Miika Järvinen

Vertaisverkkojen hakualgoritmit

Tietotekniikan
kandidaatintutkielma
15. heinäkuuta 2005

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Tekijä: Miika Järvinen
Yhteystiedot: mijakrja@cc.jyu.fi
Työn nimi: Vertaisverkkojen hakualgoritmit
Title in English: Search Algorithms in Peer-to-Peer Networks
Työ: Tietotekniikan kandidaatintutkielma
Sivumäärä: 20
Tiivistelmä: Vertaisverkkojen käyttö on jatkuvassa kasvussa ja niiden skaalautuvuus lisääntyvään käyttäjämäärään vaatii tarvittavien resurssien löytämisen tehostamista. Tässä tutkielmassa esitetään erilaisia vertaisverkkojen hakualgoritmeja sekä kerrotaan yleisesti käytössä olevien vertaisverkkojen toiminnasta.
English abstract: Using of Peer-to-Peer networks is constantly growing. In order to scale to the increased amount of users finding of resources must be improved. This thesis presents different search algorithms used in Peer-to-Peer networks and describes the search implementations of publicly used Peer-to-Peer networks.
Avainsanat: vertaisverkko, hakualgoritmi
Keywords: Peer-to-Peer network, search algorithm

Sisältö

1 Yleistä vertaisverkoista

Tietokoneverkkojen solmut on perinteisissä verkkomalleissa jaettu palvelimiin ja asiakaskoneisiin. Protokollat toimivat siten, että asiakaskoneet ottavat yhteyden palvelimeen, joka lähettää vastauksen asiakkaan pyyntöön. Palvelimen lähettämä vastaus on kooltaan yleensä moninkertainen pyyntöön verrattuna. Tämän seurauksena asiakkaan ulosmenevä kaista on hyvin vähäisessä käytössä. Vertaisverkkomallissa kaikki verkon solmut toimivat samanarvoisina ja osallistuvat tiedon levittämiseen, jolloin kaistankäyttö jakaantuu tasaisemmin. Kaistankäytön väheneminen johtaa myös kustannusten pienemiseen, koska organisaation verkko voidaan toteuttaa halvemmalla laitteistolla. Myös verkon vikasietoisuutta saadaan parannettua, koska yhden laitteen kaatuminen ei estä palvelun toimintaa.

Vertaisverkkoja on toteutettu erilaisilla tekniikoilla. Yksinkertaisin malli sisältää keskuspalvelimia, joihin muut solmut yhdistyvät ja ilmoittavat niiden tiedostojen tiedot, jotka yhdistyneestä solmusta on mahdollista kopioida. Keskuspalvelimet toimivat vertaisverkosta saatavilla olevan tiedon indeksoijina, jonka seurauksena verkkoon suoritettavat haut voidaan toteuttaa lähettämällä kysely keskuspalvelimelle. Tässä mallissa haun toteuttaminen on yksinkertaista, ja niiden tulokset kattavat koko verkon. Mallin ongelmana on kuitenkin keskuspalvelimien aiheuttama verkon haavoittuvuus: jos keskuspalvelimiin ei saada muodostettua yhteyttä, koko verkon käyttäminen on mahdotonta. Keskuspalvelimet muodostavat myös resurssillisen pullonkaulan verkkoon.

Keskuspalvelimen aiheuttaman haavoittuvuuden seurauksena luotiin vertaisverkot, joissa verkon solmut toimivat täysin itsenäisesti. Keskitetyn indeksin puuttumisen seurauksena haun toteuttaminen tälläiseen verkkoon on ongelmallista. Haun optimaalisuudelle tälläisessä verkossa voidaan asettaa erilaisia kriteerejä: haku voidaan optimoida esimerkiksi kaistankäytön, nopeuden tai haun kattavuuden suhteen. Hauille on kuitenkin tyypillistä, että kaikkia vertaisverkon solmuja ei pystytä tavoittamaan, jonka seurauksena haun tulokset jäävät puutteellisiksi.

Täysin hajautettujenkin vertaisverkkojen rakenne vaihtelee erilaisten toteutusten välillä. Osa vertaisverkoista toimii siten, että kaikki solmut ovat täysin samanarvoisia (engl. Pure Peer-to-Peer), kun taas osassa verkoista on verkon toiminnan kannalta tärkeämpiä solmuja (engl. hybrid Peer-to-Peer).

Uusimmat vertaisverkkomallit käyttävät täysin järjestämättömän vertaisverkon sijaan mallia, jossa verkon rakenne on järjestäynyt siten, että rakennetta muutetaan dynaamisesti uusien solmujen yhdistyessä osaksi verkkoa (engl. Distributed hash table, DHT).

Vertaisverkoille on useita käyttökohteita. Tunnetuin vertaisverkkojen käyttömuo-

to on tiedostojenvaihdossa käytettävät vertaisverkot, mutta vertaisverkkoja käytetään myös matemaattisessa ongelmanratkaisussa [10] sekä "Voice Over IP" -puheluissa (VoIP), jotka ovat IP-verkossa reititettyjä puheluita. [18]

Monet vertaisverkoista käyttävät tiedostojen tunnistamiseen hash-funktioita. Hash-funktioiden avulla tiedoston sisällöstä saadaan muodostettua tarkistussumma, jonka avulla sisällöltään sama tiedosto voidaan tunnistaa samaksi, vaikka tiedoston nimi olisi eri. Hash-funktioiden muodostama tarkistussumma ei kuitenkaan ole täysin yksilöivä, vaan sisällöltään erilaisilla tiedostolla voi olla sama hash-arvo. Useimmilla hash-funktioilla tämä on kuitenkin hyvin harvinaista, mutta käytössä on myös sellaisia hash-funktioita, joiden väärentäminen on helppoa. [12]

1.1 Vertaisverkkojen haut

Käyttäjä tekee vertaisverkkoihin yleisimmin tiedostojen nimiin perustuvia hakuja, mutta vertaisverkkojen sisäisessä toiminnassa tarvitaan monia muitakin hakuja. Järjestäytyneiden vertaisverkkojen tapauksessa verkon solmujen on etsittävä toisia solmuja, ja opittava optimaalisia reititystapoja. Hash-funktioita käyttävissä vertaisverkoissa on solmujen aktiivisesti etsittävä uusia lähteitä, joilta tiedostoa voitaisiin kopioida. [18]

1.2 Vertaisverkkojen tietoliikennekaistan käyttö

Vertaisverkoissa käytetään tiedonvälitykseen yleensä IP-protokollaa, jonka sisällä kuljetetaan TCP ja UDP paketteja. Näistä TCP on yhteydellinen protokolla, joka takaa pakettien saapumisen lähettäjältä vastaanottajalle oikeassa järjestyksessä. UDP-protokolla sen sijaan ei sisällä yhteyden muodostusta, ja pakettien perille saapumista ei voida taata [22]. Vertaisverkot käyttävät tiedon levittämiseen yleensä TCP-protokollaa. UDP-protokollaa käytetään aikakriittisissä VoIP-palveluissa, sekä vertaisverkkojen hallintaan liittyvissä viesteissä. Osa vertaisverkoista kuitenkin toimii käyttämättä UDP-protokollaa.

