Eddie Richter

Diego Jimenez

ECE 478 Project 1

10/13/2017

**Introduction**

In our implementation of the Distributed Coordination Function (DCF) of 802.11, we develop a slot-by-slot simulation of two different topologies with and without virtual carrier sensing (VCS). The first topology (Scenario A) consists of four nodes, two transmitters and two receivers, where all nodes are in each other’s collision domain. The second topology (Scenario B) consists of three nodes, two transmitters and one receiver, where the senders are outside of each other’s collision domains. The entire simulation was implemented in a Python environment, allowing us to use object-oriented programming and 3rd party graphing modules to easily create and test our simulation. Both members communicated efficiently on how the problem would be solved and implemented/debugged most components together. However, for bookkeeping purposes, a binary work distribution is given below:

**Eddie:**

* Created initial framework, including class structure and organization.
* Developed first simulation where just one packet could be sent. Used to test the various timing functionalities (DIFS, SIFS, data, etc.)
* Implemented statistic functionality such as counting the number of collisions, fairness index (FI), and throughput of the nodes with varying rates.
* Implemented graphing functionality within the script.
* Wrote the Introduction and Description part of the Report

**Diego:**

* Implemented Poisson distribution.
* Implemented collision functionality.
* Implemented timings for DIFS, SIFS, data counter and ACK.
* Implemented debug print statements so we could create synthetic scenarios and see how the system reacted.
* Wrote the captions for each of the graphs.

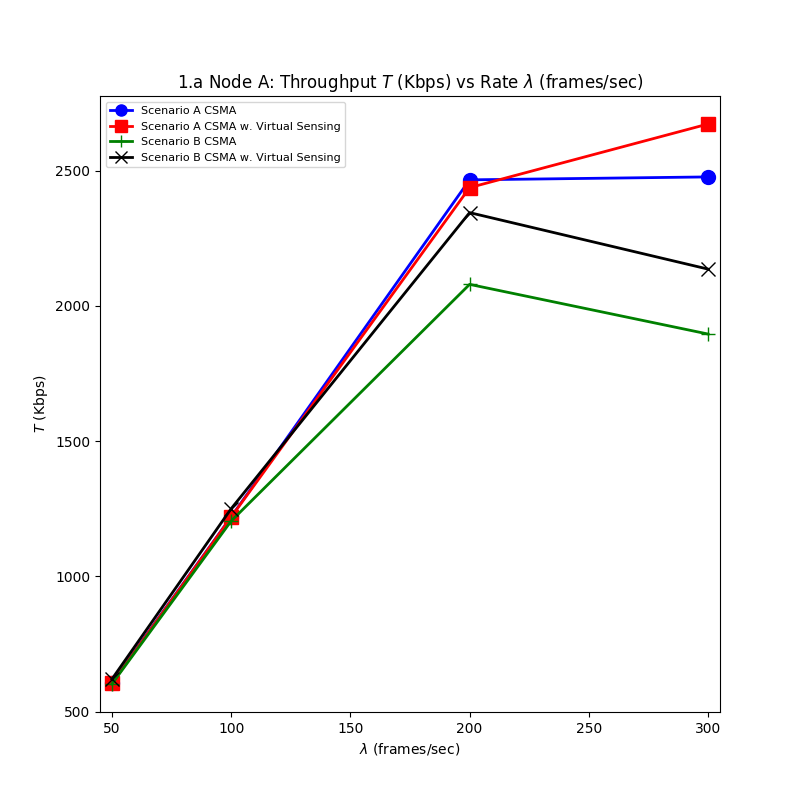
**Description**

Our simulation represents the data flow through the network at the slot level. Meaning we go through the every slot of the scenario we are simulating. Our project contains a Station class, which represents a particular station in the network, and the Spectrum class which contains all stations that are sending, and the status of the channel. At the beginning of a simulation, we initialize the topology (Scenario A or Scenario B, with or without VCS), and generate the Poisson distribution packet arrival times. We then iterate through every slot of the simulation, and check if any station is to send a packet at that particular slot.. If a stations is to send a packet it will set the spectrum to busy and go through the all steps of sending the packet. It will also go through every node in its collision domain to freeze their counters for the correct amount of time. Once the receiving station receives the packet or the RTS, it will send an ACK or a CTS to all stations in its collision domain. This will freeze all unfrozen stations in the receiver domain, which includes the hidden terminal in Station B. If another station doesn’t get frozen in time and sends a packet it will turn the spectrum status from busy to collision, and once the acknowledge counter or CTS counter of the stations is zero, it will increment that station’s number of collisions.

