Ping

Task 1: ping_test1.txt, processed_ping_test1.txt for only RTT
min/avg/max/mdev = 0.426/0.469/0.494/0.022 ms

Task 2: ping_test2.txt, processed_ping_test2.txt for only RTT
min/avg/max/mdev = 0.380/0.487/0.580/0.073 ms

The mean of task 1 is similar to task 2, and ideally, because the same physical cable is used to link the Raspberry Pi to the lab machine, the two figures should equal. However, the standard deviation of task 2 is different to task 1. This could be because the lab machine, being more complex than the single board computer of the Raspberry Pi, has to also process tasks in the background at any point in time, so this might reflect the increased RTT in task 2.

Task 3: ping_test3.txt, processed_ping_test3.txt for only RTT
min/avg/max/mdev = 0.322/0.419/0.603/0.065 ms

The interval of the ping is 1 ms but our RTT is always less than 1 ms. This means congestion is not an issue. Very likely, the same normal distribution can describe the RTT values in this task as well as task 2. It is thus likely because of the larger sample size in task 3 that we sampled from the middle of the distribution more than the tail, while in task 2 we just so happened to sample from the tail more frequently than the probability density would predict, and this skewed the perception of the results because of the smaller sample size. It is thus likely that the result of task 3 illustrates the real thing more accurately because of the larger sample size.

~~It is interesting that the minimum and mean here is lower than task 2, despite the interval being shorter and having a higher risk of congesting the network. However, the maximum here is higher. This tells me that, similar to what is happening in task 5, the link is utilised more here, so the maximum in task 3 is higher due to congestion. However, more processing power is also dedicated to process the ping, so this could be a reason why the mean is lower.~~

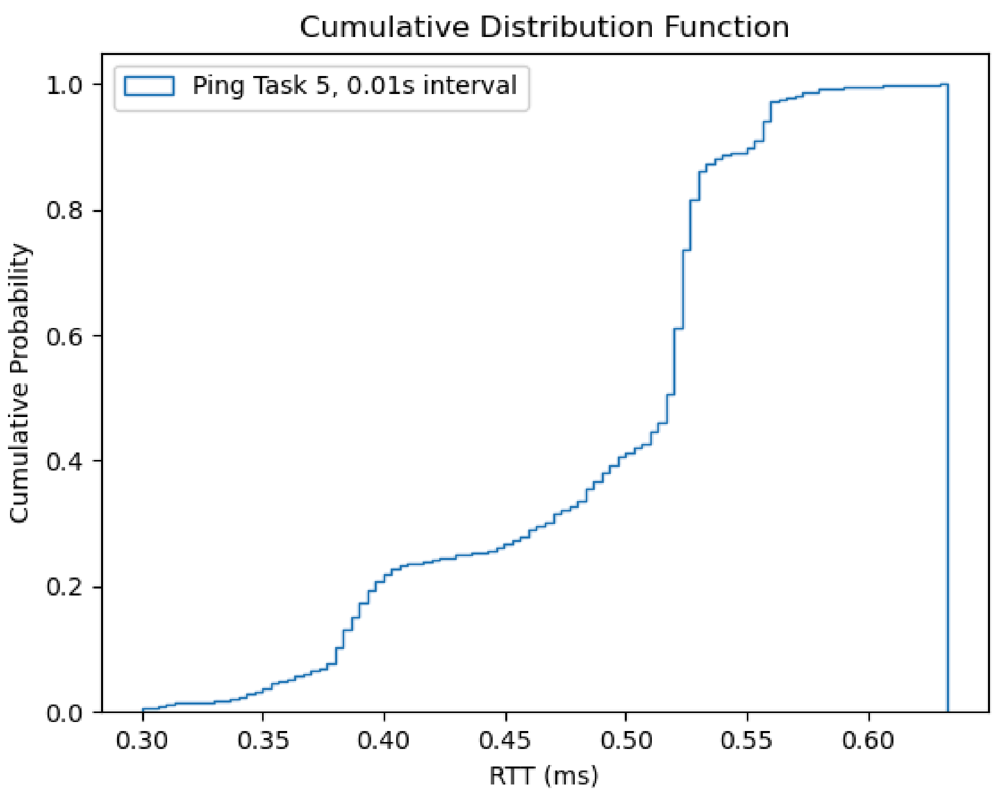
Similarly, because the target for the ping is the lab machine, the minimum value could be when the lab machine is instantaneously not that busy processing background tasks, while the maximum value could be when the lab machine is busy processing background tasks, and the mean is just an average between the two cases.

Task 4: ping_test4.txt, processed_ping_test4.txt for only RTT
min/avg/max/mdev = 0.285/0.420/1.517/0.052 ms

