Määrittelydokumentti

DHT – Distributed Hash Table

Helsinki 7.4.2004

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Marko Räihä
Risto Saarelma
Antti Salonen
Tuomas Toivonen
Tomi Tukiainen
Simo Viitanen

Asiakas

Jussi Lindgren

Johtoryhmä

Juha Taina
Turjo Tuohiniemi

Kotisivu

`http://www.cs.helsinki.fi/group/dht/`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
0.9	22.2.2004	Työstöversio
1.0	27.2.2004	Tarkastettava versio
1.1	5.3.2004	Uudelleentarkastettava versio
1.2	7.4.2004	Lopullinen versio

Sisältö

1 Johdanto

Tässä dokumentissa esitetään Kademlia-algoritmiin perustuvaan DHT-ohjelmistoon kohdistuvat toiminnalliset vaatimukset ja määritellään sen tarjoamat toiminnot. Tämän lisäksi dokumentissa pyritään määrittelemään mahdollisimman tarkasti myös ohjelmistoon kohdistuvat laadulliset vaatimukset ja toiminnallisuus, jota ohjelmiston pohjana käytettävältä GNUnetilta odotetaan.

Tämä dokumentti liittyy kevään 2004 ohjelmistotuotantoprojekti-kurssiin. Dokumentti on tarkoitettu projektiryhmän, asiakkaan ja ohjelmistotuotantoprojekti-kurssin henkilöstön käyttöön.

1.1 Terminologiaa

Seuraavassa taulukossa on selitetty lyhyesti dokumentissa käytettävää terminologiaa:

DHT	Distributed Hash Table, hajautettu hajautustaulu
GNUnet	avoin sovelluskehys turvallisille vertaisverkkoratkaisuille (http://www.ovmj.org/GNUnet/)
Kademlia	eräs kirjallisuudessa [MM] kuvattu DHT-toteutus
Vertaisverkko	Tasa-arvoisista solmukoneista koostuva verkko, P2P-verkko

1.2 Dokumentin rakenne

Luvussa 2 annetaan kokonaiskuva tuotettavasta ohjelmistosta. Luvussa 3 esitetään ohjelmistoon kohdistuvat toiminnalliset vaatimukset. Luvussa 4 määritellään ohjelmiston tarjoamat toiminnot. Luvussa 5 määritellään DHT-moduuliin kohdistuvia laadullisia vaatimuksia. Luvussa 6 kerrotaan GNUnetiin kohdistuvista vaatimuksista ja luvusssa 7 määritellään testauksessa käytettäviltä apuvälineiltä vaadittuja ominaisuuksia.

2 Kokonaiskuva

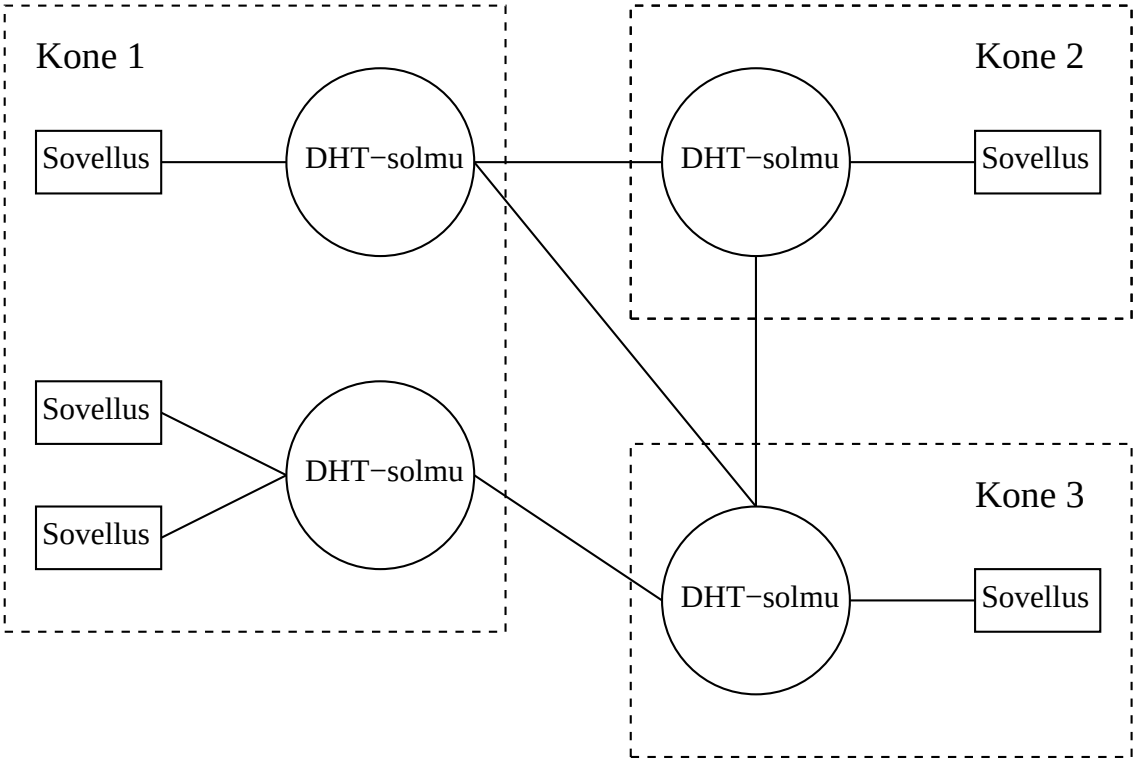
Tämä luku sisältää kuvauksen toteutettavasta ohjelmistosta. Kuvaus on korkealla tasolla ja painopiste on järjestelmäarkkitehtuurissa. Yksittäisten komponenttien määrittelyyn paneudutaan tarkemmin dokumentin muissa luvuissa.

