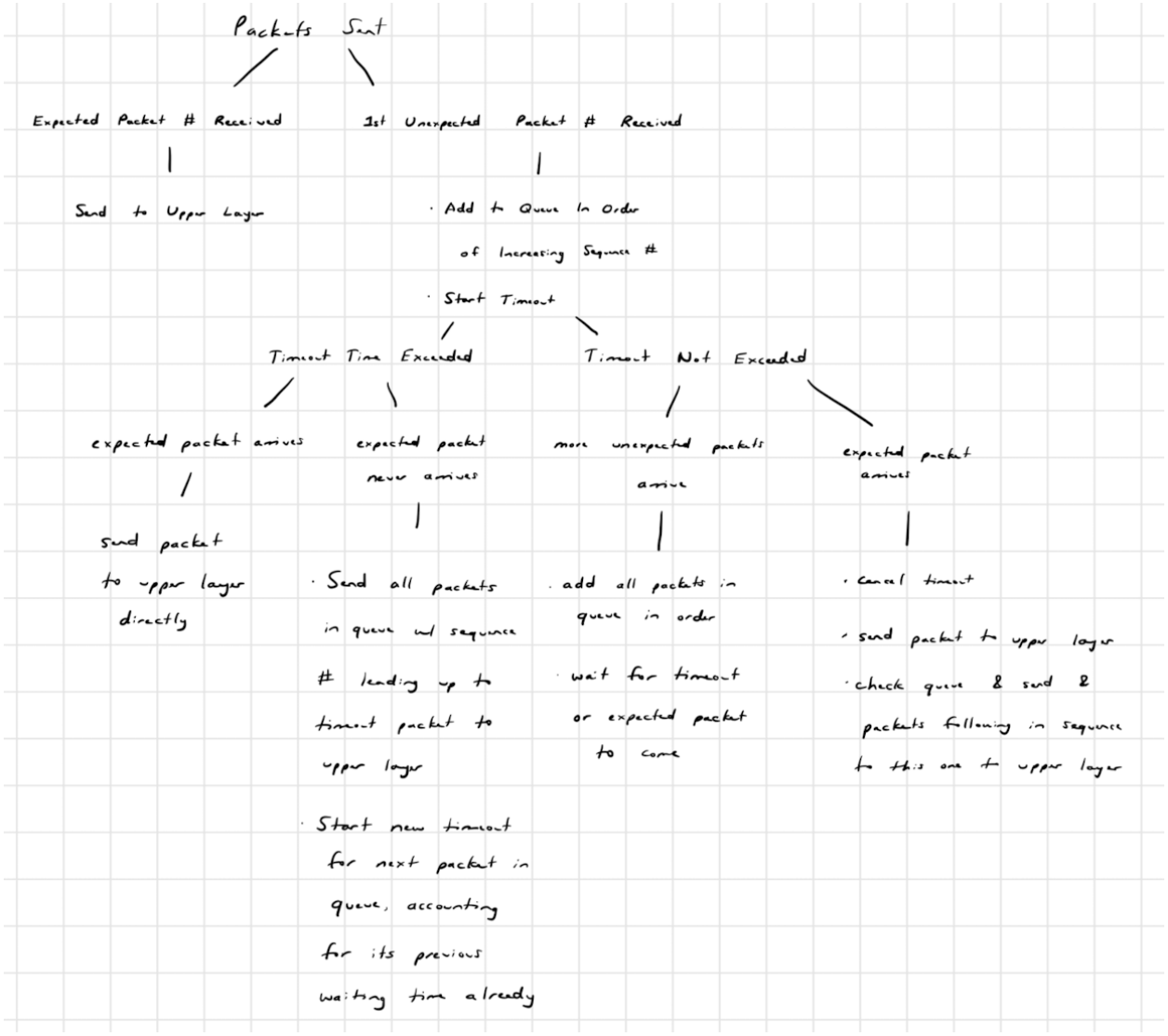
Problem 1

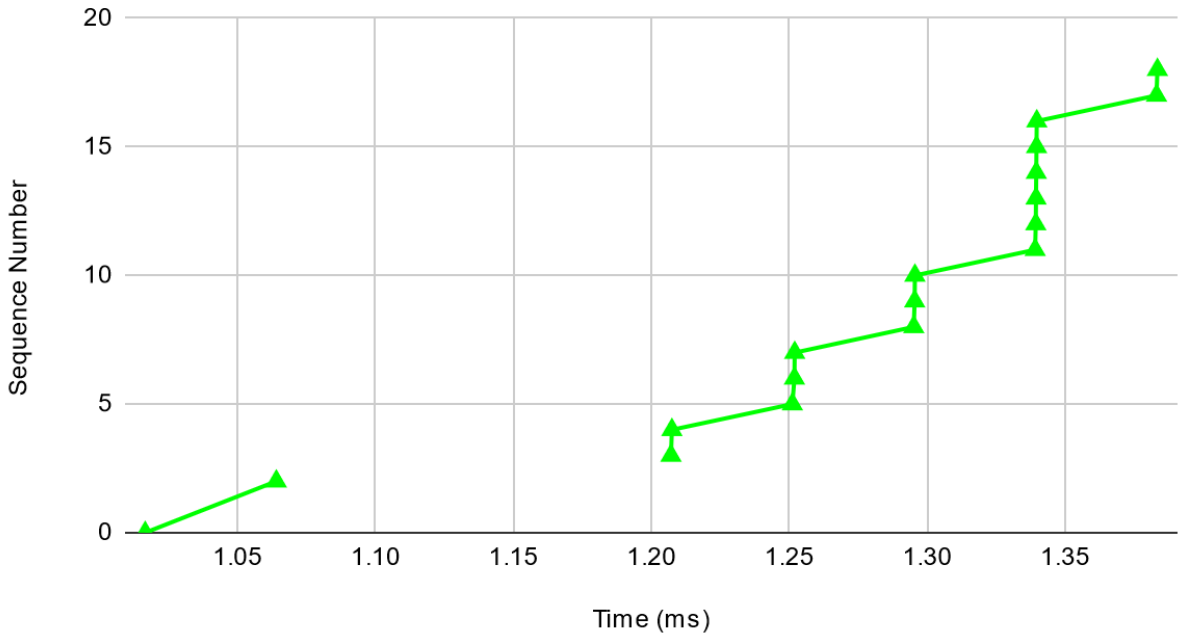
Flow Chart



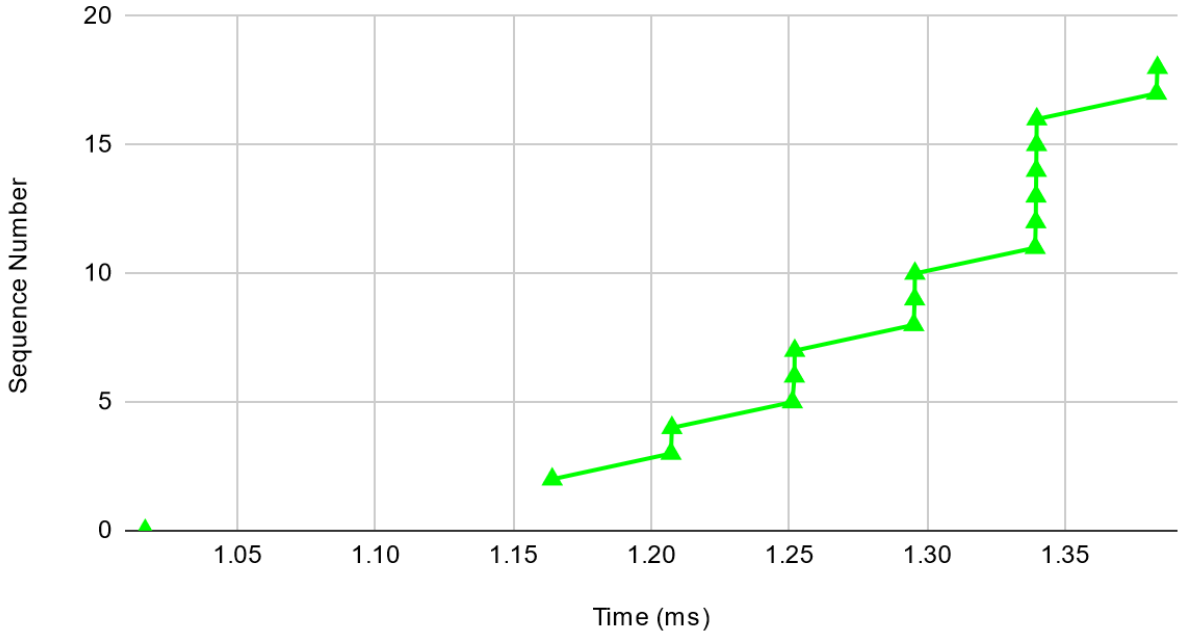
Validation

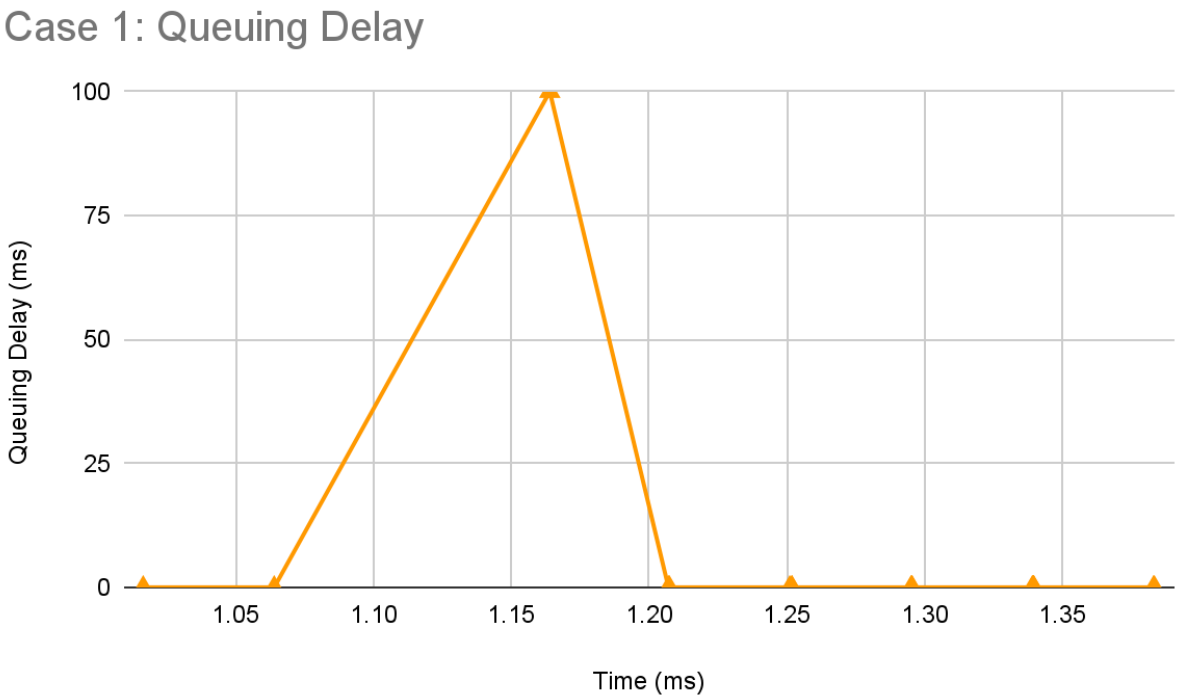
Scenario 1: Packet #1 never comes.
Expected behavior: Packets #2 and on are put into the queue. Eventually, timeout occurs after a default of 100ms. Then all packets in the queue are sent to the upper layer.

