Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matic Šincek

Analiza javnosti vsebin na omrežju BitTorrent

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: dr. Andrej Brodnik

Ljubljana, 2020

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Eno najbolj znanih in praktično uporabljenih prekrivnih vrstnik z vrstnikom omrežij je BitTorrent. V diplomski nalogi preučite koliko javno oglaševanih vsebin in koliko neoglaševanih vsebin ponujajo posamezni strežniki na omrežjih BitTorrent.

V nalogi najprej preučite delovanje omrežij BitTorrent. Nato opredelite kaj pomeni javno oglaševana vsebina in na podlagi tega preglejte primerno velik vzorec strežnikov ter na njih preglejte razmerje med obema oblikama vsebin. Izpeljite še podatke glede na združevanje strežnikov pri posameznem ISP, ASN in državi.

Rad bi se zahvalil mentorju, ki mi je vzbudil zanimanje za področje oz. temo, ki sem jo raziskal v diplomski nalogi in me mentoriral. Res mi je bil v veliko pomoč. Prav tako se zahvaljujem staršem, ki so mi s čustveno in materialno pomočjo pomagali, da sem prišel tako daleč.

Kazalo

Seznam uporabljenih kratic

HTTP	HyperText Transport Protocol	Protokol za prenašanje hiperteksta
IP	Internet Protocol	Internetni protokol
ISP	Internet Service Provider	Ponudnik internetnih storitev
AS	Autonomous System	Avtonomni sistem
URL	Uniform Resource Locator	Enolični lokator vira

Povzetek

Naslov: Analiza javnosti vsebin na omrežju BitTorrent

Avtor: Matic Šincek

V diplomski nalogi smo se lotili analiziranja pojma globoki torent. Preučili smo protokol BitTorrent in njegovo različico iz leta 2005, ki deluje brez prisotnosti strežnikov. Naredili smo sistem, s pomočjo katerega smo analiziral promet protokola ter poskušali iz prometa izluščiti vsebine, ki so dosegljive v omrežju, vendar jih ne moremo najti na straneh za odkrivanje torentov (kot npr. Thepiratebay), kar je značilno za vsebine globokega torenta. Ocenili smo razsežnost globokega torenta in delež globokega torenta v celotnem BitTorrent prometu. Prav tako smo ocenili delež globokega torenta glede na države, ponudnike internetnih storitev in avtonomne sisteme.

Ključne besede: Globoki torent, BitTorrent, princip „Enak-z-enakim“, torenti.

Abstract

Title: Content publicity analysis on the BitTorrent network

Author: Matic Šincek

In the diploma thesis we tackled the analysis of the deep torrent phenomenon. We studied the Bittorrent protocol and its version from the year 2005, which operates without the presence of servers. We created a system to analyze and extract content from the traffic that is accessible on the network but cannot be found on torrent discovery websites (like Thepiratebay), which is typical for deep torrent resources. We assessed the dimensions of the deep torrent and the share of deep torrent in Bittorrent traffic. We also assessed the share of deep torrent in relation to different countries, internet service providers and autonomous systems.

Keywords: Deep torrent, BitTorrent, „Peer-to-peer“ principle, torrents.

Poglavje 1

Uvod

1.1 Motivacija

BitTorrent je protokol, ki ga pozna marsikdo, saj omogoča tudi prenašanje piratskih vsebin. Uporabniki protokola lahko izbirajo med ogromno vsebinami, ki jih najdejo na spletnih straneh, ki so namenjene oglaševanju takih vsebin. Poleg vsebin, do katerih povezave lahko najdemo na spletu, pa se po BitTorrent omrežju delijo tudi vsebine, ki niso javno dostopne. To so vsebine, ki imajo lahko vprašljiv namen, zato je z vidika spletne varnosti pomembno, da raziskujemo njihov izvor. Zanimalo nas je, koliko je na spletu zasebnih vsebin, kakšna je njihova popularnost in od kod prihajajo. Rezultati raziskave bi lahko pomagali organizacijam, ki se ukvarjajo z analizo vsebin, ki se prenašajo po omrežju in organizacijam, ki se ukvarjajo s spletno varnstjo.

1.2 Cilj diplomske naloge

Želeli smo bolje spoznati delovanja protokola BitTorrent. Zanimalo nas je, kako pokazati prisotnost zasebnih vsebin v omrežju in kako ločiti vsebine, glede na to, ali so zasebne ali javno dostopne. Naš cilj je bil, da bi ugotovili, kakšno je razmerje med javno dostopnimi vsebinami in privatnimi vsebinami. Poleg tega nas je zanimalo tudi, če lahko ugotovimo, kje se nahajajo ponu-

dniki zasebnih vsebin, pri katerih ponudnikih internetnih storitev so, pod kateri avtonomni sistem spadajo in iz katere države prihajajo.

Poglavje 2

BitTorrent

2.1 Protokol

BitTorrent je protokol za deljenje datotek med velikim številom uporabnikov. Uporabnik protokola je oseba ali programska oprema, ki dostopa do BitTorrent omrežja in prenaša ali oddaja vsebine, ki so na voljo v omrežju. Gre za vsebine, ki jih ima in želi veliko uporabnikov. Značilnost protokola je, da datoteke prenaša po delih in od več uporabnikov hkrati, kar običajno pomeni hitrejši prenos.

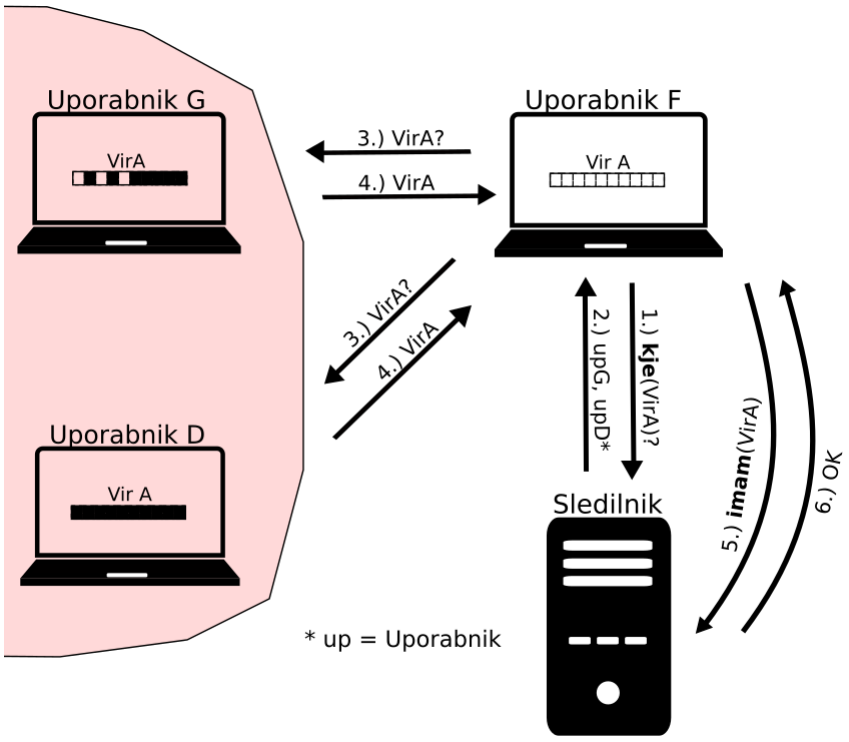
Veliko popularnost je protokol dosegel pri razpečevanju piratskega gradiva kot so filmi, glasba, e-knjige in programi zato, ker se tam potrebujejo takšni pogoji, kot jih protokol omogoča - veliko uporabnikov, ki želijo iste datoteke, ki so pogosto precej velike. Poleg piratskih vsebin se seveda protokol uporablja tudi za razpečevanje legalnih vsebin in programov, kot na primer zagonska slika operacijskega sistema Debian, dostopna na naslovu `https://www.debian.org/CD/torrent-cd/`. Protokol je leta 2001 razvil ameriški programer Bram Cohen [14].

