HY435 Lab 1

Xristos Papastamos 4569 , Panagiotis Karouzakis 3599

1. Exercise 0

   We assigned the following IP addresses to the computers using the command: $sudo ip addr change <new_address> broadcast 255.255.255.255 0 dev vlan1000@eth0

Computer	IP address	node
comm1	192.168.4.11/24	A
comm2	192.168.4.12/24	B
comm9	192.168.4.19/24	C
comm10	192.168.4.20/24	D

2. Exercise 1

   (We used the argument -J to convert the output to JSON and redirected them to a file so that we could plot them after)

   i)

   a)for TCP traffic on the server we run: $iperf3 -s .On the client we run: $iperf3 -c 192.168.12

   b) for UDP traffic we run : $iperf3 -c 192.168.4.11 -u -b 0
   to determine the max bandwidth which was 1MB

   UDP 1 percent :

   for the server:

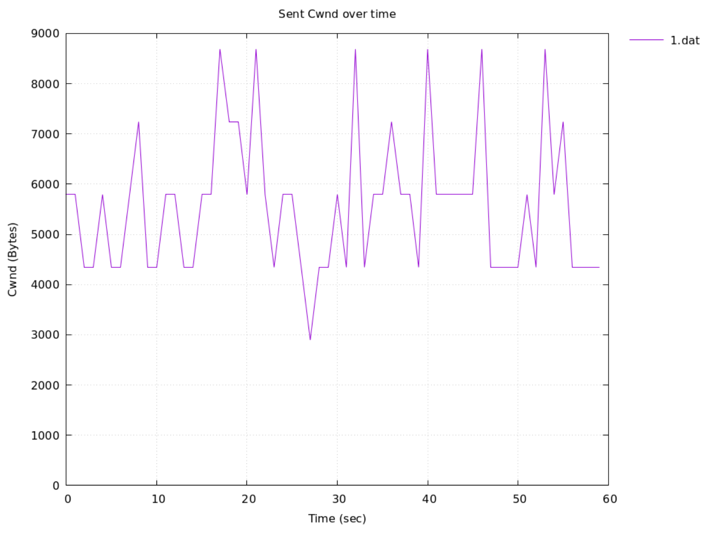
```
$iperf3 -s -V -B 192.168.4.11
```

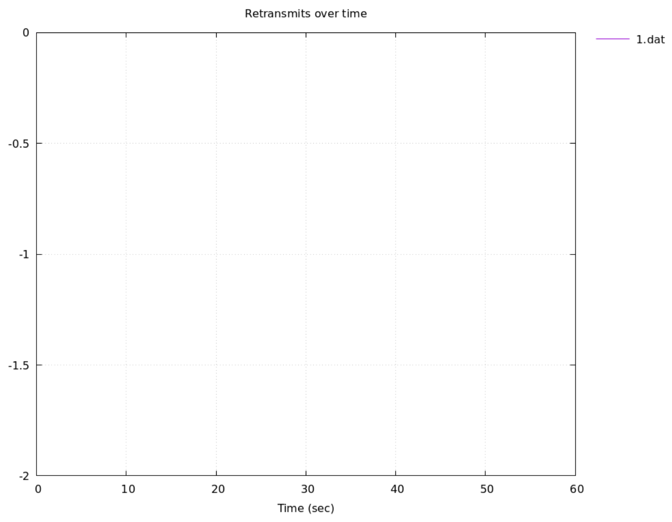
for client:

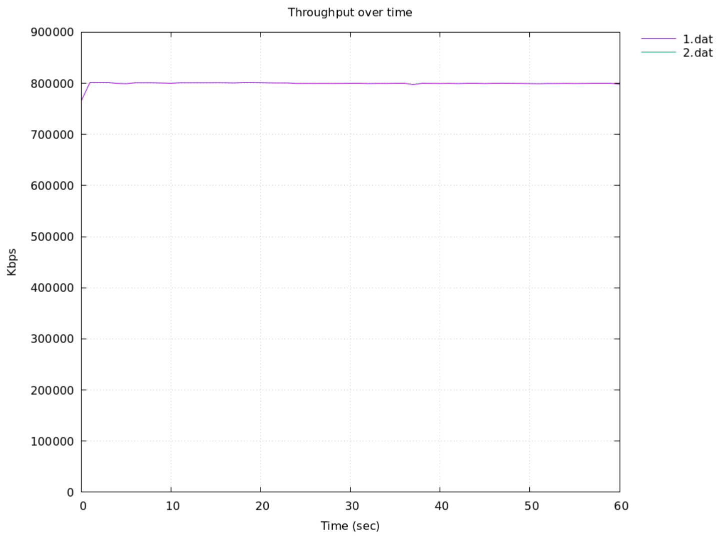
```
iperf3 -u -b 100Kb -t 60
```

All the jitter plots are in ms
we obtained the following results

TCP:

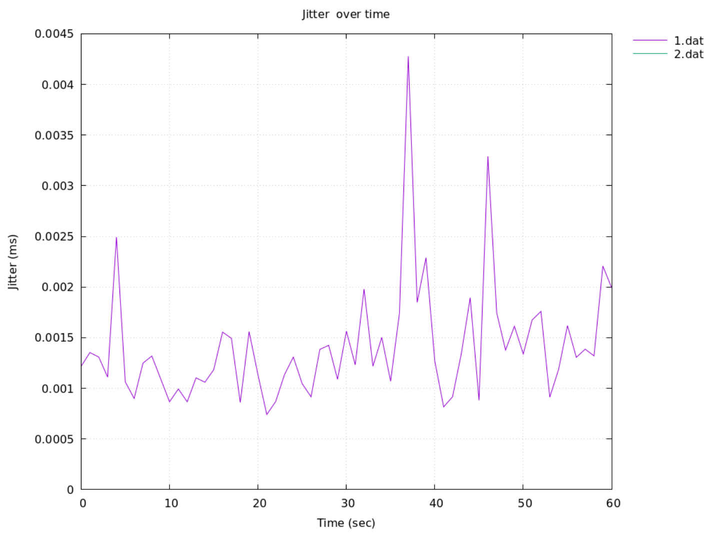


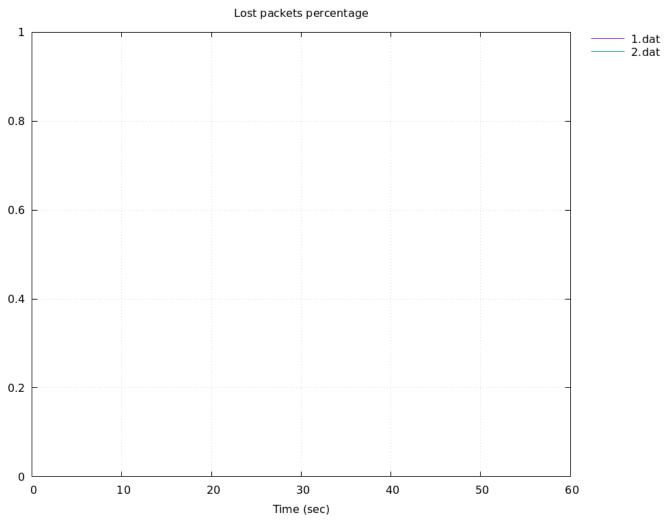


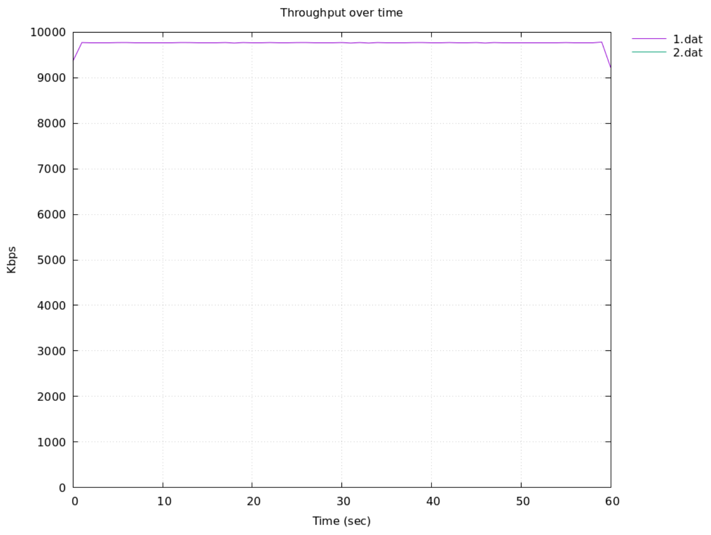


Here the congestion window adapts to the best send window based on the retransmissions detected by the protocol. There were no packet losses detected on this transfer. Having no retransmissions lead to a static throughput, with small variations caused by the adapting congestion window

UDP 1%:

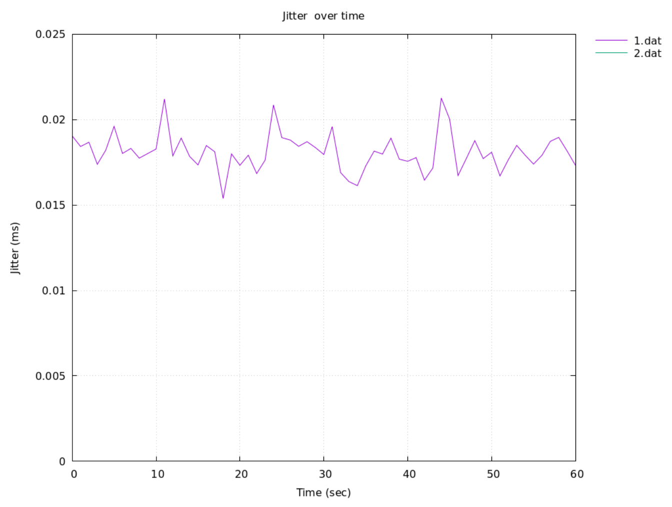


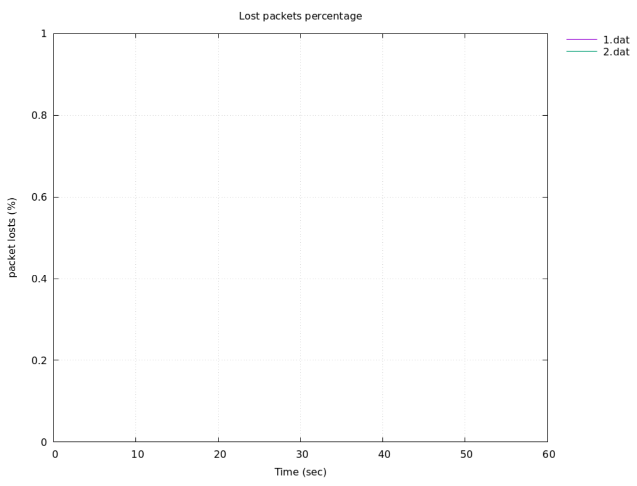


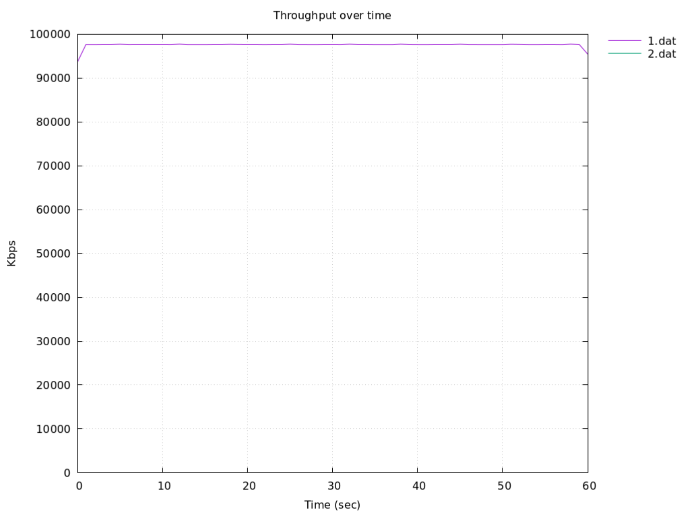


Jitter cannot be linear because the time variation between the sender and the receiver is not the same because of many variables such as link connections, network congestion, program execution. We didn't have packet's loss in this example because we don't have any interference between node A and node B.The throughput is stable at around 10Mbps