Vertaisverkoille on tyypillistä, että käyttäjä on samanaikaisesti yhteydessä useaan samaan vertaisverkkoon liittyneeseen tietokoneeseen, joita kutsutaan solmuiksi. Niitä vertaisverkon solmuja, joihin solmu on muodostanut TCP-yhteyden kutsutaan solmun naapureiksi. [18]

Monet vertaisverkot sisältävät oman protokollan viestien välittämiseen. Osa vertaisverkoista käyttää verkon kehittäjien määrittämien protokollien sijaan standardien määrittelemiä protokollia. HTTP-protokollan käyttäminen viestien välittämiseen on yleistä, ja sen käyttäminen vaikeuttaa vertaisverkkojen aiheuttaman liikenteen suodattamista palomuureissa. Monessa vertaisverkko-ohjelmassa on mahdollista dynaamisesti

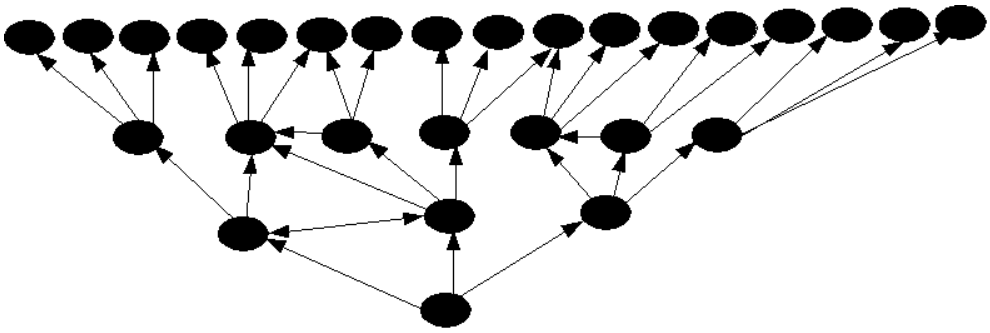
määrittää ohjelman käyttämä portti, joka myös vaikeuttaa vertaisverkkoliikenteen suodattamista. Vertaisverkkoliikenteen suodattaminen on kuitenkin organisaatioille tärkeä asia, koska vertaisverkoissa vaihdetaan usein tekijänoikeuden alaista materiaalia, jonka levittäminen organisaation verkosta voi johtaa oikeudellisiin seurauksiin. [18]

2 Tulviminen

Tulviminen (engl. Flooding) oli ensimmäisissä täysin hajautetuissa vertaisverkoissa yleisesti käytetty hakujärjestelmä, jossa hakupaketti lähetetään protokollan määrittämissä rajoissa mahdollisimman monelle solmulle. Tulviminen on tehokas tapa haun toimittamisessa usealle solmulle, mutta se käyttää paljon kaistaa. Tulviminen aiheuttaa ylimääräistä kaistankäyttöä, koska haku välitetään monille solmuille, jotka eivät pysty tarjoamaan vastausta tehtyyn kyselyyn. Solmu voi myös vastaanottaa samaa hakua kuvaavan paketin usean eri reitin kautta. Kuvassa 1 on esitetty esimerkki tulvimisesta.

Tulvimisen tehostamiseen on tehty erilaisia malleja, joiden avulla kaistankäyttöä pyritään vähentämään. Osa malleista toimii siten, että saatavat tulokset vastaavat täysin perinteisellä tulvimisella saatuja tuloksia. Osassa malleista sen sijaan saavutettavien solmujen määrä vähenee suuresti. Nämä mallit ovat erityisen tehokkaita sellaisen tiedon etsintään, joka on levinnyt usealle vertaisverkon solmulle.

Vertaisverkkojen hakupaketeissa käytetään usein laskuria (engl. Time To Live, TTL), joka kuvaa sitä, kuinka monen solmun kautta hakupaketti saadaan reitittää. Jokainen haun vastaanottanut solmu vähentää tätä arvoa vähintään yhdellä ennen haun eteenpäin lähettämistä. Tulvimisen ongelmana on myös se, että TTL-arvon kasvaessa lisääntyy myös niiden hakupakettien määrä, jotka sama solmu vastaanottaa useasta eri lähteestä. [18]



Kuva 1: Esimerkki tulvimisesta. Jokainen hakupaketin vastaanottanut solmu lähettää paketin kolmelle naapurisolmulle.

2.1 Asteittainen syveneminen

Perinteinen tulviminen toimii siten, että solmu lähettää hakupyynnön samanaikaisesti sovelluksessa määritetylle määrälle (yleensä kaikille) naapurisolmuja. Nämä solmut lähettävät paketin edelleen omille naapurisolmuillensa. Tätä jatketaan, kunnes on saavutettu protokollan määrittämä syvyys, eli TTL-laskurin arvona on nolla. Asteittaista syvenemistä (iterative deepening) käyttävässä haussa tiedon etsintä aloitetaan käyttäen pientä syvyyttä. Jos pienelle syvyydelle tehty tulos ei löydä riittävästi hakuun vastaavia resursseja, lähetetään uudelleen hakupyyntö, jossa syvyyttä on kasvatettu.

Asteittaisen syvenemisen mallilla saavutetaan samat solmut kuin perinteiselläkin tulvimisen mallilla. Jos solmu saa riittävän määrän vastauksia jo läheltä löytyviltä solmuilta, vähenee haun suorittamiseen tarvittava kaista. Mikäli haku sen sijaan joudutaan reitittämään samaan syvyyteen, kuin perinteistä tulvimista käytettäessä lisääntyy kaistan käyttö.

Beverly Yang ja Hector Garcia-Molina esittävät artikkelissaan [27], että asteittainen syveneminen voitaisiin toteuttaa käyttäen yhden syvyyttä ilmaisevan luvun sijaan kahta. Toinen luvuista määrittäisi sen, että kuinka monen verkon solmun kautta paketti reititetään automaattisesti. Jos haun aloittanut solmu ei ole saanut ajastimen lauettua riittävää määrää hakutuloksia, lähettäisi se solmuille paketin, jossa se pyytäisi laajentamaan hakua. Mallin käyttäminen vaatii, että solmut muistavat ne haut, joiden suoritus on keskeytetty, ja protokollan olisi tuettava haun yksilöintiä. Saatavana etuna kokonaan uuden kyselyn lähettämiseen nähden olisi pienempi tiedonsiirtotarve, koska hakuehtoja ei tarvitsisi lähettää uudestaan niiden solmujen läpi, joiden kautta alkuperäinen hakupaketti on jo kulkenut.

Asteittaisen syvenemisen hakuun käyttämä aika riippuu siitä, kuinka syvälle verkkoon hakua on jatkettava. Jos vastaus saadaan läheltä sijaitsevilta solmuilta, lyhenee hakuun käytetty aika, mutta suurimmassa osassa tapauksista haun suoritusaika kasvaa. Yangin ja Garcia-Molinan mittauksissa asteittainen syveneminen käytti haun suorittamiseen 90% enemmän aikaa kuin normaali tulviminen. [27]

2.2 Yhden solmun laajentaminen

Normaalissa tulvimisessa tieto lähetetään kaikille naapurisolmuille samanaikaisesti. Tällä tavalla saavutetaan nopeasti suuri määrä solmuja, mutta hakua ei pystytä keskeyttämään kun on saatu riittävä määrä vastauksia. Ratkaisun tähän ongelmaan tarjoaa malli, jossa solmu valitsee joka kerta yhden naapurisolmun, jolle se lähettää hakupaketin. Jos riittävää määrää vastauksia ei saada tältä solmulta lähetetään paketti seuraavalle solmulle. Tämä tapa on tavallista tulvimista hitaampi, mutta kuluttaa vä-

hemmän kaistaa jos riittävä määrä hakutuloksia saadaan jo aikaisessa vaiheessa tutkittavalta solmulta. Tätä tekniikkaa käytettäessä on mahdollista saavuttaa samat solmut kuin perinteisellä tulvimisella vastaavaa pakettimäärää käytettäessä.

