VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií

Přenos dat, počítačové sítě a protokoly 2023

Dokumentácia k projektu

Monitorování a detekce provozu BitTorrent

Marek Žiška **(xziska03)**

24. apríla 2023

Obsah

1 Úvod

Táto práca práca vznikla ako projekt do predmetu "Přenos dat, počítačové sítě a protokolyä je zameraná na štúdium a analýzu komunikácie protokolu BitTorrent v sieti LAN a následnú implementáciu nástroja schopného detekovať a monitorovať provoz BitTorrent.

1.1 O realizácií projektu

Nástroj bol implementovaný v jazyku Python s využitím rôznych knihovní špecifikovaných v súbore `requirements.txt`, ale najmä pomocou nástroja `tshark`. Tento nástroj bol využívaný na filtráciu pcap súborov s cieľom extrahovať údaje potrebné na analýzu BitTorrent komunikácie. Projekt som započal s cieľom pochopiť kontext, na splnenie tohto cieľa bolo nutné štúdium rôznych štúdií zaoberajúcich sa peer-to-peer sieťami, systémom BitTorrent a jeho rôznych "BitTorrent Enhancement Proposals"(BEPs). Následne som zo získaných relevantných poznatkov spísal teoretickú časť tejto dokumentácie. Nasledovalo experimentálne overovanie, ktoré prebiehalo v prostredí virtuálneho obrazu *PDS-VM.ova* s operačným systémom *Ubuntu 20.04LTS* a taktiež na hlavnom zariadení s operačným systémom *MacOS*. Z dôvodu multiplatformnosti som si ako BitTorrent klienta zvolil *qBitTorrent*, ktorý som zkompiloval z dostupných github repozitárov [5]. V rámci možností klienta som skúsil rôzne možnosti nastavení pri sťahovaní, ako napríklad využitie len nešifrovanej komunikácie, alebo využite výhradne len TCP protokolu. Pre rôzne nastavenia som si pri sťahovaní súbora z *qBitTorrent* klienta nahrával komunikáciu vo `wireshark`. Nad súbormi so zachytenou komunikáciou už následne prebiehala implementácia a testovanie nástroja.

2 Teória

2.1 Peer-to-peer siete

Sieť so vzájomným sprístupňovaním(peer-to-peer) v poslednom období naberá na popularite, aj napriek jej kontroverzie s pirátstvom v filmovom a hudobnom priemysle. Oproti klasickej architektúre klient-server má sieť peer-to-peer výhody so zvýšenou robustnosťou a dostupnosťou zdrojov, konkrétne:

- zvýšená šírka pásma (bandwidth)
- väčší ukladací priestor (storage space)
- zvýšený výpočetný výkon (comuting power)
- efektívnejšie využite výpočetného výkonu

Táto architektúra sa dá jednoducho definovať ako *"komunikačný model, v ktorom má každá zúčastnená strana rovnaké možnosti a ktorákoľvek strana môže začať komunikačnú reláciu"*.[3] Z tejto definície vyplýva že peer-to-peer sieť je z hľadiska funkcionality alternatívou klasickej klient-server architektúry. Hlavným rozdielom medzi nimi je to že v klasickej peer-to-peer sieti sú všetci zúčastnený (uzly) rovnocenný. Pre zaručenie funkcie služby a obsahu ponúkaného sieťou, je nutné aby každý zo zúčastnených poskytoval svoje vlastné zdroje(výpočetný výkon, úložisko, kapacitu sieťového pripojenia atď.). Ostatný zúčastnený následne priamo pristupujú k týmto zdieľaným zdrojom, ak sa v sieti nevyskytuje centrálna entita. Ak by peer-to-peer sieť obsahovala, takúto entitu hovoríme o *Hybrid peer-to-peer* sieti. Táto centrálna entita prepája časti poskytovaných služieb v rozdielnych zhlukoch uzlov. Ak zlyhá táto centrálna entita, celá časť siete bude nedostupná.