UDP 10%:

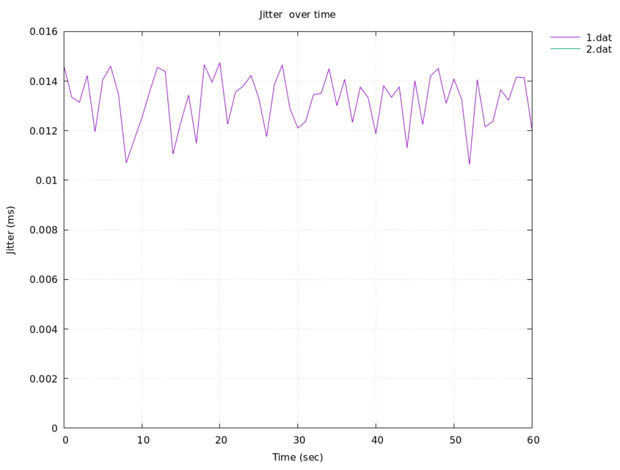


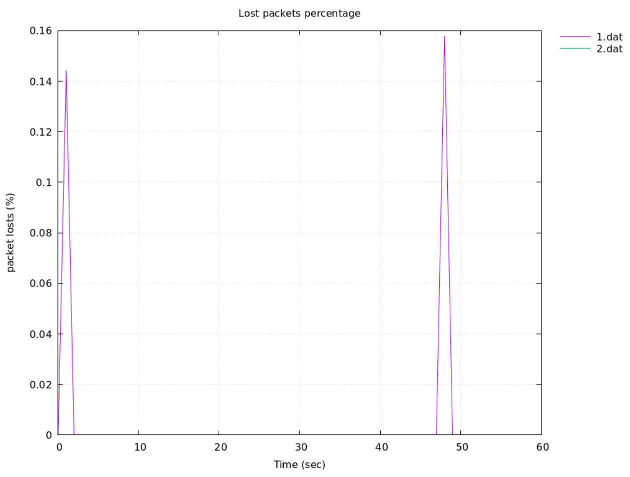


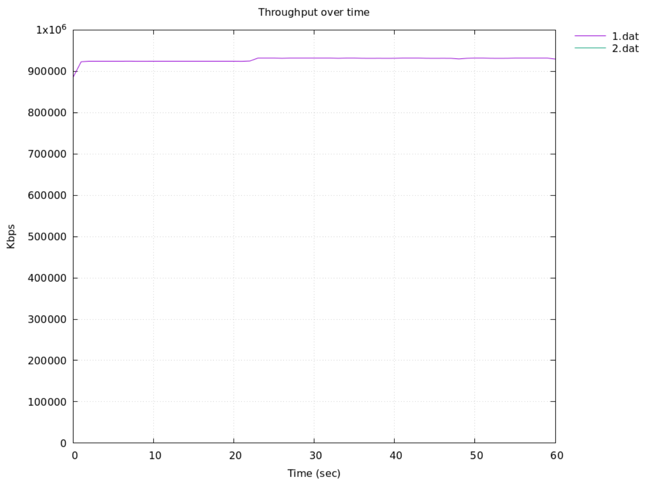


Jitter cannot be linear because the time variation between the sender and the receiver is not the same because of many variables such as link connections, network congestion, program execution. We didn't have packet's loss in this example because we don't have any interference between node A and node B. The throughput is stable at around 100Mbps.

UDP 97%:



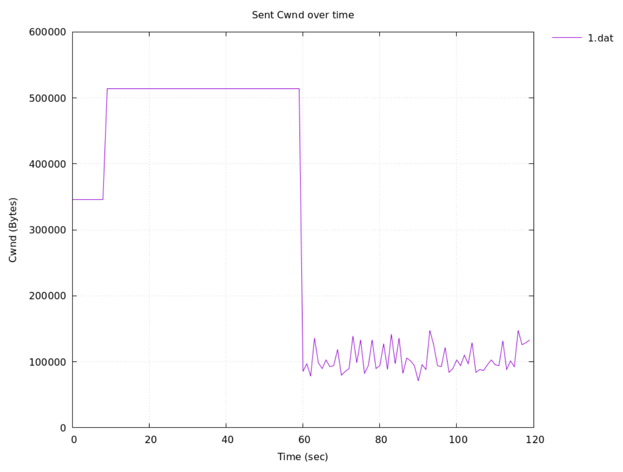


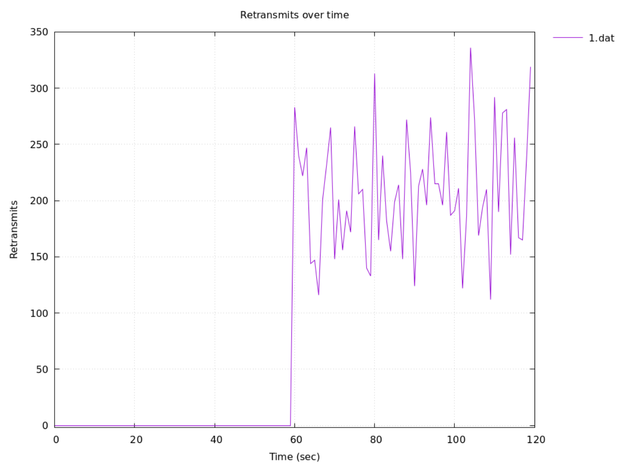


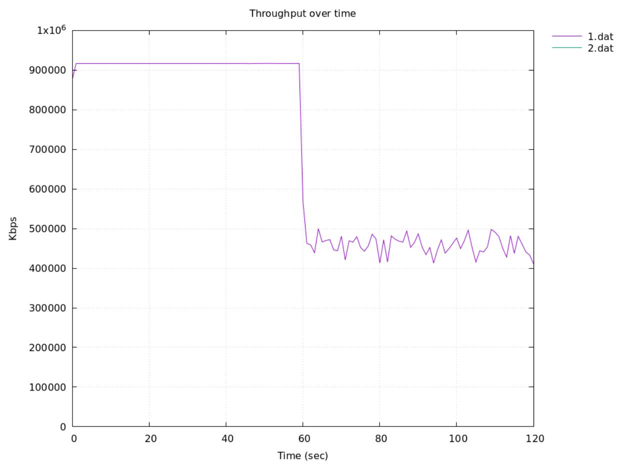
The jitter here is higher because we send data almost with the max bandwidth available so the time variation between the sender and the receiver is gonna be larger. We didn’t have packet loss in this example because we don’t have any interference between node A and node B. The throughput is higher here almost to the max bandwidth 1000MBs .

ii) Single Server multiple flow scenario:

TCP:

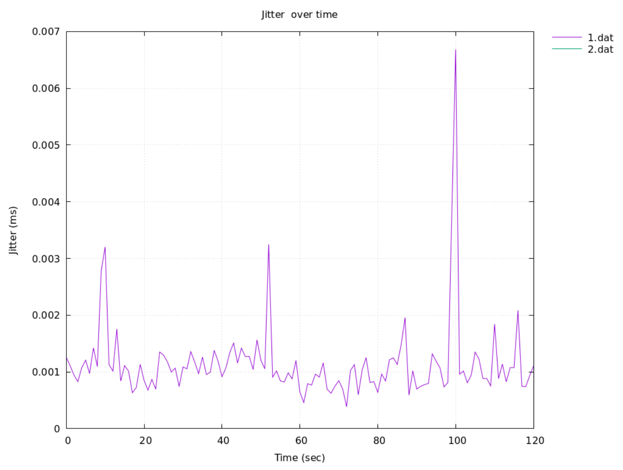


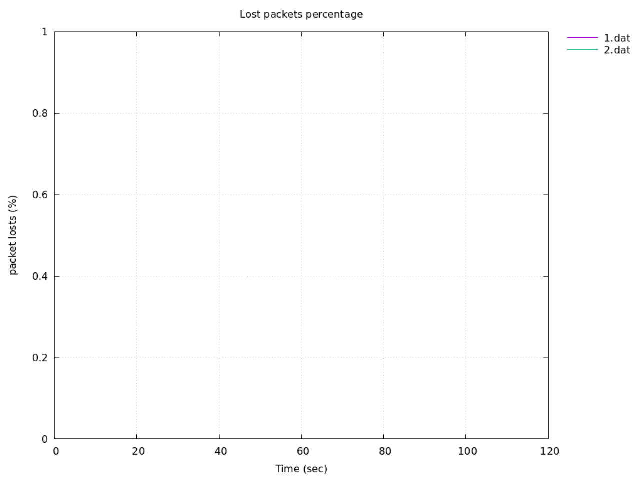


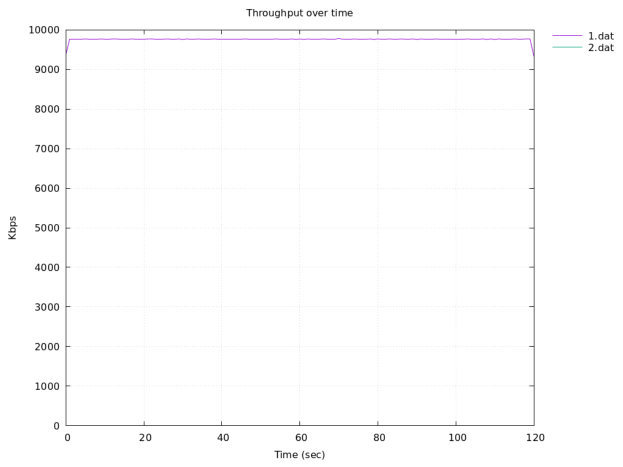


By executing the TCP test, we can observe a sudden loss in throughput when the second flow starts (60sec) due to congestion on the server that results in large amount of retransmissions and the drop of the congestion window