2.1 DHT

Projektin tarkoituksena on tuottaa vertaisverkkoon perustuva toteutus hajautetusta hajautustaulusta (distributed hash table, DHT). Hajautettu hajautustaulu tarjoaa samat perustoiminnot kuin normaali hajautustaulu, mutta mahdollistaa taulun käsittelystä aiheutuvan kuorman jakamisen useammalle koneelle, ja jos tauluun tallennettua dataa monistetaan useampaan solmuun, voidaan hajautetulla hajautustaululla parantaa luotettavuutta.

Toteutettava DHT-verkko koostuu mielivaltaisesta joukosta tasa-arvoisia solmuja (jatkossa "DHT-solmu"), joista jokainen tarjoaa saman toiminnallisuuden. Kukin DHT-solmu on ohjelmaprosessi, ja niitä voi siten sijaita useampia yhdessä verkon fyysisessä palvelimessa. DHT-verkon toimintalogiikan kannalta solmujen sijaitsemisella samassa fyysisessä palvelimessa tai aliverkossa ei ole merkitystä. DHT-solmujen välinen viestintä perustuu etäproseduurikutsuihin, ja koska solmujen välinen verkko voi olla julkinen, on DHT-solmujen välinen viestintä salattua.

DHT-solmu tarjoaa toteuttamansa toiminnallisuuden ohjelmointirajapinnan muodossa. Ohjelmointirajapinnan avulla voidaan rakentaa erilaisia DHT-verkkoa hyödyntäviä sovelluksia. DHT-verkkoa hyödyntävät sovellukset toimivat verkossa kytkeytymällä asiakas-palvelinsuhteeseen yhden DHT-solmun kanssa. Sovelluksen ja sitä palvelevan solmun oletetaan sijaitsevan vähintäänkin samassa turvallisessa aliverkossa, sillä näiden välinen viestintä on suojaamatonta. Oheisessa kaaviossa on kuvattu verkon fyysisten koneiden, DHT-solmujen, niihin kytkeytyneiden asiakkaiden suhteet.



Tässä projektissa toteutettava toiminnallisuus sisältää <avain, arvo>-parien lisäyksen ja haun, mutta ei poistoa.

2.2 GNUnet

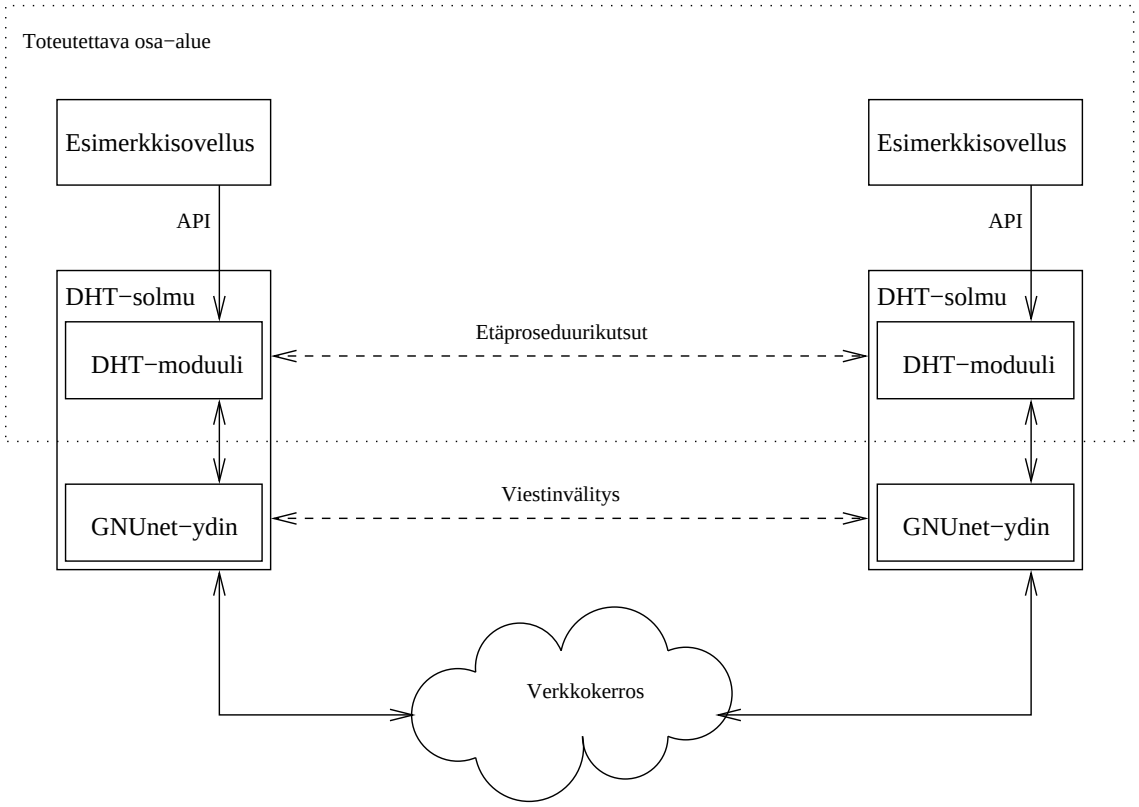
DHT-solmujen toteutuksen pohjana on GNUnet, joka on yleinen sovelluskehys vertaisverkkoa hyödyntäville sovelluksille. DHT-solmujen vaatima uusi toiminnallisuus toteutetaan GNUnetin osaksi siten, että toteutettavasta ohjelmasta tulee yksi GNUnetin moduuli (jatkossa "DHT-moduuli"). Työnjako DHT-moduulin ja GNUnetin ytimen välillä on ensisijaisesti se, että GNUnet tarjoaa DHT-moduulille sen toiminnallisuuden, millä kaksi DHT-verkon mielivaltaista solmua kommunikoivat keskenään. Lisäksi GNUnet tarjoaa yleisiä palveluita mm. muistinhallinnan ja rinnakkaisuuden toteuttamiseen.

