Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anton Luka Šijanec

Statistika omrežja BitTorrent na podlagi metapodatkov iz DHT

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentorica: izr. prof. dr. Mojca Ciglarič

Ljubljana, 2026

Kandidat: Anton Luka Šijanec
Naslov: Statistika omrežja BitTorrent na podlagi metapodatkov iz DHT
Vrsta naloge: Diplomska naloga na interdisciplinarnem univerzitetnem študijskem programu prve stopnje računalništvo in matematika
Mentorica: izr. prof. dr. Mojca Ciglarič

Opis:
Besedilo teme diplomskega dela študent prepiše iz študijskega informacijskega sistema, kamor ga je vnesel mentor. V nekaj stavkih bo opisal, kaj pričakuje od kandidatovega diplomskega dela. Kaj so cilji, kakšne metode naj uporabi, morda bo zapisal tudi ključno literaturo.

Title: BitTorrent network statistics based on metadata from DHT

Description:
The student shall transcribe the text of the diploma thesis topic from the student information system, where it was entered by the mentor. In a few sentences, they will describe what is expected of the candidate's thesis. What the objectives are, what methods should be used, and perhaps they will also list the key literature.

Hvala mentorici izr. prof. dr. Mojci Ciglarič za uporabne nasvete in odzivnost ter staršem za vso podporo.

Kazalo

Povzetek

Naslov: Statistika omrežja BitTorrent na podlagi metapodatkov iz DHT

Avtor: Anton Luka Šijanec

Diplomsko delo predstavi protokol BitTorrent in njegov podporni sistem DHT in se osredotoči na uporabo slednjega za pridobivanje podatkov, ki jih uporabniki tega porazdeljenega omrežja prenašajo, brez potrebe po privilegiranem dostopu do centralnih strežnikov. Opisana je zasnova in nekaj tehničnih značilnosti za ta namen spisane programske opreme, ki z normalnim sodelovanjem v omrežju DHT na podlagi poizvedb drugih uporabnikov pridobiva zgoščene vrednosti metapodatkov (imena datotek in njihove velikosti), ki jih nato od soležnikov prenese. Na podlagi štirih takih zajemov podatkov v letih 2023, 2024 in 2026 (skupno okoli $1{,}5 \times 10^6$ torrentov) v nalogi analiziramo stanje omrežja. Predstavljena statistika vključuje tipe datotek, ki jih uporabniki prenašajo, zastopanost odjemalcev, razširjenost protokola IPv6, podrobnosti o kvaliteti videovsebin in razširjenost uporabe protokola na državo. Navedene so tudi pomanjkljivosti izvedenega zajema in težave, na katere smo med zajemom naleteli.

Ključne besede: bittorrent, dht, p2p, kademlia.

Abstract

Title: BitTorrent network statistics based on metadata from DHT

Author: Anton Luka Šijanec

The thesis introduces the BitTorrent protocol family and its supporting protocol, the distributed hash table (DHT), focusing on using DHT to acquire data (generated by users of BitTorrent) from the BitTorrent network without requiring any privileged access to remote infrastructure servers. For the acquisition, specialised software, described in this thesis, was developed that records queries that other users' clients, by protocol design, route through our observation node in the DHT network. Those queries contain hashes of torrent metadata which are used to obtain said metadata (file names and sizes) from peers. From four independent acquisitions of data in years 2023, 2024 and 2026, containing information of around $1{,}5 \times 10^6$ torrents, we deduce the state of the BitTorrent network. Presented statistics include types of files that are downloaded by users, which client software is most used, representation of IPv6 protocol in torrent communication, details about the quality of transferred video content and BitTorrent usage per country. We also talk about the shortcomings of our acquisition approach and present possible improvements, and list some problems we encountered.

Keywords: bittorrent, dht, p2p, kademlia.

Poglavje 1

Uvod

1.1 Motivacija

Dandanes na področju računalniških komunikacij izstopajo razprave o porazdeljenih sistemih, natančneje glede opuščanja klasičnega doslej dolgo uveljavljenega koncepta strežnik-odjemalec v prid modernejšim shemam, denimo brezstrežniški infrastrukturi in komunikacijskim protokolom, ki niso odvisni od centralne entitete, temveč jo poganjajo in na nek način vzdržujejo uporabniki sami, kar naj bi omogočalo boljšo odpornost na izpade, cenzuro in drugačno zlonamerno vplivanje s strani upravljavcev. Ker taki sistemi skoraj vedno slonijo na opremi, ki je pod nadzorom velikega števila uporabnikov, so podatki kot ključni sestavni deli sistemov tudi shranjeni oziroma posredovani prek uporabniške opreme. Ko take sisteme obravnavamo, je zato pomembno vprašanje, kako dostopni so (osebni) podatki, ki se prek njih pretakajo, komu vse in v kakšni meri so prosto dostopni za prenos, in do kakšnih sklepov o porazdeljenem omrežju oziroma njegovih uporabnikih lahko z zajemom teh podatkov pridemo [21, 17].

Eno izmed najstarejših še vedno aktivno uporabljenih porazdeljenih omrežij za izmenjavo podatkov oziroma komunikacijo je protokol BitTorrent, katerega začetki segajo četrt stoletja v preteklost [8]. Protokol je načrtovan tako, da omogoča čim preprostejšo distribucijo datotek prek interneta veliki

množici ljudi, s čim manjšo obremenitvijo internetne linije objavljalca, kar doseže tako, da prenašalci datoteke po prenosu tudi sami začno oddajati drugim, ki želijo to datoteko prenesti, s čimer linijo izvornega objavljalca razbremeni.

V nalogi predstavimo, kako protokol DHT na osnovi protokola Kademlia [31], podporni sistem omrežja BitTorrent, za omrežje neinvazivno uporabimo na način, ki ni mišljen kot funkcionalnost samega protokola, temveč kot posledica njegovega delovanja, da pridobimo velike količine metapodatkov o datotekah v omrežju, ki se trenutno prenašajo. Nadalje tudi analiziramo statistično reprezentativen tako pridobljen vzorec metapodatkov iz omrežja, kjer metapodatki predstavljajo datoteke, ki se prenašajo (njihova imena in velikost), ob katerem času se prenašajo in kdo jih prenaša (naslov IP).

1.2 Cilji

Nalogo sestavljata dva ključna cilja, prvič želimo razviti program za množični zajem vzorca metapodatkov iz omrežja BitTorrent, ki ne obremenjuje omrežja (torej komunikacijske opreme ostalih uporabnikov) tako, da bi z zajemom kakorkoli znatno vplival na njegovo delovanje. Želimo si tudi, da pridobljen vzorec metapodatkov vsebinsko ni zaznamovan; vzorec mora biti nepristranski, vsebovati mora naključen vzorec datotek, ki jih prenaša naključno izbrana skupina uporabnikov. Nočemo torej vzorca, ki bi bil zaznamovan recimo geografsko (torej da bi npr. pristransko prednostno vseboval metapodatke o prenosih računalnikov v Evropi) – to vključuje tudi nepristranskost glede omrežne družine, naš zajem ne sme biti omejen samo na IPv4 ali IPv6, ker bi to zaradi različno uspešnega [7] prodora IPv6 po državah kazalo popačeno sliko.

Drugič želimo pridobljene metapodatke analizirati in iz njih izvleči najbolj ključne značilnosti uporabnikov omrežja; kaj prenašajo, kdaj, s kakšno programsko opremo, itd.

1.3 Hipoteze

Glede na že izdelane analize prometa omrežja BitTorrent, pridobljene po drugačni poti [15], domnevamo, da bodo največji delež vsebin v omrežju predstavljale videovsebine, natančneje piratski filmi. Omrežje BitTorrent ljudje po raziskavah [15] uporabljajo predvsem za ogled filmov brez ustreznih avtorskih pravic, za kar je omrežje dobro prilagojeno; filmi so velike datoteke, njihovo deljenje v peer-to-peer omrežjih za objavljalce predstavlja manjši strošek, kot če bi se posluževali klasičnega modela strežnik-odjemalec [47], poleg tega pa so manj izpostavljeni organom pregona, saj je v primeru peer-to-peer omrežij težje določiti, kdo je izvorni objavljalec neke datoteke, ker jo je moč prenesti od velikega števila naprav v internetu [39].

Ker je iz metapodatkov moč določiti tudi različico programske opreme pošiljatelja, domnevamo, da uporabniki omrežja vsled rednih avtomatskih posodobitev programske opreme in dotoka novih uporabnikov v povprečju pretežno uporabljajo najnovejše različice programov za povezavo v omrežje BitTorrent.

1.4 Stanje področja in sorodna dela

Statistične raziskave prenesenih podatkov na omrežju BitTorrent potekajo že vse od njegovega nastanka [29]. V glavnem se zaradi precej preprostejšega načina zajema osredotočajo na podatke, ki jih zberejo centralni strežniki sledilniki ali pa podatke, ki jih zbirajo na usmerjevalnikih omrežnih operaterjev [43]. Ti podatki za to nalogo niso zanimivi; mi se osredotočamo na podatke, ki jih je moč pridobiti brez relativno privilegiranega dostopa do dnevniških in analitičnih datotek na večjih sledilniških strežnikih.

Ta naloga se osredotoča na Mainline DHT (Kademlia), ki je dandanes *de facto* implementacija DHT-ja za BitTorrent. Pred trdno uveljavitvijo/standardizacijo tega DHT protokola je podporo za DHT kot prvi tak odjemalec dobil odjemalec, danes znan kot Vuze [23]. O prečesavanju Vuze DHT za statistiko omrežja piše članek [46].

Ves čas Mainline DHT preiskujejo mnogi za omrežje invazivnejši [3] programi, kot tu opisan. Njihov prvotni namen ni pridobivanje vzorca za statistično analizo, temveč izdelava zbirnega iskalnika po datotekah, ki jih je moč prenesti z odjemalec za torrente. Nekateri trenutno dostopni iskalniki torrent vsebin, ki jih poganjajo preiskovalci Mainline DHT, so [33, 14, 16, 28].

Poglavje 2

Teoretične osnove

2.1 Protokol BitTorrent

Datoteka torrent (krajše torrent) vsebuje metapodatke o končni fiksni množici nespremenljivih datotek. Kdor želi prek BitTorrent razširjati neke datoteke, ki jih ima, mora izdelati datoteko `.torrent` in jo prek poljubnega kanala posredovati bodočim prejemnikom datotek. Ta za vsako datoteko vsebuje njeno relativno pot, ki se konča z imenom datoteke (distribucija s torrenti namreč ohranja hierarhično strukturo map) in njeno dolžino v bajtih. Ključna sestavina torrenta so tudi zgoščene vrednosti (SHA-1 [12]) koščkov datotek, kar zagotavlja integriteto prenesenih podatkov, in velikost koščka, ki je poljubna, a za en torrent konstantna. BitTorrent ne definira iskanja po torrentih; če želimo prenesti datoteke nekega torrenta, moramo datoteko torrent (ali pa vsaj njeno zgoščeno vrednost – 2.1) tudi imeti. Njena struktura je naslednji bkodiran slovar:

- `announce`: HTTP URL sledilnika. Tega v naši nalogi ne uporabljamo, niti ne pridobimo, saj je izven slovarja `info`.
- `info`: Slovar z metapodatki o torrentu:
  - `private`: Oznaka, da je torrent zaseben [18]. Odjemalci v primeru zasebnih torrentov po soležnikih sprašujejo le sledilnik, naveden

v datoteki torrent, ki jo imajo. Takih torrentov v nalogi sploh ne bomo zaznali, ker jih v DHT ni.

- `name`: Ime torrenta. Če ima torrent samo eno datoteko, je to ime te datoteke.
- `piece length`: Velikost koščka.
- `pieces`: S SHA-1 zgoščene vrednosti koščkov, staknjene v en dolg niz, se pravi je dolžina celega niza deljiva z dvajset.
- `length`: To polje je prisotno le, če torrent vsebuje samo eno datoteko. V tem primeru je pod ključem `length` vpisana njena dolžina.
- `files`: Seznam datotek v torrentu, za vsako pa slovar:
  - `length`: Dolžina datoteke v bajtih.
  - `path`: Seznam komponent poti, ki se konča z imenom datoteke, torej za datoteko `slike/morje/2026/IMG_5824.jpg` bo pripadajoč seznam `["slike", "morje", "2026", "IMG_5824.jpg"]`. S takim zapisom poti se izognemo dvoumnostim z različnimi ločili poti (\ in /). Odjemalci morajo preveriti, da elementi seznama ne vsebujejo spornih podnizov, denimo `..` ali `/`.

Košček predstavlja najmanjše zaporedje bajtov fiksne dolžine[1], ki ga je moč zanesljivo in overjeno prenesti prek torrent omrežja. Po protokolu BitTorrent se prenašajo samo koščki, ne datoteke; za prenos vsebino vseh datotek staknemo skupaj (brez polnila med njimi) in dobljeno zaporedje bajtov razdelimo na koščke enake dolžine, ki jih oštevilčimo začenši z 0. Če je dolžina koščka L bajtov, to je lahko npr. 1 MiB, bo prvi košček, ki vsebuje kakšen bajt druge datoteke v torrentu, na indeksu $\left\lfloor \frac{v_0}{L} \right\rfloor$, če je v_0 velikost prve in v_1 druge datoteke v bajtih. Za prenos cele druge datoteke moramo prenesti koščke na intervalu $\left[\left\lfloor \frac{v_0}{L} \right\rfloor, \left\lfloor \frac{v_0 + v_1 - 1}{L} \right\rfloor \right]$.

[1] Izjema je seveda zadnji košček, ki je lahko krajši, če skupna dolžina datotek ni deljiva z velikostjo koščka.

Ker so v datoteki torrent navedene zgoščene vrednosti po koščkih, ne po manjših enotah, ne moremo overiti krajših zaporedij bajtov, kot dolžina koščka, čim pa pridobimo celoten košček in preverimo ujemanje zgoščene vrednosti, smo lahko prepričani, da je tak, kot je bil v datoteki izvornega izdelovalca torrenta, ne glede na to, od koga smo košček prenesli.

infohash je s SHA-1 zgoščena vrednost bkodiranega slovarja `info` v torrent. Unikatno identificira torrent, ker slovar `info` vsebuje vse ključne informacije o torrentu. Če poznamo infohash, lahko iz omrežja rekonstruiramo celotno datoteko torrent, kar počnemo v tej nalogi.

Soležnik je računalnik v omrežju BitTorrent, ki bodisi pridobiva datoteke nekega specifičnega torrenta (pijavka) od drugih soležnikov bodisi jih daje na voljo soležnikom, ki jih želijo (sejalec). Soležnik je lahko hkrati sejalec in pijavka. Čim namreč od nekega sejalca prenese in overi en košček, lahko ta košček začne deliti naprej pijavkam.

Roj je množica vseh trenutno v omrežje povezanih soležnikov nekega torrenta.

Sledilnik je strežnik, ki vzdržuje seznam aktivnih soležnikov (njihove naslove IP in vrata TCP) za posamezen torrent (glede na infohash). Sledilniki predstavljajo šibko točko omrežja, saj lahko izvajajo cenzuro (nek sledilnik lahko zavrne zahteve za določen torrent), lahko pa zaradi izpadov sploh niso dostopni. V nalogi sledilnikov ne obravnavamo; vse funkcije sledilnika namreč lahko zamenja porazdeljena razpršena tabela.

