NETWORK TOOLS AND PROTOCOLS

Lab 3: Emulating WAN with NETEM I: Latency, Jitter

Document Version: 06-14-2019

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab introduces NETEM and explains how it can be used to emulate real-world scenarios while having control on parameters that affect the performance of networks. Network parameters include latency, jitter, packet loss, reordering, and corruption. Correlation values between network parameters will also be set to provide a more realistic network environment.

Objectives

By the end of this lab, students should be able to:

1. Understand delay in networks and how to measure it.
2. Understand Linux queuing disciplines (qdisc) architecture.
3. Deploy emulated WANs characterized by large delays using NETEM and Mininet.
4. Perform measurements after introducing delays to an emulated WAN.
5. Deploy emulated WANs characterized by delays, jitters, and corresponding correlation values.
6. Modify the delay distribution of an emulated WAN.

Lab settings

The information in Table 1 provides the credentials of the machine containing Mininet.

Table 1**.** Credentials to access Client1 machine.

Device	**Account**	**Password**
Client1	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction to network emulators and NETEM.
2. Section 2: Lab topology.
3. Section 3: Adding/changing delay to emulate a WAN.
4. Section 4: Restoring original values (deleting the rules).
5. Section 5: Adding jitter to emulated WAN.
6. Section 6: Adding correlation value for jitter and delay.
7. Section 7: Delay distribution.

1 Introduction to network emulators and NETEM

Network emulators play an important role for the research and development of network protocols and applications. Network emulators provide the ability to perform tests of realistic scenarios in a controlled manner, which is very difficult on production networks. This is particularly complex for researchers who develop and test tools for *Wide Area Networks (WANs)* and for multi-domain environments.

1.1 NETEM

One of the most popular network emulators is *NETEM*[1,2], a Linux network emulator for testing the performance of real applications over a virtual network. The virtual network may reproduce long-distance WANs in the lab environment. These scenarios facilitate the test and evaluation of protocols and devices from the application layer to the data-link layer under a variety of conditions. NETEM allows the user to modify parameters such as delay, jitter, packet loss, duplication and re-ordering of packets.

NETEM is implemented in Linux and consists of two portions: a small kernel module for a queuing discipline and a command line utility to configure it. Figure 1 shows the basic architecture of Linux queuing disciplines. The queuing disciplines exist between the IP protocol output and the network device. The default queuing discipline is a simple packet first-in first-out (FIFO) queue. A queuing discipline is a simple object with two interfaces. One interface queues packets to be sent and the other interface releases packets to the network device. The queuing discipline makes the policy decision of which packets to send, which packets to delay, and which packets to drop. A classful queueing discipline, such as NETEM, has configurable internal modules.

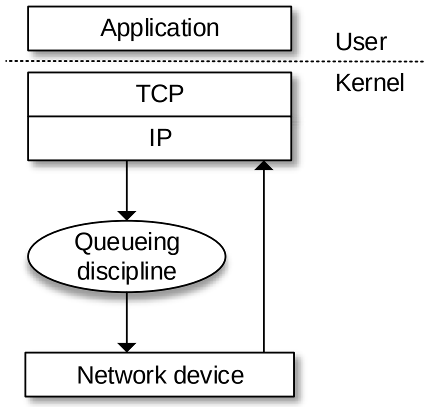


Figure 1. Linux queueing discipline.

1.2 WANs and delay

In networks, there are several processes and devices that contribute to the end-to-end delay between a sender node and a destination node. Many times, the end-to-end delay is dominated by the WAN's propagation delay. Consider two adjacent switches A and B connected by a WAN. Once a bit is pushed onto the WAN by switch A, it needs to propagate to switch B. The time required to propagate from the beginning of the WAN to switch B is the propagation delay. The bit propagates at the propagation speed of the WAN's link. The propagation speed depends on the physical medium (that is, fiber optics, twisted-pair copper wire, etc) and is in the range of 2×10^8 meters/sec to 3×10^8 meters/sec, which is equal to, or a little less than, the speed of light. The propagation delay is the distance between two switches divided by the propagation speed. Once the last bit of the packet propagates to switch B, it and all the preceding bits of the packet are stored in switch B[3].

Network tools usually estimate delay for troubleshooting and performance measurements. For example, an estimate of end-to-end delay is the Round-Trip Time (RTT), which is the time it takes for a small packet to travel from sender to receiver and then back to the sender. The RTT includes packet-propagation delays, packet-queuing delays in intermediate routers and switches, and packet-processing. As mentioned above, if the propagation delay dominates other delay components (as in the case of many WANs), then RTT is also an estimate of the propagation delay.

2 Lab topology

Let's get started with creating a simple Mininet topology using MiniEdit. The topology uses 10.0.0.0/8 which is the default network assigned by Mininet.

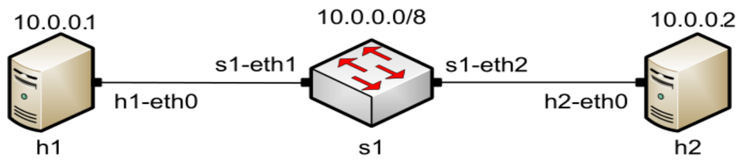


Figure 2. Lab topology.

Step 1. A shortcut to MiniEdit is located on the machine's Desktop. Start MiniEdit by clicking on MiniEdit's shortcut. When prompted for a password, type `password`.

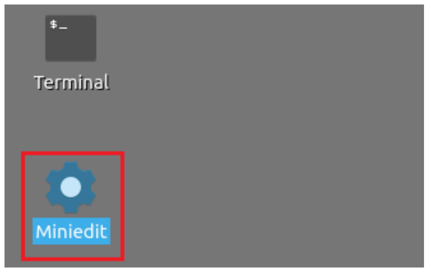


Figure 3. MiniEdit shortcut.

