Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Fakulteta za matematiko in fiziko

Anton Luka Šijanec

Zajem in analiza metapodatkov v omrežju BitTorrent s pomočjo porazdeljene razpršene tabele

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentorica: izr. prof. dr. Mojca Ciglarič

Ljubljana, 2026

Kandidat: Anton Luka Šijanec
Naslov: Zajem in analiza metapodatkov v omrežju BitTorrent s pomočjo porazdeljene razpršene tabele
Vrsta naloge: Diplomska naloga na interdisciplinarnem univerzitetnem študijskem programu prve stopnje računalništvo in matematika
Mentorica: izr. prof. dr. Mojca Ciglarič

Opis:
Opišemo osnove protokola BitTorrent in DHT. S sodelovanjem v omrežju DHT pridobivamo metapodatke o datotekah, ki se prenašajo s protokolom BitTorrent. Njihova analiza nam pokaže stanje omrežja: kategorije in legalnost vsebin v omrežju, odjemalce v uporabi, njihove različice, razširjenost IPv6 in popularnost BitTorrenta po državah. Razložimo tudi tehnične podrobnosti našega zajema.

Title: Acquisition and analysis of BitTorrent network metadata using DHT

Description:
We describe the BitTorrent and DHT protocols. By being part of the DHT network, we obtain metadata on files being downloaded with BitTorrent. Analysis of acquired data shows the state of the network: categories and legality of content in the network, clients in use and their versions, IPv6 adoption and BitTorrent popularity per country. We also explain some technical details about our acquisition methods.

Zahvaljujem se mentorici izr. prof. dr. Mojci Ciglarič za uporabne nasvete in odzivnost ter družini za vso podporo.

Kazalo

Povzetek

Naslov: Zajem in analiza metapodatkov v omrežju BitTorrent s pomočjo porazdeljene razpršene tabele

Avtor: Anton Luka Šijanec

Diplomsko delo predstavi protokol BitTorrent in njegov podporni sistem DHT (porazdeljena razpršena tabela) in se osredotoči na uporabo slednjega za pridobivanje metapodatkov iz omrežja BitTorrent. Za pridobivanje smo spisali namenski program, 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 tudi prenese. Na podlagi štirih zajemov v letih 2023, 2024 in 2026 (skupaj okoli milijon in pol torrentov) analiziramo stanje omrežja.

Ugotavljamo, da je najpopularnejši odjemalec qBittorrent, sledi μTorrent, katerega popularnost pada. Vodilne končnice imen datotek so po vrsti mp4, mkv, avi, mp3, zip in flac. Sodeč po imenih datotek je najpopularnejša ločljivost videovsebin 1080p, popularnost 480p pada, 2160p pa raste. Delež torrentov, prenesenih prek IPv6, raste, leta 2026 smo po IPv6 prenesli 17,5 % torrentov. Več kot polovico torrentov iz Mehike, Brazilije in Romunije smo prenesli preko IPv6, od slovenskih naslovov pa manj kot 5 %. Klasifikacija na kategorije z jezikovnim modelom je pokazala, da piratsko vsebino vsebuje 93 % torrentov, da je 28,9 % torrentov pornografija, sledijo TV oddaje oz. serije s 15,2 %, filmi s 14,7 % in glasba z 11,5 %.

Ključne besede: BitTorrent, DHT, P2P, Kademlia.

Abstract

Title: Acquisition and analysis of BitTorrent network metadata using DHT

Author: Anton Luka Šijanec

The thesis introduces the BitTorrent protocol and its supporting protocol, the DHT (distributed hash table), focusing on using DHT to acquire metadata from the BitTorrent network. We developed specialised software that records queries sent by other users' clients to our observation node in the DHT network. Such queries contain infohashes which are used to download metadata (names and sizes of files) from peers. From four independent acquisitions in the years 2023, 2024 and 2026, containing around a million and a half torrents, we infer the state of the BitTorrent network.

The most popular client is qBittorrent, followed by μTorrent, falling in popularity. Leading filename extensions are in order mp4, mkv, avi, mp3, zip and flac. The most popular video resolution according to filenames is 1080p, the popularity of 480p is falling and of 2160p is rising. The ratio of torrents received via IPv6 is rising; in 2026 we received 17.5 % of torrents via IPv6. More than half of torrents from Mexico, Brazil and Romania were received via IPv6, but less than 5 % of torrents received from Slovenian IP addresses were received via IPv6. Classification with a large language model shows that 93 % of torrents are for the purposes of piracy, that 28.9 % of torrents are pornography, followed by TV shows with 15.2 %, movies with 14.7 % and music with 11.5 %.

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 jih 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 [22, 18].

Eno izmed najstarejših še vedno aktivno uporabljenih porazdeljenih omrežij za izmenjavo datotek oziroma komunikacijo je protokol BitTorrent, katerega začetki segajo četrt stoletja v preteklost [6]. 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 te datoteke prenesti, s čimer linijo izvornega objavljalca razbremenijo.

V nalogi predstavimo, kako protokol DHT na osnovi protokola Kademlia [33], 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, ki se trenutno v omrežju prenašajo. Nadalje tudi analiziramo reprezentativen tako pridobljeni 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).

Glede na že izdelane analize prometa omrežja BitTorrent, pridobljene po drugačni poti [15], domnevamo, da bodo največji delež vsebin v omrežju predstavljale piratske videovsebine. Omrežje BitTorrent ljudje po raziskavah uporabljajo predvsem za ogled filmov brez ustreznih avtorskih pravic [15], za kar je omrežje dobro prilagojeno; filmi so velike datoteke, njihovo deljenje v omrežjih vsak z vsakim za objavljalce predstavlja manjši strošek, kot če bi se posluževali klasičnega modela strežnik-odjemalec [56], poleg tega pa so manj izpostavljeni organom pregona, saj je v primeru omrežij vsak z vsakim težje določiti, kdo je izvorni objavljalec neke datoteke, ker jo je moč prenesti od velikega števila naprav v internetu [42].

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

Ob prenesenih metapodatkih bomo shranjevali tudi naslove IP oddaljenih odjemalcev, od katerih metapodatke prejmemo, zato bomo lahko analizirali še razširjenost omrežnega protokola IPv6 in popularnost protokola BitTorrent po posameznih državah.

Vse opisano bomo preverili na podatkih, ki jih bomo zajeli iz omrežja.

1.2 Cilji dela

Nalogo sestavljata dva ključna cilja.

1. Razvili bomo 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. Zajeti vzorec metapodatkov vsebinsko ne sme biti 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 omrežnega protokola, naš zajem ne sme biti omejen samo na IPv4 ali samo na IPv6, ker bi to zaradi različno uspešnega prodora IPv6 po državah [5] kazalo popačeno sliko.

2. Pridobljene metapodatke bomo analizirali in na podlagi njih določili ključne značilnosti omrežja; kaj uporabniki prenašajo, kdaj, s kakšno programsko opremo itd.

1.3 Stanje področja in sorodna dela

Analize datotek, ki se prenašajo v omrežju BitTorrent, potekajo že vse od nastanka omrežja [31]. 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 [51]. Mi se osredotočamo na podatke, ki jih je moč pridobiti brez dostopa do dnevniških in analitičnih datotek na večjih sledilniških strežnikih.

Ta naloga se osredotoča na pridobivanje metapodatkov datotek z uporabo *Mainline* DHT (Kademlia), ki je dandanes *de facto* implementacija DHT-ja za BitTorrent. Pred trdno uveljavitvijo/standardizacijo tega protokola DHT

je podporo za DHT kot prvi tak odjemalec dobil odjemalec, danes znan kot Vuze [25]. O prečesavanju Vuze DHT za analizo omrežja pišejo v članku [55].

Omrežje *Mainline* DHT preiskujejo tudi mnogi za omrežje invazivnejši programi od našega [48]. Njihov prvotni namen ni pridobivanje vzorca za analizo, temveč izdelava zbirnega iskalnika po datotekah, ki jih je moč prenesti z odjemalcem za torrente. Nekateri trenutno dostopni iskalniki vsebin torrent, ki jih poganjajo preiskovalci omrežja *Mainline* DHT, so BTDigg [35, 13], bitmagnet [16] in clzhizhu (magnetni pajek) [30].

BTDigg je zasnovan za izdelavo iskalnika po datotekah, njegov namen ni analiza datotek po kategorijah. Bitmagnet se, kot naš program, trudi biti za omrežje neinvaziven, vendar še ni obstajal, ko smo mi že izvajali prve zajeme z našim programom. Clzhizhu je samo kitajska spletna stran z iskalnikom, program, ki ga uporabljajo za zajem, pa na njihovi strani ni naveden – najbrž njegova koda ni javna.

Poglavje 2

Metodologija

2.1 Kratek opis protokola BitTorrent

V tem razdelku opišemo, kako deluje protokol BitTorrent, vendar le toliko, kolikor je potrebno za razumevanje nadaljnjega besedila. Ne spuščamo se v podrobnosti o prenosu koščkov (vsebine datotek), ker tega v našem delu ne počnemo. Razložimo le podrobnosti o vzpostavitvi povezave med soležniki in prenosu metapodatkov od soležnikov.

Datoteka torrent (krajše torrent) vsebuje metapodatke o končni fiksni množici nespremenljivih datotek. Kdor želi prek BitTorrenta 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 [11]) 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) tudi imeti. Njena struktura je naslednji bkodiran slovar:

- `announce`: 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 [19]. 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 `/`.

Primer datoteke torrent, ki smo jo prenesli tekom zajema leta 2026 (odstranjen ključ `pieces`, ki vsebuje zgoščene vrednosti koščkov), je v izseku kode 1. Zavoljo berljivosti smo vsebino datoteke namesto v bkodirani obliki pisali v obliki JSON.

Izsek kode 1: Primer datoteke torrent, zajete leta 2026, z infohashem, ki se začne z `8387dddf9f`. Od soležnika je prenesen samo slovar `info`, ostale vrednosti je dodal naš program za zajem.

```
{
  "creation date": 1775704626, "encoding": "UTF-8",
  "info": {
    "files": [
      { "length": 1566287872, "path": [ "Chelovek_slon.avi" ] },
      { "length": 177408768, "path": [ "Chelovek_slon_eng.ac3" ] },
      { "length": 66480, "path": [ "Chelovek_slon_eng.srt" ] },
      { "length": 64876, "path": [ "Chelovek_slon.srt" ] }
    ],
    "name": "Chelovek_slon", "piece length": 2097152
  },
  "source": {
    "ip": "::ffff:95.221.176.<anonimizirano>/29936",
    "v": "MediaGet2 3.01.4333"
  }
}
```

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 $\left[\left\lfloor \frac{v_0}{L} \right\rfloor, \left\lfloor \frac{v_0 + v_1 - 1}{L} \right\rfloor \right]$.

Ker so v datoteki torrent navedene zgoščene vrednosti po koščkih, ne po manjših enotah, ne moremo overiti krajših zaporedij bajtov od dolžine koščka, čim pa pridobimo celoten košček in preverimo ujemanje zgoščene vrednosti,

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

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 datoteki 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 soležnik 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 Izmenjava 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 [7]. 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 poznamo 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 [7].

Podprtost razširitvenega protokola, katerega del je razširitev za prenos metapodatkov [20], 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 drug drugemu sporočita, katere razširitve podpirata [38].

Specifična razširitev za prenos metapodatkov se imenuje `ut_metadata` in je standardizirana z dokumentom BEP-0009 [20]. 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 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.

[2]Obstaja sicer tudi novejši protokol μTP, ki omogoča učinkovitejše prenašanje velikih količin koščkov prek UDP [37], vendar se v nalogi nanj ne osredotočamo, saj prenašamo relativno malo – samo metapodatke.

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 [20].

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 infohashem tega torrenta.