2.1.1 Povezovanje za izmenjavo metapodatkov in datotek

Ko soležnik bodisi prek DHT bodisi prek sledilnikov poizve o roju torrenta, se na soležnike v roju poveže s TCP in po isti povezavi zahteva koščke datotek,

dokler vseh želenih datotek ne prenese, in izpolnjuje zahteve soležnikov, ki želijo od njega koščke prenesti [9]. Tako vzpostavljena povezava je dvosmerna in vezana na en torrent; omogoča prenos koščkov v obe smeri. V tej nalogi koščkov ne prenašamo, zato se osredotočimo na specifično nam koristno funkcijo protokola, ki omogoča prenos celih datotek torrent, ko vemo infohash.

V nalogi s sodelovanjem v DHT pridobivamo zgolj infohashe, zato moramo prenesti še metapodatke – datoteke torrent. To storimo s poizvedbo v DHT o roju, ko dobimo nekaj soležnikov, se povežemo nanje po TCP[2] in opravimo začetno rokovanje BitTorrent (Slika 2.1). Po uspešno opravljenem rokovanju odjemalca drug drugemu pošiljata zaporedje sporočil, ki sestojijo iz štiribajtne dolžine (po pravilu debelega konca), enobajtnega tipa in telesa [9].

Podprtost razširitvenega protokola, katerega del je razširitev za prenos metapodatkov [19], soležnik izkaže tako, da nastavi 20. bit z desne (začnemo šteti z 0) v osmih bajtih za razširitve v sekvenci rokovanja. Brez razširitvenega protokola prenos metapodatkov ni mogoč, v takem primeru prekinemo povezavo. Naš pogoj za nadaljevanje komunikacije je torej `razširitev[5] & 0x10`. Vsa sporočila BitTorrent, ki uporabljajo razširitveni protokol, imajo tip sporočila `20`. Vsako razširitveno sporočilo se začne z enobajtnim podtipom. Če je podtip 0, gre za začetno rokovanje za razširitve, s katerim odjemalca z bkodiranim slovarjem določita številke podtipov za razširitve, ki jih bosta uporabljala, in en drugemu sporočita, katere razširitve podpirata [36].

Specifična razširitev za prenos metapodatkov se imenuje `ut_metadata` in je standardizirana z dokumentom BEP-0009 [19]. Odjemalca, ki jo podpirata, to naznanita na način, ki ga opisuje razširitveni protokol, torej tako, da med razširitvenim rokovanjem pošljeta ustrezen bkodiran slovar, npr. `{"m": {"ut_metadata": 3}, "metadata_size": 31235}`. V tem primeru bi uporabljala podtip 3 za prenos metapodatkov, obenem pa odjemalec, ki metapodatke (datoteko torrent) ima – mi jih nikoli nimamo, to označi tako, da

[2]Obstaja sicer tudi novejši protokol μTP, ki omogoča učinkovitejše prenašanje velikih količin koščkov prek UDP [35], vendar se v nalogi nanj ne fokusiramo, saj prenašamo relativno malo – samo metapodatke.

sporoči njihovo dolžino pod ključem `metadata_size`. Odjemalci običajno med rokovanjem pod ključem `v` pošljejo tudi različico svoje programske opreme, kar tudi beležimo za nadaljnjo analizo.

Ko vemo, da soležnik podpira razširitev za prenos metapodatkov in ima metapodatke, jih od njega zahtevamo po delih dolžine 16 384 B tako, da mu pošiljamo sporočila s podtipom, določenim za `ut_metadata` in bkodiranimi zahtevami `{"msg_type": 0, "piece": 123}`, kjer je 123 zaporedna številka dela, soležnik pa nam odgovarja s sporočili s podtipom za `ut_metadata` in bkodiranim odgovorom `{"msg_type": 1, "piece": 123, "total_size": 964964}`, ki mu neposredno sledijo zahtevani bajti metapodatkov [19].

Po prenosu metapodatkov lahko mi za naše potrebe povezavo prekinemo, resničen odjemalec, ki si želi prenesti vsebino datotek, pa bi nadaljeval s prenosom koščkov datotek, v kar se ne spuščamo. Šele ko prenesemo vse dele metapodatkov, obvezno preverimo njihovo integriteto – s SHA-1 zgoščena vrednost[3] celotnega slovarja `info`, ki smo ga prejeli, se mora ujemati z infohashom tega torrenta.

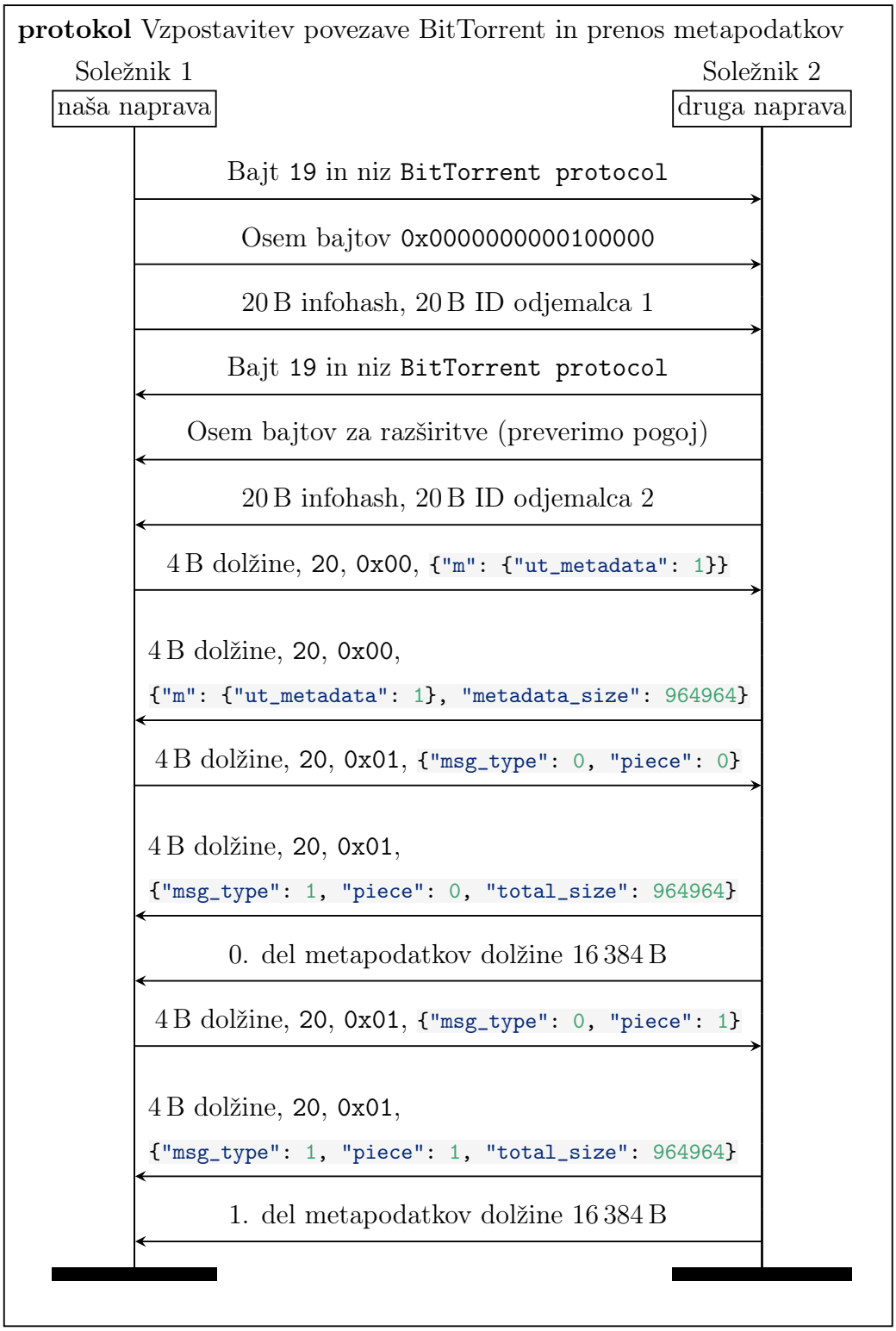
2.2 Mainline DHT

Porazdeljena razpršena tabela (DHT) je dinamičen slovar (slika 2.2). Pod ključem, ki predstavlja infohash torrenta, hrani množico soležnikov. Na najvišjem nivoju podpira dve operaciji:

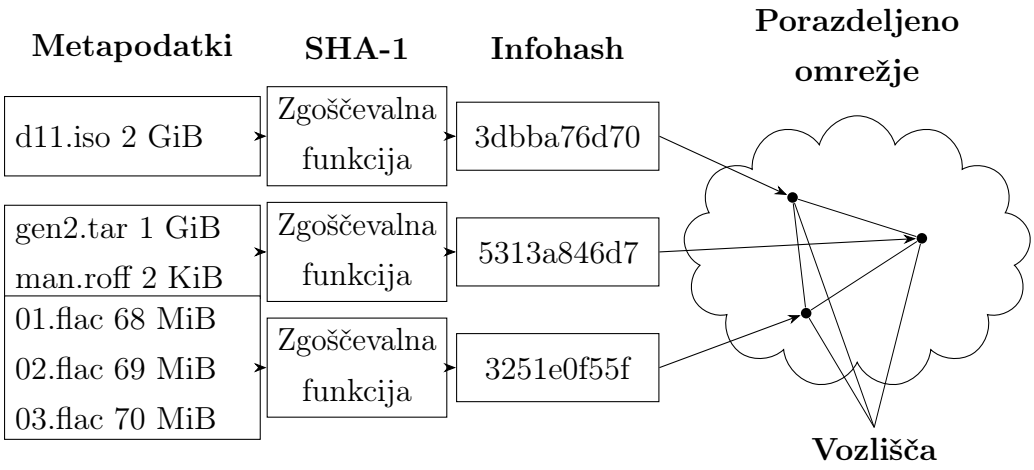
- **Oznani**[4]: Vpiše naslov IP pošiljatelja te operacije in podana vrata v DHT pod ključ, ki pripada podanemu infohashu. Kdor hoče postati soležnik in si želi, da bi se drugi povezovali nanj, s to operacijo objavi svoj naslov.

[3] V drugi različici protokola BitTorrent [10] je zgoščena vrednost prvih 20 bajtov SHA-256 bkodiranega slovarja `info`. Da ne bi izpuščali torrentov druge različice, preverimo ujemanje z obema tipoma zgoščenih vrednosti.

[4] V angleščini „announce“.



Slika 2.1: Prenos metapodatkov po povezavi TCP [19]



Slika 2.2: Shematski prikaz [22] uporabe DHT v BitTorrentu

- **Pridobi soležnike**: Vrne seznam naslovov IP in vrat soležnikov v roju torrenta glede na podani infohash.

Uporabnik v porazdeljenem omrežju DHT, ki izvede te operacije, s tem po zasnovi DHT poizvedbe pošlje prek veliko drugih uporabnikov. V tej nalogi torej kot pasivni uporabnik omrežja DHT pridobimo veliko poizvedb, ki vsebujejo infohashe torrentov, ko pa enkrat dobimo infohashe, pa lahko prenesemo še metapodatke.

Porazdeljen sistem DHT si zamislimo kot redek usmerjen graf, katerega vozlišča so posamezne naprave (odjemalci BitTorrent) kot trojice (naslov IP, vrata, dvajsetbajtni ID), povezave pa so definirane na podlagi usmerjevalnih tabel, ki jih hranijo vozlišča. Vrednosti v tabeli (soležniki) se vpisujejo v vozlišča na tak način, ki kasneje omogoča učinkovite vpoglede, torej na strateško izbrana vozlišča [27].

ID vozlišča mora biti izbran enakomerno naključno [1] na $\{0,1\}^{160}$ in mora biti shranjen na disk – med ponovnimi zagoni vozlišča DHT mora ostati nespremenjen. V primeru zamenjave bi vozlišče lahko ostalo pod

starim ID-jem, a istim naslovom IP v usmerjevalnih tabelah drugih vozlišč. Pogosto menjanje ID-ja bi povzročalo zmedo in slabše delovanje omrežja, saj je porazdeljenost podatkov in usmerjanje poizvedb za podatki po omrežju, kot bomo videli kasneje, odvisna od ID-jev vozlišč.

Usmerjevalna tabela je shranjena na vsakem vozlišču in vsebuje vozlišču sosednja vozlišča (za vsako naslov IP, vrata, dvajsetbajtni ID, stanje aktivnosti in čas zadnjega odgovora). Vozlišča so v njej razdeljena po koših glede na ID. Vsak koš vsebuje največ $K = 8$ vozlišč. Koš i na vozlišču z ID a vsebuje natanko tista oddaljena vozlišča, katerih ID-ji se z a ujemajo v prvih i bitih, ne vsebuje pa vozlišč, ki se ujemajo v prvih $i + 1$ bitih, razen če koš $i + 1$ ne obstaja[5]. Potemtakem je košev teoretično lahko največ 160, saj je prav toliko bitov v ID-ju vozlišča. V praksi jih bo mnogo manj[6], kajti verjetnost, da se dva naključna ID-ja ujemata v prvih i bitih, z i pada eksponentno – $2^{-i} = e^{-i \ln 2}$.

Vozlišča ohranjajo svoje usmerjevalne tabele v zdravem stanju tako, da sosedom periodično pošiljajo poizvedbe `ping` in beležijo odzivnost sosedov. Nedostopne označijo kot take in jih, čim je to možno (čim zasledijo nove), zamenjajo z novo pridobljenimi delujočimi.

Ko vozlišče izve za novo vozlišče, preveri, v kateri koš v usmerjevalni tabeli spada. Če je v košu kaj prostora (bodisi je v njem manj kot K vozlišč bodisi je v njem kakšno vozlišče nedosegljivo), ga vstavi v koš. Če pa na novo najdeno vozlišče sodi v koš i, koš $i + 1$ pa ne obstaja[7], vendar je koš i poln, pa koš i razpolovimo na dva (oziroma dodamo koš $i + 1$ in prerazporedimo vnose v usmerjevalni tabeli); v prvega damo vozlišča, katerih ID-ji se ujemajo z a[8] v i bitih, ne pa v $i + 1$ bitih, v drugega pa vozlišča, katerih ID-ji se z a

[5]Kako je to mogoče, da koš ne obstaja, bo razvidno kasneje; vsako vozlišče namreč začne samo z enim košem in koše dodaja po potrebi – ko spoznava nova vozlišča.

[6]Veliko košev nakazuje na izvajanje napada Sybil, glej razdelek 4.2

[7]Z drugimi besedami: na novo najdeno vozlišče sodi v isti koš kot bi spadalo vozlišče, katerega usmerjevalno tabelo izpolnjujemo (seveda vozlišče sebe ne vpiše v usmerjevalno tabelo).

[8]ID vozlišča, katerega usmerjevalno tabelo gledamo.

ujemajo v $i+1$ in več bitih.

