UNIVERSITY OF HELSINKI
DEPARTMENT OF COMPUTER SCIENCE

Software Development Project

PAKETTI

Maintenance Document

Marko Buuri
Lasse Gustafsson
Tommi Rantala
Pekka Rousu
Jaakko Saaristo

16th December 2003

Version History

Version	Date	Changes
1.0	16.12.2003	First version

Contents

1 Introduction 1
1.1 Document Purpose 1
1.2 Document Structure 1
2 Software Architecture 1
2.1 Graphical User Interface 1
2.2 Control Component 3
3 Known Problems 4
3.1 Problems in Graphical User Interface 5
3.2 Problems in Control Component 6
3.3 Other Problems 8
4 Suggestions for Future Improvements 8
4.1 Graphical User Interface 8
4.2 Control Component 9
4.3 Simple Traffic Control Configuration Protocol 9
4.4 Other 9
5 Simple Traffic Control Configuration Protocol 10
5.1 Model of Operation 10
5.2 Datagram Structure 10
5.3 XML frame structure 11
5.4 Example Conversations 15
References 19

1 Introduction

The software development project Paketti is bound to the course "Software Development Project" held by Department of Computer Science at University of Helsinki. The project is aimed to produce a graphical software for easy modification of bandwidth management settings in the Linux kernel. The future users of the software are the various research groups at Department of Computer Science. The university will also publish the software under either GNU General Public Licence [FSF91] or GNU Lesser General Public Licence [FSF99].

1.1 Document Purpose

The purpose of this document is to provide information and guidance for the future maintainer and developer of the software. Strict development schedule forced the project team to leave some desirable features unimplemented and limited the quality of the chosen implementations. This document serves as a guidebook for development of the software after the project.

1.2 Document Structure

Section 2 desribes the basic architecture of the software, thus providing help for understanding the source code. Section 3 lists the known bugs and problems in the software implementation and suggestions how to fix them. Section 4 presents suggestions on how to improve the software in the future. Section 5 contains the definition of Simple Traffic Control Configuration Protocol, the application-level protocol for transferring bandwidth management settings.

2 Software Architecture

This chapter describes the basic functionality of the two software components: GUI and Control Component. It is supposed to make understanding the code easier.

2.1 Graphical User Interface

GUI functionality can be categorized into following three main parts.

1. The main graphical user interface part (classes GUI, Element, AttributeField, NiceBox and TextWindow) also containing the edited

data.

2. The communications part (classes Connection and Message) managing network data transfer between GUI and Control Component.

3. The XML parser part (class XMLParser) managing the decoding of the data from XML form to GUI inner representation (a tree structure of Element objects).

An informative UML representation on GUI architecture can be seen in Figure 1.

