Ping

Q: Ping from the lab machine to the Raspberry Pi, 10 times, interval 0.2 seconds. The average time is around 0.46ms

```
10 packets transmitted, 10 received, 0% packet loss, time 1835ms
rtt min/avg/max/mdev = 0.367/0.460/0.486/0.031 ms
```

Figure 1: Machine to Pi

Q: Ping from the Raspberry Pi to the lab machine, 10 times, interval 0.2 seconds. The average time is around 0.56ms. We see that the time for the Rasberry Pi to send data to the lab machine takes more time as the Rasberry Pi is underperforming lab machine for sending data.

```
10 packets transmitted, 10 received, 0% packet loss, time 1852ms
rtt min/avg/max/mdev = 0.407/0.567/0.617/0.055 ms
```

Figure 2: Pi To Machine

Q: Ping from the Raspberry Pi to the lab machine, 100 times, interval 0.001 seconds (use sudo),and discuss the differences between minimum, mean and maximum results. The average time is 0.44ms. We see that the time has decreased. This is expected as with such a small interval, we are basically flooding the lab machine, and more CPU power is devoted to this task rather than computing other programs in the background. The min is the case where of minimal cpu loading and minimal program queuing, compared to the maximal case where other programs occupy considerable portion of the cpu such that is doesn't have time to respond the task as fast. The mean is the expected time of responding with the cpu loading probability distribution.

```
packets transmitted, 100 received, 0% packet loss, time 10313ms
min/avg/max/mdev = 0.343/0.446/0.535/0.040 ms
```

Figure 3: Near Flooding

Q: Ping from the Raspberry Pi to the lab machine, 10000 times using flooding (use sudo).

We see that the average is 0.435, slightly faster than the near flooding for the same reason near flooding was faster than 0.02s time interval.

```
10000 packets transmitted, 10000 received, 0% packet loss, time
4654ms
rtt min/avg/max/mdev = 0.293/0.435/0.612/0.043 ms, ipg/ewma 0.465/
0.458 ms
```

Figure 4: Flooding

Q: Ping from the Raspberry Pi to the lab machine. Run measurements with 3 different intervals (0.01, 0.001, 0.0001) and at least 1000 measurements, and draw a cdf of your measurements results (one graph per interval)