Definicija 2.1 *Metrika XOR* $d_{XOR} : \{0,1\}^{160} \times \{0,1\}^{160} \to \mathbb{Z}_{2^{160}}$ *je definirana kot ekskluzivni ali po komponentah (bitih), ki ga interpretiramo kot nepredznačeno dvojiško celo število.*

Trditev 2.1 d_{XOR} *je res metrika.*

Dokaz. Dokažimo lastnosti metrike:

- definitnost: $\forall a, b \in \{0,1\}^{160} : d_{\mathrm{XOR}}(a,b) = 0 \Leftrightarrow a = b$. Velja, ker za ekskluzivni ali velja $\forall a, b \in \{0,1\} : a \oplus b = 0 \Leftrightarrow a = b$, ničla pa je v nepredznačenem dvojiškem zapisu $(0, 0, \ldots, 0)$.
- simetričnost: $\forall a, b \in \{0,1\}^{160} : d_{\mathrm{XOR}}(a,b) = d_{\mathrm{XOR}}(b,a)$. Velja, ker velja $\forall a, b \in \{0,1\} : a \oplus b = b \oplus a$.
- trikotniška neenakost: Naj bodo $a, b, c \in \{0,1\}^{160}$ poljubni. Velja

$$d_{\mathrm{XOR}}(a,b) \oplus d_{\mathrm{XOR}}(b,c) \stackrel{\mathrm{def}}{=} a \oplus b \oplus b \oplus c = a \oplus c = d_{\mathrm{XOR}}(a,c).$$

Nadalje velja[9] $a \oplus b \leq a + b$ – bitni XOR je samo seštevanje brez prehoda, torej bo rezultat kvečjemu manjši. Iz povedanega sledi $d_{\mathrm{XOR}}(a,b) + d_{\mathrm{XOR}}(b,c) \geq d_{\mathrm{XOR}}(a,b) \oplus d_{\mathrm{XOR}}(b,c) = d_{\mathrm{XOR}}(a,c)$.

□

Ta BitTorrentov DHT deluje na principu metrike XOR. V kontekstu BitTorrentovega DHT-ja so 160-bitni nizi tako ključi v porazdeljeni razpršeni tabeli kot tudi ID-ji vozlišč, kar seveda s pridom izkoristimo.

S prej opisanim upravljanjem z usmerjevalno tabelo sčasoma s sodelovanjem v omrežju vozlišče pride v stanje, ko hrani (beri: je povezano z) le peščico vozlišč, ki so po XOR-metriki precej oddaljena od njega, in dosti več takimi, ki so po definirani metriki blizu njega.

[9] Seštevanje in primerjanje 160-bitnih nizov implicitno definiramo na njihovi predstavitvi kot nepredznačena dvojiška števila v $\mathbb{Z}_{2^{160}}$.

Komunikacija med vozlišči poteka prek UDP in je povsem neodvisna od delovanja protokola BitTorrent (uporablja lahko druga vrata, a isti naslov IP). Celotna vsebina paketa UDP je bkodiran slovar z naslednjimi osnovnimi ključi:

- `y`: Enoznakovni niz, ki določa tip paketa – paket je bodisi poizvedba (`q`) bodisi odgovor (`r`) bodisi napaka (`e`) (na napake se razen zapisovanja v dnevnik v nalogi ne oziramo in jih ne opisujemo).

- `t`: Niz, ki označuje *tranzakcijo*. Tvorec poizvedbe ga poljubno določi, odgovor nanjo pa mora vsebovati isto vrednost. Namenjen je korelaciji prejetega odgovora z oddano poizvedbo.

- `q` (samo v poizvedbah): Niz s tipom poizvedbe, možni so:

  - `ping`: Poizvedba je brez argumentov, vozlišča pričakujejo prazen odgovor.

  - `find_node`: Zahteva vsebuje argument `target` z 20-bajtnim iskalnim nizom. Odgovor pod ključem `nodes` vsebuje niz s K po XOR-metriki iskalnemu nizu najbližjih vozlišč (glede na ID) iz usmerjevalne tabele prejemnika poizvedbe. V nizu si vozlišča sledijo en za drugim, vsako pa je opisano s 26 bajti: 20-bajtni ID, štiribajten naslov IPv4 in dvobajtna vrata UDP. Če vozlišče sodeluje v omrežju IPv6 pa odgovor vsebuje še ključ `nodes6`, kjer je vsako vozlišče opisano z $20 + 16 + 2 = 38$ bajti [5].

  - `get_peers`: Zahteva vsebuje argument `info_hash` z 20-bajtnim infohashom torrenta, za katerega želimo pridobiti soležnike. Odgovor pod ključem `values` vsebuje seznam soležnikov (vsak je predstavljen kot niz IP+vrata), ki so se z `announce_peer` v vprašano vozlišče vpisali kot soležniki torrenta s podanim infohashom, pod ključem `nodes` in `nodes6` pa take podatke, kot bi jih vrnil klic `find_node` s parametrom `target`=`info_hash`. Odgovor vse-

buje še ključ `token` z naključnim nizom, potreben za kasnejši klic `announce_peer`.

– `announce_peer`: Zahteva vsebuje niz pod ključem `token`, prej prejet v poizvedbi `get_peers` (torej je treba pred njo vsaj enkrat na tem vozlišču klicati `get_peers`), s čimer preprečimo, da bi nepridipravi s ponarejenimi izvornimi naslovi IP [13] vpisali kogarkoli razen sebe v roj, ter ključa `info_hash` z 20-bajtnim infohashom torrenta, v katerega roj se pošiljatelj vpisuje, in `port` s celoštevilskimi vrati TCP, na katerih je dostopen po protokolu BitTorrent.

- `a` (samo v poizvedbah): Slovar z argumenti poizvedbe. Poleg zgoraj opisanih vedno vsebuje še ključ `id` z ID-jem pošiljatelja poizvedbe (vozlišča DHT).

- `r` (samo v odgovorih): Slovar z odgovorom. Poleg zgoraj opisanih vedno vsebuje še ključ `id` z ID-jem pošiljatelja odgovora (vozlišča DHT).

S temi poizvedbami lahko definiramo operaciji **pridobi soležnike** in **oznanjaj** (postopek 1). Zaradi takega postopka se podatki o roju nekega torrenta z infohashom h hranijo pretežno na vozliščih z ID-ji blizu h po metriki XOR. Vsled načina izgradnje usmerjevalne tabele in izdelovanja poizvedb ter naključne izbire ID-jev vozlišč pa do tistega vozlišča (in njegovih tesnih sosedov, ki so en do dva koraka proč), ki je izmed vseh v omrežju najbližje nekemu ID-ju, pridemo v $\log_2 n$ korakih (poizvedbah), kjer je n število vseh vozlišč v omrežju. To je bistveno manj, kot da bi morali vprašati vsako vozlišče, in je ključni razlog, zakaj Kademlia nasploh, ne le BitTorrentov DHT, sploh deluje.

Še vedno ni očitno, kako se vozlišče prvič poveže v omrežje. Začeti mora s povezavo z vsaj enim obstoječim vozliščem v omrežju. Obstaja več načinov, kako tako začetno vozlišče pridobiti:

- Odjemalci, ko izdelujejo datoteke torrent, lahko vanje pod ključ `nodes` vpišejo nekaj vozlišč iz svoje usmerjevalne tabele DHT v upanju, da bo

vsaj kakšno vozlišče še dosegljivo, ko bodo prenašalci torrenta datoteko odprli.

- Protokol BitTorrent sam ima razširitev za pridobivanje številke vrat UDP vozlišča DHT odjemalca. Odjemalec, ki ni povezan v DHT, ima pa kak torrent z znanim rojem, pridobljenim recimo iz sledilnika, lahko soležnike povpraša po njihovih vratih UDP, v upanju, da so oni v DHT že povezani [27].

- Programerji odjemalcev gostijo zagonska vozlišča na znanih naslovih, na katera se njihovi odjemalci povežejo in se tako priključijo v omrežje.

- Vsako vozlišče svojo usmerjevalno tabelo trajno shrani na disk, zato da se lahko po ponovnem zagonu v upanju, da je vsaj kako vozlišče še vedno dosegljivo, spet priključi v omrežje.

Omenili smo, da je za prenos datotek v torrentu dovolj, da poznamo infohash. To storimo tako, da z DHT najdemo soležnike, za kar po opisu protokola Mainline DHT očitno potrebujemo samo infohash (ključ) v razpršeni tabeli, nato pa se na enega povežemo in od njega pridobimo še datoteko torrent, za kar zopet potrebujemo samo infohash. V praksi to uporabniki pogosto izkoriščajo, saj je lažje nekomu poslati samo kratek URI kot pa relativno večjo datoteko, ki jo itak prejemnik lahko na podlagi infohasha pridobi iz omrežja. Standarden URI za infohash je oblike `magnet:?dn=ime torrenta&xt=urn:btih:infohash`. Ime torrenta sicer ni bistveno za delovanje protokola, vendar ga vseeno zapišemo v URI, da ima odjemalec kaj pokazati uporabniku, preden pridobi metapodatke.

2.3 Bencoding

Večina struktur (datoteke torrent, paketi DHT, ...) je serializirana v namensko razvito obliko bencoding (bkodiranje). Le-ta omogoča serializacijo preprostih podatkovnih tipov, ki se uporabljajo v BitTorrentu, v zaporedje

Algorithm 1 Postopek za pridobivanje soležnikov torrenta z infohashom h in oznanjanje.

Require: $T_h = \emptyset \wedge V_h = \emptyset$, U so živa vozlišča v usmerjevalni tabeli

Ensure: T_h so soležniki torrenta h, V_h so živa vozlišča blizu torrenta h

1: **procedure** Pridobivaj soležnike in oznanjaj(h)
2: **loop**
3: $N \leftarrow \underset{N \subseteq U,\ |N|=K}{\operatorname{argmin}} \sum_{n \in N} d_{\mathrm{XOR}}(h, n.\mathrm{id})$
4: **for all** $v \in V_h$ **do**
5: **if** v nedosegljivo **then**
6: $V_h \leftarrow V_h \setminus \{v\}$
7: **end if**
8: **end for**
9: **for all** $n \in N \cup V_h$ **do**
10: $s, v, z \leftarrow \mathrm{get\_peers}_n(h)$
11: $T_h \leftarrow T_h \cup s$
12: $V_h \leftarrow V_h \cup v$
13: **if** $z \wedge n \in V_h$ **then**
14: $\mathrm{announce\_peer}_v(h, z)$
15: **end if**
16: **end for**
17: **end loop**
18: **end procedure**

bajtov. Oblika bkodiranja je dovolj preprosta, da je hkrati berljiva s prostim očesom in razčlenljiva s preprostim razčlenjevalnikom.

- Nizi bajtov so serializirani v njihovo dolžino kot desetiško številko, dvopičje in bajte niza same. Niz „diploma“ bo torej serializiran v `7:diploma`.
- Cela števila so serializirana v črko `i`, desetiško zapisano vrednost številke in črko `e`. Število 325858382 bo torej serializirano v `i325858382e`.
- Seznami so serializirani v črko `l`, ki ji en za drugim sledijo serializirani elementi poljubnih tipov izmed vseh štirih navedenih, in črko `e`. Seznam `[23, "julij", 2026]` bo torej serializiran v `li23e5:juliji2026ee`.
- Slovarji so serializirani v črko `d`, ki ji en za drugim izmenično sledijo serializirani ključi (nizi) in vrednosti poljubnih tipov izmed vseh štirih navedenih, in črko `e`. Slovar `{"msg_type": 0, "piece": 123}` bo torej serializiran v `d8:msg_typei0e5:piecei123ee`. Ključi morajo biti leksikografsko urejeni.

V nalogi bomo za večjo preglednost strukture pisali v obliki JSON [4], ker je bolj berljiva od bkodiranja.

Poglavje 3

Arhitektura in načrt

Kot je razvidno iz algoritma 1 (pridobivanje soležnikov in oznanjanje), vozlišča poizvedbe `get_peers` in `announce_peer` pošiljajo prek kar nekaj drugih vozlišč, torej s sodelovanjem v omrežju vsako vozlišče sodeluje pri razreševanju mnogih poizvedb drugih članov omrežja. Normalni odjemalci za BitTorrent na take poizvedbe vestno po najboljši moči odgovorijo, vendar jih nikamor ne shranjujejo[1], ker jim ni pomembno, katere vsebine prenašajo drugi uporabniki.

Prav na ti dve poizvedbi, `get_peers` in `announce_peer`, ki obiščeta vozlišče, se bomo osredotočili v tej nalogi. Poleg tega, da bomo nanju vestno odgovarjali, si bomo zapomnili tudi infohashe torrentov, ki jih take poizvedbe vsebujejo. Tako bomo samo s sodelovanjem v omrežju kot eno samo vozlišče DHT pridobili ogromne količine infohashov torrentov. Nato se bomo prav s pomočjo DHT vpisali v roje teh najdenih torrentov z algoritmom 1 in čim bomo našli soležnike, se bomo nanje povezali in od njih zahtevali metapodatke po metodi, opisani v razdelku 2.1.1. Ko bomo naposled metapodatke prejeli in jih shranili na disk skupaj z informacijo, od koga smo jih prejeli (naslov IP, vrata in program ter različica odjemalca), se bomo izpisali[2] iz roja za ta torrent in tudi tega torrenta v bodoče več ne bomo prenašali.

[1] Razen seveda v primeru `announce_peer`, ko vozlišča shranijo podatke o soležnikih.

[2] Izpis iz roja ni definiran v standardu, toda štejemo, da je nekdo vpisan v roj, če aktivno pošilja poizvedbe `announce_peer`. Čim jih neha, bodo sčasoma vozlišča izbrisala njegov obstoj iz svoje shrambe.

Podatke bomo zajemali v več zajemih v različnih časovnih obdobjih, da bomo lahko primerjali populacijo torrentov in uporabnikov v obtoku skozi čas.

Pozoren bralec opazi, da bomo tako pretežno zbirali infohashe in posledično torrente z ID-ji, ki so po XOR-metriki blizu ID-ja našega opazovalnega vozlišča, kar na prvi pogled izgleda v kontradikciji s ciljem o reprezentativnosti podatkov. K sreči zgoščevalna funkcija SHA, uporabljena za računanje infohasha, infohasha ne izbere pristransko glede na podatke, ki jih zgosti, zato so kljub pristranskemu zajemu podatkov glede na infohash metapodatki sami vseeno enakomerno vzorčeni; neenakomerno vzorčenje glede na zgoščeno vrednost neke vsebine vseeno privede do enakomernega vzorčenja vsebine zaradi razpršilnega značaja zgoščevalne funkcije [32, 45]. Nenazadnje se DHT na to zanaša.

Pristranskost, ki se pojavi s takim načinom zbiranja poizvedb, je pristranskost zaradi popularnosti torrentov. Naše vozlišče bo slišalo veliko več poizvedb za popularne torrente. Toda vsak torrent prenesemo natanko enkrat; ko ga zaznamo drugič, ga ne prenesemo, zato bo v analizi štet le enkrat.

Domnevamo, da je za pridobivanje reprezentativnega vzorca celotnega omrežja tu predstavljen način zajema ustreznejši kot denimo pasivno prisluškovanje povezavam na usmerjevalnikih mrežnih operaterjev, ki bi bilo precej geografsko zaznamovano – najbrž bi zaznali dosti več torrentov, ki jih prenašajo lokalni uporabniki.