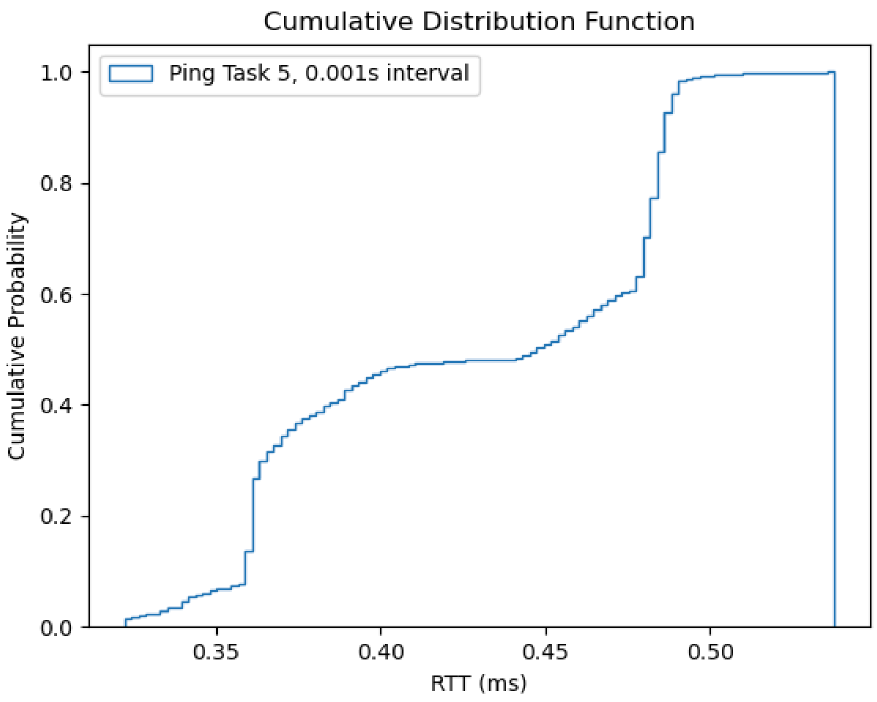
We explicitly force flooding here. We note the mean is very similar to task 3. The minimum is slightly lower. The maximum is much bigger. But interestingly, the standard deviation is lower than in task 3. This gives reason to think that we are getting a more accurate distribution to model the actual thing. Using the same reasoning about the normal distribution, the higher sample size makes it such that it is more probable to sample from the tail of the normal distribution compared to in task 3. This explains the minimum being slightly less than that of task 3. Moreover, the lower standard deviation yet with a maximum that is much higher tells me this maximum value is an outlier. It was likely an instance when the lab machine was instantaneously extremely busy handling a task in the background, and so had to delay the response to the ping.

~~We explicitly force flooding here. Therefore, the potential reason for a very high maximum is that we congest the network so much that we induce congestion in the network.~~

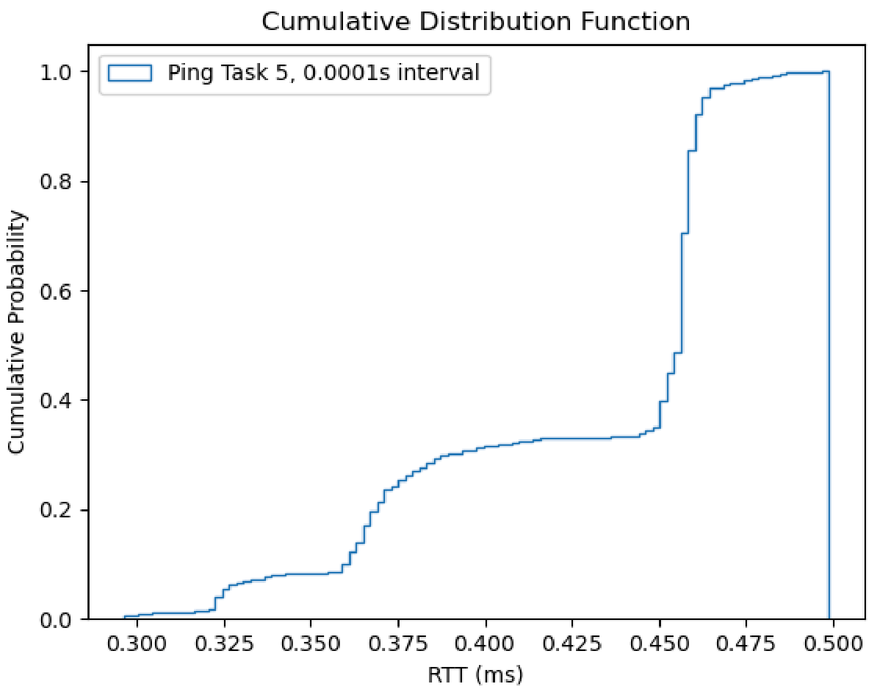
Task 5: ping_log1.txt, ping_log2.txt, and ping_log3.txt for intervals 0.01, 0.001, and 0.0001s respectively. Also processed_ping_log1.txt, processed_ping_log2.txt, processed_ping_log3.txt for only RTT respectively.



From ping_log1.txt: min/avg/max/mdev = 0.300/0.484/0.633/0.066 ms



From ping_log2.txt: min/avg/max/mdev = 0.322/0.425/0.538/0.058 ms



From ping_log3.txt: min/avg/max/mdev = 0.296/0.426/0.499/0.048 ms

We see the intervals 0.001 and 0.0001s have a similar mean, and this is decently lower than the interval of 0.01s. The minimum value is quite similar for all three intervals. As we send packets more frequently, we may expect RTT to increase as we strain and start overwhelming the network. The data I received could indicate that there is a threshold between the intervals of 0.01 and 0.001s when the computer started to dedicate more processing power to the pinging.

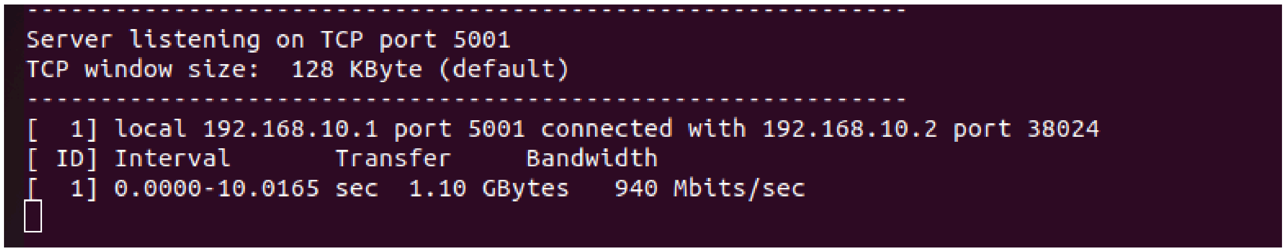
Moreover, the similar minimum values indicate that this the minimum value is due to the propagation delay in the cables between the two computers (if the cable is 10 Gbps and a packet is 512b, then it should take about 51.2ns each way -> theoretically 0.102ms for round trip) plus any delay at the receiving side both ways (such as due to background tasks that are always there). In terms of estimating propagation time of the signal in the cable then, the minimum value is the most accurate, but this will still be the upper bound on the theoretical propagation time.

