

HOWTO Play World of Warcraft and Use BitTorrent^{*†}

David Baird <dbaird@nmt.edu>

November 21, 2005

Contents

1 Terminology	1
2 Prerequisites	1
3 Shaping, Policing, and Queues	1
3.1 shape-iptables.sh	2
3.2 shape-tc-out.sh	2
3.3 shape-tc-in.sh	4
4 Measuring Performance	5
4.1 ssh-ping.py	5
4.2 Results	6
5 Resources	6

1 Terminology

L7 Filter Layer 7 Filter. This is a filter for iptables which inspects the contents of packets to determine what kind of program (e.g. web browser, email, BitTorrent, etc.) created the packet.

SFQ Stochastic Fair Queueing. This is a queue in which packets are re-ordered randomly. This ensures that one type of packet isn't given preferential treatment due to some networking anomaly.

HTB Hierarchical Token Bucket. Allows data to be divided into classes with individual rate limiting and priorities.

tc Traffic Control. This is a Linux command that is used for creating HTB and SFQ structures.

^{*}or, Crash Course on Practical Traffic Shaping in Linux

[†]note that the author himself does not play World of Warcraft

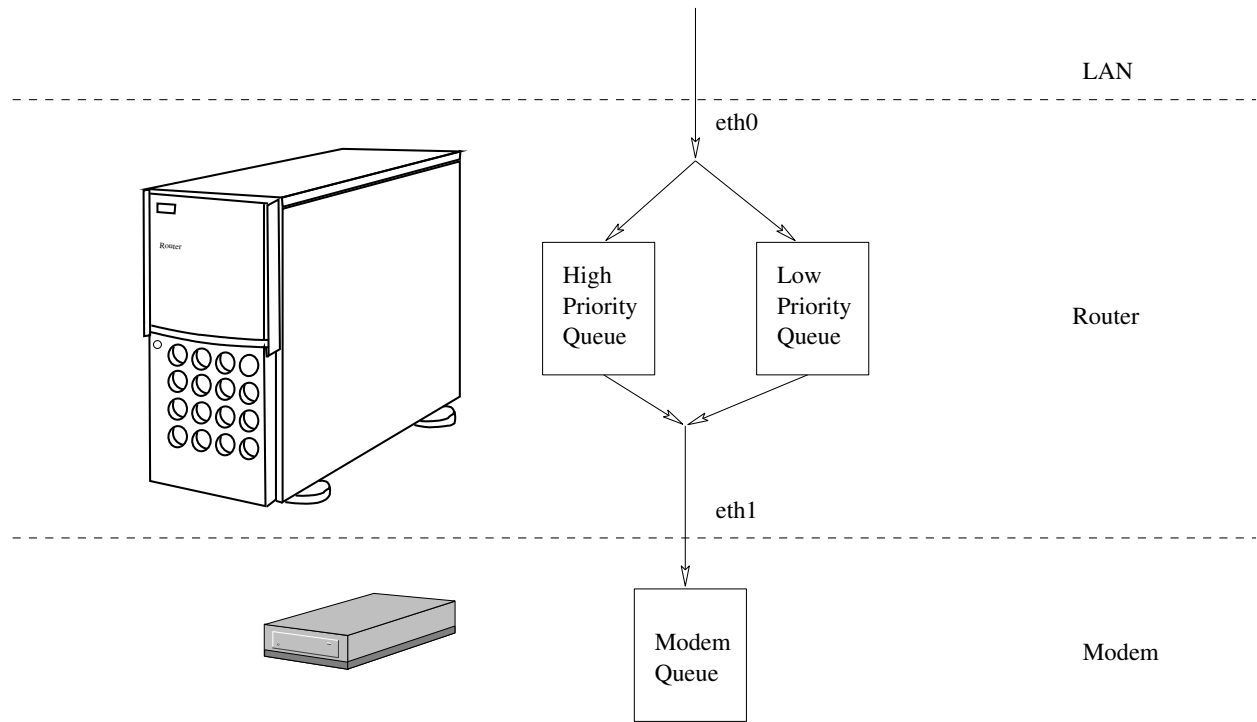


Figure 1: Queues on the router and in the modem. The modem queue is problematic and must be dealt with.

2 Prerequisites

The L7 filter is a standard part of the latest iptables, starting sometime in summer 2005. All that's necessary is to make sure you have a kernel compiled with proper support for advanced routing, iptables, filters, and QoS. I think that standard Debian kernels probably come with everything you need, or a Gentoo genkernel might also be sufficient.

3 Shaping, Policing, and Queues

Traffic shaping is the act of classifying data in order to provide quality of service. Some traffic is considered interactive, and should thus be treated to give the lowest possible latency. For example, SSH might be the most interactive, followed by HTTP, and finally BitTorrent being the least interactive.

If traffic shaping is proactive, then policing is reactive. Policing is the act of discarding or reclassifying traffic that exceeds a maximum rate.

