# Suunnitteludokumentti

Linux Traffic Control-käyttöliittymä – Ryhmä paketti2

Helsinki 27.10.2004

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

**Kurssi**

581260 Ohjelmistotuotantoprojekti (6 ov)

**Projektiryhmä**

Fabian Fagerholm
Janne Johansson
Markku Manner
Niko Mikkilä

**Asiakas**

Jukka Manner

**Johtoryhmä**

Juha Taina
Marianne Korpela

**Kotisivu**

`http://www.cs.helsinki.fi/group/paketti2`

**Versiohistoria**

| Versio | Päiväys | Tehdyt muutokset |
|---|---|---|
| 1.0 | 27.10.2004 | Ensimmäinen versio |

# Sisältö

# 1 Johdanto

Paketti2-projekti kuuluu Helsingin yliopiston tietojenkäsittelytieteen laitoksen Ohjelmistotuotantoprojekti-kurssiin. Kyseessä on jatkoprojekti syksyn 2003 Paketti-ryhmän työlle. Yhteisprojektin tavoitteena on tuottaa ohjelmisto Linux-ytimen kaistanhallinta-asetusten hallinnointiin graafisessa muodossa. Järjestelmä tulee Helsingin yliopiston tietojenkäsittelytieteen laitoksen tutkimusryhmien käyttöön. Yliopisto julkaisee ohjelmiston GNU General Public License- [FSF91] tai GNU Lesser General Public License-lisenssin [FSF99] alaisuudessa.

## 1.1 Dokumentin tarkoitus

Suunnitteludokumenttiin on koottu suunnitteluvaiheessa tehdyt ohjelman toteutusta koskevat päätökset. Dokumentti toimii projektiryhmän ohjenuorana toteutusvaiheen aikana. Dokumentissa esitetään tuotettavan ohjelmiston uudet ominaisuudet sillä tarkkuudella, että toteutus on dokumentin pohjalta suoraviivaista.

Suunnittelussa on otettu huomioon määrittelydokumentissa [Pr04] luvussa 3 listatuista toiminnoista prioriteetille 1 ja tärkeimmät prioriteetille 2 merkityt toiminnot. Luvussa 5 on esitettyjen bugien korjaukset. Muista prioriteettien 2 ja 3 toimintojen toteuttamisesta päätetään aikataulun salliessa toteutusvaiheen aikana. Ohjelmistossa jo toteutettujen ominaisuuksien suunnitelusta voi lukea Paketti-ryhmän suunnitteludokumentista [Pr03a] ja lopullisesta toteutuksesta Paketti-ryhmän ylläpitodokumentista [Pr03b].

## 1.2 Dokumentin rakenne

Luvussa 2 esitellään ohjelmiston yleisarkkitehtuuri ja luvussa 3 ohjelmiston käyttämät tietorakenteet. Luvussa 4 kuvaillaan ohjelmiston käyttämää STCCP-protokollaa ja siihen tulevia muutoksia. Luku 5 kuvailee tarkemmin käyttölittymäkomponentin suunnittelua ja luku 6 palvelinkomponentin suunnittelua. Luvussa 7 on virheiden korjaukset. Lopuksi luvussa 8 on dokumentin termistöä ja sitä seuraa lähdeluettelo.

# 2 Ohjelmisto

Tämän luvun tarkoitus on esitellä tämän hetkistä Paketti-ryhmän toteutusta ja joitakin toteutettavia muutoksia. Ohjelmiston tarkemmasta toteutuksesta voi lukea Paketti-ryhmän suunnitteludokumentista [Pr03a] ja ylläpitodokumentista [Pr03b]. Paketti2-ryhmän määrittelydokumentista [Pr04] voi lukea tehtävät muutokset tämän hetkiseen ohjelmistoon. Määrittelydokumentista nähdään myös suunniteltujen toteutusten prioriteetit ja koodit. Kyseisiä koodeja käytetään suunnitteludokumentissa selkeyttämään minkä asian toteutuksesta puhutaan.

## 2.1 Ohjelmiston yleiskuvaus

Ohjelmiston toiminnallisuus on hajautettu kahteen pääkomponenttiin. Palvelinkomponentti tarjoaa XML-rajapinnan Linux-ytimen kaistanhallinta-asetuksiin. Palvelinkomponentti on toteutettu C-kielellä. C-kielen kohtia tullaan muuttamaan ja asioita tullaan lisäämään C++-kielellä [Str97]. Käyttöliittymäkomponentti kytkeytyy tähän rajapintaan, ja mahdollistaa kohdejärjestelmän kaistanhallinta-asetuksien käsittelyn graafisessa muodossa. Käyttöliittymä on toteutettu kokonaisuudessaan Java-kielellä [SUN04].

Pääkomponentit liikennöivät keskenään Paketti-ryhmän suunnittelemalla sovellustason protokollalla Simple Traffic Control Configuration Protocol (STCCP). Protokollaa muutetaan luvussa 4 kuvatulla tavalla. Kaistanhallinta-asetukset siirtyvät XML-muotoisena komponenttien välillä.

Kuvassa 1 on tulevan Paketti2-ryhmän ohjelmiston yleisrakenne. Ohjelmiston palvelinkomponentti on kohdejärjestelmässä, jonka kaistanhallinta-asetuksia käyttäjä muokkaa hallintajärjestelmällä. Paketti-ryhmän toteutuksessa palvelinkomponentti koostuu pääkomponentista ja tc-rajapinnasta. Paketti2-ryhmä lisää palvelinkomponenttiin abstraktiotason, joka kytkee pääkomponentin tc-komentoon ja Linux-ytimen rajapintaan. Abstraktiotason tarkoitus on eristää palvelinkomponentin eri osat niin, että alla olevat kohdejärjestelmän rajapintojen mahdolliset muutokset eivät aiheuttaisi liian suurta muutostyötä ohjelmistossa. Abstraktiotaso mahdollistaa myös vaiheittaisen siirtymisen pois tc-komennon käytöstä Linux-ytimen rajapintojen suoraan käyttöön.

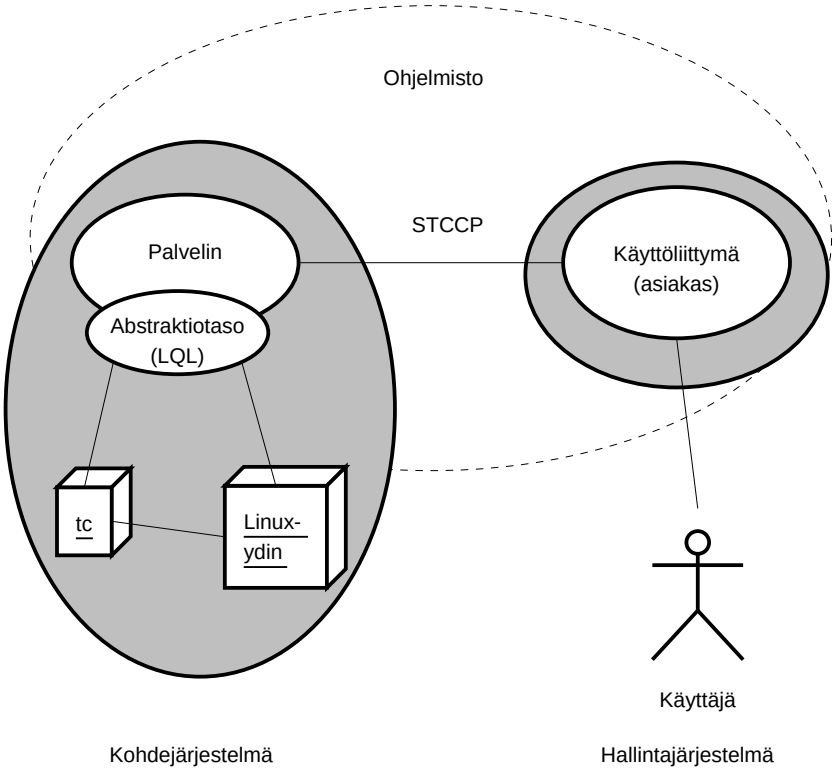
## 2.2 Paketti-sovelluksen käynnistys

Käyttäjä käynnistää paketti-sovelluksen antamalla sille parametreina kohdejärjestelmän osoitteen sekä portin, johon sovellus ottaa yhteyden. Mikäli käyttäjä antaa vain kohdejärjestelmän osoitteen, paketti-sovellus yrittää ottaa yhteyden vakio-arvona porttiin 20000. Vaihtoehtoisesti käyttäjä käynnistää ohjelman ilman parametrejä, jolloin käyttöliittymä käynnistyy ilman yhteyttä palvelinkomponenttin [KL3].

