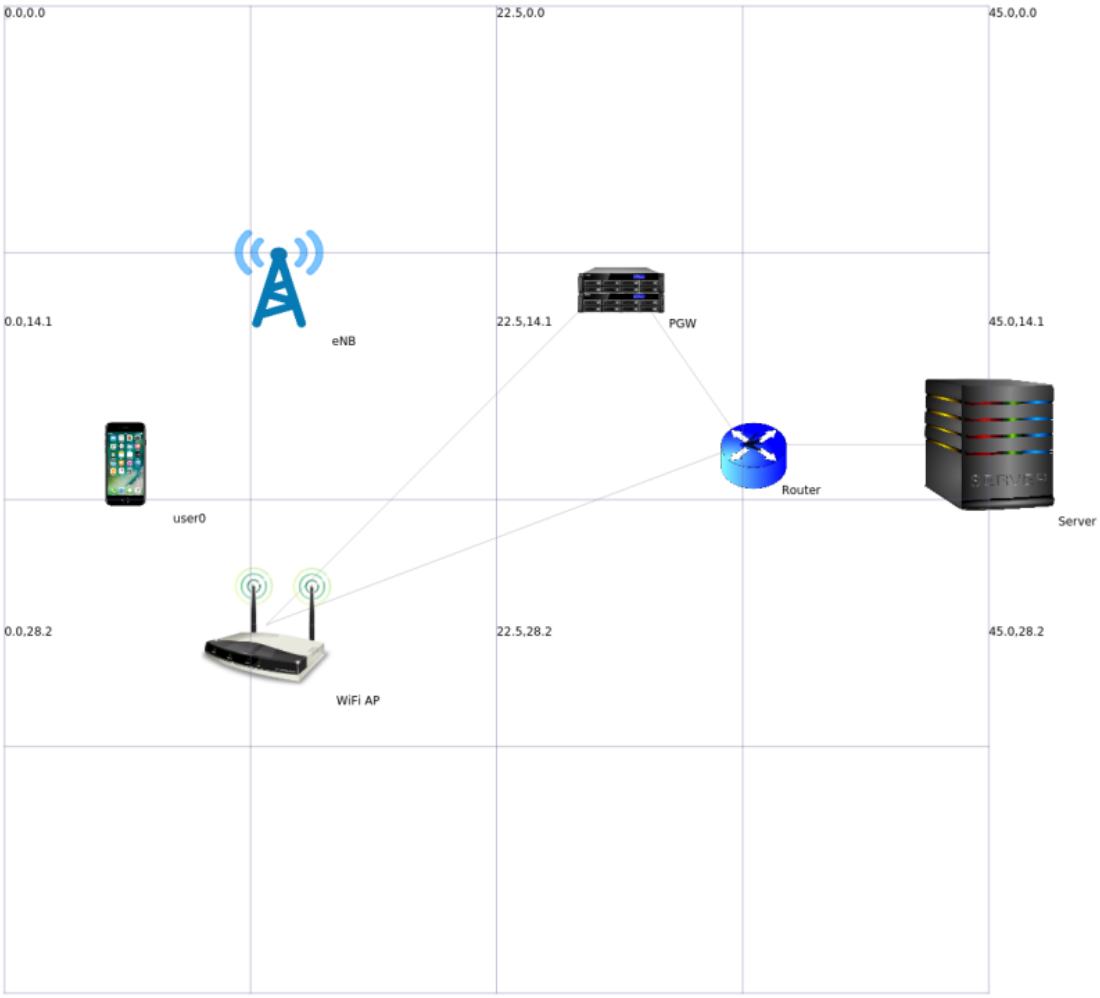
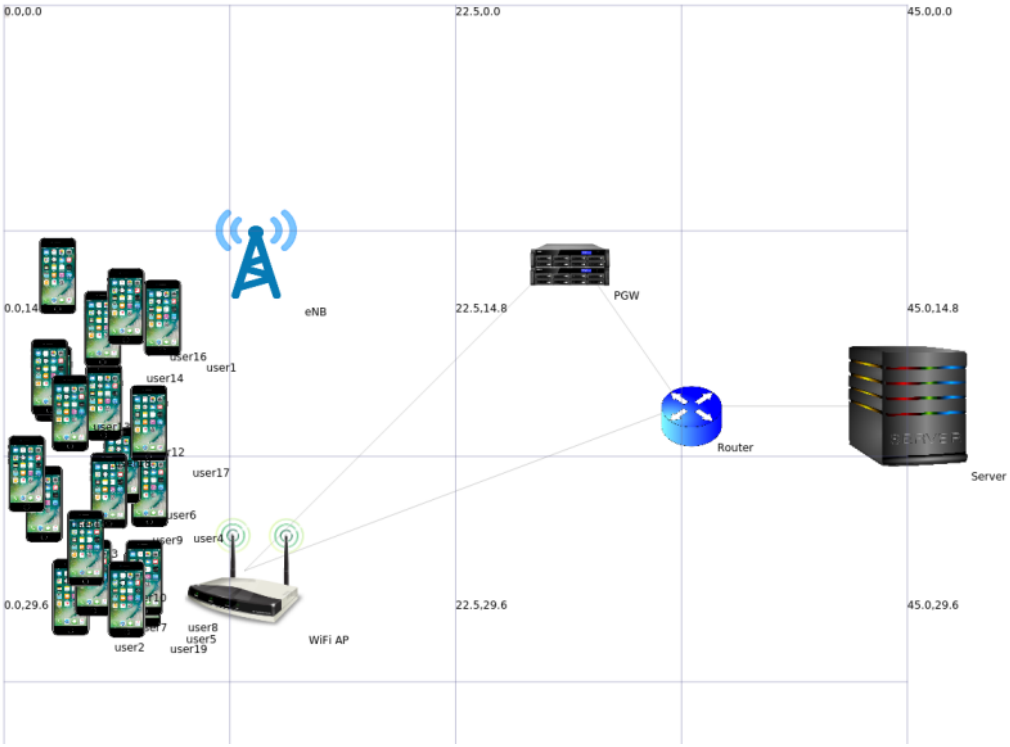
Problem 1:

1 UE



20 UEs:



Problem 2:

From the generated pcap files, all three look very similar to each other, however they are not completely the same. All three pcap files have the same amount of packets (including data, SYN, ACK, and FIN packets) delivered in total: 1078. All use the TCP protocol. The header lengths are the same, and so is the information within all of the packets sent over the network. However, the difference is when each packet is received, from server to router and router to server/wifi-ap. Since there is a network delay of 20 ms between router-to-server and router-to-wifi-ap, each packet is delivered 20 ms after it is sent to its destination. For example, when the router sends the server a message, the server will receive the message after 20 ms.

The router-to-server and router-to-wifi-ap pcap file are the same, even in time difference, because the router sends packets to each destination at the same time. These two files differ with the server-to-router pcap file though, because of RTT. Every packet delivered from the server to UE is 40 ms behind due to the 20 ms delay between the router-to-server and router-to-wifi-ap. Because the packet sent to the UE goes from server to router to wifi-ap, 20 ms and 20 ms add up to 40 ms delay in total. This can be seen in the pcap files as well: In the router-to-server pcap, the router sends the server packet 8 at 0.209. The server sends packet 8 back to the router at 0.249. There is a 40 ms difference.

Problem 3:

RTT(ms)	RWND (Bytes)	MCS	1 UE		3 UE			
			Per UE		Per UE		System level	
			Wi-Fi TCP		Wi-Fi TCP		Wi-Fi TCP	
			Peak (Mbps)	Avg (Mbps)	Peak (Mbps)	Avg (Mbps)	Peak (Mbps)	Avg (Mbps)
30 ms	64000	HtMcs 1	14.672	1.408	14.448	0.906	23.968	0.893
200 ms	64000	HtMcs 1	5.04	0.296	5.04	0.286	13.664	0.286
30 ms	1024000	HtMcs 1	23.968	2.703	23.856	1.453	24.08	0.870
200 ms	1024000	HtMcs 1	23.856	1.695	23.968	0.960	23.968	0.654
30 ms	64000	HtMcs 7	16.8	1.961	17.248	1.786	47.712	1.786
200 ms	64000	HtMcs 7	5.04	0.297	5.04	0.296	15.12	0.296
30 ms	1024000	HtMcs 7	113.568	10	114.464	5.892	114.464	3.030
200 ms	1024000	HtMcs 7	70.336	2.222	81.872	2.113	114.464	2.083

