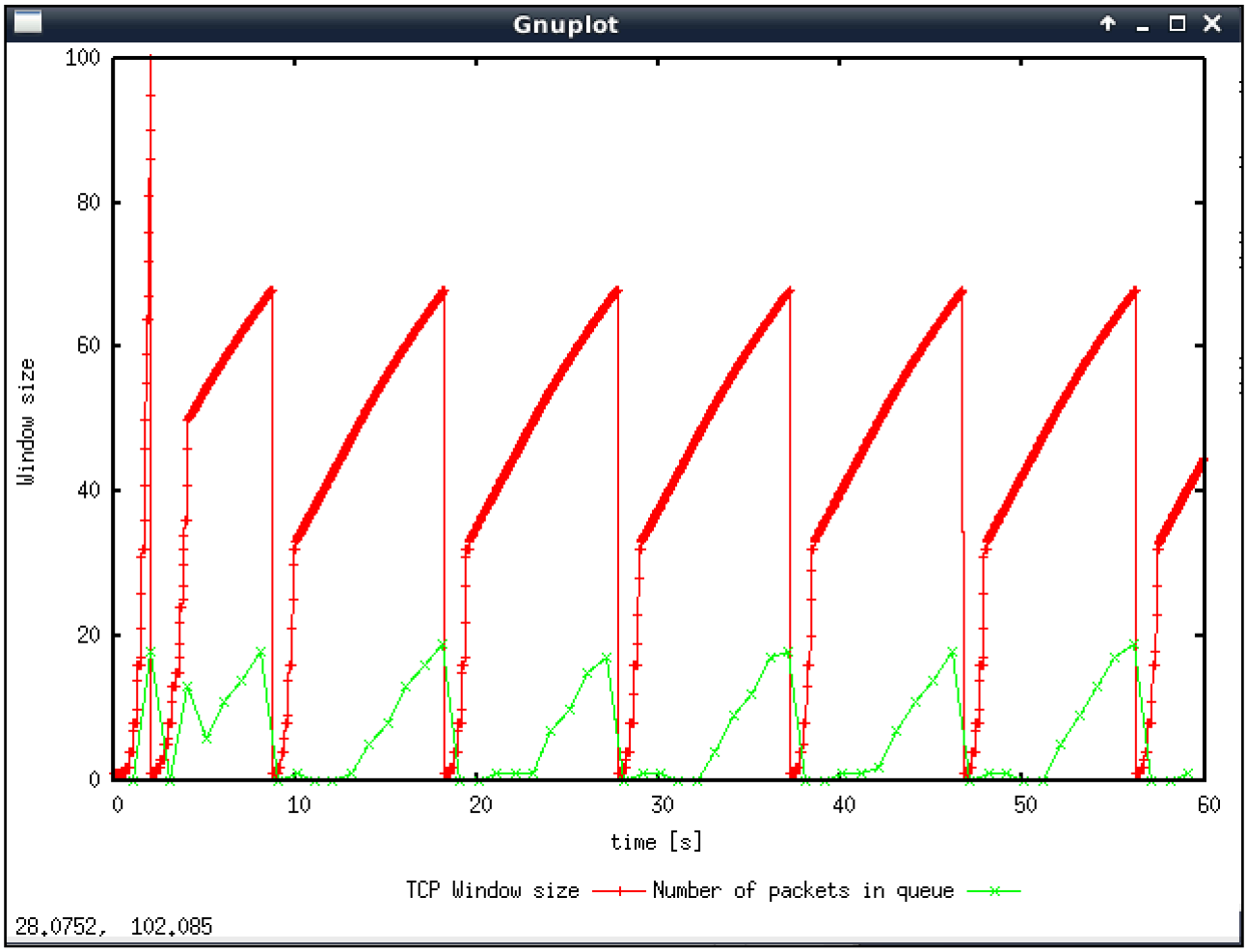
ZID: z5230310 Name: Tian Liu Date:17/07/2019

E1

Q1:

As the picture shows below:



The maximum size of the congestion window that the TCP flow reaches in this case is **100**.

When the congestion window reaches the maximum value, the threshold will be half of the previous maximum congestion window size value and the new congestion window size value will be set to 1.

The reason is that packet loss result in **timeout**.