UDP 1%:

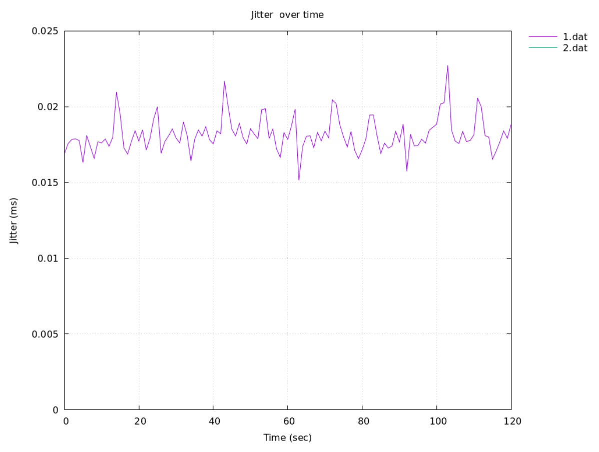


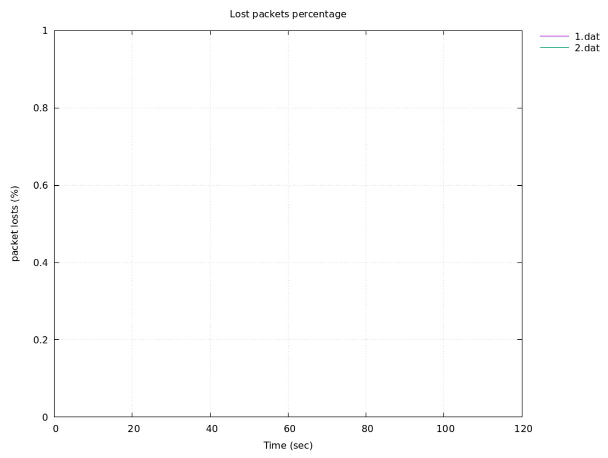


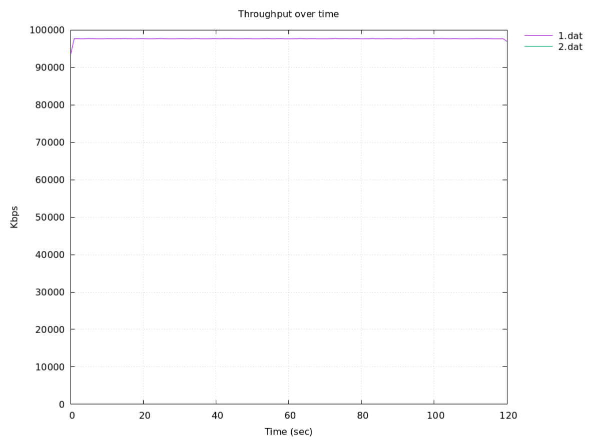


When using 1% of the UDP flow we can't see any congestion because the two flows from the two nodes can easily share the switches bandwidth, consuming a total of 2%. So, as expected the plots look like a normal UDP flow with a bandwidth of ~10Mbps

UDP 10%:

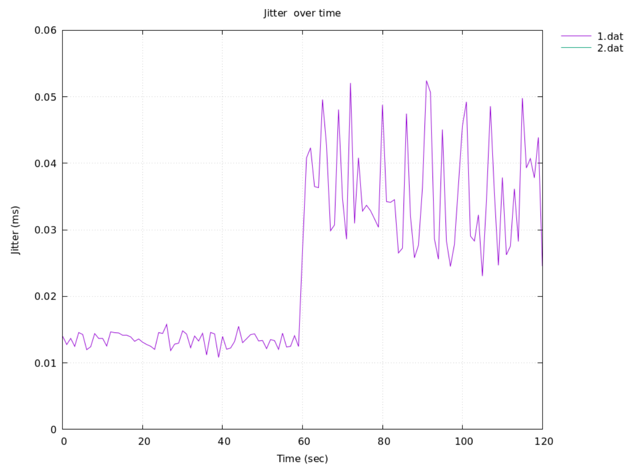


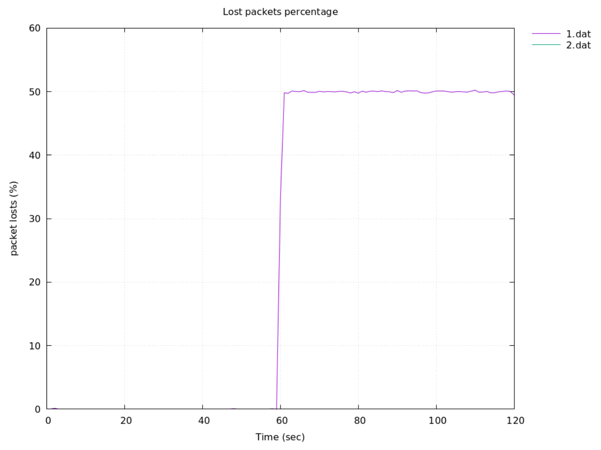


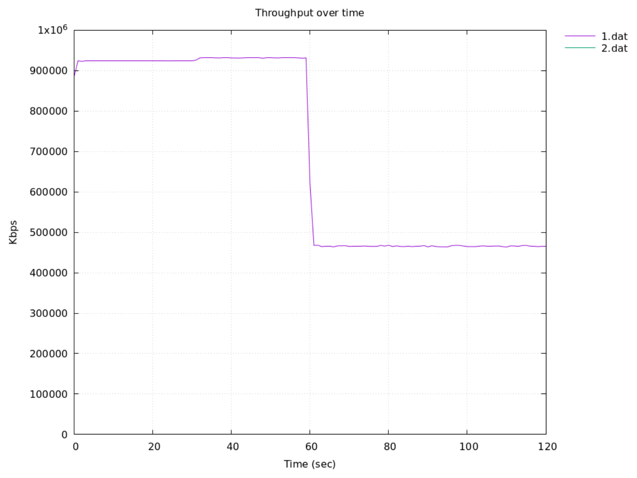


Like the 1% UDP flow, in the 10% UDP flow test everything runs normal. With a sum of 20% of the switche's bandwidth, the network does not struggle to handle the two UDP flows

UDP 97%:







At 97% UDP flow, we can clearly see a drop of the bandwidth after the second flow starts. This happens because each host tries to get 97% of the total bandwidth which cannot be accomplished by the switch. Therefore each host gets ~50% of the bandwidth which causes half of the packets to be lost and a sudden increase on the packet jitter.

c) Older versions of iperf did not provide the bandwidth option for TCP flows because the TCP implements flow control and congestion control with the congestion window and the sliding window so if a user inputs a large bandwidth to be transmitted through a TCP connection the TCP cannot violate it's protocol thus the bandwidth option for large bandwidths is redundant.

d)During the first scenario (Single flow) we didn't observe any retransmissions because the network did not have any congestion, as a result the congestion window stays stable. On the second scenario (Double flow) after the second flow starts (sec 60) we can observe a lot of retransmissions because of the congested network resulting in a big drop at the congestion window as a result of the congestion avoidance of the TCP Protocol

e) In 2.1 we didn't have any packet losses in the UDP experiments thus we cannot observe anything between packet loss and jitter

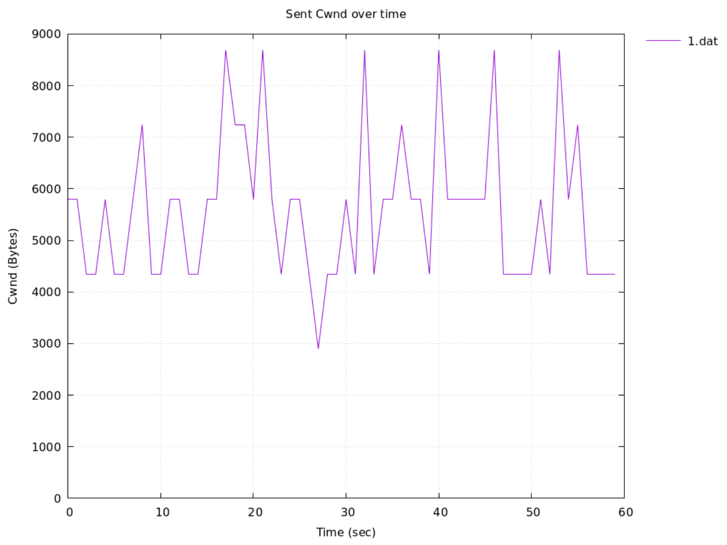
In 2.2 we have packet losses at around 50% only when using 97% of the maximum bandwidth. This is because we have 2 nodes using both 97% of the maximum bandwidth thus only the 50% of the 2 flows can be transmitted. When the packet loss goes to 50% the jitter goes higher because the time variation between the sender and the receiver is higher due to the congested network. This happens because the packets wait in the switches queue for more time (before getting forwarded to the destination node) than they normally would in a non-congested network.