Po prejemu dovolj velikega korpusa torrentov bomo vse torrente razčlenili in na njih izvedli analize populacije, denimo kakšne vrste datotek so popularne v omrežju BitTorrent, kakšna je njihova velikost, iz katerih držav so uporabniki omrežja, katere različice programske opreme za odjem torrentov uporabljajo in podobno.

Opomba Algoritem je opisan poenostavljeno. Dalo bi se ga izboljšati, recimo tako, da ne oznanjamo prvih nekaj iteracij, ker so takrat najdena vozlišča v množici V_h še relativno daleč od h po ID. Prav tako običajno

resnične implementacije oznanjajo le na nekaj h-ju najbližjih vozlišč v V_h in ne vsem [6]. Cilj Mainline DHT je informacije o soležnikih torrenta h hraniti le na vozliščih z ID blizu h, ker jih bodo iskalci soležnikov iskali prav/zgolj tam.

Poglavje 4

Implementacija

Naše vozlišče bo hkrati v veliko rojih torrentov, zato bo komuniciralo z veliko drugimi vozlišči DHT. Pričakujemo lahko, da bo v veliko usmerjevalnih tabelah, saj bo pošiljalo veliko poizvedb, torej bo veliko poizvedb tudi prejemalo. Ne želimo, da bi naši odgovori na poizvedbe zamujali, saj bi nas v takem primeru druga vozlišča iz svojih usmerjevalnih tabel zaradi neodzivnosti odstranila.

Odločili smo se za implementacijo vseh v nalogi opisanih protokolov ročno, da lahko nadziramo vse korake njihovega izvajanja in jih prilagodimo za zajem podatkov, ne za prenos datotek, kot to seveda pričakovano počno knjižnice BitTorrent. Minimalističen odjemalec za BitTorrent, ki zna le prenašati metapodatke prek TCP, minimalistično vozlišče DHT, ki pravilno odgovarja na vse poizvedbe in pošteno hrani vpisane soležnike, vendar se zavoljo lažje implementacije požvižga na poslane napake s strani drugih vozlišč, in visokonivojski razčlenjevalnik in kodirnik bkodiranja (tak, ki sprejme tudi malce nestandardno bkodiranje, denimo neurejen vrstni red ključev, a se standardov pri kodiranju strogo drži [38]), smo spisali v programskem jeziku c.

Ker program ni računsko obremenjen, pač pa komunikacijsko (IO), v celoti teče v eni niti in uporablja dogodkovno zanko in nobenih sistemskih klicev, ki blokirajo izvajanje. Jedro izvajanja je dogodkovna zanka, implementirana

s sistemskim klicem `poll(2)` [26]. Program večino časa spi, zbudi se bodisi periodično na nastavljiv časovni interval, v našem primeru 10 s, bodisi takoj, ko prejme kakšen klic iz okolja; to je lahko aktivnost (možno branje/pisanje) na kaki izmed povezav TCP za protokol BitTorrent ali pa prejem kakega paketa UDP od oddaljenega vozlišča DHT – to je lahko tako nova poizvedba kot tudi odgovor na kako našo, poslano prej.

Stanje našega programa med drugim obsega:

- njegov ID
- obstoječe internetne povezave v svet
- usmerjevalno tabelo za DHT
- torrente, katerih metapodatke želimo prenesti (smo v njihovem roju) in njihove množice T_h in V_h
- slovar, s katerim vozlišče DHT hrani soležnike, katerih veljavne poizvedbe `announce_peer` je program prejel

4.1 Knjižnice

Glavnino implementacije smo zasnovali kot dve ločeni knjižnici, vsaka s svojimi vmesniki.

4.1.1 Knjižnica za bencoding

Ker je bencoding po podatkovnih tipih podoben JSON-u, smo knjižnico zasnovali približno tako, kot je zasnovana cjevska knjižnica cJSON za delo z JSON-om. Vsak tip je predstavljen s strukturo (koda 1), ki deluje kot oprimek funkcij za denimo iskanje ključa v slovarju, dodajanje ključa v slovar, spreminjanje seznama in iteriranje po njem.

Implementirali smo tudi metodi za izpis strukture v zaporedje bajtov in razčlenjevanje iz zaporedja bajtov nazaj v interno drevesno strukturo.

Izsek kode 1: Struktura vsakega elementa v knjižnici za bkodiranje[2]

```
struct bencoding {
        struct bencoding * next; // elem ni član slovarja/seznama => NULL
        struct bencoding * prev;
        struct bencoding * child; // le če je slovar/seznam in ima otroke
        struct bencoding * parent; // NULL za korenski element
        enum benc type; // tip tega elementa in dodatne nastavitve
        struct bencoding * key; // ključ elementa, le v slovarjih
        char * value; // vsebina elementa in \0. slovar/seznam => NULL
        size_t valuelen; // dolžina pri nizih (lahko vsebujejo \0)
        long int intvalue; // vrednost celoštevilskih elementov
        int index; // smiselno le pri slovarjih in seznamih
};
```

Nekaj ključnih funkcij, ki operirajo na teh oprimkih:

- `bstr` in `bnum`: niz ali številko ovije v `struct bencoding`
- `b2json`: pretvori `struct bencoding` v json
- `binsert` in `bdetach`: v podan seznam ali slovar vstavi/odstrani element
- `bdecode` in `bencode`: razčleni bkodirano strukturo v `struct bencoding` ali stori obratno
- `bpath`: najde element slovarja na dani poti (v primeru gnezdenih slovarjev)
- `bval`: najde vrednost v slovarju

4.1.2 Knjižnica za DHT in BitTorrent

Ta knjižnica skrbi za omrežne povezave, vse zahteve DHT in vzdrževanje povezav TCP za BitTorrent. Uporabniku poda seznam oprimkov vtičnic, ki jih mora opazovati s `poll` in ob dogajanju na njih obvestiti knjižnico, način za vzpostavitev povratnih klicev v primeru najdenih infohashov in nekaj funkcij

[2]Nekatere interne komponente so zaradi berljivosti izpuščene.

za vpliv na delovanje knjižnice, denimo za dodajanje torrentov za prenos ter zagonskih vozlišč in funkcijo za opravljanje periodičnega dela, torej za pridobivanje soležnikov, oznanjanje in preverjanje stanja usmerjevalne tabele.

V izseku kode 2 je navedenih nekaj struktur za boljše razumevanje sestave knjižnice. Vsa komunikacija knjižnice poteka prek sklada IPv6, za povezavo v omrežje IPv4 se poslužuje V4MAPPED [41].

Izsek kode 2: Nekaj struktur iz knjižnice za DHT in BitTorrent[4]

```c
struct node { // opisnik oddaljenega vozlišča
        unsigned char id[20];
        struct sockaddr_in6 addr; // ::ffff:0.1.2.3 za IPv4 (V4MAPPED)
        int unanswered; // neodgovorjenih zahtev od zadnjega paketa
        time_t last_received; // čas prejema zadnjega paketa od vozlišča
        time_t last_sent; // čas pošiljanja zadnjega paketa
};
struct bucket { // opisnik koša v usmerjevalni tabeli
        unsigned char id[20]; // koš vsebuje [id, next->id)
        struct node * nodes;
};
enum state { // stanja BitTorrent TCP povezave
        blank = 0,
        handshake_sent = 1,
        handshake_received = 1 << 1,
        extension_sent = 1 << 2,
        extension_received = 1 << 3,
        incoming = 1 << 4,
        outgoing = 1 << 5
};
struct torrent { // opisnik torrenta ne nujno v prenosu
        unsigned char ut_metadata; // podtip oddaljenega soležnika
        unsigned char ut_pex; // podtip oddaljenega soležnika
        enum state state; // stanje TCP povezave
        void (* disconnection)(struct torrent *); // signal uporabniku, da
        ↪  shrani metapodatke v datoteko
        unsigned char hash[20]; // infohash
        struct peer * peers; // množica $T_h$
        struct node * nodes; // množica $V_h$
        int progress; // koliko koščkov metapodatkov imamo
```

[4]Nekatere interne komponente in strukture so zaradi berljivosti izpuščene.

```
30          int size; // kako veliki so metapodatki torrenta
31          unsigned char * metadata; // medpomnilnik za metapodatke
32  };
33  struct dht { // glavni oprimek knjižnice
34          unsigned char id[20]; // id našega vozlišča
35          int socket; // vtičnica za UDP
36          unsigned char secret[16]; // za računanje žetonov brez stanja
37          struct bucket * buckets; // usmerjevalna tabela IPv4
38          struct bucket * buckets6; // usmerjevalna tabela IPv6
39          struct torrent * torrents; // torrenti, ki jih želimo prenesti
40          void (* possible_torrent)(struct dht *, const unsigned char *,
            ↪  struct torrent *); // signal uporabniku, da je bil najden nov
            ↪  infohash
41  };
```

Nekaj internih in javnih funkcij knjižnice:

- `closer`: primerja razdalje dveh 20-bajtnih ID-jev do ciljnega po XOR-metriki
- `sendb`: pošlje vozlišču bkodiran objekt kot paket UDP
- `find_node`, `get_peers`, `announce_peer`: pošlje poizvedbo oddaljenemu vozlišču
- `persistent`: serializira stanje, potrebno za shranjevanje na disk, v seznam bajtov (ID, usmerjevalna tabela, ...)
- `token` in `valid`: podporni funkciji za kriptografske žetone `token` v DHT-jevskih `get_peers` in `announce_peer`
- `add`, `subtract`, `divide`, `midpoint`, `distance`: aritmetika na 20-bajtnih številih in metrika
- `split`, `bucket_grade`: razdeli koš, preveri njegovo zdravje
- `replied`: označi vozlišče kot zdravo, saj je odgovorilo na našo poizvedbo
- `potential_node`: najdeno novo vozlišče, ki ga lahko dodamo v usmerjevalno tabelo

- `add_torrent`: doda torrent, bodisi kot uporabniški klic, naj se začne prenos metapodatkov, bodisi kot posledica poizvedbe `announce_peer`
- `handle`: obdela dohodni paket DHT, po potrebi nanj odgovori
- `periodic` in `refresh`: uporabnik ju periodično kliče; preverjata zdravje usmerjevalne tabele, opravljata ostalo periodično delo, opisano zgoraj
- `tcp_work`: upravlja s povezavami TCP za BitTorrent, od rokovanja do prenosa metapodatkov – napolni izhodne medpomnilnike za TCP in prebere vhodne
- `work`: uporabnik funkcijo pokliče, ko `poll(2)` vrne

4.1.3 Program za obdelavo metapodatkov

Spisali smo kratek program v jeziku c za razčlenjevanje datotek torrent, ki omogoča hitro razčlenjevanje, in knjižnico v jeziku python za isti namen. Analizo smo opravljali s pythonom v okolju Jupyter Notebook in pri tem koristili to slednjo knjižnico.

4.2 Težave

4.2.1 Napad Sybil

Vozlišča si po standardu DHT Kademlia ID-je res izbirajo naključno, vendar omrežje tega v osnovni obliki[5] nikakor ne zagotavlja. Vozlišče si lahko ob zagonu izbere poljuben ID, ga tekom programa poljubno spreminja in se različnim vozliščem oglaša z različnimi ID-ji, v najhujšem primeru pa se lahko celo istemu vozlišču oglaša z več različnimi ID-ji z več različnih internetnih naslovov.

[5]Obstaja ukrep, ki napade Sybil malce oteži [34], vendar ga zaradi svoje vzvratne nekompatibilnosti z originalnim standardom DHT ni mogoče implementirati, ne da bi diskriminirali odjemalce, ki ga ne implementirajo. Za potrebe zajema statistično reprezentativnega vzorca populacije torrentov smo se torej odločili, da BEP-0042 ne implementiramo.

Gre za napad Sybil [11], katerega cilj je onesposobitev vozlišča v DHT. Napadalec se pretvarja, da je veliko vozlišč hkrati, in zastrupi usmerjevalno tabelo žrtve. Nadvse učinkovito je napad izveden tako, da so ponarejeni ID-ji takšni, ki so blizu ID-ja žrtve, torej za vsakega izmed teoretično največ 160 košev po K ID-jev na koš (napadalcu je ID žrtve znan, torej ve, kako sporne ID-je konstruirati). Ko je tabela napolnjena, je žrtev zadušena v poplavi izmišljenih ID-jev, večina povezav v njeni tabeli pa vodi do napadalca, ki na njih odgovarja neiskreno (si recimo izmišljuje soležnike in sosednja vozlišča). S tem je omejena njena sposobnost izdelovanja poizvedb v porazdeljeno razpršeno tabelo in shranjevanje soležnikov.

Napad Sybil lahko napadalcem na vzvratno kompatibilen način otežimo tako, da zaznamo neverjetno nenadno večanje usmerjevalne tabele ($\log_2 n$, kjer je n število vseh članov omrežja na svetu, je skozi čas praktično konstanta [42, 44]), ne dodajamo v usmerjevalno tabelo dvakrat istega naslova IP, ne menjamo zdravih obstoječih vozlišč v tabeli z novimi in ne polnimo košev več kot do K vozlišč.

Naša implementacija šteje, da se je napad Sybil zgodil, ko število košev v njeni usmerjevalni tabeli preseže 64. Da so v usmerjevalni tabeli samo iskrena vozlišča, bi bilo sila neverjetno, saj bi pomenilo, da je neko vozlišče z naključno izbiro ID-ja zadelo enakih prvih 64 bitov kot naš prav tako naključno izbran ID. Ko napad zazna, zbriše celotno usmerjevalno tabelo, zamenja svoj ID, se ponovno priključi v omrežje (z uporabo vozlišč za zagon, vpisanih v DNS `_dht._udp.travnik.sijanec.eu. IN SRV`) in nadaljuje z izvajanjem.

4.2.2 Ozko grlo omrežne opreme

Naša implementacija pošilja in prejema veliko paketov UDP v sekundi (ob vrhuncih tudi okoli 2000 paketov v sekundi). Čeprav porablja malo pasovne širine, le nekaj $\mathrm{Mbit\,s^{-1}}$, smo imeli tolikšne težave z zajemom na nekaterih modemih GPON, da je bilo v izogib izgubljenim paketom potrebno zmanjšati frekvenco periodičnega dela knjižnice za DHT in zmanjšati število torrentov, ki jih knjižnica hkrati prenaša.

Teh težav ni bilo na boljši mrežni opremi na strežnikih v Grčiji (GR-NET) [25] in strežnikih na Fakulteti za računalništvo in informatiko UL, le na strežniku na rezidenčnem priključku pri operaterju T-2, pa še to le zaradi slabe terminalske opreme (CPE).

