HELSINGIN YLIOPISTO

TIETOJENKÄSITTELYTIETEEN LAITOS

Ohjelmistotuotantoprojekti

PAKETTI

Muutosdokumentti

Marko Buuri
Lasse Gustafsson
Tommi Rantala
Pekka Rousu
Jaakko Saaristo

16. joulukuuta 2003

Versiohistoria

Versio	Päivämäärä	Muutokset
1.0	16.12.2003	Ensimmäinen versio

Sisältö

1 Johdanto

Paketti-projekti kuuluu Helsingin yliopiston tietojenkäsittelytieteen laitoksen Ohjelmistotuotantoprojekti-kurssiin. Projektin 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

Projektin suunnitteluvaiheen tulokset on lueteltu suunnitteludokumentissa [PAK03b], joka jäädytettiin 4.11.2003. Suunnittelua jouduttiin kuitenkin tietyiltä osin muuttamaan jäädyttämisen jälkeen. Tähän dokumenttiin on dokumentoitu ne muutokset, jotka projektin tehtiin projektin suunnitteluun sen jäädytyksen jälkeen.

1.2 Dokumentin rakenne

Dokumenttissa on oma lukunsa ohjelmiston pääosille: STCCP-protokollalle, käyttöliittymälle ja kontrollikomponentille. Muutokset on kirjattu muodossa, jossa käsitellään mahdolliset ongelmat joiden takia muutokset tehtiin, ja tehdyt ratkaisut.

2 Muutokset protokollamäärittelyyn

Protokollamäärittelyyn tehtiin seuraavat korjaukset.

Ongelma

STCCP-viestin alusta puuttuu <?xml ... ?> -deklaraatio.

Ratkaisu

Vaikka ilman deklaraatiotakin XML-instanssi on hyvinmuodostettua ([W3C00], ”2.8 Prolog and Document Type Declaration”), päätettiin tämä lisätä protokollaan.

Ongelma

STCCP-viestin DATA-elemetin sisällön merkkejä ei määritelty millään tavalla. Tämä olisi mahdollistanut esimerkiksi seuraavan viestin lähettämisen:

```
Protocol:STCCP/1
Length:300
<?xml version="1.0" encoding="UTF-8"?>
<MESSAGE><TYPE>TRANSFORM_XML_TO_TC</TYPE><DATA>
<?xml version="1.0" encoding="UTF-8"?>
<system hostname="testikone" domain="testi.net"
        gatewayip="192.168.0.65" gatewaydevice="eth0">
        <interface name="eth1" ip="192.168.0.67" netmask="255.255.255.224"
        broadcast="192.168.0.95" mac="BC:DE:F0:12:34:56">
                <pfifoqdisc limit="100"/>
        </interface>
</system>
</DATA></MESSAGE>
```

Viestin XML-muoto ei olisi kuitenkaan tällöin ollut hyvinmuodostettua, sillä tietyt erikoismerkit pitää DATA-elementin sisällä korvata vastaavilla XML-entiteeteillä. Nämä erikoismerkit ovat W3C:n määrittelemiä ([W3C00], "2.4 Character Data and Markup").

Ratkaisu

Korvataan DATA-elementin sisällä olevat merkit XML-entiteeteiksi. Siis oikea tapa lähettää esimerkiksi yllä mainittu viesti olisi:

```
Protocol:STCCP/1
Length:455
<?xml version="1.0" encoding="UTF-8"?>
<MESSAGE><TYPE>TRANSFORM_XML_TO_TC</TYPE><DATA>
&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;
&lt;system hostname=&quot;testikone&quot; domain=&quot;testi.net&quot;
        gatewayip=&quot;192.168.0.65&quot;
        gatewaydevice=&quot;eth0&quot;&gt;
        &lt;interface name=&quot;eth1&quot; ip=&quot;192.168.0.67&quot;
        netmask=&quot;255.255.255.224&quot;
        broadcast=&quot;192.168.0.95&quot;
        mac=&quot;BC:DE:F0:12:34:56&quot;&gt;
                &lt;pfifoqdisc limit=&quot;100&quot;/&gt;
        &lt;/interface&gt;
&lt;/system&gt;
</DATA></MESSAGE>
```

3 Muutokset käyttöliittymäkomponentissa

Tämä luku kuvaa käyttöliittymän toteutuksessa tehdyt muutokset suunnitteluun.