2.2 Porazdeljena razpršena tabela

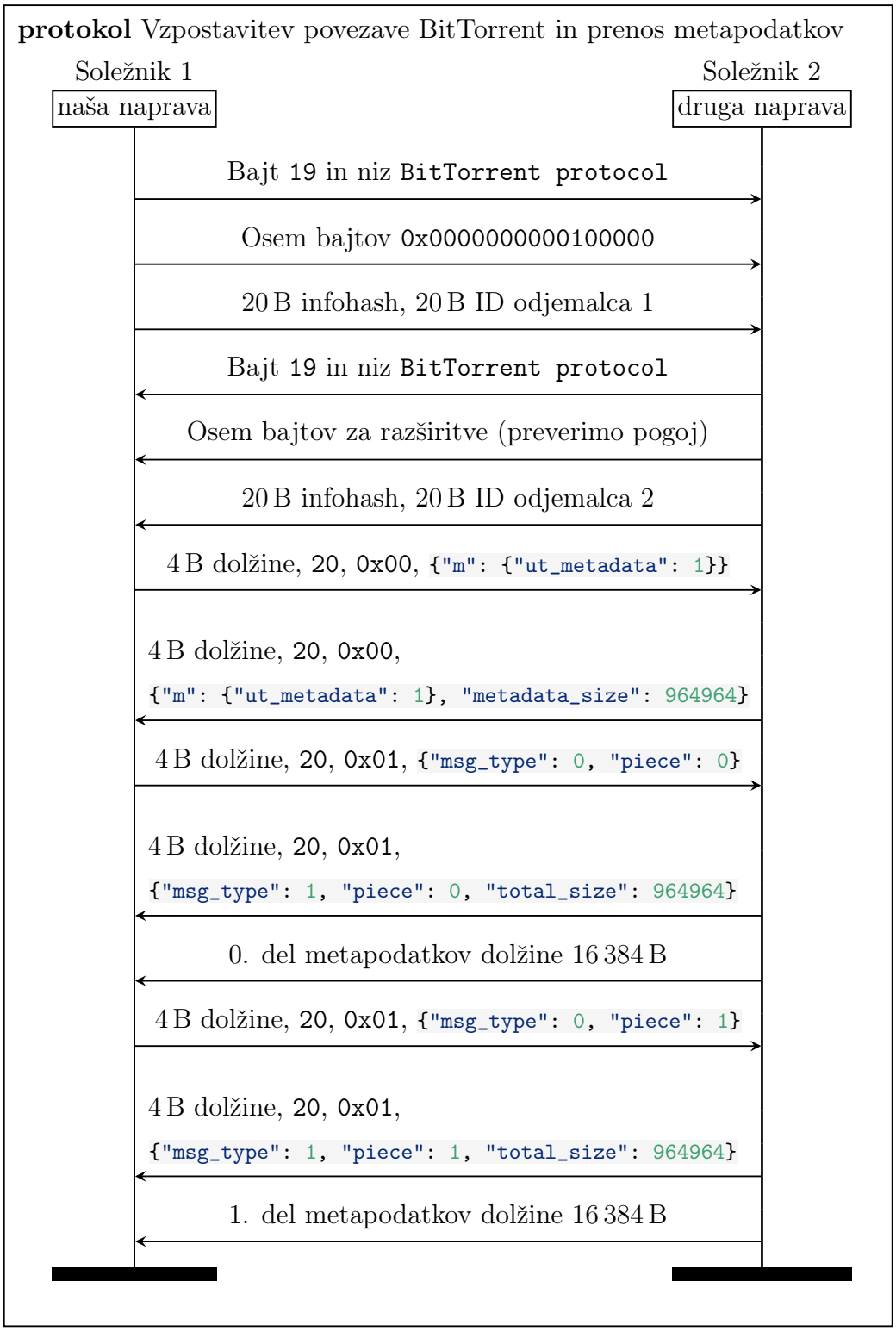
Porazdeljena razpršena tabela (DHT) je dinamičen slovar (slika 2.2). Pod ključem, ki predstavlja infohash torrenta, hrani množico soležnikov. Na najvišjem nivoju podpira dve operaciji:

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

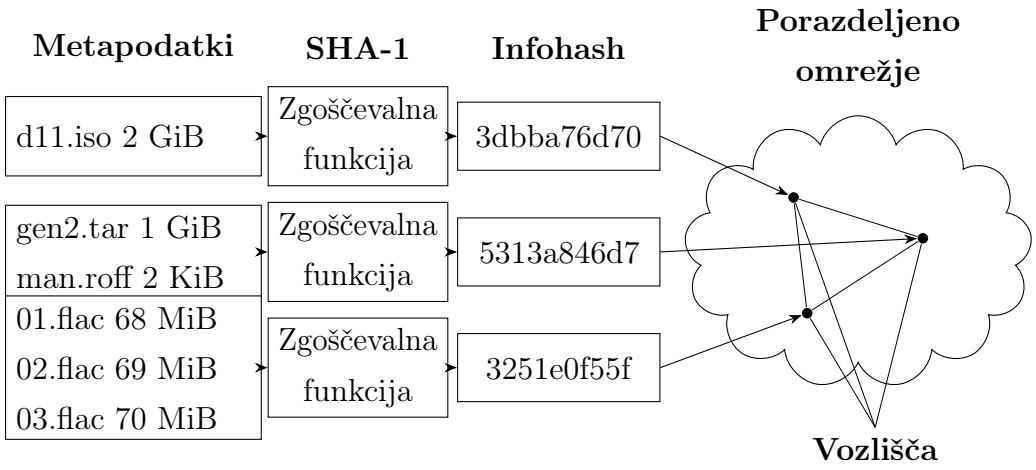
Uporabnik v porazdeljenem omrežju DHT, ki izvede te operacije, s tem po zasnovi DHT poizvedbe pošlje prek veliko drugih uporabnikov. V tej nalogi

[3] V drugi različici protokola BitTorrent [8] 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 [20].



Slika 2.2: Shematski prikaz uporabe DHT v BitTorrentu [24].

torej kot pasivni uporabnik omrežja DHT pridobimo veliko poizvedb, ki vsebujejo infohashe torrentov. Ko enkrat dobimo infohashe, 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 [29].

ID vozlišča mora biti izbran enakomerno naključno [1] na $\{0,1\}^{160}$ in mora biti shranjen na disku – 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 sta 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 kadar 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 na 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 ujemajo v $i + 1$ in več bitih.

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

[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 2.6

[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.

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

Dokaz. Dokažimo lastnosti metrike:

- definitnost: $\forall a, b \in \{0,1\}^{160} : d_{\text{XOR}}(a,b) = 0 \Leftrightarrow a = b$. 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: $\forall a, b \in \{0,1\}^{160} : d_{\text{XOR}}(a,b) = d_{\text{XOR}}(b,a)$. 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

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

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

□

Ta BitTorrentov DHT deluje na osnovi 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 metriki XOR precej oddaljena od njega, in dosti več takih, ki so po definirani metriki blizu njega.

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:

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

- `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 *transakcijo*. 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 metriki XOR iskalnemu nizu najbližjih vozlišč (glede na ID) iz usmerjevalne tabele prejemnika poizvedbe. V nizu si vozlišča sledijo drug 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 [3].

  - `get_peers`: Zahteva vsebuje argument `info_hash` z 20-bajtnim infohashem 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 infohashem, pod ključem `nodes` in `nodes6` pa take podatke, kot bi jih vrnil klic `find_node` s parametrom `target`=`info_hash`. Odgovor vsebuje 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 [12] vpisali kogarkoli razen sebe v roj, ter ključa `info_hash` z 20-bajtnim infohashem 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 **oznani** (postopek 1). Zaradi takega postopka se podatki o roju nekega torrenta z infohashem 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

Postopek 1 Postopek za pridobivanje soležnikov torrenta z infohashem h in oznanjanje.

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

Zagotovitev, dosežena tekom postopka: T_h so soležniki torrenta h, V_h so živa vozlišča blizu torrenta h (opomba: postopek teče, dokler je torrent dodan v odjemalec, množici se posodabljata tekom postopka)

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: $\mathrm{announce\_peer}_n(h, z)$
15: **end if**
16: **end for**
17: **end loop**
18: **end procedure**

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-ju. Prav tako običajno resnične implementacije oznanjajo le nekaj h-ju najbližjim vozliščem v V_h in ne vsem [4]. Cilj *Mainline* DHT je informacije o soležnikih torrenta h hraniti le na vozliščih z ID-jem blizu h, ker jih bodo iskalci soležnikov iskali prav tam.

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 [29].

- 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 tako ali tako 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 Standard za kodiranje struktur bkodiranje

Večina struktur (datoteke torrent, paketi DHT idr.) je serializirana v namensko razvito obliko bkodiranje (angl. *bencoding*). Le-ta omogoča serializacijo preprostih podatkovnih tipov, ki se uporabljajo v BitTorrentu, v zaporedje 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 drug 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 drug 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 za večjo preglednost strukture pišemo v obliki JSON [2], ker je bolj berljiva od bkodiranja.

2.4 Uporabljena metoda za pridobivanje metapodatkov

Kot je razvidno iz postopka 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[10], 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 infohashev torrentov. Nato se bomo prav s

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

pomočjo DHT vpisali v roje teh najdenih torrentov s postopkom 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[11] iz roja za ta torrent in tudi tega torrenta v bodoče več ne bomo prenašali.

Podatke o datotekah 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 metriki XOR blizu ID-ja našega opazovalnega vozlišča, kar je na prvi pogled v nasprotju s ciljem o reprezentativnosti podatkov. K sreči zgoščevalna funkcija SHA-1, uporabljena za računanje infohasha, zgoščene vrednosti ne računa 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 [34, 54]. 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. 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 predstavljeni način zajema ustreznejši kot denimo pasivno prisluškovanje povezavam na usmerjevalnikih omrežnih operaterjev, ki bi bilo precej geografsko zaznamovano – najbrž bi zaznali precej več torrentov, ki jih prenašajo lokalni uporabniki.

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

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.

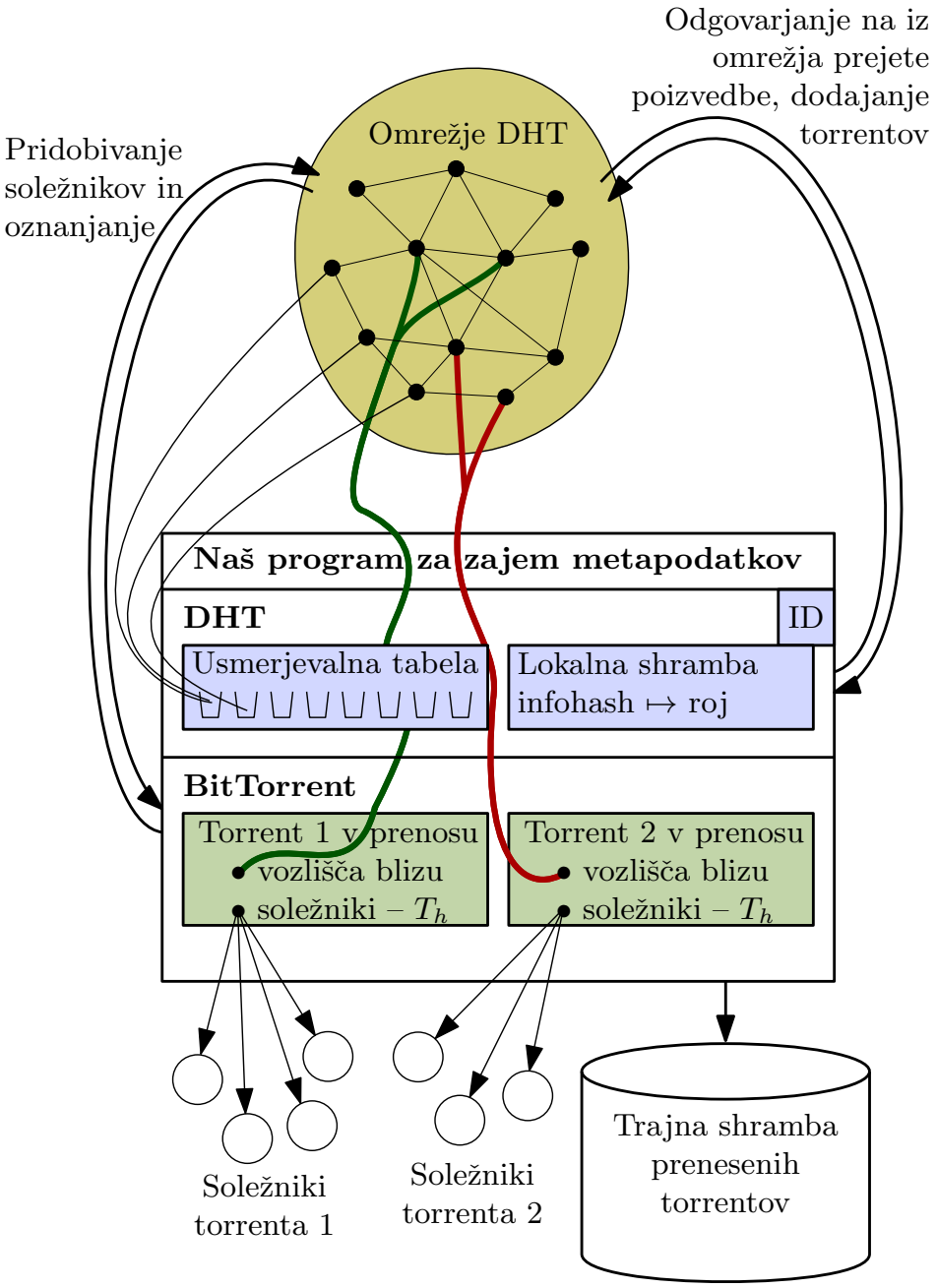
2.5 Zasnova programa in njegova implementacija

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, kar bi pomenilo, da nam ne bi več pošiljala poizvedb in nas ne bi več oglaševala kot aktivno vozlišče v odgovorih na prejete poizvedbe tretjih vozlišč.

