Západočeská univerzita v Plzni

Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Analyzátor provozu Peer-to-Peer sítí

Plzeň 2020 Bc. Antonín Vrba

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 15. května 2020

Bc. Antonín Vrba

Poděkování

Děkuji vedoucímu této diplomové práce Ing. Martinovi Šimkovi, Ph.D za jeho rady a čas strávený na konzultacích a také Ing. Jaromírovi Staňkovi za praktické poznámky při implementaci.

Abstract

This diploma thesis deals with the development of a new network analyzer for monitoring Peer-to-Peer traffic of BitTorrent applications. First, fundamental principles of Peer-to-Peer networks are explained and a comparison of the most used protocols is performed. After that, the thesis is entirely oriented on the BitTorrent network's features, including all its extensions. A torrent downloading scenario is presented to test existing BitTorrent traffic detection solutions. The results showed shortcomings in all tested cases and therefore a proper solution was designed and implemented. The newly created analyzer was tested in the same way and their results outperforms other solutions. The analyzer will be deployed and used on University of West Bohemia dormitory network.

Abstrakt

Tato diplomová práce se zabývá vývojem síťového analyzátoru pro monitorování provozu Peer-to-Peer sítě BitTorrent. Nejdříve jsou prozkoumány principy fungování Peer-to-Peer sítí a je provedeno srovnání nejpoužívanějších protokolů. Práce se následně orientuje výhradně na síť BitTorrent včetně všech jejích rozšíření. K otestování již existujících řešení pro detekci BitTorrent provozu je definován jednotný scénář torrent stahování. Výsledky poukázaly na nedostatky existujících řešení ve všech testovaných případech a proto bylo navrženo a implementováno vlastní řešení. Nově vytvořený analyzátor byl stejným způsobem otestován a svými výsledky překonává ostatní řešení. Aplikace analyzátoru bude nasazena v prostředí kolejní sítě Západočeské univerzity.

Obsah

1 Úvod

Se začátkem nového tisíciletí a s nárůstem popularity celosvětové internetové sítě se mezi uživateli rozšířil decentralizovaný způsob komunikace. Virtuálně přímé propojení počítačů dnes známé jako Peer-to-Peer neboli P2P způsobilo revoluci ve způsobu sdílení primárně multimediálního obsahu. Dnes je P2P synonymem pro šíření autorsky chráněných děl. To je hlavním důvodem, proč se univerzitní nebo korporátní sítě snaží P2P komunikaci omezovat. Pro tento účel vzniká celá řada projektů a nástrojů, které analyzují síťový provoz, se snahou co nejpřesněji P2P komunikaci detekovat. Na relevanci tohoto tématu přidává skutečnost, že celkový P2P provoz je dnes celosvětově opět na vzestupu.

Obsahem této práce bude nejdříve seznámení s pojmem Peer-to-Peer v kontextu historie a současného využití. Následně bude provedeno porovnání vlastností nejpopulárnějších P2P sítí se zaměřením na jejich praktické používání a rozdíl nabízených funkcionalit. Detailně bude popsána P2P síť BitTorrent, včetně specifikace podružných protokolů v rámci dostupných rozšíření. Dále budou dle nadefinovaného scénáře otestovány dostupné P2P analyzátory. Na základě výsledků testování bude navržen a implementován nový P2P analyzátor pro analýzu síťového provozu v reálném čase. Aplikace analyzátoru bude implementována v jazyce C++, jejíž výsledky budou ukládány do SQL databáze, zpřístupněné vytvořeným webovým rozhraním v jazyce PHP. Nově implementovaný analyzátor bude otestován a následně porovnán s dostupnými alternativami.

Vznik této práce byl iniciován požadavkem na pokročilejší detekci P2P v prostředí kolejních sítí Západočeské univerzity, kde je v současné době nasazen analyzátor P2P sonda. Tato práce navazuje na systém správy kolejní sítě KNet jako jeho rozšíření v oblasti řešení incidentů.

2 Peer-to-Peer sítě

Peer-to-Peer je označení typu distribuované počítačové sítě, která při výměně dat nepoužívá mezilehlých prvků. Zúčastněné stanice jsou v síti v roli zdroje i konzumenta obsahu, jsou tak sobě rovní. Klasická architektura klient-server se však s různou měrou vyskytuje v sítích P2P, a to hlavně v případě inicializace připojení do sítě. V této kapitole bude popsáno současné využití P2P sítí, jejich základní vlastnosti a následné porovnání konkrétních P2P sítí.

2.1 Vznik P2P sítí

Decentralizovaná distribuce dat není žádnou novinkou v oblasti počítačových sítí. První uživatelsky dostupnou službou, která využívala tuto techniku je známé diskuzní fórum Usenet z roku 1979. Již zde se objevil způsob distribuce příspěvků s absencí centrálního serveru, což dalo vzniknout nové generaci P2P protokolů. V roce 1999 se ve formě klientského programu objevila služba Napster umožňující stahování multimediálního obsahu z počítačů jiných uživatelů sítě. Volná distribuce souborů přes P2P síť významně změnila podobu zábavního průmyslu dneška. Dopady tohoto fenoménu dobře popisuje dobová práce [18], která označuje popularizaci P2P sítí jako „otevření Pandořiny skříňky“. Na konci tisíciletí se P2P sítě bohužel stávají symbolem nelegálního šíření obsahu a díky decentralizaci, tedy samotné podstatě P2P, je velmi obtížné tyto sítě efektivně blokovat.

2.2 Využití a provoz P2P sítí

Podle statistik [11] z roku 2019 je P2P provoz opět na vzestupu. Bezmála 30 % internetového provozu je přivlastňováno právě P2P přenosům. Pro představu: video streamovací služby YouTube a Netflix mají dohromady podíl na provozu 25.8 %. Když ovšem opustíme celosvětové měřítko a zaměříme se na region EMEA (Europe, Middle East and Africa), zde P2P provoz za rok 2019 dosahuje 49.5 %, což je proti roku 2018 nárůst o 14 % absolutně. Aktuální práce, které se zabývají analýzou P2P provozu a záro-

veň i způsobem blokování, předkládají podobné hodnoty provozu, a to 40 % v případě práce [20].

S příchodem různých streamovacích služeb na tuzemský trh se dal předpokládat pokles v užívání P2P přenosů. Navzdory tomu je P2P stále velmi populární způsob k získávání především multimediálního obsahu, ale i při distribuci např. rozsáhlých aktualizací počítačových her či hromadnému šíření aktualizací operačních systémů. Principy P2P používají také virtuální měny.

2.3 P2P topologie

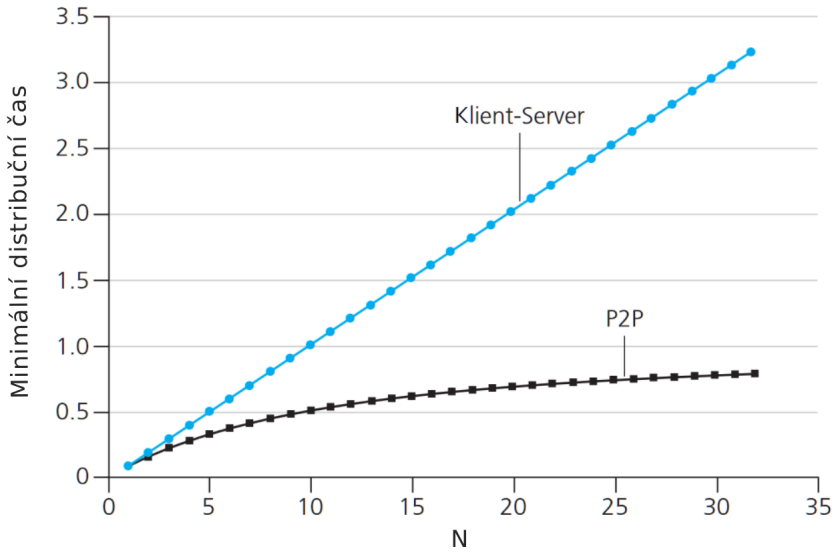
Logická topologie sítě P2P je nestálá a mění se ad hoc kdykoli se nový klient připojí. Pro označení jednoho uzlu se obecně užívá termín **peer**. Každý peer může sdílet s ostatními v sítí svůj obsah, stejně jako může žádat konkrétní obsah od ostatních - hrany v grafech případných P2P topologií jsou neorientované. Výhodou P2P uspořádané sítě proti klasické klient-server architektuře je vyšší propustnost sítě, to popisuje následující příklad z práce [15].

Pro klient-server architekturu je distribuce souboru z jednoho zdroje o velikosti F bitů do n zařízení realizované odesláním stejné kopie souboru každému, tedy nF bitů celkově. Při rychlosti odesílání u_s je vztah 2.1 minimálním časem potřebným pro dokončení distribuce souboru.

$$T_{CSmin} = \frac{nF}{u_s} \tag{2.1}$$

V případě stejného úkolu distribuce souboru z jednoho zdroje je P2P síť efektivnější. Peer vlastnící část souboru se již aktivně může podílet na další distribuci vlastní rychlostí odesílání u_i. Čím více peerů se přenosu účastní tím roste rychlost, které mohou společně dosáhnout. Zdroj se souborem musí stále do sítě dostat celý obsah v čase F/u_s. Minimální doba dokončení distribuce je tak menší, jak je patrné ze vztahu 2.2 a také z Obrázku 2.1, ve kterém je patrné vzájemné vykreslení lineární a logaritmické závislosti na vzrůstajícím počtu uzlů.

$$T_{P2Pmin} = \frac{nF}{u_s + \sum_{i=1}^{n} u_i} \tag{2.2}$$



Obrázek 2.1: Zobrazení minimální doby distribuce v závislosti na počtu peerů (převzato z [15]).

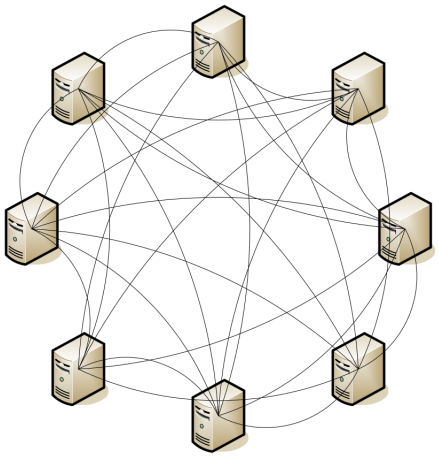
Mnoho uživatelů P2P sítí opomíjí skutečnost, že se data přenáší z počítačů ostatních uživatelů v síti nebo je prováděno stahování právě z jejich počítače. I přes komplikovanou topologii lze P2P sítě rozdělit dle [19] na úplné a hybridní.

2.3.1 Úplná Peer-to-Peer síť

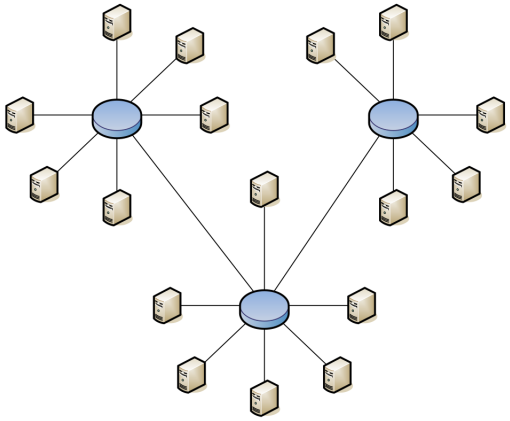
Úplnou P2P sítí se rozumí síť, ve které se nachází pouze sobě rovní uživatelé typu peer a při odebrání libovolného uzlu nedojde k žádnému výpadku konektivity v rámci zbylých uzlů sítě. Na Obrázku 2.2 je znázorněný tzv. úplný graf (každý peer propojený s každým), síť tak neobsahuje uzel, jehož odebrání by mělo za následek jakýkoliv výpadek spojení.

2.3.2 Hybridní Peer-to-Peer síť

Hybridní P2P síť se od úplné odlišuje hlavně přítomností centrálních uzlů, jak je znázorněno na Obrázku 2.3. Tyto centrální uzly (servery) mohou mít různou roli a nemusí nutně komunikovat mezi sebou. Síť je tak více zranitelná, výpadek centrálního uzlu může ochromit část sítě.



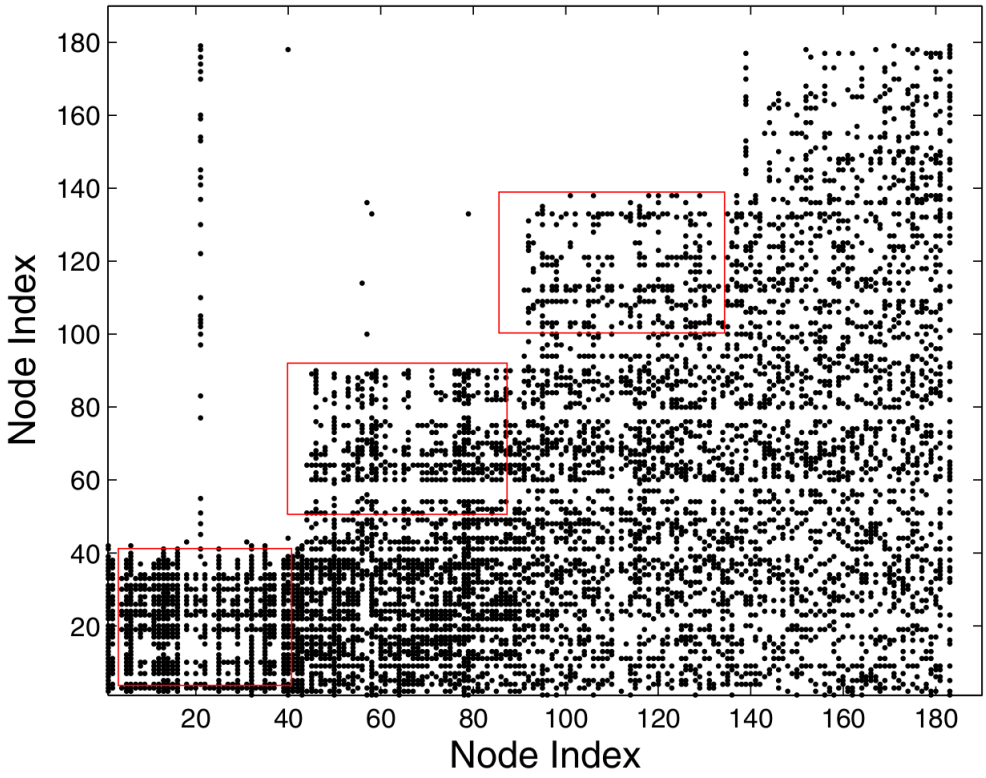
Obrázek 2.2: Diagram úplné Peer-to-Peer topologie. Převzato z [19].



Obrázek 2.3: Diagram hybridní Peer-to-Peer topologie. Převzato z [19].

2.3.3 Reálná topologie

Reálnou logickou topologii P2P můžeme popsat jako mesh, tedy smíšenou síť, která ale není úplným grafem, a to hlavně z důvodů škálovatelnosti. V práci [17] se podařilo mimo jiné dobře vizualizovat vzájemné propojení peerů a postupný rozklad zátěže s přibývajícím počtem peerů v síti. Je také vidět velké množství udržovaných spojení v rámci P2P sítě. V experimentu byl sdílen obsah o velikosti 500 MB a každých 10 minut se aktivovalo nových 50 peerů požadujících tento soubor. Na Obrázku 2.4 je matice sousednosti o velikosti $n = 200$ peerů. Matice je zachycená ve stavu 5 minut po aktivaci všech čtyř aktivací nových peerů.



Obrázek 2.4: Matice sousednosti komunikace peerů. Převzato z [17].

2.4 Srovnání P2P sítí

Jak již bylo uvedeno v sekci 2.1, síť Napster udala počátkem nového tisíciletí směr pro mnoho pozdějších P2P sítí jako např. SoulSeek nebo Audiogalaxy. V této práci je však důležité omezit se pouze na takové, které jsou aktivně využívané. Tato sekce popisuje několik vybraných P2P sítí s přihlédnutím k jejich využití a rozdílu dostupných funkcí. I když mohou být popisované P2P sítě v dnešní době nepoužívané, tak nám mohou tím jak fungují posloužit k lepšímu porozumění jak P2P v principech pracuje a vyvíjí se s dobou. Ještě před začátkem srovnání je potřeba definovat několik dále používaných termínů:

- **P2P síť** - je distribuovaná síť obsahující uzly, které komunikují díky využívání služeb sady P2P protokolů, které jsou implementované v P2P klientech.
- **P2P protokol** - je samostatný komunikační prvek, jenž bývá součástí sady P2P protokolů patřící ke konkrétní P2P síti. P2P protokoly řeší

např. problém vytváření sítě, optimalizace propustnosti sítě nebo sběr statistických informací do centrálních uzlů.

- **P2P klient** - je počítačový program umožňující připojení do P2P sítě s využitím implementovaných P2P protokolů. Může existovat více programů k připojení do stejné sítě, pokud dodrží předpis implementace P2P protokolů.

2.4.1 Gnutella

První P2P klient pro síť Gnutella byl uveřejněn v roce 2000. Byl odstraněn hlavní problém sítě Napster, a to závislost dostupného obsahu na malém množství indexovacích serverů. Síť také nebyla omezená jen pro manipulaci s audio obsahem. K připojení uzlu do sítě stačí navázat spojení s jedním uzlem, který už v síti připojený je. Získání takového počátečního uzlu může být obtížné, a tak existoval veřejný seznam stabilních, stále aktivních uzlů. Řídící protokol (jádro) používá následující zprávy mezi uzly sítě [9]:

- **PING** - test dostupnosti uzlu,
- **PONG** - odpověď na zprávu PING,
- **QUERY** - mechanismus pro vyhledávání v distribuované síti,
- **QUERY HIT** - odpověď na QUERY zprávu s adresou požadovaného souboru v síti,
- **PUSH** - používaný pokud je uzel s požadovaným souborem chráněn proti příchozím spojením.

Vyhledávání obsahu probíhá přes ostatní připojené uzly, jako prohledávání do šířky, jednotlivé uzly nabízí přes HTTP 1.1 server svůj obsah. Přenos dat tak není distribuován přes síť Gnutella, ale přímo mezi dvěma uzly. Místo označení peer se vyskytuje označení „servant“. Přenos dat je řízen HTTP příkazy jako u běžné webové komunikace.

Nevýhodou sítě Gnutella je škálovatelnost. Počet zpráv v síti lineárně narůstá s přibývajícím počtem uživatelů. Statistikami se zjistilo, že polovina obsahu je umístěna pouze na 1 % uzlů, síť tak může částečně degradovat do podoby klient-server. Decentralizované vyhledávání může působit jako

výhoda, pokud nelze mít několik stále dostupných centrálních serverů. Pokud však síť tvoří velký počet nestabilních a často nedostupných uzlů, získávání obsahu ze sítě a zvlášť vyhledávání se stává problémem. V pozdějších verzích se pak uzly sítě rozdělovaly podle toho, zda jsou stabilní či nestabilní. Gnutella také poskytuje velkou platformu pro DDOS útoky[1] [9].

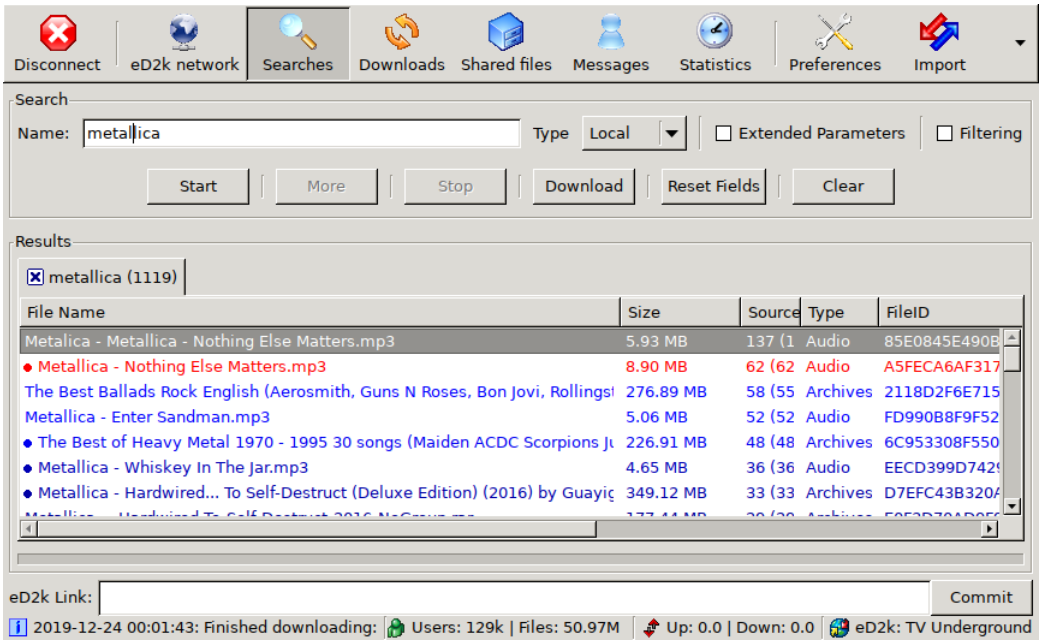
V dnešní době již síť Gnutella není aktivně využívána, klient *gtk-gnutella* pro distribuci Debian 10 není již ani zařazen v oficiálních repozitářích. Ze sítě Gnutella se později oddělila samostatná síť Ares, která však neměla takovou popularitu.

2.4.2 eDonkey

Síť eDonkey (eD2k) a její protokoly byly poprvé implementovány v klientu s názvem eDonkey2000 a později vznikl i známý klient eMule. Proti síti Gnutella se eDonkey nepokouší o kompletní decentralizaci, ale síť implicitně obsahuje stabilní servery sloužící jako slovník pro určení jaké uzly obsahují jaký obsah. Servery si vzájemně vyměňují informace o připojených uživatelích a vedou i jejich statistiky. Na serverech sítě eDonkey se nenachází klientský program, ale proprietární[2] software s názvem eserver. K indexaci obsahu se používá MD4 hash[3] o velikost 16 B. Pro stažení souborů musí uživatelé nejdříve kontaktovat zmíněné servery, získat tak hash odpovídajících souborů a podle nich se znovu dotázat na adresu uzlu, kde se vyhledávaný obsah nachází [22]. Protokoly sítě eDonkey aktivně využívají rozdělování souborů na části a tak je možné více využít potenciál P2P sítě, když se soubor stahuje z více zdrojů zároveň.