Tärkeässä asemassa tässä tekniikassa on se, että kuinka hyvin solmun valinta onnistuu. Yangin ja Garcia-Molinan tekemien mittausten mukaan järjestämättömissä verkoissa hyvänä valintana toimii se, että kuinka paljon solmusta on saatu vastauksia edellisiin hakuihin.

Tämän menetelmän toiminta tehostuu, mikäli verkon rakenne ei ole täysin järjestämätön. Tällöin hakua voidaan kohdistaa niitä solmuja kohti, jotka verkon rakenteen puolesta vastaavat parhaiten annettuun hakuun. [27]

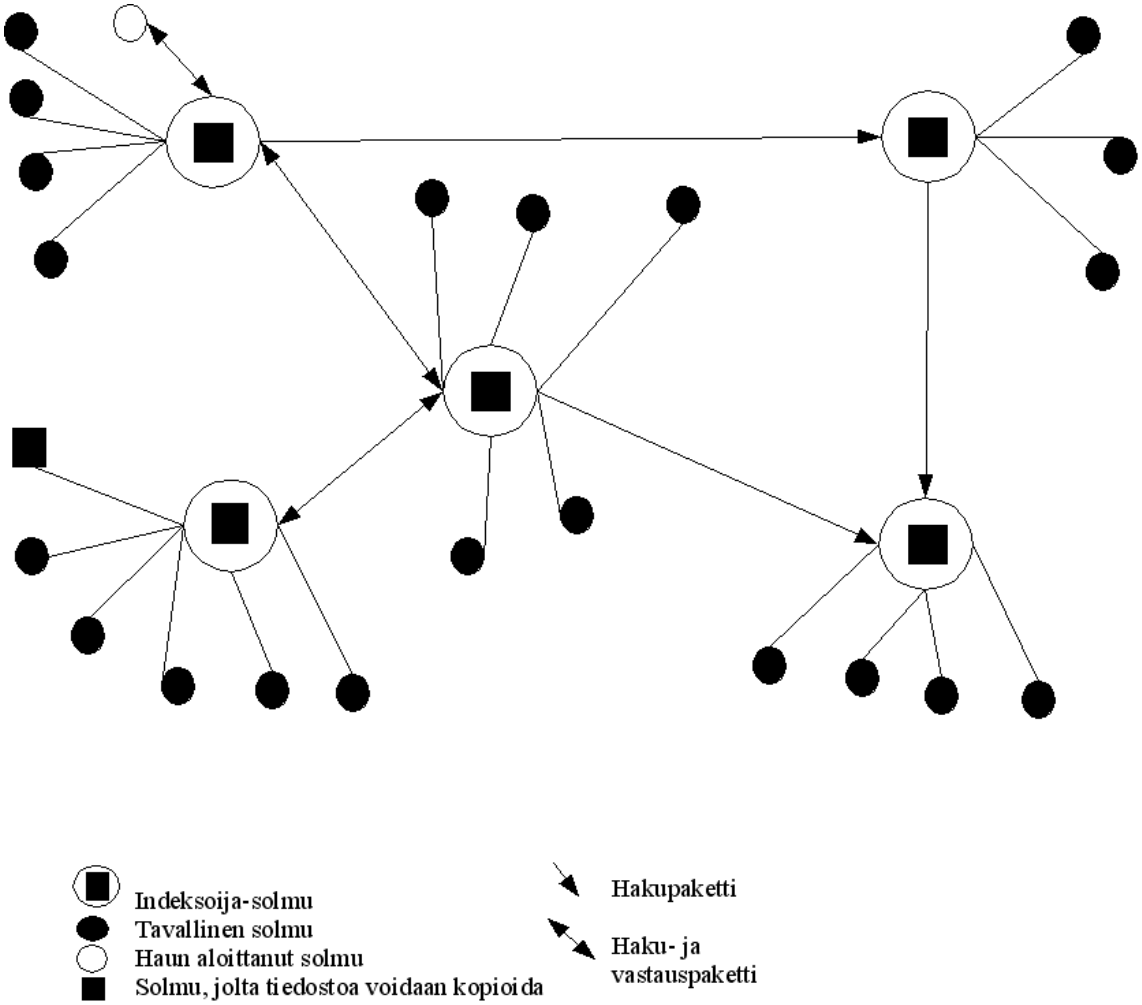
2.3 Ultrapeer

Ultrapeer-tekniikassa (tunnetaan myös nimellä Super-peer) verkkoon lisätään rakenteisuutta siten, että osasta verkon solmuista tulee verkon toiminnnan kannalta tärkeämpiä. Näiden solmujen tehtävänä on toimia hakujen paikallisina indeksoijina siten, että hakupakettia ei tarvitse lähettää kaikille niille solmuille, jotka ovat liittyneet korotettuun asemaan määrättyyn solmuun. Tämä vähentää tulvimisen aiheuttamaa liikennettä sen mukaisesti, kuinka monta solmua yhden korotetun solmun hallinnoimaan alueeseen kuuluu.

Ultrapeer-tekniikkaa voidaan käyttää monella tasolla. Yksinkertaisimmassa mallissa kaikki tavalliset solmut liittyvät yhteen korotetussa asemassa olevaan solmuun ja ilmoittavat jakamansa tiedostot tälle solmulle. Hakupaketin saapuessa tarkistaa indeksinä toimiva solmu, että jakaako joku sen hallinnoimista solmuista kyseistä tiedostoa. Jos tiedostoa jakava solmu löytyy voi indeksinä toimiva solmu verkon toteutustavasta riippuen joko itse lähettää vastauksen kyselyyn, taikka reitittää kyselyn ainoastaan niille hallinnoimilleen solmuille, joista kyseistä tiedostoa on mahdollista kopioida. [9]

Tämä malli on käytössä monessa vertaisverkossa ja tehostaa verkon toimintaa huomattavasti. Malli pienentää sekä kaistankäyttöä, että vasteaikaa. Koska mistä tahansa verkon solmusta voi muodostua korotetussa asemassa oleva solmu, eivät nämä solmut muodosta järjestelmään pistettä, jonka kaataminen estäisi koko järjestelmää toimimasta. Kuvassa 2 on esitetty tätä tekniikkaa käyttävän järjestelmän toiminta.

Toisessa mallissa kaikki verkon solmut tallentavat tiedot tietyllä syvyydellä itsestään olevien solmujen jakamista tiedostoista. Tätä mallia ei ole vielä käytetty yleisesti käytössä olevissa vertaisverkoissa, eikä malli vähennä paljoa tulvimisesta aiheutuvaa liikennettä. Tämä malli vähentää solmussa tapahtuvaa datan prosessointia, koska solmun ei tarvitse käsitellä kaikkia kauttaan kulkemia hakupyyntöjä. [27]



Kuva 2: Kaksitasoinen Ultrapeer-tekniikkaa käyttävä verkko, jossa indeksointisolmut vastaavat indeksoimiensa solmujen hakutuloksilla.

Ultrapeer järjestelmä antaa ainoastaan väliaikaisen helpotuksen verkon skaalautuvuusongelmaan.Ongelmana järjestelmässä on se, että mitä tehdään siinä vaiheessa, kun ultrapeer solmujen prosessointiteho on kokonaan käytössä. On esitetty, että verkkoihin luotaisiinkin kaksikerroksisen järjestelmän sijaan verkon koon mukaan skaalautuva n-kerroksinen järjestelmä. Sarsharin, Boykinin ja Roychowdhuryn mielestä tämä ratkaisu ei kuitenkaan toimisi oikeassa verkossa [20].