2.3 Esimerkkisovellus

GNUnetin DHT-moduulin lisäksi projektissa toteutetaan yksi DHT-moduulia hyödyntävä sovellus, joka demonstroi moduulin tarjoaman APIn käyttöä moduulin tuleville käyttäjille. Esimerkkisovellus on yksinkertainen tiedostojen jakamiseen tarkoitettu ohjelma. Toissijainen, mutta tämän projektin kannalta keskeisempi rooli esimerkkisovelluksella on järjestelmätestauksessa.

2.4 Yhteenveto

Oheisessa kaaviossa on kuvattu kahden solmun toimintaa DHT-verkossa. Kaaviossa nuolet kuvaavat kutsujen suuntaa. Katkoviivalla piirretyt nuolet esittävät loogisia yhteyksiä, normaalit nuolet fyysisiä. Huomattavaa on, että ainoastaan kaavion ylälaidassa oleva rajattu alue, joka sisältää olevat osa-alueet "Sovellus"ja "DHT-moduuli"kuuluvat projektissa toteutettaviksi. Verkkokerroksen toiminnasta ei DHT-moduuli tiedä mitään, sillä solmujen välinen kommunikaatio tapahtuu GNUnetin sille tarjoamilla palveluilla.



3 Toiminnalliset vaatimukset

Tässä luvussa käsitellään tuotettavalle ohjelmistolle asetettavat toiminnalliset vaatimukset.

3.1 Vaatimukset yleisellä tasolla

DHT-moduulin täytyy tarjota ohjelmointirajapinta, jonka kautta moduulia käyttävät ohjelmat voivat käyttää verkkoon hajautettuja hajautustauluja. Moduulin tulee myös tarjota toteutus käytettäville hajautustauluille. Tällaisen hajautustaulun täytyy olla toiminnaltaan hyvin samankaltainen kuin tavallisen lokaalin hajautustaulun: siihen täytyy voida tallentaa, ja siitä täytyy voida hakea dataa avaimien avulla. Suurimpana erona lokaaliin hajautustauluun on se, että hajautustaulun sisältö on hajautettu usealle verkossa sijaitsevalle solmulle. DHT-verkossa useat käyttäjät voivat tehdä hajautustauluun avainhakuja ja tallennuksia samanaikaisesti ja DHT-verkko osaa reitittää haut oikeille solmuille.

Kullakin DHT-solmulla on datakerros, joka huolehtii yhden solmun tietojen säilytyksestä. DHT ei ota kantaa datakerroksen toteutukseen, vaan määrittää abstraktin rajapinnan jonka solmun käyttämä datakerros tulee toteuttaa. Rajapinta määrittelee myös datakerroksen oletettu käyttäytyminen virhetilanteissa.

Hajautustaulun hajauttaminen verkkoon aiheuttaa yhden implisiittisen vaatimuksen: olemassa olevaan hajautustauluun täytyy jotenkin voida liittyä ja siitä täytyy voida myöskin erota, muutenhan kaikki hajautustaulut olisivat käytännössä hajautettuja ainoastaan yhteen verkon solmuun.