2.2 BitTorrent

BitTorrent je open-source technológia/protokol, ktorá s využitím hybridnej peer-to-peer siete poskytuje jednoduché zdieľanie súborov, s výhodami týchto sietí popísaných vyššie (2.1). Hlavnou výhodou, ktorú tento protokol poskytuje je zníženie záťaže pri nahrávaní súborov do siete. Toto je docielené využívaním upload kapacít uzlov, ktorý zo siete sťahujú súbory do vlastného zariadenia a zároveň súbor rozdeľuje na menšie časti a odosiela ich ďalším uzlom. Uzol ktorý chce potom stiahnuť súbor hľadá uzly, ktoré majú uložené jednotlivé časti žiadaného súbora. Po ich nájdení uzol započne sťahovanie simultánne z nájdených uzlov. Týmto zjednodušeným spôsobom v podstate funguje BitTorrent a je to aj príklad dôvodu zmenšenia záťaže na prostriedkoch odosielateľa. Tento princíp má ale aj svoje problémy, ako aj napríklad vplyv popularity žiadaných súborov na stiahnutie. Nepopulárne súbory sú málo sťahované, takže hľadanie uzlov s časťami tohto súbora môže byť náročné. Preto je protokol BitTorrent najefektívnejší pre populárne súbory, alebo aj pre zdieľanie operačného systému Linux, čo bol pôvodný zámer tvorcu BitTorrentu *Bram Cohen*. V BitTorrent sa pojmy "peerä "node"často používajú zameniteľne, ale majú trochu odlišný význam. *Peers* sú jednotliví klienti v sieti, ktorí sťahujú a zdieľajú súbor s ostatnými. Na druhej strane, *nodes* sa zvyčajne považujú len za koncové body v sieti, ktoré sa nezúčastňujú zdieľania súborov, a vzťahujú sa na akúkoľvek entitu v sieti vrátane *peers* a *trackers* [4].

2.3 Architektúra BitTorrent

Architektúra sa skladá z viacero entít, vrátane:

- **Klasického web servera**
- **Statického súbora "metainfo"** - malý súbor (známe ako súbory .torrent) obsahujúci informácie o zdieľaných súboroch(názvy, veľkosti a kontrolné súčty) a obsahuje aj informácie o *tracker-u*.
- **BitTorrent tracker** - server sledujúci účastníkov zúčastňujúcich sa na zdieľaní súborov.
- **Originálny downloader (sťahovač)** - Používateľ, ktorý ako prvý nahral súbor *torrent* do siete.
- **Webové prehliadače koncových užívateľov** - koncový používatelia ich používajú na vyhľadávanie a sťahovanie torrent súborov.
- **Downloader koncového užívateľa** - koncový používatelia ich používajú na sťahovanie súborov prostredníctvom protokolu BitTorrent.

2.3.1 Trackers

Trackers zohrávajú kľúčovú úlohu v uľahčení komunikácie medzi uzlami. Sú to centrálne servery, poskytujúce zoznam uzlov v roji(swarm), s informáciami o tom aké uzly majú aký konkrétny súbor(alebo ktoré časti súboru) k dispozícii na zdieľanie.

Pri sťahovaní alebo nahrávaní klient pravidelne posiela správy pre *tracker*, ktorými ho informuje o jeho celkovom stave, aké časti súborov už boli stiahnuté a ktoré ešte stále potrebuje stiahnuť. *Tracker* používa tieto informácie na aktualizáciu svojho zoznamu uzlov a na ich ďalšiu distribúciu ostatným zúčastneným, ktorí o tieto informácie požiadajú.

Niektorí *trackers* presadzujú pravidlá a zásady pre sťahovanie a zdieľanie súborov. Napríklad môžu pre účasť v komunikácií vyžadovať, aby používatelia udržiavali minimálny *upload-to-download ratio*. Taktiež môžu zaviesť čiernu listinu klientov, o ktorých je známe, že sú nespoľahliví alebo že v minulosti zdieľali materiál chránený autorskými právami.