Poglavje 5

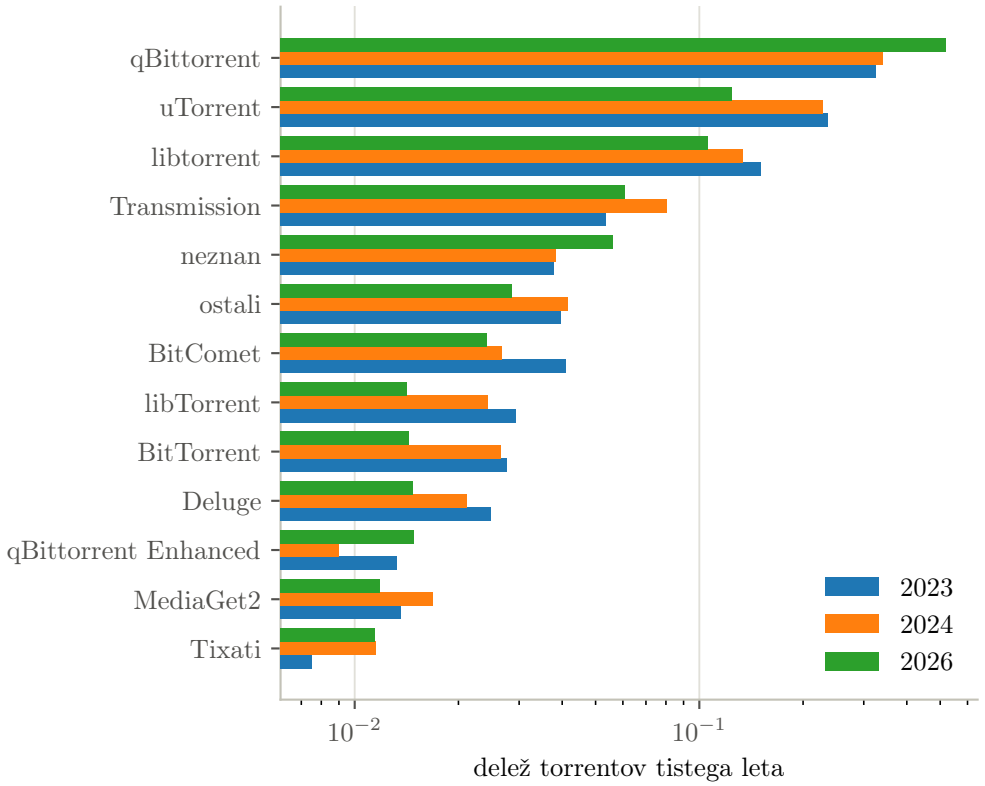
Zajeti podatki in rezultati

Podatke smo z našim izdelanim programom zajemali v štirih zajemih, prikazanih v tabeli 5.1. Skupna velikost prenesenih datotek torrent znaša 78,8 GiB. Velikost datotek v tabeli se nanaša na datoteke, na katere se torrent nanaša. Njih seveda nismo prenašali, samo metapodatke o njih. Njihova imena in dolžine so nam znane, vsebina pa ne.

Analizirali smo v okolju Jupyter Notebook [24] in grafe risali s knjižnico matplotlib [20]. Analizo smo izvajali na računalniku na FRI na 16 jedrih Opteron 6380 in zanjo porabljali 48 GiB pomnilnika. Analiza vseh podatkov skupaj se izvaja slabe tri ure.

Tabela 5.1: Zajemi, uporabljeni v analizi. Zajemi so trajali 16, 31, 5 in 84 dni (po vrsti). Stolpec p vsebuje periodo – koliko časa je v povprečju preteklo med dvema prenosoma torrenta. Izračunano na podlagi časa zajema torrenta, ki smo ga shranili v torrent.

obdobje	lokacija	torrentov	datotek	p
2023-01–2023-02	T-2 GPON	47843	3084321, 259 TiB	29 s
2023-03–2023-05	˜okeanos [25]	414620	17165425, 1891 TiB	6 s
2024-02–2024-02	T-2 GPON	62108	3725125, 344 TiB	7 s
2026-04–2026-07	FRI	1021992	80886646, 6837 TiB	7 s



Slika 5.1: Popularnost odjemalcev po letih.

5.1 Odjemalci

Za analizo uporabe odjemalcev smo za vsako posamezno leto primerjali zastopanost odjemalca (brez različice) v prejetih torrentih. Rezultati so prikazani na sliki 5.1.

5.2 Tipi datotek

Frekvenco po tipih datotek, ki se prenašajo, lahko merimo na več načinov. Ustrezen način je za vsak tip datoteke izmeriti število torrentov, v katerih ta tip predstavlja največjo velikost (slika 5.2). Alternativna načina merjenja bi bila število datotek z nekim tipom (slika 7.1) in skupna velikost datotek s

tem tipom (slika 7.2), ki nista ustrezna, saj bi prednost dala v prvem primeru tipom, ki se navadno pošiljajo kot kopica majhnih datotek (denimo slike ali glasba), v drugem primeru pa tipom, ki zavzemajo dosti velikosti na disku, torej recimo filmom.

Ker vpogleda v vsebino datotek zaradi načina zajema nimamo, lahko tip datoteke in posledično vrsto vsebine določimo le na podlagi imena oziroma natančneje datotečne končnice. V primeru stisnjenih datotek (zip) ne moremo dobro določiti, katere datoteke vsebuje.

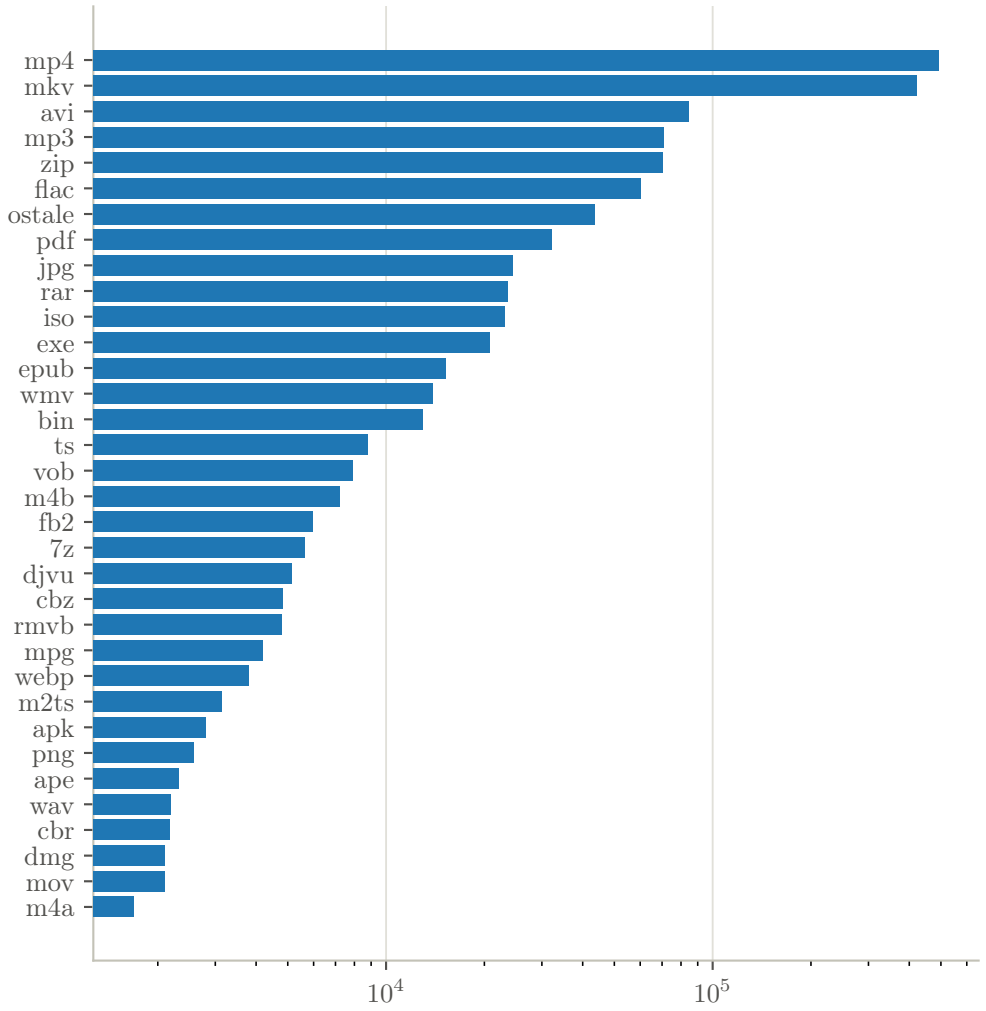
Kot pričakovano so najpogostejša vrsta datotek videovsebina, zato smo s pomočjo pythonske knjižnice [37] kategorizirali vse videovsebine glede na kodek in ločljivost (slika 5.3). Žal podatki temeljijo zgolj na metapodatkih (naslovih) in ne na vsebini datotek, torej izpustimo znaten delež populacije, ki kodeka ali ločljivosti v imenu nima vpisanega.

5.3 Različice programske opreme

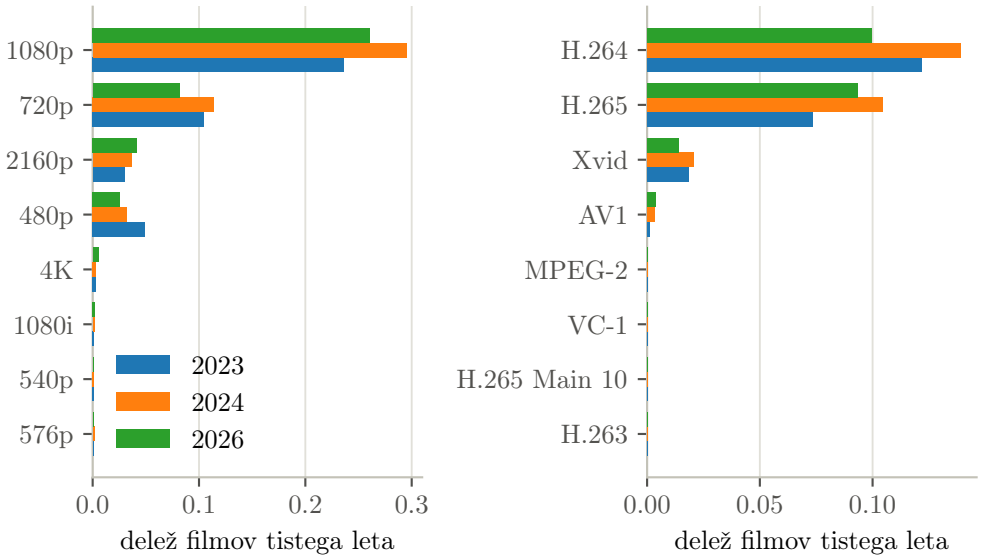
Osredotočili smo se na odjemalec qBittorrent [40], saj je najbolj zastopan v prenesenem korpusu. Pojavnost najdenih različic skozi leta (slika 5.4) potrjuje hipotezo, da bodisi uporabniki res posodabljajo programsko opremo bodisi v omrežje prihajajo novi uporabniki.

5.4 Popularnost po državah

Z uporabo podatkovne zbirke [30] smo določili, iz katere države so naslovi IP, od katerih smo prenesli torrente. Na podlagi tega ocenjujemo popularnost uporabe protokola BitTorrent v posamezni državi (slika 5.5). Podatkovno zbirko, ki smo jo uporabili za določanje države, smo prenesli julija 2026, zato mogoče vsebuje netočne podatke za naslove iz 2023, saj internetni naslovi, čeprav redko, lahko menjajo državo.



Slika 5.2: Tipi glede na število torrentov, pri katerih tip predstavlja glavnino velikosti.



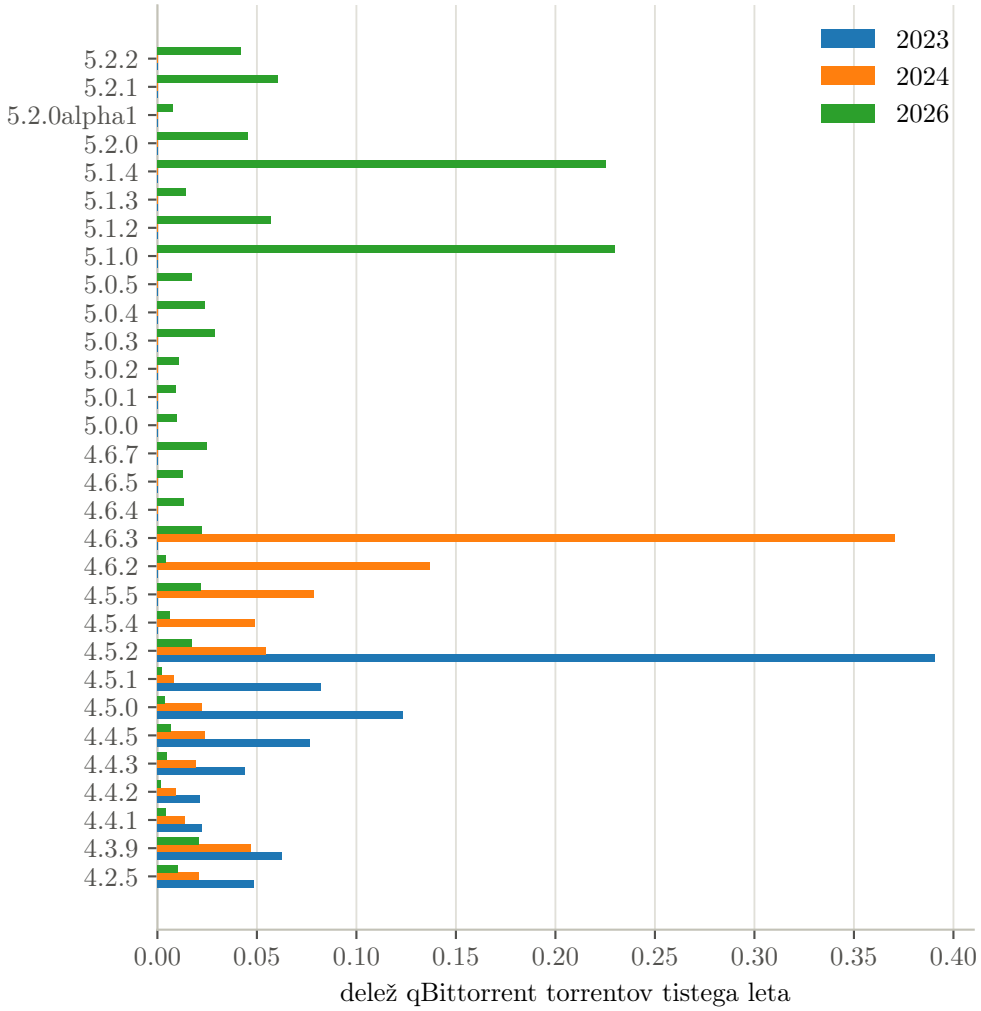
Slika 5.3: Ločljivosti in kodeki videovsebin iz naslovov datotek.

