ECE361 Lab1 Report

Ian Hu 1006939244

Santiago Ibagon 1006831591

Ethan Sovde 1006747040

Shawn Zhai 1006979389

1. Preliminaries

(a) When priority acks are used, a transmitter that successfully overhears a data transmission will defer from accessing the channel for α + 2τmax seconds, where α is the duration of an ACK and τmax is the maximum propagation delay. Justify why this is the minimum sufficient time to ensure the ACK is received collision-free.

Assuming that the transmitter overhears the data transmission right as the data is being sent, it will take at most τ_{max} seconds to be sent over the channel. The subsequent ACK will then take α, plus the τ_{max} for it to propagate back through the channel. If the other transmitter sends anything on the channel before this time, it will likely interfere with the ACK being sent.

(b) Using the default parameters above, what are the smallest and largest possible backoff periods a node may incur?

The smallest is 0 seconds, largest is $(2^8 - 1) \times 20\ \mu s = 5.1\ ms$

(c) Consider the 4node_star network from Figure 1 running Pure ALOHA.
i. Could priority ACKs increase the throughput of this network? Explain.
ii. Could carrier sensing increase the throughput of this network? Explain.

In this case, neither one would significantly increase the throughput of this topology due to the hidden terminal problem. The topology of the network means that none of the nodes can see what any of the others are sending. Since both priority ACKs and carrier sensing rely on the detection of messages from other nodes, they will not be able to prevent any collisions.

2. Pure ALOHA

(a) Create a simple topology with two nodes, a transmitter, and a sink. Configure the traffic generator to saturate the transmitter with 100 packets as soon as the experiment begins.

i. How did you configure the traffic generator to do this?

I set **ns3::UdpEchoClient::MaxPackets** to 100 and **ns3::UdpEchoClient::Interval** to 0ms, so that a total of 100 packets are enqueued at the transmitter immediately when the simulation begins.

ii. Since there is a single transmitter, no collisions are possible, yet the throughput is much lower than the data rate of the channel. What factors contribute to this overhead?

Below are the factors that contribute to this overhead.

- **Propagation delay** - even though there are no collisions, the time taken for packets to reach the destination and for ACKs to return can create a delay that reduces the effective throughput.
- **Processing delay** - both the transmitter and the receiver have processing overheads. At the transmitter, it takes time to queue, prepare, and transmit packets. Similarly, the sink has to process the received packets before acknowledging them. This processing time at both ends adds to the overall delay.
- **Jitter and Backoff Times** - jitter is the random delay added before transmission to avoid synchronization issues between nodes. Moreover, if the transmitter waits during backoff periods (even though there is no collision, a packet may still be corrupted due to other reasons in reality, not sure how this is modeled in ns3), it further reduces throughput.

- **Packet queueing delay** - Since all packets are enqueued at the transmitter at the beginning, it experiences delays due to queueing, where packets are waiting to be transmitted.

iii. Momentarily change the data rate from 10Mbps to 1Gbps. Should we expect a 100x increase in throughput from the 100x increase in data rate? How does the actual throughput compare? Be sure to return the data rate to 10Mbps before continuing.

No, we should not expect a 100x increase in throughput just because the data rate has been increased by 100 times. The actual throughput is influenced by several factors beyond the raw data rate, such as propagation delay, packet processing delay, acknowledgment delay, queueing delays, and jitter and backoff delay. These overheads do not scale with the data rate and therefore limit the potential throughput improvement. When the data rate is increased to 1 Gbps, the increase in throughput will be less than 100x due to these constant overheads that are independent of the channel's data rate.

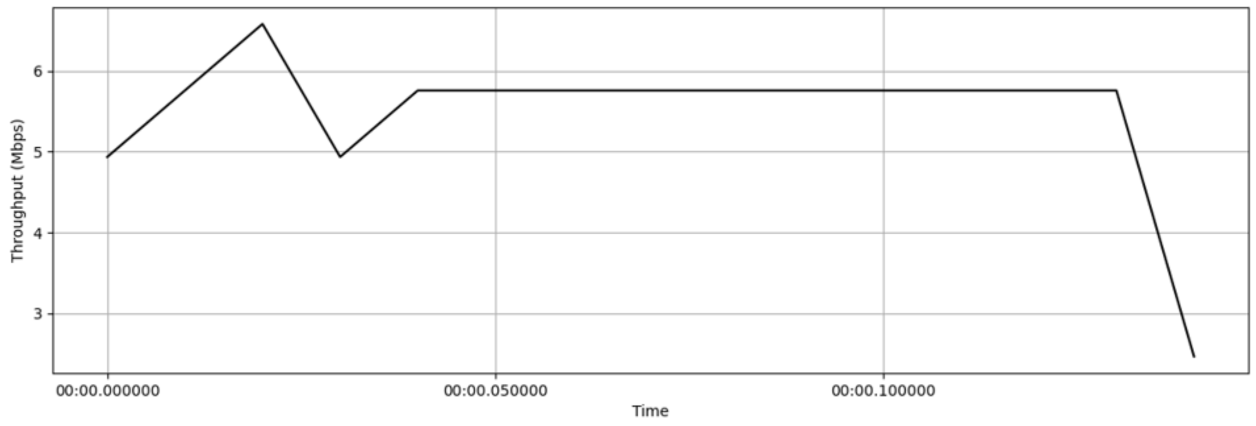


Figure 1: Throughput plot of 10Mbps data rate

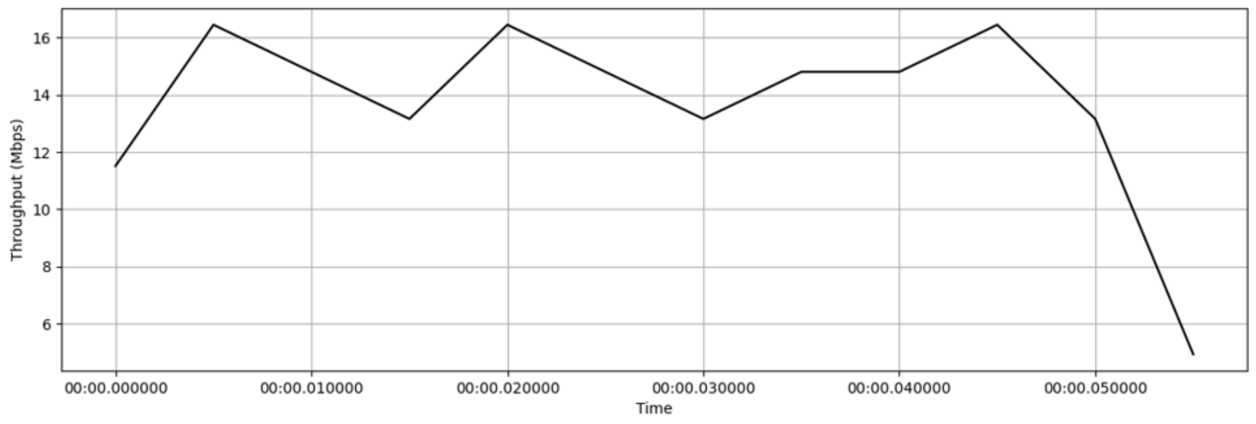


Figure 2: Throughput plot of 1Gbps data rate

The throughput is about 5.8 Mbps and 15 Mbps when the data rate is 10Mbps and 1Gbps respectively. The increase is less than 3x.