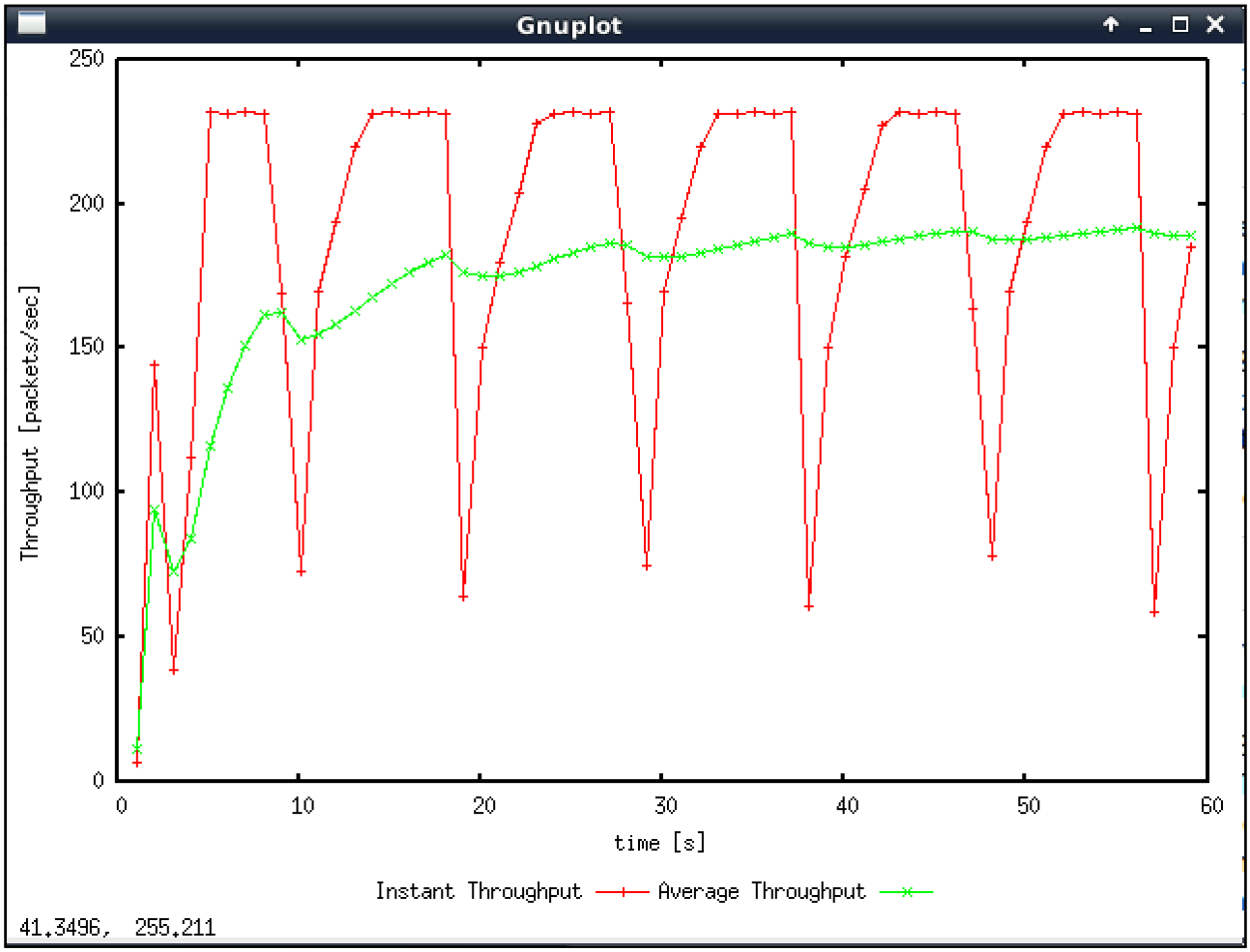
Then TCP flow begin **slow start** which increase by exponential growth until it reaches the threshold and increase by liner growth which called **congestion avoidance phrase**. Until the packet loss again, repeat the process.

Q2:

The average throughput of TCP in this case is the green line as the picture shows below:

After 20s, the throughput becomes stable, approximately **190 packets/sec**.

190 * (500 + 20 + 20) * 8 = **820800 bits/sec.**



Q3:

According the graphs of Q1 & Q2, I try to make the initial max congestion window size 50 and 66. The graphs show below:

When initial max congestion window size set to 66, there will be less congestion and the throughput tend to become stable.

