
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**


# MONITOROVÁNÍ A DETEKCE PROVOZU SÍTÍ BITTORRENT

**MONITORING AND DETECTION OF BITTORRENT TRAFFIC**

**PROJEKT DO PŘEDMĚTU PDS**
**PDS COURSE PROJECT**


**AUTOR** **Bc. ONDŘEJ ONDRYÁŠ**
**AUTHOR**


**BRNO 2023**

# Obsah

# Kapitola 1

# Architektura sítí BitTorrent

Termín BitTorrent označuje protokol, respektive skupinu protokolů a nástrojů, které umožňují decentralizovanou kolektivní distribuci souborů s využitím principu peer-to-peer (P2P).

Ve své původní verzi je BitTorrent hybridních P2P sítí. Pro realizaci sítě je kromě uzlů nutný také centrální bod – *tracker*, kterému se uzly ohlašují a od kterého dostávají informace o už připojených uzlech v síti [3]. Trackery jsou však skutečně pouhými „databázemi uzlů“ a nepodílí se na samotném řízení sítě – neměly by tedy např. rozhodovat, které stávající uzly novému uzlu nahlásí, tento výběr by měl proběhnout zcela náhodně [2].

Problémem takové architektury je, že centrální bod lze odstavit (např. s využitím útoku typu DoS nebo zásahem státních orgánů). Vzniklo tedy několik rozšíření, která umožnila existenci *trackerless torrentů*: informace o uzlech zapojených do sdílení jistého souboru se zde ukládají v **distribuovaných** hashových tabulkách *DHT*. Ty tvoří samostatnou P2P síť, která poskytuje ukládání kolekcí „klíč-hodnota“ a efektivní vyhledávání hodnot pouze s využitím lokálních informací.

## 1.1 BitTorrent s trackerem

Základní verze protokolu rozlišuje klienty v roli uzlů (*peers*), tracker a soubory *metainfo* (také označovány jako soubory .torrent). Metainfo soubor popisuje konkrétní *torrent* – soubor nebo strukturu souborů sdílenou mezi uzly. Obsahuje URL trackeru a slovník `info`, který nese mj. informace o názvu souboru nebo adresářové struktuře a o částech (*pieces*), na které je původní soubor pro potřeby protokolu rozdělen. Jednoznačným identifikátorem torrentu je *infohash*, 160b SHA-1 hash slovníku `info` vyjádřeného v kódování *bencoding* [3].

Klient, který se chce stát uzlem (peerem) v konkrétní síti uzlů sdílejících jistý torrent (označuje se jako *swarm*), se ohlašuje trackeru. Zasílá mu především infohash, náhodně vygenerovaný vlastní identifikátor *peer ID* a číslo portu, na kterém bude komunikovat s ostatními uzly (buď pomocí TCP, nebo UDP, viz kap. 1.4). Odpověď trackeru obsahuje zejména adresy známých uzlů sdílejících požadovaný torrent. Klient může trackeru oznamovat dokončení nebo ukončení provozu, také by se trackeru měl pravidelně oznamovat (v intervalech požadovaných trackerem).

V původním protokolu komunikace s trackerem probíhá prostřednictvím HTTP GET požadavku na adresu určenou v metainfo souboru [3]. Klienti běžně podporují také komunikaci pomocí HTTPS, nicméně ukazuje se, že mnoho trackerů zabezpečenou komunikaci nevyužívá [5, str. 24].

Za účelem omezení režie protokolů TCP a HTTP vznikl také odlehčený protokol pro komunikaci s trackerem, *UDP Tracker Protocol for BitTorrent*, který redukuje provoz asi o 50 % [10]. [5, str. 24] naznačuje, že přibližně 50 % trackerů využívá tento protokol. UDP protokol je dvoufázový: klient první zasílá datagram s dotazem na připojení, server odpovídá s dočasným ID spojení, které klient může následující dvě minuty využívat pro následující dotazy „announce“.

## 1.2 BitTorrent s DHT

Tato práce se zabývá implementací DHT v sítích BitTorrent obvykle označovanou jako *Mainline DHT*, která je standardizována fórem bittorrent.org v [6] a dostupná ve většině běžně využívaných klientů (viz [13]). Vychází z principu P2P hashové tabulky Kademlia [6]. DHT pracuje se 160b prostorem klíčů, ukládané položky jsou seznamy adres bittorrentových uzlů (peerů), klíče jsou příslušné infohashe. Jednotlivé uzly v síti (*nodes*) vystupují pod vlastním 160b *node ID*, které je součástí prostoru klíčů.

Při vyhledávání klíče se zasílá dotaz na už známé uzly, pro které má metrika (XOR node ID a vyhledávaného klíče) nejnižší hodnotu. Pokud uzel hodnotu přímo nezná, odpovídá adresami jemu známých nejlépe vyhovujících uzlů. Vzhledem ke způsobu, jakým se tvoří směrovací tabulka, se výchozí uzel tímto způsobem ke správnému cílovému uzlu dostane v logaritmickém čase [7].