Vsebinam, ki se prenašajo med uporabniki pravimo *viri*. Vir je pred prenašanjem navidezno razdeljen na dele enake velikosti (ponavadi 256KB

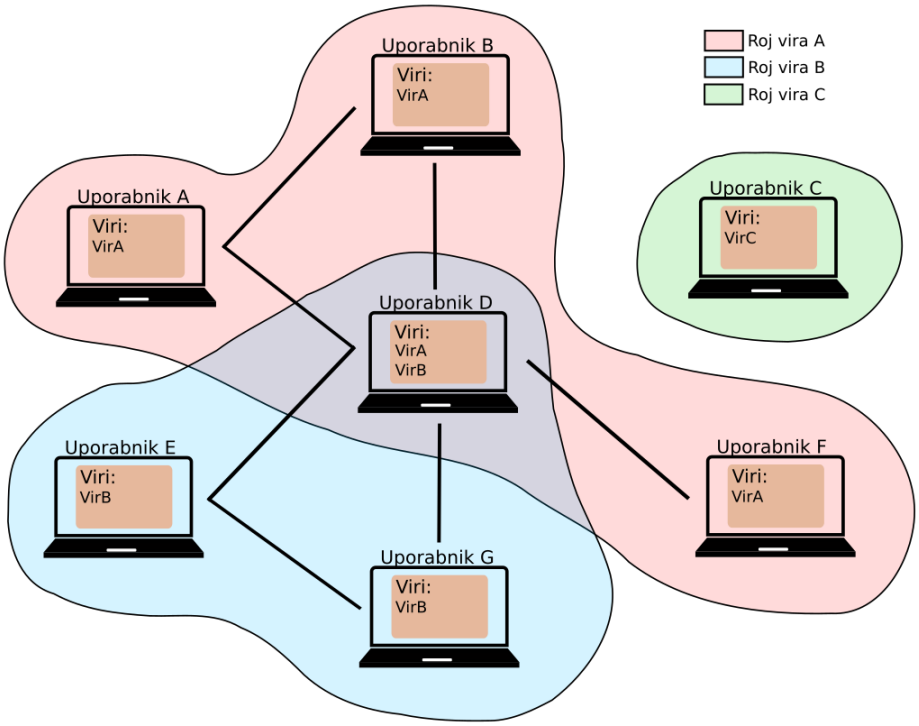
ali pa 1MB [3]). To omogoča, da lahko vir prenašamo od večih uporabnikov hkrati. Ko se nekdo odloči, da želi začeti z deljenjem nekega vira, mora vir najprej popisati z metapodatki, ki opisujejo, kako vir izgleda, kako je razdeljen in kako ga lahko uporabniki najdejo. Ta opis se shrani v datoteko, ki ji rečemo *torent*. Opis vira se mora nato na nek način deliti z drugimi uporabniki, zato, da bodo vedeli, kako izgleda in kako priti do njega. Informacija o tem, da uporabnik deli ta specifičen vir, se objavi na strežnikih imenovanih *sledilniki*. Naloga sledilnikov v omrežju je, da beležijo, kateri viri so na voljo v omrežju in kateri uporabniki jih nudijo.

Drug uporabnik, ki je prejel datoteko z opisom vira, želi začeti s prenosom le tega. Najprej mora vprašati sledilnik, kdo ta vir ima. Ko ugotovi, kateri uporabniki ga imajo, se lahko poveže z njimi in jih prosi, da mu začnejo pošiljati vir. Ker je vir razdeljen na dele, lahko za dele zaprosi različne uporabnike, prenos pa se lahko kadarkoli prekine in nadaljuje brez izgube napredka. Prav tako se lahko dele prenese v naključnem vrstnem redu, saj uporabnik, ki prenaša vir, ve, kako si sledijo in jih lahko na koncu sestavi. S tem, ko dele prenese od večih uporabnikov, od vsakega malo, pridobi tudi na hitrosti prenosa. Takoj, ko prejme nekaj delov vira, jih lahko začne deliti. To stori tako, da sledilniku, ki je odgovoren za vir, sporoči, naj ga doda na seznam uporabnikov, ki nudijo ta vir. Celoten proces je viden na sliki 2.1.

Skupinam uporabnikov, ki so navidezno povezani, ker prenašajo oziroma delijo isto datoteko, rečemo *roji* (ponazorjeno kot barva skupine uporabnikov na sliki 2.2). Vsak uporabnik, ki je v roju, ima vlogo *sejalca* ali *soležnika* [6]. Soležnik je vsak uporabnik roja, ki deli ali pa prenaša dele vira. Uporabniku, ki je začel z deljenjem nekega vira ali pa je k sebi že prenesel celotni vir in ga sedaj deli drugim, rečemo sejalec. Dokler je v roju vsaj en sejalec, to zagotavlja, da bodo lahko vsi soležniki po potrebi prejeli celoten vir in da se bo s časom število sejalcev povečalo. Večje število soležnikov, sploh pa sejalcev, pomeni hitrejše prenašanje vira za vse soležnike. Roj brez sejalcev lahko pomeni, da nihče od soležnikov ne bo uspel sestaviti celotnega vira,



Slika 2.1: Komunikacija v BitTorrent omrežju.



Slika 2.2: Roji v BitTorrent omrežju.

ker lahko, da obstaja del, ki ga nima nobeden od soležnikov. Razlog, da bi v roju zmanjkalo sejalcev je, da imajo uporabniki med prenosom možnost, da se odklopijo iz omrežja.

