VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SLEDOVÁNÍ BITTORENTOVÉHO PROVOZU V LAN

MONITORING OF BITTORRENT TRAFFIC IN LAN

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE **Bc. JAKUB KOMÁREK**
AUTHOR

BRNO 2023

Abstrakt

Tato práce se věnuje architekturám Bittorent sítí. Konkrétně se věnuje analýze Bittorent komunikace a tvorbě detekčního nástroje pro Bittorent provoz.

Citace

KOMÁREK, Jakub. *Sledování BitTorentového provozu v LAN*. Brno, 2023. Semestrální projekt. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce
,

Obsah

Kapitola 1

Úvod

Tato práce se věnuje architekturám BitTorent sítí. Konkrétně se věnuje analýze BitTorent komunikace a tvorbě detekčního nástroje pro BitTorent provoz.

V rámci analýzy bude vysvětlen hlavní princip fungování BitTorent sítí. Následovat budou experimenty s BitTorent klientem v rámci kterého bude funkce této technologie vysvětlena více podrobně. Tyto experimenty jsou zaměřeny na standardní nemodifikovaný průběh stahování souboru v programu qBitTorent. Začátek analýzy tvoří popis inicializační části klienta po nové instalaci. Poté následuje popis vyhledání a stažení zdroje.

V další kapitole následuje implementační část práce, v rámci které jsou vysvětleny využité principy z předchozí kapitoly pro účely analýzy provozu. Poté již následuje samotný popis realizace, který obsahuje výčet použitých technologií, návrhový vzor a samotný popis jednotlivých funkcionalit nástroje. Kapitola je zakončena testováním programu.

Konec práce tvoří diskuze nad výsledky implementovaného softwaru a závěr.

1.1 Popis BitTorent sítě

Síť BitTorent je tvořena z uzlů. Každý uzel má svůj vlastní identifikátor a může stahovat či nahrávat soubory. Komunikace v této sítí je zařízena pomocí dvojice protokolů BT-DHT a BITTORENT. BT-DHT slouží pro vyhledávání zdroje a BITTORENT slouží pro samotné stahování. BitTorent architektura pracuje nestrukturovaně, nicméně pro správné fungování obvykle využívá dva specifické typy uzlů - Bootstrap a Tracker.

Bootstap uzly jsou obvykle ve výchozí konfiguraci zapsány v BitTorent klientu, buď ve formě IP adresy a portu nebo ve formě doménového jména. Tyto uzly slouží pro první vstup do sítě a poskytují směrovací údaje pro další komunikaci například s Trackery. Tracker je uzel poskytující informace o uzlech, které mohou obsahovat požadovaný soubor.

Pokud chce uživatel stáhnout některý soubor v BitTorent síti, potřebuje údaje o tomto souboru, aby ho mohl dohledat a stáhnout. Tyto informace se nachází v odkazu na soubor. Tento soubor otevře ve svém Torrentovém klientu, poté započne proces získávání souboru.

Ten se obecně dělí na vyhledávací a stahovací fázi. Tyto fáze mohou pracovat zároveň. Vyhledávací fáze má za úkol lokalizovat IP adresu s portem, kde se soubor nebo jeho část může nalézat. Poté může započnout stahovací fáze v rámci, které z různých zdrojů stahujeme části daného souboru. Při této fázy zároveň můžeme potencionálně odesílat části souboru dalším zájemcům.

Dělení souboru je provedeno za pomocí bloků fixní délky (pro každý soubor může být jiná). Tyto bloky si pak jednotliví účastníci Torrent sítě na požádání vyměňují [4].

Kapitola 2

Experimenty s BitTorrent klientem

Experimenty byly prováděny za pomocí `qBitTorrent` klienta a nástroje `Wireshark`. Inicializační část analýzy probíhala ze zachycené komunikace po prvním spuštění `qBitTorrent` klienta. V rámci analýzy jsem provedl analýzu při stahování i instalaci `qBitTorrent` balíku, kde žádná užitečná komunikace neprobíhala. Stahovací část analýzy jsem provedl ze zachycené komunikace, při stahování obrazu systému Linux a jiných OpenSource softwarů. Získané poznatky byly ověřeny pomocí literatury [1, 2, 3, 4].

