KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Dainius Četrauskas

ŠIFRUOTŲ BITTORRENT SRAUTŲ APTIKIMAS

Magistro darbas

Darbo vadovas
dr. Ingrida Lagzdinytė

Kaunas, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Dainius Četrauskas

ŠIFRUOTŲ BITTORRENT SRAUTŲ APTIKIMAS

Magistro darbas

Recenzentas

doc. dr. ….

2010-05-...

Vadovas

dr. Ingrida Lagzdinytė

2010-05-...

Atliko

IFN-8/3 gr. stud.

Dainius Četrauskas

2010-05-...

Kaunas, 2010

SUMMARY

For several years we've witnessed an incredible growth in use of peer-to-peer (P2P) applications. Recent Internet traffic studies show no signs of this process slowing down. The unparalleled popularity of P2P applications and their bandwidth-intensive nature suggest that P2P traffic can have a significant effect on the underlying network infrastructure. It is therefore important to understand, characterize and accurately identify this type of traffic.

It is known by now that there are customer ready P2P traffic detection solutions gaining popularity among various ISPs. However while they might be affordable by larger ISPs, ability of smaller service providers to cope with P2P traffic is rather small.

Purpose of this work is to present, analyze and test specific characteristics of encrypted and non-encrypted BitTorrent P2P traffic, allowing it to be detected and classified. Work includes various P2P traffic detection techniques, which might not be usable as a standalone solution, but by combining them into a single framework it could be possible to achieve high BitTorrent P2P traffic detection ratio. Presented techniques vary by their nature starting with techniques based on packet payload inspection and finishing with techniques based on heuristic approach.

TURINYS

Įvadas

Peer-to-peer[1] tinklas - tai decentralizuotas duomenų keitimosi kompiuterinių įrenginių tinkle sistemos modelis. Tokiame tinkle kiekvienas tinklo elementas turi tas pačias galimybes kaip ir kiti tinklo elementai, t.y. gali atlikti tiek serverio, tiek kliento funkcijas.

Vienas iš didžiausių peer-to-peer tinklo privalumų yra resursų išskaidymas. Prie egzistuojančio peer-to-peer tinklo prisijungus naujam vartotojui dažniausiai padidėja ne tik tinklo potencialo išnaudojimas, bet ir prieinamų resursų kiekis.

Peer-to-peer principu veikiantys tinklai nėra naujiena informacinių technologijų srityje, pakanka prisiminti USENET [1] tinklą, gimusį 8-ojo dešimtmečio pabaigoje. Tačiau peer-to-peer termino ir po juo slypinčios technologijos panaudojimo duomenų mainams kompiuterių tinkle, taip kaip mes tai suprantame šiandien, atsiradimo pradžia galime laikyti 1999-uosius metus, kuomet Shawn Fanning sukūrė peer-to-peer principu veikiančią failų mainų programą pavadinimu „Napster“ [2]. „Napster” eiliniams interneto vartotojams suteikė galimybę internetu keistis turimais duomenimis.

Nuo 1999-ųjų metų iki dabartinių dienų peer-to-peer principu paremtų technologijų panaudojimas smarkiai išplito ir peržengė failų mainų ribas, bei atrado savo nišą tokiuose projektuose kaip SETI@home [3].

Nepaisant peer-to-peer technologijų teikiamos naudos tinklo vartotojams ir jo privalumų išnaudojant potencialius tinklo resursus, buvo susidurta su pasaulinio tinklo apkrovos problemomis, kurių mastas vis labiau ryškėja kasmet augant peer-to-peer principu paremtų sistemų skaičiui ir plintant jų naudojimui [4] [5].

Kadangi duomenų siuntimasis peer-to-peer tinklais dažniausiai yra grupinis procesas, t.y. failas vienu metu gali būti siunčiamas iš daugelio vartotojų ir besisiunčiantis vartotojas tą patį failą gali dalintis su kitais peer-to-peer vartotojais tuo pat metu, tad augant peer-to-peer vartotojų skaičiui atitinkamai didėja tinklo apkrova. Nevaldomą situaciją galima lyginti su vidine DDoS ataka interneto paslaugos tiekėjo tinkle, kuomet dėl resursų trūkumo tampa labai sunku arba beveik neįmanoma naudotis kitomis paslaugomis: VoIP, Video Konferencijomis, Video transliacijomis. Netgi elementarus naršymas po interneto svetaines gali tapti nepakenčiamai lėtas. Interneto paslaugos tiekėjai, siekdami išvengti minėtų pasekmių, pasitelkė metodus ir priemones galinčias aptikti peer-to-peer srautus ir juos riboti.

Tai buvo lauktas žingsnis, žvelgiant iš interneto paslaugos tiekėjų perspektyvos, iššaukęs

1 Peer-to-peer – lietuviškas žodžių junginio atitikmuo yra „taškas-į-tašką“. Darbe naudojami abu terminai, tiek angliškas, tiek lietuviškas variantai.

nelauktą peer-to-peer tinklus naudojančių interneto vartotojų bendruomenių reakciją. Peer-to-peer principu veikiančios failų keitimosi programinės įrangos gamintojai pristatė naujas peer-to-peer klientų versijas palaikančias perduodamų duomenų šifravimą, tokiu būdu praktiškai užkirsdami interneto paslaugos tiekėjams galimybę realiu laiku aptikti ir valdyti peer-to-peer srautus. Remiantis įvairia statistika peer-to-peer duomenų srautai skirtinguose pasaulinio tinklo segmentuose sudaro nuo 30% iki 70% visų perduodamų duomenų kiekio [4] [5] [6].

Pastaraisiais metais nepaprastai išaugo tyrimų, orientuotų į šifruoto peer-to-peer srauto aptikimą, skaičius. Ir nors buvo rastos priemonės[2], iš dalies padedančios interneto paslaugos tiekėjams gana tiksliai ir realiu laiku identifikuoti peer-to-peer srautus, ir juos valdyti, sprendimai yra brangūs, dažnai neįperkami mažoms ir vidutinėms įmonėms bei viešoms organizacijoms.

Šis darbas koncentruotas ties BitTorrent P2P protokolo tyrimu, siekiant nustatyti pagrindines BitTorrent protokolą naudojančių P2P tinklų, ir jais perduodamų srautų savybes, bei jas charakterizuoti.

Pasinaudojant nustatytomis ir apibrėžtomis charakteristikomis, sudarytas šifruotų BitTorrent srautų aptikimo modelis. Skyriuje nr. 4 *„Eksperimentinis šifruotų Bittorret srautų aptikimo sistemos tyrimas“* eksperimentiniu keliu tiriamas modelio veiksmingumas, gebėjimas atpažinti BitTorrent protokolu perduodamus šifruotus duomenų srautus.

Pagrindinis darbo tikslas yra sudaryti ir ištestuoti metodą, leidžiantį aptikti šifruotus BitTorrent srautus. Sudarytą metodą ištestuoti, patvirtinti arba paneigti jo veiksmingumą.

Darbo struktūra:

- Pirmame skyriuje atlikta šifruotų peer-to-peer srautų aptikimo problemos analizė. Pateiktas BitTorrent protokolo, naudojamo šifruotų peer-to-peer duomenų perdavimui, apibrėžimas, aprašyti jo veikimo principai. Apžvelgti alternatyvių sistemų, naudojamų šifruotų peer-to-peer srautų aptikimui, variantai. Įvertinami jų privalumai ir trūkumai.
- Antrame skyriuje pateikiamas šifruotų BitTorrent srautų aptikimo sistemos modelis. Aprašomi reikalavimai modeliui, siūlomi algoritmai šifruoto BitTorrent srauto aptikimui.
- Trečiame skyriuje pateikiamas Šifruotų BitTorrent srautų aptikimo sistemos realizacijos aprašas. Aprašoma sukonstruotos sistemos testavimo metodika.
- Ketvirtame skyriuje pateikiami sistemos eksperimentinio tyrimo rezultatai.
- Paskutiniame skyriuje pateikiamos suformuluotos darbo išvados.

2 Sprendimai apžvelgiami skyriuje 1.3 *„Alternatyvių sistemų analizė“*

1. Užduoties analizė

1.1 Problema

Nuolat augant tinklų ir jų vartotojų skaičiui, stipriai auga kompiuterių tinklų linijų apkrovimas. peer-to-peer duomenų srautai dažnai sudaro didžiąją kompiuterių tinklu perduodamų duomenų dalį [4] [5] [6], tačiau interneto paslaugų tiekėjai praktiškai neturi priemonių, kurias galėtų naudoti šifruoto peer-to-peer srauto aptikimui realiu laiku. Didelės kompiuterių tinklų apkrovos, kurias sukelia peer-to-peer duomenų srautai, neigiamai įtakoja kitų paslaugų (VoIP, Video konferencijos, Video transliacijos) kokybę, o dažnai ir visiškai apriboja tokių paslaugų pateikiamumą ir prieinamumą. Peer-to-peer duomenų srautai dar kitaip vadinami DDoS atakų atmaina, kylančia iš tinklo vidaus vartotojų.

Darbe siekiama ištirti šifruotų peer-to-peer duomenų srautų aptikimo galimybes bei pasiūlyti metodiką, gebančią tokius srautus aptikti ir valdyti.

Keliami klausimai:

- ar įmanoma aptikti šifruotus BitTorrent srautus analizuojant sukauptą tinklo srautą?
- ar įmanoma aptikti šifruotus BitTorrent srautus analizuojant tinklo srautą realiu laiku?
- koks metodas gali būti taikomas srautų aptikimui?

Eksperimentas skaidomas į du etapus. Pirmo etapo metu bus analizuojama aibė iš anksto surinktų ir standžioje laikmenoje saugomų statinių kompiuterių tinklu perduodamų duomenų srautų rinkinių. Analizės metu siekiama:

- Patvirtinti iškeltas hipotezes apie BitTorrent P2P srautų savybes.
- Patvirtinti oficialioje P2P BitTorrent protokolo specifikacijoje pateikiamas savybes [7].

Antras etapas skirtas dinamiškai kompiuterių tinklu perduodamų duomenų analizei. Pagrindiniai siekiai:

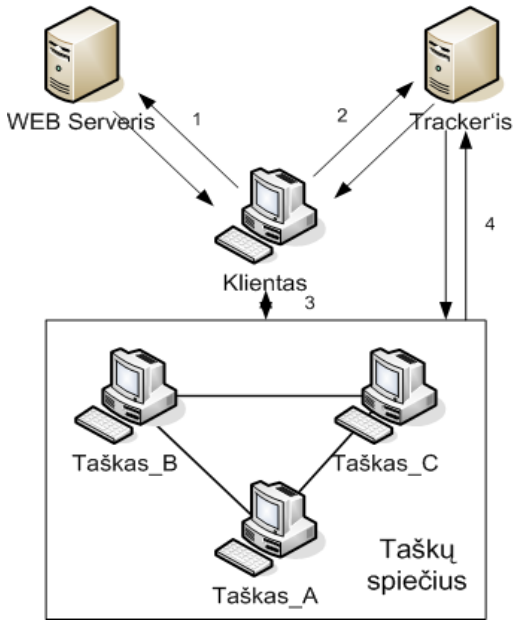
- Aptikti nešifruotus Bittrorrent P2P srautus pagal sudarytą metodiką, patikrinant jos veiksmingumą,
- Aptikti šifruotus Bittrorrent P2P srautus pagal sudarytą metodiką, patikrinant jos veiksmingumą,

1.2 BitTorrent protokolo savybės

Šioje darbo dalyje pateikiamas BitTorrent protokolo aprašymas. Aprašymas parengtas remiantis oficialia protokolo dokumentacija [7] ir darbo metu atliktos BitTorrent protokolo srautų analize.

1.2.1 BitTorrent protokolo architektūra ir duomenų mainai

BitTorrent protokolas yra koncentruotas ties centralizuoto peer-to-peer tinklo dizainu. BitTorrent protokolas centrinio serverio pagalba leidžia susijungti daugeliui klientų ir vienu metu keistis mažais to paties failo gabalais. Centrinis serveris vadinamas tracker'iu[3], nes saugo ir atnaujina informaciją apie juo besinaudojančius klientus. Paveikslėlyje nr. 1 pateikiama principinė BitTorrent protokolą naudojančio peer-to-peer tinklo schema.



Pav. 1: Principinė BitTorrent tinklo schema

1 – klientas iš WEB serverio parsisiunčia *torrent* failą, kuriame saugomas tracker'io adresas. 2 – klientas jungiasi prie tracker'io ir parsisiunčia sąrašą taškų, kurie dalijasi jo ieškomu failu. 3 – klientas jungiasi prie taškų, kurie dalijasi jo ieškomu failu, pradeda siųstis failą mažais gabaliukais ir tuo pat metu jais dalintis su kitais taškais, besisiunčiančiais tą patį failą. 4 – visi to paties failo mainuose dalyvaujantys taškai palaiko ryšį su tracker'iu. Tokiu būdu jie nuolat sužino apie naujai prisijungusius taškus ir informuoja tracker'į apie savo egzistavimą.

Remiantis aukščiau pateikta BitTorrent tinklo schema, galime išskirti pagrindinius BitTorrent tinklo elementus:

- Veikiantis WEB serveris, naudojamas kaip torrent failų talpykla.
- Statinis „metainfo“ failas, dar žinomas torrent failo vardu.
- BitTorrent tracker'is, atsakingas už susijungimų koordinavimą tarp skirtingų taškų.
- Taškas, besidalijantis resursu.

3 Artimiausias termino atitikmuo lietuvių kalboje – seklys. Tačiau angliškas terminas labiau paplitęs ir yra lengviau suvokiamas diskutuojant P2P tinklų temomis. Siekiant darbą pateikti kuo suprantamiau, apsistota ties angliškąja termino versija.

- Taškas, naudojantis ir/ar besidalijantis resursu.
- Interneto naršyklė, arba veikianti alternatyva torrent failo parsisiuntimui.

Torrent failas

Torrent failas yra būtina BitTorrent tinklo dalis. Jį galime lyginti su raktu, leidžiančiu klientui siųstis tam tikrą failą ar failus. Daugeliu atvejų klientas privalo turėti torrent failą susijusį su resursu, kurį pasiekti klientas nori. Torrent failas yra paprastas tekstinis failas, kuriuo galima keistis talpinant WEB serveriuose, FTP serveriuose, persiunčiant elektroniniu paštu ir t.t. Tekstas torrent failo viduje yra užkoduotas naudojant Bencoding [7], saugo įvairią informaciją apie naudotinus tracker'ius ir apie patį resursą, kuriuo dalintis jis skirtas. Informacija, saugoma torrent faile, reikalinga klientui norinčiam prisijungti prie tracker'io ir pradėti siųstis duomenis iš taškų, nustatyti siunčiamo failo vardą ir dydį.

Torrent failas gali būti naudojamas aprašyti vieną failą, kuriuo norima dalintis, arba aibę failų. Siekiant papildomo lankstumo failų mainuose tarp taškų, kiekvienas failas skaidomas į fiksuoto to paties dydžio gabaliukus (paskutinio gabaliuko dydis gali skirtis), kurie aprašomi torrent faile su kiekvienam iš jų priskirtu unikaliu ID, sugeneruotu naudojant SHA-1 algoritmą [8].

Detalų torrent failo struktūros aprašą galima rasti BitTorrent protokolo dokumentacijoje [7].

BitTorrent tracker'is

Tracker'is yra pagrindinė BitTorrent tinklo dalis. BitTorrent tracker'is yra HTTP/HTTPS servisas, gebantis priimti ir apdoroto HTTP GET užklausas. Tracker'is priima prisijungimus iš įvairių BitTorrent protokolą palaikančių klientų ir iš esmės yra atsakingas už susijungimų koordinavimą tarp skirtingų taškų, norinčių dalintis/pasiekti vieną ar kitą resursą.

Klientui parsisiuntus ir nuskaičius torrent failą, kliento programinė įranga susijungia su tracker'iu aprašytu torrent faile, bei pareikalauja ieškomo failo, tam naudodama unikalų ID iš torrent failo. Tracker'is nuolat atnaujina vidinę duomenų bazę, kurioje saugoma informacija apie visus klientus besidalijančius tam tikru failu ar failais. Tracker'is, atpažinęs prisijungusį klientą, persiunčia jam sąrašą taškų, kurie dalijasi jo ieškomu failu. Sąrašas sudarytas iš IP adresų ir prievadų porų, apibrėžiančių konkretų tašką. Turėdamas šią informaciją, klientas jau gali jungtis prie kitų taškų ir pradėti duomenų mainus. Tracker'io ir klientų komunikacija plačiau aptariama skyriuje 1.2.2.

Resursais besidalijantys taškai

BitTorrent tinklo klientai (taškai) skirstomi į dvi kategorijas: seeder'ius ir leecher'ius[4]. Seeder – klientas, kuris tik dalijasi failu, leidžia iš savęs failą siųstis kitiems taškams. Seeder'is turi pilną failo kopiją, jam nereikia siųstis duomenų iš kitų taškų. Leecher – normalus klientas, taškas, besisiunčiantis failą ir tuo pat metu besidalijantis turimomis failo dalimis su kitais taškais. Seeder'ių ir leecher'ių visuma suformuoja taškų spiečių[5]. Tam, kad prasidėtų duomenų mainai BitTorrent tinkle, turi egzistuoti bent vienas aktyvus taškas, turintis pilną failo kopiją ir ja besidalijantis (seeder'is).

Visi aktyvūs taškai nuolatos palaiko ryšį su tracker'iu. Tokiu būdu jie gauna informaciją apie naujai prisijungusius taškus.

1.2.2 Susijungimai tarp tracker'io ir taškų

Klientas, nustatęs BitTorrent tracker'io adresą, bandys prisijungti prie jo ir gauti taškų, besidalijančių ieškomu resursu, sąrašą. Susijungimas vykdomas naudojant UDP arba TCP transporto lygmens protokolą, bei į nurodytą tracker'io IP adresą ir prievadą siunčiant HTTP GET užklausą. Užklausos pavyzdys pateikiamas paveiksliuke nr. 2.

```
GET /announce?info_hash=%C7%A1S%D2%D6%D9%C0r%5B7X%A8%3B%E3%0E%F1MV%01%EB\
&peer_id=-AZ4314-V0DkAk69lgBo&supportcrypto=1&port=52540&azudp=52540&upl\
oaded=0&downloaded=0&left=167772160&corrupt=0&event=started&numwant=75&n\
o_peer_id=1&compact=1&key=snDGLq6n&azver=3 HTTP/1.1

User-Agent: Azureus 4.3.1.4;Windows XP;Java 1.6.0_19
Connection: close
Accept-Encoding: gzip
Host: tracker.archlinux.org:6969
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
```

Pav. 2: BitTorrent tracker'iui siunčiama kliento užklausa. Įžambus brūkšnys pirmose 3-se eilutėse žymi eilutės tęsinį

Svarbiausia yra GET eilutė. Ji tracker'iui nusiunčia svarbius statistinius duomenis. Tracker'is šiuos duomenis naudoja kliento parsiųstų / išsiųstų duomenų santykiui skaičiuoti ir nustatyti, kokių taškų sąrašą klientas turėtų gauti. Žemiau pateikiamas kiekvieno užklausoje esančio rakto aprašymas:

4 Artimiausi terminų atitikmenys lietuvių kalboje: seeder – sėjikas, leecher – siurbikas. Tačiau angliškas terminas labiau paplitęs ir yra lengviau suvokiamas diskutuojant P2P tinklų temomis. Siekiant darbą pateikti kuo suprantamiau, apsistota ties angliškomis terminų versijomis.

