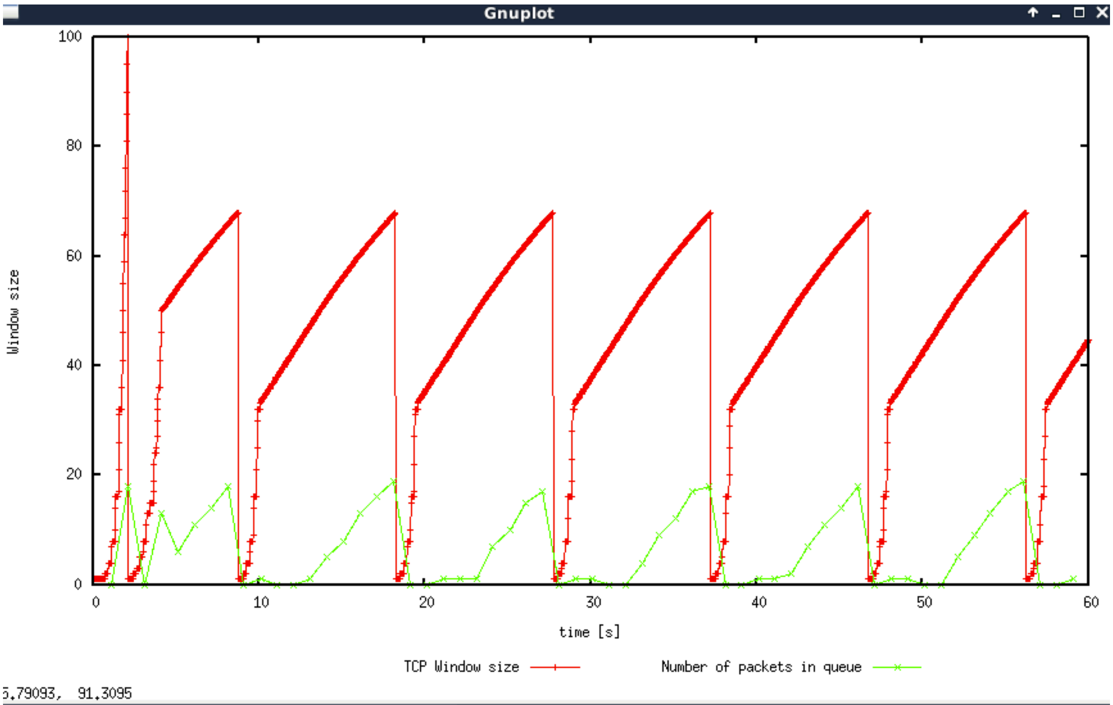
Exercise 1

Question 1. What is the maximum size of the congestion window that the TCP flow reaches in this case? What does the TCP flow do when the congestion window reaches this value? Why? What happens next? Include the graph in your submission report.



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

When the congestion window reaches this, the TCP flow reduce the congestion window size to 1 and threshold to 1/2 the size of the window.

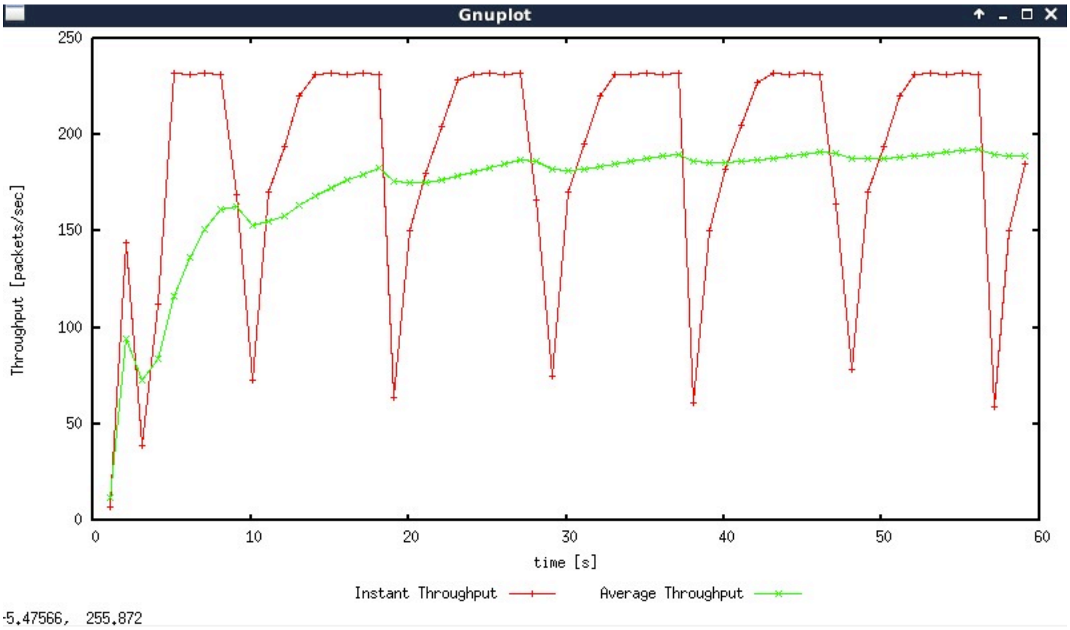
Because the queen is full and it may cause packet loss if continue increase the congestion window.

