

Traffic Shaping Basics

This presentation demonstrates how to shape all traffic as it leaves your server by using the "shaper" tool bundled with RedHat Linux and numerous other distros. Traffic shaping is very handy if you do not want your server to eat up all of your available bandwidth.

We used the configuration we will discuss tonight to shape traffic coming from the following server:

```
http://mirror.datility.net/
```

This is the server that hosts the Datility Networks Mirror Archives. These archives are provided as a public service and include current mirrors of the Linux Documentation Project, Redhat Linux, Fedora Linux and CPU Builders Linux.

You can also use a combination of "iptables" and "tc" to perform more advanced traffic shaping at your Linux firewall/router device on a per protocol and/or per host basis.

Traffic Shaping - Implementation Summary

```
apt-get install shaper

insmod shaper
shaper attach shaper0 eth0
shaper speed shaper0 256000

ifconfig shaper0 192.168.111.13 netmask 255.255.255.0 broadcast 192.168.111.255 up
route del -net 192.168.111.0/24 eth0

route delete default eth0
route add default gw 192.168.111.254 shaper0

route del -net 192.168.111.0/24 eth0
```

Traffic Shaping - Implementation Details

```
# Install "shaper" package
[root@mirror opt]# apt-get install shaper

# Start "shaper" kernel module
[root@mirror opt]# insmod shaper

# Attach a shaper ("shaper0") to existing interface ("eth0")
[root@mirror opt]# shaper attach shaper0 eth0

# Configure shaper to allow maximum speed (256kbps)
[root@mirror opt]# shaper speed shaper0 256000

# What do interfaces look like before we proceed?
[root@mirror opt]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:C0:4F:98:8D:BD
          inet addr:192.168.111.13  Bcast:192.168.111.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8287365  errors:0  dropped:0  overruns:0  frame:0
          TX packets:4528896  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:100
```

```
RX bytes:3472079981 (3311.2 Mb) TX bytes:378852151 (361.3 Mb)
Interrupt:12 Base address:0xdc40
```

```
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:16436 Metric:1
        RX packets:615 errors:0 dropped:0 overruns:0 frame:0
        TX packets:615 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:36822 (35.9 Kb) TX bytes:36822 (35.9 Kb)
```

```
# What does routing table look like before we proceed?
```

```
[root@mirror opt]# route -n
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.111.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
169.254.0.0	0.0.0.0	255.255.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	192.168.111.254	0.0.0.0	UG	0	0	0	eth0

```
# Notes before we proceed:
```

```
#
```

```
# Can download ISO image from HTTP server via 10mb LAN at ~900KBps (7.2mbps)
```

```
# SSH session seems to be unaffected by the transfer.
```

```
# Configure shaper device "shaper0" using same IP/NM/BC as existing interface
```

```
[root@mirror opt]# ifconfig shaper0 192.168.111.13 netmask 255.255.255.0 \
        broadcast 192.168.111.255 up
```

```
[root@mirror opt]# route -n
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.111.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.111.0	0.0.0.0	255.255.255.0	U	0	0	0	shaper0
169.254.0.0	0.0.0.0	255.255.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	192.168.111.254	0.0.0.0	UG	0	0	0	eth0

```
# Remove unnecessary routing entry
```

```
[root@mirror opt]# route del -net 169.254.0.0/16
```

```
[root@mirror opt]# route -n
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.111.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.111.0	0.0.0.0	255.255.255.0	U	0	0	0	shaper0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	192.168.111.254	0.0.0.0	UG	0	0	0	eth0

```
# Remove old default route (via eth0)
```

```
#
```

```
# DANGER: Do not do this unless you are on same subnet or console. Otherwise,
```