2.3.2 Peer protokol

Protokol definuje vzájomnú komunikáciu klientov pri sťahovaní a nahrávaní súborov a je založený na sérii správ uzlami. Prvé spojenie k jednému z uzlov prebieha odoslaním správy o nadviazaní spojenia (handshake), ktorá sa skladá z hlavičiek, *info hashu*, *metainfo* súbora a *peer ID* prijímateľa. Prijmajúci uzol overuje odoslané údaje, v prípade nezrovnalosti sa spojenie ukončuje (zamedzí sa tak odoslaniu nekompletných alebo zlých súborov, alebo komunikácie s nesprávnym uzlom). Po naviazaní *handshake* si klienti vymieňajú rôzne typy správ s prefixom(identifikujúceho dĺžku správy). Správa s dĺžkou 0 je známa ako *keep alive* a slúži na udržanie spojenia(jej odoslanie sa očakáva cca každé 2 minuty, môže sa to ale meniť vzhľadom na klienta). Ostatné správy môžu byť len určitého typu identifikované prvým bytom správy. Správou *have* sa uzly navzájom informujú o uložených častiach súborov alebo o tom že dokončili sťahovanie súbora. Po prijatí správy "have"si klient aktualizuje vlastný zoznam dostupných častí súborov a vyžiada si časti, ktoré ešte prípadne potrebuje.

Na prenos údajov medzi sebou si klienti taktiež vymieňajú správy *request* a *piece*. Klient odosiela správu *request* na vyžiadanie konkrétnej časti súbora od druhého uzla a ten mu odpovie správou *piece* obsahujúcou požadované údaje. Okrem týchto základných správ obsahuje peer protokol aj správy na zrušenie požiadaviek (správa *cancel*), škrtenie uzlov (správa *choking unchoking*), informovanie o strate alebo vzniku záujmu o časť súbora (*interested*, *not interested*). Podrobnejšia špecifikácia peer protokolu, správ a formátu správ je popísaná v BEP-3 [1].

2.3.3 DHT protokol

Protokol DHT (Distributed Hash Table) je decentralizovaná metóda lokalizácie uzlov v sieti BitTorrent. Podľa BEP-5 [4] funguje protokol DHT tak, že udržiava distribuovanú hešovaciu tabuľku vo všetkých zúčastnených uzloch v sieti, kde každému uzlu je priradený jedinečný identifikátor na základe jeho IP adresy a čísla portu. Protokol DHT používa metriku vzdialenosti(distance metric) na určenie blízkosti dvoch uzlov v sieti. Metrika vzdialenosti sa vypočíta pomocou funkcie *XOR*. Čím bližšie sú dva uzly z hľadiska metriky vzdialenosti, tým je pravdepodobnejšie, že si budú môcť navzájom efektívne vymieňať údaje.

Smerovací protokol *Kademlia* pre DHT je základným protokolom používaným na komunikáciu medzi uzlami v sieti. Jeho správy zahŕňajú:

- *ping* - používa sa na kontrolu, či je uzol online a či odpovedá na požiadavky.
- *find_node* sa používa na získanie informácií o konkrétnom uzle v sieti.
- *get_peers* - sa používa na získanie zoznamu, ktorí zdieľajú konkrétny súbor.
- *notify_peer* - sa používa na informovanie siete, že *peer* zdieľa konkrétny súbor.
- *announce_peer* - používa sa na informovanie siete DHT o aktuálnom stave sťahovania a nahrávania. Po prijatí tejto správy uzol v DHT sieti ukladá IP adresu a číslo portu odosielateľa vo svojej smerovacej tabuľke, aby ho prípadne mohli ostatní nájsť a pripojiť sa k nemu.
- a ďalšie.

Všetky dotazy (queries) v protokole DHT sú iniciované uzlom želajúcim si nájsť zdroj/súbor v sieti. Nasleduje iterujúce šírenie dotazu po sieti, tak aby každý uzol prijímajúci dotaz vykonal príslušnú akciu na základe typu prijatej správy.