Step 2. On MiniEdit's menu bar, click on *File* then *Open* to load the lab's topology. Locate the *Lab 3.mn* topology file and click on *Open*.

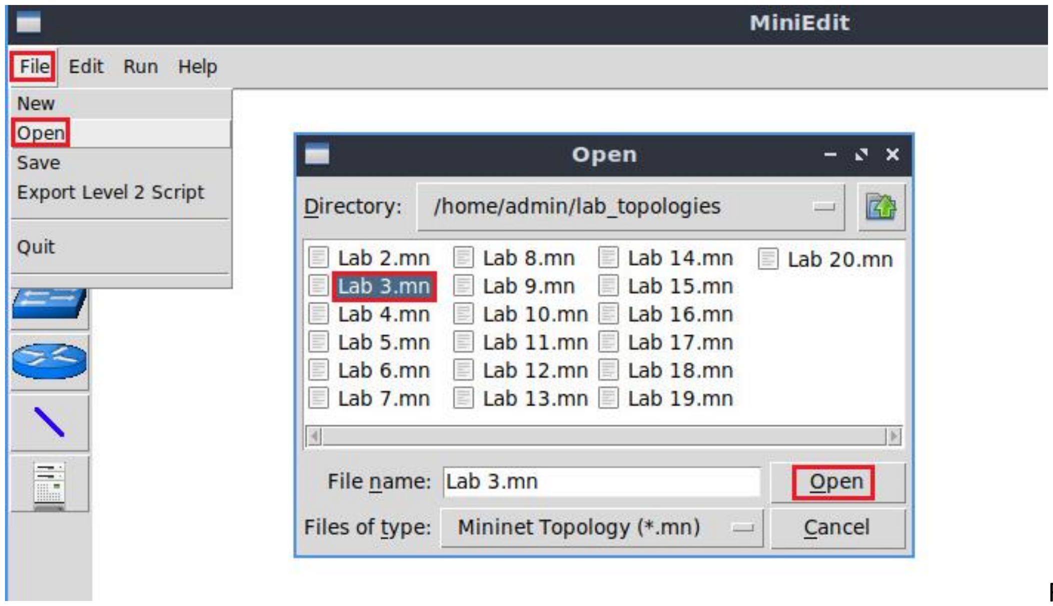


Figure 4. MiniEdit's *Open* dialog.

Step 3. Before starting the measurements between host h1 and host h2, the network must be started. Click on the *Run* button located at the bottom left of MiniEdit's window to start the emulation.

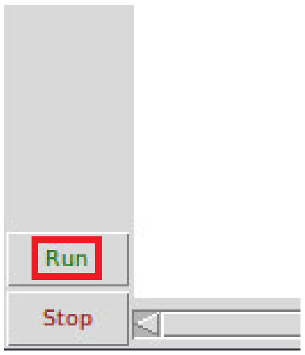


Figure 5. Running the emulation.

The above topology uses 10.0.0.0/8 which is the default network assigned by Mininet.

2.1 Starting host h1 and host h2

Step 1. Hold the right-click on host h1 and select *Terminal*. This opens the terminal of host h1 and allows the execution of commands on host h1.

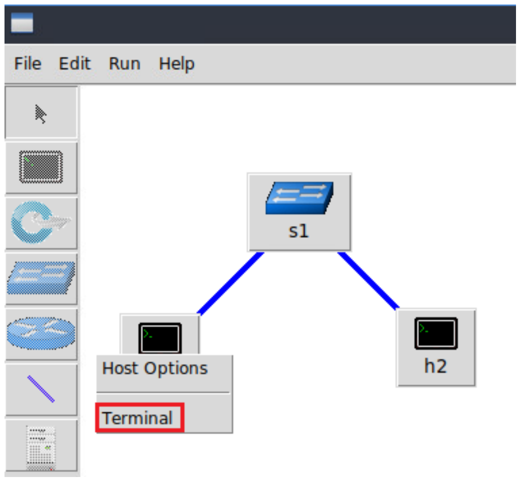
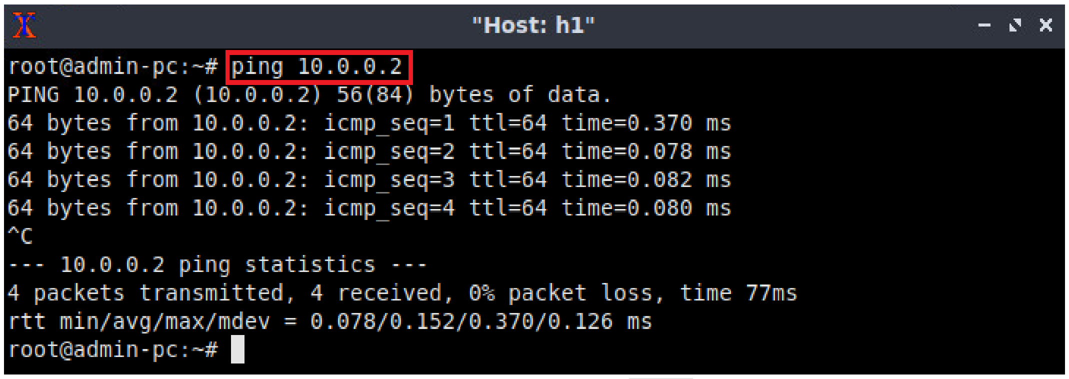


Figure 6. Opening a terminal on host h1.

Step 2. Test connectivity between the end-hosts using the `ping` command. On host h1, type the command `ping 10.0.0.2`. This command tests the connectivity between host h1 and host h2. To stop the test, press `Ctrl+c`. The figure below shows a successful connectivity test.



```
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.370 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.082 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.080 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 77ms
rtt min/avg/max/mdev = 0.078/0.152/0.370/0.126 ms
root@admin-pc:~#
```

Figure 7. Connectivity test using `ping` command.

The figure above indicates that there is connectivity between host h1 and host h2. Thus, we are ready to start the throughput measurement process.