5 Taškų spiečius: Swarm of Peers (Eng,)..

- **info_hash** – 20 baitų ilgio simbolių eilutė, sugeneruota naudojant SHA-1 maišos funkciją, kurios įvesties duomenims naudojama torrent failo „info“ sekcija [7]. Simboliai papildomai užkoduoti naudojant URL Escape kodus [9]. Tai leidžia ASCII simbolius persiųsti naudojant šešioliktainę formą.
- **peer_id** – 20 baitų ilgio simbolių eilutė, atsitiktiniu būdu sugeneruota paties kliento ir naudojama jam identifikuoti. Kai kurie klientai prijungia statinę dalį prie eilutės. Pavyzdžiui šiuo atveju naudojama Vuze (buvęs pavadinimas: Azureus) programinė įranga prijungia simbolius „-AZ4314-“, kur 4314 – naudojamos programinės įrangos versija.
- **port** – prievado numeris, kurie BitTorrent klientas naudoja įeinantiems BitTorrent prisijungimams priimti.
- **azudp** – papildomas raktas, naudojamas Vuze programinės įrangos.
- **uploaded** – baitas išreikštas duomenų kiekis, kuriuos klientas persiuntė kitiems taškams. Skaičius išreikštas dešimtainėje sistemoje.
- **downloaded** - baitas išreikštas duomenų kiekis, kuriuos klientas parsisiuntė iš kitų taškų. Skaičius išreikštas dešimtainėje sistemoje.
- **left** - baitas išreikštas duomenų kiekis, kurį klientas vis dar turi parsisiųsti. Skaičius išreikštas dešimtainėje sistemoje.
- **corrupt** – papildomas raktas, naudojamas Vuze programinės įrangos.
- **Event** – papildomas raktas, skirtas pranešti kliento susijungimo fazę šio konkretaus torrent'o atveju. Galimos trys reikšmės:
  - *started* – siunčiamas inicijuojant naują siuntimosi procesą.
  - *completed* – siunčiamas, kai norima tracker'iui pranešti, kad duomenų siuntimas baigtas.
  - *stopped* – siunčiamas, kai norima tracker'iui pranešti, kad duomenų siuntimasis sustabdytas
  - *empty* – nieko nereiškia.
- **numwant** – papildomas raktas, nurodantis tracker'iui skaičių, apibrėžiantį kiek taškų turi sudaryti taškų sąrašą, siunčiamą klientui.
- **no_peer_id** – papildomas raktas, nebenaudojamas.
- **compact** – papildomas raktas, nurodantis tracker'iui naudoti kompaktiškas žinutes atsakant, t.y. nesiųsti atsakyme kitus taškus identifikuojančių peer_id.
- **key** – papildomas raktas. Tracker'iui siunčiamas papildomas unikalus identifikatorius, kuris nebus persiųstas kitiems taškams.

- **azver -** papildomas raktas, naudojamas Vuze programinės įrangos.

Gavęs GET užklausą, tracker'is atsako klientui siųsdamas sąrašą taškų, kurie dalijasi jo ieškomu resursu.

Lentelėje nr. 1 pateikiamas standartinių prievadų, kuriuos naudoja BitTorrent tracker'iai, sąrašas. Sąrašas sudarytas remiantis programinės įrangos dokumentacija [11, 12].

Prievadas	**Protokolas**	**BitTorrent trackeris**
6969	TCP, UDP	Vuze (Azureus), OpenTracker
7000	TCP	Vuze (Azureus)
2710	TCP, UDP	XBT BitTorrent Tracker
-	-	-

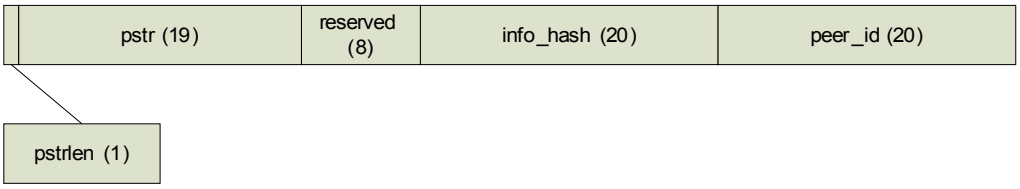
Lentelė 1: Standartiniai BitTorrent trackerių prievadai

Bendravimas tarp tracker'io ir taško gali būti šifruotas [10] ir nešifruotas. Nėra žinoma būdų, leidžiančių tiksliai aptikti šifruotus tracker'io ir taško susijungimus. Dalį šifruotų susijungimų galima aptikti tikrinant ar naudojamas vienas iš standartinių BitTorrent tracker'ių prievadų (lentelė nr. 1). Nešifruotus susijungimus tarp taško ir tracker'io galima aptikti atliekant paketų turinio analizę. Tracker'io ir taško susijungimų aptikimas detaliau nagrinėjamas skyriuje 2.6 *„Skiriamieji BitTorrent srautų bruožai“*.

1.2.3 Susijungimai tarp taškų

Norėdamas pradėt duomenų mainus, BitTorrent klientas privalo jungtis su taškais, kurių sąrašą gavo iš tracker'io. Susijungimai tarp taškų yra simetriniai, t.y. į abi puses siunčiamų žinučių formatas yra toks pat, duomenys taip pat gali keliauti į abi puses. Bendravimui tarp dviejų taškų naudojamas Peer Wire protokolas (PWP) [7], Dalis BitTorrent klientų naudoja TCP prievadus, nustatytus pagal nutylėjimą (lentelė nr. 3), tačiau BitTorrent programinė įranga leidžia vartotojui šį parametrą pakeisti į norimą reikšmę arba reikšmę, parinktą atsitiktiniu būdu.

Prieš pradedant duomenų mainus, du susijungiantys taškai pirmiausiai turi atlikti „rankos paspaudimą“. Susijungimo iniciatorius privalo išsiųsti savo rankos paspaudimą pačia pirmąja žinute. Rankos paspaudimo žinutės ilgis yra (49+ilgis(pstr)) baitų. Detalus žinutės formatas pateikiamas paveikslėlyje nr. 3.



Pav. 3: Rankos paspaudimo žinutės formatas (BitTorrent protokolo versija: 1.0)

pstrlen – žinutės dalis, skirta aprašyti pstr laukelio ilgiui. Dalies ilgis: 1 baitas.

pstr – simbolių eilutė, identifikuojanti protokolą. BitTorrent versijos 1.0 protokolui apibrėžti naudojama simbolių eilutė: „BitTorrent protocol“. Jeigu siunčiamas kitoks protokolo identifikatorius, nutolęs taškas bandymą prisijungti atmeta. Dalies ilgis: 19 baitų.

reserved – nenaudojama žinutės dalis. Paprastai užpildoma nuliais. Dalies ilgis: 8 baitai.

info_hash – simbolių eilutė, sugeneruota naudojant SHA-1 maišos funkciją. Tai ta pati simbolių eilutė, perduodama tracker'iui taško prisijungimo metu. Dalies ilgis: 20 baitų.

peer_id – simbolių eilutė, naudojama kaip unikalus BitTorrent taško identifikatorius. Tai ta pati simbolių eilutė, perduodama tracker'iui taško prisijungimo metu. Dalies ilgis: 20 baitų.

Žemiau pateikiama realaus paketo analizė, kurį siųsdamas klientas atlieka rankos paspaudimą.

```
162   46.211777   10.10.10.186      85.243.107.49     TCP   [TCP segment of
a reassembled PDU]

0000   00 0c 42 24 e2 db 00 17 9f 25 23 06 08 00 45 00  ..B$.....%#...E.
0010   00 6c 67 21 40 00 80 06 bd 82 0a 0a 0a ba 55 f3  .lg!@.........U.
0020   6b 31 27 21 12 36 3d a4 2e 32 77 e8 a6 f3 50 18  k1'!.6=..2w...P.
0030   ff ff ed 09 00 00 13 42 69 74 54 6f 72 72 65 6e  .......BitTorren
0040   74 20 70 72 6f 74 6f 63 6f 6c 80 00 00 00 00 13  t protocol......
0050   00 04 e3 18 0b 73 a4 a5 85 7c cb 33 4f a6 d5 87  .....s...|.3O...
0060   ac 77 53 47 80 c1 2d 41 5a 34 33 31 34 2d 31 4d  .wSG..-AZ4314-1M
0070   74 42 6e 67 6a 6e 61 69 74 64                    tBngjnaitd
```

Pav. 4: BitTorrent rankos paspaudimas. Analizuojamo paketo turinys pateiktas šešioliktainiu ir tekstiniu formatu

Duomenų dydis baitais	**Šešioliktainė duomenų išraiška**	**Aprašymas**
1	0x13	Aprašo sekančios žinutės dalies ilgį (19 baitų).
19	0x426974546F7272656E742070726F746F636F6C	Protokolo identifikatorius „BitTorrent protocol“.

Duomenų dydis baitais	**Šešioliktainė duomenų išraiška**	**Aprašymas**
8	0x8000000000130004	Nenaudojama žinutės dalis.
20	0xE3180B73A4A5857CCB334FA6D587AC77534780C1	simbolių eilutė, sugeneruota naudojant SHA-1 maišos funkciją.
20	0x2D415A343331342D314D74426E676A6E61697464	simbolių eilutė, naudojama kaip unikalus BitTorrent taško identifikatorius.

Lentelė 2: BitTorrent rankos paspaudimas. Paketo turinio analizė

Egzistuojančios BitTorrent peer-to-peer srautų aptikimo sistemos (plačiau aptariamos skyriuje 1.3 *„Alternatyvių sistemų analizė”*) gali aptikti srautus tarp taškų, jeigu jie nėra užšifruoti. Srautai tarp taškų aptinkami analizuojant paketų turinį ir ieškant BitTorrent rankos paspaudimo. Jeigu perduodama informacija yra šifruota, BitTorrent rankos paspaudimo aptikti neįmanoma.

Atlikus BitTorrent protokolą palaikančios programinės įrangos apžvalgą, sudarytas klientų naudojamų prievadų sąrašas. Lentelėje nr. 3 pateikiamus prievadus išvardinti klientai naudoja pagal nutylėjimą.

P2P aplikacija	Naudojami prievadai	Vartotojas gali pakeisti?
Vuze (Azureus)	TCP: 6881-6889, 44797	Taip

Lentelė 3: BitTorrent klientų pagal nutylėjimą naudojami prievadai

1.2.4 DHT protokolas

DHT[6] [13] protokolas – tai BitTorrent protokolo praplėtimas, kurio pagalba įmanoma koordinuoti duomenų mainus tarp skirtingų taškų be standartinio tracker'io (plačiau apie BitTorrent tracker'ius skyriuje 1.2.1 *“BitTorrent protokolo architektūra ir duomenų mainai”*) pagalbos. DHT tinklas yra sudarytas iš viršūnių, kurios saugo informaciją BitTorrent tinklo klientų/taškų buvimo vietą. BitTorrent klientai kartu yra ir DHT tinklo viršūnės, kuriomis galima pasinaudoti susijungimams su kitomis DHT viršūnėmis, tokiu būdu ieškant BitTorrent tinklo taškų buvimo vietos, iš kurių galima siųstis ieškomą failą naudojant BitTorrent

6 DHT – Distributed Hash Table. BitTorrent naudoja Kademila paremtą DHT protokolą [14]

protokolą. Komunikacijai tarp DHT viršūnių naudojamas UDP protokolas. BitTorrent klientų programinė įranga suteikia vartotojui galimybę DHT palaikymą įjungti arba išjungti.

1.3 Alternatyvių sistemų analizė

1.3.1 PRX Traffic Manager

„PRX Traffic Manager“ – tai bendrovės *„ipoque“* platinama visapusiška ir rentabili kompiuterių tinklo duomenų srautų valdymo priemonė, tinklų administratoriams suteikianti galimybę stebėti ir valdyti duomenų srautus, valdymą granuliuojant pagal aplikacijas ar vartotojus. *„PRX Traffic Manager“* aptinka aplikacijas naudodama visapusiško tinklo duomenų paketų tikrinimo OSI 7-ame lygmenyje ir srautų elgesio ypatumų analizės technikas. Įrenginys palaiko visus pagrindinius protokolus [15], įskaitant Peer-to-Peer apsikeitimo duomenimis protokolą, greitųjų žinučių protokolus, IP telefonijos protokolus ir protokolus, naudojamus daugialypės terpės medžiagos transliavimui. Įrenginys taip pat turi integruotą paslaugos kokybės valdymo mechanizmą, kuris atpažintus duomenų srautus leidžia riboti, blokuoti arba skirstyti pagal prioritetus.

Pagrindiniai įrenginių bruožai:

- Duomenų srauto pralaidumas iki 75 gigabitų per sekundę.
- Duomenų srauto valdymas, srautą gebant granuliuoti pagal vartotojus ar aplikacijas.
- Paketų analizė OSI 7-ame lygmenyje ir duomenų srautų elgesio analizė.
- Beveik nedidina duomenų paketų vėlinimo, tad tokių paslaugų kokybė, kaip IP telefonija, nenukenčia.
- Įrenginys visiškai nematomas tinklo vartotojams.
- Apsikeitimo duomenimis naudojant BitTorrent protokolą kontrolė, vykdoma baltųjų sąrašų principu.

Detalesnius „ipoque“ įrenginių aprašus galima rasti kompanijos svetainėje [15].

Galima i6skirti du apibendrintus metodus, kuriuos kompiuterių tinklais perduodamų srautų klasifikavimui naudoja „PRX Traffic Manager“:

- Gili paketų analizė. Gili paketų analizė – tai terminas, kuriuo siekiama apibrėžti tinklu perduodamų duomenų analizę, kuomet tarpinis tinklo įrenginys įvertina ne tik paketo antraštėje esančius duomenis.. Gili paketų analizė atliekama 3-7 OSI modelio lygmenyse. Programinė įranga, ar įrenginys, taikantis gilią paketų analizę, patikrina visą IP paketą ir priima sprendimą (pavyzdžiui: atmesti paketą, ar persiųsti) remdamasis taisyklėmis paremta logika, kuri vadovaujasi parašais[7] arba reguliariomis

7 Ang.: Signatures

išraiškomis. Kitaip tariant, paketo turinys palyginamas su tam tikroje, jau egzistuojančioje, duomenų bazėje esančiais įrašais, saugančiais informaciją apie žinomus parašus. Šį metodą galima papildyti taikant statistinį aptiktų parašų vertinimą, gaunamą naudojant statistinius ir istorinius algoritmus Toks paketų analizės būdas turi vieną trūkumą: jis labai reiklus skaičiavimo resursams. Norint peržiūrėti ir įvertinti kiekvieno paketo turinį, reikalingas didelės skaičiavimo spartos įrenginys. Tai ypač taikytina atvejams, kai siekiama apdoroti didelius duomenų srautus.

- Duomenų srautų elgesio analizė. „ipoque“ internetinėje svetainėje nepateikiama informacijos apie tai, kokie algoritmai naudojami srautų elgesio bruožų nustatymui ir analizei. Ir tai natūralu, kadangi „ipoque“ yra komercinė organizacija, o sukurtas srautų valdymo sprendimas yra mokamas.

Pagal 2008 metais EANTC organizacijos atliktų „ipoque“ gaminamų įrenginių testavimų rezultatus [16] matome, kad populiariausi P2P protokolai (BitTorrent, eDonkey, Gnuttela, FastTrack) buvo atpažinti dideliu tikslumu (daugeliu atvejų virš 90%), kai tu tarpu mažiau populiarių P2P protokolų (WinMx, Mp2P, DirectConnect) aptikimo tikslumas labai įvairus ir svyruoja nuo 1% iki 96%.

„ipoque“ sprendimas be abejonės vertas dėmesio. Tačiau jis turi keletą trūkumų:

- Kompanija neatskleidžia metodų, kuriuos naudoja kompiuterių tinklais perduodamų srautų klasifikacijai, tikslių specifikacijų. Todėl iš akademinės perspektyvos, daug naudos jie – neturi.
- Produktas yra mokamas. Tad mažo ir vidutinio dydžio įmonėms, įvairioms viešojo sektoriaus organizacijoms gali kilti finansinių sunkumų įsigyjant reikiamą įrangą.
- Tai nėra atvirojo kodo produktas.

1.3.2 IPP2P OSS programinė įranga

IPP2P – tai paketų filtro *netfilter* praplėtimas, skirtas identifikuoti P2P duomenų srautus kompiuterių tinkle. Pagrindinis IPP2P tikslas yra suteikti administratoriui įrankį, kuris padėtų filtruoti tinklu perduodamus duomenis dinamiškai ir inteligentiškai. Kadangi IPP2P kaip įskiepis yra naudojamas kartu su IPTables paketų filtru, yra galimybė atpažintu P2P srautus paženklinti, riboti ar visiškai blokuoti.

Siekiant aptikti P2P duomenų srautus, IPP2P įrankis tikrina visų paketų turinį ir ieško apibrėžtų požymių, signatūrų, parodančių, jog analizuojamas paketas priklauso vienam ar kitam P2P protokolui. Todėl įrankis nėra pajėgus atpažinti, koks yra šifruoto duomenų srauto

tipas. Detalus protokolų sąrašas, kuriuos gali aptikti IPP2P įrankis pateikiamas projekto svetainėje [17].

IPP2P yra atvirojo kodo įrankis, platinamas pagal GNU GPL [18] licenziją. Produktą galima naudoti nemokamai, projekto autorius savanoriškai paremiant pinigine auka.

1.3.3 OpenDPI programinė įranga

Kompanijos „ipoque“ bandymas sukurti PACE programinės įrangos atvirojo kodo versiją. PACE (Protocol and Application Classification Engine) – tai programinė įranga, naudojama komerciniuose „ipoque“ įrenginiuose ir leidžianti atlikti kompiuterių tinklais perduodamų duomenų srautų analizę naudojant DPI ir duomenų srautų elgesio charakterizavimo technikas.

„ipoque“ atstovai [25] pripažįsta, kad OpenDPI programinė įranga (platinama pagal LGPL [19] licenziją) geba atpažinti tik nešifruotus ar kitais būdais neužmaskuotus protokolus, tokius kaip BitTorrent, eDonkey, KaZaa/Fasttrack, SSH, SSL, HTTP, DSN ir t.t. Pilnas palaikomų protokolų sąrašas pateikiamas ARS Technika svetainėje [20].

OpenDPI projektas svarbus tuo, kad tai naujas, atvirojo kodo produktas (pasaulį išvydęs 2009 metų vasarą), leidžiantis iš arti pažvelgti į kitų žmonių idėjas, būdus spręsti darbe analizuojamai problemai.

1.3.4 Snort IDS programinė įranga

SNORT – tai atvirojo kodo sistema, gebanti realiu laiku aptikti kompiuterių tinkle vykdomas atakas ir jas neutralizuoti. SNORT sistema gali realiu laiku atlikti perduodamų paketų stebėjimą ir analizę, paketų turinio analizę.

[21] publikacijos autoriai demonstruoja, jog sėkmingai gali būti panaudotas OpenNAP, WinMX, FastTrack peer-to-peer srautų aptikimui. Publikacijos autoriai pritaiko SNORT įrankį peer-to-peer srautų aptikimui naudodami paketų turinio analizės techniką.

SNORT įrankis taip pat gali būti pritaikytas peer-to-peer srautų aptikimui pagal peer-to-peer naudojamus prievadus.

SNORT nėra pajėgus aptikti šifruotus BitTorrent peer-to-peer srautus, kadangi šifruotų BitTorrent peer-to-peer srautų neįmanoma identifikuoti remiantis tik prievadų numeriais, o esant užšifruotam paketo turiniui, jo analizę netenka prasmės.