~~From these data, we see the intervals 0.001 and 0.0001s have a similar mean, and it is significantly lower than the interval 0.01s. This is interesting, because I expect as we send packets more frequently, RTT would increase as we strain and start overwhelming the network. The data I received could indicate that there is a threshold between the intervals of 0.01 and 0.001s when the computer started to dedicate more processing power to the pinging.~~

~~In all the results, the standard deviation is significantly smaller than the range so the mean is likely to be the most accurate parameter to predict the propagation time. It is interesting to note in the third interval (0.0001s), the data is much more skewed to the higher end: the mean is much closer to the max than the min. This indicates to me that an interval of 0.0001s is approaching flooding and the network is starting to be overwhelmed, and as we lower the interval even more, the mean will approach the max.~~

iPerf

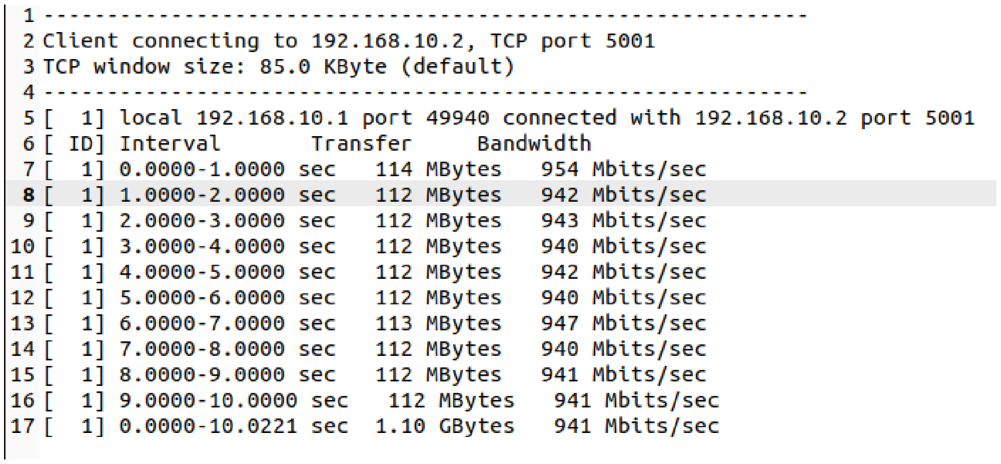
Task 1



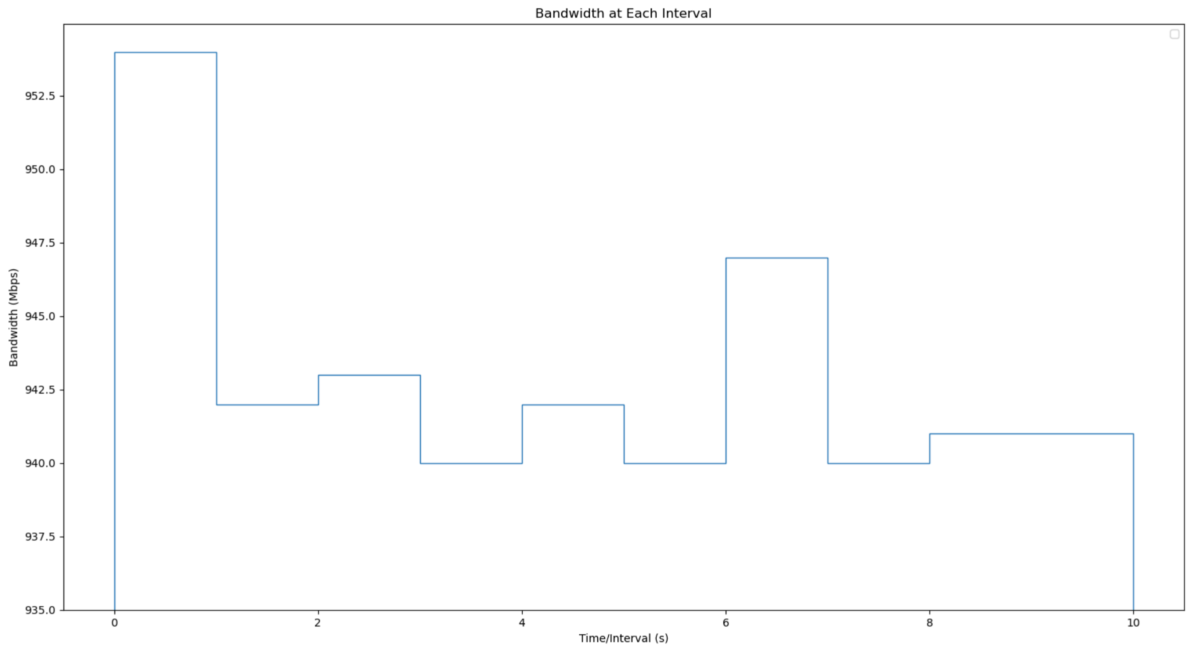
```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  1] local 192.168.10.1 port 5001 connected with 192.168.10.2 port 38024
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0165 sec  1.10 GBytes   940 Mbits/sec
```