Vinou velké popularity sítě eDonkey se kolem roku 2006 objevil článek [1] oznamující uzavření největšího serveru Razorback2 kvůli indexování souborů s autorskými právy, které byly nelegálně šířeny. Síť se ovšem rychle vzpamatovala, tento server zdaleka nebyl jediný v provozu. Ve stejném roce používaly tuto síť miliony uživatelů a byla to největší P2P síť na světě. V dnešní době je síť malá, ale stále funkční. Distribuce Debian 10 poskytuje aktualizovaný klient aMule viz Obrázek 2.5, který se v síti eDonkey automaticky připojuje k osmi veřejným serverům, na kterých je dohromady umístěno více než 50 milionů souborů, kdy drtivá většina je inzerována pouze jedním uzlem.

[1] Distributed Denial of Service - nedostupná služba z důvodu zahlcení sítě.
[2] Nezveřejněný kód většinou s autorskými právy.
[3] Jednosměrná funkce s velkým rozptylem - otisk souboru.



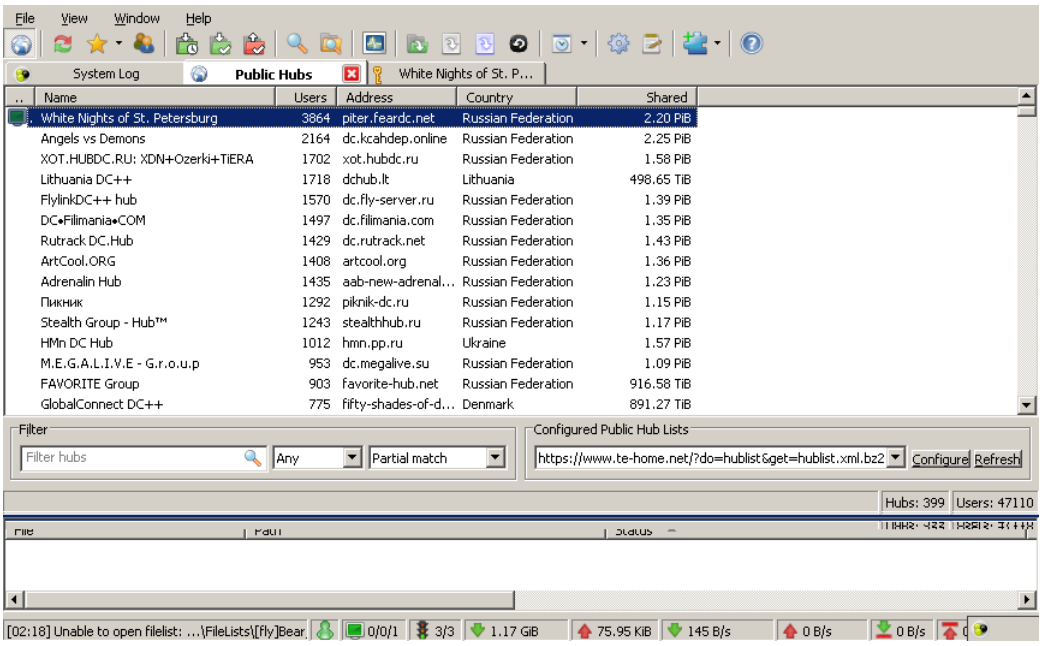
Obrázek 2.5: P2P klient aMule 2.3.2 (vyhledávací formulář seřazen podle počtu zdrojů).

K těmto serverům je aktivně připojeno dohromady kolem 130 tisíc uživatelů. V síti existuje „bodový systém“ umožňující uživatelům s větším množstvím sdílených souborů větší rychlost stahování. Během testování bylo ověřeno, že klient aMule ve výchozím nastavení stahování souboru ze sítě ihned začíná se sdílením jeho částí.

2.4.3 Direct Connect

Síť Direct Connect se v roce 1999 inspirovala již existující textovou komunikací IRC (Internet Relay Chat). Topologie je pak založená na serverech označených jako *hub* a k nim připojení klienti. Mezi klienty (peery) je vzájemně realizována topologie klient-server, jak ji už z P2P sítí známe. Hub naslouchá příchozím spojením, přeposílá informace mezi peery jako např. vyhledávání v síti nebo výměna textových zpráv. Peer kontakuje hub s dotazem k vyhledávání a hub přepošle tento dotaz všem připojeným peerům, odpověď se pak šíří přímo mezi peery. Peer v této síti disponuje tzv. sloty, tedy počtem paralelně probíhajících přenosů do sítě. Před těmito sloty se pak v případě obsazení vytváří fronty, soubor je tak vcelku přenášen z jednoho zdroje k peeru, který si žádá obsah. [21]

Práce [21] se zabývala analýzou sítě Direct Connect čítající mimo jiné aktivitu peerů v závislosti na lokalitě a také povahu dat nacházejících se v síti. V roce 2009 Bylo analyzováno 104 TB dat, ze kterých bylo zjištěno, že 58 % souborů je unikátních s majoritním podílem 43 % audio obsahu. Síť Direct Connect funguje dodnes a lze se k ní připojit stále aktivně vyvíjeným klientem DC++ pro Windows a nebo multiplatformním klientem EiskaltDC++. Při testování byl použit klient DC++ v prostředí wine[4].



Obrázek 2.6: P2P klient DC++ v0.868 (veřejně dostupné huby).

Po spuštění klientského programu dostane uživatel na výběr řádově stovky veřejně dostupných hubů viz Obrázek 2.6, které mají i několik tisíc aktivních uživatelů. Na Obrázku 2.6 je ve sloupečku *Shared* vidět celková velikost sdílených souborů uvnitř jednoho hubu. Zajímavou vlastností této sítě je, že uživatelé dobrovolně sdílí desítky TB dat ze svých osobních počítačů a kromě vyhledávání lze tento obsah i klientem DC++ procházet. Podobně tak jako síť eDonkey i Direct Connect je možné považovat svým způsobem za „P2P internetový archiv“. Velmi příjemnou vlastností klienta DC++ je absence spuštění automatického sdílení staženého souboru, a to je zásadní rozdíl proti politice klienta aMule v síti eDonkey.

[4]Prostředí pro běh Windows programů na jiných POSIX kompatibilních systémech.

2.4.4 BitTorrent

Na počátku sítě BitTorrent (dále jen BT) stála myšlenka z devadesátých let o zabezpečení souboru rozdělením na několik zašifrovaných částí a jejich následné rozmístění po síti. V roce 2001 byla vydána první otevřená beta verze a o rok později byl BT veřejně představen na konferenci se skromným cílem efektivně a jednoduše šířit software pro linuxové distribuce. Od tohoto využití se však okamžitě přešlo ke sdílení hlavně multimediálního obsahu. V průběhu několika let se z BT stává nejrozšířenější P2P síť na světě, předchozí sekce 2.2 zabývající se aktuálním P2P provozem reflektuje právě pouze provoz P2P sítě BT. Statistiky [11] neuvádí zastoupení jiných než BT P2P sítí. Na internetu již pojmenování BitTorrent automaticky symbolizuje, leč nesprávně, celý svět P2P, a to zejména nelegální sdílení obsahu. V současné době se v této síti odhaduje na desítky milionů aktivních uživatelů.

Hlavním důvodem pro rozdělení souborů na části (bloky) je podle tvůrce BT výhodnější saturace asymetrického internetového připojení např. DSL [19]. Násobně nižší rychlost odesílání dat se negativně projeví v sítích typu Direct Connect, naopak sítě jako BT nebo eDonkey dokážou stahovaný soubor odebírat z více zdrojů najednou a tak dosahovat větších rychlostí i přes to, že zdroje jednotlivě mohou odesílat data pomalu. Čím má BT více peerů zajímajících se o jeden konkrétní soubor, tím je síť efektivnější v jeho distribuci viz předešlý Obrázek 2.1. Pro další porozumění sítě BT je nutné definovat následující termíny z [19]:

- **Torrent soubor** - je textový soubor s příponou *.torrent* obsahující mimo jiné název, velikost, URL odkaz na tracker a hash souboru adresovaného v BT síti. Pro stažení obsahu z BT sítě je nutné disponovat takovým *.torrent* souborem.
- **Tracker** - je server uchovávající statistiky o stahování určitého torrentu např. počet právě stahujících peerů. Tento server nemá podíl na datových přenosech, ale sbíraná data pomáhají inzerování *.torrent* souborů přes webové rozhraní jiných indexovacích služeb, tyto servery jsou často nesprávně také označovány za trackery. Tracker využívá HTTP protokol k získávání informací od BT klientů.
- **Seed** - tzv. seeder je peer, který vlastní všechny části jednoho torrentu a už jen sdílí jeho obsah ostatním peerům.
- **Leech** - tzv. leecher je další typ peera, který je závislý na seedování

ostatních. Tento peer ještě nezískal všechny bloky torrentu, ale už se podílí na sdílení částí, jež vlastní.

- **Swarm** - skupina peerů s jedním společným torrentem. V rámci takové skupiny je platné rozdělení na peery typu seed nebo leech.

Popularitu a využití sítě BT zachycuje i současně nasazená P2P sonda protokolů na kolejní síti, která je součástí projektu KNet [23]. Detekovány jsou signatury P2P protokolů v jednotlivých paketech. Tabulka 2.1 obsahuje výčet P2P sítí, které aktuální sonda umí detekovat. Pokud uživatelé kolejní sítě dosáhnou určitého limitu paketů patřící do kategorie P2P, je vyvolán incident. Incidenty na kolejní síti se tak téměř výhradně týkají protokolů ze sítě BitTorrent.

P2P síť	počet spojení	počet incidentů
BitTorrent	6 129 498	1314
eDonkey	12 912	15
Ares	356	0
SoulSeek	314 710	22
Gnutella	1440	0

Tabulka 2.1: Výstup P2P sondy od 16-7-2017 do 29-12-2019 (896 dní).

Zde je ukončen nepatrný úvod do sítě BitTorrent, Kapitola 3 dále detailně popisuje širokou rodinu protokolů sítě BitTorrent a jeho významných rozšíření BEP[5].

2.5 Vyhodnocení srovnání vybraných P2P sítí

Pro implementaci P2P analyzátoru bude nutné omezit se pouze na jeden typ P2P sítě. Rozhodování je v tomto případě velmi jednoduché. Hlavním kritériem je aktuální využívání P2P sítě a BitTorrent je zdaleka nejpopulárnější, jak ukazují celosvětová data [11] spolu s daty z kolejní P2P sondy viz předchozí Tabulka 2.1.

Srovnání a testování dále ukázalo i jiné aspekty P2P sítí, např. integrované záplavové vyhledávání obsahu v případě sítě eDonkey, nebo možnost používat síť Direct Connect bez nutnosti automaticky sdílet stažený obsah. Tyto

[5]BitTorrent Enhancement Proposal - rozšíření protokolů BitTorrent.

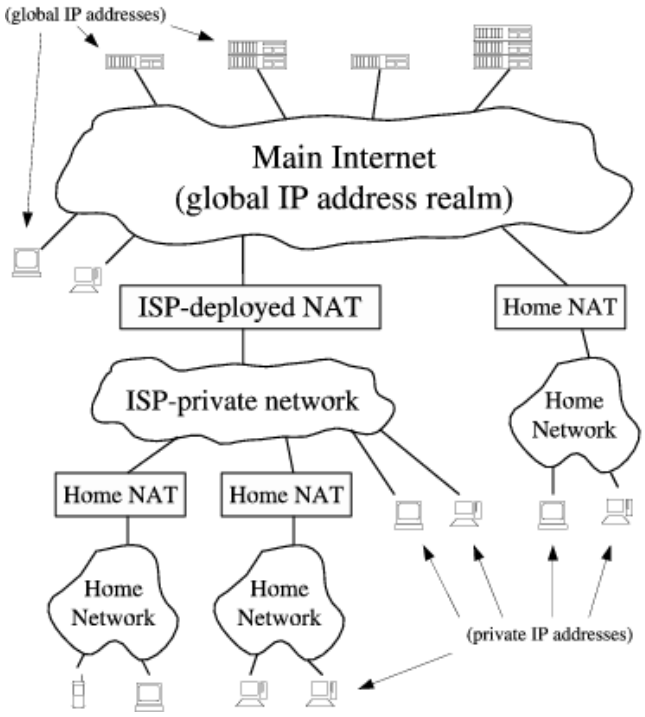
dvě zmíněné P2P sítě vykazují podobnou aktivitu a rozhodně ještě dlouho budou existovat právě kvůli pevným základům komunity kolem těchto P2P sítí.

2.6 P2P komunikace a NAT

V této sekci bude popsána metoda, která umožňuje P2P sítím komunikaci mezi jednotlivými peery v případě, že jeden nebo oba peery nedisponují veřejnou globálně routovanou IP adresou a nemohou se tak přímo kontaktovat. V internetu jak jej dnes známe je hojně využívána technika NAT (Network Address Translation) z důvodů bezpečnostních (výchozí blokování všech příchozích spojení) a také kvůli nedostatku volných IPv4 adres. Vzhledem k enormnímu růstu internetové sítě tak dnes téměř každý návrh počítačové sítě zahrnuje nasazení překladu adres a tedy využití IP adres z neveřejných rozsahů. Běžně se tak lze setkat s komunikací napříč několika NAT prvky viz Obrázek 2.7. NAT není jen standardní součástí domácích routerů, je také využíván internetovými poskytovateli (ISP). Situaci z dob, kdy každý síťový prvek využíval adresu z veřejného rozsahu se může v jisté míře podařit navrátit s širším zavedením IPv6 adresováním.

Spojení na úrovni transportní vrstvy TCP/IP modelu je definováno zdrojovou IP adresou, zdrojovým portem, cílovou IP adresou a cílovým portem. Směr takového spojení je dán prvním SYN paketem v případě TCP a nebo směrem prvního UDP datagramu. Typickým typem NATu je tzv. outbound NAT - asymetrický most mezi privátním a veřejným rozsahem sítě. Odchozí spojení jsou tak ve výchozím stavu povoleny kamkoli a příchozí spojení mohou být uskutečněny jen v případě již inicializovaného spojení zevnitř privátní sítě. Úloha NATu je tak zjevná, router při nejčastějším použití NAT 1:N (maškaráda) přepisuje u odchozích paketů zdrojovou IP adresu a port a udržuje si tabulku spojení pro případné odpovědi, kterým zase přepisuje cílovou IP adresu a port. V [8] jsou rozlišovány UDP a TCP varianty pro „hole punching“.

UDP Hole Punching umožňuje dvěma peerům navázat spojení i pokud se oba nacházejí za NATem. Peery A a B využijí třetího uzlu S, který je dosažitelný v internetu pro oba peery. Pokud chce A přímo kontaktovat B musí nejdříve pomocí S zjistit adresu B společně s portem. Aby to bylo možné, musí B periodicky udržovat spojení s S, přičemž mu zasílá dvě dvojice parametrů



Obrázek 2.7: NAT v kontextu privátních a veřejných adresních domén (převzato z [8]).

IP:port. První dvojice (privátní) jsou parametry, které B věří, že používá ke komunikaci s S, druhá (veřejná) dvojice jsou aktuální parametry, které S zjistí z příchozího spojení (pokud by B komunikoval rovnou z veřejné IP adresy budou obě dvojice parametrů stejné). Peer A poté využije získané údaje ke kontaktování B skrz NAT „díru“ díky uzlu S.

Podobně jako v případě UDP je na úrovni protokolu možné navázat přímé spojení v případě TCP je více spolehlivé, protože NAT zařízení je obeznámeno s životním cyklem TCP spojení. Problémem k řešení je však implementace API k BSD socketům, nad kterými lze používat *connect()* k inicializaci spojení nebo *listen()* a *accept()* pro akceptování spojení, ne však obojí. TCP Hole Punching ale vyžaduje na jednom portu inicializaci i akceptaci TCP spojení zároveň. Dnešní operační systémy už ale podporují TCP socket s nastavením *SO_REUSEPORT*. V případě UDP postačí peeru jeden socket pro komunikaci s pomocným serverem i mnoha jinými peery, u TCP musí peer disponovat čtyřmi sockety (naslouchání pro příchozí spojení, komunikace s pomocným serverem S a dva sockety pro komunikaci mezi peery).

3 BitTorrent

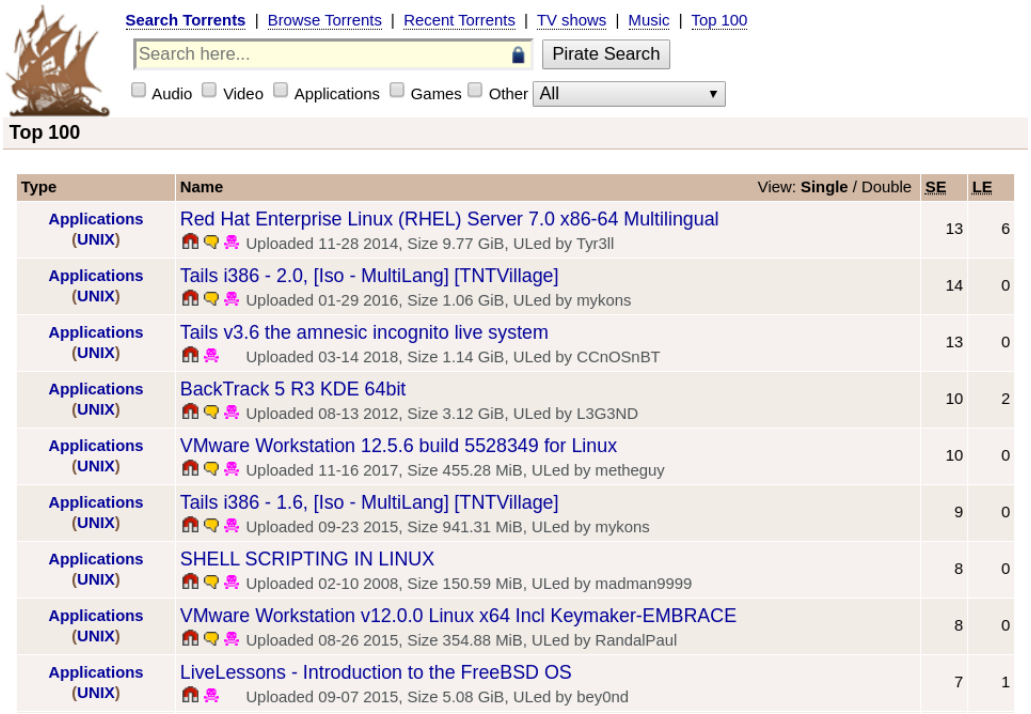
V sekci 2.4.4 byla uvedena P2P síť BitTorrent a tato kapitola bude pokračovat v popisu celé rodiny protokolů, které síť BT využívá. Protokolová rozšíření pro síť BT jsou popisována ve formátu tzv. BEPs (BitTorrent Enhancement Proposals). Každý BEP má pořadové číslo např. BEP 41 a v průběhu této kapitoly jsou i takto referencovány. Pro navržení síťového analyzátoru je nutné nejdříve zevrubně představit BT protokoly a jejich uplatnění v reálné implementaci BT klientů. Jedna důležitá funkcionalita, kterou musí implementovat BT klienti, už byla popsána v předešlé sekci 2.6. Bez techniky hole punching by síť BT jen těžko dosáhla takové popularity. Popularita zapříčinila i relativně velké množství aktivně používaných klientů jako např. μTorrent, qBittorrent a Transmission. Kromě jmenovaných „klasických“ implementací je také běžné setkat se s implementací v jazyce JavaScript s názvem WebTorrent. Každý webový prohlížeč se tak může stát plnohodnotným BT klientem. Napříč BT klienty může docházet k rozdílům v koherenci implementovaných BT protokolů. Nezávislý vývoj nejvíce používané klientské aplikace μTorrent přispěl k zavedení a rozšíření nových funkcionalit BitTorrentu, jak bude v následujících sekcích vysvětleno.

3.1 Připojení do sítě BitTorrent

Připojení uživatele do sítě BT se zájmem zúčastnit se na distribuci určitého obsahu lze již dříve vysvětlenou terminologií popsat jako připojení peera do swarmu za využití služeb trackeru. Získání *.torrent* souboru je možné např. z notoricky známého původem švédského indexovacího serveru The Pirate Bay[1] viz Obrázek 3.1. Popularitu torrentu lze určit podle počtu peerů typu seed (LE) a leech (LE).

Na Obrázku 3.2 je znázorněn jednoduchý diagram, ve kterém se peer (Alice) připojí do swarmu. V této části popisu sítě BT je ještě možné postihnout „výchozí“ chování bez účasti rozšíření, které významně mění pohled na síť BT. Torrent soubor obsahuje adresu tracker serveru, od kterého nový peer dostane sadu adres peerů zúčastněných ve swarmu. Komunikace mezi pee-

[1] https://thepiratebay.org



Search Torrents | Browse Torrents | Recent Torrents | TV shows | Music | Top 100

Search here... Pirate Search

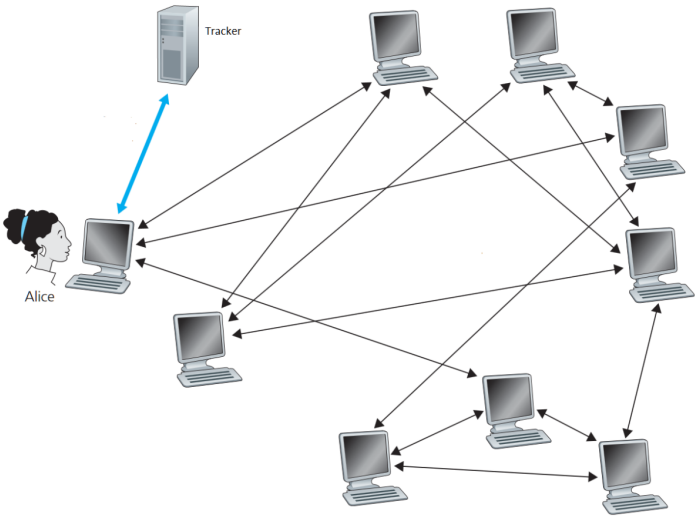
Audio Video Applications Games Other All

Top 100

Type	Name	View: **Single** / Double	SE	LE
Applications (UNIX)	Red Hat Enterprise Linux (RHEL) Server 7.0 x86-64 Multilingual Uploaded 11-28 2014, Size 9.77 GiB, ULed by Tyr3ll		13	6
Applications (UNIX)	Tails i386 - 2.0, [Iso - MultiLang] [TNTVillage] Uploaded 01-29 2016, Size 1.06 GiB, ULed by mykons		14	0
Applications (UNIX)	Tails v3.6 the amnesic incognito live system Uploaded 03-14 2018, Size 1.14 GiB, ULed by CCnOSnBT		13	0
Applications (UNIX)	BackTrack 5 R3 KDE 64bit Uploaded 08-13 2012, Size 3.12 GiB, ULed by L3G3ND		10	2
Applications (UNIX)	VMware Workstation 12.5.6 build 5528349 for Linux Uploaded 11-16 2017, Size 455.28 MiB, ULed by metheguy		10	0
Applications (UNIX)	Tails i386 - 1.6, [Iso - MultiLang] [TNTVillage] Uploaded 09-23 2015, Size 941.31 MiB, ULed by mykons		9	0
Applications (UNIX)	SHELL SCRIPTING IN LINUX Uploaded 02-10 2008, Size 150.59 MiB, ULed by madman9999		8	0
Applications (UNIX)	VMware Workstation v12.0.0 Linux x64 Incl Keymaker-EMBRACE Uploaded 08-26 2015, Size 354.88 MiB, ULed by RandalPaul		8	0
Applications (UNIX)	LiveLessons - Introduction to the FreeBSD OS Uploaded 09-07 2015, Size 5.08 GiB, ULed by bey0nd		7	1

Obrázek 3.1: Webové rozhraní indexovacího serveru The Pirate Bay.

rem a trackerem je dále popsána v sekci 3.2. Dále proběhne navázání TCP komunikace s nově získanými peery, zde se uplatní základní BT protokol popsaný v sekci 3.3. Peer (Alice) je teď součástí P2P swarmu a mezi ostatními peery hledá části souboru, které potřebuje a také již nabízí části získané, je tak peerem typu leech.



