KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Jonas Balčiūnas

Paieškos metodų analizė ir realizacija išskirstytos maišos lentelėmis grindžiamose P2P sistemose

Magistro darbas

Darbo vadovė:
Prof. Lina Nemuraitė

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Paieškos metodų analizė ir realizacija išskirstytos maišos lentelėmis grindžiamose P2P sistemose

Magistro darbas

Vadovė
Prof. Lina Nemuraitė

Recenzentas
dr. Jonas Čeponis

Atliko
IFM-2/2 gr. studentas
Jonas Balčiūnas
2008 05 26

Kaunas, 2008

Analysis and Implementation of Search Methods in P2P Systems Based on Distributed Hash Tables

SUMMARY

The key idea of DHT systems is hash table distributed over the distributed independent nodes. The DHT are decentralized, scalable, fault tolerant and have high hit guaranties for data lookup. However, they do not support arbitrary querying which flooding schemes do: users must know exact key of the resource they are looking up in the system. In the most common solution for this is external searching engine like ftp or http.

This work presents research experiment of possible methods for arbitrary querying in DHT based on the “n-grams” and “broadcasting” techniques. Experiment was carried out using experimental P2P system created for this purpose on the base of Chord algorithm. Experimental results showed that, the most expensive (in terms of message generation) process in “n-gram” is publishing of keys to network. The analysis of both methods showed that n-grams are more practical on the relatively smaller network and “broadcasting” is more effective on the networks with implemented data replication.

Turinys

Įvadas

„Kai kam gali atrodyti, kad P2P sistemos naudojamos tik nelegaliems muzikos įrašų mainams, tačiau tokia išvada būtų skubota ir neapgalvota.[1]*“*

Šis darbas skirtas Peer-to-Peer (P2P) sistemų turinio paieškos problemai. P2P sistemos ypač populiarios intelektinės nuosavybės „dalinimosi“ (angl. sharing) programose. Tai dar nereiškia, kad jos tik tam ir tinkamos bei naudojamos. Vienas žymiausių ir didžiausių projektų, naudojantis P2P technologijas, – „SETI@home“. Jo užduotis– ieškoti nežemiškos kilmės protingų būtybių. Projekte dalyvauja virš 3 milijonų vartotojų. Po visą pasaulį išskirstytų radijo teleskopų surinktų duomenų apdorojimas ir analizė yra sudėtingi, reikalauja daug resursų bei laiko. Reikėtų didelių kompiuterių centrų, norint atlikti darbą, kurį atlieka vartotojai, savanoriškai skolinantys savo kompiuterių resursus, kai jie nenaudojami.

„SETI@home“ yra tik vienas iš projektų, besinaudojančių išskirstytų sistemų privalumais. Skandinavijos šalys yra sukūrusios didžiulę paskirstytą sistemą su centru Danijoje. Šią sistemą sudaro į tinklą sujungti įvairūs moksliniai centrai, o iš išorės vartotojui ji atrodo kaip vienas super kompiuteris, esantis vienoje geografinėje vietoje. Sistemos sukūrimo priežastis paprasta: nei vienai iš šią sistemą eksploatuojančių valstybių neužtenka pajėgumų ir finansinių galimybių savarankiškai įgyvendinti projektą, todėl buvo nutarta suvienyti mokslinius centrus Danijoje, Švedijoje, Norvegijoje ir Suomijoje. Taip atsirado sistema „NorduGrid“ su apytiksliai 5000 CPU ir 75 TB diskinės vietos. Jos tikslas– sudėtingi skaičiavimai ir milžiniškų duomenų srautų analizė [8].

Kyla daug diskusijų ir klausimų dėl P2P sistemų naudojimo. Autorių teisių gynimo organizacijos pasipiktinusios, kad tokiose sistemose vartotojai nelegaliai platina autorinius darbus, kuriuos sąžiningi vartotojai turi pirkti parduotuvėse. Tuo tarpu tose šalyse, kur apribota spaudos ir žodžio laisvė, P2P yra vienas būdų bendrauti su išoriniu pasauliu.

Keičiantis informacija arba resursais, P2P tinklai turi didelių privalumų: sudaryti iš nepriklausomų kompiuterių, kurie gali laisvai įsijungti ar atsijungti, jie nereikalauja administravimo išlaidų, todėl vis plačiau naudojami ne tik interneto bendruomenėse, bet ir mokyme, versle. Dvi pagrindinės jų klasės yra srautų schemos (angl. Flooding Schemas) ir DHT

[1] „One might believe P2P systems are mainly used for illegal music-swapping and little else, but this would be a rather hasty conclusion.“ [3]

arba „*išskirstytos maišos lentelės*“ (angl. Distributed Hash Table)[3]. Viena svarbiausių problemų P2P sistemose yra objektų, kuriais dalinasi vartotojai, paieška.

DHT sistemos pasižymi dideliu plečiamumu ir nepriklausomumu, bei užklausų taiklumu, tačiau nėra efektyvesnių paieškos metodų, kaip tik rakto publikavimas išoriniame tinkle. Srautų schemose paieška atliekama pasitelkus vidinį sistemos paieškos mechanizmą. Čia galima pasirinkti paieškos žodžius, įvairius paieškos kriterijus, o paieška atliekama tinklo viduje.

DHT sistemų privalumas yra jų didelis plečiamumas ir nepriklausomumas, tačiau esami paieškos sprendimai reikalauja išorinių mechanizmų ir taip mažina DHT privalumus. Todėl šio **darbo tikslas** – padidinti paieškos DHT sistemose galimybes, sukuriant vidinį paieškos mechanizmą Chord algoritmo pagrindu veikiančiai DHT sistemai ir ištiriant jo efektyvumą. Chord algoritmas pasirinktas todėl, kad tai efektyvus, vizualiai suvokiamas, pasižymintis dideliu plečiamumu ir tolerancija klaidoms. Be to atsižvelgta ir į algoritmo testavimo sudėtingumą. Pavyzdžiui CAN algoritmo pagrindą sudaro m-dimensijų tinklas, kuris yra sunkiau suvokiamas, kai $m > 3$. Tuo tarpu Chord algoritme visa mazgų aibė yra paskirstyta apskirimu, kas palengvina tinklo struktūros vizualinį suvokimą.

Atlikus literatūros analizę [14, 4], pasirinkti du sprendimai, kurie padėtų atlikti paiešką DHT sistemose nenaudojant išorinių resursų: tai „užtvindymo pranešimais“ metodas ir n-gramomis paremtas metodas. Kadangi šių metodų praktinis pritaikomumas nėra ištirtas, tai atliekama šiame darbe. Tyrimui atlikti sukurta eksperimentinė DHT sistema, veikianti Chord algoritmo pagrindu.

Darbo uždaviniai:

- sukurti eksperimentinę Chord algoritmo pagrindu veikiančią DHT sistemą;
- pritaikyti šiai sistemai užtvindymo pranešimais mechanizmą;
- pritaikyti šiai sistemai n-gramų metodą;
- eksperimentiškai ištirti vieno iš paieškos metodų veikimą;
- įvertinti abu sprendimus ir sudaryti rekomendacijas paieškos algoritmams DHT sistemose realizuoti.

Chord algoritmo veikimu paremtai DHT sistemai realizuoti pasirinkta Java kalba, mazgų tarpusavio bendravimui naudojant Java-RMI technologiją.

Darbo struktūra:

1 skyriuje pateikiamas trumpas įvadas į P2P sistemų tematiką. Išvardinami pagrindiniai uždaviniai, kurios teks išspręsti darbo metu.

2 skyriuje analizuojamos P2P sistemų charakteristikos, pateikiami esamų sistemų pavyzdžiai ir veikimo principai. Palyginami tarpusavyje srautų schemų ir DHT tinklų paieškos mechanizmų trūkumai ir privalumai. Aprašoma paieškos problematika DHT sistemose. Išaiškinami du teoriniai paieškos problemos sprendimai aptikti literatūroje. Aprašoma šių sprendimų įgyvendinimo idėjos. Apibrėžiama magistro darbo tikslai ir uždaviniai.

3 skyriuje pateikiamas sistemos panaudos atvejų modelis ir specifikacija bei apibrėžti nefunkciniai reikalavimai.

4 skyriuje pateikiama įgyvendintos sistemos architektūros analizė: pateiktas loginis sistemos komponentų modelis bei klasių modelis. Sekų diagramomis pateikta sistemos elgsena.

5 skyriuje aprašoma eksperimento atlikto su vienu iš siūlomų metodų rezultatai. Eksperimento vykdymo eiga aprašyta priede Nr. 2.

6 skyriuje suformuojamos atlikto darbo išvados.

1. P2P sistemų analizė

Šiuo metu paskirstytos sistemos labai paplitusios. Taip susiklostė dėl to, kad jos naudojamos įvairiose srityse, pradedant didelių kompiuterių centrų skaičiavimuose, automatizuotoje kontrolėje, sensorių tinkluose, kritinėse sistemose, failų apsikeitimo ir daugelyje kitų sričių.

Apibrėžtis:

> *„Išskirstytoji sistema – tai aibė savarankiškai veikiančių ir sujungtų į tinklą kompiuterių, kurie koordinuoja savo veiklą ir bendrauja pranešimų pagalba. Atskirų sistemos elementų resursai naudojami išplėsti visos sistemos talpai ir skaičiavimo galiai. Išoriniam vartotojui tokia sistema atrodo vientisa.* [2]*“*

Apibrėžtis:

> *„Terminas P2P apibrėžia klasę sistemų ir programų, kurios decentralizuotai koordinuoja atskirų komponentų resursus bendros sistemos funkcionalumo užtikrinimui.*[3]*“*

P2P – paskirstytoji sistema, kurioje nėra pagrindinio arba **centrinio** mazgo (arba kelių **centrinių** mazgų). Tokios sistemos nepalaiko kliento-serverio architektūros. Labai dažnai serveris būna silpniausia sistemos grandis (angl. Bottleneck). Nustojus veikti vienam serveriui, visa arba dalis sistemos tampa nepasiekiama. Veikiančių komponentų P2P sistemoje gali būti daug (~10^6 eilės), todėl toks sistemos pažeidžiamumas (eliminuojant vieną iš mazgų – serverį), yra nepriimtinas.

---

[2] „*A distributed system is a set of autonomous computers connected through a network, which coordinate their activity and communicate by message passing to provide shared resources, as storage and computational power. It is perceived as a single entity by the user“*[George Couloris ir kiti 2000; šaltinis 1]

[3] „The term “peer-to-peer” (P2P) refers to a class of systems and applications that employ distributed resources to perform a function in a decentralized manner.“ [5]

1.1. Esamų P2P sistemų apžvalga

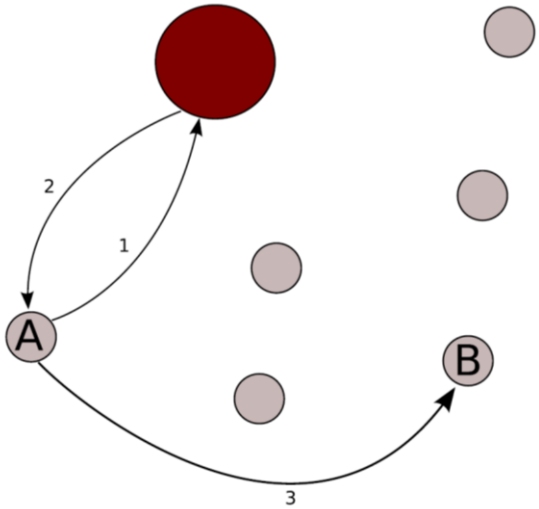
1.1.1. Napster

Napster– viena pirmųjų populiariausių savo laiku P2P sistemų. Dvi pagrindinės Napster sistemos problemos, dėl kurių ji vėliau buvo priverstinai sustabdyta, yra centrinis indeksavimo serveris ir klausimai susiję su autorinių teisių pažeidinėjimu. Sistemos veikimo principas tik **iš dalies** pagrįstas P2P architektūra. Principai, kurie buvo panaudoti sistemoje:

1. informacija apie visus vartotojų turimus failus, yra saugoma viename arba keliuose serveriuose (kliento serverio architektūra);
2. failų siuntimas tarp mazgų vyksta tiesiogiai – be serverio pagalbos (P2P principas).

Vienas pagrindinių Napster trūkumų buvo serveris. Kuo daugiau kompiuterių prisijungia prie tinklo, tuo daugiau serveris yra apkraunamas. Didesnis serverių kiekio naudojimas tik iš dalies išsprendžia problemą, kadangi vartotojų sistemoje gali būti daug ir jų skaičius gali labai svyruoti. Sutrikus vienam iš serverių, didelė tinklo dalis prarandama.

Pradinės Napster sistemos veikla buvo nutraukta. Daugelis sistemos vartotojų sistemą naudojo muzikos įrašų mainams. Nors sistemos autoriai teigė, kad sistemos tikslas nėra „neteisėtų intelektinės nuosavybės mainų“ skatinimas, tačiau teisme jiems nepavyko įrodyti fakto, kad dauguma vartotojų sistemą naudoja būtent tam. Sistemos veikimas sustabdytas uždraudus naudoti Napster paslaugą palaikančius serverius.

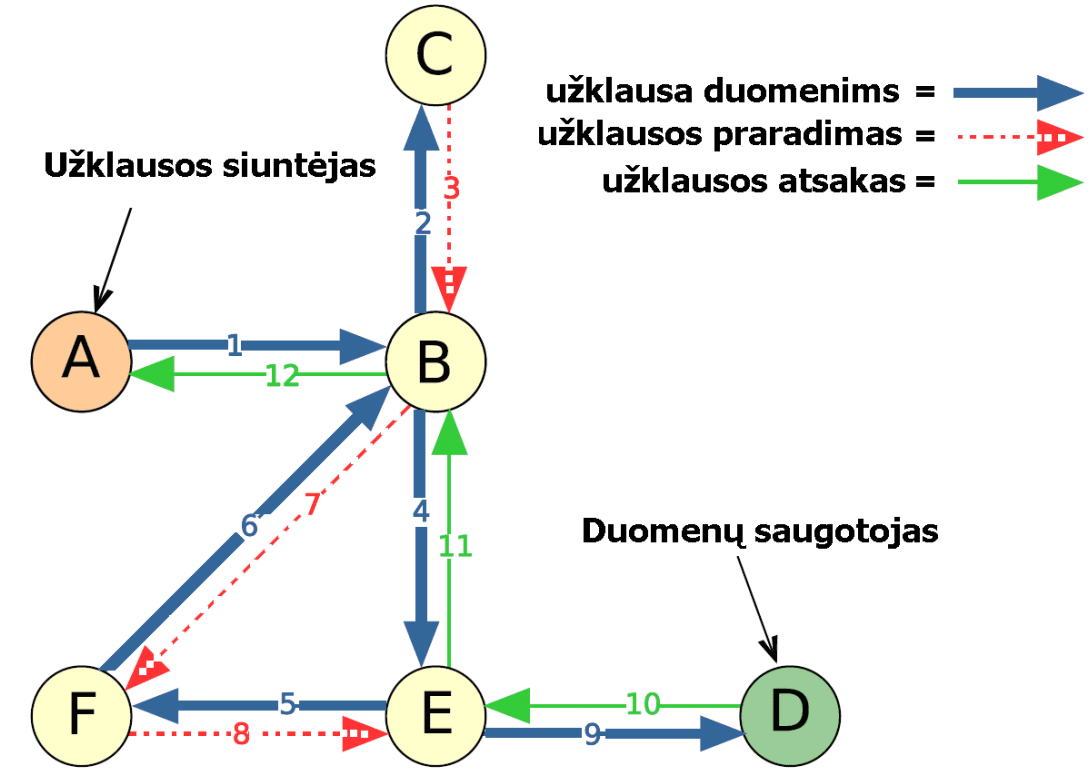


1 pav. Napster principas: (1) klientas A pateikia užklausą serveriui apie klientą, kuris turi jam reikalingą failą; (2) serveris atsako į A užklausą, suteikdamas jam B kliento IP adresą; (3) klientas A tiesiogiai užklausia B apie turimą failą

1.1.2. Freenet

„...tik su visišku anonimiškumu galima užtikrinti žodžio laisvę, ... Freenet sistemos privalumai užgožia šios sistemos naudojimo neigiamas pasekmes [4] *“*

Freenet P2P sistemoje kiekvienas klientas (mazgas) „žino“ tik apie dalį kitų mazgų–kaimynų. Taip pat kiekvienas mazgas turi dinamiškai sudarytas maršrutų lenteles (angl. Routing Tables). Kiekvienas sąrašo elementas šioje lentelėje yra sudaromas, naudojantis sėkmingais bandymais, t.y. kiekvienas elementas mazgo **n** lentelėje turi nuorodas į neseniai matytą duomenų struktūrą X ir mazgą, iš kurio buvo gauta užklausa į **n**. Paieškos užklausos siunčiamos kaimynams atsitiktine tvarka [2. pav]. Jeigu duomenų struktūra X yra randama, ji yra siunčiama atgal tuo pačiu tarpinių mazgų keliu, kuriuo vyko užklausa. Šio proceso metu visi mazgai gali atnaujinti savo nuorodas apie X duomenų struktūros sėkmingą paiešką (t.y. suteikti didesnę tikimybę, kad duomenų struktūrą galima rasti jei užklausa bus siunčiama į mazgą **m**) arba išsisaugoti X kopiją, kad kitą kartą užklausos kelias sutrumpėtų.



2 pav. Atsitiktinis mazgų parinkimo principas FreeNet tinkle užklausų siuntimo metu

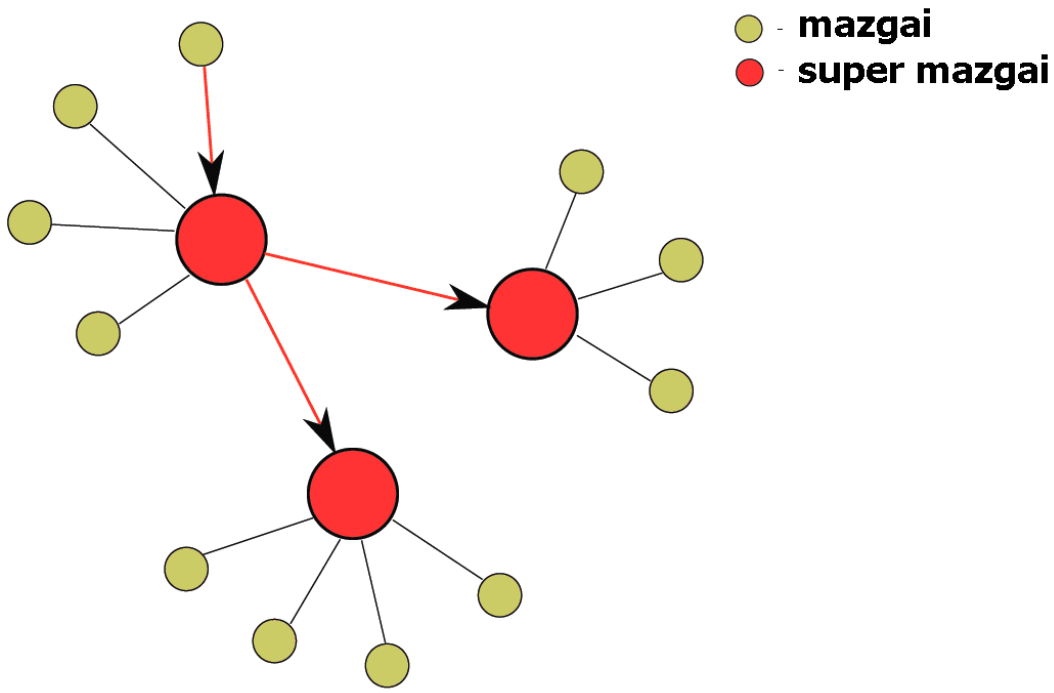
[4] „ ...only with true anonymity comes true freedom of speech, and ... the beneficial uses of Freenet outweigh its negative uses“. [7]

Tokiame tinkle žinučių, reikalingų užklausai, kiekis yra didelis. Tai vadinama srautų schema (angl. flooding sheme), kai vienu metu iš vieno mazgo visiems kaimynams yra siunčiama ta pati užklausa. Kiti mazgai savo ruoštu pasirenka, kuriems mazgams siųsti užklausą toliau. Apribojimu tokioje schemoje yra parenkama konstanta – maksimalus žinutės keliavimo iš vieno mazgo į kitą kelias. Jei žinutė neranda objekto X ir viršija šią konstantą, ji nebesiunčiama toliau kitam mazgui, o sunaikinama.