```
# you will not be able to communicate with your system!!! If you must
```

```
# do this from different subnet, add default route (below) before you
```

```
# delete the existing default route!
```

```
#
```

```
[root@mirror opt]# route delete default eth0
```

```
[root@mirror opt]# route -n
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.111.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.111.0	0.0.0.0	255.255.255.0	U	0	0	0	shaper0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo

```
# Create new default route (via shaper0)
[root@mirror opt]# route add default gw 192.168.111.254 shaper0
[root@mirror opt]# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.111.0    0.0.0.0         255.255.255.0   U        0      0      0 eth0
192.168.111.0    0.0.0.0         255.255.255.0   U        0      0      0 shaper0
127.0.0.0        0.0.0.0         255.0.0.0       U        0      0      0 lo
0.0.0.0          192.168.111.254 0.0.0.0         UG       0      0      0 shaper0

# Remove direct route to local subnet
#
# HINT: Keep this route if you'd like to place NO bandwidth restrictions on
#       the local subnet. This entry can be used to bypass the shaper.
#
[root@mirror opt]# route del -net 192.168.111.0/24 eth0
[root@mirror opt]# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.111.0    0.0.0.0         255.255.255.0   U        0      0      0 shaper0
127.0.0.0        0.0.0.0         255.0.0.0       U        0      0      0 lo
0.0.0.0          192.168.111.254 0.0.0.0         UG       0      0      0 shaper0

# Notes after we are finished:
#
# Can download ISO image from HTTP server via 10mb LAN at ~40KBps (320kbps)
# SSH session seems to be unaffected by the transfer.

# What do interfaces look like after we are finished?
[root@mirror opt]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:C0:4F:98:8D:BD
          inet addr:192.168.111.13  Bcast:192.168.111.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8287365 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4528896 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:3472079981 (3311.2 Mb)  TX bytes:378852151 (361.3 Mb)
          Interrupt:12 Base address:0xdc40

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:615 errors:0 dropped:0 overruns:0 frame:0
          TX packets:615 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:36822 (35.9 Kb)  TX bytes:36822 (35.9 Kb)

shaper0   Link encap:Ethernet  HWaddr 00:00:00:00:00:00
          inet addr:192.168.111.13  Mask:255.255.255.0
          UP RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2021 errors:0 dropped:0 overruns:0 carrier:0
          collisions:11 txqueuelen:10
          RX bytes:0 (0.0 b)  TX bytes:2720582 (2.5 Mb)

# What does routing table look like after we are finished?
[root@mirror opt]# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.111.0    0.0.0.0         255.255.255.0   U        0      0      0 shaper0
127.0.0.0        0.0.0.0         255.0.0.0       U        0      0      0 lo
0.0.0.0          192.168.111.254 0.0.0.0         UG       0      0      0 shaper0
```

Traffic Shaping - Implementation Summary (HOWTO UNDO)

```
[root@mirror opt]# route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.111.0    0.0.0.0         255.255.255.0   U      0      0      0 shaper0
127.0.0.0        0.0.0.0         255.0.0.0       U      0      0      0 lo
0.0.0.0          192.168.111.254 0.0.0.0         UG     0      0      0 shaper0

[root@mirror opt]# route add -net 192.168.111.0/24 eth0
[root@mirror opt]# route add -net 169.254.0.0/16 eth0
[root@mirror opt]# route add default gw 192.168.111.254 eth0
[root@mirror opt]# route del default shaper0
[root@mirror opt]# route del -net 192.168.111.0/24 shaper0

[root@mirror opt]# route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.111.0    0.0.0.0         255.255.255.0   U      0      0      0 eth0
169.254.0.0      0.0.0.0         255.255.0.0     U      0      0      0 eth0
127.0.0.0        0.0.0.0         255.0.0.0       U      0      0      0 lo
0.0.0.0          192.168.111.254 0.0.0.0         UG     0      0      0 eth0

[root@mirror opt]# ifconfig shaper0 192.168.111.13 netmask 255.255.255.0 broadcast
192.168.111.255 down
[root@mirror opt]# rmmode shaper
```

Traffic Shaping - Implementation Detailed Explanation

The Linux traffic shaper

Linux is well known for its ability to pump bits down a network wire. Few, if any, other systems can match its ability to make use of available network bandwidth. Given that, the "traffic shaper," which limits bandwidth out a network interface, seems a bit misconceived. Why hobble a racehorse?