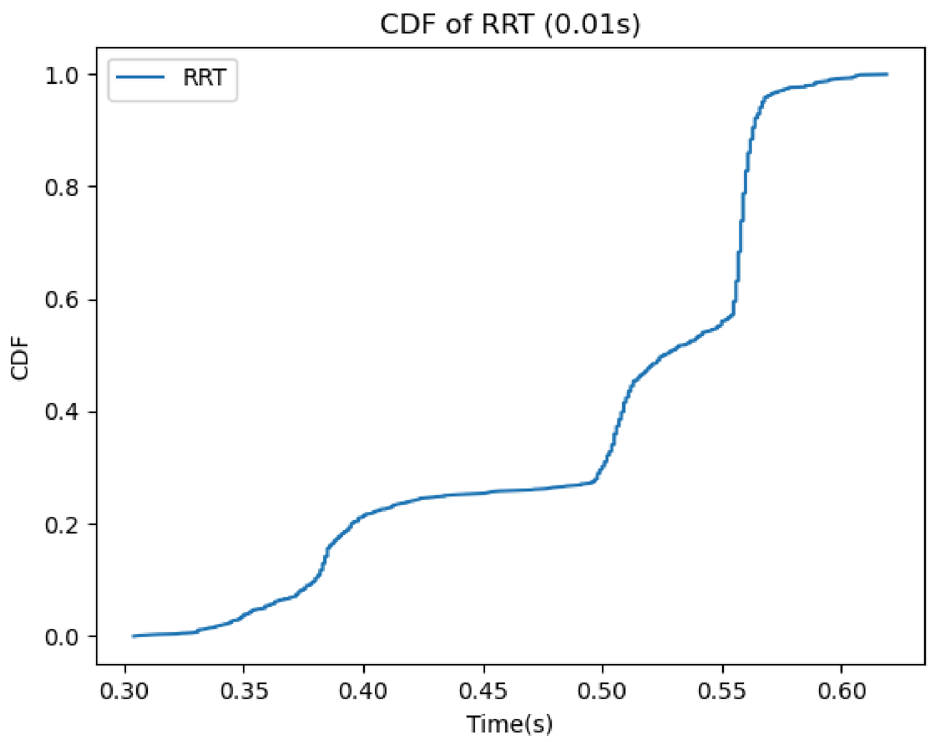

Figure 5: 0.01 RRT CDF

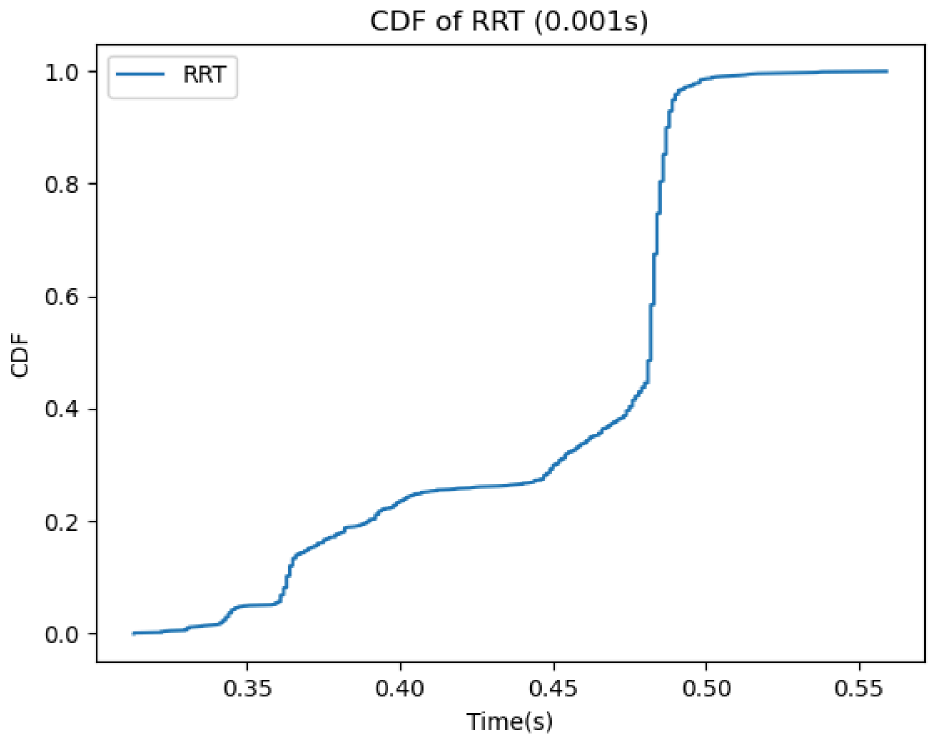

Figure 6: 0.001 RRT CDF

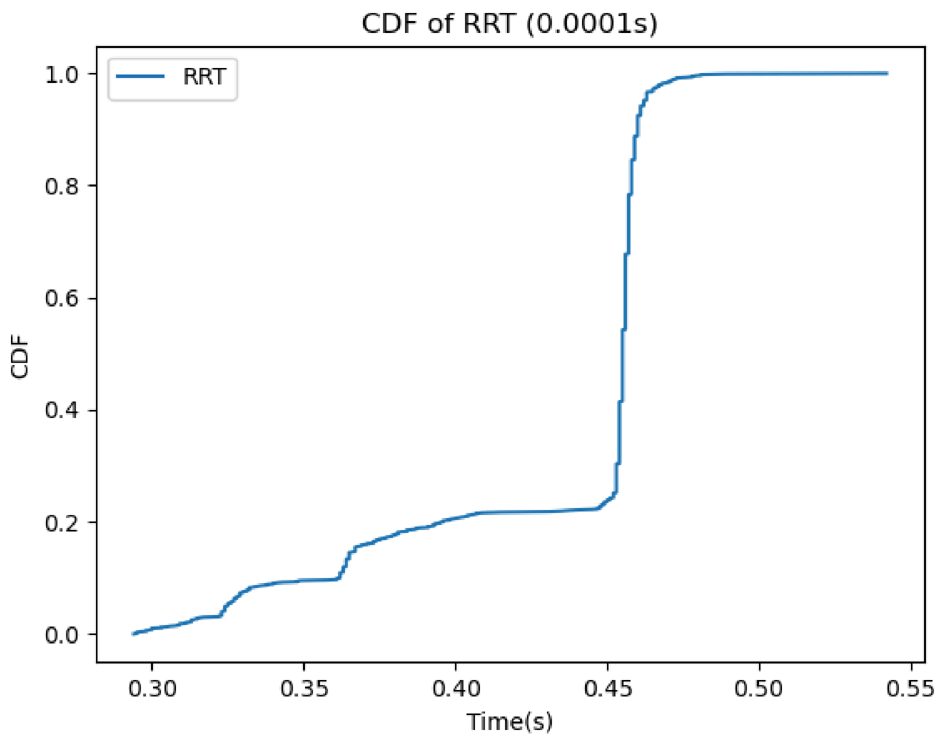

Figure 7: 0.0001 RRT CDF

We see that the graphs look very similar to each other. This is due to the clock period and the period of the internal program of the cpu, making the probabilistic distribution similar.

Q:Can you speculate why different intervals lead to different round trip results? What do you estimate is the most accurate measured parameter (e.g., min, max, mean) that can be used to estimate propagation time between the two machines? The reason is mentioned above, due to increasing allocation of computational resources to the responding of CPU as time interval gets closer to flooding.

Under normal situations, the CPU never gets close to flooding, so the non-flooding parameters are good estimators.The parameters of 0.01s interval of min/avg/max/mdev = 0.304/0.500/0.619/0.075 ms are good estimates.

Iperf

Q: What is effective bandwidith of Pi(client) and machine(server)?

```
0.0000-1.0000 sec   114 MBytes   956 Mbits/sec
[  1] 1.0000-2.0000 sec   112 MBytes   940 Mbits/sec
[  1] 2.0000-3.0000 sec   112 MBytes   940 Mbits/sec
[  1] 3.0000-4.0000 sec   113 MBytes   947 Mbits/sec
[  1] 4.0000-5.0000 sec   112 MBytes   937 Mbits/sec
[  1] 5.0000-6.0000 sec   112 MBytes   941 Mbits/sec
[  1] 6.0000-7.0000 sec   113 MBytes   947 Mbits/sec
[  1] 7.0000-8.0000 sec   113 MBytes   945 Mbits/sec
[  1] 8.0000-9.0000 sec   112 MBytes   936 Mbits/sec
[  1] 9.0000-10.0000 sec   112 MBytes   941 Mbits/sec
[  1] 0.0000-10.0203 sec  1.10 GBytes   941 Mbits/sec
```

Figure 8: Bandwidth of Lab Machine being Server

We see that the bandwith is around 940Mbits/s

Q: What about the other way around?