Torent, datoteka, ki vsebuje metapodatke o viru, je razdeljena na dva dela: informacijski del in del s podatki o sledilnikih [6]. Informacijski del vsebuje podatke o imenu in velikosti datotek v viru, velikost enega dela, število delov na katere je vir razdeljen in SHA1 podpis vsakega dela (slednje se uporablja pri preverjanju celovitosti). Del s podatki o sledilnikih vsebuje naslove sledilnikov, ki poznajo roj tega vira. Če se želi uporabnik vključiti v roj nekega vira, mora sledilniku poslati zahtevo, v katero vključi identifikator vira. Sledilnik mu nato vrne seznam soležnikov v tem roju in tega uporabnika vključi v seznam. SHA1 povzetek informacijskega dela torenta je 160 bitna vrednost, ki se uporablja kot enolični identifikator vira v omrežju. Torent mora z nami deliti tisti, ki ga je ustvaril, ponavadi preko spleta. To-

rent zagonske slike za Debian najdemo na Debianovi spletni strani, torente piratskih virov pa lahko najdemo na javnih spletnih straneh za odkrivanje torentov.

Spletne strani za odkrivanje torentov so strani, na katere uporabniki objavljajo torente za vire, ki jih delijo. Drugi uporabniki lahko na teh straneh torente poiščejo po imenu vira in jih prenesejo. Primera takšnih strani sta na primer `https://www.thepiratebay.org` ali `https://rarbg.to/`.

BitTorrent je protokol, ki deluje po principu „enak-z-enakim“, kar pomeni, da pri komunikaciji ne sodelujeta tradicionalni strežnik in odjemalec, ampak dva uporabnika, ki sta si po funkcionalnosti enakovredna in si pri pogovoru izmenjujeta vlogi oddajanja in prejemanja virov. Vir prejemamo od velikega števila uporabnikov, ki imajo posamezno omejeno oddajno hitrost, kot skupina pa ne [2]. V študiji „Really Truly Trackerless BitTorrent“ [10] so ugotovili, da sledilniki žal niso tako robustni in skalabilni kot del omrežja, ki deluje po principu „enak-z-enakim“. Ko je nek sledilnik nedosegljiv, se lahko zgodi, da se noben nov uporabnik ne more pridružiti roju.

Leta 2005 je bila razvita nova različica protokola, ki je delovala brez sledilnikov [7]. V tej različici so podatki o rojih porazdeljeni po uporabnikih v omrežju.

2.2 Brezsledilniška različica

Brezsledilniška različica protokola deluje brez sledilnikov, kar pa ne pomeni, da ideja sledilnikov ne obstaja več. Vloga sledilnika se prenese na uporabnike, ki delujejo kot porazdeljen strežnik [8]. To pomeni, da morajo uporabniki informacije o rojih, ki bi jih drugače hranili sledilniki, hraniti sami.

V tem poglavju bomo uporabnike enačili z vozlišči v omrežju. Vsako vozlišče dobi svojo tabelo z identifikatorji virov za katere vzdržuje seznam vo-

zlišč v roju. Če bi zbrali podatke iz vseh vozlišč, bi lahko sestavili seznam vseh virov, ki obstajajo.

Da bi vozlišče pridobilo podatke o roju določenega vira, mora vedeti, katera vozlišča hranijo te podatke. V ta namen uporablja BitTorrent drugotno prekrivno omrežje. Vsako vozlišče, ki se pridruži omrežju, dobi identifikator, ki je 160 bitna vrednost - iste dolžine kot identifikator vira [9]. Identifikator vozlišča določi, kako blizu je vozlišče drugim vozliščem. Bolj kot sta si vrednosti identifikatorjev blizu po velikosti, bližje sta si v omrežju. Isto velja tudi za razdaljo med identifikatorjem vozlišča in identifikatorjem vira. V omrežju velja, da bližje kot je identifikator vozlišča identifikatorju vira, večja je verjetnost, da bo imelo vozlišče informacije o roju tega vira. Če hočemo najti podatke o roju, moramo torej najti vozlišče, ki je dovolj blizu iskanemu identifikatorju vira.

Vozlišča morajo hraniti podatke o sosedih in drugih vozliščih, s katerimi se pogovarjajo zato, ker s tem gradijo omrežje. Omrežje omogoča, da najdemo podatke o rojih za vire, ki jih želimo prenesti. Vsako vozlišče hrani v svoji usmerjevalni tabeli sebi bližnja vozlišča.

Usmerjevalna tabela je sestavljena iz seznamov predpisane dolžine, ki jim pravimo koši. Koši imajo predpone, ki nakazujejo, kakšna vozlišča so shranjena v tem košu. Koš bo vseboval podatke o vozliščih z identifikatorji, ki se začnejo z istima bitoma kot predpona koša. Ko se koš napolni, se po potrebi razdeli v dva koša, pri čemer imata nova koša predpono, ki ima definiran en bit več. V usmerjevalni tabeli hranimo identifikatorje drugih vozlišč, njihove IP naslove in vrata, čas zadnje aktivnosti ter še druge podatke, kot je prikazano na sliki 2.3. Vozlišče mora samo skrbeti za to, da so podatki v koših še vedno relevantni. Če se neko vozlišče v tabeli ni odzvalo že petnajst minut, se preveri, če je še odzivno. Če ni, se zavrže.

BitTorrent ima v specifikaciji [8] opisane štiri glavne poizvedbe, ki omgočajo vozliščem, da komunicirajo:

Predpona	ID	IP	Vrata	Nazadnje viden	...
00	00000...	9.11.122.64	321	1592130882850	
	00101...	13.32.99.1	342	1593321382235	
010	01000...	26.5.11.22	551	1593321332455	
	01001...	76.100.100.2	431	1593233382831	
	01011...	76.100.100.1	443	1593343435525	
011	01101...	44.22.103.12	441	1592130882314	
	01111...	95.3.123.73	431	1593332133333	
1	10000...	25.1.22.34	441	1593321382850	
	10010...	18.36.16.44	441	1592546643656	
	11001...	92.19.10.100	551	1594324343243	

Slika 2.3: Usmerjevalna tabela v vozliščih.

ping je najbolj preprosta poizvedba, s katero se vozlišče prepriča ali je poljubno vozlišče še odzivno;

find_node je poizvedba, s katero lahko poljubno vozlišče vprašamo, če pozna vozlišče, ki ga iščemo. Parametra poizvedbe sta identifikator vozlišča, katerega hočemo vprašati in identifikator iskanega vozlišča. Če ga vprašano vozlišče pozna, nam vrne njegov IP naslov, drugače pa nam vrne podatke o nekaj najbližjih vozliščih, ki jih pozna;

get_peers je poizvedba, s katero lahko poljubno vozlišče vprašamo, če ima informacije o roju za nek vir. Kot parameter mu pošljemo identifikator vira, ki ga iščemo, vprašano vozlišče pa nam odgovori s podatki o roju, če vir pozna. Če vira ne pozna, nam pošlje nekaj najbližjih vozlišč iz svoje usmerjevalne tabele;

announce_peer uporabimo, ko se pridružujemo roju. Vozlišču, ki nam je podalo informacije o roju, sporočimo naj nas doda na seznam soležnikov v roju.

