VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

MONITOROVÁNÍ A DETEKCE PROVOZU BITTORRENT

MONITORING OF BITTORRENT TRAFFIC IN LAN

PROJEKT DO PREDMETU PDS
PROJECT FOR PDS COURSE

AUTOR PRÁCE **Bc. TIMOTEJ PONEK**
AUTHOR

BRNO 2023

Obsah

Kapitola 1

Popis sieťovej architektúry BitTorrent

Protokol BitTorrent bol pôvodne vyvíjaný za účelom jednoduchšieho zdieľania veľkých a práve populárnych súborov po sieti[3]. Architektúra klient-server umožňuje zdieľanie súboru iba nasledujúcim spôsobom: súbor sa uloží na server, a klienti si ho potom jednotlivo sťahujú. Týmto prenosom sa môže veľmi rýchlo preťažiť linka a server, ktorý musí obsluhovať veľké množstvo požiadavkov, sa stáva zahltený a nedostupný. BitTorrent tento problém rieši tak, že klient, ktorý daný populárny súbor sťahuje, zároveň nejakú časť prijatého súboru posiela ďalej po sieti ďalším klientom, ktorí majú o súbor záujem. Keďže súbor už nie je zdieľaný iba prostredníctvom serveru, ale aj klientov, zaťaženosť linky sa teoreticky znižuje (neberieme v potaz problém so zahlteným linky BitTorrent komunikáciou) a súbor je efektívnejšie a rýchlejšie distribuovaný medzi jednotlivých príjemcov.

1.1 BitTorrent podľa štandardu

Nasleduje podrobnejší popis správ, ktoré sa vymieňajú vrámci nadväzovania spojenia v sieti BitTorrent. Na tomto popise budú stáť pojmy použité v ďalších kapitolách.

Podľa štandardu[2] sa v sieti operujúcej na protokole BitTorrent nachádzajú 3 typy uzlov:

- `Tracker` - Ide o server, ktorý zaznamenáva informácie uzloch, ktoré sa vyskytujú v roji (*ang. swarm*). Klient od neho získa informácie o uzloch, ktoré vlastnia nejakú časť alebo celý zdieľaný súbor, a ďalej s týmito uzlami už kominikuje sám. Tracker teda neobsahuje žiadnu časť zdieľaného súboru, a nie je teda zapojený do procesu prenosu súboru.
- `Seed` - Uzol, ktorý má k dispozícií celý zdieľaný súbor, a teda ho už len distribuuje ostatným uzlom. Čím viac seedov v roji, tým vyššia bude prenosová rýchlosť súboru.
- `Downloader (Peer, Leech)` - Je to každý uzol, ktorý ešte nemá k dispozícií celý zdieľaný súbor, a teda súbor ešte sťahuje.

Protokol funguje na princípe „*tit for tat*“, čo v preklade znamená *čím viac dáš, tým viac dostaneš*. V princípe ide o to, že protokol BitTorrent je navrhnutý tak, aby odmeňoval uzly,ktoré už nejakú časť súboru majú a distribuujú ju ďalej, vyššou rýchlosťou sťahovania,

a uzly ktoré obmedzujú svoju rýchlosť distribúcie súboru (*ang. upload speed*) budú mať na oplátku znížený rýchlosť sťahovania.

Ak si chceme sťahnuť nejaký súbor prostredníctvom protokolu BitTorrent, najskôr si musíme sťiahnuť BitTorrent klienta. Následne si na intenete vyhľadáme **.torrent** (metainfo) súbor, ktorého štruktúra je následovná:

- `announce` - Položka ktorá obsahuje URL adresu trackera, od ktorého máme požadovať adresu uzlov so súbormi.
- `info` - Slovník s položkami, ktoré popisujú súbor/súbory, ktoré chceme stiahnuť.