Lisäksi moduulin käyttäjän täytyy tietysti voida perustaa hajautustauluja ja tieto niistä on hyvä voida antaa järjestelmän puitteissa muille DHT-solmuille, jotta ne voivat liittyä perustetun hajautustaulun solmuiksi ilman järjestelmän ulkopuolista tiedonsiirtoa. Toisaalta tiedon antamisen ulkopuolisille on oltava estettävissä, ettei kaikkia tauluja täydy pitää julkisesti näkyvillä.

Jotta datan määrä hajautustaulussa ei ainoastaan kasvaisi, täytyy järjestelmään toteuttaa myös datan vanheneminen, jonka seurauksena vanha data häviää taulusta.

Seuraavassa on vielä listattu DHT:hen kohdistuvat toiminnalliset vaatimukset yleisellä tasolla

- Tarjoaa ohjelmointirajapinnan, jonka kautta pääsee käyttämään verkkoon hajautettuja hajautustauluja
  - useiden hajautustaulujen käytön samanaikaisesti on oltava mahdollista
  - useiden sovelluksien täytyy voida käyttää DHT-moduulia samanaikaisesti
  - uusia tauluja täytyy voida luoda
  - olemassaolevaan tauluun täytyy voida liittyä
  - liitytystä taulusta täytyy voida erota

- Tarjoaa hajautetulle hajautustaululle hyödyllistä toiminnallisuutta
  - luodusta taulusta täytyy voida antaa tietoa sen ulkopuolisille verkon solmuille
  - luodusta taulusta on kuitenkin oltava mahdollista tehdä privaatti, jolloin tieto taulun olemassaolosta täytyy siirtää järjestelmän ulkopuolisesti
  - tauluun tallennettu data vanhenee

- Tarjoaa hajautustauluille tyypillisen toiminnallisuuden
  - taulusta täytyy voida tehdä avainhakuja
  - tauluun täytyy voida tallentaa <avain, arvo>-pareja
  - näiden täytyisi toimia käyttäjän näkökulmasta samalla tavalla kuin lokaaleissa hakutauluissakin

- Vikasietoisuus
  - taulujen on mahdollistettava vikasietoisuus, jonka astetta voidaan säätää taulukohtaisesti
  - säätäminen hoidetaan Kademlian määrittämällä redundanssikertoimella

4 Toiminnot

Tässä kappaleessa kuvataan se, millaisia toimintoja toteutettavan ohjelmiston on suunniteltu tarjoavan aiemmin kuvattujen vaatimusten täyttämiseksi.

Ohjelmisto jaetaan kolmeen pääkokonaisuuteen, jotka ovat:

- Ohjelmiston sisäinen toteutus
- Ohjelmointirajapinta DHT-palveluiden käyttöön
- Esimerkkisovellus

4.1 Ohjelmiston sisäinen toteutus

Ohjelmisto toteutetaan sisäisesti käyttäen Kademlia-algoritmia ja algoritmin toteuttamiseen välittömästä tarvittavia tietorakenteita ja menetelmiä. Seuraavassa on listattu tärkeät sisäiset ratkaisut, joihin on päätetty ottaa kantaa jo määrittelyvaiheessa:

- toteutuksessa ei tehdä oletuksia koskien avaimien kokoa
- toteutuksessa ei tehdä oletuksia koskien datayksiköiden kokoa
- datan tallentaminen ja lataaminen solmun sisäisesti toteutetaan abstraktin rajapinnan kautta
- solmujen väliseen kommunikointiin käytetään GNUnetin tarjoamia palveluita

4.1.1 Protokolla

Solmujen välisessä kommunikoinnissa käytetävä protokolla on Kademlia-algoritmin määrittelemä protokolla, joka koostuu neljästä etäproseduurikutsusta. Nämä ovat:

- PING
- STORE
- FIND_NODE
- FIND_VALUE.

Näitä koskevat vaatimukset on selitetty Kademliaa koskevassa artikkelissa [MM].

4.1.2 Datakerros

Datakerros on yhden DHT-solmun osa, jonka avulla DHT-moduuli hallinnoi solmuun paikallisesti tallennettuja <avain, arvo>-pareja. DHT-moduuli ei ota kantaa datakerroksen toteutukseen vaan määrittää rajapinnan, jonka kautta moduuli käyttää datakerrosta. Tästä seuraa, että DHT-verkossa olevat solmut voivat kukin käyttää eri toteutusta datakerroksesta.

