

# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

## Fakulta informačních technologií



## **Přenos dat, počítačové sítě a protokoly 2023**

### **Dokumentácia k projektu**

Monitorování a detekce provozu BitTorrent

# Obsah

<b>1</b>	<b>Úvod</b>	<b>2</b>
1.1	O realizácii projektu . . . . .	2
<b>2</b>	<b>Teória</b>	<b>2</b>
2.1	Peer-to-peer siete . . . . .	2
2.2	BitTorrent . . . . .	3
2.3	Architektúra BitTorrent . . . . .	3
2.3.1	Trackers . . . . .	3
2.3.2	Peer protokol . . . . .	4
2.3.3	DHT protokol . . . . .	4
2.4	Analýza komunikácie protokolu BitTorrent . . . . .	5
2.4.1	Prvé pripojenie do P2P siete . . . . .	5
2.4.2	Sťahovanie súborov . . . . .	5
2.4.3	Peers keep-alive monitoring . . . . .	6
2.5	Metódy detekcie aktivít popísaných vyššie . . . . .	6
2.5.1	Detekcia bootstrap serverov . . . . .	6
2.5.2	Detekcia uzlov/susedov . . . . .	6
<b>3</b>	<b>Implementácia</b>	<b>7</b>
3.1	Všeobecné info . . . . .	7
3.2	Init . . . . .	7
3.3	Peers . . . . .	7
<b>4</b>	<b>Záver</b>	<b>8</b>

# 1 Úvod

Táto práca vznikla ako projekt do predmetu "Prenos dat, počítačové siete a protokoly" a 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 realizácii projektu

Nástroj bol implementovaný v jazyku Python s využitím rôznych knižovní 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 naštudovať rôzne štúdie zaoberajúce 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 20.04 LTS* 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 využitie 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 nabera na popularitu, aj napriek jej kontroverziám 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 výpočtový výkon (computing power)
- efektívnejšie využitie výpočtového 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 zúčastnení (uzly) rovnocenní. Pre zaručenie funkcie služby a obsahu ponúkaného sieťou, je nutné, aby každý zo zúčastnených poskytoval svoje vlastné zdroje (výpočtový výkon, úložisko, kapacitu sieťového pripojenia atď.). Ostatní zúčastnení následne priamo prístupujú 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 preberá 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 súboru. 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 odosielať. 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 súboru 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 “peer” a “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 viacerých entít, vrátane:

- **Klasického web servera**
- **Statického súboru “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** - koncoví používatelia ich používajú na vyhľadávanie a sťahovanie torrent súborov.
- **Downloader koncového užívateľa** - koncoví používatelia ich používajú na sťahovanie súborov prostredníctvom protokolu BitTorrent.

### 2.3.1 Trackers

*Trackers* zohrávajú kľúčovú úlohu v uľahčení komunikácie medzi uzlami. Sú to centrálné 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 komunikácii 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. Prijímajúci uzol overuje odoslané údaje, v prípade nezrovnalosti sa spojenie ukončuje (zamedzí sa tak odoslaniu nekompletných alebo zlých súborov, alebo komunikácie s nesprávnym uzlom). Po nadviazaní *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 hešovacíu 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 žiadajúcim 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 (bootstrap node/ bootstrap server). Tento uzol je podobný *trackeru*, s tým že pomáha nadviazať

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 pripojiť do siete, skontaktuje sa s jedným alebo viacerými bootstrap uzlami, aby získal zoznam ďalších aktívnych uzlov v sieti a tie zase na ďalšie a ďalšie, pokiaľ 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 siete 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 na lokalizáciu iných uzlov v sieti.

Z komunikácie je možné vypožorovať, že ešte pred tým než klient naviaže *Bittorrent handshake* 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.transmissionbt.com” alebo “dht.aelitis.com”. Až po prijatí odpovede na tento DNS dotaz, komunikácia započne zasielať správy *get\_peers* protokolu BT-DHT (pozri obrázok 1).

163	2.522019304	195.146.128.62	10.0.2.15	DNS	183	Standard query response 0x24b8 A dht.aelitis.co
164	2.523579649	195.146.128.62	10.0.2.15	DNS	238	Standard query response 0xcd89 AAAA dht.aelitis
165	2.524052250	10.0.2.15	195.146.128.62	DNS	112	Standard query 0x9b46 AAAA ec2-34-229-89-117.co
166	2.547262431	195.146.128.62	10.0.2.15	DNS	183	Standard query response 0x9b46 AAAA ec2-34-229-
167	2.649830512	10.0.2.15	87.98.162.88	BT-DHT	153	BitTorrent DHT Protocol
168	2.649899833	10.0.2.15	34.229.89.117	BT-DHT	153	BitTorrent DHT Protocol
169	2.649953148	10.0.2.15	67.215.246.10	BT-DHT	153	BitTorrent DHT Protocol
170	2.649999669	10.0.2.15	185.157.221.247	BT-DHT	153	BitTorrent DHT Protocol
171	2.650110039	10.0.2.15	82.221.103.244	BT-DHT	153	BitTorrent DHT Protocol
172	2.693926041	87.98.162.88	10.0.2.15	BT-DHT	317	BitTorrent DHT Protocol reply=8 nodes
173	2.694066438	10.0.2.15	134.19.188.91	BT-DHT	146	BitTorrent DHT Protocol

  