Obrázek 3.2: Diagram BT swarmu včetně trackeru (Převzato z [15]).

3.1.1 Magnet URI

Jednotným identifikátorem zdroje (URI) byl dosud *.torrent* soubor. V dnešní době se však již hromadně přechází na magnet URI, což je v podstatě jen odkaz umožňující připojení do swarmu. Na Obrázku 3.1 je u každé položky k dispozici ikona ve tvaru magnetu, zde je možné tento odkaz získat. V následujícím Výpisu 3.1 je formát magnet URI a reálný magnet získaný z indexovacího serveru The Pirate Bay.

```
magnet:?xt=urn:btih:<info-hash>
        &dn=<name>
        &tr=<tracker-url>
        &x.pe=<peer-address>

magnet:?xt=urn:btih:35f5a50fe20d5b719103c2eb9cdee8ad92130e53
        &dn=BackTrack+5+R3++KDE+64bit
        &tr=udp://tracker.leechers-paradise.org:6969
        &tr=udp://tracker.openbittorrent.com:80
        &tr=udp://open.demonii.com:1337
```

Výpis 3.1: Formát a ukázka magnet URI.

BEP 9 [7] definuje jednotlivé parametry magnet URI následovně:

- **<info-hash>** - 20 bajtový jednoznačný identifikátor každého torrentu. Jedná se o hash SHA-1, který je v tomto případě reprezentován 40 hexadecimálními znaky. Infohash se počítá přes všechny části torrentu (název, každá část jeho dat a celková velikost).
- **<tracker-url>** - adresa a port trackeru, kterých může být uvedeno i více. Jelikož HTTP působilo zbytečnou režii v komunikaci mezi peerem a trackerem, začalo se postupně přecházet na UDP tracker protokol popsaný v BEP 15. Magnet URI nemusí obsahovat položku žádného trackeru, v tom případě se využívá služeb protokolu DHT viz sekce 3.6.
- **<peer-address>** - adresa peera nebo více peerů, kteří se mají nacházet ve swarmu. Tento parametr se ale v praxi nevyužívá.

3.2 HTTP

Jak již bylo řečeno, BT používá pro komunikaci s trackerem textový protokol HTTP. Tracker odpovídá na dotazovací metodu GET a existují pouze dvě zprávy: *scrape* nebo *announce*. Tyto zprávy jsou v pravidelných intervalech opakovány. BT klient vkládá do GET metody různé parametry ve známém formátu *parametr=hodnota* oddělených znakem '&'. Dle RFC 1738[2] je také textový obsah zprávy kódován a bajty bez textové reprezentace[3] v URL jsou pak v podobě "%nn"např. znak '!' je kódován jako "%21".

3.2.1 Zpráva scrape

Jako první posílá peer trackeru zprávu *scrape*, která slouží k zjištění stavu stahovaného torrentu. Parametrem tohoto dotazu je již zmiňovaný infohash, který je částečně viditelný u Výpisu 3.2 v URL zakódovaném tvaru. Součástí takové zprávy je také konkrétní verze BT klienta.

```
GET /scrape?info_hash=%0f%2a%3a%df%e8.%1c%92%b3%9...
HTTP/1.1
Host: bttracker.debian.org:6969
User-Agent: uTorrent/355(111915293)(45341)
```

Výpis 3.2: Část textu zprávy *scrape*.

Tracker pak ke konkrétní hodnotě infohash parametru odpoví aktuálními statistikami, které BT klient využije k určení, zda tento torrent má potenciál k dokončení stahování: *complete* (počet peerů typu seed ve swarmu), *downloaded* (kolikrát tracker zaznamenal dokončení stahování) a *incomplete* (počet peerů typu leech ve swarmu).

3.2.2 Zpráva announce

Zprávou typu *announce* oznamuje BT klient trackeru svůj stav stahování určitého torrentu. Odpovědí je mu od trackeru seznam peerů, kteří jsou součástí swarmu identifikovaného podle infohash parametru. Podle BEP 3 [5]

[2] http://www.faqs.org/rfcs/rfc1738.html
[3] https://www.w3schools.com/tags/ref_urlencode.ASP

obsahuje Výpis 3.3 několik parametrů metody GET, tyto parametry jsou dále rozepsány v seznamu.

```
GET /announce?info_hash=%0f%2a%3a%df%e8.%1c ...
&peer_id=-UT355W-%1d%b1%c3t%a8%807%7dF%0d%c5%dc
&port=18640
&uploaded=0
&downloaded=0
&left=351272960
&event=started
&numwant=200
```

Výpis 3.3: Část textu HTTP zprávy *announce*.

- **peer_id**, **port** - parametry jednoznačně identifikující peera (BT klientem generovaný 20 znakový identifikátor a naslouchající port).
- **uploaded**, **downloaded**, **left** - parametry definující průběh stahování.
- **event** - oznámení pro tracker o záměru stahování nebo dokončení (*started*, *completed*, *stopped*).
- **numwant** - počet peerů, které požaduje peer od trackeru, tato hodnota silně závisí na druhu BT klienta.

3.2.3 Bencoding

Odpovědi na zprávy typu *scrape* a *announce* jsou kódovány v tzv. B-kódování „benkódování“. Toto kódování se používá napříč protokoly BT sítě a je také součástí první BT specifikace [5]. Typickou vlastností tohoto kódování je reprezentace čísel jako textu v desítkové notaci. Rozlišují se zde následující 4 datové typy: řetězec, číslo, seznam a slovník:

1. **Řetězec** je ve tvaru `délka:ASCII_hodnota` např. `4:spam` reprezentuje řetězec "spam".
2. **Číslo** začíná znakem 'i' a končí znakem 'e' např `i3e` je číslo 3 a `i-3e` pak znamená hodnotu opačnou.
3. **Seznam** je uvozen znakem 'l', který následuje libovolný počet prvků ukončených znakem 'e' např. `l4:spam4:eggse` je seznamem řetězců "spam"a "eggs".

4. **Slovník** slouží pro reprezentaci dvojic `klíč:hodnota`. Začíná znakem 'd', pokračuje libovolným počtem dvojic a je ukončen znakem 'e'. Příkladem slovníku nechť je `d3:cow3:moo4:spam4:eggse` pro dvojice `cow:moo` a `spam:eggs`.

Skládání výše uvedených typů je demonstrováno Výpisem 3.4:

```
d1:ad2:id20:abcdefghij0123456789e1:q4:ping1:t2:aa1:y1:qe

"t":"aa", "y":"q", "q":"ping",
"a":{"id":"abcdefghij0123456789"}
```

Výpis 3.4: Bencoding v praxi (vnořený slovník).

3.3 Protokol BitTorrent

V této práci již byla vysvětlena úloha trackeru v síti BT, komunikace mezi peery je vyjednávána dle protokolu BitTorrent, podle kterého se i celá síť nazývá a je klíčovým prvkem při přenosu dat respektive částí souboru. Původní dokumentace [5] označuje tento protokol jako „peer wire protocol“, nicméně program Wireshark[4] tento protokol rozeznává pod řetězcem "bittorrent". Podle článku tvůrce BitTorrentu [16] řeší tento protokol problém distribuce souboru při velkém množství připojených peerů, kteří vlastní různé části tohoto souboru. Dochází k častým změnám swarmu a nově připojený peer by měl bez zbytečné zátěže ostatních peerů spolehlivě získat všechny části souboru. Zjednodušeně řečeno je tento protokol zodpovědný za zvolení peera a části, která se bude přenášet.

Spojení mezi peery je symetrické ve smyslu stejných vzájemných zpráv. Komunikace obou peerů začíná s dvojicemi vzdálených a lokálních stavových parametrů: *choked* = 1 a *interested* = 0. Parametr *choked* reprezentuje „nasycenost“ - dočasná nemožnost řešit požadavky a přenosy. Parametr *interested* jednoduše znamená zájem o data druhého peera např. stažení části souboru, který požaduje peer A od peera B je podmíněno následovným nastavením stavových parametrů: *A.interested* = 1 a *B.choked* = *0*. Každý peer si drží tabulku takových parametrů pro každé spojení a podle toho se řídí další rozhodování algoritmů.

[4]Volně dostupný síťový analyzátor https://www.wireshark.org/.

Každé spojení mezi peery začíná po navázání TCP spojení dvojicí zpráv *handshake* se strukturou patrnou z Tabulky 3.1. Na prvním místě je délka názvu protokolu *pstrlen* následována řetězcem "Bittorrent protocol"jako *pstr*. Po *reserved* prostoru se nachází dva již známé parametry: infohash a identifikátor peera.

PSTRLEN	PSTR	RESERVED	INFOHASH	PEER ID
1 B		8 B	20 B	20 B

Tabulka 3.1: Struktura obsahu *handshake* zprávy.

Kromě zprávy *handshake* jsou ostatní vypsané zprávy protokolu BT [5] strukturované dle Tabulky 3.2. Zprávy začínají definicí jejich délky (*msg len*), pokračují jejich typem a případnými daty. V následujícím seznamu se nachází vybrané zprávy, číselné dodatky u názvů reprezentují *msg type* položku z Tabulky 3.2. Pro udržení spojení mezi peery se používá speciální (prázdný) typ zprávy *keepalive* zasílaný v pravidelném intervalu dvou minut.

MSG LEN	MSG TYPE	PAYLOAD
4 B	1 B	

Tabulka 3.2: Základní struktura BT zpráv.

- **Choke (0)**, **unchoke (1)**, **interested (2)** a **not interested (3)** - zprávy neobsahující žádná data navíc. Jejich záměrem je šířit změny již popsaných stavových parametrů.
- **Have (4)** - tato zpráva informuje druhého peera o tom, že byla úspěšně získána část souboru a o zaslání indexu této části v obsahu zprávy.
- **Bitfield (5)** - zasláním zprávy obsahující bitové pole oznamuje peer druhé straně počáteční stav částí souborů, které již vlastní.
- **Request (6)** - tato zpráva slouží jako žádost o specifickou část souboru, v obsahu zprávy se nachází index části a také bajtový posun s délkou sloužící k případnému získání ještě menšího kusu dat. Běžnou velikostí jedné části souboru v BT síti je 512 kB.

V této sekci byla popsána struktura a význam protokolu BitTorrent - vyjednávání částí souborů k oboustranné distribuci. Práce [16] postihuje celé strategie podle kterých byly implementovány algoritmy pro optimální výběr a pořadí částí souborů od různých peerů. Protokol také při distribuci dat zahrnuje principy reciprocity peerů přibližující chování BT sítě k takovému,

jež lze popsat jako Paretovo optimum[5] - rovnovážný stav při kterém zlepšení prostředků jednoho uzlu ubere prostředky jinému. Strategie algoritmů protokolu BT uvedené v práci [16] nejsou z pohledu síťového analyzátoru tolik podstatné a nebudou tak v této sekci dále popisovány.

3.4 μTorrent Transport Protokol

Motivací dle rozšíření BEP 29 [4] k vytvoření μTorrent Transport Protokolu známého ve zkratce také jako μTP nebo uTP byl přechod z transportního protokolu TCP na UDP, a to hlavně kvůli přebytečné režii TCP. BT klienti používají protokoly BT i μTP společně a samotná data tak mohou být mezi peery přenášena pomocí TCP i UDP. TCP spojení ale sdílejí rovnoměrně propustnost. Zapnutý BT klient s velkým množstvím TCP spojení tak přirozeně omezí propustnost jiného programu, např. internetovému prohlížeči.

Protokol μTP musí implementovat vlastní řešení kongesce - zahlcení sítě. V případě TCP se datový tok reguluje změnou parametrů protokolu s posuvným okénkem na základě počtu ztracených packetů. μTP používá vlastní protokol s přítomností posuvného okénka pod názvem LEDBAT (Low Extra Delay Background Transport). LEDBAT je definován v RFC[6] 6817 s doporučeným použitím pro aplikace přenášející velké množství dat na pozadí. LEDBAT zajišťuje, že jeho UDP spojení ustupují v propustnosti konkurenčním spojením.

3.5 LSD protokol

Protokol LSD (Local Service Discovery nebo také Local Peer Discovery) je od roku 2015 funkční rozšíření sítě BT popsané v BEP 14 [3]. Úkolem tohoto protokolu je inzerování swarmu do lokální sítě skrze multicast adresu `239.192.152.143:6771`. Po inicializaci BT klienta dochází k přihlášení do již zmíněné multicast skupiny protokolem IGMPv2, kde se vyskytují textové LSD zprávy jako z Výpisu 3.5.

Zpráva z Výpisu 3.5 využívá formátování jako textový protokol HTTP. BT klient zde uvádí naslouchající UDP port pro jiné uživatele na síti se zájmem

[5] Vilfredo Pareto 1848-1923 byl italský matematik a ekonom.
[6] https://tools.ietf.org/html/rfc6817

```
BT-SEARCH * HTTP/1.1
Host: 239.192.152.143:6771
Port: 18640
Infohash: A1C3ED8571F7DAE9EA853A23CDEF5723F74CCD11
```

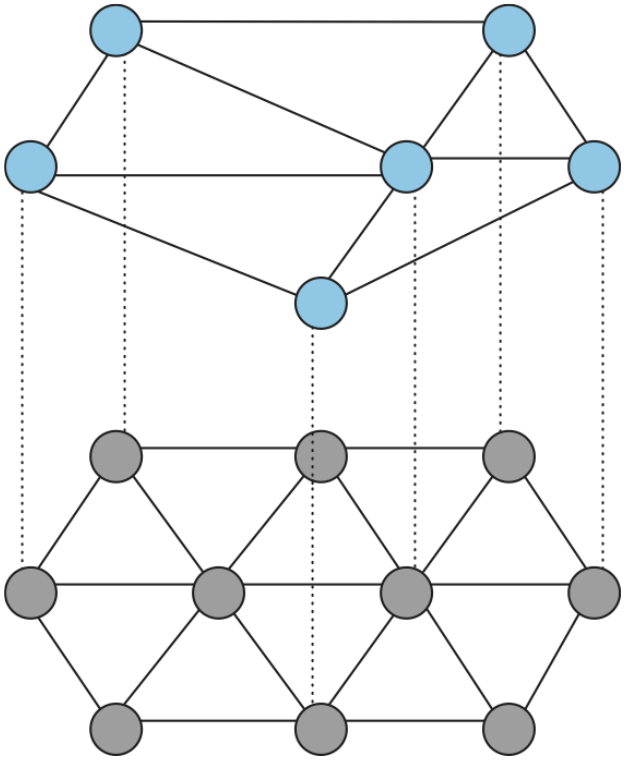
Výpis 3.5: Ukázka pravidelně zasílané LSD zprávy.

o připojení do swarmu pro stahování torrentu identifikovaného infohashem. Pokud se klient účastní několika různých swarmů, zasílá takových zpráv více. Toto poměrně minimalistické rozšíření tak může zprostředkovat spojení dvou peerů uvnitř lokální sítě a tedy šetřit provoz směřující mimo síť lokální. Toto rozšíření není tak výrazné, ale klientské aplikace jej ve výchozím nastavení mají aktivované a zjištění tohoto protokolu na lokální síti je velice přesvědčivým důkazem o používání sítě BT. Pro uživatele, kteří o tomto rozšíření nevědí může být překvapivé, že jejich BitTorrentový klient pravidelně informuje lokální síť o probíhajícím stahování. Jiní připojení uživatelé se mohou jen přihlásit do multicast skupiny a naslouchat textovým LSD zprávám.

3.6 DHT protokol

Rozšíření BEP 5 [6] přidává do BT sítě protokol DHT (Distributed Hash Table), který dále znemožňuje zjednodušený pohled na BT síť jako např. reprezentoval Obrázek 3.2 na začátku této kapitoly. Každý peer se díky DHT implementaci stává také trackerem. V této sekci bude vysvětleno jak je možné zúčastnit se stahování tzv. trackerless torrentů - připojení se do swarmu jen na základě infohash hodnoty. Také bude dále rozlišeno označení „peer“ jako uživatel používající BT nebo i μTP protokoly a označení „uzel“ pro uživatele využívající navíc DHT protokol. Na Obrázku 3.3 je znázorněna tzv. překryvná síť jako klíč k pochopení paradigmatu DHT protokolu. Nad fungující počítačovou sítí je vytvořena síť uzlů, kde se na aplikační úrovni samostatně řeší adresování a směrování.

BT klient, který se pokusí stáhnout torrent pouze na základě infohash hodnoty, je bez již existujícího připojení do DHT sítě odkázán na tzv. „bootstrap nodes“, tedy výchozí DHT uzly, sloužící jako brána do virtuální DHT sítě, z níž chce nový uživatel získat adresy peerů, které mu poskytnou data týkající se požadovaného torrentu. BT klienti mají předdefinovány výchozí DHT uzly, např. jeden z uvedených ve Výpisu 3.6. Do DHT sítě se BT klient také



Obrázek 3.3: Virtuální (překryvná) síť (převzato z [13]).

dokáže připojit i prostřednictvím jiné právě probíhající BT komunikace. Nastavením posledního rezervního bitu BT zprávy *handshake* signalizuje připojeným peerům, aby mu byl zaslán jejich UDP port vyhrazený pro protokol DHT. Před dalším popisem protokolu DHT je ještě potřeba ujasnit vztah mezi pojmy swarm a DHT síť. Swarm je uskupení aktivních peerů, kteří mezi sebou sdílejí torrent specifikovaný infohashem. Takových swarmů může být několik a jeden BT klient se může účastnit hned několika najednou. DHT sítí je myšlena ta síť uzlů, do které se připojí každý BT klient, který disponuje DHT funkcionalitou. V této síti se pak uzel účastní distribuce adres peerů, které patří k dotazovaným torrentům definovaných infohashem.

```
router.utorrent.com:6881
router.bittorrent.com:6881
dht.transmissionbt.com:6881
```

Výpis 3.6: Adresy a porty tří známých výchozích DHT uzlů.

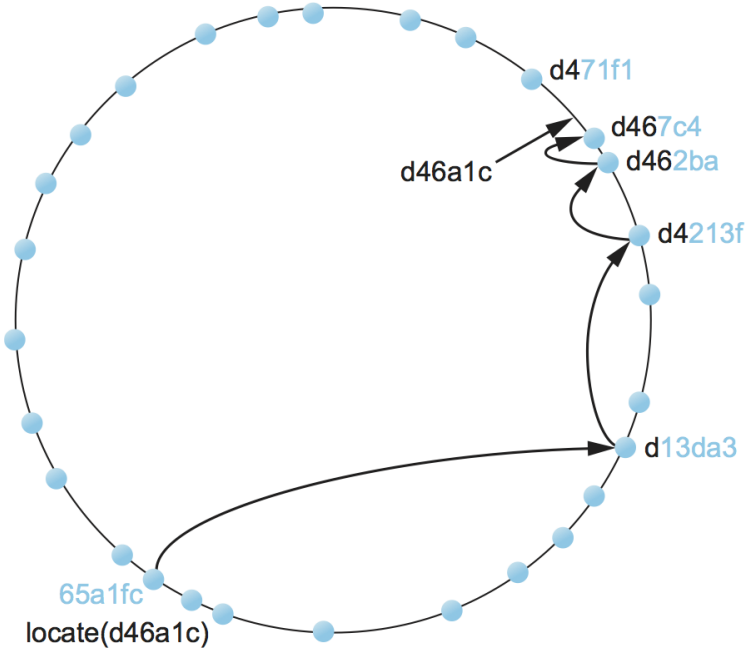
3.6.1 Princip protokolu DHT

V této sekci bude popsáno jak DHT protokol přistupuje k řešení dvou základních problémů: mapování objektů mezi distribuované uzly a směrování požadavků na mapované objekty. [13] Objektem je zde míněn infohash torrentu a k němu pak patřičný seznam peerů (IP a port). K mapování se používá hash funkce pro objekty i uzly a obě tyto entity tak mají unikátní identifikátor v prostoru 2^{160} bitů. V případě objektů se jako *ObjectID* určí již známý infohash. Uzly pak vystupují pod hashem *NodeID*, který je generován pro každý uzel náhodně. Přirozeně se nabízí použít formulaci funkce `hash(IP_adresa_uzlu)`, ale na jednom počítači může být aktivních i více různých BT klientů. Jelikož objekty i uzly vystupují ve stejném prostoru, je možné zavést pravidlo, že každý uzel mapuje objekty, které jsou mu nejblíže. Zjištění vzdáleností je určeno dle BEP 5 [6] jako bitová operace `distance(A,B) = |A xor B|`, kde výsledkem je čtyřbajtové číslo.