1.3.5 Alternatyvių sistemų apibendrinimas

Sistemų vertinimui, lyginimui, ir trūkumams atspindėti naudosime šiuos kriterijus:

- Sistemos prieinamumas. Nusako vieną iš dviejų atvejų: a) sistema yra atvirojo kodo ir nemokama (Taip) b) sistema nėra atvirojo kodo, ir yra mokama (Ne).
- Sistemos suderinamumas su technine įranga. Nusakoma, ar: a) sistemos veikimui užtikrinti būtina naudoti sistemos gamintojo tiekiamą techninę įrangą (Ne) b) sistemos veikimui užtikrinti nebūtina naudoti sistemos gamintojo tiekiamą techninę įrangą (Taip).
- Sistemos gebėjimas identifikuoti šifruotus BitTorrent peer-to-peer srautus. Nusakoma, ar: a) sistema geba identifikuoti šifruotus BitTorrent peer-to-peer srautus (Taip) b) sistema nemoka identifikuoti šifruotų peer-to-peer srautų (Ne).

Sistemos pavadinimas	Sistemos prieinamumas.	Sistemos suderinamumas	Atpažįsta šifruotus BitTorrent srautus
PRX Traffic Manager	Ne	Ne	Taip
IPP2P	Taip	Taip	Ne
OpenDPI	Taip	Taip	Ne
SNORT	Taip	Taip	Ne

Lentelė 4: Alternatyvių sistemų palyginimas

Iš visų alternatyvių sistemų tik PRX Traffic Manager geba atpažinti šifruotus BitTorrent peer-to-peer srautus. Tačiau ši sistema yra mokama ir ne atvirojo kodo programinė įranga. Sistemos IPP2P, OpenDPI ir SNORT prieinamos plačiajam vartotojų ratui, yra nemokamos ir atvirojo kodo sistemos, tačiau negali aptikti šifruotų BitTorrent peer-to-peer srautų.

Darbu siekiama ištirti ir parodyti metodiką, leidžiančią aptikti šifruotus BitTorrent peer-to-peer srautus. Patvirtinus ir eksperimentiniu keliu įrodžius metodikos veiksmingumą, būtų žengtas pirmas žingsnis jos integravimui į srauto klasifikavimo įrankius.

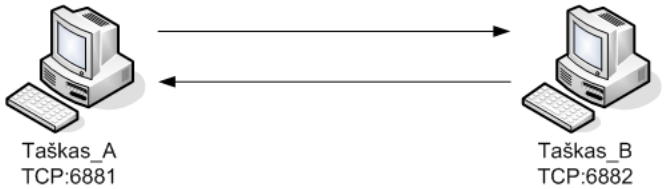
1.4 Alternatyvių metodų analizė

Kompiuterių tinklais perduodamų srautų klasifikavimas ir valdymas nuo seno yra mokslinių tyrinėjimų subjektas. Pastaraisiais metais paplitus P2P aplikacijų naudojimui ir jų valdymo problemai tapus aktualia tema, ši konkreti sritis taip pat sulaukė dėmesio. Įvairūs autoriai siūlo įvairius teorinius metodus, skirtus įvairių rūšių peer-to-peer srautų aptikimui, kuriuos galiausiai juos galima suskirstyti į tris grupes:

1) Peer-to-peer srautų aptikimas naudojant prievadų analizę.
2) Peer-to-peer srautų aptikimas atliekant aplikacijos lygmens analizę
3) Peer-to-Peer srautų aptikimas atliekant transporto lygmens analizę.

1.4.1 Peer-to-peer srautų aptikimas naudojant prievadų analizę

Tikriausiai vienas iš pirmųjų ir seniausių metodų, naudojamų kompiuterių tinklais perduodamų srautų klasifikacijai. Metodas paremtas paprasta koncepcija, pagal kurią kiekviena P2P aplikacija turi standartinį prievadą ar prievadų grupę, kurių pagalba klausosi naujų susijungimų iš kitų taškų. Todėl tinklo srautų klasifikavimo ir/ar filtravimo programinei įrangai tereikia stebėti tinklo duomenų srautą ir ieškoti atitikimų tarp žinomų prievadų, kuriuos naudoja P2P aplikacijos, ir prievadų, naudojamų duomenims perduoti stebimame kompiuterių tinkle. Prievadų analizės metodo pritaikymo pavyzdžiai pateikiami [22] ir [23].



Pav. 5: Principinė dviejų BitTorrent taškų susijungimo schema. Taškas_A įeinantiems susijungimams priimti naudoja TCP 6881 prievadą, Taškas_B - 6882

Paveikslėlyje nr. 5 pateikiamas supaprastintas dviejų BitTorrent taškų susijungimo variantas, kuomet naudojami standartiniai prievadai (prievadų sąrašas skyriuje 1.2.3 *„Susijungimai tarp taškų“*, lentelė nr. 3) įeinantiems prisijungimams priimti. Šiuo atveju BitTorrent susijungimo tarp dviejų taškų atpažinimui pakanka pirmojo sesijos paketo analizės, įvertinant transporto lygmens informaciją, t.y. gavėjo arba siuntėjo prievado numerį.

Toks peer-to-peer srautų aptikimo metodas labai lengvai pritaikomas, tačiau jį apeiti taip pat nesudėtinga. Dauguma šiuolaikinės programinės įrangos (pavyzdžiui BitTorrent klientas Vuze), skirtos duomenų mainams peer-to-peer tinkluose turi funkciją, kuri vartotojui leidžia pačiam nurodyti norimą naudoti prievadą naujiems susijungimams, arba jį pasirinkti atsitiktiniu būdu.

Kai kurie peer-to-peer klientai taip pat gali panaudoti HTTP protokolą peer-to-peer srautų

užmaskavimui, panaudojant TCP 80 prievadą [24], papildomai apsunkindami kompiuterių tinklu perduodamų srautų klasifikavimą.

Dėl išvardintų priežasčių, kompiuterių tinklais perduodamų srautų klasifikavimas remiantis tik prievadais yra bergždžias reikalas, mat dideli kiekiai peer-to-peer duomenų srautų liktų neatpažinti ir nefiltruoti.

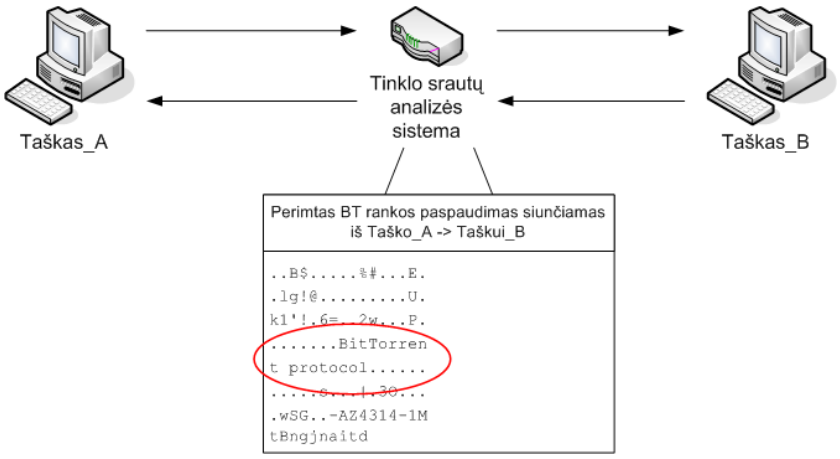
Kita vertus, informacija apie prievadus gali būti panaudota standartinių aplikacijų (HTTP, SMTP, DNS, FTP, SSH ir t.t.) duomenų srautams klasifikuoti.

1.4.2 Peer-to-peer srautų aptikimas atliekant aplikacijos lygmens analizę

Naudojant aplikacijos lygmens metodą, kompiuterių tinklais perduodamų srautų klasifikatorius tikrina siunčiamų paketų turinį ir ieško parašų, atitinkančių parašus susietus su konkrečiu protokolu ar aplikacija, saugomus paruoštoje duomenų bazėje. Aplikacijos lygmens metodas reikalauja vartotojo duomenų, perduodamų pakete, analizės. Aplikacijos parašas gali būti saugomas viename arba išbarstytas per keletą persiunčiamų paketų. Pastaruoju atveju, norint aptikti aplikacijos parašą, reikėtų surinkinėti visus sesijos duomenis, tačiau tai reikalautų didelių skaičiavimo pajėgumų, lyginant su parašo aptikimu viename pakete. Pagrindiniai metodo privalumai yra:

- kompiuterių tinklu perduodamų duomenų srautų filtras gali atpažinti, kokia aplikacija generuoja tam tikrą srautą. Tokiu būdu administratorius gali lanksčiau filtruoti kompiuterių tinklu perduodamus srautus.
- Srautai klasifikuojami dideliu tikslumu.

Paveikslėlyje nr. 6 pateikiama principinė schema, vaizduojanti BitTorrent srautų atpažinimą naudojant aplikacijos lygmens analizės metodą.



Pav. 6: Principinė BitTorrent peer-to-peer srauto atpažinimo schema naudojant aplikacijos lygmens analizę

Sistema analizuoja persiunčiamų paketų turinį, ieškodama BitTorrent protokolo identifikatoriaus (žiūrėti skyrelį 1.2.3 *„Susijungimai tarp taškų“*). Radus ieškomą identifikatorių, sesija gali būti klasifikuojama kaip peer-to-peer. Jungiantis dviems BitTorrent taškams, rankos paspaudimo metu perduodamas unikalus taško ID, kurį generuojant pridedama ir kliento programinę įrangą aprašanti informacija [7], Paveikslėlyje nr. 6 pateikiamo paketo turinyje esantys simboliai „-AZ4314-“ leidžia nustatyti, kad Taškas_A naudoja Vuze programinės įrangos 4.3.1.4 versiją.

Tačiau aplikacijos lygmens metodas:

- reikalauja daugiau skaičiavimo pajėgumų ir įrenginio, kuriame veikia filtravimo programinė įranga, lyginant su prievadų analizės metodu.
- Gali padidinti paketų vėlinimą (tai vėlgi priklauso nuo techninės įrangos pajėgumų).
- Kelia abejonių dėl perduodamų duomenų privatumo išsaugojimo, mat tinklo paketų, kuriais perduodami vartotojų duomenys, turinį dažniausiai analizuoja trečios šalies programinė įranga.
- Naudoja iš anksto paruoštą aplikacijos parašų duomenų bazę, kuri nuolatos turi būti atnaujinama, mat aplikacijos nuolat tobulinamos, keičiasi. Taip pat paruošti ir atnaujinti duomenų bazę pakankamai keblu, nes daugelis peer-to-peer aplikacijų neturi išsamios dokumentacijos, pagal kurią būtų galima aprašyti jų parašus.
- Nesuteikia galimybės atpažinti užšifruotų peer-to-peer duomenų srautų.

1.4.3 Peer-to-Peer srautų aptikimas atliekant transporto lygmens analizę

Tai euristinis metodas, paremtas įvertinimu įvairių peer-to-peer srautams būdingų savybių, atsispindinčių transporto lygmeny [16]. Jis nereikalauja paketų turinio analizės, o labiau yra koncentruotas į susijungimų tarp taškų elgesio bruožus. Peer-to-peer aplikacijos gali šifruoti perduodamus duomenis, naudoti atsitiktinius prievadus, tačiau susijungimų tarp taškų bruožai transporto lygmeny išlieka tokie patys [22].

[22] ir [23] publikacijų autoriai vykdė eksperimentus bandydami įrodyti metodo veiksmingumą. Publikacijose metodas buvo testuojamas naudojant qtorrent, valknut ir aMule aplikacijas, atitinkamai naudojančias šiuos peer-to-peer protokolus: BitTorrent, Direct Connect, eDonkey. [23] pateikiami rezultatai yra įspūdingai teigiami ir daug žadantys - teigiama, kad peer-to-peer srautus pavyko identifikuoti 99% tikslumu. Tačiau pasirinktas ir

ištestuotas labai mažas BitTorrent protokolą atstovaujančių aplikacijų kiekis (1), todėl tai kelia abejonių dėl rezultatų pagrįstumo.

Transporto lygmens metodo trūkumai:

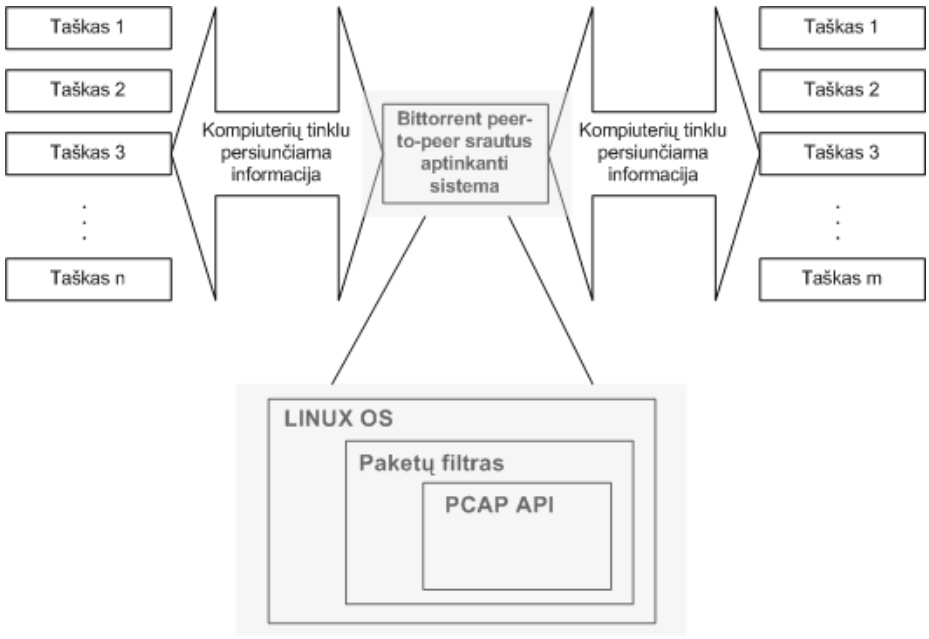
- Kai kurios ne peer-to-peer aplikacijos (pvz DNS, NFS, internetiniai žaidimai) turi panašius susijungimų bruožus, kaip ir peer-to-peer aplikacijos. Tai mažina metodo patikimumą.
- Metodo taikymas reikalauja papildomos informacijos apie egzistuojančius susijungimus kaupimo, tad reikia daugiau resursų, lyginant su prievadų analizės metodu.
- Kadangi tai euristinis metodas, gan sudėtinga užtikrinti aukštą teisingo peer-to-peer srautų aptikimo santykį, todėl metodas turi būti kruopščiai ištestuotas kiekvienam peer-to-peer protokolui.

1.5. BitTorrent peer-to-peer srautų aptikimo sistema

Darbu siekiama patvirtinti BitTorrent peer-to-peer srautų požymius, transporto ir aplikacijų lygmens metodų taikomumą BitTorrent peer-to-peer srautams identifikuoti; sudaryti ir kruopščiai ištestuoti techniką, leidžiančią aptikti šifruotus BitTorrent peer-to-peer srautus pasinaudojant transporto ir aplikacijų lygmens metodais.

Darbu numatoma sukurti koncepcinę sistemą, kuri praktiškai įgyvendins ir leis ištestuoti sudarytą BitTorrent peer-to-peer srautų aptikimo techniką. Abstrakti sistemos diagrama pateikiama 7 pav.

Darbu nesiekiama sukurti komercinio lygio sistemos, leidžiančios realiu laiku aptikti ir klasifikuoti peer-to-peer srautus. Tai daugiau koncepcinio modelio aprašymas ir pagrindimas, sudedantis pamatus praktiškai pritaikomo įrankio vystymui.



Pav. 7: Abstrakti BitTorrent peer-to-peer srautų aptikimo sistemos schema

Pagrindiniai sistemos, skirtos šifruotų BitTorrent peer-to-peer srautų aptikimui, komponentai:

- Linux Operacinė Sistema. Linux operacinė sistema pasirinkta dėl aibės nemokamų įrankių kompiuterių tinklais perduodamų duomenų srautams tyrinėti, bibliotekų reikalingų srauto analizei atlikti, kodo kompiliatorių.
- BT P2P aptikimo programa. Tai programa, atliksianti tinklu perduodamų paketų perėmimą, informacijos apie sesijas, kuriems perimti paketai priklauso, kaupimą ir analizę, įvertinimą, ar sesija ir BitTorrent peer-to-peer ar ne.
- *libpcap* biblioteka [26]. Tai biblioteka su API, skirta kompiuterių tinklais perduodamų duomenų srautų perėmimui.

Sistemos modelis detaliau nagrinėjamas skyriuje 2. *„Sistemos reikalavimų specifikacija ir analizė“*.

Numatomos sistemos funkcijos:

- Aptikti šifruotus BitTorrent peer-to-peer srautus analizuojant failą, kuriame saugoma duomenų srautų kopija, padaryta naudojant kompiuterių tinklais perduodamų srautų stebėjimo įrankį.
- Aptikti šifruotus BitTorrent peer-to-peer srautus analizuojant realiu laiku kompiuterių tinklu perduodamus duomenis.

1.6 Išvados

Dalis šiuo metu egzistuojančių įrankių gali aptikti šifruotus BitTorrent srautus. Tačiau įrankiai, gebantys aptikti šifruotus BitTorrent srautus yra mokami, todėl sunkiau prieinami mažų įmonių, organizacijų, ar namų tinklų vartotojams. Nemokamų ir/ar atvirojo kodo įrankių, leidžiančių aptikti šifruotus BitTorrent srautus rasta nebuvo.

Aptartos BitTorrent protokolo savybės, ir kitų autorių darbų apžvalga leidžia manyti, kad sudaryti metodą, skirtą šifruotų BitTorrent srautų aptikimui – įmanoma.

Sudaryta abstrakti būsimos sistemos schema. Tolesniuose darbo etapuose pateikiami reikalavimai sistemai, sistema realizuojama pasirinkta programavimo kalba, atliekamas sistemos testavimas ir eksperimentinis tyrimas.