V případě použití DHT se klientská aplikace de facto připojuje do dvou samostatných P2P sítí. V síti DHT se označuje jako *node*, v síti BitTorrent pak jako *peer*. Klient (resp. uzel, node) si po spuštění vygeneruje node ID a začne prozkoumávat své blízké okolí, čímž postupně naplňuje svou směrovací tabulku. Na začátku k tomu potřebuje znát adresu alespoň jednoho dalšího uzlu (který už směrovací tabulku naplněnou má). K tomuto účelu jsou typicky v klientech přednastavené adresy (resp. doménová jména) několika známých startovacích (*bootstrap*) uzlů. Standard [6] sice říká, že by měly být startovací uzly součástí individuálních metainfo souborů, nicméně se zdá, že v praxi se tak neděje a při využití DHT se spoléhá na schopnost klienta. Standard zároveň uvádí, že by si klienti měli směrovací tabulku uchovávat i mezi spuštěními.

[4] uvádí jako možnost získání startovacího uzlu také náhodné procházení prostoru IP adres. Vzhledem k rozsáhlosti systému BitTorrent je pravděpodobnost nalezení stanice nezanedbatelná, postup je však v dnešní době komplikován tím, že každý DHT uzel typicky využívá jiný port, což prostor rozšiřuje $2^{16}$-krát.

K vyhledávání a oznamování hodnot (seznamů sousedů/peerů) se v síti DHT využívají tři typy zpráv: `find_node`, `get_peers` a `announce_peer`; pro účely řízení je dostupná také zpráva `ping`. Nosičem zpráv je jednoduchý protokol KRPC, který využívá UDP, nezajišťuje spolehlivý přenos.

## 1.3 Další možnosti výměny informace o uzlech (peerech)

Další možností, jak získat informace o stavu sítě, je rozšíření protokolu BitTorrent *Peer Exhange* (*PEX*). Rozšíření umožňuje uzlu (peerovi) vyměňovat si vzájemně s už známými sousedy (které získal např. z trackeru nebo z DHT) informace o jiných známých uzlech, což umožňuje rychlejší konvergenci pohledu na síť [12].

Někteří klienti implementují také rozšíření *Local Service Discovery* (*LSD*), které slouží k oznamování přítomnosti sousedů (peerů) v lokální síti [11]. Rozšíření využívá multicas-

tovou komunikaci – klient se na začátku automaticky připojí do skupiny 239.192.152.143 (resp. f15::efc0:988f pro IPv6), kde čeká na oznámení od ostatních a zároveň zasílá informace o sobě na port 6771.

## 1.4 Komunikace mezi uzly (peery)

Bittorrentové uzly (v roli peerů) mezi sebou komunikují s využitím *BitTorrent Peer Protocol* [3]. Protokol je přenášen buď pomocí TCP, nebo s využitím vlastního transportního protokolu *uTorrent Transport Protocol* (*uTP*, který je založen na UDP [8].

Klasický protokol pro příchozí komunikaci využívá libovolné TCP porty, ačkoliv specifikace zmiňuje, že typicky by mělo jít o porty 6881 až 6889 [3]. Při použití uTP nejsou UDP porty specifikovány. Stejně tak nejsou určeny ani porty využité při komunikaci týkající se DHT, mnoho předkonfigurovaných startovacích DHT uzlů však využívá právě číslo portu 6881. Klienti obvykle port volí náhodně – autoři [4, str. 711] vysledovali v internetu DHT komunikaci na všech 65 536 možných portech.

Pro DHT a peer protocol (zasílaný přes uTP) může klient využívat stejný UDP port. Zpráva `announce_peer` v DHT umožňuje použít parametr `implied_port`, do DHT je pak příjemcem jako port uzlu (peera) vložen zdrojový port DHT zprávy.

HTTP(S) i UDP trackery mohou běžet na libovolném portu, ten je součástí adresy v metainfo souboru.

# Kapitola 2

# Experimenty s klientem

Při experimentech byl využit klient **qBitTorrent**[1] ve verzi 4.4.1. Klient je založen na knihovně *libtorrent*[2]. Testování bylo prováděno primárně na virtuálním stroji se systémem Xubuntu 22.04. Bylo provedeno celkem dvanáct experimentů, které měly ukázat chování klienta v různých situacích. osm experimentů bylo provedeno s nastavením virtuálního síťového adaptéru v režimu LAN, aby nebyl zachytáván šum z lokální sítě. Pro další tři byl nastaven adaptér do režimu mostu, stroj se tedy choval jako součást lokální sítě. K dispozici nebyla síť, ve které by bylo možné navazovat IPv4 spojení z internetu.

V následujícím textu jsou experimenty označovány čísly, která odpovídají názvům odevzdaných *pcapng* souborů. Kompletní data, na kterých bylo experimentováno a testováno, jsou dostupná zde[3].

## 2.1 První spuštění

V prvních čtyřech experimentech bylo pozorováno chování klienta po prvním spuštění ihned po instalaci, po následném spuštění a po spuštění s vypnutým DHT. Klient uživatele vyzve k jednorázovému odsouhlasení, že rozumí následkům sdílení dat, a ihned po přechodu do hlavního okna programu se na pozadí začne zapojovat do DHT sítě. Začíná dotazy na překlad několika doménových jmen, která jsou pevně uložena v kódu klienta[4], jde o adresy známých startovacích serverů DHT:

- `dht.libtorrent.org`
- `router.bittorrent.com`
- `router.utorrent.com`
- `dht.transmissionbt.com`
- `dht.aelitis.com`

Klient pak na získané adresy serverů (na porty také uložené v kódu) okamžitě zasílá UDP datagramy DHT protokolu. Postupuje zřejmě v souladu s principy směrovací tabulky

[1] https://www.qbittorrent.org/
[2] https://www.libtorrent.org/
[3] https://drive.google.com/drive/folders/1xxJUkvanOKXdyviWrBy1otZR6tb5SWmU
[4] https://github.com/qbittorrent/qBittorrent/blob/v4_5_x/src/base/bittorrent/sessionimpl.cpp#L1783

sítě Kademlia a [6], tedy snaží se pro jednotlivé části prostoru klíčů (*k-buckets*) hledat několik reprezentativních uzlů. Jednotlivé bootstrap servery k tomu používá, zdá se, nahodile, dotazované klíče jsou tedy odlišné. Využívají se zprávy typu `get_peers`, a nikdy `find_node`, které jsou k tomuto účelu navrženy. Klient postupuje iterativně, tedy po přijetí odpovědi posílá nové požadavky na vrácené známé bližší uzly pro daný infohash, až dokud nedostane v odpovědi seznam `peers`.