Na Obrázku 3.4 je znázorněn proces, při kterém se uzel s označením `65a1fc` snaží získat informace k objektu `d46a1c`. Celý kruh reprezentuje zmenšený 28 bitový prostor. Každý uzel si udržuje svou směrovací tabulku (*NodeID* a IP adresa). Při velikostech, které může DHT dosáhnout by ale bylo nevhodné udržovat adresy všech uzlů. Místo toho jsou udržovány jen uzly relativně blízko vlastní hodnoty *NodeID* (vyšší i nižší) a také několik málo uzlů vzdálených (opačná strana kruhu na Obrázku 3.4). Uzel `65a1fc` na Obrázku 3.4 tak při hledání informací o `d46a1c` kontaktuje uzel `d13da3`, protože si byl s hledaným objektem nejblíže, požadavek se tímto způsobem dostane až k uzlu s označením `d467c4`, který již má k dispozici informace o požadovaném objektu. Je důležité si též uvědomit, že komunikace provedená v této překryvné síti je v důsledku přenášena internetem, využívá se tedy nižších protokolových služeb. V krajním případě se může každý uzel fyzicky nacházet na jiném světovém kontinentě a dotazy v rámci překryvné sítě tak mohou mít vyšší odezvu.

3.6.2 Zprávy DHT protokolu

Pro výměnu UDP zpráv mezi DHT uzly se používá KRPC protokol, který definuje strukturu komunikace. KRPC zprávy jsou formátovány jako slovníky (klíč a hodnota) využívající bencoding, ten byl popsán v předchozí sekci 3.2.3. Zprávy mohou být tří typů: *query*, *response* a *error* a obsahují tři klíče nejvyšší úrovně: `"t"` pro definování ID transakce (žádost a odpověď), `"y"`



Obrázek 3.4: Lokalizace objektů v překryvné síti (převzato z [13]).

pro určení typu zprávy a `"v"` oznamující verzi klienta. V následujících podsekcích budou popsány používané DHT zprávy uvedené v rozšíření BEP 5 [6].

Ping

Zpráva ověřující dostupnost DHT protistrany. Je zde pouze jeden parametr a to *NodeID* odesílatele. Výpis 3.7 upřesňuje umístění *NodeID* parametru jako vnořený slovník. V dalších ukázkách zpráv již bude pro přehlednost použita ukázka struktury samotných parametrů. U Výpisu 3.7 je tak ještě možné sledovat plný tvar zpráv obsahující transakční ID, typ KRPC zprávy, typ DHT zprávy a parametry.

```
{"t":"aa", "y":"q", "q":"ping", "a":{"id":"A"}}

{"t":"aa", "y":"r", "r": {"id":"B"}}
```

Výpis 3.7: Zpráva *ping* a odpověď.

Find node

Tato zpráva slouží k získání informací k hledanému uzlu. Parametrem je *NodeID* odesílatele a *TargetID* cíle. Odpověď obsahuje identifikátor odesílatele a následně seznam nejbližších uzlů z jeho směrovací tabulky. Výpis 3.8 představuje situaci, kde uzel `A` posílá zprávu uzlu `B` s dotazem na uzel `C`. V odpovědi se pak uzel `A` dozví o bližších uzlech `D` a `E`. Uzly uvedené v odpovědi jsou v tzv. kompaktním tvaru (*NodeID*, IP adresa a port) - celkem 26 bajtů.

```
{"id" : "A", "target" : "C"}

{"id" : "B", "nodes" : "D, E"}
```

Výpis 3.8: Parametry zprávy *find_node* a odpověď.

Get peers

V této zprávě je konečně součástí infohash hodnota sloužící jako klíč pro cílový uzel k vydání seznamu konkrétních peerů v kompaktním tvaru 6 B (IP adresa a port), od kterých by dotazující uzel mohl získat obsah konkrétního torrentu. Pokud cílový uzel nemá konkrétní peery k infohash hodnotě, tak odpoví seznamem nejbližších uzlů jako v předchozím typu DHT zprávy *find_node*. Zde se v praxi projevuje umístění uzlů a objektů do stejného „hash prostoru“ viz předchozí Obrázek 3.4. V odpovědi se navíc nachází hodnota pro klíč s názvem *token* viz Výpis 3.9, tento atribut bude popsán v dalším typu zprávy *announce_peer*.

```
{"id" : "A", "info_hash" : "20 B infohash"}

{"id" : "B", "token" :"...", "values" : ["peer1, ..."]}
```

Výpis 3.9: Parametry zprávy *get_peers* a odpověď.

Announce peer

Tento typ DHT zprávy informuje uzel, ze kterého přišla odpověď na zprávu *get_peers* o tom, že se aktivně účastní stahování tohoto torrentu. Pro ověření

je použitý výše zmíněny *token* atribut. Uzel indexující seznam peerů pro určitý infohash si tak do seznamu přidá další IP adresu společně s portem. Výpis 3.10 ukazuje strukturu parametrů této zprávy, která díky obsažené hodnotě infohash může dobře sloužit jako ukazatel aktivního BT stahování na síti.

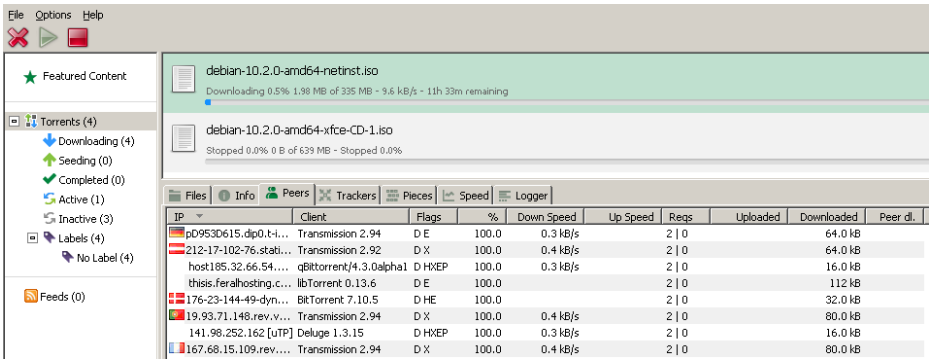
```
{"id" : "A", "info_hash" : "20 B infohash",
"port" : 53878, "token" : "..."}

{"id" : "B"}
```

Výpis 3.10: Parametry zprávy *announce_peer* a odpověď.

3.7 Klientské aplikace

Pro dokončení této kapitoly o síti BitTorrent budou představeny příklady klientských aplikací. Mezi známé klientské aplikace pro platformu Windows patří např. stejnojmenný BitTorrent, který má v kódu stejný základ jako dnes nejpoužívanější klient μTorrent. V unixovém prostředí se pak využívá klientů Transmission nebo qBittorrent. Klientský software může vystupovat jako součást jiných programů a nebo počítačových her a je využíván pro aktualizaci těchto programů. Každý si také může vyvíjet BT klienta s použitím volné knihovny libtorrent[7]. Na Obrázku 3.5 je vidět podoba již zmíněného klienta μTorrent, který se účastní stahování obsahu se zobrazenými peery.

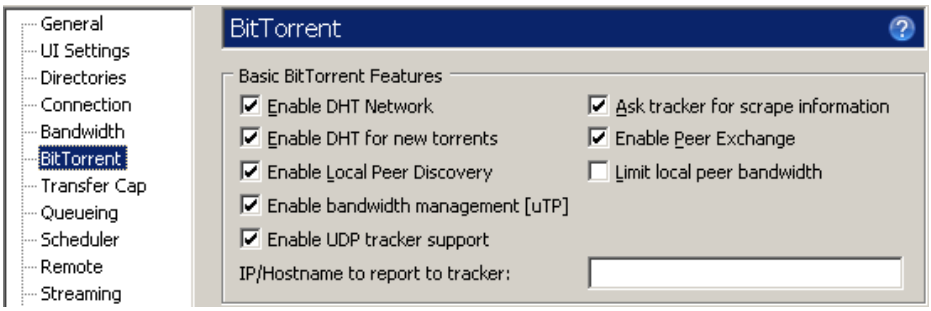


Obrázek 3.5: Stahování v klientský programu μTorrent 3.1.3.

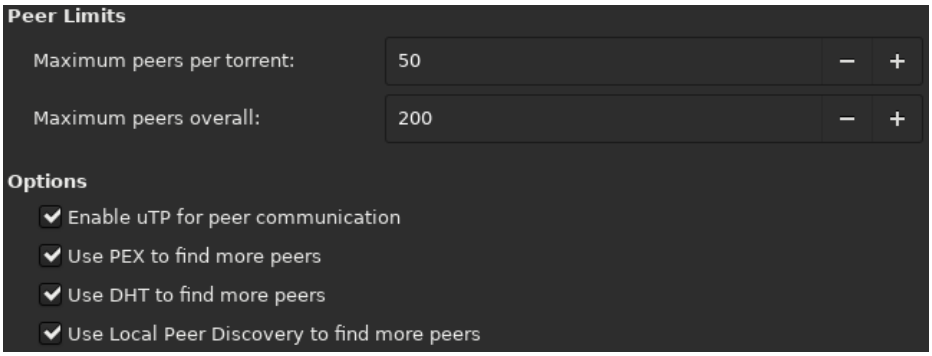
Na Obrázku 3.6 je znázorněno výchozí nastavení programu μTorrent. Jsou zde přítomné nám již známé položky jako je DHT, LSD (pod názvem „Local

[7]https://www.libtorrent.org/

Peer Discovery“) nebo μTP. Označena je zde i položka „Peer Exchange“. Jedná se o rozšíření popsáné v BEP 11 [2], které urychluje a zefektivňuje výměnu peerů v rámci swarmu, a to hlavně v prostředí s centrálním trackerem. Toto rozšíření není důležité pro implementaci síťového analyzátoru a nebude tak v této práci dále popisováno. Na Obrázku 3.7 je zachyceno nastavení pro klientský program Transmission. I zde jsou ve výchozím stavu povolena všechna rozšíření stejně jako v předchozím případě u programu μTorrent. Klientské aplikace také disponují limitem pro počet současně připojených peerů, který se pohybuje kolem hodnoty 200 viz Obrázek 3.7. Testování různých klientských aplikací ukázalo, že lze očekávat podobné chování na základě stejné množiny aktivních BT rozšíření. Implementace síťového analyzátoru podle dat zachycených pro jednu testovanou aplikaci pak bude univerzální i pro ostatní.



Obrázek 3.6: Nastavení programu μTorrent 3.1.3.



Obrázek 3.7: Nastavení programu Transmission 2.94.

4 Zachycení síťové komunikace

Aby bylo možné zkoumat a dále analyzovat provoz z počítačových sítích je nutné nejdříve takový provoz správně zachytit. V unixovém prostředí k tomu postačí nástroj Tcpdump[1], který využívá knihovnu *libpcap*. Tato knihovna s poskytnutím práv pro manipulaci se síťovým rozhraním obstará vytvoření socketu typu `PF_PACKET`, což umožní získat kopii všech přenášených dat na tomto rozhraní. Jinými slovy je vytvořen most ze síťového rozhraní přes jádro operačního systému až do uživatelského programu. Pokud nevyužíváme analýzu dat v reálném čase, je možné zachycenou síťovou komunikaci uložit jako tzv. *.pcap* soubor, formátovaný dle *libpcap* dokumentace. U síťového analyzátoru Wireshark je však větší šance setkat se s vylepšeným formátem *.pcapng*.

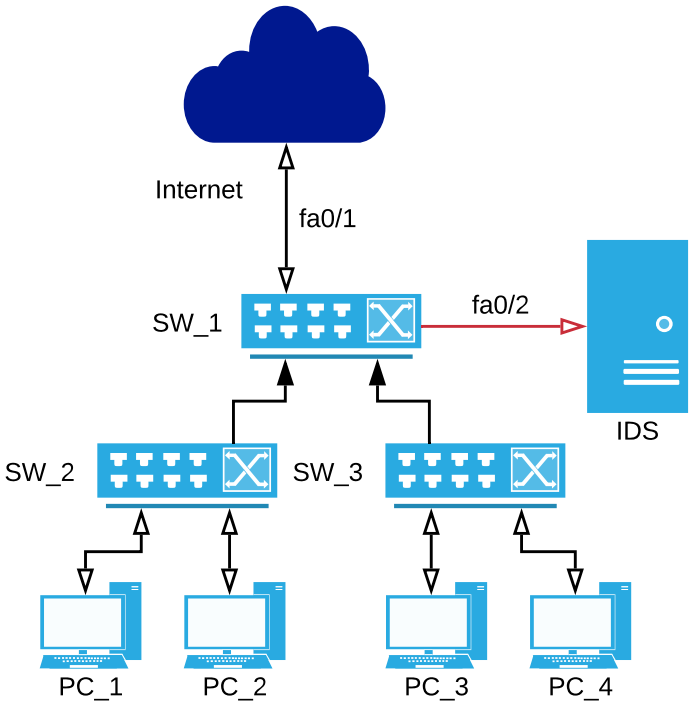
Rozsah zachycení síťové komunikace je podmíněn místem. Pokud nám jde např. o monitorování BitTorrent klienta, který je aktivní na našem počítači, postačí nástroje jako Tcpdump nebo Wireshark s právem přistoupit k lokálním síťovým rozhraním. Pokud je úkolem detekovat přítomnost BT protokolů na celém segmentu sítě s více uživateli, je zapotřebí disponovat přístupem ke komunikačním prostředkům, které pracují s jejich agregovaným provozem.

4.1 Port mirroring

Technika zrcadlení portů též nazývaná jako SPAN (Switched Port Analyzer) spočívá v duplikaci síťového provozu z jednoho zdrojového portu na druhý port určený pro monitorování. Většinou se jedná o port switche, kde se nachází tzv. uplink, tedy připojení konektivity, která je přímo využívána klientskými stanicemi. Na Obrázku 4.1 je znázorněný IDS (Intrusion Detection System) server, který monitoruje síťovou komunikaci všech čtyř uživatelů připojených na internet přes hierarchii switchů. Právě síťový port *fa0/1* s internetovou konektivitou je zde zvolen pro zrcadlení provozu do portu *fa0/2*.

[1] https://www.tcpdump.org/

Port *fa0/2* je znázorněn jednosměrnou hranou, protože zrcadlený port již nelze použít jako klasický port k přístupu do sítě. Ve výchozím nastavení síťová karta nepropouští do jádra operačního systému rámce, které nejsou adresovány pro toto zařízení. Aby bylo možné provést zachycení zrcadlené síťové komunikace je potřeba zajistit přepnutí síťového rozhraní do režimu `PROMISC`. Promiskuitní režim umožní síťové kartě předat dále ke zpracování rámce, které by normálně byly zahozeny.



Obrázek 4.1: Diagram zapojení IDS.

5 Testování analyzátorů

5.1 Příprava testovacího scénáře

Aby bylo možné objektivné srovnat výsledky několika existujících řešení pro zachycení P2P komunikace, je nejdříve potřeba definovat jednotný scénář. Tento scénář bude popisovat účast uživatele na stahování obsahu z BitTorrent sítí. Scénář počítá se čtyřmi současně stahovanými torrenty při využití dvou různých klientských aplikací: μTorrent a Transmission. Dva torrenty budou předány klientům jako *.torrent* soubory a druhé dva prostřednictvím magnet odkazu získaného z portálu The Pirate Bay viz následující seznam:

1. μTorrent 3.1.3
   - *debian-live-10.3.0-amd64-gnome.iso* (2.57 GB, *.torrent*)
   - *Frozen (2013) 720p BrRip x264 - YIFY* (844 MB, magnet)
2. Transmission 2.94
   - *debian-live-10.3.0-amd64-xfce.iso* (2.57 GB, *.torrent*)
   - *Warcraft 3 + Frozen Throne 1.21* (1.09 GB, magnet)

Pro získání dostatečného množství informací není nutné dokončit stahování výše zmíněných položek, obě klientské aplikace budou již od počátku limitovat svůj provoz na symetrických 100 kB/s a pro každý torrent budou mít povoleno využívat 50 peerů. Test bude trvat 5 minut a pak budou obě aplikace ukončeny, tak zůstane zaznamenávaný *.pcap* soubor kompaktní a snadno přenositelný. Jedním z aspektů testování bude reakce analyzátorů na pouze spuštěné klientské aplikace. Během prvotního testování se prokázalo, že i klientská aplikace bez úlohy stahování můžou aktivně využívat BT protokoly např. ke statistickým účelům. Cílem tohoto testování je získat přehled v oblasti dostupných monitorovacích systémů a porovnat jejich výsledky vzhledem k připravenému scénaři stahování.

5.2 Wireshark

Tento velice známý síťový analyzátor je k získání zdarma pod licencí GNU GPLv2 [14] (stejná jako pro Linux Kernel), díky tomuto licencování je v komunitě síťových administrátorů velmi populární a při řešení problémů spojených se síťovou komunikací je Wireshark většinou první volbou. Tento software ovšem nelze nasadit pro kontinuální monitorování sítí, ale používá se spíše jako ladící a průzkumný nástroj. V našem případě Wireshark bude sloužit k prvnímu prozkoumání zachycené P2P komunikace. V rámci scénáře byl zaznamenán soubor `scenar.pcapng` o celkové velikosti 143 MB (382 tisíc paketů).

5.2.1 Analýza vybraných protokolů

První operace se zachycenými pakety bude odebírat nepotřebné protokoly výrazem: `(!mdns && !arp && !ssdp)`. Dále jsou odstraněny všechny pakety týkající se přenosů na lokální síti, tedy se společnou zdrojovou a cílovou adresou v síti `147.228.0.0/16`. Těmito operacemi byl počet paketů redukován na 123 tisíc. K vytvoření statistiky protokolů Wireshark umožňuje selekci protokolů na základě jejich jména např. `"bittorrent"` nebo `"http"`. Protokol jako DHT je ale nutné nejdřívě povolit v sekci *Enabled Protocols* pod názvem `BT-DHT`. Implementace detekce DHT protokolu je ve verzi 2.6.8 ne úplně dokonalá a většinou se jená o tzv. *malformed packet*, tedy špatně rozpoznaný paket. Cestou jak obejít špatnou detekci je připojení výrazu `(!_ws.malformed)`, pak se již detekce zdá korektní. Tabulka 5.1 reprezentuje zpracovaný výskyt BT protokolů v rámci testování podle předdefinovaného scénáře. Ze statistických dat je výrazný poměr transportních protokolů ve prospěch UDP (74.9 %) a z něj tvoří zprávy protokolu DHT 3.75 %. Je zjevné, že zprávy HTTP a LSD se v rámci stahování vyskytují v řádech desítek. Řídicí zprávy protokolu BitTorrent v rámci TCP zaujímají podíl 1.7 %. Zbytek paketů téměř absolutně tvoří samotná data binárně přenášená přes TCP a UDP spojení.

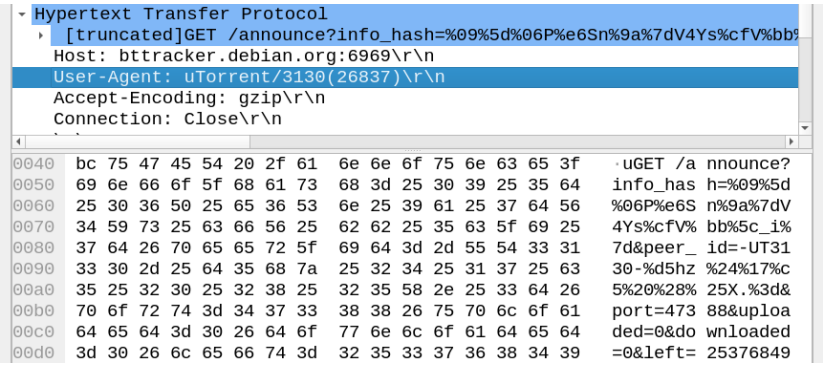
Dalším statistickým poznatkem je přítomnost 1984 „živých“ unikátních IP adres se kterými při stahování čtyř torrentů probíhala komunikace iniciovaná BT klienty. Pro navázání spojení k těmto IP adresám bylo vytvořeno 890 nových TCP a UDP portů. Tyto nezvykle vysoké charakteristiky mohou např. posloužit jako primitivní ukazatel toho, zda uživatel používá P2P.

L4 protokol	BT protokol	Počet paketů	Podíl [%]
TCP		29595	**23.9**
	BitTorrent	2140	1.7
	HTTP	20	0.0
UDP		92672	**74.9**
	DHT	4639	3.75
	LSD	4	0.0

Tabulka 5.1: Statistika podílu vybraných protokolů.

5.2.2 Příklady zachycených zpráv

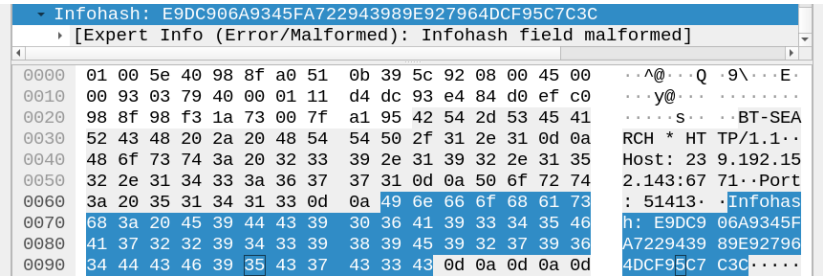
V této podsekci budou představeny konkrétní příklady vybraných zachycených zpráv z výstupu grafického rozhraní programu Wireshark. Na Obrázku 5.1 je zpráva *announce*, která byla popsána v předchozí sekci 3.2.2. Díky textovému protokolu HTTP lze snadno extrahovat informace o použitém klientovi a stahovaném torrentu podle kódované infohash hodnoty.



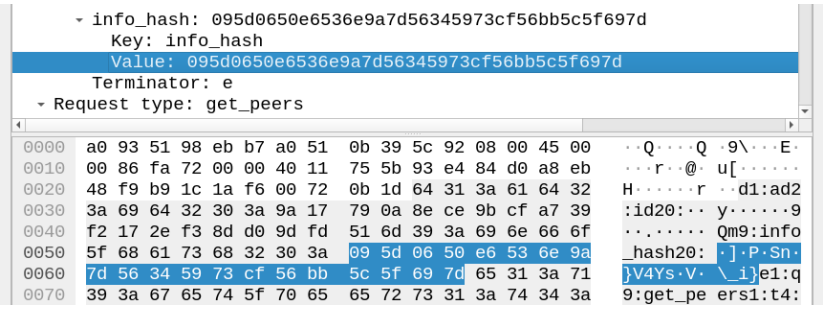
Obrázek 5.1: HTTP zpráva *announce*.