Datakerroksen rajapinnalle määritettävät palvelut ovat:

- uuden taulun luonti
  - datakerros varaa uuden taulun, johon moduuli voi tallentaa <avain, arvo>-pareja
  - kutsussa määritellään kuinka monta arvoa voi yhdelle avaimelle tallentaa.
- <avain, arvo>-parin tallennus
  - datakerros tallentaa <avain, arvo>-parin kutsussa määritettyyn tauluun
  - avaimelle voi tallettaa useampia arvoja, mutta jos tallennettavien arvojen maksimimäärä on saavutettu, datakerros palauttaa virheen
  - datakerros välittää kutsujallelle viitteen talletettuun <avain, arvo>-pariin
    - viitettä tarvitaan <avain, arvo>-parin poistamiseen
- arvojen haku avaimen perusteella
  - hakee kaikki avaimeen liittyvät arvot datakerroksesta ja palauttaa kutsujalle listan tallennusviitteitä
- arvon poisto avaimen ja tallennusviitteen perusteella
  - poistaa yhden <avain, arvo>-parin datakerroksesta
- taulun poisto

Rajapintaan määritellään yhtenäinen virhetilanteiden käsittely. Virhetilanteet, jotka datakerros voi viestittää DHT-moduulille ovat:

- haettaessa arvoa talletettu data on korruptoitunut
- taululle varattu tallennustila on loppunut
- tauluun talletettujen arvojen määrä annetulle avaimelle on ylittänyt raja-arvon
- muu virhe

4.2 Ohjelmointirajapinta

Ohjelmointirajapinnan avulla sovellus voi hyödyntää DHT-moduulin tarjoamaa toiminnallisuutta. Tästä johtuen moduulin tarjoama API-rajapinta on keskeisessä asemassa määrittelyä tehtäessä. Rajapinnan avulla täytyy voida käyttää kaikkia toiminnallisissa vaatimuksissa määriteltyjä toimintoja.

Ohjelmointirajapinta määritellään tarjoamaan seuraavat palvelut:

`create`	luo uuden hajautustaulun
`join`	liittää solmun olemassaolevaan hajautustauluun
`leave`	erottaa solmun hajautustaulusta
`insert`	lisää <avain, arvo>-parin hajautustauluun
`fetch`	hakee annettua avainta vastaavat datat hajautustaulusta
`dropInsertedData`	poistaa itse julkaistun <avain, arvo>-parin hajautustaulusta
`listInsertedData`	listaa kaikki itsejulkaistut <avain, arvo>-parit
`listTables`	tuottaa listan kaikista hajautustauluista, joihin solmu on liittynyt

4.2.1 `create`

`create`-palvelu luo paikalliseen solmuun uuden hajautustaulun. Palvelu tarvitsee 2 parametria:

- `table_metadata` - Sisältää kaiken luotavaan tauluun liittyvän metadatan.
- `table_config` - Sisältää kaiken luotavaan tauluun liittyvän konfiguraatiotiedon.

Taulun luonnin yhteydessä taulun luonut solmu liittyy automaattisesti tauluun. Luonnin jälkeen solmu voi käyttää hajautustaulun haku- ja tallennuspalveluita.

Hajautustaulun luonnin yhteydessä taululle määräytyy tunniste (table_id).

4.2.2 `join`

`join`-palvelu liittää solmun olemassa olevaan hajautustauluun. Palveluun määritellään 2 parametria:

- `address` - toisen, tauluun jo liittyneen, solmun osoite
- `table_id` - tunniste, jonka avulla tauluun jo liittynyt solmu tunnistaa liityttävän taulun

Liittymisen yhteydessä liittyjäsolmulle toimitetaan taulun metadata ja konfiguraatiotiedot. Liittymisen jälkeen uudelle solmulle voidaan (Kademlia-algoritmin mukaisesti) siirtää osa taulun alkioista. Liittymisen jälkeen tauluun liittynyt solmu voi käyttää hajautustaulun haku- ja tallennuspalveluita.

4.2.3 `leave`

`leave`-palvelu erottaa solmun hajautustaulusta ja tarvitsee yhden parametrin:

- `table_id` - tunniste, jolla solmu tunnistaa taulun

Taulusta eroava solmu voi eroamisen yhteydessä hävittää vastuullaan olleet hajautustaulun <avain,arvo>-parit. Eroamisen jälkeen eroajasolmu ei voi enää käyttää hajautustaulun haku- ja tallennuspalveluita.

4.2.4 `insert`

`insert`-palvelu tallentaa <avain,arvo>-parin hajautustauluun ja tarvitsee 3 parametria:

- `table_id` - sen taulun tunniste johon tallennus suoritetaan
- `key` - avain jolla tallennettu data voidaan myöhemmin hakea hajautustaulusta
- `value` - arvo joka tallennetaan hajautustauluun

Tallentamisen yhteydessä <avain,arvo>-pari tallennetaan hajautustauluun myöhempiä hakuja varten. Solmut pitävät kirjaa itse tallentamistaan <avain,arvo>-pareista.

4.2.5 `fetch`

`fetch`-palvelu hakee kaikki avaimeen liittyvät data-alkiot hajautustaulusta ja tarvitsee 2 parametria:

- `table_id` - sen taulun tunniste josta haku suoritetaan
- `key` - avain jolla haku suoritetaan

Hakemisen yhteydessä hakupalvelun kutsujalle palautetaan lista datayksiköitä, jotka liittyvät hakutaulussa annettuun avaimeen.