Task 2

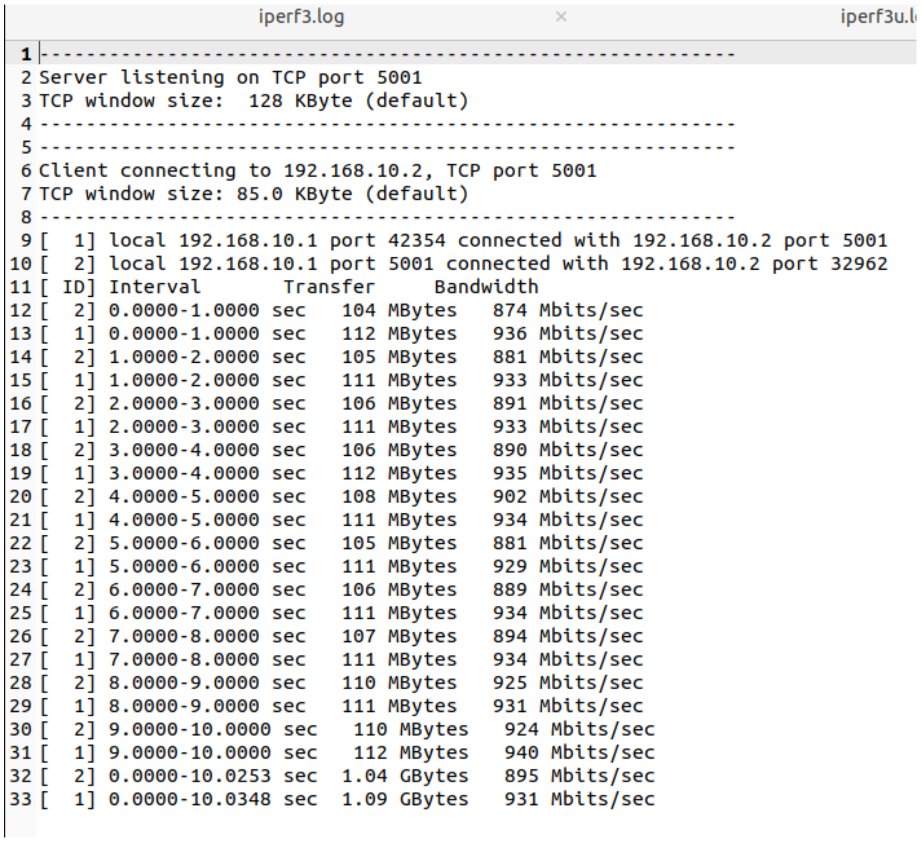
iperf.log



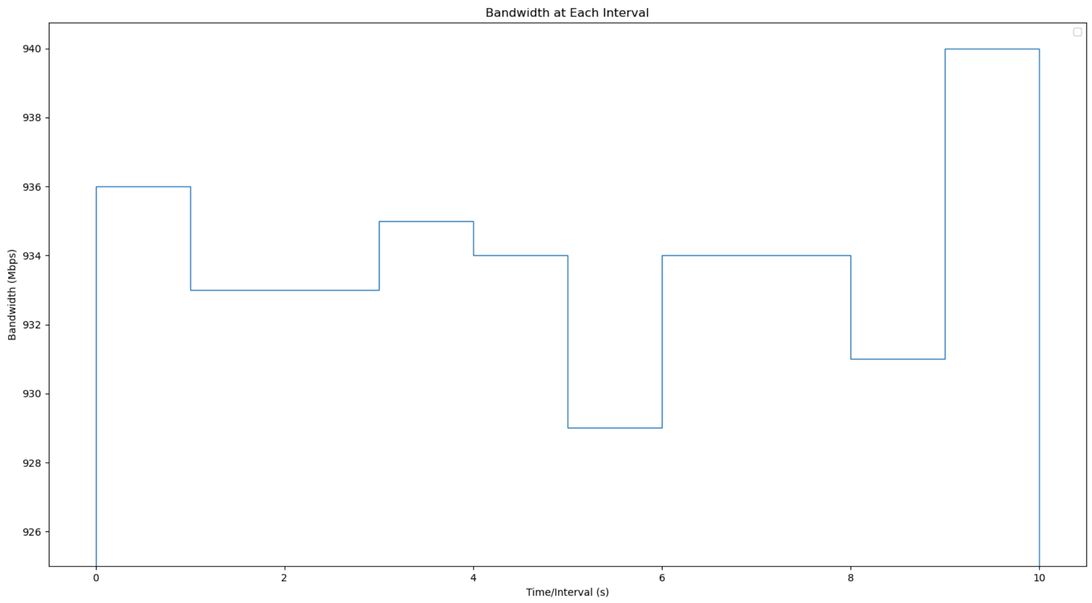
```
------------------------------------------------------------
Client connecting to 192.168.10.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  1] local 192.168.10.1 port 49940 connected with 192.168.10.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec   114 MBytes   954 Mbits/sec
[  1] 1.0000-2.0000 sec   112 MBytes   942 Mbits/sec
[  1] 2.0000-3.0000 sec   112 MBytes   943 Mbits/sec
[  1] 3.0000-4.0000 sec   112 MBytes   940 Mbits/sec
[  1] 4.0000-5.0000 sec   112 MBytes   942 Mbits/sec
[  1] 5.0000-6.0000 sec   112 MBytes   940 Mbits/sec
[  1] 6.0000-7.0000 sec   113 MBytes   947 Mbits/sec
[  1] 7.0000-8.0000 sec   112 MBytes   940 Mbits/sec
[  1] 8.0000-9.0000 sec   112 MBytes   941 Mbits/sec
[  1] 9.0000-10.0000 sec   112 MBytes   941 Mbits/sec
[  1] 0.0000-10.0221 sec  1.10 GBytes   941 Mbits/sec
```