Figure 1 shows queues which packets get piled up in. The DSL modem queue is troublesome because it ruins all the care the router has done about prioritizing packets. DSL modems are designed with large queues to increase their maximum bandwidth, but this has a detrimental effect on latency because it does not provide any quality of service like the router does. Thus, a hack is needed to disable the modem's queue.

To disable the modem's queue, the maximum data rate is reduced at the router. In other words,

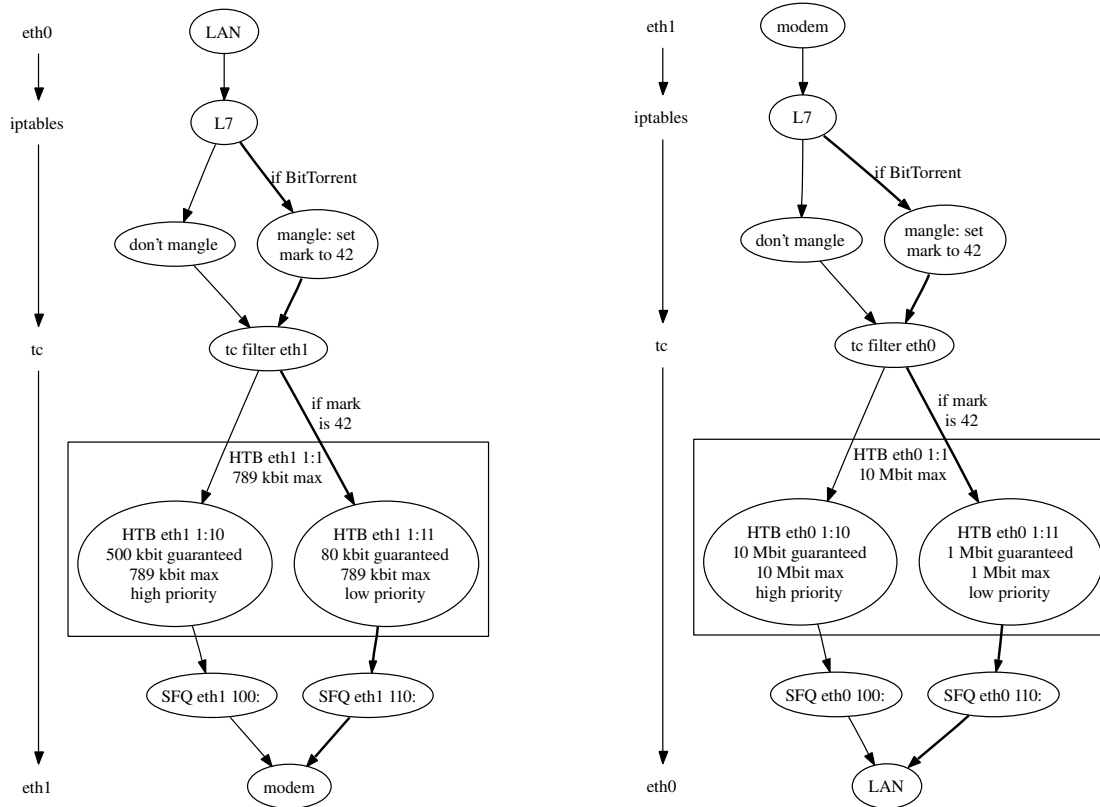


Figure 2: Path packets take when being transmitted (diagram on left) and received (diagram on right). The bold edges indicate the path of BitTorrent packets.

a small amount of bandwidth must be thrown away in order to prevent the modem's queue from filling up (i.e. effectively disabling the modem's queue). The exact amount of bandwidth to throw away is discussed in Section 4.

3.1 shape-iptables.sh

```

1 mark_bt=42 # bittorrent marker
2
3 iptables -t mangle -N bt_mangle
4 iptables -t mangle -A POSTROUTING -m layer7 --l7proto bittorrent -j bt_mangle
5 iptables -t mangle -A bt_mangle -j MARK --set-mark $mark_bt

```

3.2 shape-tc-out.sh

This script is demonstrated by the diagram on the left in Figure 2. See Section 4 for notes about tuning the performance of this script.

```

1 mark_bt=42
2 prio_high=10