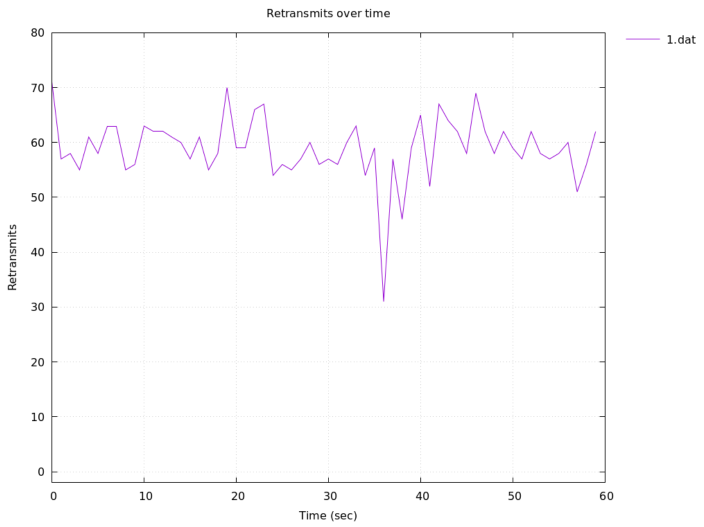
When 2 nodes transmit segments to a single server instead of 1 node the server becomes congested and so the speed in which he responds back and processes segments is slower, especially when the output is close to the max bandwidth

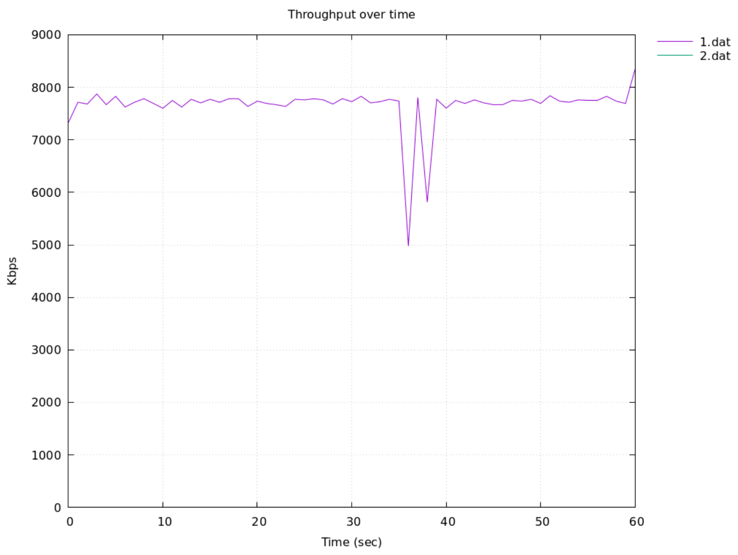
3. Exercise 2

In this topology we use a hub, which operates in the physical layer. This means that any incoming packets to any of its interfaces will get transmitted to every interface. That can cause traffic congestion or packet collisions which can lower the throughput (as we can see below)

Single Client -> Single Server:
TCP:

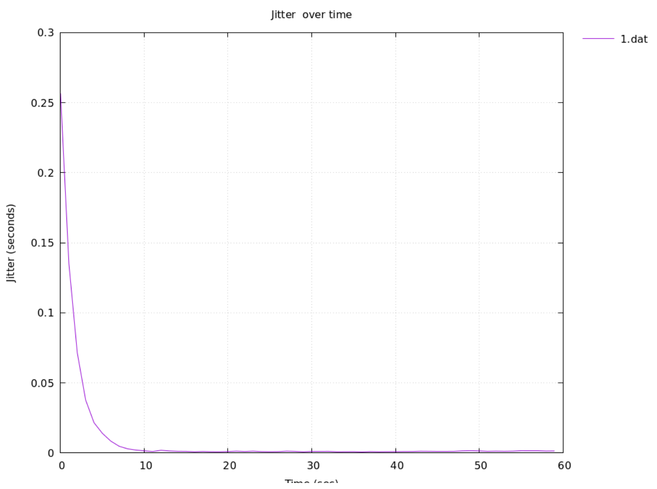


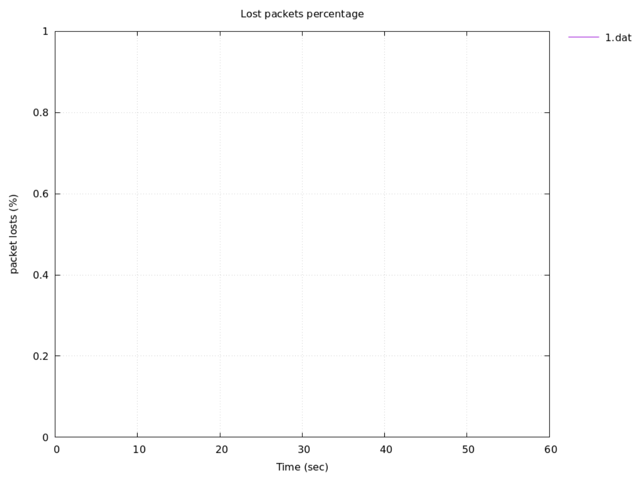


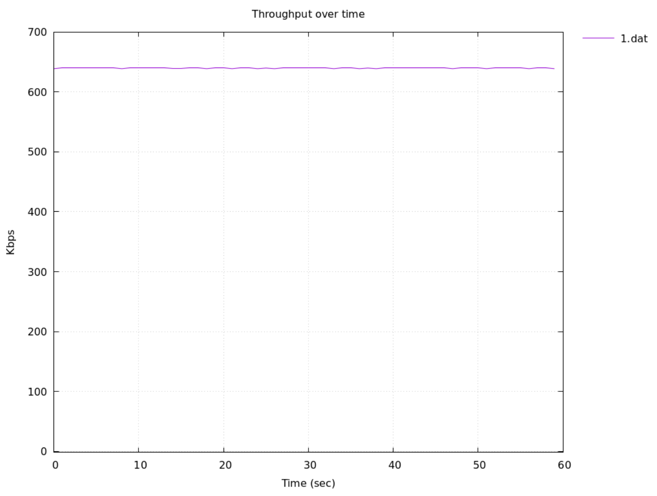


When using the TCP protocol, we can observe a natural transmission with moderate retransmissions, a congestion window increasing and decreasing and a pretty stable throughput with some exceptions between 30 and 40 seconds

UDP 1%:

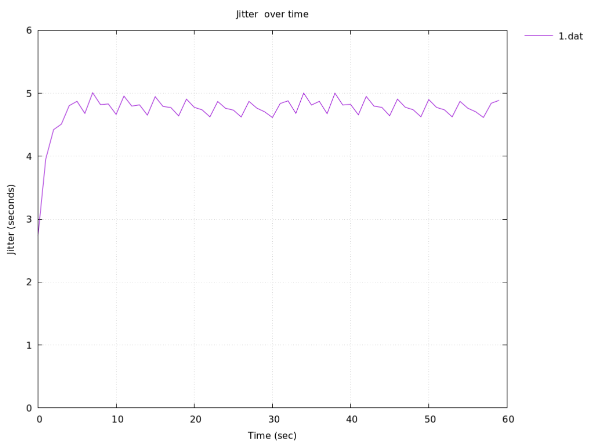


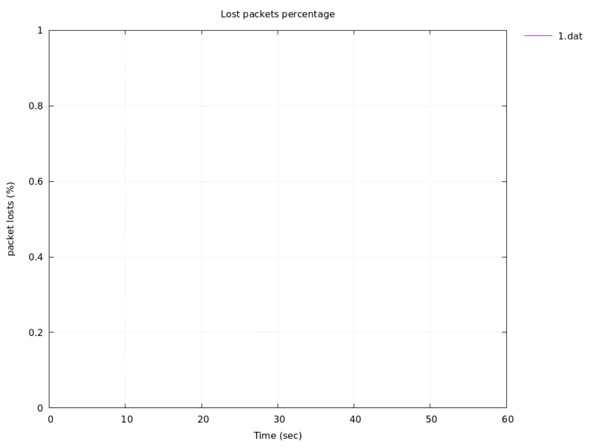


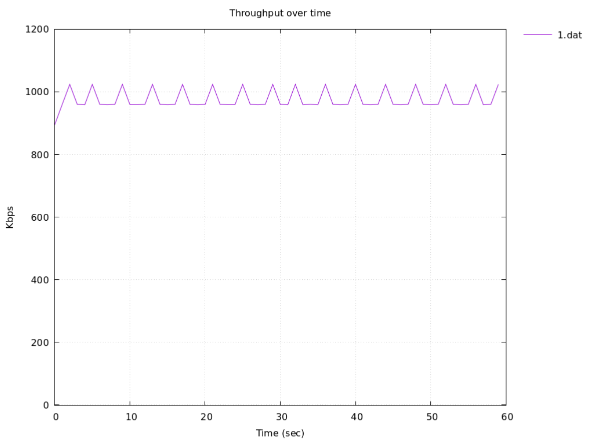


At 1% of the maximum UDP bandwidth the transmission is smooth, without retransmissions and with a stable bandwidth of ~650Kbps. The exponential loss of the jitter happens for a lot of reasons. One reason that the jitter is higher at the start is because the server's OS has a context switch when a Segment is transmitted from the client thus the big delay that later vanishes.

UDP 10%:

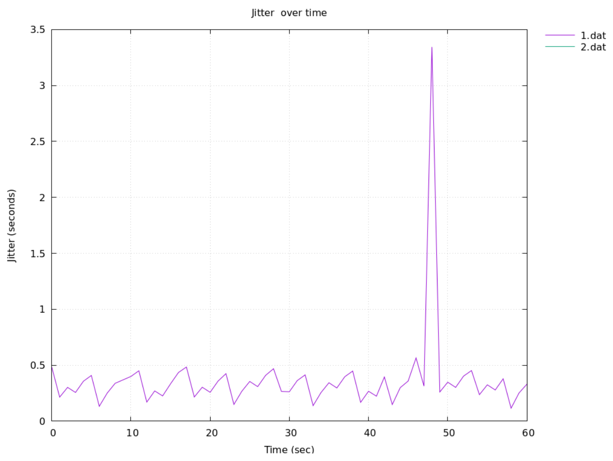


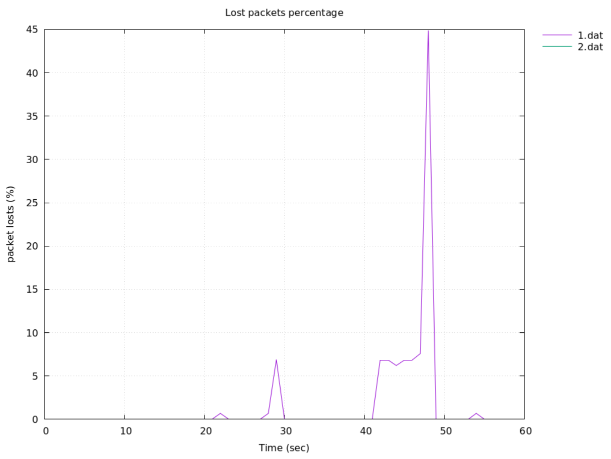


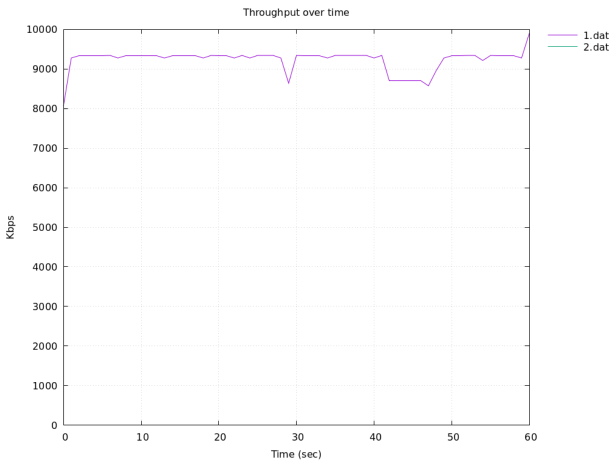


At 10% UDP bandwidth, a normal transmission is observed, with low jitter and a stable throughput. Jitter is lower at the start because throughput is lower as well.

