# Testausyhteenveto

DHT – Distributed Hash Table

Helsinki 27.5.2004

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 |
|---|---|---|
| 1.0 | 27.5.2004 | Luovutettu versio |

# Sisältö

# 1 Yleiskuva

DHT-projektin testauksessa pyrittiin mahdollisimman kattavaan testien automaatioon. Yksikkötestauksessa käytettiin Extreme Programming -malleissa käytettyjä ohjelmallisia testejä, jotka toteutettiin käyttäen xUnit-mallia. Järjestelmätestausta varten laadittiin joukko komentorivityökaluja, joilla oli mahdollista joustavasti testata DHT-ohjelmointirajapinnan eri toiminnallisuuksia.

Yksikkötestauksen malli oli toimiva ja käytetty CuTest-työkalu riittävän monipuolinen, mutta silti yksinkertainen käyttää. Aikataulun tiivistyessä loppua kohden jouduttiin yksikkötestausta ja osittaista järjestelmätestausta lomittamaan. Tuloksena osa yksiköiden toiminnallisuudesta testattiin osana järjestelmää sen sijaan, että kaikille yksiköille olisi laadittu kattavat yksikkötestit.

Järjestelmätestausta varten laadittiin joukko työkaluja.

**generate-conf.pl** Luo halutun määrän asetustiedostoja. Jokainen tiedosto laaditaan käyttäen samaa pohjaa, `gnunet-template.conf`. Erillisiä tiedostoja käytetään käynnistettäessä joukko GNUnet-solmuja. Jokaiselle solmulle annetaan samat binäärihakemistot, mutta esimerkiksi portit ja hakemistot ajonaikaiselle datalle.

**boot-network.pl** Käynnistää peer-to-peer -testiverkon eli halutun määrän GNUnet-solmuja käyttäen automaattisesti luotuja asetustiedostoja.

**stop-network.pl** Ajaa alas kaikki testiverkon GNUnet-solmut.

**distribute-hosts.sh** Jakaa kaikkien testiverkon GNUnet-solmujen tunnisteet kaikille muille testiverkon solmuille, jotta verkon solmujen välinen viestintä mahdollistuu.

Työkaluista laadittiin kaksi versiota. Toinen versio käynnisti testiverkon solmut saman fyysisen koneen sisällä eri TCP/UDP -porteissa. Työkalujen varianteilla puolestaan voitiin käynnistää testiverkko, jonka jokainen solmu oli erillisessä fyysisessä koneessa. Käytännössä testattessa ajettiin testiverkkoa yhden koneen sisällä.

Varsinainen järjestelmätestaus suoritettiin manuaalisesti ajettavilla komentorivityökaluilla, joista jokainen testasi yhtä DHT-ohjelmointirajapinnan toiminnallisuutta (funktiota). Suoritetut testit ja niiden tulokset kirjattiin wikiin. Kehittyneempää järjestelmätestausta varten laadittiin myös prototyyppi skripteistä, joilla komentorivityökaluista olisi voitu muodostaa sekvenssejä. Järjestelmän kokonaisuuden toiminnassa ei kuitenkaan päästy sille tasolle, että automatisoitujen sekvenssien ajamiselle vielä olisi ollut tarvetta.

Tämän dokumentin toisessa luvussa listataan ajetut järjestelmätestit ja niiden tulokset. Kolmannessa luvussa puolestaan listataan yksikkötestit ja niiden tulokset.

# 2 Järjestelmätestit

Järjestelmätestit suoritettiin käyttämällä testitarkoituksiin tuotettuja komentorivityökaluja. Jokainen työkalu vastasi yhtä asiakassovelluksille tarkoitetun DHT-ohjelmointirajapinnan funktiota. Komentorivityökalut ja niitä vastaavat funktiot on listattu taulukossa 1.