S temi poizvedbami so realizirane glavne funkcije, ki jih opravljajo vozlišča. S ping poizvedbo ugotovimo ali je treba vozlišče zamenjati, ker ni več aktivno. S ponovnim pošiljanjem find_node poizvedbe lahko z vedno več odkritimi vozlišči najdemo neko vozlišče in odkrijemo še vsa vozlišča na poti ter si tako napolnimo usmerjevalno tabelo.

Algoritem za iskanje vozlišča izgleda takole:

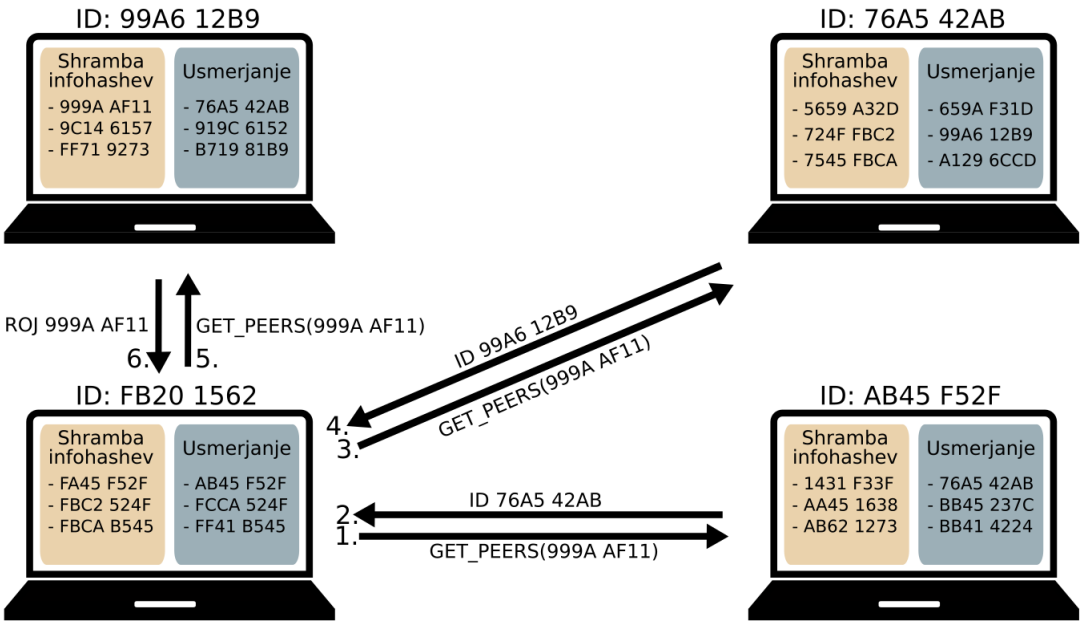
1. Poišči najbližje poznano vozlišče iskanemu iz svoje usmerjevalne tabele.
2. Pošlji temu najbližjemu vozlišču find_node zahtevo za iskano vozlišče.
3. Če poizvedba vrne iskano vozlišče, zaključi program.
4. Drugače s pridobljenimi vozlišči posodobi usmerjevalno tabelo in ponovi korak 1.

S ponovnim pošiljanjem get_peers poizvedbe najdemo poljuben roj, z announce_peer pa se mu pridružimo.

Algoritem za iskanje soležnikov izgleda takole:

1. Poišči najbližje poznano vozlišče iskanemu identifikatorju vira iz usmerjevalne tabele.
2. Pošlji temu najbližjemu vozlišču get_peers zahtevo za iskan identifikator vira.
3. Če poizvedba vrne roj vira, zaključi program.
4. Drugače s pridobljenimi vozlišči posodobi usmerjevalno tabelo in ponovi korak 1.

Na sliki 2.4 je na poenostavljen način predstavljena uporaba get_peers poizvedbe. Vozlišče z identifikatorjem FB20 1562 želi najti soležnike v roju vira z identifikatorjem 999A AF11. Zaporedno pošilja get_peers poizvedbe vozliščem, bližje in bližje tistemu vozlišču, ki pozna iskani vir. Prvi dve vozlišči ne poznata vira, zato ga napotita k najbližjemu vozlišču, ki bi ga lahko poznal. Vozlišče 99A6 12B9 pozna iskani vir in mu tako vrne podatke



Slika 2.4: Primer uporabe get_peers poizvedbe za iskanje vira 999A AF11.

o soležnikih v roju.

Poleg štirih glavnih poizvedb, ki podpirajo osnovne funkcionalnosti vozlišč v omrežju, je protokol dobil še nekatere razširitve, ki so jih predlagali raziskovalci. Te podpirajo funkcionalnosti, ki ne zadevajo nujno prenosa virov. Ena od takih razširitev, je razširitev 51 [1], ki omogoča, da lahko prosimo neko vozlišče, da nam pošlje seznam virov za katere pozna informacije o roju. S tem olajšamo delo raziskovalcem, ki bi želeli takšne informacije pridobiti na kakšen drug način, ki je bolj časovno potraten in manj učinkovit.

Poglavje 3

Metodologija

3.1 Globoki torent

V raziskavi „On Understanding the Exsistence of a Deep Torrent“ [11] se navežejo na globoki torent z analogijo na globoki splet. Globoki splet je del spleta, ki ga iskalniki ne indeksirajo, ampak je dostopen z določenimi orodji na naslovih, ki jih moramo poznati. Vsebine na njem so pogosto zaščitene z avtentikacijo. V raziskavi „The deep web and the darknet: A look inside internet's massive blackbox„ [12] so leta 2015 ocenili, da je velikost podatkov, shranjenih na 60 največjih globoko-spletnih straneh, štiridesetkrat večja, kot cel površinski splet.

Pri globokem torentu gre za vire, ki se pretakajo po BitTorrent omrežju, njihovih torentov pa ne moremo najti na spletnih straneh za odkrivanje torentov. Skriti so javnosti in so dostopni samo posameznikom, ki jim je nekdo povedal, kje jih najdejo. Takšne definicije globokega torenta smo se v raziskavi držali tudi mi. Viri, katerih torente lahko najdemo na prej omenjenih straneh, so zato javni in so jih v raziskavi [11] poimenovali površinski torent. S sprehajanjem po BitTorrent omrežju so zbirali vozliša in vire, ki jih delijo, nato pa razdelili vire glede na pripadnost globokemu oziroma površinskem torentu. Ocenili so, da je velikost globokega torenta štirikrat

večja od površinskega torenta. Raziskovanje področja globokega torenta je lahko zanimivo z vidika kibernetske varnosti, saj je tako kot globoki splet to eden od načinov manj nadzorovanega razpečevanja virov s vprašljivo motivacijo, moralnostjo in legalnostjo.

3.2 Beli in črni viri

Teoretično lahko vire razdelimo v dve kategoriji - beli in črni. Viri, ki jih povezujemo z globokim torentom, so črni, viri, ki jih povezujemo s površinskim torentom, pa so beli.