Ilman yhteyttä käyttöliittymällä voi muuttaa aluksi vain eth0-verkkorajapinnan asetuksia, jotka voi lähettää palvelinkomponenttiin vasta kun yhteys on avattu [KL3]. Olemassaolevien, esimerkiksi tiedostosta ladattujen tai yhteyden sulkemisen jälkeen käsiteltäviksi jäävien asetusten muokkaaminen on kuitenkin mahdollista myös muiden kuin eth0-rajapinnan osalta. Jos käyttäjä antaa käyttöliittymälle väärän osoitteen tai portin, paketti-sovellus ei saa yhteyttä palvelinkomponenttiin tai kohdejärjestelmä lähettää ERROR-viestin käyttöliittymälle, käynnistyy käyttöliittymä ilman yhteyttä.

## 2.3 Ohjelmiston toiminnot

Muunnos TC- ja XML-muotojen välillä [KL4, KL5] toteutetaan erillisinä työkaluina nykyisen ohjelmiston palvelinkomponentissa suoritettavan muunnoksen sijaan. Ratkaisun perusteena on palvelinkomponenttiin tehtävät muutokset, joiden seurauksena siellä oleva




Kuva 1: Ohjelmiston yleisrakenne.

TC-jäsennin on tarpeen vain kyseisen TC-XML-muunnoksen tekemiseksi. Pääkäyttäjäoikeuksin suoritettavan koodin määrän vähentämiseksi ja tietoliikenteen selkeyttämiseksi TC-jäsennin ja XSLT-muunnos XML dokumentista TC-muotoon siirretään erillisiksi työkaluiksi käyttöliittymäkomponentin yhteyteen.

Koska nykyinen TC-jäsennin on kirjoitettu C-kielellä, sen liittäminen suoraan käyttöliittymäkomponenttiin ei ole järkevää. Erillisenä työkaluna sen käyttö on kuitenkin kohtuullisen vaivatonta etenkin käyttäjille, jotka tarvitsevat TC-skriptien käyttömahdollisuutta. Toisaalta jäsentimeen joudutaan kirjoittamaan myös uutta koodia lisättävien jonomallien vuoksi. Mikäli aika riittää, työkalut pyritään kirjoittamaan suoraan Javalla. Tällöin niiden kutsuminen käyttöliittymästä onnistuu vaivattomasti ja suoritusympäristöstä riippumatta. Toteutettavien luokkien tulee kuitenkin olla käytettävissä myös erillisinä komentokehotteessa toimivina työkaluina, jotta niitä voidaan helposti kutsua skripteistä.

# 3 Tietosuunnitelma

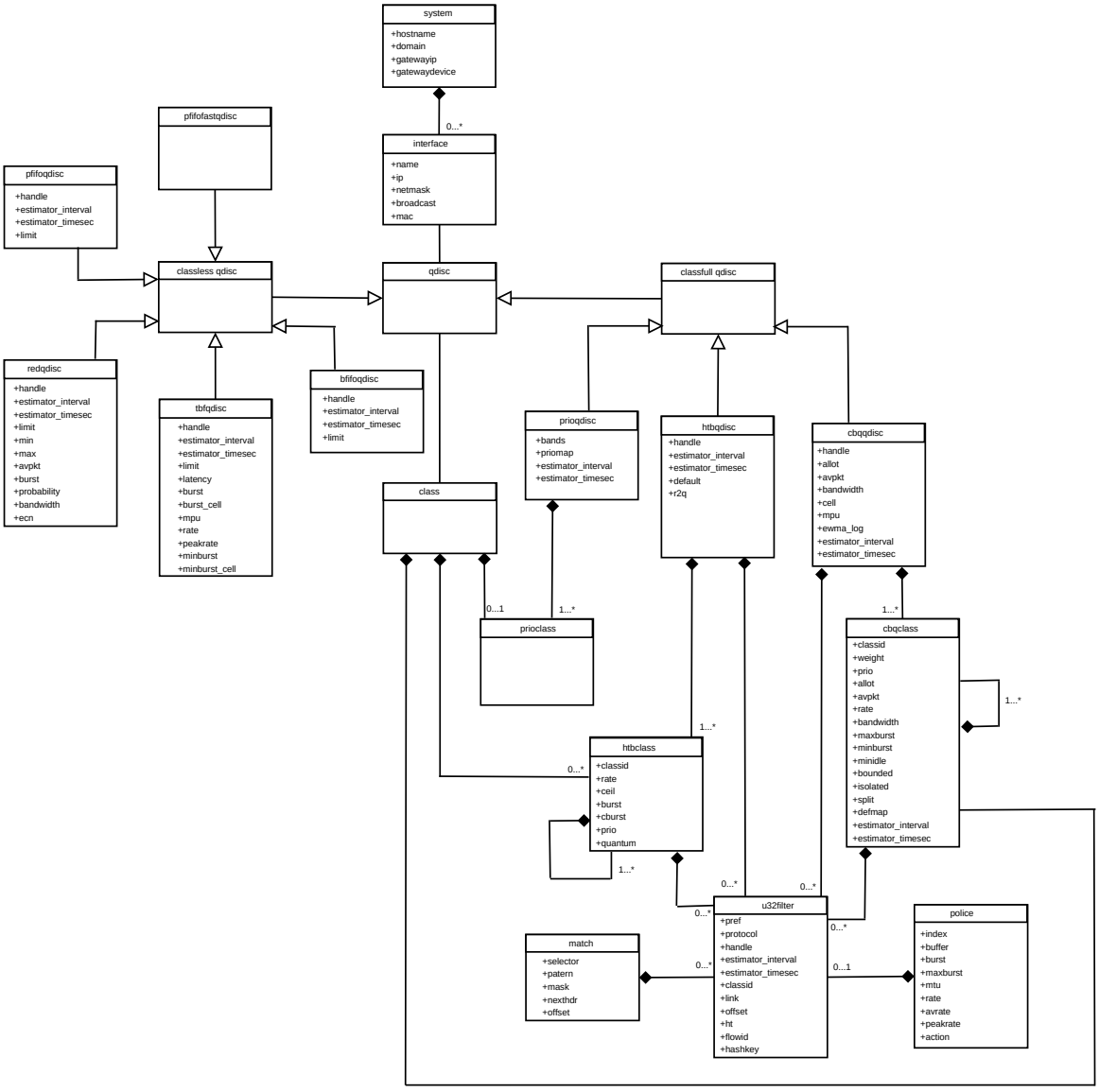
Järjestelmän kaistanhallinta-asetukset on kuvattu Paketti-ryhmän XML DTD:stä laajennetun XML Scheman avulla. Schema määrittelee ohjelmiston palvelin- ja käyttöliittymäkomponenttien väliseen kaistanhallinta-asetusten siirtoon käytettävän XML-tietorakenteen. Komponentit käsittelevät XML-rakennetta ja sen sisältämiä kaistanhallinta-asetuksia sisäisessa toiminnassaan eri tavoin. Kuva 2 esittää XML Schemaa kuvaavan UML-mallin.

Mallin System-luokka on rakenteen juuri, johon liittyy valinnainen määrä interface-luokkia. Interface-luokat ilmentyvät kohdejärjestelmässä eth0, eth1, eth2, ..., eth99. Jokaiseen tälläiseen käyttöön määrittelyyn liityntään on liitetty yksi jonomalli. Mikäli jonomalli on HTB, voidaan siihen liittää HTB-luokkia tai U32-suodattimia. Jos jonomalli on CBQ, siihen voidaan liittää CBQ-luokkia, ja PRIO-jonomalliin voidaan liittää PRIO-luokkia. Luokat voivat edelleen sisältää muita jonomalleja ja HTB:n ja CBQ:n tapauksessa suodattimia. Luokattomiin jonomalleihin ei voida lisätä muita luokkia.

U32-suodattimen valinnaisen tc-komentoparametrin selector sisältö kuvataan omina match-luokkinaan. Samoin valinnaisen police-luokan sisältö kuvataan omana luokkanaan police.