UDP 97%:



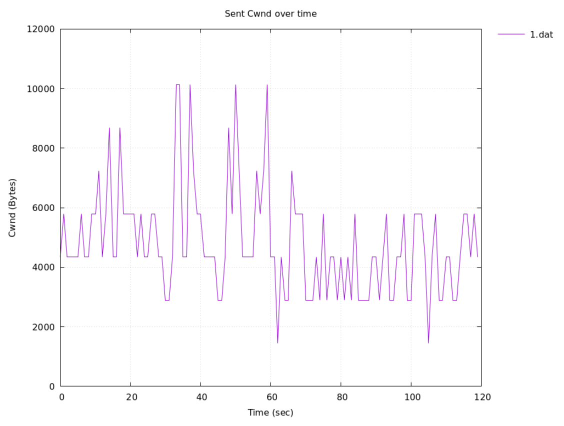


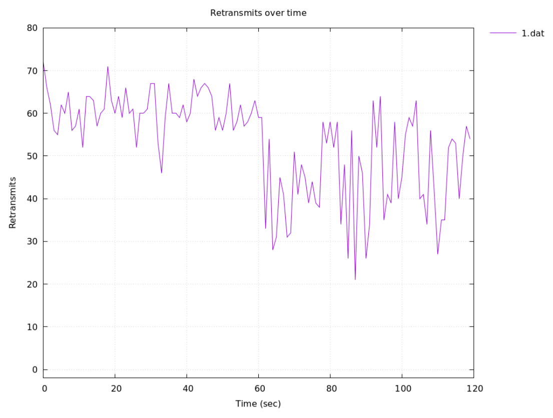


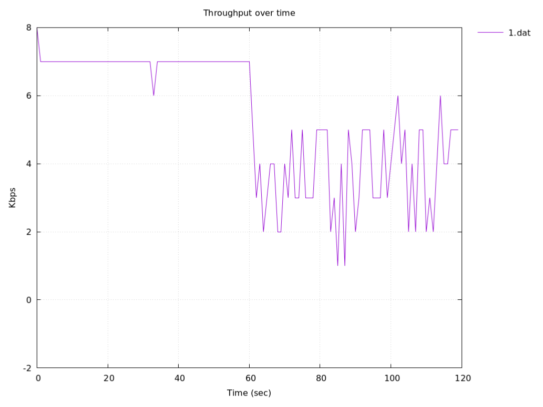
At 97% UDP traffic we can see a big packet loss between sec 40 and 50 that caused an increase to jitter and a decrease to the throughput. We cannot be sure why this happened, one thing that could possibly cause this is the receive buffer of the receiver filling up with the fast incoming traffic.

Two Client -> Single Server:

TCP:

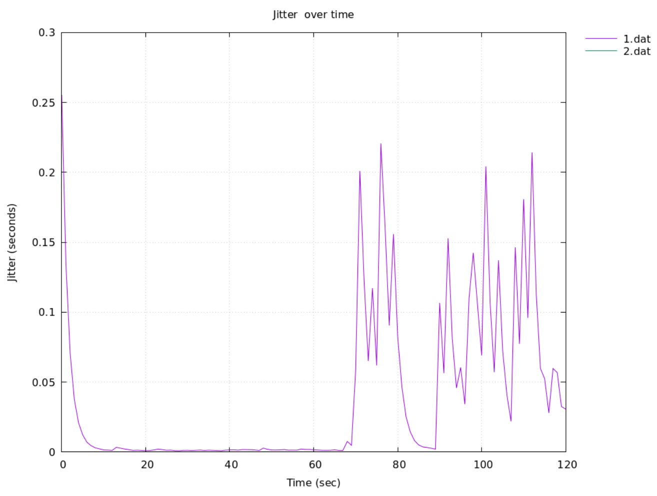


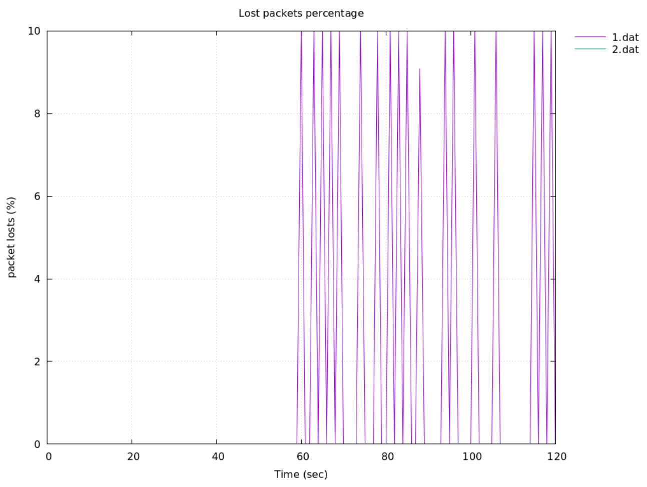


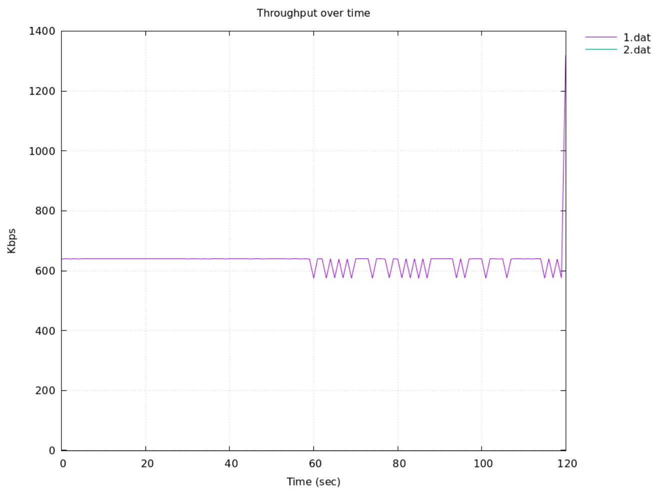


As soon as the second flow starts (>60sec) we can see the throughput cuts to about half in order to satisfy both clients requesting the medium. This only happens when a hub is used, because the hub operates on the physical layer, that means that all packets go to all of the hub’s interfaces, which causes congestion on all of the hubs nodes

UDP 1%:

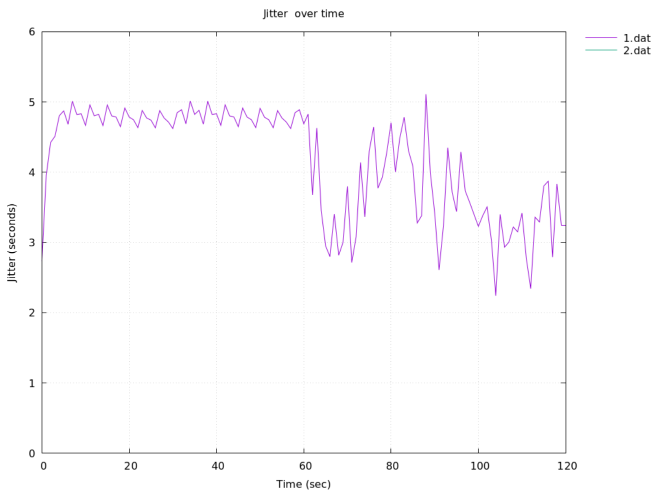


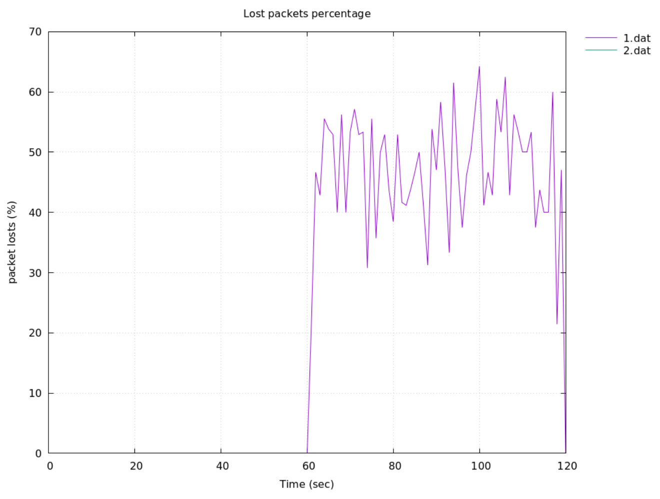


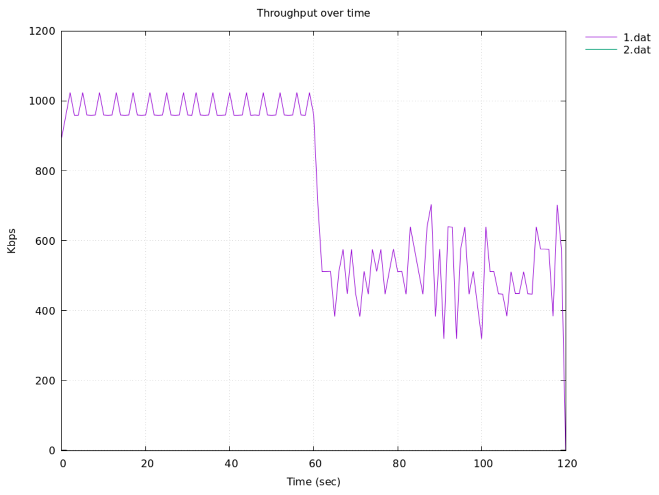


Here we can observe many packet losses although the traffic is low (2% total). This happens because the packets can collide in the hub and as a result they are considered destroyed. Packet loss here is around 10% because we have very small throughput overall 2% but a massive increase in packet loss percentage.