Tokiu principu vykstant užklausoms, pasitaiko pranešimo kelionės ciklų, kurie sukuria perteklinius pranešimus. Pranešimų kelias yra funkcija, kurios sudėtingumas – O(**z**) (kur z – maksimalus žinutės kelias nuo pradinio iki galinio mazgo). Tačiau pasirinkus per mažą konstantos z reikšmę yra rizikuojama, kad paieška veiks labai neefektyviai, o pasirinkus z per didelį, rizikuojama, kad sistema vienos užklausos metu gali išnaudoti per daug tinklo resursų. Įvykus kelioms lygiagrečioms užklausoms rizikuojama, kad tinklas bus ne užtvindytas, o „paskandintas“ pranešimų srautuose [1].

1.1.3. Gnutella protokolas

Tinklo „paskandinimo“ pranešimais išvengimui Gnutella protokole pristatyta „super mazgų“ (angl. super nodes) architektūra. Super mazgai yra sujungti tarpusavyje ir sudaro aibę mažų potinklių [3]. Kiekviena užklausa iš „paprasto mazgo“ yra organizuojama hierarchiškai, t.y. paprastas mazgas visada iš pradžių užklausia super mazgą. Tuo tarpu super mazgas gali apklausti visus arba dalį prie jo prisijungusių paprastų mazgų arba nusiųsti užklausą į dalį jam žinomų super mazgų – kaimynų. Toks principas pavaizduotas 3-čiajame paveiksle.



3 pav. Gnutella protokolo super mazgų architektūra [10]

Nustojus veikti super mazgui, gali nustoti veikti didelė tinklo dalis. Šiai problemai spręsti Gnutella protokolu paremtuose P2P tinkluose yra numatyta ne tik tvarka, kada „išrinkti“ mazgą į super mazgų aibę, bet ir kada pašalinti mazgą iš super mazgo, taip pat kaip aptikti, kad super mazgas nebepasiekiamas, arba kaip aptikti, kad super mazgas buvo tik laikinai nepasiekiamas.

1.2. DHT algoritmų palyginimas

DHT (angl. Ditributed Hash Table) – paskirstytos maišos lentelės, tai P2P sistemos, kurios turi maišos lentelėms būdingą funkcionalumą. DHT išsiskiria tuo, kad tinklas nereikalauja, nei severio, nei super-mazgų. Jis decentralizuotas (anlg. decentralized). Kiekvienas tokio tinklo mazgas turi identišką funkcionalumą (gali skirtis tik saugumo parametrų nustatymai). DHT tinklai pasižymi plečiamumu (angl. Scalability), tolerancija sutrikimams. Dėl šių priežasčių tokia tinklų architektūra dažnai pasirenkama jungiant tinklus iš daugiau nei 10^6 kompiuterių.

Apibrėžtis.

DHT (angl. Distributed Hash Table) algoritmais paremtos sistemos − tai klasė decentralizuotų išskirstytųjų sistemų, kurios teikia maišos lentelėms (angl. Hash Tables) būdingą funkcionalumą[5] *“.*

DHT sistemose kiekviename iš mazgų yra saugomos poros **raktas-reikšmė. K**iekvienas mazgas gali išrinkti **reikšmes,** susietas su duotu **raktu**. Atsakomybė už teisingą **raktų** susiejimą (angl. mapping) su **reikšmėmis** paskirstyta tarp mazgų tokiu būdu, kad pasikeitimai esamų mazgų aibėje sukelia mažiausią kiekį klaidų ir nepageidaujamo duomenų praradimo. Tai ir užtikrina DHT sistemų plečiamumą su labai dinamišku vartotojų skaičiaus kitimu. Yra daug algoritmų, palaikančių DHT funkcionalumą, iš kurių pagrindiniai yra šie:

- Pastry
- Cademlia
- Tapestry
- CAN (Content Addressable Network)
- Chord
- Koorde

Pastry algoritmo idėja yra ta, kad kiekvienas mazgas tinkle turi unikalų numerį– **nodeId**. Kai duota žinutė ir raktas, kiekvienas mazgas efektyviai perduota žinutę į arčiau esantį mazgą, kurio nodeId yra artimesnis duotam raktui. Kiekvienas Pastry mazgas stebi kelis savo kaimynus (mazgus, kurių **nodeId** yra artimiausias jo paties **nodeId**), ir praneša sistemai apie naujų mazgų prisijungimą, mazgų avarijas ir atsistatymus. Sistemos, pagrįstos šiuo algoritmu, yra visiškai decentralizuotos, pasižymi dideliu plečiamumu ir yra save organizuojančios – jos automatiškai prisitaiko prie naujų mazgų prisijungimo/atsijungimo ir neplanuoto mazgų „dingimo“ [9].

CAN algoritmu paremtos sistemos remiasi d-dimensijų virtualia Dekarto koordinačių sistema (angl. Cartesian Coordinate System). Ši koordinačių sistema yra pagrįsta ir logiška, tačiau neturi jokių sąsajų su fizine koordinačių sistema [11]. Kaip ir visose DHT sistemose, CAN naudoja šią virtualią d-dimensijų koordinačių sistemą porų **raktas-reikšmė** saugojimui. **Raktas** yra vienareikšmiškai atvaizduojamas į tašką **P**, kuris priklauso šiai koordinačių sistemai. Koordinačių sistema yra suskirstyta sritimis [4 pav.]. Kiekvienam prisijungusiam prie sistemos mazgui yra priskiriama sritis, už kurią jis atsakingas. CAN algoritmas taip pat užtikrina, kad mazgui **M** sugedus arba paliekant tinklą mazgo kaimynai pasirūpina sritimi, už kurią buvo atsakingas **M**. Jei sistemoje yra „daug“ mazgų, tai vieno mazgo gedimas veikia tik jo

[5] G.Couloris ir kiti [1]

kaimyninius mazgus. Žinučių siuntimas ir užklausimas vadovaujasi „godžiu“ persiuntimu pirmyn (angl. Greedy Forwarding). Gavęs žinutę su raktu **k**, mazgas persiunčia žinutę savo kaimynui, kurio turima sritis koordinačių sistemoje yra artimesnė **k** reikšmei. Šiuo atveju vidutinis pranešimo kelio ilgis nuo siuntėjo iki gavėjo yra:

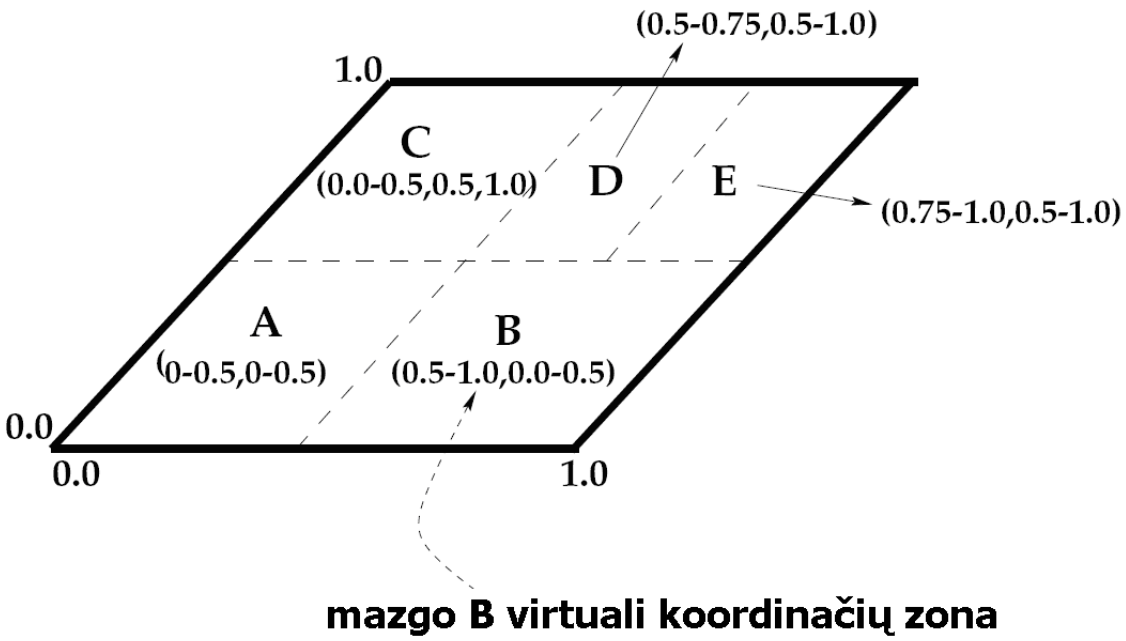
$$\frac{d}{4}\left(n^{\frac{1}{d}}\right)$$

Čia:

d – koordinačių sistemos dimensijų skaičius;

n – sistemos aktyvių mazgų skaičius.

Iš čia matome, kad pranešimų siuntimo kelias yra funkcija, kurios sudėtingumas $O\left(n^{\frac{1}{d}}\right)$.



4 pav. 2 dimensijų CAN koordinačių erdvė paskirstyta 5 mazgams. Koordinačių sistemoje kiekvienas raktas k patenka į kurią nors sričių, už kurią atsakingas vienas iš sistemoje esančių mazgų

Chord algoritmu paremtos sistemos savo veikimo principu yra panašus tiek į CAN, tiek ir į Pastry sistemas. Algoritmo pagalba padidinamas sistemos plečiamumas išvengiant reikalavimo, kad kiekvienas sistemos mazgas žinotų apie kitą sistemos mazgą. Kiekvienam mazgui reikia tik mažos maršrutų duomenų dalies. N dydžio tinkle kiekvienam mazgui tereikia $O(\log N)$ kitų

mazgų adresų. Susisiekimas su bet kuriuo tinklo mazgu reikalauja daugiausiai $O(\log N)$ pranešimų [5].

DHT sistemose apsieinama be super mazgų. DHT algoritmai suteikia tinklui labai didelį plečiamumą. Tokiose sistemose be papildomos kompiuterinės ir programinės įrangos sėkmingai veikia tinklai, sudaryti iš milijonų aktyvių vartotojų. Jų struktūra leidžia mazgų bendravimui siųsti mažiau pranešimų, o paieška tokiose sistemose yra taiklesnė (angl. *Higher Hit Guarantee*), palyginus su srautų schemomis. Struktūrizuotose P2P sistemose (pvz., *Pastry, Tapastry, CAN* arba *Chord*) kiekvienam iš sistemoje dalyvaujančių mazgų yra paskiriama atsakomybė už tam tikrą dalį sistemos resursų. Tai atliekama suskirstant visą resursų (raktų arba adresų) aibę į intervalus. Kiekvienam mazgui priskiriamas tam tikrą intervalas. Kai mazgas nori surasti resursą, pavyzdžiui, Chord algotitme, žinant tik raktą, įmanoma aptikti resursą per $O(\log N)$ pranešimų skaičių. Tam tereikia saugoti $O(\log N)$ kaimynų adresų kiekviename iš mazgų [1 lentelė]. Pavyzdžiui, *Chord* algoritmu grįstose DHT sistemose, kai maksimalus tinklo dydis yra $1024 = 2^{10}$, DHT leidžia iš mazgo **A** nusiųsti pranešimą į bet kuri mazgą **Z** per daugiausiai $\log_2 1024 = 10$ tarpinių žinučių. Mazgui **A** tereikia saugoti 10 savo kaimynų adresų. Detalesnis algoritmo principas aprašytas skyriuje 2.1. .

1 lentelė. Pranešimų tarp mazgų skaičius (angl. hops between nodes) ir maršruto lentelės dydis įvairiuose DHT algoritmuose

Algoritmas	**Paieškos ilgis (šuolių tarp mazgų skaičius)**	**Maksimalus maršruto lentelės dydis kiekviename mazge**	**Paaiškinimai**
Chord	$\log_2(N)$	$\log_2(N)$	*N – maksimalus mazgų skaičius (sistemos dydis)*
Tapestry	$\log_b(N)$	$b\log_b(N)$	*b – ieškojimo srities dekodavimo pagrindas*
CAN	$\frac{d}{4}n^{1/d}$	$2d$	*d – CAN tinklo koordinačių sistemos dimensijų skaičius*

1.3. DHT ir srautų schemų palyginimas

P2P sistemos suskirstomos į dvi pagrindines grupes:

1. srautų schemos (dar vadinamos Gnutella protokolu)
2. DHT algoritmais paremtos sistemos

Lyginant su srautų schemomis, pagrindinis DHT trūkumas yra, tas kad jos nepalaiko laisvų užklausų (angl. arbitrary quering)– norėdamas rasti resursą, vartotojas privalo žinoti jo raktą. Originaliose DHT sistemose nėra galimybės paieškai pagal raktažodžius (Gnutella protokole tai vienas iš pagrindinių paieškos metodų), kadangi neefektyvu saugoti raktu kiekvienai užklausos išraiškai.

DHT sistemose visi duomenų objektai susiejami (angl. mapping) raktu. Tai maišos lentelės principas: duomenų turinio raktas maišos algoritmo pagalba susiejamas su turiniu. Šis raktas yra priskiriamas už jį atsakingam tinklo mazgui. Nežinant rakto, nežinoma, kuriam iš mazgų siųsti užklausą ir rasti mus dominantį objektą.

2 lentelė. Srautų schemų ir DHT algoritmu paremtų P2P sistemų savybių palyginimas

Savybės	Srautų schemos	DHT
Užklausos	pasirenkamosios	raktu grįstos užklausos
Užklausomis generuojamas pranešimų srautas	O(N) (čia N - sistemos dydis)	O(log(N))
Užklausos taiklumas (anlg. Hit guarantee)	žemas	aukštas
Susijungimo grafas	atsitiktinis	struktūrizuotas

1.4. Paieškos DHT tinkluose tyrimo uždavinio formuluotė

Apibrėžtis. Duotas duomenų objektas X (gali būti failas, sakinys, simbolis ir pan.), kuris saugomas dinamiškoje mazgų aibėje. Kaip rasti X?

Tarkime, X yra failo pavadinimas– „penketas.bmp“. Srautų schemose paieška atliekama taip: užklausos iniciatorius M išsiunčia pranešimą kaimynams (arba super-mazgui). Kaimynai, naudodami godaus siuntimo pirmyn schemą, (angl. Greedy Forwarding) persiunčia šį pranešimą savo kaimynams. Tinklas užtvindomas pranešimais. Tie mazgai, kurie turi failą arba duomenų struktūrą „penketas.bmp“, atsako į užklausą užklausos pradininkui [3].

1.4.1. Kaip problemą sprendžia kiti

Vienas iš problemos sprendimo būdų egzistuojančiose DHT sistemose pasinaudojus išoriniais paieškos mechanizmais, tokiais kaip http arba ftp. Duomenų objektas X yra susiejamas raktu R. Raktas R patalpinamas į išorinę sistemą, iš kur gali būti lengvai pasiekiamas DHT tinklo

vartotojui. Tik sužinojęs raktą (naudodamasis papildomais parametrus, tokius kaip: objekto X tipas, dydis, paskirtis, pavadinimas ir pan.) vartotojas gali suformuluoti užklausą šiam objektui rasti. Pavyzdžiui, Bit-torrent (populiari DHT algoritmais paremta sistema) visus raktus spausdina daugybėje išorinių serverių. Tokiu atveju pats DHT tinklas yra decentralizuotas, bet paieškos mechanizmas yra centralizuotas (nors ir egzistuoja daug atskirų serverių, kurie padeda surasti norimą raktą).

Kitas, rečiau sutinkamas problemos sprendimo būdas – panaudoti tinklo užtvindymo mechanizmą. Kaip ir srautų schemomis paremti tinklai, DHT tinklas dalinai arba pilnai užtvindomas pranešimais. Kadangi DHT yra struktūrinis tinklas, užtenka N pranešimų (N mazgų turinčiame tinkle), kad visi mazgai gautų inicijuotą pranešimą. Tuo tarpu srautų schemose dėl ciklų susidarymo papildomai gali būti sukurta iki 95% perteklinių pranešimų. 5-jame paveiksle pavaizduota, kiek pranešimų reikia viso tinklo užtvindymui srautų schemomis paremtose ir DHT sistemose [13].

5 pav. Pranešimų skaičius reikalingas užtvindymo mechanizmui įgyvendinti: Gnutella protokole (a); DHT sistemose (b)

1.4.2. Kodėl šių sprendimų neužtenka arba jie nepakankami?

Išorinis paieškos metodas yra patogus, tačiau jis nėra pakankamas. Nėra įgyvendintų paieškos sistemų, kuriose galima būtų rasti visus esamus DHT raktus. Tai sumažina paieškos lankstumą, nes neradus reikiamo objekto rakto vienoje sistemoje, mes dar nesame tikri kad tinkle

tikrai to objekto nėra, todėl esame priversti ieškoti rakto kitoje sistemoje, nors jei ir paieškos kriterijai visiškai nesiskirai nuo pirmojo bandymo kriterijų.

Jau rašant šį darbą, buvo uždrausta naudoti TorrentSpy.com svetainė – viena iš populiariausių išorinių DHT paieškos sistemų. Jei JAV vyriausybė ir toliau sėkmingai uždarinės tokias sistemas, greitai DHT tinklų kūrėjai privalės pereiti prie vidinių paieškos mechanizmų.

Analizės metu pastebėti užtvindymo pranešimais mechanizmo privalumai:

- Užklausų lankstumas (užklausų tipų gali būti labai daug);
- Lyginant su srautų schemų užtvindymo mechanizmais, DHT užtvindyme nėra žinučių pertekliškumo (nes nesusidaro komunikavimo ciklai). Maksimalus žinučių kiekis norint pranešti visam tinklui kažkokį pranešimą yra lygus N (N – aktyvių mazgų kiekis tinkle)[13];
- Metodo efektyvumas padidėja, jeigu sistemoje įgyvendintas duomenų kopijų palaikymas. Tokiu atveju, padidėja tikimybė rasti reikiamą rezultatą neužtvindant viso tinklo pranešimais[14].

Užtvindymo pranešimais mechanizmo įgyvendinimas DHT tinkle turi tokius trūkumus:

- mažėja užklausų taiklumas (tik tuo atveju, kai pasirenkamas dalinis tinklo užtvindymas);
- jei vartotojų tinkle daug, pranešimų srautas gali būti labai didelis ir neįgyvendinamas;
- sunkiau užtikrinti tinklo saugumą (gali atsirasti vartotojų, kurie naudojasi tuo ketindami pakenkti tinklo stabilumui).

N-gramų metodo privalumai:

- Paieškos žodyje (kriterijuje pagal kurį ieškoma objekto) esančios gramatinės klaidos mažai įtakoja rezultatų taiklumą. Pavyzdžiui padarius klaidą paieškos žodyje „*little nice song*“ parašius tik su viena t: „*litle nice song*“ pastebėsime, kad gramatinė klaida įtakos tik tas n-gramas, į kurias įeina praleistoji *t* raidė.
- Paieškų taiklumas yra didesnis, nei užtvindymo schemų paieška, kadangi užklausos apie failų n-gramas siunčiamos tiesiai į tuos mazgus, kurie atsakingi už raktus, sudarytus iš n-gramų.

N-gramų metodo trūkumai pastebėti analizės metu:

- Netolygus raktų pasiskirstymas dideliuose tinkluose Pvz. skirtingų 3-gramų variantų skaičius $26 \cdot 26 \cdot 26 = 17576$ (26 – raidės yra angliškoje UTF formato

abėcėlėje). Jei mazgų tinkle yra 20000 tuomet vienam mazgui tenka mažiau nei po vieną rakto variantą. Todėl atsiranda mazgų, kurie neturės priskirtų raktų.

- Maišos algoritmo kolizijos problema. Kartais du žodžiai, kurių reikšmės skiriasi gali būti transformuojami į du vienodus raktus.
- Naudojant n-gramas dideliuose tinkluose (tarkim, dagiau nei 50000 vartotojų) labai padidėja raktų skaičius saugomas kiekviename faile. Todėl n-gramų naudojimas mažina DHT sistemų plečiamumą.

1.4.3. Darbo tikslas ir uždaviniai

Darbo tikslas – padidinti paieškos DHT sistemose galimybes, sukuriant vidinį paieškos mechanizmą Chord algoritmo pagrindu veikiančiai DHT sistemai ir ištiriant jo efektyvumą

Uždaviniai:

- sukurti eksperimentinę Chord algoritmo pagrindu veikiančią DHT sistemą;
- pritaikyti šiai sistemai užtvindymo pranešimais mechanizmą;
- pritaikyti šiai sistemai n-gramų metodą;
- eksperimentiškai ištirti bent vieno iš paieškos metodų veikimą;
- įvertinti abu sprendimus ir sudaryti rekomendacijas paieškos algoritmams DHT sistemose realizuoti.

