QoS Connection Tuning HOWTO

Emmanuel Roger, fly4@prout.be $Revision: 0.59 $ $Date: 2001/12/15 00:37:52 $

This quick howto describes how to enhance your connection speed and response time with QoS, iptables and iproute2.

Contents

1 Introduction

This paper was written because I wanted to listen to shoutcast mp3 streams without being cut and wanted to have a good response time with ssh even when uploading lots of data via ftp or dcc.

Basically it explains how to setup 2 classes of traffic for your upstream to get a more "interactive" connection and why to do it this way.

1.1 What's the problem ?

When you upload data, it gets trough your modem as fast as it can, so if you upload from eg: ftp server, a lot of data packets will be enqueued into the output device queue.

Fine it's the way it must be working, but what if you want to send other interactive traffic at the same time ? It will be enqueued too, but it'll have to wait that all the packet enqueued before are dequeued.

This explains why when you upload at almost your full bandwidth, you get response time of seconds, ping a server when you upload, you'll see.

This makes also ssh and telnet slower, irc lags, when listening to mp3 streams, the music is cut frequently, and when you download some file, the download slows.

1.2 How could I solve this ?

In a perfect world, the OS would detect which traffic needs higher output priority. But we aren't in a perfect world.

So we just divide the traffic into two classes:

Interactive or real-time

This is the kind of traffic which will be given the higher priority, but the less bandwidth.

It is composed of small packets (tcp ack, irc, ssh, telnet, http requests etc...)

Data

This kind of traffic is given the lower priority, but most of the bandwidth.

It is composed of large packets (ftp data, http responses, email with large attachments, pr0n ...)

In opposition to traffic classification by port, ip, or tcp flags, classification by size of packets doesn't need to be redefined every time you add a service to your machine or network and suits well an adsl or cable modem connection.

There are better ways to class different kinds of traffic, but this is easy and seems to work well for the average user.

2 Requirements

Some software is needed to perform this enhancement:

- *Kernel* `http://www.kernel.org` You will need the sources of a 2.4 kernel, the latest ;)
- *iptables* `http://netfilter.samba.org` You need the source to get the `length` match to mark packets so the kernel knows in which class to put them.
- *iproute2* `ftp://ftp.inr.ac.ru/ip-routing/iproute2-current.tar.gz` This tool allows you to control the advanced routing and queueing disciplines of the linux kernel.

3 Compiling and installing

I won't explain compilation process in details, just read the README's and various HOWTO's.

Untar every package, first compile iproute2+tc.

Go into the iptables source directory and "make patch-o-matic". You will be prompted with a list of kernel patches, the one we are interested in is the "length match".

Note that since kernel version 2.4.16 you don't need this step, the length match is included in the official kernel.

After patching, configure your kernel, including:

```
Networking options  --->
    Kernel/User netlink socket
    Network packet filtering (replaces ipchains)
        IP: Netfilter Configuration  --->
            IP tables support (required for filtering/masq/NAT)
            LENGTH match support
            MARK target support
    QoS and/or fair queueing  --->
        select everything here.
```

Either builtin or as modules.

Compile and install as usual.

Next compile iptables and install it too.

reboot.

4 Configure

This configuration example is for an adsl link with 1024 kbits downstream and 512 Kbits upstream. Perhaps you will need fine tuning depending on your connection speed or bandwidth needs.

We will match the packets by their size and mark them acordingly. A packet of size 0 -> 500 will be considered interactive traffic and marked with fwmark value "3"

```
iptables -t mangle -A OUTPUT -m length --length 0:500 -j MARK --set-mark 3
```

Next packets of size 500 -> 1500 will be marked as data traffic with value "4".

```
iptables -t mangle -A OUTPUT -m length --length 500:1500 -j MARK --set-mark 4
```

We will create a root cbq qdisc for our device ppp0:

```
tc qdisc add dev ppp0 root handle 10: cbq bandwidth 10Mbit avpkt 1000 mpu 64
```

This creates a root queuing discipline with handle 10: for the device ppp0 of type cbq, with maximum bandwidth of 10Mbits (pppoe), average packet size will be of 1000 bytes and minimum packet size of 64 (ethernet).

The 2 classes:

- first one will get low initial bandwidth but the higher priority and ability to steal bandwidth from the second.
- second one will limit the rate of the outgoing data to 461 kbits with the less priority.

As a general rule for a workstation connected to internet you would calculate the rate values of the two classes like this:

```
Download bandwidth: 1024 kbits
Upload bandwidth: 512 kbits


Interactive class rate = Download bandwidth / 20
Data class rate = Upload bandwidth - Interactive class rate
```

So 51 and 461.

These values are working well here, perhaps you'll need to fine tune to suit your needs, just make sure to have enough bandwidth in the interactive class to acknowledge all incoming tcp data.

```
tc class add dev ppp0 parent 10:0 classid 10:1 cbq bandwidth 10Mbit rate 51Kbit allot 1514 prio 1 maxburst 10 avpkt 100 isolated
tc class add dev ppp0 parent 10:0 classid 10:2 cbq bandwidth 10Mbit rate 461Kbit allot 1514 prio 8 maxburst 2 avpkt 1500 bounded
```

At least we need to tell the kernel which class we want to send packets to:

```
tc filter add dev ppp0 parent 10:0 protocol ip handle 3 fw flowid 10:1
tc filter add dev ppp0 parent 10:0 protocol ip handle 4 fw flowid 10:2
```

Well that's done.

5 Testing

Testing was realised easily: uploading 2 files on a host with more bandwidth, listening to two mp3 streams simultaneously, and downloading latest kernel from local mirror.

ping time never gets more than 200ms, ssh is interactive (ever tried to type with a 2 sec ping ?) and online gaming works well too.

6 Example script

```
#!/bin/bash
#Example config for an adsl link with 1024Kbit downstream and 128Kbit upstream

iptables -t mangle -A OUTPUT -m length --length 0:500 -j MARK --set-mark 3
iptables -t mangle -A OUTPUT -m length --length 500:1500 -j MARK --set-mark 4

tc qdisc add dev ppp0 root handle 10: cbq bandwidth 10Mbit avpkt 1000 mpu 64
```

```
tc class add dev ppp0 parent 10:0 classid 10:1 cbq bandwidth 10Mbit rate 51Kbit allot 1514 prio 1 maxburst 10 avpkt 100 isolated
tc class add dev ppp0 parent 10:0 classid 10:2 cbq bandwidth 10Mbit rate 77Kbit allot 1514 prio 8 maxburst 2 avpkt 1500 bounded

tc filter add dev ppp0 parent 10:0 protocol ip handle 3 fw flowid 10:1
tc filter add dev ppp0 parent 10:0 protocol ip handle 4 fw flowid 10:2
```

7 Further information

Linux 2.4 Advanced Routing & Traffic Control HOWTO `http://www.ds9a.nl/2.4Routing/`

Sally Floyd *CBQ* `http://www.aciri.org/floyd/cbq.html` papers.

Differentiated Services on Linux `http://diffserv.sf.net`.

8 Changes

- add example script
- corrected formula
- fixed some typos
- added a generic formula to calculate classes rates
- explanation of the qdisc line
- patching not needed with 2.4.16
- added Changelog ;)
- correction of kernel options
- added diffserv url
- license changed to FDL

9 Thanks

Thanks first to the netfilter and kernel teams for providing great software.

Greats also to everyone at *adsl-bc.org* `http://www.adsl-bc.org` who provided feedback.

And to everyone on *undernet* `http://www.undernet.org` #linuxbe and #linuxfr channels.

Emmanuel Roger