3 Adding/changing delay to emulate a WAN

The user invokes NETEM using the command line utility called `tc` [4, 5]. With no additional parameters, NETEM behaves as a basic FIFO queue with no delay, loss, duplication, or reordering of packets. The basic `tc` syntax used with NETEM is as follows:

```
sudo tc qdisc [add|del|replace|change|show] dev dev_id root netem opts
```

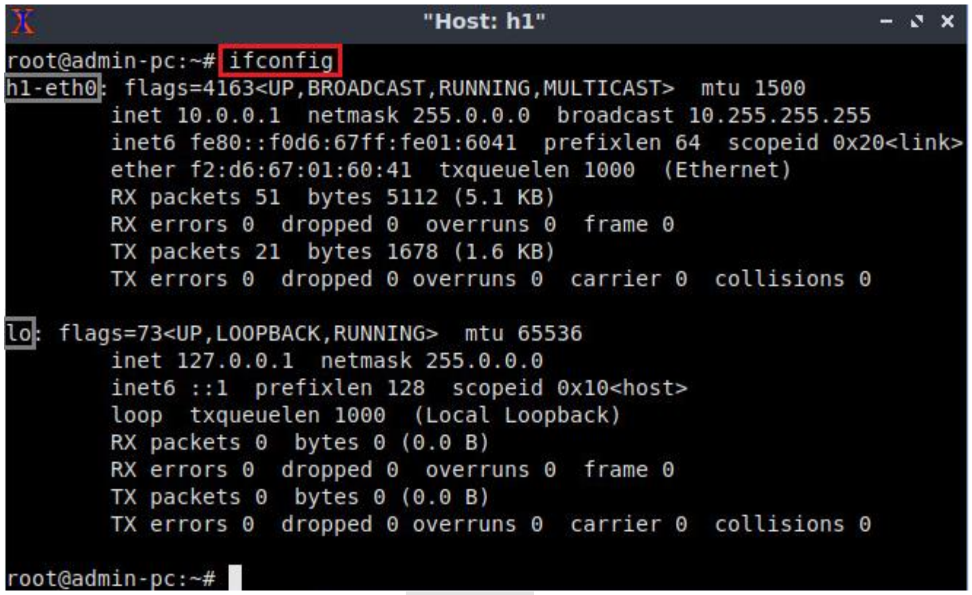
- `sudo`: enable the execution of the command with higher security privileges.
- `tc`: command used to interact with NETEM.

- `qdisc`: a queue discipline (qdisc) is a set of rules that determine the order in which packets arriving from the IP protocol output (see Figure 1) are served. The queue discipline is applied to a packet queue to decide when to send each packet.
- `[add | del | replace | change | show]`: this is the operation on qdisc. For example, to add delay on a specific interface, the operation will be `add`. To change or remove delay on the specific interface, the operation will be `change` or `del`.
- `dev_id`: this parameter indicates the interface to be subject to emulation.
- `opts`: this parameter indicates the amount of delay, packet loss, duplication, corruption, and others.

3.1 Identify interface of host h1 and host h2

According to the previous section, we must identify the interfaces on the connected hosts.

Step 1. On host h1, type the command `ifconfig` to display information related to its network interfaces and their assigned IP addresses.

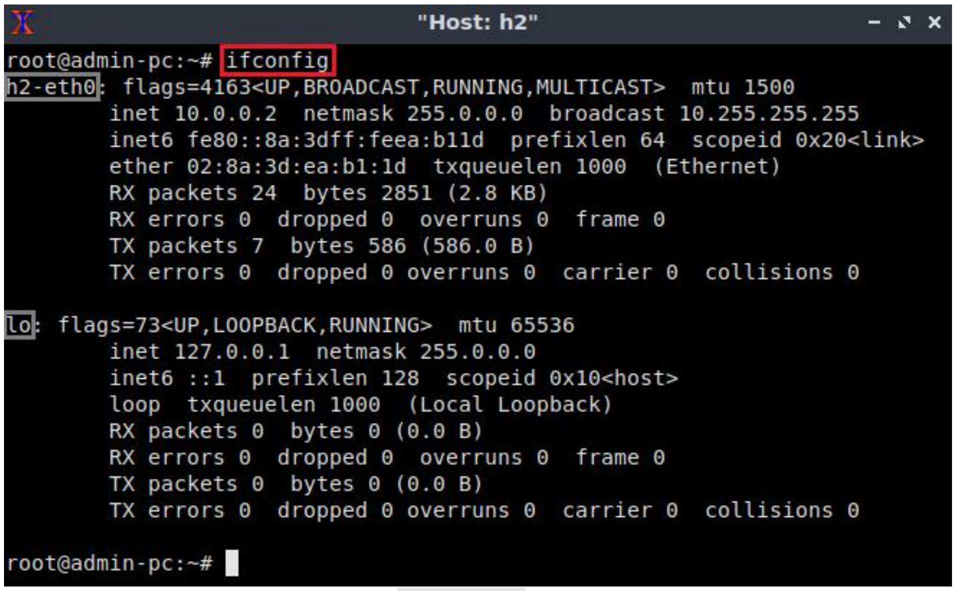


```
root@admin-pc:~# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::f0d6:67ff:fe01:6041  prefixlen 64  scopeid 0x20<link>
        ether f2:d6:67:01:60:41  txqueuelen 1000  (Ethernet)
        RX packets 51  bytes 5112 (5.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 21  bytes 1678 (1.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@admin-pc:~#
```

Figure 8. Output of `ifconfig` command on host h1.

The output of the `ifconfig` command indicates that host h1 has two interfaces: *h1-eth0* and *lo*. The interface *h1-eth0* at host h1 is configured with IP address 10.0.0.1 and subnet mask 255.0.0.0. This interface must be used in `tc` when emulating the WAN.