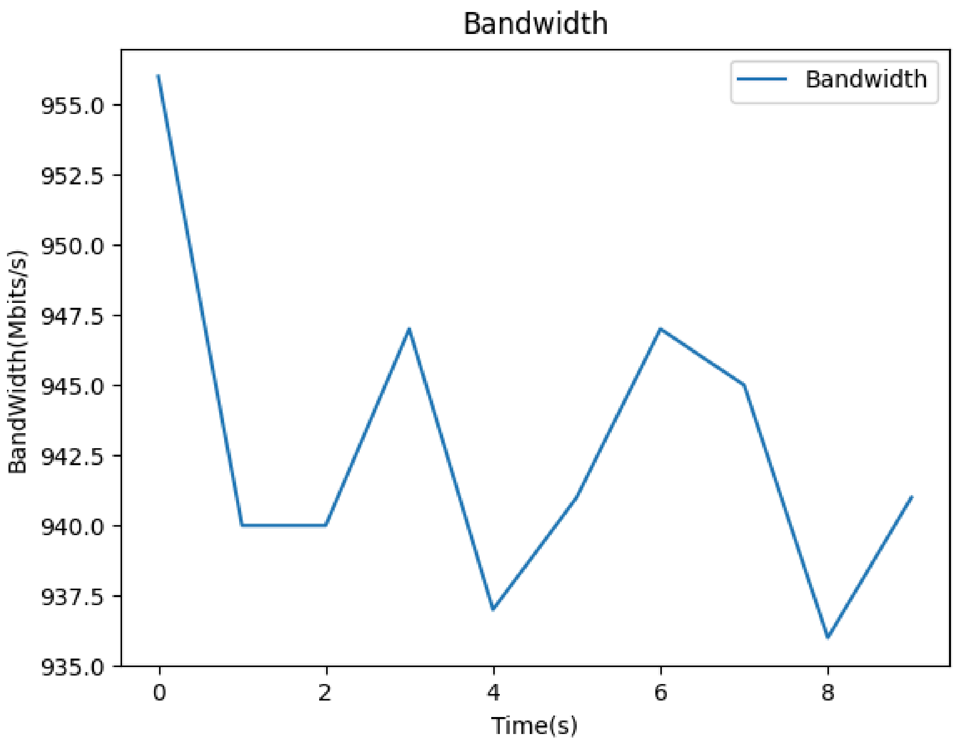

Figure 9: Bandwidth of Rasberry Pi Server

We see that the bandwidth is approximately the same as in the previous case due to symmetrical connection nature of the system.

Q: What about for bidirectional Iperf?

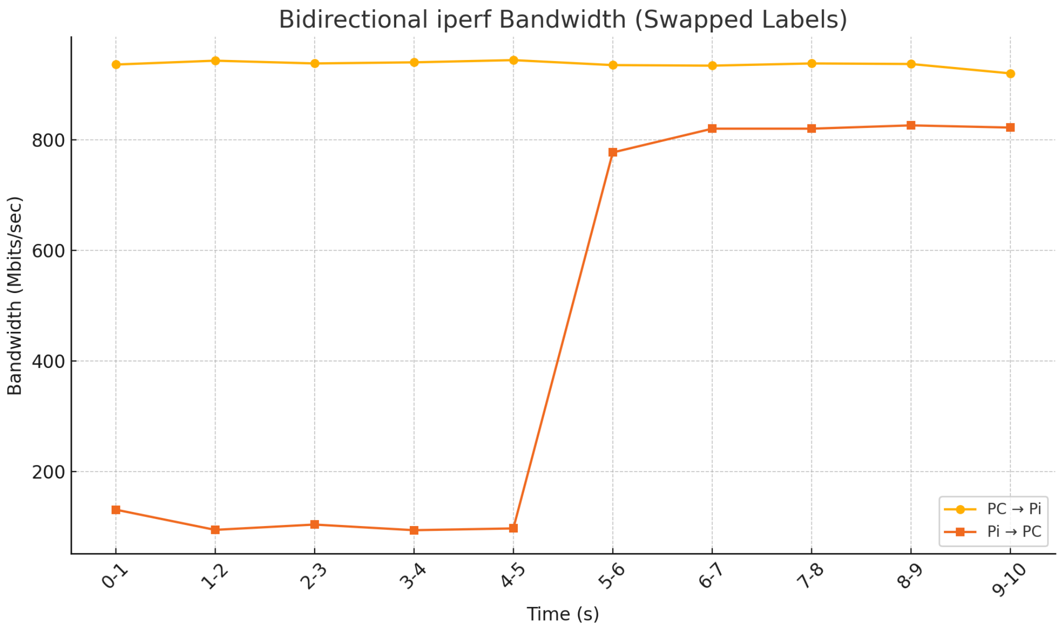

Figure 9: Bidirectional Iperf

We see that bidirectional iperf yields very different result. This is because of the traffic congestion and the limited CPU loading capacity of the rasberry

Pi to be a send data and receive data at the same time, but after 5s, the bandwidth goes up due to buffer adjustment within the pi.

Q: Run one way iperf using UDP, from the lab machine to the Raspberry Pi, 5 sec long, with varying bandwidth (100Kb/s, 1Mb/s, 100Mb/s).