The connection enters slow start until it reaches the threshold in the next.

Question 2: From the simulation script we used, we know that the payload of the packet is 500 Bytes. Keep in mind that the size of the IP and TCP headers is 20 Bytes, each. Neglect any other headers. What is the average throughput of TCP in this case? (both in number of packets per second and bps)

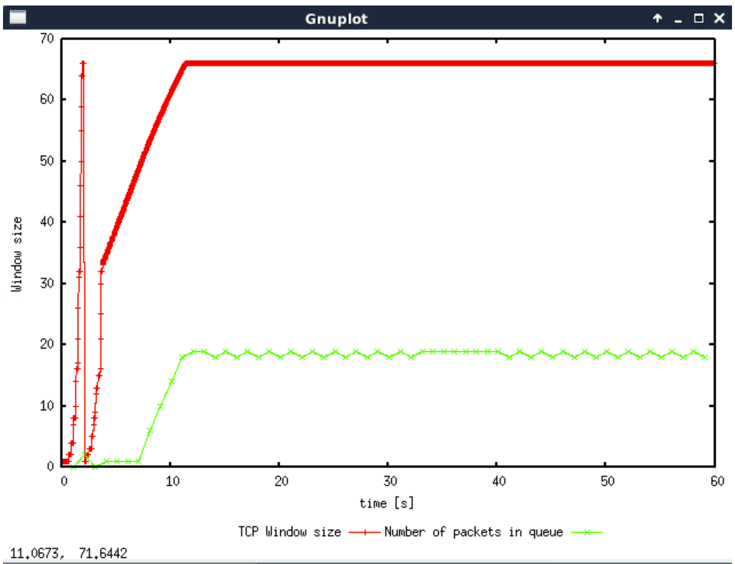
From the graph, we can see that is about 190 packets per second.

