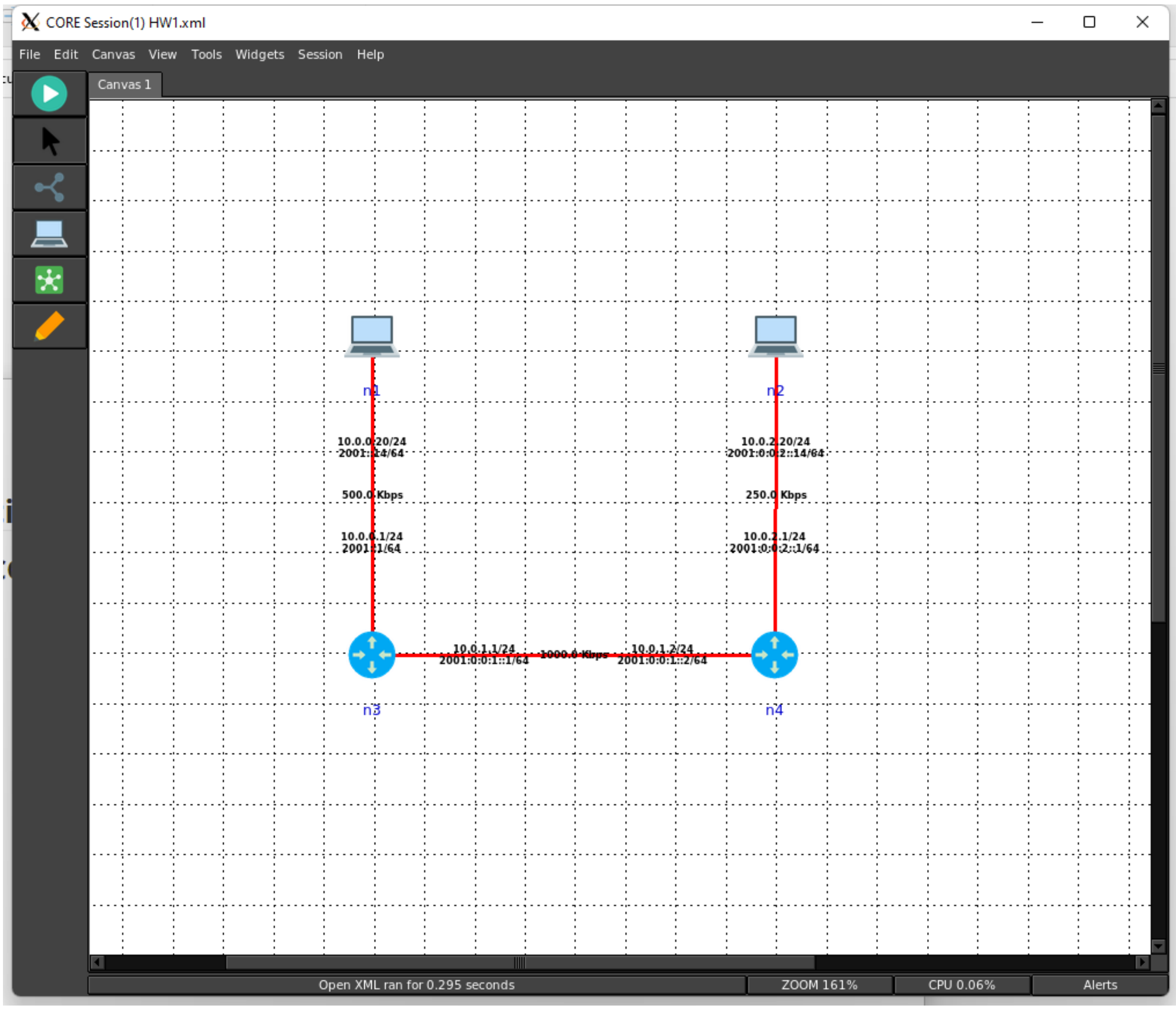
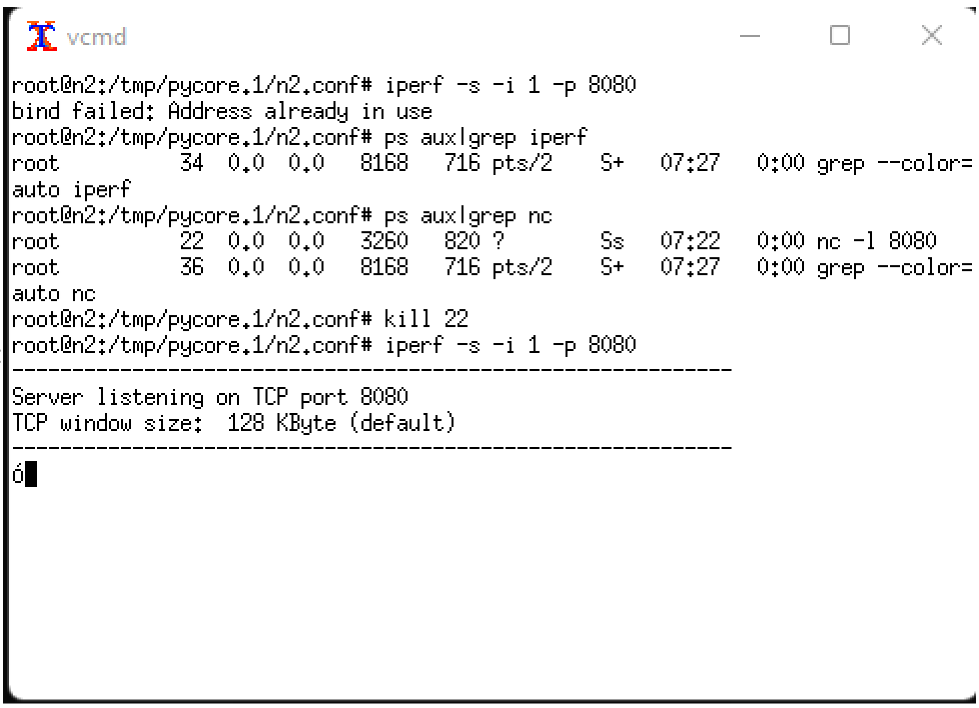
Part 1: Setting up a TCP client and server (2 points)

1) Run the CORE scenario



2) Open a terminal on node n2. Run a server on port 8080:

- a:



```
vcmd
root@n2:/tmp/pycore.1/n2.conf# iperf -s -i 1 -p 8080
bind failed: Address already in use
root@n2:/tmp/pycore.1/n2.conf# ps aux|grep iperf
root          34  0.0  0.0   8168   716 pts/2    S+   07:27   0:00 grep --color=
auto iperf
root@n2:/tmp/pycore.1/n2.conf# ps aux|grep nc
root          22  0.0  0.0   3260   820 ?        Ss   07:22   0:00 nc -l 8080
root          36  0.0  0.0   8168   716 pts/2    S+   07:27   0:00 grep --color=
auto nc
root@n2:/tmp/pycore.1/n2.conf# kill 22
root@n2:/tmp/pycore.1/n2.conf# iperf -s -i 1 -p 8080