Palvelinkomponentti lukee XML-muotoiset kaistanhallinta-asetukset tietorakenteeseen, josta se edelleen syöttää ne Linux-ytimelle. Vastaavasti ytimeltä luettavat asetukset sijoitetaan tietorakenteeseen ja kirjoitetaan XML-muotoon.

Käyttöliittymäkomponentti muuntaa vastaanottamansa kaistanhallinta-asetukset käyttöliittymäkomponentteihin sidotuksi tietorakenteeksi, jota käytetään käyttöliittymän näkymän muodostamiseen sekä asetusten muokkaamisen mahdollistamiseen. Käyttöliittymä muuntaa tietorakenteen XML-muotoon, kun asetukset lähetetään palvelinkomponentille käyttöön otettavaksi kohdejärjestelmässä.




Kuva 2: Kaistanhallista asetusten UML-malli.

# 4 STCCP-protokolla

Simple Traffic Control Configuration Protocol (STCCP) on Paketti-ryhmän määrittelemä sovellustason protokolla kaistanhallinta-asetusten ja statistiikan siirtoon [Pr03a]. Protokolla perustuu asiakas-palvelin -mallin mukaiseen protokolla-arkitehtuuriin ja hyödyntää TCP/IP protokollaa. Protokolla on osittain XML-pohjainen.

Paketti2-ryhmä määrittelee protokollan uudelleen hieman tiukemmin. Edellisessä versiossa ei otettu kantaa protokollan XML-kehyksen DATA-elementin sisällöstä. Tässä elementin sisältö määritellään.

## 4.1 STCCP-viestin rakenne

STCCP-viesti koostuu kolmesta osasta:

- Otsikkokentistä,
- XML-kehyksestä ja
- XML-kehyksen sisällä olevasta XML-muotoisesta datasta.

Jälkimmäinen on valinnainen joissakin tilanteissa. STCCP-viestin rakenteen kuvauksessa käytetään seuraavia merkintöjä:

**LF** on rivinvaihto, heksadesimaalilukuna 0xA, desimaalilukuna 10.

**Kokonaisluku** on ei-negatiivinen kokonaisluku k, jolle pätee
$0 <= k < (2^{32} = 4294967296)$.

**Merkki** on mielivaltainen UTF-8 -merkistön merkki.

**Merkkijono** on mielivaltainen jono XML-syntaksin hyväksymiä merkkejä.

STCCP-viestit käyttävät Unicode UTF-8 -merkistöä [Ali04]. Viestien muoto on:

```
Protocol:STCCP/<kokonaisluku>LF
Length:<kokonaisluku>LF
<xml-kehys>
```

Protocol-otsikkokenttä ilmaisee protokollan versionumeron ja Length-otsikkokenttä ilmaisee XML-kehyksen pituuden.

## 4.2 XML-kehyksen rakenne

STCCP-viestin sisältö tallennetaan XML-muotoiseen viestikehykseen. XML-kehyksen DTD-kuvaus on seuraava:

```
<!ELEMENT MESSAGE ((SEQUENCE, ACK)?, TYPE, DATA)>
<!ELEMENT SEQUENCE (#CDATA)>
<!ELEMENT ACK (#CDATA)>
<!ELEMENT TYPE (#CDATA)>
<!ELEMENT DATA (#CDATA)>
```

XML-kehyksen elementtien sisältö ja merkitys on ensimmäisen version mukainen. DATA-elementtiä määritellään kuitenkin tarkemmin. Sen sisältö on

- Paketti-ryhmän määrittelemän XML-Scheman mukainen XML-dokumentti, jossa dokumentin merkit koodataan XML-standardin mukaisella tavalla, kun viestin tyyppi on COMMIT, tai
- virheilmoitus merkkijonona, kun viestin tyyppi on ERROR, tai
- tyhjä, kun viestin tyyppi on GET_SETTINGS_XML, GET_SETTINGS_TC, GET_STATISTICS_XML, GET_STATISTICS_TC tai QUIT.

REPLY-tyyppinen viesti on joko XML-dokumentti tai merkkijono, riippuen siitä minkä tyyppisen viestin vastaus on kyseessä. Tarkka kuvaus protokollan ensimmäisestä versiosta on Paketti-ryhmän suunnitteludokumentissa [Pr03a].

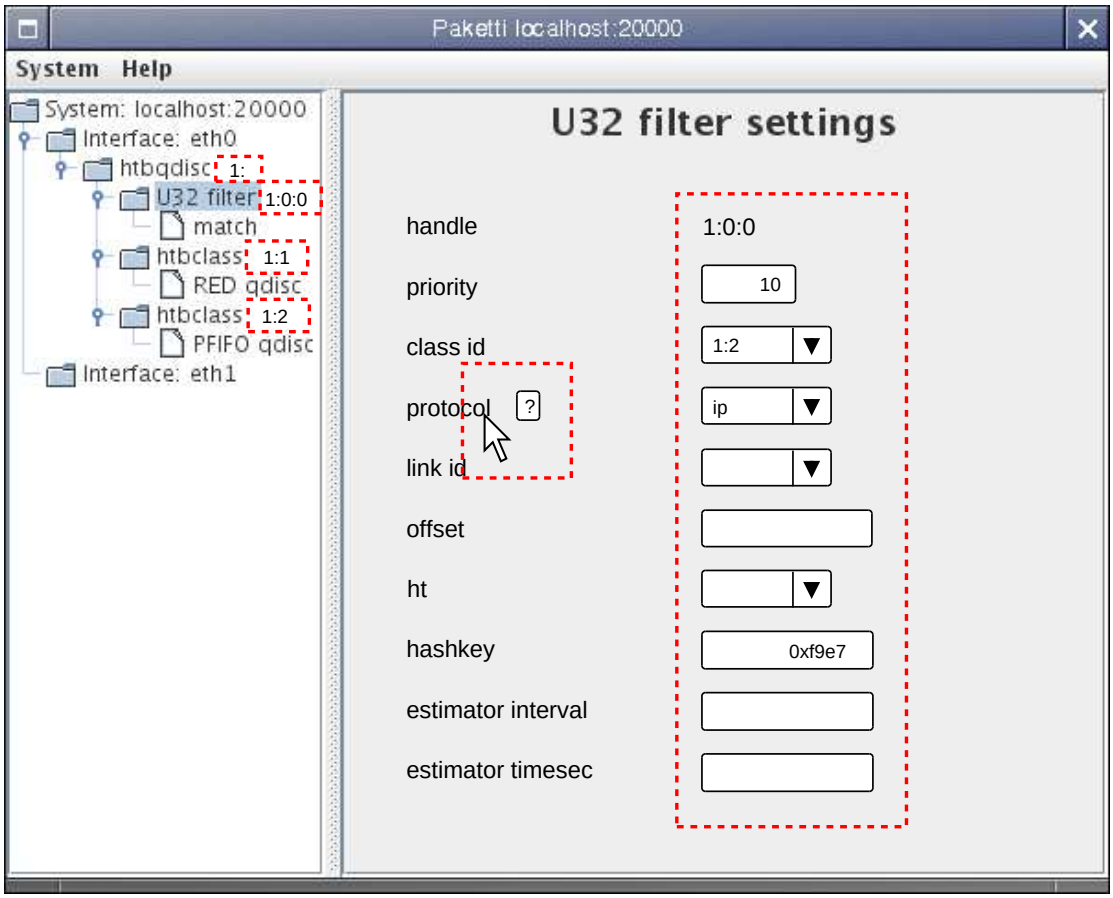
## 4.3 STCCP versio 2

Paketti2-ryhmä laajentaa STCCP-protokollaa siten, että se tukee paremmin U32-suotimen käyttöä [KL2], [S1]. Viestityypit TRANSFORM_TC_TO_XML ja TRANSFORM_XML_TO_TC poistetaan protokollasta, ja kyseinen toiminnallisuus siirtyy erilliseen työkaluun [KK2], [KK8]. STCCP-protokollan versionumero nostetaan 2:een.

# 5 Käyttöliittymäkomponentti

Tämä luku kuvaa käyttöliittymäkomponentin toimintojen suunniteltua toteutusta ja komponentin rakennetta. Paketti-ryhmän suunnitteludokumentin sivuilla 19-30 on kuvattu tarkasti nykyisen käyttöliittymän ulkoasu ja toimintojen käyttö [Pr03a]. Luvussa 5.1 käsitellään vain käyttöliittymän ulkoasuun tehtäviä muutoksia.