Štruktúra slovníka `info`:

- `name` - Názov sťahovaného súboru (alebo zložky).
- `piece lenght` - Každý sťahovaný súbor je rozdelený do kúskov. Táto položka určuje veľkosť každého takéhoto kúsku (s výnimkou posledného, ktorý môže byť kratší).
- `pieces` - Položka, ktorá obsahuje SHA1 hash súboru alebo súborov. V prípade jedného súboru ide o reťazec o dĺžke 20 znakov, inak ide o reťazec $x * 20$ znakov, kde x reprezentuje počet súborov vrámci torrentu.
- `lenght` alebo `files` - Položka `lenght` sa vyskytuje v prípade, ak ide o jeden súbor, a reprezentuje dĺžku súboru v bajtoch. V opačnom prípade je prítomná položka `files`, ktorej obsahom je pole slovníkov s kľúčmi:

  `lenght` - Dĺžka súboru v bajtoch.

  `path` - List UTF-8 kódovaných reťazcov, ktoré reprezentujú jednotlivé podpriečinky. Posledná položka v liste zodpovedá názvu súboru.

Po stiahnutí **.torrent** súboru si ho otvoríme v BitTorrent klientovi. Ten kontaktuje tracker na URL adrese, uvedenej v `announce` položke s GET requestom, ktorý obsahuje nasledujúce položky:

- `info_hash` - SHA1 hash (o veľkosti 20 bajtov) súboru, ktorý chceme sťahovať.
- `peer_id` - Náhodne vygenerovaný reťazec o veľkosti 20 znakov, ktorý downloader používa ako svoje id.
- `ip` - Voliteľný parameter, nesúci IP adresu, na ktorej sa peer nachádza.
- `port` - Číslo portu, na ktorom chce peer počúvať.
- `uploaded` - Celkový počet uploadnutých bajtov.
- `downloaded` - Celkový počet stiahnutých bajtov.
- `left` - Počet bajtov, ktoré peer ešte potrebuje stiahnuť (pozn. nemôže byť spočítané ako $vekossboru - poetstiahnutchbajtov$, kvôli zaisteniu integrity sťahnutého súboru)
- `event` - Voľiteľná položka nadobudajúca hodnoty: **started** - posiela sa na začiatku sťahovania, **completed** - posiela sa po dokončení sťahovania súboru, **stopped** - označuje zastavenie sťahovania súboru peerom, **empty** - táto hodnota má rovnaký význam ako keby položka `event` nebola prítomná.

Tracker následne na GET request odpovie. Ak odpoveď obsahuje položku `failure reason`, ide o nejakú chybu, a táto položka obsahuje reťazec popisujúci prečo nastala chyba. V opačnom prípade obsahuje odpoveď dve položky: `interval` a `peers`.

- `interval` - Počet sekúnd, ktoré downloader bude čakať medzi jednotlivými požiadavkami.
- `peers` - List slovníkov s položkami `peer id`, `ip`, `port`, ktoré majú rovnaký význam ako bol uvedený v GET requeste vyššie.

Ďalej BitTorrent klient pokračuje komunikáciou s jednotlivými uzlami, ktoré dostal v položke `peers` od trackera. Od týchto uzlov dostáva jednotlivé časti súboru, ktoré si ďalej vymieňa s ostatnými peermi, ktorí majú o súbor záujem. Podľa štandardu existujú tieto typy správ:

- `0 - choke`
- `1 - unchoke`
- `2 - interested`
- `3 - not interested`
- `4 - have` - Ide o signalizáciu, že downloader práve dosťahoval časť súboru. Obsahom je číslo, reprezentujúce index stiahnutej časti súboru.
- `5 - bitfield` - Obsahom tejto položky je bitové pole, ktorého bity na jednotlivých indexoch sú nastavené na 1 ak downloader už má danú časť súboru, inak na 0. Je posielaný ako prvá správa.
- `6 - request` - Obsahuje položky **index**, **begin** a **lenght**, kde posledné menované je zvyčajne mocnina čísla 2, s výnimkou prípadu kedy ide o poslednú časť súboru.
- `7 - piece` - Obsahuje položky **index**, **begin** a **piece**, čo je požadovaná časť súboru.
- `8 - cancel` - Majú rovnaký obsah ako `request` správy. Sú používané ku koncu sťahovania, počas tzv. „end game mode“, kde sa peer snaží čo najrýchlejšie získať zvyšné časti súboru od všetkých peerov. Tieto správy sa používajú na to, aby sa zastavili zbytočné posielania častí súborov od peerov, ktoré downloader už dostal.