------------------------------------------------------------
Server listening on TCP port 8080
TCP window size:  128 KByte (default)
------------------------------------------------------------
ó
```

- `-s` means run in server mode
- `-i 1` means 1 second between periodic bandwidth reports
- `-p 8080` means server listens on port 8080

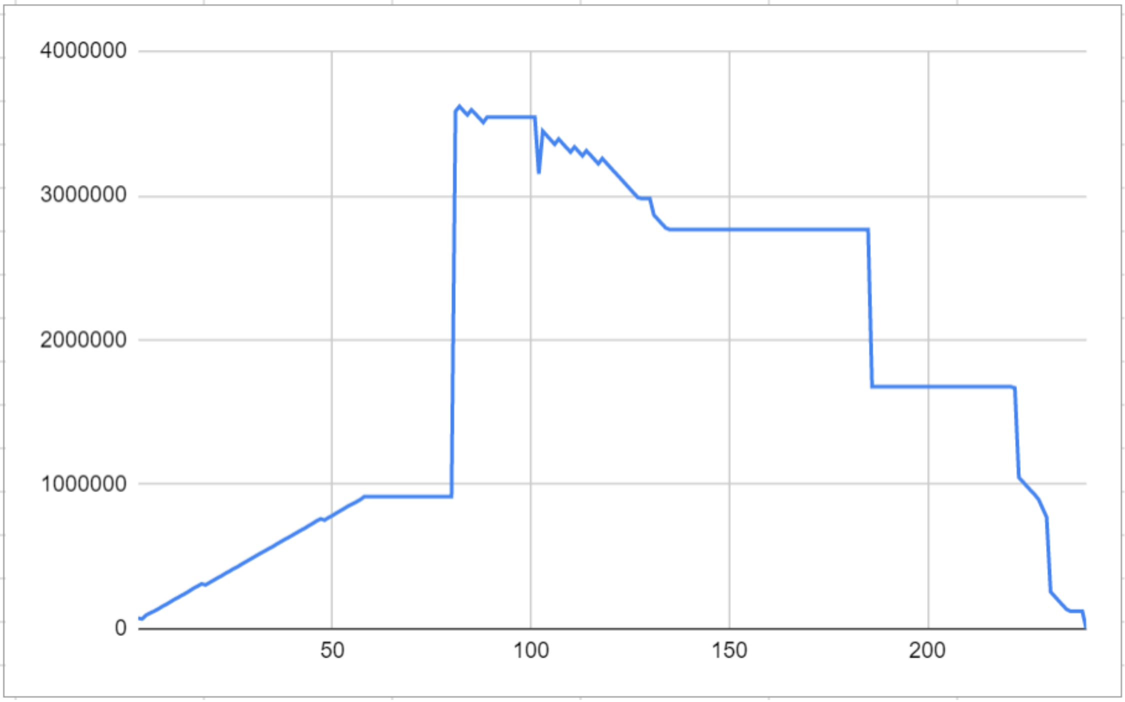
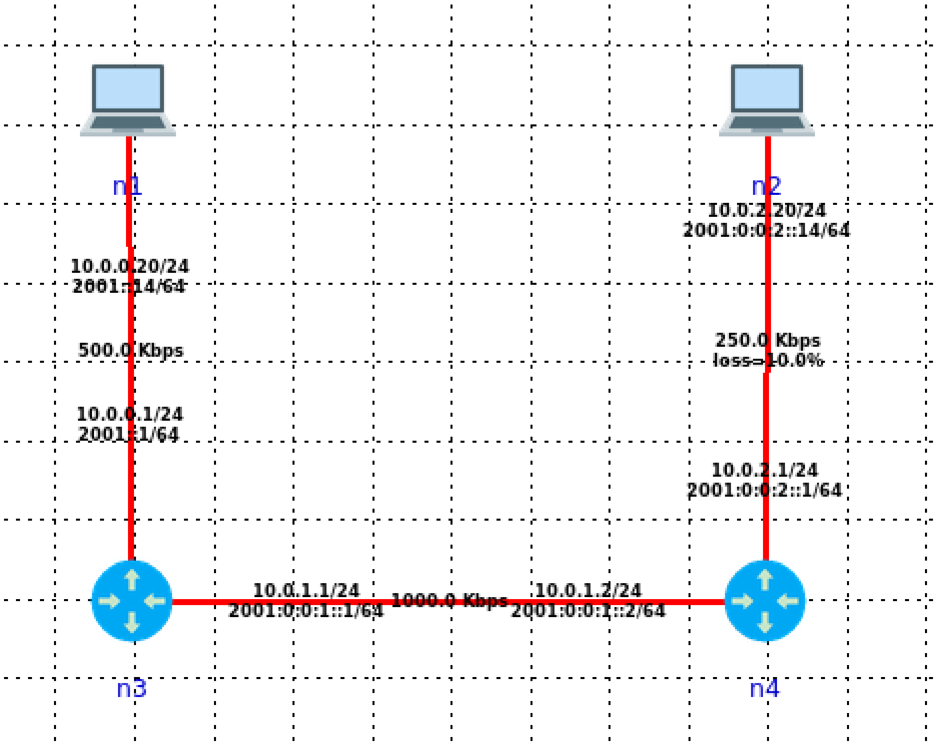
3) Open another terminal on node n2. Use the command netstat to list servers

- ```
  root@n2:/tmp/pycore.1/n2.conf# netstat -anp
  Active Internet connections (servers and established)
  Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
  tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      37/iperf
  Active UNIX domain sockets (servers and established)
  Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
  root@n2:/tmp/pycore.1/n2.conf# 
  ```
- Proto/Recv-Q/Send-Q/Local Address/Foreign Address/State/PID

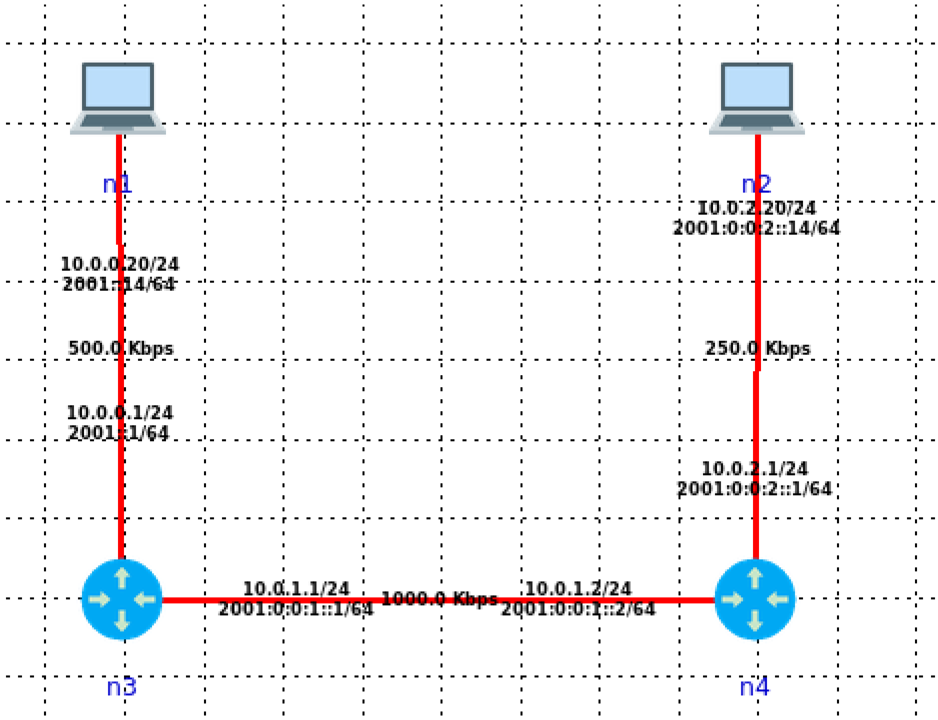
4) Now run a client on node n1 connecting to node n2

- ```
  root@n1:/tmp/pycore.1/n1.conf# iperf -c 10.0.2.20 -i 1 -t 120 -p 8080
  ------------------------------------------------------------
  Client connecting to 10.0.2.20, TCP port 8080
  TCP window size: 85.0 KByte (default)
  ------------------------------------------------------------
  [  3] local 10.0.0.20 port 55758 connected with 10.0.2.20 port 8080
  [ ID] Interval       Transfer     Bandwidth
  [  3]  0.0- 1.0 sec   107 KBytes   880 Kbits/sec
  [  3]  1.0- 2.0 sec  14.1 KBytes   116 Kbits/sec
  [  3]  2.0- 3.0 sec  36.8 KBytes   301 Kbits/sec
  [  3]  3.0- 4.0 sec  55.1 KBytes   452 Kbits/sec
  [  3]  4.0- 5.0 sec  36.8 KBytes   301 Kbits/sec
  [  3]  5.0- 6.0 sec  42.4 KBytes   348 Kbits/sec
  [  3]  6.0- 7.0 sec  43.8 KBytes   359 Kbits/sec
  [  3]  7.0- 8.0 sec  46.7 KBytes   382 Kbits/sec
  [  3]  8.0- 9.0 sec  42.4 KBytes   348 Kbits/sec
  [  3]  9.0-10.0 sec  46.7 KBytes   382 Kbits/sec
  [  3] 10.0-11.0 sec  46.7 KBytes   382 Kbits/sec
  [  3] 11.0-12.0 sec  45.2 KBytes   371 Kbits/sec
  [  3] 12.0-13.0 sec  43.8 KBytes   359 Kbits/sec
  [  3] 13.0-14.0 sec  42.4 KBytes   348 Kbits/sec
  ```
- `-c` means run in client mode and connect to 10.0.2.20
- `-t 120` means set transmit time to 120 senconds

5) Open another terminal on node n1.