3.1 Luokkarakenne ja elementtien parametritiedot

Alkuperäinen suunnittelu

Alunperin käyttöliittymä suunniteltiin hyväksi todetun oppikirjamallin mukaan jaettavaksi osiin (pakkauksiin) view ja model, sekä erilliseen verkkoliittymäosaan connection (suunnitteludokumentin luvut 5.3 ja 5.4). View-model-jaottelun etuna olisi ollut käyttöliittymän (view) itsenäisyys suhteessa käyttöliittymäkomponentin sisäisiin tietorakenteisiin (model). Haittana olisi seurannut ylimääräistä koodia osien välisen toiminnan organisoinnissa: jokainen kaistanhallinta-asetusten elementti oli jaettu kahteen eri luokkaan (P- ja E-alkuiset luokkanimet). E-alkuiset luokat toteuttivat Element-rajapinnan ja JTree-rakenteen alkiot olivat Node-luokan ilmentymiä, joista kukin sisälsi olioviitteen Element-tyyppiseen olioon. Elementtien parametrit oli kovakoodattu E-alkuisiin luokkiin. Suunnittelun aikana esitettiin myös geneerisen ratkaisun tekemistä siten, että elementit generoitaisiin ajonaikaisesti jonnekin tallennetusta datasta; joko XML-dokumentin DTD-kuvauksesta tai esim. erillisestä tekstitiedostosta. Ratkaisu kuitenkin hylättiin johtuen puuttuvasta toteutuskokemuksesta sekä ajanpuutteesta.

Ongelma

Suunnittelussa päätettiin huolimattomasti, että käyttöliittymäkomponentti käyttää sisäisenä tietorakenteenaan JTree-puurakennetta, mikä käyttöliittymäelementtinä olisi kuulunut View-osaan. Suunnittelua olisi voitu muuttaa siten, että käyttöliittymäkomponentilla olisi ollut model-osassa erillinen puurakenne dataa varten, mistä käyttöliittymä päivittää JTree-näkymän. Tästä kuitenkin olisi seurannut vieläkin enemmän ylimääräistä koodia, mitä model-view-jaottelu tuotti jo entisestään.

Toinen ongelma oli E-alkuisissa luokissa ja parametrien kovakoodauksessa. Koska luokkia oli reilu 10 ja jokaiseen oli kovakoodattu parametrien kentät, get- ja set-metodit sekä tekstikenttäelementit kuuntelijoineen, olisi yksinomaan tästä tullut vajaa 1000 riviä koodia. Tämä vaikutti myös ohjelmiston ylläpidettävyyteen, sillä uuden elementin lisääminen olisi vaatinut koodin lisäämistä useisiin paikkoihin.

Suunnittelu oli jo suurelta osalta ehditty toteuttaa, mutta ongelma tuotiin kuitenkin uudelleen suunnitteluun palaverissa 14.11.2003.

Ratkaisu

Model-view-jaottelusta luovuttiin ja nämä yhdistettiin pakkaukseen gui. Elementtien parametrien kovakoodauksen sijaan siirryttiin alunperinkin esitettyyn geneeriseen malliin, minkä toteuttaminen oli tässä vaiheessa selventynyt. Elementtien attribuuttikenttien luontiin löydettiin valmis DTD-jäsentäjä [BOU03], jolla attribuutit luetaan suoraan XML-dokumentin DTD-kuvauksesta. Tällöin käyttöliittymäelementtien muuttaminen onnistuu suoraan DTD-kuvausta muuttamalla. Kaikista E- ja P-alkuisista spesifisten elementtien luokista luovuttiin. Element-rajapintaluokka muutettiin ilmentymäluokaksi, joka generoi itselleen tarvittavat attribuuttikentät DTD-jäsentäjän luoman ElementType-olion perusteella. Lisäksi Node-luokka yhdistettiin Element:in kanssa siten, että Element perii DefaultMutableTreeNode-luokan ja JTree-rakenteen alkioiksi sijoitetaan suoraan Element-olioita.