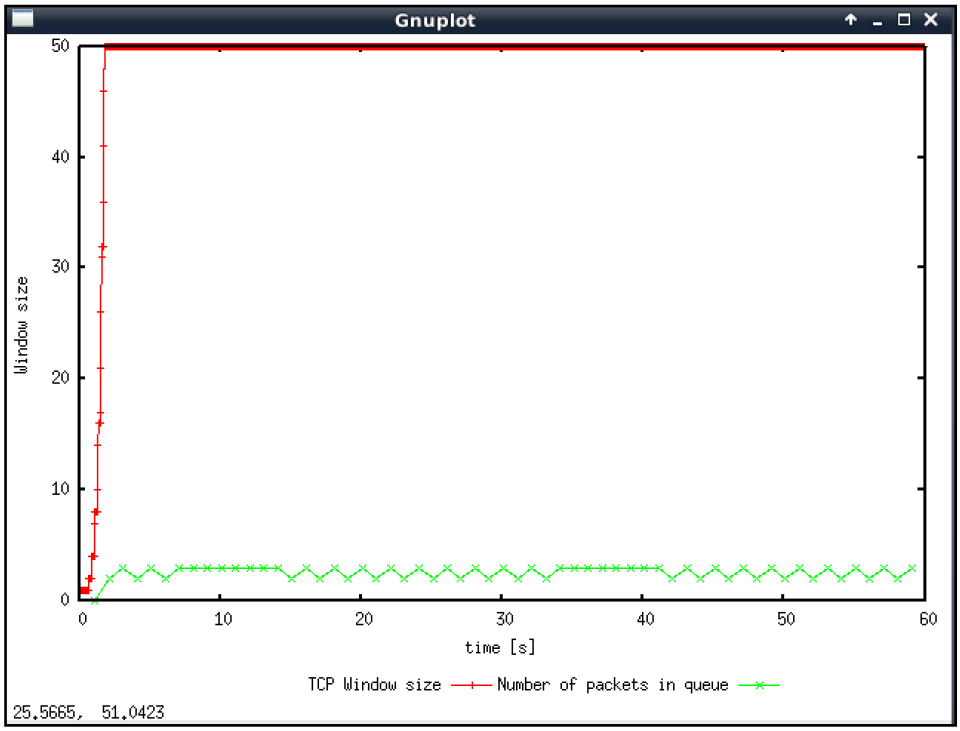
When set to 50, TCP flow stops oscillating and a straight line can be found in the graph.

And the average is around **231 packets/sec**. Can found on the fourth line of *WindowMon.tr*.

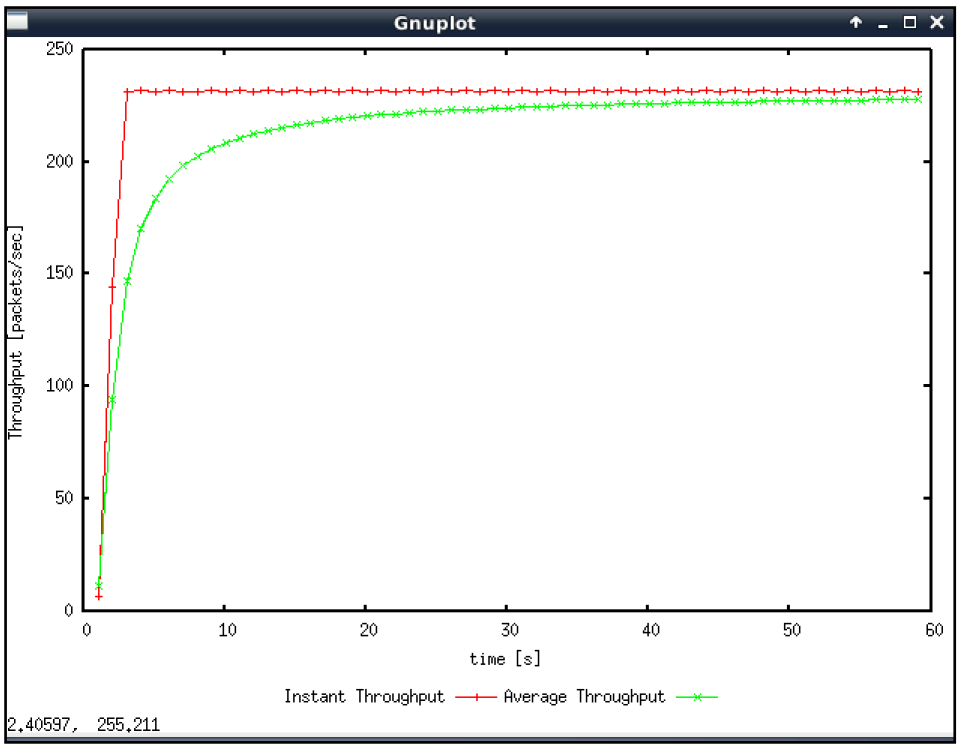
```
 1 1.1000000000000001 0  0.0 7.0 0 11.666666666666664
 2 2.1000000000000001 0  0.0 144.0 2 94.375
 3 3.1000000000000001 0  0.0 231.0 3 146.92307692307691
 4 4.0999999999999996 0  0.0 232.0 2 170.55555555555557
 5 5.0999999999999996 0  0.0 231.0 3 183.69565217391306
 6 6.0999999999999996 0  0.0 232.0 2 192.32142857142858
 7 7.0999999999999996 0  0.0 231.0 3 198.18181818181819
 8 8.0999999999999996 0  0.0 231.0 3 202.5
 9 9.0999999999999996 0  0.0 232.0 3 205.93023255813955
10 10.1 0  0.0 231.0 3 208.54166666666669
11 11.1 0  0.0 232.0 3 210.75471698113208
12 12.1 0  0.0 231.0 3 212.5
13 13.1 0  0.0 232.0 3 214.04761904761907
14 14.1 0  0.0 231.0 3 215.29411764705884
15 15.1 0  0.0 232.0 2 216.43835616438358
16 16.100000000000001 0  0.0 231.0 3 217.37179487179486
17 17.100000000000001 0  0.0 232.0 2 218.25301204819274
18 18.100000000000001 0  0.0 231.0 3 218.97727272727272
19 19.100000000000001 0  0.0 232.0 2 219.67741935483869
20 20.100000000000001 0  0.0 231.0 3 220.2551020408163
21 21.100000000000001 0  0.0 232.0 2 220.82524271844659
22 22.100000000000001 0  0.0 231.0 3 221.29629629629628
23 23.100000000000001 0  0.0 232.0 2 221.76991150442475
24 24.100000000000001 0  0.0 231.0 3 222.16101694915253
25 25.100000000000001 0  0.0 232.0 2 222.56097560975607
```