Case 1: Pre Sequence Numbers Sent



Case 1: Post Sequence Numbers Sent

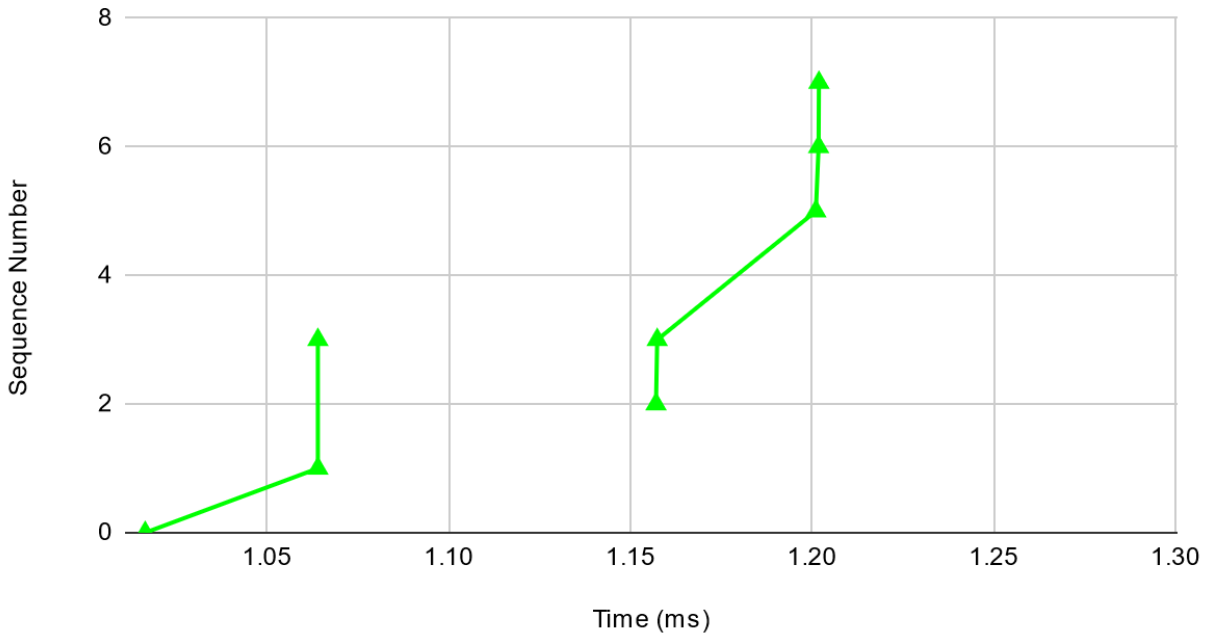




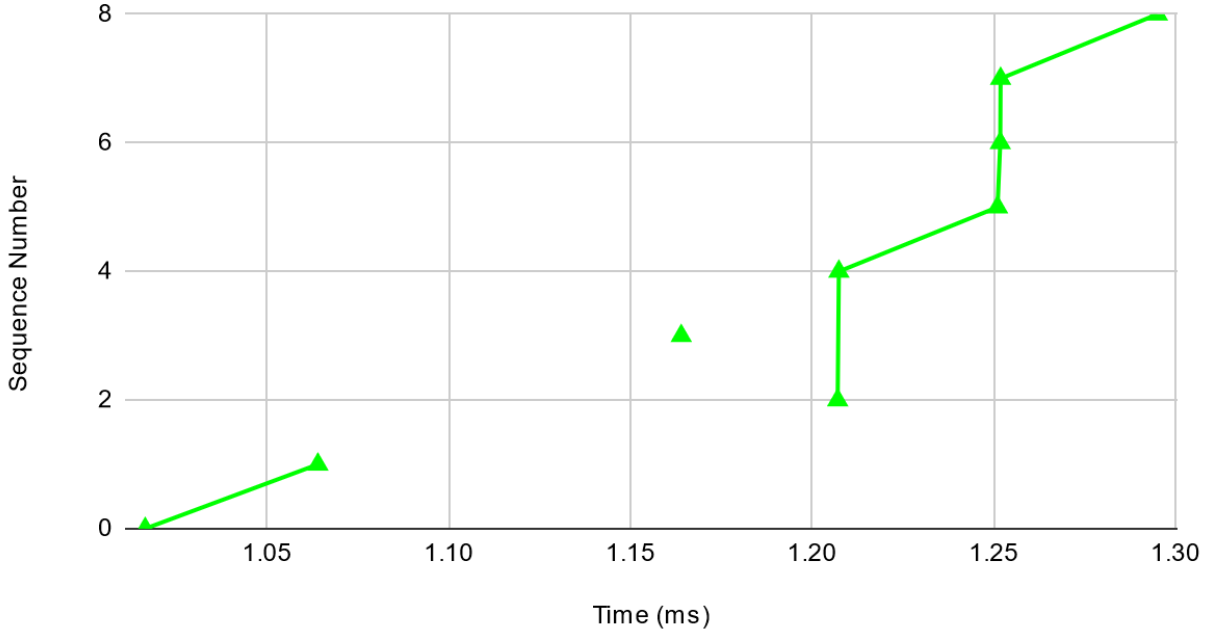
Observed behavior: Packet #1 is lost. Packet #2 arrives at 1.064. It arrives at the upper layer at 1.164ms: after the timeout of 100ms is complete. Then, all following packets are also sent to the upper layer. We can see that Packet #2 waits in the queue for 100 ms until timeout occurs. Following Packet #2, packets are delivered in order of sequence number to the upper layer.

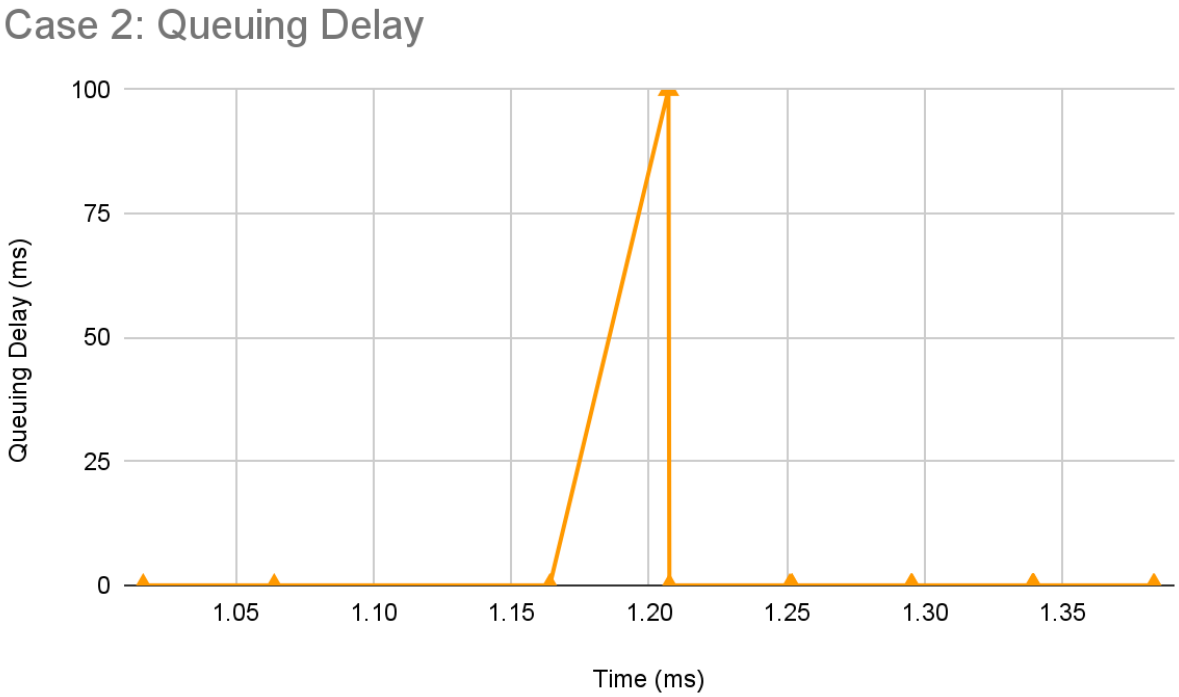
Scenario 2: Packet #2 arrives out of order but arrives after timeout.
Expected behavior: Packet #1 is delivered directly to the upper layer. Packet #3 is put into the queue once it arrives. After a default timeout of 100 ms, all packets in the queue are sent to the upper layer. When packet #2 finally arrives, it is also sent to the upper layer directly.