(b) Configure the traffic generator to saturate transmitters with 100 packets as soon as the experiment begins. Run the 4node_connected topology. Now run the 7node_connected topology, but reduce the number of packets from 100 to 50, so the total number of packets introduced to the network is the same as in the 4-node experiment.

i. Discuss your results in terms of the throughput, average delay, and the time required to deliver all packets.

- **Throughput**

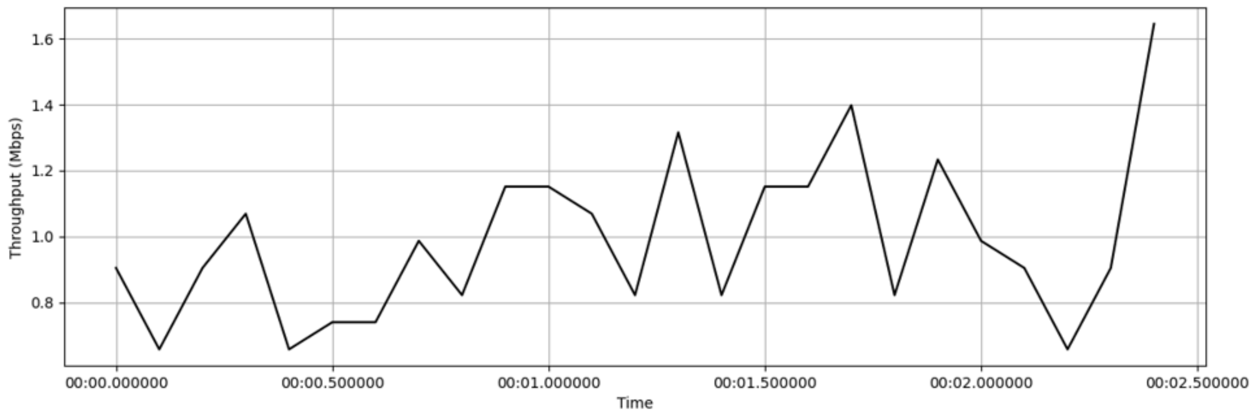


Figure 3: Throughput plot of 4-node configuration

For the 4-node configuration, the throughput fluctuates between around 0.7 Mbps to 1.6 Mbps, with occasional dips. These variations could be due to intermittent delays between transmissions and backoff times. However, the throughput remains relatively high, considering the smaller number of nodes competing for the channel.

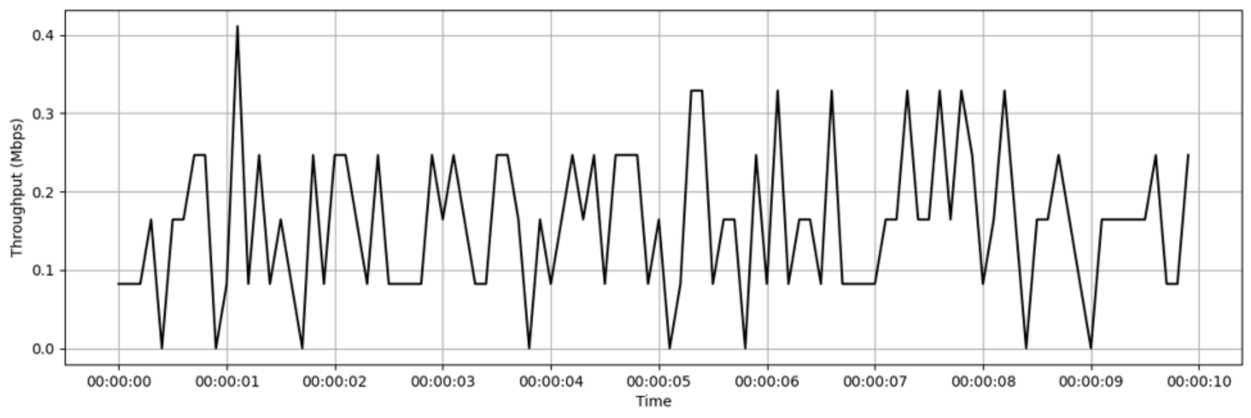


Figure 4: Throughput plot of 7-node configuration

For the 7-node configuration, though the total number of packets that need to be transmitted is the same, the throughput is significantly lower, ranging between 0 Mbps and 0.4 Mbps. This is expected since more nodes (twice as many as in the 4-node configuration) are competing for the channel, leading to more collisions and backoff periods. Consequently, fewer packets are successfully transmitted per unit time, resulting in lower overall throughput.