231 * 540 * 8 = **997920 bits/sec**. Which is very close to the link capacity(1Mbps).

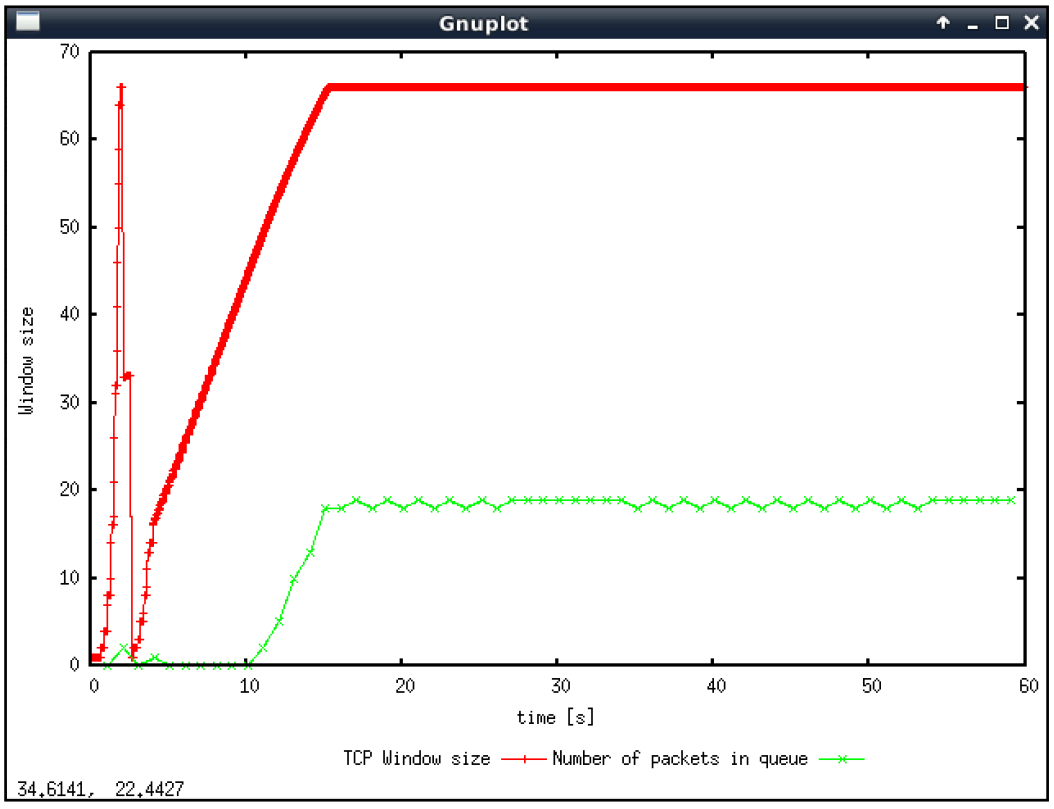
Window size = 50.

