Wireless Network - ALOHA Simulation

Student: Guilherme Afonso Galindo Padilha

# Introduction

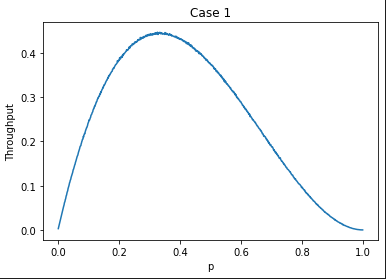
This project aims to simulate both slotted and pure ALOHA, gather results and compare with theoretical analysis. The ALOHA Protocol is a random-access protocol first introduced for wireless LAN. ALOHA is a simple protocol where each user sends data whenever they want. Since the users don’t listen before sending, collisions are common in this protocol. A random delay (or backoff) is added whenever a collision occurs to prevent further collisions from happening. There are 2 different implementations of ALOHA: Slotted and Pure. Both implementations work similarly, but the slotted is more organized and provides better maximum throughput.

# Slotted ALOHA

A simplified version of Slotted ALOHA was implemented using python and ipython notebook. We assume that the transmission time of a packet is 1 slot and that all nodes N are synchronized. Every node will have an infinite number of data to send, and it will always try to send. However, when a collision happens, the node will attempt to retransmit the data with probability p. All simulations will run for 10000 slots. Three cases were studied in this project:

1. Case 1: 3 nodes, different ‘p’ values.
2. Case 2: 100 nodes, different ‘p’ values.
3. Case 3: different N nodes, and using the best ‘p’ from Case 2.

## Case 1

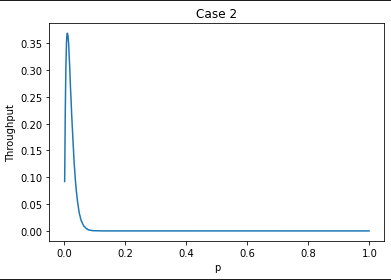


The following settings were considered when running the simulation:

* Range of ‘p’: 0.001~1.0 (1000 values).
* Number of runs for each ‘p’: 20 (total of 20000 runs).

The best throughput found was slightly above 0.4, and the corresponding ‘p’ was around 0.333. Compared to the theoretical results, the best throughput was slightly different, probably due to differences in implementations and also because of this was a simplified version of the Slotted ALOHA. However, the best ‘p’ was exactly the same as expected according to the theoretical 1/N.

## Case 2

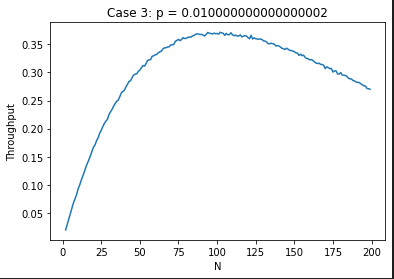


The following settings were considered when running the simulation:

* Range of ‘p’: 0.001~1.0 (1000 values).
* Number of runs for each ‘p’: 10 (total of 10000 runs).

The best throughput this time was almost the same as the theoretical value of 0.37, and the best ‘p’ was also, as expected, around 0.01. Probably due to the increased sample, the slight variations from implementations wasn’t enough to change the expected results by much.

## Case 3



The following settings were considered when running the simulation:

* Range of ‘N’: 2~200 (100 values).
* Number of runs for each ‘p’: 10 (total of 1000 runs).

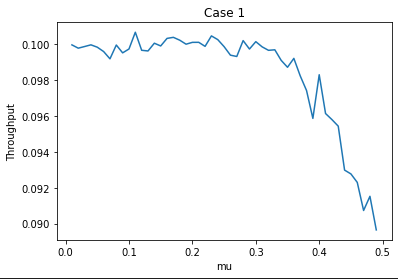
The results from this experiment confirmed the validity of the previous experiment. As expected, the ideal number of nodes for the probability 0.01 is N=100.

# Pure ALOHA

A simplified version of Pure ALOHA was implemented using python and ipython notebook. We assume that the transmission time of a packet is 1 unit time, and each node has only one packet to send. Each node arrives to the system according to a Poisson process of rate LAMBDA, and it tries to transmit as soon as it arrives. However, when a collision happens, the “collided” nodes will attempt to retransmit the data after a delay that follows an exponential distribution of rate MU. To simulate the arrivals using Poisson Process and inter-arrival times, the python function ‘*random.expovariate*’ was used. All simulations will run until the 10000th node arrives. The throughput is calculated by the ratio of successful packets sent by time of last arrival. Four cases were studied in this project:

1. Case 1: LAMBDA = 0.1, different MU values.
2. Case 2: LAMBDA = 0.2, different MU values.
3. Case 3: LAMBDA = 0.3, different MU values.
4. Case 4: different LAMBDA values, and using the best MU from Case 2.

