Quick Start Guide for iperf and tc

By Steven Gordon on Wed, 19/01/2011 - 9:09am

A simple application to test network performance (throughput) is `iperf`. Basically, you run `iperf` as a server on one computer, and then as a client on another computer. The software sends packets between client and server to see measure the throughput. Read on to see the basic commands or view the video to see slightly more detailed demo (including both UDP and TCP tests). Note that video does not show `tc` in use - read on to see `tc` explained.

On your server computer, you need an accessible IP address. If both client and server computers are inside the same LAN then that is easy: use `ifconfig` to find your servers IP. However if your server is behind a NAT/firewall device (e.g. your computer on your home ADSL connection), accessing it from outside (e.g from work) may be a problem. Port forwarding is probably needed (you'll have to work that out yourself).

On the server computer run `iperf` in server mode:

```
$ iperf -s
```

Now on the client computer, if the server has IP 192.168.1.5:

```
$ iperf -c 192.168.1.5
```

By default `iperf` runs a 10 second TCP test and then reports throughput ('bandwidth') at both client and server. Here's an example output at the client:

```
------------------------------------------------------------
Client connecting to 192.168.1.5, TCP port 5001
```

```
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.116 port 47873 connected with 192.168.1.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec    115 MBytes  95.8 Mbits/sec
```

Read the man page and online instructions (e.g. 1 [3] 2 [4]) for using different iperf options (e.g. -u for UDP, -t for test time, -p for server port, -b for UDP sending rate).

In Linux, `tc` is a traffic control application. It allows you to control packets being sent by your computer. You can use `tc` to do many things - here we will only use it to delay and drop packets. The netem [5] website gives more details and examples for these operations, while the Linux Advanced Routing & Traffic Control Howto [6] covers `tc` in depth.

In the following command it is assumed your Ethernet interface is `eth0` - you can find out using `ifconfig`.

To add a random delay of between 90 and 110ms (average 100ms) to every packet your computer sends:

```
$ sudo tc qdisc add dev eth0 root netem delay 100ms 10ms
```

(Note that `sudo` is needed, as only a privileged user can modify the kernel operation.)

To test that it works, `ping` another computer and record the average RTT. To remove this delay, use the *del* option:

```
$ sudo tc qdisc del dev eth0 root netem delay 100ms 10ms
```

Now perform the ping again and you should see the difference in RTT from the delay introduced by `tc`.

To drop 1% of packets (randomly):

```
$ sudo tc qdisc add dev eth0 root netem loss 1%
```

And again you can remove it by replacing *add* with *del*. Also you can change existing commands by using *change* instead of *add* and *del*.

For more details about potential problems of dropping packets with `tc`, and alternatively using `iptables`, see my writeup here [7].

Content: Howto [8]
Interest: Ubuntu Linux [9]
iperf [10]
tc [11]

Source URL: http://sandilands.info/sgordon/quick-start-guide-for-iperf-and-tc

Links:

[1] http://sandilands.info/sgordon/quick-start-guide-for-iperf-and-tc
[2] http://sandilands.info/sgordon/user/2
[3] http://openmaniak.com/iperf.php
[4] http://sourceforge.net/projects/iperf/
[5] http://www.linuxfoundation.org/collaborate/workgroups/networking/netem
[6] http://lartc.org/howto/
[7] http://sandilands.info/sgordon/dropping-packets-in-ubuntu-linux-using-tc-and-iptables
[8] http://sandilands.info/sgordon/taxonomy/term/212
[9] http://sandilands.info/sgordon/taxonomy/term/302
[10] http://sandilands.info/sgordon/taxonomy/term/306
[11] http://sandilands.info/sgordon/taxonomy/term/312