Obrázek 5.2 prezentuje pravidelně vysílanou zprávu protokolu LSD, který byl popsán v předchozí sekci 3.5. Zvýrazněná infohash hodnota může opět posloužit k označení uživatele, jenž se účastní P2P přenosů. Jedná se pouze o jeden typ zprávy a lze snadno detekovat např. podle cílové multicast adresy a následně porovnáním počátečního statického řetězce. Následný Obrázek 5.3 ukazuje obsah DHT zprávy *get_peers*, kterou se BT klienti dostávají k adresám peerů sdílející požadovaný obsah. Z předchozí sekce 3.6 víme, že přítomnost protokolu DHT znamená účast uzlu v překryvné síti a většinu z 4649 DHT zpráv tak tvoří požadavky vnějších uzlů na jiný obsah mapovaný do DHT prostoru.

Poslední ukázka zachycené zprávy je na Obrázku 5.4 *handshake* protokolu

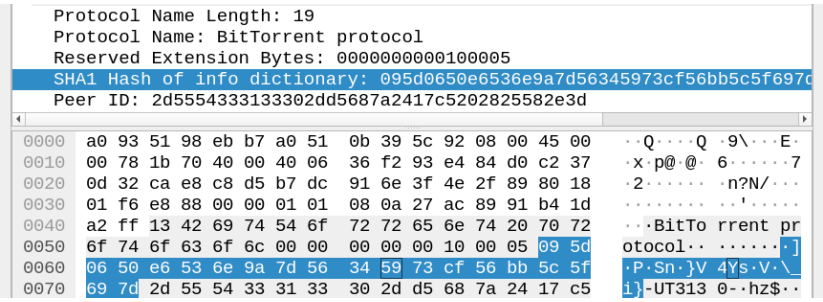


Obrázek 5.2: Zpráva LSD protokolu.



Obrázek 5.3: Zpráva *get_peers* DHT protokolu.

BitTorrent. DHT uzel poskytuje získaný seznam peerů k navázání komunikace protokolem BT, která se již týká výměny samotných dat obsažených v torrentu. Společným jmenovatelem všech čtyř ukázkových zpráv je přítomnost infohash hodnoty, podle níž lze minimálně tyto typy zpráv agregovat a přiřadit k uživateli s identickou zdrojovou IP adresou. Přítomnost zpráv *handshake* umožňuje sestavit seznam peerů, respektive cílových IP adres a portů. Tento seznam doplněný o čísla portů je dále možné využít např. k monitorování množství reálně přenesených dat. Nástrojem Wireshark byl představen praktický pohled na BT síť. Bylo ověřeno, že jednotlivé protokoly je možné zachytit a přesně detekovat uživatele využívající P2P na lokální síti.



Obrázek 5.4: Zpráva *handshake* BitTorrent protokolu.

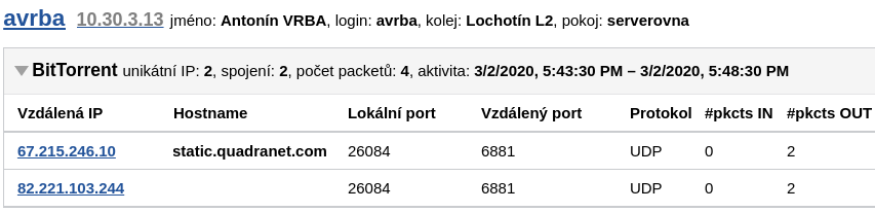
5.3 P2P sonda

V rámci projektu KNet [23] bylo implementováno interaktivní webové rozhraní k modulu *ipp2p* do iptables. Iptables je jeden z Linuxových uživatelských nástrojů pro nastavování firewall pravidel do jádra operačního systému, respektive do rozhraní *netfilter*[1]. Debian balíček `xtables-addons` obsahuje modul *xt_ipp2p*, který má parametry určující jaké P2P sítě bude detekovat. Nasazení tohoto modulu je pro zrcadlený provoz realizováno jako přidání pravidla z Výpisu 5.1 do tabulky *mangle* v sekci *PREROUTING*. Toto jediné pravidlo umožňuje paketům přicházejícím na rozhraní *eth0* projít skrz modul *ipp2p*, kde proběhne detekce protokolů sítě BT a při pozitivní detekci bude vytvořen LOG záznam do `/var/log/syslog` se specifickým prefixem.

```
/sbin/iptables -t mangle -A PREROUTING -m ipp2p --bit
    -i eth0 -j LOG --log-prefix 'P2P:BitTorrent'
```

Výpis 5.1: Iptables pravidlo používající *ipp2p* modul.

Služba rsyslog[2] přesměruje výstupní logy s prefixem do Unix socketu, kde si tyto záznamy odebírá webové rozhraní viz Obrázek 5.5, který prezentuje jak P2P sonda zobrazuje zachycenou komunikaci dvou spuštěných BT klientů z testovacího scénáře (bez aktivního stahování). P2P sonda poskytuje uživatelský komfort v rámci napojení na systému KNet (zobrazené detaily uživatele), ale po provedení scénářového testu se webové rozhraní zaplní velkým množstvím záznamů jako na Obrázku 5.6. Bez možnosti další agregace těchto záznámů je velmi snížená celková čitelnost. Za pět minut testu sonda zachytila 1485 unikátních IP adres a 3535 paketů v rámci 1640 spojení. P2P sonda doporučí vytvoření BitTorrent incidentu, pokud uživatel překročí limit 100 unikátních IP adres, 200 spojení a ještě 400 paketů.



Obrázek 5.5: Webové rozhraní P2P sondy.

[1] https://www.netfilter.org/
[2] https://www.rsyslog.com/

BitTorrent unikátní IP: **1485**, spojení: **1640**, počet packetů: **3535**, aktivita: **3/2/2020, 5:43:30 PM**

Vzdálená IP	Hostname	Lokální port	Vzdálený port	Protokol	#pkcts IN	#pkcts OUT
46.119.186.122	46-119-186-122.brc	26084	18069	UDP	2	4
79.129.199.27	athedsl-4445995.h	26084	62786	UDP	1	1
184.167.0.50	184-167-000-050.re	61603	27050	TCP	0	1
24.144.103.29	user-0c90pot.cabl	26084	51413	UDP	1	1
219.140.59.36		26084	34316	UDP	0	1

Obrázek 5.6: Webové rozhraní P2P sondy.

Správce sítě využívající výstupu P2P sondy ze záznamů jako na Obrázku 5.6 určí orientační rozsah BT komunikace, ale po vytvoření incidentu v systému KNet může vzniknout spor např. o počet stahovaných torrentů nebo zda se jednalo o aktualizaci softwaru a ne o šíření obsahu s autorskými právy. Pokud se zaměříme na kód[3] uvnitř modulu *ipp2p* (cca 1000 řádek v jazyce C), tak nalezneme velké množství jednoduchých porovnání zaměřených na obsah paketů. Výpis 5.2 reprezentuje zjednodušenou ukázku tří vybraných porovnání navracející číselný kód v případě pozitivní detekce. První podmínka kontroluje délku obsahu paketu a poté porovnává obsah samotný v síťové endianitě[4], kde se v tomto případě jedná o konkrétní řídicí zprávu BT protokolu. Detekce DHT protokolu je prováděna komparací řetězce `"d2:id20:"` bez ohledu na typ zprávy a směr komunikace. BT zpráva *handshake* je pak v poslední ukázce detekována podle obsaženého řetězce `"BitTorrent protocol"`. Zkoumáním kódu modulu *ipp2p* lze jednoznačně říci, že se jedná o minimalistickou a bezestavovou detekci sítě BitTorrent bez rozlišování typů zpráv a BT protokolů. P2P sondu je tak možné vnímat jako „dobrý základ“ pro detekci P2P sítí.

```
if (len >= 32 && get_u32(pkt, 8) == htonl(0x00000402))
        return IPP2P_BIT + 1;

if (memcmp(pkt + 4, "d2:id20:", 8) == 0)
        return IPP2P_BIT + 2;

if (memcmp(pkt + 1, "BitTorrent protocol", 19) == 0)
        return IPP2P_BIT + 3;
```

Výpis 5.2: Ukázka kódu modulu *ipp2p* v jazyce C.

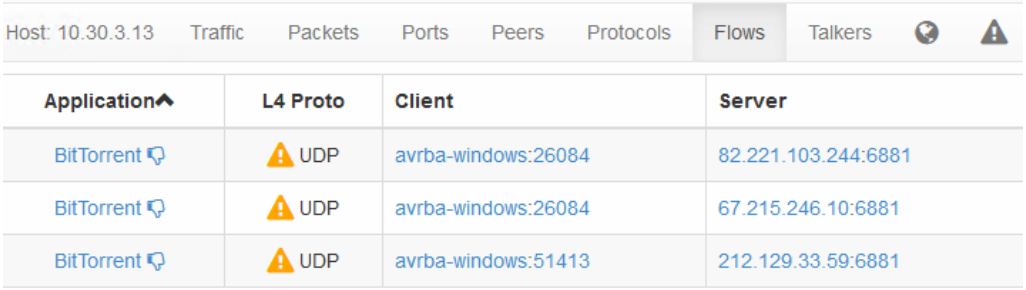
[3] https://sourceforge.net/p/xtables-addons

[4] Endianita je pořadí bajtů při reprezentaci čísel v operační paměti počítače.

5.4 Ntopng

Ntop je známá sada nástrojů pro monitorování síťové komunikace. Jelikož se jedná o projekt s otevřenými GitHub repozitáři[5] je možné nástroje zdarma vyzkoušet ve školním prostředí, přičemž ntop nabízí i zpoplatněné verze Enterprise. V rámci testování byla zvolena verze 3.8.2 ze stabilní vývojové větve. Ntopng je analyzátor síťového provozu s webovým rozhraním, které je napojené na výstup zprostředkovaný knihovnou nDPI. Tato knihovna se specializuje na detekci protokolů aplikační úrovně [10]. Nahlédnutím do repozitáře nDPI[6] je patrné, že ze strany komunity probíhají neustálé aktualizace dílčích částí kódu pro detekci specifických protokolů.

Na Obrázku 5.7 je webové rozhraní ntopng, které vypisuje zachycené BitTorrent flow uživatele s pouze spuštěnými BT klienty dle scénáře. Flow v kontextu nDPI knihovny je komunikační entita určená pěticí parametrů: typ transportní protokolu a dvojice IP adres s porty (zdrojové a cílové). Po provedení scénářového testu zachytil analyzátor ntopng stovky flow relací zařazených do kategorie BitTorrent viz Obrázek 5.8. Ntopng navíc proti P2P sondě extrahuje infohash a velmi orientačně znázorňuje množství přenesených dat.



Obrázek 5.7: Zobrazení flow relací v ntopng.

I přes přidanou hodnotu ntopng v podobě přehledné vizualizace BT přenosů včetně agregace do různých grafů je nasazení tohoto softwaru v prostředí kolejní sítě KNet nevhodné hned z několika důvodů. Identifikace uživatelů je v ntopng možná jen podle IP adres bez podpory reverzních doménových jmen. Ve webovém prostředí nelze nijak nastavit upozornění nebo filtraci pro uživatele právě používající BitTorrent síť. Největším problémem je však prohlížení historie monitorování. Správci sítě je umožněno nahlédnout detailně pouze do flow, která jsou posledních několik minut aktivní (sledování

[5] https://github.com/ntop
[6] https://github.com/ntop/nDPI/tree/dev/src/lib/protocols

L4 Proto	Client	Server	Duration	Total Bytes	Info
UDP	avrba-windows:51413	109.226.209.73:54732	05:31	246.47 KB	e9dc906a9345fa722943989e...
UDP	avrba-windows:51413	121.58.221.194:45877	05:37	167.32 KB	91acf1734f3bd8e64d4430cf...
UDP	avrba-windows:51413	77.46.177.151:28958	05:39	280.61 KB	91acf1734f3bd8e64d4430cf...
UDP	avrba-windows:51413	46.249.166.114:12825	03:08	4.81 KB	91acf1734f3bd8e64d4430cf...
TCP	avrba-windows:51412	35.136.125.195:58019	< 1 sec	422 Bytes	25b4cd46e389e96f80ee42e4...
TCP	avrba-windows:51383	203.221.248.28:25155	00:05	1.75 KB	25b4cd46e389e96f80ee42e4...

Obrázek 5.8: Část zachycených flow z kategorie BitTorrent.

flow v reálném čase) a je možné je i exportovat jako *.pcap* soubor. Při požadavku na zkontrolování uživatelů používající protokoly BT za posledních několik týdnů tak ntopng nelze použít.

5.5 Suricata

Mezi další IDS vytvářené otevřenou komunitou patří Suricata[7]. Jedná se o velmi modulární a výkonnou platformu pro monitorování sítě. Pro distribuci Debian 10 je z oficiálních repozitářů dostupný stejnojmenný balíček ve verzi 4.1.2. Po instalaci balíčku stačí jen nastavit název rozhraní se zrcadleným síťovým provozem do konfiguračního souboru `suricata.yaml` a spustit software Suricata jako službu. Ve výchozím nastavení obsahuje Suricata řádově tisíce pravidel, které jsou definovány v externích souborech, na které odkazuje konfigurace. Výpis 5.3 představuje ukázku syntaxe pravidla ze souboru `emerging-p2p.rules` (celkem 278 pravidel pro detekci P2P sítí). Dle dokumentace [12] jsou pravidla rozdělena do tří hlavních sekcí: akce, základní definice a volitelné parametry. Ve Výpisu 5.3 je zvolena akce *alert*, která při pozitivní detekci vytvoří záznam se specifickým prefixem do souboru `/var/log/suricata/fast.log`. Do základní definice spadá protokol UDP, libovolná zdrojová IP adresa z privátního rozsahu a libovolná cílová adresa z rozsahu veřejného. Součástí je též kontrola obsahu paketu vůči ostatním parametrům tohoto pravidla.

Stejně jako v předchozích dvou případech i Suricata dokáže v podobné míře do logovacího souboru zaznamenat aktivitu pouze zapnutých BT klientů a po dokončení scénářového testu bylo zachyceno celkem 47 případů detekce BT protokolů. Ve Výpisu 5.4 je prezentována část výsledků, ze kte-

[7] https://suricata-ids.org/

```
alert udp $HOME_NET any -> $EXTERNAL_NET any
(msg:"ET P2P BitTorrent DHT find_node request";
content:"d1|3a|ad2|3a|id20|3a|";
...
```

Výpis 5.3: Ukázka části pravidla pro detekci DHT zprávy.

rých je podlenázvu zřejmé, které BT klienty uživatel při stahování používal a dále přítomnost protokolu DHT včetně typu použitých zpráv. Ve výchozím stavu jsou pravidla na detekci BT protokolů jednoduchá a bylo by nutné dopsat příslušná pravidla např. pro extrakci hodnoty infohash. Suricata též umožňuje pracovat s flow relacemi, ale bylo by nutné správně přiřazovat jednotlivá flow k BT provozu. Suricata ve výchozím stavu nedisponuje žádným webovým rozhraním, pro vyhledávání nebo agregaci nasbíraných dat je doporučeno přeposílat logovací zprávy do centrálního úložiště logů jako je např. Graylog[8].

```
BTWebClient uTorrent in use
{TCP} 10.10.80.21:52464 -> 104.24.98.205:80
Bittorrent P2P Client User-Agent (Transmission/1.x)
{TCP} 10.10.80.21:59344 -> 104.27.130.137:80
P2P BitTorrent DHT ping request
{UDP} 10.10.80.21:47388 -> 82.221.103.244:6881
P2P BitTorrent DHT announce_peers request
{UDP} 10.10.80.21:47388 -> 211.171.89.230:41172
...
```

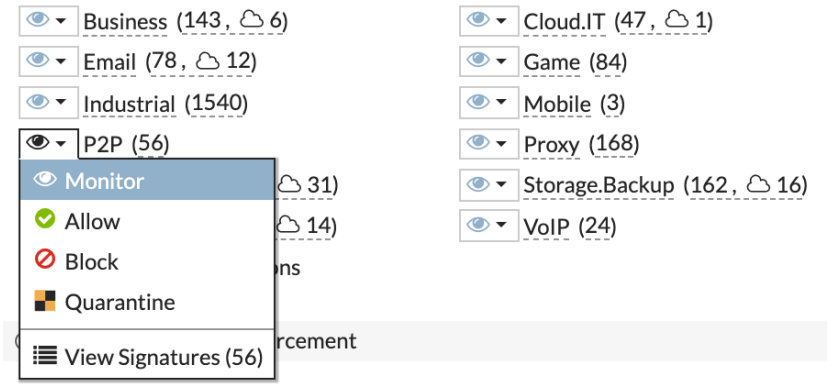
Výpis 5.4: Část výstupu z IDS Suricata.

5.6 Fortinet FortiGate 3700D

V rámci testování byl Západočeskou univerzitou poskytnut profesionální firewall FortiGate 3700D[9] s propustností až 160 Gbit/s. Výstupy z analýzy provozu jsou dále uchovávány k pozdějšímu zpracování zařízením FortiAnalyzer 3000E. V rámci DPI funkcionality nabízí FortiGate monitorování nebo blokování konkrétních aplikací podle jejich signatury, kterých má v databázi řádově tisíce. Na Obrázku 5.9 je znázorněna část webového rozhraní

[9]https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiGate_3700D.pdf

pro výběr aplikačního filtru na zařízení FortiGate. Kategorie P2P obsahuje 56 položek a kromě BitTorrentu jsou zde také sítě jako Gnutella nebo Direct Connect.



Obrázek 5.9: Výběr aplikací k monitorování na zařízení FortiGate.

V rámci testování podle scénáře bylo zaznamenáno 1726 shod s BitTorrent signaturou včetně pozitivní detekce pouze zapnutých BT klientů. Na Obrázku 5.10 je zobrazena část výstupu, obsahující všechny informace, které FortiGate implementace poskytuje. Tyto informace postačí k prokázání toho, že se na síti vyskytuje uživatel s BT aplikací. Dle Fortinet databáze aplikací[10] se jedná pouze o signatury zpráv samotného protokolu BitTorrent. Z rozhraní ani z úplných LOG záznamů není možné získat hodnotu infohash torrentu. Webové rozhraní též neumožňuje více agregovat BT přenosy a ani neposkytuje informace o objemu stažených dat torrentů. Položky ve sloupečku `Sent/Received` z Obrázku 5.10 odráží pouze velikost zpráv protokolu BitTorrent.

Destination IP = "147.228.244.231" | Application = "BitTorrent" | Action = "all-accept" | Add Filter

#	Source	Destination IP	Service	Application	Sent/Received
1	104.245.79.213	147.228.244.231	tcp/47388	BitTorrent	34.7 KB/1.8 KB
2	178.48.168.156	147.228.244.231	tcp/47388	BitTorrent	34.4 KB/1.5 KB
3	80.108.204.6	147.228.244.231	tcp/47388	BitTorrent	34.4 KB/2.5 KB
4	118.208.211.221	147.228.244.231	tcp/47388	BitTorrent	34.5 KB/1.8 KB
5	122.162.224.118	147.228.244.231	tcp/47388	BitTorrent	35.9 KB/2.7 KB
6	97.113.224.27	147.228.244.231	tcp/47388	BitTorrent	7.3 KB/2.1 KB
7	118.208.211.221	147.228.244.231	tcp/47388	BitTorrent	34.5 KB/1.8 KB
8	122.162.224.118	147.228.244.231	tcp/47388	BitTorrent	35.9 KB/2.7 KB

Obrázek 5.10: Výběr aplikací k monitorování na zařízení FortiGate.

Na první pohled se jeví úroveň detekce profesionálního zařízení FortiGate horší než ta, kterou nabízí komunitní nástroje zdarma. Kde ovšem toto za-

[10]https://fortiguard.com/appcontrol/6/bittorrent

řízení exceluje je dynamické vytváření pravidel pro úplnou blokaci BT provozu. Na předchozím Obrázku 5.9 je na výběr i položka „Block“ a pak je provoz BitTorrent zablokován. Testování ukázalo, že tato blokace dostačuje a BT klientské aplikace končily stahování s chybou nebo stahování neprobíhalo z důvodu nemožnosti navázat spojení se vzdálenými peery. Aby bylo možné rozpoznat např. DHT protokol, FortiGate nabízí rozhraní pro tvorbu signatur s vlastní syntaxí.

5.7 Vyhodnocení testování

V této kapitole byl představen testovací scénář, jenž byl v sekci 5.2 analyticky prozkoumán programem Wireshark. Následovalo testování současné P2P sondy, nástroje ntopng, Suricata a nakonec bylo k testování zapůjčeno profesionální zařízení FortiGate 3700D. Všechny zmíněné platformy s různou měrou, ale vždy spolehlivě, detekují BitTorrent provoz, jak znázorňuje Tabulka 5.2.

Velké rozdíly nalézáme v možnostech zobrazení zachycených informací určené pro správce sítě, kteří řeší blokování uživatelů porušujících používáním sítě BitTorrent autorská práva. Pouze ntopng dokáže ve výchozím stavu extrahovat infohash torrentu, ale tento systém se potýká s problémy ohledně archivování zaznamenaných událostí. Z pohledu agregace zachycených dat nejlépe vychází současně nasazená P2P sonda. Pakety BT komunikace vizuálně agreguje podle uživatele a dále kumulativně podle cílové adresy. P2P sonda se ale potýká s limity detekce firewall modulu *ipp2p* a neposkytuje žádné informace o objemu stažených dat, stějně jako všechny ostatní testované systémy. Pro splnění všech kritérií z Tabulky 5.2 bude nutné navrhnout a implementovat nový síťový analyzátor zaměřený specificky na BitTorrent.