2.1 Inicializace klienta

Inicializace je zahájena rezolucí `DNS` několika doménových jmen - např. dht.libtorrent.org., router.bittorrent.com., atd. Na těchto adresách sídlí `BootStrap` uzly. Z názvů doménových jmen lze usuzovat, že `BootStrap` servery obvykle ve svém doménovém jméně obsahují klíčová slova `"Torrent"`, "dht"a "router". Po získaní `IP` adres `Bootstrap` serverů se provedla s některými takto získanými `Bootstrap` servery komunikace pomocí `BT-DHT` protokolu.

Při této komunikaci se klient snažil rozšířit svou routovací tabulku, pravděpodobně o další `BootStrap` uzly. K tomu používal zprávy `GetPeers`. Po kontaktování `BootStap` serverů obdržel odpověď v podobě sady uzlů. Tato sada obsahovala dvojice ID uzlu a `IP` adresa s portem služby. Obvykle tyto sady obsahovaly 8 nebo 16 uzlů. Stejný postup provedl klient cyklicky nad některými novými uzly. Po určitém počtu opakování se stav klienta ustálil a komunikace částečně ustanula. Klient udržoval spojení s některými nově objevenými uzly za pomocí `keepalive` zpráv.

2.2 Stažení souboru

Odkazy na soubor

Metadata o souboru lze získat dvěma způsoby. Prvním způsobem je využití `Torrent` odkazovacího souboru (koncovka .torrent). V tomto souboru jsou přítomny metainformace pro potřeby stažení souboru. Nachází se v něm `InfoHash` - hash souboru sloužící k adresaci zdroje, adresa `Tracker` serveru/ů, seznam názvů souborů, souhrnný název, celková velikost, komentář, datum vytvoření a název autora. Viz. obrázek 2.1.

Druhým způsobem je využití takzvaného magnetu, kdy na webu, kde se odkaz na soubor nachází klikneme na tlačítko, které zařídí přenos potřebných informací do `Torrent` klienta. Obvykle tyto data nejsou tak obsáhlá jako u předchozího souboru a obsahují nutné minimum informací, obvykle `InfoHash` a adresy některých `Tracker` serverů.

Vyhledávání peerů

Apriorními informacemi, které klient využije při stahování jsou: `InfoHash`, seznam `Trackerů`, celková velikost souboru a velikost Chunku. Klient započne stahování kontaktováním `Tracker` serverů, které nalezne buď při inicializaci nebo z odkazovacího souboru.

Tyto uzly kontaktuje spolu se svým `InfoHashem` pomocí zprávy `GetPeers`. Odpovědí na tento požadavek je buďto seznam uzlů (`nodes`) nebo seznam peerů (`Peers`). Výjimkou nejsou ani odpovědi obsahující obojí. V případě varianty `nodes` zpráva obsahuje seznam dvojic ID uzlu a jejich adresu `IP` s portem. V případě druhé varianty `Peers` jsou v seznamu pouze `IP` adresy s porty.

`Nodes` jsou vráceny pokud daný uzel nemá informace o daném InfoHashi a vrací tudíž přibližnou lokaci zdroje. `Peers` jsou vráceny pokud server zná lokaci těchto zdrojů.

Dá se tedy říci, že stahování se skládá z vyhledávací a stahovací části. Ve vyhledávací části pomocí uzlů hledá adresy peerů, ze kterých by mohl soubor stahovat. Toto vyhledávání je řešeno za pomocí struktury připomínající binární strom a proto je možné se postupným doptáváním okolních uzlů docílit lokalizace zdroje.

Po nashromáždění dostatečné množství peerů si klient vybere nejvhodnější uzly, pravděpodobně podle délky odezvy či propustnosti a započne stahovací fáze. Neznamená to však, že v průběhu stahování nejsou dohledány další `Peer` uzly.

Průběh stahování

Do teď byl používán pouze protokol `BT-DHT` nad `UDP` vrstvou. Pro stahování je použit protokol `BITTORENT` standartě nad `TCP` vrstvou, klient také používal odlehčenou variantu protokolu nad `UDP` vrstvou.

Během komunikace si účastníci sítě vyměňují svoje `Chunky` - kousky souboru. `Chunk` se skládá z několika kousků (`Pieces`). Aby bylo jasné, co daný kousek obsahuje, vždy se posílá spolu s daty jeho délka, offset a pozice kousku. Z toho lze přesně odvodit pozici na které kousek leží.

