LINUX TRAFFIC SHAPING - BASICS

Traffic shaping is another one of those tools which make our lives a lot easier. It allows us to ensure that the traffic which is important will go through while allowing us to penalize non-critical traffic. As always, Linux has traffic shaping capabilities built in. In this article we will be going through the basics by looking at a sample script, but even this should allow you to start meaningfully segmenting your traffic.

Getting Started
First off you will need a linux firewall through which all your data traffic flows, this gives you a single point at which you can do traffic shaping. Now the start of our script looks like this..

```
TC=/sbin/tc
EXT=eth1
```

Here we set the variables we will be using in the rest of the script. You will see that the *TC* variable is set to */sbin/tc*. The *tc* command comes with the *iproute2* package and it is the executable we will use to setup our shaping. Next we define the network interface on which we want t do shaping. Now bear in mind that we will generally only shape outgoing traffic, because thats what we have control over, unlike traffic coming in. Next we do..

```
$TC qdisc del dev $EXT root
```

Here you see the previously declared variables being used, and what we are doing in this line is clearing away any old rules. This is always a good idea so that any settings you make are made without any previous settings conflicting.

Setup the Main Segments
Now we start getting into the more detailed stuff. The first thing we do is create a *qdisc* (queuing discipline) that all our other settings will use..

```
$TC qdisc add dev $EXT root handle 1:0 htb default 90
```

Lets step through the line word for word; the *tc* command is using the *qdisc* functionality to add a setting to the specified *dev* (device) this setting is called *root* and is referenced by the *handle 1:0*, it will use the queuing discipline of *htb* and the rule for *default* traffic is the one specified later on by the handle *90*. Next we create the traffic class in this *qdisc* you have just setup..

```
$TC class add dev $EXT parent 1:0 classid 1:1 htb rate 350kbit ceil 350kbit
```

Here the *tc* command is using the *class* functionality to add a setting to the specified *dev* (device), the *parent* for this setting is the setting using the handle *1:0*, the handle/identifier for this setting is *1:1* using the *htb* queuing discipline and the line *rate* is *350kbit* with a maximum *ceiling* of *350kbit*. When you specify your *rate* and *ceiling* it is best to use a figure that is about 10% lower then your total bandwidth. What you have now done is created a root *qdisc* that the rest of the settings will use, and the main class setting sets the total segment which you will slice up.

Slicing the Bandwidth
Now that we have the main settings we can setup the bandwidth segments as we want..

```
$TC class add dev $EXT parent 1:1 classid 1:5 htb rate 50kbit ceil 50kbit
$TC class add dev $EXT parent 1:1 classid 1:10 htb rate 200kbit ceil 200kbit
```

I will just go through the first line here as the second one follows the same idea. The *tc* command is using the *class* functionality to *add* a setting to the specified *dev* (device) which has a *parent* with the handle of *1:1*, and the identifier for this setting is *1:5*. This setting uses the *htb* queuing discipline, and the total bandwidth allowed through this segment is a *rate 50kbit*, with a maximum *ceiling* of *50kbit* (your *ceiling* can be higher than your *rate*, but try keeping it simple for now). Assuming these are the only two specific slices we want to make, we then next to create out default class..

```
$TC class add dev $EXT parent 1:1 classid 1:90 htb rate 350kbit ceil 350kbit
```

The main thing to see here is that the default slice I have setup to use the full bandwidth, you do not have to do this, many people use whatever is left over after the rest of the bandwidth allocations. Also, in my main initial *qdisc* setup I specified a default segment of *90*.

Classing the Bandwidth Slices
Now we declare *qdisc* settings for all the segmented bandwidth *class* settings, including the default one..

```
$TC qdisc add dev $EXT parent 1:5 handle 10:0 sfq perturb 10
$TC qdisc add dev $EXT parent 1:10 handle 20:0 sfq perturb 10
$TC qdisc add dev $EXT parent 1:90 handle 90:0 sfq perturb 10
```

Again I will only go through the first line as they all use the same type of settings. The *tc* command is using the *qdisc* functionality to *add* a setting to the specified *dev* (device) using the *parent* with the identifier of *1:5*, while this settings *handle*/identifier is *10:0*. This setting uses the *sfq* setting with a *perturb* setting of *10* (these settings you should not really have to change). Again notice that the last line is for the default traffic and uses the *handle* of *90:0*

Putting Traffic into the Segments
Ok, we have created all our bandwidths segments using the *classes* and the *qdiscs*, now lets use them. There are a couple of ways to classify packets, but I will carry on using the *tc* command for the purposes of this article..

```
$TC filter add dev $EXT parent 1:0 protocol ip u32 match ip dport 80 0xffff classid 1:5
$TC filter add dev $EXT parent 1:0 protocol ip u32 match ip dport 1863 0xffff classid
1:5
$TC filter add dev $EXT parent 1:0 protocol ip u32 match ip dport 25 0xffff classid
1:10
```

Using the first line as an example; the *tc* command is using the *filter* functionality to *add* a setting to the specified *dev* (device) the *parent* which this filter sits under is the one with a handle of *1:0*. We are looking at the *protocol* suite of *ip*. We will use the *u32* functionality to match the *ip dport* (destination port) of *80*, and send it to the *class* with the identifier/handle of *1:5*. The *u32* section can also match source port, specified hosts, etc. Also you will see that in the example I was pushing port 80 and port 1863 through the 50kbit section, while port 25 was going through the 200kbit segment. This means that any other traffic would use the default bandwidth segment.

Final Words
Traffic shaping is a very powerful tool, and it can really help us control our networks. But remember this article only covered off the basics of what you can do, but even with this you can start making a difference. As always, have fun and learn.