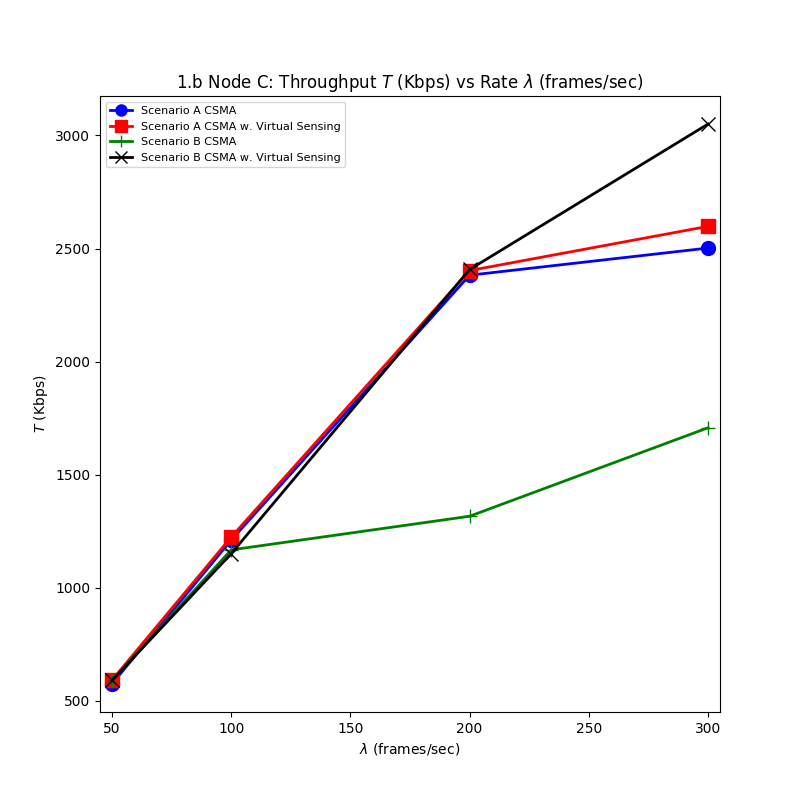
The simulation can be run by navigating to the directory of our simulation and running the command “python driver.py”. The script will go through the eight combination of lambda values, the two topologies, and with VCS turned on and off. It will then save all the ten figures required for the report in the directory of driver.py as PNGs. The figures were generated using the python module matplotlib to generate MATLAB like plots.