Step 2. In host h2, type the command `ifconfig` as well.



```
root@admin-pc:~# ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.2  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::8a:3dff:feea:b11d  prefixlen 64  scopeid 0x20<link>
        ether 02:8a:3d:ea:b1:1d  txqueuelen 1000  (Ethernet)
        RX packets 24  bytes 2851 (2.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7  bytes 586 (586.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@admin-pc:~# 
```

Figure 9. Output of `ifconfig` command on host h2.

The output of the `ifconfig` command indicates that host h2 has two interfaces: *h2-eth0* and *lo*. The interface *h2-eth0* at host h1 is configured with IP address 10.0.0.2 and subnet mask 255.0.0.0. This interface must be used in `tc` when emulating the WAN.

3.2 Add delay to interface connecting to WAN

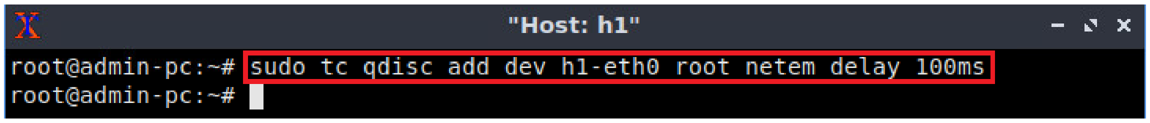
Network emulators emulate delays by introducing them to an interface. For example, the delay introduced to a switch A's interface that is connected to a switch B's interface may represent the propagation delay of a WAN connecting both switches. In this section, you will use `netem` command to insert delay to a network interface.

Step 1. In host h1, type the following command:

```
sudo tc qdisc add dev h1-eth0 root netem delay 100ms
```

This command can be summarized as follows:

- `sudo`: enable the execution of the command with higher security privileges.
- `tc`: invoke Linux's traffic control.
- `qdisc`: modify the queuing discipline of the network scheduler.
- `add`: create a new rule.
- `dev h1-eth0`: specify the interface on which the rule will be applied.
- `netem`: use the network emulator.
- `delay 100ms`: inject delay of 100ms.



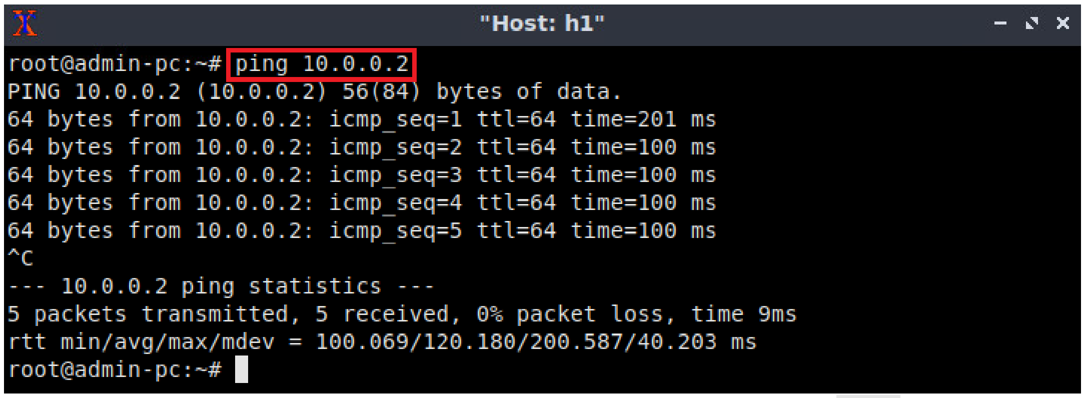
```
root@admin-pc:~# sudo tc qdisc add dev h1-eth0 root netem delay 100ms
root@admin-pc:~# 
```

Figure 10. Adding 100ms delay to the interface *h1-eth0*.

The above command adds a delay of 100 milliseconds (ms) to the output interface, exclusively.

Step 2. The user can verify now that the connection from host h1 to host h2 has a delay of 100 milliseconds by using the `ping` command from host h1:

```
ping 10.0.0.2
```



```
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=201 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=100 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 9ms
rtt min/avg/max/mdev = 100.069/120.180/200.587/40.203 ms
root@admin-pc:~#
```

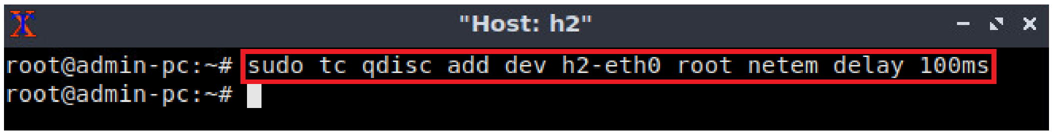
Figure 11. Verifying latency after emulating delay using `ping`.

The result above indicates that all five packets were received successfully (0% packet loss) and that the minimum, average, maximum, and standard deviation of the Round-Trip Time (RTT) were 100.069, 120.180, 200.587, and 40.203 milliseconds respectively.

Note that the above scenario emulates 100 milliseconds latency on the interface of host h1 connecting to the switch. In order to emulate a WAN where the delay is bidirectional, a delay of 100 milliseconds must also be added to the corresponding interface on host h2.