The rate at any data of TCP connection: $190 * (500 + 20 + 20) * 8 = 820.8Kbps$

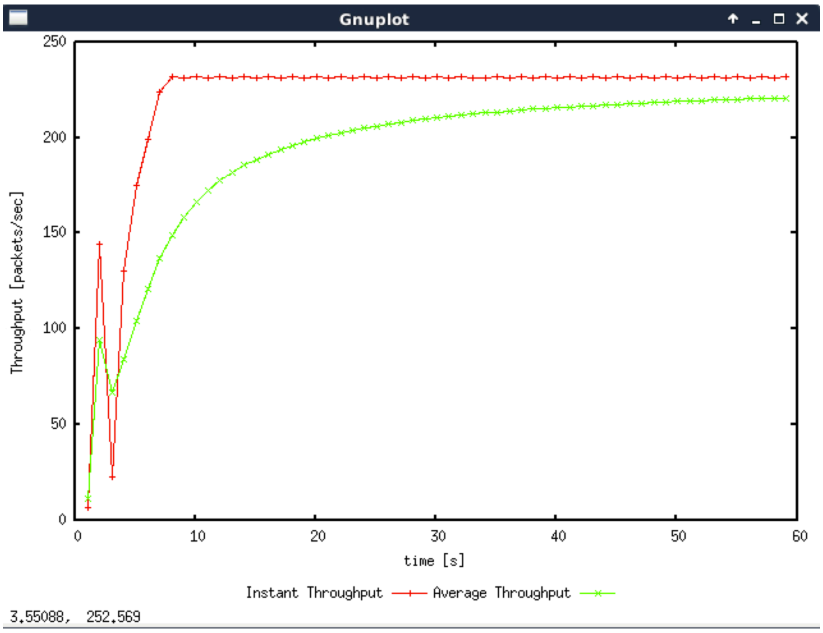


Question 3: Rerun the above script, each time with different values for the max congestion window size but the same RTT (i.e. 100ms). How does TCP respond to the variation of this parameter? Find the value of the maximum congestion window at which TCP stops oscillating (i.e., does not move up and down again) to reach a stable behaviour. What is the average throughput (in packets and bps) at this point? How does the actual average throughput compare to the link capacity (1Mbps)?

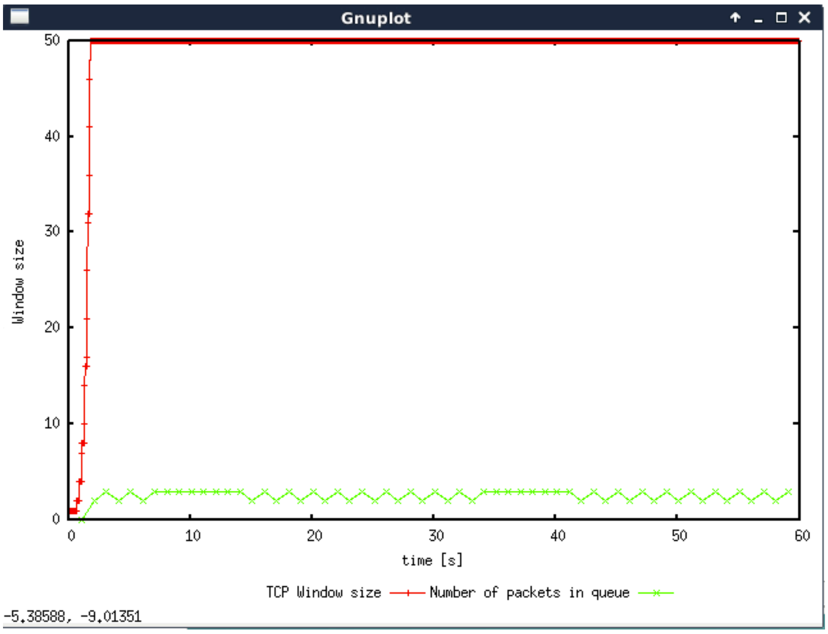
After some attempts, we can see that:
If 50<= initial maximum congestion window size <=66, the TCP will stop oscillating after the first slow start phase and the congestion window will be 1/2 of that size, the queue will never get full can packet will not be lost.
If initial maximum congestion window size<50, TCP will get balance after the slow start phase.



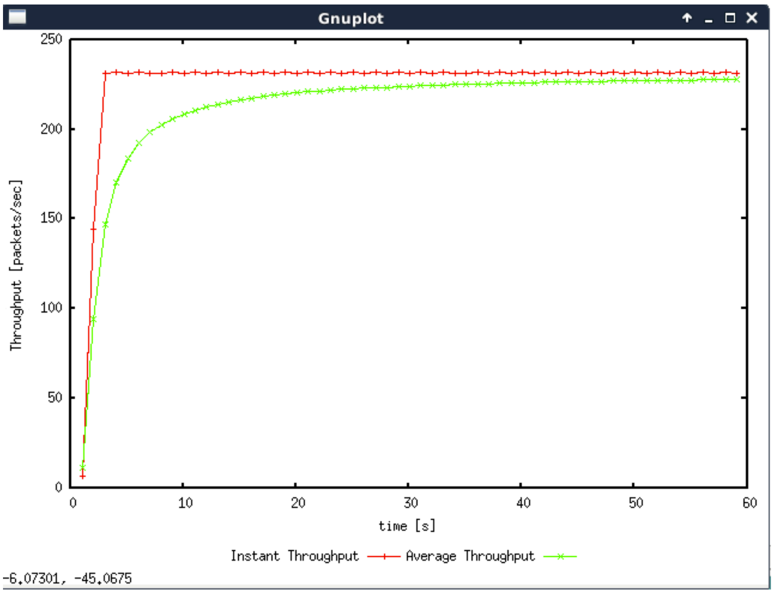
Window changing when the congestion window size = 66