3 Kävelijä-algoritmit

Kävelijä-algoritmit ovat hakualgoritmeja, joilla pyritään vähentämään hakuun tarvittavaa kaistankäyttöä. Kävelijä-algoritmeille on tyypillistä, että solmuja tavoitetaan vähemmän kuin tulvimisessa. Algoritmit ovat kuitenkin tehokkaita haun suorittamiseen etenkin silloin, kun tieto on monistunut useaan vertaisverkon solmuun.

3.1 Satunnaiskävelijä

Satunnaiskävelijä (random walk) algoritmissa solmu, jolle kysely lähetetään valitaan satunnaisesti. Haun vastaanottanut solmu lähettää sen jälleen satunnaisesti jollekin muulle solmulle kuin sille, mistä paketti alunperin saatiin. Tätä toimintaa jatketaan kunnes paketin TTL-laskuri saa arvon nolla. Satunnaisen toiminnan ansiosta algoritmilla voidaan päästä syvemmälle verkkoon kuin tulvimista käytettäessä. Matemaattisesti on mahdollista todistaa, että satunnaiskävelijä algoritmi lähestyy sellaisia solmuja, joista on paljon yhteyksiä. [20] Algoritmi on tehokas, mikäli verkko sisältää indeksoijina toimivia solmuja, tai verkon rakenne on usein muuttuva.

Vertaisverkoista tietoa etsittäessä käyttäjille on tyypillistä, että sama haku suoritetaan uudestaan, jotta löydettäisiin uusia lähteitä halutulle tiedolle. Tulvimisessa tämä toiminta voi johtaa täsmälleen samojen solmujen läpikäymiseen, kun taas satunnaiskävelijä algoritmia käytettäessä on todennäköisempää, että etsityt solmut muuttuvat voimakkaasti. Satunnaiskävelijä algoritmia käytettäessä on usein tapana jopa lähettää useita hakupaketteja samanaikaisesti. Nämä hakupaketit reitittyvät eri osiin verkkoa ilmoittaen löytämistään vastauksista kyselyn aloittaneelle solmulle. Satunnaiskävelijä algoritmi voidaan toteuttaa myös siten, että käytyään läpi tietyn määrän solmuja tiedustellaan alkuperäisen haun lähettäneeltä, että halutaanko kyselyn vielä jatkavan kulkua verkossa. Tällä tavalla saadaan vähennettyä kaistankäyttöä, koska hakupaketit eivät enää liiku verkossa kun tarvittava määrä hakutuloksia on saatu kerättyä.

Satunnaiskävelijä algoritmi on sitä tehokkaampi, mitä useampi solmu kyseistä tiedostoa jakaa. Gkantsidis, Mihail ja Saberi suorittivat mittauksia, [11], joissa he vertasivat perinteistä tulvimista ja satunnaikävelijä algoritmia. Saatujen tulosten mukaan satunnaiskävelijä algoritmi löysi samalla pakettimäärällä enemmän erillisiä solmuja kuin tulviminen ja epäonnistuneiden hakujen määrä oli pienempi lukuunottamatta tilannetta, jossa verkon rakenne muuttui voimakkaasti. Epäonnistuneena hakuna mittauksissa pidettiin hakua, joka ei löytänyt haettua resurssia verkosta.

Algoritmien toimintaa testattiin verkossa, jossa verkon rakennetta muutettiin askelittain vaihtamalla solmujen välisiä yhteyksiä. Satunnaiskävelijä algoritmilla epäonnistuneiden hakujen määrä pysyi koko mittausten ajan noin 20%:ssa kun taas tulvimi-

sessa epäonnistuneita hakuja saatiin 2% muutoksella yli 60%. Kun rakennetta muutettiin 40% epäonnistuneita hakuja tapahtui tulvimisessa 20%. Tulviminen havaittiinkin tehokkaaksi, mikäli verkon rakenne on paljon muuttuva. [11]

3.2 Korkean asteen solmujen painottaminen

Adamic, Lukose, Puniyani ja Huberman ovat suorittaneet mittauksia [1], joiden mukaan satunnaisen valinnan sijaan kannattaisi aina valita seuraajaksi lokaalisti korkeimman asteen omaava solmu. Jos vastausta ei saada tältä solmulta siirryttäisiin seuraavaksi korkeimpaan. Tehtyjen mittausten mukaan tällä menetelmällä saataisiin 10000 solmun verkossa 80% peitto, satunnaisen kävelijän yltäessä ainoastaan 50% peittoon samalla määrällä suoritettuja askeleita.

Kummassakin mallissa tavoitetaan nopeasti 40% peitto, mutta tämän jälkeen uusien solmujen löytyminen hidastuu huomattavasti. Mittausten mukaan malli sopii potenssijakautuneita verkkoja paremmin Poisson-jakautumista noudattaviin verkkoihin. Potenssijakautuneissa verkoissa, joissa suurimmalla osalla solmuista on pieni määrä naapureita mutta pienellä osalla suuri määrä, menetelmä siirtyi alussa tapahtuneen nopean uusien solmujen löytymisen jälkeen 80% sellaisiin solmuihin, joissa se oli jo käynyt. Poisson-jakautuneessa verkossa, jossa kullakin solmulla on suunnilleen yhtä paljon naapureita, vastaavaksi arvoksi saatiin 50%. [1]

3.3 Suodatushaku

Sarshar, Boykin ja Roychowdhury esittävät artikkelissaan [20], että satunnaiskävelijä ja korkean asteen solmujen painottaminen eivät ole tarpeeksi skaalautuvia hakumenetelmiä suurissa vertaisverkoissa käytettäviksi. He esittävät ”Percolation search” -algoritmiksi (suodatushaku) kutsumansa algoritmin, joka löytää tiedon verkosta $\theta(logN)$ paketilla, jossa N kuvaa verkon kokoa. Algoritmin kehittäjien mukaan sellaisten hakujen määrä, jotka eivät tuota tulosta vaikka sellainen verkosta löytyisikin on pienempi, kuin käytettäessä satunnaiskävelijä- tai ”korkean asteen solmujen painottamis” -hakuja.

Suodatushaku jakaantuu kolmeen vaiheeseen. Uuden solmun yhdistyessä verkkoon ilmoittaa se jakamiensa tiedostojen tiedot käyttäen satunnaiskävelijä algoritmia. Satunnaiskävelijän tekemä askelten määrä valitaan sovellustasolla.

Suodatushaku algoritmissa haun suorittaminen jakaantuu kahteen vaiheeseen. Ensimmäisessä vaiheessa solmu lähettää haun käyttämällä satunnaiskävelijä algoritmia. Hakupaketin vastaanottaneet solmut tulvivat liikenteen naapurisolmuilleen suodatuskynnyksen (percolation threshold) määrittämällä todennäköisyydellä. Koska satunnaiskävelijä algoritmi lähestyy sellaisia solmuja, joista on paljon yhteyksiä on todennäköis-

tä, että haluttua tietoa jakava solmu saavutetaan. [20]

3.4 Huhun levittäminen

Marius Portmann ja Aruna Seneviratne esittävät artikkelissaan ”huhun levittäminen” (engl. rumor mongering) -algoritmien käyttämistä vertaisverkkojen hakutoteutuksena.