CNAME: ec2-34-229-89-117.compute-1.amazonaws.com	0000	08 00 27 38 c9 37 52 54 0
ec2-34-229-89-117.compute-1.amazonaws.com: type A, class IN, addr 34.229.89.117	0010	00 a9 36 47 00 00 40 11 f
Name: ec2-34-229-89-117.compute-1.amazonaws.com	0020	02 0f 00 35 d7 d4 00 95 3
Type: A (Host Address) (1)	0030	00 03 00 00 00 01 03 64 6
Class: IN (0x0001)	0040	69 73 03 63 6f 6d 00 00 0
Time to live: 305422 (3 days, 12 hours, 50 minutes, 22 seconds)	0050	01 00 05 23 b9 00 0e 03 6
Data length: 4	0060	03 63 6f 6d 00 c0 2d 00 0
Address: 34.229.89.117	0070	2b 11 65 63 32 2d 33 34 2
	0080	31 31 37 09 63 6f 6d 70 7
	0090	61 7a 6f 6e 61 77 73 03 6

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 komunikácii klient opakovane odosiela na DHT uzly množstvo správ *get\_peers* pre získanie ďalších uzlov v sieti. Následne som mal problém nájsť v komunikácii akékoľvek iné typy správ DHT protokolu. Len v nešifrovanej komunikácii 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 komunikácii prebiehala výmena súbora protokolom BitTorrent a správ *request piece*, popísanej v sekcii 2.3.2.

369	6.288461405	10.0.2.15	104.19.22.22	HTTP	499	GET /announce?info_hash=%d8%00%c8%ea%d6%a2f%3b%de%5c%3d%b75%d2%80G%0b%7d%2f%846peer_id=-qB4030-(ZZ0GAX0!JpD&port=8999&uploaded=0&downloaded=0&left=1630798836
370	6.288778305	104.19.22.22	10.0.2.15	TCP	60	8080 → 49890 [ACK] Seq=1 Ack=356 Win=65535 Len=0
371	6.306601318	93.158.213.92	10.0.2.15	UDP	62	1337 → 8999 Len=20
372	6.309338075	45.154.253.5	10.0.2.15	UDP	62	80 → 8999 Len=20
373	6.322101641	82.164.59.66	10.0.2.15	TCP	60	45319 → 36857 [FIN, ACK] Seq=27214 Ack=317 Win=65535 Len=0
374	6.322129290	10.0.2.15	82.164.59.66	TCP	54	36857 → 45319 [ACK] Seq=317 Ack=27215 Win=64240 Len=0
375	6.346615839	fe80::6753:9ff:96d...	ff02::fb	MDNS	180	Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _nfs._tcp.local, "QM"
376	6.346859311	10.0.2.15	224.0.0.251	MDNS	160	Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _nfs._tcp.local, "QM"
377	6.354648185	83.146.97.6	10.0.2.15	UDP	62	6969 → 8999 Len=20

  

```

Internet Protocol Version 4, Src: 10.0.2.15, Dst: 104.19.22.22
Transmission Control Protocol, Src Port: 49890, Dst Port: 8080, Seq: 1, Ack: 1, Len: 355
Hypertext Transfer Protocol
  [truncated] GET /announce?info_hash=%d8%00%c8%ea%d6%a2f%3b%de%5c%3d%b75%d2%80G%0b%7d%2f%846peer_id=-qB4030-(ZZ0GAX0!JpD&port=8999&uploaded=0&downloaded=0&left=1630798836
    [truncated] Expert Info (Chat/Sequence): GET /announce?info_hash=%d8%00%c8%ea%d6%a2f%3b%de%5c%3d%b75%d2%80G%0b%7d%2f%846peer_id=-qB4030-(ZZ0GAX0!JpD&port=8999&uploaded=0&downloaded=0&left=1630798836
      [GET /announce?info_hash=%d8%00%c8%ea%d6%a2f%3b%de%5c%3d%b75%d2%80G%0b%7d%2f%846peer_id=-qB4030-(ZZ0GAX0!JpD&port=8999&uploaded=0&downloaded=0&left=1630798836]
      [Severity Level: Chat]
      [Group: Sequence]
      Request Method: GET
      Request URI [truncated]: /announce?info_hash=%d8%00%c8%ea%d6%a2f%3b%de%5c%3d%b75%d2%80G%0b%7d%2f%846peer_id=-qB4030-(ZZ0GAX0!JpD&port=8999&uploaded=0&downloaded=0&left=1630798836
      Request URI Path: /announce
      Request URI Query [truncated]: info_hash=%d8%00%c8%ea%d6%a2f%3b%de%5c%3d%b75%d2%80G%0b%7d%2f%846peer_id=-qB4030-(ZZ0GAX0!JpD&port=8999&uploaded=0&downloaded=0&left=1630798836
  
```

Obr. 2: HTTP /announce dotaz.

## 2.4.3 Peers keep-alive monitoring

Keďže 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 komunikácii BitTorrent protokolu, vyfiltrujem DNS provoz a budem hľadať doménové mená korešpondujúce známym bootstrap serverom. Dokážem tieto servery nájsť 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*. Keďže z vypozerovanej komunikácie a jej analýze som zistil že správa *find\_node* sa v komunikácii 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 filtermi sa z hľadiska projektu ukázali byť *bt-dht.ip*, *bt-dht.id* a *bt-dht.port*. V rámci odpovedí 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* identifikácie 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 zvýši 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 sekcii 2.4.1 a 2.5.1. Na vyfiltrovanie DNS provozu som využil nástroj *tshark*. Keďže ma zaujímali dotazy typu A, doménové meno 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 z 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 sekcii 2.3.3 a 2.5.2 využíval som pakety BT-DHT a filtrovanie nástrojom *tshark*. Po vyfiltrovaní do *csv* súbora a *pandas dataframe* 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 participujúcich 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](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](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](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 prostredí tshark* [online]. April 2023. Dostupné z:  
<https://www.wireshark.org/docs/dfref/b/bt-dht.html>.