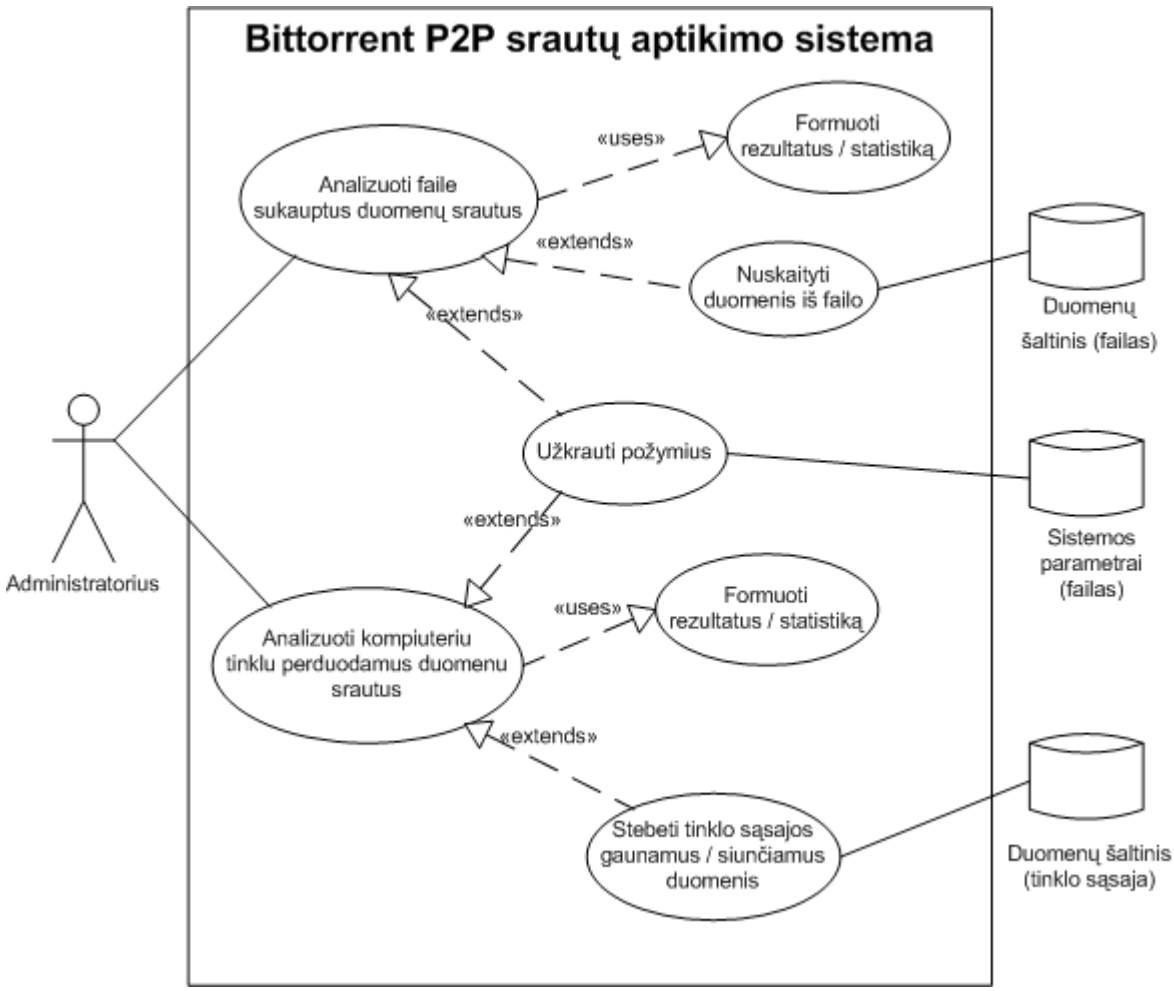
2. Sistemos reikalavimų specifikacija ir analizė

Šiame skyriuje aptariami sistemos reikalavimai, atliekama jų analizė. Sistemos reikalavimų analizei ir projektavimui atlikti pasirinktas objektinis modelis, grindžiamas UML. Proceso automatizavimui pasirinktas Microsoft Office Visio 2003 įrankis.

2.1 Reikalavimų modelis

2.1.1 Sistemos panaudojimo atvejų modelis

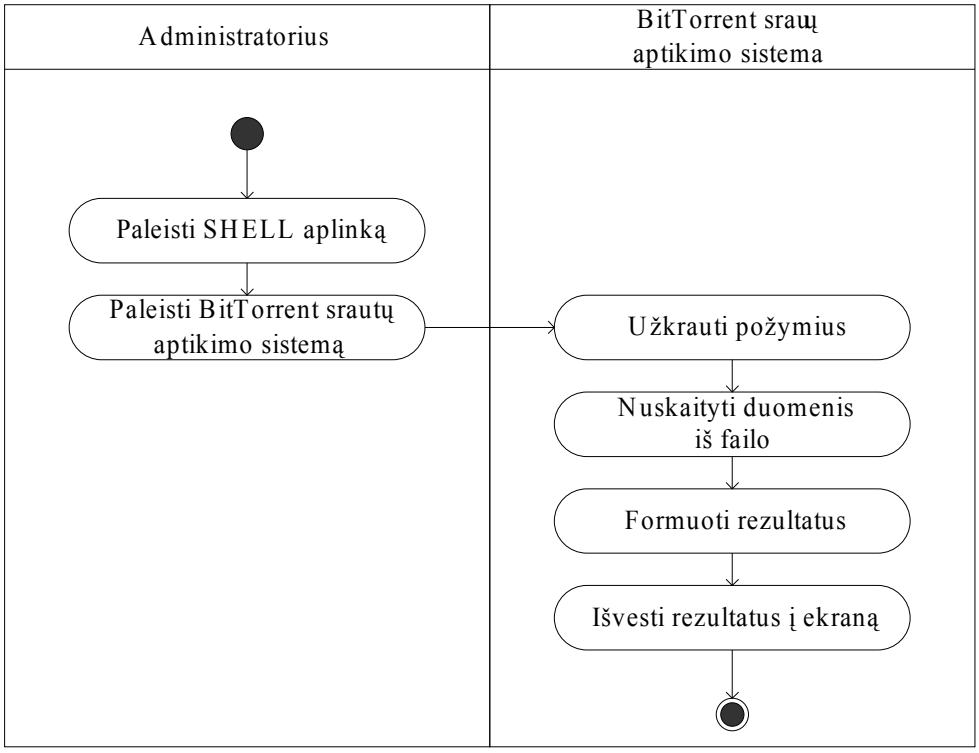
Sistemos panaudojimo atvejų diagrama pateikta sekančiame paveiksle. Joje pavaizduoti sistemos aktoriai ir jų atliekamos funkcijos.



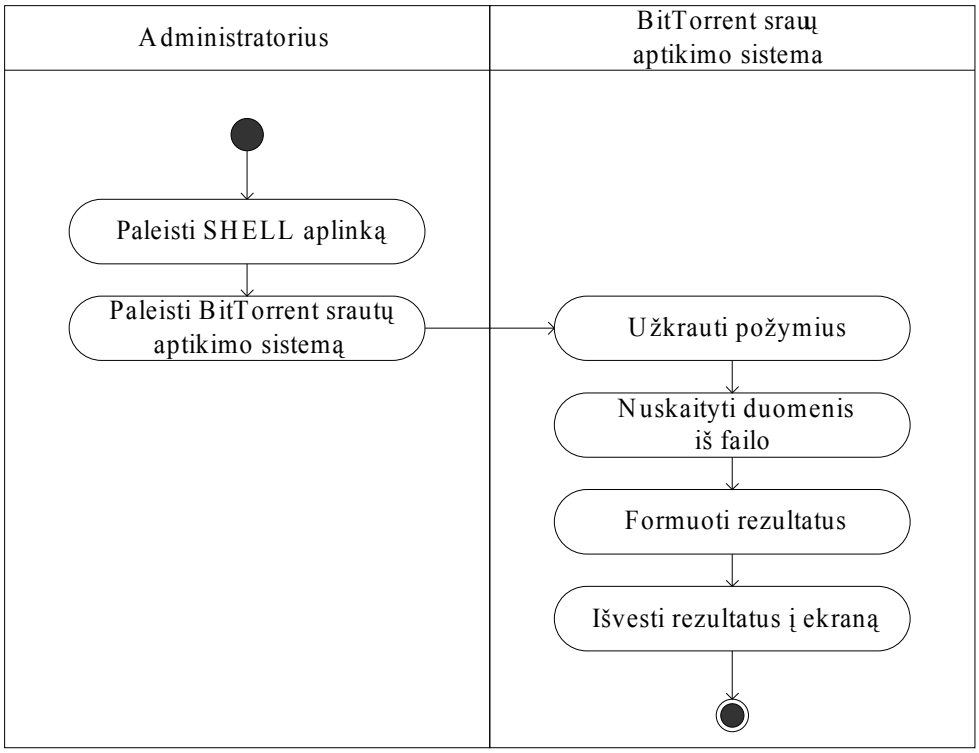
Pav. 8: BitTorrent P2P srautų aptikimo sistemos panaudojimo atvejų modelio diagrama

- Administratorius – sistemos administratorius, turintis privilegijas perimti duomenų srautus, perduodamus per tinklo sąsają. Administratorius paleidžia BitTorrent peer-to-peer srautų aptikimo sistemą, nurodo jos darbo režimą, įvertina gautus rezultatus.
- Duomenų šaltinis (failas) – failas, kuriame saugomi praeityje sukaupti ir jau nekintantys kompiuterių tinklais perduodamų duomenų srautai.
- Sistemos parametrai (failas) – sistemos parametrų (protokolų ir aplikacijų parašai, charakteristikos, įvairūs sistemos darbo nustatymai, detaliau analizuojami 3.1 *„Sistemos veikimo aprašymas“* skyriuje) saugykla.
- Duomenų šaltinis (tinklo sąsaja) – tinklo sąsaja, kuria perduodamus duomenų srautus tinklo administratorius nori stebėti, siekdamas identifikuoti BitTorrent peer-to-peer srautus. Stebima tinklo sąsaja duomenų srautai perduodami realiu laiku, srautų analizė ir rezultatų išvedimas taip pat vyksta realiu laiku.

Panaudojimo atvejus aprašančios veiklos diagramos pateikiamos paveiksliukuose nr. 9-10.



Pav. 9: Panaudojimo atvejų „Analizuoti faile sukauptus duomenų srautus“ veiklos diagrama



Pav. 10: Panaudojimo atvejų „Analizuoti kompiuterių tinklu perduodamus duomenų srautus“ veiklos diagrama

2.1.2 Panaudojimo atvejų specifikacijos

Šiame skyriuje pateikiamos panaudojimo atveju specifikacijos. Panaudojimo atvejai sudaryti remiantis panaudojimo atvejų diagramomis (žiūrėti skyrelį 2.1.1).

1. PANAUDOJIMO ATVEJIS: Analizuoti faile sukauptus duomenų srautus

Vartotojas / Aktorius: Administratorius
Aprašas: kompiuterių tinklais perduodamų srautų analizė, juos nuskaitant iš statinio duomenų failo. Srautai duomenų faile sukaupti praeityje ir nekintantys. Duomenų failo formatas: PCAP.
Prieš sąlyga: Egzistuoja duomenų failas.
Sužadinimo sąlyga: Pageidaujama atlikti duomenų failo analizę.
Po sąlyga: Baigta tinklo paketų, saugomų nurodytame duomenų faile, analizė.

2. PANAUDOJIMO ATVEJIS: Analizuoti kompiuterių tinklu perduodamus duomenų srautus

Vartotojas / Aktorius: Administratorius
Aprašas: kompiuterių tinklais perduodamų srautų analizė, juos nuskaitant iš einamuoju laiko momentu nuskaitant nuo nurodytos tinklo sąsajos.
Prieš sąlyga: Egzistuoja nurodyta tinklo sąsaja.
Sužadinimo sąlyga: Pageidaujama atlikti kompiuterių tinklais perduodamų srautų stebėjimą ir analizę.
Po sąlyga: Administratorius sustabdo sistemos veikimą.

2.1.3 Funkciniai sistemos reikalavimai

Šiame skyriuje pateikiami funkciniai BitTorrent srautų aptikimo sistemos reikalavimai.

Reikalavimas: 1	**Reikalavimo tipas:** F	**Įvykio/panaudojimo atvejis**: 1	
Aprašymas:	Sistema turi pranešti, jeigu nurodytas duomenų failas neegzistuoja.		
Pagrindimas:	Užtikrina sistemos veikimą.		
Šaltinis:	Administratorius.		
Tikimo kriterijus:	Sistema įspėja vartotoją, jeigu nurodytas duomenų failas neegzistuoja.		
Užsakovo tenkinimas:	5	**Užsakovo netenkinimas:** 0	
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2010-03-11		

Reikalavimas:	2	**Reikalavimo tipas:** F	**Įvykio/panaudojimo atvejis**: 1
Aprašymas:	Sistema turi pranešti, kai duomenų faile esančios informacijos analizė baigta.		
Pagrindimas:	Palengvina vartotojo darbą.		
Šaltinis:	Administratorius.		
Tikimo kriterijus:	Sistema įspėja vartotoją, kai duomenų faile esančios informacijos analizė baigta.		
Užsakovo tenkinimas:	5	**Užsakovo netenkinimas:** 0	
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2010-03-11		

Reikalavimas:	3	**Reikalavimo tipas:** F	**Įvykio/panaudojimo atvejis**: 1
Aprašymas:	Sistema turi pranešti, jeigu sistemos parametrų failas neegzistuoja.		
Pagrindimas:	Užtikrina sistemos veikimą.		
Šaltinis:	Administratorius.		
Tikimo kriterijus:	Sistema įspėja vartotoją, jeigu sistemos parametrų failas neegzistuoja arba jo nepavyksta nuskaityti.		
Užsakovo tenkinimas:	5	**Užsakovo netenkinimas:** 0	
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2010-03-11		

Reikalavimas:	4	**Reikalavimo tipas:** F	**Įvykio/panaudojimo atvejis**: 2
Aprašymas:	Sistema turi pranešti, jeigu sistemos nurodyta tinklo sąsaja neegzistuoja.		
Pagrindimas:	Užtikrina sistemos veikimą.		

Šaltinis:	Administratorius.		
Tikimo kriterijus:	Sistema įspėja vartotoją, jeigu nurodyta tinklo sąsaja neegzistuoja.		
Užsakovo tenkinimas:	5	**Užsakovo netenkinimas:**	0
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2010-03-11		

Reikalavimas:	5	**Reikalavimo tipas:**	F	**Įvykio/panaudojimo atvejis**:	2
Aprašymas:	Sistema baigia darbą tik vartotojui davus atitinkamą komandą.				
Pagrindimas:	Užtikrina sistemos veikimą.				
Šaltinis:	Administratorius.				
Tikimo kriterijus:	Vartotojui nusiuntus nuspaudus CTRL+C aktyvioje SHELL aplinkoje, iš kurios paleista BitTorrent srautų aptikimo sistema, sistema baigia darbą.				
Užsakovo tenkinimas:	5	**Užsakovo netenkinimas:**	0		
Priklausomybės:	Nėra	**Konfliktai:**	Nėra		
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2010-03-11				

Reikalavimas:	6	**Reikalavimo tipas:**	F	**Įvykio/panaudojimo atvejis**:	2
Aprašymas:	Baigusi darbą, sistema išveda rezultatus į ekraną,				
Pagrindimas:	Palengvina darbą vartotojui.				
Šaltinis:	Administratorius.				
Tikimo kriterijus:	Sistemai baigus darbą, vartotojui į ekraną išvedami darbo rezultatai. Rezultatai pateikiami tekstiniu pavidalu.				
Užsakovo tenkinimas:	5	**Užsakovo netenkinimas:**	0		

Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2010-03-11		

Reikalavimas: 3	**Reikalavimo tipas:** F	**Įvykio/panaudojimo atvejis**: 2	
Aprašymas:	Sistema turi pranešti, jeigu sistemos parametrų failas neegzistuoja.		
Pagrindimas:	Užtikrina sistemos veikimą.		
Šaltinis:	Administratorius.		
Tikimo kriterijus:	Sistema įspėja vartotoją, jeigu sistemos parametrų failas neegzistuoja arba jo nepavyksta nuskaityti.		
Užsakovo tenkinimas:	5	**Užsakovo netenkinimas:** 0	
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2010-03-11		

2.1.4 Sistemos nefunkciniai reikalavimai

Skyriuje pateikiami nefunkciniai reikalavimai kuriamai BitTorrent peer-to-peer srautų aptikimo sistemai.

2.1.4.1 Reikalavimai sistemos išvaizdai

- Sistemos naudos CLI (Command Line Interface).
- Sistemos paleidimas vykdomas per komandinę eilutę iš *shell* aplinkos. *Shell* aplinkos kintamieji atsakingi už teksto atvaizdavimo spalvas ir šriftus nustatomi paties vartotojo, įtakos sistemos veikimui jie neturi.
- Sistemos paleidžiamojo failo pavadinimas turi būti trumpas (iki 10 simbolių) ir atspindintis sistemos paskirtį.
- Sistemos veikimo rezultatų išvedimui naudojama standartinė išvestis [27].

2.1.4.2 Reikalavimai sistemos panaudojamumui

Sistema orientuota į galutinį vartotoją, kuris turi tvirtas žinias IT srityje. Vartotojui, norinčiam

suprasti sistemos tikslą, jos veikimo principą ir gebėti interpretuoti sistemos pateikiamus rezultatus įdėmus pateikiamos dokumentacijos skaitymas reikalaujamas. Papildomos žinios šiose srityse rekomenduojamos:

- Kompiuterių tinklai, bendros žinios.
- Linux Operacinės Sistemos pradmenys.
- Naudojimosi *shell* aplinka pradmenys.

Sistemos atliekamas argumentų, perduodamų per komandinę eilutę, tikrinimas ribotas. Todėl vartotojas privalo būti tikras įvedamų duomenų teisingumu. Neteisingų duomenų įvedimo atveju sistema neveiks, veiks nestabiliai arba pateiks neteisingus rezultatus.

Vartotojas, neturintis reikiamų žinių išvardintose srityse, nebus pajėgus interpretuoti sistemos pateikiamų rezultatų.

Kartu su sistema pateikiamų ir BitTorrent peer-to-peer protokolo aptikimui naudojamų požymių sąrašas saugomas sistemos parametrų faile. Bet koks požymių sąrašo redagavimas gali sutrikdyti sistemos darbą, todėl yra nerekomenduojamas.

Tinklo įrenginį, naudojamą kompiuterių tinklo perduodamų srautų stebėjimui atlikti, sistema pagal nutylėjimą perveda į PROMISCUOUS būseną. Kai kuriuose kompiuterių tinkluose taisyklės draudžia jungti prie tinklo įrenginius su įjungta PROMISCUOUS būsena. Prie sistemos naudojimą vartotojui rekomenduojama susipažinti su kompiuterių tinklo, prie kurio prijungta BitTorrent peer-to-peer srautų aptikimo sistema, taisyklėmis. Pagal poreikį PROMISCUOUS režimą galima išjungti arba įjungti.

2.1.4.3 Reikalavimai vykdymo charakteristikoms

- Srautų apdorojimo greitis - įgyvendinti technologinius sprendimus, kurie užtikrintų pakankamą duomenų apdorojimo greitį, įvertinant naudojamos techninės įrangos pajėgumus ir operacinės sistemos sistemos apkrovą.
- Duomenų saugumas – nekaupti stebimais kompiuterių tinklais perduodamiems srautams priklausančių paketų turinio. Tokiu būdu išvengiama potenciali situacija, kuomet pažeidžiamas kompiuterių tinklo vartotojų privatumas.
- Užtikrinti sistemos veikimo stabilumą su kūrėjo pateikiamais parametrais.

2.1.4.4 Reikalavimai veikimo sąlygoms

- Veikianti tinklo sąsaja kompiuterių tinklais perduodamų srautų stebėjimui.

- Serveris, monitorius, klaviatūra.
- Priėjimas prie aktyvaus kompiuterių tinklo.
- ArchLinux operacinė sistema su 2.6.32 branduolio versija [28]
- Libpcap biblioteka, 1.0.0 versija [26]
- jNetPcap biblioteka, 1.2 versija. [29]
- Java(TM) SE Runtime Environment 1.6
- Standartinių Linux OS bibliotekų rinkinys, korektiškam programos veikimui.

2.1.4.5 Reikalavimai sistemos priežiūrai

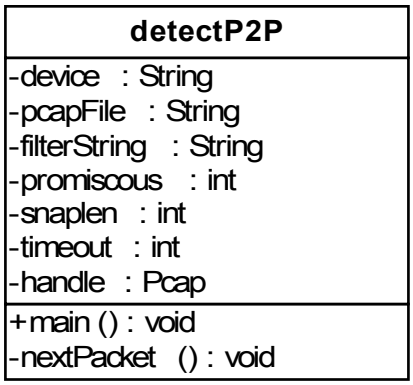
Kadangi kuriama sistema yra koncepcinio pobūdžio sistema, kurios tikslas yra padėti ištestuoti sudarytą šifruotų BitTorrent peer-to-peer srautų aptikimo techniką, reikalavimai sistemos priežiūrai nepateikiami.

2.4.5 Reikalavimai duomenims

Informacija apie tinklo paketus duomenų faile turi būti saugoma PCAP formatu.
Atskirų reikalavimų duomenims, nuskaitomiems tiesiai nuo tinklo sąsajos nėra. Kadangi duomenų nuskaitymui naudojama LIBPCAP biblioteka, sistemos gaunami duomenys jau atitinka PCAP formatą.