Käyttöliittymästä on poistettu seuraavat suunnittelussa olleet luokat:

- Model
- Node
- View
- PHtbclass
- PPfifofastqdisc
- PPolice
- PSystem
- PU32filter
- PHtbqdisc
- PRedqdisc
- PPfifoqdisc
- PMatch
- PBfifoqdisc
- EHtbclass
- EPfifofastqdisc

- EPolice
- ESystem
- EU32filter
- EHtbqdisc
- ERedqdisc
- EPfifoqdisc
- EMatch
- EBfifoqdisc

Poistettujen luokkien tilalla on seuraavat uudet luokat:

Gui *extends JFrame implements ActionListener*

Gui-luokka on käyttöliittymän pääluokka, jonka tehtävä oli vanhan suunnittelun mukaan luokalla View.

Konstruktori

`public Gui(Connection connection, XMLParser xmlParser)`

Gui saa konstruktorin parametrina

1. Connection-olion, jota se käyttää tietoliikenneviestien lähettämiseen kontrollikomponentille, sekä
2. XMLParser-olion, jonka avulla se jäsentää kontrollikomponentilta tulleet XML-muotoiset viestit sekä muodostaa puurakenteen elementit (luokka Element) jäsentämällä XML-dokumenttien DTD-kuvausta.

Metodit

`public void saveFile(String text) throws IOException`

Näyttää tiedostovalintaikkunan (JFileChooser), jolla käyttäjä voi valita tiedoston. Tallettaa parametrin *text* sisällön tekstinä tiedostoon.

`public void mouseClicked(MouseEvent e)`

Metodia kutsutaan, kun puurakenteen elementtiä klikataan. Oikealla painikkeella päivittää oikeanpuoleiseen näkymään valitun elementin attribuuttikentät. Vasemmalla painikkeella näyttää popup-valikon, josta käyttäjä voi lisätä tai poistaa elementtejä.

`public void actionPerformed(ActionEvent e)`

Metodia kutsutaan, kun käyttäjä klikkaa sovelluksen päävalikon elementtiä. Käynnistää klikatusta elementistä riippuen sopivan toiminnan.

`public void errorMessage(Exception e, String text)`

Näyttää virhedialogin, minkä tekstinä on *text* sekä poikkeuksen *e* viesti.

Element *extends DefaultMutableTreeNode*

Element-oliot kiinnitetään Gui-luokassa sijaitsevaan JTree-rakenteeseen. Vanhassa suunnittelussa Element oli rajapinta, jonka erilliset elementtiluokat toteuttivat. Nyt elementtiolio muodostetaan XML:n DTD-kuvauksen perusteella.

Konstruktori

`public Element(ElementType type, String[] initValues)`

Luokka Gui luo Element-olion käyttämällä luokkaa XMLParser elementin attribuuttitietojen sekä niiden alkuarvojen määrittämiseen. Parametri *type* on DTD-jäsentäjän palauttama luokka, joka sisältää elementin nimen sekä tiedon sallituista alielementeistä ja niiden määrästä. alkuarvot on taulukossa *initValues*. Elementti luo attribuuttitietojen perusteella Box-olion, joka sisältää editoitavat tekstikentät attribuuteille.

Metodit

`public StringBuffer toXML(StringBuffer toXML)`

Palauttaa elementin sekä kaikkien sen alielementtien yhteisen XML-esitysmuodon sisältöineen katenoimalla sen annettuun StringBufferiin sen append-metodilla.

`public String getName()`

Palauttaa elementin nimen.