Step 3. In host h2's terminal, type the following command:

```
sudo tc qdisc add dev h2-eth0 root netem delay 100ms
```

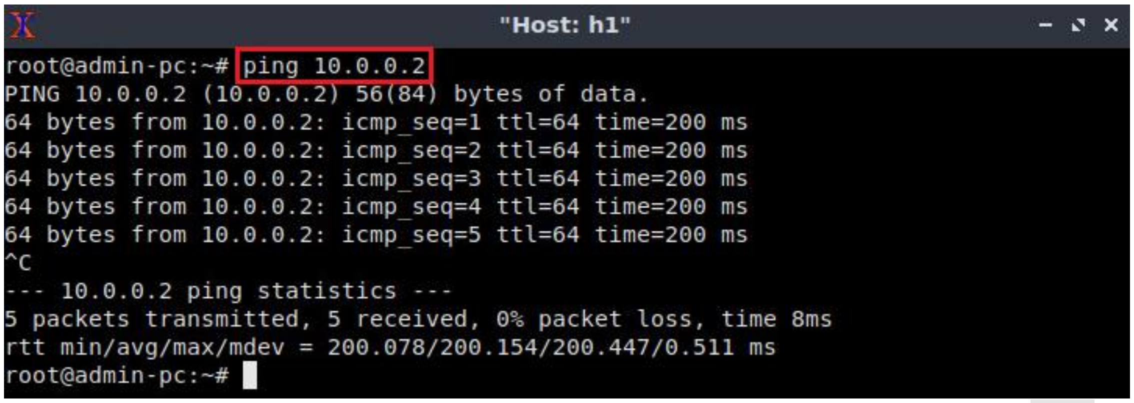


```
root@admin-pc:~# sudo tc qdisc add dev h2-eth0 root netem delay 100ms
root@admin-pc:~#
```

Figure 12. Adding 100ms delay to the interface *h2-eth0*.

Step 4. The user can verify now that the connection between host h1 and host h2 has an RTT of 200 milliseconds (100ms from host h1 to host h2 plus 100ms from host h2 to host h1) by retyping the `ping` command on host h1's terminal:

```
ping 10.0.0.2
```



```
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=200 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=200 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=200 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=200 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=200 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 8ms
rtt min/avg/max/mdev = 200.078/200.154/200.447/0.511 ms
root@admin-pc:~#
```

Figure 13. Verifying latency after emulating delay on both host h1 and host h2 using `ping`.

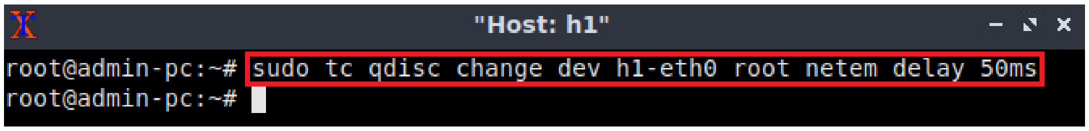
The result above indicates that all five packets were received successfully (0% packet loss) and that the minimum, average, maximum, and standard deviation of the Round-Trip Time (RTT) were 200.078, 200.154, 204.447, and 0.511 milliseconds respectively.

3.3 Changing the delay in emulated WAN

In this section, the user will change the delay from 100 milliseconds to 50 milliseconds in both sender and receiver. The RTT will be 100 milliseconds now.

Step 1. In host h1's terminal, type the following command:

```
sudo tc qdisc change dev h1-eth0 root netem delay 50ms
```



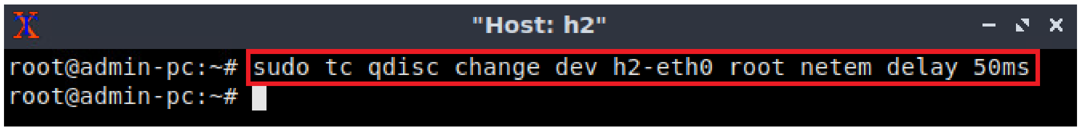
```
root@admin-pc:~# sudo tc qdisc change dev h1-eth0 root netem delay 50ms
root@admin-pc:~#
```

Figure 14. Changing delay on the interface *h1-eth0*.

The new option added here is `change`, which changes the previously set delay to 50 milliseconds.

Step 2. Apply also the above step on host h2's terminal to change the delay to 50ms:

```
sudo tc qdisc change dev h2-eth0 root netem delay 50ms
```

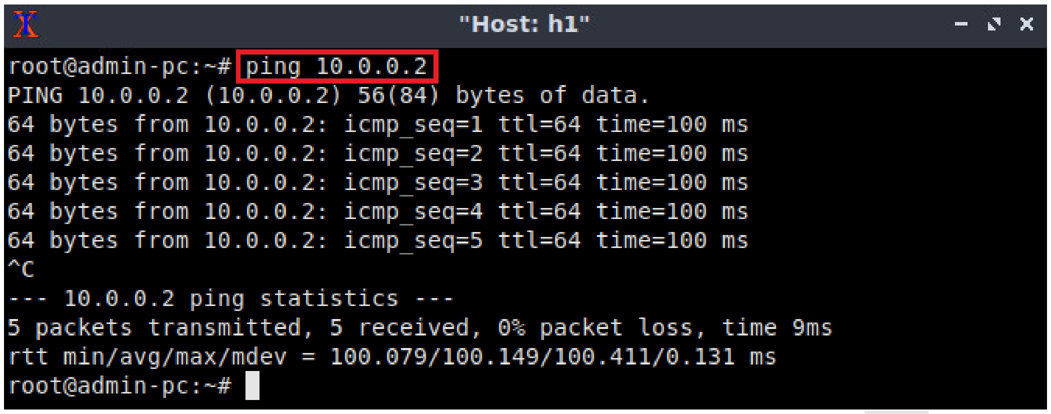


```
root@admin-pc:~# sudo tc qdisc change dev h2-eth0 root netem delay 50ms
root@admin-pc:~#
```

Figure 15. Changing delay to the interface *h2-eth0*.

Step 3. The user can verify now that the connection from host h1 to host h2 has a delay of 100 milliseconds by using the `ping` command from host h1's terminal:

```
ping 10.0.0.2
```



```
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=100 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 9ms
rtt min/avg/max/mdev = 100.079/100.149/100.411/0.131 ms
root@admin-pc:~#
```

Figure 16. Verifying latency after emulating 100ms delay using `ping`.