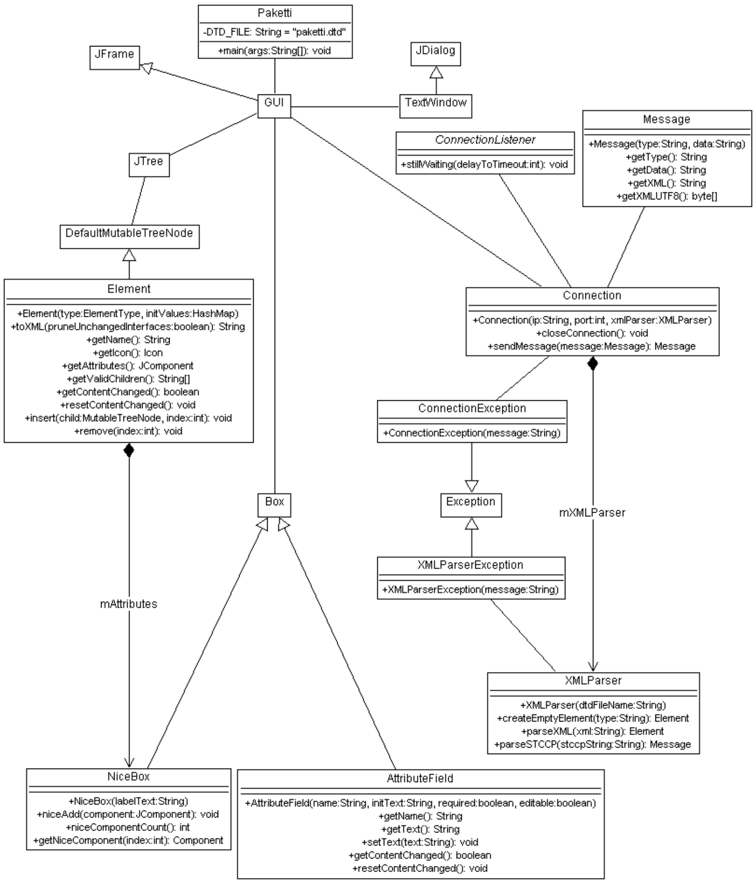

Figure 1: UML Representation on GUI

Basically, the data concerning the bandwidth management settings includes the following.

1. A group of elements such as RED qdisc, HTB class etc.

2. The values of each element's related attributes.

3. The element structure, which is a *tree*.

The data is stored using the actual Swing components in the following way.

Elements are represented by class Element which is instantiated respectively to every created element. An Element object contains the element's related attribute data in separate AttributeField objects. An AttributeField object consists of a JLabel name and either a JTextField in case the attribute is of editable type, or a JLabel otherwise. An Element object stores the AttributeFields in a convenience graphical component NiceBox, which is the actual graphical component rendered in the GUI right hand frame. Thus every Element contains a NiceBox which contains a set of AttributeField objects.

An Element object is created using directly the DTD file defining the settings. The software utilizes a DTD parser created by Ron Bourret [BOU03]. The DTD parser is used to create ElementType objects out of the parsed DTD file. An ElementType object is passed to Element in the constructor call, and Element creates necessary content for itself as described in the ElementType object.

Element extends Java's DefaultMutableTreeNode, which gives the ability to store Elements in a tree structure using the DefaultMutableTreeNode method add. This tree structure is compatible with the JTree object in the GUI left hand frame and can thus be rendered directly.

The data communications makes it necessary to encode and decode the settings data between Element and XML (String) representations. An Element object can return an XML representation of it's subtree using the recursive hand-made method toXML. XML strings are parsed to Element structures in class XMLParser by first using DOMParser in Xerces to parse the String into DOM nodes and then recursively creating Element objects passing the DTD parser ElementType objects to Element constructor.

Error situations are handled using exception classes XMLParserException and ConnectionException. ConnectionException is thrown in class Connection when a communication error occurs. XMLParserException is thrown when any error occurs in class XMLParser. The exceptions are handled in GUI by invoking the method errorDialog to show the exception contained error message.

2.2 Control Component

Control Component functionality can be categorized into the following main parts.

1. Fetching current settings and statistics with tc binary.
2. Transformation of settings from XML to tc commands and back.
3. Commiting settings.
4. Communication with client.

The functionality is divided into several implementation files: pakettid.c, stccp.c, transform.c, etc. These files include several functions that handle a given task.

Control Component is invoked by xinetd whenever the client makes a connection to the configured TCP port. The main function in pakettid.c then reads requests from the client using functions from stccp.c, and calls the functions defined in other files to process the request. The main function continues to serve the client until the connection is closed or until a major error occurs.

Transformations are done whenever the client requests current settings, commits new settings or requests a transformation from tc to XML or XML to tc.

Current bandwidth management settings are transformed into tc commands using an XSL Transformation stylesheet. The transformation is done using libxslt.

When the client requests current settings in XML form, current settings are transformed into tc commands, and the tc commands are transformed into XML form with a hand-made parser. The XML tree is formed using libxml. If the client request the settings in tc form, the commands are sent as is.

When receiving a commit request, Control Component uses an XSL Transformation stylesheet to transform the settings into tc commands. The commands are then run, and current settings are then returned to the client. If there are any errors when applying the new settings, the old settings are restored.

An error reporting facility was designed into error.c. If there are any errors while serving a request, the functions called from main() use those error setting functions to set up an accurate error state. After calling the function, main() then checks for any errors. If there were any, the error message is then sent to the client and the processing of the request halts.

3 Known Problems

The project testing phase revealed some problems we were unable to fix within the project schedule. The problems are documented in this section.