Správy `choke`, `unchoke`, `interested` a `not interested` nemajú žiadny obsah. Správy `choke` a `unchoke` sa používajú v algoritmoch, ktoré majú za úlohu prevenciu zahltenia linky BitTorrent komunikáciou.

1.2 DHT protokol

DHT (distributed hash table) protokol rieši problém s tým, že tracker je v pôvodnom protokole *single point of failure*. Je implementovaný nad UDP. V DHT terminológií je peer klient, počúvajúci na TCP porte a operujúci nad BitTorrent protokolom. Uzol (*ang. node*) je klient počúvajúci na UDP porte a operujúci nad DHT protokolom. Tento protokol poskytuje 4 typy správ[1]:

- `ping` - Používa sa na zisťovanie dostupnosti daných uzlov.
- `find_node` - Používa sa na získanie informácií o konkrétnom uzle. Uzol, ktorý obrdrží takúto správu odpovedá reťazcom s informáciami o K (zväčša 8) najbližších dobrých uzloch, ktoré má vo svojej smerovacej tabuľke.
- `get_peers` - Pomocou tejto správy si uzol pýta informácie o uzloch, ktoré mu môžu poskytnúť žiadaný súbor. Dotazované uzly na základe infohash-u súboru, ktorý je obsiahnutý v správe, buď vrátia list uzlov ktoré si medzi sebou daný súbor vymieňajú alebo vrátia K najbližších uzlov k infohash-u.
- `announce_peer` - Táto správa oznamuje ostatným uzlom, že peer sťahuje žiadaný súbor (ako bolo vysvetlené vyššie, **peer** - TCP spojenie na sťahovanie súborov, **uzol** - UDP spojenie na provoz DHT)

Vzdialenosť, ktorá bola vyššie niekoľkokrát spomenutá, sa počíta podobne ako v systéme Kademila, pomocou logického XOR-u medzi dvoma hodnotami. Čím menší výsledok XOR-u, tým menšia vzdialenosť.

Kapitola 2

Použitý BitTorrent klient

Za účelom vypracovania projektu som si vybral klienta `qBittorrent`, pretože jeho správanie by malo čo najviac odpovedať správaniu popísanom v BitTorrent štandarde. Ako testovacie súbory som zvolil súbory zo stránky *academictorrents.com*, konkrétne súbory sú v poznámke pod čiarou[1]. Na vygenerovanie bežného sieťového provozu, som použil **.torrent** súbory z tejto stránky, rovnako ako aj **Magnet linky**. Všimol som si, že pri použití **.torrent** súborov klient stiahol dotazované súbory priamo zo vzdialeného serveru, bez použita BitTorrent protokolu. BitTorrent bol použitý až pri ďalšom zdieľaní súboru ostaným peerom v sieti.

[1] súbory https://academictorrents.com/details/18cf38dccb2548a5213233c1c11e3ae4438d4ca3 a https://academictorrents.com/details/1d16994c70b7fff8bfe917f83c397b1193daee7f