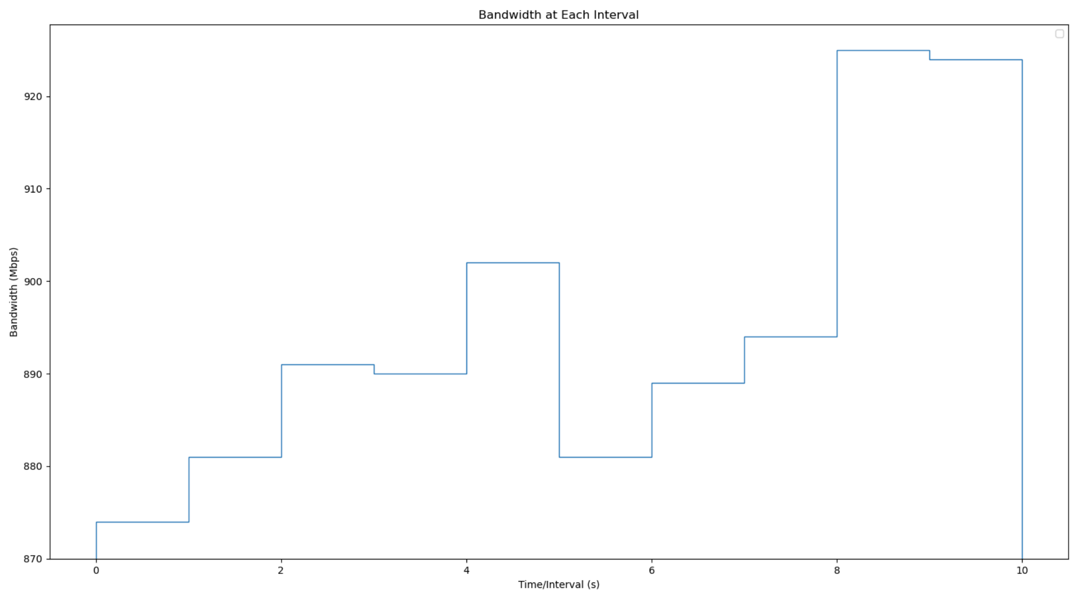
Task 3
Iperfbi.log (server side)
Iperfbi_client.log (client side, shown below)



```
 1 ------------------------------------------------------------
 2 Server listening on TCP port 5001
 3 TCP window size:  128 KByte (default)
 4 ------------------------------------------------------------
 5 ------------------------------------------------------------
 6 Client connecting to 192.168.10.2, TCP port 5001
 7 TCP window size: 85.0 KByte (default)
 8 ------------------------------------------------------------
 9 [  1] local 192.168.10.1 port 42354 connected with 192.168.10.2 port 5001
10 [  2] local 192.168.10.1 port 5001 connected with 192.168.10.2 port 32962
11 [ ID] Interval       Transfer     Bandwidth
12 [  2] 0.0000-1.0000 sec   104 MBytes   874 Mbits/sec
13 [  1] 0.0000-1.0000 sec   112 MBytes   936 Mbits/sec
14 [  2] 1.0000-2.0000 sec   105 MBytes   881 Mbits/sec
15 [  1] 1.0000-2.0000 sec   111 MBytes   933 Mbits/sec
16 [  2] 2.0000-3.0000 sec   106 MBytes   891 Mbits/sec
17 [  1] 2.0000-3.0000 sec   111 MBytes   933 Mbits/sec
18 [  2] 3.0000-4.0000 sec   106 MBytes   890 Mbits/sec
19 [  1] 3.0000-4.0000 sec   112 MBytes   935 Mbits/sec
20 [  2] 4.0000-5.0000 sec   108 MBytes   902 Mbits/sec
21 [  1] 4.0000-5.0000 sec   111 MBytes   934 Mbits/sec
22 [  2] 5.0000-6.0000 sec   105 MBytes   881 Mbits/sec
23 [  1] 5.0000-6.0000 sec   111 MBytes   929 Mbits/sec
24 [  2] 6.0000-7.0000 sec   106 MBytes   889 Mbits/sec
25 [  1] 6.0000-7.0000 sec   111 MBytes   934 Mbits/sec
26 [  2] 7.0000-8.0000 sec   107 MBytes   894 Mbits/sec
27 [  1] 7.0000-8.0000 sec   111 MBytes   934 Mbits/sec
28 [  2] 8.0000-9.0000 sec   110 MBytes   925 Mbits/sec
29 [  1] 8.0000-9.0000 sec   111 MBytes   931 Mbits/sec
30 [  2] 9.0000-10.0000 sec   110 MBytes   924 Mbits/sec
31 [  1] 9.0000-10.0000 sec   112 MBytes   940 Mbits/sec
32 [  2] 0.0000-10.0253 sec  1.04 GBytes   895 Mbits/sec
33 [  1] 0.0000-10.0348 sec  1.09 GBytes   931 Mbits/sec
```

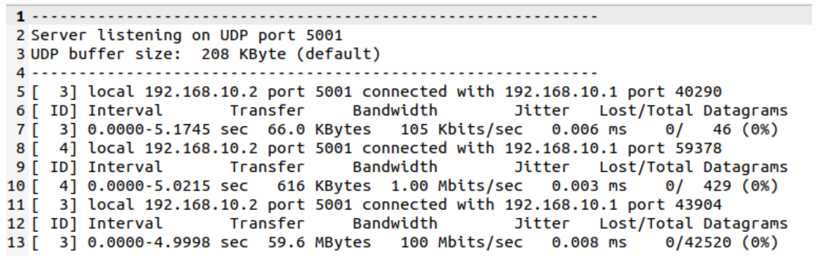


This one is regarding ID 1. Based on line 2, 6, and 9, this means that ID 1 is from the lab machine to the Raspberry Pi.