Barva vira, ki ga vozlišče deli, nam pove tudi nekaj o vozlišču. Če vozlišče deli samo bele vire, potem ga definiramo kot belega. V primeru, da lahko povežemo z vozliščem vsaj en črn vir, ga definiramo kot črnega. Barvo vozlišča smo nato definirali kot količnik med številom črnih virov in številom vseh virov povezanih z vozliščem. Vozlišče z barvo ena pomeni, da je vozlišče belo, barva manj kot ena pa pomeni neko stopnjo črnosti.

3.3 Zbiranje podatkov

Iz omrežja smo pridobili podatke o belih in črnih virih, ki se delijo v omrežju in o vozliščih, ki jih delijo. Začeli smo z zbiranjem identifikatorjev virov. Medtem, ko bi bele vire lahko zbrali iz spletnih katalogov torentov, smo mi želeli najti tudi črne.

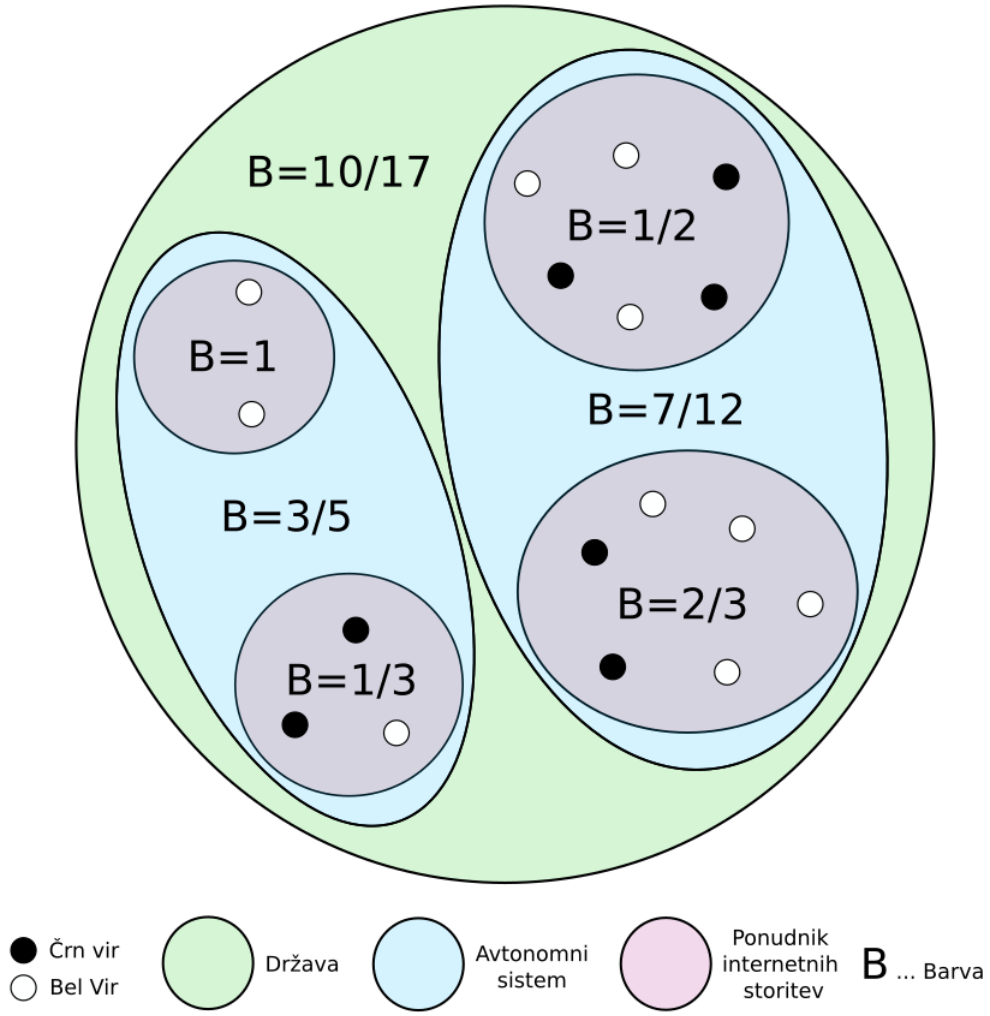
Vire smo morali zato pridobiti s pomočjo poizvedb brezsledilniške različice protokola. Začeli smo tako, da smo s sprehajanjem po omrežju pridobili čim večje število vozlišč. Od vozlišč smo pridobili seznam virov za katere imajo podatke o roju, nato pa smo od njih pridobili še podatke o rojih. Tako smo iz omrežja pridobili naključne identifikatorje virov in naslove vozlišč, ki jih trenutno delijo ali prenašajo.

Za vsak identifkikator vira smo preverili, če lahko zanj na spletnih straneh za odkrivanje torentov, najdemo pripadajoči torent ter tako določili barvo tega vira. Ko smo pobarvali vire, smo lahko prešteli tudi, kolikšen delež virov, ki jih deli posamezno vozlišče, je belih in koliko črnih. Ker smo pobarvali vire, smo lahko pobarvali tudi vozlišča. Vredno je omeniti, da se je analiza delala v nekem trenutnem posnetku stanja v omrežju, ki pa se skozi čas spreminja.

3.4 Združevanje podatkov

Imeli smo torej podatke o nekaterih virih, ki se delijo v omrežju in podatke o tem, katera vozlišča so odgovorna za deljenje virov. Za vsako vozlišče smo pridobili še podatke o lastništvu IP naslova. Preverili smo, iz katere države je, pri katerem ponudniku internetnih storitev (ang. *internet service provider ali ISP*) je registriran in pod kateri avtonomni sistem (ang. *autonomous system ali AS*) spada omrežje, v katerem je vozlišče. Avtonomni sistem je organizacija, ki je odgovorna za neko IP območje. Identificira ga globalno unikatna številka.

Podatke smo združili glede na državo, ponudnika storitev in številko avtonomnega sistema vozlišča, kot je prikazano na sliki 3.1. Tako kot smo pobarvali vozlišča, smo lahko pobarvali tudi države, ponudnike internetnih storitev in avtonomne sisteme.



Slika 3.1: Način združevanja podatkov v skupine.

Poglavje 4

Implementacija sistema

4.1 Uporabljene tehnologije

Pri načrtovanju rešitve smo tehnologije izbrali tako, da bi si čimbolj olajšali delo pri zbiranju in agregiranju podatkov. Tako smo se odločili za Javo kot glavni programski jezik in MySQL podatkovno bazo za hranjenje podatkov. Za vizualizacijo podatkov smo uporabili programsko orodje R.

