COMP3331 LAB 5
Abanob Tawfik
Z5075490

Contents

Exercise 1: Understanding TCP Congestion Control using ns-2

Question 1: Run the script with the max initial window size set to 150 packets and the delay set to 100ms (be sure to type "ms" after 100)

What is the maximum size of the congestion window that the TCP flow reaches in this case?

The maximum size of the congestion window that the TCP flow reaches in this case is 100 packets. This occurs during the slow start while it increases exponentially.

What does the TCP flow do when the congestion window reaches this value? Why?

When the congestion window reaches size 100 packets, either 3 duplicate ACK's have occurred or a timeout, causing the congestion window size to reset, and the new threshold is 50 (half of 100, which is when the first loss/triple duplicate ACK occurred). In TCP-Tahoe, regardless of whether a triple duplicate ACK or a timeout occurs, the congestion window size drops down to 1, and the threshold is set to half of the congestion window.

What happens next?

Slow start phase will initiate with window size 1, and the congestion window size will increase exponentially up until the threshold which is now 50, at window size 50 we begin congestion avoidance which is a linear increase up until loss. This is seen in Figure 1.

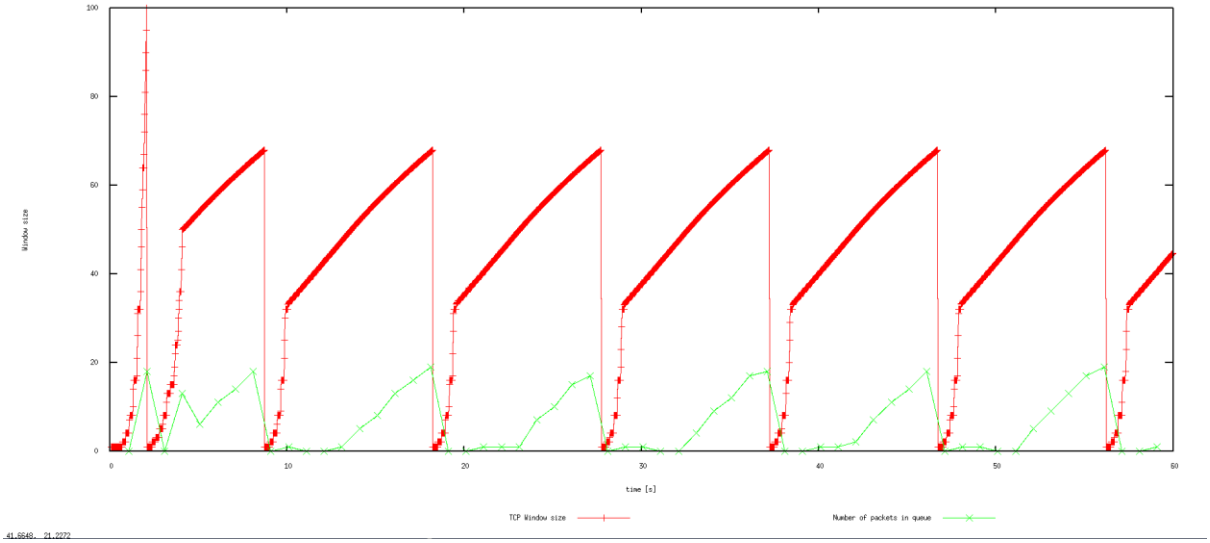

Figure 1 plot of the TCP window size and queued packets

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)

The average throughput in packets/second can be determined from the WindowMon.tr file as **188.97610921501706.** This is seen in Figure 2.

Throughput in packets/second = 188.97610921501706 packets per second.