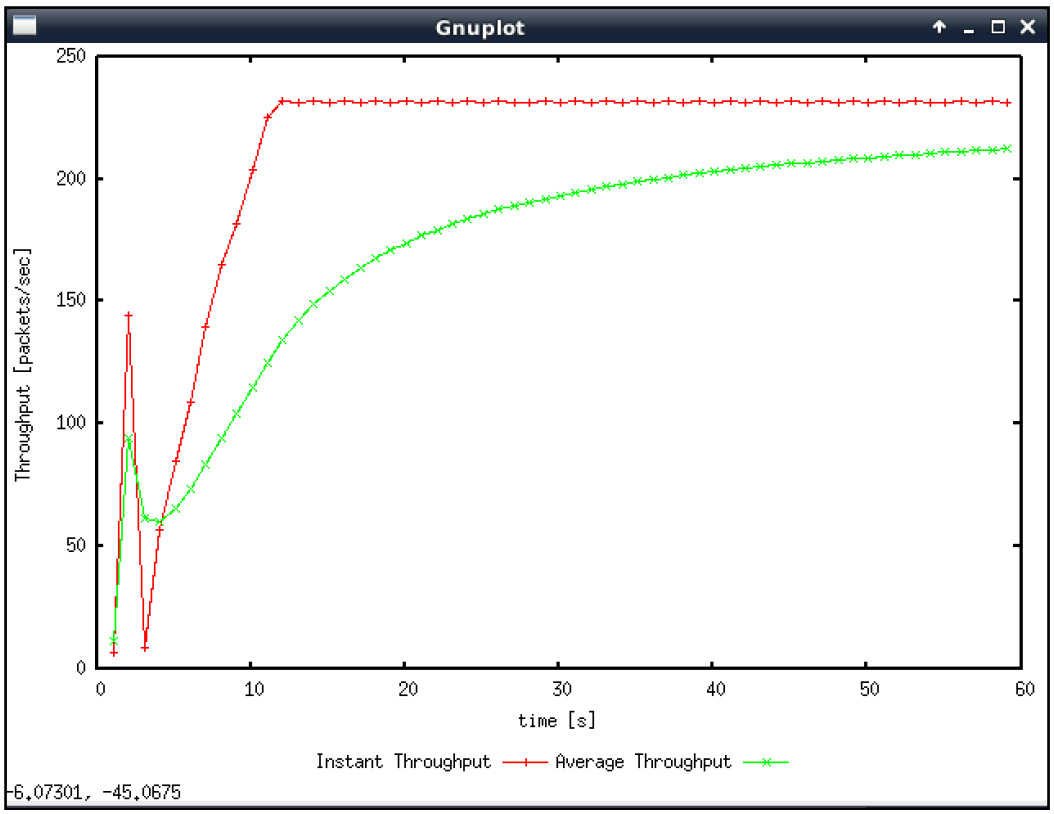


ZID: z5230310 Name: Tian Liu Date:17/07/2019



Window size = 66.





Q4:

In TCP Reno, the sender set the new congestion window size to half of the current size, and then increase linearly until next packet loss event happen. This means that most of the packet loss events are detected due to the triple duplicate ACKs rather than timeout. For timeout, the sender the windows back to 1 and then into slow start phase.