Java je objektno usmerjeni visokonivojski programski jezik. Zanj smo se odločili zato, ker nudi številne knjižnice, ki so nam olajšale delo. V našem primeru je bila uporabljena knjižnica OkHttp[5], ki ponuja metode za preprostejše grajenje HTTP poizvedb in ravnanje z odgovori. Prav tako smo uporabili knjižnico mldht[13], ki nam olajša komunikacijo z BitTorrent omrežjem. Pomaga nam z nižjimi nivoji specifikacije, nudi pomoč pri posodabljanju same usmerjevalne tabele, poskrbi za komunikacijo z drugimi vozlišči, prav tako pa nam ponuja objekte, ki predstavljajo preprostejše koncepte kot so sporočila in odgovori brezsledilniških poizvedb.

Razvojno okolje, ki smo ga uporabili je IntelliJ IDEA[4], orodje razvijalca JetBrains, ki nam med drugim ponuja avtomatsko barvanje in dopolnjevanje kode ter razhroščevalnik. Osnovna „Skupnostna“ verzija orodja je brezplačna

in je za namen te diplomske naloge priskrbela vse, kar smo potrebovali.

Omejitev pri izbiri podatkovne baze za našo uporabo ni bilo. MySQL smo izbrali, ker ponuja MySQL delovno mizo (ang. workbench), orodje, ki nam olajša delo s podatkovno bazo, pisanje poizvedb in vpogled v podatke. Ker smo uporabili JDBC gonilnik, ki poenoti vmesnik za dostop do podatkovne baze v jeziku Java, bi lahko med razvojem podatkovno bazo tudi kadarkoli zamenjali z manjšo spremembo v kodi. Tudi MySQL ima „Skupnostno“ verzijo, ki je brezplačna. R je skriptni jezik za manipulacijo in kalkulacijo podatkov ter grafični prikaz. Uporabili smo ga za izris grafov podatkov, ki smo jih pridobili.

4.2 Implementacija

Za analizo smo potrebovali podatke o tem, katere vire delijo vozlišča. Tega nismo mogli narediti neposredno, ker poizvedbe v omrežju tega ne omogočajo. Pomagali smo si s knjižnico mldht. Zbiranje podatkov o rojih je potekalo v treh korakih.

1. Zbiranje naključnih vozlišč v omrežju. To smo storili s pošiljanjem *find_node* zahtev na naključne identifikatorje vozlišč. Zanimalo nas je za katere vire imajo vozlišča podatke o rojih.

2. Vzorčenje podatkov o identifikatorjih virov, ki jih imajo vozlišča. Da bi pridobili seznam identifikatorjev virov, za katera ima vozlišče podatke o rojih, smo uporabili zahtevo za vzorčenje virov iz razširitve 51. Zahtevo smo poslali vozliščem zbranim v prvem koraku. Pridobljene identifikatorje virov smo si shranili v bazo.

3. Zbiranje podatkov o rojih zbranih virov. Podatke o rojih smo pridobili tako, da smo poslali *get_peers* zahtevo za vsak pridobljen identifikator vira. Tako smo pridobili podatke o tem, katera vozlišča delijo nek določen vir. Te podatke smo shranili v bazo.

Zbiranje vozlišč smo izvedli takole:

```
RPCServer rpc =
    dht.getServerManager().getRandomServer();

while (notProcessed.size() != 0) {
    KBucketEntry toProcess = notProcessed.get(0);
    notProcessed.remove(toProcess);

    //z obracanjem identifikatorja pridobimo
    //zelo nakljucen nov identifikator
    String keystr =
        new StringBuilder(toProcess.getID().toString(false))
        .reverse().toString();
    toProcess =
    new KBucketEntry(toProcess.getAddress(), new Key(keystr));

    Key key = new Key(keystr);
    Prefix prefix  = new Prefix(key, 150);
    key = prefix.first();

    NodeLookup nodeLookup =
        new NodeLookup(key, rpc, dht.getNode(), false);
    nodeLookup.start();
}
```

Vozlišča, ki smo jih pridobili in jih še nismo obdelali, smo dodali na seznam za procesiranje, prav tako pa smo z njimi izvedli zbiranje virov:

```
KeyspaceSampler sampler =
    new KeyspaceSampler
    (rpc, dht.getNode(), prefix, nodeLookup, null);

sampler.start();
```

S pridobljenimi identifikatorji virov smo izvedli še poizvedbo za roj:

```
for(String ih : unprocessedInfohashes) {
    PeerLookupTask peerLookup =
        new PeerLookupTask(rpc, dht.getNode(), new Key(ih));
    peerLookup.start();
}
```

Pridobljene vire smo nato barvno klasificirali. Na spletu smo našli več spletnih strani, ki so nam omogočale iskanje torentov. Poleg že prej omenjenih smo našli še:

- `https://katcr.to/`,
- `https://www.limetorrents.info/`,
- `https://torrentdownloads.unblockall.org/`,
- `https://torrentz2.eu/`,
- `https://torlock.unblockit.top/`,
- `https://zooqle.unblockit.top/` in
- `https://yts.mx/`

Vse te strani omogočajo iskanje torentov po imenu vira, nekatere strani pa omogočajo iskanje vira po identifikatorju. Te so bile za nas ključnega pomena, saj nismo imeli imen virov, ampak njihove identifikatorje. Našli smo štiri takšne in sicer:

- `http://btdig.com/`,
- `https://ext.to/`,
- `https://1337x.to/` in
- `https://btcache.me/`

Slednje so nam ponujale možnost, da smo preko URL-ja GET zahteve podali identifikator vira in nam vrnile podatke o viru, če je ta obstajal, drugače pa niso vrnile ničesar. Zadnje tri od the strani so omogočale, da smo na njih naredili GET zahtevo iz javanske kode, prva pa je naše zahteve blokirala. Pomagali smo si s knjižnico OkHttp, primer za eno od spletnih strani pa izgleda takole:

```
public TT checkIfKeyIsPublic(String infohash) {
    String website = null;

    //ext.to
    Request request = new Request.Builder()
            .url("https://ext.to/search/?q=" + infohash)
            .build();
    try (Response r = client.newCall(request).execute()) {
        website =  r.body().string();
    } catch (IOException e){
        e.printStackTrace();
    }
    if(!website.contains("No results found"))
        return (TT.WHITE);

    return TT.BLACK;
}
```

Odvisno od odgovora spletne strani smo vir označili kot črn ali bel. Da bi lahko bili še bolj prepričani, bi morali o torentu za identifikatorje virov vprašati več spletišč, vendar teh v času našega raziskovanja nismo odkrili.

Pri združevanju na nivoju vozlišč smo preverili, pri koliko različnih virov deli neko vozlišče. Prešteli smo vire povezane z vsakim vozliščem in vozliščem določili barvo.

Spletna stran `http://ip-api.com/` ponuja api za geolokacijo. V GET zahtevi ji podamo IP naslov, za katerega želimo podatke, vrne pa nam podatke o državi, ISP-ju in številki AS ter še druge podatke o lokaciji. Nato smo zbrali vozlišča povezana z neko državo, ISP in AS ter glede na barve vozlišč pobarvali še te.