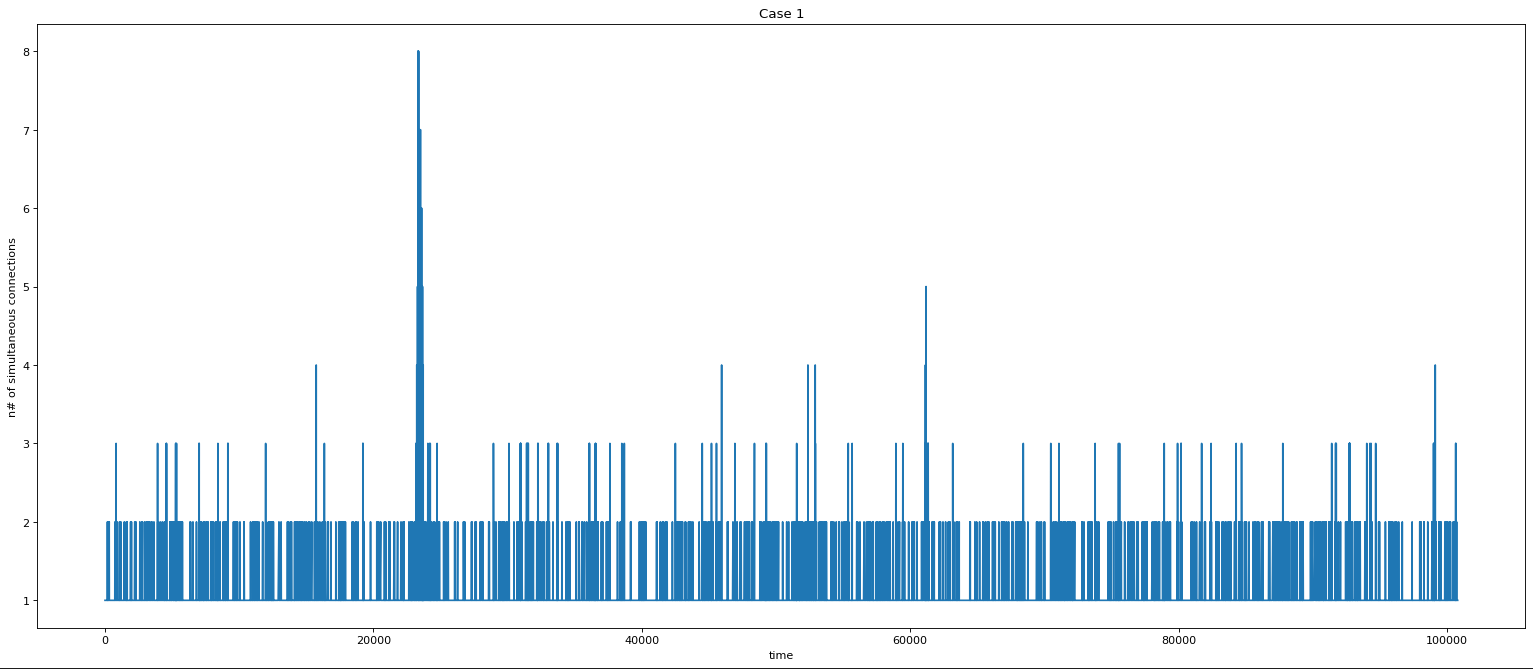
## Case 1



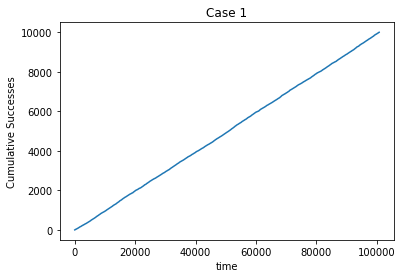
The following settings were considered when running the simulation:

* Range of MU: 0.01~0.3 (30 values).
* Number of runs for each ‘p’: 10 (total of 300 runs).

The best throughput achieved was around 0.1, considerably lower than the theoretical values.



The above graph shows the amount of simultaneous connections vs. time. When it is equal to 1, it means that a successful transmission was finished. When it is above 1, a collision happens and all those nodes are backlogged. Idle times are not shown in this graph. The graph shows both successful transmissions and backlogged nodes.