Klient po spuštění náhodně volí jeden port, na kterém bude provozovat všechny druhy komunikace. Zároveň pomocí protokolu PCP žádá směrovač o vytvoření mapování v tabulce NAT (resp. PAT) pro umožnění komunikace zvnějšku (v souvislosi s tím se také připojuje do multicastové skupiny 239.255.255.250 pro uPnP). Zvláštní je, že nejprve žádá o mapování portu pro TCP provoz, pro UDP provoz zasílá požadavek až o něco později, když už DHT komunikace probíhá.

Klient po spuštění také ihned zahajuje připojení stanice do multicastové skupiny pro LSD. Pokud je proces klienta ukončen, stanice zasílá IGMP a MLD zprávy pro odhlášení ze skupin.

Použitý klient dále po spuštění překládá také jméno `download.db-ip.com` a navazuje HTTPS komunikaci s tímto serverem. Jde však o stažení databáze geolokačních informací pro IP adresy, které jsou užity v GUI. Z hlediska bittorrentové komunikace je toto nevýznamné.

Experiment č. 2 byl proveden pro potvrzení předchozích pozorování. Před spuštěním byly odstraněny všechny lokální soubory klienta. Komunikace zde probíhala v zásadě stejným způsobem, v zachycených datech chybí pouze některé DNS dotazy, to je ale způsobeno jejich výskytem v mezipaměti systému. Největší objem komunikace zde tvoří právě nevýznamné stahování databáze.

## 2.2 Druhé spuštění

Experiment č. 3 zachycuje druhé spuštění klienta. Otázkou zde bylo, zda si klient DHT směrovací tabulku ukládá, jak naznačuje specifikace. Komunikace však i zde měla obdobný charakter jako v předchozích případech, klient nejprve zjistil adresy startovacích uzlů, poté je oslovil a pokračoval iterativně na základě jejich odpovědí. To naznačuje, že DHT směrovací tabulky klient neukládá.

Zjištění bylo potvrzeno dodatečnými experimenty. Klient byl spuštěn a po krátkém čase ukončen. Poté byla na úrovni systému zablokována DNS komunikace. Po dalším spuštění už nebyl zaznamenán žádný DHT provoz, protože klient neznal adresy startovacích uzlů.

Klient umožňuje v nastavení deaktivovat službu DHT. Experiment č. 4 ukazuje, že v takovém případě žádná komunikace se startovacími uzly neprobíhá. Zůstává však požadavek směrovači na nastavení mapování v NAT, neboť klient na stejném portu provozuje kromě DHT také samotný bittorrentový provoz.

## 2.3 Stahování bez využití trackeru

Experiment č. 5 spočíval ve stažení obrazu linuxové distribuce Arch Linux. Od obvyklých torrentů se odlišuje tím, že poskytovaný metainfo soubor neobsahuje žádné adresy trackerů, spoléhá tak výhradně na DHT. Použito bylo standardní nastavení klienta, tj. náhodný port (pro uTP i DHT komunikaci), povolené transportní protokoly TCP i uTP, povolené DHT, PEX a LSD.

Specifikace uvádí, že by DHT uzel měl vracet v odpovědi `get_peers` 8 známých nejbližších uzlů (pokud sám neobsahuje seznam peerů). V komunikaci se vyskytly uzly, které tento počet striktně dodržují i v případě, kdy znají dalších uzlů méně – pak jednu adresu v odpovědi opakují. Nachází se zde i několik případů, kdy je vráceno uzlů méně nebo více, v jednom případě dokonce 260. Některé odpovědi mají slovník s počtem záznamů prázdný, některé jej vůbec neobsahují.

Typická délka dat v DHT odpovědi se pohybuje mezi 283 B a 317 B (v závislosti na počtu uzlů a délkách polí, která uzel volí dle vlastního uvážení. Odpovědi, které nesou hodnotu (seznam peerů), jsou přirozeně větší. Nejčastější délka seznamu byla 100, tyto odpovědi mají délky od 1 107 B do 1 127 B.