Nonetheless, situations do arise where one wants to put some limits on what a Linux box can put out. Maybe the system lives at an ISP's co-location hub and bandwidth is metered (and expensive). Or a user may simply not want to allow an FTP server to take up a company's entire leased line to the net. Or the need may arise to clamp down on the usage of one particular, problematic site.

Until recently there has been no ability to throttle bandwidth usage in the stable Linux kernel series. The 2.1 development kernels have had the traffic shaper for a while, but it's only with 2.0.36 that this driver has been added to the stable kernel (thanks to a backporting effort by Alan Cox). The traffic shaper is a simple device, but it gives fairly precise control over your system's outgoing network traffic (incoming traffic is rather harder to control, and is not affected directly by the traffic shaper).

To use the traffic shaper facility, you'll need (1) a suitably configured 2.0.36 or 2.1 kernel, and (2) the "shaper" utility. Happily for the author, Red Hat 5.2 includes both right out of the box, though little associated documentation is included. In fact, about the only shaper documentation in existence is (1) the source, and (2) this file that comes with the kernel source. Fortunately, the shaper is a fairly simple thing to configure.

The first step is to configure your networking normally. The traffic shaper creates a pseudo networking device that is used by the system, but it relies on the underlying

ethernet (or whatever) interface to actually carry the traffic. Once you set up the shaper interface, do not shut down the underlying physical interface, or unpleasant things will happen. This restriction would appear to make the traffic shaper unsuited for non-dedicated connections, such as PPP links.

The traffic shaper must be built as a kernel module. If you want the module to autoload with kernel, you will need to add a line like:

```
alias shaper0 shaper
```

to `/etc/conf.modules`. Either that or just use `insmod` at boot time to load the shaper module. The `insmod` command, like those that follow, can be placed in `/etc/rc.d/rc.local` (on Red Hat systems) to have it executed at boot time. The next step is to attach the shaper device to the physical interface, and to set the speed limit. That is done with a pair of commands like:

```
/sbin/shapecfg attach shaper0 eth0  
/sbin/shapecfg speed shaper0 64000
```

The first line hooks the pseudo-interface "shaper0" onto the real interface "eth0" - the first ethernet interface. The second sets the maximum speed to 64 Kbits per second. According to the documentation, the workable range is from 9600 to 256,000 bits per second. I have verified that higher speeds work pretty well (i.e. the metering is accurate), but the traffic is bursty. Then you need to configure the shaper0 interface for networking, using the usual sort of `ifconfig` and `route` commands.

```
ifconfig shaper0 host netmask mask broadcast bcast up  
route add -net net netmask mask dev shaper0
```

Of course, all of the parameters in italics must be replaced with suitable values for your network. The host/address, mask, and broadcast parameters need to match those of the underlying physical interface.

At this point, assuming you have left your system's normal networking setup in place, you likely have two different routes for your local network. Things seem to work that way, but it's inelegant. You may want to delete the route pointing directly at the physical interface. You may also want to change the default route out of your system, assuming you set one statically, so that traffic for the world goes through the shaper:

```
route delete default eth0  
route add default gw gateway shaper0
```

That's all it takes. An easy, if imprecise, test to verify that the shaper is working for you is to simply FTP a large file from another machine. FTP kindly prints out the bandwidth it gets; in the absence of other interference on the net you should get something very close to the value you gave in the `shapecfg` command.

Advanced use of the traffic shaper could involve the use of IP alias interfaces and fancy routes to impose limits on traffic to specific sites. There is also a patch by Mike McLagan on ftp.linux.org which allows setting point to point routes which allows even more fine per-host control of bandwidth use with the traffic shaper.

Information Source:
<http://lwn.net/1998/1119/shaper.html>