	P2P sonda	**ntopng**	**Suricata**	**FortiGate**
BitTorrent detekce	✓	✓	✓	✓
Agregace událostí	✓	✗	✗	✗
Extrakce infohash hodnoty	✗	✓	✗	✗
Přenesená torrent data	✗	✗	✗	✗
Webové rozhraní	✓	✓	✗	✓
Ukládání historie událostí	✓	✗	✓	✓

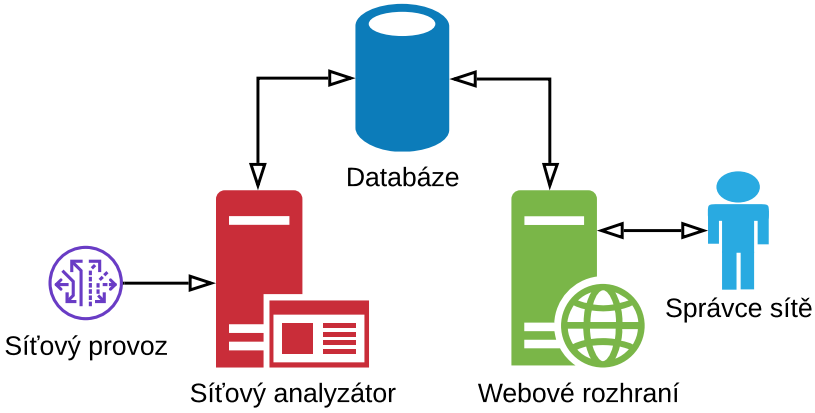
Tabulka 5.2: Porovnání testovaných systémů.

6 Návrh

Předmětem této kapitoly bude diskuze nad návrhem síťového analyzátoru zaměřeného na síť BitTorrent. Na základě předchozích kapitol jsou známé jednotlivé protokoly umožňující BT přenosy a během testování již existujících analyzátorů je možné pozorovat implementační rozdíly a hlavní komponenty, ze kterých se analyzátory skládají. Návrh vlastního řešení lze rozdělit na následující tři základní části:

1. **Síťový analyzátor**
2. **Databáze**
3. **Webové rozhraní**

Na Obrázku 6.1 je znázorněn vztah tří výše zmíněných komponent. Ze zrcadleného síťového provozu rozpozná analyzátor charakteristiky protokolů BitTorrent a výstup zaznamená do relační databáze. Správce sítě přistupuje do databáze prostřednictvím webového rozhraní. Cílová platforma pro všechny zmíněné komponenty bude Linuxová distribuce Debian. Podrobnější návrh těchto komponent bude rozveden v dalších sekcích této kapitoly.



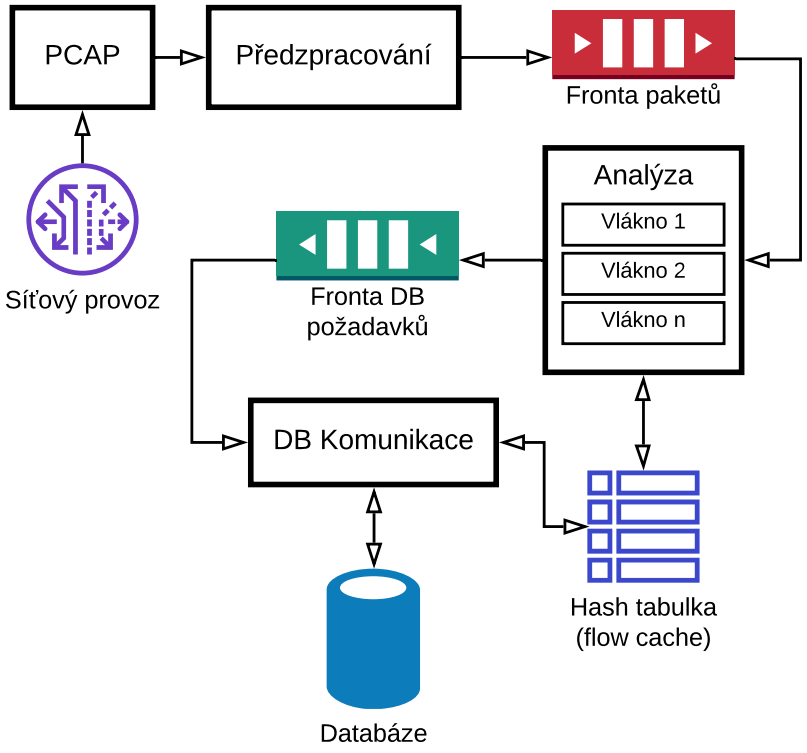
Obrázek 6.1: Vztah základních prvků analyzátoru.

6.1 Síťový analyzátor

Hlavním aspektem síťového analyzátoru je jeho výkon. Kolejní síť KNet je v různých lokalitách vybavena konektivitou s kapacitou až 10 Gbit/s, což může v závislosti na povaze síťového provozu znamenat jednotky milionů paketů za sekundu. Práce s databází v reálném čase tak není možná a analyzátor musí mezi svými logickými celky implementovat systém front, které zabezpečí dostatečně robustní rozhraní pro výměnu zpráv a požadavků. V jednotlivých modulech analyzátoru se automaticky počítá s nasazením paralelního zpracování dat. Bude se tak jednat o vícevláknovou aplikaci. Ideálním programovacím jazykem pro takové nasazení je C++. V následujícím seznamu jsou zaznamenány požadavky důležité k návrhu síťového analyzátoru:

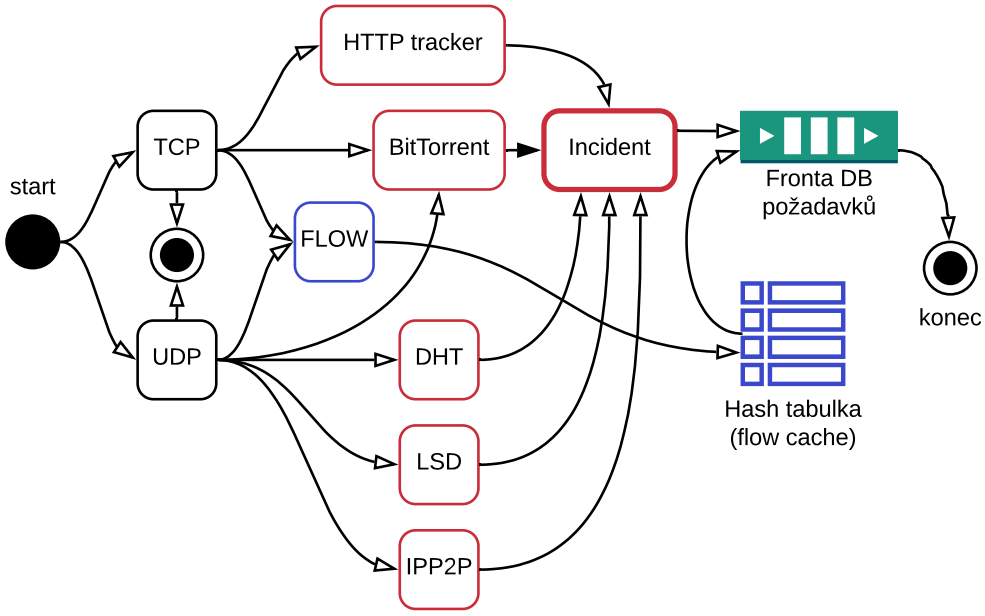
- Analyzovat síťový provoz v reálném čase z 10 Gbit/s rozhraní,
- Umožnit i „offline“ analýzu *.pcap* souborů,
- Zachytit infohash, peery a agregovat protokoly sítě BitTorrent,
  - HTTP tracker
  - LSD
  - BitTorrent
  - DHT
- Určit orientační objem přenášených dat jednotlivých torrentů,
- Výsledky analýzy a statistické údaje průběžně ukládat do databáze.

Na základě požadavků lze sestavit diagram čitelný z Obrázku 6.2. V Kapitole 4 byla zmíněna knihovna *libpcap*, podle které se nazývá první vstupní modul, jenž předává zachycené pakety k předzpracování. Předzpracování alokuje paměťové zdroje pro zachycené pakety a umístí odkazy na tyto struktury do fronty. Samotná vícevláknová analýza odebírá pakety z fronty a aplikuje na ně pravidla, podle kterých se určuje BT provoz. Pokud je detekce pozitivní, je vytvořen DB požadavek a zařazen fronty. Vlákno obsluhující komunikaci s databází zpracovává postupně požadavky z fronty. Specifickou součástí pro výpočet objemu přenesených dat je hash tabulka, podle které jednotlivá vlákna analyzátoru kumulují velikosti obsahu paketů. Tato flow cache je pravidelně databázovým vláknem aktualizována a její obsah je synchronizován s položkami v databázi.



Obrázek 6.2: Vztah základních prvků analyzátoru.

Pokud se zaměříme pouze na stavy uvnitř vlákna analyzátoru, je zapotřebí sestavit stavový diagram viz Obrázek 6.3. Paket vybraný z fronty je klasifikován nejdříve dle protokolu transportní úrovně a poté podle typu incidentu. Drtivá většina paketů skončí bez povšimnutí v prvním konečném stavu a vlákno může okamžitě pokračovat na další pakety z fronty. Pakety také mohou zamířit do stavu monitorování aktivních flow využívající hash tabulku. Diagram na Obrázku 6.3 obsahuje dvě zatím nevysvětlené položky: incident a IPP2P. Incidentem je označena entita organizující výskyty BT protokolů podle infohash hodnoty. IPP2P je označení pro různé BitTorrent zprávy, které nelze jednozačně přiřadit k infohash hodnotě, ale stále je snaha tyto zprávy monitorovat a přiřadit pro uživatele jako neidentifikovatelný, avšak stále platný BT incident. Jak již bylo osvětleno přechozím Obrázkem 6.2, analýzu paralelně provádí více vláken a je tak potřeba předvídat přístupy k sdíleným datům např. již zmíněným frontám a ošetřit toto jednání aby nedocházelo k souběhu.



Obrázek 6.3: Stavový diagram paketu uvnitř vlákna analyzátoru.

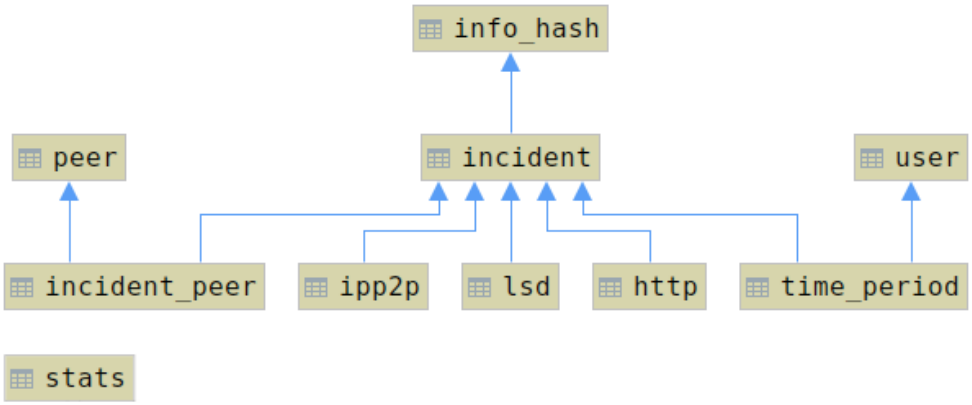
6.2 Databáze

Prostředkem pro archivaci zachycených incidentů a statistik bude relační databáze MariaDB[1]. Jedná se o otevřenou komunitní verzi známé databáze MySQL. Účelem je zachycené informace přehledně organizovat, aby webové rozhraní jednoduše dokázalo výsledky prezentovat uživateli.

Na Obrázku 6.4 je ER (Entity Relationship) diagram vytvořený na základě struktury databáze, kterou používá P2P sonda [23]. Všechny relace jsou naznačeny šipkou a po jejím směru jsou k sobě entity ve vztahu `1:N`. Předchozí návrh byl rozšířen ze 4 tabulek na 10 se zaměřením na incident identifikovaný podle infohash hodnoty jako hlavní entita. Tomu také odpovídá relace mezi tabulkou `incident` a `info_hash`. Dynamicky vytvářený uživatel z tabulky `user` je přes rozkladovou tabulku pro určení času `time_period` spojen s incidentem. Na takový incident odkazují tabulky `ipp2p`, `lsd` a `http`, ve kterých se akumulují jednotlivé zachycené zprávy a signatury. K incidentu se také vážou peery se kterými uživatel komunikuje. K tabulce `peer` je připojen incident pomocí rozkladové tabulky `incident_peer`. Díky takovéto konstrukci jsou IP adresy peerů unikátní v rámci celého systému a až roz-

[1] https://mariadb.org/

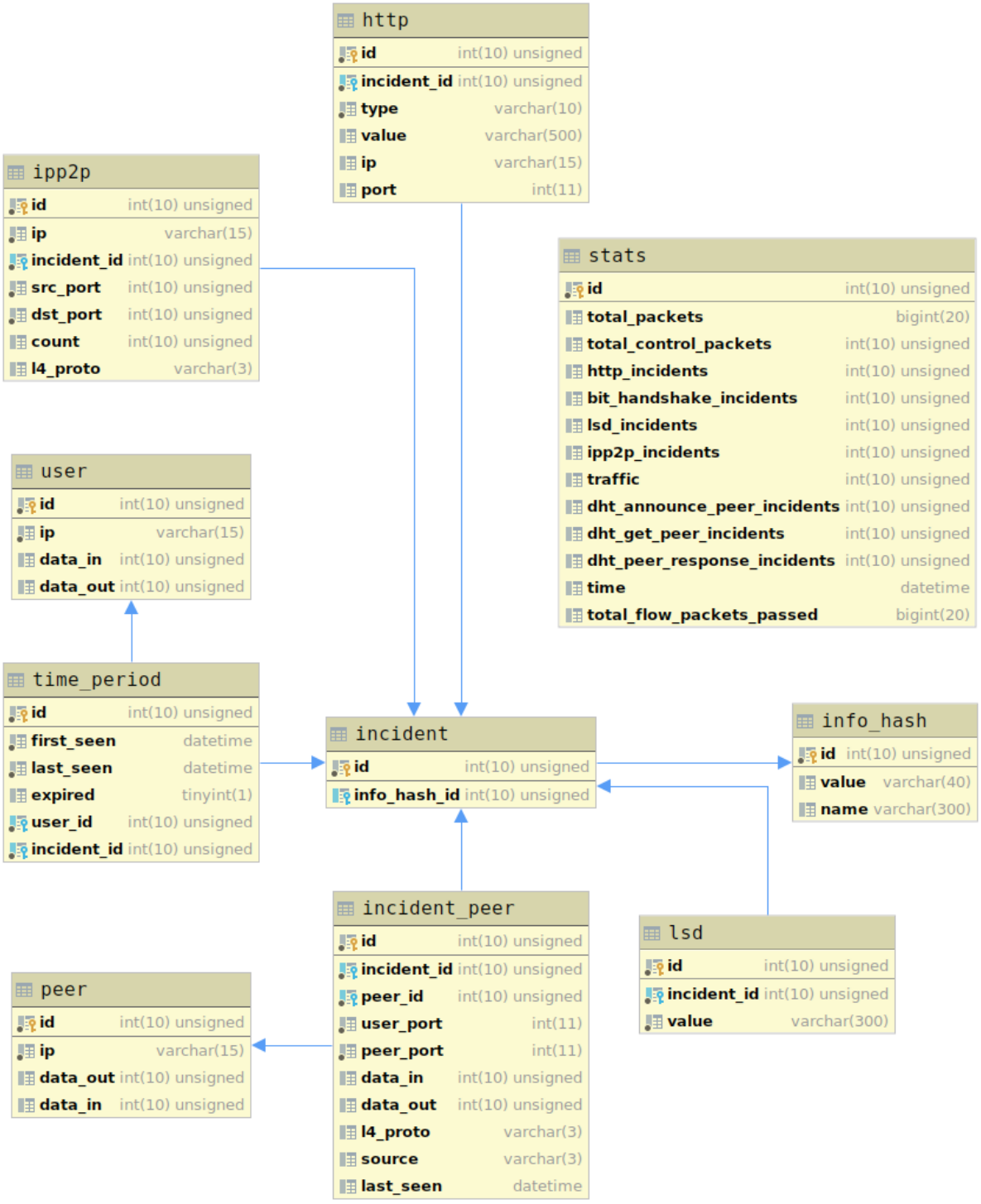
kladová tabulka specifikuje parametry spojení a tím jednoznačně definuje aktivní flow. Poslední tabulkou, která ještě nebyla popsána je `stats`. Slouží pro pravidelné ukládání statistických dat. Tabulka `stats` nemá žádné vazby na ostatní tabulky.



Obrázek 6.4: ER diagram databázových tabulek.

Předchozí diagram je v úplné podobě na Obrázku 6.5, zde jsou k vidění datové typy a názvy jednotlivých sloupců tabulek databázového návrhu. Význam většiny položek tohoto diagramu by měl být zřejmý díky předchozímu popsání principů a postupů v rámci této práce. Některé položky si ovšem zaslouží dodatečné vysvětlení.

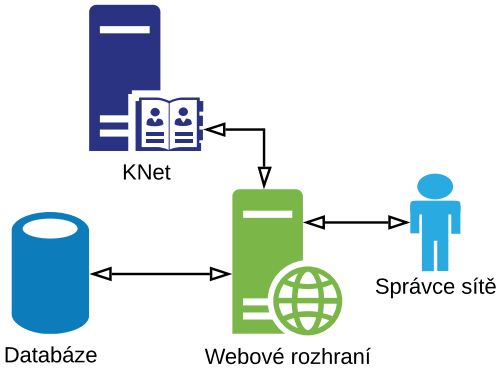
V tabulkách: `user`, `incident_peer` a `peer` se vyskytují stejné sloupečky `data_in` a `data_out`. Tyto položky slouží ke kumulaci objemu potvrzených přenesených torrent dat na různých úrovních: uživatel, flow nebo peer. Díky těmto informacím lze velmi rychle určit např. to, který vzdálený peer sdílel do lokální sítě nejvíce dat. Incident je určen časovým úsekem a pro jeho uzavření je přítomna položka `time_period.expired`. Analyzátor při každé nové aktualizaci konkrétního incidentu nastaví položku `time_period.last_seen` na aktuální čas. Databázové vlákno pak periodicky kontroluje všechny incidenty a určuje jejich expiraci porovnáním poslední aktivity a nastavené hodnoty prodlevy. Podobný princip využití časových značek se používá pro zjištění neaktivního datového přenosu. Položka `incident_peer.last_seen` je aktualizována kdykoli je zaznamenán datový tok. Při sestavování flow cache hash tabulky tak lze použít jen flow záznamy aktivní např. před minutou a nebude tak docházet k případnému přeplnění této tabulky neaktivními prvky.



Obrázek 6.5: Položky jednotlivých tabulek databáze.

6.3 Webové rozhraní

Na základě předchozího testování různých analyzátorů a zvláště současně nasazeného webového rozhraní P2P sondy je možné specifikovat požadavky návrhu vlastního webového rozhraní, které bude komplementem pro již navržený analyzátor a databázi. Na Obrázku 6.6 je naznačena role systému KNet. Práce [23] definuje a dokumentuje rozsáhlé možnosti API rozhraní. API rozhraní bude primárně určené k identifikaci uživatele, kterého podle IP adresy určí analyzátor. Z webového rozhraní pak také bude umožněné přesměrování na případné zablokování uživatele. Pro implementaci byl zvolen interpretovaný programovací jazyk PHP, a to hlavně kvůli snadné uchopitelnosti pro začátečníka v oblasti vývoje webových aplikací a relativní jednoduchosti oproti technologii jako je Node.js[2].



Obrázek 6.6: Vztah webového rozhraní a systému KNet.

V následujícím seznamu budou specifikovány požadavky na zobrazovací schopnosti rozhraní:

- Možnosti zadání časového rozmezí pro vyhledání incidentů,
- Zobrazení filtrovaných incidentů včetně grafu s jejich výskytem,
- Přítomnost infohash hodnoty a množství doposud přenesených dat,
- Incident rozdělen na podsekce HTTP, BitTorrent, IPP2P a DHT,
- Označení torrentu vlastním názvem nebo poznámkou.

[2]https://nodejs.org/

7 Implementace

V této kapitole budou popsány konkrétní sekce a struktury jednotlivých implementací analyzátoru i webového rozhraní. Pro celý projekt byl zřízen Git[1] repozitář na školním GitLab[2] serveru pro sledování postupu implementace, verzování a případné hlášení chyb. Tento repozitář je také v poslední verzi přítomen na přiloženém CD viz Příloha A. Návod pro nasazení analyzátoru na server je v Příloze B.

7.1 Síťový analyzátor

V průběhu návrhu byl vybrán programovací jazyk C++ 17 vzhledem k výkonnostním požadavkům. V následujícím seznamu jsou přehledně popsány jednotlivé moduly programu nazvané podle stejnojmenné dvojice souborů *.cpp* a *.h*, kromě souboru *main.cpp*. Další podsekce pak lépe ilustrují význam a funkčnost těchto modulů.

Main - inicializace a ukončování všech komponent analyzátoru.

Logger - instance pro logování do souboru.

Config - načítání konfigurace ze souboru.

Bencode - dekódování zpráv v bencode formátu.

Packet - zpracování a alokace vstupních paketů.

Callbacks - analýza paketů pro identifikaci BT provozu.

Connector - rozhraní mezi vlákny analyzátoru a vytvářením incidentů.

Sql - napojení na databázi a implementace jednotlivých dotazů.

7.1.1 Main

Spuštění programu v C++ začíná vždy funkcí *main()*. Ta v tomto případě přijímá v rámci spuštění dva argumenty: umístění souboru pro logování

[1] https://git-scm.com/

[2] https://ipmil.civ.zcu.cz/knet/p2p-analyzer

a umístění konfiguračního souboru. Jsou zde inicializována vlákna analyzátoru včetně vlákna pro práci s databází. Součástí počátečních příprav analyzátoru je také práce s *pcap* rozhraním pro zachytávání paketů. *Pcap* je opatřen vstupním filtrem v Berkeley syntaxi (stejně jako používá *tcpdump*) např. v podobě `(tcp or udp) and net 10.0.0.0/8`. Filtr je kompilován a přiřazen k instanci zachytávání paketů společně s ukazatelem na funkci *got_packet()* v modulu `callbacks`, do které jsou zasílány všechny pakety splňující podmínky vstupního filtru. Aby program mohl být korektně ukončen, reaguje na SIGINT signál vyvolaný např. stiskem kláves CTRL + C příslušnou funkcí *int_handler()*, která obstarává ukončení všech vláken programu.

7.1.2 Logger

Pro logování je přítomna třída podle návrhového vzoru Singleton (jedináček). V rámci celého programu tak existuje pouze jedna instance, se kterou se může pracovat jako `LOG->Info("záznam")` díky makru nahrazující získání instance `Logger::getLogger()`. Tato třída také zajišťuje sběr statistických dat z různých částí programu a sbírané statistiky jsou pravidelně využívány databázovým vláknem. Současně je přítomno i vlastní vlákno pro výpis pravidelných statistik do standardního výstupu.