Celotna koda programa je dostopna na `https://github.com/MaticSincek/lociranje-globokega-torenta`.

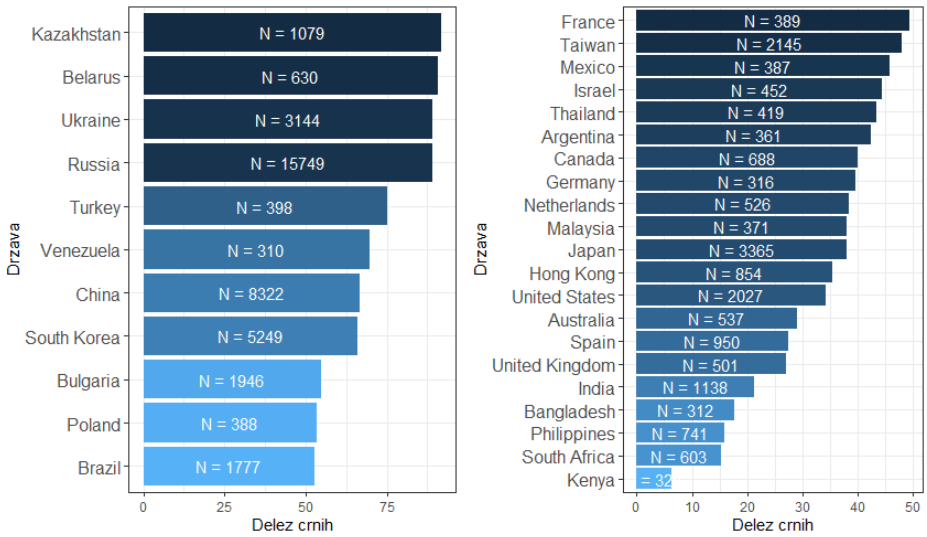
4.3 Rezultati

Pri fazi zbiranja podatkov smo iz omrežja zbrali 2848 različnih virov. Te vire je med sabo delilo 64352 uporabnikov, 7502 ponudnikov internetnih storitev in 5381 avtonomnih sistemov. Uporabniki so bili iz skupno 207 držav.

Zbiranje virov je bilo najhitrejše in je bilo opravljeno v manj kot eni uri. Čas iskanja podatkov o roju posameznega vira je trajal povpečno 16 sekund, skupno pa je zbiranje podatkov o rojih trajalo približno 12 ur in pol. Lokacijske poizvedbe za IP naslove so trajale okoli 46 sekund za 100 IP naslovov. S to hitrostjo so bili vsi naslovi obdelani v osmih urah in pol. Agregacija podatkov je bila opravljena prej kot v minuti.

Da bi preverili, ali smo vire uspešno pobarvali, smo delež črno pobarvanih virov primerjali z deležem črno pobarvanih virov v raziskavi [11] . V našem primeru je bil delež črnih virov 70,29%, v raziskavi pa je bil delež črnih 67.47%.

Da bi ugotovili, kje na svetu se nahaja največ uporabnikov, ki delijo črne vire, smo izrisali dva grafa. Prvega smo izrisali na podlagi podatkov o državah. Zaradi preglednosti smo izris omejili na države, ki so bile povezane z najmanj 300 viri. Na sliki 4.1 je graf, ki na x osi prikazuje delež črnih virov, ki jih delijo uporabniki. Na levi so države, ki pri našem vzorcu delijo



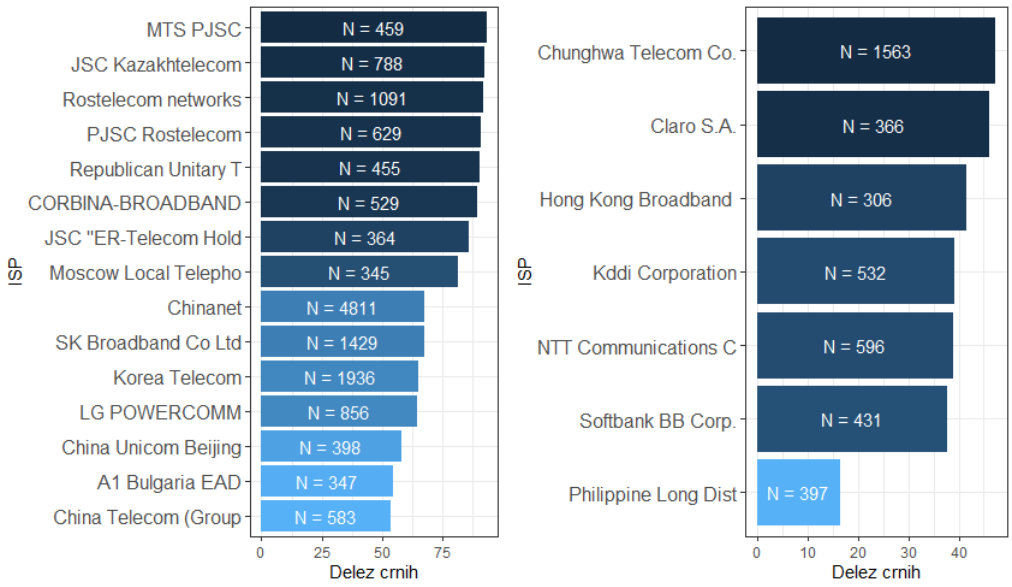
Slika 4.1: Delež zbranih črnih virov glede na državo.

več kot polovico črnih virov, na desni pa so tiste države, ki delijo manj kot polovico črnih. Najbolj so izstopale so države Kazahstan, Belorusija, Ukrajina in Rusija z deležem črnih virov 88% ali več, po drugi strani sta izstopali Filipini in Južna Afrika z deležem črnih virov 16% ali manj.