Using LAMBDA as 0.1, there weren’t many collisions, but the amount of idle time was large. Figure 3 shows that, even though the number of successes neared maximum, the amount of idle time was enough to make the total throughput low.

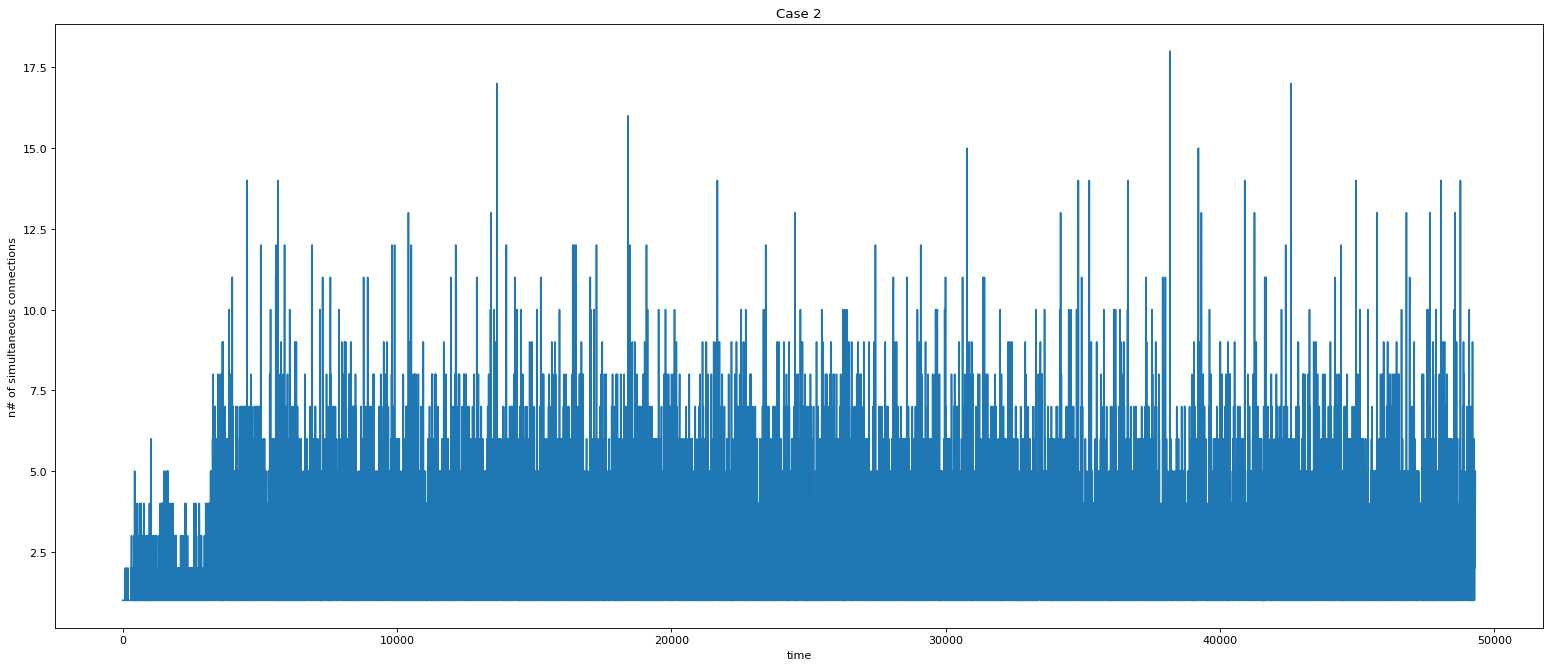
## Case 2



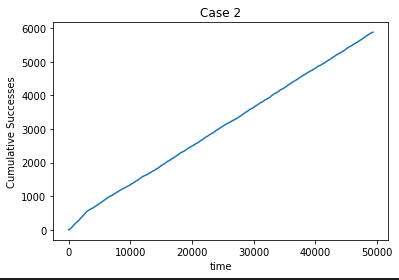
The following settings were considered when running the simulation:

* Range of MU: 0.01~0.3 (30 values).
* Number of runs for each ‘p’: 10 (total of 300 runs).

The best throughput achieved was slightly better than the previous experiment, but still lower than the theoretical values.



Increasing the LAMBDA increased the number of collisions considerably. Since there are more collisions (less time between arrivals), the choice of MU was relevant to the throughput in this experiment.



Even though the amount of successes was lower than the previous experiment, the time was also lower, thus increasing the total throughput.