`public Icon getIcon()`

Palauttaa elementin kuvakkeen, jota käytetään puunäkymässä. Kuvakkeena käytetään kansiokuvaketta, mikäli elementille on mahdollista lisätä alielementtejä, muuten tiedostokuvaketta.

`public JComponent getAttributes()`

Palauttaa elementin attribuuttikentät Box-oliona. Box sisältää JLabel-otsikon, mistä ilmenee elementin nimi, sekä JLabelit ja JTextFieldit elementin attribuuteille.

`public String[] getValidChildren()`

Palauttaa elementin sallittujen alielementtien nimet. Gui käyttää tätä tietoa muodostaessaan valikkoa alielementtien lisäämiseen.

`public boolean getContentChanged()`

Palauttaa true, mikäli elementin attribuuttien tekstikenttiä on editoitu elementin luonnin tai edellisen resetContentChanged-kutsun jälkeen.

`public void resetContentChanged()`

Nollaa elementin attribuuttien tekstikenttien muutostiedot.

3.2 Muokattavien rajapintojen valinta

Alkuperäinen suunnittelu

Alunperin suunniteltiin, että puunäkymässä näkyvät rajapinnat valitaan erillisestä valikosta View->Interfaces (suunnitteludokumentin luku 5.2.4, kuva 16). Tällöin käyttäjä olisi valinnut itse ne rajapinnat, joita haluaa muokata, ja Commit-toiminnon yhteydessä olisi lähetetty kontrollikomponentille vain valitut rajapinnat.

Ongelma

Ilmeni, että rajapintojen valinta ja päivittäminen kuormittaa käyttäjää ja tekee ohjelmasta hankalan käyttää.

Ratkaisu

Rajapintojen valintamahdollisuus poistettiin kokonaan ja toteutettiin puunäkymä siten, että siinä näkyvät kerralla kaikki kohdejärjestelmän rajapinnat. Puunäkymä pitää kirjaa kunkin rajapinnan alielementteihin tehdyistä muutoksista ja Commit-toiminnon yhteydessä lähetetään kontrollikomponentille käyttöön otettavaksi vain ne rajapinnat, joita on käyttäjä on muokannut edellisen Commit- tai Load from daemon-toiminnon jälkeen.

3.3 Import XML, Export XML ja Import tc script

Alkuperäinen suunnittelu

Alkuperäisessä suunnittelussa ei ollut mahdollisuutta ladata tai tallettaa asetuksia paikallisesti.

Ongelma

Asiakas toivoi mainittusa toiminnallisuuksia vahvasti, mikäli vain aikaa riittäisi.

Ratkaisu

Toteuttamisesta koituva vaiva oli vähäinen, joten lisättiin Import XML, Export XML ja Import tc script -toiminnallisuudet päävalikkoon.

3.4 Tiedonsiirron odotusikkuna

Alkuperäinen suunnittelu

Alkuperäisessä suunnittelussa tiedonsiirron aikana asiakas näki pelkän odotusikkunan.

Ongelma

Pelkkä ikkuna on epäinformatiivinen.

Ratkaisu

Lisättiin odotusikkunaan "Progess bar" joka näyttää tiedonsiirron aikakatkoon (timeout) jäljelläolevan ajan.

3.5 XML-jäsentäjä

Alkuperäinen suunnittelu

Alunperin XML-jäsentäjäluokka XMLParser suunniteltiin model-pakkaukseen (suunnitteludokumentin luku 5.4.4), ja toteutettavaksi kirjastoluokkana.

Ongelma

Model-pakkauksesta luovuttiin ja käyttöön otettu DTD-jäsentäjä haluttiin myös sijoittaa XMLParser-luokkaan. Koska DTD-jäsentäjän alustus vaati kaistanhallinta-asetusten DTD-kuvaksen lukemisen levyltä, ei sitä ole järkevää sijoittaa kirjastoluokaksi.

Ratkaisu

XMLParser siirrettiin omaan pakkaukseensa xml. Luokan metodit muutettiin ilmentymämetodeiksi ja luokasta luodaan ilmentymä sovelluksen käynnistyksen yhteydessä.