Kapitola 3

Metódy detekcie BitTorrent komunikácie

Pre zachytávanie komunikácie som použil aplikáciu Wireshark. Táto aplikácia poskytuje vhodné display filtre, ako napr. `bittorrent`, pre zobrazenie iba bittorrent komunikácie (výmena jednotlivých častí súboru medzi peermi a pod.) alebo `bt-dht`, pre zobrazenie správ vygenerovaných vrámci tvorby dht tabuľky (správy ako `get_peers`, `find_node`). Rád by som vypichol jeden z rozdielov medzi verziami 3.6.13 (dostupná z hlavného repozitára aplikácií pre Ubuntu) a 4.0.5 (dostupná iba cez repozitár aplikácií Flatpak). Všimol som si že verzia 3.6.13 nemá display filter pre `bt-dht` (alebo má a vôbec bt-dht komunikáciu nedetekuje), a túto komunikáciu klasifikuje ako obyčajnú UDP komunikáciu. Preto som musel nainštalovať najnovšiu verziu Wiresharku, ktorá už správne klasifikuje `bt-dht` komunikáciu (túto skutočnosť som zistil pri testovacom odchyte paketov na systéme Windows).

Pre rýchle vyhľadavanie rôznych kľúčových slov v paketoch som si odchytenú komunikáciu vyexportoval do .json formátu, ktorý je mi dobre známy a teda sa mi s ním jednoducho pracovalo. Za účelom rýchleho vyhľadávania by ale bolo možné exportovať pcap do hociakého textového súboru. V získanom prevoze som následne analyzoval, ako sa klient pri prvom spustení dotazuje na ip adresy bootstrap serverov a získava od nich odpovede pre inicializáciu dht tabuľky. Klient sa vždy cez DNS služby dotazuje na ip adresy známych bootstrap serverov. Po získaní zodpovedajúcich odpovedí s ip adresou serverov ich klient následne kontaktuje s požiadavkom `get_peers`. V mnou zachytenom prevoze nedostal odpoveď iba od jedného bootstrap serveru. Uzly, ktoré získa v odpovediach od bootstrap serverov sú považované za bootstrap uzly.

Následne som hľadal vo vyexportovanom .json-e kľúčové slová ako *piece, have, get <announce>*, ktoré by naznačovali, že ide o BitTorrent komunikáciu.

Kapitola 4

Popis implementovanej aplikácie

Aplikácia bola implementovaná v jazyku Python, s využitím voľne dostupných knižníc. Menovite boli použité knižnice:

- `scapy` - Knižnica na tvorbu a zasielanie rôznych paketov. Bola použitá na načítanie **.pcap** súboru, a následné spracovanie jednotlivých paketov. Táto knižnica automaticky detekuje rôzne hlavičky paketu, a týmto spôsobom mi uľahčila spracovanie paketov.
- `bencodepy` - Umožňuje dekódovanie obsahu paketu, ktorý bol zakódovaný metódou *bencoding*.
- `dht-node` (pip package simple-dht-node) - Použitá na extrakciu `<compact-node-info>` z bt-dht odpovedí na dotaz `get_peers`. Extrahuje id, ip adresu a port dht uzlov obsiahnutých v odpovedi.

Aplikácia prechádza .pcap súbor jedntolivo, paket po pakete. Za účelom prepínača `-init` prechádza pakety s DNS hlavičkami, ktoré sa dotazujú na preklad doménových mien známych bootstrap uzlov. Známe bootstrap servery som vyhľadal na internete, a patria medzi ne tieto doménové adresy *router.utorrent.com, router.bittorrent.com", dht.transmissionbt.com, router.bitcomet.com, dht.aelitis.com, dht.libtorrent.org, dht.vuze.com.* Každý BitTorrent klient môže mať nejaký bootstrap server naviac, u `qBittorrent`u je to *dht.libtorrent.org.* Aplikácia si uloží ip adresy bootstrap serverov získané z DNS odpovedí, a následne hľadá v paketoch s UDP hlavičkami odpovede na dotaz `get_peers`. Knižnica scapy žiaľ nedetekuje bt-dht komunikáciu podobne ako to robí Wireshark. Preto aplikácia detekuje túto komunikáciu spôsobom, že sa najskôr skontroluje či, ide o UDP paket (takýto paket je podozrivý z bt-dht komunikácie), následne sa skontroluje, či zdrojová ip adresa paketu zodpovedá ip adrese jedného z bootstrap serverov (zaujímajú nás iba servery, ktoré nám odpovedali/sú online). Ak áno, pokračuje sa dekódovaním obsahu paketu, z čoho sa získa slovník, a extrahovaním informácií o uzloch z položky `nodes` tohto slovníka. Získané uzly sú prehlásené za bootstrap uzly. Takéto kontrolovanie paketu prebieha dovtedy, dokedy sa nedostane odpoveď od všetkých ip adries bootstrap jednotlivých bootstrap serverov. Potom sa paket už nekontroluje na prítomnosť bootstrap uzlov.