- Average Delay

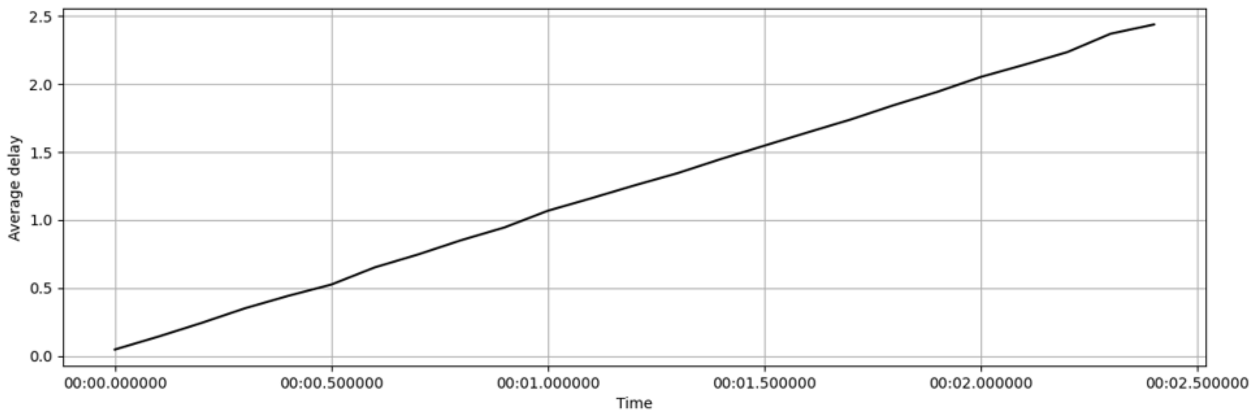


Figure 5: Average Delay plot of 4-node configuration

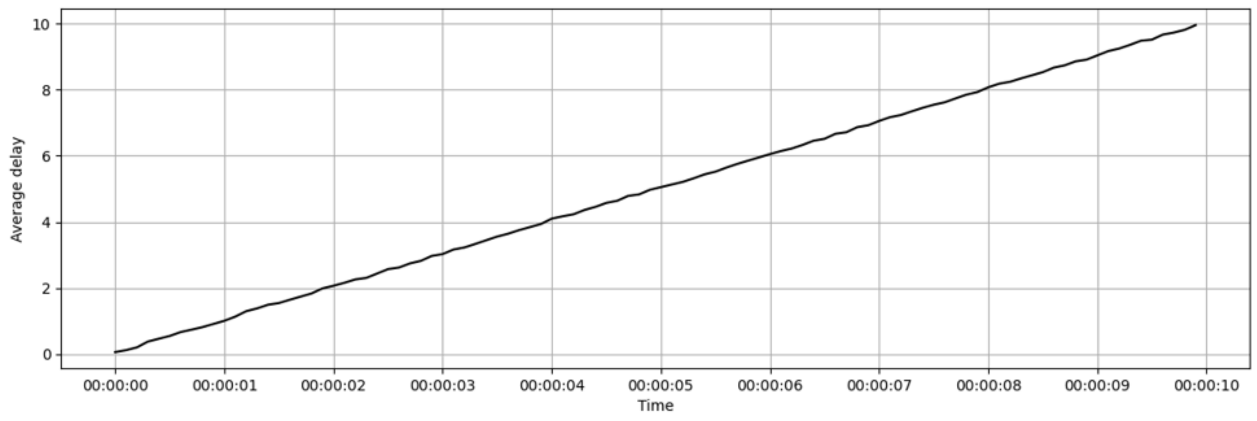
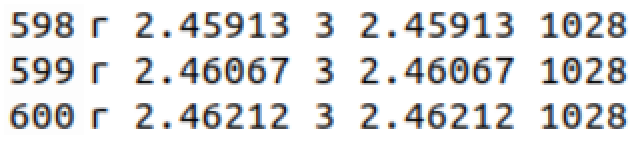


Figure 6: Average Delay plot of 7-node configuration

For the average delay, there's no significant difference. In both configurations the average delay is simply the current time elapsed, indicating that all packets that are not yet transmitted have been waiting since the start of the session to be transmitted.Therefore the slopes are both 1. The only difference here is the simulation time.

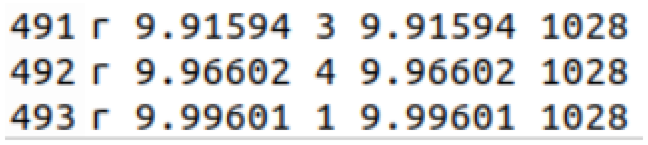
- Total Time



```
598 r 2.45913 3 2.45913 1028
599 r 2.46067 3 2.46067 1028
600 r 2.46212 3 2.46212 1028
```

Figure 7: End of aloha.tr of 4-node configuration

For the 4-node configuration, the total transmission time is about 2.46 seconds as reflected by the time stamp of the last packet received by the sink node in aloha.tr. This is considerably fast as backed up by its moderate slope in the delay plot and high throughput.



```
491 r 9.91594 3 9.91594 1028
492 r 9.96602 4 9.96602 1028
493 r 9.99601 1 9.99601 1028
```

Figure 8: End of aloha.tr of 7-node configuration

In the 7-node configuration, as suggested by the last line of aloha.tr, the transmission is not even completed due to the simulation being terminated after 10 seconds. 107 out of 300 packets are not transmitted. This is expected as this configuration has much lower throughput and steeper slope for the delay plot compared to the 4-node configuration.

ii. Do you think a "good" choice of backoff parameters (minimum, maximum, and factor) depends on the network size? If so, find a backoff configuration that improves the performance of 4node_connected but diminishes the performance of 7node_connected, and rationalize your choice of parameters. If you disagree, argue why the choice of backoff parameters is independent of the network size.

Yes, a good choice of backoff parameters depends on the network size. Below is the configuration I found that improves the performance of 4node_connected but diminishes the performance of 7node_connected, where I decrease the backoff time slot by 1ns and keep the other two parameters as default.

ns3::AlohaMac::BackoffFactor = 19ns

ns3::AlohaNetDevice::MinBackoffExponent = 4

ns3::AlohaNetDevice::MaxBackoffExponent = 8

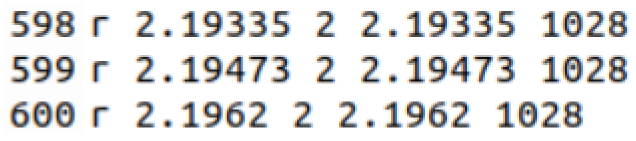

```
598 r 2.19335 2 2.19335 1028
599 r 2.19473 2 2.19473 1028
600 r 2.1962 2 2.1962 1028
```

Figure 9: End of aloha.tr of 4-node configuration with adjusted backoff parameters

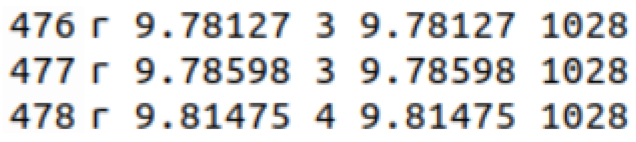

```
476 r 9.78127 3 9.78127 1028
477 r 9.78598 3 9.78598 1028
478 r 9.81475 4 9.81475 1028
```

Figure 10: End of aloha.tr of 7-node configuration with adjusted backoff parameters

As we can see from the end of aloha.tr of each simulation with the adjusted back off time slot, the 4-node configuration is able to complete transmitting 300 files in about 2.20 seconds, which is faster than before. On the other hand, the 7-node configuration is unable to transmit 122 out of 300 packets within the 10 second limit. Therefore, its performance becomes worse under this parameter setting.

3. ALOHA with priority ACKs

(a) Verify your answer to question 1(c) by running the 4node_star network with and without ns3::AlohaMac::UsePriorityAck enabled. Discuss your results.