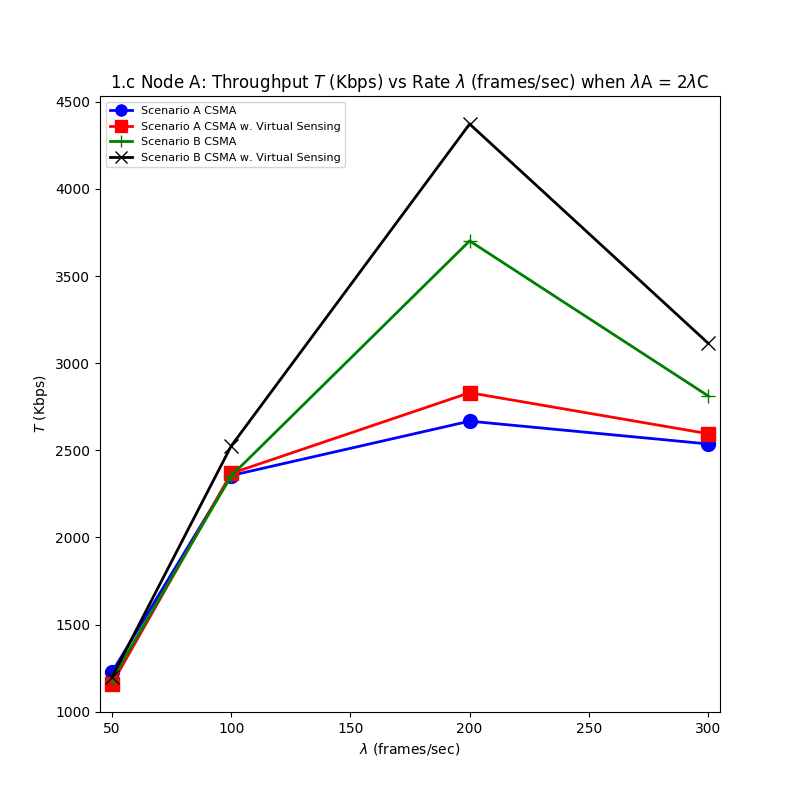
**Graphs**

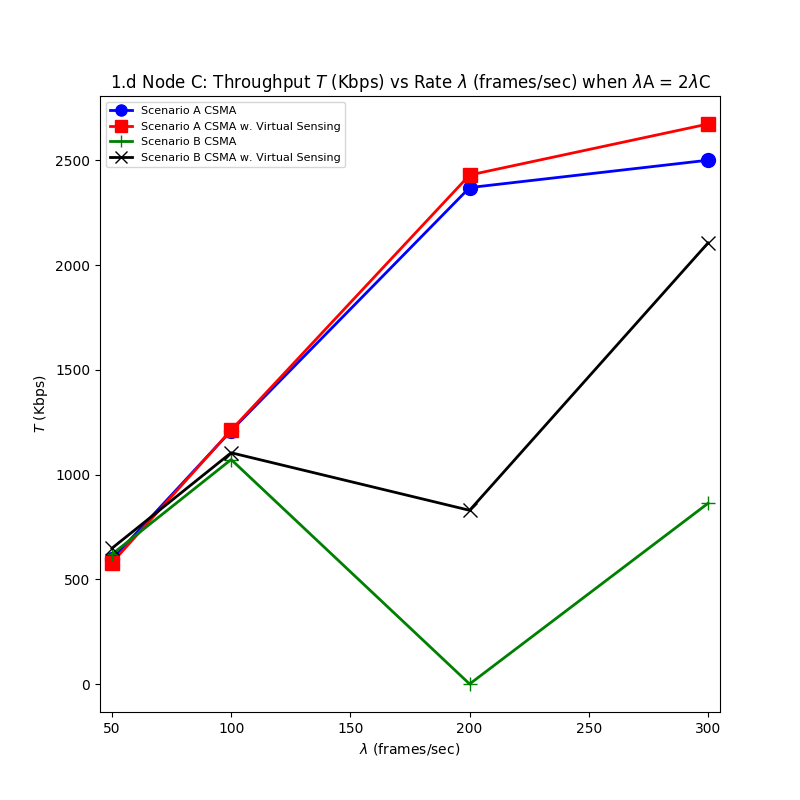
Each of these graphs have four lines, each representing one of the four Scenarios. The blue line is for Scenario A without VCS, red line is for Scenario A with VCS, green line is for Scenario B without VCS and the black line is for Scenario B with VCS.