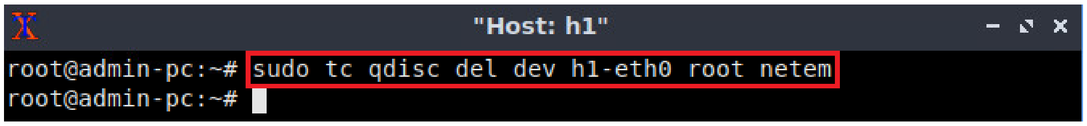
The result above indicates that all five packets were received successfully (0% packet loss) and that the minimum, average, maximum, and standard deviation of the Round-Trip Time (RTT) were 100.079, 100.149, 100.411, and 0.131 milliseconds respectively.

4 Restoring original values (deleting the rules)

In this section, the user will restore the default configuration in both sender and receiver by deleting all the rules applied to the network scheduler of an interface.

Step 1. In host h1's terminal, type the following command:

```
sudo tc qdisc del dev h1-eth0 root netem
```



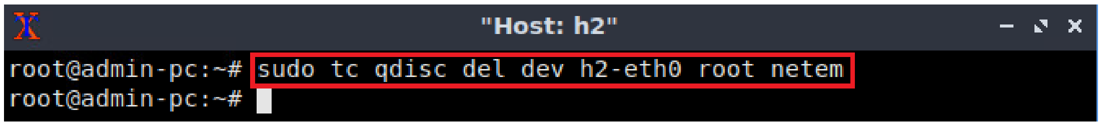
```
root@admin-pc:~# sudo tc qdisc del dev h1-eth0 root netem
root@admin-pc:~#
```

Figure 17. Deleting all rules on interface *h1-eth0*.

The new option added here is `del`, which deletes the previously set rules on a given interface. As a result, the `tc` qdisc will restore its default values of the device *h1-eth0*.

Step 2. Apply the same steps to remove rules on host h2. In host h2's terminal, type the following command:

```
sudo tc qdisc del dev h2-eth0 root netem
```



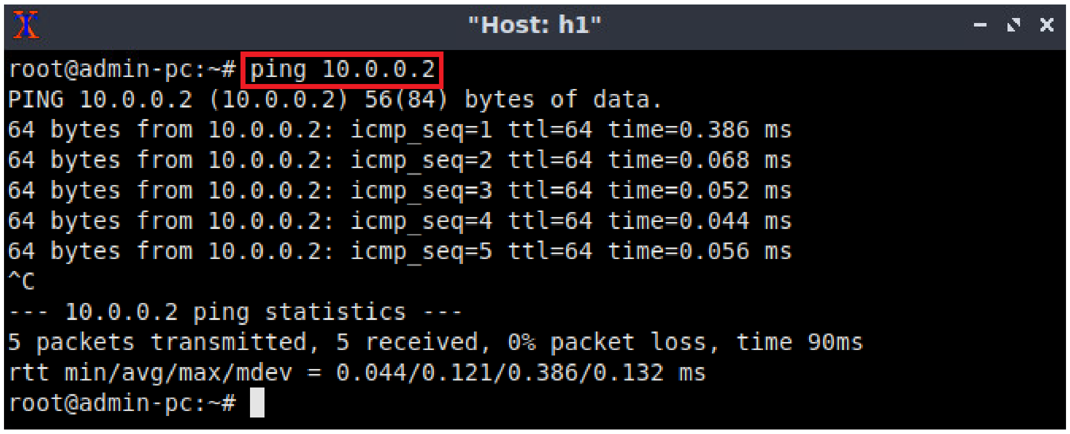
```
root@admin-pc:~# sudo tc qdisc del dev h2-eth0 root netem
root@admin-pc:~#
```

Figure 18. Deleting all rules on interface *h2-eth0*.

As a result, the `tc` queueing discipline will restore its default values of the device *h2-eth0*.

Step 3. The user can now verify that the connection from host h1 to host h2 has no explicit delay set by using the `ping` command from host h1's terminal:

```
ping 10.0.0.2
```



```
"Host: h1"
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.386 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.068 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.044 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.056 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 90ms
rtt min/avg/max/mdev = 0.044/0.121/0.386/0.132 ms
root@admin-pc:~#
```

Figure 19. Verifying latency after deleting all rules on both devices.

The result above indicates that all five packets were received successfully (0% packet loss) and that the minimum, average, maximum, and standard deviation of the Round-Trip Time (RTT) were 0.044, 0.121, 0.386, and 0.132 milliseconds respectively.

5 Adding jitter to emulated WAN

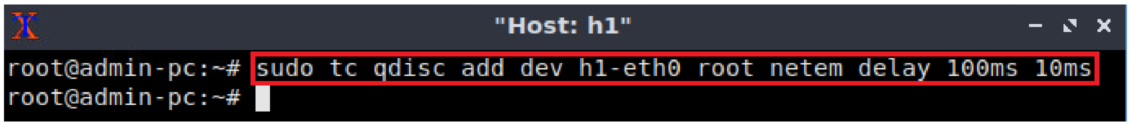
Networks do not exhibit constant delay; the delay may vary based on other traffic flows contending for the same path. Jitter is the variation of delay time. The delay parameters are described by the average value (μ), standard deviation (σ), and correlation. By default, NETEM uses a uniform distribution, so that the delay is within μ ± σ.

5.1 Add jitter to interface connecting to WAN

In this section, the user will add delay of 100 milliseconds with a random variation of ± 10 milliseconds. Before doing so, make sure to restore the default configuration of the interfaces on host h1 and host h2 by applying the commands of Section 4. Then, apply the commands below.