Case 2: Pre Sequence Numbers Sent



Case 2: Post Sequence Numbers Sent

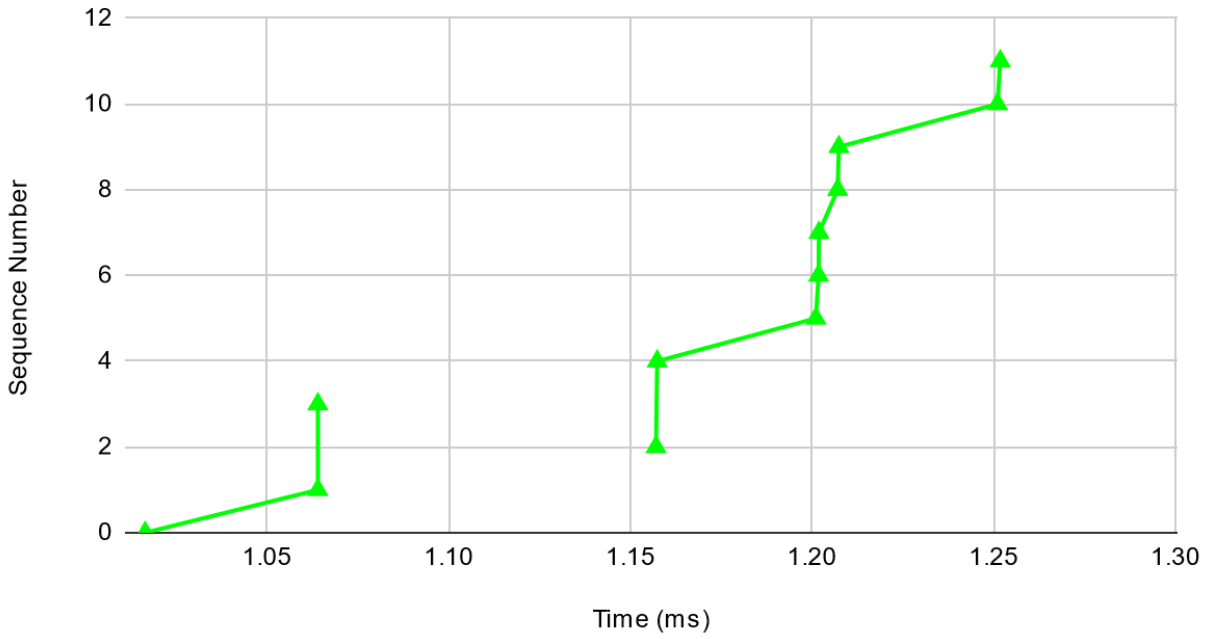




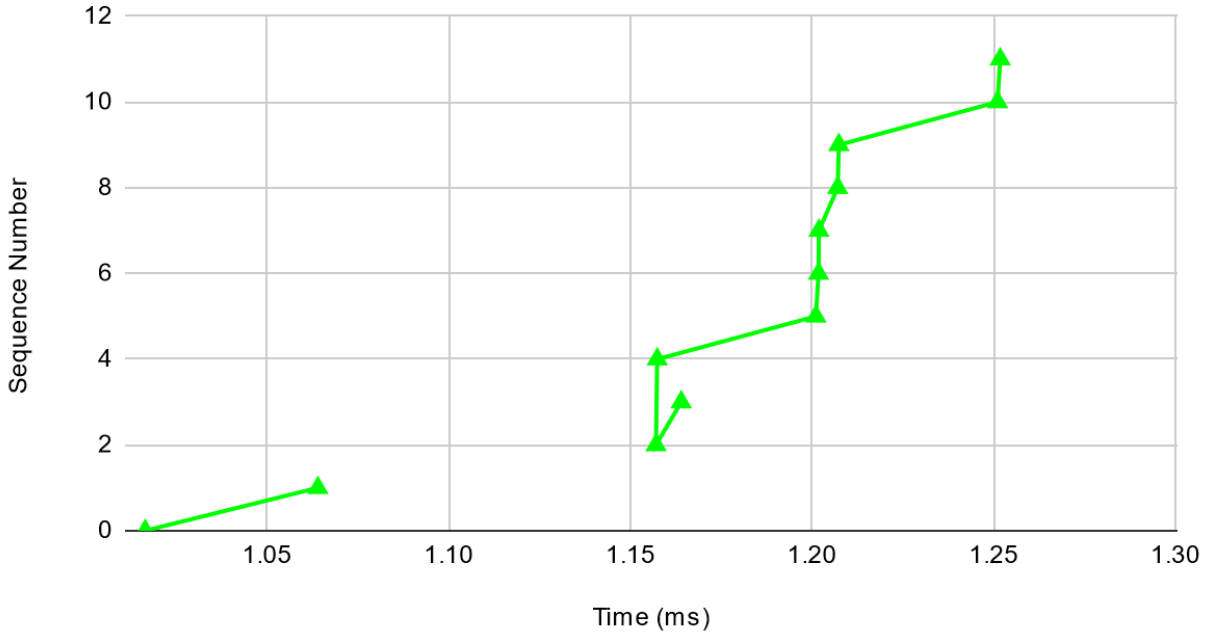
Observed behavior: Packet #3 is received before packet #2. Because of that, #3 is added to the queue. After a delay of 100ms in the queue, packet #3 is finally sent because the timeout is up. Packet #2 arrives after packet #3 is sent, and is also sent to the upper layer directly. We can see that the order of sequence numbers delivered to the upper layer goes: 1, 3, 2, 4, 5, 6, etc.

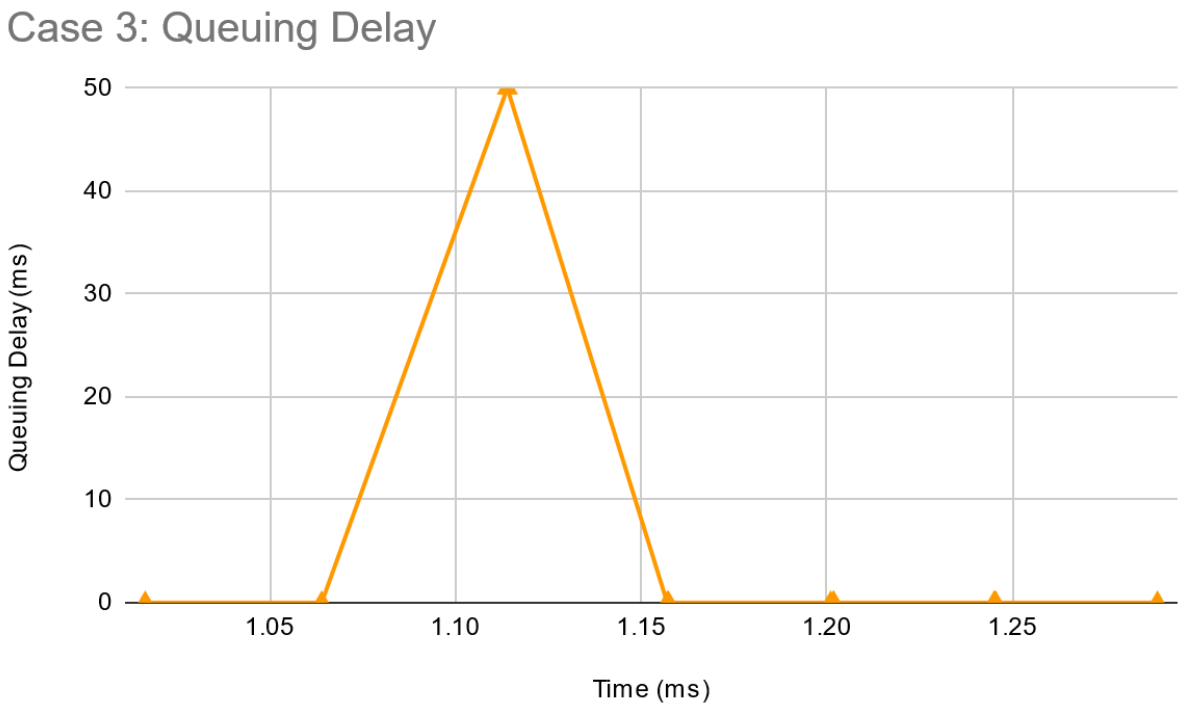
Scenario 3: We set inOrderTimeout = 50 ms instead of the default 100ms. Packet #2 arrives out of order but arrives after timeout.
Expected behavior: Packet #3 will be put into the queue for 50 ms. After timeout, it will get sent to the upper layer, along with other consecutive packets accumulated in the queue.

Case 3: Pre Sequence Numbers Sent



Case 3: Post Sequence Numbers Sent



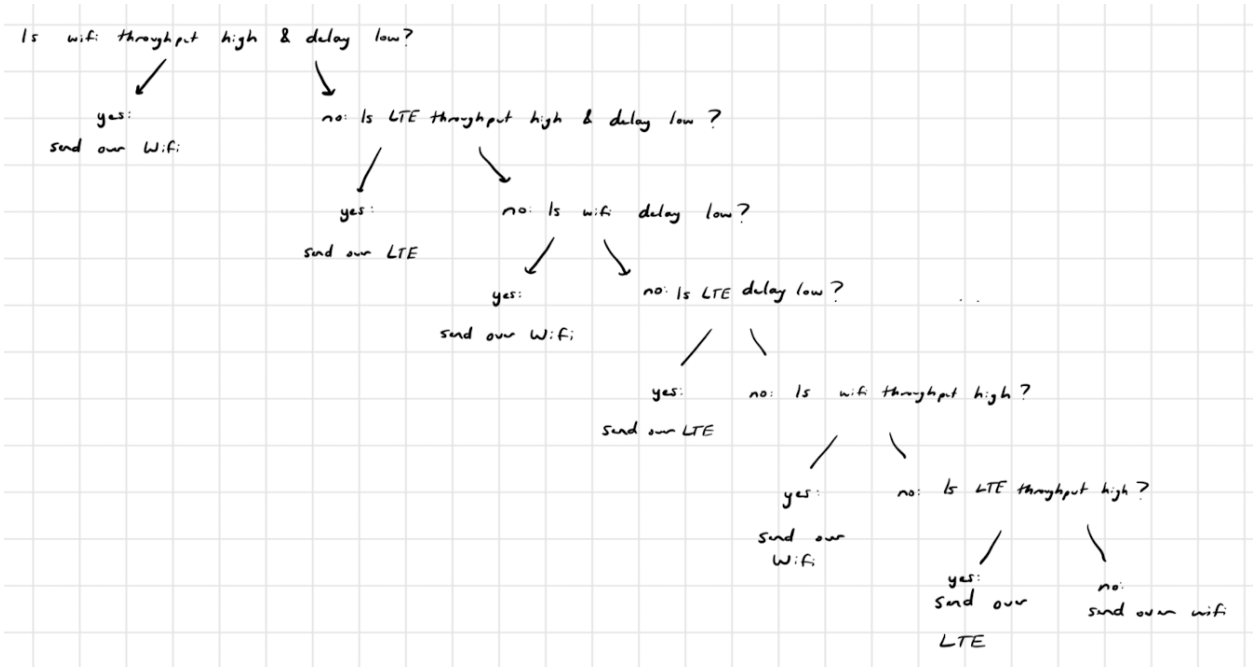


Observed behavior: Packet #3 is received before packet #2. Because of that, #3 is added to the queue. After a delay of 50ms in the queue, packet #3 is finally sent because the timeout is up. Packet #2 arrives after packet #3 is sent, and is also sent to the upper layer directly. We can see that the order of sequence numbers delivered to the upper layer goes: 1, 3, 2, 4, 5, 6, etc.

From these scenarios, we can verify that our in-packet ordering mechanism works as expected.

Problem 2

Flow Chart



1st Cycle Validation

Expected Performance and Algorithm Behavior

In our first iteration, we determined whether or not it would be better for the router to send packets along the Wi-Fi or LTE path by calculating scores for each path. To determine these two scores, we considered the delay along each path from router to UE and the throughput of each path in real time. We expected this algorithm to split packets evenly between Wi-Fi and LTE paths based on network conditions (differing RTTs, unstable LTE MCS values). For example, if the LTE path had a higher delay and lower throughput, the Wi-Fi path would be used more often, and vice versa. The performance of this algorithm was expected to be at least the performance if only the LTE or Wi-Fi path was used alone.

Simulations for Validation

See Appendix for command lines used. Testing on 1 UE.
The following baseline results will be used for validation #1 and validation #2. This is because these baseline measurements only consider the Wi-Fi only route, or the LTE only route. Thus, they are not affected by our aggregation algorithm.