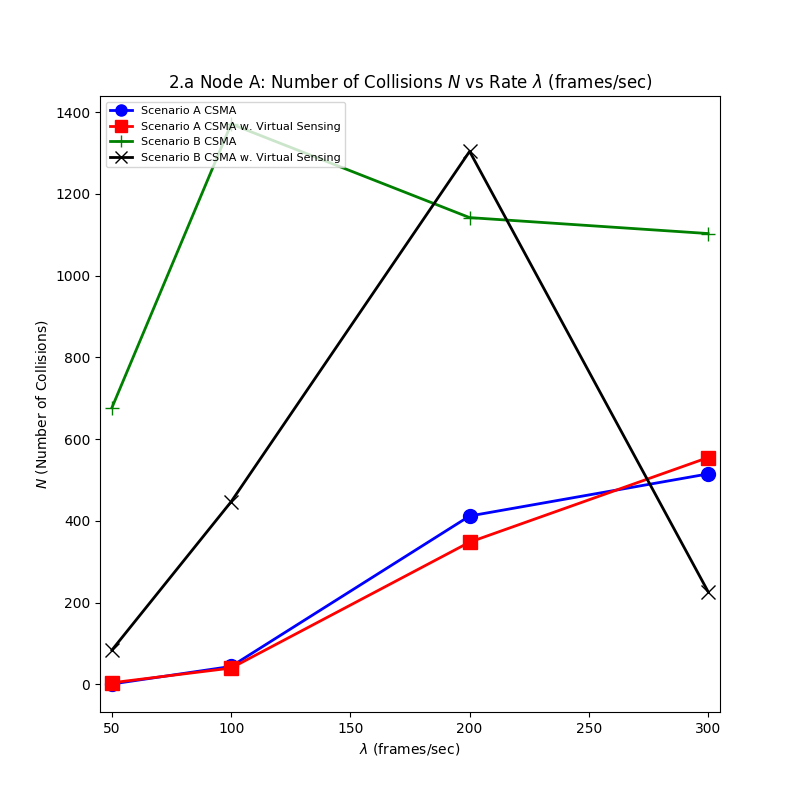


Graph 1.a - In this graph we see the throughput of node A in each of the four scenarios as lambda changes. In this graph, the lambda values of node A and C are the same. This figure demonstrates how the throughput (in Kbps) changes as lambda increases. Generally, as lambda increases, or the amount of information being sent increases, the throughput over the medium will increase. The throughput seen in Scenario A with and without VCS are similar. One key differences between these two is that using VCS adds overhead to every transmission. So when there is a low amount of contention over the medium, without VCS has a higher throughput as it has less overhead. However, if a collision occurs, VCS contains less data to be retransmitted since collisions get caught in the RTS. Scenario A with and without VCS will have very similar throughputs until lambda has increased to 300 frames/sec, when there is more contention on the medium. This added contention means there will be more collisions, so the throughput of VCS will be higher as collisions are less costly. Scenario B has a lower throughput than Scenario A because Scenario B has a hidden terminal. Scenario B without VCS has the worst throughput because one station will not detect when the other station is transmitting, resulting in multiple collisions and lots of information retransmitting. Scenario B with VCS performs better than without VCS because the receiver can stop the other station from sending information after a CTS has been received. In general, Scenario A performs better than Scenario B because Scenario A allows for the all stations to be frozen the instant a stations starts transmitting.