Prepínač `-peers` tiež vyžaduje prechádzanie paketov podobným spôsobom ako prepínač `-init`. Avšak narozdiel od `-init` prechádza všetky pakety podozrivé z bt-dht (nepozerá sa na ich ip adresu). Znovu sa tu najskôr dekóduje obsah paketu, ak bolo dekódovanie úspešné, pokračuje sa extrakciou dht uzlov, ktoré sa pridajú do slovníka *peerNodes* s dht uzlami (ak tam ešte nie sú). Tento slovník má ako kľúče id dht uzlov a hodnoty dvojice <informácie

o uzle, počet vymenených správ>. Počet vymenených správ s uzlom sa vždy inkrementuje, keď je v komunikácií detekovaná správa od uzlu s id, ktoré už je prítomné v *peerNodes*.

V aplikácií je zvlášť oddelené spracovanie ipv4 a ipv6 paketov, ale okrem iného formátu ip adresy a extrakcie <compact-nodes-info> z odpovede na dotaz `get_peers` sa v ničom inom nelíši, štruktúra obsahu paketu je rovnaká.

Kapitola 5

Testovanie aplikácie a diskusia výsledkov

Aplikácia bola implementovaná a testovaná na priloženom súbore *com1.pcap*, ktorý obsahuje komunikáciu odchytenú počas prvého spustenia klienta `qBittorrent`, a sťahovania malého súboru textového súboru *Stability of Randomized Learning Algorithms*. Výsledok pri spustení s prepínačom `-init` (obrázok 5.1) a `-peers` (obrázok 5.2) je možné vidieť na priložených obrázkoch. Aplikácia bola taktiež testovaná na dvoch ďalších odchytených BitTorrent komunikáciach (súbor *com2.pcap, com2_magnet,pcap*), zo sťahovania väčšieho zip súboru.

```
------------INIT-------------
ip: 111.92.119.254; port: 5851
ip: 2001:4451:945:d100:88d8:d29b:1328:d626; port: 47065
ip: 2804:d59:9170:be00:d86d:25ff:fe4b:d8ba; port: 60186
ip: 2804:d51:461f:2e00:c0af:6161:45d7:d20f; port: 40532
ip: 95.144.141.117; port: 44599
ip: 158.88.32.19; port: 2360
ip: 181.41.127.23; port: 50208
ip: 139.216.24.108; port: 11593
ip: 68.203.51.172; port: 41156
ip: 177.192.124.253; port: 52401
ip: 187.85.16.66; port: 1032
ip: 49.150.65.119; port: 3171
ip: 88.147.153.133; port: 4868
ip: 223.239.95.32; port: 51990
ip: 112.79.125.208; port: 36245
ip: 42.105.181.86; port: 32843
ip: 223.185.126.0; port: 36661
ip: 106.198.142.20; port: 24335
ip: 49.216.232.55; port: 46200
ip: 106.206.69.174; port: 11089
ip: 2604:3d09:676:9c00:8aa3:26e1:d255:2a6d; port: 17446
ip: 172.58.208.0; port: 58848
ip: 200.125.89.82; port: 53143
ip: 189.239.41.236; port: 54060
ip: 188.6.4.169; port: 62339
ip: 117.222.199.234; port: 1545
ip: 85.66.243.51; port: 51413
ip: 180.92.29.215; port: 10306
ip: 151.234.36.233; port: 54898
ip: 119.179.249.51; port: 5060
ip: 188.163.45.5; port: 6767
ip: 142.54.207.201; port: 35913
ip: 157.48.251.134; port: 59631
ip: 218.156.107.237; port: 7720
ip: 2.101.159.177; port: 23830
ip: 136.158.40.240; port: 61175
ip: 172.58.229.210; port: 64467
```