Komunikace s `Peer` uzlem započne zprávou `Handshake`, ve které se nachází hash souboru (`InfoHash`) a `NodeId`. `Handshake` je obvykle potvrzen klientovy. V případě, že klient nemá žádné Chunky stahovaného souboru (typicky na začátku stahování), zašle zprávu `Bitfield` ve které zasílá bitovou mapu všech svých Chunků (prvně obsahuje samé nuly). `Peer` mu poté zašle libovolné Chunky.

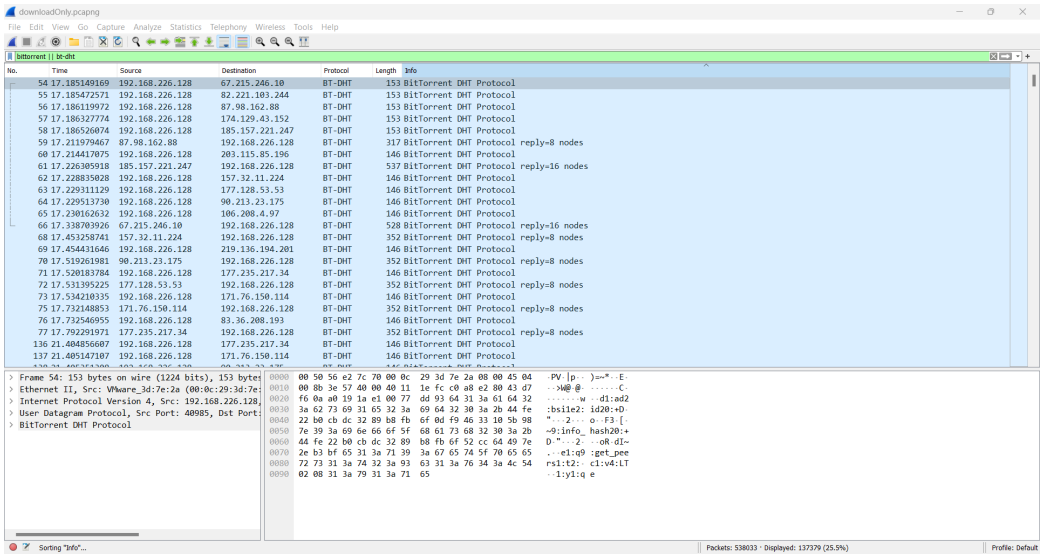
Pokud již některé Chunky má, zasílá `Peer` uzlu žádosti o konkrétní kousky pomocí zprávy `Reguest Piece`, která obsahuje index Chunku, offset kousku a délku kousku. Pokud `Peer` má daný kousek, zašle mu ho pomocí zprávy `Piece`. Pokud je kousek delší než je maximální délka zprávy, je kousek doručen v následujících zprávách typu `Extended`.

Klient a `Peer` uzly si občasně vyměňují zprávy `Bitfield`, aby se navzájem informovali o datech kterými disponují. Klient může projevit zájem o konkrétní blok dat zprávou `Interested`. Na což `Peer` reaguje zprávou `Unchoke` a často zašle klientovy nový port na kterém započne výměna kousků. Poté již probíhá výměna pomocí zpráv `Reguest Piece` a `Piece`. V případě že `Peer` nedisponuje hardwarovou kapacitou pro obsloužení klienta zašle zprávu `Choke` čímž přeruší komunikaci.

Tímto způsobem klient provádí stahování z několika peerů zároveň. Je nutné podotknout, že jeden požadavek nemusí nutně znamenat jednu odeslanou zprávu. Jedna zaslaná zpráva může obsahovat vícero požadavků různých typů - například se zpráva může sestávat z vícera požadavků request pieces a přiložené bitmapy.

```
        name: Smrtonosná pasca
    filename: [SkT]Smrtonosna_past___Die_Hard_(komplet,1080p,CZ).torrent
     comment:
        date: 23.02.2021 01:20:46 PM (1614104446)
  created_by: uTorrent/3000
       files: (5)
              1: Smrtonosná pasca I (r.1988 - 1920x812).avi
              2: Smrtonosná pasca IV (2007 - 1920x800).avi
              3: Smrtonosná pasca V - Opět v akci  (r.2013 - 1920x1080).mkv
              4: Smrtonosná pasca III (r.1995 - 1920x816).mkv
              5: Smrtonosná pasca II (r.1990 - 1920x812).mkv
        size: 10318519246
    announce: http://sktorrent.eu/torrent/announce.php?pid=95ff38b5e44395a619f52365d1ea027f
announce_list:
              - https://announce1.sktorrent.eu/torrent/announce.php?pid=95ff38b5e44395a619f52365d1ea027f
              - udp://tracker.opentrackr.org:1337
              - udp://ipv4announce.sktorrent.eu:6969/announce
   info_hash: e00ceb61175110ae02f90d14801e5fd5457790e1
```