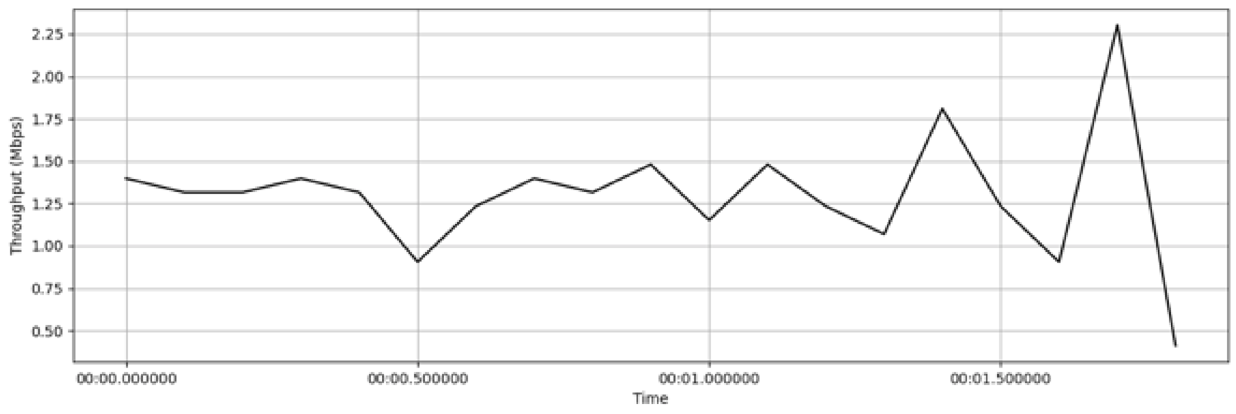


Figure 11. Throughput plot of 4-node configuration with priority ACKs disabled

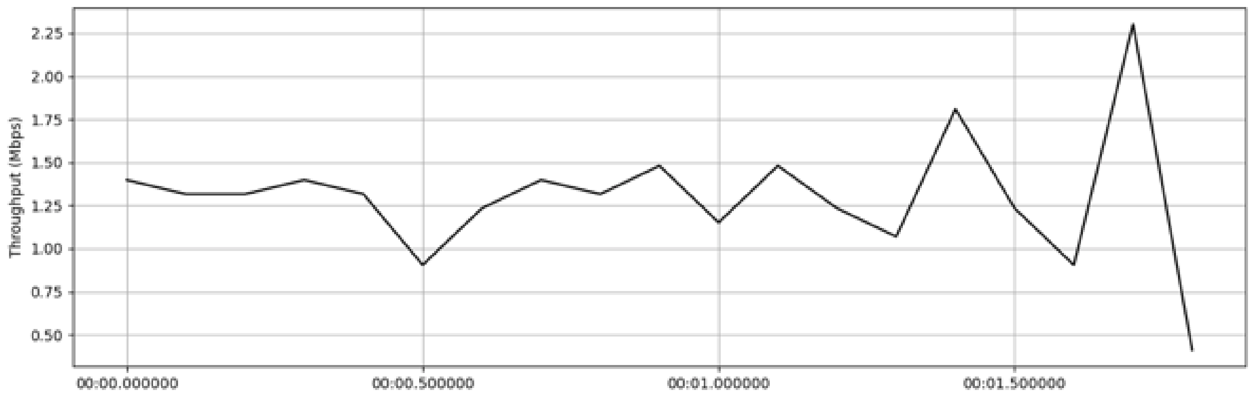


Figure 12. Throughput plot of 4-node configuration with priority ACKs enabled

In question 1(c), we concluded that priority ACKs will not increase throughput because of the hidden node problem. The through plots of the 4node_star network with priority ACKs enabled and disabled are exactly the same, which verifies the result.

(b) Run 7node_connected with ns3::AlohaMac::UsePriorityAck disabled and enabled. Discuss your results in terms of the throughput, average delay, and the time required to deliver all packets.