2.4 BitTorrent srautų aptikimo sistemos klasių aprašai

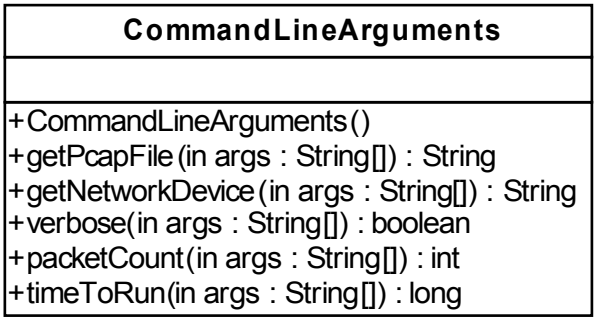
Sistemos realizacijai naudojama JAVA programavimo kalba. JAVA – tai objektinė, paremta klasėmis programavimo kalba. Šiame skyriuje pateikiamos pagrindinės projektuojamos sistemos klasės ir jų aprašai.



Pav. 11: Klasės "DetectP2P" UML diagrama

Pagrindinė sistemos klasė. Klasė *detectP2P* atsakinga už programos inicializaciją: valdo tinklo įrenginio paruošimą nuskaitymui,perima paketus ir perduoda juos tolimesniam apdorojimui. Toliau pateikiamas detalus klasės kintamųjų ir metodų aprašas.

device – kintamasis naudojamas tinklo sąsajos vardui saugoti. Sąsajos vardas programai perduodamas per komandinės eilutės parametrus. Nenurodžius įrenginio vardo, programa pagal nutylėjimą naudos tinklo sąsają pavadinimu „*eth1*“.
pcapFile – kintamasis naudojamas duomenų failo vardui saugoti. Nurodžius neegzistuojantį failą, arba failo visiškai nenurodžius, programa pagal nutylėjimą bandys skaityti duomenis iš tinklo įrenginio, kurio pavadinimą saugo *device* kintamasis.
filterString – kintamasis naudojamas saugoti PCAP filtrui. Mums reikalingi tik TCP ir UDP srautai,todėl filtrui pagal nutylėjimą nurodomas „*TCP and UDP*“ parametras.
promiscous – kintamasis saugo būsenos, į kurią bus pervestas tinklo įrenginys, pavadinimą. Pagal nutylėjimą tinklo įrenginys pervedamas į būseną pavadinimu „*Pcap.MODE_PROMISCUOUS*“.
snaplen – kintamasis apibrėžia maksimalų dydį paketo, kurį gali priimti ir apdoroti programa. Perimti tinklo duomenų paketai, kurių dydis didesnis už *snaplen* apkarpomi. Pagal nutylėjimą priimami tinklo duomenų paketai, kurių dydis ne didesnis kaip 1500 baitų.
nextPacket – metodas atsakingas už naujai atvykusių paketų apdorojimą. Iškviečiamas kiekvieną kartą atvykus naujam tinklo duomenų paketui.



Pav. 12: Klasės "CommandLineArguments" UML diagrama

Klasė *CommandLineArguments* skirta komandine eilute perduodamiems parametrams apdoroti. Viso programa priima 5 argumentus:

- Failo vardas
- Tinklo įrenginio vardas

- Informatyvumo parametras
- Perimamų paketų kiekis
- Programos veikimo laikas

Visi argumentai, jų paskirtis ir naudojimas plačiau aptariame skyriuje 3.1 *„Sistemos veikimo aprašymas“*.

Žemiau pateikiamas detalus klasės metodų aprašas.

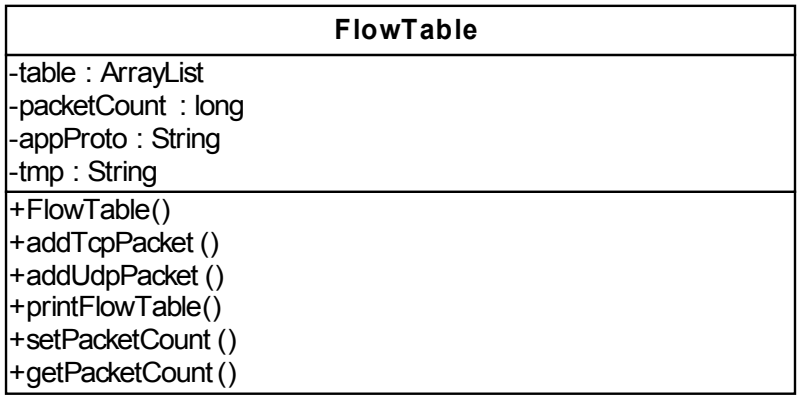
getPcapFile – patikrina ar per komandinės eilutės argumentus programai perduodamas duomenų failo vardas, ar failas egzistuoja. Tenkinant abi sąlygas, gražina duomenų failo vardą.

getNetworkDevice - patikrina ar per komandinės eilutės argumentus programai perduodamas tinklo įrenginio. Tenkinant sąlygą, gražina tinklo įrenginio vardą. Priešingu atveju gražina pagal nutylėjimą naudojamo tinklo įrenginio vardą.

verbose – patikrina ar komandinės eilutės argumentų pagalba programai perduodamas argumentas, nurodantis, ar vartotojas nori gauti sesijų lentelės turinį ar ne. Teigiamu atveju sesijų lentelės turinys išvedamas į ekraną tekstiniu pavidalu, programai pabaigus darbą.

packetCount – patikrina ar per komandinės eilutės argumentus programai perduodamas paketų skaičius, kuriuos perėmusi programa turi baigti darbą.

timeToRun - patikrina ar per komandinės eilutės argumentus programai perduodamas laikas, kuriam praėjus programa turi baigti darbą.



Pav. 13: Klasės "FlowTable" UML diagrama

FlowTable klasė naudojama duomenų apie perimtas sesijas saugojimui.

table – sąrašas, kuriame saugomas aptiktų UDP ir TCP sesijų sąrašas.

packetCount – nuo programos startavimo pradžios priimtų tinklo duomenų paketų skaičius.

appProto – einamosios sesijos protokolo aprašą saugantis kintamasis.

tmp – kintamasis, skirtas saugoti laikiniems duomenims.

FlowTable – klasės konstruktorius.

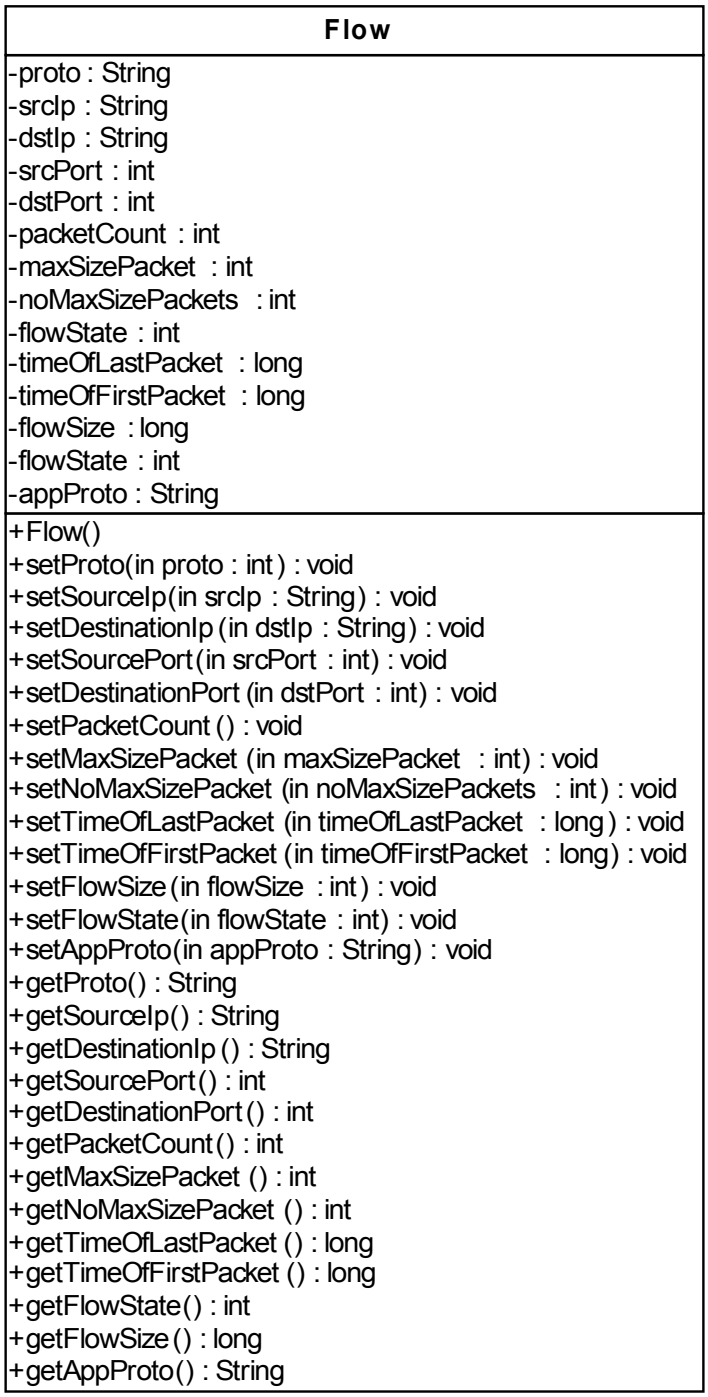
addTcpPacket – atvykus naujam TCP paketui, metodas atpažįsta sesiją, kuriai priklauso paketas, ir , jeigu sesijos duomenys jau saugomi sąraše, atnaujina jos duomenis. Jeigu sesiją identifikuojančių duomenų sąraše nėra, sukuriamas naujas sesijos įrašas.

addUdpPacket – atvykus naujam UDP paketui, metodas atpažįsta sesiją, kuriai priklauso paketas, ir , jeigu sesijos duomenys jau saugomi sąraše, atnaujina jos duomenis. Jeigu sesiją identifikuojančių duomenų sąraše nėra, sukuriamas naujas sesijos įrašas.

printFlowTable – metodas, į standartinę išvesti atspausdinantis sesijų lentelės turinį.

setPacketCount – metodas vykdomas atvykus naujam tinklo duomenų paketui. Padidina gautų tinklo duomenų paketų skaičių vienetu.

getPacketCount – gražina gautų tinklo duomenų paketų skaičių.



Pav. 14: Klasės "Flow" UML diagrama

Flow klasė naudojama individualių sesijų duomenims saugoti ir tvarkyti. Kiekvienai naujai aptiktai UDP arba TCP sesijai sukuriamas naujas Flow klasės tipo objektas ir talpinamas į sesijų sąrašą. Žemiau pateikiamas detalus klasės kintamųjų ir metodų aprašas.

proto – sesijos protokolas. TCP arba UDP.

srcIp – IP adresas tinklo taško, kuris inicijavo sesiją.

dstIp – IP adresas tinklo taško, kuris užmezgė srcIp taško inicijuotą sesiją.

srcPort – tinklo prievadas, naudojamas sesijai užmegzti srcIp taške.

dstPort - tinklo prievadas, naudojamas sesijai priimti ir palaikyti dstIp taške.

packetCount – tinklo duomenų paketų, priklausančių konkrečiai sesijai, skaičius.

maxSizePacket – maksimalus aptiktas paketo dydis sesijoje.

noMaxSizePackets – sesijoje esančių paketų kiekis, kurių dydis lygus *maxSizePacket*.

flowState – sesijos būsena. Sesija gali būti neaktyvi (jeigu 30 minučių negauna naujų paketų) arba aktyvi.

timeOfLastPacket – paskutinio sesijai priklausančio tinklo duomenų paketo priėmimo laikas išreikštas milisekundėmis.

timeOfFirstPacket - pirmojo sesijai priklausančio tinklo duomenų paketo priėmimo laikas išreikštas milisekundėmis.

flowSize – sesijos dydis baitais.

flowState – sesijos būsena. Saugoma reikšmė nusako, ar sesija yra aktyvi ar ne.

appProto – sesijos protokolo tipas.

Flow – Klases konstruktorius

setProto – nustato sesijos protokolą į pageidautiną reikšmę.

setSourceIp – nustato sesijos šaltinio IP adresą į norimą reikšmę

setDestinationIp – nustato antrojo sesijos dalyvio IP adresą į norimą reikšmę.

setSourcePort – nustato sesijos šaltinio prievadą į norimą reikšmę.

setDestinationPort – nustato sesijos antro dalyvio prievadą į norimą reikšmę.

setPacketCount – padidina sesijos paketų skaičių vienetu.

setMaxSizePacket – nustato maksimalų aptiktą duomenų paketo dydį į norimą reikšmę.

setNoMaxSizePacket – nustato paketų su maksimaliu dydžiu skaičių į norimą reikšmę.

setTimeOfLastPacket – nustato sesijos paskutinio gauto duomenų paketo gavimo laiką į norimą reikšmę.

setTimeOfFirstPacket - nustato sesijos pirmo gauto duomenų paketo gavimo laiką į norimą reikšmę.

setFlowSize – nustato sesijos metu perduotų duomenų dydį į norimą reikšmę.

setFlowState – nustato sesijos būseną į norimą reikšmę.

setAppProto – nustato sesijos protokolo tipą į norimą reikšmę.

getProto – gražiną sesijos protokolo reikšmę.

getSourceIp – gražina sesijos šaltinio IP adresą.

getDestinationIp – gražina antrojo sesijos dalyvio IP adresą.

getSourcePort – gražina sesijos šaltinio prievadą.

getDestinationPort – gražina antrojo sesijos dalyvio naudojamo prievado numerį.

getPacketCount – gražina esamą paketų, priklausančių tam tikrai sesijai, kiekį.

getMaxSizePacket – gražina maksimalų aptiktą sesijai priklausančio paketo dydį

getNoMaxSizePacket – gražina skaičių paketų, kurių dydis lygus maksimaliam sesijos paketo dydžiui.

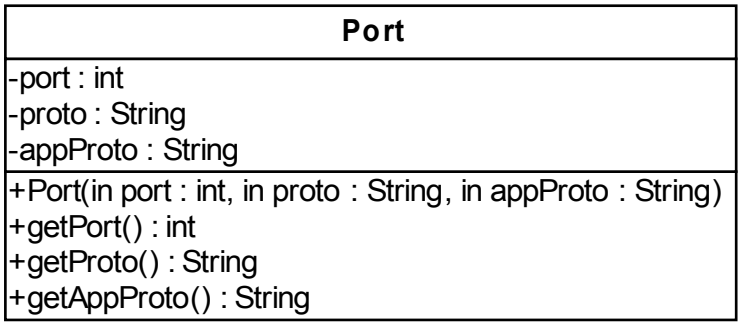
getTimeOfLastPacket – gražina paskutinio gauto sesijai priklausančio paketo gavimo laiką.

getTimeOfFirstPacket – gražina pirmojo gauto sesijai priklausančio paketo gavimo laiką.

getFlowSize – gražina sesijos metu perduotų duomenų kiekį.

getFlowState – gražina sesijos būseną.

getAppProto – gražina sesijos protokolo tipą.



Pav. 15: Klasės Port UML diagrama

Port klasė skirta iš parametrų failo įvedamų požymių saugojimui. Žemiau pateikiamas klasės kintamųjų ir metodų aprašas.

port – prievado numerį saugantis kintamasis.

proto – protokolo vardą saugantis kintamasis

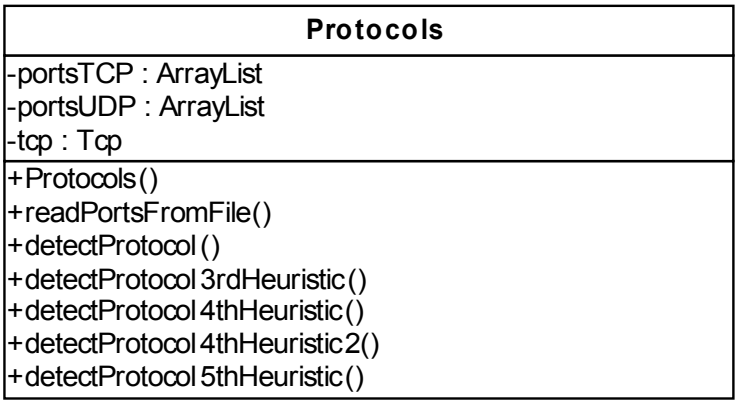
appProto – aplikacijos protokolo tipą saugantis kintamasis.

port – klasės konstruktorius.

getPort – gražina prievado numerį.

getProto – gražina protokolo tipą.

getAppProto – gražina aplikacijos protokolo tipą.



Pav. 16: Klasės Protocols UML diagrama

Klasės *Protocols* metodai naudojami tinklo srautų (tarp jų ir BitTorrent) atpažinimui.

portsTCP – žinomų TCP prievadų sąrašas.

portsUDP – žinomų UDP prievadų sąrašas.

tcp – kintamasis, naudojamas TCP antraštės aprašo inicializavimui. Dalyvauja bandant atpažinti protokolus.

Protocols – klasės konstruktorius.

readPortsFromFile – iš parametrų failo nuskaito informaciją apie žinomus prievadus.

detectProtocol – metodas, kurio pagalba bandoma atpažinti tiriamos sesijos protokolo tipą naudojant gilią paketų analizę ir žinomus prievadus.

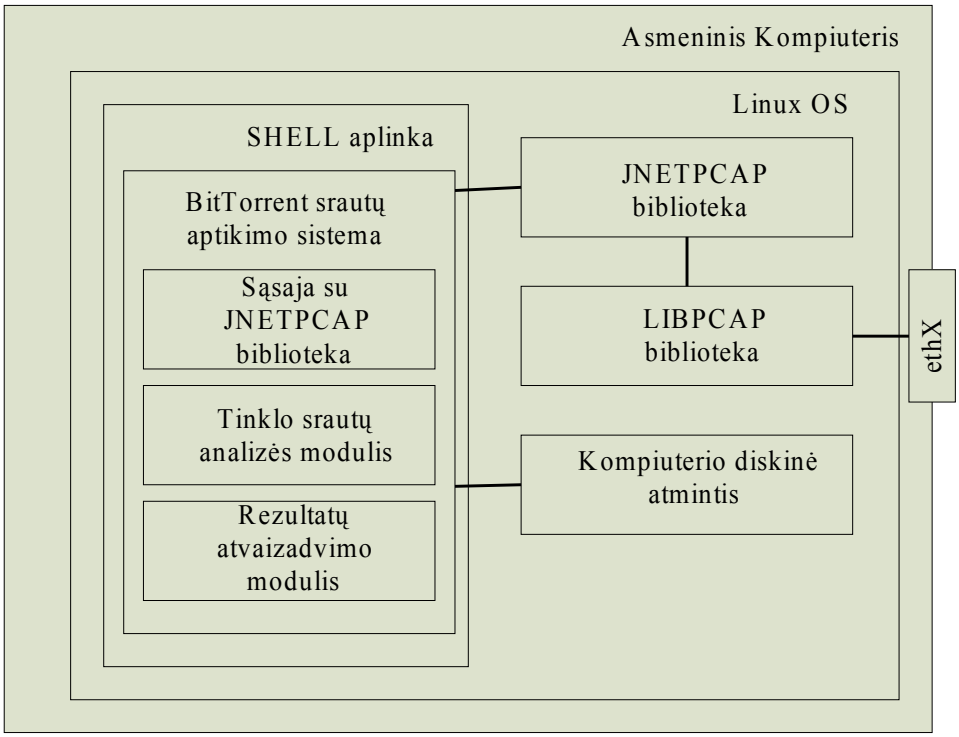
detectProtocol3rdHeuristic – metodas, kurio pagalba bandoma atpažinti tiriamos sesijos protokolo tipą naudojant trečią taškas į tašką srautų bruožą (2.6.1 „Taškas į tašką srautų bruožai“).

detectProtocol4thHeuristic – metodas, kurio pagalba bandoma atpažinti tiriamos sesijos protokolo tipą naudojant ketvirtą taškas į tašką srautų bruožą (2.6.1 „Taškas į tašką srautų bruožai“).

detectProtocol4thHeuristic2 – metodas, kurio pagalba bandoma atpažinti tiriamos sesijos protokolo tipą naudojant ketvirtą taškas į tašką srautų bruožą (2.6.1 „Taškas į tašką srautų bruožai“).

detectProtocol5thHeuristic – metodas, kurio pagalba bandoma atpažinti tiriamos sesijos protokolo tipą naudojant penktą taškas į tašką srautų bruožą (2.6.1 „Taškas į tašką srautų bruožai“).