UDP 10%:

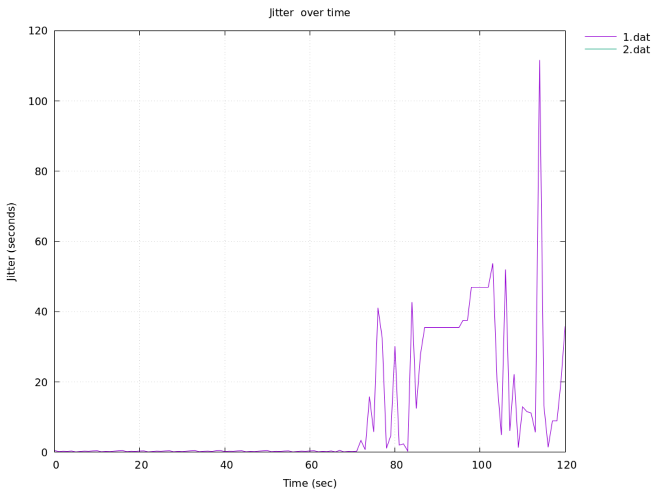


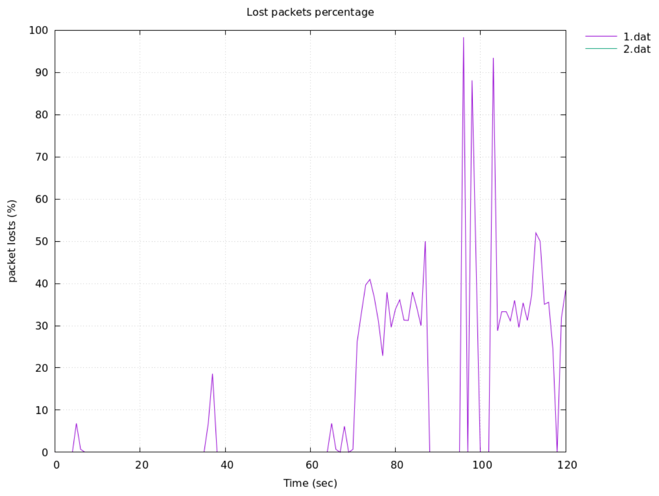


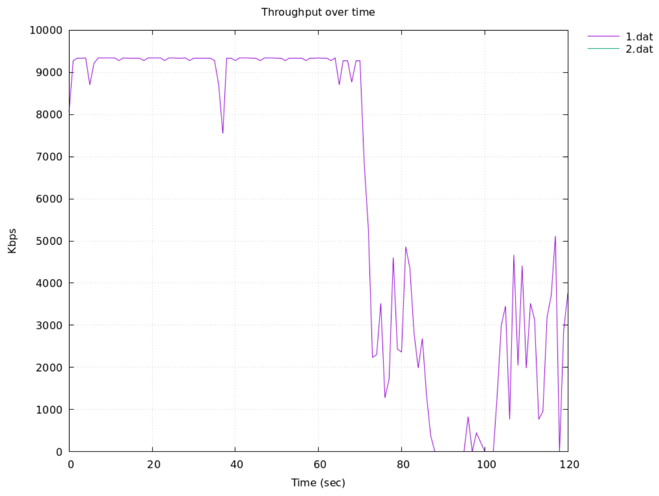


Similarly to the 1% scenario, we can observe a big packet loss due to packet collision in the hub. In this case the packet loss percentage is higher than before because the rate that the nodes send packets is increased

UDP 97%:







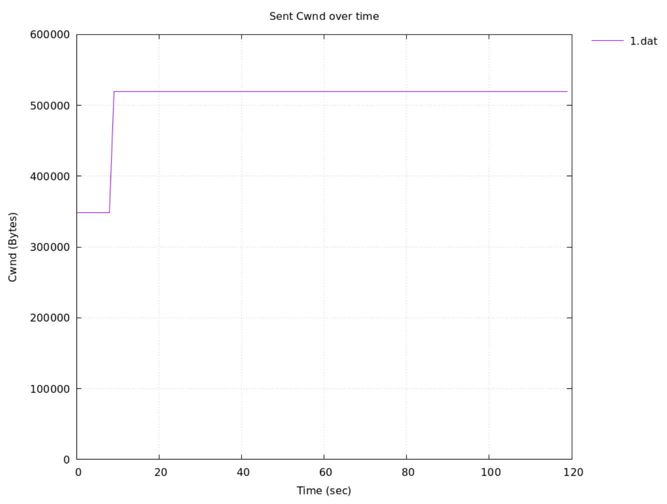
At the 97% UDP bandwidth scenario we observed large peaks of packet loss due to packet collision which leads to a big drop on the connection’s throughput

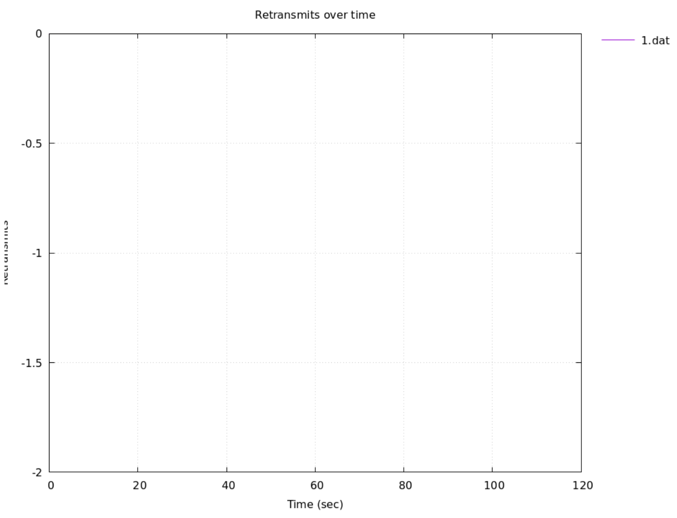
4. Exercise 3

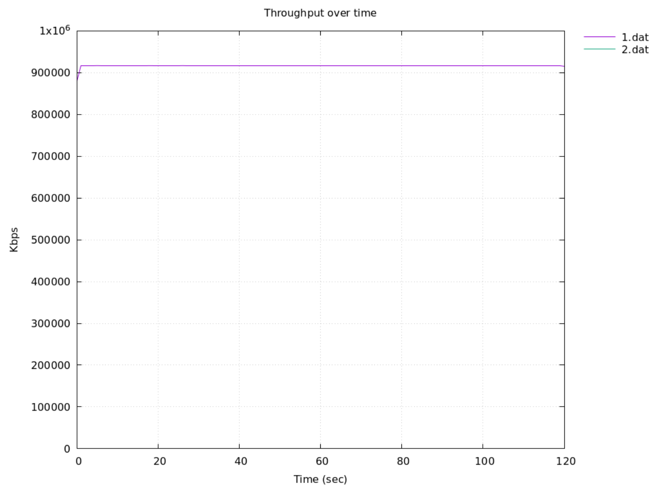
4.1

On all the plots below of 4.1, because we use a switch (which operates in the link layer), network congestion is not created within the network when we start transmitting from node C to node D. This happens because the switch forwards the packets only to the interface that they are destined to. So the transmission between node A and node B does not get affected.

TCP:

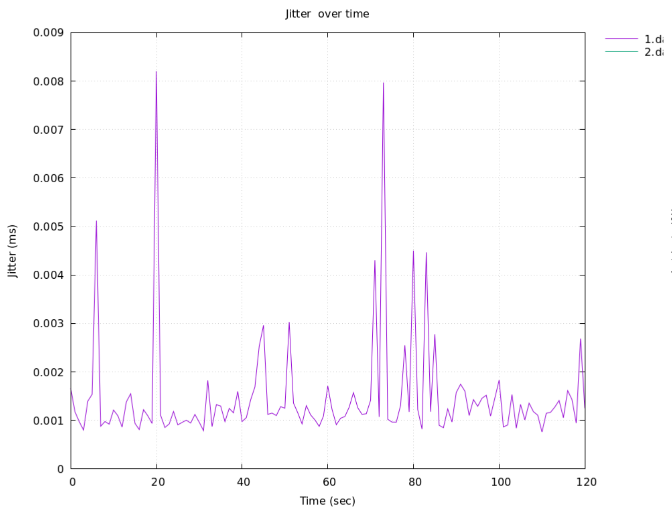


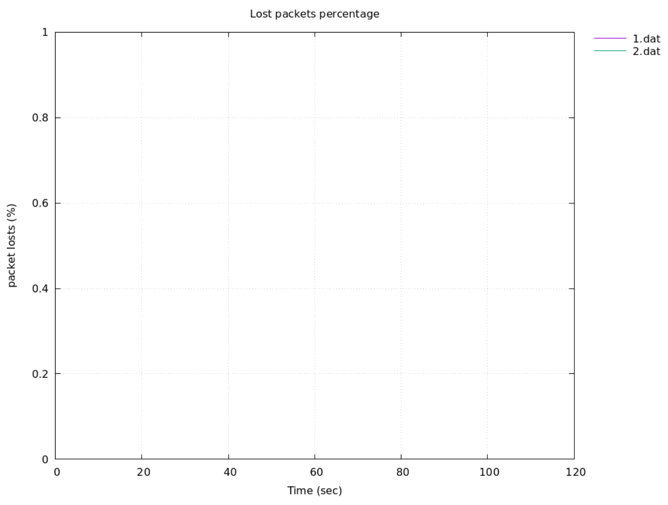


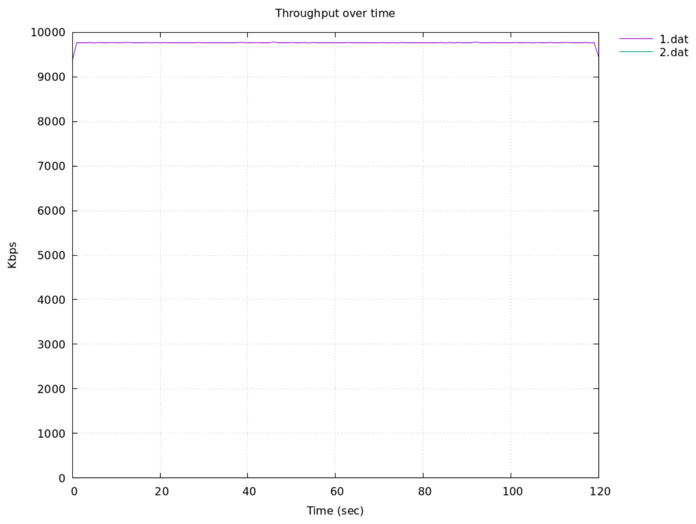


A normal TCP flow can be observed even after the second flow starts (>60sec). In comparison with the 2.2 TCP experiment, now we have two servers in different computers which leads to no congestion

UDP 1% :