3.1 Problems in Graphical User Interface

Problem 1: User interface throws a NullPointerException when user presses any key in the keyboard when no attribute field is selected to receive the key input. We were unable to locate the cause for the problem. The Exception stack trace is listed in the following:

```
java.lang.NullPointerException
at javax.swing.JTree.getNextMatch(JTree.java:2504)
at javax.swing.plaf.basic.BasicTreeUI$KeyHandler.keyTyped(BasicTreeUI.java:2624)
at java.awt.Component.processKeyEvent(Component.java:5048)
at javax.swing.JComponent.processKeyEvent(JComponent.java:2385)
at java.awt.Component.processEvent(Component.java:4902)
at java.awt.Container.processEvent(Container.java:1566)
at java.awt.Component.dispatchEventImpl(Component.java:3598)
at java.awt.Container.dispatchEventImpl(Container.java:1623)
at java.awt.Component.dispatchEvent(Component.java:3439)
at java.awt.KeyboardFocusManager.redispatchEvent(KeyboardFocusManager.java:1688)
at java.awt.DefaultKeyboardFocusManager.dispatchKeyEvent(DefaultKeyboardFocusManager.java:593)
at java.awt.DefaultKeyboardFocusManager.preDispatchKeyEvent(DefaultKeyboardFocusManager.java:765)
at java.awt.DefaultKeyboardFocusManager.typeAheadAssertions(DefaultKeyboardFocusManager.java:698)
at java.awt.DefaultKeyboardFocusManager.dispatchEvent(DefaultKeyboardFocusManager.java:559)
at java.awt.Component.dispatchEventImpl(Component.java:3468)
at java.awt.Container.dispatchEventImpl(Container.java:1623)
at java.awt.Window.dispatchEventImpl(Window.java:1585)
at java.awt.Component.dispatchEvent(Component.java:3439)
at java.awt.EventQueue.dispatchEvent(EventQueue.java:450)
at java.awt.EventDispatchThread.pumpOneEventForHierarchy(EventDispatchThread.java:197)
at java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:150)
at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:144)
at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:136)
at java.awt.EventDispatchThread.run(EventDispatchThread.java:99)
```

Solution: There might be a bug either in Java Runtime Environment or perhaps somewhere in Swing JTree component. Upgrading to a new JRE might be worth trying.

Problem 2: Importing big files (>10MB) using the "Import XML" action can cause an Out of memory error.
Suggested solution: Add a file size check before importing.

Problem 3: There is a constant scenario of System menu disappearing from under the mouse. It can be produced this way: 1. do something that produces a separate window or dialog, for example invoke Show stats, 2. close the window/dialog, 3. click on the System menubar item, 4. move the mouse down on the appearing menu. The menu disappears but reappears if you reselect the System menubar item.
Suggested solution: Problem might be related to component focus.

Problem 4: Import tc script does not change XML reserved characters from the imported tc script if the imported script for some reason does contain non-XML-encoded XML. Export XML saves the exported settings in (correct) XML-encoded XML format though.
Suggested solution: Replace every XML reserved characters from the tc script. (reserved characters are listed in XML specification)

Problem 5: After a network transfer involving action, such as Commit, if an error dialog is displayed and the user clicks on the main window, the progress bar appears in front of the error dialog covering it in an ugly way.
Suggested solution: Rid of the progress bar properly before displaying the error dialog.

Problem 6: The settings are displayed with inconsistent terms in different places. For example htbqdisc and RED qdisc.
Suggested solution: There is an array: `protected static final String[][] ELEMENT_NAME_COSMETICS` in class GUI, which contains the (element name, rendered name) pairs for elements to be renamed. Some names were missed unintentionally. The naming problem persists though in remove confirmation dialog, which queries: Are you sure you want to remove "redqdisc"...

Problem 7: The maximum character limit for the attribute fields is constant 30. This actually might be too little for some special attribute cases, for example Match element attribute sampleargs.
Suggested solution: The constant MAX_TEXT_LENGTH is defined in class AttributeField. Change it.

3.2 Problems in Control Component

Problem 8: If the current bandwidth management settings are empty, and the user tries to commit erroneous settings, Control Component is unable to revert back to the empty settings.
Suggested solution: This could be fixed in main_commit(): if current_settings() returns a NULL pointer, then call commit() with a string that clears any settings.