Baseline Table 0A: Wi-Fi Only

Wifi RTT(ms)	RWND (Bytes)	MCS	Avg Wifi Throughput (Mbps)
30 ms	1024000	HtMcs7	72.73
200 ms	1024000	HtMcs7	17.78

Baseline Table 0B: LTE Only without MCS file

LTE RTT (ms)	RWND (Bytes)	Channel Bandwidth (MHz)	Avg LTE Throughput (Mbps)
30 ms	1024000	20	50
200 ms	1024000	20	16.67

Baseline Table 0C: LTE Only with MCS file

LTE RTT (ms)	RWND (Bytes)	Channel Bandwidth (MHz)	Avg LTE Throughput (Mbps)
30 ms	1024000	20	36.36
200 ms	1024000	20	13.56

These following measurements are only used for validation #1, as they test how well we aggregate between LTE and Wi-Fi routes with our algorithm.

Table 1A: Aggregated LTE and Wi-Fi without MCS File

LTE RTT(ms)	WiFi RTT(ms)	RWND (Bytes)	MCS	Channel Bandwidth (MHz)	Avg LTE Throughput (Mbps)	Avg Wifi Throughput (Mbps)	Total Avg Throughput (Mbps)
30 ms	30 ms	1024000	HtMcs7	20	0	114.29	160
200 ms	200 ms	1024000	HtMcs7	20	0	23.53	28.57
200 ms	30 ms	1024000	HtMcs7	20	0	38.10	47.06
30 ms	200 ms	1024000	HtMcs7	20	0	38.10	47.06

Table 1B: Aggregated LTE and Wi-Fi with MCS File

LTE RTT(ms)	WiFi RTT(ms)	RWND (Bytes)	MCS	Channel Bandwidth (MHz)	Avg LTE Throughput (Mbps)	Avg Wifi Throughput (Mbps)	Total Avg Throughput (Mbps)
30 ms	30 ms	1024000	HtMcs7	20	0	114.29	160
200 ms	200 ms	1024000	HtMcs7	20	0	23.53	28.57
200 ms	30 ms	1024000	HtMcs7	20	0	38.10	47.06
30 ms	200 ms	1024000	HtMcs7	20	100	80	100

Figure 1C_1: Expected vs Actual Throughput Without MCS File

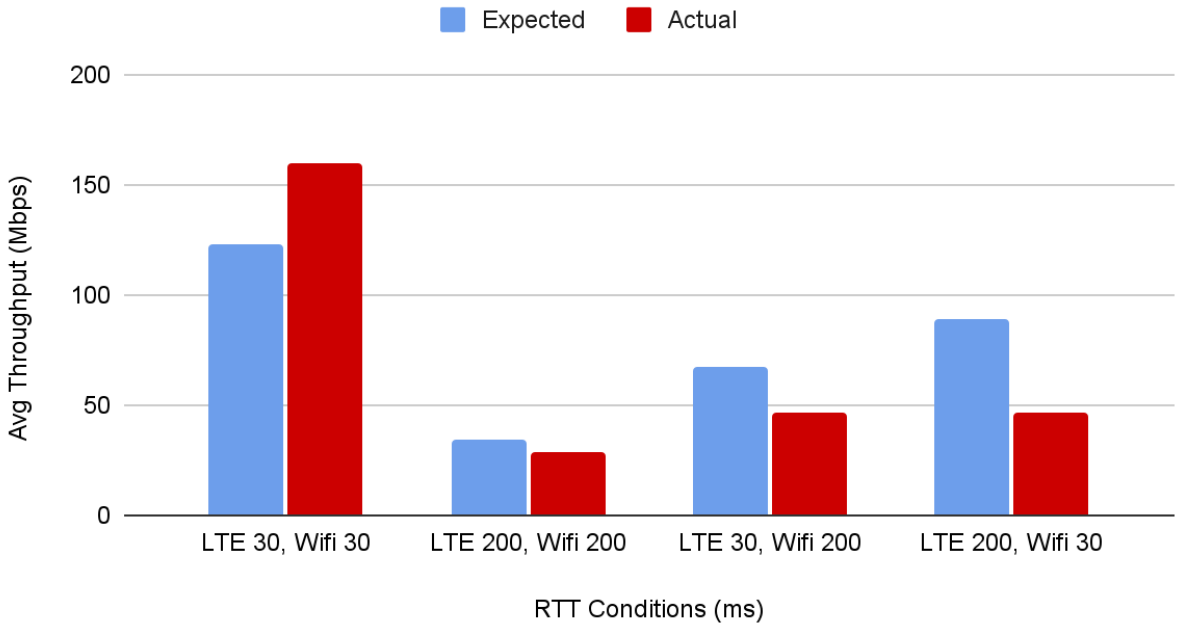
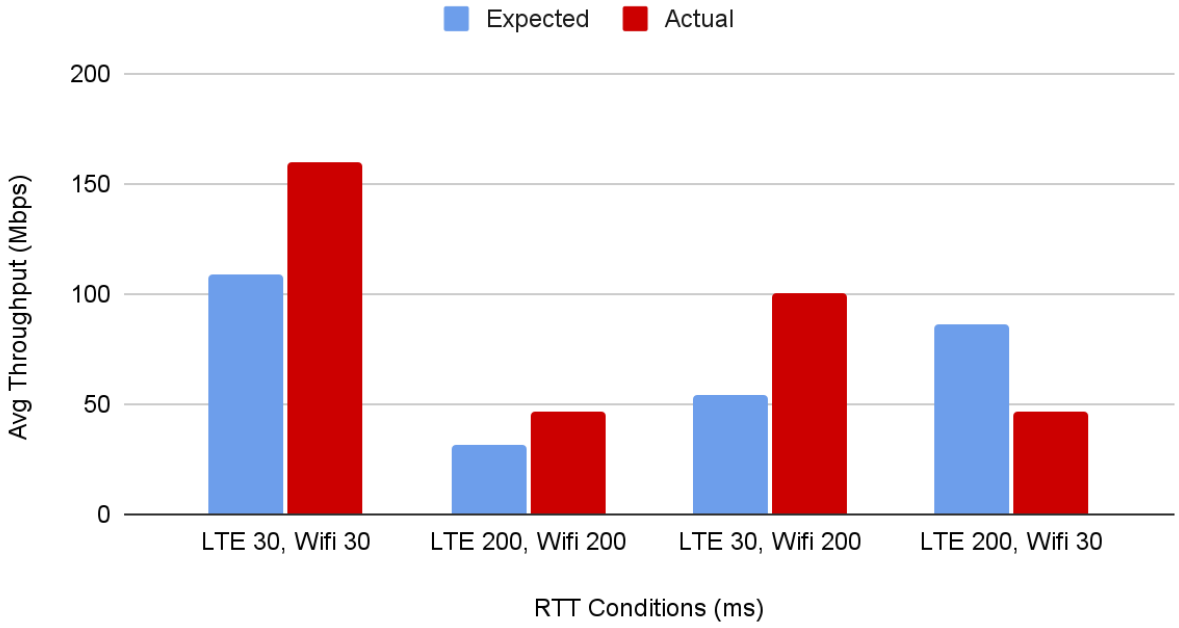


Figure 1C_2: Expected vs Actual Throughput With MCS File



Analysis of Algorithm Performance

After simulation, we observed some expected behavior under various network conditions. Our algorithm did not split the traffic between LTE and Wi-Fi at all; in reality, it only sent packets to the Wi-Fi path due to how we calculated the LTE and Wi-Fi scores. However, the actual throughput of scenarios tested met the expected throughput in certain simulations run.

Our reasoning behind the calculation of the LTE and Wi-Fi scores was to just take into account the delay and throughput along each path by subtracting them. If the throughput was really high, but the delay was really high as well, the difference between the two would account for whether the path was “worth” taking. We also multiplied this difference by a weight because we thought conditions would be more stable on the Wi-Fi path, and hence its score would always be better than the LTE score. In order to ensure the LTE path would be taken at least once, we made the Wi-Fi weight lower than the LTE one. As a result, when the difference between throughput and delay was multiplied with the weight, the LTE path would have a fair chance to be chosen.

This reasoning did not work as intended, and the algorithm performed somewhat poorly. When we actually executed the code, only the Wi-Fi path was chosen in most cases, except for the last trial where LTE RTT was 30 ms and Wi-Fi RTT was 200 ms with the MCS file. Other than that 1 trial, LTE average throughput was 0 Mbps, meaning the LTE path was not used much at all in other conditions tested, even when LTE RTT was 30 ms and Wi-Fi RTT was 200 ms without the MCS file.

When testing without the MCS file, we can see that our actual average throughput did not meet the expected average throughput (Figure 1C_1) except for the first scenario tested, where LTE and Wi-Fi RTT are both 30 ms. On the other hand, with the MCS file, we observe three out of the four cases meeting our expectations; the only case where the actual throughput is lower than the expected throughput is when LTE RTT is 200 ms and Wi-Fi RTT is 30 ms (Figure 1C_2).
From these observations, we can verify that while under some network conditions our algorithm performs as expected in terms of throughput, it does not evenly aggregate between Wi-Fi and LTE paths. This could be due to how we calculated the Wi-Fi score and LTE score. Due to the way we set up the algorithm, the LTE score would always end up being lower than the Wi-Fi score, no matter what network conditions there were. As a result, the LTE path would scarcely be chosen, because the LTE score never beats the Wi-Fi score oftenly. It might be best to scrap this calculation, because it was not considering each path fairly based on actual network conditions.

Furthermore, as we were running these conditions, we realized that we were not accurately calculating the throughput on each path. We calculated throughput by finding the size of each packet sent across a path, then dividing that by the amount of delay from router to UE across that path. Over time, we accumulated bytes of data sent, and were dividing by real-time delay. However, the definition of throughput does not depend on network delay from router to UE; it is

based on how many bytes of data you send over a certain period of time. Thus, we had to fix the way we were calculating throughput as well, because we weren't accounting for it correctly in this algorithm, which could also be why performance was so low.
(Note: The data above was obtained after fixing throughput calculations in the code, and so they are accurate as to what throughput actually was with the aggregation algorithm.)

From these observations, there were a few steps we had to take to enhance our current algorithm:

- Firstly, we must redesign how we calculate throughput on both paths in order to get real time network conditions accurately. We must collect how many bytes of data we receive on the UE end from each path within a given period of time.
- Secondly, we must redesign our scoring system for Wi-Fi and LTE so that it actually takes into account throughput and network delay with more sound reasoning than just a difference. Perhaps setting the score based on whether certain network conditions were met would be a better idea.