Vse v nalogi opisane protokole bomo implementirali ročno, da lahko nadziramo vse korake njihovega izvajanja in jih prilagodimo za zajem metapodatkov, ne za prenos datotek, kot to seveda pričakovano počnejo knjižnice BitTorrent, uporabljene v odjemalcih. 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 zavoljo lažje implementacije ne obravnava prejetih napak 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 [41]), bomo spisali v programskem jeziku c.

Shematski prikaz našega programa za zajem je na sliki 2.3.

Ker program ni računsko obremenjen, pač pa vhodno/izhodno, v celoti teče v eni niti, uporablja dogodkovno zanko in sistemskih klicev ne uporablja



Slika 2.3: Shematski prikaz našega implementiranega programa med delovanjem.

na način, ki bi blokiral izvajanje niti. Jedro izvajanja je dogodkovna zanka, implementirana s sistemskim klicem `poll(2)` [28]. 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 in pri vozliščih v njej podatek o času zadnje uspešne vzpostavitve stika
- 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

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

2.5.1 Knjižnica za bkodiranje

Ker je bkodiranje po podatkovnih tipih podobno JSON-u, smo knjižnico zasnovali približno tako, kot je zasnovana cjevska knjižnica cJSON [14] za delo z JSON-om. Vsak tip je predstavljen s strukturo (koda 2), 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 2: Struktura vsakega elementa v knjižnici za bkodiranje. Nekatere interne komponente so zaradi berljivosti izpuščene.

```c
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 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

2.5.2 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 infohashev in nekaj funkcij

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 3 je navedenih nekaj struktur za boljše razumevanje sestave knjižnice. Vsa komunikacija knjižnice poteka prek sklada IPv6, za povezavo v omrežje IPv4 se poslužuje V4MAPPED [46].

Izsek kode 3: Nekaj struktur iz knjižnice za DHT in BitTorrent. Nekatere interne komponente in strukture so zaradi berljivosti izpuščene.

```
1  struct node { // opisnik oddaljenega vozlišča
2          unsigned char id[20];
3          struct sockaddr_in6 addr; // ::ffff:0.1.2.3 za IPv4 (V4MAPPED)
4          int unanswered; // neodgovorjenih zahtev od zadnjega paketa
5          time_t last_received; // čas prejema zadnjega paketa od vozlišča
6          time_t last_sent; // čas pošiljanja zadnjega paketa
7  };
8  struct bucket { // opisnik koša v usmerjevalni tabeli
9          unsigned char id[20]; // koš vsebuje [id, next->id)
10         struct node * nodes;
11 };
12 enum state { // stanja povezave TCP za BitTorrent
13         blank = 0,
14         handshake_sent = 1,
15         handshake_received = 1 << 1,
16         extension_sent = 1 << 2,
17         extension_received = 1 << 3,
18         incoming = 1 << 4,
19         outgoing = 1 << 5
20 };
21 struct torrent { // opisnik torrenta ne nujno v prenosu
22         unsigned char ut_metadata; // podtip oddaljenega soležnika
23         unsigned char ut_pex; // podtip oddaljenega soležnika
24         enum state state; // stanje TCP povezave
25         void (* disconnection)(struct torrent *); // signal uporabniku, da
           ↪ shrani metapodatke v datoteko
26         unsigned char hash[20]; // infohash
27         struct peer * peers; // množica T_h
28         struct node * nodes; // množica V_h
29         int progress; // koliko delov metapodatkov imamo
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 metriki XOR
- `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 idr.)
- `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

2.5.3 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 [26] in pri tem uporabljali slednjo knjižnico.

2.6 Med testiranjem zaznani in odpravljeni težavi

Med testiranjem našega programa smo naleteli na dve težavi, ki smo ju pred zajemanjem podatkov o datotekah za analizo odpravili. Programa nismo spreminjali iz zajema v zajem, vse zajeme smo delali z enako različico programa.

2.6.1 Napad Sybil

Vozlišča si po standardu DHT Kademlia ID-je res izbirajo naključno, vendar omrežje tega v osnovni obliki[12] nikakor ne zagotavlja. Vozlišče si lahko ob zagonu izbere poljuben ID, ga med 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.

Gre za napad Sybil [9], 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 omejeno izdelovanje 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 [49, 52]), 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

[12]Obstaja ukrep, ki napade Sybil malce oteži [36], 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 reprezentativnega vzorca populacije torrentov smo se torej odločili, da BEP-0042 ne implementiramo.

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.

S takim delovanjem naš program, tudi če postane tarča napada Sybil, vseeno lahko nadaljuje z zajemanjem. Ta težava je z opisanim postopkom za nas rešena.

2.6.2 Ozko grlo omrežne opreme

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

Težavo smo odpravili tako, da smo vse naslednje zajeme izvajali na boljši omrežni opremi, torej na strežnikih v Grčiji (GRNET) [27], strežnikih na Fakulteti za računalništvo in informatiko UL in na rezidenčnem priključku z boljšim modemom GPON. Težava se je namreč pojavila le na strežniku na rezidenčnem priključku pri operaterju T-2, pa še to le v primeru slabe terminalske opreme (modema GPON).

Poglavje 3

Zajeti podatki in rezultati

Podatke o datotekah smo z našim izdelanim programom zajemali v štirih zajemih, prikazanih v tabeli 3.1. Skupna velikost prenesenih datotek torrent znaša 78,8 GiB. Velikost datotek v tabeli se nanaša na datoteke, ki jih vsebujejo torrenti. Teh datotek seveda nismo prenašali, samo metapodatke o njih. Njihova imena in dolžine so nam znane, vsebina pa ne.

Tabela 3.1: 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.

obdobje	lokacija	torrentov	datotek	p
2023-01–2023-02	T-2 GPON	47 843	3 084 321, 259 TiB	29 s
2023-03–2023-05	~okeanos [27]	414 620	17 165 425, 1891 TiB	6 s
2024-02–2024-02	T-2 GPON	62 108	3 725 125, 344 TiB	7 s
2026-04–2026-07	FRI	1 021 992	80 886 646, 6837 TiB	7 s

Podatke smo analizirali v okolju Jupyter Notebook [26] in grafe risali s knjižnico matplotlib [21]. 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.

3.1 Tipi datotek, ki se prenašajo s protokolom BitTorrent

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 3.1). Alternativna načina merjenja bi bila število datotek z nekim tipom (slika A.1) in skupna velikost datotek s tem tipom (slika A.2), ki odgovarjata na drugačno vprašanje, 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 prostora na disku, torej recimo filmom.

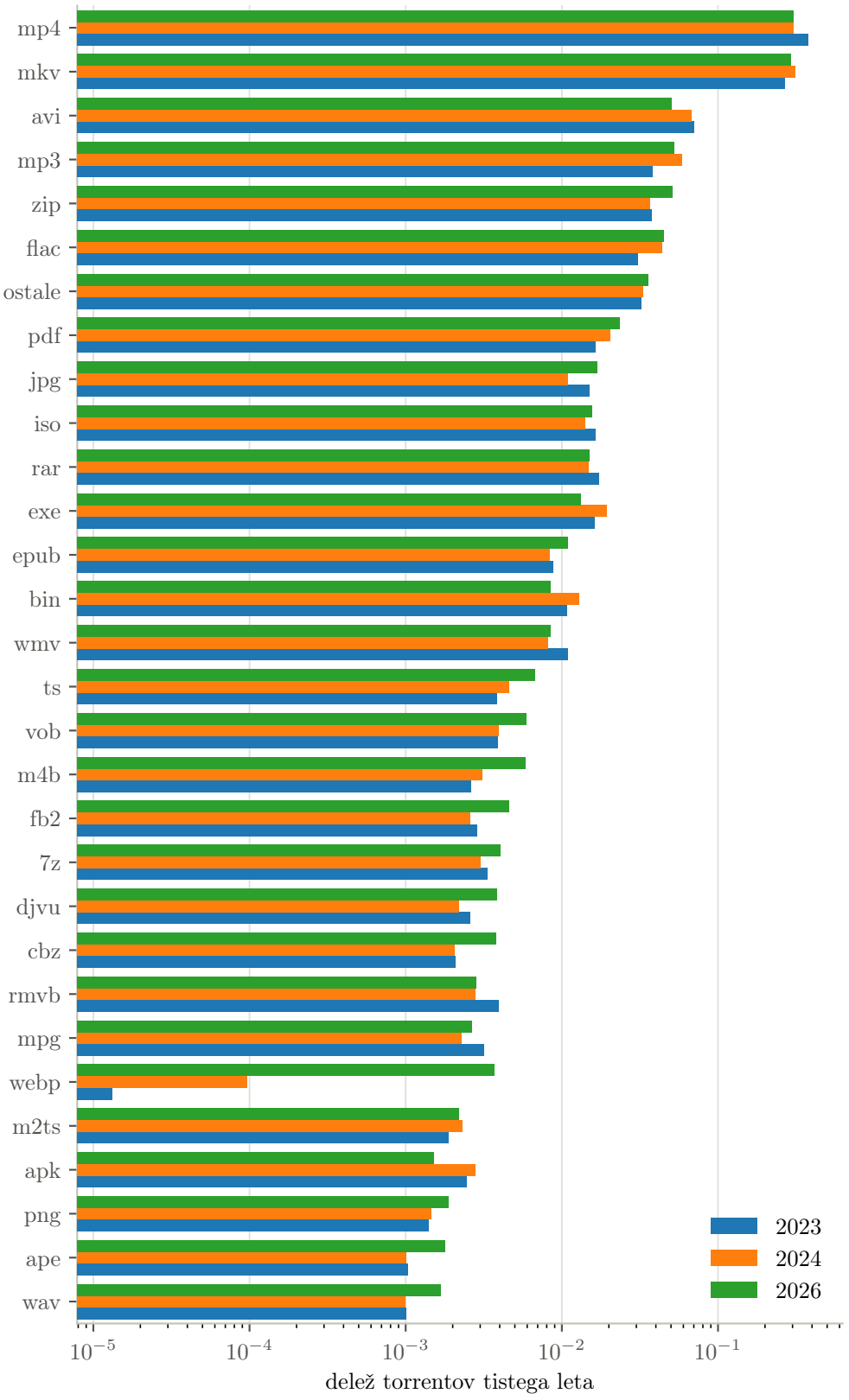
Opazimo rast uporabe oblike webp od leta 2023 do leta 2026. Na spletu opazimo sunkovito rast zanimanja za webp oktobra 2022, ko je iskalnik Google začel privzeto uporabljati webp za vse slike [17]. Velik del torrentov v 2026 s slikami v obliki webp predstavljajo stripi, izdelani s pogonom Ren'Py [45].

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 vsebujejo.