Paketti-ryhmä ei toteuttanut käyttöliittymän luokkarakennetta alkuperäisen suunnitelman mukaan, eikä nykyistä rakennetta ole dokumentoitu tarkasti missään Paketti-ryhmän dokumentissa. Luvussa 5.2 käydään läpi nykyinen rakenne kokonaisuutena sekä kuvataan siihen tehtävät muutokset.




Kuva 3: Käyttöliittymän ulkoasun muutoksia.

## 5.1 Käyttöliitymän ulkoasu

Osa käyttöliittymän ulkoasuun tehtävistä muutoksista on esitetty kuvassa 3, joka esittää kaistanhallinta-asetusten muokkaamista ohjelman pääikkunassa. Katkoviivalla merkityt kohdat ovat uusia tai niiden toiminnallisuutta laajennetaan. Nykyisen käyttöliittymän ulkoasu kuitenkin säilytetään pääosin.

### 5.1.1 Kaistanhallinta-asetusten attribuutit

Käyttöliittymässä parannetaan kaistanhallinta-asetusten attribuuttien käsittelyä [KL1, KL2]. Joissain tapauksissa attribuutilla on vain pieni määrä mahdollisia arvoja, jolloin niiden esittäminen listana alasvetovalikossa on suotavaa. Kuvassa 3 tällainen lista on esimerkiksi u32-suodattimen protokollavalinta. Lisäksi attribuuttikenttien pituus kuvaa niiden vaatiman syötteen pituutta. Kenttien pituus on kuitenkin rajoitettava ikkunan koon mukaan. Attribuuttikentissä numerot tasataan oikeaan reunaan ja teksti vasempaan.

Ohjelma varoittaa väärästä syötteestä [KL6], esimerkiksi attribuutin raja-arvon ylityksestä välittömästi kohdistuksen siirryttyä toiseen komponenttiin kirjoittamalla virheelli-

sen tekstikentän viereen tekstin “Error” tai piirtämällä virhettä osoittavan huutomerkki-ikonin. Tekstiä tai kuvaa painamalla saa auki ohjelman antaman virheilmoituksen. Virheilmoitus esitetään dialogi-ikkunassa. Kun virhe on korjattu, teksti tai ikoni poistetaan.

Elementin otsikon ja attribuuttien nimien vieressä on painikkeet, jotka avaavat kyseistä oliota koskevan kohdan kappaleessa 5.3 kuvatussa ohjejärjestelmässä. Kuvan 3 tavoin painikkeet voivat myös olla piilotetut, jolloin ne saa esiin osoittamalla ensin otsikkoa.

### 5.1.2 Elementtien väliset viittaukset

Eräs ongelma kaistanhallinta-asetusten puumallissa on viittaaminen puun elementistä muihin kuin sen vanhempiin ja lapsiin [KL1]. Tämä on tarpeen esimerkiksi haluttaessa määrittää mihin luokkaan jokin suodatin siirtää paketteja. Linux-ytimen kaistanhallintajärjestelmässä kullekin jonomallille, luokalle ja suodattimelle määritellään 32-bittinen viitenumero, “handle”. TC-komentokielessä tämä numero esitetään jonomalleilla muodossa 123:, luokilla muodossa 123:456 ja suodattimilla se on muotoa 123:456:789.

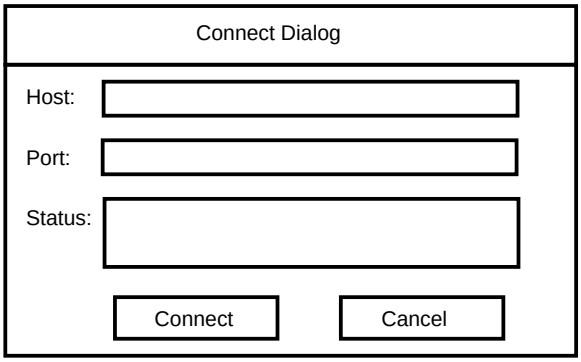
Kohtuullinen ratkaisu viittausongelmaan on antaa automaattisesti jokaiselle elementille sen tyypin mukaan TC-muotoinen numero. Numero esitetään puussa elementin nimen vieressä. Kun suodattimen asetuksissa halutaan määrittää kohdeluokka, se yksinkertaisesti valitaan alasvetovalikon listasta, jossa kaikki luokat ja niiden viitenumerot on lueteltu. Toinen vaihtoehto olisi toteuttaa jokin vedä ja pudota -toiminto, mutta sen löytäminen saattaisi olla vaikeaa käyttäjälle, joka ei ennestään tunne ohjelman toimintaa.

### 5.1.3 Käyttöliittymän yhteyden luominen

Käyttäjä voi luoda yhteyden kohdejärjestelmään joko käynnistämällä käyttöliittymän kohdejärjestelmän parametreilla tai avaamalla yhteyden käyttöliittymän sisältä [KL3]. Yhteyden luominen käyttöliittymän sisältä tapahtuu valitsemalla System-valikosta Connect-vaihtoehto. Tällöin aukeaa kuvan 4 mukainen uusi ikkuna, johon syötetään kohdejärjestelmän IP- tai DNS-osoite ja portti. Ikkuna pysyy auki kunnes yhteys kohdejärjestelmään saadaan luotua tai käyttäjä sulkee ikkunan Cancel-painikkeesta. Ohjelma kirjoittaa Status-tekstialueeseen tilannetietoa.

Jos pyyntö onnistui ja yhteys kohdejärjestelmään saatiin joko antamalla parametrit käynnistyksessä tai yhdistämällä erikseen käyttöliittymän sisältä, voi palvelinkomponentin asetuksia muuttaa. Yhteyden luomisen jälkeen ohjelma hakee automaattisesti kohdejärjestelmän asetukset muokkaamista varten. Jos käyttäjä oli määritellyt kaistanhallinta-asetuksia ennen yhteyden luomista, ohjelma varmistaa haluaako käyttäjä hakea asetukset automaattisesti vai pitää määrittelemänsä asetukset. Jos asetuksia ei haeta automaattisesti kohdejärjestelmältä, käyttäjä voi hakea ne myöhemmin valitsemalla System-valikosta Load From Daemon.

Mikäli yhteys kohdejärjestelmään on luotu ja käyttäjä haluaa ruveta muuttamaan toisen kohdejärjestelmän asetuksia, täyty hänen ensiksi valita System-valikosta Disconnect-vaihtoehto. Tällöin tämän hetkinen yhteys katkaistaan. Mikäli käyttäjällä oli tehtynä muutoksia, mitä hän ei ole lähettänyt nykyiselle kohdejärjestelmälle, käyttöliittymä varmistaa




Kuva 4: Uuden yhteyden luominen.

käyttäjältä, haluaako hän joka tapauksessa katkaista yhteyden vai jättää yhteyden auki mahdollistaakseen asetusten lähettämisen kohdejärjestelmälle. Kun käyttäjä on sulkenut yhteyden, voi hän avata uuden yhteyden edellä mainitulla tavalla.

## 5.2 Rakenne

Kuvassa 5 on Paketti-ryhmän toteuttaman käyttöliittymäkomponentin luokkakaavio. Muutoksia tehtäessä perusrakenne pyritään pitämään vähintään yhtä dynaamisena. Luvussa 3 esitellyn tietosuunnitelman mukaista jäykkää luokkarakennetta ei siis ohjelmoida, vaan käyttöliittymä rakennetaan yksinkertaisista elementtejä ja mahdollisesti attribuutteja mallintavista olioista suoraan määritellyn XML Scheman perusteella.

Seuraavaksi esitellään yksityiskohtaisesti kuhunkin käyttöliittymäluokkaan tehtäviä muutoksia.