2nd Cycle Validation

Expected Performance and Algorithm Behavior

In our second iteration, we again determined whether or not it would be better for the router to send packets along the Wi-Fi or LTE path by calculating scores for each path. To determine these two scores, we considered the delay along each path from router to UE and the throughput of each path in real time. If the delay was low and throughput was high for a certain path, we were more likely to choose that path. Otherwise, we prioritized lower delays over higher throughputs when calculating our scores (i.e. if a path had a lower delay time but also high throughput, we would choose it over the path with higher delay time). We also resorted to Wi-Fi as the default path in the case where LTE and Wi-Fi conditions were similar. Thus, we expected this algorithm to split packets more towards the Wi-Fi path if network conditions were similar between LTE and Wi-Fi, however more evenly between Wi-Fi and LTE paths if network conditions differed greatly and it would be more advantageous to use the LTE path. For example, if the Wi-Fi path had a higher delay and lower throughput, the LTE path would be used more often, and vice versa. The performance of this algorithm was expected to be at least the performance if only the LTE or Wi-Fi path was used alone, if not double the performance.

Simulations for Validation

See Appendix for command lines used. Testing on 1 UE.
Baseline measurements from Validation #1 will be used for analysis.

Table 2A: Aggregated LTE and Wi-Fi without MCS File

LTE RTT(ms)	WiFi RTT(ms)	RWND (Bytes)	MCS	Channel Bandwidth (MHz)	Avg LTE Throughput (Mbps)	Avg Wifi Throughput (Mbps)	Avg Total Through

							put (Mbps)
30 ms	30 ms	1024000	HtMcs7	20	160	114.29	160
200 ms	200 ms	1024000	HtMcs7	20	30.77	23.53	28.57
200 ms	30 ms	1024000	HtMcs7	20	0	38.10	47.06
30 ms	200 ms	1024000	HtMcs7	20	114.29	0	133.33

Table 2B: Aggregated LTE and Wi-Fi with MCS File

LTE RTT(ms)	WiFi RTT(ms)	RWND (Bytes)	MCS	Channel Bandwidth (MHz)	Avg LTE Throughput (Mbps)	Avg Wifi Throughput (Mbps)	Avg Total Throughput (Mbps)
30 ms	30 ms	1024000	HtMcs7	20	133.33	144.29	133.33
200 ms	200 ms	1024000	HtMcs7	20	30.77	23.53	28.57
200 ms	30 ms	1024000	HtMcs7	20	0	38.10	47.06
30 ms	200 ms	1024000	HtMcs7	20	88.89	80	100

The following figures were made from results run with the MCS file.