Problem 9: Control Component doesn't always send message back if it receives a TRANSFORM_TC_TO_XML request which contains random data.
Suggested solution: The problem might be in transform_tc_to_xml() not setting a proper error status when returning a NULL pointer. A check might also be added to main_transform_tc_to_xml().

Problem 10: Tc reports attribute "ecn" for RED qdisc and "divisor" for U32 filter which are not send to Paketti.
Suggested solution: Support for these attributes should be add to Con-

trol Component. Implementation of "ecn" required modification to function parse_and_add_attributes() in transform.c. "Divisor" is not designed to be implemented so it requires modification to DTD and GUI as well as to function parse_and_add_attributes().

Problem 11: Control Component does not return all attributes when committing RED qdisc or U32 filter. The attributes not returned by tc are "estimator" values for all qdiscs and "avpkt", "bandwidth", "burst" and "probablity" for RED qdisc.
Suggested solution: This cannot be fixed using tc.

Problem 12: Control Component will return Match element attribute "sampleargs" in the form returned by tc, which is typically of the form "00b80000/00ff0000 at 0". In addition, Control Component will remove the "at 0" from the string.
Suggested solution: Fixing this would include converting mentioned hex decimal string to the format used in tc parameter input, which would probably be intractable.

Problem 13: If the user uses "police" and "match" in bandwidth management settings at the same time, Control Component doesnt produce valid tc commands.
Suggested solution: The problem is in the XSL Transformation file: in the template that matches the match element, the test for position should be removed.

Problem 14: If the user uses more than one "match" in any processed settings, Control Component won't produce valid tc commands.
Suggested solution: When the previous problem is solved, this will also be fixed.

Problem 15: If there is more than one "match" attribute in any processed settings, Control Component will only process the first one.
Suggested solution: Add support for multiple "match" attributes. The implementation requires changes in transform.c at functions tc_to_xml() and parse_match().

Problem 16: Control Component does not send information to the syslog service.
Suggested solution: This feature isn't implemented in the current version. The future implementation would require inclusion of syslog.h in file paket-

tid.h. Errors can be then logged by using function syslog() from error_set() in error.c. Logging of outgoing messages would be best handled in stccp.c at functions send_reply() and send_error(). Logging of incoming messages can be done at read_request() in stccp.c.

3.3 Other Problems

Problem 17: Software is neither GPL or LGPL compatible. It uses Xerces XML-parser which is published under Apache license [ASF03a], which is known to be incompatible with both GNU licences.
Suggested solution: Replace Xerces with a GPL compatible parser, or modify the user interface to use the SAX interface instead of DOM.

4 Suggestions for Future Improvements

This section makes suggestions for future upgrades on the software.

4.1 Graphical User Interface

- Modify the user interface to be more user friendly so that the user doesn't need to know anything about the tc tool
- Change error messages to be more user friendly
- Provide bandwidth management presets for typical situations
- Add user guides (wizards)
- Add support for different languages
- Statistical information displayed in graphical format in the user interface
- Add support for the STCCP protocol request GET_SETTINGS_TC
- Add support for using the GUI in offline mode
- Add support for changing to another Control Component without restarting the user interface
- Show recently used files in the System menu

4.2 Control Component

- Change pakettid to use sockets instead of xinetd, because xinetd outputs stderr to the user. This can break the STCCP protocol.
- Modify the tc tool to report all the missing parameters, eg. RED qdisc, U32 filter, estimators etc.
- Use the Linux kernel API instead of the tc tool to get current bandwidth management settings
- Block concurrent management sessions, because if two administrators modify the bandwidth management settings at the same time, the behaviour is undefined
- Add support for modifying interface and system networking parameters, such as ip addresses, etc. that are currently shown in GUI as unalterable
- Add support for other than eth* interfaces

4.3 Simple Traffic Control Configuration Protocol

- Message authentication with digital signatures
- Message TYPE and DATA content encryption
- Compression of the messages

4.4 Other

- Transport level encryption
- User authentication using digital signatures or passwords
- Support for the qdiscs Priority (PRIO), Token Bucket Filter (TBF), Stochastic Fairness Queueing (SFQ), Class Based Queueing (CBQ), Differentiated Service (DSMARK) and Weighted Fair Queueing (WFQ).
- Support for the filters fw, route and tcindex
- Design the XML form used with the GET_STATISTICS_XML request, and implement support for it in the user interface and in pakettid
- Add support for setting up iptables (firewall) rules

5 Simple Traffic Control Configuration Protocol

Simple Traffic Control Configuration Protocol (STCCP) is an application level protocol for transmitting bandwidth management settings and statistics. It is defined by Paketti software project group. The protocol is based on the Client-Server Model architecture on TCP/IP and encodes the request and reply data in XML frames.