Chord algoritmo veikimu paremtai DHT sistemai realizuoti pasirinkta Java kalba, mazgų tarpusavio bendravimui naudojant Java-RMI technologiją. Sistemai sukurti nebus naudojama jokia iš jau įgyvendintų ir viešai platinamų *Chord* algoritmo versijų. Detalus algoritmo veikimo principas su stabilizacijos proceso patobulinimu aprašytas I.Stoca ir R.Moris straipsnyje [2].

1.5. Analizės išvados

1. P2P sistemų analizė rodo dideles šių sistemų perspektyvas, kadangi jos leidžia be papildomų sąnaudų vartotojų bendruomenėms kurti ir naudoti bendrus informacinius resursus. Tokios sistemos gali būti pritaikomos mokymui, moksliniams tyrimams, socialiniams tinklams ir kt.

2. Pagrindiniai P2P sistemų tipai yra srautų ir DHT sistemos, iš kurių DHT yra efektyvesnės, kadangi jos nereikalauja centrinio serverio ir turi didesnes plečiamumo galimybes.

3. Tačiau paieška DHT sistemose nėra lanksti. Esami sprendimai reikalauja išorinio paieškos serverio, o tai mažina DHT sistemų nepriklausomumą, todėl tikslinga sukurti vidinius paieškos mechanizmus.

4. Remiantis literatūros analize, rasti du galimi vidinės paieškos algoritmai: užtvindymas pranešimais ir n-gramų metodas. Kadangi šie algoritmai aprašyti tik teoriškai, tolesnio tyrimo uždavinys – realizuoti juos ir ištirti jų tinkamumą paieškai DHT tinkluose.

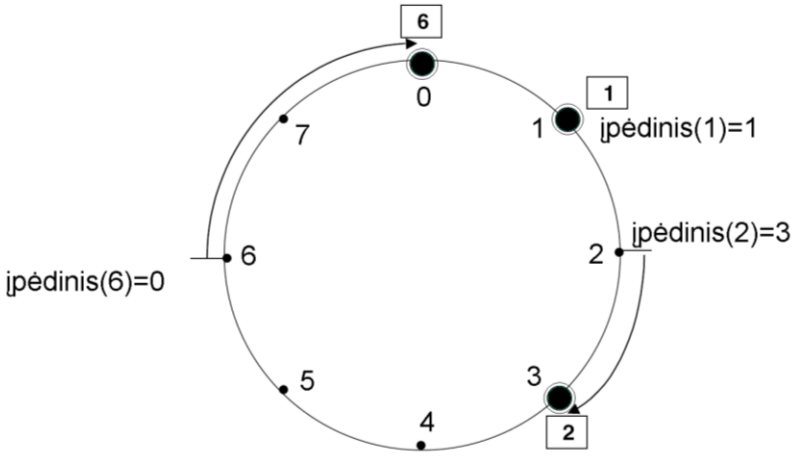
5. Paieškos algoritmų tyrimui eksperimentinė DHT sistemą bus kuriama Chord algoritmo pagrindu, kadangi, sprendžiant iš literatūros analizės, tai vienas perspektyviausių DHT tinklų realizavimo metodų.

2. Patobulintos paieškos DHT sistemose algoritmai

2.1. Chord algoritmas

Nors aiškaus lyderio darp DHT algoritmų dar nėra, tačiau Pastryt, Chord ir CAN yra vienu populiariausių ir perspektyviausių DHT algoritmų. Be to Chord algoritmas pasirinktas dėl jo paprastumo. Lyginant su kitais algoritmais jis greičiau išmokstamas, intuityviau suprantamas. Jo charakteristikos nenusileidžia kitų DHT algoritmų savybėms. Algoritmo pasirinkimui įtakos turėjo testavimo etapas. Stengėmės pasirinkti tokį DHT algoritmą, kurį testuoti būtų paprasčiau ir greičiau.

Kiekvienas mazgas šiame algoritme atsakingas už tam tikrą visos raktų aibės dalį. Visa raktų aibė yra išskirstyta apskritimu [6 pav.]. Kiekvienas iš mazgų turi unikalų numerį *nodeId*. Mazgas m vadinamas rakto r įpėdiniu (anlg. *Successor*), jeigu $r = m$ arba m yra pirmasis pagal laikrodžio rodyklę mazgas raktų apskritime (tuo atveju $r < m$). 1 paveiksle pavaizduoti trys tinklo mazgai su nodeId**:** 0, 1, 3. Mazgas 0 yra raktų 4, 5, 6, 7 ir 0 įpėdinis, mazgas 1 yra rakto 1 įpėdinis, o mazgas 3 yra raktų 2 ir 3 įpėdinis.



6 pav. Chord raktų aibės apskritimas susidedantis iš mazgų, kurių identifikatoriai yra 0, 1, 3; raktas 1 priskirtas mazgui 1, raktas 2 – mazgui 3, raktas 6 – mazgui 0.

Tam, kad *Chord* algoritmu paremtas tinklas veiktų teisingai, užtenka, kad kiekvienas iš mazgų turėtų savo pirmojo kaimyno pagal laikrodžio rodyklę adresą. Algoritmo veikimui to pakanka, tačiau nėra efektyvu, nes pranešimas, siunčiamas iš mazgo 1 į mazgą 0 [6 pav.], turi apkeliauti visą apskritimą. Todėl kiekvienas tinkle esantis mazgas turi lentelę iš $m = \log_2 N$

įrašų (čia N – maksimalus tinklo mazgų skaičius). Mazgo A i-tajame maršrutų lentelės įraše saugomas adresas mazgo, kuris adresų apskritime yra nutolęs mažiausiai per 2^{i-1} adresų. Iš 1 paveikslo matome, kad 0-nio mazgo pirmoje maršruto lentelės pozicijoje bus $2^{i-1} = 2^{1-1} = 1 \rightarrow 1$-mojo mazgo adresas, antroje: $2^{i-1} = 2^{2-1} = 2 \rightarrow 3$-čiojo mazgo adresas, trečioje $2^{i-1} = 2^{3-1} = 4 \rightarrow 0$-nio mazgo adresas [2].

2.1.1. Pagrindinis funkcionalumas, kurį suteikia DHT

Chordo algoritmas kaip ir visi DHT algoritmai įgalina tokias pagrindines funkcionalumo procedūras:

- **put(key, value)** – išsaugo duotą raktą **key** ir bei reikšmę **value** tinkle;
- **value[] get(key)** – kviečiant šį metodą, gaunamos visos tinkle saugomos **value** reikšmės;
- **remove(key, value)** – iš tinklo pašalinami ir raktas ir jo nešama reikmė.

remove() operacijoje kaip parametrai imama visa porą (key, value), nes negalima pašalinti tik raktą, arba tik prie jo prisietą reikšmę. Pašalinus tik raktą, lieka reikšmės **value**, kurių neįmanoma rasti. Pašalinus tik reikšmes **value**, lieka raktai kurie nesusieti su jokiomis reikšmėmis.

2.1.2. Chord algoritmo darbinės operacijos

Chord branduolyje yra trys operacijos nuo kuriomis vadovaujasi visas tinklas. Žymėjimas **n.methodName()** reiškia, kad metodas, kurio vardas **methodName()** yra vykdomas mazge **n**.

Visi trys metodai [7,8,9 pav.] yra vykdomi periodiškai ir jų tikslas rasti duoto rakto įpėdinį **successor(id)**. Metodas grąžina įpėdinį rakto **id**. Jį naudoja kiekvienas mazgas, kai tikrina ar jo įpėdinis nepasikeitė (ar tarp esamo mazgo ir jo senoj įpėdinio neįsiterpė kitas mazgas).

```
1  n.find_successor(id){
2    n'=find_predecessor(id);
3    return n'.follower;
4  }
```

7 pav. Pseudo kodas metodo ieškančio duoto rakto įpėdinio

Antrasis metodas tikrina ar raktas **id** patenka į tam tikrą intervalą Chord apskritime. Čia operacija **id.notInLeftOpenedRightClosedInterval(a, b)** grąžins **true**, jei $id \notin (a,b]$, kur a ir b yra mazgų adresai tinkle.

```
1  n.find_predecessor(id){
2    n' = n;
3    while (id.notInLeftOpenedRightClosedInterval(n', n'.follower) )
4        n' = n'.closest_preceding_finger(id);
5    return n';
6  }
```

8 pav. Pseudo kodas metodo ieškančio duoto rakto pirmtako (angl. predecessor)

Trečiasis metodas užbaigia pagrindinių Chord metodų seriją.

```
1  n.closest_preceding_finger(id){
2    for (int k = m–1 downto 0)
3        if (finger[k].node.inOpenedInterval (n, id))
4            return finger[k].node;
5  }
```

9 pav. Pseudo kodas metodo tikrinančio ar nėra pasenę įrašai apie esamus įrašus maršrutų lentelėje

Daugiau apie Chord atliekamas operacijas: [2].

2.2. Paieškos metodas, pritaikant DHT sąsajai užtvindymo mechanizmą

Vienas iš būdų užtikrinti laisvas užklausas ir išplėsti DHT objektų paieškos galimybes– įgyvendinti nestruktūrizuotos paieškos galimybes jau egzistuojančiame DHT tinkle. Išplatinant užklausą iš anksto pasirinktam mazgų skaičiui, galima atlikti tiek paiešką pagal raktą, tiek atliekant laisvas užklausas, taip sujungiant struktūrizuotų tinklų efektyvumą su nestruktūrizuotų P2P sistemų lankstumu. Tą galima atlikti dviem būdais: užtvindymo pranešimais mechanizmą arba naudojant n-gramomis pagrįstą metodą.

Užtvindymo pranešimais mechanizmo tikslas– sumažinti pranešimų srauto dydį, kurį generuoja mazgo sukurta užklausa, lyginant su srautų schemose generuojamu pranešimų srautu. Užklausos pradininkas pradeda resurso paiešką išsiųsdamas užklausą su mažu TTL gyvavimo laiko parametru (angl. TTL – *Time To Live*) keliems iš savo kaimynų. TTL apriboja užklausos

skverbimosi į tinklą gylį. Pradinė užklausa yra vadinama žvalgybine. Ji padeda apytiksliai apskaičiuoti ieškomo resurso populiarumą. Jeigu pradinio bandymo metu nesulaukiama norimo rezultato, užklausos pradininkas siunčia papildomą užklausą kitam savo kaimynų poaibiui su nauju TTL parametru. Naujas TTL parenkamas atsižvelgiant į pirmosios užklausos rezultatus ir norimą taiklių rezultatų skaičių. Procesas kartojamas tol, kol aptinkamas laukiamas resursų skaičius arba apklausiami visi kaimynai [14].

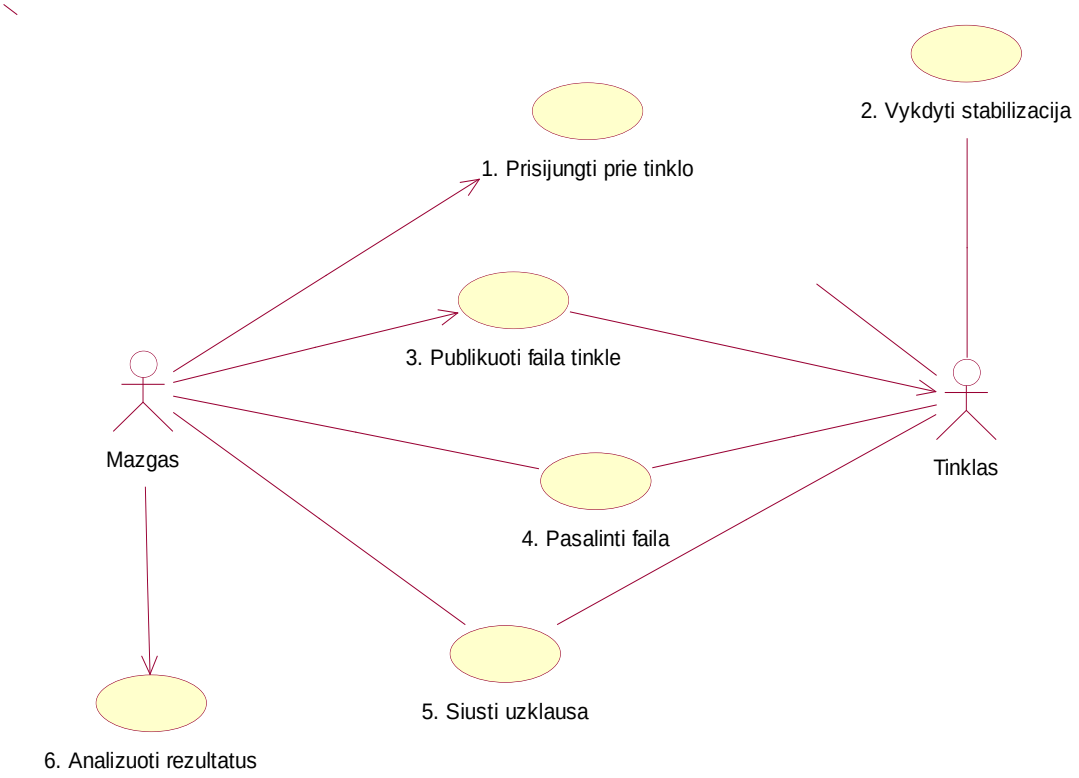
2.3. Paieškos metodas, pritaikant DHT sąsajai n-gramų algoritmą

Praplėsti paieškos galimybėms Chord tinkle pasirinkome n-gramų metodą [4]. N-gramos – tai tam tikros žodžio dalys, kurių ilgis yra n. Pavyzdžiui, visos žodžio *„kompiuteris“* 4–gramos yra: *komp, ompi, mpiu, piut, iute, uter, teri, eris*. Kiekvienam duomenų objektui tenka sudaryti ne vieną, o kelis raktus. Raktai gali būti sukuriami tarkim failo pavadinimą suskaidžius į n-gramas ir jas transformuojant maišos funkcijos pagalba. Taip kiekvienam failui sudaromos ne viena, o kelios poros **raktas-turinys**, kur raktas – failo pavadinimo n-grama, transformuota panaudojant rakto funkciją, o turinys– failo duomenų objektas (kuriame gali būti nurodyta visas failo vardas, tipas; vieta, kur tas failas galėtų būti ieškomas; dydis arba papildoma informacija).

Įgyvendinus n-gramų metodą paieškai atlikti, užklausos iniciatorius neprivalo apklausinėti savo kaimynų. Užklausos iniciatorius turi pateikti **užklausos sakinį** (tarkim, ieškomo failo pavadinimo fragmentą). Užklausos sakinys suskaidomas į pasirinkto ilgio n-gramas. Jos transformuojamos į raktą maišos algoritmo funkcijos pagalba. Gaunama daugybė raktų, kurie nurodo ieškomą failą arba failus, turinčius pavadinime tokias pat n-gramas. Šis algoritmas skiriasi nuo laisvųjų užklausų algoritmo tuo, jog užklausos siunčiamos tiems mazgams, kuriuose tikimasi rasti rezultatus, tuo tarpu laisvosios užklausos siunčiamos bet kuriems kaimynams tikintis, kad jie persiųs užklausą į savo kaimynus, kurie gal būt turės užklausos pradininką dominančių resursų.

3. Eksperimentinės P2P sistemos reikalavimai

3.1. Panaudos atvejų modelis



10 pav. Sukurtos sistemos, veikiančios Chord algoritmo pagrindu panaudos atvejų diagrama

3.2. Panaudos atvejų specifikacijos

1. PANAUDOS ATVEJIS: Prisijungti prie tinklo

Vartotojas/Aktorius: Mazgas

Aprašas: Mazgo prisijungimas prie jau tinklo. Jei tinklo dar nėra vienintelis mazgas norintis prisijungti prie tinklo, sukuria tą tinklą. Antrasis mazgas jungdamasis prie tinklo turėtų žinoti pirmojo mazgo adresą, kad galėtų prisijungti prie tinklo.

Prieš sąlyga: Tinklas turi būti jau sukurtas ir vykdyti periodinį stabilizavimo procesą.

Sužadinimo sąlyga: Mazgas pageidauja būti prijungtas prie tinklo.

Po-sąlyga: Mazgas prijungiamas prie DHT tinklo ir tampa jo dalimi (atnaujinami įpėdinio ir pasekėjo duomenys, atnaujinami maršrutų lentelės duomenys).

Komentaras: šiame darbe teigiu, kad DHT algoritmai yra visiškai decentralizuoti. Taip ir yra, tačiau visiškos decentralizacijos visuose procesuose pasiekti praktiškai labai sunku. Pagal originalų Chord algoritmą mazgas norintis prisijungti prie tinklo „turi žinoti“ bent vieno iš tinkle esančių mazgų adresų. Žiūrit iš jungimosi proceso perspektyvos, galima teigti, kad tas mazgas, kurio adresą turi žinoti mazgas ir yra serveris. Bet iš kitos pusės negalime pavadinti to mazgo, esančio tinkle ir padedančio mums prisijungti severio, kadangi prisijungimui įvykdyti, mums reikia bent vieno, **bet kurio** veikiančio tinkle mazgo adreso.

2. PANAUDOS ATVEJIS: Vykdyti stabilizaciją

Vartotojas/Aktorius: Kiekvienas iš tinkle esančių mazgų (visas tinklas).

Aprašas: Kiekvienas tinkle esančių mazgų periodiškai vykdo tinklo stabilizavimo procesus.

Prieš sąlyga: Tinklas turi būti jau sukurtas (tinkle turi būti bent vienas mazgas).

Sužadinimo sąlyga: Tinklo sukūrimas.

Po-sąlyga: – .

3. PANAUDOS ATVEJIS: Publikuoti failą tinkle

Vartotojas/Aktorius: Mazgas

Aprašas: Kiekvienas tinkle esančių mazgų gali publikuoti pasirinktu failus.

Prieš sąlyga: Mazgas turi būti prisijungęs prie tinklo.

Sužadinimo sąlyga: Pageidavimas skelbti duomenis apie naują failą.

Po-sąlyga: Failo duomenys ir raktas patalpinami tinkle.

4. PANAUDOS ATVEJIS: Pašalinti failą

Vartotojas/Aktorius: Mazgas

Aprašas: Mazgas, kuris laiko failą gali pašalinti raktą ir duomenis apie failą iš tinklo.

Prieš sąlyga: Mazgas turi būti prisijungęs prie tinklo ir laikyti failą (pašalinimo komanda gali ateiti iš tinklo).

Sužadinimo sąlyga: Pageidavimas pašalinti failą iš tinklo.

Po-sąlyga: Duomenys apie raktą pašalinami iš mazgų, failas pašalinamas iš jį saugančio mazgo.

5. PANAUDOS ATVEJIS: Siųsti užklausą

Vartotojas/Aktorius: Mazgas

Aprašas: Mazgas gali siuntinėti užklausas į tinklą.

Prieš sąlyga: Mazgas turi būti prisijungęs prie tinklo.

Sužadinimo sąlyga: Pageidavimas rasti dominantį failą.

Po-sąlyga: Išsiuntęs užklausą į tinklą mazgas laukia rezultatų. Rezultatus atsiunčia tinkle esantys mazgai, kurios pasiekė užklausa.

6. PANAUDOS ATVEJIS: Analizuoti gautus rezultatus

Vartotojas/Aktorius: Mazgas

Aprašas: Mazgas gautus rezultatus analizuoja ir lygina ar jie atitinka užklausą.

Prieš sąlyga: Mazgas turi būti gavęs rezultatus iš tinklo.

Sužadinimo sąlyga: Gauti rezultatai iš tinklo.

Po-sąlyga: Mazgas išveda rezultatus į failą.