V rámci experimentů č. 6 a 7 bylo testováno chování při vynucení transportního protokolu TCP nebo uTP. Za povšimnutí stojí závěr komunikace v zachycené komunikaci č. 7, kde po vypnutí klienta na stanici stále přicházejí UDP datagramy jak se zprávami peer protokolu (uTP), tak s DHT požadavky od jiných uzlů. Je tedy zjevné, že se klient při svém chodu zapojil do sítě DHT a poskytoval informace i ostatním uzlům. DHT odpovědi generované v průběhu komunikace klientem je zde možné vyfiltrovat např. výrazem:

```
udp.srcport == 44176 && udp contains "y1:r"
```

V experimentu č. 8 byla sledována komunikace klienta po spuštění a přidání několika torrentů s trackery až do dokončení stahování. Klient i v tomto případě vyměňoval informace o peerech pomocí DHT, nicméně zachycena byla i komunikace s trackery. Jeden kontaktovaný tracker využíval HTTP, ostatní UDP. Datagramy s prvním kontaktem trackeru je možné vyfiltrovat např. podle specifikací definovaného řetězce výrazem:

```
udp.payload contains 17:27:10:19:80
```

Za povšimnutí zde stojí také značný objem TLS komunikace. Incializační *Client Hello* zpráva ukazuje, že se jedná o server `download.manjaro.org`. Evidentně jde o využití rozšíření *WebSeed* definované v [1], které umožňuje do metainfo souboru zahrnout URL souboru na běžném HTTP nebo FTP serveru. Klienti tak při stahování mohou využít i běžný klient-server přístup.

Obdobné výskedjy

### 2.3.1 Využití Peer Exchange protokolu

Experiment č. 12 zkoumal, jak se klient zachová, pokud je v průběhu stahování vypnut a při následném spuštění nemá dostupné startovací uzly. Použit byl podobný postup jako v kap. 2.2, zde se však po spuštění začal stahovat výše zmíněný obraz. V průběhu stahování byl klient ukončen, následně byla v systému zablokována DNS komunikace.

Po dalším spuštění se klient, v souladu s předchozím zjištěním, nezačal připojovat do DHT sítě. Ukázalo se ale, že klient pro běžící stahování ukládá adresy sousedů (peerů), stahování tedy bylo obnoveno. Počet dostupných sousedů se dokonce zvětšoval, a to díky rozšíření Peer Extension, které umožnilo výměru informací o síti v rámci už navázané komunikace se sousedy. Poukazuje to na značnou odolnost sítě (alespoň pokud je skupina uzlů sdílejících jeden torrent dostatečně rozsáhlá).

Uzly v rámci peer komunikace přenášejí také informaci o tom, zda podporují DHT (resp. zda jsou také DHT uzlem) a pokud ano, na jakém portu DHT komunikaci očekávají. Dalo by se očekávat, že klient při zjištění sousedů s podporou DHT využije tyto sousedy jako startovací uzly a pokusí se do sítě DHT připojit (ostatně podobně to ukládá specifikace [6]). Použitý klient tak ovšem nečiní a při počátečním neúspěchu se do DHT nepřipojuje.

## 2.4 Přenos v síti LAN

Experimenty č. 9 a 10 sledovaly výměnu souboru mezi dvěma počítači umístěnými ve stejné síti. V exp. 9 byl torrent vytvořen na jiné stanici. Ta o něm okamžitě umístila informaci do DHT, a to pomocí IPv4 i IPv6 komunikace. Stanice se o sobě však dříve dozvěděly prostřednictvím LSD: sledovaná stanice ohlásila v multicastové skupině, že je peerem pro daný torrent (zpráva obsahuje infohash), tato zpráva byla doručena zdrojové stanici, která následně iniciovala bittorrentové spojení pomocí uTP směrované na lokální adresu stanice. Zajímavé je, že později obě stanice přešly na IPv6 provoz, a to dokonce paralelně na dvou adresách, které zdrojová stanice měla přiřazeny. Těžko říct, zda záměrně, nebo shodou okolností byl na jedné z nich provoz přenášen pomocí TCP, na druhé pomocí uTP. V komunikaci je možné vysledovat také úspěšné získání IPv4 i IPv6 adres zdrojové stanice z DHT (přes veřejnou IPv4 však byla stanice nedostupná).

Experiment č. 10 proběhl obdobně, zde však byl na klientské stanici zakázán protokol IPv6. Na počátku klient úspěšně našel venkovní IPv4 adresu v DHT a pokusil se na ni připojit, ovšem neúspěšně. Po chvíli s ním ale navázala zdrojová stanice spojení na základě LSD požadavku.

## 2.5 Šifrování provozu

Šifrování je v kontextu bittorrentové komunikace považováno spíše za metodu obfuskace obsahu, kterou uzly mohou provádět, aby se vyhýbaly detekci a omezování provozu např. ze strany poskytovatelů připojení [9]. V současné době pro něj neexistuje standard BEP, do klientů je přidávána po vzoru implementace Vuze[5].

V klientovi qBitTorrent je možné šifrování povolit, zakázat nebo vynutit. Vzhledem k nevelkému rozšíření však jeho vynucení i u torrentů s jinak podstatnou skupinou sdílejících uzlů, jako jsou např. obrazy linuxových distribucí, vede ke znatelnému snížení počtu sousedů, a tedy i ke snížení rychlosti, viz obr. 2.1.

| Name | Down Speed | ETA | Peers |
|---|---|---|---|
| xubuntu-22.10-desktop-amd64.iso | 48,7 KiB/s | 15h 55m | 1 (3) |
| ubuntu-22.10-live-server-amd64.iso | 1,3 MiB/s | 16m | 1 (8) |
| ubuntu-20.04.6-live-server-amd64.iso | 48,7 KiB/s | 7h 54m | 1 (9) |
| xubuntu-22.04.2-desktop-amd64.iso | 34,8 KiB/s | 1d 19h | 1 (14) |
| ubuntu-22.04.2-live-server-amd64.iso | 47,5 KiB/s | 13h 38m | 1 (30) |
| ubuntu-22.10-desktop-amd64.iso | 48,7 KiB/s | 21h 44m | 1 (33) |
| ubuntu-22.04.2-desktop-amd64.iso | 45,5 KiB/s | 1d 10h | 1 (49) |

Obrázek 2.1: Nízké přenosové rychlosti při vynucení šifrování.