Napríklad, ak si uzol želá nájsť iných *peers* zdieľajúcich konkrétny súbor, pošle správu *get_peers* do siete DHT. Správa sa rozšíri po sieti a uzly s informáciami o požadovanom súbore odpovedia zoznamom *peers* zdieľajúcich tento súbor. Protokol DHT obsahuje aj špeciálny typ uzla nazývaný bootstrap (bootstrat node/ bootstrap server). Tento uzol je podobný *trackeru*, s tým že pomáha naviazať

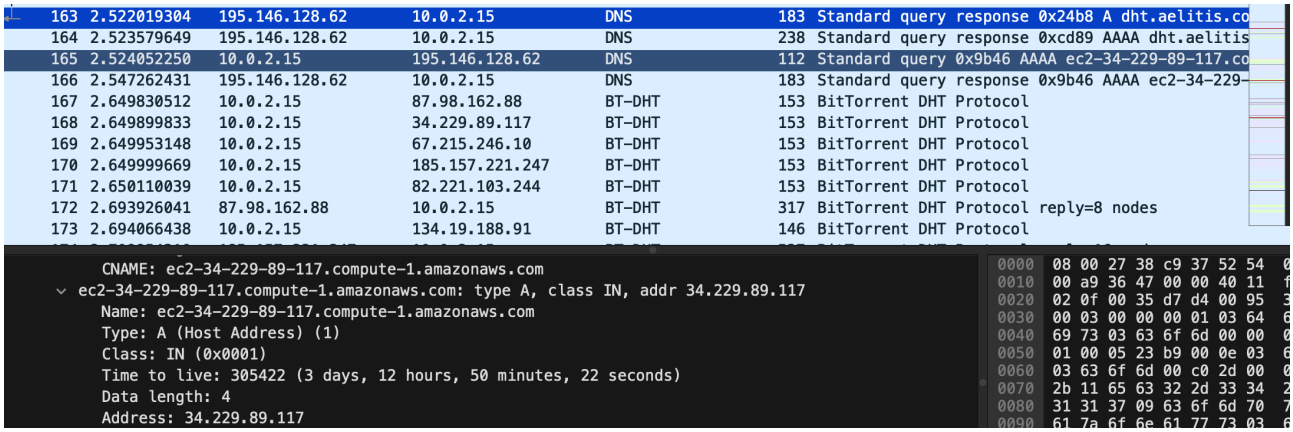
kontakt s inými uzlami. Bootstrap uzly sú ale dopredu známe uzly, ktoré sa používajú na iniciovanie kontaktu so sieťou DHT(vhodné keď nemám žiadne iné uzly uložené v lokálnej pamäti alebo inak). Keď sa nový uzol snaží pripojí do sieti, skontaktuje sa s jedným alebo viacerými bootstrap uzlami, aby získal zoznam ďalších aktívnych uzlov v sieti a tie zase na ďalšie a ďalšie, pokým nedosiahneme dostačujúceho počtu uzlov na rýchly prenos.

2.4 Analýza komunikácie protokolu BitTorrent

2.4.1 Prvé pripojenie do P2P siete

Keď sa chce klient pripojiť k roju BitTorrent, potrebuje získať zoznam partnerov, ktorí majú súbor, ktorý chce stiahnuť. Klient môže získať tento zoznam priamo z *tracker-a*, špecifikovaného v .torrent súbore alebo môže použiť protokol DHT na priame vyhľadanie partnerov. V protokole DHT je pre prvé spojenie do sieti potrebné pripojiť k bootstrap serveru(ako som spomínal vyššie 2.3.3). Klient odošle dotaz *find_node* na bootstrap server, ktorý odpovie zoznamom blízkych uzlov. Klient potom môže tieto uzly ďalej používať v protokole DHT naa lokalizáciu iných uzlov v sieti.