5.1 Model of Operation

The model of operation for transmitting the settings is as follows:

- Client opens a TCP connection to server
- Client transmits a request datagram to server
- Server transmits a reply datagram to client
- (After receiving the reply, client may send another request)
- (Server replies ...)
- Client transmits a QUIT request and closes the TCP connection

The model is clarified in the following figure. The notation $*$ implicates a possibility for zero or more repetitions.

```
      Client ---- open TCP/IP connection ---> Server

.---> Client ------------ request ----------> Server
|     Client <------------ reply ------------ Server
|_____|
   *
      Client --------- QUIT-request --------> Server
      Client --- close TCP/IP connection ---> Server
```

5.2 Datagram Structure

The following notations are used in the datagram structure definition:

```
       LF = line feed
            hex decimal 0xA, decimal 10
  integer = a non-negative integer k
            in range 0 <= k < 2^32 = 4294967296
character = an arbitrary character of UTF-8 charset
   string = an arbitrary sequence of characters
            approved by XML syntax
```

STCCP datagrams utilize Unicode UTF-8 charset. The datagrams are of the following form:

```
version_number LF
length LF
xml_frame
```

The `version_number` field is of the following form:

```
Protocol:STCCP/integer
```

The value of `integer` is 1 in the first version of the protocol, 2 in the second and so on. The `length` field is of the following form:

```
Length:integer
```

The value of `integer` corresponds to the total length of `xml_frame` in bytes. The structure of the last field, `xml_frame`, is defined in Section 5.3.

An example STCCP datagram of type GET_STATISTICS_TC:

```
Protocol:STCCP/1 LF
Length:54 LF
<MESSAGE><TYPE>GET_STATISTICS_TC</TYPE><DATA /></MESSAGE>
```

5.3 XML frame structure

The content of an STCCP datagram is stored in an XML frame. The frame structure is defined by the following DTD:

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

The TYPE element contains the datagram type in uppercase letters. The several types come in two categories: requests and replies. The requests are sent only by a client, and the replies are sent only by a server. The server transmits a reply for every received request, except for the QUIT request.

The DATA element contains the actual transmitted data. This basically consists of bandwidth management settings, error information in case of ERROR reply, or statistics information. However there is no restriction to transmit other kinds of data as well.

The element pair SEQUENCE and ACK is optional and included in the DTD only for future upgrades. They are not utilized in this protocol version.

Requests

The following defines each request along with it's message type, the corresponding server reply and possible error conditions. Server replies with REPLY reply if the request was accepted, otherwise with ERROR. The error conditions common to all requests are listed in the definition of ERROR reply.

GET_SETTINGS_XML

Description: GET_SETTINGS_XML requests the current bandwidth management settings in XML form. Server may ignore the datagram DATA element, which should be empty.

Reply: DATA element contains the bandwidth management settings in XML form.

Error conditions: see ERROR

```
<MESSAGE><TYPE>GET_SETTINGS_XML</TYPE><DATA /></MESSAGE>
```

GET_SETTINGS_TC

Description: GET_SETTINGS_TC requests the current bandwidth management settings in the form of a tc script. Server may ignore the datagram DATA element, which should be empty.

Reply: DATA element contains the bandwidth management settings in the form of a tc script. The tc commands are separated with line feeds.

Error conditions: see ERROR

```
<MESSAGE><TYPE>GET_SETTINGS_TC</TYPE><DATA /></MESSAGE>
```

GET_STATISTICS_XML

Description: GET_STATISTICS_XML requests statistical information in XML form. Server may ignore the datagram DATA element, which should be empty.

Reply: DATA element contains statistical information in XML form. The actual XML form is not defined in this protocol version.