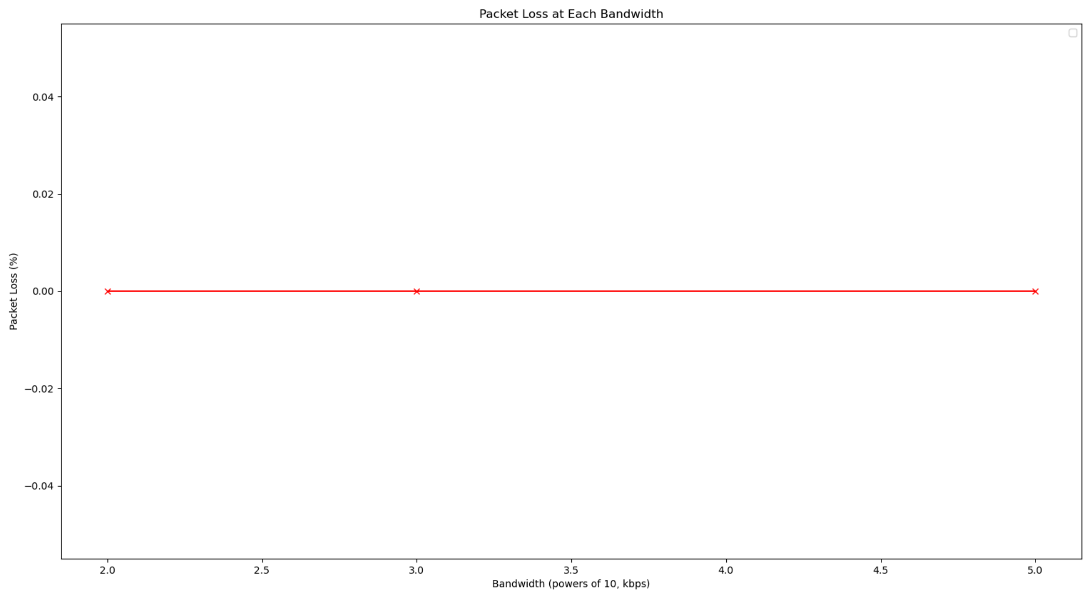


This one is regarding ID 2. Conversely, then, this is from the Raspberry Pi to the lab machine. ID 2 bandwidth is on average lower than ID 1 bandwidth (seen from lines 32-33), likely because of the background tasks of the lab machine.

Task 4:
Iperfu.log (shown below)



```
------------------------------------------------------------
Server listening on UDP port 5001
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.2 port 5001 connected with 192.168.10.1 port 40290
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3] 0.0000-5.1745 sec  66.0 KBytes   105 Kbits/sec   0.006 ms    0/   46 (0%)
[  4] local 192.168.10.2 port 5001 connected with 192.168.10.1 port 59378
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  4] 0.0000-5.0215 sec   616 KBytes  1.00 Mbits/sec   0.003 ms    0/  429 (0%)
[  3] local 192.168.10.2 port 5001 connected with 192.168.10.1 port 43904
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3] 0.0000-4.9998 sec  59.6 MBytes   100 Mbits/sec   0.008 ms    0/42520 (0%)
```



For 100 Kbps: Transferred 66 KB @105Kbps, ~~jitter/standard deviation of 0.006ms.~~
For 1 Mbps: Transferred 616 KB @ 1 Mbps, ~~jitter/standard deviation of 0.003ms.~~
For 100 Mbps: Transferred 59.6 MB (59600-ish KB) @ 100 Mbps, ~~jitter/standard deviation of 0.008ms (greatest).~~

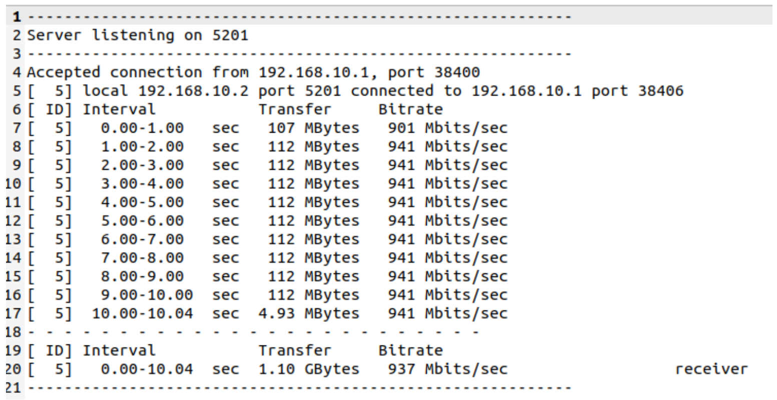
For all tests, no packets were lost.
Above, the data transferred is almost equal to bandwidth multiplied by 5 seconds, divided by 8 (converting bits to bytes), as expected. Alongside the 0 packet loss, this gives me the impression that the network is not congested and was able to keep up with the high rate of data that was sent. The throughput is thus good and as expected.

~~While the jitter of the second test is about half the first and the third, it is not that different in absolute terms. The jitter thus might seem different simply because this is such a small time scale, and the numbers given were only correct to 1 significant figure. In reality, this variation is likely to be normal.~~

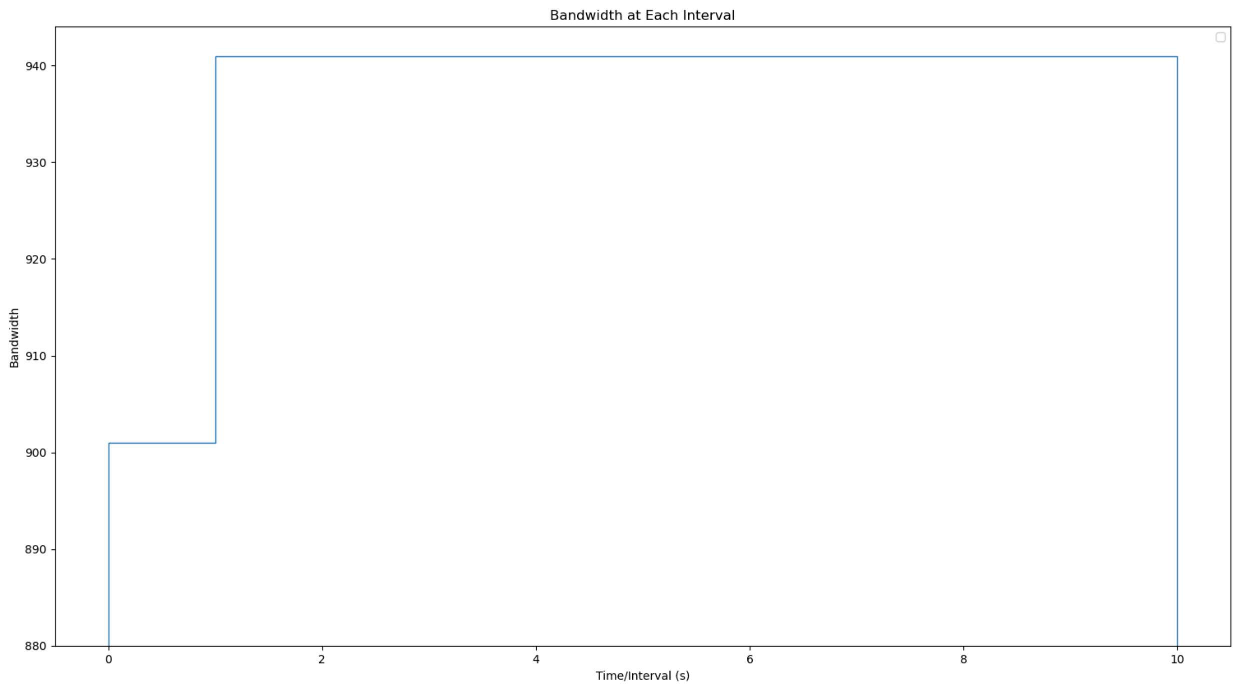
iPerf3

Task 1

Iperf3.log (shown below)