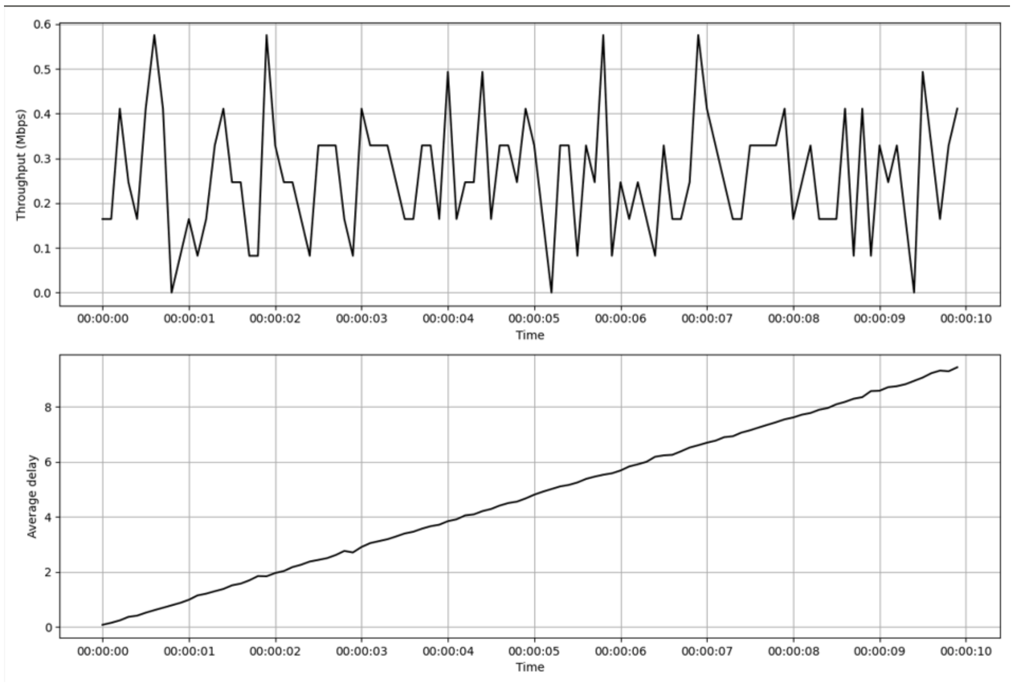


Figure 13. Throughput and Average delay config of 7-node with priority ACKs

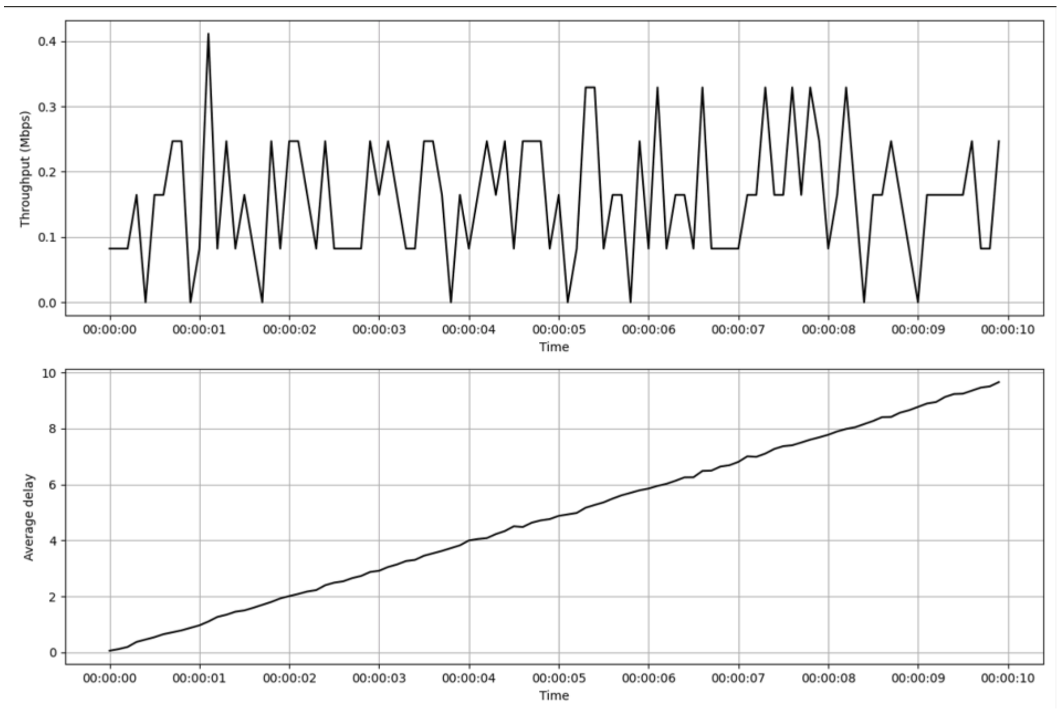
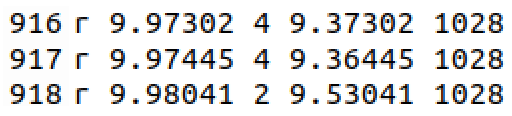
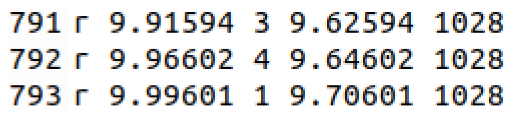


Figure 14. Throughput and Average delay config of 7-node without priority ACKs



```
916 r 9.97302 4 9.37302 1028
917 r 9.97445 4 9.36445 1028
918 r 9.98041 2 9.53041 1028
```

Figure 15. End of aloha.tr of 7-node with priority ACKs configuration



```
791 r 9.91594 3 9.62594 1028
792 r 9.96602 4 9.64602 1028
793 r 9.99601 1 9.70601 1028
```

Figure 16. End of aloha.tr of 7-node without priority ACKs configuration

From figure 13 and figure 14, the throughput of ALOHA with priority ACKs has a peak value of 0.6 Mbps and average about 0.3 Mbps. While ALOHA without priority ACKs has a peak value of 0.4Mbps and average of 0.15Mbps. Hence, the throughput of ALOHA with priority ACKs is greater than the one without priority ACKs.

For the average delay, there's no significant difference. The slope of the average time plot of ALOHA with priority ACKs is slightly smoother than one without priority ACK.

From figure 15 and 16, which are the last 3 lines of aloha.tr for both cases, both of them can't receive 600 packets in 10ms. However, ALOHA with priority ACK receives 318 out of 600 and ALOHA without priority ACK receives only 193 out of 600. As a result, ALOHA with priority ACKs is able to send more packets than ALOHA without priority ACKs in a limited time. In the 7node_connected configuration, nodes are closer to each other than in the 4node_star configuration, so that the hidden node is not a significant problem anymore. Therefore, adding priority ACKs in this case improves the performance of ALOHA.

(c) Does ALOHA with priority ACKs have a higher throughput with large or small packets? Is this in line with the analytical result? If not, explain the discrepancy.

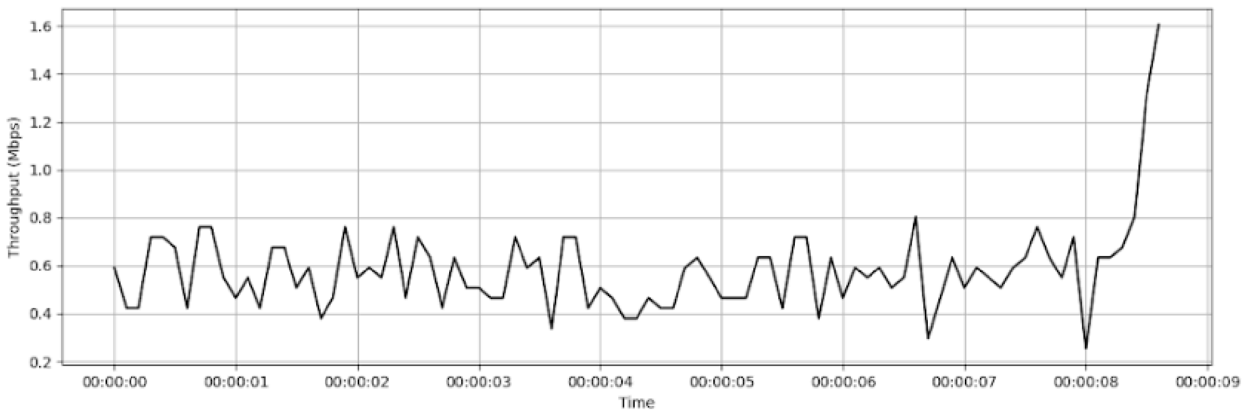
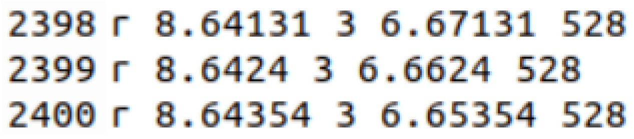


Figure 17. Throughput plot of 7-node config with priority ACK and smaller packet size



```
2398 r 8.64131 3 6.67131 528
2399 r 8.6424 3 6.6624 528
2400 r 8.64354 3 6.65354 528
```

Figure 18. End of aloha.tr of of 7-node config with priority ACK and smaller packet size

For this question, we changed packet size from 1000B to 500B and packet count from 100 to 200. In this way, the total amount of data to be transmitted remains the same, which would give us fair comparison. From figure 17, when the packet size is adjusted to 500B, the throughput has a peak of 1.6Mbps and average about 0.6Mbps. The throughput is higher than the one with 1000B packet size. From figure 18, we can see that all 1200 packets of size 500B are able to be transmitted within the 10s limit, while we saw previously that some of the 600 packets of size 1000B were not able to be transmitted within this time limit. Therefore, ALOHA with priority ACKs have a higher throughput with smaller packets. The result is in line with the analytical result since smaller packets occupy the communication channel for a shorter period, reducing the chance of collisions, which are a primary cause of throughput loss in ALOHA. Additionally, when collisions do occur, the time lost in retransmitting smaller packets is significantly less than that for larger packets, allowing the network to recover more quickly.