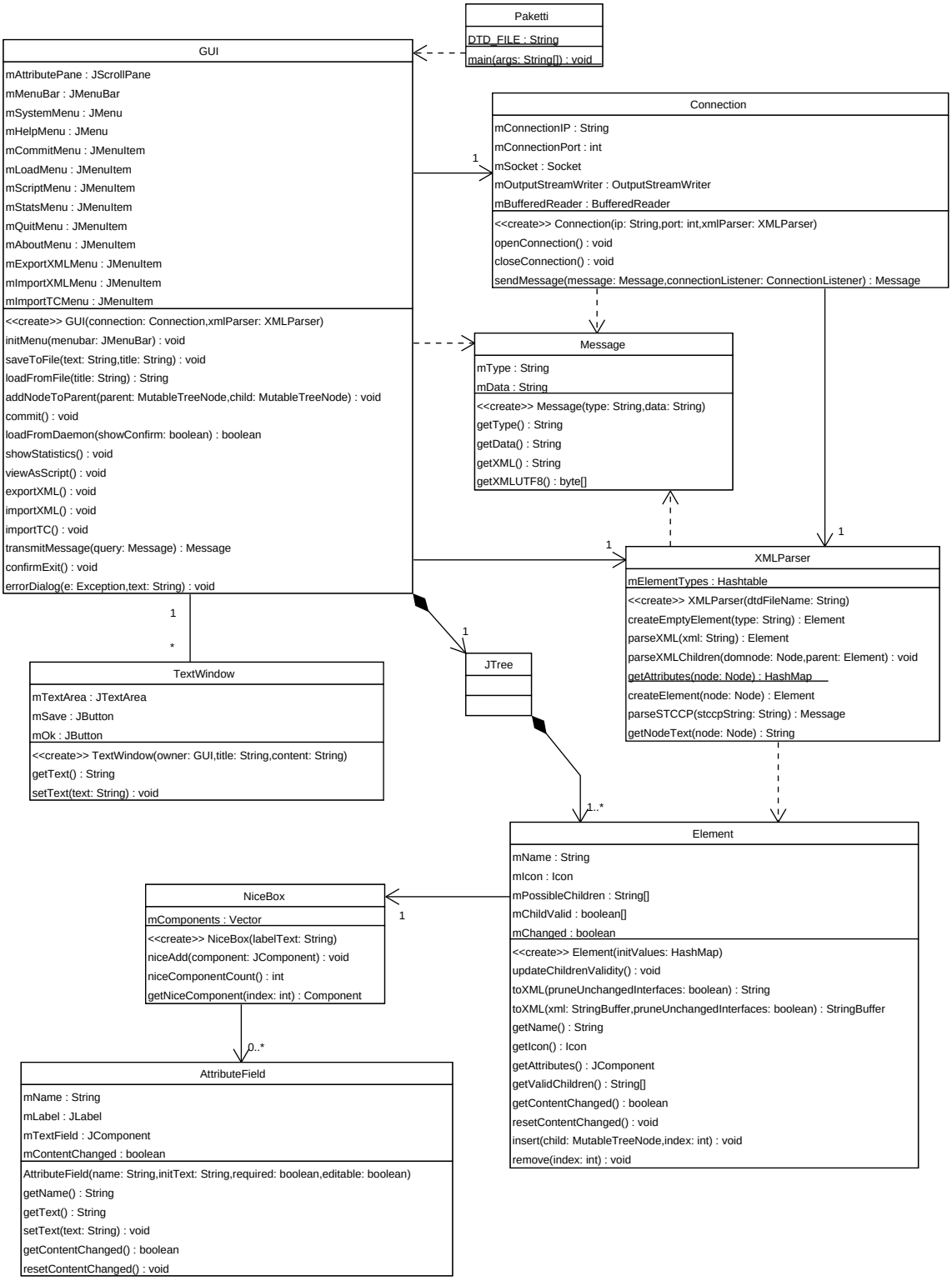
### Paketti

Pääluokka, joka sisältää vain main-metodin. Luo XMLParser- ja GUI-oliot sekä ottaa yhteyden kohdejärjestelmään.

- **public static void main(String[] args)**
  Metodiin tulee muutoksia, jotta ohjelma on mahdollista käynnistää ilman suoraa yhteyden ottamista kohdejärjestelmään. Komentoriviparametrit muutetaan valinnaisiksi. Lisäksi XMLParseria varten voidaan tarvittaessa muuttaa Xercesin luokkien hakupolkuja dynaamisesti, jotta ohjelma saadaan toimimaan sekä J2SE 1.5.0:n sisäisen Xerces-version, että muiden Java-ympäristöjen ja ulkoisen Xercesin kanssa.

### GUI

Rakentaa graafisen käyttöliittymän. Rakentaminen voidaan toteuttaa selkeämmin tekemällä kullekin käytettävälle Swing-oliolle alustusmetodi. Toteutetaan prioriteetilla 2.




Kuva 5: Nykyisen käyttöliittymäkomponentin luokkakaavio.

- **public void disconnect()**
  Uusi metodi, joka sulkee avatun yhteyden.

- **public void connect()**
  Uusi metodi. Avaa ikkunan, jonka avulla voi avata uuden yhteyden.

- **public GUI(XMLParser xmlParser)**
  Uusi konstruktori, joka avaa käyttöliittymän ilman yhteyttä palvelinkomponenttiin. Vanhan konstruktorin toiminnallisuus siirretään osittain tähän. Siistitään JTreen alustusta ja siirretään se omaan metodiinsa initTree().

- **public GUI(Connection connection, XMLParser xmlParser)**
  Nykyinen konstruktori. Muutetaan siten että kutsuu ensin uutta konstruktoria ja asettaa sitten yhteyden sallimat toiminnot käytettäviksi.

- **private void initMenu(JMenuBar menubar)**
  Lisätään Disconnect- ja Connect-vaihtoehdot System-valikon alle. Lisätään myös koodia, joka ottaa system-valikosta toiminnot Disconnect, Commit, Load from Daemon, View as Script ja Show Stats pois käytöstä mikäli yhteyttä ei ole luotu. Connect-vaihtoehto otetaan pois käytöstä mikäli yhteys on luotu.

- **private JTree initTree()**
  Uusi metodi JTree-olion alustamista varten.

- **public void tryToConnect()**
  Kun connect()-metodi on avannut uuden ikkunan, tryToConnect()-metodi pyrkii avaamaan uuden yhteyden käyttäjän ilmoittamaan IP- tai DNS-osoitteeseen.

- **public void actionPerformed(ActionEvent e)**
  Lisätään muutama koodirivi, jotta uusia toimintoja voidaan käyttää.

- **public void mouseClicked(MouseEvent e)**
  Puuelementtien tapahtumankäsittelijämetodi, johon on koodattu toimintamalleja erikoistapauksissa. Näistä pyritään pääsemään eroon scheman määritysten avulla.

- **public void viewAsScript()**
  Metodi, joka pyytää palvelinkomponentilta XML-tc-muunnoksen. Poistuu toistaiseksi, koska muunnosta varten toteutetaan erilliset komentorivityökalut.

- **public void importTC()**
  Metodi, joka pyytää palvelinkomponentilta tc-XML-muunnoksen. Poistuu toistaiseksi, koska muunnosta varten toteutetaan erilliset komentorivityökalut.

**XMLParser**

Lukee ja kirjoittaa XML-muotoisia kaistanhallinta-asetuksia kutsumalla Xercesin DOM-apia. Validoi dokumentit DTD:n avulla. Metodit säilyvät pääosin samoina, mutta validointi muutetaan schemaan perustuvaksi.

### Connection

STCCP-yhteyttä mallintava luokka, joka käyttää XMLParser-luokkaa viestien jäsentämiseen. Luokkaan tehdään tarvittavat muutokset protokollan uutta versiota varten. Metodit säilyvät samoina.

### Element

Luokka puunäkymän elementeille. Luotaessa uusi elementti, konstruktorille annetaan halutun elementin tyyppi. Ohjelma lukee suoraan DTD:stä tarvittavat tiedot attribuuteista ja sallituista alielementeistä. Osa toiminnallisuudesta on kuitenkin toteutettu suoraan koodiin: esim. lisättäessä alielementtejä, jäljelle jäävät sallitut elementit tarkistetaan koodattujen sääntöjen perusteella metodissa updateChildrenValidity().

Luokan metodien nimet ja tarkoitukset pyritään säilyttämään, mutta koodi muuttuu lähes kokonaan. DTD-käsittelystä siirrytään schemaan ja koodista tehdään mahdollisimman dynaamista, jotta uusien jonomallien ja muiden elementtien lisäämiseksi riittää scheman muokkaaminen.

Ohjejärjestelmää varten Element-luokkaan lisätään help-painikkeen toteutus. Painike sijoitetaan elementin nimen viereen attribuuttikenttien yläpuolelle.