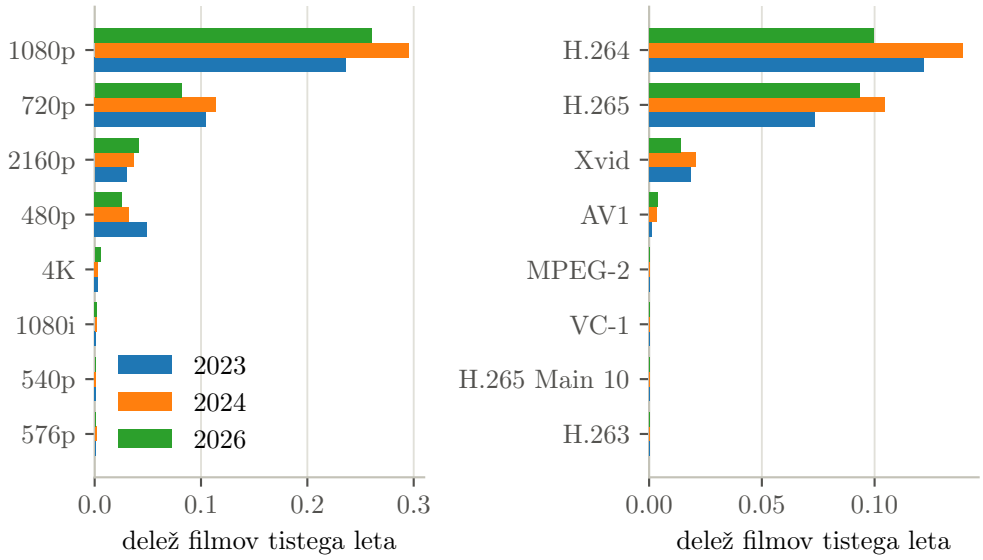
Kot pričakovano je najpogostejša vrsta datotek videovsebina, zato smo s pomočjo pythonske knjižnice [40] kategorizirali vse videe glede na kodek in ločljivost (slika 3.2). Žal podatki temeljijo zgolj na metapodatkih (imenih) in ne na vsebini datotek, torej izpustimo znaten delež populacije, ki kodeka ali ločljivosti v imenu nima vpisanega.

3.2 Opredelitev legalnosti vsebine in kategorizacija z jezikovnim modelom

V članku [53] iz leta 2011 so delež piratske vsebine v omrežju BitTorrent ocenili na 97 %. Podobno analizo smo avtomatizirano izdelali tudi na našem korpusu. Z odprtim jezikovnim modelom Qwen3.6-35B-A3B [44], ki smo ga z



Slika 3.1: Frekvenčna porazdelitev tipov datotek glede na število torrentov, pri katerih tip predstavlja glavnino velikosti.



Slika 3.2: Ločljivosti in kodeki videovsebin iz imen datotek.

orodjem Ollama [39] poganjali na Arnesovi superračunalniški gruči na dveh grafičnih karticah Tesla V100S, smo 2799 torrentom (vsem tistim iz zajema leta 2026, katerih infohash se začne z bajtom `0x4a`) priredili oznako legalnosti in oznako tipa vsebine izmed tipov, ki so jih uporabili v [53], da lahko dobljene rezultate primerjamo. Deleže torrentov po kategorijah prikazuje tabela 3.2. Uporabljeni poziv je v prilogi v izseku kode 5. Za klicanje modela smo uporabili program `pi` [10].

Pri 441 torrentih (15,76 %) je bil izhod modela v neustrezni obliki. V tem primeru smo isti pogovorni seji dodali vnovičen poziv, naj se izhod popravi (izsek kode 6).

Za validacijo smo 119 torrentov, ki se jim infohash začne s polbajti `0x4ae`, klasificirali ročno. Rezultat ročne klasifikacije je dostopen v prilogi v izseku kode 4.

Po legalnosti smo torrente z jezikovnim modelom razločili v skupine, prikazane v tabeli 3.3.

Tabela 3.2: Torrenti iz vzorca velikosti 2799 po kategorijah, določenih z jezikovnim modelom. Za lažjo primerjavo smo izbrali enake kategorije, kot so jih izbrali v članku [53].

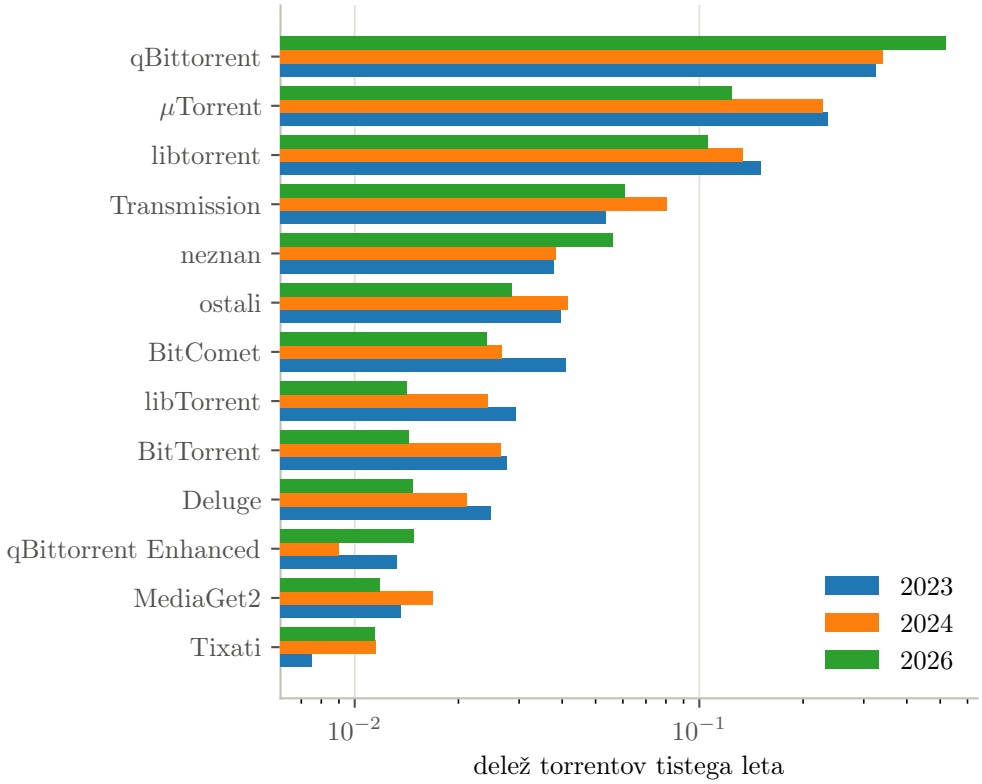
kategorija	število torrentov	delež
anime	263	9,40 %
book	175	6,25 %
childporn	23	0,82 %
documentary	28	1,00 %
game	107	3,82 %
hentai	86	3,07 %
movie	383	13,68 %
music	321	11,47 %
pictures	35	1,25 %
porn	700	25,01 %
software	114	4,07 %
tvshows	426	15,22 %
other	101	3,61 %
unknown	21	0,75 %
various	14	0,50 %
error	2	0,07 %

Tabela 3.3: Torrenti iz vzorca velikosti 2799 po legalnostih, določenih z jezikovnim modelom.

legalnost	število torrentov	delež
piracy	2603	93,00 %
legitimate	66	2,36 %
illegal	52	1,86 %
unknown	76	2,72 %
error	2	0,07 %

3.3 Najpopularnejši odjemalci za BitTorrent

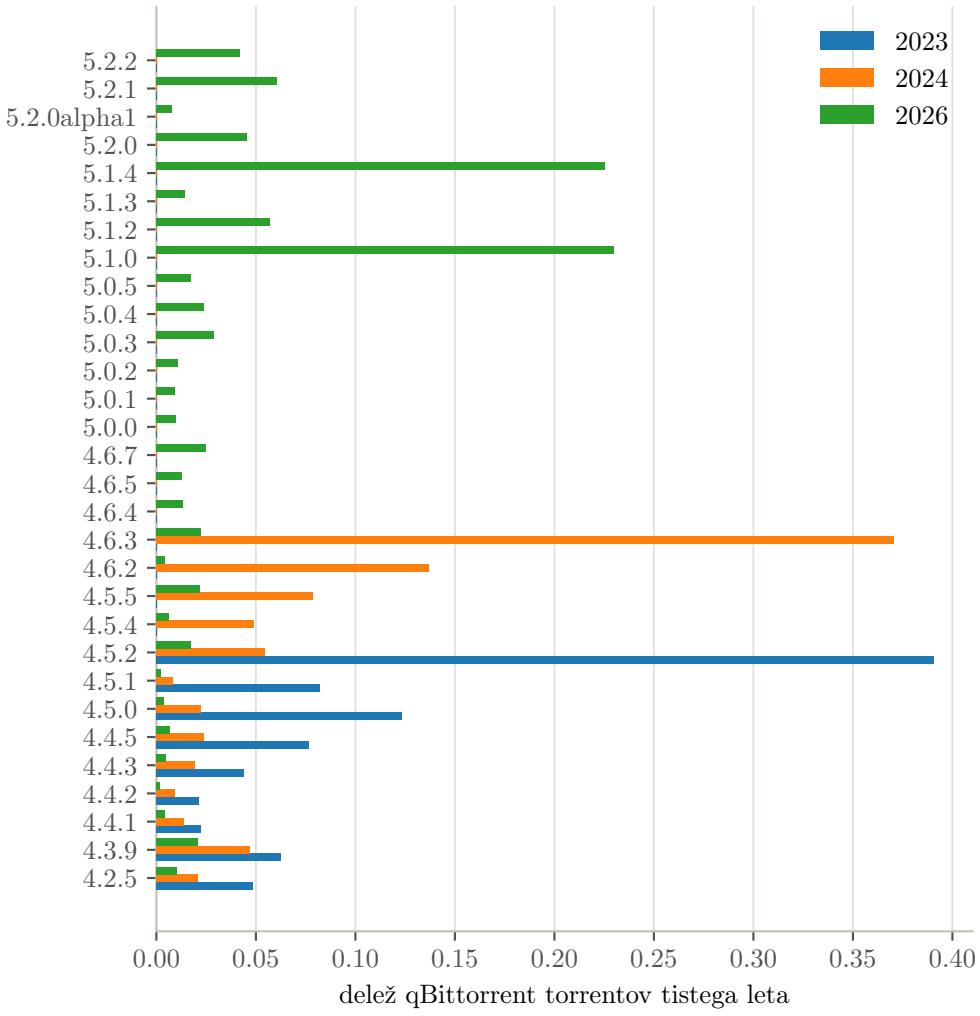
Za analizo uporabe odjemalcev smo za vsako posamezno leto primerjali zastopanost odjemalca (brez različice) v prejetih torrentih. Rezultati so prikazani na sliki 3.3. Najpopularnejši odjemalec je odprtokodni qBittorrent, sledi mu μTorrent. Domnevamo, da popularnost slednjega pada zato, ker ni odprtokoden in v brezplačni različici vsebuje oglase.



Slika 3.3: Popularnost odjemalcev po letih.

3.4 Uporabljene različice programske opreme odjemalcev za BitTorrent

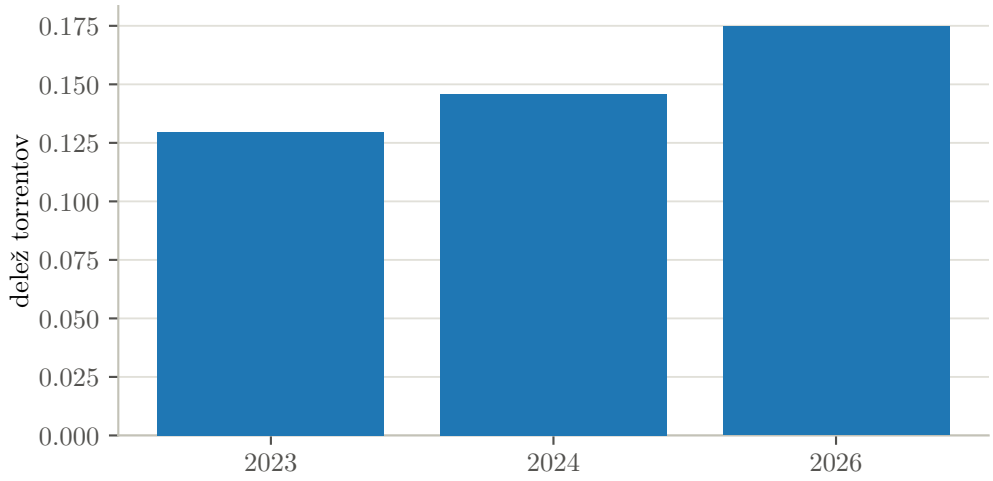
Osredotočili smo se na odjemalca qBittorrent [43], saj je najbolj zastopan v prenesenem korpusu. Pojavnost najdenih različic skozi leta (slika 3.4) potrjuje domnevo, da bodisi uporabniki res posodabljajo programsko opremo bodisi v omrežje prihajajo novi uporabniki.