”Huhun levittäminen” -algoritmissa kukin solmu tallentaa tiedon siitä, että kuinka monesti se on tietyn hakupyynnön nähnyt ja mille naapurisolmuista sen edelleenlähettänyt. Jos solmu ei ole nähnyt hakupyyntöä protokollan määrittämää kertaa, valitsee se satunnaisesti protokollan määrittämän määrän naapurisolmuja, joille se ei ole vielä lähettänyt kyseistä hakupyyntöä ja lähettää sen niille.

Portmann ja Seneviratne esittävät myös ”huhun levittäminen” -algoritmista laajennetun version, jossa kukin solmu tietää moneenko solmuun sen naapurisolmut ovat yhteydessä. Algoritmista käytetään artikkelissa ’deterministic rumor mongering’ nimeä. Tämä algoritmi toimii siten, että kun solmu näkee hakupyynnön ensimmmäistä kertaa, lähettää se sen niille solmuille, jotka ovat yhteydessä ainoastaan yhteen solmuun (ainoa reitti kyseiseen solmuun on hakupaketin vastaanottaneen solmun kautta). Muut solmut asetetaan yhteyksien määrän mukaan nousevaan järjestykseen. Hakupaketti lähetetään satunnaisesti protokollan määrittämälle määrälle naapurisolmuja, joista on vähiten yhteyksiä, ja jotka eivät ole vastaanottaneet hakupakettia.

Portmannin ja Seneviratnen tekemien mittausten mukaan ”huhun levittäminen” -algoritmilla saavutetaan noin 40% tulvimista pienemmällä kaistankulutuksella 90% prosenttia tulvimisella saavutetuista solmuista, kun kukin solmu käsitteli hakupyynnön korkeintaan kaksi kertaa lähettäen sen molemmilla kerroilla kahdelle solmulle. ”Deterministinen huhun levittäminen” -algoritmi saavutti kaikki verkon solmut samoilla parametreilla. Mittausten mukaan algoritmien vasteaika oli noin kaksinkertainen tulvimiseen verrattuna. [19]

4 Järjestäytyneet vertaisverkot

Monet uudet vertaisverkkojen toteutukset käyttävät järjestäytymättömän verkon sijaan rakennetta, jossa verkon solmut muodostavat topologian, jonka muutoksista solmut ilmoittavat toisilleen. Tämä mahdollistaa haun toteuttamisen siten, että haun tulokset kattavat kaikki vertaisverkon solmut ilman keskuspalvelimien käyttöä.

Hakuindeksi levitetään verkon solmujen kesken siten, että kukin solmuista vastaa osan hakuindeksin tallennuksesta ja osaa reitittää kyselyjä kohti sellaisia solmuja, jotka vastaavat haussa annetun tiedoston indeksoinnista. Hakuindeksin osat on verkossa

monistettu siten, että yhden verkon solmun kaatuminen ei estä kyseisen tiedon indeksitietojen hakemista verkosta. Hakuindeksin ylläpitäminen vaatii solmujen välistä yhteydenpitoa, ja kuluttaa tämän seurauksena kaistaa. Koska solmujen poistuminen ja liittyminen verkkoon vaatii hakuindeksin päivitystä, pidetään kunkin solmun hallinnoima hakuindeksin osa pienenä. [8]

Gupta, Liskov ja Rodrigues ovat esittäneet [14], että hakuindeksi voitaisiin toteuttaa myös siten, että kukin solmu tuntisi kaikki verkon solmut. Heidän mallissaan verkko jaettaisiin ryhmiin, jotka sisältävät kolmen tyyppisiä solmuja. Alimman tason verkon solmut ilmoittaisivat verkon rakenteen muutoksista oman alueensa ylimmän tason solmulle. Ylin taso ilmoittaisi muutokset muille ylimmän tason solmuille sekä oman alueensa keskitason solmuille. Keskitason solmu lähettäisi tiedon ensimmäiselle hallinnoimalleen alimman tason solmulle. Alimman tason solmut lähettäisivät tiedon aina seuraajalleen, kunnes saavutaan keskitason solmun hallinnoiman alueen loppuun. Rakenteen muutos on esitetty kuvassa 3.

Tehtyjen mittausten mukaan tällä tavalla olisi mahdollista luoda verkko, joka skaalautuisi noin miljoonaan solmuun asti siten, että solmujen verkkoon liittyminen ja poistuminen ei aiheuttaisi kohtuuttomasti liikennettä. [14]

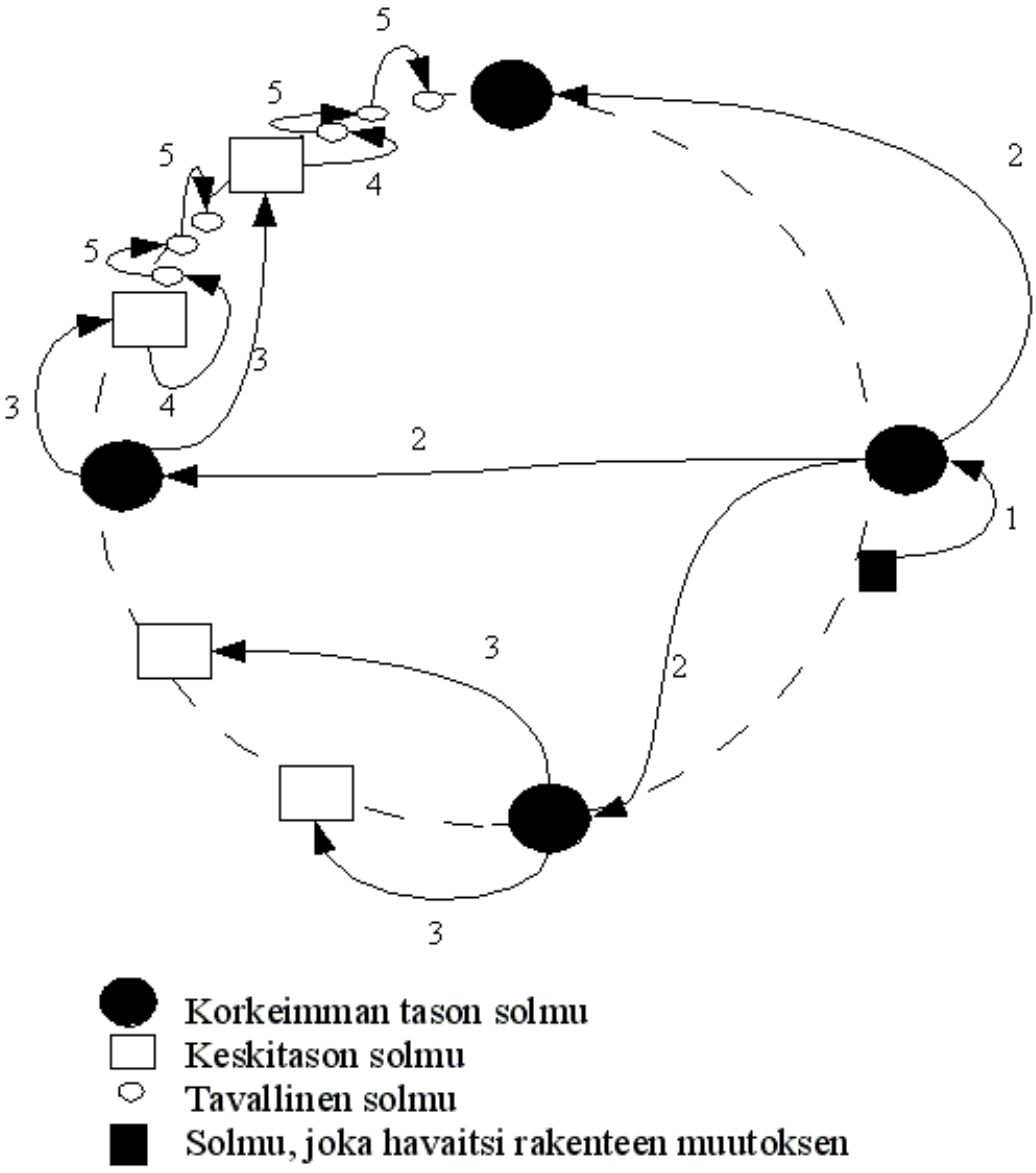
Hajautettua hakuindeksiä (distributed hash table, DHT) käyttävien verkkojen toiminta perustuu tiedosta otettuihin hash-arvoihin. Käytetyn hash-funktion hyvä jakautuvuus on DHT-verkon toiminnan kannalta tärkeätä, jottei verkkoon muodostuisi solmuja, jotka ovat vastuussa suuren osan verkon indeksoinnista.