Obrázek 2.1: Ukázka dat nacházejících se v `Torrent` souboru za pomocí nástroje https://www.tools4noobs.com/online_tools/torrent_decode/



Obrázek 2.2: Analýza nástrojem Wireshak

```
root@Power-Machine:/mnt/c/workSpace/PDS# make run ARGS="-pcap logs/downloadOnly.pcapng  -init -packetLim 25000"
venv/bin/python3 main.py -pcap logs/downloadOnly.pcapng  -init -packetLim 25000
Bootstraps servers:
185.157.221.247:25401 (dht.libtorrent.org.)
67.215.246.10:6881 (router.bittorrent.com.)
82.221.103.244:6881 (router.utorrent.com.)
185.125.190.59:6926 (torrent.ubuntu.com.)
root@Power-Machine:/mnt/c/workSpace/PDS# make run ARGS="-pcap logs/downloadOnly.pcapng  -peers -packetLim 40000"
venv/bin/python3 main.py -pcap logs/downloadOnly.pcapng  -peers -packetLim 40000
Active peers:
   0#   185.125.190.59: 6926, InfoHash: 2d7142343532302d5772323304937794a626c2146 Dwn/Up:       0/   81920 B, Ratio: 1.000, Pieces Send/recv    0/   5, Connections:  0
   1#  192.168.226.128:57799, InfoHash: 543033492d2d3031302e6863576a6c6743563567 Dwn/Up:  540672/       0 B, Ratio: 0.000, Pieces Send/recv   33/   0, Connections:  5
   2#    89.134.88.185: 9916, InfoHash: 2d7142343532302d6d73314a584646304654673 3 Dwn/Up:       0/  344064 B, Ratio: 1.000, Pieces Send/recv    0/  21, Connections:  1
   3#      5.79.77.54:55686, InfoHash: 2d7142343532302d28704349737462307550283 1 Dwn/Up:       0/   98304 B, Ratio: 1.000, Pieces Send/recv    0/   6, Connections:  1
   4#    66.63.167.116:60243, InfoHash: 2d7142343532302d456d457a722e37312d493348 Dwn/Up:       0/   16384 B, Ratio: 1.000, Pieces Send/recv    0/   1, Connections:  1
   5#    165.22.47.211:63793, InfoHash: 2d7142343532302d2d4363422a677a3770364c4c Dwn/Up:       0/       0 B, Ratio: 1.000, Pieces Send/recv    0/   0, Connections:  1
   6#  123.129.130.103:10086, InfoHash: 2d7142343532302d4d5845686937725643777e32 Dwn/Up:       0/       0 B, Ratio: 1.000, Pieces Send/recv    0/   0, Connections:  1
root@Power-Machine:/mnt/c/workSpace/PDS# make run ARGS="-pcap logs/downloadOnly.pcapng  -download -packetLim 40000"
venv/bin/python3 main.py -pcap logs/downloadOnly.pcapng  -download -packetLim 40000
Files:
Info_hash 99c82bb73505a3c0b453f9fa0e881d6e5a32a0c1
Totaly sended pieces 33 ,Totaly sended 540672 B, Maximal Piece Size 16384 B
Download begined by:
    1#  192.168.226.128:57799, InfoHash: 543033492d2d3031302e6863576a6c6743563567 Dwn/Up:  540672/       0 B, Ratio: 0.000, Pieces Send/recv   33/   0, Connections:  5
Peers:
   0#   185.125.190.59: 6926, InfoHash: 2d7142343532302d5772323304937794a626c2146 Dwn/Up:       0/   81920 B, Ratio: 1.000, Pieces Send/recv    0/   5, Connections:  0
   2#    89.134.88.185: 9916, InfoHash: 2d7142343532302d6d73314a584646304654673 3 Dwn/Up:       0/  344064 B, Ratio: 1.000, Pieces Send/recv    0/  21, Connections:  1
   3#      5.79.77.54:55686, InfoHash: 2d7142343532302d28704349737462307550283 1 Dwn/Up:       0/   98304 B, Ratio: 1.000, Pieces Send/recv    0/   6, Connections:  1
   4#    66.63.167.116:60243, InfoHash: 2d7142343532302d456d457a722e37312d493348 Dwn/Up:       0/   16384 B, Ratio: 1.000, Pieces Send/recv    0/   1, Connections:  1
   5#    165.22.47.211:63793, InfoHash: 2d7142343532302d2d4363422a677a3770364c4c Dwn/Up:       0/       0 B, Ratio: 1.000, Pieces Send/recv    0/   0, Connections:  1
   6#  123.129.130.103:10086, InfoHash: 2d7142343532302d4d5845686937725643777e32 Dwn/Up:       0/       0 B, Ratio: 1.000, Pieces Send/recv    0/   0, Connections:  1
```