Error conditions: An error occured requesting statistics from target system.

```
<MESSAGE><TYPE>GET_STATISTICS_XML</TYPE><DATA /></MESSAGE>
```

GET_STATISTICS_TC

Description: GET_STATISTICS_TC requests statistical information in the form output by tc. Server may ignore the datagram DATA element, which should be empty.

Reply: DATA element contains statistical information in the form output by tc.

Error conditions: An error occured requesting statistics from target system.

```
<MESSAGE><TYPE>GET_STATISTICS_TC</TYPE><DATA /></MESSAGE>
```

COMMIT

Description: COMMIT requests server to run the accompanied bandwidth management settings into the target system. DATA element contains the settings-to-be-run in XML form.

Reply: DATA element contains the settings-that-were-run in XML form.

Error conditions: DATA element is empty.
DATA element contains malformed XML.
The settings-to-be-run contained by DATA element are invalid.
An error occured while running the settings.

```
<MESSAGE><TYPE>COMMIT</TYPE><DATA>string</DATA></MESSAGE>
```

TRANSFORM_TC_TO_XML

Description: TRANSFORM_TC_TO_XML requests server to transform the accompanied bandwidth management settings to XML form. DATA element contains the settings-to-be-transformed in tc form.

Reply: DATA element contains the transformed settings in XML form.

Error conditions: DATA element is empty.
The settings-to-be-transformed commands are invalid or unsupported.

```
<MESSAGE><TYPE>TRANSFORM_TC_TO_XML</TYPE><DATA>string</DATA></MESSAGE>
```

TRANSFORM_XML_TO_TC

Description: TRANSFORM_XML_TO_TC requests server to transform the accompanied bandwidth management settings to tc form. DATA element contains the settings-to-be-transformed in XML form.

Reply: DATA element contains the transformed settings in tc form.

Error conditions: DATA element is empty.
DATA element contains malformed XML.
The settings-to-be-transformed are invalid or unsupported.

```
<MESSAGE><TYPE>TRANSFORM_XML_TO_TC</TYPE><DATA>string</DATA></MESSAGE>
```

QUIT

Description: QUIT notifies server that the connection will be closed. Server may ignore the datagram DATA element, which should be empty.

Reply: Server sends no reply.

Error conditions: -

```
<MESSAGE><TYPE>QUIT</TYPE><DATA /></MESSAGE>
```

Replies

REPLY

Description: Server transmits a REPLY datagram in response to every accepted GET_SETTINGS_TC, GET_SETTING_XML, GET_STATISTICS_TC, GET_STATISTICS_XML, COMMIT, TRANSFORM_XML_TO_TC or TRANSFORM_TC_TO_XML request. If the request type implied reply data, the data is contained in DATA element.

```
<MESSAGE><TYPE>REPLY</TYPE><DATA>string</DATA></MESSAGE>
```

ERROR

Description: Server transmits an ERROR datagram if the request received was not accepted for some reason. The reply DATA element contains additional information on the error cause.

```
<MESSAGE><TYPE>ERROR</TYPE><DATA>string</DATA></MESSAGE>
```

The following possible error conditions are common to all requests:

- The request format was invalid, and
- Server internal error occured while processing the request.

5.4 Example Conversations

The following presents two examples of the use of STCCP in client-server conversation. The first example shows client requesting for statistics followed by server reply. The second example shows client requesting transformation of tc commands to XML form followed by server reply. It is worth noticing that for the `xml_frame` field contained XML document to be well-formed, the XML data *inside* DATA element must be encoded according to the W3C XML recommendation.

Basic Example

Client:

```
Protocol:STCCP/1 LF
Length:54 LF
<MESSAGE><TYPE>GET_STATISTICS_TC</TYPE><DATA /></MESSAGE>
```