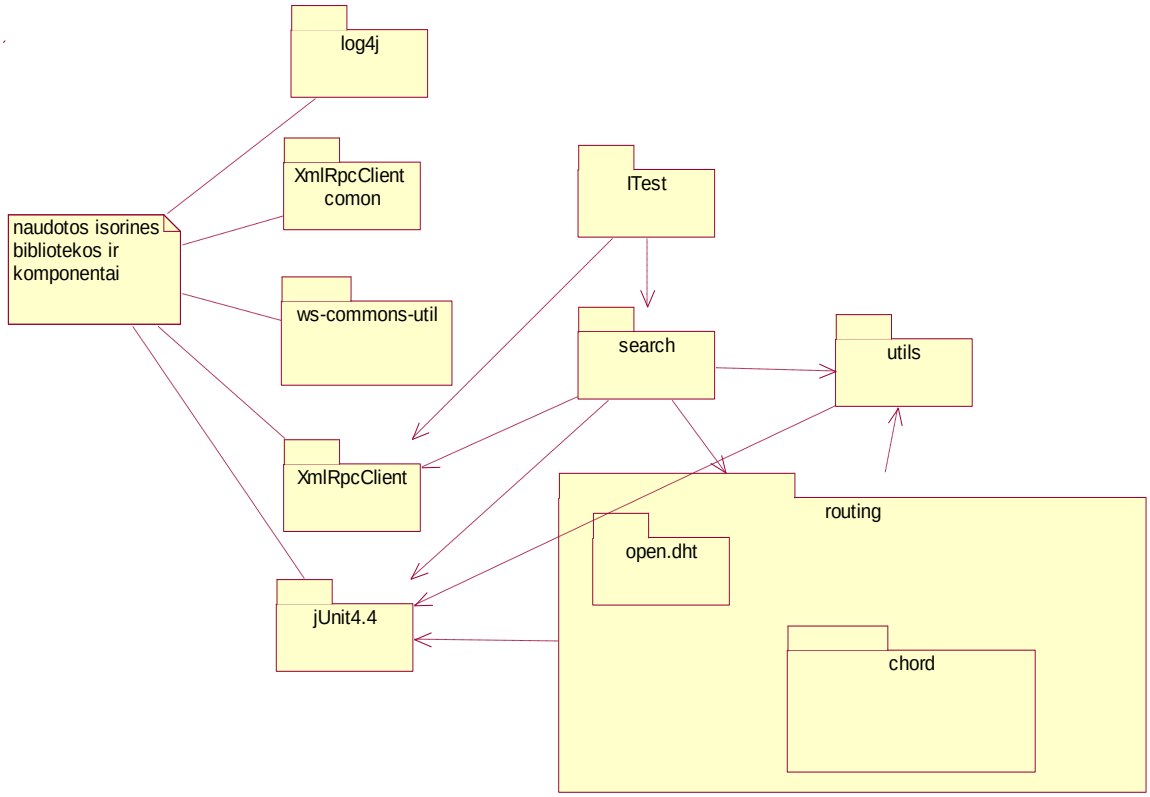
3.3. Nefunkciniai reikalavimai sistemai

Tokie nefunkciniai reikalavimai buvo suformuoti kuriamai programinei įrangai:

- Užklausų lankstumas;
- Sistemos plečiamumas;
- Tolerancija klaidoms;
- Decentralizacija;
- Minimalus žinučių srauto generavimas paieškos proceso metu.

4. Išskirstytos P2P sistemos projektas

4.1. Loginė sistemos architektūra



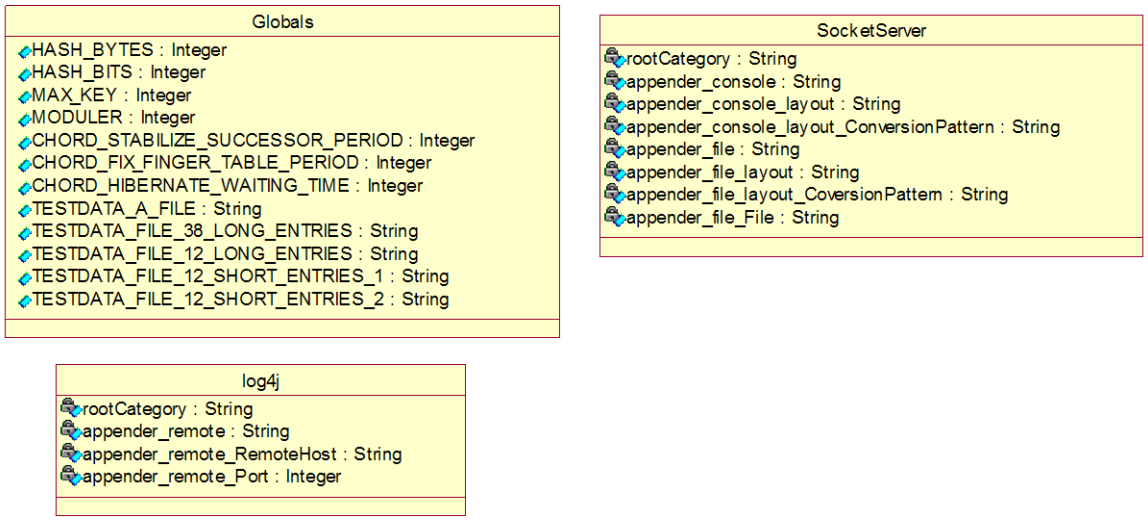
11 pav. Realizuotos programinės įrangos paketų diagrama

4.2. Klasių modelis

Globals– klasė sauganti pagrindinius Chord algoritmo parametrus ir kintamuosius, failų iš kurių imami duomenys pavadinimus.

SocketServer– konfigūracijų klasė SocketServeriu.

Konfigūracijų klasė log4j įvykių sekimo serveriui.



12 pav. Paketo „conf“ klasių diagrama

ITest – klasė inicijuojanti programos paleidimą.

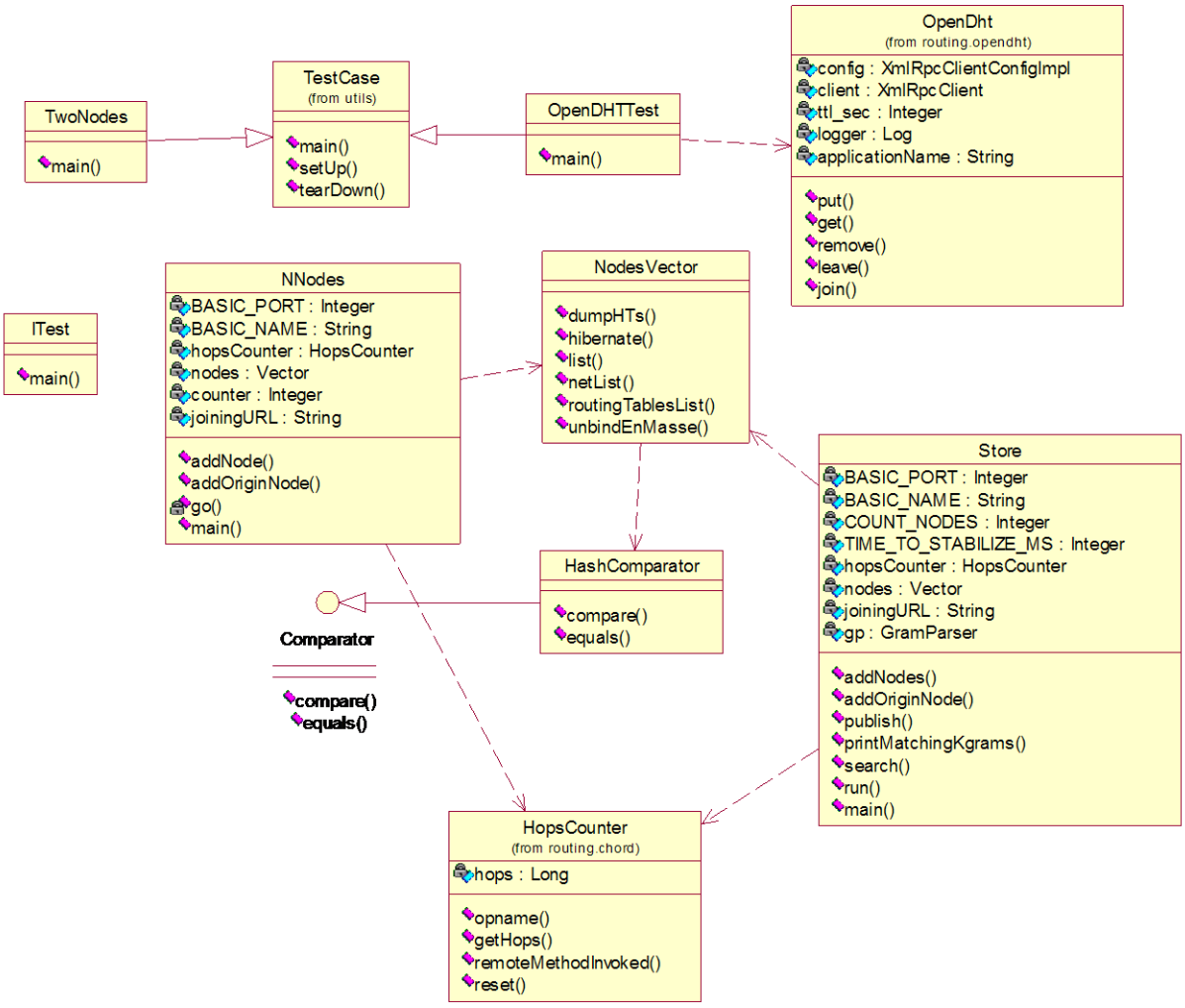
TestCase – klasė, kurią įgyvendina testavimo vienetų testavimo klasės.

NodesVector–Chord mazgų vektorius.

Store– saugo NodeesVector ir atlieka papildomus veiksmus su mazgais.

HospCounter– klasė, kuri skaičiuoja kiek bendrai buvo atlikta RMI šaukinių per visus sistemoje esančius mazgus.

TwoNodes– testavimo klasė, kuri tikrina tinklo iš dviejų mazgų paleidimą.



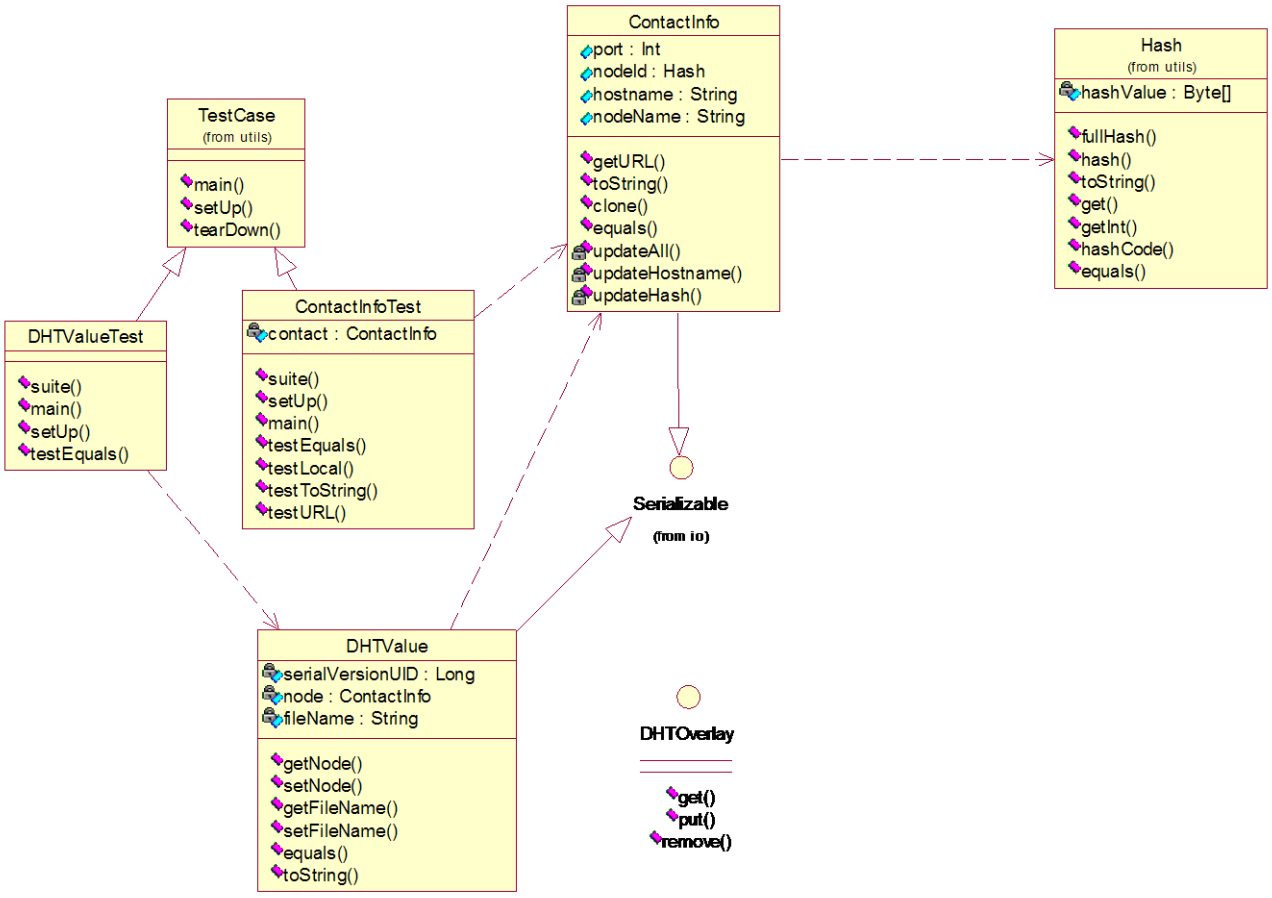
13 pav. paketo „ITest“ klasių diagrama

ContactInfo– klasė, kurioje saugomas kiekvieno mazgo vardas, prisijungimo adresas tinkle, maišos funkcijos reikšmė (angl. Hash).

Hash– klasė darbui su maišos funkcijomis (naudojama tiek mazgų, tiek duomenų maišos raktams sudaryti).

DHTValue– klasė kurioje saugomas pora: mazgo informacija, failo pavadinimas.

DHTOverlay– sąsaja, kurioje DHT sistemų kaprindinės operacijos (gauti, padėti, pašalinti).



14 pav. Paketo „routing“ klasių diagrama

ChorNode– Chord mazgų klasė. Įgyvendina sąsajas DHTOverlay, ChordNodeItf, Stabilizable.

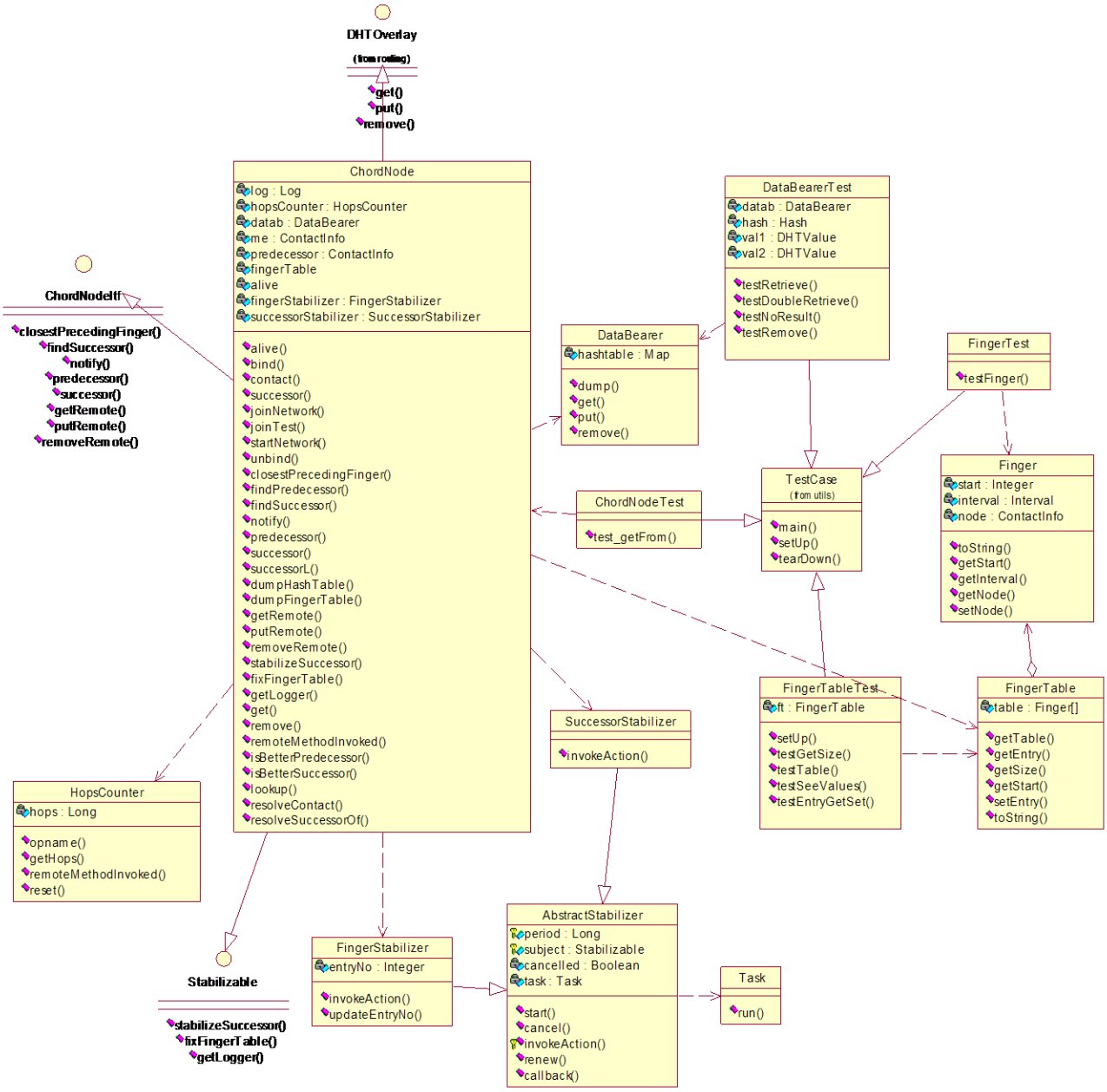
DataBearer– klasė sauganti reikšmes (raktas, turinys), kurias saugo duotasis chord mazgas.

FingerTable– maršrutų sudarymo lentelė. Joje saugomi adresai kaimynų, kuriuos „žino“ mazgas.

Finger– vienas maršrutų lentelės įrašas (duomenys mazgo kaimyno).

FingerStabilizer– klasė, kuri stabilizavimo proceso metu vykdo patikrinimą ar maršrutų lentelė nėra pasenusi (periodiškai tikrina, ar mazgas, kurio adresas žinomas vis dar prisijungęs prie sistemos).

SuccessorStabilizer– klasė, kuri stabilizavimo metu tikrina ar artimiausias kaimynas pagal laikrodžio rodyklę (pasiekėjęs) vis dar prisijungęs prie sistemos.



15 pav. Paketo „routing.chord“ klasių diagrama

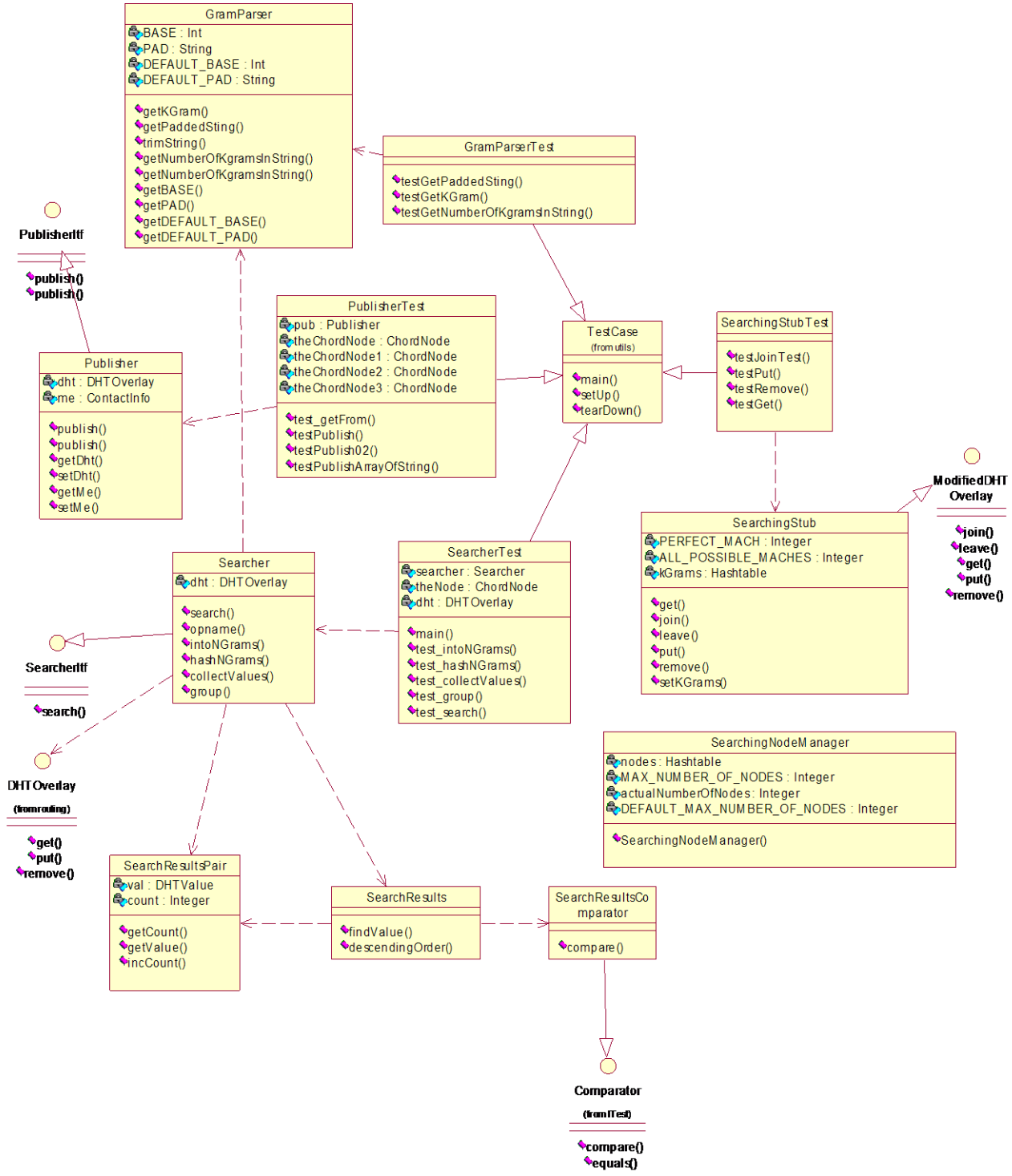
GramParser– n-gramų formavimo klasė.