Since each packet consists of 500 bytes including the header

Packet = payload + headers bytes

Packet = 500 + 20 + 20 bytes

Packet = 540 bytes (or 540*8 bits)

Throughput in bits/second = 540 * (8) * (188.97610921501706) bits per second

= **816376.791809** bits per second

```
59  59.100000000000001 41  0.0036883771140698092 185.0 1 188.97610921501706
60
```

Figure 2 average Throughput in packets/second for the duration

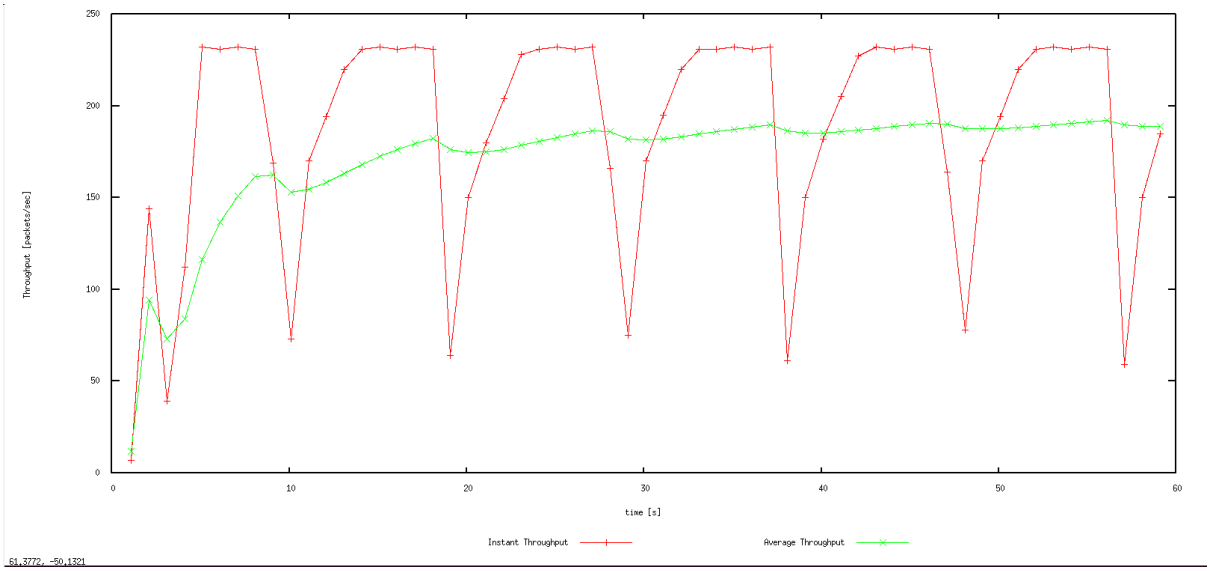

Figure 3 the average and instanteous throughput for the duration

Question 3: Rerun the above script, each time with different values for the max congestion window size but the same RTT).

How does TCP respond to the variation of this parameter?

TCP will make the slow start threshold the value of that parameter, it will also make sure that the window size cannot go over the value input (so if value = 20 the maximum is also 20). When this value increases TCP will continue oscillating, when it decreases drastically, there will be no oscillations however it is not fully efficient as it could fit more packets without congesting. There is a window size such that, there will be no oscillations and the flow will be stable and efficient. See Figure 4 and Figure 5.