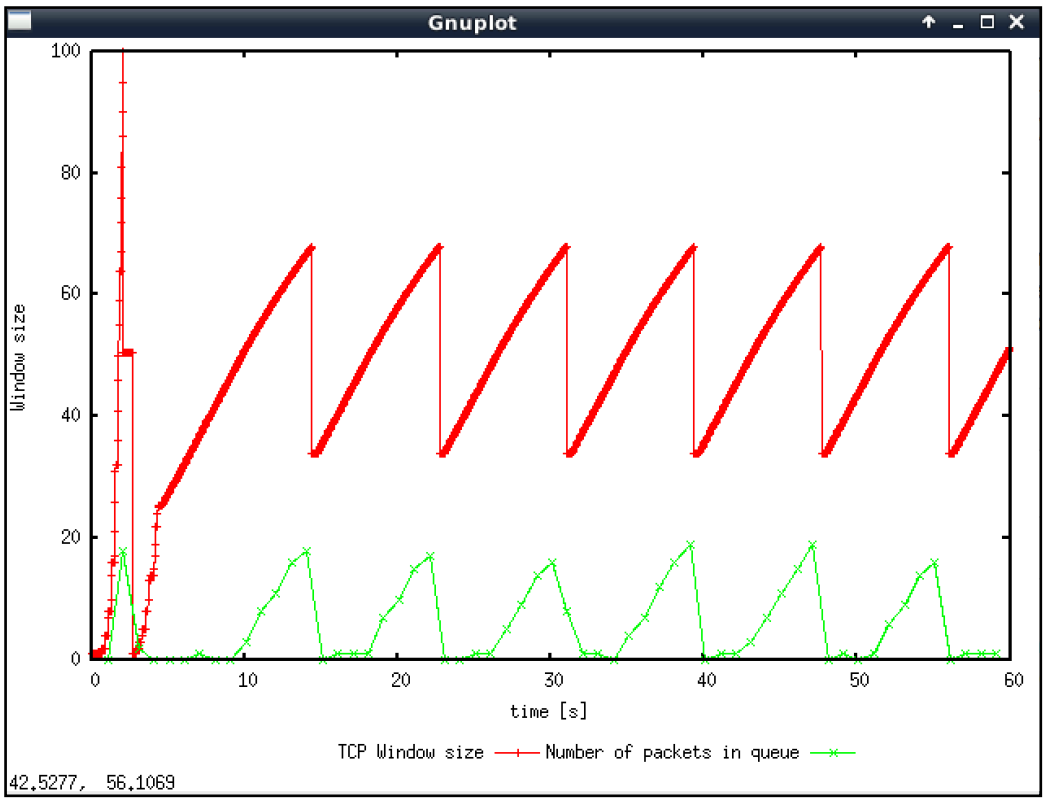
But TCP Tahoe, will set the window size to 1 following the slow start phase for both triple duplicate ACKs and timeout.

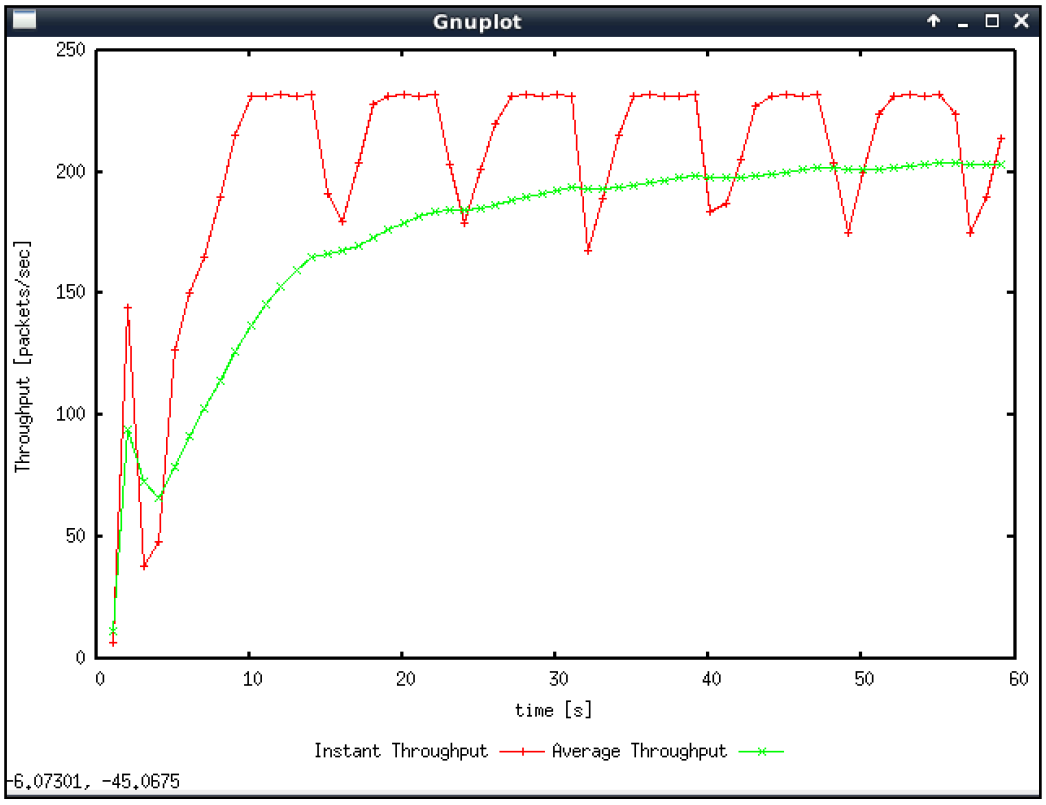
The picture of TCP Reno shows below:

TCP Reno throughput is **200 packets/sec**.

200 * 540 * 8 = **864000 bits/sec**. which is much higher than TCP Tahoe.