Publisher– klasė, naudojama duomenų publikavimui tinkle.

Searcher– klasė, dalyvaujanti paieškos procesuose tinkle.

SearchingResultPair– paieškos rezultatas (n-gramų metodo).

SearchingNodeManager– klasė, kurią naudoja ieškantysis mazgas paieškai atlikti.



16 pav. Paketo „search“ klasių diagrama

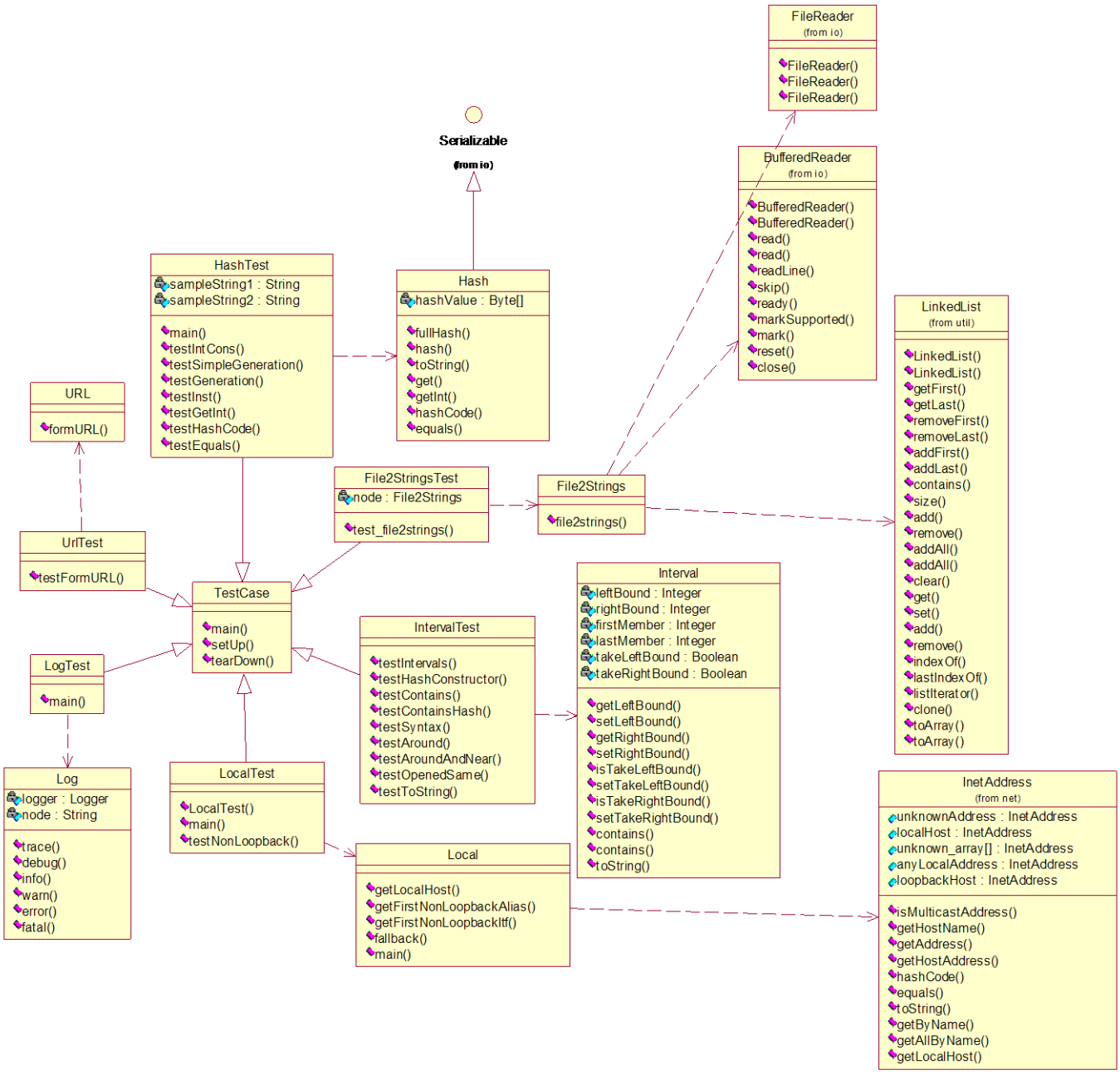
Hash– klasė operacijoms su maišos algoritmai atlikti.

File2String– klasė naudojama duomenims iš failo paversti String tipo objektais.

Local– klasė mazgų tarpusavio bendravimui (adresų radimui).

Inteval– klasė, kuri naudojam Chord algoritme (apsprendžia ar duota maišos funkcijos reikšmė įeina į intervalą, tarkim ar c įeina į intervalą [a,b]).

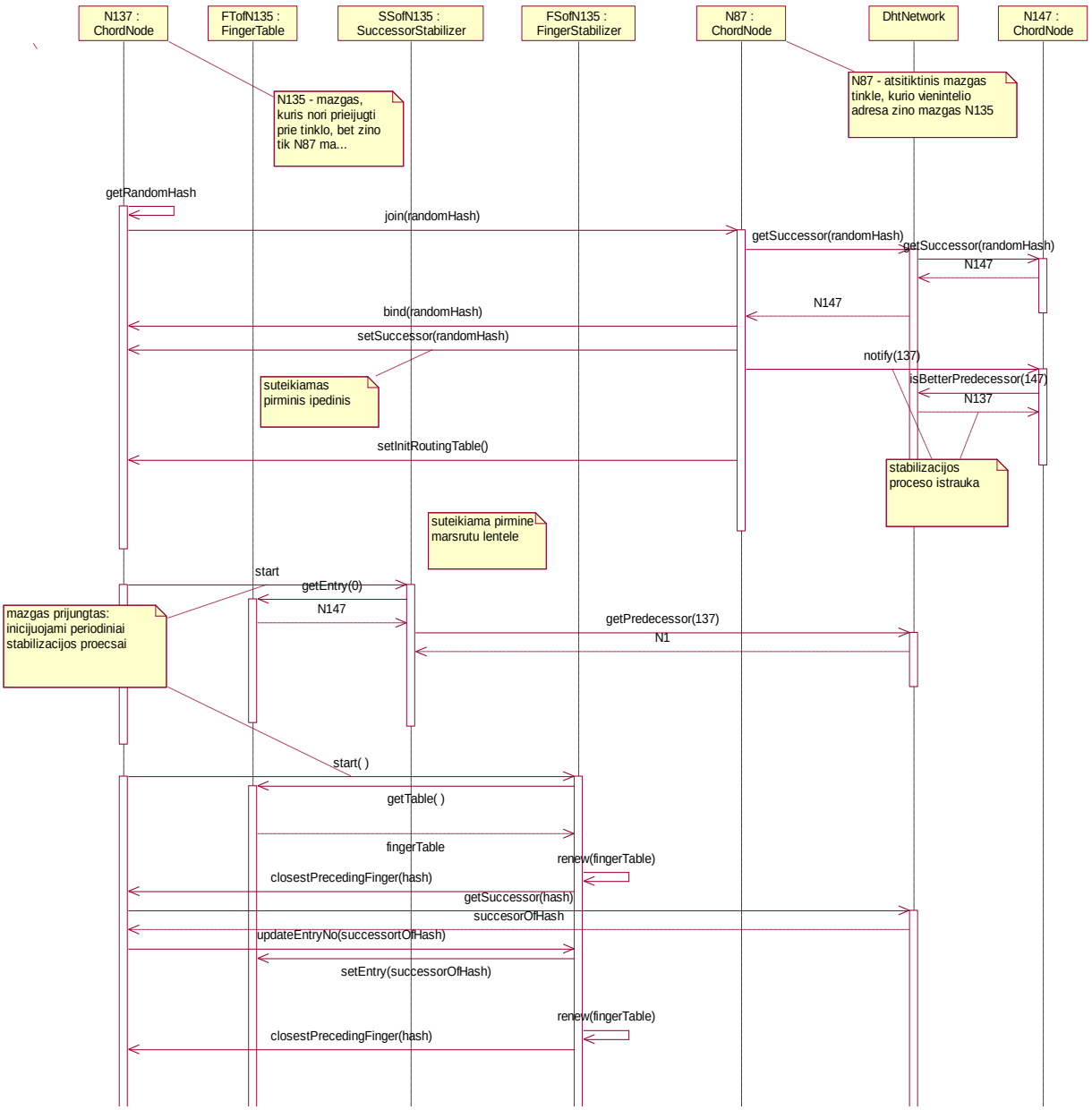
Log– klasė padedanti registruoti mazgų atliekamus veiksmus į failą.



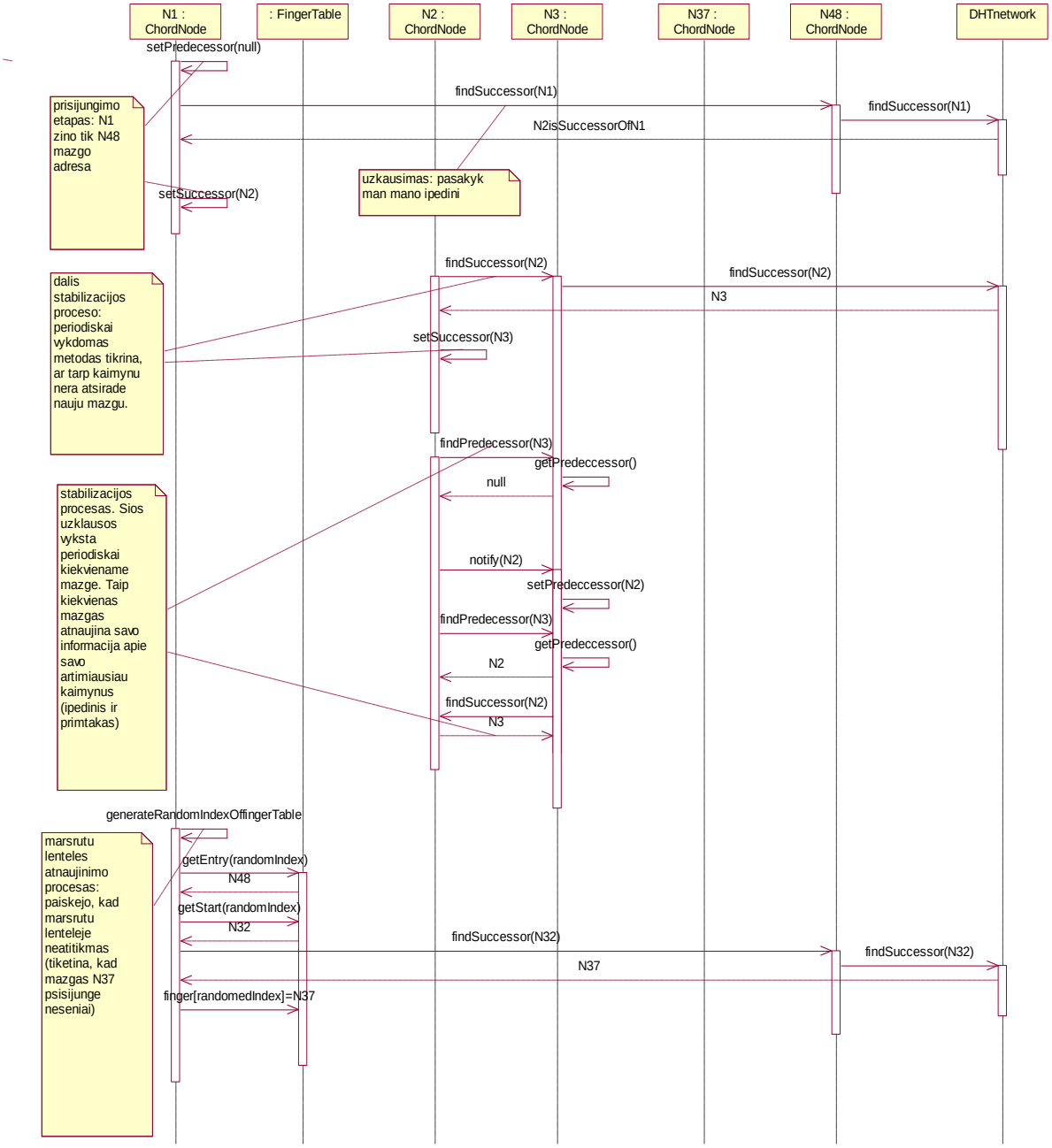
17 pav. Paketo „utils“ klasių diagrama

4.3. Sistemos elgsenos modelis

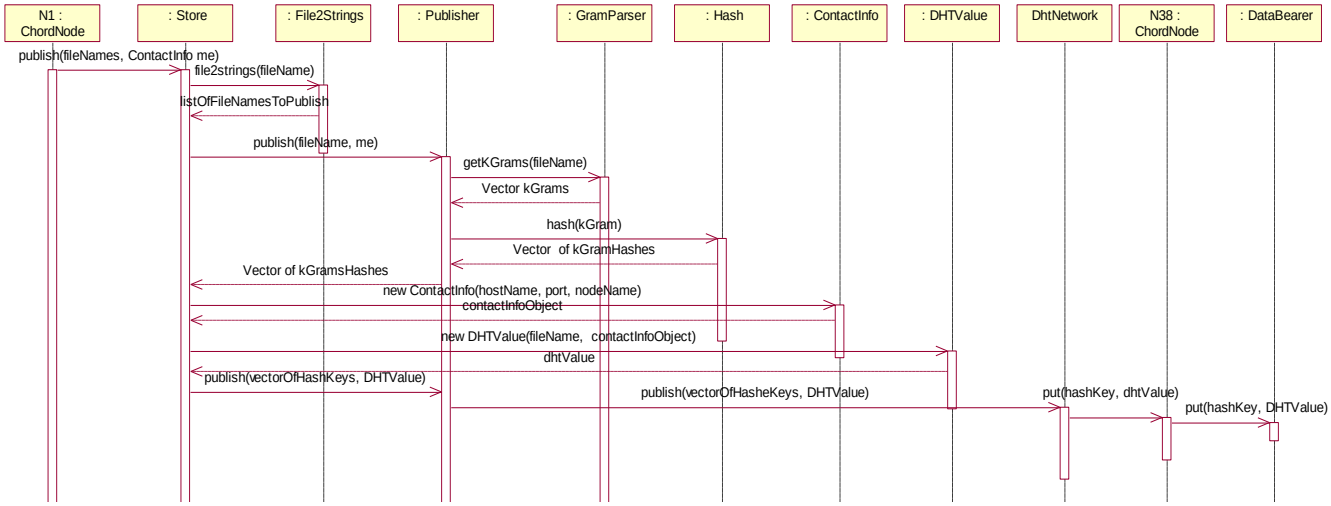
Sekų diagramos atspindi pagrindinius procesus vystančius sistemoje. Kiekvienam iš panaudos atvejų pavaizduota sekų diagrama.



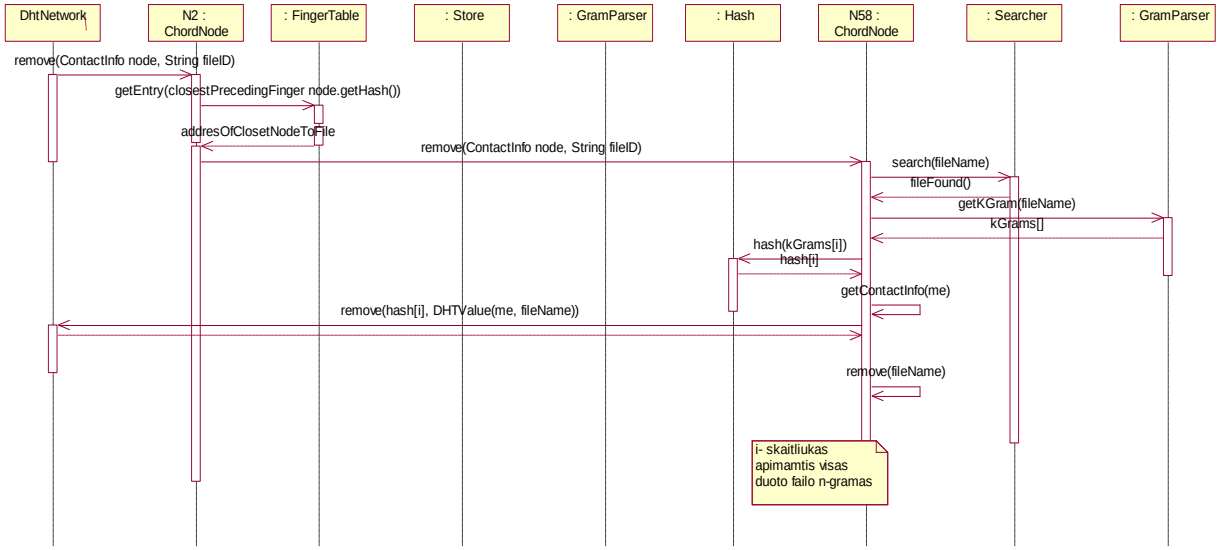
18 pav. PA „1. Prisijungti prie tinklo“ sekų diagrama



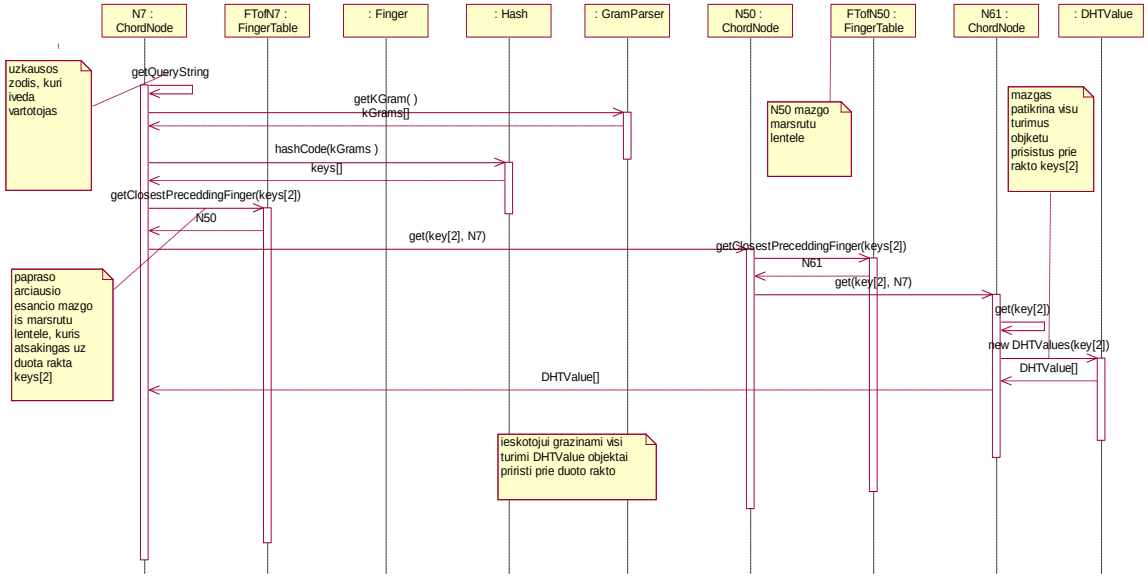
19 pav. PA „2. Vykdyti stabilizaciją“ sekų diagrama



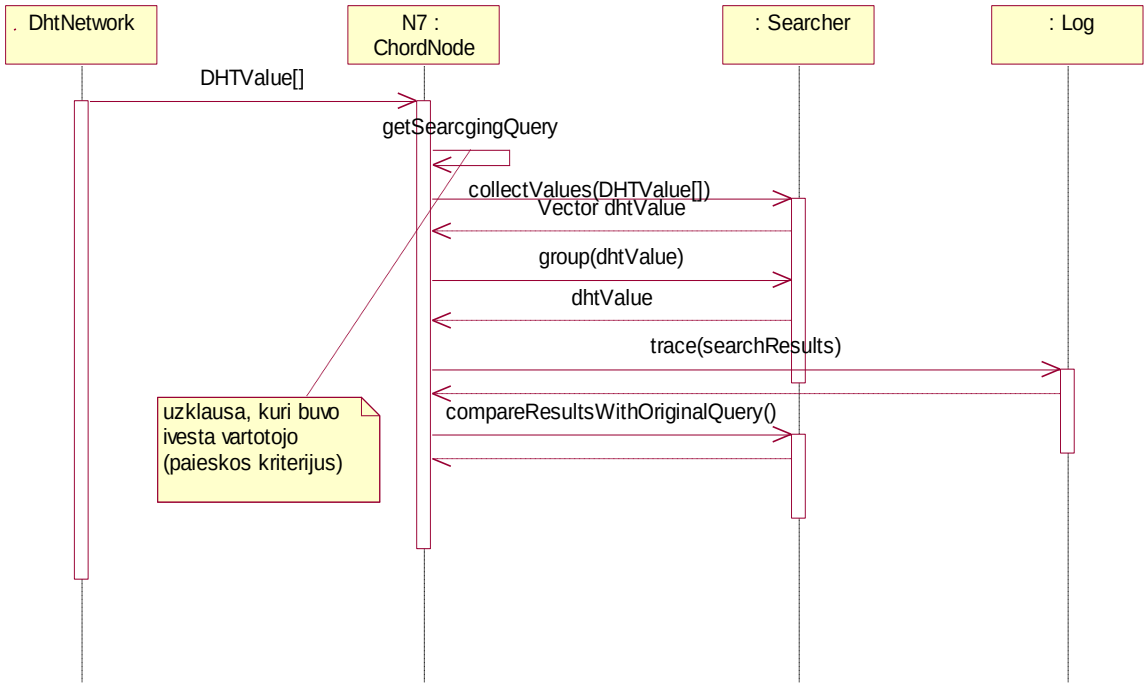
20 pav. PA „3. Publikuoti failą tinkle“ sekų diagrama



21 pav. PA „4. Pašalinti failą“ sekų diagrama



22 pav. PA „5. Siųsti užklausą“ sekų diagrama



23 pav. PA „6. Analizuoti rezultatus“ sekų diagrama

5. Eksperimentinis paieškos algoritmų tyrimas

5.1. N-gramų algoritmo eksperimentinio tyrimo metodika

Eksperimento sąlygos:

- Vienas kompiuteris;
- Vidutinis į tinklą publikuojamų failų skaičius: 50;
- Bandymams atlikti pasirinkta mazgų imtis: 10, 20, 50, 100, 200;
- Bandymams atlikti buvo naudojamos 3-gramos;
- Bandymų skaičius po 4 kiekvienai iš tinklo pasirinktų mazgų imčiai.