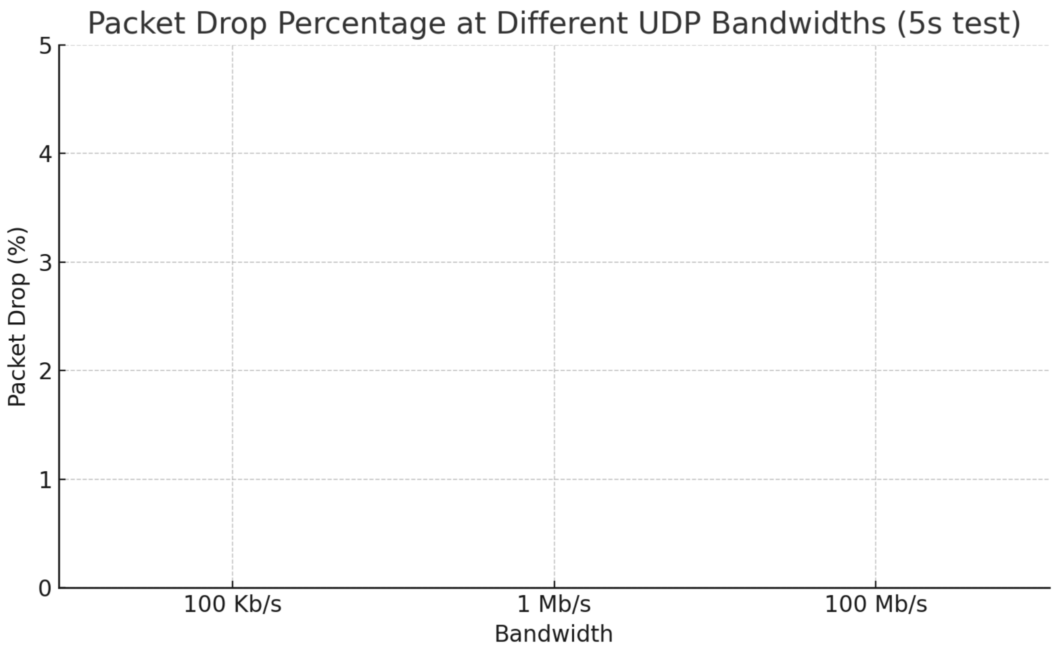

Figure 10: 0 Loss

Sadly there is no packet drop. This is due to packet drop does not include retransmission, and the nodes would keep requesting retransmission until all datagram is received.

```
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec  14.4 KBytes   118 Kbits/sec
[  1] 1.0000-2.0000 sec  12.9 KBytes   106 Kbits/sec
[  1] 2.0000-3.0000 sec  11.5 KBytes  94.1 Kbits/sec
[  1] 3.0000-4.0000 sec  12.9 KBytes   106 Kbits/sec
[  1] 4.0000-5.0000 sec  11.5 KBytes  94.1 Kbits/sec
[  1] 0.0000-5.1747 sec  66.0 KBytes   105 Kbits/sec
[  1] Sent 47 datagrams
[  1] Server Report:
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-5.1743 sec  66.0 KBytes   105 Kbits/sec   0.003 ms 0/46 (0%)
```

```
[  1] local 192.168.10.1 port 40618 connected with 192.168.10.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec   125 KBytes  1.02 Mbits/sec
[  1] 1.0000-2.0000 sec   122 KBytes  1000 Kbits/sec
[  1] 2.0000-3.0000 sec   122 KBytes  1000 Kbits/sec
[  1] 3.0000-4.0000 sec   122 KBytes  1000 Kbits/sec
[  1] 4.0000-5.0000 sec   122 KBytes  1000 Kbits/sec
[  1] 0.0000-5.0219 sec   616 KBytes  1.00 Mbits/sec
[  1] Sent 430 datagrams
[  1] Server Report:
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-5.0216 sec   616 KBytes  1.00 Mbits/sec   0.001 ms 0/429 (0%)
```

```
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec  11.9 MBytes   100 Mbits/sec
[  1] 1.0000-2.0000 sec  11.9 MBytes   100 Mbits/sec
[  1] 2.0000-3.0000 sec  11.9 MBytes   100 Mbits/sec
[  1] 3.0000-4.0000 sec  11.9 MBytes   100 Mbits/sec
[  1] 4.0000-5.0000 sec  11.9 MBytes   100 Mbits/sec
[  1] 0.0000-5.0002 sec  59.6 MBytes   100 Mbits/sec
[  1] Sent 42522 datagrams
[  1] Server Report:
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-4.9998 sec  59.6 MBytes   100 Mbits/sec   0.007 ms 0/42521 (0%)
```

Figure 11: 100k&1m&100m

Not much to talk about, but notice that the fluctuation and jitter number for each cases, as we will be discussing these in iperf3.