Server:

```
Protocol:STCCP/1 LF
Length:1976 LF
<MESSAGE><TYPE>REPLY</TYPE><DATA>[qdisc] LF
qdisc sfq 13: quantum 1492b perturb 10sec LF
 Sent 1569980 bytes 11521 pkts (dropped 0, overlimits 0) LF
 LF
 qdisc sfq 12: quantum 1492b perturb 10sec LF
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0) LF
 LF
 qdisc sfq 11: quantum 1492b perturb 10sec LF
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0) LF
 LF
 qdisc sfq 10: quantum 1492b perturb 10sec LF
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0) LF
 LF
 qdisc htb 1: r2q 1 default 13 direct_packets_stat 0 LF
 Sent 1569980 bytes 11521 pkts (dropped 0, overlimits 453) LF
 LF
[class] LF
class htb 1:11 parent 1:1 leaf 11: prio 1 rate 30Kbit ceil 126Kbit LF
burst 1637b cburst 1760b LF
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0) LF
 lended: 0 borrowed: 0 giants: 0 LF
```

```
 tokens: 349439 ctokens: 89446 LF
 LF
class htb 1:1 root rate 126Kbit ceil 126Kbit burst 1760b cburst 1760b LF
 Sent 1569888 bytes 11520 pkts (dropped 0, overlimits 0) LF
 rate 1bps LF
 lended: 2973 borrowed: 0 giants: 0 LF
 tokens: 84977 ctokens: 84977 LF
 LF
class htb 1:10 parent 1:1 leaf 10: prio 0 rate 10Kbit ceil 126Kbit LF
burst 1611b cburst 1760b LF
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0) LF
 lended: 0 borrowed: 0 giants: 0 LF
 tokens: 1031680 ctokens: 89446 LF
 LF
class htb 1:13 parent 1:1 leaf 13: prio 3 rate 10Kbit ceil 126Kbit LF
burst 1611b cburst 1760b LF
 Sent 1569888 bytes 11520 pkts (dropped 0, overlimits 0) LF
 rate 3bps LF
 lended: 8547 borrowed: 2973 giants: 0 LF
 tokens: 975360 ctokens: 84977 LF
 LF
class htb 1:12 parent 1:1 leaf 12: prio 2 rate 75Kbit ceil 126Kbit LF
burst 1695b cburst 1760b LF
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0) LF
 lended: 0 borrowed: 0 giants: 0 LF
 tokens: 144724 ctokens: 89446 LF
 LF
[filter] LF
filter parent 1: protocol ip pref 1 fw LF
filter parent 1: protocol ip pref 1 fw handle 0xb classid 1:11 LF
filter parent 1: protocol ip pref 2 fw LF
filter parent 1: protocol ip pref 2 fw handle 0xc classid 1:12 LF
filter parent 1: protocol ip pref 3 fw LF
filter parent 1: protocol ip pref 3 fw handle 0xd classid 1:13 LF
filter parent 1: protocol ip pref 49152 fw LF
filter parent 1: protocol ip pref 49152 fw handle 0xa classid 1:10 LF
</DATA></MESSAGE>
```

Client:

```
Protocol:STCCP/1 LF
Length:44 LF
<MESSAGE><TYPE>QUIT</TYPE><DATA /></MESSAGE>
```

Advanced Example

Client:

```
Protocol:STCCP/1 LF
Length:384 LF
```

```
<MESSAGE><TYPE>TRANSFORM_TC_TO_XML</TYPE><DATA>
tc qdisc add dev eth0 root handle 1: htb default 10 r2q 9 LF
tc class add dev eth0 parent 1: classid 1:10 htb rate 2mbit burst 3000 LF
tc qdisc add dev eth0 handle 10: parent 1:10 pfifo limit 100 LF
tc qdisc add dev eth1 root handle 1: htb LF
tc class add dev eth1 parent 1: classid 1:10 htb rate 2mbit burst 3000 LF
tc qdisc add dev eth1 parent 1:10 pfifo LF
tc qdisc add dev eth2 root pfifo LF
</DATA></MESSAGE>
```

Server:

```
Protocol:STCCP/1 LF
Length:1639 LF
<MESSAGE><TYPE>REPLY</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;eth0&quot; ip=&quot;192.168.0.66&quot;
netmask=&quot;255.255.255.224&quot; broadcast=&quot;192.168.0.95&quot;
mac=&quot;01:23:45:67:89:0A&quot;&gt;&lt;htbqdisc handle=&quot;1:&quot;
default=&quot;10&quot; r2q=&quot;9&quot;&gt;&lt;htbclass
classid=&quot;1:10&quot; rate=&quot;2mbit&quot; burst=&quot;3000&quot;&gt;
&lt;pfifoqdisc handle=&quot;10:&quot; limit=&quot;100&quot;/&gt;
&lt;/htbclass&gt;&lt;/htbqdisc&gt;&lt;/interface&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;htbqdisc handle=&quot;1:&quot;&gt;
&lt;htbclass classid=&quot;1:10&quot; rate=&quot;2mbit&quot; &gt;
&lt;pfifoqdisc/&gt;&lt;/htbclass&gt;&lt;/htbqdisc&gt;&lt;/interface&gt;
&lt;interface name=&quot;eth2&quot; ip=&quot;192.168.0.68&quot;
netmask=&quot;255.255.255.224&quot; broadcast=&quot;192.168.0.95&quot;
mac=&quot;78:9A:BC:DE:F0:12&quot;&gt;&lt;pfifoqdisc/&gt;&lt;/interface&gt;
&lt;/system&gt;</DATA></MESSAGE>
```