Obr. 5.1: Výsledok pri spustení s prepínačom `-init`

Prepínač `-init` vypisuje ip adresu a port dht uzlov získaných skrz dotazy `get_peers` na bootstrap servery tak ako to vyžaduje zadanie

```
------------PEERS------------
ip: 10.0.0.31; port: 8999; id: 5ccc052c5ebaa0f32fe08b7c8cffa22602125f22; nmb of messages exchanded: 7
ip: 109.247.154.151; port: 49681; id: 0c079d40dba8c2337273f681b2e66d54c370adb7; nmb of messages exchanded: 5
ip: 185.16.39.228; port: 48271; id: 18cf38dccb2548a521323aae1863fb4b932715c3; nmb of messages exchanded: 3
ip: 202.61.226.152; port: 6883; id: 18cf38dccb2548a521323f45d1c1e067df80d5f3; nmb of messages exchanded: 3
ip: 195.3.220.58; port: 6881; id: 18cf38dccb2548a52132ae018b8cb02dbf088c6c; nmb of messages exchanded: 3
ip: 35.167.186.212; port: 6881; id: 18cf38dccb2548a521328a95232b9b9b0d324ab3; nmb of messages exchanded: 3
ip: 87.142.11.62; port: 50357; id: 5d6bce594fd9df09a4e935e28dda9a18366051ba; nmb of messages exchanded: 2
ip: 87.98.162.88; port: 6881; id: 3c00727348b3b8ed70baa1e1411b3869d8481321; nmb of messages exchanded: 1
ip: 95.144.141.117; port: 44599; id: 80333dcb5306cb4d22d1c51d82046b1b7c5dbfe7; nmb of messages exchanded: 1
ip: 181.41.127.23; port: 50208; id: 6b91f3808b1913eaf93c0cb6c5c187acdcb05578; nmb of messages exchanded: 1
ip: 139.216.24.108; port: 11593; id: 4b65808957112417e9e060a6b64af1c425e89cc1; nmb of messages exchanded: 1
ip: 68.203.51.172; port: 41156; id: 2f9af1361b1fbbd116fc8936be1e52718a62bb4f; nmb of messages exchanded: 1
ip: 88.147.153.133; port: 4868; id: f69dc4f44ce9ba8c266a931b176b01c2c0e7ac3c; nmb of messages exchanded: 1
ip: 112.79.125.208; port: 36245; id: 8042cb45e1f569c2be7a47f2db31f46082671d69; nmb of messages exchanded: 1
ip: 106.198.142.20; port: 24335; id: d28b4df4791e32799bff9675ba82d56a01091c61; nmb of messages exchanded: 1
ip: 49.216.232.55; port: 46200; id: 2e400949a8c9177664ccaf5373e990c4da547d05; nmb of messages exchanded: 1
ip: 185.157.221.247; port: 25401; id: 1c11e01be8e78d765a2e63339fc99a66320db754; nmb of messages exchanded: 1
ip: 219.145.34.178; port: 9292; id: 585b20584a1773039878af7513dc46a6dbb27d1b; nmb of messages exchanded: 1
ip: 71.220.177.133; port: 61464; id: 46027fbc856ca42461de757b3af58c6049d3129f; nmb of messages exchanded: 1
ip: 85.140.12.213; port: 26288; id: 54b1e001dfc67b978dbe89dcd723e588b6e712dc; nmb of messages exchanded: 1
ip: 121.172.178.61; port: 40563; id: 5ee880d47db485841fcb07d485814167da3f55bd; nmb of messages exchanded: 1
ip: 120.233.127.236; port: 6891; id: 59b64255e9556e5501aef97a5e84a09d2651ccfd; nmb of messages exchanded: 1
ip: 176.36.148.189; port: 51413; id: 551242abebd168aaee70621b438c7751e2e43ee3; nmb of messages exchanded: 1
ip: 216.184.43.240; port: 14485; id: 5520911dd8baef6d72eca82260682ff642710a28; nmb of messages exchanded: 1
ip: 218.147.196.190; port: 33193; id: 4dc907996314eb1929d6884ab396166a40a8c288; nmb of messages exchanded: 1
ip: 185.4.78.225; port: 61489; id: 4d82c11dde4ed12af94381f93729a297004247aa; nmb of messages exchanded: 1
ip: 202.190.12.63; port: 47094; id: 5a88e4292153a17b528ec7c57d2bd23aa081d04e; nmb of messages exchanded: 1
ip: 106.195.93.97; port: 3957; id: 5f59f325d4be9f73710c477f9b601f5400136a3f; nmb of messages exchanded: 1
ip: 188.163.4.130; port: 49271; id: 582bb7b43ea62cbe22b3db099935b099ca9e0919; nmb of messages exchanded: 1
ip: 14.43.184.109; port: 41043; id: 5a48d32ba377fb87b12eb1955924937480670c9b; nmb of messages exchanded: 1
ip: 31.134.188.116; port: 1088; id: 5ba39d7ba4f1a1f7356ce06b86aeba3b3d53affd; nmb of messages exchanded: 1
```