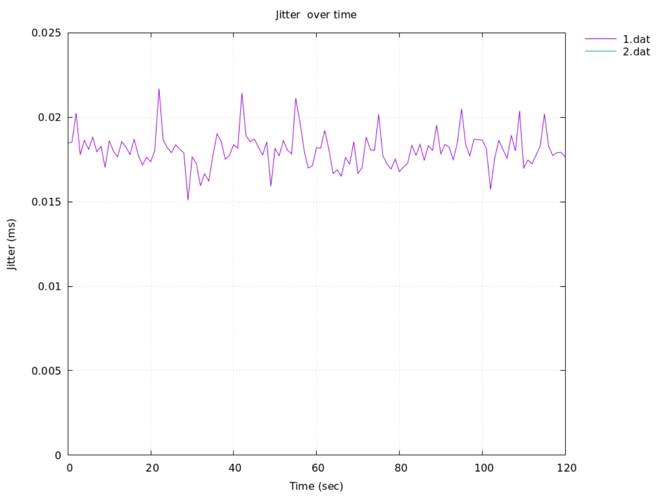


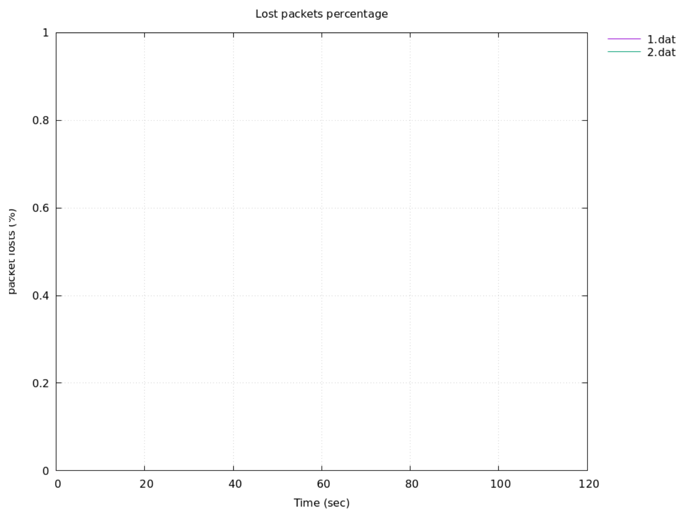


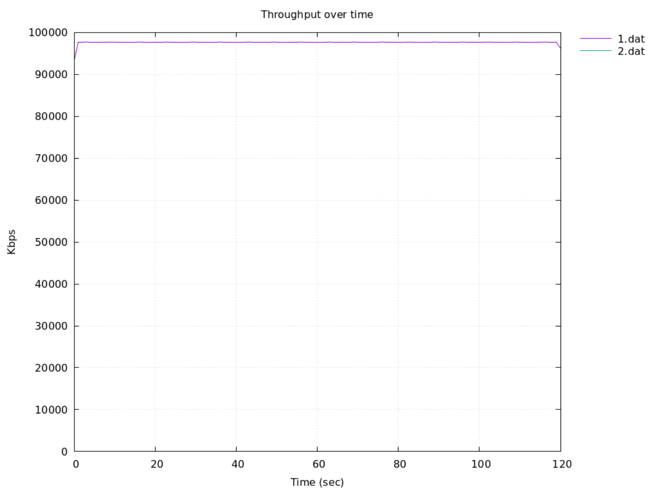


The time variation between client and server has spikes because we send very small amounts of data so the jitter changes at a very small rate which is normal due to many factors. We don't have any packet loss here because we send very small amounts of data. Throughput is stable because we have very small data to send and no network congestion

UDP 10%:

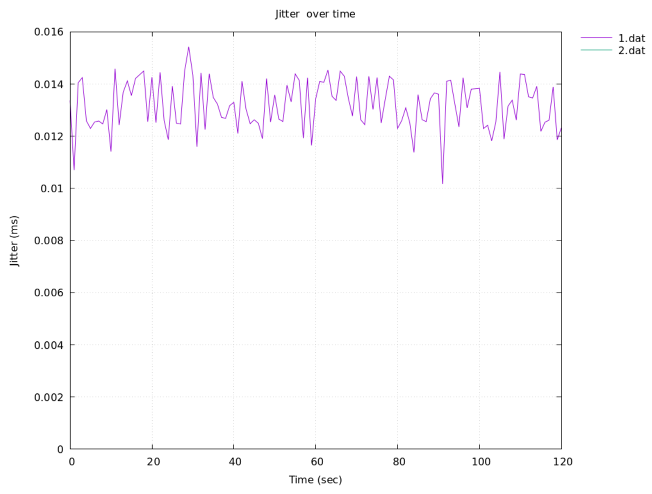


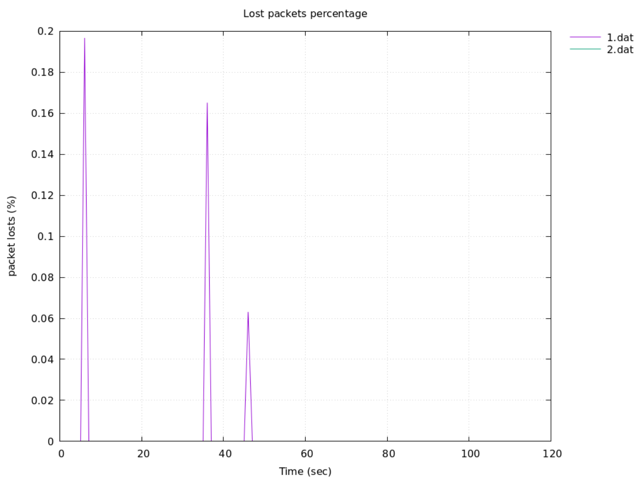


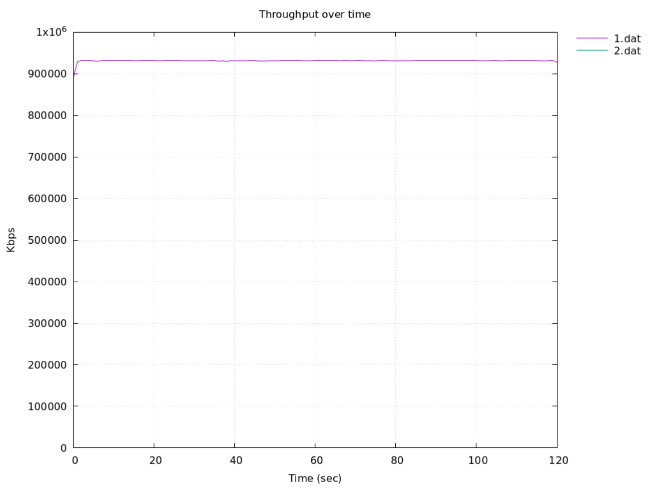


Jitter is around 0.02ms because we increased the data a bit thus the blocks require a little more time to transmit between client and server. We don't have any packet loss here because we don't send much data and we don't have any network congestion. Throughput is stable because we don't have network congestion.

UDP 97%:





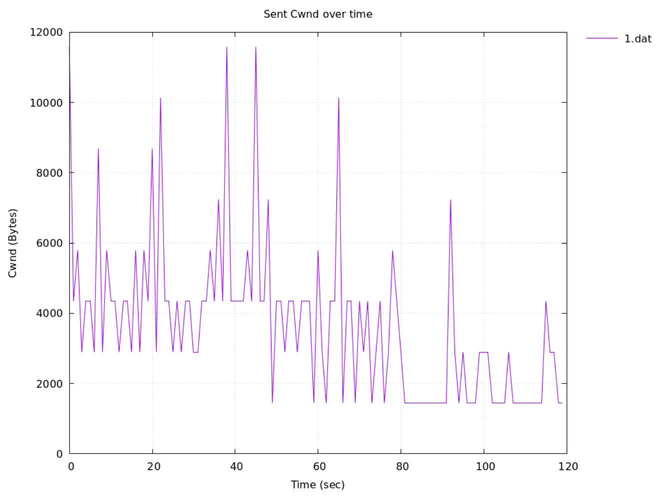


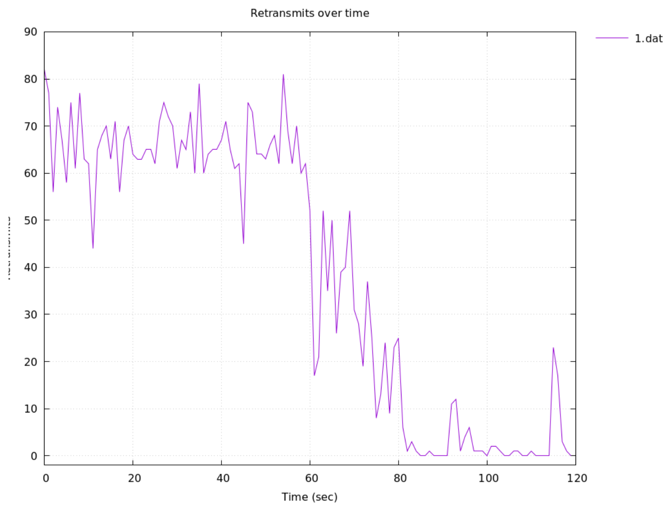
Because we send a lot of data here we have a number of reasons for the lost packets such as the receive buffer of the server got full and dropped a few packets. Or the queue of the switch got full and a few packets got dropped . Jitter is around 0.016 ms here because we send a lot of data so jitter is increased. The throughput is stable at around 900000Kbps we can see small downwards spikes if we zoom very much that is the result of the packet losses we had seen from the previous plot.

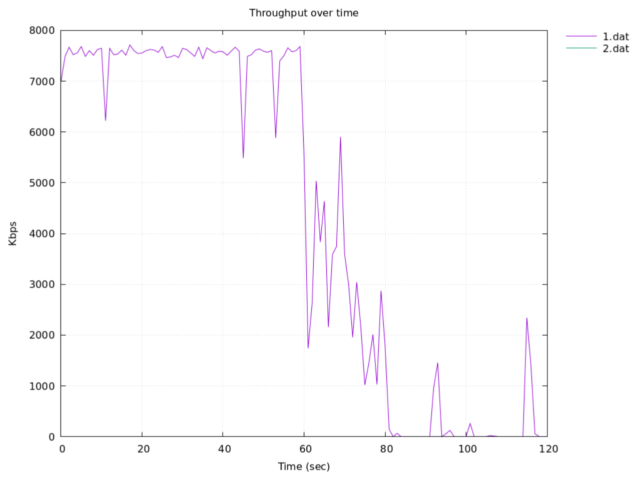
4.2

Because here we use a hub which operates in the physical layer it creates a lot of congestion because it forwards the packet to every interface.We can clearly observe in our plot the congestion in our network.