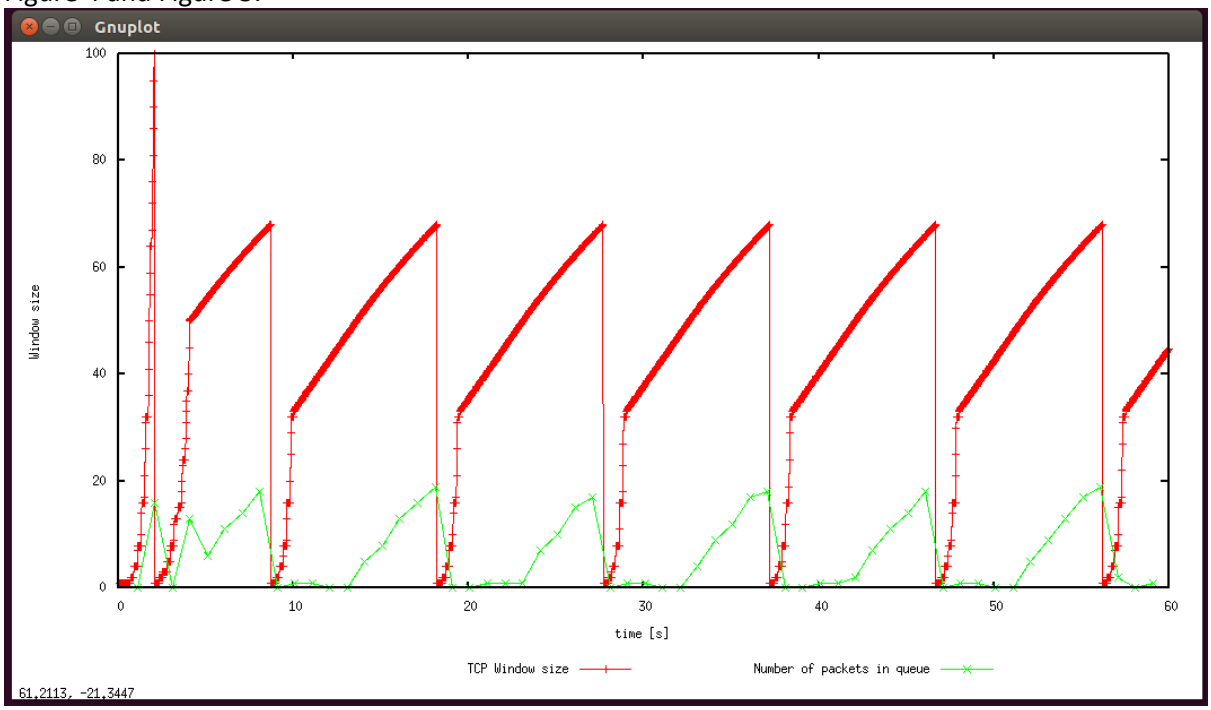

Figure 4 plot when maximum size = 100

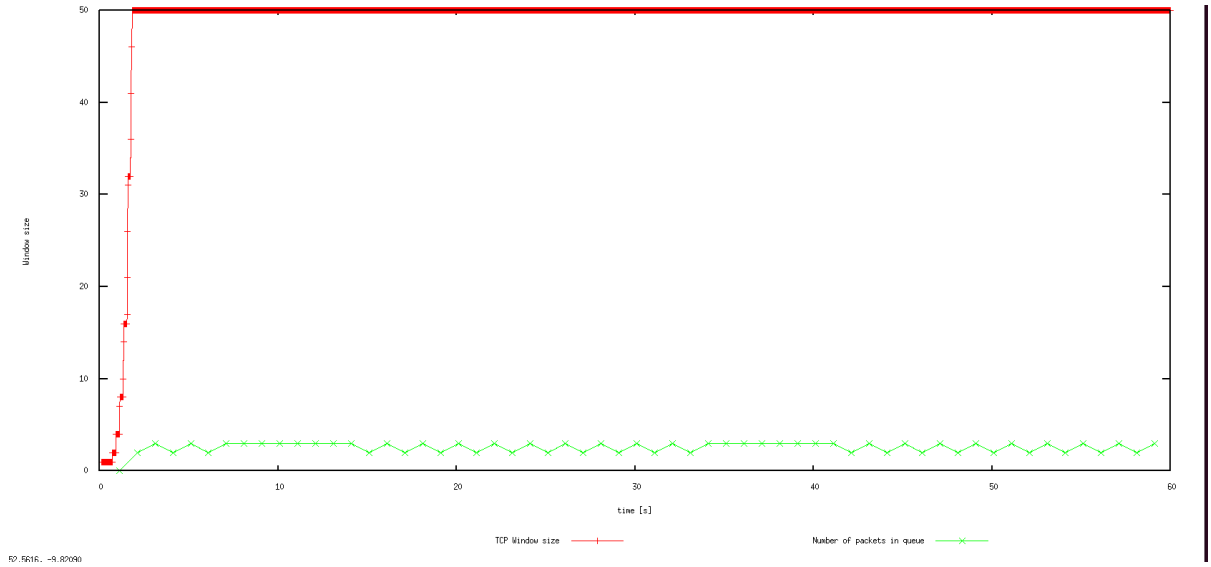

Figure 5 plot when maximum size = 50

Find the value of the maximum congestion window at which TCP stops oscillating to reach a stable behaviour.