Obrázek 2.3: Ukázka výstupu implementovaného nástroje programu.

Kapitola 3

Implementační část

3.1 Použité metody a návrh

Program jsem navrhl jako průtokový parser jednotlivých paketů - pakety jsou přiváděny na vstup hlavní parsovací třídy, kde ovlivňují vnitřní stav programu. Po zpracování všech paketů jsou vypsány výsledky. Program byl koncipován tak, aby byl odolný proti nestandardním stavům, pokud v rámci parsování paketu dojde k chybě, program se zotaví přechodem na další paket.

Pro analýzu `Bootstrap` serverů byly použity `DNS` dotazy a následná analýza komunikace po obdržení požadovaných adres.

Analýza souborů a peerů byla provedena s využitím `BitTorent` protokolu, konkrétně pomocí zpráv `Handshake`, `Bitmap` a `Piece`.

3.2 Implementace a použité nástroje

Nástroj je napsán v Pythonu verze 3.8 (měl by být kompatibilní s vyššími verzemi) s použitím knihovny `Scapy`. Inicializaci do virtuálního prostředí `Venv` provedete příkazem `make`. Pokud nechcete využít `Venv`, doinstalujte knihovnu `Scapy` a projekt by měl být spustitelný. Spuštění provedete pomocí příkazu make run ARGS=<vaše argumenty> v případě užití prostředí venv nebo pomocí příkazu python3 main.py <vaše argumenty>. Program umí pracovat pouze s `PCAP` soubory. Pro větší optimalizaci předfiltrujte `PCAP` soubor.

Program se skládá ze souborů main.py a packetParse.py. Main.py slouží k inicializaci programu a načtení vstupů. PacketParse obsahuje hlavní logiku aplikace.

Argumenty

- -pcap <file.pcap>: vstupní PCAP soubor
- -init: vrací seznam `Bootstrap` uzlů
- -`Peers`: vrací seznam aktivních uzlů - probíhal mezi nimi užitečný provoz
- -download: vrací podrobnosti o stahovaném souboru
- -packetLim: horní limit zpracovaných paketů - volitelný argument
- -h : výpis nápovědy

3.2.1 Módy běhu

Init

Tento mód má za úkol extrahovat adresy `Bootstrap` serverů. Využil jsem poznatku, že adresy `Bootsrap` serverů nejsou zabudovány v klientovy v podobě `IP` adres, nýbrž jsou nahrazeny doménovými jmény nad kterými je následně provedena rezoluce. Tudíž jsem si poznamenal všechny adresy které jsou získány `DNS` rezolucí v komunikaci a sledoval jsem jestli s těmito adresami je navázána komunikace pomocí `BT-DHT` protokolu komunikace. Pokud mezi těmito adresami byla provedena užitečná komunikace, byl uzel označen za `Bootstrap`ový.

`Peers`

Mód autor pojal jako detekci aktivních `Peers`, to jsou uzly z kterými byla provedena užitečná komunikace a proto jsme si jistí, že uzly nejsou chybné a lze s nimi navázat spojení. Uzly autor extrahoval ze správ `Handshake` a informace o průběhu stahování byly vypreparovány ze zpráv `Piece`. Extrakce těchto informací tedy proběhla za pomocí zpráv `BITTORENT` protokolu.