## 2.1 Testitulokset

### 2.1.1 dht-create, single node

**OK** Lokien perusteella uusi taulu luodaan datakerrokselle ja Kademliaan. Lokeissa mainittu taulun tunnistava hash palautetaan asiakassovellukselle. (toivotuo, 20.5.2004)

### 2.1.2 dht-create, multiple nodes

**N/A** Ei testattu.

### 2.1.3 dht-fetch, single node

**ERROR** Yritetään hakea paikalliseen tauluun lisättyä avain-arvo -paria. Lokien perusteella pari löytyy datakerroksesta. Client-server ilmaisee luovansa viestin, jossa yksi pari. dht-fetch kuitenkin corettaa ilmeisesti lukiessaan vastausta client-server -protokollan yli. (toivotuo, 20.5.2004)

**ERROR** dht_api::handleFetchResultMsg() - message size is invalid: 17 != 16 (szviitan 20.5.2004)

**OK** Korjattu dht_api.c:stä ja client-server.c:stä useita virheitä ja nyt toimii. (mjraiha 24.5.2204)

| **Komentirivityökalu** | **Funktio** |
|---|---|
| dht-create.c | create() |
| dht-join.c | join() |
| dht-leave.c | leave() |
| dht-list.c | listTables () |
| dht-insert.c | insert() |
| dht-fetch.c | fetch() |
| dht-inserted-drop.c | dropInsertedData() |
| dht-inserted-list.c | listInsertedData() |

Taulukko 1: DHT-ohjelmointirajapinnan ja -moduulin testaukseen käytetyt komentorivityökalut ja niiden vastaavuudet ohjelmointirajapinnan funktioihin.

#### 2.1.4 dht-fetch, multiple nodes

**ERROR** Ei voi testata ennenkuin insert toimii oikein (szviitan 20.5.2004)

#### 2.1.5 dht-insert, single node

**OK** Ensin luotu taulu (dht-create) ja liitytty siihen (dht-join). dht-insert'llä lisätty tauluun foo, bar. Lokien perusteella datakerros tallettaa avain-arvo -parin oikein paikalliseen tietovarastoon. (toivotuo, 20.5.2004)

**ERROR** Jokainen <avain, arvo>-pari pitäisi aina ensin tallettaa verkkoon ja jos se onnistui niin vasta sitten lokaaliin datalayeriin, sillä jos jokainen solmu tallettaa kamat aina ensin omaan datalayeriin niin dht:ssä voi loppujen lopuksi olla yhtä monta arvoa yhdelle avaimelle kuin solmuja verkossa. (arvojen määrä on rajoitettu TableConfigissa) (szviitan 20.5.2004)

#### 2.1.6 dht-insert, multiple nodes

**ERROR** Insert ei lähetä liikennettä verkon yli. (szviitan 20.5.2004)

#### 2.1.7 dht-join, single node

**OK** Taulu luotu ensin dht-create'lla. Sitten suoritettu dht-join. Lokien perusteella taulu löytyy Kademliasta. Asiakassovellukselle palautetaan handle liityttyyn tauluun eli käytännössä taulun tunnistava hash. (toivotuo, 20.5.2004)

#### 2.1.8 dht-join, multiple nodes

**ERROR** Toimii muuten hyvin, mutta TableMetaData ja TableConfig ei siirry (szviitan 20.5.2004)

#### 2.1.9 dht-leave, single node

**OK** (Ei kuvailua.)

#### 2.1.10 dht-leave, multiple nodes

**N/A** Ei testattu.

#### 2.1.11 dht-list, single node

**ERROR** Luodaan ensin kaksi taulua paikalliseen solmuun. Annetaan sen jälkeen dht-list. Lokien perusteella taulut löytyvät datakerroksesta ja annetaan client-server -kerrokselle. Vastaukset eivät kuitenkaan ole kunnollisia asiakassovelluksella. Taulujen hashit saadaan ilmeisen ok (oikea määrä ja sisältö), mutta taulujen nimet ovat kuraa ("satunnaisesta"muistiosoitteesta?). (toivotuo, 20.5.2004)

**OK** Pari pientä feelua oli viestin kirjoittamisessa. (ttukiain 23.5.2004)

#### 2.1.12 dht-list, multiple nodes

**ERROR** Kts. yllä oleva virhe. Sata liikkuu nodelta toiselle oikeassa muodossa. client-server ja dht_api välillä data kuitenkin muuttuu. (szviitan 20.5.2004)

#### 2.1.13 dht-inserted-drop, single node

**N/A** Ei testattu.

#### 2.1.14 dht-inserted-drop, multiple nodes

**ERROR** Poiston pitäisi propagoitua verkkoon. 24H viiveellä. (szviitan 20.5.2004)

#### 2.1.15 dht-inserted-list, single node

**ERROR** Message size is invalid: 55 != 54. (szviitan 20.5.2004)

**ERROR** Korjattu virheitä ja vastaus saadaan. Arvot vain on korruptoituneet. (mjraiha 24.5.2004)

# 3 Yksikkötestit

Yksikkötestit suoritettiin käytten automatisoituja testejä. Työkaluna käytettiin ns. xUnit-mallia C-kielelle soveltavaa CuTest -työkalua[1]. Taulukossa 2 esitetään yhteenveto testien tuloksista.