4.2.6 `dropInsertedData`

`dropInsertedData`-palvelu poistaa solmun itse tallentaman <avain,arvo>-parin hajautustaulusta. Palvelu tarvitsee 2 parametria:

- `table_id` - sen taulun tunniste johon poisto suoritetaan
- `key` - avain johon talletettu arvo poistetaan

Poiston ei tarvitse poistaa ko. <avain,arvo>-paria hajautustaulusta välittömästi. Riittää, että ko. tieto on kokonaan poistunut hajautustaulusta 24 tunnin kuluttua.

4.2.7 `listInsertedData`

`listInsertedData`-palvelu listaa kaikki solmun määrättyyn hajautustauluun itse tallentamat <avain,arvo>-parit. Palvelu tarvitsee yhden parametrin:

- `table_id` - sen taulun tunniste johon poisto suoritetaan

Hakupalvelun kutsuja saa listan tauluuntallentamistaan <avain,arvo>-pareista.

4.2.8 `list_tables`

`list_tables`-palvelu hakee tiedot kaikista tauluista joihin tietty solmu on liittynyt, ja tarvitsee yhden parametrin:

- `address` - sen solmun osoite jolta tiedot haetaan

Hakemisen yhteydessä hakupalvelun kutsuja saa listan taulujen tunnisteista ja niihin liittyvistä metadatoista. Sovellusohjelma voi hyödyntää tätä tietoa mm. tauluihin liittymiseen.

4.3 Esimerkkisovellus

Esimerkkisovellus on tarkoitettu DHT-moduulin testaamiseen ja toimivaksi esimerkiksi sovellusohjelmasta, joka hyödyntää DHT-moduulin tarjoamaa ohjelmointirajapintaa. Esimerkkisovellus on yksinkertainen tiedostojen jakamiseen tarkoitettu ohjelma. Sovelluksella on graafinen käyttöliittymä.

Esimerkkisovelluksen ei tarvitse toimia kaikilla eri alustoilla, joilla GNUnet sekä DHT tulevat toimimaan, mutta sen tulee toimia ainakin Linux/i386 alustalla.

DHT:llä toteutettavan hajautetun hajautustaulun pitää sisältää tiedot jaettavina olevista tiedostoista (tiedoston nimi ja sen kuvaus avainsanoilla) ja linkit joiden kautta ne ovat noudettavissa järjestelmän ulkopuolelta.

Esimerkkisovellus toteuttaa seuraavat toiminnot:

- liittyminen jo olemassaolevaan tiedostojen jakoverkkoon
- uuden tiedostojen jakoverkon luominen
- tiedostonimilistan hakeminen avainsanoilla (esim. ”kekkonen”, ”promootio”, jne.)
- hakuosoitelistan hakeminen tiedoston nimellä
- tiedostonsiirron käynnistäminen tiedoston nimen ja hakuosoitteen avulla
- tiedoston lisääminen verkkoon
  - tiedoston nimi, avainsanajoukko ja osoite, mistä se on noudettavissa

  - jos johonkin avainsanajoukossa olevaan avainsanaan liittyy hyvin monta tiedostonnimeä, sallitaan se, ettei tiedoston nimeä liitetä mainittuun avainsanaan

- hakutulosjoukkojen selailu
  - avainsanojen avulla haetun tiedostonnimilistan selaaminen
  - tiedoston nimen avulla haetun osoitelistan selaaminen
- poistuminen verkosta

5 Laadulliset vaatimukset

Tässä luvussa tarkastellaan DHT-ohjelmistoon kohdistuvia laadullisia vaatimuksia. Tällaisia vaatimuksia ovat muunmuassa talletettujen avaimien löytymistodennäköisyydet ja ohjelmointirajapinnan selkeys. Yleisesti ottaen kaikki DHT-ohjelmistoon kohdistuvat vaatimukset, joita ei ole mainittu toiminnallisien vaatimuksien yhteydessä, kuuluvat laadullisten vaatimusten joukkoon.

5.1 Ohjelmakoodin laadulliset vaatimukset

Koska DHT-toteutuksemme liitetään GNUnetiin, ohjelmakoodin on luonnollisesti oltava GNUnetissa käytetyn ohjelmointityylin mukaista. Lisäksi DHT-toteutuksen käyttäjät muodostavat kansainvälisen ryhmän, jolloin suomenkielisen ohjelmakoodin tuottaminen olisi este DHT:n järkevälle käytölle. Tästä johtuen ohjelmakoodin on oltava englanninkielistä. On selvää, että DHT:n on lisäksi tarkoituksenmukaista tarjota mahdollisimman selkeä API palveluidensa käyttöön.

Seuraavassa on listattuna DHT:n ohjelmakoodiin liittyvät laadulliset vaatimukset:

- Koodin on oltava GNUnet-tyylistä
- Koodin on oltava englanninkielistä
- DHT:n tarjoaman API:n täytyy olla selkeä

5.2 DHT:n toiminnallisuutta koskevat laatuvaatimukset

Tässä aliluvussa tarkastellaan DHT:n tarjoamaan toiminnallisuuteen liittyviä laadullisia vaatimuksia, joihin toimintojen määrittely ei itsessään ota kantaa.