Slika 3.4: Različice programske opreme qBittorrent skozi leta.

3.5 Razširjenost IPv6 glede na vir metapodatkov

Na sliki 3.5 je opazna rast razširjenosti IPv6 skozi leta. Ocenimo jo kot razmerje med torrenti, prenesenimi prek IPv6, in vsemi prenesenimi torrenti. Podobne podatke o prenosu torrentov glede na omrežni protokol za vsako državo posebej smo prikazali na sliki 3.6. Vse podatke smo zajemali na lokacijah s povezavo v IPv6 in IPv4.

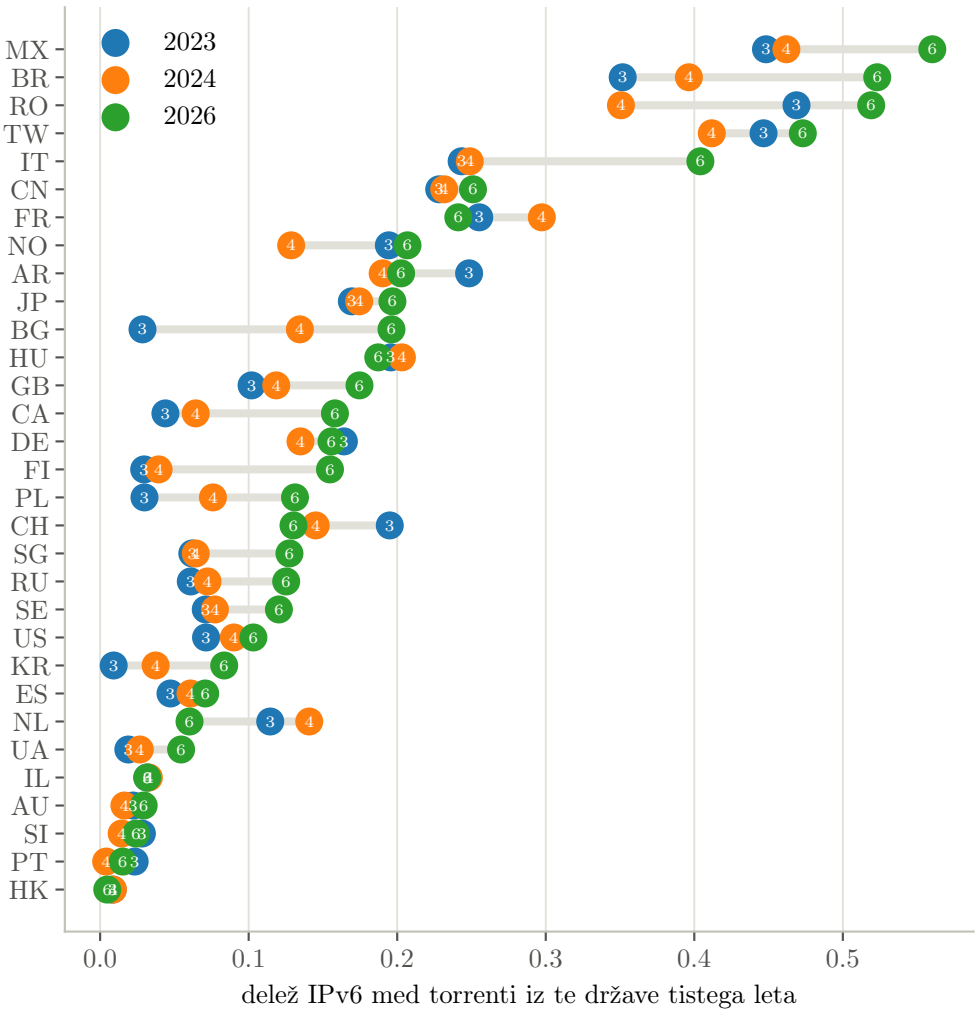


Slika 3.5: Delež torrentov, ki smo jih prejeli preko IPv6, v posameznem letu, nakazuje postopno širjenje omrežja IPv6.

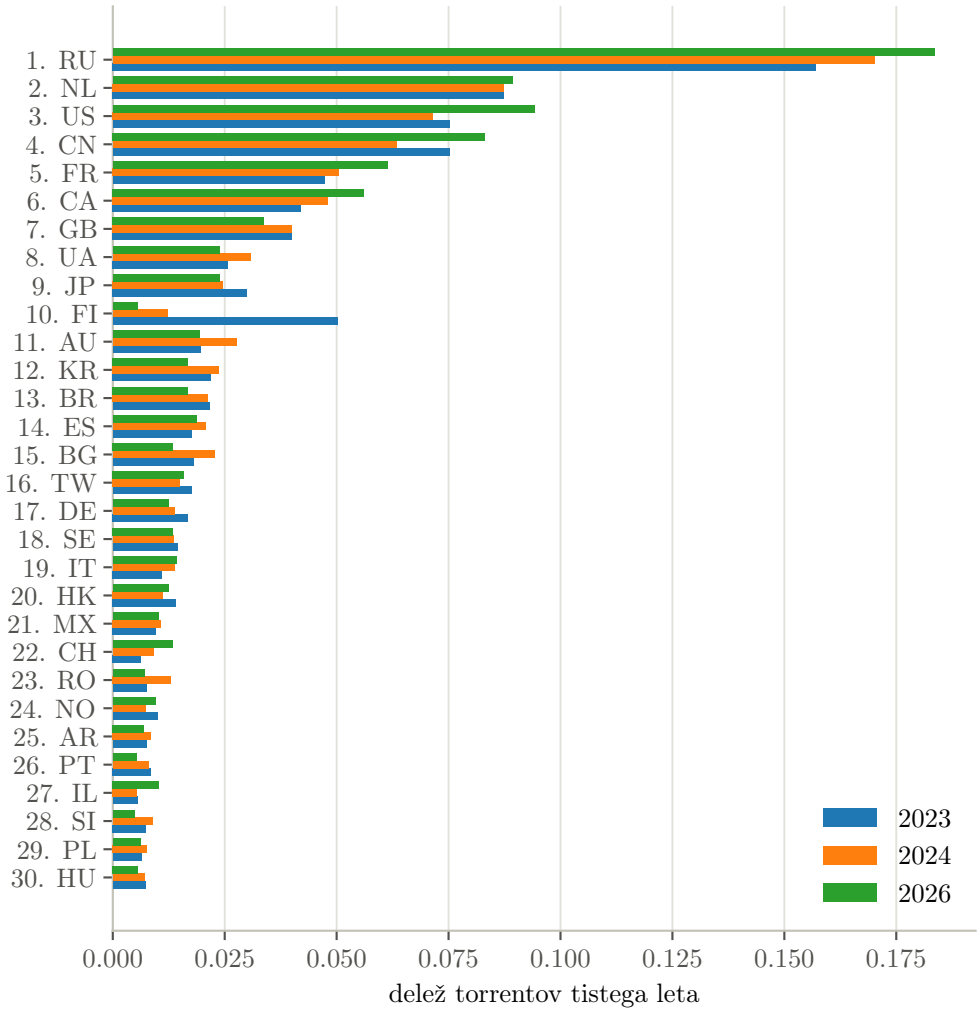
3.6 Popularnost BitTorrenta po državah

Z uporabo podatkovne zbirke [32] 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 3.7).

Podatkovno zbirko, ki smo jo uporabili za določanje države, smo prenesli julija 2026, zato mogoče vsebuje netočne podatke za naslove iz 2023, saj



Slika 3.6: Rast uporabe IPv6 skozi leta po državah. Prikazan je delež po IPv6 prenesenih torrentov z naslovov IP posameznih držav skozi leta. Prikazane so tiste države, od katerih smo prejeli največ torrentov.



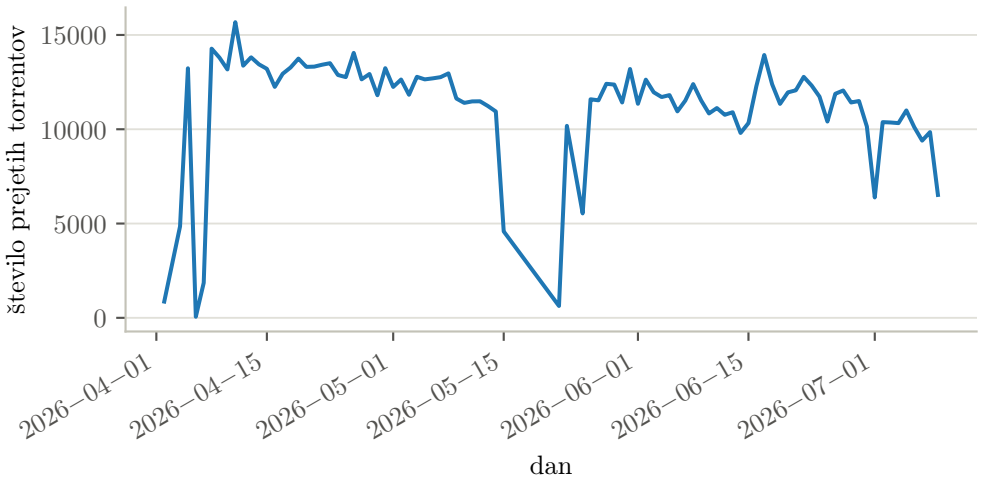
Slika 3.7: Primerjava držav, iz katerih so odjemalci posredovali največ metapodatkov. Oznake držav po [23].

internetni naslovi, čeprav redko, lahko menjajo državo.

Opazimo, da je bilo veliko torrentov prejetih s Finske v letu 2023, kasneje pa se je število bistveno zmanjšalo. Podroben pogled pokaže, da je 72 % torrentov, prejetih s Finske, z vsebino istega vira oziroma skupine ljudi. Prejeti so bili iz samo osmih različnih naslovov IP. Nalagalec sodeč po nedostopnih domenah iz imen torrentov ne nalaga več oziroma ne vzdržuje svojih najetih strežnikov na Finskem.

3.7 Količina prejetih torrentov na dan

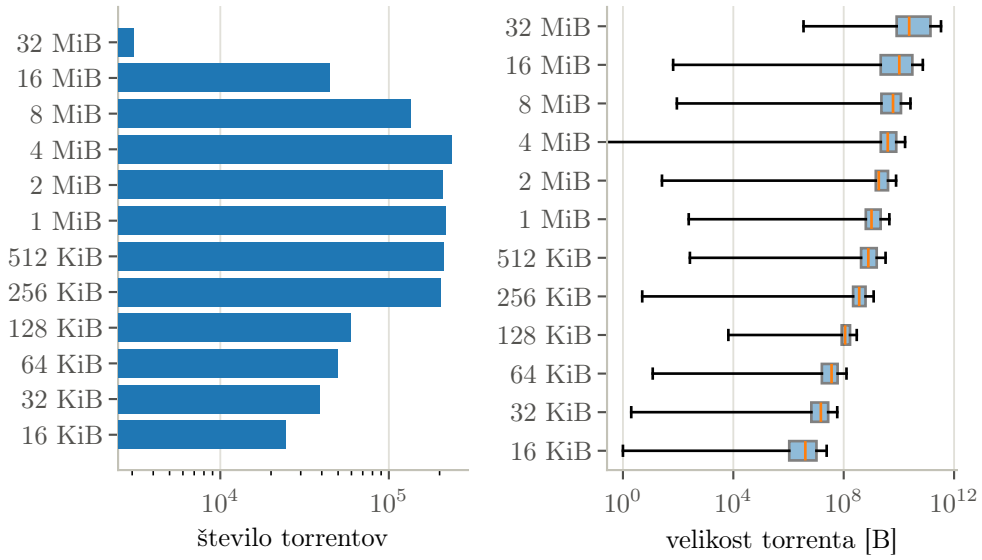
Graf na sliki 3.8 prikazuje, koliko torrentov na dan smo prejeli med zajemom leta 2026.