7.1.3 Config

Druhou a poslední třídou v tomto programu je též Singleton třída *Config*. Výpis 7.1 obsahuje všechny konfigurovatelné položky analyzátoru potřebné před spuštěním. Základní je nastavení položky `mode`, podle které je program v režimu čtení paketů ze souboru *.pcapng* nebo z příslušného síťového rozhraní v reálném čase. Parametr pod názvem `network` určuje IP rozsah lokální sítě, jejíž příchozí a odchozí provoz je analyzátorem zpracováván. Následují parametry k navázání databázového spojení a dále jednotlivé funkcionality analyzátoru aktivované hodnotami `0` nebo `1`. Parametr `FLOW` aktivuje monitorování datových toků na základě aktivních peerů přiřazených k incidentu. To, jak dlouho jsou drženy neaktivní flow položky v hash tabulce určuje parametr `flow_activity_delay`. Incident je označený jako expirovaný na základě hodnoty `incident_expiration` parametru. V závěru konfiguračního souboru je nastavení počtu vláken, které paralelně analyzují pakety. Po na-

čtení souboru s konfigurací proběhne zpracování a všechny hodnoty jsou dostupné v programu přes příslušné metody.

```
#[file , realtime]
mode:realtime

pcap_file:../resources/all.pcapng
net_interface:wlp3s0
network:10.0.0.0/24

db_hostaname:localhost
db_port:3306
db_user:user
db_pass:password
db_name:p2p_analyzer

#[0, 1]
HTTP:1
LSD:1
IPP2P:1
BIT:1
DHT:1
FLOW:1

#[minutes]
incident_expiration:60
flow_activity_delay:1

processing_threads:5
```

Výpis 7.1: Položky konfiguračního souboru.

7.1.4 Bencode

Pro dekódování bencode zpráv, které používá např. protokol DHT, byla použita již existující knihovna[3]. Tato malá, ale velmi užitečná knihovna určená pro volné použití, má závislost na knihovně Boost[4] v minimální verzi 1.23.

[3]https://github.com/jimporter/bencode.hpp
[4]https://www.boost.org/

7.1.5 Packet

Tento modul obstarává zpracování vstupního paketu do struktury, která je poté zařazena do fronty k pozdější analýze. Výpis 7.2 obsahuje deklaraci struktury *packet*. Jak lze očekávat, obsahuje IP adresu a port zdroje i cíle. Výčtovými typy je definován transportní protokol (UDP, TCP) a směr provozu (IN, OUT).

```
struct packet{
    struct in_addr src_ip, dst_ip;
    u_short src_port = 0, dst_port = 0;
    L4 l4;
    IO io;
    unsigned int payload_size;
    u_char *payload;
    bool p2p_protocol_packet = false;
    std::chrono::steady_clock::time_point time;
    bool fragmented = false;
};
```

Výpis 7.2: Deklarace *packet* struktury.

Obsah paketu je alokován knihovní funkcí *malloc()* a do *packet* struktury je umístěn jen ukazatel a velikost. Přítomny jsou dvě *boolean* proměnné pro označení paketu zúčastněného v P2P přenosech a případná fragmetace paketu. Pokud je v rámci detekce nutné řešit stavovost paketů např. pro potvrzení BT přenosu z platné odpovědi, je přítomna časová proměnná na určení zpoždění odpovědi nebo zrušení čekání.

7.1.6 Callbacks

Funkce modulu pro analýzu byla formálně popsána v sekci 6.1 změnami stavu paketu v rámci navrženého stavového diagramu. Implementace tohoto diagramu se nachází ve funkci *processing_func()*, ze které jsou pozitivní detekce předávány do fronty DB požadavků spravované modulem *Connector*. Vlákno analyzátoru musí co nejrychleji odevzdat informace o incidentu k dalšímu zpracování a věnovat se dalším paketům. Manipulace s frontou DB požadavků je patrná z pseudokódu na Výpisu 7.3. Díky konstrukci *std::bind()* je možné svázat ukazatel na funkci *incident()* a její parametry hodnotou. Databázové vlákno pak jen vybere funkci z fronty a provede ji voláním *f()*.

Pro každý druh incidentu je tak definována jiná funkce s příslušnými parametry a z těchto entit se vytváří fronta DB požadavků.

```
void incident(...){
    // prace s incidentem (DB, LOG)
}

std::vector<std::function<void()>> db_fronta;
auto binding = std::bind(&incident, ...);
db_fronta.insert(db_fronta.begin(), binding)

std::function<void ()> f = db_fronta.back();
db_fronta.pop_back();
f();
```

Výpis 7.3: Ukázka manipulace s frontou DB požadavků

7.1.7 Connector

Hlavním úkolem modulu *Connector* je spojení analýzy paketů a vykonávání DB operací. Součástí je také práce s mezipamětí pro výpočet objemu přenesených dat na základě sledování aktivních flow navázaných mezi uživateli sítě a vzdálenými peery. Úloha paralelního zpracovávání DB požadavků byla rozdělena mezi dvě nezávislá vlákna:

`db_query_thread` - pouze zpracovává frontu DB požadavků jak bylo vysvětleno v předchozí podsekci.

`db_periodic_thread` - pravidelně přidává do fronty DB požadavků další úlohy: kontrola expirace incidentů, kontrola aktivity jednotlivých flow a aktualizace flow cache hash tabulky.

Aby bylo možné sledovat v reálném čase aktivní flow, je v tomto modulu implementována hash tabulka s 16 bitovým klíčem. Velikost klíče určuje i velikost tabulky. To znamená 65 535 položek. Pro rozptýlení flow dat byla implementována velmi jednoduchá a rychlá hash funkce viz Výpis 7.4. Jedná se o bitový součet IP adres, bitový součin portů a přičtení hodnoty výčtového typu transportního protokolu `0` nebo `1`. Díky operaci modulo se pak výsledek

může umístit do 16 bitové proměnné. Rozptyl takové hash funkce se ukázal jako dostatečný pro řádově stovky současně sledovaných flow. Pokud dojde ke kolizi hash hodnot, je další flow zařazeno do seznamu tak, jak lze očekávat od abstraktního datového typu jako je hash tabulka.

```
u_int16_t hash = ((user_ip | peer_ip) +
                  (user_port & peer_port)
                   + L4) % UINT16_MAX;
```

Výpis 7.4: Hash funkce pro flow položky.

Programovací jazyk C++ 17 disponuje asociativním datovým kontejnerem *std::unordered_multimap* podporujícím vložení položek se stejným klíčem v případě kolize. Flow položky obsahují svůj „komparátor“ používaný v případě, že vyhledávání dosáhne druhé úrovně (procházení seznamu). Při shodě analyzovaného paketu s některým z flow záznamů se přičte množství přenášených dat do kategorie *IN* nebo *OUT*. Vlákno *db_periodic_thread* pravidelně propaguje nashromážděné množství flow dat do databáze a zároveň provede obnovu flow cache položek s ohledem na čas poslední aktivity.

Již předem je zřejmé, že nebude možné zachytit 100 % přenášených torrent dat. Jednou z nedokonalostí může být nedostatečný počet zjištěných peerů, se kterými uživatel komunikuje v rámci stahování jednoho torrentu. Další může nastat díky relativně velkým prodlevám v řádu desítek sekund, než se zpětně z databáze načtou aktivní flow ke sledování. Strategií *best-effort* byla ovšem implementována tato funkcionalita, která není dostupná v žádném z testovaných systémů pro analýzu BitTorrent sítě. Správce sítě tak bude disponovat informacemi o **prokazatelném** objemu stahovaných nebo sdílených torrent dat.

7.1.8 Sql

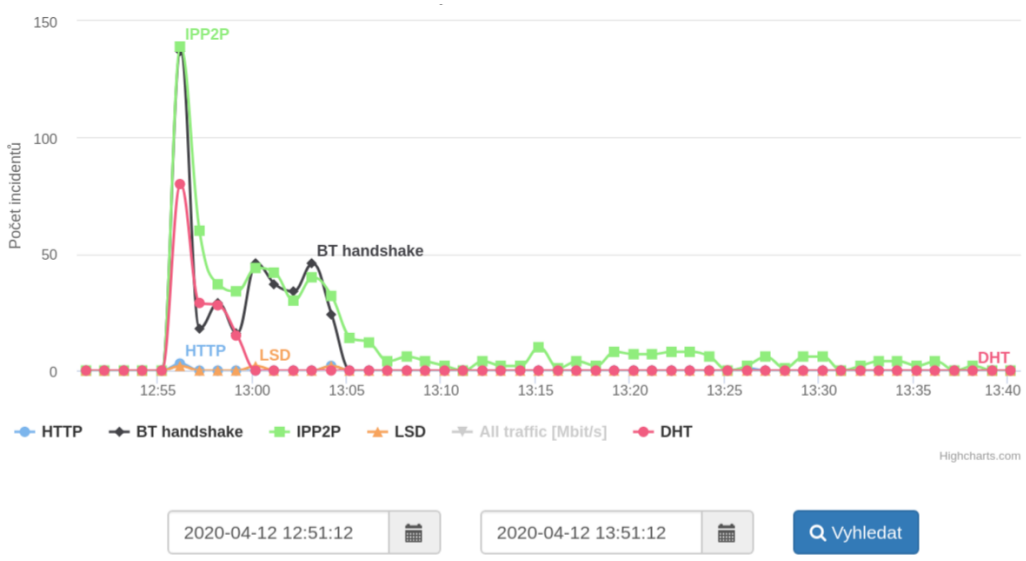
Poslední modul se skládá z 32 funkcí, které dle zadaných parametrů vytvářejí a pracují s SQL dotazy. Aby z jazyka C++ bylo možné komunikovat s MariaDB je do projektu přidán tzv. MariaDB Connector, který lze v distribuci Debian 10 nalézt v oficiálním repozitáři jako *mariadb-connector-c*.

7.2 Webové rozhraní

Při implementaci webové aplikace v jazyce PHP byl zvolen minimalistický postup bez použití již existujících PHP frameworků nebo šablonovacích systémů. Na straně uživatele bylo však pro další interakci nutné použít programovací jazyk JavaScript s knihovnou jQuery[5] pod licencí MIT (libovolné použití a úpravy pod podmínkou uvedení originálního autorství). V následujících podsekcích bude představeno rozložení webového rozhraní společně s jeho ovládacími prvky a poté budou popsány implementační detaily.

7.2.1 Graf historie BitTorrent aktivity

Na Obrázku 7.1 je vytvořený graf za pomoci Highcharts[6] knihovny pod nekomerční studentskou licencí. Uživatel si vybere ze dvou vzájemně propojených prvků pro výběr časového rozmezí, ze kterého je vytvořen graf BitTorrent aktivity. K vytváření jsou použity statistická data z databázové tabulky `stats`, které graf přijímá ve formátu JSON[7].



Obrázek 7.1

Graf je interaktivní a podporuje změnu škály výskytu různých incidentů na základě skrytých nebo zobrazených položek v legendě grafu. Kromě již

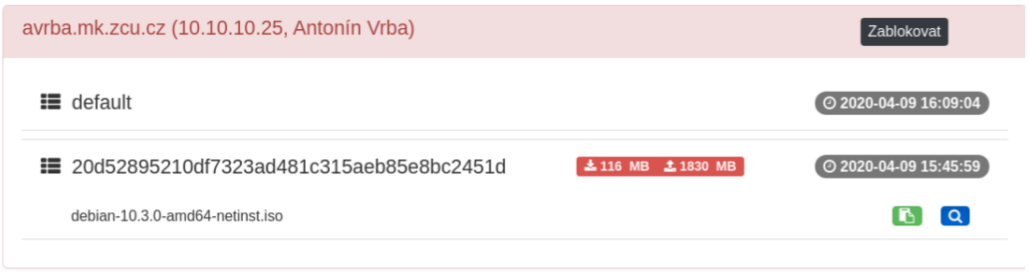
[5]https://jquery.com/
[6]https://www.highcharts.com/
[7]https://www.json.org/json-cz.html

známých položek je dostupné i zobrazení množství veškerého provozu procházejícího analyzátorem v Mbit/s. Obrázek 7.1 odráží orientační výskyt BT protokolů zachycených analýzou při startu BT klienta s aktivní torrent úlohou v čase 12:55.

7.2.2 Zobrazení incidentů

Další komponenta webového rozhraní je výpis incidentů časově filtrovaných stejnými prvky jako je ovládán graf. Na Obrázku 7.2 je příklad zaznamenaných incidentů vztažených ke konkrétnímu uživateli registrovaného v lokální síti. Mezi základní ovládací prvky patří možnost volby „Zablokovat“, to přesměruje správce sítě do systému KNet, kde je již vyřešený systém vytváření blokací uživatelů.

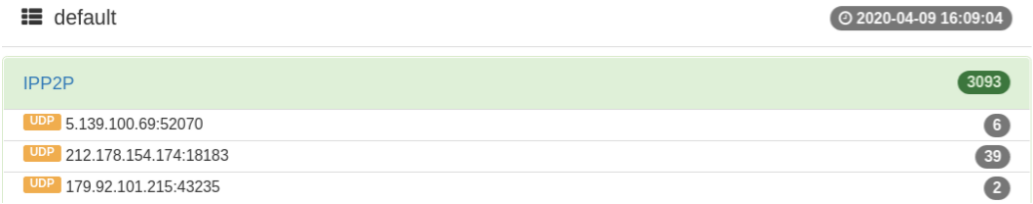
Rozlišuje se mezi tzv. „default“ výchozím incidentem a incidentem popsaným infohash hodnotou. Do výchozího incidentu patří nepřiřazené IPP2P BitTorrent signatury. Incident s infohash hodnotou lze vyhledávací ikonou otevřít v novém okně a zkontrolovat tak manuálně, jestli se infohash nevyskytuje na některém z veřejně dostupných seznamů torrentů. Druhá ikona slouží k nastavení jména torrentu. Pokud uživatel použije DHT zprávu *announce_peer*, analyzátor jméno torrentu automaticky doplní do databáze. Incident s infohash hodnotou disponuje také údaji o potvrzených obousměrně přenesených torrent datech.



Obrázek 7.2: Příklad incidentu uživatele lokální sítě.

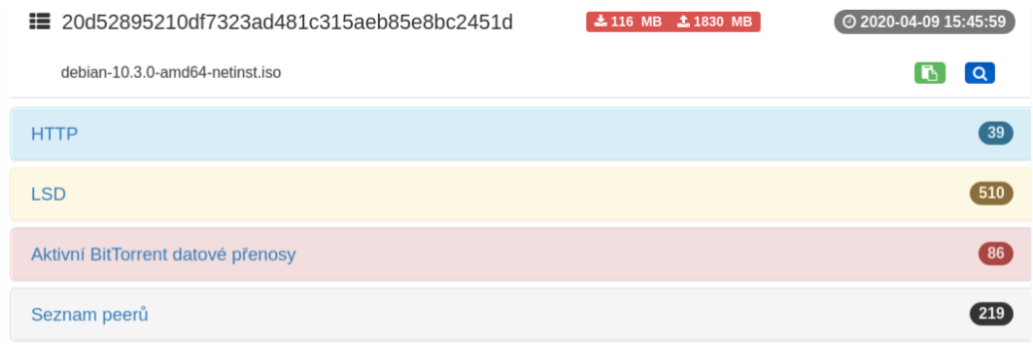
Na Obrázku 7.3 je část výpisu detailu výchozího incidentu. Zobrazeny jsou cílové IP adresy, porty, použité transportní protokoly a příslušný počet zachycených nezařazených signatur. Výpis s podobnou informační hodnotou nabízí aktuálně nasazená P2P sonda.

Obrázek 7.4 nabízí pohled na druh incidentu identifikovaného podle infohash hodnoty. Obsahuje čtyři dále rozšiřitelné kategorie: HTTP, LSD, kompletním



Obrázek 7.3: Detail výchozího incidentu.

seznamem peerů a nejdůležitější kategorii „Aktivní BitTorrent datové přenosy“, kde se nachází informace o přenesených datech s konkrétními peery. Všechny čtyři kategorie jsou v „nerozbaleném“ stavu doplněny v pravé části o počet položek v rámci této kategorie. Po „rozbalení“ kategorie HTTP jsou zobrazeny adresy vzdálených tracker serverů se kterými si uživatel vyměnil zprávy typu *announce* a *scrape*. LSD obsahuje obsah zpráv vysílaných na multicast adresu. Kategorie Seznam peerů skrývá kompletní výpis všech peerů se kterými uživatel v rámci tohoto incidentu komunikoval jak protokolem BitTorrent tak i DHT.



Obrázek 7.4: Kategorie incidentu popsaného infohash hodnotou.

Na posledním Obrázku 7.5 ilustrujícím vzhled webového rozhraní je detail kategorie aktivních BT přenosů. Toto zobrazení také umožňuje reverzní překlad IP adresy na doménové jméno.

Obrázek 7.5: Kategorie incidentu popsaného infohash hodnotou.

7.2.3 Struktura webové aplikace

V GitLab repozitáři obsahujícím celý projekt se nachází podsložka *web* s následujícími *.php* soubory:

`config` - konfigurační soubor.

`index` - HTML struktura, JavaScript kód pro graf a zpětné volání.

`content` - generování výpisu incidentů do HTML dle časového rozmezí.

`api` - implementace API do systému KNet.

`input` - vstupní bod pro zpětné volání do aplikace.

`sql` - práce s databází.

Konfigurace

Konfigurační soubor obsahuje položky jako na Výpisu 7.5. První čtyři parametry specifikují připojení webové aplikace do databáze a jsou následovány parametrem adresy `web_hostname` pod kterou webová aplikace bude vystupovat. Pro napojení na systém KNet je nejdříve v tomto systému nutné vygenerovat `id` a `secret` a s těmito hodnotami pak žádat o dočasný *token* na adrese `token_url`. Adresa `endpoint_url` pak slouží k API volání, jehož rozhraní je specifikované Knet dokumentací v práci [23].

```
$host='localhost';
$db = 'p2p_analyzer';
$username = 'user';
$password = 'pass';
$web_hostname = "http://localhost:8888";
$id= "...";
$secret = "...";
$endpoint_url = "https://knet.zero.zcu.cz/api/v1";
$token_url = "https://knet.zero.zcu.cz/api/oauth2/token";
```

Výpis 7.5: Položky konfigurace webové aplikace.

Asynchronní komunikace

Součástí knihovny jQuery je i asynchronní způsob práce s HTTP dotazy nazývaný AJAX (Asynchronous JavaScript and XML). Využití technologie

AJAX přináší výhodu nemuset aktualizovat stránku při získávání nových dat z webového serveru. Takto je např. vyřešeno získávání dat pro tvorbu grafu nebo zadávání názvu pro incident identifikovaný infohash hodnotou. Výpis 7.6 ukazuje jak je s použitím jQuery knihovny jednoduché vytvořit asynchronní HTTP POST požadavek s parametry a následným zpracováním dat ze serveru.

```
$.ajax({
type: "POST",
url: "http://localhost:8888/input.php",
data: {param: "param"},
    success: function(data){
    // zpracovani dat
    },
    error: function(){
    // chyba
    }
});
```

Výpis 7.6: HTTP POST požadavek zprostředkovaný přes AJAX.

Výchozí chování aplikace

Při zobrazení obsahu *index.php* je správci sítě představen graf za poslední hodinu s přednastavenými kalendáři. Až stiskunutí tlačítka „Vyhledat“ zobrazí případné incidenty z vybraného časového období. Změna časového rozmezí ovlivní i vykreslení grafu. Správce sítě pozná uživatele zablokovaného v systému KNet podle červeného zbarvení záhlaví, které je jinak modré.

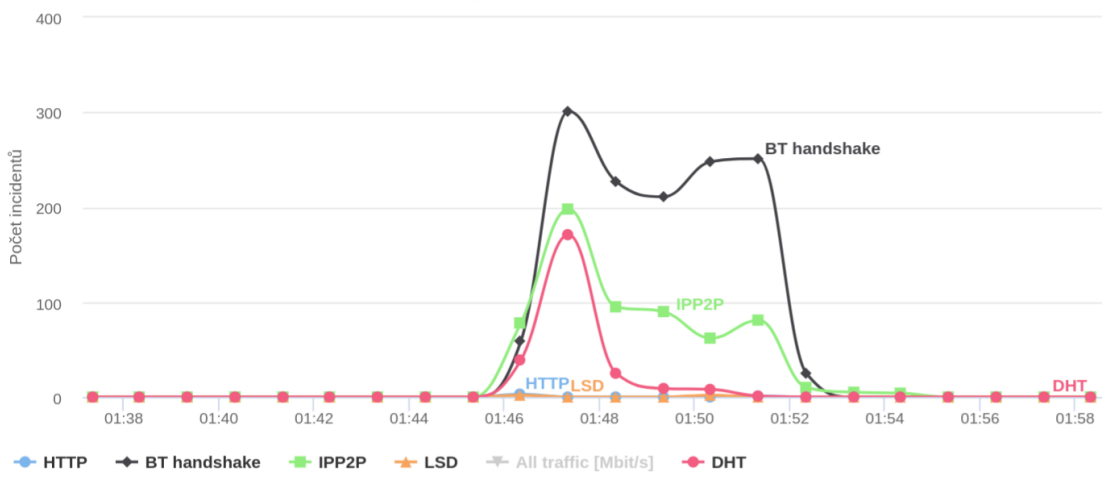
Jelikož je tato aplikace minimalistická není implementována žádná správa uživatelů a stejně jako P2P sonda je přístup řešen na straně webového serveru - možná autentizace pomocí hesla, nebo přístup jen z určitých počítačů na síti.

8 Testování

V Kapitole 5 byl definován scénář pro testování analyzátorů a v této kapitole bude podle stejného scénáře otestován i nově implementovaný analyzátor. Výsledky testování budou porovnány v rámci již dříve představených analyzátorů.

