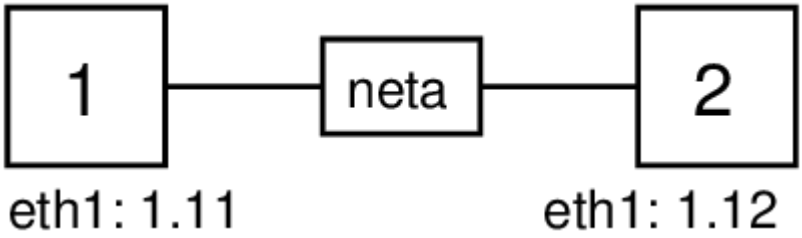
Setting Link Data Rate and Delay in a Linux Virtual Network

Submitted by Steve on Sun, 26/10/2014 - 9:48am

virtnet [1] allows you to quickly setup multiple Linux virtual machines and connect them in a pre-selected topology. Assuming you have installed virtnet [2] and created a topology [3], the following shows how to set the data rate and delay for the links in your virtual network. I use topology 2, shown below, as an example.



Square 1 represents one Linux virtual machine called node1 (with IP address 192.168.1.11) and square two is node2 (192.168.1.12). neta represents a virtual switch; you don't have access to this, as it is internal to VirtualBox. The best way to understand this is to assume there is a link between node1 and node2. Here I will show you how to use software on each of the Linux nodes to emulate link data rate and delay. The software is called tc [4], but I have provided are simpler script called vn-link. You use it to set the outgoing link characteristics on each node.

Before we set the data rate and delay, lets first measure the current values. To measure delay, use ping:

```
network@node1:~$ ping -c 192.168.1.12
PING 192.168.1.12 (192.168.1.12) 56(84) bytes of data.
64 bytes from 192.168.1.12: icmp_req=1 ttl=64 time=0.859 ms
64 bytes from 192.168.1.12: icmp_req=2 ttl=64 time=1.13 ms
64 bytes from 192.168.1.12: icmp_req=3 ttl=64 time=1.06 ms
64 bytes from 192.168.1.12: icmp_req=4 ttl=64 time=1.04 ms
64 bytes from 192.168.1.12: icmp_req=5 ttl=64 time=1.05 ms

--- 192.168.1.12 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 0.859/1.031/1.131/0.095 ms
```

You see the delay is quite small - about 1 ms. This is the delay incurred by the VirtualBox software. Now lets test the data rate by transferring packets across the link and measuring the throughput. I will use iperf to do this (see my quick start guide for iperf [5]). First start the server on node2:

```
network@node2:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

Now start the client on node1:

```
network@node1:~$ iperf -c 192.168.1.12
------------------------------------------------------------
Client connecting to 192.168.1.12, TCP port 5001
TCP window size: 21.6 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.11 port 42172 connected with 192.168.1.12 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  2.02 GBytes  1.73 Gbits/sec
```

After the 10 second test, you notice the "bandwidth" (which means throughput) is very high (in my case about 1.7 Gb/s; yours may differ) . This throughput is limited by the host computer: the VirtualBox software, and possibly your CPU, hard disk and memory.

Now lets set the data rate and delay to specific values. For example, to set the data rate and delay of the link from node1 to node2 to 1 Mb/s and 10 ms respectively, do the following on node1:

```
network@node1:~$ sudo bash ~/virtnet/bin/vn-link --interface eth1 --set \
--rate 1000kbit --delay 10.0ms
Old values
  Rate: not set
  Delay: not set
  Jitter: not set
RTNETLINK answers: No such file or directory
New values
  Rate: 1000kbit
  Delay: 10.0ms
  Jitter: not set
```

`vn-link` is just a Bash script that calls tc. The above command assumes it is in the directory `virtnet/bin` in your home directory. Alternatively you may `cd` into the directory and run it from there. Don't forget to use `sudo`, as it is necessary to run with administrator privileges to change the interface options/

And now to set the data rate and delay of the link from node2 to node1 to 100 Mb/s and 5 ms respectively, do the following on node2:

```
network@node2:~$ sudo bash ~/virtnet/bin/vn-link --interface eth1 --set \
--rate 100000kbit --delay 5.0ms
Old values
  Rate: not set
  Delay: not set
  Jitter: not set
RTNETLINK answers: No such file or directory
New values
  Rate: 100000kbit
  Delay: 5.0ms
  Jitter: not set
```

To be sure that everything is setup correctly, you can test the links using `ping` and `iperf`. First

test the delay:

```
network@node1:~$ ping -c 5 192.168.1.12
PING 192.168.1.12 (192.168.1.12) 56(84) bytes of data.
64 bytes from 192.168.1.12: icmp_req=1 ttl=64 time=16.9 ms
64 bytes from 192.168.1.12: icmp_req=2 ttl=64 time=16.0 ms
64 bytes from 192.168.1.12: icmp_req=3 ttl=64 time=16.0 ms
64 bytes from 192.168.1.12: icmp_req=4 ttl=64 time=16.2 ms
64 bytes from 192.168.1.12: icmp_req=5 ttl=64 time=16.2 ms

--- 192.168.1.12 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 16.062/16.310/16.929/0.337 ms
```

You see the round-trip-time (RTT) is about 16 ms. That consists of the 10 ms delay we added for node1 to node2, the 5 ms delay we added for node2 to node 1 and the 1 ms delay due to VirtualBox. So if you want a 50 ms RTT, then you could set the link delays to be 24.5 ms each. Lets try. On node 1 (for the link to node2):

```
network@node1:~$ sudo bash ~/virtnet/bin/vn-link --interface eth1 --set \
--rate 1000kbit --delay 24.5ms
Old values
  Rate: 1000Kbit
  Delay: 10.0ms
  Jitter: not set
New values
  Rate: 1000kbit
  Delay: 24.5ms
  Jitter: not set
```

And on node2 (for the link to node1):

```
network@node1:~$ sudo bash ~/virtnet/bin/vn-link --interface eth1 --set \
--rate 100000kbit --delay 24.5ms
Old values
  Rate: 100000Kbit
  Delay: 5.0ms
  Jitter: not set
New values
  Rate: 100000kbit
  Delay: 24.5ms
  Jitter: not set
```

And ping again:

```
network@node1:~$ ping -c 5 192.168.1.12
PING 192.168.1.12 (192.168.1.12) 56(84) bytes of data.
64 bytes from 192.168.1.12: icmp_req=1 ttl=64 time=50.5 ms
64 bytes from 192.168.1.12: icmp_req=2 ttl=64 time=50.4 ms
64 bytes from 192.168.1.12: icmp_req=3 ttl=64 time=50.4 ms
64 bytes from 192.168.1.12: icmp_req=4 ttl=64 time=50.3 ms
64 bytes from 192.168.1.12: icmp_req=5 ttl=64 time=50.6 ms
```

```
--- 192.168.1.12 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 50.337/50.489/50.614/0.171 ms
```

You see a RTT of about 50 ms. So you can control the overall RTT, to an accuracy of within 1 or 2 ms, by setting the delays on each link.

Finally, lets test the throughput. One node2 start the iperf server:

```
network@node2:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

Now start the client on node1:

```
network@node1:~$ iperf -c 192.168.1.12
------------------------------------------------------------
Client connecting to 192.168.1.12, TCP port 5001
TCP window size: 21.6 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.11 port 42173 connected with 192.168.1.12 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-11.6 sec  1.62 MBytes  1.18 Mbits/sec

------------------------------------------------------------
```

We see the throughput of 1.18 Mb/s, which is close the data rate set from node1 to node2 of 1 Mb/s. You note that the throughput is slightly above the data rate. The reason is that our emulation of the data rate using tc and vn-link is not exact. However from my limited testing it is accurate within about 100 or 200 kb/s when setting the link data rate from 1 Mb/s up to 10 Mb/s. This is sufficient for simple tests.

`vn-link` is work-in-progress, and really was meant to hide the details of `tc`. Two other useful options supported are to include jitter (variance of the delay) by adding the option `--jitter 1.0ms` to the above commands, and clearing the values (returning to the defaults) by running:

```
network@node1:~$ sudo bash ~/virtnet/bin/vn-link --interface eth1 --clear
Current values
  Rate: not set
  Delay: not set
  Jitter: not set
```

If you want to set other characterstics, like the link error rate, then I suggest learning `tc` and `iptables` - see my quick start guide [5] and how to drop packets [6].

Content: Howto [7]
Interest: Networking [8]
VirtualBox [9]