Throughput when the congestion window size = 66



Window changing when the congestion window size = 50

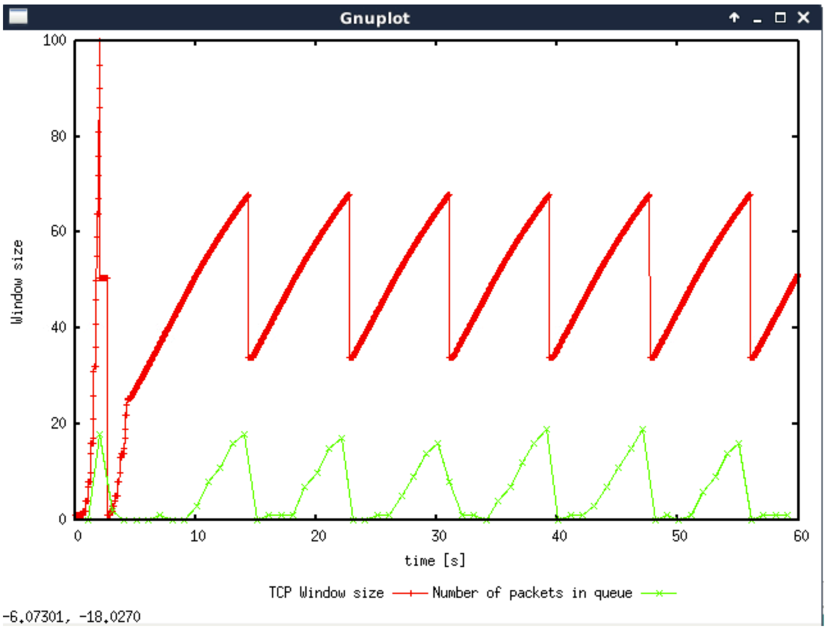


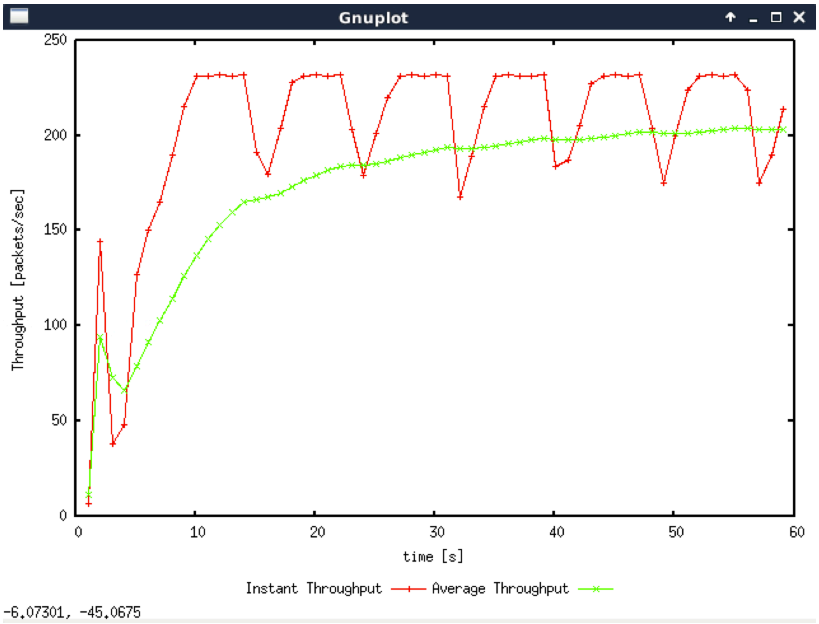
Throughput when the congestion window size = 50

As we can see, the average packet throughput is around 225 packets per second.

The average throughput is $225 * 500 * 8 = 900\ Kbps$, which is 90% to the link capacity , it's almost full.

Question 4: Repeat the steps outlined in Question 1 and 2 (NOT Question 3) but for TCP Reno. Compare the graphs for the two implementations and explain the differences. (Hint: compare the number of times the congestion window goes back to zero in each case). How does the average throughput differ in both implementations?