O každém peeru se ukládají počty spojení, počty odeslaných a přijatých paketů, celkového množství odeslaných kousků a ratio. Počty spojení jsou inkrementovány v případě potvrzeného handshaku (pokud `Handshake` není potvrzen, je tato skutečnost brána jako nedokončení spojení i když, Peer může odesílat data).

V tomto módu běhu jsou na více extrahovány všechny adressy peerů a nodů (nodes), které jsou následně vypsány po výpisu aktivních peers. Tyto položky byly vzaty ze zpráv protokolu `BT-DHT`. Vzhledem k obvyklému počtu těchto položek je uživatel vyzván k potvrzení výpisu, pokud počet položek přesáhne 30.

Download

Tento mód slouží pro extrakci informací o stahovaném souboru. `InfoHash` je získán ze zprávy `Handshake`. Stejně tak jsou získány adresy kontributorů (stejně jako v módu Peers). Velikost souboru je odhadnuta za pomocí zpráv `Piece` a `Bitmap`. Ze zpráv `Piece` je extrahována informace o velikosti dílků (uvažována je nejdelší velikost dílku) a nejdelší offset dílku. Velikost dílku byla nejčastěji 16384B, ale podle standardu může být různá. Součástí výpisu jsou i počty přenesených dílků. Z `Bitmapy` jsem extrahoval její délku a jednoduchým výpočtem jsem dosáhl celkové velikosti.

U malých souborů jsem narazil na problém v podobě nepřítomnosti zprávy Bitmap a proto nešla spočítat informace o celkové délce soubory.

3.3 Testování

Testování bylo prováděno na virtuálním stroji `Ubuntu` a `qBitTorent` klientem. Vždy se testovalo stažení jednoho souboru. Výstupy byly kontrolovány křížově s logy pomocí ruční analýzy pomocí programu `Wireshark`. Stahovanými soubory byly povětšinou větší soubory (nad 200MB). U velmi malých souborů neproběhla užitečná komunikace k detekci velikosti souboru.

V rámci testování autor shledal program za funkční. Testovací `PCAP` soubory jsou přiloženy v archivu.

Kapitola 4

Diskuze a závěr

Detekce `Bootstrap` uzlů pomocí `DNS` rezoluce se ukázala býti velmi efektivní. Zachycení účastníků komunikace pomocí zpráv `Handshake` byla také poměrně spolehlivou metodou detekce. K následném seznamu získaných Nodes a Peers autor žádné výtky neměl.

Informace o stahovaném souboru byly pravdivé. Jedinou výjmou bylo získání velikosti souboru u malých souborů - jednoduše však lze odvodit velikost podle počtu stažených dat.

Metodika programu se v době psaní dokumentace zdála býti správnou a relativně spolehlivou. Pro lepší efektivitu programu by autor radil program přepsat do rychlejšího jazyku (pravděpodobně C++). Předfiltrování komunikace by také pomohlo optimalizaci programu.

V rámci dalšího zlepšování bych zvážil monitoring samotných TCP spojení, aby jsme mohli zanalyzovat celý kontext komunikace, což by však vedlo na horší efektivitu programu.

4.1 Závěr

V rámci tohoto dokumentu byla popsána architektura a funkcionalita `BitTorent` protokolu. Tato analýza byla provedena na základě reálných dat z experimentů a podpůrné literatury. Konkrétní experimenty byly popsány a poznatky z nich prezentovány.

V implementační kapitole byl představen nástroj pro skenování internetového provozu. Součástí představení byl popis vnitřní architektury, popis spuštění a vysvětlení použitých metod. Práce byla zakončena testováním a diskuzí.

Literatura

[1] *Bittorrent Protocol Specification v1.0* [online]. Únor 2017 [cit. 2023-04-17]. Dostupné z: https://wiki.theory.org/BitTorrentSpecification#Peer_wire_protocol_.28TCP.29.

[2] Cohen, B. *The BitTorrent Protocol Specification* [online]. 2022 [cit. 2023-04-17]. Dostupné z: https://www.bittorrent.org/beps/bep_0003.html.

[3] Matoušek, P. *Sítě peer-to-peer (P2P)* [online]. 2023 [cit. 2023-04-17]. Dostupné z: https://moodle.vut.cz/pluginfile.php/573594/mod_resource/content/1/pds-p2p.pdf.

[4] Woodford, C. *BitTorrent* [online]. 2022 [cit. 2023-04-17]. Dostupné z: https://www.explainthatstuff.com/howbittorrentworks.html.