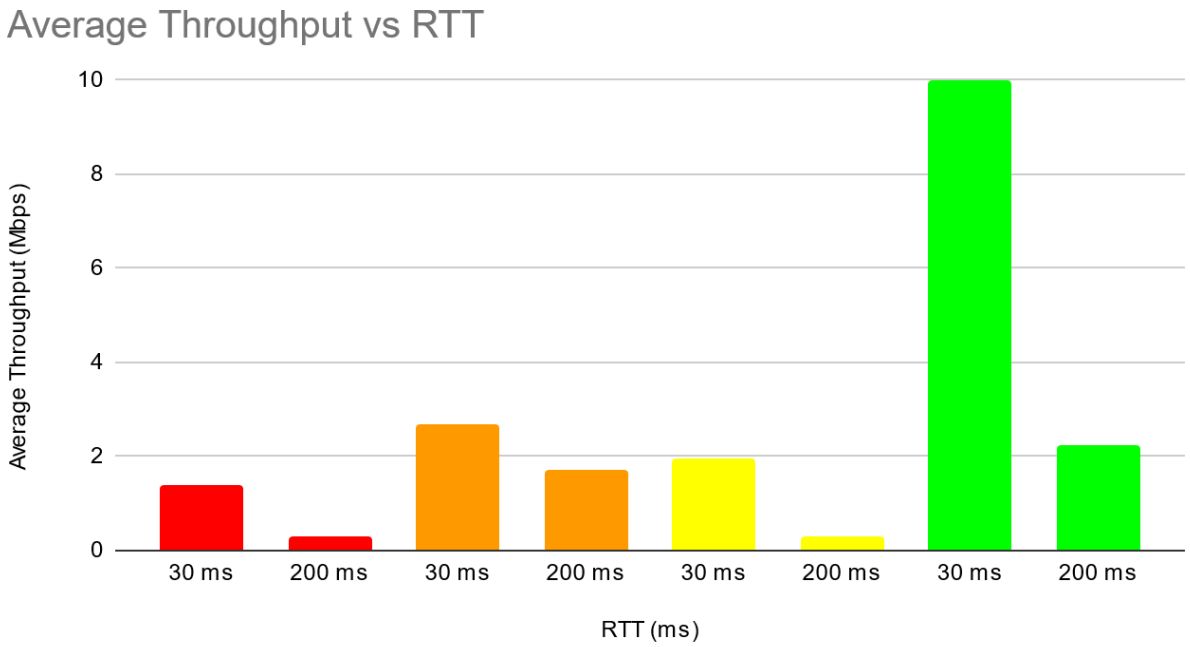
In this problem, the throughput is the measurement of the rate in which it takes to send a certain file size from source to destination via the network. We already know the file size that is being sent over the network: 10Mb. All we need is to find the duration it takes to deliver the entire file from server to UE by using application start and end time. We can subtract these values to get the duration. The throughput of a TCP session relies on how long it takes to receive all packets necessary.

Now, let us interpret the data. Some trials definitely meet expectations, while others vary. Here are some noticeable patterns to observe:

- Increasing RTT decreases throughput:
    - When increasing the RTT for a window size of 64000 and MCS of 1, the average throughput goes down from 1.408 to 0.296 when looking at only 1 UE. The same difference can be said for 3 UEs: the average throughput goes down significantly

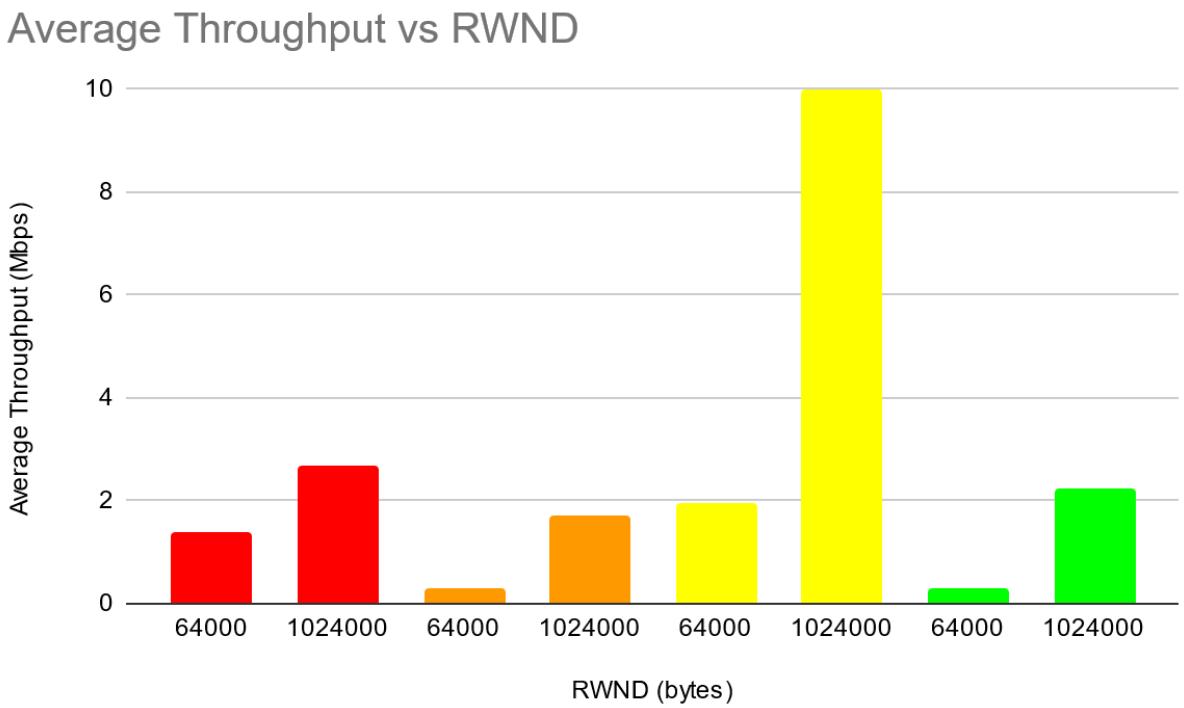
when increasing RTT. This makes sense, as less packets can be delivered in a short period of time due to RTT delay. The same observation can be seen for a RWND of 1024000 and MCS of 1, for a RWND of 64000 and MCS of 7, and for a RWND of 1024000 and MCS of 7.