5.2.1 Siirretyn datan oikeellisuus

Yleisesti ottaen DHT:n täytyy pystyä tarjoamaan luotettava datansiirtopalvelu. Seuraavat vaatimukset määrittelevät, mitä luotettavuudella tarkoitetaan tässä yhteydessä.

- Tallentaminen: lähdesolmussa oleva sovelluskerros tallentaa dataa DHT:hen
  - Tässä yhteydessä luotettavuudella tarkoitetaan sitä, että DHT:hen tallennettava data ei muutu matkalla lähdesolmun sovelluskerroksesta kohdesolmujen datakerrokseen. Yksittäinen data, jonka lähdesolmun sovelluskerros DHT:hen tallentaa, annetaan kohdesolmujen datakerroksille yksittäisenä datana, joka on täsmälleen alkuperäisen kaltainen.
- Lataaminen: kohdesolmussa oleva sovelluskerros lataa dataa DHT:stä

  - Tässä yhteydessä luotettavuudella tarkoitetaan sitä, että DHT:stä ladattava data ei muutu matkalla lähdesolmun datakerroksesta kohdesolmun sovelluskerrokseen. Yksittäinen data, joka saadaan lähdesolmun datakerroksesta, annetaan kohdesolmun sovelluskerrokselle yksittäisenä datana, joka on täsmälleen lähdesolmun datakerroksesta saadun kaltainen.

5.2.2 Säilytetyn datan oikeellisuus

DHT-toteutus ei voi taata datakerroksessa säilytetyn datan oikeellisuutta, mutta

- kunkin solmun datakerros on vastuussa datansäilytysvirheistään
  - datakerroksen oletetaan tuottavan virheilmoituksen, jos sinne tallennettu data on korruptoitunut
  - datakerros voi jättää virheellisen datan luovuttamatta, jolloin jokin muu lähdesolmu luovuttaa sen kohdesolmulle lataamisen yhteydessä
- jos solmussa oleva data korruptoituu ja datakerros ei luovuta sitä DHT:lle
  - DHT-toteutus voi yrittää hakea datan DHT:sta ja tallentaa sen uudelleen
  - DHT-toteutus voi poistaa dataan liittyvän avaimen omasta tallennettujen avainten listastaan

5.2.3 Palveluiden toimivuus

Ennen kuin asiakas pääsee hyödyntämään DHT:n varsinaista toiminnallisuutta, täytyy taulun solmujen välisen kommunikaation toimia kohtuullisesti. Tämän jälkeen asiakkaan täytyy päästä liittymään johonkin tauluun, jonka jälkeen DHT:n täytyy tarjota toimiva datan tallennus- ja latauspalvelu. Seuraavan listan kohdat ovat tähän liittyviä vaatimuksia.

- Solmujen välisten etäproseduurikutsujen täytyy toimia luotettavasti; mikäli asiakassolmu kutsuu palvelinsolmun etäproseduuria ja tiedonsiirto solmujen välillä toimii kohtuullisesti GNUnetin paketinvälityksen tasolla, täytyy tiedonsiirron solmujen välillä toimia kohtuullisesti myös DHT:n toteuttaman etäproseduurikutsu-abstraktion tasolla
  - GNUnetin paketinvälityksen tasolla tapahtuvat paketin katoamiset voivat hidastaa etäproseduurikutsu-abstraktion tasolla näkyvää toimintaa
  - GNUnetin paketinvälityksen tasolla tapahtuvien paketin katoamisten ylittäessä tietyn rajan, vähintään 15 % lähetetyistä paketeista, voi etäproseduurikutsujen tasolla oleva yhteys katketa kokonaan
- Tauluun liittymisen täytyisi toimia luotettavasti; mikäli asiakas yrittää liittyä tauluun sellaisen taulussa jo olevan solmun kautta, joka on toiminnassa ja johon asiakas saa yhteyden, täytyy liittymistapahtuman epäonnistumisen todennäköisyyden hyvin pieni

  - liittymisalgoritmin luotettavuuden täytyy olla vähintään Kademlia-algoritmin tasolla

- Datan tallentamisen DHT:hen pitäisi toimia luotettavasti; todennäköisyys sille, että tallennustapahtuma päättyy virheellisesti, vaikka syöte on hyväksyttävä, on oltava lähellä nollaa.
  - tallennusalgoritmin luotettavuuden täytyy olla vähintään Kademlia-algoritmin tasolla

- Datan lataamisen DHT:sta pitäisi toimia luotettavasti; mikäli tauluun on tallennettu lataamisessa käytetty avain, on lataustapahtuman virheellisen päättymisen todennäköisyyden oltava lähellä nollaa.
  - latausalgoritmin luotettavuuden täytyy olla vähintään Kademlia-algoritmin tasolla

5.2.4 Palveluiden tehokkuus ja skaalautuvuus

DHT-palveluun kohdistuu suuria odotuksia skaalautuvuuden suhteen. Tämä tarkoittaa käytännössä sitä, että DHT-tauluun liittyneiden solmujen määrä ei saa vaikuttaa suuresti palvelun vasteaikoihin. Lisäksi palvelun odotetaan olevan kohtuullisen tehokas, tarkoittaen sitä, että toimintojen vasteajat eivät alun perinkään saa olla kovin suuret. Seuraavassa listassa on täsmennetty tähän liittyviä laadullisia vaatimuksia.