Z komunikácie je možné vypozorovať, že ešte pred tým než klient naviaže *Bittorrent hanshake* alebo si vymení správy z ostatnými uzlami v sieti, prebieha DNS komunikácia. V rámci tejto DNS komunikácie dochádza k prekladu doménových adries pomocou záznamu typu A. Prekladané doménové adresy sú obzvlášť zaujímavé, pretože po ich vyhľadaní si na internete som zistil, že ide práve o dobre známe bootstrap servery. Medzi také servery patrí napríklad “router.bittorrent.com”, “dht.aelitis.com” alebo “dht.transmissionbt.com”. Až po prijatí odpovede na tento DNS dotaz, komunikácia započne zasielať správy *get_peers* protokolu BT-DHT (pozri obrázok 1).

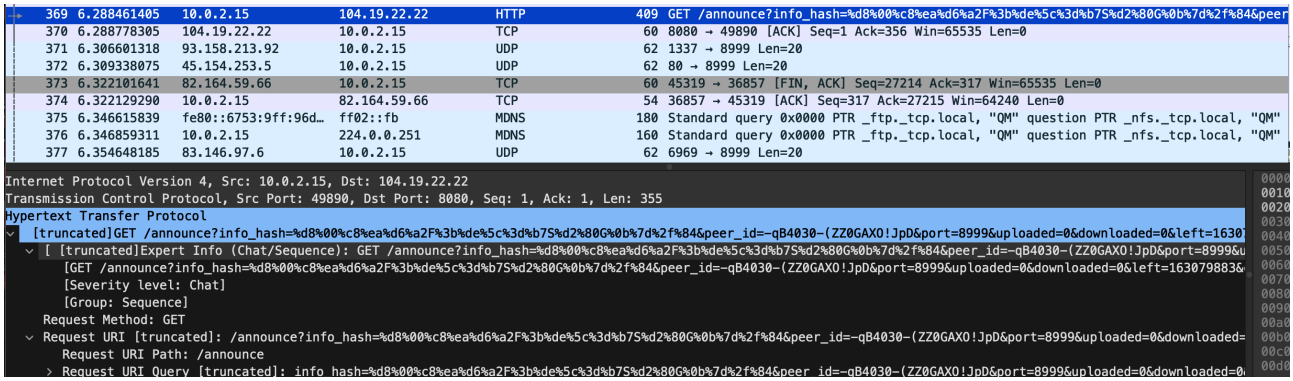


Obr. 1: DNS záznam s odpoveďou na preklad bootstrap serveru a následné naviazanie spojenia s danou adresou protokolom BT-DHT

2.4.2 Sťahovanie súborov

V tejto BT-DHT komunikácií klient opakovane odosiela na DHT uzly množstvo správ *get_peers* pre získanie ďalších uzlov v sieti. Následne som mal problém nájsť v komunikácií akékoľvek iné typy správ DHT protokolu. Len v nešifrovanej komunikácií som našiel *announce_peer* správy, ktoré naznačujú začiatok sťahovania alebo nahrávania súborov. Následne som pozoroval *HTTP GET /announce* správu s obsahom *info hash* a *peer id*, port a ďalšie (pozri Obrázok 2). V BitTorrent je info hash jedinečný identifikátor pre torrent súbor, ktorý sa vypočíta pomocou hašovacej funkcie SHA-1. Používa sa na identifikáciu súboru torrent a na zabezpečenie toho, aby všetci partneri v sieti zdieľali rovnakú verziu súboru. Tento info hash sa zasiela centrálnemu *trackeru* alebo bootstrap serveru práve

pomocou tejto správy, centrálny *tracker* následne odpovie s uzlami, ktoré daný *info hash* a teda aj súbor obsahujú [2]. Následne v zachytenej komunikácií prebiehala výmena súbora protokolom BitTorrent a správ *request piece*, popísanej v sekcií 2.3.2.



Obr. 2: HTTP /announce dotaz.