Tällä hetkellä käytössä olevissa DHT verkoissa ei ole mahdollista tehdä kyselyä, joka perustuu tiedostojen osittaisiin nimiin, mutta hakutoiminnon toteutuksesta on luotu malleja, joiden käyttäminen on mahdollista nykyisten DHT-toteutusten päällä. Näitä menetelmiä ei kuitenkaan vielä pidetä tarpeeksi skaalautuvina [17].

DHT-verkoissa haku saadaan toteutettua pienellä pakettimäärällä siten, että pahimman tapauksen, jossa tietoa ei löydy verkosta, kaistankulutus on logaritmisesti verrannollinen verkon kokoon. Jos tiedosto löytyy verkosta saadaan vastaus nopeasti ja pienellä pakettimäärällä.

5 Yleisesti käytettävien verkkojen toteutukset

Tässä luvussa kerrotaan yleisesti käytössä olevien vertaisverkkojen rakenteesta ja niissä käytettävien hakualgoritmien toiminnasta. Keskuspalvelimen sisältävistä verkoista kuvataan BitTorrent [25] -verkon toiminta. Järjestämättömistä verkoista esitetään Gnutella [15], Freenet [16] sekä KaZaAn käyttämä FastTrack[9]. DHT-verkoista esimerkkinä esitetään Chord-kirjaston [23] toiminta.



Kuva 3: Rakenteen muutos Guptan, Liskovin ja Rodriguesin mallissa.

5.1 BitTorrent

BitTorrent [25] on Bram Cohenin kehittämä GPL-lisenssin [13] alla julkaistu keskuspalvelimia sisältävä vertaisverkko, joka on saavuttanut suuren suosion. CacheLogicin tekemien mittausten [6] mukaan yli puolet vertaisverkkoliikenteestä on BitTorrent-liikennettä.

BitTorrent-verkko sisältää trackeriksi kutsuttuja keskuspalvelimia, joihin käyttäjät

yhdistyvät ja ilmoittavat jakamiensa tiedostojen hash-summat. Kun solmu haluaa kopioida tiedostoa, lähettää se pyynnön trackerille, joka lähettää sellaisten solmujen IP-osoitteet ja porttinumerot, joista kyseisen tiedoston kopiointi on mahdollista. Solmut lähettävät trackerille tiedon kopioinnin edistymisestä, ilmoittaen trackerille lähetetyn ja vastaanotetun tiedon määrän.

Tiedostojen kopiointi perustuu BitTorrent-verkossa *.torrent*-tiedostoihin, jotka sisältävät SHA1-algoritmilla [21] laskettuja tiedostojen hash-arvoja. Kukin tiedosto jaetaan pieniin palasiin, joiden koko on materiaalin verkkoon laittavan käyttäjän määritettävissä. Internetissä levitettävät *.torrent* tiedostot sisältävät kunkin palan ja koko tiedoston SHA1-tarkistussumman. Koska tiedostosta tiedetään useita hash-arvoja, voidaan tiedoston vioittunut pala havaita nopeasti ja kopioida uudestaan. [5]

5.2 Gnutella

Gnutella [15] on yksi ensimmäisistä täysin hajautettuun malliin perustuvista vertaisverkoista, mutta sen käyttö on edelleen aktiivista. Verkon solmun liittyessä Gnutella-verkkoon muodostaa se yhteyden vähintään yhteen asiakasohjelmassa ennalta määrättyyn solmuun. Ensimmäisissä Gnutella-standardeissa verkon kaikki solmut olivat tasa-arvoisia, ja pitivät yllä tietoa ainoastaan itse jakamistaan tiedostoista. Kaistankäytön vähentämiseksi myöhemmin ilmestyneissä Gnutella-standardeissa on otettu käyttöön ultrapeer-tekniikka [4].

Alkuperäisessä Gnutella-verkossa haku verkossa perustuu tulvimiseen siten, että solmu lähettää hakupaketin kaikille naapurisolmuilleen. Tähän pakettiin merkitään TTL-arvo, jota jokainen paketin vastaanottanut solmu vähentää yhdellä. Jos TTL-arvo on nollaa suurempi, lähettää hakupaketin vastaanottanut solmu sen eteenpäin solmuille, joihin se on yhdistynyt. Jos solmulla on hakupakettia vastaava tiedosto, lähettää se vastauksen samaa reittiä pitkin. Hakupaketit sisältävät haun yksilöivän id-kentän, jonka avulla solmu voi hylätä haun saadessaan sen useasta eri lähteestä.

Käyttämällä TTL-arvoa 6 ja lähettämällä paketti aina kuudelle verkon solmulle saadaan tällä hyvin yksinkertaisella hakualgoritmilla etsittyä tietoa yli 8000:sta solmusta. Tämä on kuitenkin ainoastaan teoreettinen maksimi, koska sama hakupaketti voi tulla solmuun useita eri reittejä pitkin. Gnutella-verkko koostuukin useasta erillisestä TTL-arvon määrittämistä verkoista. [15]

5.3 Freenet

Freenet-järjestelmää kehitettäessä painotettiin verkon käyttäjien anonymiteettiä. Tämän seurauksena Freenet-verkosta on tullut Internetin käyttöä rajoittavissa maissa

hyvin suosittu. Anonymiteetti on Freenet-järjestelmään toteutettu siten, että tietoa reititetään kohteeseen useiden eri verkon solmujen kautta. Tämän seurauksena tiedon alkuperäisen lähettäjän selvittäminen on vaikeaa. Liikenne on salattua, ja protokolla tukee versiointia siten, että alkuperäisen salaisen avaimen tietävä henkilö pystyy päivittämään tiedoston sisällön.

Freenet-järjestelmässä ainoa tapa vastaanottaa haluamansa tieto on tietää sen hash-arvo, eikä järjestelmä sisällä erillistä tiedostojen nimiin perustuvaa hakumahdollisuutta. Verkkoon liittyneet solmut eivät myöskään voi itse valita tietoa, joka kyseisestä solmusta on mahdollista kopioida, vaan verkon solmut toimivat yhtenä suurena tiedostojärjestelmänä, jossa kaikki tieto on hajautettu ja monistettu useiden solmujen kesken.