Obr. 5.2: Výsledok pri spustení s prepínačom `-peers`

Prepínač `-peers` vypisuje ip adresu, port, a id dht uzlov a počet vymenených správ s daným uzlom (to sa zisťuje podľa id uzla, ktorý je zapísaný v obsahu paketu s odpoveďou). Keďže z odchytenej komunikácie nie je možné jasne určiť ktorá ip adresa je ip adresa klienta, je aj táto adresa vypísaná vrámci prepínača `-peers`. Teoreticky je možné za ip adresu klienta považovať adresu s najväčším počtom vymenených správ. Ip adresu klienta by bolo možné zistiť z paketov http služby, ktoré obsahujú políčko `user_agent` a zisťovať, či jeho hodnota obsahuje názov nami použitého BitTorrent klienta (v mojom prípade qBittorrent). Toto by ale nebolo obecné, môže sa stať že táto hodnota je zámerne vyplnená niečím nepravdivým alebo môžem byť v odchytenej komunikácií použitý klient ktorého nepoznám.

Kapitola 6

Záver

Nebol plne implementovaný prepínač `-download` zo zadania, z dôvodu časovej tiesne. Vytvorenú aplikáciu by bolo vhodné otestovať na viacerých .pcap súboroch a odladiť prípadné chyby a nedostatky vo výstupe. Aplikácia umožňuje spracovávať .pcap súbory, čo je rozhodne výhodou oproti druhej možnosti, ktorá bola povolená, a teda nie je potrebné predspracovanie .pcap súboru pomocou ďalšieho skriptu.

Výsledky je možné použiť pri ďalšom skúmaní BitTorrent komunikácie. Ďalšie vylepšenia aplikácie by mohli spočívať v použití knižnice `dpkt` namiesto `scapy`, ktorá umožňuje rýchlejšie načítať pakety z .pcap súborov, avšak neposkytuje až tak príjemné API pre spracovanie paketov.

Literatúra

[1] Andrew Loewenstern, A. N. *DHT Protocol* [online]. 2020 [cit. 2023-04-10]. Dostupné z: https://www.bittorrent.org/beps/bep_0005.html.

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

[3] Garakh, I. *What is BitTorrent?* [online]. 2022 [cit. 2023-04-10]. Dostupné z: https://blog.passwork.pro/what-is-bittorrent/.