- ```
  vcmd
  Active UNIX domain sockets (servers and established)
  Proto RefCnt Flags       Type       State         I-Node   PID/Program name
  Path
  Active Internet connections (servers and established)
  Proto Recv-Q Send-Q Local Address           Foreign Address         State
  PID/Program name
  tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN
  23/iperf
  tcp        0 3191393 10.0.0.20:55766         10.0.2.20:8080          FIN_WAIT1
   -
  Active UNIX domain sockets (servers and established)
  Proto RefCnt Flags       Type       State         I-Node   PID/Program name
  Path
  Active Internet connections (servers and established)
  Proto Recv-Q Send-Q Local Address           Foreign Address         State
  PID/Program name
  tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN
  23/iperf
  tcp        0 3191393 10.0.0.20:55766         10.0.2.20:8080          FIN_WAIT1
   -
  Active UNIX domain sockets (servers and established)
  Proto RefCnt Flags       Type       State         I-Node   PID/Program name
  Path
  ```
  ```
  vcmd
  Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
  tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      37/iperf
  tcp        0      0 10.0.2.20:8080          10.0.0.20:55766         ESTABLISHED 37/iperf
  Active UNIX domain sockets (servers and established)
  Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
  Active Internet connections (servers and established)
  Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
  tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      37/iperf
  tcp        0      0 10.0.2.20:8080          10.0.0.20:55766         ESTABLISHED 37/iperf
  Active UNIX domain sockets (servers and established)
  Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
  Active Internet connections (servers and established)
  Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
  tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      37/iperf
  tcp        0      0 10.0.2.20:8080          10.0.0.20:55766         ESTABLISHED 37/iperf
  Active UNIX domain sockets (servers and established)
  Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
  Active Internet connections (servers and established)
  Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
  tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      37/iperf
  tcp        0      0 10.0.2.20:8080          10.0.0.20:55766         ESTABLISHED 37/iperf
  Active UNIX domain sockets (servers and established)
  Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
  ```
- The `recv-Q` and `send-Q on node 2 is always 0, the recv-Q on node1 is always 0, the send-Q increase first and drops in the last(takes 240 seconds to clear the send-Q, which is exactly 2 times of 120 seconds)
  ```

- 

- The Send-Q is the Queue for packet sending, the Recv-Q is the Queue for packet receiving.

Part 2: Benchmarking (4 points)

1) Test 1, throughput:

1. Run the CORE scenario
2. How does iperf measure throughput?
    1. iperf should be counting packet during certain time period to calculate the average throughput
3. Which link in this scenario is the bottleneck link?
    1. The n4->n2 link should be the bottleneck link for
4. Run iperf server on node n2 as shown in part 1
5. Run iperf client on node n1 as shown on part 1
6. The server will report instantaneous throughput per second
7. When the server stops reporting, the last line is the average throughput. What is the average throughput?
    1.
    ```
    [  4] 234.0-235.0 sec  28.3 KBytes   232 Kbits/sec
    [  4] 235.0-236.0 sec  29.7 KBytes   243 Kbits/sec
    [  4] 236.0-237.0 sec  29.7 KBytes   243 Kbits/sec
    [  4] 237.0-238.0 sec  28.3 KBytes   232 Kbits/sec
    [  4] 238.0-239.0 sec  29.7 KBytes   243 Kbits/sec
    [  4] 239.0-240.0 sec  28.3 KBytes   232 Kbits/sec
    [  4] 240.0-241.0 sec  29.7 KBytes   243 Kbits/sec
    [  4]  0.0-241.5 sec  6.49 MBytes   226 Kbits/sec
    ```
    2. 226 Kbits/sec
8. In your opinion, why is the average throughput smaller than the bottleneck bandwidth?
    1. The bottleneck is just the upper limit of average throughput, there might be packet loss/retry/wait time included to result this smaller average value.
9. Stop the scenario
10. Change rate of link n2-n4 to 750kbps? Repeat starting from (a) to (i)

1.
```
[  4] 177.0-178.0 sec  58.0 KBytes   475 Kbits/sec
[  4] 178.0-179.0 sec  59.4 KBytes   487 Kbits/sec
[  4] 179.0-180.0 sec  58.0 KBytes   475 Kbits/sec
[  4] 180.0-181.0 sec  58.0 KBytes   475 Kbits/sec
[  4]  0.0-181.4 sec  9.43 MBytes   436 Kbits/sec
]
```
2. The Average increased to 436Kbits/s, Now the bottleNeck Link changed to n1->n3

2) Test2,loss

- Reload the scenario (to reset rate of link n2-n4)
- Add a 10% loss on the link n2-n4
    - 

- Run the CORE scenario
- Run a ping test from n1- n4 for 1 minute (till you see icmp_seq=60)
    i. Ping 10.0.2.20
    ii. Control-c to stop
- When you stop ping, it will report statistics. What is the packet loss reported by ping? Can you justify the results (are the results 10% or more? If more, why?)