Figure 2C_1: Bits Sent vs Time for LTE & Wifi RTT 30ms

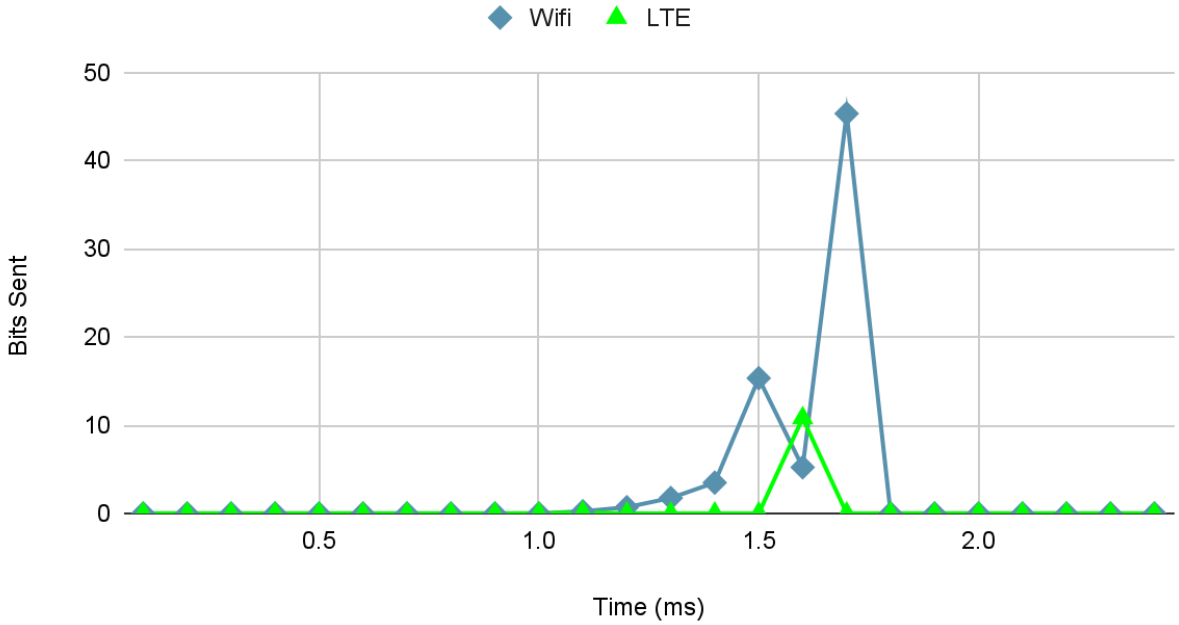


Figure 2C_2: Bits Received vs Time for LTE & Wifi RTT 30 ms

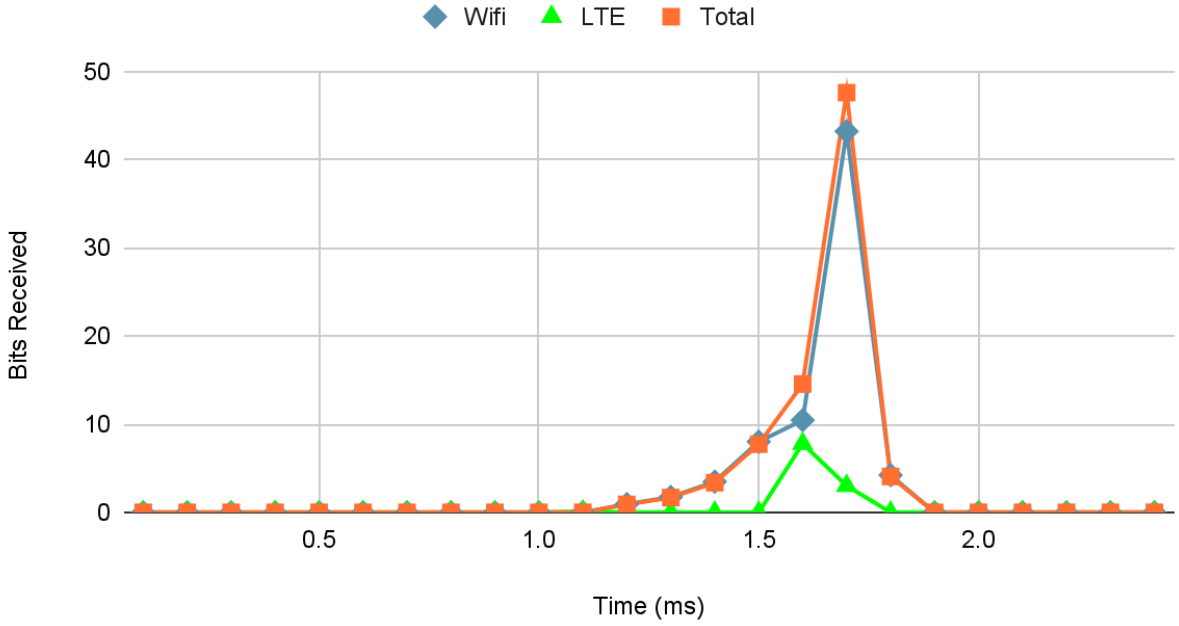


Figure 2D_1: Bits Sent vs Time for LTE &Wifi RTT 200ms

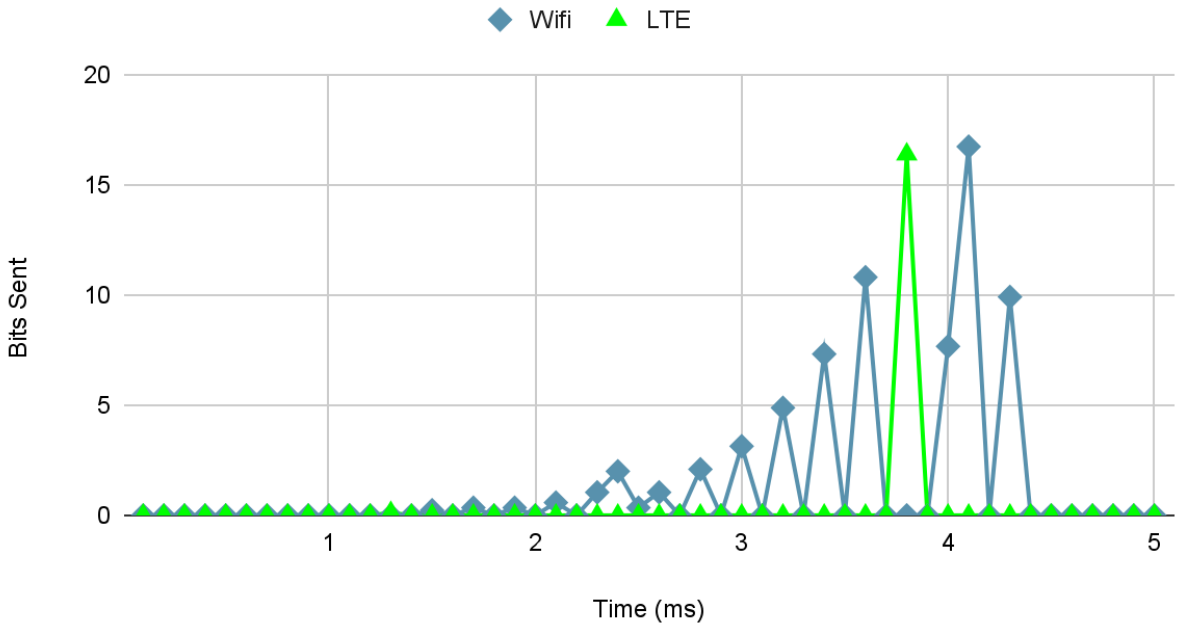


Figure 2D_2: Bits Received vs Time for Wifi & LTE RTT 200 ms

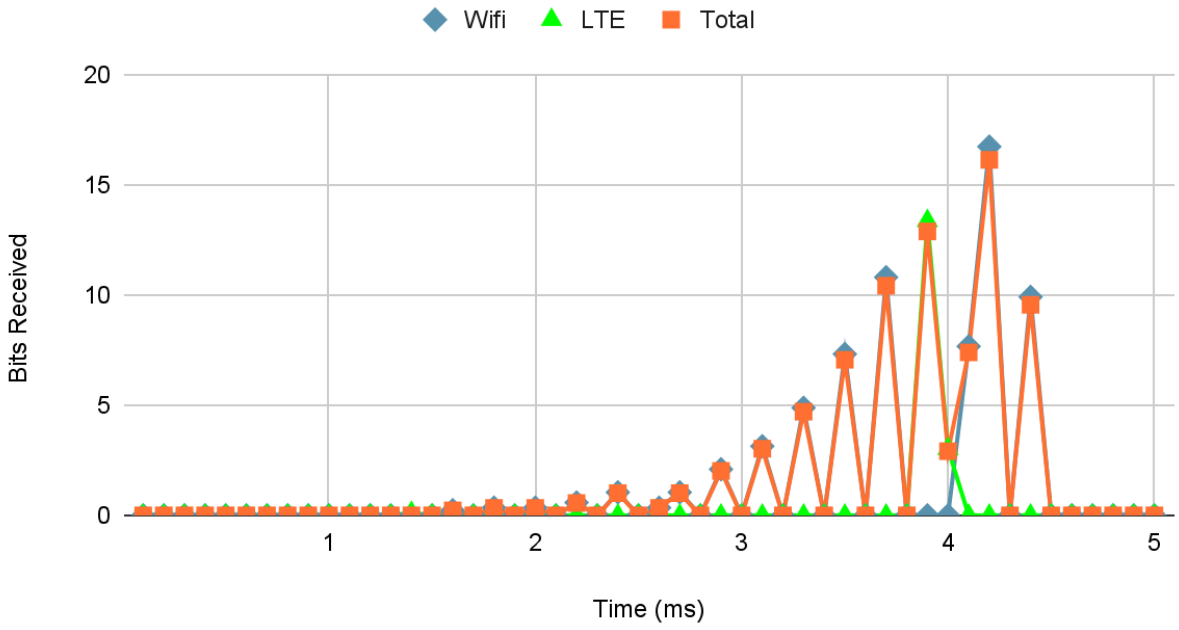


Figure 2E_1: Bits Sent vs Time for LTE RTT 200ms & Wifi RTT 30ms

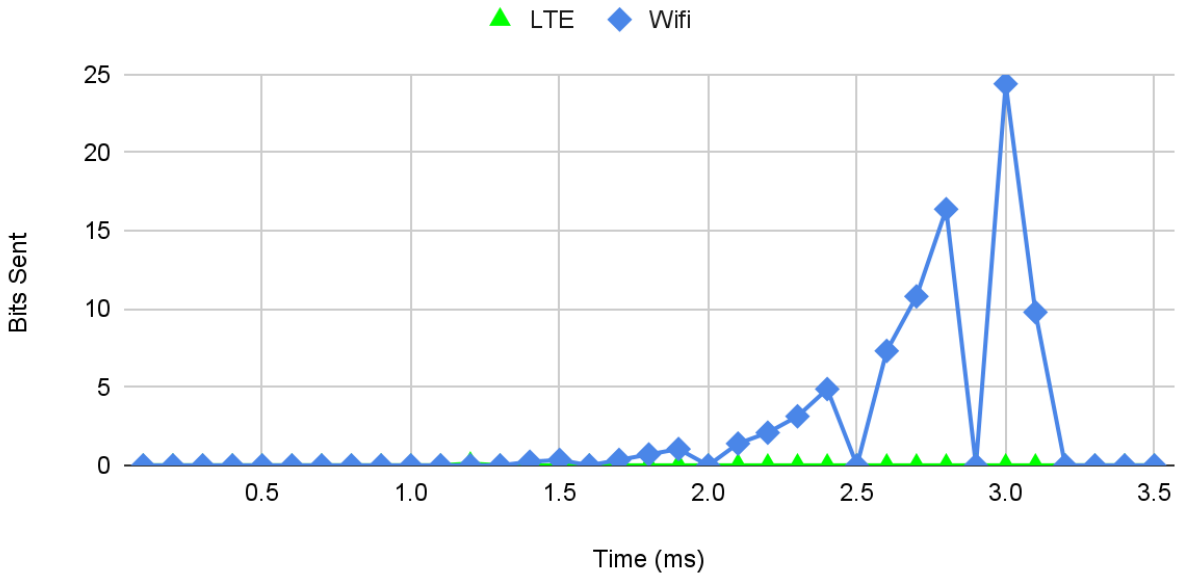


Figure 2E_2: Bits Received vs Time for LTE RTT 200ms & Wifi RTT 30ms

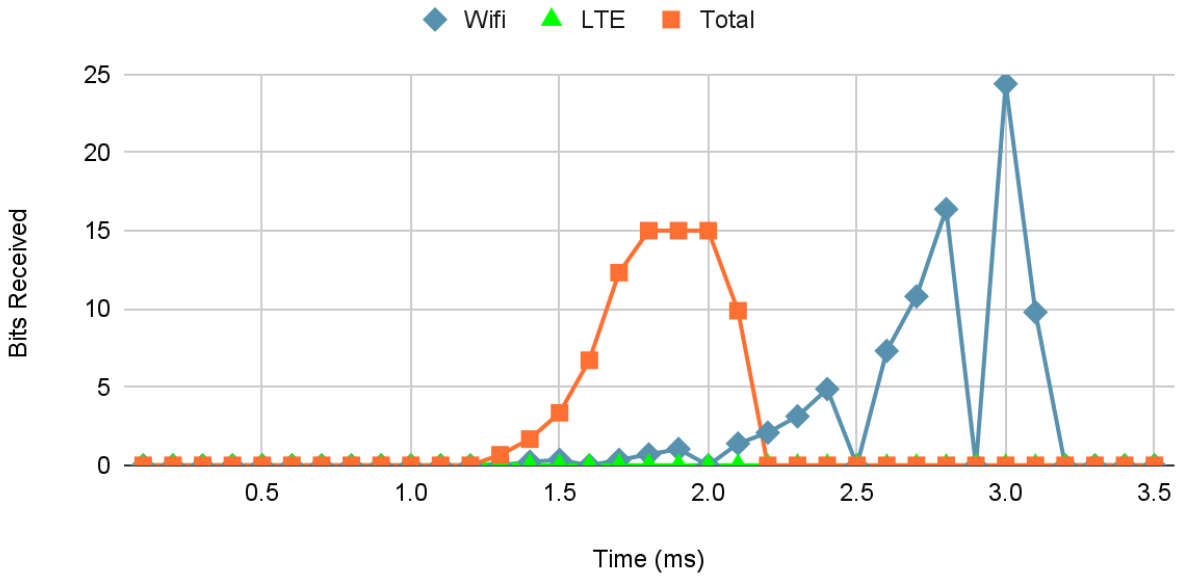


Figure 2F_1: Bits Sent vs Time for LTE RTT 30ms & Wifi RTT 200ms

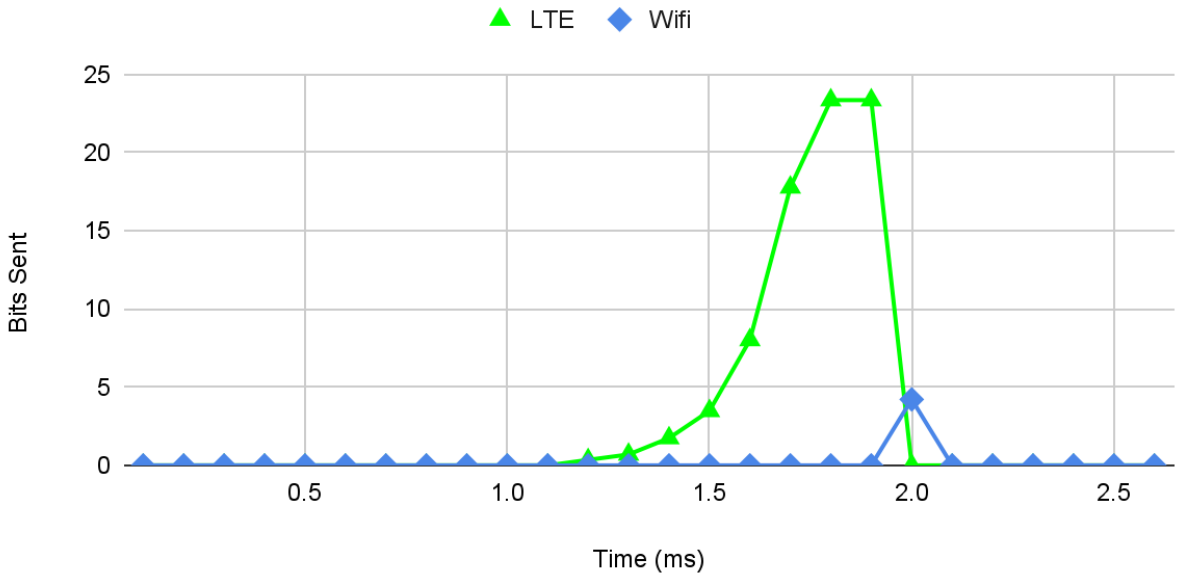


Figure 2F_2: Bits Received vs Time for LTE RTT 30ms & Wifi RTT 200ms

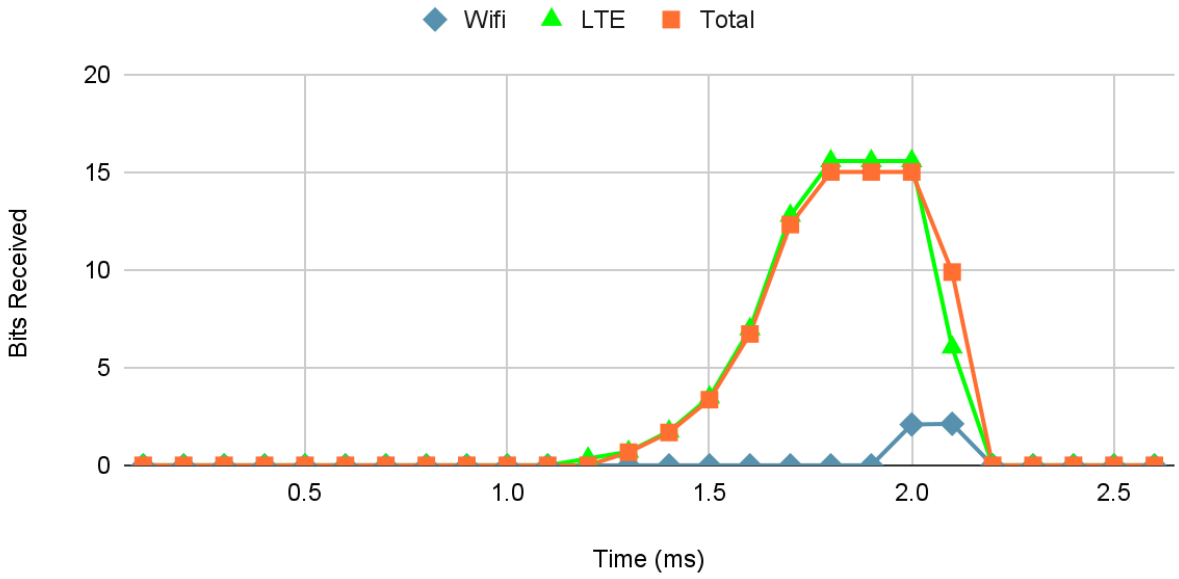


Figure 2G_1: Expected vs Actual Throughput without MCS File

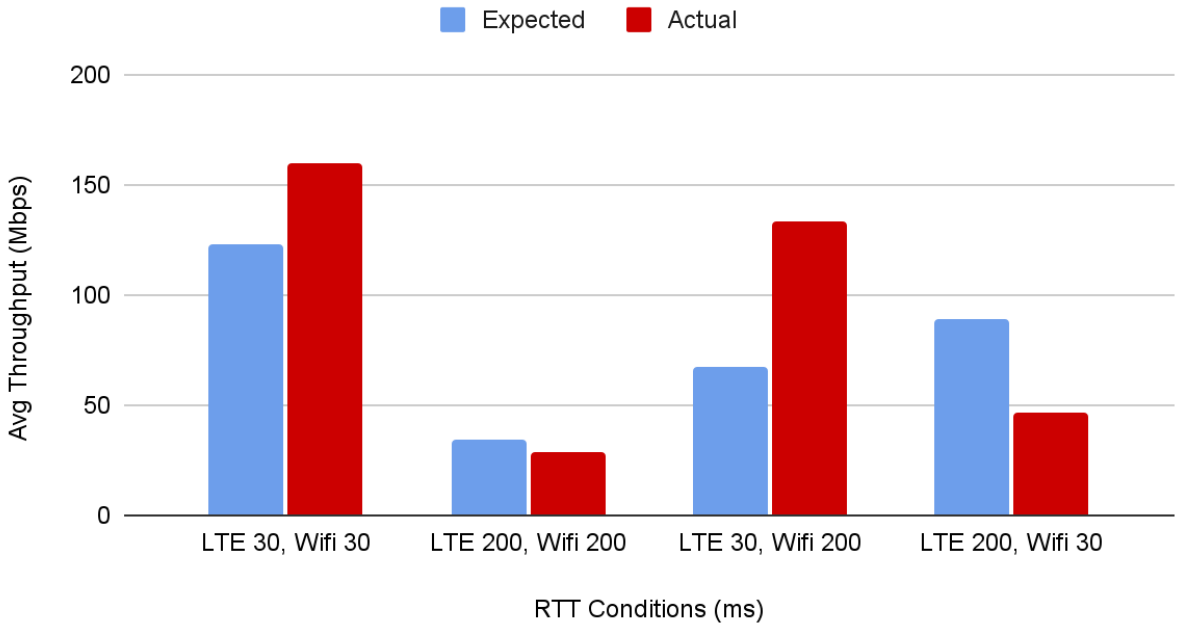
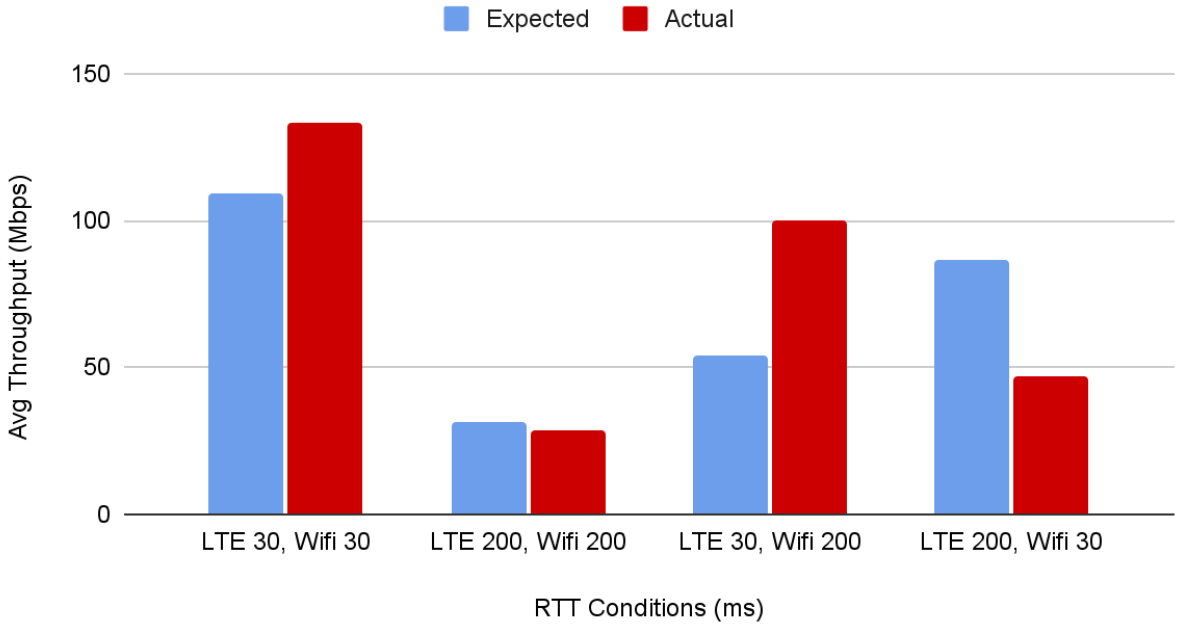


Figure 2G_2: Expected vs Actual Throughput with MCS File



Analysis of Algorithm Performance

After simulation, we observed some expected behavior under various network conditions. Our algorithm did split the traffic between LTE and Wi-Fi depending on which path had a lower RTT, and the actual throughput of some scenarios tested met the expected throughput.

Let us observe the case where LTE RTT is 200 ms and Wi-Fi RTT is 30 ms. From Figure 2E_1, which shows the input rate along the LTE and Wi-Fi paths, we can see that more traffic is being sent towards the Wi-Fi path. Since our algorithm prioritizes lower delay when choosing a path to take, it makes sense that we observe more bits of data being sent along the Wi-Fi route. In Figure 2E_2, we also see that more data is being received on the Wi-Fi end of the tunnel at the UE, which further aligns with our expectation of the algorithm behavior. On the other hand, when Wi-Fi RTT is 200 ms and LTE RTT is 30 ms, we can see that more traffic gets sent to the LTE path from Figure 2F_1. Again, since the LTE path has a lower delay, our algorithm favors that path and sends more traffic towards it. In Figure 2F_2, we also see that more data is being received on the LTE end of the tunnel at the UE, which aligns with our expectation of the algorithm behavior. In terms of aggregation methods, the results obtained from simulation are what we expect to happen: whichever RTT is lower gets more traffic.

In the case that Wi-Fi and LTE have similar network conditions (i.e. the same delay), we observe that both LTE and Wi-Fi are utilized, however Wi-Fi is utilized much more often (Figure 2C_1 and Figure 2D_1). This is because we default to the Wi-Fi path if Wi-Fi and LTE have similar conditions. Therefore, if Wi-Fi and LTE have similar delay values, we will still choose the Wi-Fi path over the LTE path, explaining why throughput on the Wi-Fi end is higher than LTE throughput (Figure 2C_2 and Figure 2D_2) when Wi-Fi RTT equals LTE RTT.