Iperf3

Q: Plot of Iperf3 Bandwidth

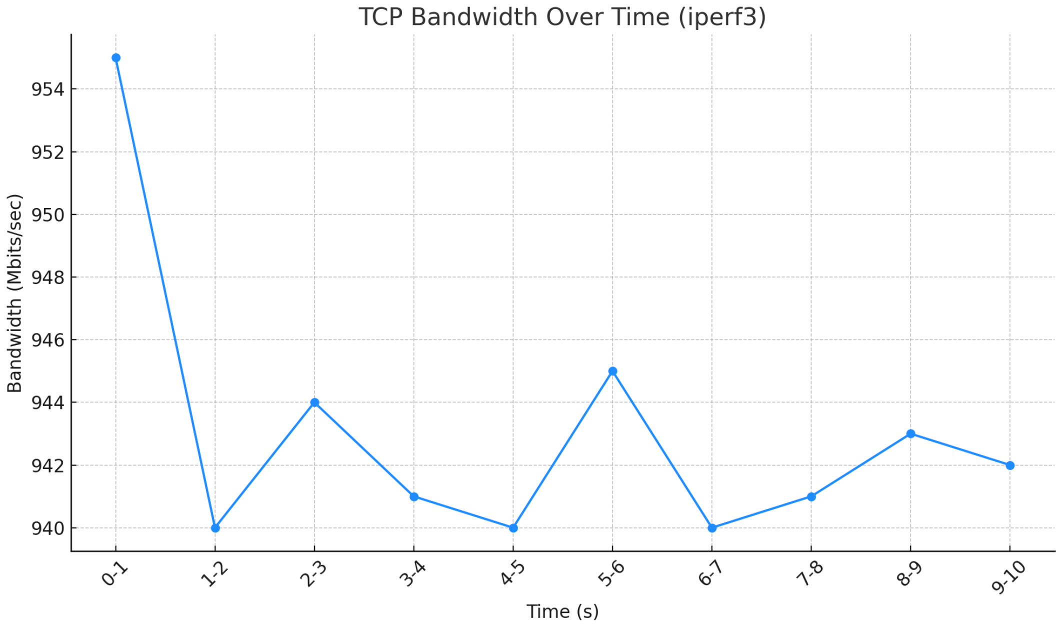

Figure 12: Iperf3 Bandwidth

We see that the result is similar to the iperf case.

Q:Run one way iperf3 using UDP, from the lab machine to the Raspberry Pi, 5 sec long, with varying bandwidth (100Kb s, 1Mb/s, 100Mb/s).