2.4.3 Peers keep-alive monitoring

Keď že sa klient pre stiahnutie súbora zvyčajne pripája k viacerým uzlom, klient používa *keep-alive* monitorovanie na udržiavanie zoznamu aktívnych partnerov a na odstránenie partnerov, ktorí už nie sú k dispozícii. Dotaz *get_peers* sa používa na získanie zoznamu partnerov, ktorí majú konkrétny súbor. Klient pravidelne posiela dotazy *get_peers* do DHT siete, aby aktualizoval svoj zoznam dostupných partnerov. Klient tiež používa správy *keep-alive* na monitorovanie dostupnosti svojich pripojených kolegov. Ak partner určitý čas neodpovedá, klient ho odstráni zo zoznamu aktívnych partnerov.

2.5 Metódy detekcie aktivít popísaných vyššie

2.5.1 Detekcia bootstrap serverov

Ako som popísal vyššie (viz. 2.4.1) na detekciu bootstrap serverov využijem fakt, že pred komunikáciou bittorrent a BT-DHT protokolu prebieha translácia doménových jmen, v ktorých sa ukázalo že sa ukazujú aj hľadané bootstrap servery. Teda v prípade že v zachytenej komunikácií BitTorrent protokolu, vyfiltrujem DNS provoz a budem hľadať doménové mená korešpondujúce známym bootstrap serverom. Dokážem tieto servery nájst a preložiť ich na ip adresu a port.

2.5.2 Detekcia uzlov/susedov

Ako som popísal vyššie (viz. 2.3.3) pre vyhľadávanie uzlov v sieti sa využíva protokol BT-DHT a jeho správa *get_peers* alebo *find_node*. Keď že z vypozorovanej komunikácie a jej analýze som zistil že správy *find_node* sa v komunikácií môjho klienta v BitTorrent protokole, vyhodnotil som že na získanie susedov v sieti použijem práve správu *get_peers*. Pre získanie paketov a dát BT-DHT som znova využil *tshark* a jeho filtračné možnosti. V rámci filtrov, ktoré tshark poskytuje [6]. Najužitočnejšími filtrami sa z hľadiska projektu ukázali byť bt-dht.ip, bt-dht.id a bt-dht.port. V rámci odpovedi na správu *get_peers* sa nachádza zoznam uzlov, ktorý dotazovaný uzol odoslal dotazovanému. Tieto filtre docielia to že pri správach BT-DHT komunikácie filtrujú obsah týchto paketov ako ip adresy, identifikátory, porty a "bencoded string". Týmto pádom viem získať zoznam uzlov, ktoré boli klientovi poskytnuté a údaje o týchto uzly ukladať. V prípade počítania spojení daného uzla, som využil filter bt-dht.bencoded.string, ktorý obsahuje *transaction id* identifikúce určité spojenie medzi uzlami. Následne už len stačilo porovnávať cieľovú a zdrojovú ip adresu s ip adresami

získaných uzlov. Ak sa našla zhoda tak sa porovnáva *transaction id* s uloženým zoznamom *transaction ids* pre daný uzol. Pokiaľ sa v zozname nenašiel, pridá sa a zvýší sa počet pripojení.

3 Implementácia

Nástroj bol implementovaný v jazyku Python, bližší popis štruktúry súborov projektu s návodom ako program spustiť je v priloženom *readme.txt*. Implementačnú časť popíšem v odsekoch identifikujúce jednotlivé argumenty, ktoré sme mali za úlohu implementovať.

3.1 Všeobecné info

Skript primárne pracuje s súbormi *.pcapng*, ktoré su v rámci programu prevádzané na *csv* reprezentáciu. Tento prevod je docielený s využitím nástroja *tshark*. Vygenerované súbory sú následne spracované pomocou *pandas* knižnice, pre jednoduchšiu prácu s dátami. Pri využití prepínača na pre vstup *csv* súbora, program nekontroluje správnosť formátu a preto sa doporučuje využívať len *csv* súbory vygenerované skriptom alebo práve *.pcapng* súbory. Primárne skript očakáva *.pcapng* súbory uložené v zložke `pcaps` a prevádzané *csv* súbory ukladá do `csvs`. Program sa skladá primárne z dvoch tried a jednej triedy reprezentujúcej susedov(peers/nodes). V triede `PreProcessor` primárne prebieha prevod *.pcapng* súborov do očakávanej podoby obsahujúcej informácie kľúčové pre splnenie podcieľov jednotlivých prepínačov. V triede `BTMonitor` následne sa preprocesované data spracovávajú a vyhodnocujú za účelom daného prepínača.