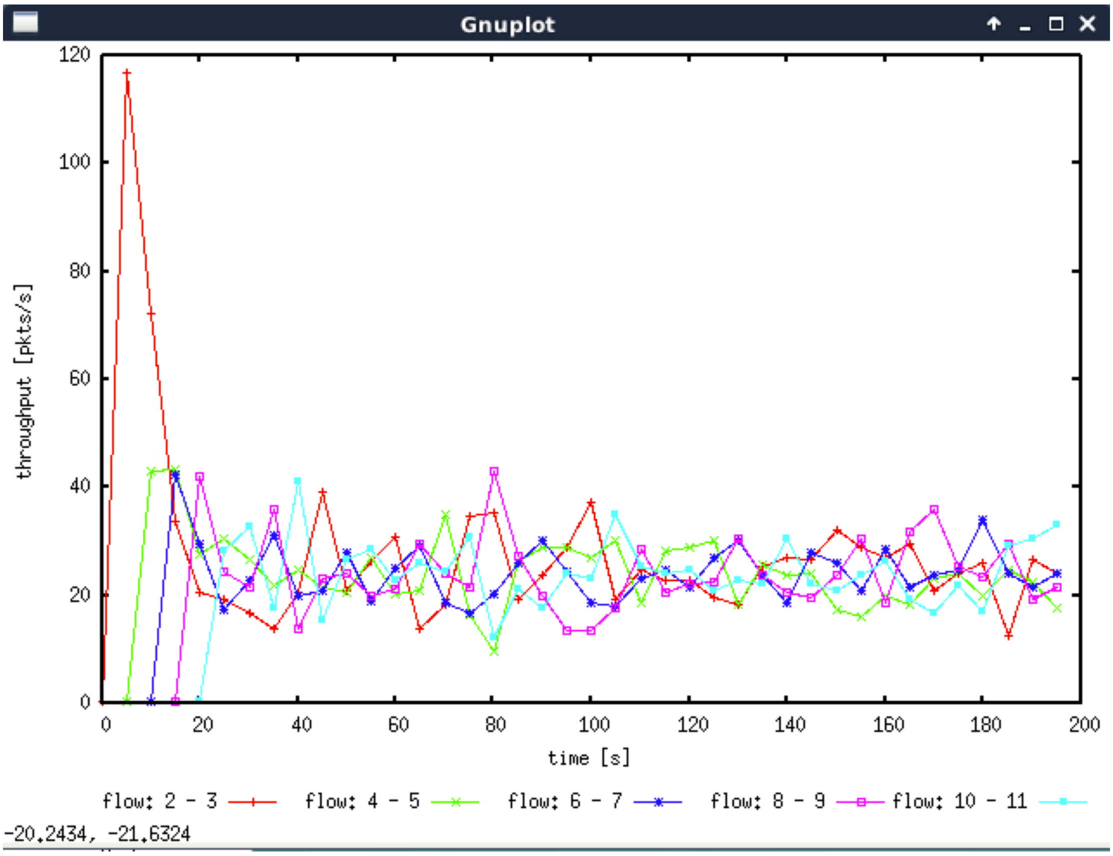


As we can see from the graphs:

TCP reno doesn’t enter slow start phase for all cases. It will cut the congestion window to half and go on increasing linearly.However, TCP tahoe always cut the congestion window size to 1 for any cases.

And the throughput of TCL reno is around 200 which is a bit higher than TCP tahoe, because TCP reno doesn’t need so much slow start phases.