3.6 DTD-kuvaus

Alkuperäinen suunnittelu

Suunnittelussa oli u32filter-elementin attribuuttilistassa attribuutti police sekä suurimmassa osassa elementeistä attribuutti estimator.

Ongelma

Huomattiin, että police on attribuuttina virheellisesti, sillä se toteutetaan elementtinä. Lisäksi huomattiin, että estimator-attribuutin arvona tulee aina olla kaksi parametria, joiden kysyminen käyttöliittymässä yhdessä tekstikentässä on harhaanjohtavaa.

Ratkaisu

Police poistettiin. Estimator muutettiin kahdeksi attribuutiksi: estimator_interval ja estimator_timesec. Uusi DTD-kuvaus on tämän dokumentin liitteenä.

3.7 Message-luokka

Alkuperäinen sunnitelma

Suunnittelussa Message-luokka sisälsi getData()-ja getType()-metodit.

Ongelma

Message-luokan sisältämä tieto tarvittiin XML-muotoisena, jonka toteutus päätettiin sisältää Message-luokkaan.

Ratkaisu

Tehtiin kaksi uutta metodia: getXML() ja getXMLUTF8().

3.8 XMLParser-ja Connection-luokkien virheiden hallinta

Alkuperäinen suunnitelma

Alkuperäisessä suunnittelussa ei otettu kantaa XMLParser-ja Connection-luokkien virhetilanteiden hallintaan.

Ongelma

Mahdollisten poikkeusten määrä oli kyseisissä luokissa suuri. Poikkeukset heitetään yläluokalle eli GUI-luokalle, josta olisi tullut huomattavasti koodimäärältään suurempi jos jokainen erilainen poikkeus olisi käsitelty erikseen.

Ratkaisu

Molemmille luokille tehtiin omat poikkeusluokat: XMLParserException ja ConnectionException.

4 Muutokset kontrollikompontissa

Tämä luku kuvaa kontrollikomponentin toteutuksessa tehdyt muutokset suunnitteluun.

4.1 transform.c

Ongelma

transform_tc_to_xml() hakee paikallisen järjestelmän verkkoliityntöjen asetukset, vaikka toiminta ei aina ole toivottavaa.

Ratkaisu

Funktion määrittelyyn on lisätty boolean-parametri, joka määrittelee käytetäänkö paikallisen järjestelmän verkkoliityntöjen tietoja tc-komentojen muunnossa XML-muotoon.

Ongelma

tc_to_xml() ei välttämättä havaitse koskevatko tietyt tc-komennot samaa verkkoliityntää.

Ratkaisu

Funktion määrittelyyn on lisätty merkkijonoparametri, joka sisältää sen verkkoliitynnän nimen, johon rekursiopuu liittyy. Rekursio purkautuu kun se etenee tc-komentoon, jossa ei esiinny tätä verkkoliitynnän nimeä.

Ongelma

tc_to_xml() ei saa parametrina XML-dokumentin isäelementtiä.

Ratkaisu

Funktion määrittelyyn on lisätty xmlNodePtr-tyyppinen osoitin isäelementtiin. Samalla dokumenttiosoitin sekä isäelementin nimi poistettiin parametreista.

Muutos

Funktiota clean_tc() ei tarvittu, koska XSL-muunnoksen tuottamia tc-komentoja ei tarvitse siistiä. Muunnos onnistuttiin tekemään niin, että se ei tuota tyhjiä rivivaihtoja.

4.2 Virheenkäsittely

Ongelma

Komponenttiin ei suunniteltu yhtenäistä virheenkäsittelyä.