## 3.1 dht_test

Testaa hajautetun hajautustaulun ylläpitoon käytetyn Kademlia-algoritmin toteutusta.

**Testitiedosto** `.../src/dht/module/test/dht_test.c`

**Testiohjelma** `dht_test`

**Testattu koodi** `.../src/dht/module/dht.c`

**Tulokset**

```
start testing dht
doing testDhtCreate
done testDhtCreate
doing testDhtJoin
done testDhtJoin
doing testDhtLeave
doing testDhtInsert
done testDhtInsert
doing testDhtFetch
done testDhtFetch
doing testDhtTables
done testDhtTables
doing testDhtInserted
done testDhtInserted
doing testDhtDrop
# Results of dht_test:
..F....F
```

[1] `http://cutest.sourceforge.net/`

| Testiryväs | Testejä | Onnistuneita | Epäonnistuneita |
|---|---|---|---|
| dht_test | 8 | 6 | 2 |
| datalayer_test | 4 | 4 | 0 |
| util_test | 2 | 2 | 0 |

Taulukko 2: Yhteenveto yksikkötestien tuloksista.

```
There were 2 failures:
1) testDhtLeave: test/dht_test.c:124: expected <1> but was <153>
2) testDhtDrop: test/dht_test.c:216: expected <1> but was <153>

!!!FAILURES!!!
Runs: 8 Passes: 6 Fails: 2
```

## 3.2 datalayer_test

Testaa hajautettujen hajautustaulujen sisällön tallettamiseen käytetyn datakerroksen toimivuutta.

**Testitiedosto** `.../src/dht/module/test/datalayer_test.c`

**Testiohjelma** `datalayer_test`

**Testattu koodi** `.../src/dht/module/datalayer.c` ja `.../src/dht/module/datalay`

**Tulokset**

```
start
getSuite
testflags
new_DHT_LocalStoreUnit testdata
get_DataUnitFromStr testdata
flags: 3
flags: 2
flags: 0
flags: 1
test_datalayerTableCreation.
init store
create a table
testing names ’TAULU’ == ’TAULU’
testing hashes 12345 == 12345
iterate past end
doAssert def
doAssert set.iterator
test_datalayerDataInsertionAndIteration.
init store
create a table
new_DHT_LocalStoreUnit testdata
```

```
get_DataUnitFromStr testdata
store unit
store unit again
new_DHT_LocalStoreUnit testdata
get_DataUnitFromStr testdata
get iterator
new_DHT_LocalStoreUnit testdata
get_DataUnitFromStr testdata
iterate to next
new_DHT_LocalStoreUnit testdata
get_DataUnitFromStr testdata
testing hashes 6A3D0FD084D8FB74F6A06B84F45E2E7BBDD2DBA1 == 6A3D0FD084
iterate past end
doAssert unit
doAssert iterator
test_datalayerDataInsertionAndDeletion.
init store
create a table
store unit
new_DHT_LocalStoreUnit testdata
get_DataUnitFromStr testdata
store another different unit
new_DHT_LocalStoreUnit testdata
get_DataUnitFromStr testdata
iterate all units
iterate all units again
iterate all units and delete them
delete  unit
delete  unit
iterate past end
try get the units after deletion
delete table
try to search the table
....

OK (4 tests)
```

## 3.3 util_test

Testaa lähinnä client-server -rajapinnan käyttämien apurutiinien toimivuutta.

**Testitiedosto** `.../src/dht/module/test/util_test.c`

**Testiohjelma** `util_test`

**Testattu koodi** `.../src/dht/util/networking.c`

**Tulokset**

```
doing conversion test
51F4087E0A58C8910CE6A763C9A0723C6B32D763 == 51F4087E0A58C8910CE6A763C
51F4087E0A58C8910CE6A763C9A0723C6B32D763 == 000000000A58C8910CE6A763C
free value
free key
doing equality test
# Results of dht_util_test:
..

OK (2 tests)
```