- Comparison is clearly shown in graph below for 1 UE, where red indicates RWND of 64000 and MCS of 1, orange is RWND of 1024000 and MCS of 1, yellow is RWND of 64000 and MCS of 7, and green is RWND of 1024000 and MCS of 1.

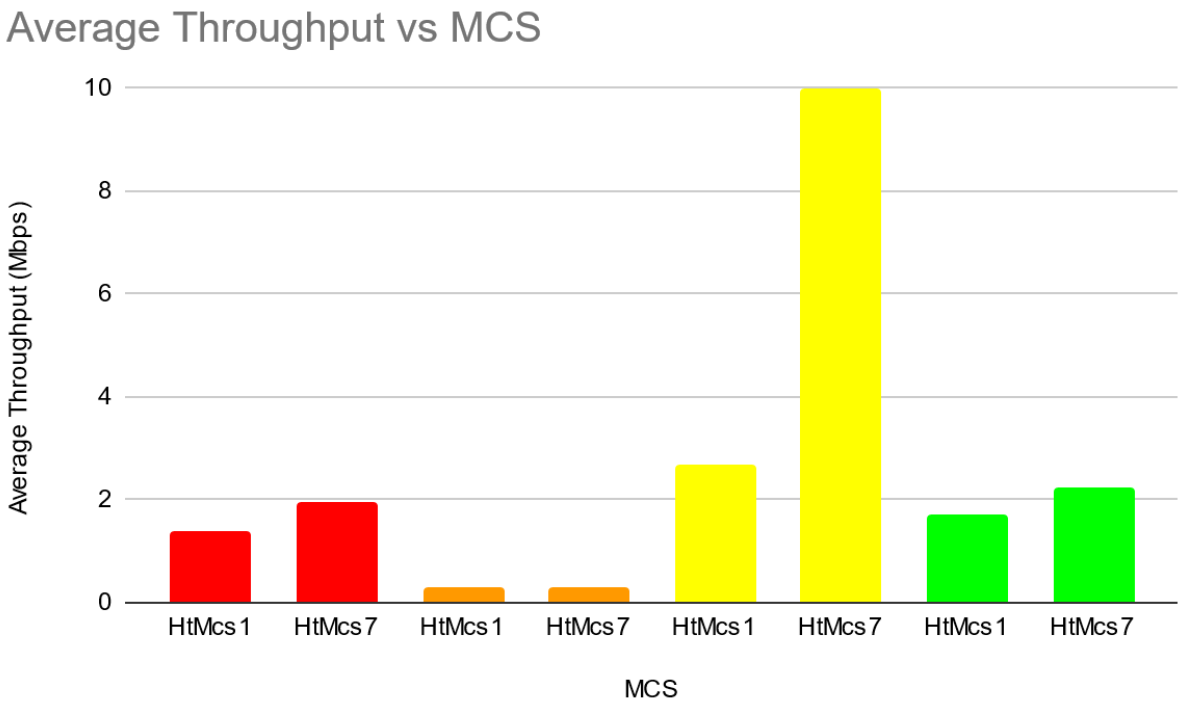


- Decreasing RWND decreases the throughput:
  - When decreasing the RWND from 1024000 to 64000, throughput is decreased. This is because less information can be placed within a packet, decreasing the total information that can be sent in a shorter period of time. This is seen when you compare an RTT of 200 ms and MCS of 7: at 1024000, the throughput is 2.222 and at 64000 it is 0.297 for one UE. The same decreasing average throughput is present for 3 UEs with these parameters as well. This difference can be seen for other comparing trials too, like for RTTs of 200 ms and MCS of 1 and RTT of 30 ms and MCS of 7.
  - The only trial that does not follow this particular notion is for RTT of 30 ms and MCS of 1, specifically for the system level averages. At a RWND of 1024000, the average throughput is 0.870 whereas for RWND of 64000 it is 0.893. At a smaller RWND there is a higher average throughput. This is because the system level throughput aggregates the data for the 3 UEs. However if we actually look at the average throughput for the 3 UEs per UE, at a RWND of 1024000, the average throughput is 1.453 whereas for RWND of 64000 it is 0.906, following our decreasing trend.
  - Comparison is clearly shown in graph below for 1 UE, where red indicates RTT of 30 ms and MCS of 1, orange is RTT of 200 ms and MCS of 1, yellow is RTT of 30 ms and MCS of 7, and green is RTT of 200 ms and MCS of 7. Increasing the window size from

64000 bytes to 1024000 bytes increases average throughput.



- Increasing MCS increases average throughput:
  - When increasing MCS from 1 to 7, the average throughput is increased. This is because more symbols are present to represent more information in one packet, so more information can be sent at once. This can be seen when looking at 30 ms RTT and RWND of 1024000: at MCS of 1, the average throughput is 2.703 and at MCS of 7 the average throughput is 10 for 1 UE. This is a significant increase, and can be observed for 3 UEs as well. The same increase is present in other trials as well, like for RTT of 30 ms and RWND of 64000 and RTT of 200ms and RWND of 1024000.
  - The only trial that is slightly similar in throughput is RTT of 200ms and RWND of 64000, where at MCS of 1, the average throughput is 0.296 and at MCS of 7 the average throughput is 0.297. The second is slightly higher, which is what we expect, however the values are very close in value unlike the other trials. The difference is not as significant. In this case, the RWND might be the limiting factor: even increasing MCS does not affect performance due to the smaller RWND value.