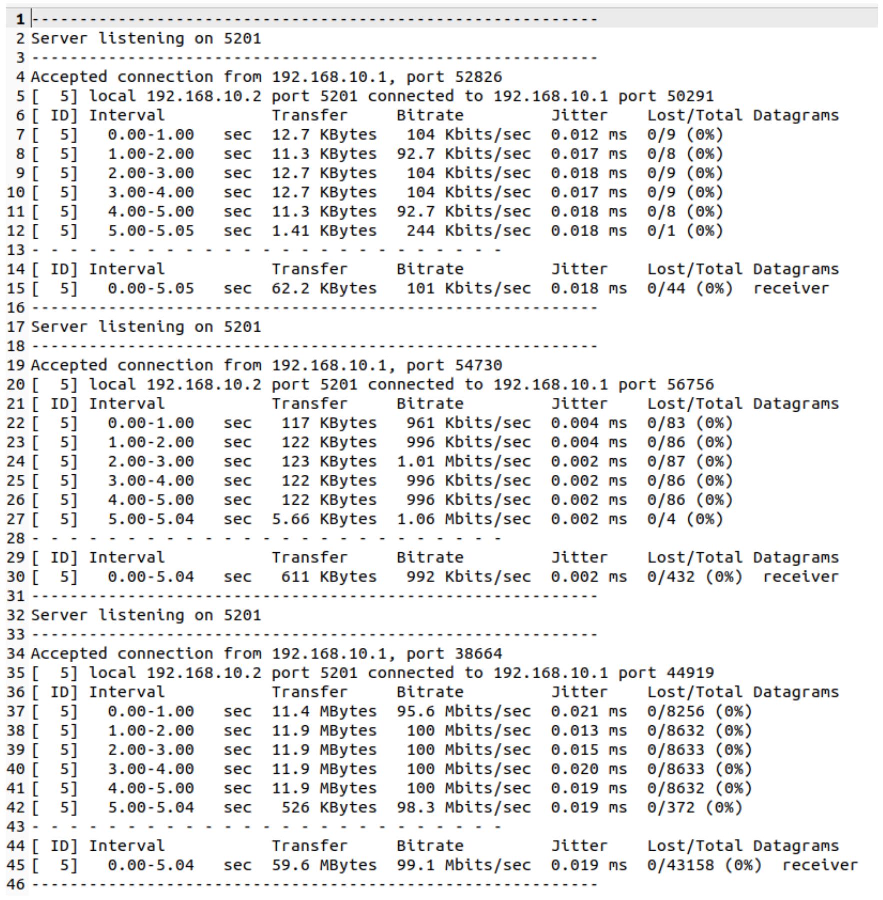


```
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.10.1, port 38400
[  5] local 192.168.10.2 port 5201 connected to 192.168.10.1 port 38406
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec   107 MBytes   901 Mbits/sec
[  5]   1.00-2.00   sec   112 MBytes   941 Mbits/sec
[  5]   2.00-3.00   sec   112 MBytes   941 Mbits/sec
[  5]   3.00-4.00   sec   112 MBytes   941 Mbits/sec
[  5]   4.00-5.00   sec   112 MBytes   941 Mbits/sec
[  5]   5.00-6.00   sec   112 MBytes   941 Mbits/sec
[  5]   6.00-7.00   sec   112 MBytes   941 Mbits/sec
[  5]   7.00-8.00   sec   112 MBytes   941 Mbits/sec
[  5]   8.00-9.00   sec   112 MBytes   941 Mbits/sec
[  5]   9.00-10.00  sec   112 MBytes   941 Mbits/sec
[  5]  10.00-10.04  sec  4.93 MBytes   941 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.04  sec  1.10 GBytes   937 Mbits/sec                  receiver
-----------------------------------------------------------
```