Slika 3.8: Količina zajetih torrentov na dan v zajemu leta 2026. Zajem smo na začetku nekajkrat prekinili, vidna je tudi ena večdnevna prekinitev.

3.8 Tipične velikosti koščka v datotekah torrent

Izvorna različica BitTorrenta, kot smo omenili, v torrentih vsebuje zgoščene vrednosti koščkov dolžine, določene v datoteki. Na sliki 3.9 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 3.9: Porazdelitev določene velikosti koščka in primerjava izbrane velikosti koščka z velikostjo celotnega torrenta za nekaj najpopularnejših velikosti koščka.

3.9 Rekordni primerki: velike datoteke, naslovi IP, vrata TCP

Tabeli 3.4 in 3.5 prikazujeta največje odkrite datoteke v Sloveniji in največje odkrite datoteke nasploh.

Tabela 3.4: Deset 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

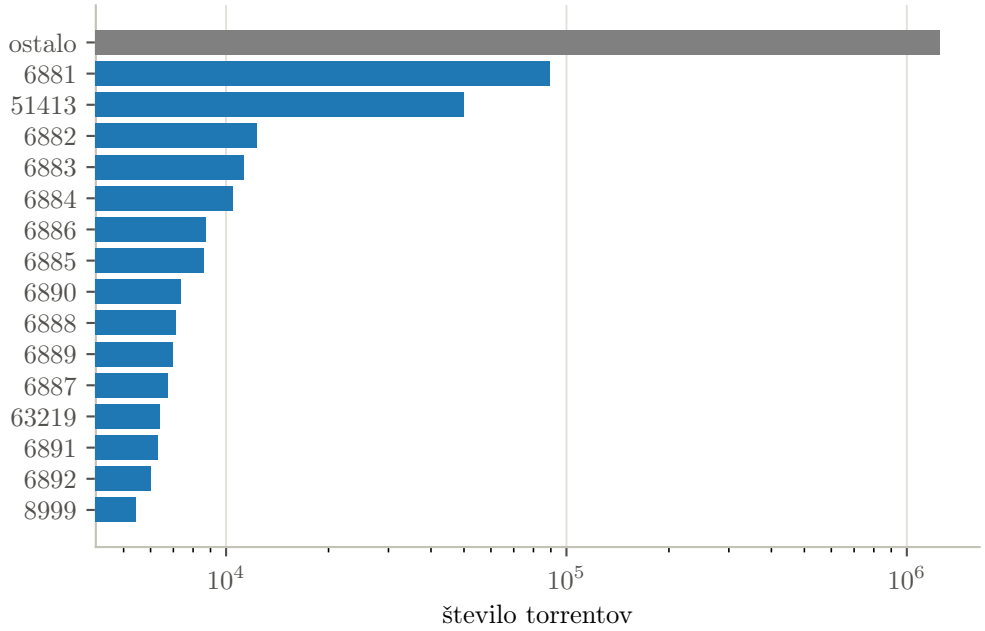
Tabela 3.5: Deset največjih posameznih datotek (globalno).

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

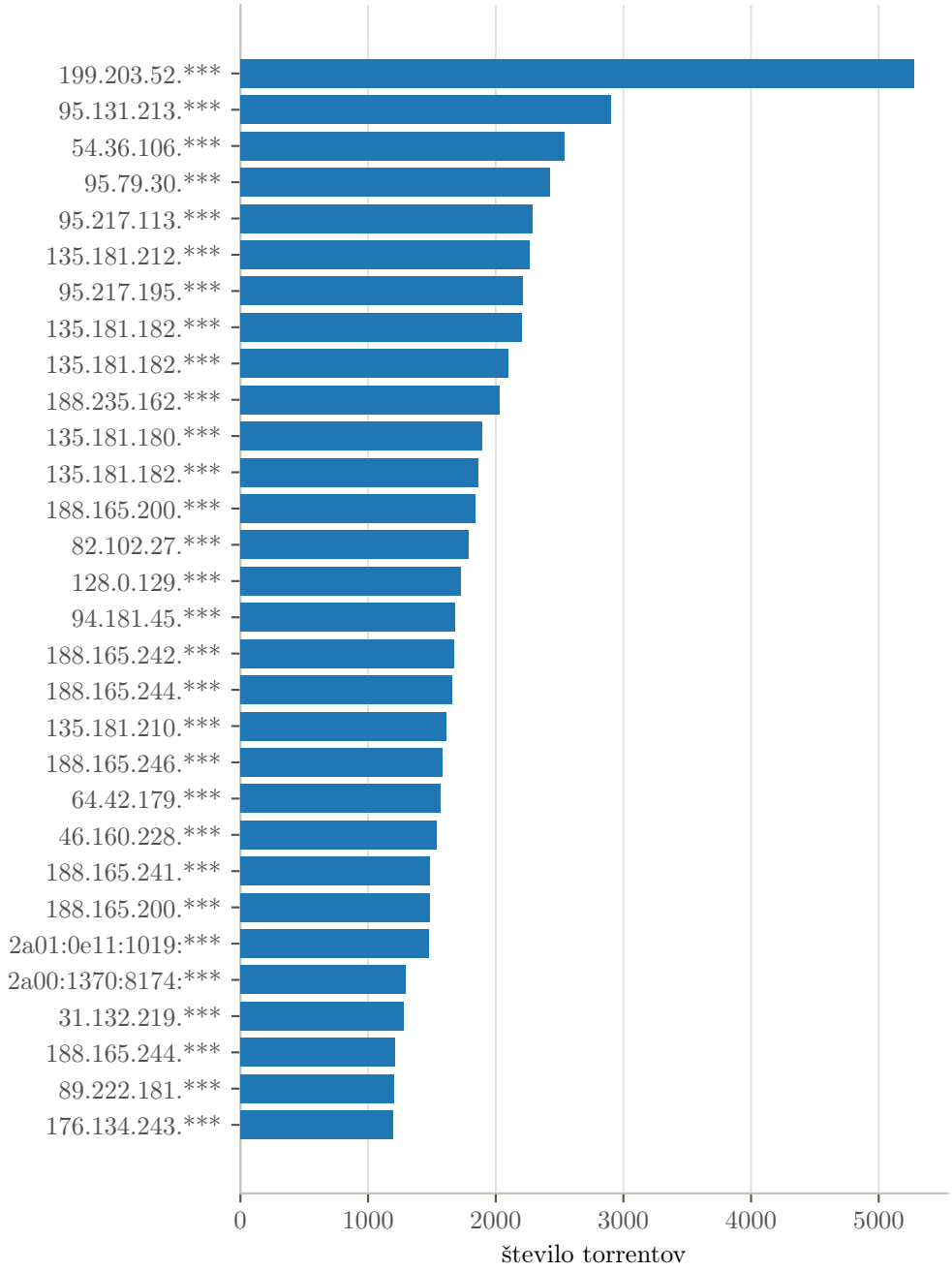
Najpogosteje uporabljena vrata TCP odjemalcev, po katerih smo prejeli torrente, so prikazana na sliki 3.10, najpogosteje uporabljeni naslovi IP pa na sliki 3.11. Od 75,37 % naslovov IP smo prejeli natanko en torrent, torrente so nam poslale naprave na 559 419 različnih naslovih IP. Vrata 6881 so predlagana vrata v standardu BitTorrent [7], 51413 pa privzeta vrata programa Transmission [50].

Nekateri odjemalci za BitTorrent imajo funkcijo naključne izbire vrat ob vsakem zagonu. Nekateri omrežni operaterji so namreč z namenom preprečevanja širjenja piratskih vsebin na požarnih pregradah blokirali vrata TCP, značilna za promet BitTorrent. Zaradi naključnosti izbire vrat take blokade niso nujno učinkovite.

Od 43 161 naslovov IP smo prejeli vsaj dva torrenta v razmaku vsaj enega meseca. Izmed njih jih je svojega odjemalca posodobilo vsaj 6920 (16,03 %). Toliko izmed njih smo namreč videli najprej z nižjo in nato z višjo različico odjemalca.



Slika 3.10: Najpogosteje uporabljena oddaljena vrata TCP, na katera smo se povezovali za prenos metapodatkov.



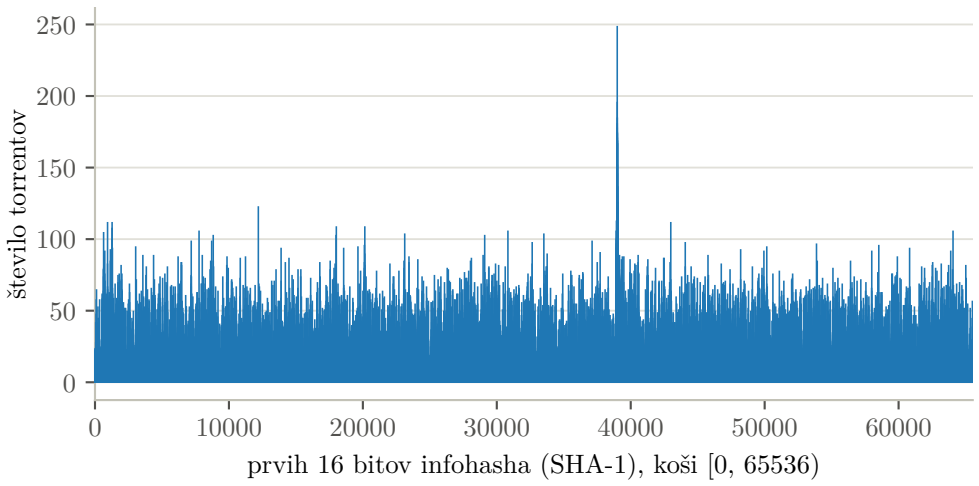
Slika 3.11: Naslovi IP, od katerih smo prejeli največ metapodatkov.

3.9.1 Porazdeljenost zgoščenih vrednosti zajetih metapodatkov

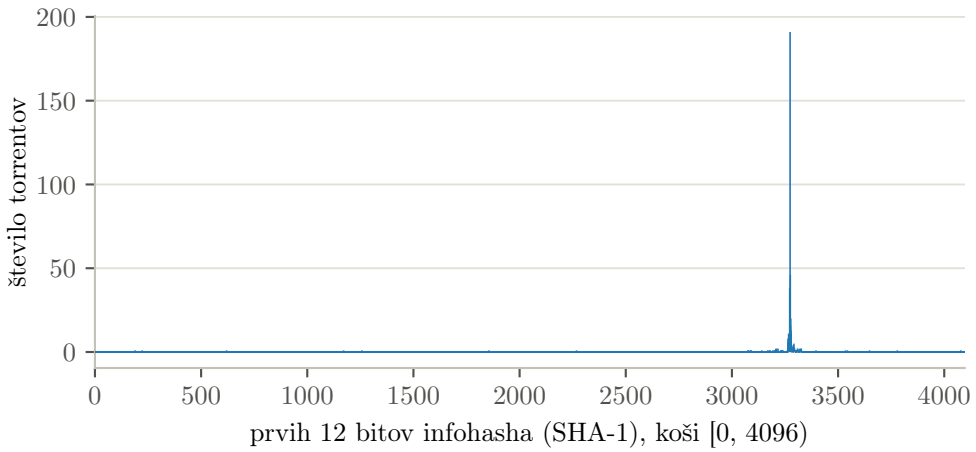
Predvidevali smo, da infohashi najbrž ne bodo enakomerno porazdeljeni, zato smo pripravili izrise porazdelitve prejetih infohashev na sliki 3.12.

Omenili smo, da smo za reševanje iz napada Sybil med zajemom menjali ID. Ker smo ga vedno zamenjali na povsem naključnega, iz slike z gostoto porazdelitve infohashev na celotnem korpusu ni dobro razvidno, da bi bili pristranski glede izbire infohashev.