Exercise 2



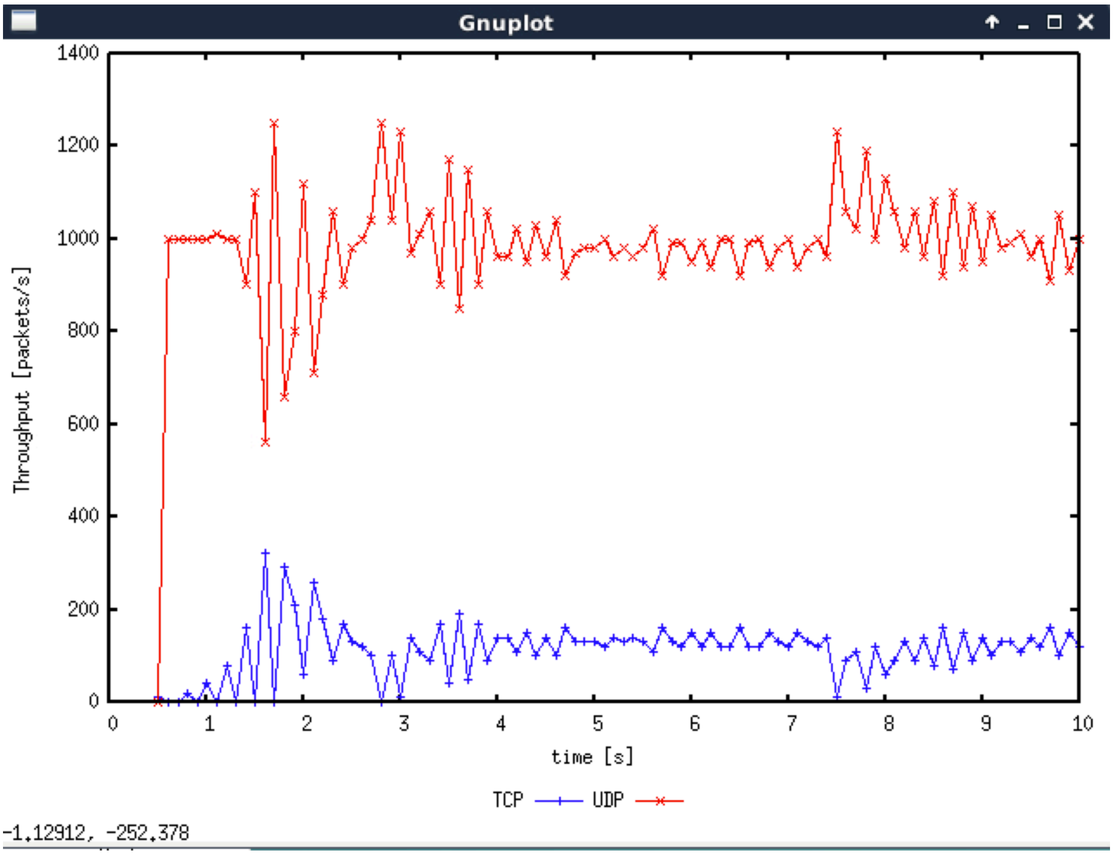
Question 1: Does each flow get an equal share of the capacity of the common link (i.e., is TCP fair) ? Explain which observations lead you to this conclusion.

Yes, after all of the flows connected to the common link, the throughput for all connection is nearly similar, although there are still some fluctuations.
This is the result of TCP AIMD congestion control algorithm. When several flow share a common link, AIMD achieve long-term fairness by changing the window size.

Question 2. What happens to the throughput of the pre-existing TCP flows when a new flow is created? Explain the mechanisms of TCP which contribute to this behaviour. Argue about whether you consider this behaviour to be fair or unfair.

As we can see, when a new flow is created, the throughput of other existing TCP flows are reduced because of the slow start phase of the new flow. After that, all flows will detect losses and they need to adapt congestion window size to avoid overwhelming the link.
I think this is fair. Because when a new flow join the common link, the other flows should reduce bandwidth for the new one. And after some adapts, all the flows will get an equal share of the capacity of the common link.

Exercise 3



Question 1: How do you expect the TCP flow and the UDP flow to behave if the capacity of the link is 5 Mbps ?

I think UDP will get much more throughtput than TCP because of the lack of congestion control.
TCP will reduce its its transmission rate if there is congestion but UDP will not.

Question 2: Why does one flow achieve higher throughput than the other? Try to explain what mechanisms force the two flows to stabilise to the observed throughput.

TCP will reduce its its transmission rate if there is congestion but UDP will not because TCP have congestion control.
So, UDP will transmits packets at a higher and constant rate and TCP will reduce its transmission rate when it detects congestion in the network.

Question 3: List the advantages and the disadvantages of using UDP instead of TCP for a file transfer, when our connection has to compete with other flows for the same link. What would happen if everybody started using UDP instead of TCP for that same reason?

Advantages: The transmission rate of UDP is much faster than using TCP, and the rate will not be restrained by congestion in the network except the transmitting rate is higher than link capacity.

Disadvantage: We need a reliable data transfer because UDP is an unreliable data transfer protocol.

If everybody started using UDP instead of TCP for that same reason, the network will be more and more serious congestion and everybody will lose their packets. So, whether the data could be transmitted without a loss will depend on luck.