4. CSMA (25 Points)

(a) Verify your answer to question 1(c) by running the 4node_star network with and without ns3::AlohaMac::UsePriorityAck and ns3::AlohaMac::EnableCarrierSensing enabled. Discuss your results.

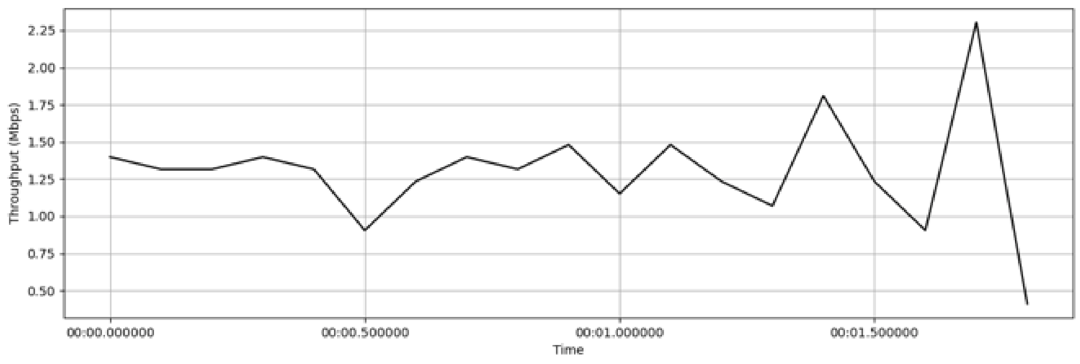
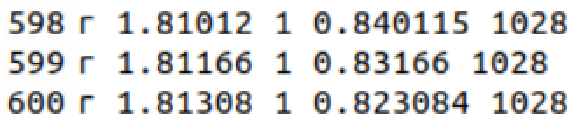


Figure 19. Throughput plot of 4node_star config with both settings disabled



```
598 r 1.81012 1 0.840115 1028
599 r 1.81166 1 0.83166 1028
600 r 1.81308 1 0.823084 1028
```

Figure 20. End of aloha.tr of 4-node star config with both settings disabled

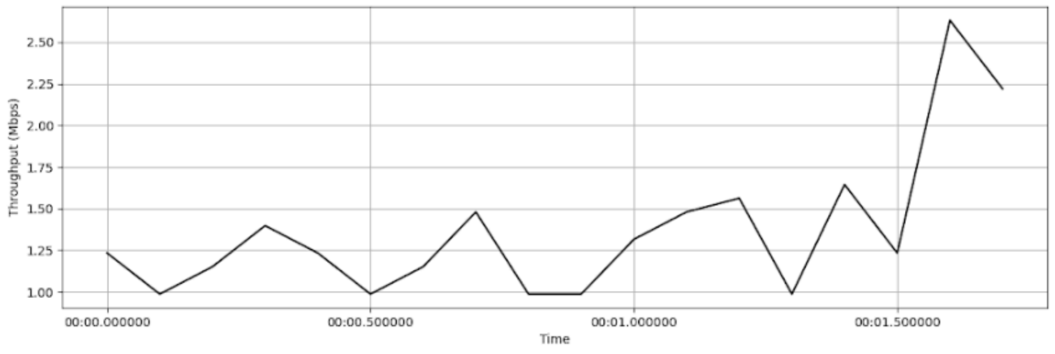
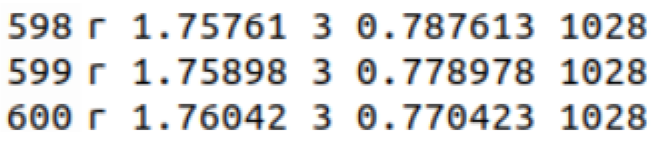


Figure 21. Throughput plot of 4node_star config with both settings enabled



```
598 r 1.75761 3 0.787613 1028
599 r 1.75898 3 0.778978 1028
600 r 1.76042 3 0.770423 1028
```

Figure 22. End of aloha.tr of 4-node star config with both settings enabled

UsePriorityAck and EnableCarrierSensing do not impact the results significantly. Enabling the options does decrease the time to send all packets by 0.05 seconds, but this is not a large enough margin to assign significance to. It likely stems from slight differences in how the simulation decides to send packets. The throughput plots are similar for both options.

(b) Enable ns3::AlohaMac::UsePriorityAck. Run 7node_connected with ns3::AlohaMac::UseCarrierSensing disabled and enabled.

i. Discuss your results in terms of the throughput, average delay, and the time required to deliver all packets.