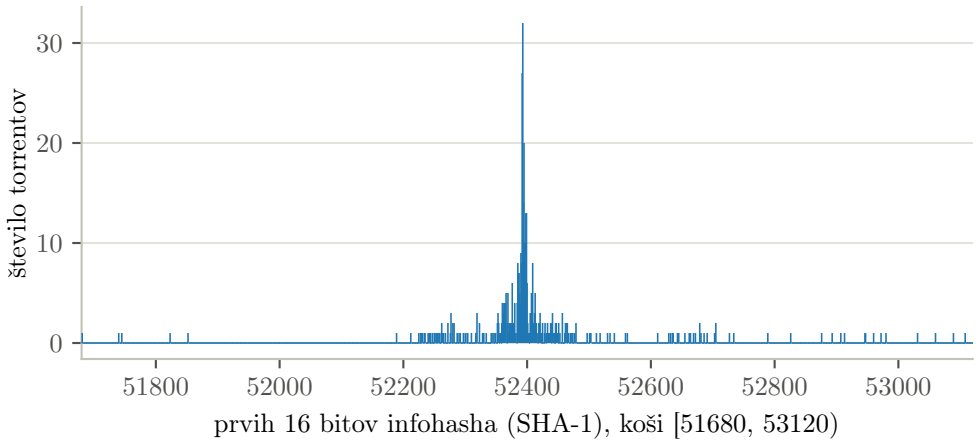
Da bi bolje prikazali pristranskost ob nespremenljivem ID-ju, smo pognali namenski zajem s konstantnim ID-jem (zajemali smo, dokler nismo naleteli na napad Sybil). Slika 3.13 (in povečava na sliki 3.14) prikazuje gostoto porazdelitve tako pridobljenih torrentov.



Slika 3.12: Gostota porazdelitve infohashev prejetih torrentov. Enakomerna porazdelitev infohashev bi imela $\frac{1546563}{65536} \approx 24$ torrentov na koš.



Slika 3.13: Gostota porazdelitve infohashev prejetih torrentov v majhnem namenskem zajemu 518 torrentov.



Slika 3.14: Gostota porazdelitve infohashev prejetih torrentov v majhnem namenskem zajemu 518 torrentov – povečava na območje z veliko torrenti.

3.10 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 anonimizirane 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*).

Poglavje 4

Diskusija in zaključek

Skupno smo v vseh zajemih zbrali 1 546 563 torrentov, ki opisujejo datoteke v skupni velikosti 9,112 PiB. Podatke smo analizirali in z njimi predstavili različne lastnosti omrežja BitTorrent.

Opazili smo padec uporabe odjemalca μTorrent in rast uporabe odjemalca qBittorrent, ki je v vseh zajemih vodil po popularnosti. Prikazali smo, katere različice programa qBittorrent so najpogosteje uporabljene in ocenili, da je v enem mesecu svoj odjemalec posodobilo vsaj 16 % uporabnikov.

Vrsto vsebine, ki jo uporabniki omrežja BitTorrent prenašajo, smo ocenili na dva načina; glede na datotečno končnico in s klasifikacijo z jezikovnim modelom. Klasifikacija glede na končnico potrjuje, da je večina torrentov v omrežju namenjena prenosu videoposnetkov, sledijo pa zvočni posnetki. Klasifikacija z jezikovnim modelom podrobneje pokaže, da je 28,9 % torrentov s pornografsko vsebino (kategorije *porn*, *hentai* in *childporn*), ki ji sledijo televizijske oddaje in serije s 15,2 %, filmi s 14,7 % (kategoriji *movie* in *documentary*) ter glasba z 11,5 %. Tu je na mestu razmislek o načinu distribucije – za en film bomo našli veliko manj torrentov kot za eno enako popularno serijo, kajti serija sestoji iz več epizod, ki jih često distribuirajo po epizodah. Ker torrent ni spremenljiv, ko ga enkrat izdelamo, je za epizode, ki jih piratski nalagalci objavljajo sproti, treba izdelati vsakič nov torrent, kar vpliva na naš način merjenja popularnosti kategorij po številu torrentov v prid serijam.

Pri obravnavi navedenih podatkov o popularnosti ločljivosti in kodekov je treba upoštevati dejstvo, da so izluščeni iz imen datotek. Nalagalcem piratskih vsebin v ime nikakor ni treba vpisati teh podatkov, zato podatek o ločljivosti in kodeku najbrž vpišejo pretežno takrat, ko so na izbiro kodeka oziroma ločljivost ponosni – ko je posnetek kvaliteten, torej zapisan z dobrim kodekom oziroma v visoki ločljivosti. Prisotnost slabših kodekov v omrežju je zato mogoče podcenjena – nalagalci se s slabšim kodekom preprosto v imenih datotek ne bahajo. To bi bilo vredno preveriti v nadaljnjih raziskavah, kjer bi prenašali tudi vsebino datotek.

Države, iz katerih prihajajo uporabniki omrežja BitTorrent, so dober pokazatelj kvalitete internetne infrastrukture. Seveda so zato na vrhu lestvice države z veliko računalniško pismenimi prebivalci, ki imajo kvalitetno povezavo v internet. Kvaliteto interneta smo po državah merili tudi z razmerjem med prometom BitTorrent na modernejši šesti različici internetnega protokola in prometom BitTorrent na stari četrti različici internetnega protokola. Mehika, Brazilija, Romunija in Tajvan po povezljivosti z IPv6 izstopajo.

Za boljšo predstavo o datotekah v omrežju smo objavili imena največjih datotek, ki jih prenašajo uporabniki slovenskih naslovov IP in imena največjih datotek, ki jih prenašajo uporabniki omrežja BitTorrent na vsem svetu.

Navedli smo še nekaj tehničnih lastnosti protokola, denimo, katera vrata TCP pogosto uporabljajo odjemalci. Sodeč po grafu sklepamo, da večina odjemalcev uporablja naključno izbrana vrata, kar je ugodno za izogibanje preprostim blokadam protokola BitTorrent na požarnih pregradah, po popularnosti pa sledijo v standardu opisana vrata (688x) in privzeta vrata programa Transmission (51413). Nadalje prikažemo tudi gostoto porazdelitve infohashev po celotnem prostoru vseh možnih infohashev, saj zaradi specifik načina zajema metapodatkov infohashi prejetih metapodatkov niso enakomerno razporejeni po spektru vseh možnih, kar potrdimo z namenskim zajemom z enim konstantnim ID-jem.

Naša zastavljena cilja o izdelavi programa za zajem in o analizi pridobljenih podatkov sta bila dosežena. Program je po sklepu o postopku 1 res

nepristranski na vsebino (je pa uteženo pristranski glede na popularnost torrentov) in omrežja ni motil, saj se je držal standardov za izdelavo odjemalca BitTorrent in prenašal omejeno količino torrentov hkrati. Zajeli smo veliko metapodatkov, na podlagi njih smo lahko sklepali o sestavi omrežja.

Domnevo o tipu vsebine, ki jo uporabniki prenašajo, lahko potrdimo. Piratske videovsebine so res najpopularnejša oblika datotek glede na končnico. Prav tako na podlagi razdelka 3.4 o različicah odjemalcev qBittorrent sklepamo, da uporabniki res po večini uporabljajo aktualne različice programske opreme.

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 najučinkoviteje z dodatkom standarda PeX (izmenjava soležnikov) [47], 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 v primerjavi z nemogočim prenašanjem celotne vsebine datotek znatno zmanjšali porabo pasovne širine.

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 (pridobljeno 19. 7. 2026).

[2] Tim Bray. *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC 8259. Dec. 2017. DOI: 10.17487/RFC8259. URL: https://www.rfc-editor.org/info/rfc8259 (pridobljeno 19. 7. 2026).

[3] Juliusz Chroboczek. *BEP 32: BitTorrent DHT Extensions for IPv6*. BitTorrent Enhancement Proposal 32. BitTorrent.org, okt. 2009. URL: https://www.bittorrent.org/beps/bep_0032.html (pridobljeno 9. 7. 2026).

[4] Juliusz Chroboczek. *dht.c*. 2009. URL: https://github.com/jech/dht/blob/master/dht.c (pridobljeno 9. 7. 2026).

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

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

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

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

[9] 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.

[10] Earendil Inc. and Contributors. *Pi: A Minimal LLM Agent Harness*. 2026. URL: https://pi.dev (pridobljeno 20. 7. 2026).

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

[12] 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.

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

[14] Dave Gamble in cJSON contributors. *cJSON: Ultralightweight JSON parser in ANSI C*. 2009. URL: https://github.com/DaveGamble/cJSON (pridobljeno 12. 7. 2026).

[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*. URL: https://bitmagnet.io/ (pridobljeno 10. 7. 2026).

[17] Google. *Google Trends: Search interest for webp worldwide, 2004–present*. 2026. URL: https://trends.google.com/trends/explore?date=all&q=webp (pridobljeno 20. 7. 2026).

[18] 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).

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

[20] 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).

[21] 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.

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

[23] International Organization for Standardization. *ISO 3166-1:2020 Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*. Geneva, CH: International Organization for Standardization, 2020. URL: https://www.iso.org/standard/72482.html (pridobljeno 19. 7. 2026).

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

[25] 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).

[26] 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.

[27] 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.

[28] Linux man-pages project. *poll(2) — Linux Programmer's Manual*. 2024. URL: https://man7.org/linux/man-pages/man2/poll.2.html (pridobljeno 19. 7. 2026).

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

[30] *Magnet Spider*. URL: https://clzhizhu.com (pridobljeno 9. 7. 2026).

[31] 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.

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

[33] 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.

[34] 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.

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

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

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

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

[39] Ollama Inc. *Ollama*. 2023. URL: https://github.com/ollama/ollama (pridobljeno 21. 7. 2026).

[40] platelminto. *parse-torrent-title*. URL: https://github.com/platelminto/parse-torrent-title (pridobljeno 10. 7. 2026).

[41] 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 (pridobljeno 19. 7. 2026).

[42] 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).

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

[44] Qwen Team. *Qwen3.6-35B-A3B*. Apr. 2026. URL: https://huggingface.co/Qwen/Qwen3.6-35B-A3B (pridobljeno 21. 7. 2026).

[45] Ren'Py Project. *The Ren'Py Visual Novel Engine*. URL: https://www.renpy.org/ (pridobljeno 20. 7. 2026).

[46] 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 (pridobljeno 21. 7. 2026).

[47] the8472. *BEP 11: Peer Exchange (PEX)*. BitTorrent Enhancement Proposal 11. BitTorrent.org, 2015. URL: https://www.bittorrent.org/beps/bep_0011.html (pridobljeno 21. 7. 2026).

[48] the8472. *DHT Infohash Indexing*. 2017. URL: https://www.bittorrent.org/beps/bep_0051.html (pridobljeno 28. 2. 2023).

[49] 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.

[50] Transmission Project. *Transmission*. Ver. 4.1.3. 2026. URL: https://transmissionbt.com/ (pridobljeno 21. 7. 2026).

[51] 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.

[52] 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.

[53] Paul A. Watters, Robert Layton in Richard Dazeley. „How much material on BitTorrent is infringing content? A case study". V: *Information Security Technical Report* 16.2 (2011), str. 79–87. ISSN: 1363-4127. DOI: https://doi.org/10.1016/j.istr.2011.10.001. URL: https://www.sciencedirect.com/science/article/pii/S1363412711000616.

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

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

[56] Jackie Zhong. *The Past, Present, and Future of P2P Network: Applications and Challenges*. Dec. 2021. URL: http://www.cse.wustl.edu/~jain/cse570-21/ftp/zhong520/index.html (pridobljeno 21. 7. 2026).

Dodatek A

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, je na voljo na zahtevo prek elektronske pošte na naslov anton@sijanec.eu.