Step 1. In host h1's terminal, type the following command:

```
sudo tc qdisc add dev h1-eth0 root netem delay 100ms 10ms
```



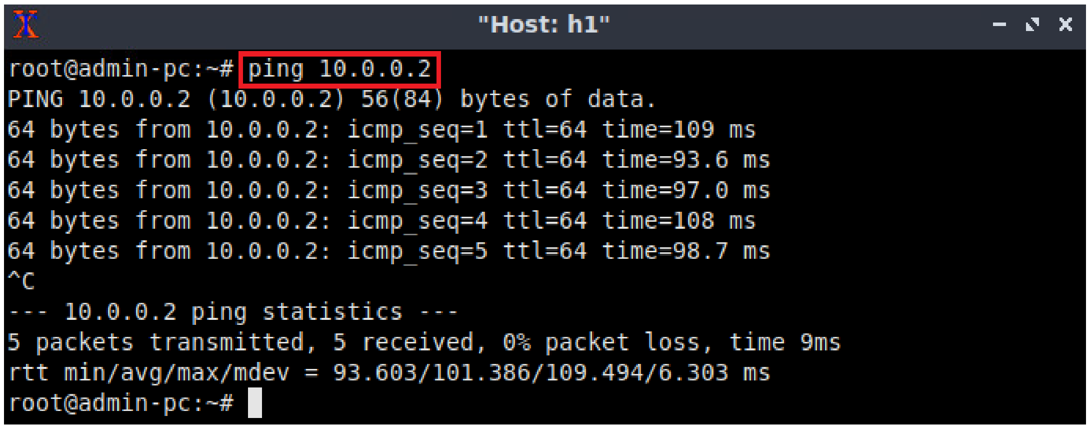
```
"Host: h1"
root@admin-pc:~# sudo tc qdisc add dev h1-eth0 root netem delay 100ms 10ms
root@admin-pc:~#
```

Figure 20. Add 100ms delay with ± 10 millisecond.

The new value added here represents jitter which defines the delay variation. Therefore, all packets leaving host h1 via interface *h1-eth0* will experience a delay of 100ms, with a random variation of ± 10ms.

Step 2. The user can now verify that the connection from host h1 to host h2 has 100ms delay with ± 10 millisecond random variation by using the `ping` command on host h1's terminal:

```
ping 10.0.0.2
```



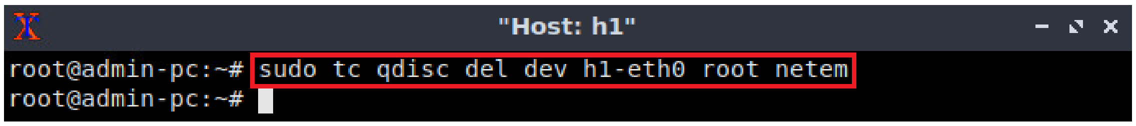
```
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=109 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=93.6 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=97.0 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=108 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=98.7 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 9ms
rtt min/avg/max/mdev = 93.603/101.386/109.494/6.303 ms
root@admin-pc:~#
```

Figure 21. Verifying RTT after adding 100 millisecond delay and 10 millisecond jitter on interface *h1-eth0*.

The result above indicates that all five packets were received successfully (0% packet loss) and that the minimum, average, maximum, and standard deviation of the Round-Trip Time (RTT) were 93.603, 101.386, 109.494, and 6.303 milliseconds respectively. Note that we are only adding jitter to the interface of host h1 at this point.

Step 3. In host h1's terminal, type the following command to delete previous configurations:

```
sudo tc qdisc del dev h1-eth0 root netem
```



```
root@admin-pc:~# sudo tc qdisc del dev h1-eth0 root netem
root@admin-pc:~#
```

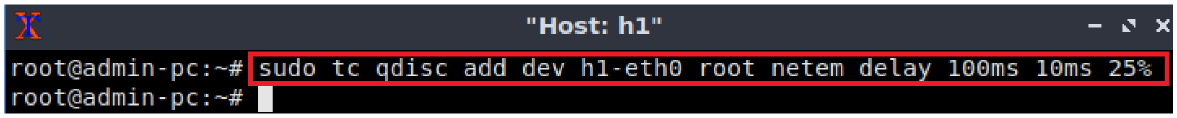
Figure 22. Deleting all rules on interface *h1-eth0*.

6 Adding correlation value for jitter and delay

The correlation parameter controls the relationship between successive pseudo-random values. In this section, the user will add a delay of 100 milliseconds with a variation of ± 10 milliseconds while adding a correlation value. Before doing so, make sure to restore the default configuration of the interfaces on host h1 and host h2 by applying the commands of Section 4. Then, apply the commands below.

Step 1. In host h1 terminal, type the following command:

```
sudo tc qdisc add dev h1-eth0 root netem delay 100ms 10ms 25%
```



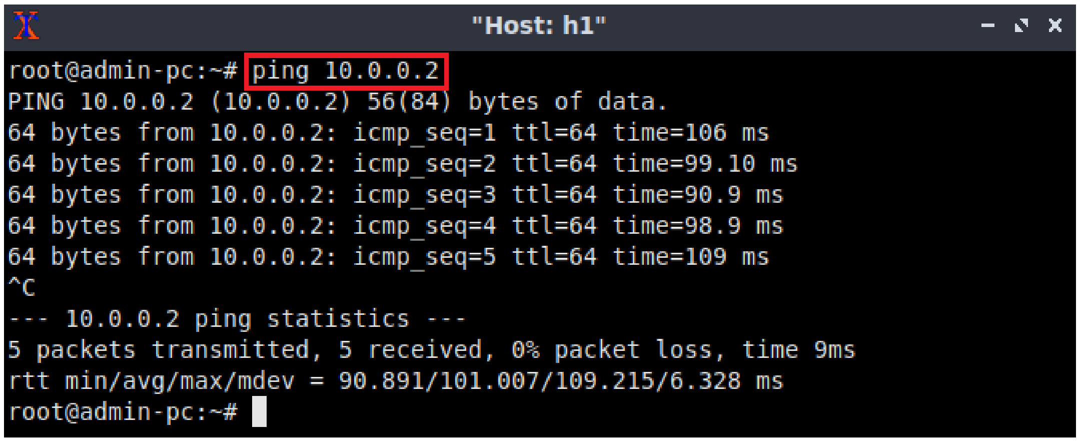
```
root@admin-pc:~# sudo tc qdisc add dev h1-eth0 root netem delay 100ms 10ms 25%
root@admin-pc:~# 
```

Figure 23. Adding a correlation value of 25%.