2.5 Realizacijos modelis



Pav. 17: Sistemos išdėstymo diagrama

Sistemos elementai:

- Asmeninis kompiuteris – tai kompiuterinis įrenginys (darbo autoriaus atveju – asmeninis kompiuteris), gebantis veikti su Linux operacine sistema, priimti bei siųsti duomenis kompiuterių tinklu.
- Linux OS – Linux operacinė sistema.
- SHELL aplinka – tai programa, suteikianti vartotojui sąsają su operacine sistema.
- BitTorrent srautų aptikimo sistema – darbo metu sukurtas įrankis, kurio pagalba buvo testuojama darbe pateikta BitTorrent srautų aptikimo metodika. Sistema sudaryta iš trijų modulių:
  - Sąsajos su JNETPCAP biblioteka.
  - Tinklo srautų analizės modulio.
  - Rezultatų atvaizdavimo modulio.
- JNETPCAP biblioteka – biblioteka, leidžianti JAVA programavimo kalba parašytoms programoms prieiti prie LIBPCAP bibliotekos teikiamo funkcionalumo.
- LIBPCAP biblioteka – biblioteka, skirta tinklu perduodamų srautų perėmimui.
- Kompiuterio diskinė atmintis – vieta, skirta saugoti sistemos parametrų ir duomenų failams.

2.6 Skiriamieji BitTorrent srautų bruožai

Šioje darbo dalyje aprašomi BitTorrent srautų bruožai, būdingi šifruotiems ir nešifruotiems taškas į tašką, taškas į tracker'į susijungimams. Remiantis apibrėžtais bruožais, konstruojamas BitTorrent srautų aptikimo algoritmas ir įgyvendinama koncepcinė sistemos realizacija (žiūrėti 3. *„Šifruotų BitTorrent srautų aptikimo sistemos realizacija“*).

2.6.1 Taškas į tašką srautų bruožai

Remiantis literatūros ir kompiuterių tinklais perduodamų paketų analize, išskiriami penki požymiai, kurie yra būdingi BitTorrent taškas į tašką srautams.

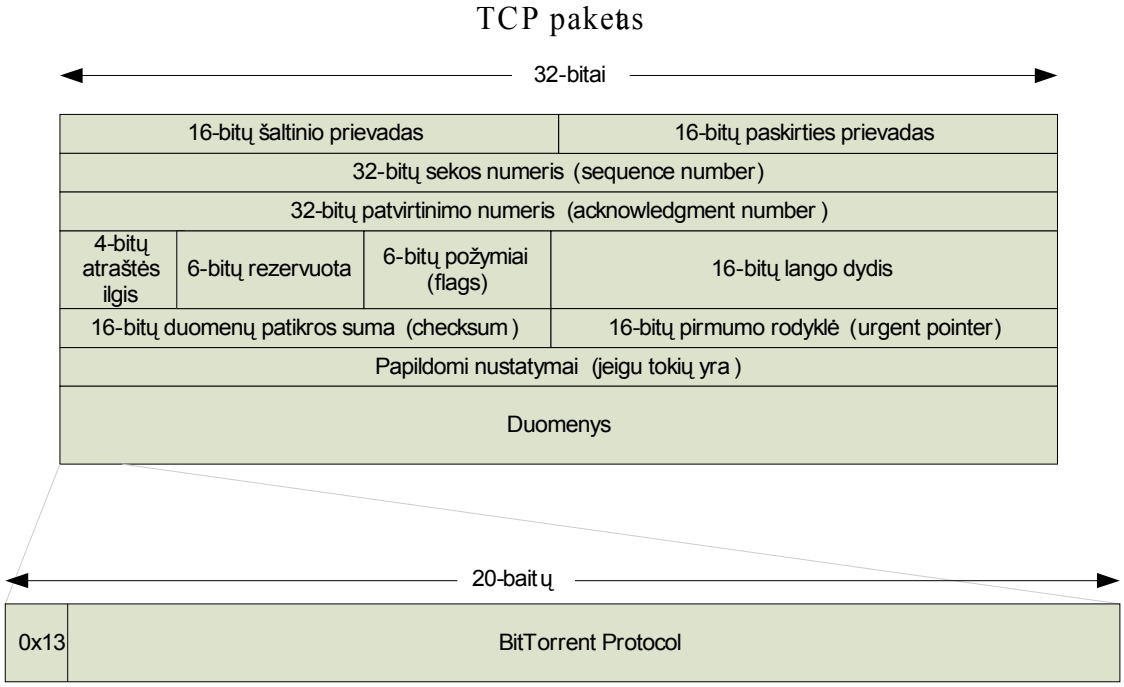
Žemiau pateikiamas protokolo bruožų sąrašas leidžia:

- Nustatyti tinklo įrenginį, kuris aktyviai naudoja BitTorrent peer-to-peer protokolą duomenims persiųsti.
- Nustatyti ar tam tikra sesija, sudaryta tarp dviejų nutolusių tinklo įrenginių naudojant TCP arba UDP protokolą, yra BitTorrent P2P sesija ar ne.

1. BitTorrent parašas

1.2.3 *„Susijungimai tarp taškų“* skyriuje atliekama BiTorrent protokolo analizė, pateikiamas išsamus BitTorrent rankos paspaudimo aprašymas tarp dviejų taškų. Prieš pradėdamas siųstis duomenis iš nutolusio taško, klientas privalo inicijuoti rankos paspaudimą, kurio metu persiunčiamas BitTorrent protokolo identifikatorius.

BitTorrent protokolo identifikatorius yra 19 baitų ilgio simbolių eilutė „BitTorrent Protocol“. Prieš šią eilutę visada įrašomas 1 baitas, aprašantis protokolo identifikatoriaus ilgį baitais. Kadangi „BitTorrent Protocol“ eilutės ilgis yra 19 baitų, prieš identifikatorių esančio baito reikšmė visada lygi 0x13 (dešimtainis skaičius 19 užrašytas šešioliktaine išraiška). Pirmasis baitas ir BitTorrent protokolo identifikatorius visada rašomas tinklu perduodamo TCP paketo duomenis saugančio laukelio pradžioje (paveikslėlis nr. 18).



Pav. 18: TCP paketo struktūra. Nurodomas duomenų laukelis, ir BitTorrent protokolo parašo saugojimo vieta

Aptikus šį BitTorrent protokolo identifikatorių, sesija tarp dviejų taškų gali būti priskiriama BitTorrent P2P srautų kategorijai.

2. Standartinių prievadų naudojimas

Dalis BitTorrent klientų pagal; nutylėjimą naudoja standartinius prievadus naujų prisijungimų priėmimui. Aptikus sesijas, kuriose bent vienas iš susijungusių taškų naudoja vieną iš standartinių BitTorrent prievadų, jas galima ženklinti kaip peer-to-peer sesijas. Pagal nutylėjimą naudojamų prievadų sąrašas pateikiamas skyriuje 1.2 *„BitTorrent protokolo savybės”*.

Metodas nėra patikimas, nes vartotojai gali pakeisti susijungimams naudojamą prievadą.

3. Lygiagretus UDP ir TCP protokolų naudojimas

Esant žemiau išvardintoms sąlygoms, BitTorrent taškams būdingas lygiagretus UDP ir TCP protokolų naudojimas.

- BitTorrent klientas naudoja DHT taškų, besidalijančių jo ieškomu resursu, radimui.
- BitTorrent klientui DHT pagalba pavyksta rasti tašką, besidalijantį jo ieškomu resursu.
- BitTorrent klientas pradeda siųstis duomenis iš naujai atrasto taško, besidalijančio jo ieškomu resursu.

Aptikus lygiagretų TCP ir UDP protokolų naudojimą tarp dviejų nutolusių taškų, galime laikyti tai vienu iš požymių, kad TCP ir UDP sesijos tarp šių taškų yra BitTorrent P2P sesijos.

4. Pakartotinis prievadų naudojimas

Įprastuose TCP ar UDP susijungimuose bent vienas iš dviejų prievadų parenkamas atsitiktinai. Vargu ar tikėtina situacija, kuomet susijungimai su tais pačiais sesijos atributais (šaltinio IP, paskirties IP, šaltinio prievadas, paskirties prievadas, protokolas, paslaugos tipas) gali pasikartoti tam tikro apibrėžtumo laiko intervale.

Tačiau tarp BitTorrent taškų tai įmanoma, jeigu abi pusės naudoja fiksuotą prievadą duomenų perdavimui. Failo persiuntimas dažnai vykdomas mažais gabalais, todėl daugkartiniai susijungimai su tais pačiais sesijos atributais įmanomi trumpam laiko intervale.

Jeigu tam tikram, sąlyginai trumpam, laiko intervale aptinkami bent du susijungimai su tais pačiais sesijos atributais tarp dviejų nutolusių tinklo įrenginių – sesiją galime priskirti BitTorrent P2P srautų kategorijai.

Taip pat, mažai tikėtina, kad tas pats tinklo įrenginys (apibrėžiamas IP adresu) pakartotinai pasirinks tą patį prievadą TCP ar UDP susijungimams, nebent tai yra serveris, teikiantis kokią nors paslaugą.

Jeigu tas pats tinklo įrenginys apibrėžtam laiko intervale naudoja tą patį TCP ar UDP prievadą daugiau negu 5 kartus, sesijas susijusias su naudojamu prievadu ir tinklo įrenginiu galime priskirti BitTorrent P2P srautų kategorijai.

5. Didelis sesijos duomenų dydis

BitTorrent klientai dažnai siunčiasi failus, kurių dydis yra > 50 MB (filmai, žaidimai, muzika). Taip pat BitTorrent taškai yra „kantrūs“, t.y. susijungimai tarp taškų gali trukti ilgai, nes siunčiami dideli duomenų kiekiai.

Todėl sesijas, kurių dydis yra didesnis nei 50 MB arba trukmė ilgesnė nei 15 minučių, galime priskirti BitTorrent P2P srautų kategorijai.

Visų BitTorrent srautų bruožų veikimas grindžiamas su prielaida, kad žinomi, ne BitTorrent P2P srautai, yra atpažįstami prieš bandant atpažinti BitTorrent srautus pagal apibrėžtas charakteristikas.

Visos penkios įvardintos BitTorrent srautų charakteristikos taikomos kompleksiškai.

2.6.2 Susijungimų tarp BitTorrent taškų ir tracker'ių bruožai

Remiantis literatūros ir kompiuterių tinklais perduodamų paketų analize, išskiriami du požymiai, kurie yra būdingi BitTorrent duomenų srautams tarp taško ir tracker'io.

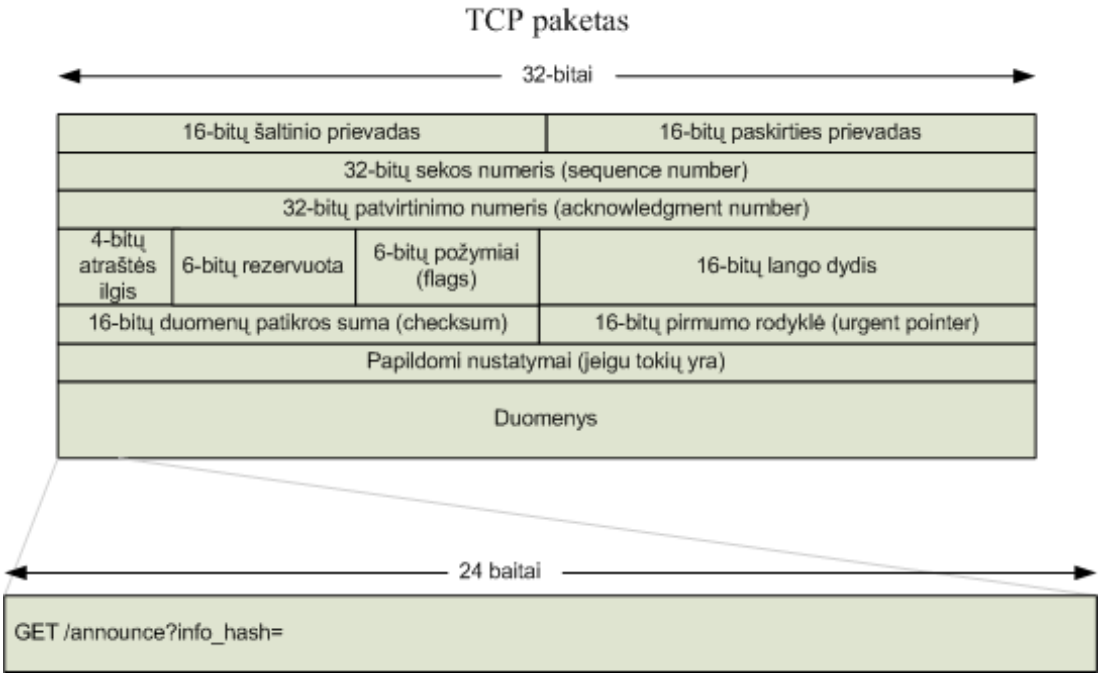
Žemiau pateikiamas protokolo bruožų sąrašas leidžia:

- Nustatyti tinklo įrenginį, kuris aktyviai naudoja BitTorrent protokolą duomenims persiųsti tarp tracker'io ir taško.
- Nustatyti ar tam tikra sesija, sudaryta tarp dviejų nutolusių tinklo įrenginių naudojant TCP arba UDP protokolą, yra BitTorrent sesija tarp taško-tracker'io.

1. GET /announce užklausa

1.2.2 *„Susijungimai tarp tracker'io ir taškų“* skyriuje atliekama BiTorrent protokolo analizė, pateikiamas išsamus BitTorrent bendravimo tarp taško ir BitTorrent tracker'io aprašymas. Kreipdamasis į tracker'į, taškas visada siunčia *GET /announce?info_hash=* užklausą.

Užklausos ilgis yra 24 baitai. Užklausa visada įrašoma tinklu perduodamo TCP paketo duomenis saugančio laukelio pradžioje (paveikslėlis nr. 19).



Pav. 19: TCP paketo struktūra. Nurodomas duomenų laukelis, ir BitTorrent parašo, būdingo susijungimams tarp taško ir tracker'io, saugojimo vieta

Aptikus šį BitTorrent taško/tracker'io sesijos identifikatorių, būdingą sesijoms tarp taško ir tracker'io, sesija gali būti priskiriama BitTorrent P2P srautų kategorijai. Šis metodas neveikia su šifruotais srautais.

2. Standartinių prievadų naudojimas

Dalis BitTorrent tracker'ių pagal; nutylėjimą naudoja standartinius prievadus prisijungimams iš taškų priimti. Aptikus sesijas, kuriose bent vienas iš naudojamų prievadų yra standartinis BitTorrent prievadas, jas galima priskirti BitTorrent P2P srautų kategorijai . Pagal nutylėjimą naudojamų prievadų sąrašas pateikiamas skyriuje 1.2 *„BitTorrent protokolo savybės”*.

Metodas nėra patikimas, nes vartotojai gali pakeisti prievadą, kurį BitTorrent tracker'is naudoja taškų prisijungimams priimti, tačiau jis padeda aptikti taško/tracker'io bendravimą jeigu perduodami duomenys yra šifruoti, bet susijungimui naudojamas standartinis prievadas.

2.6.2 Apibendrinimas

Šiame skyriuje buvo įvardinti ir aptarti bruožai, būdingi BitTorrent susijungimams tarp taškų, ir tarp taškų ir tracker'ių. Naudojant įvardintus BitTorrent susijungimų bruožus, galima aptikti BitTorrent srautus (šifruotus ir ne) tarp taškų, ir tarp taškų ir tracker'ių.

Darbo metu sukurto įrankio realizacija aptariama skyriuje 3. *„Šifruotų BitTorrent srautų aptikimo sistemos realizacija“*. Sudaryto BitTorrent srautų aptikimo algoritmo veiksmingumas tiriamas eksperimentinėje darbo dalyje.

3. Šifruotų BitTorrent srautų aptikimo sistemos realizacija

Šioje darbo dalyje pateikiamas išsamus BitTorrent srautų aptikimui sukurtos programos aprašymas, į kurį įeina programos veikimo aprašymas, programos testavimo modelis ir duomenys, bei apibendrinančios išvados.

3.1 Sistemos veikimo aprašymas

Darbo metu sukurta ir naudota programa neturi grafinės vartotojo sąsajos. Programos paleidimas vykdomas iš SHELL aplinkos, naudojant komandinę eilutę. Programos paleidimo komandinėj eilutėj nurodomi argumentai, apibrėžiantys programos darbo režimą.

Programos paleidimas naudojant komandinę eilutę:

```
java detectP2P [ -f pcap_file ]
               [ -i network_interface ]
               [ -t time ] [ -c count ] [ -v ]
```

Toliau pateikiamas komandinės eilutės argumentų paskirties aprašymas (lentelė nr. 5).

Argumentas	Aprašymas
-f	Nurodomas PCAP formato duomenų failas, kuriame saugomi analizei paruoštas tinklo duomenų srautas.
-i	Nurodomas egzistuojantis tinklo prievadas, kuriuo perduodamus paketus norima perimti ir analizuoti.
-t	Nurodomas laikas, kuriam prabėgus programa baigia darbą. Laikas nurodomas milisekundėmis. Dėl *libpcap* bibliotekos savybių, neįmanoma nutraukti paketų perėmimo, jeigu prievadas negauna jokių paketų. Tokiu atveju programa baigs darbą tada, kai gaus pirmą paketą po nurodyto laiko pabaigos.
-c	Nurodomas paketų kiekis, kurį perėmus programa baigia darbą.
-v	Startavus programą su šiuo parametru, baigusi darbą programa į ekraną išves sesijų lentelės turinį.