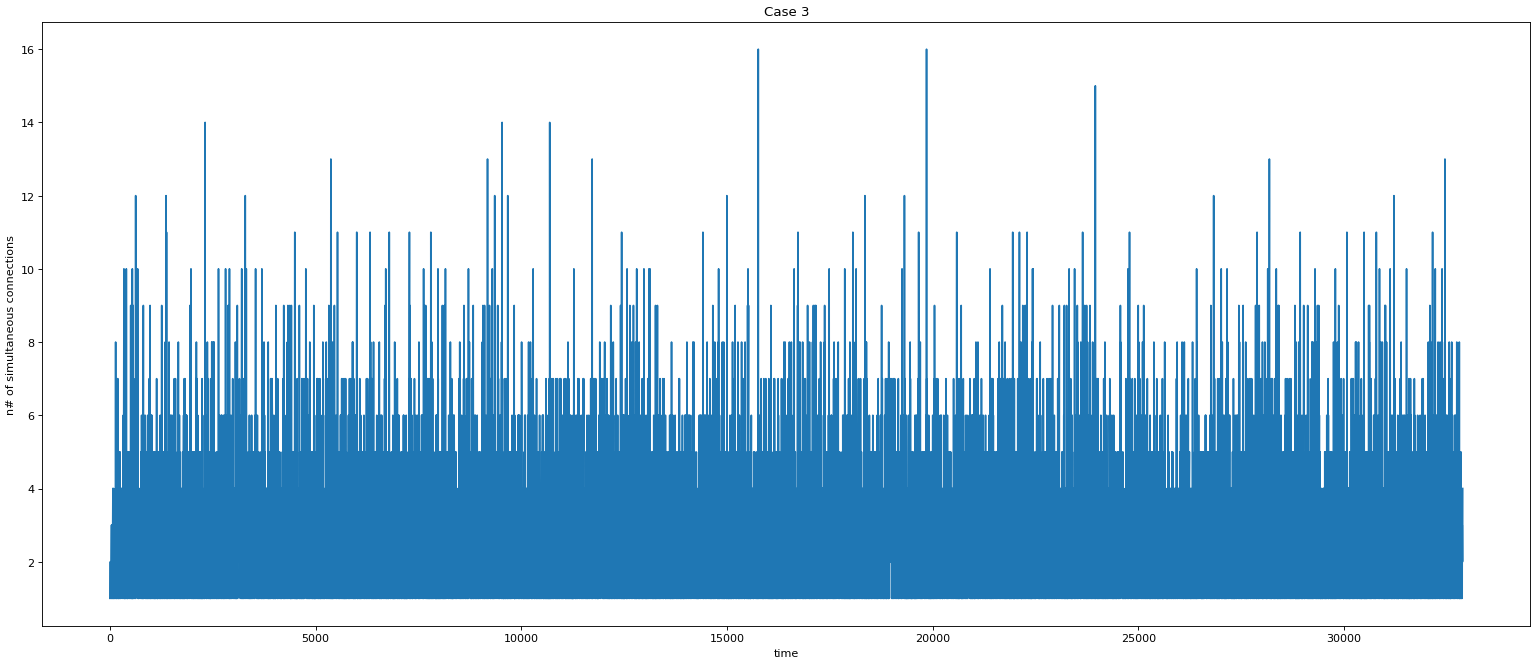
## Case 3



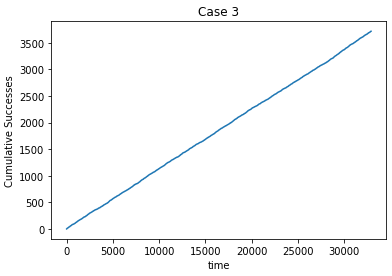
The following settings were considered when running the simulation:

* Range of MU: 0.01~0.3 (30 values).
* Number of runs for each ‘p’: 10 (total of 300 runs).

The best throughput achieved was similar to the previous experiment.

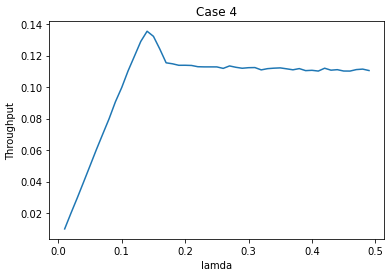


The amount of collisions happening in this experiment was higher than the one in the previous experiment. This confirms the hypothesis that decreasing the interval between arrivals will decrease the idle time, but increase the number of collisions.



Again, we can observe less successful packets delivered, but also in less time used.