Dėl ribotų kompiuterio ir laiko resursų, negalėjome sau leisti atlikinėti bandymo realiomis sąlygomis. Bandymai atliekami viename kompiuteryje sukūrus DHT tinklus. Kiekvienas mazgas su tinklu bendrauja Java RMI technologijos pagalba.

Kiekvieno bandymo metu į tinklą publikuojama vidutiniškai 50 failų pavadinimų su pilnais katalogų vardais. Naudojant katalogų vardus pailgėja failų vardai, o kartu ir failą nurodančių raktų skaičius.

Maksimalų mazgų skaičių apribojome iki 200, nes ankstesnių bandymų metu padidinus tinklo dydį iki 260 mazgų, atsirasdavo Java pranešimai apie operatyviosios atminties nepakankamumą. Prisiminkime, kad eksperimentas atliekamas lokaliame kompiuteryje ir pasirinkome ribotą mazgų poaibį. Deja dėl resursų ir laikų trūkumo negalėjome atlikinėti eksperimento realiomis sąlygomis arba bent jau su DHT tinklu kuriame būtų keliasdešimt tūkstančių mazgų (tarkim sukurtų ant 50 kompiuterių).

Algoritmo tyrimui pasirinkome tokius **laiko vertinimo kriterijus**:

Laikas reikalingas viso tinklo stabilizacijai atlikti su pasirinktu mazgų skaičiumi

Laikas reikalingas visiems duomenims apie failus publikuoti tinkle

Laikas reikalingas atlikti failo paieškai (užklausa ir atsakymas į ją)

Kadangi bandymai atliekami viename kompiuteryje, tai laiko metrikos tik iš dalies atspindi realiomis sąlygomis veikiantį tinklą. Mes suprantame, kad laiko charakteristikų rezultatai [24 pav.] neturėtų būti traktuojami kaip simuliacija realių sąlygų. Kadangi pranešimo keliavimas realiu tinklu palyginus su RMĮ metodo įvykdimu lokaliame kompiuteryje gali užtrukti nuo kelių

iki kelių tūkstančių kartų ilgiau. Tačiau net tokiomis sąlygomis iš rezultatų galime spręsti apie tai, kad algoritmas veikia. Ir kaip ir galima buvo tikėtis ilgiausiai užtrunka publikavimo fazė.

Algoritmo tyrimui pasirinkome tokius **RMI kvietimų kiekio kriterijus**:

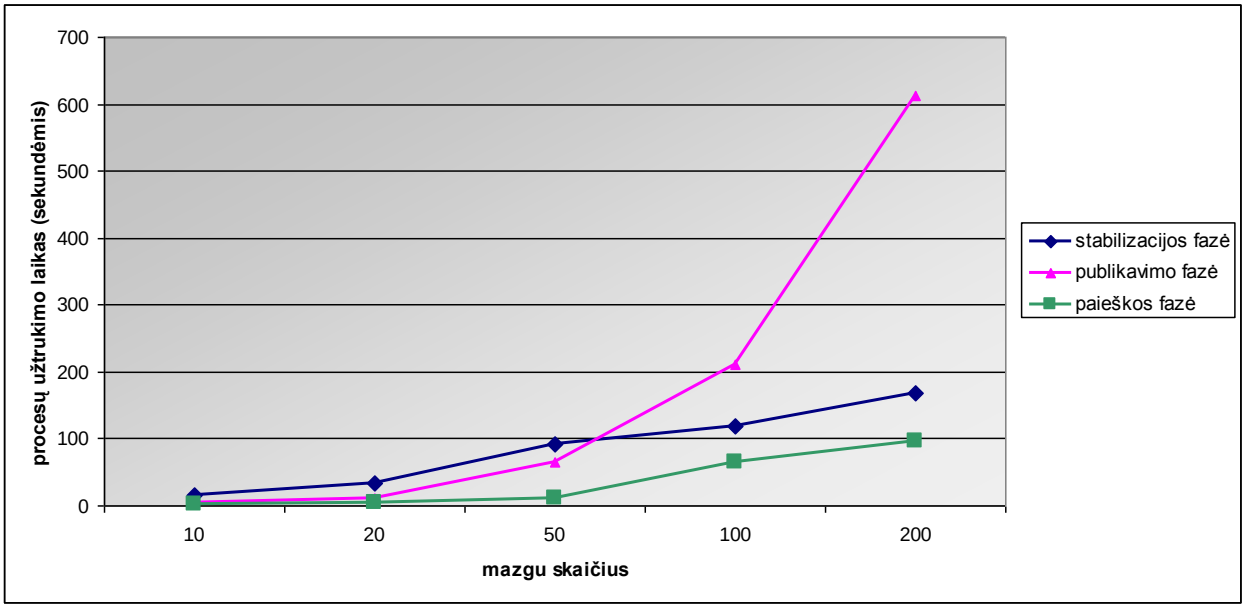
RMI iškvietimai reikalingi viso tinklo stabilizacijai atlikti;

RMI iškvietimai reikalingi visiems duomenims apie failus publikuoti tinkle

RMI iškvietimai reikalingi atlikti failo paieškai (užklausa ir atsakymas į ją)

5.2. N-gramų metodo tyrimo rezultatų įvertinimas

Atlikus po 4 bandymus su kiekviena iš 24 paveiksle pavaizduota mazgų imtimi ir skyriuje 5.1. minėtomis sąlygomis, gavome tokius 3-gramų eksperimento rezultatus [24 pav. ir 3 lentelė].

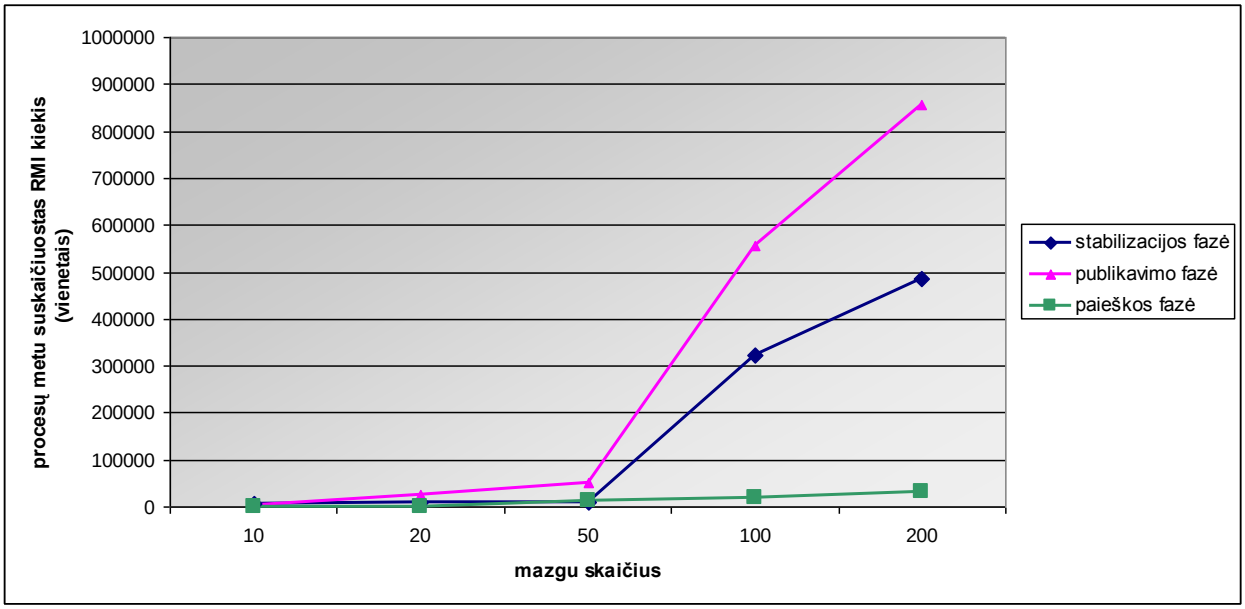


24 pav. Tinklo stabilizavimo, failų publikavimo ir paieškos procesų sunaudojamas laikas (sekundėmis)

3 lentelė. Procesų vykdymo laikas

Mazgų skaičius	**10**	**20**	**50**	**100**	**200**
stabilizacijos fazė	15	34	92	120	169
publikavimo fazė	4	12	64	210	612
paieškos fazė	1,5	4	12	64	96

Iš 24 paveiklso ir 3 lentelės galime spręsti, kad algoritmas veikia teisingai. Atsižvelgdami į tai, kad eksperimentas buvo vykdomas vienam kompiuteryje turėtume pripažinti, kad pranešimų srauto skaičiavimo metrika yra kur kas artimesnė realiems DHT tinklams veikiantiems ant nutolusių mašinų. Kaip jau minėta negalime RMI iškvietimų laiko atžvilgiu lyginti su realiais tinklais siunčiamų pranešimų vėlinimu. Tačiau turėdami šias laikines metrikas galime bent jau palyginti ir interpretuoti rezultatus tarp aprašytų trijų procesų. Taigi iš rezultatų matome, kad daugiausiai laiko užtrunka publikavimo fazė. Nors stabilizavimo fazė trukdavo ilgiausiai, kai mazgų skaičius buvo iki 50.



25 pav. Tinklo stabilizavimo, failų publikavimo ir paieškos procesų generuojamas RMI šaukinių kiekis

4 lentelė. Procesų generuojamas pranešimų kiekis

mazgai	10	20	50	100	200
stabilizacijos fazė	6317	8279	10874	324120	486641
publikavimo fazė	4764	26641	49541	557410	857654
paieškos fazė	217	331	12574	18547	32512

Iš 25 pav. arba 4 lentelės matome, kad daugiausiai RMI šaukinių generuoja failų publikavimas į tinklą. Stabilizavimo fazėje šaukinių padidėja tinklui išaugus iki 100 mazgų. Tai galima būtų paaiškinti tuo, kad paleidus 100 magų vienu metu, tinklas yra visiškai nestabilus. Tai nelabai atitinka net realaus labai dinamiško modelio. Esant DHT tinklui iš tarkim 4000 mazgų yra tik labai maža tikimybė, kad 90 % ar 100% mazgų prisijungs arba atsijungs vienu metu (kad ir koks dinamiškas ir nepastovus būtų). Tuo tarpu mūsų sukurtos sąlygos atspindi, kad pradinėje

gyvavimo fazėje į tinklą pasijungia visi 100% mazgų. Tuo ir galima būtų paaiškinti didelį pranešimų srautą stabilizavimo fazėje didėjant mazgų kiekiui.

Paieškos fazėje, kaip ir galima buvo tikėtis, sugeneruotas pranešimų kiekis yra kur kas mažesnis nei stabilizavimo ir publikavimo fazėse. Tačiau tinklui su 200 mazgų tai yra didelis žinučių kiekis. Tai iš dalies galima būtų paaiškinti tuo, jog visų procesų metu stabilizavimo fazė nebuvo išjungta t.y. kadangi tinklo mazgų maršrutų stabilizavimo mechanizmas susietas su naujų mazgų prijungimu prie tinklo.

Priede Nr. 3 aprašyta eksperimento su n-gramomis eiga.

5.3. Eksperimentų išvados

1. Panaudotos laikinės metodo matavimo metrikos nors ir nėra adekvačios realių tinklų laikinių charakteriskų (dėl to kad pranešimo siuntimas tinklu gali užtrukti nuo kelių iki kelių tūkstančių kartų ilgiau, nei pranešimo siuntimas tarps mazgų lokaliame kompiuteryje), tačiau, jos parodo bendrą tendenciją, kad n-gramų metode stabilizacijos ir publikavimo etapai yra labiau reiklūs laiko resursui nei paieškos fazė.

2. Iš sugeneruotų procesų metu pranešimų srautų metrikos galima, teigti, kad mazgų kiekiui didėjanti nuo 50 iki 200 stabilizavimo ir ypač publikavimo fazėje generuojamų pranešimų srautas yra kelis arba net keliolika kartų didesni.

3. Apibendrinus abi charakteristikas galima teigti, kad didėjant mazgų kiekiui tinkle didėja ir išnaudojamų resursų kiekis (šiuo atveju tinklo pralaidumo ir sunaudojamo laiko bendravimui). Lyginant su stabilizavimo ir publikavimo fazėmis tinklo dydžio augimui labiausiai atspari yra paieškos fazė. Net ir dvigubai padidėjus mazgų kiekiui nuo 50 iki 100 ir nuo 100 iki 200 žinučių kiekis generuojamas paieškos fazėje padidėja mažiau kaip 2 kartus.

6. Išvados

1. Literatūros analizės metu paaiškėjo, kad DHT algoritmais paremtos sistemos pasižymi didesniu plečiamumu, nei srautų schemos, tačiau joms trūksta užklausų lankstumo. Egzistuoja metodai ir algoritmai, kurie galėtų praplėsti jau egzistuojančius DHT tinklus ir suteikti jiems užklausų lankstumą, nepakeičiant jų stipriųjų pusių: plečiamumo, tolerancijos klaidoms, decentralizacijos.

2. Vienas tokių metodų, vadinamų n-gramomis, buvo įgyvendintas ir išbandytas *Chord* algoritmo pagrindu sukurtoje DHT sistemoje. Analizė ir eksperimentas parodė šio metodo privalumus, tačiau atskleidė ir keletą trūkumų. Pagrindinis metodo trūkumas yra, tai kad publikavimo ir stabilizacijos procesai generuoja kelis kartus didesnį pranešimų srautą, nei paieškos procesas.

3. Kitas būdas padidinti užklausų efektyvumą – įgyvendinti nestruktūrizuotos paieškos galimybes jau egzistuojančiame DHT tinkle, išplatinant užklausą iš anksto pasirinktam mazgų skaičiui. Tai metodas, kuris būdingas srautų schemoms, tačiau iš literatūros [13] žinome, kad panaudojus šią techniką DHT sistemose visiškai nebelieka perteklinių žinučių, kurias kartais generuoja srautų schemos. Todėl tinklui turinčiam N aktyvių mazgų užtenka N pranešimų, norit pranešimą iš vieno mazgo išplatinti visam tinklui (su prielaida, kad tinklas yra stabilus – mažai prisijungiančių ir atsijungiančiu mazgų tuo metu).

4. Du siūlomi metodai Chord algoritmu paremtoje sistemoje gali būti papildomu funkcionalumu šiuo metu dažniausiai naudojamoms technikoms tokioms kaip raktų platinimas pasitelkiant išorinius serverius. Naudojant vidinės, o ne išorines paieškos technikas DHT sistemoms sutiekiamas užklausų lankstumas būdingas srautų schemoms.

5. Eksperimento rezultatai su n-gramų metodu parodė, kad tinklo stabilizacijos ir failų publikavimo procesai yra kur kas imlesni srautų generavimui. Iš eksperimento paaiškėjo, kad didėjant mazgų skaičiui tinkle pranešimų srautai stabilizacijoje ir publikavime gali būti kelis, o kartais ir keliolika kartų didesni už paieškos generuojamus pranešimų srautus.

6. 2008 gegužės mėnesį konferencijoje „Informacinės technologijos 2008“ (XIII tarpuniversitetinė magistrantų ir doktorantų konferencija) organizuota KTU patalpose buvo skaitytas pranešimas paruoštas magistrinio darbo tezių pagrindu.

7. Literatūra

1. G. Couloris, J. Dollimore, T. Kindberg, „**Distributed Systems: concepts and deign**“ (fourth edition) 2005.
2. I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan. „**Chord: A scalable Peer-To-Peer lookup service for internet applications**“.
3. H. Balakrishanan, M. Frans Kaashoek, D. Karger, R. Morris, I. Stoica „**Looking up Data in P2P Systems**“.
4. M. Harren, J. Hellerstein, R. Huebsch, B. Loo, S. Shenker, I. Stoica „**Complex queries in dht-based peer-to-peer networks**“ 2002.
5. D. Tsoumakos, N. Roussopoulos „**Analysis and Comparison of P2P Search Methods**“ 2006.
6. I. Clarke, O. Sandberg, B. Wiley, and Theodore W. Hong „**Freenet: A distributed anonymous information storage and retrieval system. Lecture Notes in Computer Science**“, 2001.
7. http://en.wikipedia.org/wiki/Freenet. Paskutinį kartą žiūrėta 2008-02-28.
8. http://www.cs.aau.dk/~bnielsen/DS-E07/ Brian Nielsen, lecture notes on „Introduction to Distributed Systems (Autumn 2007)“. Paskutinį kartą žiūrėta 2008-01-14.
9. Antony Rowstron, Peter Druschel „**Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems**“, 18th OFO/ACM International Conference on Distributed Systems Platforms. Heidelberg, Germany 2001.
10. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf „**The Gnutella Protocol Specification v0.41Document Revision 1.2**“ paskutinį kartą žiūrėta 2008-03-03.
11. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker „**A Scalable ContentAddressable Network (CAN)**“.
12. Sameh El-Ansary, Luc Onana Alima, Per Brand, Seif Haridi „**A Framework for Peer-To-Peer Lookup Services based on k-ary search**“.
13. Samuel El-Ansary, Luc Onana Alima, Par Brand, Seif Haridi **„Efficient Broadcast in Structured P2P Networks“.**
14. P.Trunfio, D.Talia, A.Ghodsi, Seif Haridi „**Implementing Dynamic Quering Search in k-ary DHT-based Overlays**“. CoreGRID Technical Report, Number TR-0119, 2007-12-28.

15. Daniel Stuzbach, Reza Rjaie „Improving Lookup Performance over a Widely-Deployed DHT“. University of Oregon, USA, 2005.

Terminų ir santrumpų žodynas

DHT (angl. Distributed Hash Table) išskirstyta maišos lentelė – tai išskirstytos atminties sistemos, kurios palaiko **maišos lentelių** funkcionalumą.

Gyvavimo laiko parametras (angl. TTL – Time To Live)– paremetras, kuris apriboja užklauso gyvavimo laiką. Reikalingas tam, kad sumažinti užklausos skverbimosi gylį, naudojamas tada, kai užklausą reikia sunaikinti;

Godus persiuntimas pirmyn (angl. gready forwarding) – technika naudojama P2P sistemose mazgų tarpusavio komunikavimui;

Išskirstyta (paskirstyta) sistema (angl. Distributed System) – į tinklą sujungti kompiuteriai arba įrenginiai, kurie koordinuoja tarpusavio darbą pranešimais.