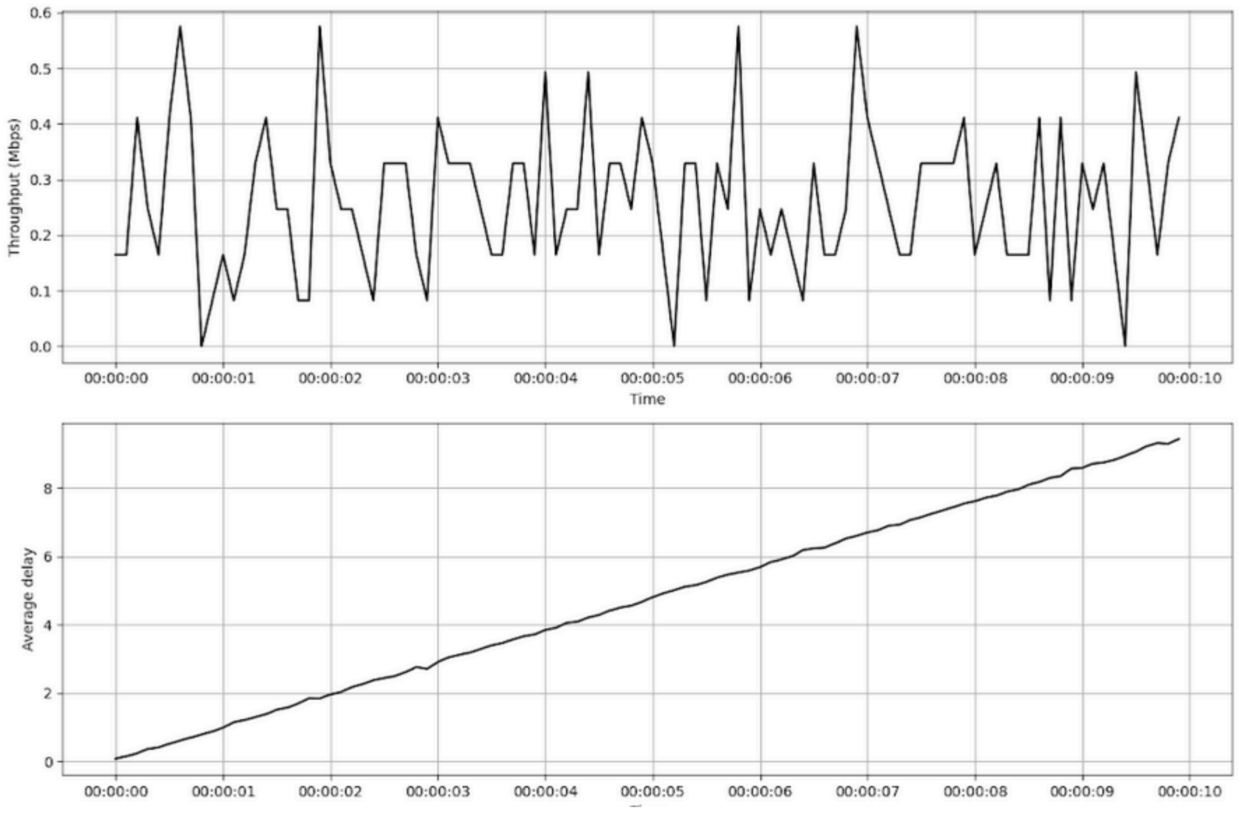


Figure 23. Throughput and delay plots of 7-node connected config with UseCarrierSensing disabled

Throughput when UseCarrierSensing was disabled varied significantly, between 0 and 0.6 Mbps, averaging around 0.25 Mbps. Average delay steadily increased during the run, indicating that packets were taking the full 10 seconds to send.

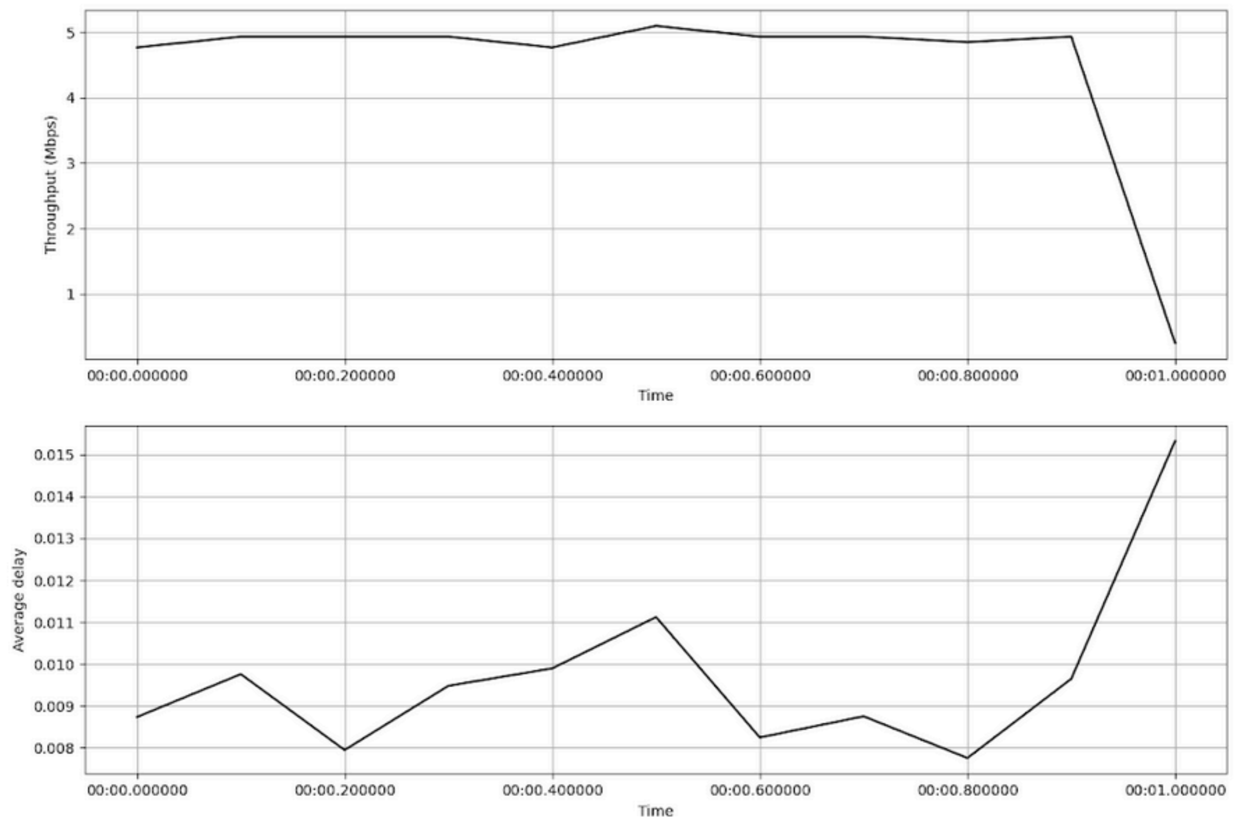


Figure 24. Throughput and delay plots of 7-node connected config with UseCarrierSensing enabled

When UseCarrierSensing was enabled, throughput shot up to a steady 5 Mbps, while delay stayed low, indicating packets weren't having to wait to be fully delivered. The total time to deliver decreased significantly to 1 second.

ii. Based on your results, do you think that ALOHA and CSMA are equally fair? Why or why not?

ALOHA and CSMA are not equally fair due to the way they handle access to the shared communication medium. ALOHA allows any node to transmit at any time, giving an equal opportunity for each node to send packets. Unfortunately, equal opportunity is not inherently fair as some nodes may experience repeated collisions, reducing fairness in practice as some nodes may get more access than others. In contrast, CSMA uses carrier sensing to check if the medium is free before transmission, which helps reduce collisions and makes the protocol more efficient. However, nodes closer to the channel's center or with better sensing capabilities may gain an advantage, potentially leading to less fairness than expected in certain scenarios.

As expected, the fairness plots for CSMA indicate nearly perfect fairness, while ALOHA varies significantly, averaging 0.5 compared to CSMA's 1.0 average.