Komunikaci s vynuceným šifrováním zachycují experimenty č. 13 a 14. V prvním případě zvládl klient navázat spojení pouze s jedním uzlem. Z průběhu komunikace je patrné, že klient použil HTTPS tracker, komunikace s protistranou probíhala přes TCP. V jistých fázích komunikace je možné pozorovat sekvenci: dva delší pakety od protistrany ke klientovi (cca 1 500 B), jeden krátký ACK paket od klienta k protistraně, který nenese žádná data ($< 100$ B).

[5] https://web.archive.org/web/20220901061247/http://wiki.vuze.com/w/Message_Stream_Encryption

# Kapitola 3

# Možnosti detekce

Jak bylo naznačeno výše, bittorrentový provoz se skládá z několika druhů komunikace, kterou je možné sledovat různými způsoby. Samostatně je možné sledovat komunikaci s trackery, komunikaci v síti DHT a komunikaci mezi uzly (peery), případně také LSD dotazy.

Většina níže popsaných metod je založena na hledání signatur uvnitř paketů (*deep packet inspection*). V některých případech se nabízí také doplnění o filtrování na základě velikosti přenášených paketů. Naznačeny jsou také možné postupy založené na statistických vlastnostech komunikace.

Použitý klient provozuje všechnu UDP komunikaci a příchozí TCP komunikaci na stejném portu. Tohoto pozorování by mohlo být využito pro efektivnější filtraci: např. by bylo možné uvažovat pouze komunikaci na portech s vyšším objemem přenesených dat nebo sledovat, na kterých portech probíhá TCP i UDP komunikace.

## 3.1 Komunikace s trackery

Pro identifikaci komunikace s trackery je nutné volit odlišné přístupy pro HTTP, HTTPS a UDP trackery. Není zde možné využít identifikaci podle čísel portů, neboť trackery často naslouchají na nestandardních portech.

V případě HTTP komunikace je možné filtraci provádět podle několika prvních bajtů v TCP datech. Zprávy mají následující charakteristiky:

1. První tři bajty dat obsahují řetězec `GET`,
2. za ním následuje dotazovaný HTTP zdroj ukončený znakem nového řádku `\n`,
3. prakticky všechny trackery jsou dostupné na adrese `/announce`,
4. i pokud by byla adresa zdroje jiná, mezi parametry požadavku se musí nutně vyskytovat řetězec `info_hash=`.

Při UDP komunikaci s trackery je nutné odlišit počáteční „navazování spojení“ a následné ohlašování uzlu výměnou za seznam ostatních uzlů. Počáteční požadavek od klienta je možné jednoznačně identifikovat tak, že obsah datagramu začíná 64b sekvencí `0x41727101980`. Další komunikaci je však nutné rozlišovat podle ID spojení, které tracker v odpovědi vrátí. Zprávy s navazováním spojení a ohlašovací dotaz je možné vyfiltrovat také podle pevně určené délky UDP dat (16, 16 a 98 B), následné odpovědi však mají velikost dynamickou podle počtu vrácených peerů (alespoň 20 B).

Zřejmě nejméně spolehlivé možnosti detekce budou dostupné v případě HTTPS komunikace s trackery. Zde je asi jedinou možností hledání pravděpodobných řetězců jako „torrent“ nebo „tracker“ v TLS handshake zprávách, ve kterých může být uvedeno jméno serveru. I v případě úspěšného detekování však nebude možné z odpovědí trackeru zjišťovat seznam uzlů.

Použití statistických metod pro detekci komunikace s trackery by bylo omezené, neboť objem této komunikace není velký a neprobíhá pravidelně (čas do dalšího ohlášení oznamuje tracker ve své odpovědi).

## 3.2 Komunikace v síti DHT

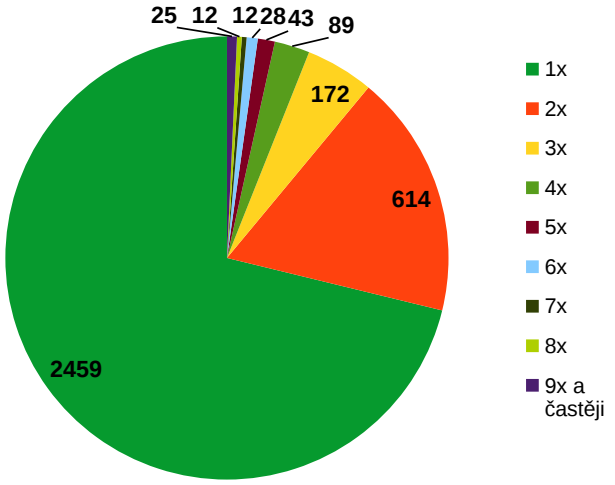
DHT zprávy jsou vždy přenášeny přes UDP a nepoužívá se pro ně šifrování. Je náročnější pro ně určit dostatečně spolehlivou signaturu, avšak dá se vycházet ze skutečnosti, že zprávy jsou vždy slovníky v kódování bencoding.