Izsek kode 4: Podatki ročne klasifikacije 119 torrentov.

```
valid=119 invalid=0 categories={'porn': 35, 'tvshows': 13, 'movie': 13,
↪  'pictures': 1, 'game': 7, 'anime': 9, 'book': 10, 'music': 17,
↪  'various': 1, 'hentai': 5, 'software': 2, 'other': 1, 'unknown': 5}
↪  legalities={'piracy': 113, 'legitimate': 1, 'unknown': 5}
↪  catdeležˇ={'porn': 0.29411764705882354, 'tvshows': 0.1092436974789916,
↪  'movie': 0.1092436974789916, 'pictures': 0.008403361344537815, 'game':
↪  0.058823529411764705, 'anime': 0.07563025210084033, 'book':
↪  0.08403361344537816, 'music': 0.14285714285714285, 'various':
↪  0.008403361344537815, 'hentai': 0.04201680672268908, 'software':
↪  0.01680672268907563, 'other': 0.008403361344537815, 'unknown':
↪  0.04201680672268908} legdeležˇ={'piracy': 0.9495798319327731,
↪  'legitimate': 0.008403361344537815, 'unknown': 0.04201680672268908}
```

Izsek kode 5: Poziv za klasifikacijo z jezikovnim modelom.

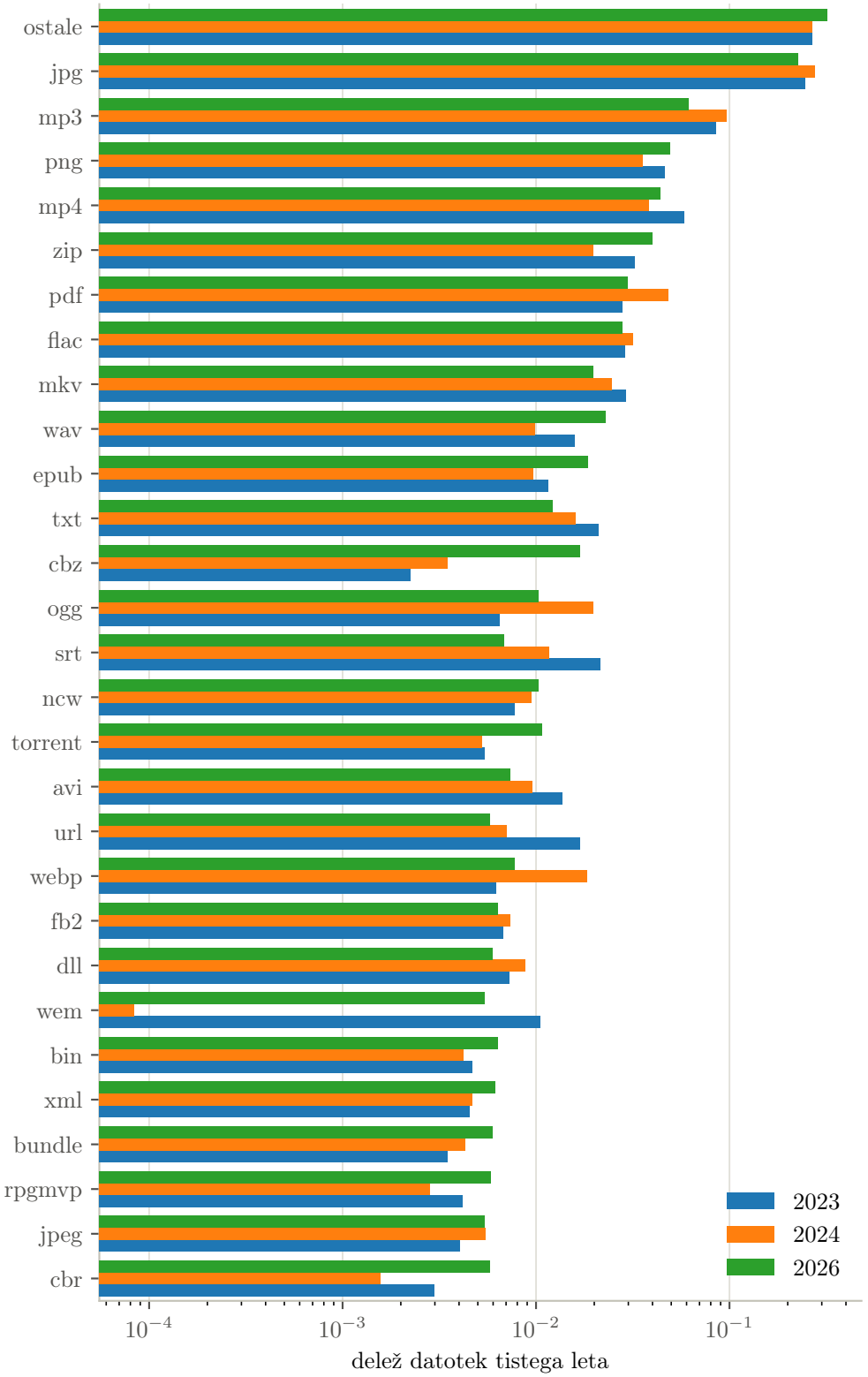
```
You're given a text file describing a torrent. Your task is to determine whether the torrent contains pirated content (is
      used for piracy, copyright infringement), other illegal content or is legitimate.

Your output is two space separated strings and nothing else on the first line, without quotes, as follows:
```

```
First string of first line: "piracy" if the torrent is piracy, "illegal" if torrent has illegal content other than piracy,
      "legitimate" if the torrent is legitimate, "unknown" if you can't determine or are uncertain about whether the
      torrent is piracy or not, or "error" if there was some error that prevented you from making a decision (for example
      you couldn't read the file or a failed search query -- note that search queries are optional).

Second string of first line: one of anime,book,childporn,documentary,game,hentai,movie,music,pictures,porn,software,
      tvshows,other,unknown,various,error for the category of the torrent.

On the second line of the output, provide a short explanation for your decision.

Do not add any markdown formatting to your output text. It needs to be exactly two lines long, no blank lines or code
      blocks. Before outputting text, make really sure again that it is two lines long and the first line contains only
      one the correct format. Your output will be parsed automatically. You may search the web to get more information
      about the torrent.
```

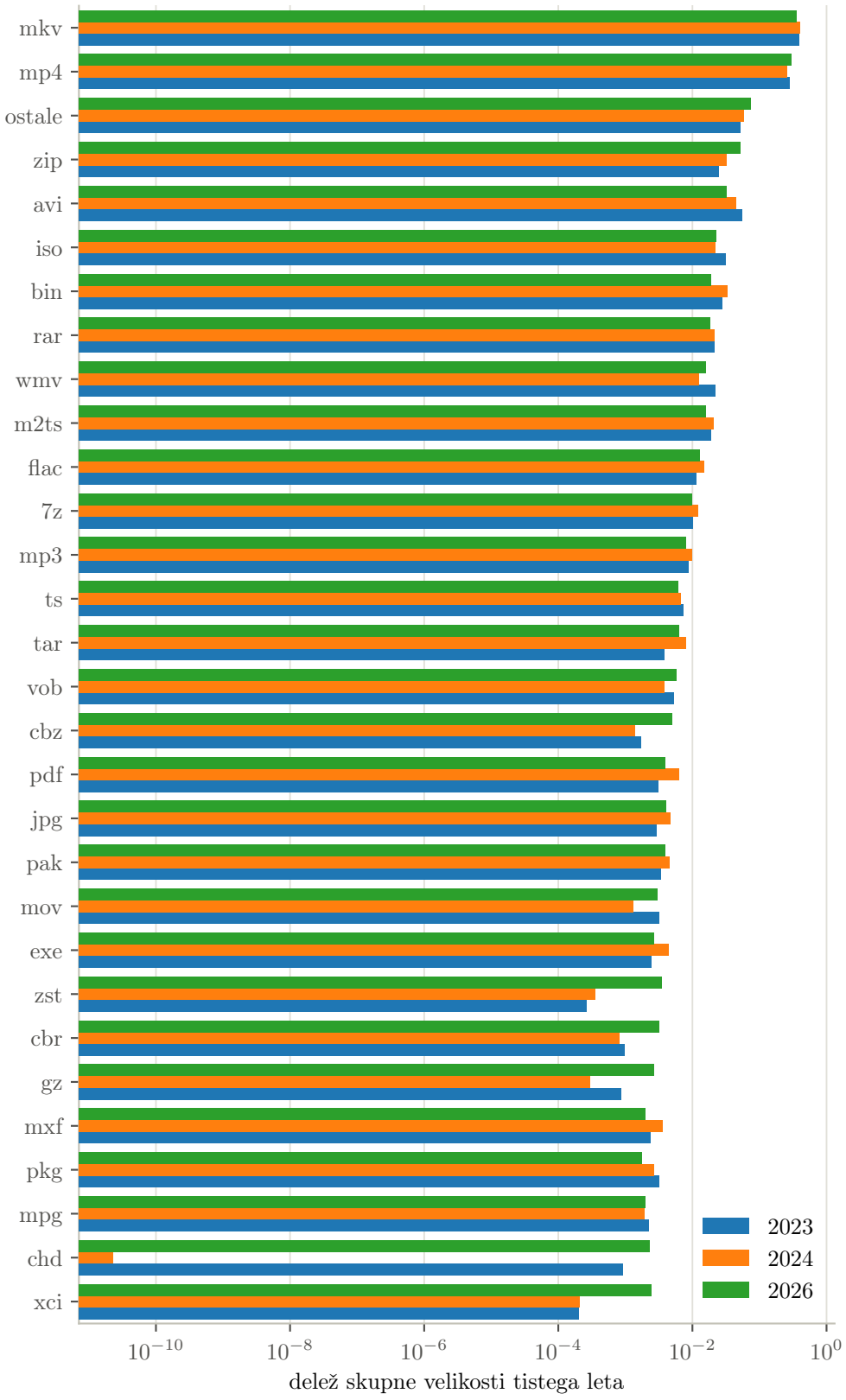
Izsek kode 6: Skript za popravilo nepravilno strukturiranih izhodov jezikovnega modela.

```
#!/bin/bash
set -xeuo pipefail
if [ ! `pi --thinking xhigh -p "output the number 6 and nothing else"` =
↪  "6" ]
then
        echo we have a bad model >&2
        exit 1
fi
while read line
do
        ih=`rev <<<$line | cut -d. -f2 | cut -d/ -f1 | rev`
        firstline=`head -n1 corrected/$ih.txt`
        legitimacy=`cut -d\  -f1 <<<"$firstline" || :`
        category=`cut -d\  -f2 <<<"$firstline" || :`
        legitimacies="piracy\nillegal\nlegitimate\nunknown\n" # error\n
        categories="anime\nbook\nchildporn\ndocumentary\ngame\nhentai"
        categories=$categories"\nmovie\nmusic\npictures\nporn\nsoftware"
        categories=$categories"\ntvshows\nother\nunknown\nvarious\n"
        set +e
        echo -e "$legitimacies" | grep "^$legitimacy$"
        legitimacystatus=$?
        echo -e "$categories" | grep "^$category$"
        categorystatus=$?
        set -e
        if [ ! `wc -w <<<"$firstline"` -eq 2 ] || [ ! $legitimacystatus
        ↪  -eq 0 ] || [ ! $categorystatus -eq 0 ]
```

```
25          then
26                  echo | pi --thinking xhigh --session sessions/$ih.jsonl -p
                    ↪  "You did not answer in the correct format. First line
                    ↪  must be of two words, separated by only space. First
                    ↪  word must be one of piracy, illegal, legitimate,
                    ↪  unknown, error and it specifies the legitimacy of the
                    ↪  torrent. Second word must be one of anime, book,
                    ↪  childporn, documentary, game, hentai, movie, music,
                    ↪  pictures, porn, software, tvshows, other, unknown,
                    ↪  various, error and it specifies the category of the
                    ↪  torrent. Please output in the correct format again
                    ↪  (including the brief reasoning for your choice on the
                    ↪  second line). Do not add any formatting to the output.
                    ↪  If you need the parsed torrent again, read it from the
                    ↪  file 'text/$ih.txt'. If you need the original prompt
                    ↪  (your original instructions) again, read it from the
                    ↪  file 'prompt.txt'." > corrected/$ih.txt
27                  # exit
28          fi
29  done < <(find corrected -name '*.txt')
```



Slika A.1: Tipi datotek glede na število datotek v vseh torrentih.



Slika A.2: Tipi datotek glede na skupno velikost datotek v vseh torrentih.