Asiakasohjelmissa valitaan Freenetin käyttöön annettavan levytilan määrä ja tämä levytila täyttyy verkossa liikkuvasta liikenteestä. Solmu tallentaa tähän tilaan osan solmun läpi kulkeneesta Freenet-liikenteestä. Kun käyttäjän valitsema levytila on kokonaan käytetty, korvataan uudella tiedolla sitä tietoa, jonka pyytämisestä on kulunut pisin aika.

Haku järjestelmässä perustuu hash-arvoihin. Kukin solmu pitää yllä indeksiä solmun läpi kulkeneista hakupyynnöistä ja niihin saaduista vastauksista. Solmun vastaanottaessa hakupyynnön, tarkistaa solmu ensimmäisenä, että pystyttäisiinkö tietoa kopioimaan kyseisestä solmusta. Jos tiedon kopioiminen on mahdollista, lähettää solmu pyydetyn tiedon. Jos kopiointi ei ole mahdollista, valitsee solmu hakuja tallentavasta indeksistä, että mikä solmuista olisi paras lähde kyseiselle tiedolle ja lähettää pyynnön tälle solmulle. Jokaisen uudelleenlähetyksen yhteydessä vähennetään hakupakettiin liitettyä TTL-arvoa yhdellä. Jos TTL-kentän arvoksi tulee nolla lähetetään paketti, joka ilmoittaa kyselyn epäonnistuneen. Saadessaan tiedon kyselyn epäonnistumisesta lähettää solmu kyselypaketin indeksinsä mukaan seuraavaksi parhaalle lähteelle. Kielteinen vastaus lähetetään kehän muodostumisen välttämiseksi, myös mikäli havaitaan, että solmu vastaanottaa saman kyselyn useampaan otteeseen.

Kun solmu on vastaanottanut haluamansa tiedon, purkaa se sen salauksen materiaalin alunperin verkkoon laittaneen ilmoittamaa julkista avainta käyttämällä.

Freenet-verkon toiminta ei rajoitu ainoastaan tiedostojen levittämiseen. Protokollan päälle onkin toteutettu esimerkiksi anonyymejä keskustelualueita. Myös sensuurin sulkemien sivujen levittäminen Freenet-protokollan avulla on yleistä (tätä toimintaa kutsutaan nimellä Freesites). [16]

5.4 FastTrack

FastTrack on Sharman Networksin [24] hallinnoima tiedostojenvaihtoon keskittynyt verkko. Verkon solmut tunnistautuvat keskitetyille autentikointipalvelimille, jonka jälkeen ne toimivat Gnutellan protokollasta laajennetun protokollan mukaisesti. Autentikointipalvelimien käyttämisen seurauksena verkko sisältää järjestelmän, jonka sammuttaminen kaataa koko verkon toiminnan.

FastTrack verkkoon voidaan liittyä useilla asiakasohjelmistoilla, joista suosituin on Sharman Networksin ylläpitämä KaZaA. Sharman Networks on kehittänyt myös suositun VoIP-ohjelmiston, Skypen, jonka toiminta vastaa FastTrack järjestelmää.

FastTrack järjestelmässä käytetään Ultrapeer-tekniikkaa siten, että solmut on jaettu kahden eri tyypin solmuihin: tavallisiin solmuihin ja supernodeiksi kutsuttuihin solmuihin. Solmun liittyessä verkkoon ilmoittavat autentikointipalvelimet sille supernode solmun, johon se yhdistyy. Supernodejen tarkoituksena on toimia hakujen paikallisina indeksoijina ja keskustella muiden supernodejen kanssa. Mistä tahansa nopean prosessorin ja nopean tietoliikennekaistan sisältävästä verkon solmusta voi tulla supernode.

Käyttäjän yhdistyessä supernodeen se lähettää jakamiensa tiedostojen tiedot supernodelle. Hakupyynnöt lähetetään supernodelle, joka tutkii, että jakaako jokin supernoden alueella olevista solmuista kyseistä tiedostoa. Alkuperäisessä hakupaketissa asetetaan TTL-kentän arvoksi seitsemän, ja jokainen supernode, jonka kautta paketti kulkee vähentää arvoa yhdellä. Jokainen supernode on yhdistynyt keskimäärin viiteen toiseen supernodeen, joille hakupyyntö lähetetään niin kauan kuin TTL-kentän arvoksi tulee nolla. Hakupaketin saaneet supernodet vastaavat supernodeen liittyneiden solmujen tuloksilla (katso kuva 2 sivulla 6).

Tiedostojen etsiminen perustuu tiedostonnimeen, mutta tiedostojen kopiointi tapahtuu tiedostosta otetun hash-arvon perusteella, jonka supernode ilmoittaa hakutuloksien yhteydessä. Samaa tiedostoa voidaan hash-funktioiden seurauksena kopioida useammasta lähteestä. FastTrack verkossa käytetty hash-järjestelmä (UUHash) [26] on kuitenkin puutteellinen, jonka seurauksena vihamielinen solmu voi lähettää viallisen tiedoston osan, jonka virheellisyyttä hash-järjestelmä ei havaitse, koska vioittuneen tiedoston ja alkuperäisen tiedoston tarkistussumma on sama.

FastTrack-verkossa kulkeva data on Freenetin tavoin salattu, mutta data liikkuu suoraan tiedoston alkuperäisen lähteen ja tiedostoa kopioivan solmun välillä. FastTrack-verkon käyttämä salaus on kuitenkin pystytty selvittämään, jonka seurauksena kolmannen osapuolen on mahdollista selvittää siirretty tieto. Salauksen toiminnnan selvittäminen on mahdollistanut myös FastTrack-protokollaa käyttävien Open Source ohjelmistojen kehittämisen. [9]

Suositun VoIP-ohjelmisto Skypen toiminta perustuu FastTrack-verkosta laajennettuun protokollaan. Skypessä on otettu käyttöön vahva AES [2] salaus ja mahdollistettu kahden palomuurilla suojatun solmun väliset yhteydet. Kahden palomuurilla suojatun solmun välinen keskustelu on toteutettu siten, että Skype-verkosta etsitään solmu, jonka tietoliikennekaista ei ole aktiivisessa käytössä ja johon pystyy muodostamaan yhteyden palomuurin takaa. Molemmat osapuolet ottavat yhteyden tähän solmuun, ja tieto reititetään osapuolten välillä käyttäen kolmatta solmua tiedonvälitykseen. Kolmas osapuoli ei kuitenkaan salauksen ansiosta pysty selvittämään siirrettyä dataa. [3]

5.5 Chord

Chord [23] on MIT:ssä kehitetty kirjasto, joka tarjoaa ohjelmoijalle rajapinnan DHT-järjestelmään, jossa avaimen perusteella tiedosta vastaavan solmun löytäminen tapahtuu $\theta(Log(N))$ paketilla, jossa N kuvaa verkon kokoa. Chord-kirjasto ei siis tarjoa täydellistä vertaisverkon toteutusta. Chord kirjastoa käyttäen on luotu esimerkiksi Cooperative File System, jota käyttämällä on mahdollista toteuttaa hajautettu tiedontallennusjärjestelmä Linux- ja Unix-järjestelmiin [7].

Chord käyttää solmun tunnistamiseen IP-osoitteesta SHA1-algoritmilla [21] muodostettua tarkistussummaa, jonka seurauksena IP-osoitteen täytyy yksilöidä verkon solmu (NAT:n käyttäminen ei ole mahdollista). Chordin käyttämä protokolla edellyttää myös yhteyden muodostamista solmuun, jonka seurauksena Chordin käyttämä portti ei saa olla suojattu palomuurilla.