### AttributeField

Otsikon ja tekstikentän sisältävä käyttöliittymäluokka, joka mallintaa Element-olion yhden attribuutin. AttributeField ei välttämättä vaadi suuria muutoksia, mutta sitä vastaavia luokkia on tehtävä jokaiselle erilaiselle attribuuttityypille erikseen. Nykyinen tietomalli tukee vain merkkijonoattribuutteja, jolloin AttributeField riittää. Scheman avulla attribuuttien tyypit voidaan määritellä tarkasti ja joissain tapauksissa voidaan antaa lista vaihtoehtoisista arvoista tai raja-arvot, joiden välissä attribuutin arvon on oltava.

Tarvitaan siis ainakin kaksi erilaista Swing-komponenttia: tekstikenttä ja alasvetovalikko, sekä kullekin datatyypille omat tarkistuksensa. Paras vaihtoehto olisi jakaa tietomalli ja käyttöliittymäkomponentit erillisiin luokkiin. Tämän toteuttaminen vaatii kuitenkin lisäsuunnittelua, jota tehdään mahdollisuuksien mukaan.

Alustava ratkaisu on toteuttaa kaksi luokkaa: tekstikentän sisältävä AttributeField ja valintalistan sisältävä AttributeSelector. Näille luokille toteutetaan lisäksi syötettyjen arvojen tarkastus joko generoimalla attribuutin elementin XML-koodi ja validoimalla se scheman avulla, tai erityisellä schemaa parsivalla tarkistusmetodilla.

Ohjejärjestelmää varten AttributeField-luokkaan lisätään help-painikkeen toteutus. Painike sijoitetaan attribuutin nimen ja tekstikentän tai listan väliin.

**NiceBox**

Box-luokkaa laajentava layout AttributeField-olioiden sijoitteluun. Sisältää helppokäyttöisiä metodeja sisällytettyjen olioiden hakua varten. Ei vaadi muutoksia ja saattaa jäädä tarpeettomaksi jos Element- ja AttributeField-luokkien toteutusta järkeistetään.

**Message**

STCPP-viestiolio. Mahdollisesti pieniä muutoksia protokollan uuden määrittelyn tukemiseksi.

**TextWindow**

Tekstimuotoisen datan esikatselu- ja tallennusikkuna. Käytetään tc-statistiikan esittämiseen ja tallentamiseen tiedostoon. Ei vaadi suuria muutoksia, mutta mahdollista graafisen statistiikan esittämistä varten on suunniteltava toinen luokka [KL9].

**PakettiHelp**

Uusi luokka ohjejärjestelmää varten. Ohjejärjestelmän laajempi kuvaus on luvussa 5.3. Käytetään singleton-oliona, joka on luokan ilmentymä.

- **public static PakettiHelp create(String key)**
  Staattinen luokkametodi, joka palauttaa viitteen olemassaolevaan PakettiHelp-olioon tai luo tarvittaessa uuden olion.
- **public void open(String key)**
  Avaa ohjeikkunan, jossa on key-koodin kuvaaman toiminnon ohjeistus. Jos ikkuna on jo olemassa, se päivittyy näyttämään kyseisen toiminnon kuvausta.

## 5.3 Sisäänrakennettu ohjejärjestelmä

Käyttöliittymään toteutetaan sisäänrakennettu ohjejärjestelmä [KL1], jonka avulla käyttäjä voi tarkistaa eri ominaisuuksien ja kenttien merkityksiä.

Itse ohjeet on kirjoitettu HTML-tiedostoon. Kaikki toiminnot on kuvattu samassa tiedostossa, ja niissä on ankkurit (<a name='x'>) viittauksia varten. Tiettyyn toimintoon viittaaminen tapahtuu siis muodossa tiedostonimi.html#ankkurinimi. Sallitut HTML-elementit on kuvattu taulukossa 1.

Ohjejärjestelmä kytketään käyttöliittymään kahdella tavalla: valikkoon, josta saa ohjeen pääsivun näkyviin, ja eri kenttien kohdalla oleviin kysymysmerkki-ikoneihin, joista aukeaa kyseisen kentän ohje. Ohjejärjestelmän ikkuna on mahdollisimman yksinkertainen, ja kaikki navigaatio toteutetaan HTML:n avulla. Dokumentaatio jakaantuu muutamaan

| | |
|---|---|
| <html> | HTML-dokumentin juurielementti. |
| <head> | Dokumentin metatietoja sisältävä elementti. |
| <title> | Dokumentin aihe. |
| <body> | Dokumentin sisällön juurielementti. |
| <h[1-3]> | Eri tasojen otsikkoja. |
| <p> | Tekstikappale. |
| <br> | Rivinvaihto. |
| <a> | Ankkuri (<a name=’x’>) tai viite (<a href=’x’>). |
| <ul>, <ol>, <li> | Listojen tekemiseen tarvittavat elementit. |

Taulukko 1: Ohjetiedostossa sallitut HTML-elementit.

HTML-tiedostoon, kuitenkin niin, että dokumentaation jokainen luku on aina yhdessä tiedostossa.

# 6 Palvelinkomponentti

Palvelinkomponentin toimintaa on tarkoitus muuttaa siten, että ennen xinetd:n avulla käynnistetty prosessi muutetaan kokonaan erilliseksi daemonikseen, jonka voi käynnistää joko suoraan komentoriviltä tai erillisen käynnistys-scriptin avulla. Tämä helpottaa käyttäjien poissulkemisessa ja mahdollistaa toiminnallisuuden, jossa voimassa olleet asetukset voidaan automaattisesti palauttaa tietokoneen uudelleenkäynnistyksen jälkeen.

Lisäksi tarkoituksena on luopua TC-komennon käyttämisestä palvelinkomponentin backendinä ja siirtyä käyttämään kaistanhallinta-asetukset kernelin omilla API-kutsuilla toteuttavaa LQL-kirjastoa, joka on GPL-lisensioitu ja sopii täten projektiin mainiosti.

## 6.1 LQL

LQL [Lp04] on C-kielellä kirjoitettu ohjelmointikirjasto, joka toimii wrapperina Linuxin kaistanhallinta-asetusten säätämiseen tarkoitetulle rajapinnalle. Kirjasto vaikuttaa tukevan kaikkia meidän ohjelmistoltamme vaadittuja ominaisuuksia, joten sen hyväksikäyttäminen on varsin loogista vaihtoehtona sille, että vastaava kirjasto kirjoitettaisiin itse netlink-sockettien avulla.

LQL:n ohjelmoinnissa on käytetty hyväksi Glib [GTK04]-kirjastoa, joka lisää C-ohjelmointiin monia valmiita tietorakenteita ja eräänlaisen oliojärjestelmän. Lisäksi kirjaston API-funktiot on dokumentoitu käyttämällä GTK-Doc [GTK01] notaatiota. Päätös käyttää LQL:a tarkoittaa siis näiden kolmen paketin lisäämistä ohjelmiston vaatimuksiin.

## 6.2 Daemon

Käynnistettäessä daemon prosessia ohjelman on luotava lapsiprosessi forkkaamalla ja äitiprosessin on lopetettava oma suorituksensa. Näin ohjelma käynnistyy taustalle ja suoritus palaa välittömästi daemonin käynnistäneelle prosessille.

Lapsiprosessin on tämän jälkeen luotava uusi sessio jatkaakseen suoritusta daemonina. Joissain tapauksissa on suositeltavaa forkata uudelleen, mutta meidän ympäristömme ei vaadi sitä. Lisäksi lapsiprosessin on syytä muuttaa työhakemistoon sellaiseksi, ettei se sijaitse levyllä joka saatettaisiin haluta unmountata.

Ennen varsinaisen suorituksen aloittamista, voidaan osana alkuvalmisteluja nollata perityt tiedoston luontiin liittyvät maskit ja sulkea tarpeettomat tiedostokuvaajat.

Koska sovellusta ei enää käynnistä xinetd, ei kukaan huolehdi verkkokommunikoinnista sovelluksen puolesta. Tästä syystä sovellukseen joudutaan itse kirjoittamaan kaikki verkkoyhteyden muodostamiseen ja kommunikointiin liittyvä koodi. Ohjelmoinnin helpottamiseksi ja yhtenäisen tyylin luomiseksi verkkokoodi on tarkoitus koota yhteen paikkaan ja muokata siitä sellainen, että se käyttäytyy kuten yleisimmät streamit [Str97] C++ ohjelmissa.