- Tauluun liittymisen tehokkuus
  - liittymisalgoritmin aikavaativuusluokan tulee olla korkeintaan sama kuin Kademlia-algoritmissa
  - tässä solmun katsotaan liittyneen tauluun, kun se voi alkaa suorittaa hakuja
    - kaiken datansiirron ei tarvitse olla ohi

- Tauluun tallettaminen ja taulusta hakeminen
  - tallennus- ja hakualgoritmien aikavaativuusluokan tulee olla korkeintaan sama kuin Kademlia-algoritmissa, eli suhteessa korkeintaan verkon solmujen määrän logarimtiin $O(\log(n))$
  - tässä operaation katsotaan olevan valmis, kun varsinaisen datan siirto alkaa

- Taulusta eroaminen
  - eroamisalgoritmin aikavaativuusluokan tulee olla korkeintaan sama kuin Kademlia-algoritmissa
  - taulusta eroamisen katsotaan olevan loppunut, kun kaikki datansiirto DHT-tauluun loppuu

6 Vaatimukset GNUnet-alustalta

DHT-moduuli toteutetaan GNUnet-alustan päälle, ja tämän alustan on toteutettava joitakin moduulin vaatimia palveluita. Oleellisin vaatimus on tiedonsiirtopalvelut eli GNUnetin on tarjottava jonkinlainen tiedonsiirtoyhteys verkon solmujen välille. Lisäksi käytetään GNUnetin tarjoamia säikeistykseen ja muistinhallintaan liittyviä palveluita.

6.1 Tiedonsiirto GNUnetin välityksellä

GNUnet mahdollistaa tiedonsiirron monilla eri protokollilla (mm. TCP, UDP ja SMTP). DHT-moduulin pitää toteuttaa luotettava tiedonsiirto GNUnetin tarjoaman epävarman toteutuksen yli, sillä kaikki GNUnetin tarjoamat tiedonsiirtopalvelut ovat epävarmoja. GNUnet siirtää ainoastaan vakiokokoisia datapaketteja, joten DHT-moduulin on paloiteltava vakiokokoa pidempi lähetettävä data riittävän pieniksi paloiksi ja yhdistettävä useana pakettina vastaanotetettu data. GNUnet siirtää datan aina kryptattuna.

6.2 GNUnetin säikeistyspalvelut

GNUnet tarjoaa palvelut säikeiden luomiseen sekä semafori- ja mutex-rakenteet. GNUnet määrittelee myös tiettyjä konventioita joiden tarkoituksena on estää lukkiumia koodissa. DHT-moduulissa käytetään GNUnetin tarjoamia säikeistyspalveluita ja noudatetaan GNUnetin dokumentaatiossa kuvailtuja konventioita säikeistettyyn ohjelmointiin.

6.3 Muita GNUnetin tarjoamia palveluita

GNUnetin kirjasto `libgnunet_util` tarjoaa esimerkiksi muistinhallintaan ja hajautukseen liittyviä palveluita. DHT-moduuli käyttää GNUnetin muistinvarauspalveluita ja hyödyntää mahdollisimman paljon GNUnetin hajautusfunktioita.

6.4 Yhteenveto

GNUnet-alusta näyttäisi tarjoavan tarvittavat palvelut DHT-moduulin toteuttamiseen. Oleellisin palvelu on solmujen välinen tiedonsiirto. Muistinhallintaan ja säikeistykseen liittyvissä palveluissa GNUnetin tarjoamat palvelut tarjonnevat C:n peruskirjastoja parempaa alustariippumattomuutta ja auttavat moduulin koodia toimimaan yhdessä muun GNUnetin kanssa.

7 Vaatimukset testaukselle

Yksikkötestauksessa pyritään Extreme Programming (XP) mallin mukaiseen yksikkötestauksen automatisointiin. Käytännössä siis pyritään toteuttamaan ensin testitapaukset ja vasta sen jälkeen tapaukset toteuttava ohjelmakoodi.

7.1 Vaatimuksia yksikkötestien työkaluille

- Testien laatimisen tulisi olla helppoa ja selkeää.
- Testit voitava ajaa yksinkertaisesti yhdellä komennolla.
- Testien tuloksena oltava numerisoituva raportti: koodi läpäisee x prosenttia testeistä, y prosenttia epäonnistui.
- Raportin on myös selvästi listattava epäonnistuneet testit ja testin epäonnistumisen syy.

Lähteet

MM Maymounkov, P. ja Mazières, D., Kademlia: A peer-to-peer information system based on the xor metric.