```

```

3 prio_low=11
4 # Raw ADSL up/down rate in kbit/sec = 1536/896
5 # ceil should be determined experimentally by measuring latency;
6 # increase ceil until it has a detrimental effect on latency
7 ceil=789 # [kbits/sec] experimentally determined optimal value
8 dev_todsl=eth1
9
10 tc qdisc del dev $dev_todsl root
11 tc qdisc add dev $dev_todsl root handle 1: htb default $prio_high
12 tc class add dev $dev_todsl parent 1: classid 1:1 \
13     htb rate ${ceil}kbit ceil ${ceil}kbit
14 tc class add dev $dev_todsl parent 1:1 classid 1:$prio_high \
15     htb rate 500kbit ceil ${ceil}kbit prio 0
16 tc class add dev $dev_todsl parent 1:1 classid 1:$prio_low \
17     htb rate 80kbit ceil ${ceil}kbit prio 1
18 tc qdisc add dev $dev_todsl parent 1:$prio_high handle ${prio_high}0: \
19     sfq perturb 10
20 tc qdisc add dev $dev_todsl parent 1:$prio_low handle ${prio_low}0: \
21     sfq perturb 10
22 tc filter add dev $dev_todsl parent 1:0 protocol ip prio 1 handle $mark_bt \
23     fw classid 1:$prio_low
24
25 tc -s qdisc show dev $dev_todsl

```

3.3 shape-tc-in.sh

This script is demonstrated by the diagram on the right in Figure 2. This script polices the incoming BitTorrent traffic and prevents it from exceeding 1 Mbit, thus leaving 500 kbits for other traffic. Although this does not guarantee quality of service, this does help improve performance a little bit. True quality of service must be managed on the other end of the DSL which we unfortunately do not have control over.

```

1 mark_bt=42
2 in_ceil=1536 # [kbits/sec]
3 bt_in_ceil=1000 # [kbits/sec]
4 bt_in=$bt_in_ceil
5 prio_high=10
6 prio_low=11
7 dev_tolan=eth0
8
9 tc qdisc del dev $dev_tolan root
10 tc qdisc add dev $dev_tolan root handle 1: htb default $prio_high
11 tc class add dev $dev_tolan parent 1: classid 1:1 htb rate 10mbit ceil 10mbit
12 tc class add dev $dev_tolan parent 1:1 classid 1:$prio_high \
13     htb rate 10mbit ceil 10mbit prio 0
14 tc class add dev $dev_tolan parent 1:1 classid 1:$prio_low \

```

```

15     htb rate ${bt_in}kbit  ceil ${bt_in_ceil}kbit prio 1
16 tc qdisc add dev $dev_tolan parent 1:$prio_high handle ${prio_high}0: \
17     sfq perturb 10
18 tc qdisc add dev $dev_tolan parent 1:$prio_low  handle ${prio_low}0: \
19     sfq perturb 10
20 tc filter add dev $dev_tolan parent 1:0 protocol ip prio 1 handle \
21     $mark_bt fw classid 1:$prio_low
22
23 tc -s qdisc show dev $dev_tolan

```

4 Measuring Performance

The script in Section 3.3 has a variable called `ceil` which is determined experimentally.

4.1 ssh-ping.py

```

1  #!/usr/bin/python
2  # 2005-11-21 04:59:39 UTC  David Baird <dbaird@nmt.edu>
3  import pexpect # http://pexpect.sourceforge.net/
4  import time
5
6  def mean(values):
7      total = 0.0
8      for value in values:
9          total += value
10     return total / len(values)
11
12 # open an interactive SSH connection to a remote computer:
13 ssh = pexpect.spawn('ssh rainbow.nmt.edu cat')
14
15 dt_history = []
16 i = 0
17 while 1:
18     # send the ping:
19     t0 = time.time()
20     print '>>ssh, '%d' % i
21     # wait for the response:
22     ssh.expect('%d' % i)
23     ssh.expect('%d' % i)
24     tf = time.time()
25     # compute and display the ping time:
26     dt = tf - t0
27     dt_history.append(dt)
28     # <stdout> += ping time, average of last 10 ping times:
29     print '%1.5f %1.5f' % (dt, mean(dt_history[-10:]))

```

```
30     i += 1
31     time.sleep(.2)
```

4.2 Results

The following table contains data for a 1.5 Mbit upload, 896 kbit download DSL modem. The “DSL upload” column indicates the value of the `ceil` variable in Section 3.3. The “round trip ping time” values are measured using the `ssh-ping` script in Section 4.1.

round trip ping time [msec]	shaping	DSL download	DSL upload (<code>ceil</code>)	BitTorrent
80	None	Full speed	Full speed	None
500	None	Full speed	Full speed	seed (upload) only
100	L7 + HTB,SFQ	Full speed	789 kbit	seed (upload) only
800	None	Full speed	Full speed	upload and download
350	L7 + HTB,SFQ	?	?	upload and download

5 Resources

<http://lartc.org/howto/> Linux Advanced Routing & Traffic Control HOWTO

<http://www.rns-nis.co.yu/~mps/linux-tc.html> tc - traffic control; Linux QoS control tool

<http://luxik.cdi.cz/~devik/qos/htb/> HTB is packet scheduler. It is currently included in stock Linux kernels from 2.4.20.

<http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm> HTB Linux queuing discipline manual - user guide

<http://sourceforge.net/projects/htbinit/> HTB.init is a shell script derived from CBQ.init that allows for easy setup of HTB-based traffic control on Linux.