The value of the maximum congestion window size which stops oscillation and transmission is stable, is when the maximum window size is equal to **66** This can be seen in Figure 6, the initial drop is due to slow start increasing exponentially, however since the threshold for slow start will drop in half, it will perform congestion avoidance the next time, and will increase and reach stability at maximum window size 66, at window size max = 67 we get oscillations. There is a case where if congestion max window is 50, there will be no initial drop, as it will reach the maximum before there can be any loss in the slow start phase. However, since the maximum window size is 50, we cannot get optimal throughput and utilise the link as much as in the case of window size 66. This is seen in Figure 5.

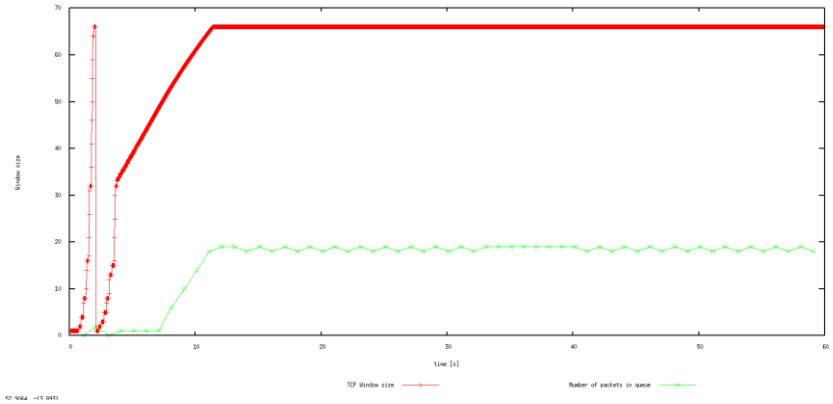

Figure 6 stable transmission at window size max = 66

What is the average throughput (in packets and bps) at this point?

The average throughput in packets/second can be determined from the WindowMon.tr file as **220.81911262798636.** This is seen in Figure 7.

Throughput in packets/second = 220.81911262798636 packets per second.

Since each packet consists of 500 bytes including the header

Packet = payload + headers bytes

Packet = 500 + 20 + 20 bytes

Packet = 540 bytes (or 540*8 bits)

Throughput in bits/second = 540 * (8) * (**220.81911262798636**) bits per second

= **953938.566553** bits per second

```
59  59.100000000000001 14  0.0010792476102374346 232.0 18 220.81911262798636
60
```

Figure 7 average throughput for the duration

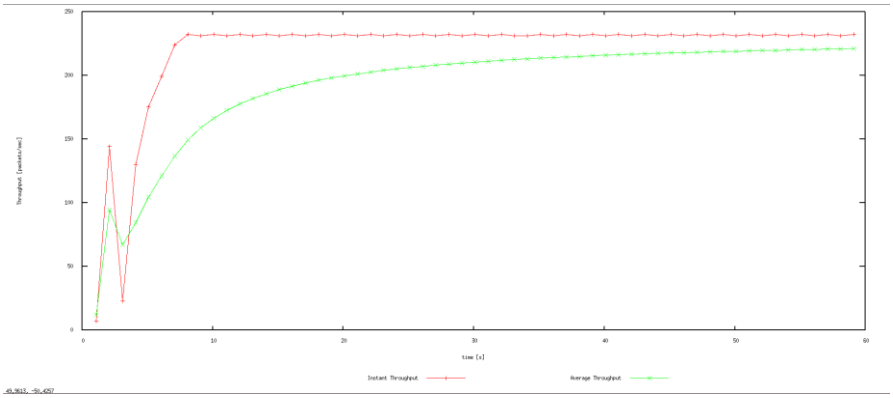

Figure 8 plot of throughput for the duration instanteous and average

How does the actual average throughput compare to the link capacity (1Mbps)?

The average throughput is
953938.566553 bits/second
The link capacity is
1000000 bits/second
Hence our utilisation of the link becomes
100 – 100*(1000000-953938.566553)/(1000000)
= **95.3938566553%** utilisation of the link.
The average throughput is almost at link capacity in this case.

Question 4: Repeat the steps outlined in Question 1 and 2 but for TCP Reno. Compare the graphs for the two implementations and explain the differences. How does the average throughput differ in both implementations?

What is the maximum size of the congestion window that the TCP flow reaches in this case?

The maximum size of the congestion window that the TCP flow reaches in this case is 100 packets. This occurs during the slow start while it increases exponentially. This is the same behaviour as TCP Tahoe had.