Zprávy začínají sekvencí `dI:X`, kde `d` označuje typ slovník, `I` je ASCII znak číslice označující délku prvního klíče (žádné v protokolu použité zprávy nemají klíče delší než 9 znaků) a `X` je řetězec znaků délky `I`. Dotazy je možné odlišit přítomností sekvence `1:y1:q`, odpovědi přítomností sekvence `1:y1:r`. Zprávy by měly být vždy zakončeny znakem `e`, který ukončuje slovník.

Zprávy použitého klienta s dotazy `find_node` mají vždy velikost obsahu 104 B, zprávy s dotazy `ping` mají 65 B (tyto nejsou příliš časté). Velikost odpovědí se liší podle počtu vrácených uzlů, obvykle jsou delší než 280 B (pokud je přeneseno typických 8 adres DHT uzlů).

Vhodným statistickým ukazatelem by zde mohl být počet různých cílových IP adres pro jeden zdrojový UDP port (nebo naopak různých zdrojových adres pro jeden cílový UDP port) – v typické komunikaci se kontaktuje velké množství uzlů DHT, většina z nich ale pouze jednou. Graf na obr. 3.1 ukazuje četnost počtů DHT zpráv zaslaných na stejnou IP adresu v exp. 7 (běžné stahování několika torrentů). Z celkem 3 454 adres bylo 2 459 (71,2 %) kontaktováno pouze jednou, dalších 614 (17,8 %) pouze dvakrát.

Dalším možným ukazatelem by mohl být poměr přijatých a celkového množství zpráv na daném UDP portu. Při experimentech se tento poměr pohyboval mezi 15 a 27 %.

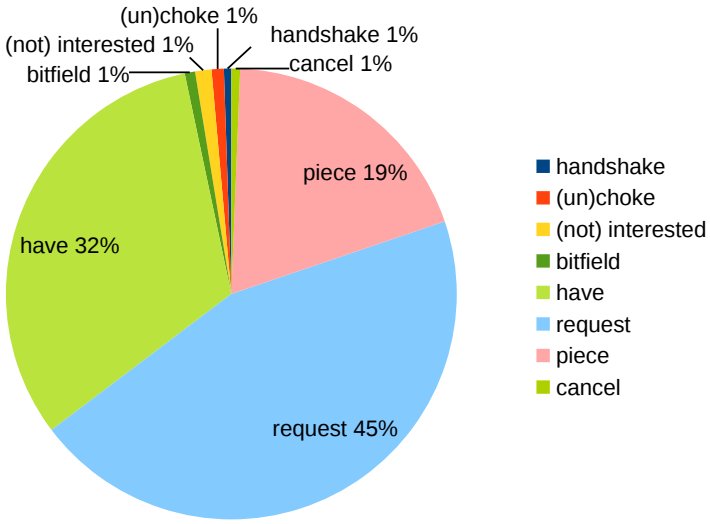



Obrázek 3.1: Četnost počtů DHT zpráv zaslaných na jednu adresu v experimentu č. 7.

## 3.3 Komunikace mezi uzly (peery)

Pro filtrování komunikace mezi uzly je nutné analyzovat jak TCP, tak UDP provoz, nicméně uTP implementuje transportní vrstvu nad UDP, samotné zprávy mezi uzly jsou při jeho použití totožné. Bittorrentové spojení je možné určit na základě počáteční výměny mezi uzly, kde prvních 21 B aplikačních dat obsahuje řetězec `19BitTorrent protocol`.

Pokud by zachycená data neobsahovala počáteční handshake, detekce bude složitější. Protokol je tvořen proudem zpráv, každá zpráva začíná 4 B s délkou zbytku zprávy, následuje 1 B s typem zprávy. Signatury je možné sestrojit pro řídicí zprávy protokolu `choke`, `unchoke`, `interested` a `not interested`, které nenesou žádná data a mají fixní délku: zde půjde o sekvence bajtů `00000001 0X`, kde `X` je typ zprávy od 0 do 3. Fixní délku by měly mít také zprávy `have`, `request` a `cancel`. Ve zprávách `piece`, které nesou samotná data, hledání podle signatur prakticky není možné, neboť je pro ně charakteristický pouze jeden bajt. Poměr typů zpráv v exp. 6 ukazuje graf na obr. 3.2.




Obrázek 3.2: Poměr typů zpráv protokolu BitTorrent v experimentu č. 6.

# Kapitola 4

# Implementace

Nástroj pro monitorování provozu v síti BitTorrent byl implementován v jazyce Python s využitím knihovny scapy. Není využito předzpracování, nástroj čte přímo soubory ve formátu pcapng (zřejmě je možné použít také klasický formát pcap, ale s tím probíhalo pouze minimální testování).

Rozpoznávání provozu je v principu založeno na podrobném zkoumání komunikace a sestavování stavu sítě. Program čte ze vstupu jednotlivé pakety, případně skládá IP fragmenty, a pro každý nalezený UDP a TCP paket spouští několik trasovacích akcí. Tyto akce si udržují seznamy vhodně identifikovaných toků, které postupně obohacují o zjištěné informace. Původní ambicí bylo implementovat detekci bezstavově, což by více odpovídalo klasickým (a v praxi použitelným) metodám filtrace, nicméně pro účely získání požadovaných informací ze záznamu komunikace je tento přístup jednodušší a předpokládám, že přesnější (také s přihlédnutím k absenci požadavku na rychlost).