    - ```
      64 bytes from 10.0.2.20: icmp_seq=55 ttl=62 time=11.4 ms
      64 bytes from 10.0.2.20: icmp_seq=57 ttl=62 time=11.4 ms
      64 bytes from 10.0.2.20: icmp_seq=59 ttl=62 time=11.3 ms
      64 bytes from 10.0.2.20: icmp_seq=60 ttl=62 time=11.2 ms
      64 bytes from 10.0.2.20: icmp_seq=61 ttl=62 time=11.2 ms
      64 bytes from 10.0.2.20: icmp_seq=63 ttl=62 time=11.3 ms
      64 bytes from 10.0.2.20: icmp_seq=64 ttl=62 time=11.3 ms
      64 bytes from 10.0.2.20: icmp_seq=65 ttl=62 time=11.1 ms
      64 bytes from 10.0.2.20: icmp_seq=66 ttl=62 time=11.3 ms
      ^C
      --- 10.0.2.20 ping statistics ---
      66 packets transmitted, 51 received, 22.7273% packet loss, time 65377ms
      rtt min/avg/max/mdev = 11.143/11.321/11.437/0.065 ms
      ```
    - The loss is about 22.72% and is a lot higher than 10%, the main reason behind this is the loss is applied both to inbound traffic and outbound traffic, resulting a double loss rate in round trip.
- Stop the scenario
      ```

3) Test3, end to end delay

- a. Read on RTT: https://developer.mozilla.org/en-US/docs/Glossary/Round_Trip_Time_(RTT)
- b. Reload the scenario (to reset loss on link n2-n4)
- c. Add a 100ms delay on the link n2-n4
    - 

- d. Run the CORE scenario
- e. Run a ping test from n1-n4 for 1 minute (till you see icmp_seq=60)
    - ```
      64 bytes from 10.0.2.20: icmp_seq=49 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=50 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=51 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=52 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=53 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=54 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=55 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=56 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=57 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=58 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=59 ttl=62 time=211 ms
      64 bytes from 10.0.2.20: icmp_seq=60 ttl=62 time=211 ms
      ^C
      --- 10.0.2.20 ping statistics ---
      60 packets transmitted, 60 received, 0% packet loss, time 59083ms
      rtt min/avg/max/mdev = 211.250/211.396/211.534/0.044 ms
      root@n1:/tmp/pycore.1/n1.conf#
      ```
- f. When you stop ping, it will report statistics. What is the RTT (round trip time) reported by ping? Can you justify the results (is delay 100ms or more? If more, why?)
    - The ping's RTT is around 211 ms, is a lot more than 100ms. Because the ping reports RTT which consist outbound time and inbound time. So this 100ms latency is applied twice on the result RTT.
- g. Stop the scenario
      ```

4) Test4, per link latency

- a. Rerun last scenario (keep 100ms delay on the link n2-n4)
- b. Run a traceroute from n1-n4:
    - traceroute -n 10.0.2.20
    - ```
      root@n1:/tmp/pycore.1/n1.conf# traceroute -n 10.0.2.20
      traceroute to 10.0.2.20 (10.0.2.20), 30 hops max, 60 byte packets
       1  10.0.0.1  2.920 ms  4.516 ms  6.179 ms
       2  10.0.1.2  7.827 ms  9.475 ms  11.096 ms
       3  10.0.2.20  217.140 ms  220.302 ms  223.646 ms
      ```
- c. Compare the latency results from ping to traceroute. What information does each provide?
    - Ping mainly provide packet loss and overall RTT
    - Traceroute provides RTT for each node in it's path.
    - Describe a situation when you would want to use ping to measure end to end latency
        - When I want to check if there is packet loss in the link or I'm only interested in overall RTT
    - Describe a situation when you would want to use traceroute to measure per link latency
        - When I want to check latency for each link to my destination
      ```