What does the TCP flow do when the congestion window reaches this value? Why?

When the congestion window reaches size 100 packets, initially the window size drops to half and so does the slow start threshold indicating that three duplicate ACK's have returned, there is a brief transmission and the window size drops to 1 and the slow start threshold is reduced in half to 25. This is due to a timeout.

What happens next?

There is a brief transmission, however on receiving the timeout, the window size drops to 1 and slow start phase initiates till window size is 25, due to the slow start threshold halving twice from 100 (25). Afterwards we entered congestion avoidance phase. With the oscillations for TCP Reno, on triple duplicate ACK's our congestion window only halves, and depending on when this occurs, we enter congestion avoidance once again (slow start only occurs when window size < slow start threshold).

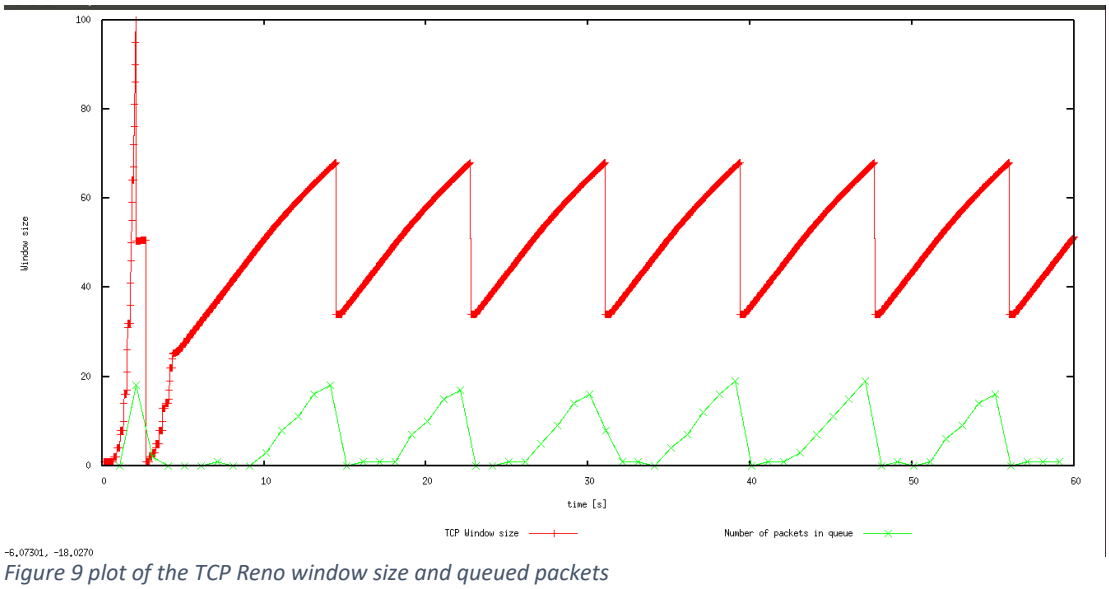

Figure 9 plot of the TCP Reno window size and queued packets

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)

The average throughput in packets/second can be determined from the WindowMon.tr file as **188.97610921501706.** This is seen in Figure 2.

Throughput in packets/second = 203.41296928327645 packets per second.

Since each packet consists of 500 bytes including the header

Packet = payload + headers bytes

Packet = 500 + 20 + 20 bytes

Packet = 540 bytes (or 540*8 bits)

Throughput in bits/second = 540 * (8) * (203.41296928327645) bits per second

= **878744.027304** bits per second

```
59  59.100000000000001 41  0.0034275204815248286 214.0 1 203.41296928327645
60
```

Figure 10 average Throughput in packets/second for the duration (TCP Reno)