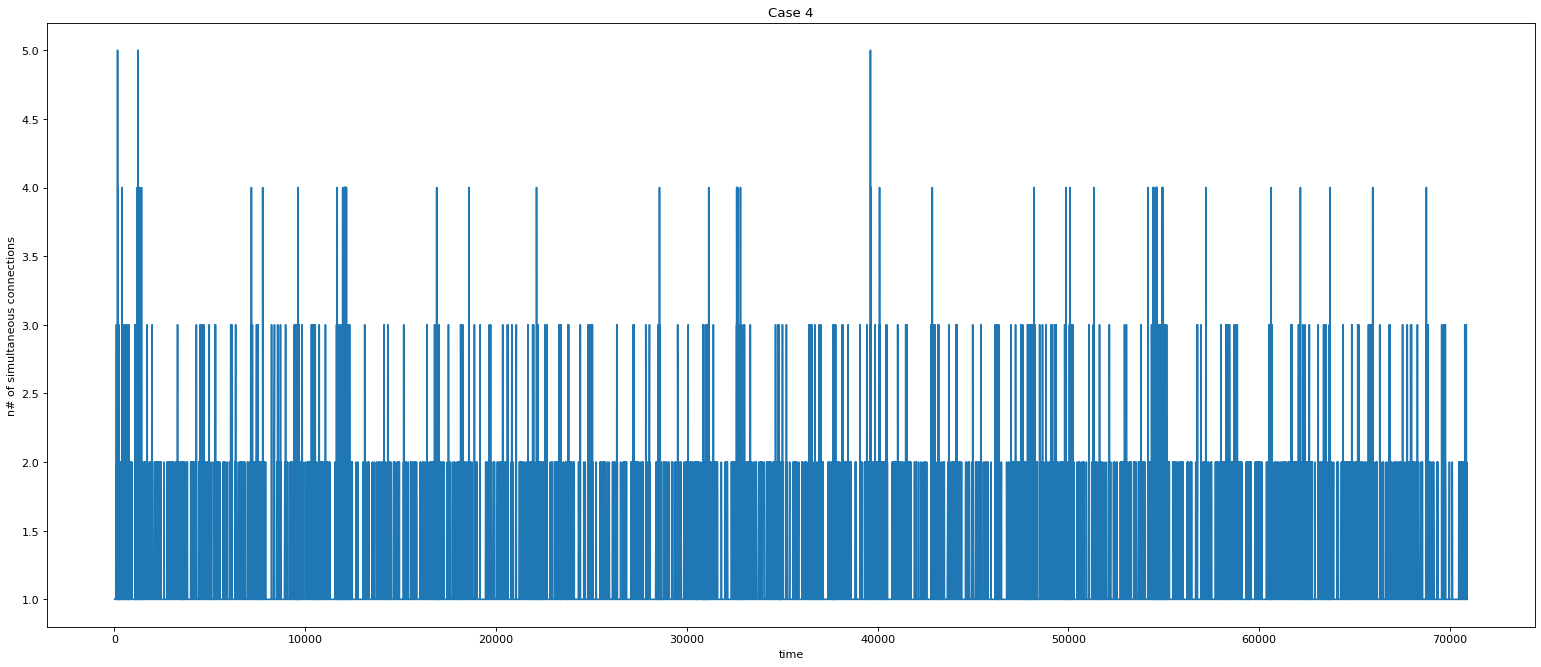
## Case 4



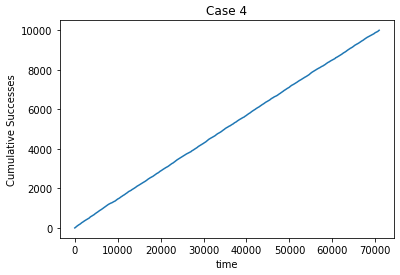
The following settings were considered when running the simulation:

* Range of LAMBDA: 0.01~0.3 (30 values).
* Number of runs for each ‘p’: 10 (total of 300 runs).

The best throughput achieved was higher than all the previous experiments and much closer to the expected value, suggesting that maybe the maximum theoretical value wasn’t achieved due to the low amount of sampling.



If compared to the collision graph of Case 2, this experiment shows a much cleaner graph.



The above graph is similar to the graph from Case 1, but now with less steps due to higher LAMBDA. There was still too much idle time between packets, which means that the algorithm can still be improved.

# Conclusion

The Slotted ALOHA experiment was able to achieve the expected results successfully. The Pure ALOHA, unfortunately, couldn’t achieve the expected results, probably due to differences in the implementation or because the number of runs for each case was too low. Each run of the Pure ALOHA took long, and that limitation caused the number of runs to be considerably lower than the Slotted ALOHA experiment. Although some results were slightly different, overall the experiment was successful.