Chord-järjestelmässä solmut muodostavat järjestetyn rengasmaisen topologian, jossa solmun paikka määräytyy IP-osoitteesta otetun SHA1-tarkistussumman mukaan. Kukin solmu tuntee edellisen ja seuraavana renkaassa olevan solmun ja pienen määrän muita verkkoon liittyneitä solmuja.

Hakuindeksi hajautetaan verkossa siten, että ensimmäinen solmu, josta tiedoston tietoja on mahdollista saada on se solmu, jonka IP-osoitteen tarkistussumman on pienempi tai yhtäsuuri kuin etsittävän avaimen SHA1-tarkistussumma modulo 2^n, jossa n vastaa avaimen pituutta. Jos solmu ei pysty määrittämään suoraan sitä solmua, joka vastaa tiedoston indeksoinnista, lähettää se hakupyynnön sille tuntemalleen solmulle, joka on reititystietojen perusteella lähinnä kyseisen avaimen indeksoijasolmua.

Uuden solmun liittyessä verkkoon ilmoitetaan sille sen hallinnoiman hakuindeksin osien tiedot. Kun solmu poistuu verkosta, ilmoittaa se poistumisestaan edeltäjä- ja seuraajasolmuilleen siirtäen hallinnoimiensa hakuindeksin osien hallinnan näille solmuille. Reititystaulujen korjaaminen tehdään ajastetusti ja silloin kun havaitaan, että solmuun ei saada muodostettua yhteyttä. [23]

6 Yhteenveto

Tutkielmassa on esitetty erilaisia vertaisverkkojen hakualgoritmeja. Kaikkia tutkielmassa esitetyistä hakualgoritmeista ei olla vielä käytetty julkisesti saatavilla olevissa vertaisverkoissa.

Tutkielmassa kerrotaan lyhyesti järjestäytyneistä vertaisverkoista. On kuitenkin todennäköistä, että järjestäytyneet verkot syrjäyttävät järjestäytymättömät verkot tiedostojenvaihtoon keskittyvissä vertaisverkoissa nopeiden hakutoimintojen ansiosta. Muunnoksen tapahtuminen kuitenkin edellyttää skaalautuvien tiedostonimiin perustuvien hakujen saamista osaksi DHT-järjestelmiä.

Viitteet

[1] Lada A Adamic, Rajan M. Lukose, Amit R. Puniyani ja Bernardo A. Huberman, *Search in power law networks*, Physical review e volume 64 2001, saatavilla pdf-muodossa osoitteesta http://www.hpl.hp.com/research/idl/papers/plsearch/pre46135.pdf

[2] Advanced Encryption Standard (AES), saatavilla pdf-muodossa osoitteesta http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf,

[3] Salman A. Baset ja Henning Schulzrinne,*An analysis of the Skype Peer-to-Peer Internet Telephony protocol*, 2004, saatavilla pdf-muodossa osoitteesta http://www1.cs.columbia.edu/ library/TR-repository/reports/reports-2004/cucs-039-04.pdf

[4] *Bearshare technical faq*, saatavilla html-muodossa osoitteessa http://www.bearshare.com/help/faqtechnical.htm#10, viitattu 4.7.2005.

[5] *BitTorrent - Protocol*, saatavilla html-muodossa osoitteesta http://www.bittorrent.com/protocol.html, viitattu 13.7.2005

[6] *CacheLogic - P2P Traffic Analyses*, saatavilla html-muodossa osoitteesta http://www.cachelogic.com/research/slide1.php, viitattu 14.7.2005

[7] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris ja Ion Stoica, *Wide-area cooperative storage with CFS*,Symposium on Operating System Principles 2001, saatavilla pdf-muodossa osoitteesta http://pdos.csail.mit.edu/papers/cfs:sosp01/cfs_sosp.pdf.

[8] Distributed Hash Table, saatavilla html-muodossa osoitteesta http://en.wikipedia.org/wiki/Distributed_hash_table, viitattu 6.7.2005.

[9] *FastTrack - Wikipedia, the free encyclopedia*, saatavilla html-muodossa osoitteesta http://en.wikipedia.org/wiki/FastTrack, viitattu 2.7.2005.

[10] *GIMPS The Great Internet Prime Search*, saatavilla html-muodossa osoitteessa http://www.mersenne.org/, viitattu 13.7.2005

[11] Christos Gkantsidis, Melena Mihail ja Amin Saberi, *Random Walks in Peer-To-Peer Networks*, Infocom 2004, saatavilla pdf-muodossa osoitteesta http://www.ieee-infocom.org/2004/Papers/03_4.PDF

[12] *Hash collision*, saatavilla html-muodossa osoitteesta http://encyclopedia.thefreedictionary.com/Hash+collision, viitattu 30.6.2005

[13] GNU General Public License, saatavilla html-muodossa osoitteesta http://www.gnu.org/copyleft/gpl.html

[14] A. Gupta, B. Liskov, and R. Rodrigues. Efficient routing for peer-to-peer overlays, NSDI 2004, saatavilla pdf-muodossa osoitteesta http://pmg.lcs.mit.edu/ anjali/nsdi-onehop.pdf

[15] Gene Kan, *Gnutella*, *Peer to Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly, 2001, s. 94–122.

[16] Adam Langley, *Freenet*, *Peer to Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly, 2001, s. 123–132.

[17] Jinyang Li, Boon Thau Loo,Joseph M. Hellerstein, M. Frans Kaashoek, IPTPS 2003, saatavilla pdf-muodossa osoitteesta http://www.cs.berkeley.edu/ boonloo/papers/search_feasibility.pdf

[18] Nelson Minar ja Marc Hedlund, *A Network of Peer: Peer-to-Peer Models Through the History of the Internet,Peer to Peer: Harnessing the Power of Disruptive Technologies*,2001, s.8-20

[19] Marius Portmann, Aruna Seneviratne, Cost-effective Broadcast for Fully Decentralized Peer-to-peer Networks, Euronetlab seminar 2002, saatavilla pdf-muodossa osoitteesta http://www.euronetlab.net/seminar/aruna_paper.pdf

[20] Nima Sarshar, P. Oscar Boykin, Vwani P. Roychowdhury, *Percolation Search in Power Law Networks Making Unstructured Peer-To-Peer Networks Scalable*, 4th IEEE International Conference on Peer-to-Peer Computing 2004, saatavilla pdf-muodossa osoitteesta http://femto.org/p2p2004/papers/sarshar.pdf

[21] The Internet Society,*RFC 3174 - US Secure Hash Algorithm 1 (SHA1)*, 2001, saatavilla txt-muodossa osoitteesta http://www.ietf.org/rfc/rfc3174.txt

[22] William Stallings, *Data And Computer Communications*, Prentice Hall, 1997, s. 520-522, 621.

[23] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, ja Hari Balakrishnan, *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*, SIGCOMM 2001, saatavilla pdf-muodossa osoitteesta http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf

[24] Sharman Networks, saatavilla html-muodossa osoitteessa http://www.sharmannetworks.com/

[25] *The Official Bittorrent Homepage*, saatavilla html-muodossa osoitteesta http://www.bittorrent.com/

[26] UUHash, saatavilla html-muodossa osoitteesta http://en.wikipedia.org/wiki/UUHash, viitattu 2.7.2005.

[27] Beverly Yang ja Hector Garcia-Molina,*Improving Search in Peer-to-Peer Systems*, International Conference on Distributed Computing Systems 2002, saatavilla pdf-muodossa osoitteessa http://www-db.stanford.edu/ byang/pubs/p2psearch_long.pdf