When testing without the MCS file, we can see that our actual average throughput did not meet the expected average throughput (Figure 2G_1) in cases where LTE RTT is 200ms. The same could be observed with results run with the MCS file as well (Figure 2G_2). For some reason, our actual throughput falls short of the expected throughput when LTE RTT is 200ms, both in stable LTE MCS conditions and unstable LTE MCS conditions. This could be due to how our algorithm calculates the LTE score. If LTE throughput is high and LTE delay is low, the LTE score will be high. However if LTE delay is high, even if the LTE throughput is high, the algorithm will assign the LTE score to be lower. Thus a higher LTE RTT will always result in a lower LTE score, despite LTE throughput obtained. This may result in less traffic being aggregated towards the LTE path in general, even when it is more efficient to send packets that way. Our algorithm simply waits to send packets towards the Wi-Fi path despite the LTE path being clear resulting in a lower actual throughput than expected.

From these observations, we can verify that our algorithm performs somewhat well under certain network conditions. The lower the RTT on Wi-Fi and LTE paths, the better. If one path has a longer RTT, it will not be utilized as much as the other path. Furthermore, it seems that unstable MCS conditions for the LTE does not greatly impact the performance of the algorithm–the results are similar whether the MCS conditions are stable or not. The algorithm is dynamically able to choose between Wi-Fi and LTE paths depending on network conditions and

aggregate between them–at least better than our first cycle validation algorithm, where the LTE path would not be used at all in certain network conditions. Still, though, there are many more optimizations that we could make to our algorithm to ensure even better performance if we had more time.

In order to further enhance our algorithm, we can also consider the traffic input bit rate measurement at the router toward LTE and Wi-Fi paths with our algorithm, besides the real-time throughput (bps) and the one-way packet delay (ms) measurement at the UE for both LTE and Wi-Fi path. We could use this input bit rate measurement to determine whether or not there was an even load between LTE and Wi-Fi paths; if one was being used more than the other, the less used path would be taken into consideration more often. Furthermore, we could come up with a better ranking system to determine LTE and Wi-Fi scores. Instead of assigning high scores based on only the path RTT, we could assign higher scores based on throughput as well. For example, maybe the algorithm can switch between assigning higher scores based on delay and throughput–for 100 ms, it assigns a higher score if the delay recorded is low, and then for the next 100 ms, it assigns a higher score if the throughput recorded is high. We can integrate these properties instead of statically relying on an ordering system that favors lower delay at all times.

Problem 3

DL using Wi-Fi: circular path

RTT (ms)	Number of UEs	Average Throughput (Mbps)
30	15	16.327
200	15	6.452
30	5	33.333
200	5	13.559

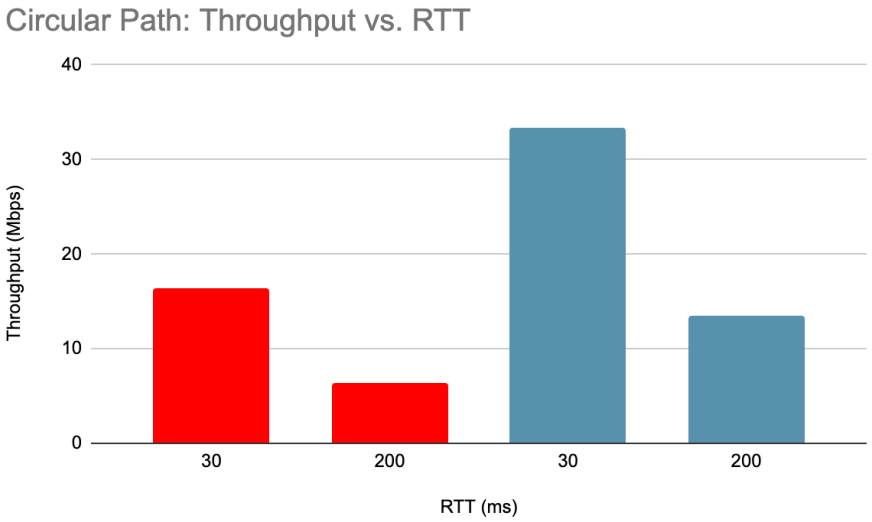
DL and UL using Wi-Fi

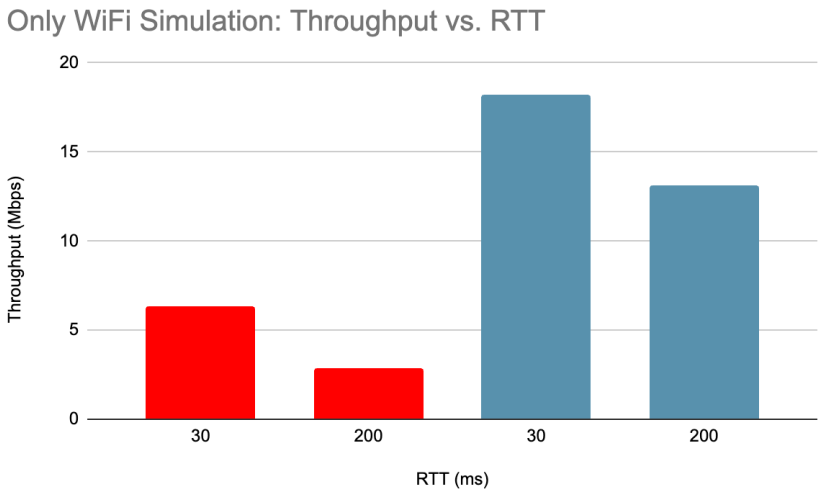
RTT (ms)	Number of UEs	Average Throughput (Mbps)
30	15	6.349
200	15	2.837
30	5	18.182
200	5	13.115

Consistent observation across both simulations: as RTT increases, throughput decreases.