Ratkaisu

Virheenkäsittelyyn toteutettiin neljä funktiota, jotka sijaitsevat tiedostossa error.c.

void error_set(int err_type, char *err_msg, char *filename, unsigned int line)

Asettaa virheen, jonka tyyppi on err_type. Virheviesti muodostetaan yhdistämällä virheilmoitus err_msg, tiedostonnimi filename, sekä rivinumero line. Virheilmoitus on kokonaisuudessaan esimerkiksi ”Internal error. (stccp.c:123)”, jossa ”Internal error” on parametrina saatu virheilmoitus, ”stccp.c” lähdetiedoston nimi, ja ”123” rivinumero josta tätä funktiota kutsuttiin.

Virhetilanteet käsittelee pakettid.c:n funktio main(), joka suorittaa virheen tapahtuessa alla kuvatun virheen tyypistä riippuvan toiminnon.

void error_clear()

Asettaa virhetilanteen oletustilaan, jossa virheen tyyppi on PAKETTID_NO_ERROR, ja virheilmoitus on tyhjä merkkijono.

int error_type()

Palauttaa asetetun virheen tyypin. Komponentissa käytetään seuraavia virhetyyppejä:

PAKETTID_NO_ERROR - ei virhettä
PAKETTID_NONTERMINATING_ERROR - virhetilanne, asiakasohjelmistolle lähetetään virheilmoitus
PAKETTID_TERMINATING_ERROR - virhetilanne, asiakasohjelmistolle lähetetään virheilmoitus, jonka jälkeen kontrollikomponentti sulkee hallintayhteyden
PAKETTID_CONNECTION_CLOSE - virhetilanne, konrollikomponentti sulkee hallintayhteyden

char *error_message()

Palauttaa virheilmoituksen.

4.3 Dynaaminen muistinhallinta

Ongelma

Komponenttiin ei suunniteltu menetelmää dynaamisen muistinhallinnan yhtenäiseen käsittelyyn. Joidenkin funktioiden muistinhallinta oli suunniteltu hoidettavan dynaamisilla taulukoilla, joiden kokoa olisi kasvatettu tarvittaessa.

Ratkaisu

Dynaamista muistinhallintaa varten toteutettiin mekanismi, joka sijaitsee tiedostossa arena.c. Muistinvaraus perustuu ns. arena-varaukseen, jossa muistia varataan määrätyn kokoisissa osissa, areenoissa. Näistä areenoista allokoidaan tarvittavan kokoisia muistialueita ohjelmiston tarpeisiin. Muistin vapauttaminen tapahtuu vapauttamalla kaikki areenat kerralla, kun ohjelmisto palautuu alkutilaansa. Toiminnolla varmistetaan, että ohjelmisto ei vuoda muistia.

Areenan käyttöönoton takia alunperin suunniteltuja dynaamisia taulukoita ei toteutettu, joten suurin mahdollinen varattava tila on ARENA_SIZE:n kokoinen. Pienempiä arena.h:ssa määriteltyjä tilakokoja ovat MEDIUM_CHUNK, SMALL_CHUNK ja TINY_CHUNK.

void *allocate_space(unsigned int size)

Palauttaa jostakin areenasta dynaamisesti varatun muistialueen, jonka koko on size tavua.

void free_space()

Vapauttaa kaikki dynaamisesti varatut areenat.

Viitteet

[BOU03] Bourret, R., Ron Bourret:n DTD-jäsentäjän kotisivu. `http://www.rpbourret.com/dtdparser/`, 19.11.2003

[FSF91] Free Software Foundation, Inc., GNU General Public License, 1991. `http://www.gnu.org/copyleft/gpl.html`, 1.10.2003

[FSF99] Free Software Foundation, Inc., GNU Lesser General Public License, 1999. `http://www.gnu.org/copyleft/lesser.html`, 1.10.2003

[PAK03b] Paketti-projektiryhmä, Suunnitteludokumentti, 2003. `http://www.cs.helsinki.fi/group/paketti/`

[W3C00] World Wide Web Consortium, Extensible Markup Language (XML) 1.0, 2000. `http://www.w3.org/TR/REC-xml`, 28.9.2003