TCP :

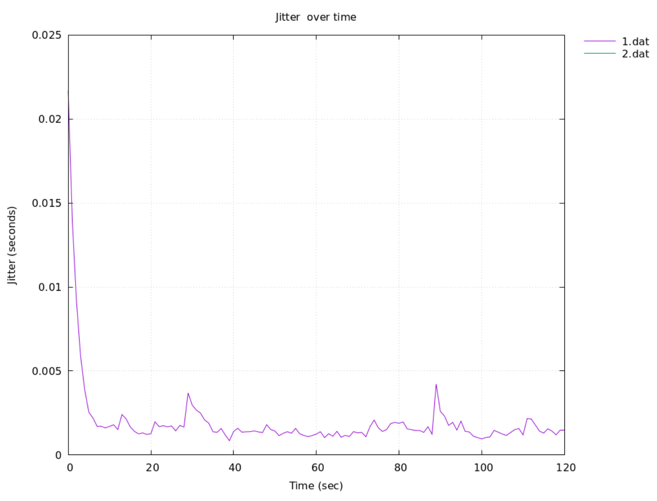


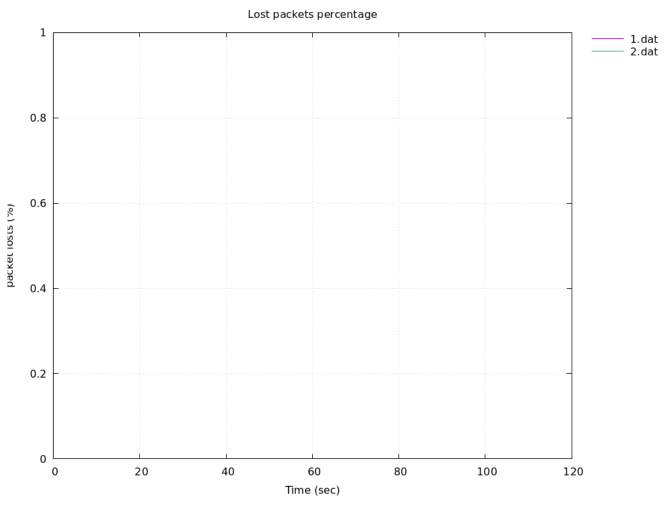


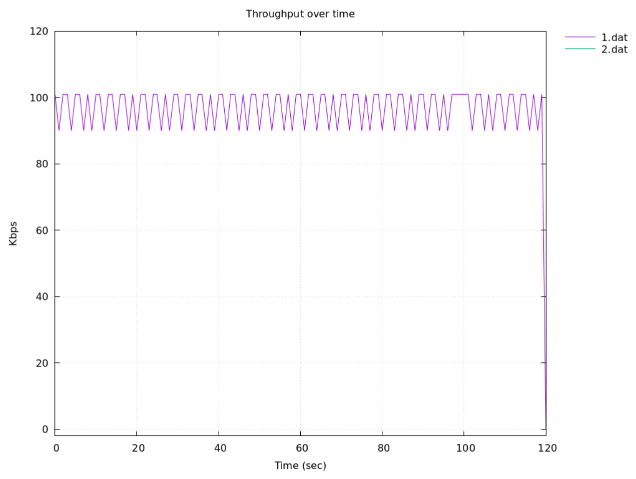


The congestion window after the 60 seconds is smaller most of the time than before or equal because there is congestion in the network due to the hub. From 0 to 60 seconds we can see spikes in the congestion window because of the TCP protocol. From 0 to 60 seconds we have a lot of retransmits around 60-80, that's because we have a large window size. After 60 seconds the retransmits get lower and lower because the window size decreases. The throughput from 0 to 60 seconds we have small downward spikes because of the sudden decreases in the window size. From 60 to 120 we have decreased window size because of the congestion in the network so as expected the throughput is lower.

UDP 1%:

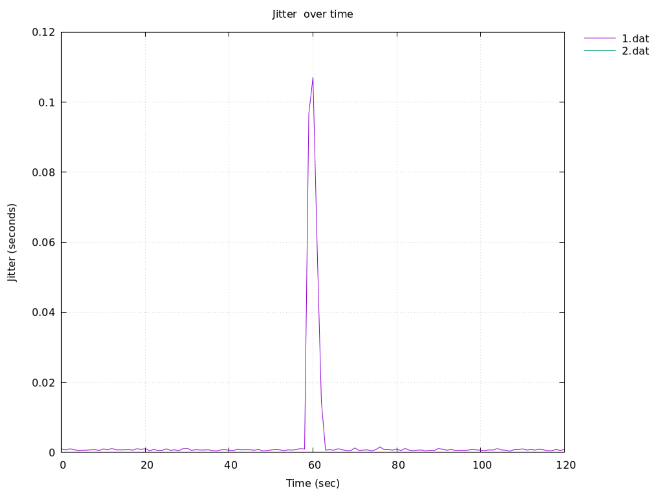


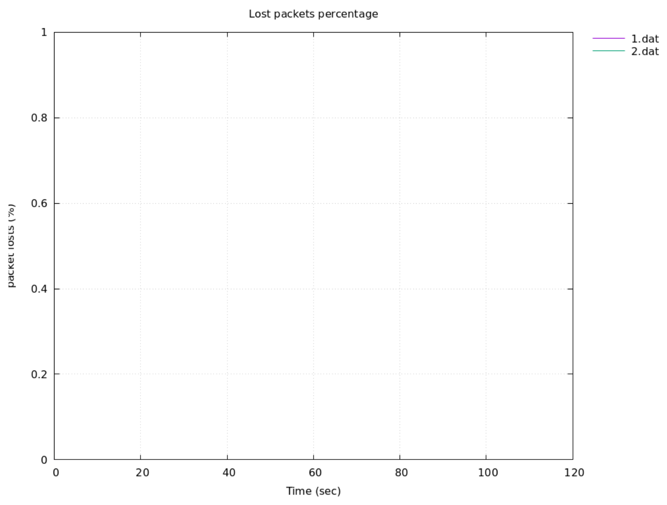


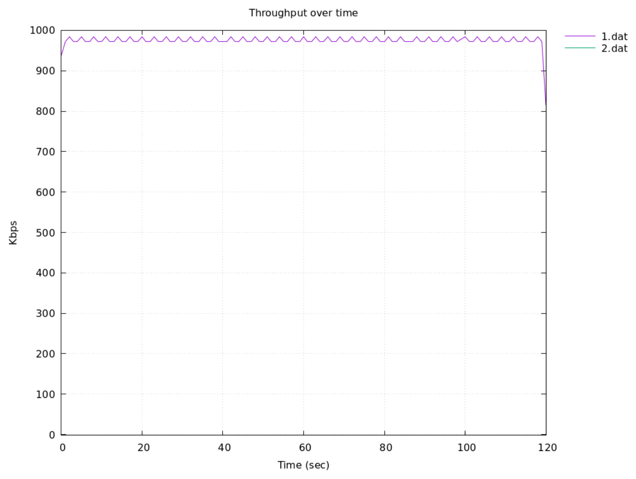


The jitter starts from 0.025ms and goes down to 0.005 ms and lower. This happens for a lot of reasons . The main reason is that in the beginning we have a context switch at the server OS and thus the big jitter at the start. After that the jitter is stable. We don't have any packet loss here because we send very small datagrams. The throughput is at around 100 Kbps and is almost stable. No congestion can be detected here because the sum of the flow is 2% which can easily be handled by the switch.

UDP 10%:

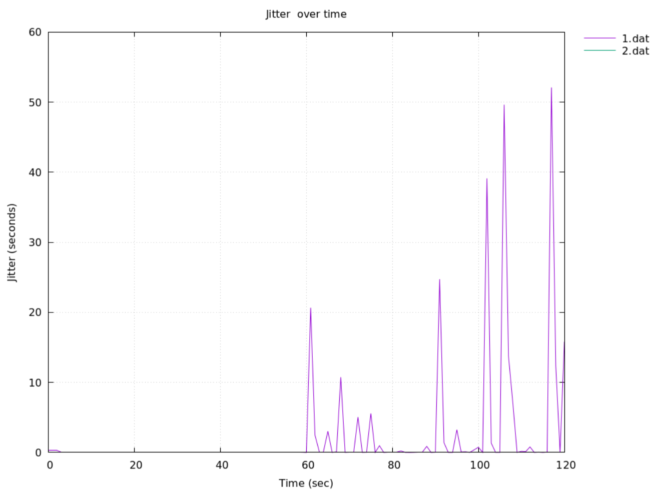


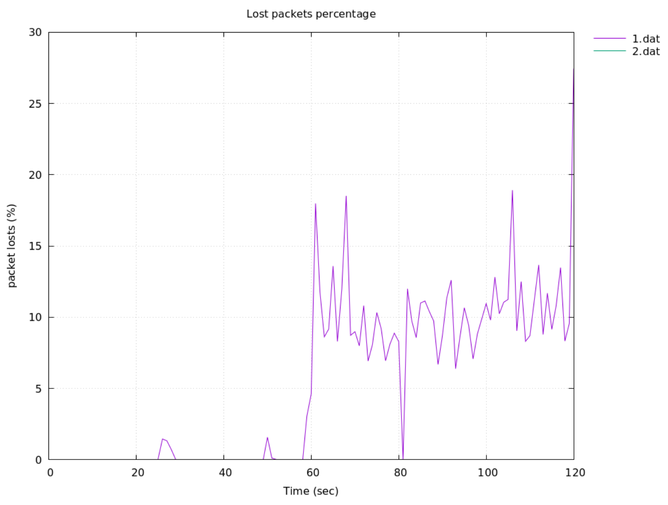


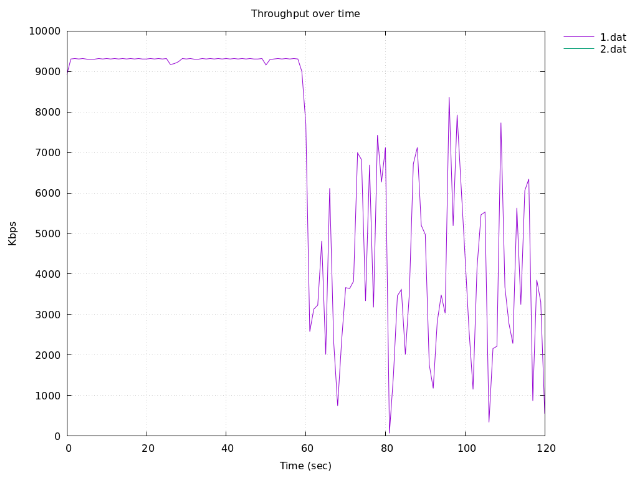


Like the 1% UDP flow scenario, the hub can easily handle the sum of 20% UDP flow without any packet losses or collisions. It's a bit unexpected to have a peak in jitter without lost packets, maybe something happened to the server and the delay between the sender and the receiver increased sharply.

UDP 97%:







At 0-60 seconds the jitter, the throughput and the lost packets percentage are all stable. From 60-120 seconds we created a congestion in the network from the hub so we have jitter spikes, increased packet loss percentages, and a decreased throughput with big spikes due to the congestion in the hub.

Comparing the current topology to the topology of the first scenario, we can understand that when two clients try to “speak”to one server, the congestion is much greater than when they try to “speak” to two different servers (over one switch/hub in both cases). This can be explained because the single server has one link for two clients whereas the two servers use two links for two clients