## 6.3 Daemonin käynnistäminen

Palvelinprosessin voi käynnistää joko suoraan komentoriviltä tai erillisellä käynnistysscriptillä. Komentoriviltä ohjelmalle on mahdollista antaa parametrina käynnistymiseen tarvittava konfiguraatiotiedosto parametrin -f kanssa. Toinen tuettu parametri on –help, joka tulostaa lyhyen ohjeen palvelimen käynnistämiseen liittyen. Jos komentoriviparametreja ei anneta, etsii palvelin asetuksia järjestelmän asetustiedostosta.

Asetustiedostoa etsitään kahdesta eri paikasta, kunnes tiedosto löytyy. Jos asetuksia ei löydy, joutuu palvelin lopettamaan suorituksensa. Asennustietojen etsintä tapahtuu järjestyksessä, PAKETTID_CONF ympäristömuuttujan kuvaamasta sijainnista ja hakemistosta /etc/pakettid.

Käynnistyttyään palvelin jää kuuntelemaan sille määrättyä porttia ja yhteydenoton tapahtuessa forkkaa palvelinprosessin kommunikoimaan yhteydenmuodostajan kanssa. Forkkaamisen jälkeen prosessi tarkastaa onko ympäristössä jo toinen käyttäjä, jolloin palvelin katkaisee yhteyden lähettämällä kyseistä tilaa kuvaavan viestin asiakassovellukselle.

## 6.4 Palvelimen tilan tallentaminen

Palvelin päivittää sisäistä tilaansa jokaisen onnistuneen muutoksen jälkeen. Tilatiedot tallennetaan myös jokaisen commitin yhteydessä tilatiedostoon, josta tila voidaan palauttaa uudelleenkäynnistyksen jälkeen. Tiedot tallennetaan verkkokommunikointiin käytettävässä XML-muodossa, josta ne on helppo tulkita. Tallennus tapahtuu hakemistoon /var/lib/pakettid.

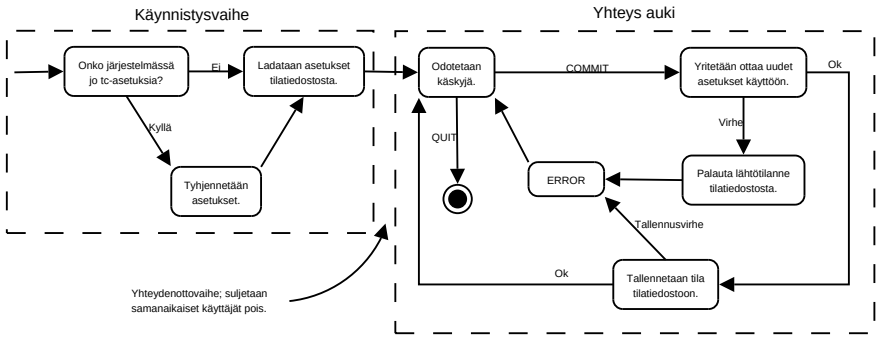
Tilan tallentaminen oikealla tavalla varmistaa asetusten käyttöönoton atomisuuden. Joko kaikki asetukset tulevat voimaan, tai ne hylätään kokonaan. Kuvassa 6 kuvataan asetusten käyttöönottoa daemonin käynnistymisestä lähtien.

Kun daemon käynnistyy, se tarkistaa, onko järjestelmässä tällä hetkellä kaistanhallinta-asetuksia käytössä. Jos on, ne tyhjennetään. Tämän jälkeen ladataan omat kaistanhallinta-asetukset tilatiedostosta. Jos tilatiedosto on tyhjä tai ei ole olemassa (esim. daemonin ensiasennuksen jälkeen), jatketaan tyhjillä asetuksilla.

Käynnistysvaiheen jälkeen daemon siirtyy odottamaan asiakasohjelman yhteydenottoa. Tätä vaihetta ja siihen kuuluvaa samanaikaisten käyttäjien poissulkeminen kuvataan tarkemmin luvussa 6.3.

Kun asiakasohjelma on ottanut yhteyden, daemon suorittaa sen antamia käskyjä. Kuvassa on selkeyden vuoksi jätetty pois kaikkia käskyjä paitsi COMMIT-käsky, jolla annetut asetukset otetaan käyttöön. Kun COMMIT-käsky annetaan, daemon yrittää ottaa annetut asetukset käyttöön. Jos käyttöönotossa ilmenee virhe, alkuperäinen tila palautetaan tilatiedostosta (tai palataan tyhjiin asetuksiin jos tilatiedosto on tyhjä tai puuttuu) ja lähetetään asiakkaalle virheilmoitus. Jos käyttöönotto onnistui, yritetään tallentaa tila tilatiedostoon. Virheestä raportoidaan asiakkaalle. Jos tallennus onnistuu, palataan odottamaan käskyjä.

QUIT-käsky johtaa lopputilaan, ja palataan odottamaan asiakasohjelman yhteydenottoa.




Kuva 6: Asetusten käyttöönotto ja atomisuuden varmistaminen.

## 6.5 Asetustiedosto

Palvelimen käynnistyessä etsittävät asetukset tallennetaan erittäin yksinkertaisessa muodossa, koska tarvetta ilmaisuvoimaisemmalle muodolle ei ole. Asetukset esitetään muodossa muuttuja=arvo, kukin muuttuja omalla rivillään. Riveillä sijaitsevat tyhjätilamerkit jätetään parsittaessa huomiotta.

Asetustiedostoon on tarkoitus tallentaa kuunneltava IP-osoite ja portti, lokitiedoston ja ohjelman tilan tallentavan tiedoston sijainnit sekä yhteyden aikakatkaisun viive sekunneissa.

IP-osoite voi olla mikä tahansa IP-osoitteen muotoinen merkkijono. Samoin portiksi kelpuutetaan mikä tahansa validi porttinumero. Asetustiedostojen sijaintien tulee olla ennestään olemassa olevia hakemistopolkuja ja viiveeksi voi antaa minkä tahansa sekuntiarvon. Viive määrää ajan, jonka palvelin odottaa asiakassovellukselta tultavaa viestiä ennen yhteyden katkaisemista. Jos aikakatkaisun viiveeksi asetetaan 0, ei aikakatkaisua suoriteta lainkaan.

| **Muuttuja** | **Arvo** |
|---|---|
| IP | 0.0.0.0 - 255.255.255.255 |
| PORT | 0 - 65535 |
| LOGDIR | Mikä tahansa validi hakemistopolku |
| STATEDIR | Mikä tahansa validi hakemistopolku |
| IDLE | 0 - |

Taulukko 2: Asetustiedoston mahdolliset muuttujat ja niiden mahdolliset arvot.

# 7 Virheiden korjaukset

Määrittelyn yhteydessä vahvistetut Paketti-ryhmän ohjelmiston virheet korjataan uudessa toteutuksessa seuraavan suunnitelman mukaisesti:

**O1: Xerces XML-jäsennin ei ole GPL-yhteensopiva.**

Käytetään JDK 1.5.0:aa, joka sisältää Xerces-jäsentimen. Koska Xercesiä ei tarvitse sisällyttää Paketti-ohjelman jakeluun ja paketointiin, GPL-lisenssi on käytettävissä.

**O2: NullPointerException painettaessa näppäintä käyttöliittymässä.**

Esiintyy JDK 1.4.2:lla ja vanhemmilla kun syöte ohjautuu JTree-komponentille. Todennäköisesti korjattavissa lisäämällä JTree-olioon näppäimistökäsittelijä. Ei esiinny JDK:n versiossa 1.5.0, joten korjaus ei ole välttämätön, mutta toteutetaan aikataulun salliessa.

**O3: Isojen XML-tiedostojen lukeminen voi aiheuttaa Out of memory -virheen.**

Lisätään tarkistus yli 10MB tiedostoille XML-import-toiminnon yhteyteen.

**O4: Tc-skriptien tuonnissa ei huomioida XML-koodin varattuja merkkejä.**

Ongelma poistuu koska Tc-skriptejä ei enää siirretä tietoliikenneyhteyden yli, vaan niitä käsitellään ainoastaan paikallisesti XML-TC-muuntotyökalulla.

**O5: Jonomallien ja suotimien nimitykset eivät ole kaikkialla yhtenäiset.**

Korjataan muuttamalla nimien käsittelyä. Nimet saadaan XML-jäsentimeltä kuten ennen, mutta niitä ei näytetä suoraan käyttöliittymäkomponenteissa. Sen sijaan niitä käytetään avaimina resurssitiedostoon kirjattuihin kokonaisiin nimiin ResourceBundle-olion avulla [KL8].