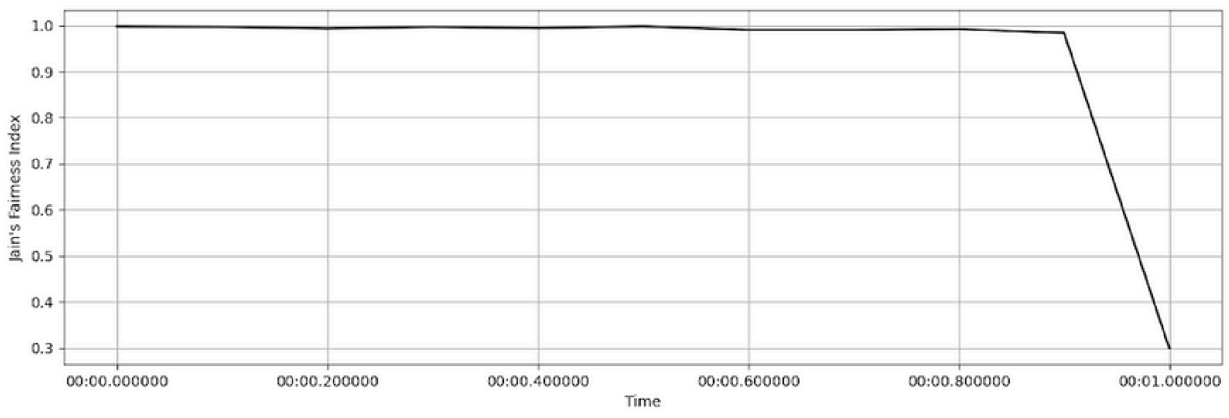


Figure 25. Fairness over time plot of 7-node connected config with UseCarrierSensing enabled

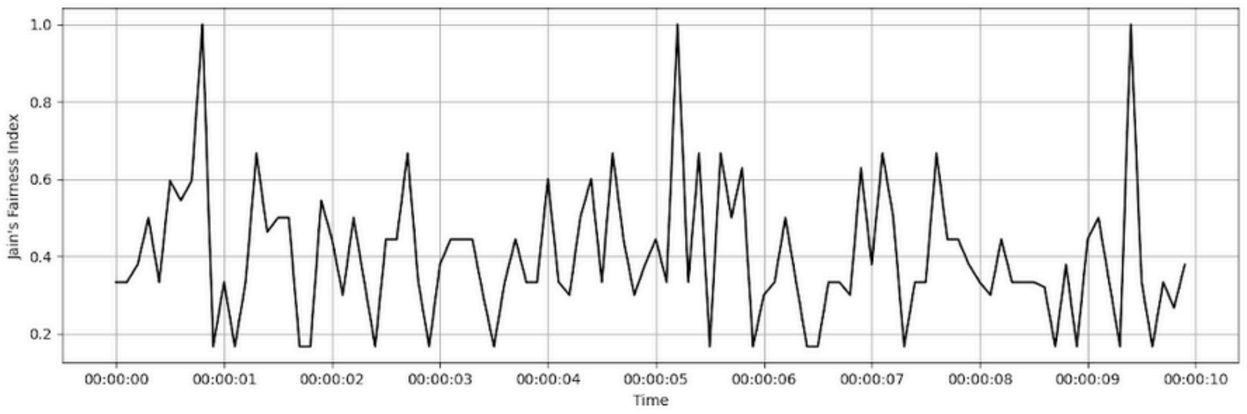


Figure 26. Fairness over time plot of 7-node connected config with UseCarrierSensing disabled

(c) Figure 3 shows the CSMA sender algorithm implemented in ns-3. What could be done to improve the fairness of the protocol?

Instead of a random backoff integer, we could implement a case where nodes wait for less and less over time, meaning older tasks are more likely to be active when the channel is clear.

(d) Does CSMA have a higher throughput with large or small packets? Is this in line with the analytical result? If not, explain the discrepancy.

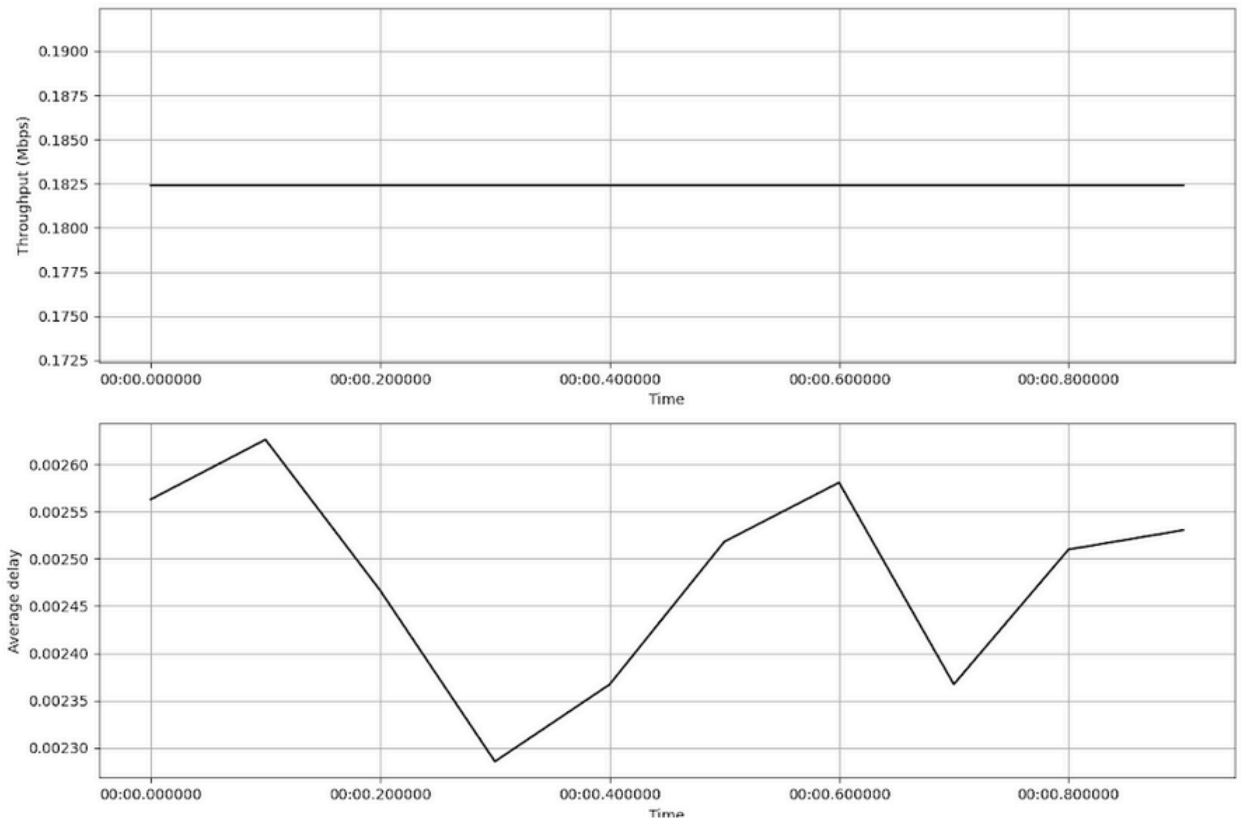


Figure 27: Throughput and delay plots of 7-node connected config with UseCarrierSensing enabled, with 10 byte packets.

Larger packet sizes work better with CSMA. This makes sense with the analytical model, since with many small packets, the anti-collision measures mean that often the channels go unused, or the overhead of the measures slows down total throughput. With larger packets, the delay isn't much compared to the packet transmission times, meaning that the channel gets more efficiently used and doesn't stay very empty. Figure 27 below shows the impact small packets can have on the throughput rate, dropping it all the way down to 0.1825 Mbps.