Išplečiamumas (angl. scalability) – viena iš P2P sistemų charakteristikų;

Kaimynas-įpėdinis (angl. successor) – Chord algoritme tai pirmas mazgas sistemoje, kurio nodeId yra **didesnis** už duotojo mazgo nodeId;

Kaimynas-pirmtakas (angl. predecessor) – Chord algoritme tai pirmas mazgas sistemoje, kurio nodeId yra **mažesnis** už duotojo mazgo nodeId;

Laisvosios užklausos (angl. arbitrary quering) –P2P sistemų funkcionalumo aspektas būdingas srautų schemoms. Tai galimybė užklausti savo kaimynų bet kokios iš anksto susitarto užklausos apie ieškomus objektus. Paprastai šis mechanizmas nėra būdingas DHT sistemoms aspektas;

Maišos lentelė (angl. Hash Table) – tai duomenų struktūra kurios elementai poros **raktas-reikšmė**. Tokioje lentelėje **raktas** gali būti susietas (angl. mapped) su keliomis reikšmėmis.

Maršrutų lentelė (angl. routing table) – tai mazgo kaimynų adresų lentelė. Chord algoritmo atveju paskutinėje šios lentelės vietoje yra adresas mazgo, kuris nutolęs per puse adresų nuo esamo mazgo, prieš paskutinėje– mazgo, kuris nutoles per ketvirtį, o pirmoje, mazgo kuris yra šio mazgo įpėdinis (pats artimiausias kaimynas);

P2P sistemos (angl. peer-to-peer systems) – sistemos, kurių atskiri komponentai pranešimų pagalba (nenaudojant centrinio mazgo) koordinuoja tarpusavio veiksmus.

Srautų schema (angl. flooding shema) – vienas iš P2P sistemų veikimo principų, kai tinklo dalyvis siunčia pranešimus savo kaimynams, tuo tarpu kaimynai persiunčia pranešimą

daliai savo žinomų mazgų. Šios schemos pagrindiniai trūkumai: negarantuojamas tikslus rezultatas, bei tinklo perkrovimas pertekliniais pranešimais;

Susiejimas (angl. mapping) – procedūra, kurios metu su vienos duomenų reikšmės atvaizduojamos kitomis (nebūtinai vienareikšmiškai);

Taiklumas (angl. hit guarantee) – tikimybė, kad duoto rakto reikšmė bus rasta, kai žinoma, kad tinkle toks raktas egzistuoja.

8. Priedai

8.1. Priedas Nr.1. pranešimas „Paieškos metodai išskirstytos maišos lentelėmis grindžiamose P2P sistemose“ skaitytas XIII tarpuniversitetinėje magistrantų doktorantų konferencijoje „Informacinės technologijos 2008“

PAIEŠKOS METODAI IŠSKIRSTYTOS MAIŠOS LENTELĖMIS GRINDŽIAMOSE P2P SISTEMOSE

Jonas Balčiūnas, Lina Nemuraitė

Kauno Technologijos Universitetas

Straipsnyje nagrinėjama resursų paieškos išskirstytos maišos lentelėmis (DHT) grindžiamose P2P sistemose problema. Palyginus su srautų schemų sistemomis (Gnutalla protokolas), DHT tinklai pasižymi didesniu plečiamumu ir „taiklesne“ paieška, tačiau nepalaiko laisvų užklausų – vartotojas turi žinoti ieškomo resurso raktą. Egzistuojančiuose DHT tinkluose ši problema sprendžiama naudojant išorinius raktų platinimo mechanizmus. Straipsnyje pateikiami algoritmai, leidžiantys įgyvendinti laisvas užklausas paties DHT tinklo viduje ir tokiu būdu sujungti DHT sistemų lankstumą su srautų sistemų paieškos efektyvumu.

Įžanga

Keičiantis informacija, Peer-to-Peer (P2P) tinklai turi didelių privalumų: sudaryti iš nepriklausomų kompiuterių, kurie gali laisvai įsijungti ar atsijungti, jie nereikalauja administravimo išlaidų, todėl vis plačiau naudojami ne tik interneto bendruomenėse, bet ir mokyme, versle. Dvi pagrindinės jų klasės yra srautų schemos (angl. Flooding Schemas) ir DHT arba „išskirstytos maišos lentelės“ (angl. Distributed Hash Table). Srautų schemomis paremti tinklai dar vadinami nestruktūrizuotomis P2P sistemomis, tuo tarpu visos DHT yra be išimties struktūrizuotos.

Viena iš svarbiausių problemų P2P sistemose yra objektų, kuriais dalinasi vartotojai, paieška. Straipsnyje nagrinėjama paieškos problema DHT sistemose. Čia aprašyti du galimi šios problemos sprendimo būdai. Vienas iš algoritmų yra įgyvendintas *Chord* algoritmu paremtoje DHT sistemoje. Aptariami abiejų šių algoritmų privalumai, trūkumai, bei pateikiami preliminarūs vieno iš algoritmų bandymų rezultatai.

Srautų schemų ir DHT lyginamoji analizė

Srautų schemų pavyzdys galėtų būti Gnutella protokolas [4]. Srautų schemose dažnai pasitelkiama supermazgų architektūra. Supermazgas – vienas iš sistemos dalyvių, kuris atsakingas už savo potinklio koordinavimą. Kiekvienas supermazgas turi supermazgų „kaimynų“ poaibį. Jei supermazgas „miršta“, visas paprastų mazgų poaibis dažniausiai lieka atskirtas nuo likusio tinklo. Tokiu atveju iš esamo poaibio mazgų išrenkamas naujas supermazgas, kuris paskiriamas toliau koordinuoti potinklio darbą.

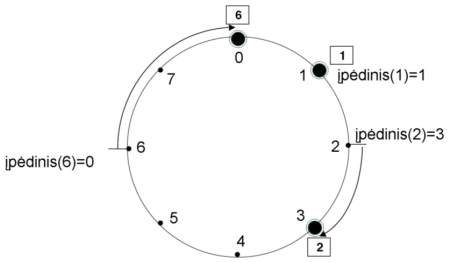
DHT grindžiamose sistemose apsieinama be supermazgų. DHT algoritmai suteikia tinklui labai didelį išplečiamumą (angl. *scalability*). Tokiose sistemose be papildomos kompiuterinės ir programinės įrangos sėkmingai veikia tinklai, sudaryti iš milijonų aktyvių vartotojų. Jų struktūra leidžia mazgų bendravimui siųsti mažiau pranešimų, o paieška tokiose sistemose yra taiklesnė (angl. *higher hit guarantee*), palyginus su srautų schemomis. Struktūrizuotose P2P sistemose (pvz., *Pastry, Tapastry, CAN* arba *Chord*) kiekvienam iš sistemoje dalyvaujančių mazgų yra paskiriama atsakomybė už tam tikrą dalį sistemos resursų. Tai atliekama suskirstant visą resursų raktų (arba adresų) aibę į intervalus. Kiekvienam mazgui priskiriamas tam tikrą intervalas. Kai mazgas nori surasti resursą, žinodamas tik raktą, DHT leidžia aptikti resursą per $O(\log N)$ pranešimų skaičių. Tam tereikia saugoti $O(\log N)$ kaimynų adresų kiekviename iš mazgų (1 lentelė). Pavyzdžiui, *Chord* algoritmu grįstose DHT sistemose, kai maksimalus tinklo dydis yra $1024 = 2^{10}$, DHT įgalina iš mazgo **A** nusiųsti pranešimą į bet kuri mazgą **Z** per daugiausiai $\log_2 1024 = 10$ tarpinių žinučių. Mazgui **A** tereikia saugoti 10 savo kaimynų adresų [1].

1 lentelė. Pranešimų tarp mazgų skaičius (angl. hops between nodes) ir maršruto lentelės dydis įvairiuose DHT algoritmuose

Algoritmas	**Paieškos ilgis (šuolių tarp mazgų skaičius)**	**Maksimalus maršruto lentelės dydis kiekviename mazge**	**Paaiškinimai**
Chord	$\log_2(N)$	$\log_2(N)$	*N – maksimalus mazgų skaičius (sistemos dydis)*
Tapestry	$\log_b(N)$	$b\log_b(N)$	*b – ieškojimo srities dekodavimo pagrindas*
CAN	$\frac{d}{4}n^{1/d}$	$2d$	*d – CAN tinklo koordinačių sistemos dimensijų skaičius*

Chord algoritmas

DHT algoritmuose kiekvienas iš mazgų yra atsakingas už tam tikrą visos raktų aibės dalį. *Chord* algoritme visa raktų aibė yra išskirstyta apskritime (1 pav.). Kiekvienas iš mazgų turi unikalų numerį *nodeId.* Mazgas m vadinamas rakto r įpėdiniu (anlg. *successor*), jeigu $r = m$ arba m yra pirmasis pagal laikrodžio rodyklę mazgas raktų apskritime (tuo atveju $r < m$). 1 paveiksle pavaizduoti trys mazgai su tokiais nodeId**:** 0, 1, 3. Mazgas 0 yra raktų 4, 5, 6, 7 ir 0 įpėdinis, mazgas 1 yra rakto 1 įpėdinis, o mazgas 3 yra raktų 2 ir 3 įpėdinis.



1 pav. Chord raktų aibės apskritimas susidedantis iš mazgų, kurių identifikatoriai yra 0, 1, 3; raktas 1 priskirtas mazgui 1, raktas 2 – mazgui 3, raktas 6 – mazgui 0.

Tam, kad *Chord* algoritmu paremtas tinklas veiktų teisingai, užtenka, kad kiekvienas iš mazgų turėtų savo pirmojo kaimyno pagal laikrodžio rodyklę adresą. Algoritmo veikimui to pakanka, tačiau nėra efektyvu, nes pranešimas, siunčiamas iš mazgo 1 į mazgą 0, turi apkeliauti visą apskritimą. Todėl kiekvienas iš tinkle esančių mazgų turi lentelę iš $m = \log_2 N$ įrašų (čia N – maksimalus tinklo mazgų skaičius). Mazgo A i-tajame maršrutų lentelės įraše saugomas adresas mazgo, kuris adresų apskritime yra nutolęs mažiausiai per 2^{i-1} adresų, t.y. 1 paveiksle matome, kad 0-nio mazgo pirmoje maršruto lentelės pozicijoje bus $2^{i-1} = 2^{1-1} = 1 \to 1$-mojo mazgo adresas, antroje: $2^{i-1} = 2^{2-1} = 2 \to 3$-čiojo mazgo adresas, trečioje $2^{i-1} = 2^{3-1} = 4 \to 0$-nio mazgo adresas [7].

Chord algoritmo veikimu paremta DHT sistema buvo įgyvendinta Java kalba, mazgų tarpusavio bendravimui naudojant Java-RMI technologiją. Sistemai sukurti nebuvo naudojama jokia iš jau įgyvendintų ir viešai platinamų *Chord* algoritmo versijų. Detalus algoritmo veikimo principas su stabilizacijos proceso patobulinimu aprašytas straipsnyje [7]. Dėl galimų algoritmo interpretacijų išlieka tikimybė, kad realizuota sistema nėra optimaliai veikianti *Chord* versija.

Siūlomas algoritmas paieškai įgyvendinti DHT tinklo viduje

Palyginus su srautų schemomis, DHT trūkumas yra tas, kad jos nepalaiko laisvų užklausų (angl. *arbitrary quering*) – norėdamas rasti resursą, vartotojas privalo žinoti jo raktą. Originaliose

DHT sistemose nėra galimybės „reguliariosioms išraiškoms“ [3] arba paieškai pagal raktažodžius, kadangi neefektyvu saugoti raktus kiekvienai užklausos išraiškai. Populiariose P2P srautų sistemose ši problema sprendžiama išplatinant raktus įvairiuose išoriniuose mechanizmuose. Remdamiesi [5, 8, 9], pateiksime du algoritmus, kuriais galima atlikti resursų paiešką DHT tinklo viduje.

Vienas iš būdų užtikrinti laisvas užklausas – įgyvendinti nestruktūrizuotos paieškos galimybes jau egzistuojančiame DHT tinkle, išplatinant užklausą iš anksto pasirinktam mazgų skaičiui. Tuomet DHT būtų galima atlikti tiek paiešką pagal raktą, tiek laisvas užklausas, taip sujungiant struktūrizuotų tinklų efektyvumą su nestruktūrizuotų P2P sistemų lankstumu. Tą galima atlikti dviem būdais: naudojant n-gramomis pagrįstą metodą arba laisvųjų užklausų mechanizmą.

Šiame tyrime objektų paieškai *Chord* tinkle pasirinktas n-gramomis paremtas metodas [5]. N-gramos – tai tam tikros žodžio dalys, kurių ilgis yra n. Pavyzdžiui, visos žodžio *„kompiuteris“* 4–gramos yra: *komp, ompi, mpiu, piut, iute, uter, teri, eris*. Kiekvienam maišos algoritmu apdorojamam objektui sudaromas raktas, suskaidytas į n-gramas. Taip gaunama daug raktų, kurie skeliami tinkle. Kiekvienai n-gramai yra sudaroma pora raktas-turinys, kur raktas – n-grama, transformuota panaudojant rakto funkciją, o turinys – objektas, kuriame nurodyta visas failo vardas; vieta, kur tas failas galėtų būti ieškomas; dydis arba papildoma informacija.

Norint atlikti paiešką, užklausos iniciatoriui nebereikia apklausinėti savo kaimynų kaip laisvųjų užklausų algoritme. Užklausos iniciatorius turi pateikti užklausos sakinį arba bent ieškomo failo pavadinimo skiemenis. Užklausos sakinys suskaidomas į pasirinkto ilgio n-gramas. Jos transformuojamos rakto funkcija. Gaunama daugybė raktų, kurie nurodo ieškomą failą arba failus, turinčius pavadinime tokius pat skiemenis Šis algoritmas skiriasi nuo laisvųjų užklausų algoritmo tuo, jog užklausos siunčiamos tiems mazgams, kuriuose tikimasi rasti rezultatus, tuo tarpu laisvosios užklausos siunčiamos bet kuriems kaimynams tikintis, kad jie persiųs užklausą į savo kaimynus, kurie gal būt turės iniciatorių dominančių resursų.

Dinaminių užklausų tikslas yra sumažinti pranešimų srauto dydį, kurį generuoja mazgo sukurta užklausa, lyginant su srautų schemose generuojamu pranešimų srautu. Užklausos pradininkas pradeda resurso paiešką išsiųsdamas užklausą su mažu TTL parametru (angl. TTL – *Time To Live*) keliems iš savo kaimynų. TTL apriboja užklausos skverbimosi į tinklą gylį. Pradinė užklausa yra vadinama žvalgybine. Ji padeda apytiksliai apskaičiuoti ieškomo resurso populiarumą. Jeigu pradinio bandymo metu nesulaukiama norimo rezultato, užklausos pradininkas siunčia papildomą užklausą kitam savo kaimynų poaibiui su nauju TTL parametru.

Naujas TTL parenkamas atsižvelgiant į pirmosios užklausos rezultatus ir norimą rezultatų skaičių. Procesas yra kartojamas tol, kol aptinkamas laukiamas resursų skaičius arba apklausiami visi kaimynai [11].

N-gramų metodo analizė

Realizavus n-gramų metodą *Chord* algoritmu paremtoje DHT sistemoje, buvo atliekamas eksperimentinis tyrimas: viename kompiuteryje paleidžiama 100 *Chord* algoritmo mazgų; pasibaigus stabilizacijai (kai visų mazgų maršrutų lentelės nebesikeičia), tinkle publikuojami failai, kurių vardai suskaidomi į n-gramas ir jos transformuojamos į raktus, naudojant rakto funkciją. Duomenys apie failus talpinami mazguose, atsakinguose už tam tikrus raktus. Užklausos iniciatoriui pateikus užklausos sakinį, jis suskaidomas į n-gramas, kurios rakto funkcija transformuojamos į raktus ir šių raktų užklausos siunčiamos į kiekvieną mazgą.

Eksperimento sąlygos: n-gramų ilgis $n = 3,4,5$; maksimalus tinklo dydis $N = 2^{10} = 1024$; tinkle esančių mazgų skaičius: $m = 10,50,100$. Eksperimento metu išaiškėjo tokie šio metodo privalumai:

- didelis rezultatų skaičius – mazgas „užklausėjas“ gauna daug rezultatų, kuriuos analizuodamas gali atsirinkti, kas jį domina. Pavyzdžiui, vartotojas gali užklausti visų sistemoje esančių *.ppt rinkmenų;
- užklausų taiklumas – kadangi kiekviena n-grama iš užklausos sakinio transformuojama į raktą, užklausa bus nusiųsta būtent tam mazgui, kuris atsakingas už tą raktą. Užklausos siuntimas į mazgą turint raktą užima daugiausiai $\log_2(N)$ „šuolių“ nuo užklausos iniciatoriaus iki bet kurio mazgo **Z**;
- rašybos klaidų toleravimas – kadangi iš užklausos sakinio padaroma daug raktų, rašybos klaidos šiame sakinyje nedaug veikia gautus rezultatus. Tą užtikrina n-gramų mechanizmas. Kuo n-grama trumpesnė, tuo daugiau iš bet kurio užklausos sakinio sudaroma raktų ir tuo mažiau svarbi gramatinė klaida bet kurioje užklausos sakinio vietoje;
- mažas užklausos žinučių kiekis – teoriškai galima pastebėti, kad žinučių kiekis užklausos metu neturėtų viršyti $v\log_2(N)$, kur v – vidutinis užklausos sakinio n-gramų skaičius, N – maksimalus tinklo dydis. Tačiau eksperimento metu pastebėta, kad didėjant mazgų ir failų tinkle kiekiui, užklausoms reikalingų pranešimų kiekis didėja greičiau nei $v\log_2(N)$ funkcija. Neoptimalus sistemos veikimas gali būti susijęs su *Chord* algoritmo interpretavimu.

Eksperimento metu išryškėję metodo trūkumai:

- svarbi žodžių išdėstymo tvarka užklausos sakinyje – taip yra todėl, kad į n-gramų formavimą įtraukiamas tarpo simbolis. Problemą galima išspręsti, neįtraukant tarpo simbolio ir darant n-gramas tik iš užklausos sakinio žodžių;
- n-gramos ilgio įtaka sistemai – kuo trumpesnės n-gramos, tuo didesnės maišos lentelės yra generuojamos kiekviename mazge. Tačiau didinant n-gramų ilgį, mažėja tolerancija užklausų klaidoms – užklausos sakinyje esant gramatinėms klaidom, mažėja rezultatų skaičius;
- maišos algoritmo kolizijos problema (angl. *hash collision problem*) – pasireiškia tada, kai sutampa visiškai skirtingų n-gramų raktai. Pavyzdžiui, jei failas turi 4-gramą *„abcd“*, o kitas failas turi 4-gramą *„em0c“* ir rakto funkcija transformuoja abi reikšmes į vieną rakta r1, tuomet nurodžius šį raktą užklausose gaunami abu rezultatus, nors domina tik vienas iš jų;
- didelė maišos lentelė kiekviename iš mazgų – kuo sistemoje daugiau failų, tuo daugiau raktų reikia tų failų n-gramoms publikuoti ir kiekvienas mazgas turi skirti daugiau vietos šiems duomenims saugoti. Kai sistemoje yra 1000 mazgų ir kiekviename mazge po 100 failų, sistemos efektyvumas gali smarkiai sumažėti.
- apkrovos disbalansas dėl n-gramų skaičiaus santykio su tinklo mazgų skaičiumi. Logiškai mąstant, angliškoje abėcėlėje yra 26 skirtingos raidės (failai dažniausiai vadinami UTF-8 formato simboliais). Jei imsime 3-gramas, turėsime $26 \cdot 26 \cdot 26 = 17576$ skirtingas variacijas. Jei tinklą sudaro daugiau nei 26^3 mazgų, kai kurie iš mazgų nesaugos jokios informacijos apie tinkle esančius resursus. Tuo tarpu kai kurie mazgai bus apkrauti labiau;
- duomenų rezultatų pertekliškumas – kai sistemoje publikuojama labai daug failų, susiduriama su duomenų pertekliškumu dėl to, kad, pavyzdžiui, n-gramos *„mp3“* raktas gali turėti labai daug failų.