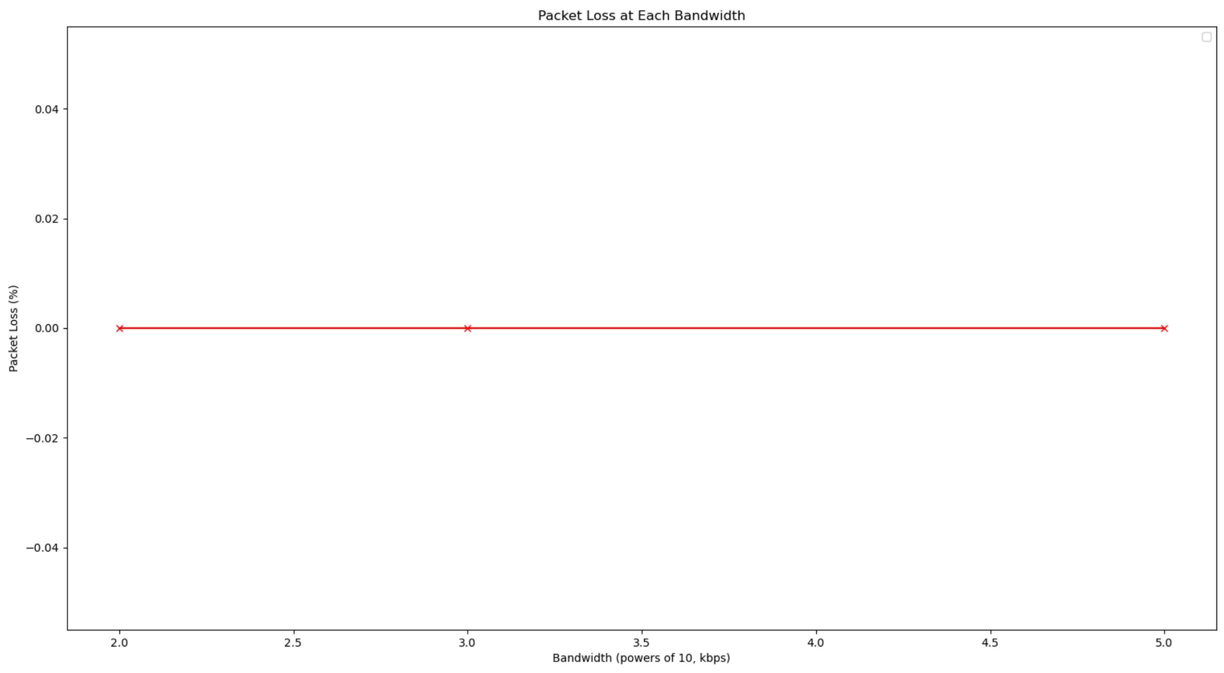


Task 2

Iperf3u.log (shown below)



```
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.10.1, port 52826
[  5] local 192.168.10.2 port 5201 connected to 192.168.10.1 port 50291
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec  12.7 KBytes   104 Kbits/sec  0.012 ms  0/9 (0%)
[  5]   1.00-2.00   sec  11.3 KBytes  92.7 Kbits/sec  0.017 ms  0/8 (0%)
[  5]   2.00-3.00   sec  12.7 KBytes   104 Kbits/sec  0.018 ms  0/9 (0%)
[  5]   3.00-4.00   sec  12.7 KBytes   104 Kbits/sec  0.017 ms  0/9 (0%)
[  5]   4.00-5.00   sec  11.3 KBytes  92.7 Kbits/sec  0.018 ms  0/8 (0%)
[  5]   5.00-5.05   sec  1.41 KBytes   244 Kbits/sec  0.018 ms  0/1 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-5.05   sec  62.2 KBytes   101 Kbits/sec  0.018 ms  0/44 (0%)  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.10.1, port 54730
[  5] local 192.168.10.2 port 5201 connected to 192.168.10.1 port 56756
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec   117 KBytes   961 Kbits/sec  0.004 ms  0/83 (0%)
[  5]   1.00-2.00   sec   122 KBytes   996 Kbits/sec  0.004 ms  0/86 (0%)
[  5]   2.00-3.00   sec   123 KBytes  1.01 Mbits/sec  0.002 ms  0/87 (0%)
[  5]   3.00-4.00   sec   122 KBytes   996 Kbits/sec  0.002 ms  0/86 (0%)
[  5]   4.00-5.00   sec   122 KBytes   996 Kbits/sec  0.002 ms  0/86 (0%)
[  5]   5.00-5.04   sec  5.66 KBytes  1.06 Mbits/sec  0.002 ms  0/4 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-5.04   sec   611 KBytes   992 Kbits/sec  0.002 ms  0/432 (0%)  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.10.1, port 38664
[  5] local 192.168.10.2 port 5201 connected to 192.168.10.1 port 44919
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec  11.4 MBytes  95.6 Mbits/sec  0.021 ms  0/8256 (0%)
[  5]   1.00-2.00   sec  11.9 MBytes   100 Mbits/sec  0.013 ms  0/8632 (0%)
[  5]   2.00-3.00   sec  11.9 MBytes   100 Mbits/sec  0.015 ms  0/8633 (0%)
[  5]   3.00-4.00   sec  11.9 MBytes   100 Mbits/sec  0.020 ms  0/8633 (0%)
[  5]   4.00-5.00   sec  11.9 MBytes   100 Mbits/sec  0.019 ms  0/8632 (0%)
[  5]   5.00-5.04   sec   526 KBytes  98.3 Mbits/sec  0.019 ms  0/372 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-5.04   sec  59.6 MBytes  99.1 Mbits/sec  0.019 ms  0/43158 (0%)  receiver
-----------------------------------------------------------
```



For 100 Kbps: Transferred 62.2 KB @ 101 Kbps, ~~jitter/standard deviation of 0.018ms.~~
For 1 Mbps: Transferred 611 KB, @ 992 Kbps, ~~jitter/standard deviation of 0.002ms.~~
For 100 Mbps: Transferred 59.6 MB (59600-ish KB) @ 99.1 Mbps, ~~jitter/standard deviation of 0.019ms (greatest).~~

For all tests, no packets were lost.
Above, the data transferred is almost equal to bandwidth multiplied by 5 seconds, divided by 8 (converting bits to bytes), as expected. Alongside the 0 packet loss, this gives me the impression that the network is not congested and was able to keep up with the high rate of data that was sent. The throughput is thus good and as expected.

We see that the bitrate for iperf3 at each tested bandwidth is less than or equal to the corresponding bandwidth in iperf. iperf3 thus performs slightly worse than iperf, although not by a significant amount (only about 1% worse max in this experiment). A quick Google search reveals that iperf3 is a rewrite of iperffrom the ground up, and its goal is to have a simpler code base than regular iPerf (https://software.es.net/iperf/faq.html). This website states that older version of iperf3 do not support CPU core multi-threading. Since it only uses one CPU core, it will start suffering at higher bitrates, which we see a little of with the test at the two higher bandwidths

~~Similar to regular iPerf, here the jitter for 1 Mbps is the lowest, but here it is significantly lower than than the other bandwidths. The data transferred with iPerf3 is also less than or equal to the same bandwidth with iPerf.~~

~~A quick Google search reveals that iPerf3 is a rewrite of iPerf from the ground up, and its goal is to have a simpler code base than regular iPerf (https://software.es.net/iperf/faq.html). This makes me think that iPerf3 trades simplicity for some performance and accuracy. The same website also shows that, if the version of iPerf3 we use is older, then it only uses one CPU core which may explain why there is more jitter in iPerf3.~~