Client:

```
Protocol:STCCP/1 LF
Length:1640 LF
<MESSAGE><TYPE>COMMIT</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;eth0&quot; ip=&quot;192.168.0.66&quot;
netmask=&quot;255.255.255.224&quot; broadcast=&quot;192.168.0.95&quot;
mac=&quot;01:23:45:67:89:0A&quot;&gt;&lt;htbqdisc handle=&quot;1:&quot;
default=&quot;10&quot; r2q=&quot;9&quot;&gt;&lt;htbclass
classid=&quot;1:10&quot; rate=&quot;2mbit&quot; burst=&quot;3000&quot;&gt;
&lt;pfifoqdisc handle=&quot;10:&quot; limit=&quot;100&quot;/&gt;
&lt;/htbclass&gt;&lt;/htbqdisc&gt;&lt;/interface&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;htbqdisc handle=&quot;1:&quot;&gt;
&lt;htbclass classid=&quot;1:10&quot; rate=&quot;2mbit&quot; &gt;
&lt;pfifoqdisc/&gt;&lt;/htbclass&gt;&lt;/htbqdisc&gt;&lt;/interface&gt;
&lt;interface name=&quot;eth2&quot; ip=&quot;192.168.0.68&quot;
netmask=&quot;255.255.255.224&quot; broadcast=&quot;192.168.0.95&quot;
mac=&quot;78:9A:BC:DE:F0:12&quot;&gt;&lt;pfifoqdisc/&gt;&lt;/interface&gt;
&lt;/system&gt;</DATA></MESSAGE>
```


Server:


```
Protocol:STCCP/1 LF
Length:1639 LF
<MESSAGE><TYPE>REPLY</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;eth0&quot; ip=&quot;192.168.0.66&quot;
netmask=&quot;255.255.255.224&quot; broadcast=&quot;192.168.0.95&quot;
mac=&quot;01:23:45:67:89:0A&quot;&gt;&lt;htbqdisc handle=&quot;1:&quot;
default=&quot;10&quot; r2q=&quot;9&quot;&gt;&lt;htbclass classid=
&quot;1:10&quot; rate=&quot;2mbit&quot; burst=&quot;3000&quot;&gt;
&lt;pfifoqdisc handle=&quot;10:&quot; limit=&quot;100&quot;/&gt;
&lt;/htbclass&gt;&lt;/htbqdisc&gt;&lt;/interface&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;htbqdisc handle=&quot;1:&quot;&gt;
&lt;htbclass classid=&quot;1:10&quot; rate=&quot;2mbit&quot; &gt;
&lt;pfifoqdisc/&gt;&lt;/htbclass&gt;&lt;/htbqdisc&gt;&lt;/interface&gt;
&lt;interface name=&quot;eth2&quot; ip=&quot;192.168.0.68&quot;
netmask=&quot;255.255.255.224&quot; broadcast=&quot;192.168.0.95&quot;
mac=&quot;78:9A:BC:DE:F0:12&quot;&gt;&lt;pfifoqdisc/&gt;&lt;/interface&gt;
&lt;/system&gt;</DATA></MESSAGE>
```


Client:


```
Protocol:STCCP/1 LF
Length:44 LF
<MESSAGE><TYPE>QUIT</TYPE><DATA /></MESSAGE>
```

Bibliography

[ASF03a] Apache Software Foundation, The Apache Software License. http://www.apache.org/LICENSE,15.12.2003

[BOU03] Bourret, R., Home Page of Ron Bourret's DTD Parser. 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