8.1 Pohled správce sítě

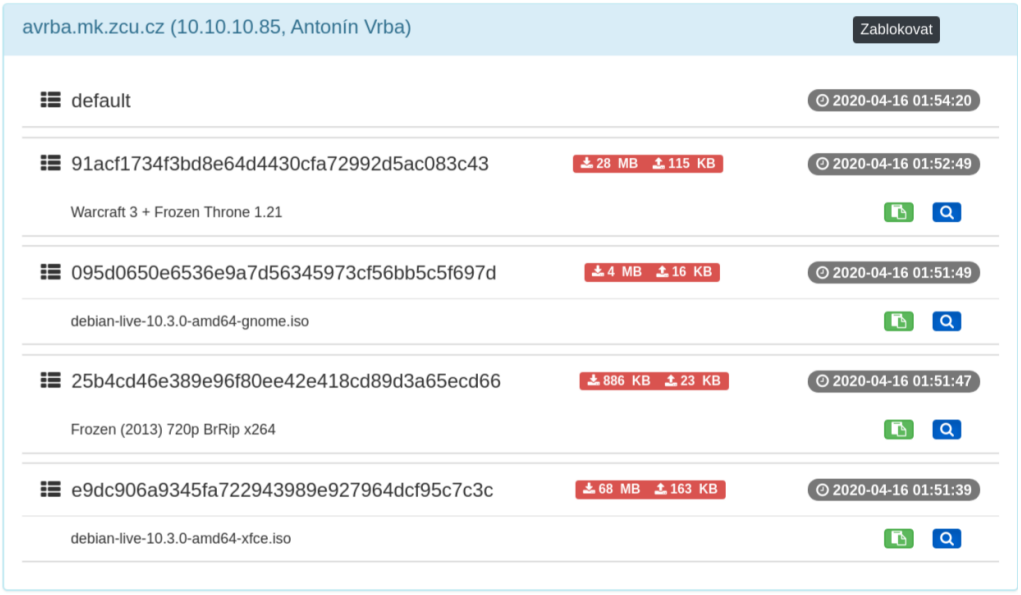
Scénář BitTorrent stahování po dobu 5 minut se okamžitě projevil na podobě grafu reprezentující četnost výskytů zpráv různých BT protokolů. Na Obrázku 8.1 je patrný nárůst aktivity v řádu stovek zpráv za minutu (čas 1:46). Správce sítě si může určit libovolný časový úsek a okamžitě vidět aktivitu BitTorrent protokolů v závislosti na nastaveném časovém rozmezí.



Obrázek 8.1: Grafická reprezentace BitTorrent provozu v čase.

Grafová reprezentace může pomoci zpřehlednit situaci před zobrazením konkrétních incidentů a na Obrázku 8.2 už se nachází výpis všech incidentů, které analyzátor zachytil. Scénář obsahoval 4 torrenty a všechny 4 byly podle infohash hodnoty identifikovány. Následné signatury se klasifikovaly podle těchto nově založených incidentů a nezařazené se přesunuly do kategorie IPP2P v incidentu „default“.

Tabulka 8.1 koresponduje s výsledky zobrazenými na webovém rozhraní



Obrázek 8.2: Zobrazení incidentů z webového rozhraní po provedení testů.

z Obrázku 8.2, které lze dále individuálně specifikovat a rozšiřovat v rámci kategorií jednotlivých incidentů. Analyzátor automaticky zjistil název dvou torrentů stahovaných z aplikace μTorrent, zbylé dva bylo možné vyhledat a nastavit manuálně. Při bližším zjištění se ukázalo, že Transmission explicitně neuvádí název torrentu v DHT zprávách a analyzátor ho tak nezachytil. Podle statistik z flow analýzy bylo za 5 minut zjištěno celkem 101 MB stažených a 317 KB odeslaných torrent dat. Přesnost této flow analýzy se liší dle použité klientské aplikace a stahovného torrentu, ale dle i jiných testování se pohybuje od 40 % do 80 % potvrzených dat. Z Tabulky 8.1 lze také potvrdit očekávané chování a to, že torrenty inicializované přes *.torrent* soubory pravidelně komunikují přes HTTP tracker protokol narozdíl od torrentů inicializovaných z tzv. magnet URI. Protokol LSD se podařil zachytit u aplikace μTorrent, ale ne u aplikace Transmission i přes správné nastavení aplikace. Pro ověření funkčnosti a vyloučení specifického nastavení analyzátoru pouze na μTorrent byla otestována i aplikace qBittorrent a v jejím případě byly správně rozpoznané jména torrentů i LSD signatury.

V průběhu testování analyzátor zachytil 595 unikátních IP adres peerů a 816 inicializovaných relací mezi lokálním uživatelem a adresami těchto peerů. Tabulka 8.1 reflektuje tyto hodnoty a pro položku „Aktiních peerů“ byl nastaven práh alespoň 1 KB přenesených torrent dat. Ve stejném čase čtyř zmíněných incidentů došlo k zachycení 623 neklasifikovaných IPP2P zpráv. Úspěšně byla v síti detekována přítomnost „prázdných“ klientských BT apli-

BitTorrent klient	μTorrent 3.1.3		Transmission 2.94	
Způsob inicializace	*.torrent*	magnet	*.torrent*	magnet
Infohash hodnota	095d..697d	25b4..cd66	e9dc..7c3c	91ac..3c43
Zjištený název	✓	✓	✗	✗
Stažená data	4 MB	886 KB	68 MB	28 MB
Odeslaná data	16 KB	23 KB	163 KB	115 KB
HTTP	2		2	
LSD	2	2		
Počet peerů	86	263	73	197
Aktivních peerů	11	20	52	70
IPP2P	623			

Tabulka 8.1: Shrnutí výsledků k scénářovému testu.

kací bez probíhajícího torrent stahování. Tato informace je ovšem ve výchozím nastavení skrytá a nevytváří tak zbytečné viditelné incidenty.

8.2 Stabilita a výkon analyzátoru

Jelikož je analyzátor navržen pro nepřetržitý provoz, byl z tohoto hlediska dlouhodobě testován na serveru umístěném v prostředí jednoho z páteřních switchů kolejní sítě. Server disponoval následujícími parametry:

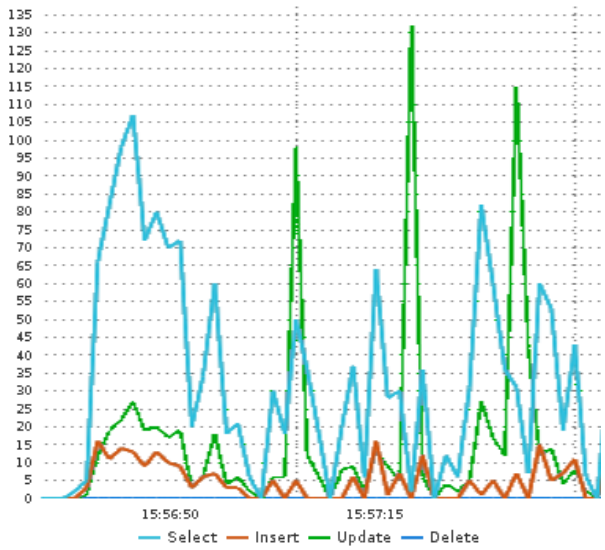
CPU - Intel Core i5-3570K 3.40 GHz,

RAM - 16 GB,

NIC (mirror) - QLogic 10 Gbit/s Ethernet.

Test probíhal 60 dní vůči provozu ze zrcadleného 10 Gbit/s rozhraní. Software analyzátoru po celou tuto dobu vykazoval stabilitu a stále správně reagoval na nový BitTorrent provoz. Bylo zachyceno 150 tisíc různých BT signatur a z nich byly vytvořeny desítky incidentů. Ze statistických dat byl zjištěn největší datový tok procházející analyzátorem 900 Mbit/s, zpracováván současně pěti vlákny s průměrným vytížením všech jader procesoru na 15 %. Využití operační paměti dosahovalo do 5 GB, ale vzhledem k přidělovaným paměťovým zdrojům tato hodnota nebude škálovat se zvyšujícími se nároky lineárně jako škálují nároky na výpočetní výkon CPU.

Na Obrázku 8.3 se nachází graf vygenerovaný volně dostupným softwarem DBeaver[1] reprezentující četnost databázových operací při běhu scénářového testu. Nejnáročnější operace je zde ihned patrná a je to chování flow cache mechanismu, který každých 10 sekund synchronizuje monitorovaná data zpět do databáze (zelené špičky). S rostoucí flow cache tabulkou se databáze během dlouhodobého testování dostala do jednorázových špiček až 1000 dotazů za sekundu a toto databáze bez problému zvládla. Nebylo tak nutné strategii komunikace mezi analyzátorem a databází dále optimalizovat.



Obrázek 8.3: Graf četnosti databázových operací.

8.3 Vyhodnocení a porovnání

Test výkonu prokázal schopnost analyzátoru zpracovávat v realném čase teoreticky až 5 Gbit/s síťového provozu na počítačové platformě staré 8 let (výkon dnešního průměrného notebooku). Nově implementovaný analyzátor splňuje jako jediný všechny funkce z předchozí sekce 5.7. Správně detekuje BitTorrent protokoly a výsledky agreguje do incidentů orientovaných na stahovaný obsah podle infohash hodnoty. Výsledky jsou dostupné z webového rozhraní s možností zpětného prohlížení událostí včetně získávání dodatečných informací o uživateli pomocí API ze systému KNet. Unikátní je pak funkcionalita umožňující zjistit prokazatelné množství přenesených torrent dat. Žádné jiné z testovaných řešení neumožňuje takový informační komfort, při řešení incidentů spojených s BitTorrent přenosy.

[1] https://dbeaver.io/

9 Možnosti rozšíření

Po dokončení celé aplikace (analyzátor, databáze a webové rozhraní) se ihned nabídlo několik možností jak aplikaci dále zdokonalit. Hlavním směrem rozšíření by měl být obecnější přístup k samotné analýze. Nyní je analyzátor orientován jen pro detekci sítě BitTorrent, ovšem kolejní síť je také místem s restrikcí pro výskyt NAT zařízení a VPN tunelování. Bylo by ideální mít pouze jeden analyzátor, který bude detekovat BitTorrent a také další zmíněné restrikce. Aplikace byla vyvíjena s předpokládaným nasazením v místech, kde se nachází kolejní páteřní switche. Takto bude najednou spuštěno více instancí aplikace, včetně databáze i webového rozhraní. V budoucnu by však analyzátor mohl být umístěný přímo v místě zařízení provádějící NAT pro všechny koleje. Právě kvůli tomuto předpokládanému nasazení byl už z počátku kladen velký důraz na propustnost a efektivitu analyzátoru.

Pro zvýšení komfortu pro práci s incidenty, je možné rozšířit propojení se systémem KNet, např. automatické blokace po překročení určitého množství BT signatur. V rámci této úpravy bude nutné upravit i položky pro vytváření incidentů v systému KNet. Webové rozhraní lze vylepšit volitelným pohledem pro sjednocené incidenty různých uživatelů podle svého obsahu, např. pokud se více uživatelům lokální sítě spustí bez jejich vědomí BT aktualizace počítačové hry, tak je vhodné řešit takovou situaci centralizovaně. Rozšiřování funkcí webového rozhraní nad rámec tohoto řešení bude vhodné zpřehlednit použitím šablonovacího systému pro zvýšení čitelnosti PHP kódu. Během implementace byl také diskutován mechanismus hodnocení závažnosti incidentu podle infohash hodnoty, ale pokud již existuje možnost zadání jména torrentu, byla by tato informace redundantní. Už nyní webové rozhraní filtruje např. signatury spojené s „na prázdno“ spuštěnými BT klienty bez aktivního torrent stahování. Pro zvýšení přehlednosti lze přidat další mechanismy pasivního filtrování incidentů dle různých kritérií např. počet připojených peerů nebo množství přenesených dat. Také by bylo užitečné zprostředkovat správci sítě manuální filtrování při vyhledávání incidentů.

Díky zřízenému GitLab repozitáři bude umožněno v průběhu nasazování aplikace analyzátoru sledovat požadavky pro další případné úpravy.

10 Závěr

V prvních čtyřech kapitolách této práce bylo teoreticky popsáno, co se nachází pod pojmem Peer-to-Peer. Po vysvětlení vzniku a využití byly představeny P2P topologie a s tím spojené principy P2P sítí. K dalšímu pochopení fungování P2P sítí byly porovnány nejpopulárnější z nich s ohledem na jejich vlastnosti. Dále jsou uvedeny informace dokazující absolutní převahu sítě BitTorrent v podílu všech P2P přenosů. Tato síť pak byla detailně popsána s ohledem na její nejvýznamnější rozšíření. Při popisu byl kladen důraz na strukturu zpráv a chování BT protokolů využitelné v analýze síťového provozu. Po teoretické části bylo provedeno testování vybraných analyzátorů podle vytvořeného testovacího scénáře. Dle výsledků nesplňoval žádný z testovaných analyzátorů všechny požadavky na pokročilejší P2P analýzu.

Díky získaným znalostem z teoretické přípravy a dokončenému testování existujících řešení byl vytvořen návrh a podle něj byla vyvinuta zcela nová aplikace analyzátoru. V návrhu byla vysvětlena strategie pro zachycení a manipulaci s BT signaturami a následně byl implementován síťový analyzátor v jazyce C++, jehož moduly byly v práci zdokumentovány. Pro uložení výsledků byla navržena struktura SQL databáze, blíže specifikovaná vloženými relačními diagramy. K zajištění zobrazení nových a historických incidentů z databáze bylo navrženo webového rozhraní implementované v jazyce PHP. Webové rozhraní analyzátoru integruje API kolejního systému KNet a tím výrazně zvyšuje komfort při řešení incidentů pro síťové správce. Testováním nově implementovaného analyzátoru byla prokázána jeho dlouhodobá stabilita a především schopnost v reálném čase dosáhnout dostatečné propustnosti k monitorování BitTorrent komunikace v prostředí kolejní sítě Západočeské univerzity.

Vysoká úroveň zpracování zachycených informací, ve smyslu agregace signatur dle individuálního torrent stahování, dělá z nově představené aplikace nejpokročilejší analyzátor pro úlohu monitorování BitTorrent přenosů ze všech testovaných systémů. Vůbec poprvé je díky této práci možné sledovat prokazatelné množství přenesených torrent dat v kontextu uživatelem sdíleného nebo stahovaného obsahu. Velké množství získaných dat při detekci sítě BitTorrent bylo dosaženo hlavně díky zaměření na individuální BT protokoly s využitím obsahu jejich vlastních zpráv pro získání klíčových informací. Nasazení nově představené aplikace výrazně usnadní řešení

incidentů pro správce kolejní sítě a může plně nahradit v současné době používané řešení v podobě analyzátoru P2P sonda. Při současném trendu narůstajícího množství BitTorrent přenosů může tato práce sloužit i jako zdroj ucelených informací k uvedení do problematiky světa P2P sítí.

Literatura

[1] *Raids close file-sharing server* [online]. BBC NEWS, 2006. [cit. 25.12.2019]. Dostupné z: http://news.bbc.co.uk/2/hi/technology/4743052.stm.

[2] *BEP 11, Peer Exchange (PEX)* [online]. [cit. 22.2.2020]. Dostupné z: https://www.bittorrent.org/beps/bep_0011.html.

[3] *BEP 14, Local Service Discovery* [online]. [cit. 16.1.2020]. Dostupné z: https://www.bittorrent.org/beps/bep_0014.html.

[4] *BEP 29, uTorrent transport protocol* [online]. Ludvig Strigeus, Greg Hazel, Stanislav Shalunov, Arvid Norberg, Bram Cohen. [cit. 14.1.2020]. Dostupné z: https://www.bittorrent.org/beps/bep_0029.html.

[5] *BEP 3, BEPThe BitTorrent Protocol Specification* [online]. Bram Cohen. [cit. 5.1.2020]. Dostupné z: https://www.bittorrent.org/beps/bep_0003.html.

[6] *BEP 5, DHT Protocol* [online]. Andrew Loewenstern, Arvid Norberg. [cit. 16.1.2020]. Dostupné z: https://www.bittorrent.org/beps/bep_0005.html.

[7] *BEP 9, Extension for Peers to Send Metadata Files* [online]. Greg Hazel, Arvid Norberg. [cit. 5.1.2020]. Dostupné z: https://www.bittorrent.org/beps/bep_0009.html.

[8] *Peer-to-Peer Communication Across Network Address Translators* [online]. Bryan Ford Massachusetts Institute of Technology, Pyda Srisuresh Caymas Systems, Inc., Dan Kegel. [cit. 30.12.2019]. Dostupné z: Peer-to-PeerCommunicationAcrossNetworkAddressTranslators.

[9] *The Gnutella Protocol* [online]. Gayatri, Albert-Ludwigs-Universität Freiburg, 2007. [cit. 30.12.2019]. Dostupné z: http://archive.cone.informatik.uni-freiburg.de/teaching/seminar/p2p-networks-w06/submissions/mzpro2.pdf.

[10] *Dokumentace ntop* [online]. [cit. 4.3.2020]. Dostupné z: https://www.ntop.org/support/documentation/documentation/.

[11] The Global Internet Phenomena Report. 2019. Dostupné z: https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/Internet%20Phenomena/Internet%20Phenomena%20Report%20Q32019%2020190910.pdf.

[12] *Dokumentace Suricata* [online]. [cit. 7.3.2020]. Dostupné z: https://suricata.readthedocs.io/en/suricata-4.1.2.

[13] *Computer Networks: A Systems Approach* [online]. Larry Peterson and Bruce Davie. [cit. 17.1.2020]. Dostupné z: https://book.systemsapproach.org.

[14] *Wireshark Dokumentace* [online]. Gerald Combs, Guy Harris, Gilbert Ramirez. [cit. 26.2.2020]. Dostupné z: https://www.wireshark.org/docs/wsug_html_chunked.

[15] Bezděk, D. Monitorování Peerů Sdílející Torrenty. Master's thesis, Vysoké Učení Technické v Brně, Fakulta Informačních Technologií, Ústav Informačních Systémů, 2018. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=180931.

[16] Cohen, B. Incetives Build Robustness in BitTorrent. 2003. Dostupné z: https://www.cs.swarthmore.edu/~newhall/readings/bittorrentecon.pdf.

[17] Di Wu, X. H. C. Z. K. W. R. P. D. Understanding Peer Exchange in BitTorrent Systems. 2019. Dostupné z: https://www.researchgate.net/publication/224173879_Understanding_Peer_Exchange_in_BitTorrent_Systems/link/0a85e53513d4229e45000000/download.

[18] Green, M. Napster Opens Pandora's Box: Examining How File-Sharing Services Threaten the Enforcement of Copyright on the Internet. *OHIO STATE LAW JOURNAL*. 2002. Dostupné z: https://kb.osu.edu/bitstream/handle/1811/70499/OSLJ_V63N2_0799.pdf.

[19] Jahn Arne Johnsen, S. S. B. L. E. K. Peer-to-peer networking with BitTorrent. 2005. Dostupné z: http://web.cs.ucla.edu/classes/cs217/05BitTorrent.pdf.

[20] MEHDI, M. Interception of P2P Traffic in a Campus Network. *Romanian Journal of Information Technology and Automatic Control*. 2019. Dostupné z: https://www.researchgate.net/publication/334213518_Interception_of_P2P_Traffic_in_a_Campus_Network/link/5d1f6001a6fdcc2462c3b0aa/download.

[21] Molin, K. Measurement and Analysis of the Direct Connect Peer-to-Peer File Sharing Network. Master's thesis, University of Gothenburg, Department of Computer Science and Engineering, 2009. Dostupné z: https://gupea.ub.gu.se/bitstream/2077/22088/1/gupea_2077_22088_1.pdf.

[22] Oliver Heckmann, A. M. R. S. A. B. The eDonkey File-Sharing Network. 2004. Dostupné z: https://pdfs.semanticscholar.org/242a/8ad865c8b2d40caafecd4a88a4af99b75624.pdf.

[23] Smitka, J. Systém pro řízení přístupu do kolejní sítě ZČU. Master's thesis, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky, 2016. Dostupné z: https://dspace5.zcu.cz/handle/11025/23678.

Přílohy

A Obsah přiloženého DVD

- Dokumentace

  src - zdrojové soubory LaTeXa obrázky

  out - PDF s textem diplomové práce

- Software verze 0.7 (GitLab repozitář)

  analyzer - zdrojové soubory C++, CMake soubor pro sestavení a *pcapng* soubory pro testování

  setup - konfigurace webového serveru a systemd konfigurace

  sql - definice databáze a instalační skripty.

  web - PHP soubory webové aplikace

  README.md - instalační pokyny

- Kompletní CLion projekt

B Nasazení analyzátoru na server (Debian 10)

V této příloze je do tří sekcí shrnuto nezbytnými kroky nasazení nově implementovaného analyzátoru.

B.1 Databáze

```
# instalace potrebnych balicku
apt install mariadb-server git

# klonovani projektu
git clone git@ipmil.civ.zcu.cz:knet/p2p-analyzer.git
cd p2p-analyzer

# vytvoreni uzivatele a pristupu k DB p2p_analyzer
mysql> CREATE USER 'user'@'localhost'
       IDENTIFIED BY 'password';
mysql> GRANT ALL PRIVILEGES ON p2p_analyzer.*
       TO 'user'@'localhost';
mysql> FLUSH PRIVILEGES;

# import DB pomoci predprivaneho skriptu clear.sh
bash p2p-analyzer/sql/clear.sh
```

B.2 Analyzátor

```
# instalace potrebnych balicku
apt install build-essential cmake libboost-dev libmariadb-dev

# sestaveni a kompilace pomoci nastroje CMake a make
cd p2p-analyzer/analyzer/src
cmake CMakeLists.txt
```

```
make

# rozmisteni souboru analyzatoru
mkdir /opt/p2p-analyzer
cp p2p-analyzer /opt/p2p-analyzer/
cp ../config/config /opt/p2p-analyzer/

# rucni konfigurace analyzatoru
vim /opt/p2p-analyzer/config

# nastaveni systemd sluzby automaticky po startu
cd ../..
cp setup/p2p-analyzer.service /etc/systemd/system
chmod 664 /etc/systemd/system/p2p-analyzer.service
systemctl daemon-reload
systemctl enable p2p-analyzer.service
systemctl start p2p-analyzer.service
```

B.3 Webové rozhraní

```
# instalace potrebnych balicku
apt install apache2 php7.0-mysql

# kopirovani PHP souboru webove aplikace
cp /web/ /var/www/

# nastaveni weboveho serveru
cp p2p-analyzer/setup/p2p-analyzer.conf
   /etc/apache2/sites-available/
a2ensite p2p-analyzer

# vytvoreni basic auth pristupu
htpasswd -c /etc/apache2/.htpasswd user

# nastaveni webove aplikace
vim /var/www/web/config.php
```