## 4.1 Trasování DNS komunikace

Klienti pro vyhledávání startovacích uzlů a trackerů typicky využívají DNS. Pokud scapy v UDP datagramu detekuje DNS provoz, jeho obsah je zanalyzován speciálním způsobem (a nepokračuje pak do dalších trasovacích akcí). V tabulce jsou uchovávány DNS odpovědi, které obsahují klíčová slova jako *dht*, *tracker*, *torrent*. Tyto záznamy jsou poté použity ke zpřesnění dalších zjištění, viz dále.

## 4.2 Trasování DHT komunikace

Pro každý UDP datagram se testuje, zde začíná a končí charakteristickými sekvencemi (viz 3.2). Pokud ano, spouští se zpracování možné DHT zprávy. Obsah je dekódován (chyba dekódování značí, že nejde o DHT zprávu). Pokud jde o DHT požadavek, do tabulky je vložen (nebo je v ní nalezen) objekt reprezentující potenciální DHT uzel identifikovaný cílovou IP adresou, cíl. portem a identifikátorem transakce zprávy. V případě, že jde o `announce_peer` požadavek, je vytvořen také nový záznam v tabulce peerů o odesilateli požadavku. Pro datagramy s DHT odpovědí se v tabulce hledá záznam o jeho zdroji. Pokud je nalezen, zpráva je tím spárována s požadavkem. Ze zpráv obsahujících seznamy peerů se vytvoří záznamy v tabulce peerů, obdobně se rovnou vytvoří záznamy o DHT uzlech.

Takto je zároveň možné, pro účely režimu `-init`, odlišit startovací uzly od dalších, později nalezených: pokud v tabulce DHT uzlů nebyl pro příchozí **požadavek** uzel nalezen,

znamená to, že se jeho adresa vzala odjinud než z dříve zaznamenaných DHT odpovědí, tj. například je napevno uložena v klientovi. Odlišování startovacích uzlů od později nalezených je (očekávatelně) spolehlivé, pouze pokud vstupní soubor obsahuje počáteční komunikaci klienta. Zavedena je tedy ještě další heuristika: nalezený DHT uzel je považován za startovací, pokud příslušná DHT zpráva byla maximálně $N$ UDP datagramů vzdálená od předchozí přijaté zprávy s DHT uzly. Připojování do DHT se totiž dá považovat za jistý shluk zpráv, které proběhnou ve velmi blízké časové souslednosti a pouze jednou. Výchozí hodnota je 50, dá se určit parametrem `-bootstrap-cutoff`. Pro vstupy, které nezachycují začátek komunikace, by se patrně hodila ještě menší hodnota.

Další možnou heuristikou jsou DNS dotazy: za startovací uzly můžeme považovat pouze ty, jejichž adresa byla předtím zjištěna v DNS rezoluci. Tato heuristika je však značně slabá, např. protože systém může překlad mít uložený v mezipaměti, takže se pak neobjeví v záchytu komunikace na fyzickém rozhraní. Nastavením výše uvedeného parametru na 0 se vynutí použití výhradně této charakteristiky.

Podotkněme, že s použitím tohoto postupu je možné informace o DHT uzlech vypisovat už v průběhu čtení komunikace. Program však všechny výpisy realizuje až po přečtení celého vstupu, a to kvůli ostatním trasovacím akcím, u kterých by „on-line“ režim nepodával dostatečně přesné informace.

## 4.3 Trasování komunikace mezi uzly (peery)

Pro sestavení informací o stahovaných souborech (a pro zpřesnění informací o uzlech, ačkoliv zde poskytuje vcelku solidní obsah informací i samotné sledování DHT komunikace) je nutné sledovat zprávy peer protokolu mezi uzly. Zde však nepostačuje sledování jednotlivých zpráv jako u DHT, protože protokol má charakter proudu: na začátku jsou vyměněny režijní informace (v rámci handshake), poté jsou už zprávy přenášeny v podobě sekvence délka – typ – data. Pro získávání ucelených informací o komunikaci je tedy nutné rekonstruovat proud transportního protokolu a číst přenos od začátku do konce.

Sítě BitTorrent využívají pro přenos dva transportní protokoly, běžné TCP a vlastní uTP, který tvoří vrstvu nad UDP, ale podobně jako TCP poskytuje řízení toku a spojení. Nástroj proto obsahuje vlastní mechanismy (`tcp_tracer`, `utp_tracer`), kterým jsou dodávány TCP segmenty, resp. UDP datagramy, a které rekonstruují přenášený proud bajtů, v podstatě tedy implementují stavové automaty těchto protokolů. Rekonstruované zprávy jsou dodávány do BTP (BitTorrent Protocol) trasovače, který z dodávaného proudu bajtů postupně čte aplikační zprávy. Tyto tři mechanismy byly převzaty z veřejného repozitáře https://github.com/elektito/bttools a do značné míry upraveny pro potřeby tohoto nástroje.

Z BTP provozu jsou aktuálně využity zprávy `piece`, `bitfield` a `request`, které nesou informace o přenášených částech (*pieces*) souboru. Z provozu samotného není vždy možné přesně určit charakteristiky přenášených souborů: metadata jsou součástí metainfo souboru, mezi peery se pouze může vyměňovat s využitím rozšíření protokolu. Reportované velikosti jsou tedy určeny jako odhad podle polí `begin` a `offset` označujících pozici uvnitř části, která se v dané zprávě přenáší. Velikost části je určena jako nejbližší mocnina dvojky k největšímu nalezenému součtu těchto dvou hodnot. Zpráva `bitfield` přenáší bitovou masku s informacemi o už stažených částech, její velikost je tedy použita jako odhad počtu částí (s přesností na násobky osmi).