Liite 1: Kaistanhallinta-asetusten DTD-kuvaus

```
<!ELEMENT system (interface*)>
<!ELEMENT interface (pfifofastqdisc | pfifoqdisc | bfifoqdisc |
                     redqdisc | htbqdisc)>
<!ATTLIST system
        hostname CDATA #REQUIRED
        domain CDATA #IMPLIED
        gatewayip CDATA #IMPLIED
        gatewaydevice CDATA #IMPLIED
>
<!ATTLIST interface
        name CDATA #REQUIRED
        ip CDATA #IMPLIED
        netmask CDATA #IMPLIED
        broadcast CDATA #IMPLIED
        mac CDATA #IMPLIED
>
<!-- classless qdiscs -->
<!ELEMENT pfifofastqdisc EMPTY>
<!ELEMENT pfifoqdisc EMPTY>
<!ELEMENT bfifoqdisc EMPTY>
<!ELEMENT redqdisc EMPTY>
<!ATTLIST pfifoqdisc
        handle CDATA #IMPLIED
        estimator_interval CDATA #IMPLIED
        estimator_timesec CDATA #IMPLIED
        limit CDATA #IMPLIED
>
<!ATTLIST bfifoqdisc
        handle CDATA #IMPLIED
        estimator_interval CDATA #IMPLIED
        estimator_timesec CDATA #IMPLIED
        limit CDATA #IMPLIED
>
<!ATTLIST redqdisc
        handle CDATA #IMPLIED
        estimator_interval CDATA #IMPLIED
        estimator_timesec CDATA #IMPLIED
        limit CDATA #REQUIRED
        min CDATA #REQUIRED
        max CDATA #REQUIRED
        avpkt CDATA #REQUIRED
```

```
        burst CDATA #REQUIRED
        probability CDATA #IMPLIED
        bandwidth CDATA #IMPLIED
        ecn (true | false) #IMPLIED
>
<!-- /classless qdiscs -->
<!-- classful qdiscs -->
<!ELEMENT htbqdisc (htbclass+, u32filter*)>
<!ATTLIST htbqdisc
        handle CDATA #IMPLIED
        estimator_interval CDATA #IMPLIED
        estimator_timesec CDATA #IMPLIED
        default CDATA #IMPLIED
        r2q CDATA #IMPLIED
>
<!-- /classful qdiscs -->
<!-- classes -->
<!ELEMENT htbclass ((htbclass* | pfifoqdisc | bfifoqdisc |
                     redqdisc | htbqdisc)?, u32filter*)>
<!ATTLIST htbclass
        classid CDATA #REQUIRED
        rate CDATA #REQUIRED
        burst CDATA #IMPLIED
        ceil CDATA #IMPLIED
        cburst CDATA #IMPLIED
        prio CDATA #IMPLIED
        quantum CDATA #IMPLIED
>
<!-- /classes -->
<!-- filters -->
<!ELEMENT u32filter (match*, police?)>
<!ELEMENT match EMPTY>
<!ELEMENT police EMPTY>
<!ATTLIST u32filter
        pref CDATA #IMPLIED
        protocol CDATA #IMPLIED
        estimator_interval CDATA #IMPLIED
        estimator_timesec CDATA #IMPLIED
        handle CDATA #IMPLIED
        classid CDATA #IMPLIED
        link CDATA #IMPLIED
        offset CDATA #IMPLIED
        ht CDATA #IMPLIED
```

```
        hashkey CDATA #IMPLIED
>
<!ATTLIST match
        sampletype (ip | ip6 | udp | tcp | icmp | u8 | u16 | u32)
                   #IMPLIED
        sampleargs CDATA #IMPLIED
>
<!ATTLIST police
        index           CDATA #IMPLIED
        buffer          CDATA #IMPLIED
        burst           CDATA #IMPLIED
        maxburst        CDATA #IMPLIED
        mtu             CDATA #IMPLIED
        avrate          CDATA #IMPLIED
        rate            CDATA #IMPLIED
        peakrate        CDATA #IMPLIED
        action (continue | drop | shot | pass | reclassify)
               #REQUIRED
>
<!-- /filters -->
```