**O6: Attribuuttikenttien pituus on rajoitettu 30 merkkiin.**

Korjataan samalla kun attribuuttien tyyppien ja raja-arvojen käsittelyä kehitetään.

**O7: Palaaminen tyhjiin kaistanhallinta-asetuksiin ei onnistu virhetilanteessa.**

Virhe tapahtuu silloin, kun kohdejärjestelmässä on tyhjät kaistanhallinta-asetukset, ja käyttäjä yrittää asettaa virheellisiä asetuksia. Virhe korjataan palvelinpuolella tekemällä asetusten käyttöönotosta atominen operaatio; joko asetukset tulevat kokonaan voimaan, tai eivät ollenkaan. Tätä kuvataan tarkemmin luvussa 6.4. Virhe korjataan asiakaspuolella validoimalla käyttäjän syötteitä.

**O8: Tuki attribuuteille “ecn” ja “divisor” puuttuu.**

Arvojen puuttuminen johtuu vanhassa toteutuksessa käytetyn tc-komennon rajoituksista. Siirryttäessä Linuxin kernelrajapinnan käyttöön, ongelmasta pyritään pääsemään eroon.

**O9: Joissain tapauksissa arvo palautetaan eri muodossa kuin missä se on syötetty.**

Siirretään kaikki yksikköjen muunnostoiminnot käyttöliittymään, ja STCCP-yhteyden yli kulkee ainoastaan Linux-ytimen käyttämiä yksikköjä jokaisen attribuutin kohdalla. Käyttöliittymässä käytetään erillistä asetusta, joka määrää käytettävän yksikön, esim tavua, kilotavua, tai megatavua.

**O10: Virheellinen toiminta joillain attribuuteilla.**

Virhe on tc-komennon parametrien luonnissa. XSLT-transformaatio joka luo tc-komentotiedostoa ei toimi oikein. Korjataan XSLT-transformaatio sekä siirrytään Linux-rajapinnan käyttöön, jolloin kyseinen virhe ei vaikuta kohdejärjestelmän toimintaan.

# 8 Termistö

Seuraavassa annetaan selitykset tässä dokumentissa käytetyille termeille.

| | |
|---|---|
| **asiakas** | toinen osapuoli asiakas-palvelin-mallisessa järjestelmässä; palvelun käyttäjä. Ks. myös käyttöliittymäkomponentti |
| **daemon** | taustalla ajettava, yleensä pitkäkestoinen ohjelma, jolla ei ole ohjaavaa päätettä. Ks. myös palvelinkomponentti. |
| **forkkaaminen** | uuden, vanhempansa resursseja jakavan lapsiprosessin luominen prosessin sisällä (engl. *forking*) |
| **hallintajärjestelmä** | järjestelmä, jolta käyttöliittymää käytetään |
| **jono** | pakettijono; kaistanhallinnassa paketit sijoitetaan jonoihin, joiden toimintaa säännöstelee jonomalli [Hub04] |
| **jonomalli** | kaistanhallinnan väline; algoritmi, joka säännöstelee jonoa (engl. *queuing discipline* lyh. *qdisc*) [Hub04] |
| **kaistanhallinta** | tietoliikenteen hallinta laatutakuiden toteuttamiseksi (engl. *bandwidth management*) |
| **kohdejärjestelmä** | järjestelmä, jonka kaistanhallinta-asetuksia sovelluksella muutetaan |
| **konrollikomponentti** | ks. palvelinkomponentti |
| **käyttöliittymä** | ohjelmiston graafinen käyttöliittymä |
| **käyttöliittymä-komponentti** | ohjelmiston asiakasosa, joka sisältää käyttöliittymän |
| **laatutakuu** | asetusten kokonaisuus, jolla pyritään parantamaan jonkun palvelun käytettävyyttä tietoliikenneverkossa (engl. *quality of service*), lyh. *QOS* |
| **Linux** | vapaasti levitettävä, avoimen lähdekoodin kehitysmenetelmällä tuotettu käyttöjärjestelmäydin; myös GNU/Linux-käyttöjärjestelmästä yleisesti käytetty termi |
| **ohjelmisto** | Paketti2-ryhmän tuottama ohjelmisto kaistanhallinta-asetusten määrittelyyn graafisesti |
| **paketti** | käyttöliittymäkomponentin binääritiedosto |
| **pakettid** | palvelinkomponentin binääritiedosto |
| **palvelin** | toinen osapuoli asiakas-palvelun-mallisessa järjestelmässä; palvelun tarjoaja |
| **palvelinkomponentti** | ohjelmiston palvelinosa, joka tarjoaa rajapinnan kaistanhallinta-asetuksiin |
| **pyyntö** | STCCP-viesti, jolla asiakas tekee palvelimelle palvelupyynnön |

| | |
|---|---|
| **pääkäyttäjä** | käyttäjä, jolla on laajempi pääsy järjestelmän toimintoihin; UNIX-sukuisissa järjestelmissä yleensä “root” |
| **STCCP** | *Simple Traffic Control Configuration Protocol*; Paketti-projektin määrittelemä ja Paketti2-projektin laajentama protokolla palvelinkomponentin ja käyttöliittymän väliseen tiedonsiirtoon |
| **STCCP-viesti** | STCCP-protokollan mukainen yksittäinen viesti, joko pyyntö tai vastaus |
| **suodatin** | kaistanhallinnan väline IP-pakettien jakamiseen luokkiin (engl. *filter* tai *classifier*) |
| **vastaus** | STCCP-viesti, jolla palvelin vastaa asiakkaan pyyntöön |
| **ydin** | käyttöjärjestelmän keskeiset toiminnot sisältävä ydinohjelma; tässä yhteydessä Linux-ydin (engl. *kernel*) |
| **XML-dokumentti** | Puumalliin perustuva tiedon tallennus- ja siirtomuoto. Koostuu sisäkkäisten elementtien muodostamasta puusta, elementtien attribuuteista ja tekstidatasta. |
| **XML Schema** | XML-pohjaisen kielen kuvausmuoto; rajoittaa XML-dokumentin elementtien sijoittelua ja attribuuttien arvoja määriteltyjen sääntöjen mukaan. |
| **XSL** | *Extensible Stylesheet Language* |
| **XSLT** | *XSL Transformations*. Kuvauskieli, jonka avulla XML-dokumentit voidaan muuntaa lähes mihin tahansa ei-XML-pohjaiseen muotoon, esimerkiksi PDF- tai kuvatiedostoksi. |

# Lähteet


Ali04 Aliprand, Joan et. al., *The Unicode Standard*. Addison-Wesley, 2004. URL `http://www.unicode.org/`.

FSF91 Gnu general public license, 1991. URL `http://www.gnu.org/licenses/gpl.html`.

FSF99 Gnu lesser general public license, 1999. URL `http://www.gnu.org/licenses/lgpl.html`.

GTK04 GTK-projekti, Gtk-kirjaston kotisivu, 2004. URL `http://www.gtk.org/`.

GTK01 GTK+ Reference Documentation Project, Rdp:n kotisivu, 2001. URL `http://www.gtk.org/rdp/`.

Hub04 Hubert, B. et. al., Linux advanced routing and traffic control (lartc) howto, 2004. URL http://lartc.org/howto/.

Lp04 LQL-projekti, Lql-kirjaston kotisivu, 2004. URL http://www.coverfire.com/lql/.

Pr03a Paketti-ryhmä, Suunnitteludokumentti, 2003. URL http://www.cs.helsinki.fi/group/paketti/dokumentit/suunnitteludokumentti.ps.

Pr03b Paketti-ryhmä, Ylläpitodokumentti, 2003. URL http://www.cs.helsinki.fi/group/paketti/dokumentit/yllapitodokumentti.ps.

Pr04 Paketti2-ryhmä, Määrittelydokumentti, 2004. URL http://www.cs.helsinki.fi/group/paketti2/doc/maarittelydokumentti.pdf.

Str97 Stroustrup, B., *The C++ programming language*. Addison-Wesley, kolmas painos, 1997. URL http://www.research.att.com/~bs/3rd.html.

SUN04 Javatm 2 jdk, standard edition, 5.0, 2004. URL http://java.sun.com/j2se/1.5/docs/index.html.