Dinaminių užklausų mechanizmo analizė

Nors su šiuo algoritmu eksperimentas nebuvo atliktas, tačiau galima išskirti tokius šio metodo privalumus:

- maža tinklo apkrova, lyginant su srautų schemomis – kadangi pranešimui išsiųsti visiems tinklo mazgams *Chord* algoritmu paremtose sistemose reikia tik m žinučių (m – prie tinklo prisijungusių mazgų skaičius), o tai yra efektyviau, nei įprastuose srautų schemų mechanizmuose (standartinėje srautų schemoje galimi užklausų ciklai, kurie nutraukiami tik pasibaigus TTL laikui).

- užklausų analizės lygiagretumas – esant labai didaliam tinklui ir jame esančių failų skaičiui, kiekvienas mazgas gali saugoti nemažą kiekį duomenų. Todėl mazgas, gavęs užklausą, pirmiausia siunčia ją kitam savo kaimynui ir tik po to pats ją analizuoja. Taip analizė atliekama visuose užklausą priėmusiuose mazguose, ne tik iniciatoriuje kaip n-gramų metode.

Trūkumai, kuriuos galima įžvelgti dinaminių užklausų algoritme:

- mažesnis užklausų taiklumas – kai tinklas didelis, neracionalu siuntinėti pranešimus visiems tinklo dalyviams. Todėl lieka tikimybė, kad pasirinkus tik dalį apklausiamo tinklo, dominantis rezultatas bus saugomas toje dalyje, kuri liko neapklausta;
- efektyvumas priklauso nuo tinkle platinamų resursų kopijų. [9] šaltinyje skelbiama, kad laisvų užklausų mechanizmas efektyvesnis spartos ir užklausų taiklumo atžvilgiu, kai tinkle platinamos resursų kopijos. Tuomet didėja tikimybė, kad užklausos iniciatoriaus žinutė greičiau pasieks reikiamą resursą. Bet kadangi žinutės bus platinamos tik dalyje tinklo, išlieka tikimybė, kad nepopuliarius resursus, turinčius mažai kopijų, rasti bus sunkiau, t.y. nukentės užklausų taiklumas.

Išvados

DHT pasižymi didesniu plečiamumu, nei srautų schemos, tačiau jiems trūksta užklausų lankstumo. Analizės metu paaiškėjo, kad egzistuoja metodai ir algoritmai, kurie galėtų praplėsti jau egzistuojančius DHT tinklus ir padaryti juos lankstesnius.

Vienas iš būdų padidinti užklausų efektyvumą – įgyvendinti nestruktūrizuotos paieškos galimybes jau egzistuojančiame DHT tinkle, išplatinant užklausą iš anksto pasirinktam mazgų skaičiui. Tuomet DHT būtų galima atlikti tiek paiešką pagal raktą, tiek laisvas užklausas, taip sujungiant struktūrizuotų tinklų efektyvumą su nestruktūrizuotų P2P sistemų lankstumu.

Vienas tokių metodų, vadinamų n-gramomis, buvo išbandytas *Chord* algoritmo pagrindu sukurtoje DHT sistemoje. Eksperimentas parodė šio metodo privalumus, tačiau atskleidė ir keletą trūkumų, kurių būtų galima išvengti kitoje DHT versijoje. Numatyta ištirti ir dinaminių užklausų veikimą bei palyginti su n-gramų metodu.

Literatūros sąrašas

[1] S. El-Ansary, L.O.Alima, P.Brand, S.Haridi. „**Efficient Broadcast in Structured P2P Networks**“. 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS'03), Berkley, USA, 2003, puslapiai 1-3, 5.

[2] H.Balakrishnan, M.F.Kaashoek, D.Karger, R.Morris, I.Stoca. **„Looking Up Data in P2P Systems**“. 2003-02, Communication of the ACM/Vol 46, No. 2, puslapiai 45-47.

[3] M.Castro, M.Costa, A.Rowston. **„Debunking some myth about Structured and Unstructured Overlays“**, 2nd Symp. on Networking Systems Desing and Implementation (NSDI'05), Boston, USA, 2005.

[4] A.Fisk. „**Gnutella Dynamic Query Protocol v0.1**“. 2003 http://www.limewire.com/developer/dynamic_query.html, paskutinį kartą žiūrėta 2008-04-07.

[5] M.Harren, J.M.Hellerstein, R.Huebsch ir kt.. „**Complex Queries in DHT-based Peer-to-peer Networks**“, 2001, puslapiai 1-3.

[6] H.Jiang, S.Jin. **„Exploiting Dynamic Querying like Flooding Techniques in Unstructured Peer-to-Peer Networks“**. 13th IEEE Int. Conf. on Network Protocols (ICNP 2005), Boston, USA, 2005.

[7] I.Stoica, R.Morris, D.Karger, M.F. Kaashoek, H.Balakrishnan. „**Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications**“. MIT Laboratory for Computer Science, Tech. Report TR-819, puslapiai 149-160.

[8] D.Stutzbach, R.Rajaie. „**Improving Lookup Performance over a Widely-Deployed DHT**“. Departament of Computer & Information Science, University of Oregon, 2005, puslapiai 2-10.]

[9] P.Trunfio, D.Talia, A.Ghodsi, Seif Haridi „**Implementing Dynamic Quering Search in k-ary DHT-based Overlays**“. CoreGRID Technical Report, Number TR-0119, 2007-12-28, puslapiai 3, 4, 8.

Search Methods for P2P Systems Based on Distributed Hash Tables

In this article we are solving resource search problem in DHT-based P2P systems. DHT systems are more scalable than flooding schemas and have higher hit guaranty. However, they do not support arbitrary querying: users must know the exact key of the resource. In most popular DHT implementations this problem is solved by propagating keys to web sites. We present analysis of searching resources on DHT using “arbitrary querying” and “n-gram methods”. These techniques do not require external extensions (web sites), unifying the flexibility of flooding schemas with scalability of DHT.

8.2. Priedas Nr. 2 „Eksperimento su n-gramomis eigos pavyzdys“

Eksperimento scenarijaus sąlygos:

- Tinklas iš 20 mazgų (lokalus kompiuteris)
- Bandyme pasirinktos 3-gramos
- Po stabilizacijos proceso 2 mazgai publikuoja po 12 failų (viso 24 failai tinkle)

Pirmiausia tinklas pradeda savo darbą nuo vieno mazgo įsijungimo į tinklą. Vienas mazgas tinkle jau tinklas. Tinkle paleidus 20 mazgų prasideda rezultatų kaupimas:

New network started: 192.168.6.139:9300/nnn (6a75)

{10:53:49,865} - #192.168.6.139:9300/nnn (6a75)# - follower set to: 192.168.6.139:9300/nnn (6a75)

{10:53:49,865} - #192.168.6.139:9300/nnn (6a75)# - predecessor set to: 192.168.6.139:9300/nnn (6a75)

Iš rezultatų matome, kad pirmo mazgo sistemoje id = 6a75. Jis buvo startavo 10:53:49. Jo IP adresas 192.168.6.139, portas 9300. Kadangi tai kol kas pirmasis mazgas tinkle, pagal originalų Chord algoritmą jo pirmtakas ir įpėdinis yra jis pats.

Paskutinis mazgas, kurio id=4b65 prisijungęs prie tinklo 10:53:51.283. Jo įpėdiniu tampa 6a75.

{10:53:51,283} - #192.168.6.139:9319/nnn (4b65)# - Server rmi://192.168.6.139:9319/nnn bound and running.

{10:53:51,293} - #192.168.6.139:9319/nnn (4b65)# - follower set to: 192.168.6.139:9300/nnn (6a75)

Visi 20 mazgų turi tą patį IP adresą (kadangi šiuo momentu visi mazgai dirba lokaliame kompiuteryje). Skiriasi tik portai. Visų mazgų įpėdiniai yra nurodyti į mazgą 6a75, nes stabilizacijos procesas pradedama po visų mazgų prijungimo.

{10:53:50,374} - #192.168.6.139:9300/nnn (6a75)# - Finger table entry #:1 fixed with: 192.168.6.139:9300/nnn (6a75)

{10:53:50,381} - #[...] (6a75)# - Notified by better predecessor, changing from: 192.168.6.139:9300/nnn (6a75)

{10:53:50,382} - #192.168.6.139:9300/nnn (6a75)# - predecessor set to: 192.168.6.139:9301/nnn (93f1)

Mazgas 6a75 pakeitė savo maršrutų lentelės pirmąjį narį į save patį: (finger[1].node = 6a75) ir buvo perspėtas apie geresnį savo pirmtaką: 93f1. Ko gero, kad šis mazgas bus perspėtas apie dar geresnį savo pirmtaką netrukus, nes mazgas 93f1 yra pakankamai „toli“ (adresų atžvilgiu) nuo mazgo 6a75.

Paskutinis stabilizacijos proceso pranešimas:

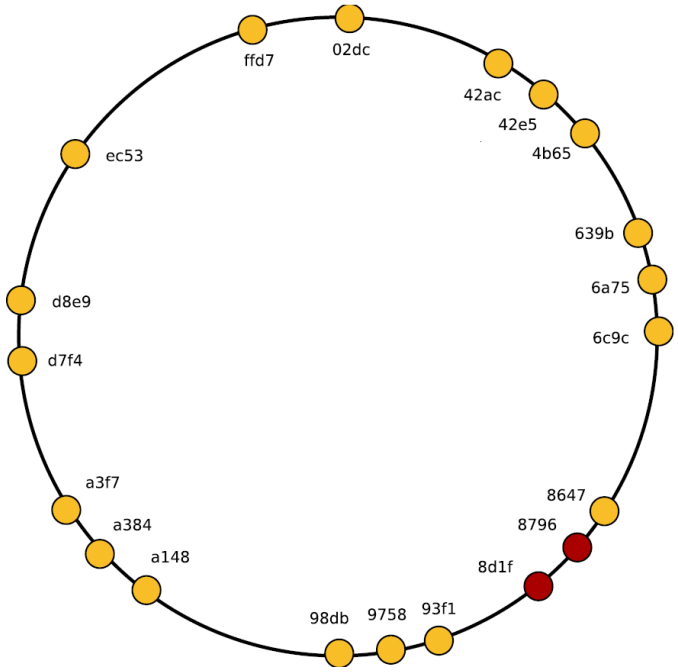
{10:54:05,340} - #192.168.6.139:9315/nnn (02dc)# - Finger table entry #:15 fixed with: 192.168.6.139:9308/nnn (42ac)

Mazgas 02dc pataisė savo 15-tąjį maršrutų lentelės įrašą į mazgą 42ac 10:54:05. Žinome, kad stabilizacijos procesas baigtas, nes pranešimų serveris nebeperspėjamas apie **įpėdinio** ir

pirmtako bei maršrutų lentelės pasikeitimus. Kaip po stabilizacijos atorodo visa mazgų išsidėstymo schema pavaizduota 26 pav.

5 lentelė. Susiformavusio tinklo schema :

įpėdinis	mazgas	pasiekėjas
ffd7	02dc	42ac
02dc	42ac	42e5
42ac	42e5	4b65
42e5	4b65	639b
4b65	639b	6a75
639b	6a75	6c9c
6a75	6c9c	8647
6c9c	8647	8796
8647	8796	8d1f
8796	8d1f	93f1
8d1f	93f1	9758
93f1	9758	98db
9758	98db	a148
98db	a148	a384
a148	a384	a3f7
a384	a3f7	d7f4
a3f7	d7f4	d8e9
d7f4	d8e9	ec53
d8e9	ec53	ffd7
ec53	ffd7	02dc



26 pav. Mazgų išsidėstymo vardų schema po Chord algoritmo pradinės stabilizacijos

Praėjo 15 sekundžių nuo stabilizacijos proceso pradžios ir pagal turimą RMI iškvietimo skaitliuką, nustatėme, kad stabilizacijos procesas pareikalavo 5512 RMI iškvietimų (tai skaičius pranešimų tarp mazgų).

Publikavimo etapas. Du mazgai publikuoja tinkle po 12 failų. Failų vardai prisekti priede Nr. 3.

Pirmos dvi operacijos publikavimo proceso metu:

{10:54:05,343} () - #[...] (d7f4)# - Databearer updated with d202/Pretty Vacant.mp3@192.168.6.139:9305/nnn (8647)

{10:54:05,348} () - #[...] (8d1f)# - Databearer updated with 882b/Pretty Vacant.mp3@192.168.6.139:9305/nnn (8647)

Kiekvienas iš mazgų turi klasę „Databaearer“. Šios klasės objektuose saugoma kiekvieno iš mazgų turimos poros (rakta, reikšmė). Iš rezultatų matome, kad mazgas d7f4 gavo raktą d202, kuris sudarytas iš 3-gramos nurodančios į failą „Pretty Vacant.mp3“. Šį failą publikavo mazgas 8647. Mazgas 8d1f1 gavo kitą raktą iš 3-gramos kuri publikavo mazgas 882b. Kuris susietas su tu pačiu failu „Pretty Vacant.mp3“. Pastebime, kad kiekvienas gautas raktas yra „labai arti“ (Chord adresų apskritimo atžvilgiu) paties mazgo adreso ir jo neviršija. Vadinasi procedūra **find_successor()** veikia teisingai.

Iš mazgo d8e9 gaunamos paskutinės dvi 3-gramos:

{10:54:08,793} () - #[...] (42ac)# - Databearer updated with 0fc2/Straight Lines.mp3@192.168.6.139:9306/nnn (d8e9)

{10:54:08,807} () - #[...] (a148)# - Databearer updated with 99f8/Straight Lines.mp3@192.168.6.139:9306/nnn (d8e9)

Kaip pavyzdį pateikiame raktų, kurie buvo atvaizduoti į mazgą 8d1f, sarašą:

<<>> 192.168.6.139:9307/nnn (8d1f) <<>>

8cf5

Waiting All Day.mp3@192.168.6.139:9306/nnn (d8e9)

8b54

E.M.I..mp3@192.168.6.139:9305/nnn (8647)

882b

Pretty Vacant.mp3@192.168.6.139:9305/nnn (8647)

8b20

Pretty Vacant.mp3@192.168.6.139:9305/nnn (8647)

8aa8

Anarchy In The UK.mp3@192.168.6.139:9305/nnn (8647)

8c5b

Straight Lines.mp3@192.168.6.139:9306/nnn (d8e9)

87dd

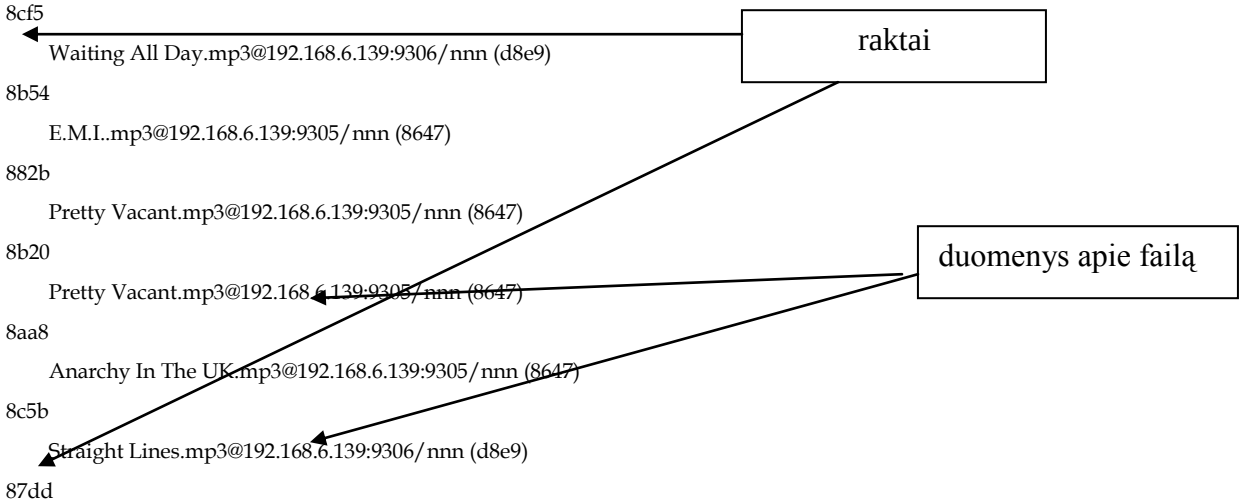
Those Thieving Birds (Part 1) Strange Behaviour [...] (Part 2).mp3@192.168.6.139:9306/nnn (d8e9)

8bfe

Pretty Vacant.mp3@192.168.6.139:9305/nnn (8647)

8aba

Those Thieving Birds (Part 1) Strange Behaviour [...] (Part 2).mp3@192.168.6.139:9306/nnn (d8e9)



Straight Lines.mp3@192.168.6.139:9306/nnn (d8e9)

Pastebėkime, kad dvi reikšmės yra atvaizduojamos į raktą 8aba, tai reiškia, kad yra du failai, kurie turi tas pačias 3-gramas. Taip pat gali būti, kad raktai sutampa dėl maišos funkcijos kolizijos. Jei pažvelgtume atidžiau, pastebėtumėte, kad abu failai, kurių 3-gramos atvaizduojamos į raktą 8aba turi bendrą 3-gramą „**str**“.

Taip pat pastebime, kad kai kurie iš mazgų (9b1b) neturi jokių jiems priskirtų raktų:

<<>> 192.168.75.195:9310/nnn (9b1b) <<>>

<<>> 192.168.75.195:9311/nnn (884b) <<>>

87dd

Those Thieving Birds (Part 1) Strange Behaviour [...] (Part 2).mp3@192.168.75.195:9306/nnn (0758)

871a

Silverchair - Young Modern.m3u@192.168.75.195:9306/nnn (0758)

882b

Pretty Vacant.mp3@192.168.75.195:9305/nnn (7cd8)

Atliekant paieškos etapą buvo pasirinkti tokie paieškos užklausa „god save the queen“, Sistemoje yra publikuojamas failas labai panašiu pavadinimu.

Rezultatai kuriuos gauname:

Chosen searching node: 192.168.6.139:9308/nnn (42ac)

Searching

Hit: 16 times, file: God Save The Queen.mp3, at: 192.168.6.139:9305/nnn (8647)

-===- Matching k-grams -===-

8ec1, k-gram = god

58f2, k-gram = od

9813, k-gram = d s

7e32, k-gram = sa

3eb2, k-gram = sav

a65d, k-gram = ave

4751, k-gram = ve

9cea, k-gram = e t

83a8, k-gram = th

ea6e, k-gram = the

62a3, k-gram = he

5bd1, k-gram = e q

15db, k-gram = qu

c377, k-gram = que

011e, k-gram = uee

2d6c, k-gram = een

-==========================-

Pastebime, kad atlikus paieška failas „God Save The Queen.mp3“ surinko daugiausiai taiklių paieškos rezultatų: 16. Jei pažvelgsime atidžiau, tai pastebėsime, kad iš užklausos „god save the queen“ galima padaryti 16-ka 3-gramų ir visos jų „taikliai pataikė“ į mūsų ieškomą failą.

Kitas pakankamai aukštai vertinamas rezultatų rinkinys buvo failas „Those Thieving Birds (Part 1) Strange Behaviour Those Thieving Birds (Part 2).mp3“:

Hit: 5 times, file: Those Thieving Birds (Part 1) Strange Behaviour Those Thieving Birds (Part 2).mp3, [...]
-===- Matching k-grams -===-
9cea, k-gram = e t
9cea, k-gram = e t
83a8, k-gram = th
83a8, k-gram = th
83a8, k-gram = th
-==========-

Jis surinko penkis sutapimus su mūsų užklausos n-gramų raktais. Taip atsitiko dėl to, kad kai kurios šio pavadinimo 3-gramos sutampa su užklauso 3-gramomis.

Buvo pabandyta, kaip užklausos klaidos įtakoja randamus failus. Su pasirinkta užklausa „good sav the qeuen“ buvo aptikti tik 8 sutapimai su failu „God Save The Queen.mp3“ kitas aukščiausiai įvertintas rezultatas buvo tik su 3 sutapimais (hits).

8.3. Priedas Nr. 3 „Failų pavadinimai, kurie buvo atrinkti publikavimui eksperimento su n-gramomis metu“

Failas1

Pretty Vacant.mp3
Submission.mp3
No Feelings.mp3
New York.mp3
Anarchy In The UK.mp3
Bodies.mp3
God Save The Queen.mp3
E.M.I..mp3
Problems.mp3
Liar.mp3
Seventeen.mp3
Holidayis In The Sun.mp3

Failas2:

Those Thieving Birds (Part 1) Strange Behaviour Those Thieving Birds (Part 2).mp3
The Man That Knew Too Much.mp3

All Across The World.mp3

Mind Reader.mp3

Insomnia.mp3

If You Keep Losing Sleep.mp3

Young Modern Station.mp3

Reflections Of A Sound.mp3

Low.mp3

Waiting All Day.mp3

Silverchair - Young Modern.m3u

Straight Lines.mp3