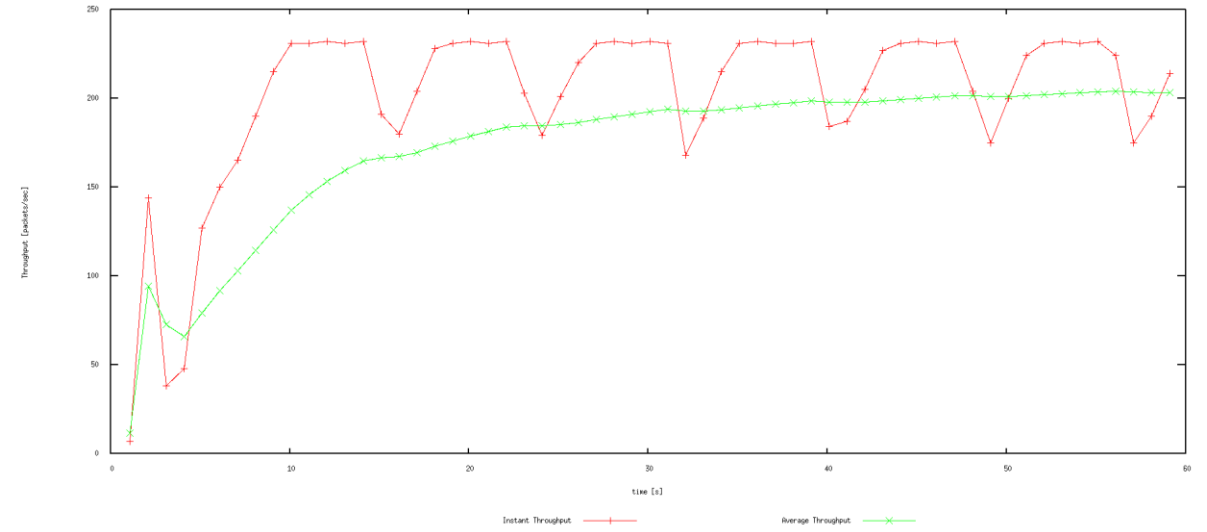

Figure 11 the average and instanteous throughput for the duration (TCP Reno)

Exercise 2: Flow Fairness with TCP

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.

Each flow gets an equal share of the capacity of the link. Initially each link gets a different amount of the capacity, however as time increases, each flow decreases and increases until they average around a similar value. This is done through additive increase, multiplicative decrease. This is shown in Figure 12.

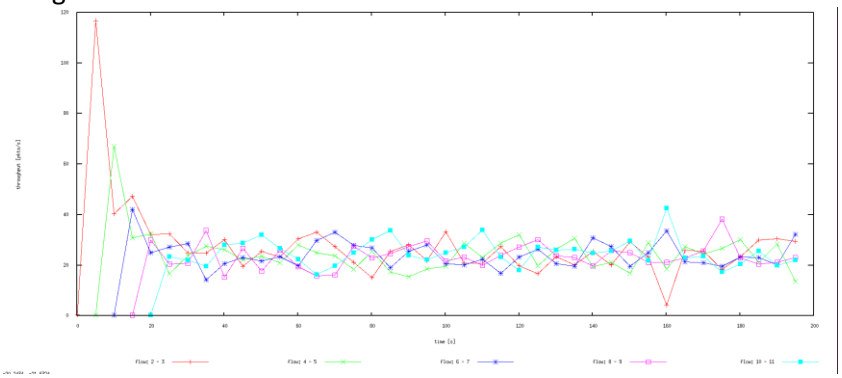

Figure 12 tcp flow distribution

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 a new flow is created, the pre-existing TCP throughputs will decrease in order to give an equal share to each flow. The mechanism of TCP that contributes to this behaviour is additive increase, multiplicative decrease (AIMD) which is a congestion control algorithm. This behaviour is fair as multiple flows using this mechanism will converge to receive an equal share of the link, whilst efficiently utilising the link. This is seen in Figure 13, where AIMD tends to oscillate around the fairness and efficiency lie, and in Figure 14 which illustrates the AIMD sharing dynamics.

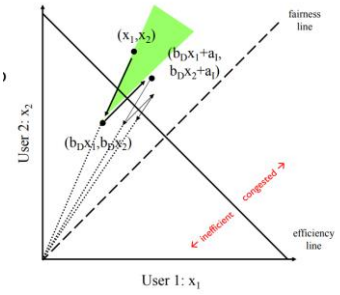

Figure 13 this diagram shows two users in a connection using AIMD to be on both the fairness and efficiency line.