5.5 Prejetih torrentov na dan

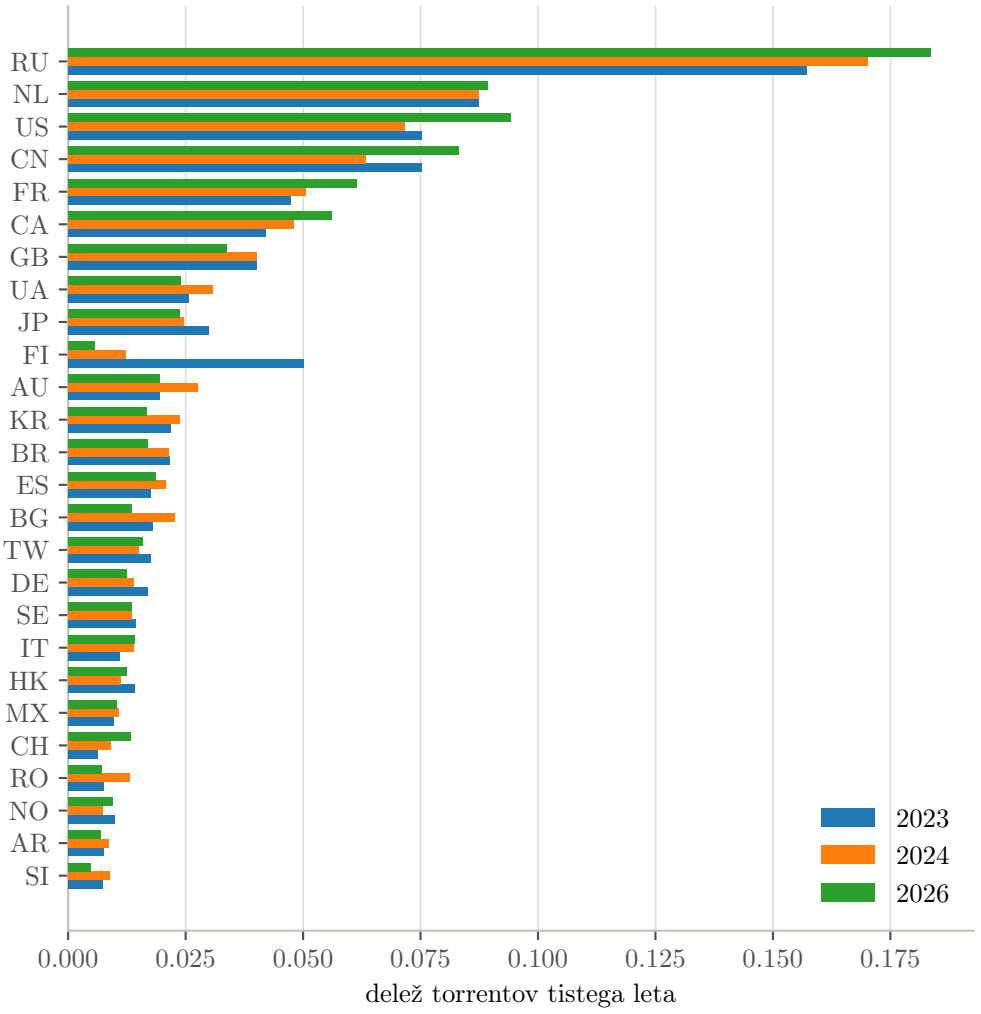
Graf na sliki 5.6 prikazuje, koliko torrentov na dan smo prejeli med zajemom leta 2026.

5.6 Velikost koščka

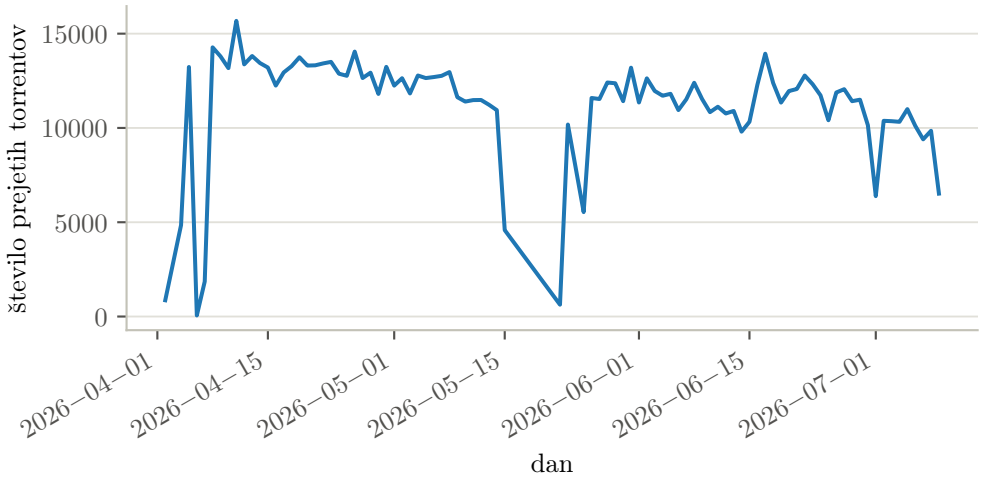
Izvorna različica BitTorrenta, kot smo omenili, v torrentih vsebuje zgoščene vrednosti koščkov dolžine, določene v datoteki. Na sliki 5.7 smo prikazali najpopularnejše dolžine koščka in primerjali dolžino koščka z velikostjo celotnega torrenta. Odjemalci namreč prav glede na velikost predlagajo neko dolžino koščka, vendar jo lahko uporabniki pred izdelavo torrenta spremenijo.



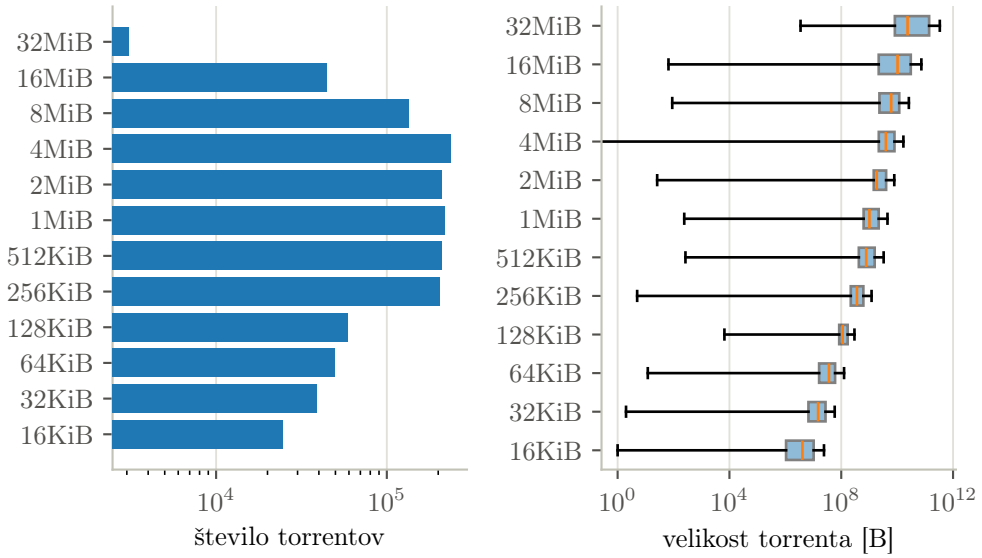
Slika 5.4: Različice programske opreme qBittorrent skozi leta.



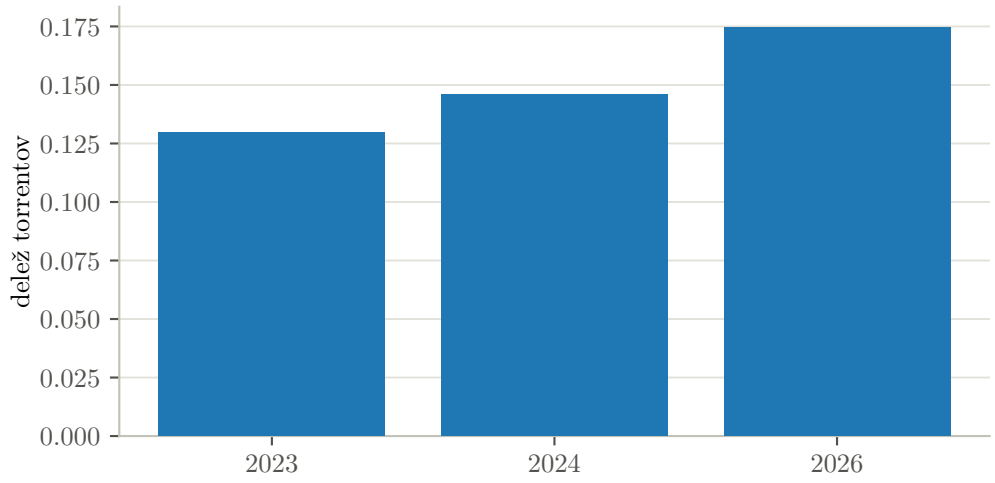
Slika 5.5: Države, od katerih smo prejeli največ metapodatkov, in Slovenija skozi leta.



Slika 5.6: Količina zajetih torrentov na dan v zajemu leta 2026. Zajem smo na začetku nekajkrat prekinili, vidna je tudi ena večdnevna prekinitev.



Slika 5.7: Porazdelitev velikosti koščkov.



Slika 5.8: Rast uporabe IPv6 skozi leta.

5.7 Razširjenost IPv6

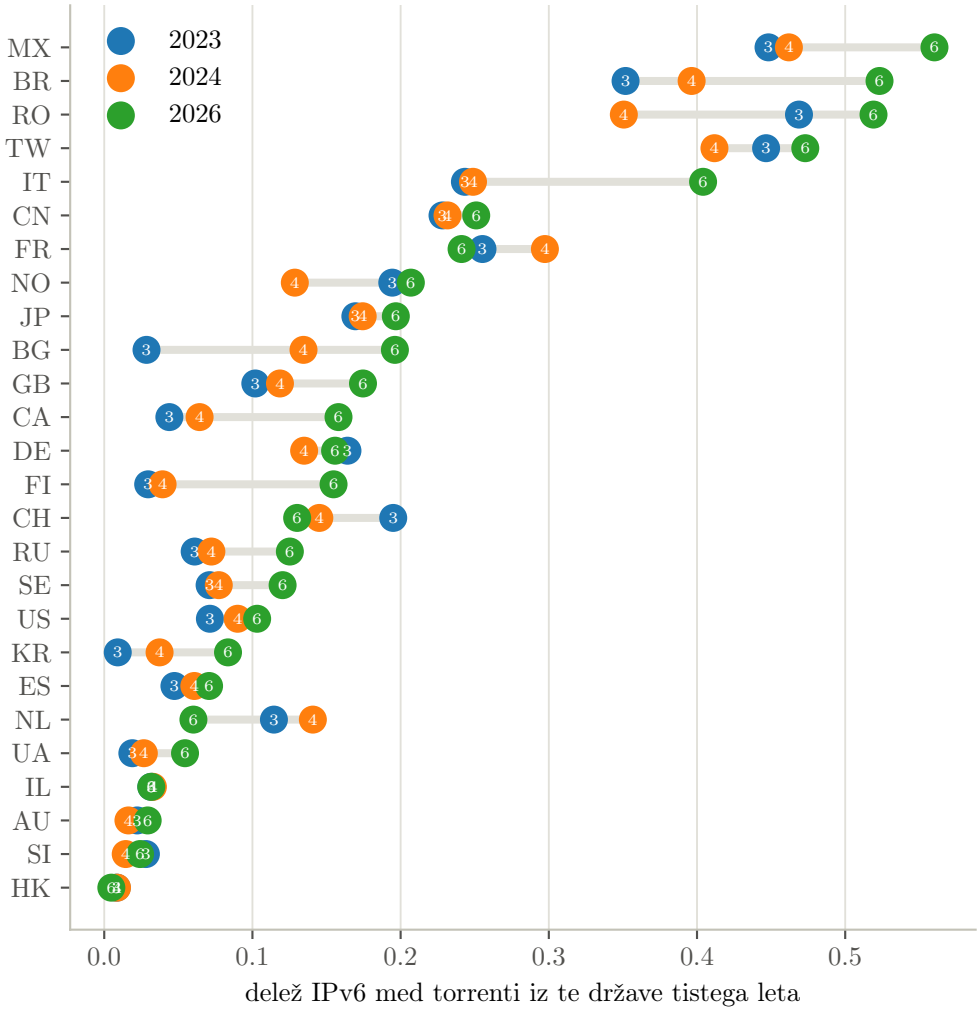
Na sliki 5.8 je opazna rast razširjenosti IPv6 skozi leta. Ocenimo jo kot razmerje med torrenti, prenesenimi prek IPv6 in vsemi prenesenimi torrenti. Podobno statistiko o prenosu torrentov glede na omrežni protokol za vsako državo posebej smo prikazali v sliki 5.9.

5.8 Rekordni primerki v populaciji

Tabeli 5.2 in 5.3 prikazujeta največje odkrite datoteke v Sloveniji in največje odkrite datoteke nasploh.

Najpogosteje uporabljena vrata TCP odjemalcev, po katerih smo prejeli torrente, so prikazana na sliki 5.10, najpogosteje uporabljeni naslovi IP pa na sliki 5.11. Od 75,37 % naslovov IP smo prejeli natanko en torrent.

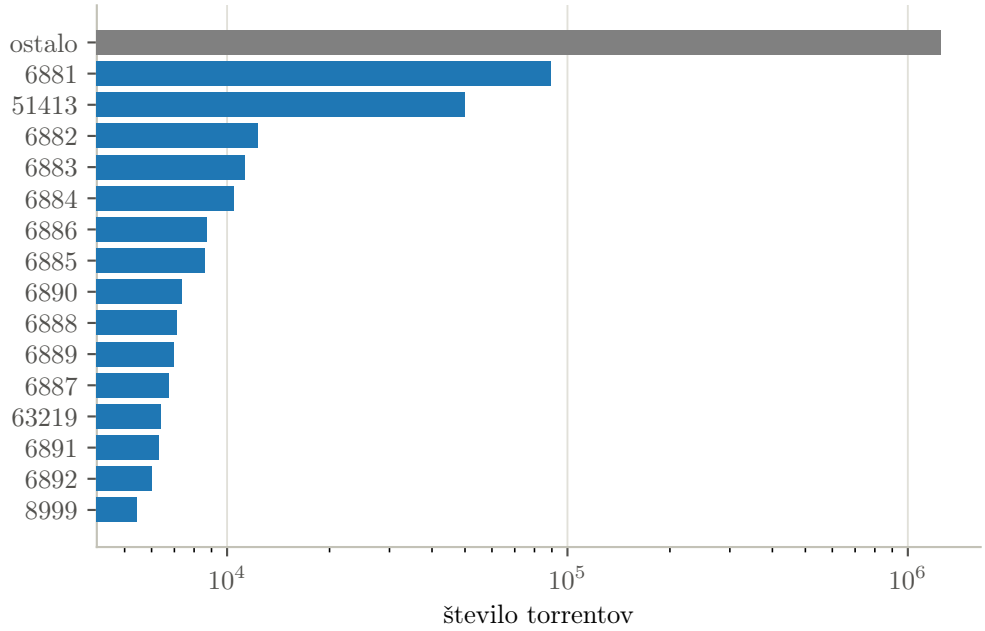
Od 43161 naslovov smo prejeli vsaj dva torrenta v razmaku vsaj enega meseca. Izmed njih jih je svoj odjemalec posodobilo vsaj 6920 (16,03 %). Toliko izmed njih smo namreč videli najprej z nižjo in nato z višjo različico odjemalca.