ZID: z5230310 Name: Tian Liu Date:17/07/2019



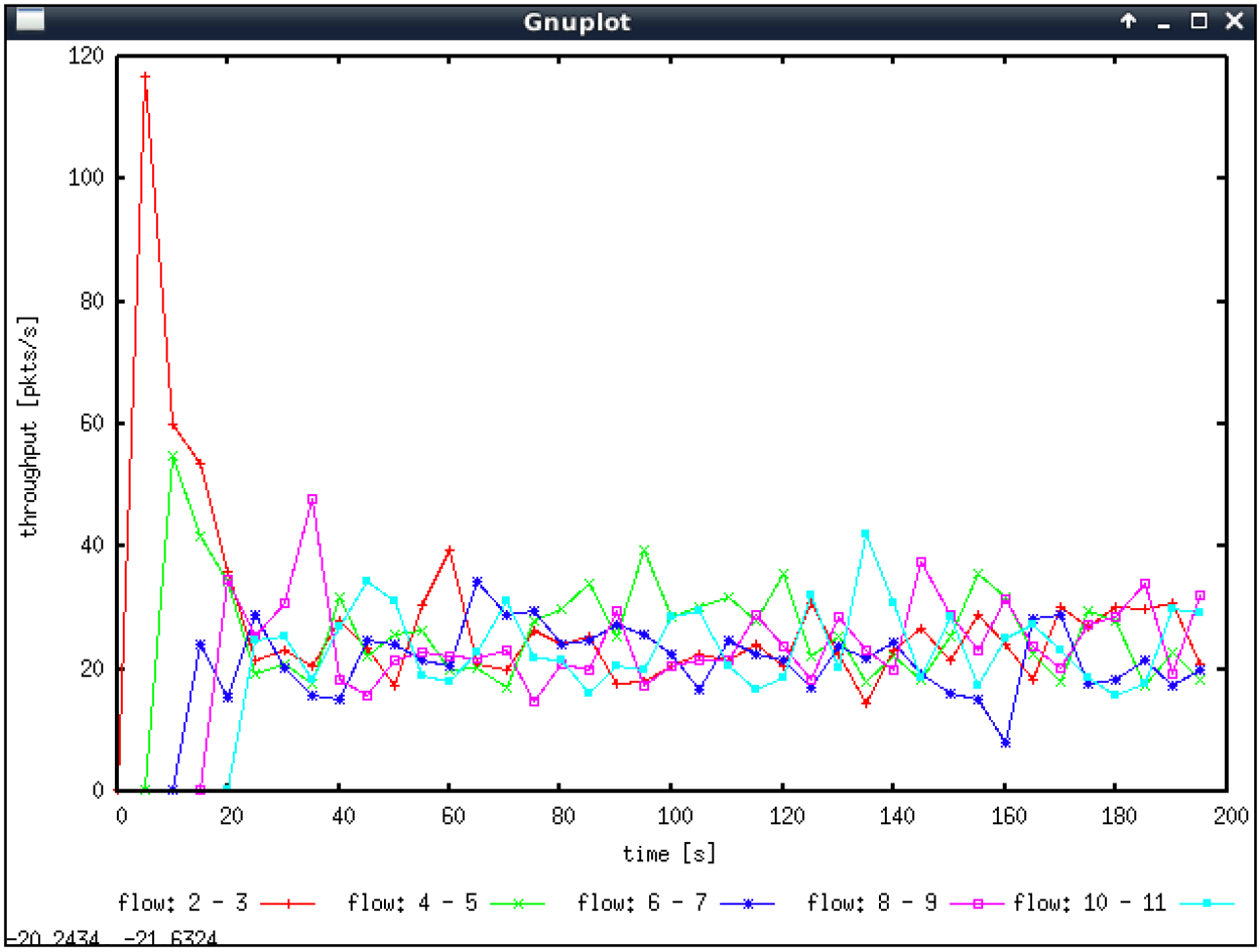


E2

Q1:

As the picture shows below:

We can conclude that each flow get an equal share of the capacity of the shared link. As we can find that when a new flow connects to the link, the pre-existing TCP flows will decrease. After all 5 TCP flows connected, the throughput are roughly similar. And the overall throughput is stable, which can be said that the TCP is fair.