3.2 Init

Pre implementáciu tohto prepínača bola znalosť, ktorú som spomínal v sekcií 2.4.1 a 2.5.1. Na vyfiltrovanie DNS provozu som využil nástorj *tshark*. Keďže ma zaujímali dotazy typu A, doménové méno bootstrap servera vyfiltroval som si z provozu *qry_type* a *qry_name*. Na základe znalosti dobre známych bootstrap serverov som si vytvoril zoznam ich doménových jmen. Následne som už len hľadal taký vyfiltrovaný záznam, kde sa zhodovala doména s jedným s známych bootstrap serverov. Po nájdení takého som využil python funkciu `getaddrinfo()`, s ktorou som preložil doménové meno bootstrap servera na ip adresu a port, čo bolo požadované zo zadania. Výsledkom bol zoznam obsahujúci *tuple* portov a ip adries, ktoré sa vypisuje na stdout.

3.3 Peers

Tak ako som popísal vyššie v sekcií 2.3.3 a 2.5.2 využíval som pakety BT-DHT a filtrovanie nástrojom *tshark*. Po vyfiltrovaní do csv súbora a *pandas dataframu* som hľadal záznamy, ktoré obsahovali v stĺpci pre IP adries ID uzlov a portov pre odpoveď na správu *get_peers*. Pokiaľ som taký záznam našiel vytvoril som inštanciu z triedy `Neighbour`, ktorá mi reprezentovala jeden uzol/suseda v sieti. Následne po vytvorení tejto inštancie som hľadal v záznamoch cieľovú alebo zdrojovú ip adresu zhodujúcou sa s ip adresou jedného z susedov. Ak sa našla tak pomocou metódy `check_connection` dochádzalo k kontrole a počítaní počtu spojení. Využil sa na to zoznam *transaction ids*. Ktoré sa získavali z stĺpca vyfiltrovaného pomocou bt-dht.bencoded.string. V rámci tohoto stĺpca sa nachádzalo toto *transaction id* a to sa vkladalo na vstup tejto metódy spolu s ip adresou cieľovou a zdrojovou. V prípade nájdenia nového spojenia sa počet spojení zvýšil. Následne sa získaný zoznam susedov vypísal na výstup s instančnou metódou *__str__*.

4 Záver

Funkcionalita nástroja bola overovaná v prostredí wiresharku, na základe nezrovnalostí bol nástroj upravovaný a momentálna verzia dokáže detekovať bootstrap servery a vypísať zoznam uzlov participujucich v sieti. Vzhľadom na časovú vyťaženosť počas semestra som nedokázal naimplementovať všetky časti nástroja. Implementácia momentálne chýba podporu argumentov *-download* a *-rtable*.

Literatúra

[1] COHEN, B. Bep_0003.rst_post. [online]. January 2008. Dostupné z: https://www.bittorrent.org/beps/bep_0003.html.

[2] HARRISON, D. *Bep_0023.rst_post* [online]. May 2008. Dostupné z: https://www.bittorrent.org/beps/bep_0023.html.

[3] JOHNSEN, K. L. E. B. S. S. Peer-to-peer networking with BitTorrent. 2005. Dostupné z: https://web.cs.ucla.edu/classes/cs217/05BitTorrent.pdf.

[4] NORBERG A., L. A. Bep_0005.rst_post. [online]. January 2008. Dostupné z: https://www.bittorrent.org/beps/bep_0005.html.

[5] SLEDGEHAMMER999. *QBittorrent - A BitTorrent client in Qt* [online]. January 2015. Dostupné z: https://github.com/qbittorrent/qBittorrent.

[6] TSHARK. *Filtrovanie BT-DHT v prostredi tshark* [online]. April 2023. Dostupné z: https://www.wireshark.org/docs/dfref/b/bt-dht.html.