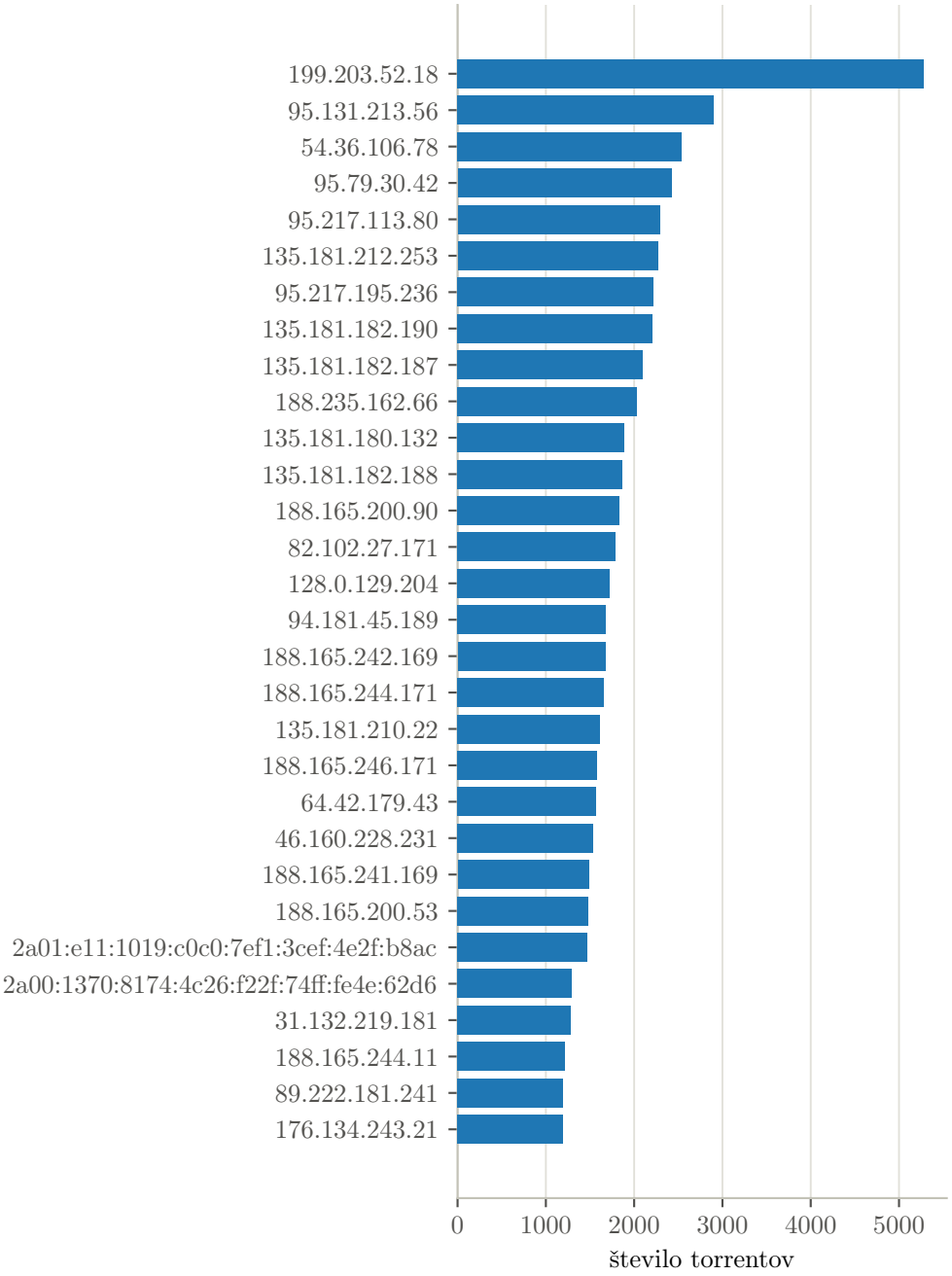
Slika 5.9: Rast uporabe IPv6 skozi leta po državah.

Tabela 5.2: 10 največjih posameznih datotek iz Slovenije

velikost	infohash	ime
155 GiB	`ceafd85e`	rune-stalker.2. heart.of.chornobyl.v1.7.iso
113 GiB	`5b9c5367`	repack-warhammer.3.iso
95 GiB	`b3af5e6d`	Content/DOSMagazines.zip
88,4 GiB	`a7da1c58`	rune-grand.theft.auto.v.enhanced.iso
82,4 GiB	`a5a18729`	Dune.1984.2160p.BluRay.REMUX. HEVC.DTS-HD.MA.5.1-FGT.mkv
78 GiB	`09b53cde`	The Long Kiss Goodnight 1996 2160p UHD Blu-ray Remux HEVC DV TrueHD 7.1 Atmos-HDT.mkv
76,6 GiB	`07215cb9`	Setup-1.bin
69,4 GiB	`99bf377e`	Blade.Runner.2049.Open.Matte. INTERNAL.HDR-X.DoVi.Ver.1.0.4.TrueHD.Atmos.7.1-TEKNO3D.mkv
64,5 GiB	`98f974dd`	In.Bruges.2008.4K.HDR.DV.2160p. BDRemux Ita Eng x265-NAHOM.mkv
62,4 GiB	`c15a97ca`	Shrek.2001.2160p.BluRay.REMUX. HEVC.DTS-X.7.1-FGT.mkv



Slika 5.10: Najpogosteje uporabljena oddaljena vrata TCP, na katera smo se povezovali za prenos metapodatkov.



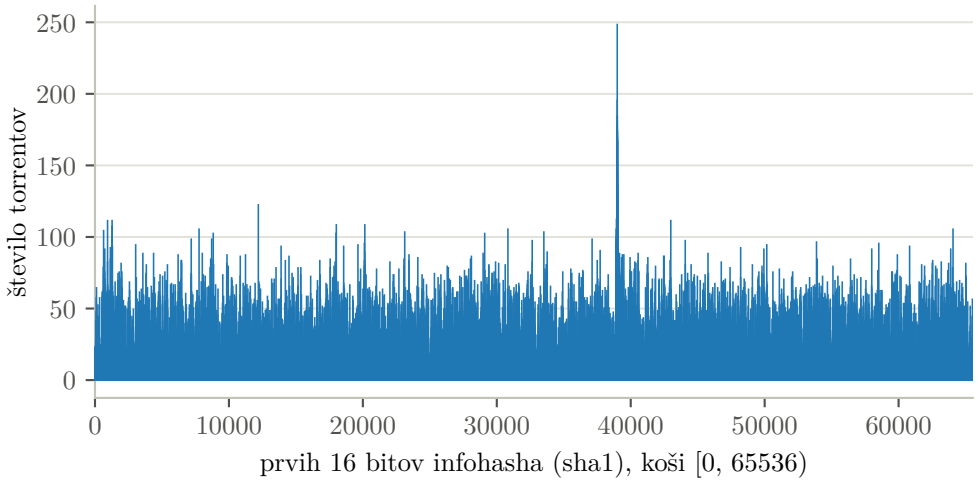
Slika 5.11: Naslovi IP, od katerih smo prejeli največ metapodatkov.

Tabela 5.3: 10 največjih posameznih datotek (globalno)

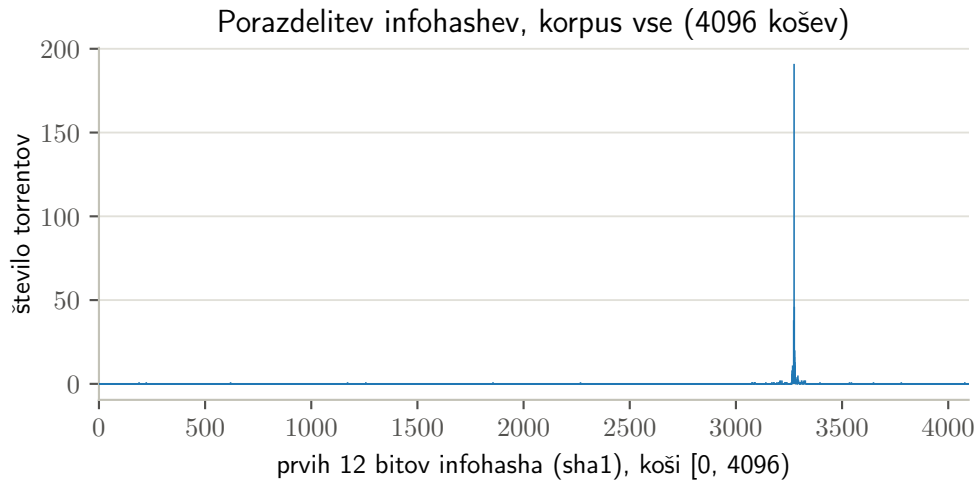
velikost	infohash	ime
3,35 TiB	`37eb5bd3`	annas-archive-ia-lcpdf-per_c.tar
2,81 TiB	`51e30019`	Pegasus G V1.1 without Android only Windows.CHT.7z
2,27 TiB	`e48b620b`	annas-archive-ia-lcpdf-b.tar
2,13 TiB	`5e80fab7`	annas-archive-ia-acsm-m.tar
2,09 TiB	`a883ce9e`	annas-archive-ia-acsm-p.tar
2,04 TiB	`fa023aae`	annas-archive-ia-acsm-a.tar
1,97 TiB	`5bccfd69`	annas_archive_spotify_2025_07_coverart.tar
1,95 TiB	`42c3e52b`	Comfyui_Mie_Models.zip
1,86 TiB	`503655cf`	annas-archive-ia-acsm-l.tar
1,83 TiB	`9cf53879`	annas-archive-ia-acsm-b.tar

Trdili smo, da infohashi najbrž ne bodo enakomerno porazdeljeni, zato smo pripravili izrise porazdelitve prejetih infohashov na sliki 5.12.

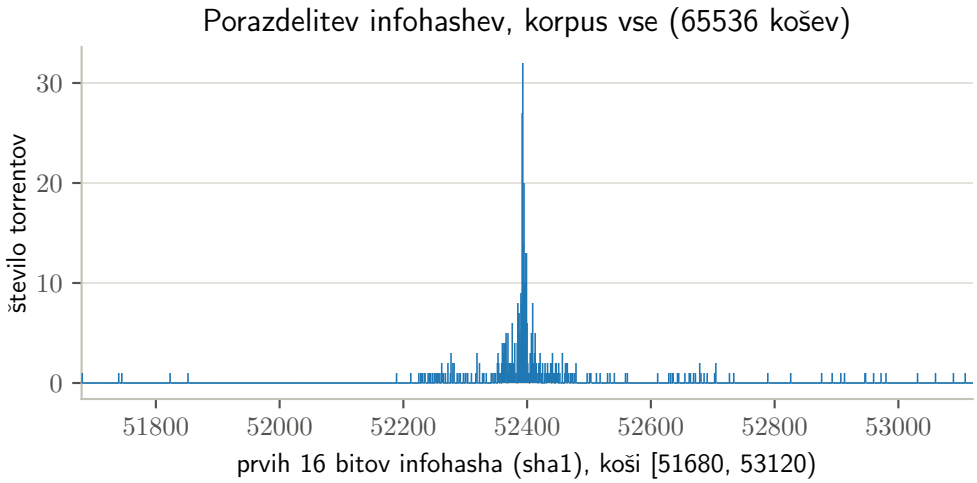
Omenili smo, da smo za reševanje pred napadom Sybil med zajemom menjali ID. Ker smo ga vedno zamenjali na povsem naključnega, iz slike z gostoto porazdelitve infohashov na celotnem korpusu ni razvidno, da bi bili pristranski do izbire infohashov. Da bi bolje prikazali pristranskost ob nespremenljivem ID-ju, smo pognali namenski zajem s konstantnim ID (zajemali smo dokler nismo naleteli na napad Sybil). Slika 5.13 (in povečava na sliki 5.14) prikazuje gostoto porazdelitve tako pridobljenih torrentov.



Slika 5.12: Gostota porazdelitve infohashov prejetih torrentov.



Slika 5.13: Gostota porazdelitve infohashov prejetih torrentov v majhnem namenskem zajemu 518 torrentov.



Slika 5.14: Gostota porazdelitve infohashov prejetih torrentov v majhnem namenskem zajemu 518 torrentov – povečava na območje z veliko torrenti.

Poglavje 6

Zaključek

Predstavljeni podatki kažejo sliko uporabe omrežja BitTorrent. Med analizo opažena ključna pomanjkljivost podatkov, ki izvira že iz načina zajema, je, da nimamo vsebine datotek, poleg tega pa imamo za vsak torrent shranjenega samo enega člana roja – tistega, od katerega smo metapodatke prejeli.

V bodočih raziskavah na tem področju bi bilo vsled tega smiselno pridobivati podatke o celotnem roju, najlažje in najučinkovitejše z dodatkom standarda PeX (izmenjava soležnikov) [2], ter podatke o vsebini datotek od soležnikov. Za določanje podrobnosti tipa datoteke (kodek, vsebina arhiva ZIP, jezik dokumentov, itd.) bi bilo dovolj prenesti le prvi ali zadnji košček datoteke, s čimer bi napram nemogočemu podvigu prenašanja celotne vsebine datotek znatno zmanjšali porabo pasovne širine.

Naša zastavljena cilja o izdelavi programa za zajem in o analizi pridobljenih podatkov sta bila dosežena. Program je po sklepu o algoritmu 1 res nepristranski in omrežja ni motil, saj se je držal standardov za izdelavo odjemalca BitTorrent. Zajeli smo dovolj podatkov, da smo lahko sklepali o sestavi omrežja.

Hipotezo tipu vsebine, ki jo uporabniki prenašajo, potrdimo. Videovsebine so res najpopularnejša oblika datotek glede na končnico. Prav tako na podlagi razdelka 5.3 o različicah odjemalcev qBittorrent sklepamo, da uporabniki res po večini uporabljajo aktualne različice programske opreme.

6.1 Etika in varstvo podatkov

Uporabnike omrežja BitTorrent programska oprema za povezavo v omrežje običajno pred prvim zagonom obvesti o javnosti njihove komunikacije v omrežju. Prav tako odjemalci jasno kažejo naslove IP drugih uporabnikov v omrežju med prenosom podatkov, kar pomeni, da se uporabniki zavedajo, da je njihova aktivnost v omrežju drugim znana. Iz tega sklepamo, da naša raziskava ni pretirano posegala v osebne podatke uporabnikov omrežja.

V nalogi nismo objavili naslovov IP domačih uporabnikov. Objavili smo le naslove, ki rekordno izstopajo po količini torrentov, ki jih prenašajo. Razumno lahko sklepamo, da ne gre za posamezne fizične osebe, temveč za strežnike, ki prenašajo torrente po navodilih velike množice oseb (seedbox).

Literatura

[1] Donald E. Eastlake 3rd, Steve Crocker in Jeffrey I. Schiller. *Randomness Requirements for Security*. RFC 4086. Jun. 2005. DOI: 10.17487/RFC4086. URL: https://www.rfc-editor.org/info/rfc4086.

[2] The 8472. *BEP 11: Peer Exchange (PEX)*. BitTorrent Enhancement Proposal 11. Last modified 2017. BitTorrent.org, 2015. URL: https://www.bittorrent.org/beps/bep_0011.html.

[3] The 8472. *DHT Infohash Indexing*. 2017. URL: https://www.bittorrent.org/beps/bep_0051.html (pridobljeno 28. 2. 2023).

[4] Tim Bray. *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC 8259. Dec. 2017. DOI: 10.17487/RFC8259.

[5] Juliusz Chroboczek. *BEP 32: BitTorrent DHT Extensions for IPv6*. BitTorrent Enhancement Proposal 32. Status: Draft; last modified 21 Jul 2016. BitTorrent.org, okt. 2009. URL: https://www.bittorrent.org/beps/bep_0032.html (pridobljeno 9. 7. 2026).