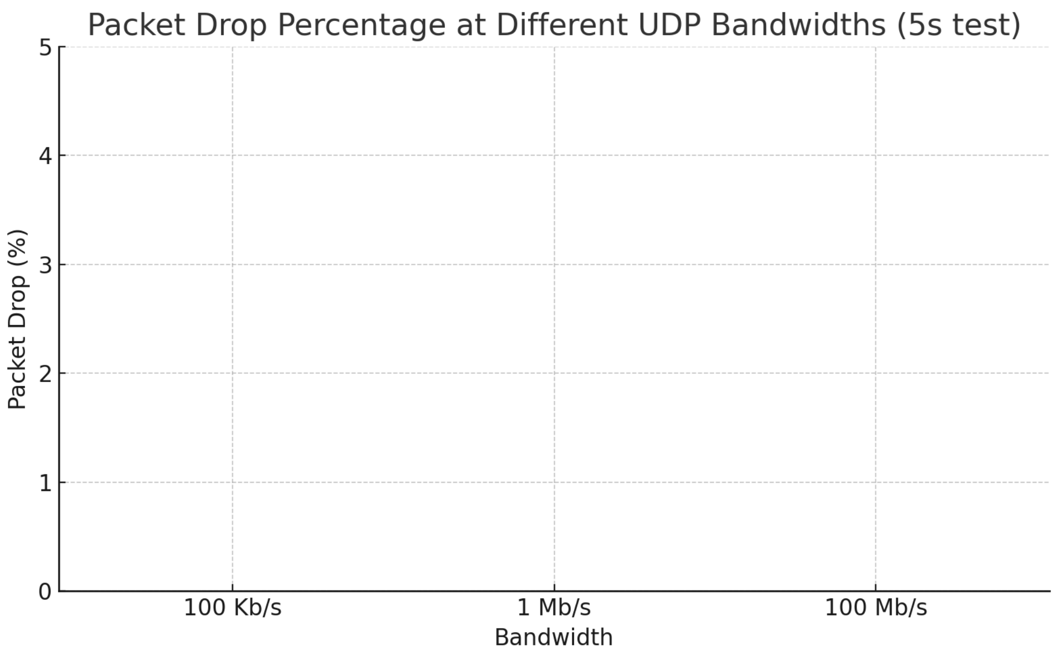

Figure 13: 0 Loss

We see that iperf3 also has 0 loss.

```
[  5] local 192.168.10.1 port 42161 connected to 192.168.10.2 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec  12.7 KBytes   104 Kbits/sec  9
[  5]   1.00-2.00   sec  12.7 KBytes   104 Kbits/sec  9
[  5]   2.00-3.00   sec  11.3 KBytes  92.7 Kbits/sec  8
[  5]   3.00-4.00   sec  12.7 KBytes   104 Kbits/sec  9
[  5]   4.00-5.00   sec  12.7 KBytes   104 Kbits/sec  9
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-5.00   sec  62.2 KBytes   102 Kbits/sec  0.000 ms  0/44 (0%)  sender
[  5]   0.00-5.04   sec  62.2 KBytes   101 Kbits/sec  0.009 ms  0/44 (0%)  receiver
```

```
[  5] local 192.168.10.1 port 48095 connected to 192.168.10.2 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec   123 KBytes  1.01 Mbits/sec  87
[  5]   1.00-2.00   sec   122 KBytes   996 Kbits/sec  86
[  5]   2.00-3.00   sec   122 KBytes   996 Kbits/sec  86
[  5]   3.00-4.00   sec   123 KBytes  1.01 Mbits/sec  87
[  5]   4.00-5.00   sec   122 KBytes   996 Kbits/sec  86
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-5.00   sec   611 KBytes  1.00 Mbits/sec  0.000 ms  0/432 (0%)  sender
[  5]   0.00-5.04   sec   611 KBytes   993 Kbits/sec  0.006 ms  0/432 (0%)  receiver
```

```
[  5] local 192.168.10.1 port 34847 connected to 192.168.10.2 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec  11.9 MBytes  99.9 Mbits/sec  8626
[  5]   1.00-2.00   sec  11.9 MBytes   100 Mbits/sec  8633
[  5]   2.00-3.00   sec  11.9 MBytes   100 Mbits/sec  8633
[  5]   3.00-4.00   sec  11.9 MBytes   100 Mbits/sec  8632
[  5]   4.00-5.00   sec  11.9 MBytes   100 Mbits/sec  8633
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-5.00   sec  59.6 MBytes   100 Mbits/sec  0.000 ms  0/43157 (0%)  sender
[  5]   0.00-5.04   sec  59.6 MBytes  99.2 Mbits/sec  0.010 ms  0/43157 (0%)  receiver
```

Figure 14: 100k&1m&100m

Q:Discuss any observed differences between iperf and iperf3 results.

We see that iperf3 has more jitter, which might be due to the fact that iperf3 is designed for single-core purpose, or that our data amount is just not sufficient to display actual jitter. Also iperf3 has more stable bandwidth(it's bandwidth fluctuation of 4k is smaller compared to the 8k fluctuation of iperf), which might also due to insufficient data.