The new value added here represents the correlation value for jitter and delay. Therefore, all packets leaving the device host h1 on the interface *h1-eth0* will experience a 100ms delay time, with a random variation of ± 10 millisecond with the next random packet depending 25% on the previous one.

Step 2. Now, the user can test the connection from host h1 to host h2 by using the `ping` command on host h1's terminal:

```
ping 10.0.0.2
```



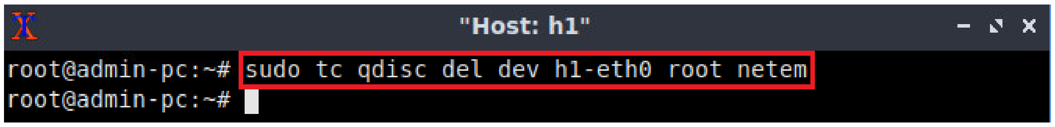
```
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=106 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=99.10 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=90.9 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=98.9 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=109 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 9ms
rtt min/avg/max/mdev = 90.891/101.007/109.215/6.328 ms
root@admin-pc:~# 
```

Figure 24. Verifying latency after setting the correlation value.

The result above indicates that all five packets were received successfully (0% packet loss) and that the minimum, average, maximum, and standard deviation of the Round-Trip Time (RTT) were 90.891, 101.007, 109.215, and 6.328 milliseconds respectively.

Step 3. In host h1's terminal, type the following command to delete previous configurations:

```
sudo tc qdisc del dev h1-eth0 root netem
```



```
root@admin-pc:~# sudo tc qdisc del dev h1-eth0 root netem
root@admin-pc:~# 
```

Figure 25. Deleting all rules on interface *h1-eth0*.

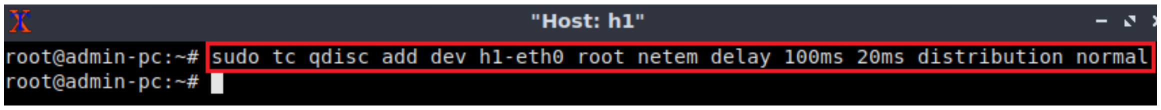
7 Delay distribution

NETEM permits user to specify a distribution that describes how delays vary in the network. Usually delays are not uniform, so it may be convenient to use a non-uniform distribution such as normal, pareto, or pareto-normal. For this test, the user will specify a normal distribution for the delay in the emulated network. Before doing so, make sure to restore the default configuration of the interfaces on host h1 and host h2 by applying the commands of Section 4. Then, apply the commands below.

Step 1. In host h1's terminal, type the following command:

```
sudo tc qdisc add dev h1-eth0 root netem delay 100ms 20ms distribution normal
```

The new option added here (`distribution`) represents the delay distribution type. We define the delay to have a normal distribution, which provides a more realistic emulation of WAN networks. As a result, all packets leaving the host h1 on the interface *h1-eth0* will experience delay time which is normally distributed between the range of 100ms ± 20ms.

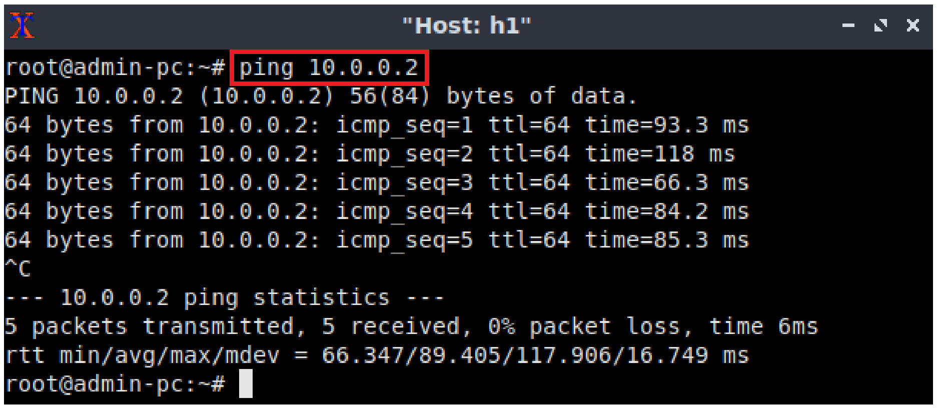


```
root@admin-pc:~# sudo tc qdisc add dev h1-eth0 root netem delay 100ms 20ms distribution normal
root@admin-pc:~#
```

Figure 26. Adding normal distribution of delay.

Step 2. The user can now verify if the configuration was successfully done in the previous step (Step 1) by using the `ping` command on host h1's terminal:

```
ping 10.0.0.2
```



```
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=93.3 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=118 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=66.3 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=84.2 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=85.3 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 6ms
rtt min/avg/max/mdev = 66.347/89.405/117.906/16.749 ms
root@admin-pc:~#
```

Figure 27. Verifying latency after using normal distribution.

The result above indicates that all five packets were received successfully (0% packet loss) and that the minimum, average, maximum, and standard deviation of the Round-Trip Time (RTT) were 66.347, 89.405, 117.906, and 16.749 milliseconds respectively.

This concludes Lab 3. Stop the emulation and then exit out of MiniEdit.

References

1. Linux foundation. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem.
2. S. Hemminger, “Network emulation with NETEM,” Linux conf au. 2005, pp. 18-23. 2005.
3. J. Kurose, K. Ross, “Computer networking, a top-down approach,” 7th Edition, Pearson, 2017.
4. How to use the linux traffic control panagiotis vouzis [Online]. Available: https://netbeez.net/blog/how-to-use-the-linux-traffic-control.
5. M. Brown, F. Bolelli, N. Patriciello, “Traffic control howto,” Guide to IP Layer Network, 2006.