## 4.4 Trasování komunikace s trackery

V odevzdaném nástroji je implementováno také sledování komunikace s UDP trackery. Datagramy jsou filtrovány s pomocí charakteristických signatur, využito je opět stavové řízení: výměna vždy začíná ustavením pseudospojení, v reakci na tyto zprávy se ukládají informace do tabulky. Pokud datagram projde počátečním filtrem, hledá se buď signatura ustavující zprávy, nebo záznamy o předchozí komunikaci. Podle informací z trackeru se pak doplňuje tabulka peerů. V současné bittorrentové komunikaci však zásadně převažuje využití DHT, proto je toto spíše okrajová záležitost. Sledování komunikace s HTTP trackery implementováno nebylo, a to zejména z časových důvodů, nicméně kód je k tomu připraven a v zásadě by šlo o triviální rozšíření.

# Kapitola 5

# Zhodnocení

V rámci projektu jsem nastudoval principy fungování bittorrentových sítí a protokolů, které se v nich využívají. Provedl jsem sadu experimentů, ve kterých jsem sledoval chování klienta qBitTorrent a charakter komunikace v různých situacích a konfigurací. Na základě poznatků jsem implementoval nástroj, který umožňuje ze zachyceného přenosu na síťovém rozhraní extrahovat informace o proběhlé komunikaci v síti DHT a BitTorrent. Oproti požadavkům zadání poskytuje některé informace navíc, například odhady podílů uzlů na přijatém souboru, seznamy všech DHT uzlů nebo přesnější informace mezi vztazích DHT uzlů a bittorrentových uzlů. Nástroj jsem otestoval na delších záznamech skutečné komunikace. Pro objektivní ověření jeho funkčnosti mi schází vhodné metriky, jeho výstupy však vypadají dostatečně věrohodně.

Použité řešení umožňuje podrobnou analýzu komunikace i nad rámec požadavků ze zadání, protože rekonstruuje proudy komunikace a ukládá si vcelku podrobné informace o síti. To je však na druhou stranu vykoupeno nízkou rychlostí, která vyplývá z nutnosti udržovat různé informace o tocích, vyhledávat v nich a mj. nad každým realizovat konečný automat transportního protokolu. Předpokládám, že výkon programu by bylo možné zlepšit lepší prací s vyhledávacími tabulkami, např. vhodným promazáváním nepotřebných záznamů.

# Literatura


[1] Burford, M. *WebSeed - HTTP/FTP Seeding (GetRight style)* [online]. BitTorrent.org, 2017 [cit. 2023-04-10]. Dostupné z: https://www.bittorrent.org/beps/bep_0019.html.

[2] Cohen, B. *Incentives Build Robustness in BitTorrent* [online]. 2003 [cit. 2023-05-10]. Dostupné z: http://bittorrent.org/bittorrentecon.pdf.

[3] Cohen, B. *The BitTorrent Protocol Specification* [online]. BitTorrent.org, 2017 [cit. 2023-04-10]. Dostupné z: https://www.bittorrent.org/beps/bep_0003.html.

[4] Dinger, J. a Waldhorst, O. P. Decentralized Bootstrapping of P2P Systems: A Practical View. In: Fratta, L., Schulzrinne, H., Takahashi, Y. a Spaniol, O., ed. *NETWORKING 2009*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, s. 703–715. ISBN 978-3-642-01399-7.

[5] Florek, D. *Detekce provozu protokolu BitTorrent*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Libor Polčák, Ph.D. Dostupné z: https://www.fit.vut.cz/study/thesis/20313/.

[6] Loewenstern, A. a Norberg, A. *DHT Protocol* [online]. BitTorrent.org, 2020 [cit. 2023-04-10]. Dostupné z: https://www.bittorrent.org/beps/bep_0005.html.

[7] Maymounkov, P. a Mazières, D. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In: Druschel, P., Kaashoek, F. a Rowstron, A., ed. *Peer-to-Peer Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, s. 53–65. ISBN 978-3-540-45748-0.

[8] Norberg, A. *UTorrent transport protocol* [online]. BitTorrent.org, 2017 [cit. 2023-04-10]. Dostupné z: https://www.bittorrent.org/beps/bep_0029.html.

[9] Sar, E. van der. *Interview with Bram Cohen, the inventor of BitTorrent* [online]. TorrentFreak, 2007 [cit. 2023-05-10]. Dostupné z: https://torrentfreak.com/interview-with-bram-cohen-the-inventor-of-bittorrent/.

[10] Spek, O. van der. *UDP Tracker Protocol for BitTorrent* [online]. BitTorrent.org, 2017 [cit. 2023-04-10]. Dostupné z: https://www.bittorrent.org/beps/bep_0015.html.

[11] The 8472. *Local Service Discovery* [online]. BitTorrent.org, 2017 [cit. 2023-04-10]. Dostupné z: https://www.bittorrent.org/beps/bep_0014.html.

[12] The 8472. *Peer Exchange (PEX)* [online]. BitTorrent.org, 2017 [cit. 2023-04-10]. Dostupné z: https://www.bittorrent.org/beps/bep_0011.html.

[13] Wikipedia contributors. *Comparison of BitTorrent clients — Wikipedia, The Free Encyclopedia: Features I* [online]. 2023 [cit. 2023-05-10]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Comparison_of_BitTorrent_clients&oldid=1147257461.