[6] Juliusz Chroboczek. *dht.c*. GitHub repository jech/dht; copyright 2009–2011; BitTorrent DHT library implementation in C. 2009. URL: https://github.com/jech/dht/blob/master/dht.c (pridobljeno 9. 7. 2026).

[7] Cloudflare Data Insights Team. *IPv6 adoption*. 2022. URL: https://radar.cloudflare.com/reports/ipv6 (pridobljeno 9. 7. 2026).

[8] Bram Cohen. *BitTorrent - a new P2P app*. Internet Archive. 2001. URL: http://finance.groups.yahoo.com/group/decentralization/message/3160 (pridobljeno 15. 4. 2007).

[9] Bram Cohen. *The BitTorrent Protocol Specification*. 2017. URL: https://www.bittorrent.org/beps/bep_0003.html (pridobljeno 28. 2. 2023).

[10] Bram Cohen. *The BitTorrent Protocol Specification v2*. 2017. URL: https://www.bittorrent.org/beps/bep_0052.html (pridobljeno 28. 2. 2023).

[11] John R. Douceur. „The Sybil Attack“. V: *Peer-to-Peer Systems*. Ur. Peter Druschel, Frans Kaashoek in Antony Rowstron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, str. 251–260. ISBN: 978-3-540-45748-0.

[12] D. Eastlake in P. Jones. *US Secure Hash Algorithm 1 (SHA1)*. RFC 3174. Sep. 2001.

[13] Toby Ehrenkranz in Jun Li. „On the state of IP spoofing defense“. V: *ACM Transactions on Internet Technology* 9.2 (maj 2009), str. 1–29. ISSN: 1557-6051. DOI: 10.1145/1516539.1516541. URL: http://dx.doi.org/10.1145/1516539.1516541.

[14] Nina Evseenko. *Btdigg BitTorrent DHT search engine*. 2011. URL: http://btdig.com (pridobljeno 28. 2. 2023).

[15] Arnau Gavaldà-Miralles in sod. „Impact of heterogeneity and socioeconomic factors on individual behavior in decentralized sharing ecosystems“. V: *Proceedings of the National Academy of Sciences* 111.43 (okt. 2014), str. 15322–15327. ISSN: 1091-6490. DOI: 10.1073/pnas.1309389111. URL: http://dx.doi.org/10.1073/pnas.1309389111.

[16] Mike Gibson in sod. *bitmagnet*. Self-hosted BitTorrent indexer, DHT crawler, content classifier, and torrent search engine. URL: https://bitmagnet.io/ (pridobljeno 10. 7. 2026).

[17] Andrew Griffin. *'I Know What You Download': Website claims to let people see everything their friends have torrented*. 2017. URL: https://www.independent.co.uk/tech/torrent-website-download-safe-legal-privacy-i-know-what-you-friends-spying-a7504266.html (pridobljeno 28. 2. 2023).

[18] David Harrison. *Private Torrents*. 2008. URL: https://www.bittorrent.org/beps/bep_0027.html (pridobljeno 28. 2. 2023).

[19] Greg Hazel in Arvid Norberg. *Extension for Peers to Send Metadata Files*. 2017. URL: https://www.bittorrent.org/beps/bep_0009.html (pridobljeno 28. 2. 2023).

[20] J. D. Hunter. „Matplotlib: A 2D graphics environment“. V: *Computing in Science & Engineering* 9.3 (2007), str. 90–95. DOI: 10.1109/MCSE.2007.55.

[21] *I know what you download*. URL: http://iknowwhatyoudownload.com (pridobljeno 28. 2. 2023).

[22] Jnlin. *File:DHT en.svg*. Wikimedia Commons. 2007. URL: https://commons.wikimedia.org/wiki/File:DHT_en.svg (pridobljeno 9. 7. 2026).

[23] Ben Jones. *BitTorrent's DHT Turns 10 Years Old*. 2015. URL: https://torrentfreak.com/bittorrents-dht-turns-10-years-old-150607/ (pridobljeno 28. 2. 2023).

[24] Thomas Kluyver in sod. „Jupyter Notebooks – a publishing format for reproducible computational workflows“. V: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ur. F. Loizides in B. Schmidt. IOS Press. 2016, str. 87–90.

[25] Vangelis Koukis, Constantinos Venetsanopoulos in Nectarios Koziris. „ okeanos: Building a Cloud, Cluster by Cluster“. V: *IEEE Internet Computing* 17.3 (2013), str. 67–71. DOI: 10.1109/MIC.2013.43.

[26] Linux man-pages project. *poll(2) — Linux Programmer's Manual*. Section 2: System Calls. 2024. URL: https://man7.org/linux/man-pages/man2/poll.2.html.

[27] Andrew Loewenstern in Arvid Norberg. *DHT Protocol*. 2020. URL: https://www.bittorrent.org/beps/bep_0005.html (pridobljeno 28. 2. 2023).

[28] *Magnet Spider*. Torrent metasearch engine. URL: https://clzhizhu.com (pridobljeno 9. 7. 2026).

[29] Alexandre M Mateus in Jon M Peha. „Quantifying global transfers of copyrighted content using BitTorrent“. V: *39th Research Conference on Communication, Information and Internet Policy (TPRC)*. TPRC. 2011.

[30] MaxMind. *GeoLite2 Free Geolocation Data*. 2024. URL: https://dev.maxmind.com/geoip/geolite2-free-geolocation-data (pridobljeno 31. 7. 2024).

[31] Petar Maymounkov in David Mazieres. „Kademlia: A peer-to-peer information system based on the xor metric“. V: *Peer-to-Peer Systems: First International Workshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers*. Springer, 2002, str. 53–65.

[32] Alfred J. Menezes, Paul C. van Oorschot in Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and Its Applications. Boca Raton, FL: CRC Press, 1996. ISBN: 0-8493-8523-7.

[33] Abhijeet Mukherjee. *BTDigg: A Trackerless Torrent Search Engine*. 2011. URL: https://www.makeuseof.com/tag/btdigg-trackerless-torrent/ (pridobljeno 28. 2. 2023).

[34] Arvid Norberg. *DHT Security extension*. 2016. URL: https://www.bittorrent.org/beps/bep_0042.html (pridobljeno 28. 2. 2023).

[35] Arvid Norberg. *uTorrent transport protocol*. 2017. URL: https://www.bittorrent.org/beps/bep_0029.html (pridobljeno 28. 2. 2023).

[36] Arvid Norberg, Ludvig Strigeus in Greg Hazel. *Extension Protocol*. 2017. URL: https://www.bittorrent.org/beps/bep_0010.html (pridobljeno 28. 2. 2023).

[37] platelminto. *parse-torrent-title*. Python package, PyPI: parse-torrent-title. URL: https://github.com/platelminto/parse-torrent-title (pridobljeno 10. 7. 2026).

[38] Jon Postel. *Transmission Control Protocol*. RFC 761. §2.10, "Robustness Principle". DARPA, Information Sciences Institute, jan. 1980. URL: https://www.rfc-editor.org/rfc/rfc761.

[39] Antonio Prado in sod. *Live-Event Blocking at Scale: Effectiveness vs. Collateral Damage in Italy's Piracy Shield*. RIPE Labs. Sep. 2025. URL: https://labs.ripe.net/author/antonio-prado/live-event-blocking-at-scale-effectiveness-vs-collateral-damage-in-italys-piracy-shield/ (pridobljeno 9. 7. 2026).

[40] qBittorrent Project. *qBittorrent*. URL: https://www.qbittorrent.org (pridobljeno 10. 7. 2026).

[41] Myung-Ki Shin in sod. *Application Aspects of IPv6 Transition*. RFC 4038. Section 4.2. IETF, mar. 2005. URL: https://datatracker.ietf.org/doc/html/rfc4038#section-4.2.

[42] Juan Pablo Timpanaro in sod. „BitTorrent's Mainline DHT Security Assessment". V: *2011 4th IFIP International Conference on New Technologies, Mobility and Security*. 2011, str. 1–5. DOI: 10.1109/NTMS.2011.5721044.

[43] Matteo Varvello, Moritz Steiner in Koen Laevens. „Understanding BitTorrent: A reality check from the ISP's perspective". V: *Computer Networks* 56.3 (2012), str. 1054–1065. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2011.10.031. URL: https://www.sciencedirect.com/science/article/pii/S1389128611004488.

[44] Liang Wang in Jussi Kangasharju. „Measuring large-scale distributed systems: case of BitTorrent Mainline DHT". V: *IEEE P2P 2013 Proceedings*. 2013, str. 1–10. DOI: 10.1109/P2P.2013.6688697.

[45] Wikipedia contributors. *Hash function*. 2024. URL: https://w.wiki/An3L (pridobljeno 29. 7. 2024).

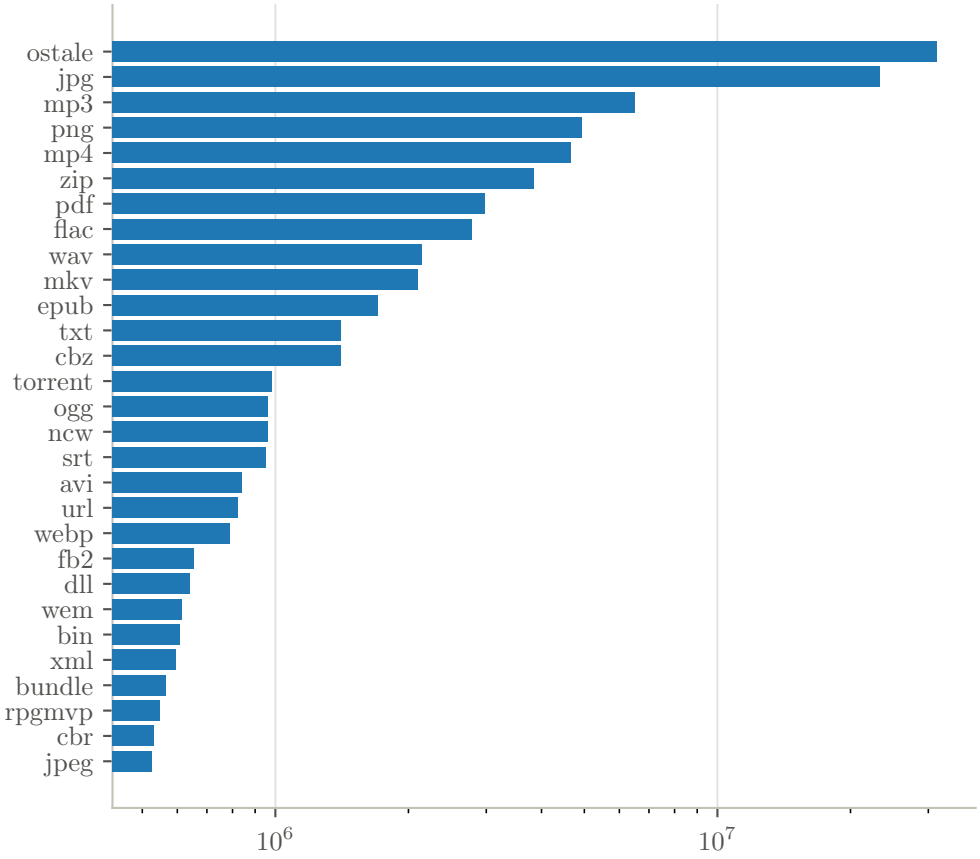
[46] Scott Wolchok in J Halderman. „Crawling BitTorrent DHTs for fun and profit“. V: *4th USENIX Workshop on Offensive Technologies (WOOT '10)*. Avg. 2010.

[47] Jackie Zhong. *The Past, Present, and Future of P2P Network: Applications and Challenges*. `http://www.cse.wustl.edu/~jain/cse570-21/ftp/zhong520/index.html`. Dec. 2021.

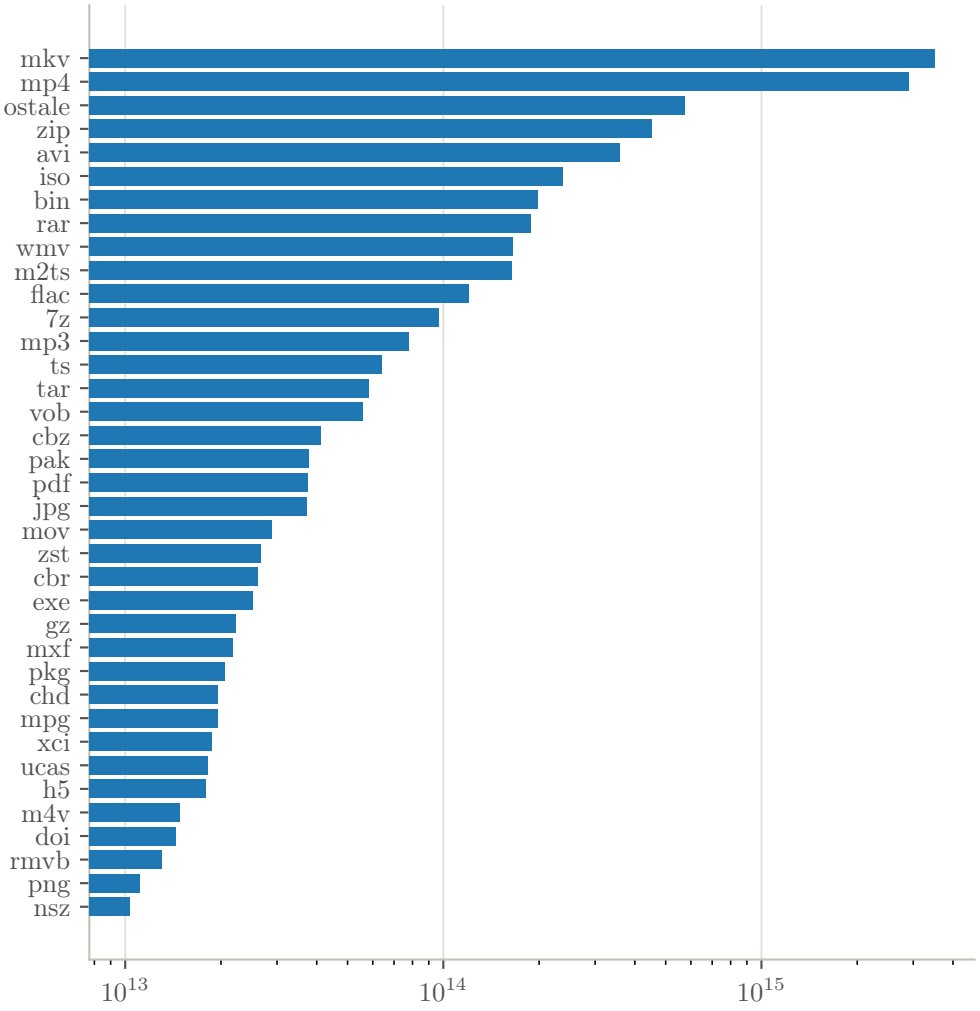
Poglavje 7

Priloge

Izvorna koda programa za zajem, ki smo ga implementirali, je dostopna na `http://ni.sijanec.eu/sijanec/travnik`. Korpus, na katerem smo izvajali analizo, pošljemo na zahtevo na elektronski naslov anton@sijanec.eu.



Slika 7.1: Tipi glede na število datotek v vseh torrentih.



Slika 7.2: Tipi glede na skupno velikost datotek v vseh torrentih.