- Specifically considering average throughput (as this is a better indicator for the average throughput performance across the entire TCP session) it can be clearly observed from above that increasing RTT for all scenarios (only WiFi or sending UL traffic across LTE) decreases the throughput.
    - Notable differences can be seen for instance in the 15 UE case using the circular path simulation. Increasing the RTT from 30 ms to 200 ms shows that the average throughput drops notably from 16.327 Mbps to 6.452 Mbps.
    - A similar trend can be observed in the same simulation for the 5 UE simulation scenario.

To further exemplify the above observation, consider the following bar charts where the red blocks represents 15 UE scenarios in the simulations and the navy blocks represent 5 UE scenarios in the simulations:





Circular Path average throughput values were consistently higher across all RTT delays:

- In the circular path scenario, where UL traffic was sent across LTE, we can observe in the above bar chart that the average throughput values were higher than for the WiFi only simulation.
- As mentioned in the lecture notes, WiFi uplink congestion is a large issue in terms of performance given the number of devices that can share a particular WiFi network. Hence, offloading some of the uplink congestion to LTE ameliorates this issue as seen above.
- Furthermore, LTE networks are typically optimized to more efficiently transmit uplink, leading to particularly better UL throughput compared to WiFi networks.
- Overall, this observation supports the notion of network load balancing: if the characteristics of the UL and DL traffic are different, routing them over networks optimized for their respective directions can lead to better overall performance.

Circular Path simulation average throughput experienced more dramatic declines when RTT increased:

- This is more of a nuanced observation, however, it is worthwhile to notice that even though the circular path network simulation saw higher throughput values in comparison to the WiFi only scenario, it also observed larger declines in throughput when the RTT was increased from 30 ms to 200 ms.
- There is not a single reason for why this occurred:
    - If your simulation follows a circular path, the signals in the LTE network have to travel a greater distance, leading to longer delays that can affect how fast data moves. Especially when there are longer round-trip times (RTT), this delay can noticeably impact the data transfer speed.
    - Similarly, the circular simulation might come across various LTE cells, which could result in handovers or switching between cells. These transitions can briefly disrupt the connection, impacting how quickly data is transferred. The smoothness of the handover process and the network's capability to smoothly

switch between cells can influence the overall performance and could be a reason for why there was a more dramatic decline in average throughput.

Appendix

Problem 1 Validation

Case 1

```
./waf --run "scratch/prj4_px_1-1 --OutputFileName=output/prj
--Scenario=3 --NumberUE=1 --Transport=1 --wifiMcs=HtMcs7
--tcpRcvBufBytes=1024000 --DataSizeforTCP=1000000 --
delayValueforRHtoR=5 --delayValueforLte=10 --delayValueforWifi=10
--chBwMHz=20 --simTime=7 --aggPath=lteAndWifi"
```

Case 2

```
./waf --run "scratch/prj4_px_1-2 --OutputFileName=output/prj
--Scenario=3 --NumberUE=1 --Transport=1 --wifiMcs=HtMcs7
--tcpRcvBufBytes=1024000 --DataSizeforTCP=1000000 --
delayValueforRHtoR=5 --delayValueforLte=10 --delayValueforWifi=10
--chBwMHz=20 --simTime=7 --aggPath=lteAndWifi"
```

Case 3

```
./waf --run "scratch/prj4_px_1-2 --OutputFileName=output/prj
--Scenario=3 --NumberUE=1 --Transport=1 --wifiMcs=HtMcs7
--tcpRcvBufBytes=1024000 --DataSizeforTCP=1000000 --
delayValueforRHtoR=5 --delayValueforLte=10 --delayValueforWifi=10
--chBwMHz=20 --simTime=7 --inOrderTimeout=50 --aggPath=lteAndWifi"
```

Problem 2 Validation

Baseline Wifi only conditions

```
./waf --run "scratch/prj4_px
--OutputFileName=output/prj4-1_wifi_only_rtt30 --Scenario=1
--NumberUE=1 --Transport=1 --wifiMcs=HtMcs7 --tcpRcvBufBytes=1024000
--DataSizeforTCP=10000000 --delayValueforRHtoR=5
--delayValueforWifi=10  --simTime=30"

./waf --run "scratch/prj4_px
--OutputFileName=output/prj4-1_wifi_only_rtt200 --Scenario=1
--NumberUE=1 --Transport=1 --wifiMcs=HtMcs7 --tcpRcvBufBytes=1024000
--DataSizeforTCP=10000000 --delayValueforRHtoR=10
--delayValueforWifi=90  --simTime=30"
```

Baseline LTE only conditions

```
./waf --run "scratch/prj4_px
--OutputFileName=output/prj4-1_lte_only_rtt30 --Scenario=2
--NumberUE=1 --Transport=1 --DataSizeforTCP=10000000
--tcpRcvBufBytes=1024000 --delayValueforRHtoR=5
--delayValueforLte=10 --chBwMHz=20 --simTime=20"

./waf --run "scratch/prj4_px
--OutputFileName=output/prj4-1_lte_only_rtt200 --Scenario=2
--NumberUE=1 --Transport=1 --DataSizeforTCP=10000000
--tcpRcvBufBytes=1024000 --delayValueforRHtoR=10
--delayValueforLte=90 --chBwMHz=20 --simTime=20"
```

Validation #1: Aggregated Conditions

```
# wifi RTT and LTE RTT 30ms
./waf --run "scratch/prj4-1_px
--OutputFileName=output/prj4-1_agre_rtt_30 --Scenario=3 --NumberUE=1
--Transport=1 --wifiMcs=HtMcs7 --tcpRcvBufBytes=1024000
--DataSizeforTCP=1000000 --delayValueforRHtoR=5 --delayValueforLte=10
--delayValueforWifi=10 --chBwMHz=20 --simTime=7 --aggPath=lteAndWifi"

# wifi RTT and LTE RTT 200ms
./waf --run "scratch/prj4-1_px
--OutputFileName=output/prj4-1_agre_rtt_200 --Scenario=3 --NumberUE=1
--Transport=1 --wifiMcs=HtMcs7 --tcpRcvBufBytes=1024000
--DataSizeforTCP=1000000 --delayValueforRHtoR=10
--delayValueforLte=90 --delayValueforWifi=90 --chBwMHz=20 --simTime=7
--aggPath=lteAndWifi"

# wifi RTT 30 and LTE RTT 200ms
./waf --run "scratch/prj4-1_px
--OutputFileName=output/prj4-1_agre_wifi_rtt_30_lte_rtt_200
--Scenario=3 --NumberUE=1 --Transport=1 --wifiMcs=HtMcs7
--tcpRcvBufBytes=1024000 --DataSizeforTCP=1000000
--delayValueforRHtoR=10 --delayValueforLte=90 --delayValueforWifi=5
--chBwMHz=20 --simTime=7 --aggPath=lteAndWifi"

# LTE RTT 30 and wifi RTT 200ms
./waf --run "scratch/prj4-1_px
--OutputFileName=output/prj4-1_agre_lte_rtt_30_wifi_rtt_200
--Scenario=3 --NumberUE=1 --Transport=1 --wifiMcs=HtMcs7
--tcpRcvBufBytes=1024000 --DataSizeforTCP=1000000
```

```
--delayValueforRHtoR=10 --delayValueforLte=5 --delayValueforWifi=90
--chBwMHz=20 --simTime=7 --aggPath=lteAndWifi"
```

Validation #2: Aggregated Conditions

```
# wifi RTT and LTE RTT 30ms
./waf --run "scratch/prj4_px
--OutputFileName=output/prj4-2_agre_rtt_30 --Scenario=3 --NumberUE=1
--Transport=1 --wifiMcs=HtMcs7 --tcpRcvBufBytes=1024000
--DataSizeforTCP=1000000 --delayValueforRHtoR=5 --delayValueforLte=10
--delayValueforWifi=10 --chBwMHz=20 --simTime=7 --aggPath=lteAndWifi"

# wifi RTT and LTE RTT 200ms
./waf --run "scratch/prj4_px
--OutputFileName=output/prj4-2_agre_rtt_200 --Scenario=3 --NumberUE=1
--Transport=1 --wifiMcs=HtMcs7 --tcpRcvBufBytes=1024000
--DataSizeforTCP=1000000 --delayValueforRHtoR=10
--delayValueforLte=90 --delayValueforWifi=90 --chBwMHz=20 --simTime=7
--aggPath=lteAndWifi"

# wifi RTT 30 and LTE RTT 200ms
./waf --run "scratch/prj4_px
--OutputFileName=output/prj4-2_agre_wifi_rtt_30_lte_rtt_200
--Scenario=3 --NumberUE=1 --Transport=1 --wifiMcs=HtMcs7
--tcpRcvBufBytes=1024000 --DataSizeforTCP=1000000
--delayValueforRHtoR=10 --delayValueforLte=90 --delayValueforWifi=5
--chBwMHz=20 --simTime=7 --aggPath=lteAndWifi"

# LTE RTT 30 and wifi RTT 200ms
./waf --run "scratch/prj4_px
--OutputFileName=output/prj4-2_agre_lte_rtt_30_wifi_rtt_200
--Scenario=3 --NumberUE=1 --Transport=1 --wifiMcs=HtMcs7
--tcpRcvBufBytes=1024000 --DataSizeforTCP=1000000
--delayValueforRHtoR=10 --delayValueforLte=5 --delayValueforWifi=90
--chBwMHz=20 --simTime=7 --aggPath=lteAndWifi"
```