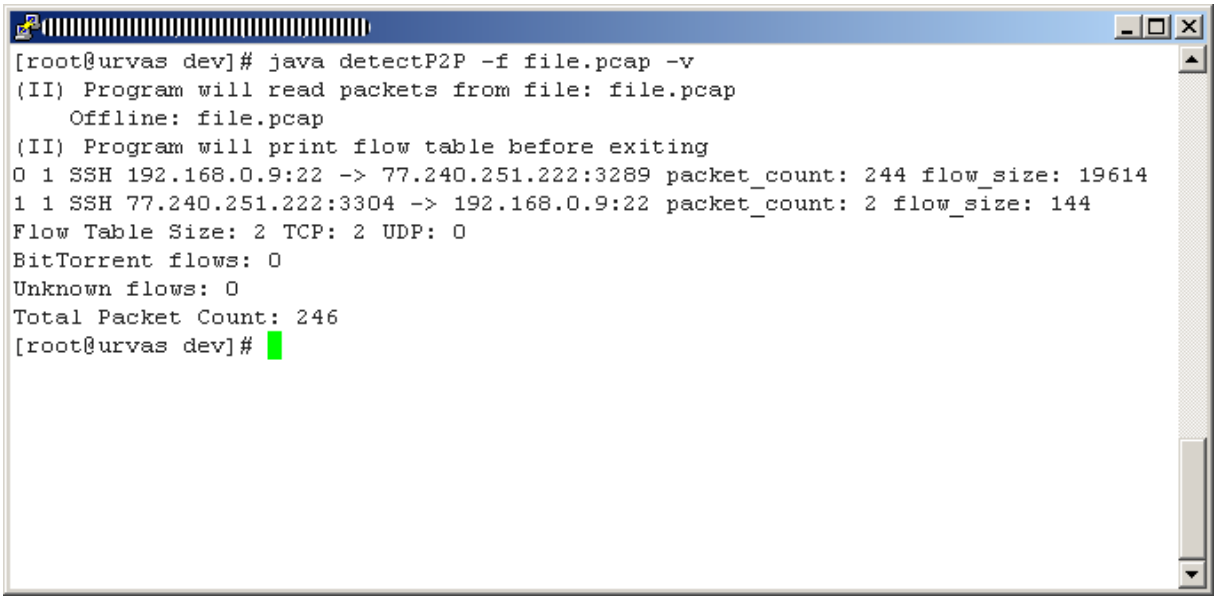
Lentelė 5: Programos komandinės eilutės argumentų paskirties aprašymas

Visi komandinės eilutės argumentai yra neprivalomi. Tačiau jeigu naudojamas -i argumentas, t.y. paketai skaitomi ne iš failo, bet aktyvaus tinklo prievado, rekomenduojama nurodyti argumentus -t arba -c, priešingu atveju programa dirbs be galo ir vienintelis būdas užbaigti programos darbui - nužudyti programos procesą. Nužudžius programos procesą, programos darbo metu sukaupti analizės rezultatai vartotojui nepateikiami.

Jeigu parametrai -t ir -c nurodomi kartu, programa baigs darbą pasiekus arba laiko limitą, arba paketų limitą. Priklauso nuo to, kuris limitas pasiektas pirmas.

Jeigu programa startuojama nenurodžius nei vieno iš argumentų -f arba -i, t.y. nenurodžius duomenų failo vardo ir nenurodžius stebimos tinklo sąsajos, programa pagal nutylėjimą startuos ir bandys naudoti tinklo prievadą pavadinimu *eth1*.

Paveikslėlyje nr. 20 pateikiamas programos paleidimo pavyzdys.



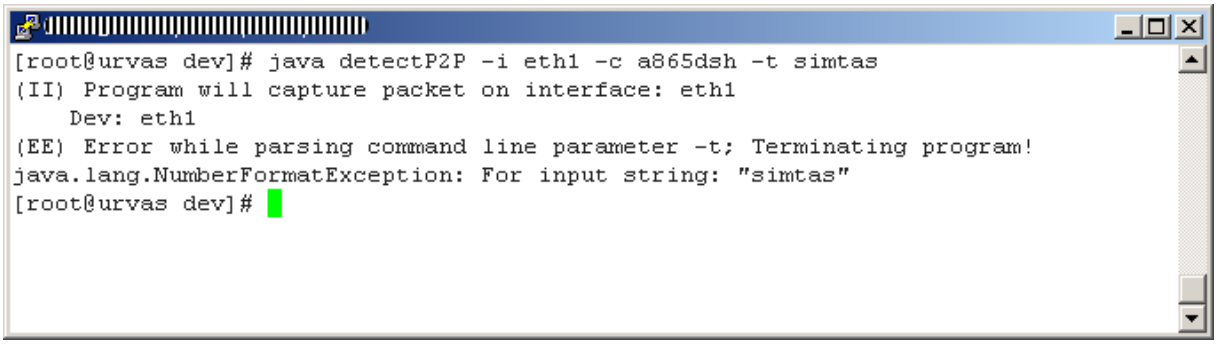
Pav. 20: Programos paleidimas. Rezultatai tekstiniu pavidalu pateikiami vartotojui

Pateiktame pavyzdyje (20 pav.) prie testinio serverio prisijungiama *SSH* protokolu, naudojant *Putty* klientą. BitTorrent srautų aptikimo programa paleidžiama naudojant komandinę eilutę:

```
java detectP2P -f file.pcap -v
```

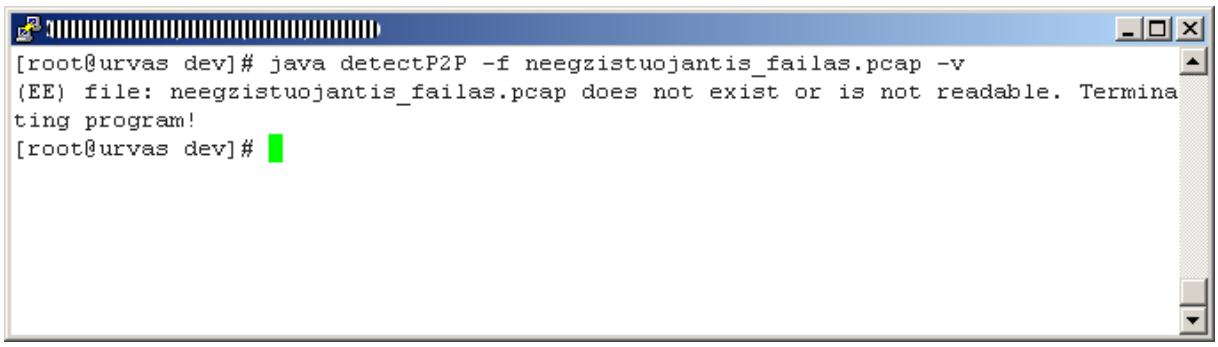
Programai nurodoma, kad duomenys turi būti skaitomi iš failo *file.pcap*, o rezultatai pateikiami vartotojui programai baigus darbą.

Programoje įdiegti apsaugos mechanizmai, tikrinantys perduodamų argumentų korektiškumą. Pavyzdžiui, programos veikimo laikas (nurodomas argumentu -t) turi būt pateiktas skaitiniu pavidalu; failas (nurodomas argumentu -f) turi egzistuoti. Paveikslėliuose 21 ir 22 pateikiamas programos veikimo pavyzdys esant nekorektiškiems argumentams.



Pav. 21: Programos reakcija į neteisingai nurodytus -c ir -t argumentus

Programa startuojama (21 pav.) nurodant neteisingas reikšmes parametrams -c ir -t. Programa gali priimti tik skaitines argumentų reikšmes, tuo tarpu pavyzdyje pateikiamos reikšmės susideda iš neskaitinių simbolių. Programą startuojant su neteisingais parametrais, programa vartotojui praneša apie aptiktą klaidą ir baigia darbą.



Pav. 22: Programos reakcija, nurodžius neegzistuojantį duomenų failą

Programa startuojama (22 pav.) nurodant neegzistuojančio duomenų failo vardą (argumentas -f). Programą startuojant su neteisingais parametrais, programa vartotojui praneša apie aptiktą klaidą ir baigia darbą.

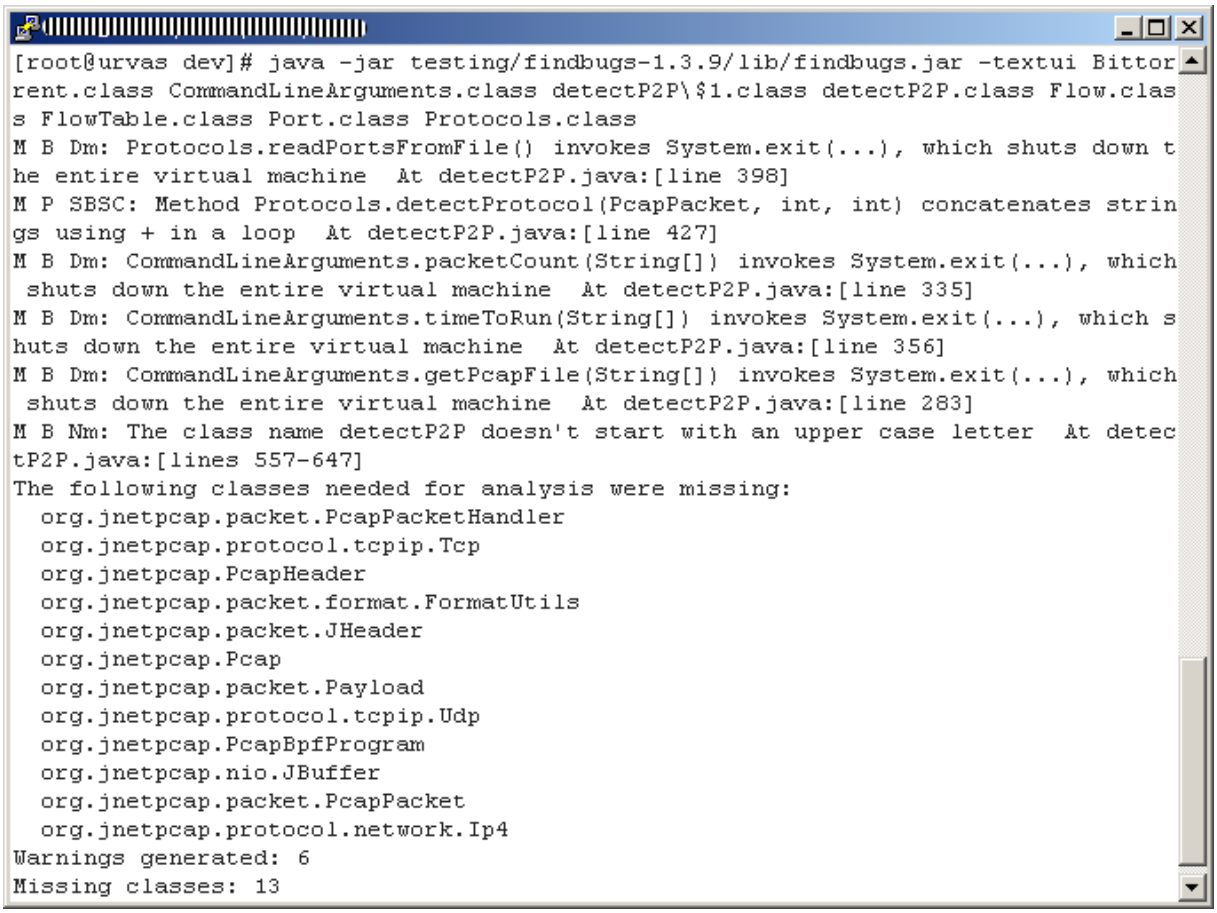
3.2 Statinė programos kodo analizė

Statinei programos kodo analizei atlikti į pagalbą buvo pasitelktas atvirojo kodo nemokamas įrankis FindBugs [30]. Testavimo metu buvo naudojama 1.3.9 FindBugs įrankio versija.
Sąrašas klasių, testuotų naudojant statinės kodo analizės metodą:

- Bittorrent.class
- CommandLineArguments.class
- detectP2P$1.class
- detectP2P.class
- Flow.class
- FlowTable.class
- Port.class
- Protocols.class

Įrankio paleidimui naudojama komandinė eilutė:
[baka@urvas dev]$ java -jar findbugs.jar -textui <argumentai>
Kur <argumentai> - visų testuojamų JAVA klasių pavadinimai atskirti tarpo simboliu.

Testavimo rezultatai pateikiami paveiksliuke nr. 20.



Pav. 23: Statinės kodo analizės rezultatai

Statinės kodo analizės metu ištestuota 100% programos kodo. Klasės, priklausančios *jnetpcap* bibliotekai testuojamos nebuvo, kadangi tai trečios šalies produktas.

Testavimo metu (23 pav.) sugeneruoti šeši perspėjimai apie galimas bėdas programos kode. Ištyrus kiekvieną pranešimą, buvo prieita prie išvados, kad visi sugeneruoti perspėjimai yra kosmetinio pobūdžio ir realaus poveikio programos veikimui neturi.

3.3. Sistemos testavimas

Šioje darbo dalyje pateikiama informacija apie magistrinio darbo metu realizuotos sistemos testavimą: tikslai, apimtys, testavimo procesas, rezultatai. Skyriaus pabaigoje pateikiamos apibendrinančios išvados.

3.3.1 Testavimo tikslai ir objektai

Testavimo objektai yra:

- JAVA programavimo kalba parašytas koncepcinis įrankis, kurio pagalba siekiama

ištirti metodo, skirto šifruotų BitTorrent peer-to-peer srautų aptikimui, veiksmingumą. Metodo ir susijusių BitTorrent srautų charakteristikų aprašymas pateikiamas 2.6 *„Skiriamieji BitTorrent srautų bruožai“* skyriuje.

- Algoritmas, realizuoti euristiniai metodai, kurių pagalba siekiama aptikti šifruotus BitTorrent peer-to-peer srautus.

Testavimo fazės metu siekiama patvirtinti, kad įrankio realizacija atitinka iškeltus reikalavimus, srautams apdoroti skirtas algoritmas gražina rezultatus, kurių tikimasi.

3.3.2 Testavimo apimtis

Programos ir algoritmo testavimui naudojamos testavimo rūšys:

- Validavimo[8]
- Našumo[9]

Pagrindinis dėmesys sutelkiamas į algoritmo realizacijos testavimą. Parinkus tinkamai parinktą įvesties duomenų aibę, siekiama patvirtinti, kad algoritmas veikia korektiškai ir pateikia laukiamus rezultatus.

Našumo testavimo metu įvertinama įrankio veikimo sparta esant skirtingų įvesties duomenų aibių dydžiui.

3.3.3 Testavimo strategija

Įrankis geba aptikti šifruotus ir nešifruotus BitTorrent srautus. BitTorrent srautų aptikimo algoritmas sudarytas iš septynių metodų, gebančių aptikti BitTorrent srautus pagal skirtingas charakteristikas (žiūrėti skyrių 2.6.1 *„Taškas į tašką srautų bruožai“*). Penki metodai skirti BitTorrent duomenų srautų identifikavimui tarp taškų, du metodai skirti BitTorrent srautų identifikavimui tarp taškų ir tracker'io.

Kiekvienam metodui ištestuoti buvo paruošta po tris rinkinius specialių duomenų. Įvesties duomenis sudarančių sesijų protokolai buvo nustatyti rankiniu būdu, naudojant paketų analizatorių *tcpdump*. Kadangi įvesties duomenis sudarančių sesijų protokolai žinomi, lengva įvertinti įrankio pateikiamų rezultatų korektiškumą ir įsitikinti, kad BitTorrent srautams identifikuoti naudojami metodai veikia teisingai.

8 Validavimo – angl. *validation*
9 Našumo – angl. *performance*

3.3.4 Įvesties duomenų rinkiniai

Sistemos testavimui skirti įvesties duomenys saugomi *PCAP* formato failuose. Testavimui skirti duomenų rinkiniai surinkti ir išanalizuoti naudojant kompiuterių tinklu perduodamų paketų analizatorius *tcpdump* ir *wireshark*. Trumpas testavimui skirtų įvesties duomenų aprašas pateikiamas žemiau, įvesties duomenų rinkinius suskirstant pagal BitTorrent srautų identifikavimo metodus.

Taškas į tašką srautų aptikimo metodai

1. BitTorrent parašas

Failo pavadinimas	Paketų skaičius	Sesijų skaičius	BitTorrent sesijų skaičius	Duomenų dydis (baitais)	BT Duomenys šifruoti
bt_signature_1.pcap	244	1	1	164567	NE
bt_signature_2.pcap	51329	27	27	44099795	NE
bt_signature_3.pcap	11377	20	5	9481909	NE

Lentelė 6: 1-ajam BitTorent srautų aptikimo metodui skirti įvesties duomenų rinkiniai. Visuose įvesties rinkiniuose BitTorrent sesijos yra tarp taškų, nešifruotos.

2. Standartinių prievadų naudojimas

Failo pavadinimas	Paketų skaičius	Sesijų skaičius	BitTorrent sesijų skaičius	Duomenų dydis (baitais)	BT Duomenys šifruoti
std_port_1.pcap	19055	1	1	18097983	NE
std_port_2.pcap	70	1	1	6314	TAIP
std_port_3.pcap	4889	10	3	3814260	NE

Lentelė 7: 2-ajam BitTorrent srautų aptikimo metodui skirti įvesties duomenų rinkiniai. Visuose įvesties duomenų rinkiniuose BitTorrent sesijos yra tarp taškų

3. Lygiagretus UDP ir TCP protokolų naudojimas

Failo pavadinimas	Paketų skaičius	Sesijų skaičius	BitTorrent sesijų skaičius	Duomenų dydis (baitais)	BT Duomenys šifruoti
par_udptcp_1.pcap	2263	8	5	0	TAIP
par_udptcp_2.pcap	2373	12	5	0	TAIP
par_udptcp_3.pcap	0	2	2	5276	TAIP

Lentelė 8: 3-ajam BitTorrent srautų aptikimo metodui skirti įvesties duomenų rinkiniai. Visuose įvesties duomenų rinkiniuose BitTorrent sesijos yra tarp taškų

4. Pakartotinis prievadų naudojimas

Failo pavadinimas	Paketų skaičius	Sesijų skaičius	BitTorrent sesijų skaičius	Duomenų dydis (baitais)	BT Duomenys šifruoti
rep_port_1.pcap	137	6	6	12414	TAIP
rep_port_2.pcap	128949	0	6	12493525	TAIP
rep_port_3.pcap	113713	10	6	0	TAIP

Lentelė 9: 4-ajam BitTorrent srautų aptikimo metodui skirti įvesties duomenų rinkiniai. Visuose įvesties duomenų rinkiniuose BitTorrent sesijos yra tarp taškų

5. Didelis sesijos duomenų dydis

Failo pavadinimas	Paketų skaičius	Sesijų skaičius	BitTorrent sesijų skaičius	Duomenų dydis (baitais)	BT Duomenys šifruoti
flow_size_1.pcap	81224	6	1	64000000	TAIP
flow_size_2.pcap	111241	80	1	84000000	TAIP
flow_size_3.pcap	251241	84	2	124000000	TAIP

Lentelė 10: 5-ajam BitTorrent srautų aptikimo metodui skirti įvesties duomenų rinkiniai. Visuose įvesties duomenų rinkiniuose BitTorrent sesijos yra tarp taškų

BitTorrent taškų ir tracker'ių srautų aptikimo metodai

1. GET /announce užklausa

Failo pavadinimas	Paketų skaičius	Sesijų skaičius	BitTorrent sesijų skaičius	Duomenų dydis (baitais)	BT Duomenys šifruoti
tracker_get_1.pcap	10	1	1	1662	NE
tracker_get_2.pcap	1155	8	1	350145	NE
tracker_get_3.pcap	1171	16	1	352881	NE

Lentelė 11: 1-ajam BitTorrent srautų tarp taško ir tracker'io aptikimo metodui skirti įvesties duomenų rinkiniai. Visuose įvesties duomenų rinkiniuose BitTorrent sesijos yra tarp taško ir tracker'io

2. Standartinių prievadų naudojimas

Failo pavadinimas	Paketų skaičius	Sesijų skaičius	BitTorrent sesijų skaičius	Duomenų dydis (baitais)	BT Duomenys šifruoti
tracker_port_1.pcap	10	1	1	1662	NE
tracker_port_2.pcap	1155	8	1	350145	NE
tracker_port_3.pcap	1171	16	1	352881	NE

Lentelė 12: 2-ajam BitTorrent srautų tarp taško ir tracker'io aptikimo metodui skirti įvesties duomenų rinkiniai. Visuose įvesties duomenų rinkiniuose BitTorrent sesijos yra tarp taško ir tracker'io. Siekiant ištestuoti šį metodą, 1-asis metodas laikinai buvo išjungtas.

Sistemos testavimo rezultatai aptariami skyriuje 3.4 *„Sistemos testavimo rezultatai“*.