  - Comparison is clearly shown in graph below for 1 UE, where red indicates RTT of 30 ms and RWND is 64000 bytes, orange is RTT of 200 ms and RWND is 1024000 bytes, yellow is RTT of 30 ms and RWND of 64000, and green is RTT of 200 ms and RWND of 1024000.



Problem 4:

RTT	MCS	1 UE	
		System Level	
		Wi-Fi UDP	
		Peak (Mbps)	Avg (Mbps)
30 ms	HtMcs 1	25.2	25.032
200 ms	HtMcs 1	25.088	25.032
30 ms	HtMcs 7	122.192	121.615
200 ms	HtMcs 7	122.08	121.699

To measure the link throughput, it is better to use UDP traffic instead of the results obtained from using TCP in Problem 3 because TCP average throughput is based on session duration: you send a certain amount of data, and the throughput is measured by seeing how long it takes to send all of that data over. In UDP, however, you set a data rate and simply pass information along: there is no "start" and "stop" like there is for TCP. The server may be sending a lot of data over, however if the data rate at a node along the path to the UE is lower, then the throughput on the UE end will be dependent on the bottleneck, not the actual server's data rate. With UDP, we can set the data rate and see how much the UE throughput is; if it is lower than the data rate set, we know that the network has a bottleneck somewhere. In reference to the UE, that bottleneck will be the link throughput, since that is actually the rate at which the UE receives packets.

In comparison to the TCP trials we did, the UDP trials are much more consistent when you change a parameter. For example, when changing RTT, the throughput is really not affected much: at 30 ms for MCS of 1, the average is 25.032 and it stays the same when RTT is increased to 200ms. The same goes for MCS of 7 and RTT of 30ms, which goes from 122.615 and remains around the same value when increased to 200ms, around 121699. Increasing RTT does not impact average throughput like it did for the TCP sessions. This is because UDP does not use ACK packets like TCP does; RTT must be accounted for in TCP sessions because an ACK packet must be delivered back to the sender in order to confirm packet delivery, which can have a delay due to RTT. In UDP, the sender does not depend on the receiver's feedback. However, changing the MCS value did significantly change the average throughput. When keeping RTT of 30 ms at MCS of 1, the average throughput is at 25.032 and goes up to 121.6155 when MCS increases to 7. We can see that changing MCS actually has an impact on average throughput whereas RTT does not for UDP. Changing MCS will allow UDP to deliver more information within a packet at once though, because more symbols can be used to store more bits, which significantly increases throughput.