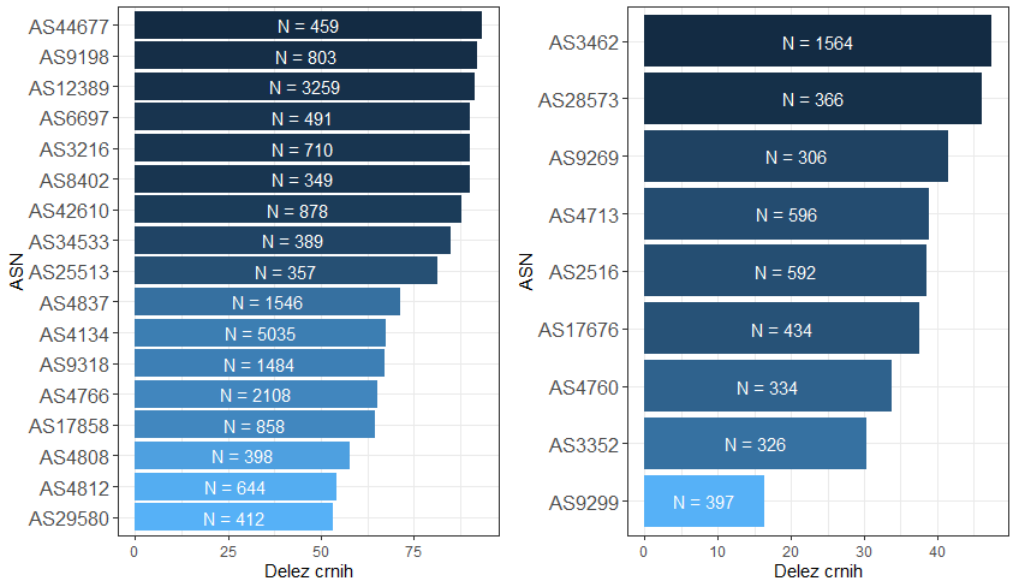
Na sliki 4.2 je graf, ki smo ga izrisali na podlagi podatkov o ponudnikih internetnih storitev. Tudi tukaj smo omejili ponudnike na tiste, ki so imeli pod sabo vsaj 300 virov. Na levo stran smo postavili tiste, ki so imeli več kot polovični delež črnih virov in na desno tiste, ki so imeli manj kot polovični delež črnih virov. Izstopa Filipinski ponudnik s 17 procentnim deležem črnih virov, veliko pa je tudi ponudnikov z 90 in več procentnim deležem črnih virov.

Pri grafu, na katerem so viri zbrani glede na avtonomni sistem, je stanje podobno. Največji delež črnih virov imajo AS, ki so iz držav, ki imajo visok delež črnih virov.

Rezultati kažejo na to, da je v času analize v nekaterih državah in pri nekaterih ponudnikih storitev delež deljenih vsebin, ki so črne, zelo visok, v primerjavi z velikostjo globokega torenta v omrežju.



Slika 4.2: Delež zbranih črnih virov glede na ISP.



Slika 4.3: Delež zbranih črnih virov glede na AS.

4.4 Vrednotenje rezultatov

Ocena deleža globokega torenta na našem vzorcu podatkov iz omrežja je 70.3%. V raziskavi [11] so delež globokega torenta ocenili na 67%, zato lahko rečemo, da je bil naš vzorec reprezentativen.

Ker ne moremo preiskati vseh spletnih strani za odkrivanje torentov, za določen vir nismo mogli zagotovo reči, da je črn, če na teh straneh ne moremo najti torenta, ki ga opisuje. Takšni viri bi lahko bili beli, mi pa smo jih poimenovali kar črni, pri čemer se zavedamo, da je zato pri rezultatu nekoliko odstopanja.

Pri zbiranju so bile nekatere države in nekateri ponudniki storitev zastopani bolj kot drugi. Velik faktor pri tem je seveda njihova velikost, vseeno pa so bile zaradi naključnosti nekatere države in ponudniki storitev premalo zastopani. Zato smo pri vizualizaciji izpostavili samo države, ki so imele število vozlišč višje od neke meje, ki smo jo določili z opazovanjem vzorca. V raziskavi [11] so z opazovanjem posameznih območij omrežja ocenili, da je v celotnem omrežju približno 41216000 različnih virov, medtem, ko smo jih mi odkrili 2848. Z daljšim opazovanjem omrežja bi pridobili na količini podatkov in posledično na njihovi točnosti.

Poglavje 5

Zaključek

V diplomski nalogi smo osnovali in zgradili sistem, ki zna pridobiti identifikatorje virov, ki se ponujajo v BitTorrent omrežju in podatke o vozliščih, ki jih ponujajo. Pri tem smo dobili dober vpogled v delovanje omenjenega protokola. Sistem zna bele vire ločiti od črnih, kjer so beli viri tisti, ki so oglaševani na spletnih straneh za oglaševanje le teh. Poleg tega zna vozlišča združiti glede na podatke o geolokaciji in nam tako prikaže deleže globokega torenta v odvisnosti od države, ponudnika internetnih storitev in avtonomnega sistema.

Potrdili smo oceno raziskave[11], da globoki torent predstavlja dve tretjini vseh torentov. Zdi se nam, da je zaradi velike prisotnosti globokega torenta raziskovanje takšnega fenomena še toliko bolj pomembno.

Ugotovili smo, da je v nekaterih državah delež črnih virov, ki se delijo, večji kot v drugih, prav tako je različno popularen med ponudniki internetnih storitev in v različnih avtonomnih sistemih. Z našo raziskavo smo dokazali, da je nadziranje globokega torenta do neke mere možno. Z daljšim opazovanjem in večjimi količinami podatkov bi bili rezultati še bolj natančni.

Literatura

[1] The 8472. Dht infohash indexing. Technical report, BitTorrent Inc., 2016. [Dostopano: 16.5. 2020].

[2] Bram Cohen. Incentives build robustness in bittorrent. Technical report, BitTorrent Inc., 2003. [Dostopano: 25.6. 2020].

[3] Bram Cohen. The bittorrent protocol specification. Technical report, BitTorrent Inc., 2008. [Dostopano: 16.5. 2020].

[4] Jesse Wilson et al. Kozlova, anna and gromov, peter et al. Dosegljivo: `https://github.com/JetBrains/intellij-community`, 2019. [Dostopano: 8.9.2020].

[5] Jesse Wilson et al. okhttp. Dosegljivo: `https://github.com/square/okhttp/`, 2019. [Dostopano: 8.9.2020].

[6] J. Fonseca, B. Reza, and L. Fjeldsted. Bittorrent protocol version 1.0. Technical report, University of Copenhagen, 2005. [Dostopano: 22.6. 2020].

[7] BitTorrent Inc. Azureus - java bittorrent client. Dosegljivo: `https://web.archive.org/web/20061201095553/http://azureus.sourceforge.net/changelog.php`, 2006. [Dostopano: 22.8. 2020].

[8] Andrew Loewenstern and Arvid Norberg. DHT protocol. Technical report, BitTorrent Inc., 2008. [Dostopano: 16.5. 2020].

[9] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. Technical report, New York University, 2002. [Dostopano: 22.6. 2020].

[10] Charles P. Fry and Michael K. Reiter. Really truly trackerless bittorrent. Technical report, School of Computer Science, Carnegie Mellon University, 2006.

[11] A. Rafael Rodriguez-Gomez, Gabriel Macia-Fernandez, and Alberto Casares-Andres. On understanding the existence of a deep torrent. *IEEE Communications Magazine*, 55(7):64–69, 2017.

[12] D. Sui, J. Caverlee, and D. Rudesill. The deep web and the darknet: A look inside internet's massive blackbox. Technical report, Wilson Center, 2015.

[13] the8472 et al. mldht. Dosegljivo: `https://github.com/the8472/mldht`, 2017. [Dostopano: 8.9.2020].

[14] Clive Thompson. The bittorrent effect. Dosegljivo: `https://www.wired.com/2005/01/bittorrent-2/`, 2005. [Dostopano: 16.5. 2020].