3.3.5 Sistemos našumo testavimas

Testuojant sistemos našumą, siekiama įvertinti sistemos darbo spartos kitimą, kintant paketų, kuriuos reikia apdoroti, skaičiui.

Testavimui naudojama 40 įvesties duomenų rinkinių, kuriuose saugomų paketų skaičius auga po ~2000 pradedant nuo pirmo rinkinio. Testavimo rezultatai pateikiami paveiksliuke nr. 24.

Pav. 24: Sistemos darbo laiko pokytis, augant paketų skaičiui, kuriuos reikia apdoroti

Laiko matavimams atlikti buvo naudojamas Linux operacinės sistemos įrankis *time*.

Kompiuterio, naudoto sistemos našumo testavimui, parametrai:

Modelis: Pentium III

CPU sparta: 1Ghz

Procesorių skaičius: 2

Operatyvinės atminties kiekis: 2GB

Testavimo rezultatai aptariami skyriuje 3.4 *„Sistemos testavimo rezultatai“*.

3.4 Sistemos testavimo rezultatai

Sistemos testavimo metu buvo ištestuotas sistemos našumas, patikrintas algoritmo veikimo korektiškumas.

Algoritmo korektiškumo tikrinimo atveju, visi gauti rezultatai sutapo su įvesties duomenų aprašu (žiūrėti 3.3.4 *„Įvesties duomenų rinkiniai“*). Todėl galima teigti, kad algoritmas veikia korektiškai ir duoda norimus rezultatus.

Sistemos našumo (3.3.5 *„Sistemos našumo testavimas“*) testavimas atskleidė sistemos darbo spartą. Tarp dviejų naudojamų metrikų (laikas/paketų skaičius) yra stipri tiesinė priklausomybė. Tai reiškia, kad tam pačiam paketų kiekiui apdoroti skirtingu laiko momentu reikia tiek pat laiko sąnaudų, t.y. sistemos darbo sparta stabili.

Skyriuje 4. *„Eksperimentinis šifruotų BitTorret srautų aptikimo sistemos tyrimas“* atliekamas sistemos eksperimentinis sistemos tyrimas. Atliekant eksperimentinį sistemos tyrimą, remiamasi šio skyriaus išvadomis, patvirtinančiomis, kad BitTorrent srautų aptikimo algoritmas veikia korektiškai.

4. Eksperimentinis šifruotų BitTorrent srautų aptikimo sistemos tyrimas

Magistrinio darbo realizacijos veikimui ir efektyvumui patikrinti buvo atliktas eksperimentas, kurio metu buvo generuojami BitTorrent ir kitų aplikacijų duomenų srautai, bei analizuojamas sistemos gebėjimas atpažinti šifruotus BitTorrent duomenų srautus.

4.1 Eksperimentui atlikti naudota aplinka

Eksperimentas vykdomas dviem etapais:

- Eksperimentas, vykdant pasyvų sistemos testavimą, kuomet sistemos testavimui naudojama aibė iš anksto sukauptų duomenų. Duomenys saugomi PCAP formato failuose.
- Eksperimentas, vykdant aktyvų sistemos testavimą. Aktyvaus testavimo metu sistema stebi ir analizuoja realiu laiku perduodamus duomenų srautus.

Eksperimentas buvo atliekamas laboratorijoje, kurios schema pateikta 25 paveiksle.

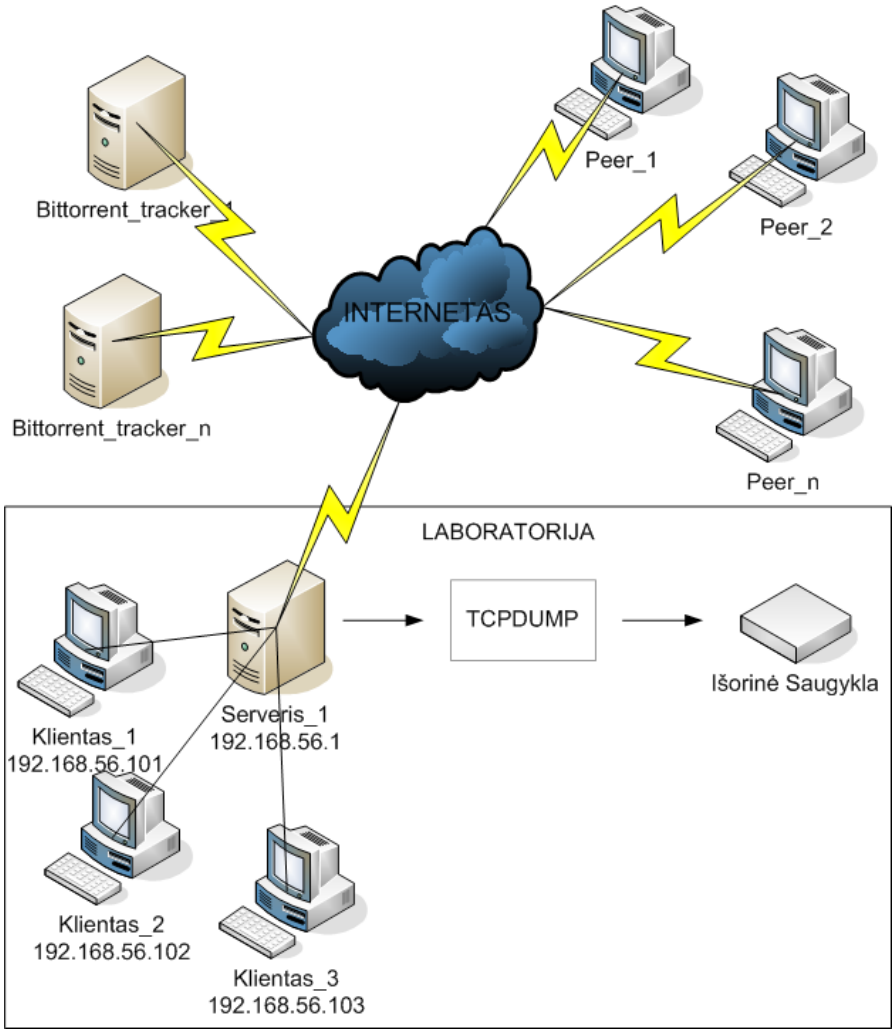
Testavimui naudojamą laboratoriją sudaro:

- Trys klientinės sistemos, atliekančios vartotojų roles. Vartotojai naudojasi BitTorrent, HTTP, FTP, TELNET, SSH, DNS paslaugomis.
- Vienas serveris, atliekantis maršrutizatoriaus rolę. Šis įrenginys yra centrinis taškas, per kurį keliauja visi duomenų srautai tarp laboratorijoje esančių klientų ir interneto.
- Išorinė saugykla, kurioje saugojami *tcpdump* įrankio pagalba surinkti duomenys.

Visi įrenginiai įjungti į *ethernet* tinklą, kuris per pagrindinį serverį pasiekia internetą.

Eksperimento metu naudoti BitTorrent peer-to-peer klientai:

- rtorrent
- qbittorrent
- vuze
- ktorrent
- deluge
- transmission



Pav. 25: Eksperimentui atlikti naudota laboratorija

Pasyviam sistemos testavimui skirtų duomenų rinkimui buvo naudojamas *tcpdump* įrankis. Renkant duomenis saugomos ir paketų antraštės, ir perduodami duomenys.

Aktyvaus sistemos testavimo etapu, sistema buvo įdiegta įrenginyje pavadinimu

Serveris_1.Sistema stebėjo ir analizavo visus persiunčiamus duomenis, o aptikusi BitTorrent sesiją informaciją apie ją išsaugodavo.

Srautų generavimas dalinai automatizuotas. Dalies srautų generavimas automatizuotas pasinaudojant sisteminiais Linux operacinės sistemos įrankiais. Pavyzdžiui HTTP srautui generuoti buvo pasitelktas analogiška komanda:

```
for i in "http://bbc.co.uk" "http://delfi.lt"
"http://slashdot.org" ; do wget -r $i ; sleep 1000 ; done
```

Tai ciklas, leidžiantis *wget* komandos pagalba paeiliui siųsti nurodytas interneto svetaines.
Dalis srautų buvo generuojama rankiniu būdu, pavyzdžiui BitTorrent srautai buvo generuojami paleidžiant BitTorrent klientą ir nurodant, kokius failus jam siųstis.

BitTorrent klientai eksperimento metu galėjo siųstis duomenis, ir priimti įeinančius prisijungimus iš BitTorrent klientų, veikiančių tame pačiame potinklyje.

Eksperimento metu gauti rezultatai pateikiami skyriuje 4.2 *„Eksperimento eiga“*.

4.2 Eksperimento eiga

Duomenų, skirtų pasyviam eksperimento etapui, kaupimas vyko nenuosekliai (laiko atžvilgiu). Bendras duomenų kaupimo laikas – 24 valandos.
Aktyvaus eksperimento etapo metu, tinklas buvo stebimas 10 valandų.
Žinomiems duomenų srautams (SSH, HTTP, DNS, TELNET, NETBIOS ir t.t.) identifikuoti buvo naudojamas tik prievadų metodas. Tam tikslui sistema iš parametrų failo nusiskaitydavo sąrašą prievadų ir juos naudojančias paslaugas. Į standartinias aplikacijas aprašančių prievadų sąrašą įtraukti prievadai nuo 1 iki 1023 imtinai. Standartinių prievadų sąrašas sudarytas remiantis IANA informacija [31]. Žinomų aplikacijų identifikavimui lygiai taip pat gali būti panaudojami ir kiti metodai, pavyzdžiui gili paketų analizė. Šiame darbe prievadų analizės, siekiant identifikuoti ne BitTorrent srautus pakako, nes nei viena eksperimento metu naudota BitTorrent aplikacija nesinaudojo standartiniais prievadais, kurių numeris yra < 1024. Šiuo aspektu eksperimento aplinka buvo kontroliuojama. Realiomis gyvenimo sąlygomis papildomi metodai turi būti taikomi žinomų aplikacijų identifikavimui, jeigu norima itin tiksliai nustatyti, kuriai aplikacija konkreti duomenų sesija priklauso.

Sukauptų duomenų analizė atlikta naudojant darbo metu sukurta šifruotų BitTorrent srautų aptikimo sistemą.

Eksperimento metu gauti rezultatai pateikiami skyriuje 4.2 *„Eksperimento eiga“*.

4.3 Eksperimento rezultatai

Pasyvaus eksperimento metu buvo sukaupta 4127 MB duomenų. Duomenų analizės rezultatai pateikiami lentelėje nr. 13.

Aktyvaus eksperimento metu stebimu tinklu, naudojant UDP arba TCP protokolą buvo perduota 5084 MB duomenų. Duomenų analizės rezultatai pateikiami lentelėje nr. 14

Kategorija	Sesijų kiekis	Sesijų kiekis (%)
as	32366	88.45
bt	4066	11.12
nz	161	0.43

Lentelė 13: Duomenų analizės rezultatai. Pasyvus eksperimentas

Kategorija	Sesijų kiekis	Sesijų kiekis (%)
as	18986	79.01
bt	4914	20.45
nz	127	0.54

Lentelė 14: Duomenų analizės rezultatai. Aktyvus eksperimentas

Aiškumo ir paprastumo dėlei, srautai buvo suskirstyti į tris kategorijas:

- as – šiai kategorijai priskiriami atpažintos ne BitTorrent duomenų sesijos.
- bt – šiai kategorijai priskiriami atpažintos BitTorrent duomenų sesijos.
- nz – šiai kategorijai priskiriamos neatpažintos duomenų sesijos.

Taikant darbe sukurtą algoritmą, pirmiausiai atpažįstami duomenų srautai, priklausantys žinomoms aplikacijoms. Jie aptinkami naudojant standartinių prievadų analizės metodą. Po šios fazės, neatpažintas duomenų srautas papildomai analizuojamas naudojant BitTorrent charakteristikas.

Eksperimento rezultatai patvirtina, kad sukurtas algoritmas geba atpažinti šifruotus ir nešifruotus BitTorrent duomenų srautus, naudodamasis darbe apibrėžtomis BitTorrent srautų charakteristikomis, sudarytomis pagal BitTorrent srautų elgesio bruožus.. Iš pateiktų rezultatų taip pat matome, kad ne visi duomenų srautai yra atpažįstami., todėl algoritmo patikimumas, net ir atliekant testavimus laboratorijos sąlygomis, nėra 100%.

IŠVADOS

1. Atlikta egzistuojančių sistemų, skirtų BitTorrent srautų aptikimui, apžvalga, analizė ir palyginimas. Sistemų analizės tikslas – nustatyti sistemų tinkamumą problemai spręsti. Patvirtinti arba paneigti poreikį naujų metodų, sistemų kūrimui. Iš apžvelgtų sistemų tik viena geba aptikti šifruotus BitTorrent srautus, tačiau ji yra mokama ir ne atvirojo kodo.
2. Apibrėžtos BitTorrent srautų charakteristikos leido sudaryti algoritmą, gebantį atlikti šifruotų ir nešifruotų BitTorrent srautų tarp taškų, tarp taškų ir tracker'io, analizę, srautų analizės pagalba siekiant atpažinti BitTorrent srautus.
3. Algoritmo realizacija atlikta naudojant JAVA programavimo kalbą. Sukurtas portabilus įrankis, galintis veikti UNIX šeimos operacinėse sistemose. Sukurtas įrankis yra koncepcinė algoritmo realizacija. Jos pagalba buvo patvirtintas apibrėžtų BitTorrent charakteristikų ir pagal jas sudaryto algoritmo veiksmingumas. Tai atsispindi darbo dalyje, aprašančioje testavimą ir eksperimentus.
4. Sukurtas įrankis gali atlikti duomenų srautų analizę pasyviu ir aktyviu režimu. Pasyvios analizės atveju duomenys skaitomi iš *PCAP* formato duomenų failo, tuo tarpu aktyvios analizės atveju duomenys gaunami realiu laiku perimant kompiuterių tinklu perduodamus paketus.
5. Realizuotas BitTorrent srautų aptikimo algoritmas reikalauja papildomos informacijos kaupimo, apie kompiuterių tinklais perduodamus paketus ir iš jų sudarytas sesijas tarp tinklo įrenginių. Papildomos informacijos kaupimas ir analizė atliekama realiu laiku. Tai reikalauja papildomų skaičiavimo ir atminties resursų, todėl algoritmo praktinis panaudojimas esant dideliems duomenų kiekiams gali būti keblus. Reikalinga algoritmo optimizacija, bei kruopšti panaudojamos techninės įrangos analizė.
6. Eksperimento metu patvirtintas įrankio gebėjimas aptikti šifruotus ir nešifruotus BitTorrent duomenų srautus laboratorijos sąlygomis. Darbo metu sukurtas metodas, BitTorrent srautų aptikimui naudoja paketų turinio analizę ir požymius, būdingus BitTorrent srautų elgesiui.
7. Eksperimentui naudotos duomenų aibės yra pakankamai mažos. Realaus tikrų vartotojų srauto eksperimentams nepavyko gauti dėl techninių kliūčių. Plėtojant darbą, algoritmo testavimas didesnėmis duomenų aibėmis turi būti įtrauktas tarp tikslų.
8. Darbe charakterizuoti ir tirti tik BitTorrent duomenų srautai. Egzistuoja aibė kitų taškas į tašką protokolų, kurie taip pat naudoja šifravimą. Plėtojant darbą, kitų taškas į

tašką protokolų tyrimas gali būti įtrauktas tarp tikslų.

Terminai

Prievadas – skaitiniu pavidalu išreiškiamas identifikatorius, kurį naudoja 4-am OSI modelio lygmeniui priskiriami protokolai. Prievadai naudojami siekiant atpažinti skirtingus duomenų srautus, kuriuos tam tikras protokolas gali apdoroti. RFC: 793

Literatūra

1. The Usenet Newsgroups, http://www.livinginternet.com/u/u.htm, [žiūrėta 2010-05-14]
2. „Meet the Napster“ By Karl Taro Greenfeld, http://www.time.com/time/printout/0,8816,998068,00.html, [žiūrėta 2010-05-14]
3. The science of SETI@home, http://setiathome.berkeley.edu/sah_about.php, [žiūrėta 2010-05-14]
4. Hendrik Schulze, Klaus Machalski (ipoque), „Internet Study 2007“ - 2007
5. Hendrik Schulze, Klaus Machalski (ipoque), „Internet Study 2008/2009“ - 2009
6. ipoque, „P2P Survey 2006“ - 2006
7. The BitTorrent Protocol Specification, http://www.BitTorrent.org/beps/bep_0003.html, [žiūrėta 2010-05-14]
8. RFC 3174 „US Secure Hash Algorithm 1 (SHA1)“
9. RFC 3986 „Uniform Resource Identifier (URI): Generic Syntax“
10. David Harrison, Anthony Ciani, Arvid Norberg, Greg Hazel, „Tracker Peer Obfuscation“, http://BitTorrent.org/beps/bep_0008.html, [žiūrėta 2010-05-14]
11. XBT Tracker, http://xbtt.sourceforge.net/tracker, [žiūrėta 2010-05-14]
12. Opentracker – An open and free BitTorrent tracker, http://erdgeist.org/arts/software/opentracker, [žiūrėta 2010-05-14]
13. "DHT Protocol", http://BitTorrent.org/beps/bep_0005.html, [žiūrėta 2010-05-14]
14. Petar Maymounkov, David Mazi`eres, *“Kademlia: A Peer-to-peer Information System Based on the XOR Metric”* - 2002
15. „PRX Traffice Manager“ įrenginių aprašymai, http://www.ipoque.com/products/prx-traffic-manager, [žiūrėta 2009-06-14]
16. EANTC (European Advanced Network Test Center), „Peer-to-Peer Traffic Management Test“ – 2008
17. About IPP2P, http://www.ipp2p.org, [žiūrėta 2009-06-14]
18. GNU General Public License, http://www.gnu.org/licenses/gpl.html, [žiūrėta 2010-03-31]
19. GNU Lesser General Public License, http://www.gnu.org/licenses/lgpl.html, [žiūrėta 2010-03-31]
20. Nate Anderson, “Deep packet inspection engine goes open source”, http://arstechnica.com/open-source/news/2009/09/deep-packet-inspection-engine-goes-open-source.ars, [žiūrėta 2010-03-31]
21. Angelo Spognardi, Alessandro Lucarelli, Roberto Di Pietro, „A Methodology for P2P File-Sharing Traffic Detection“ - 2005
22. Alok Madhukar, Carey Williamson, „A longtitudal Study of P2P Traffic Classification, 2005
23. Marcell Perényi, Trang Dinh Dang, András Gefferth, Sándor Molnár, “Identification and Analysis of Peer-to-Peer Traffic”, 2006

24. John Hurley, Emi Garcia-Palacios, and Sakir Sezer. “Classification of P2P and HTTP Using Specific Protocol Characteristics”. Published on: The Internet of the Future (ISBN 978-3-642-03699-6); Pages: 31-40
25. ipoque's Industry Leading Deep Packet Inspection Engine Goes Open Source, http://www.ipoque.com/news-and-events/news/ipoque's-industry-leading-deep-packet-inspection-engine-goes-open-source.html, [žiūrėta 2010-03-31]
26. TCPDump / libpcap, http://www.tcpdump.org, [žiūrėta 2010-04-04]
27. Standard Output Definition, http://www.linfo.org/standard_output.html, [žiūrėta 2010-04-04]
28. ArchLinux, http://www.archlinux.org, [žiūrėta 2010-04-04]
29. http://jnetpcap.com, [žiūrėta 2010-04-04]
30. http://findbugs.sourceforge.net, [žiūrėta 2010-04-04]
31. http://www.iana.org, [žiūrėta 2010-04-04]