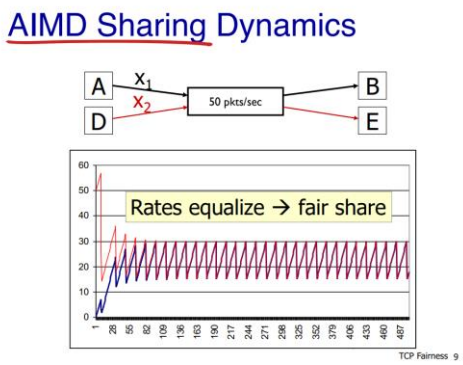

Figure 14 This diagram shows the fluctuation in flowrate equalising to give both x1 and x2 an equal share of the link

Exercise 3: TCP competing with UDP

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 expect if the capacity of the link is increased to 5Mbps, that both flows would have an increased throughput, however UDP will have a higher throughput than TCP.UDP has no congestion control whereas TCP has built in congestion control. This is observed in Figure 15.

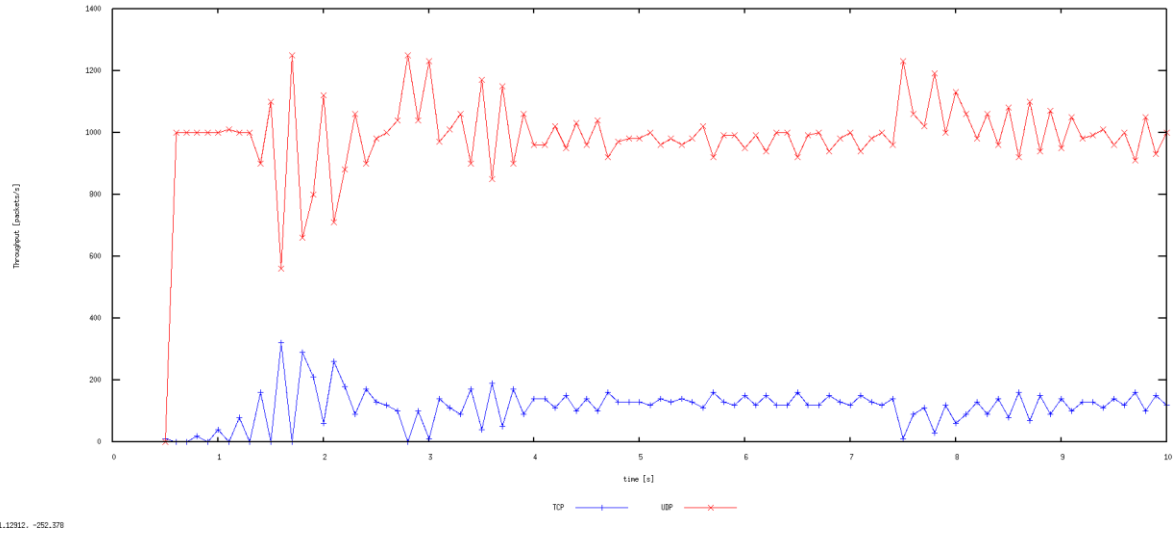

Figure 15 throughput for link with capacity 5Mbps

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.

UDP achieves a much higher throughput than TCP due to having no form of congestion control mechanism. UDP is simply a best effort protocol, whereas TCP has congestion control mechanisms built in place in order to not overload the link and provide a stable connection. TCP congestion control will change its window size based on conditions as seen above in Figure 15, there are times where the throughput drops to 0, which is when timeout occurs and the window size changes. There are also drops which indicate triple duplicate ACK's. in general, there are also fluctuations in both UDP and TCP which seem to come from a network that has changing conditions. The TCP line on the plot has a similar plot to the UDP line, when the UDP throughput drops drastically it can be observes as loss from the TCP plot in the same position.

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 to using UDP instead of TCP would be that

1. There is a much higher overall throughput using UDP instead of TCP
2. If only a few people were using UDP on the network they would get much faster data transfer, as it will transfer based off the link bandwidth.

Disadvantages to using UDP instead of TCP would be that

1. There is no form of congestion control, so a link could end up collapsing and blocking due to packets being sent to a blocked link.
2. There is no form of guarantee that you will receive the file, and receive it in order without any data corruption (UDP is a best effort protocol).

If everyone begins using UDP instead of TCP for the advantages, the network would most likely block, and new packets being sent, would simply be dropped as the queue is full.