Q2:

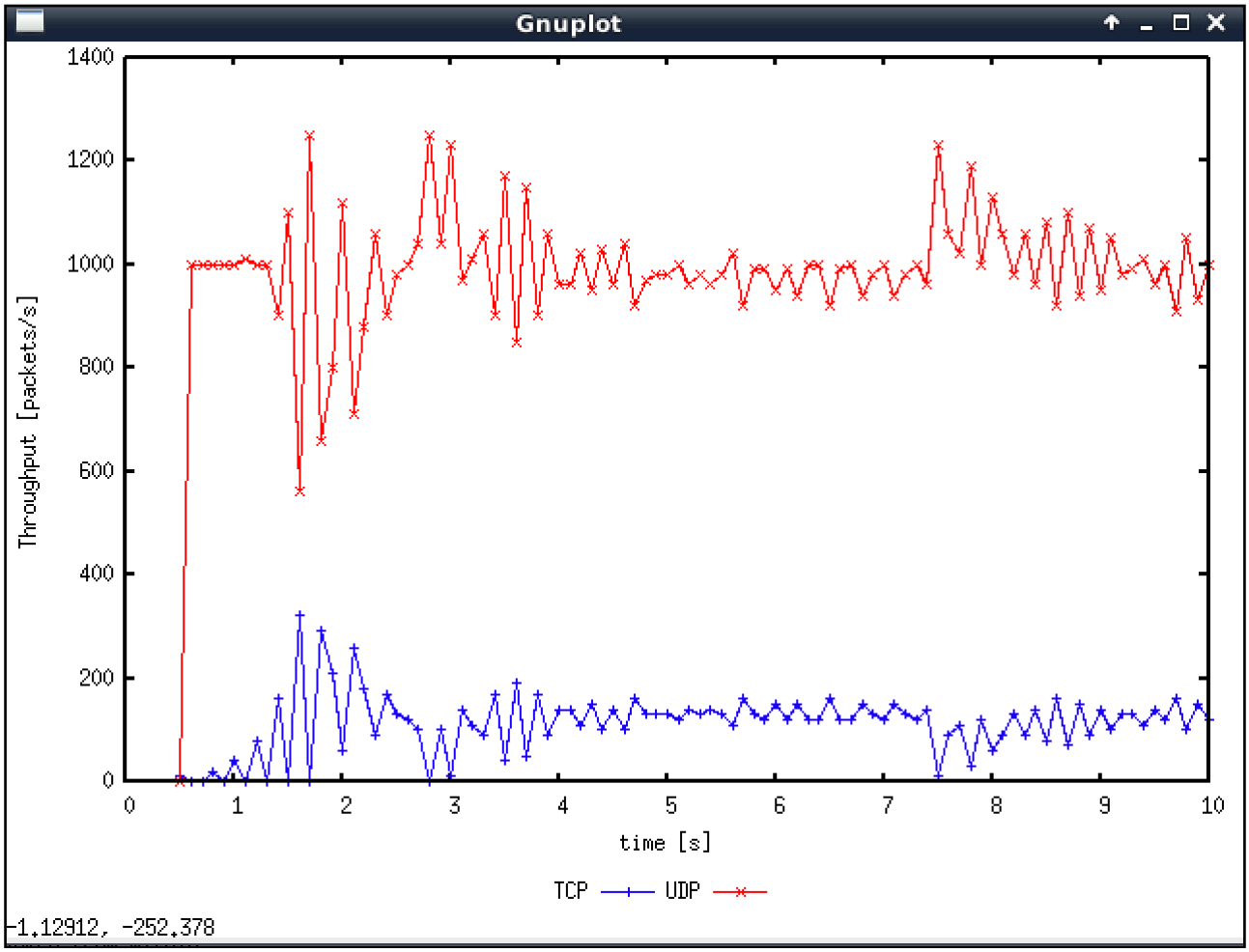
As the picture of Q1, the throughput of the pre-existing TCP flows will be shrink when a new flow is created. TCP guarantee each flow get an equal share of the capacity of the link. This is because new flows can be quickly increase its congestion windows size. And every TCP connections detected losses include timeout and duplicate ACKs, and change the congestion window size . This is fair because once a new flow is added, the share of all pre-existing flows need reduce.

E3

Q1:

The throughput of UDP will be much larger than TCP since UDP don't have any congestion control, and UDP flow will not reduce its transmission rate if there are any congestion.

As the graph shows below:



Q2:

The reason is that UDP doesn't implement any congestion control, so when packet loss occur, the transmission rate of UDP won't be influenced, but the transmission rate of TCP will reduced. So TCP flows will be forced to have a lower throughput due to more aggressive UDP flows.

Q3:

Advantage: use UDP to transfer file, it will keep transferring data at an unrestrained speed because UDP is not affected by the congestion in the network, which can reduce time for transferring files.

Disadvantage: UDP will add more burden on the network and it is not reliable. The file transfer protocol running over the UDP have to use reliable data transfer. But UDP does not provide this service.

If everybody started using UDP instead of TCP, everyone can have high transmission rate, the whole network will collapse and finally doesn't work.