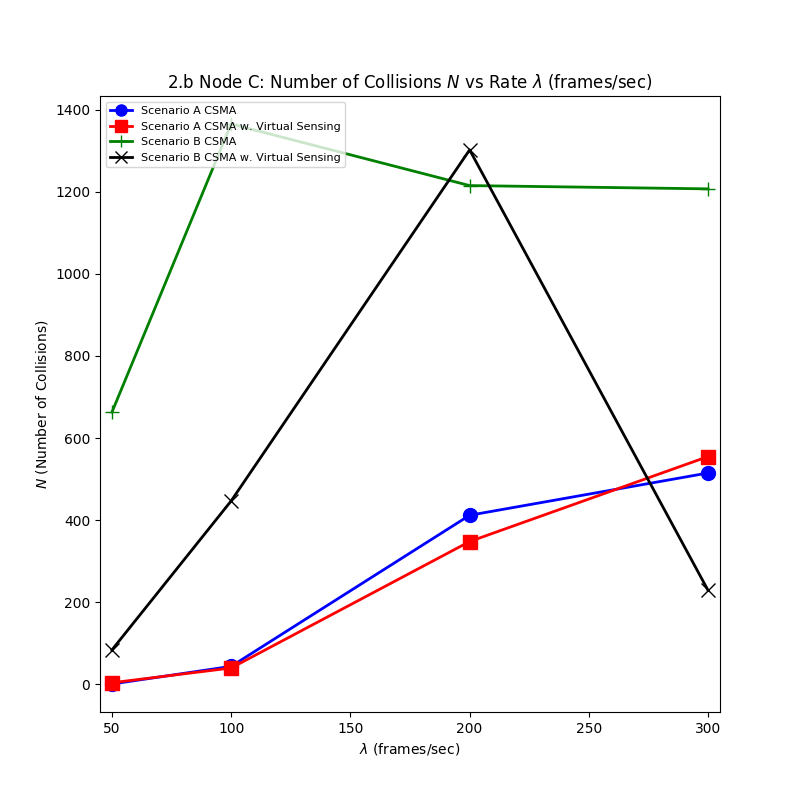
   
Graph 1.b - In this graph we see the throughput of node C in each of the four scenarios as lambda changes. In this graph, the lambda values of node A and C are the same. This figure demonstrates how the throughput (in Kbps) increases as lambda increases. In this graph, we see very similar results to those in Graph 1.a. This is to be expected because the lambda values for station A and C are the same. Meaning that both stations have the same rate at which they intend to transfer information. One way to verify the results is to add the throughput of the same scenario in station A and station C. The total bandwidth of the line is 6 Mbps, therefore, the sum of the throughput for station A and C in each scenario should be less than 6 Mbps since there are collisions and overhead information.

  
Graph 1.c - In this graph we see the throughput of node A in each of the four scenarios as lambda changes. In this graph, the lambda values of node A are double those of node C. This figure demonstrates how the throughput (in Kbps) increases as the lambda value of station C increases. Comparing this graph to Graph 1.a, it can be observed that the throughput for node A is higher than before. One observation that can be made from Scenario A is that the plateauing effect is larger in Graph 1.c than Graph 1.a. It is seen that the throughput in Scenario maxes out at about 2,500Kbps. On the other hand, in Scenario B, the throughput is higher than what it was in Graph 1.a. This difference in throughput is due to the hidden terminal. The hidden terminal will prioritize the stations that have a higher framerate.

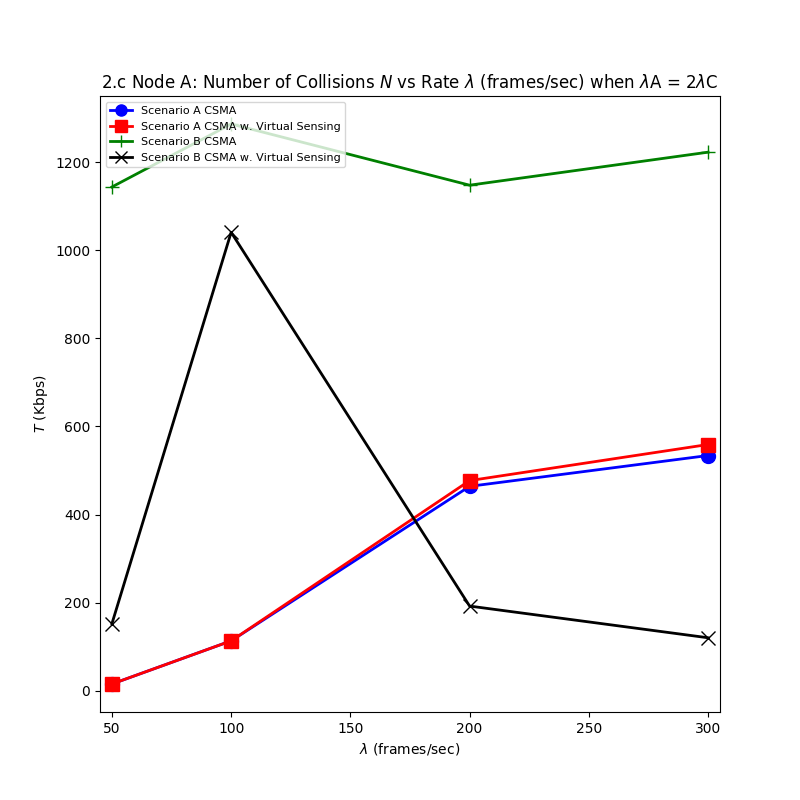
  
Graph 1.d - In this graph we see the throughput of node C in each of the four Scenarios as lambda changes. In this graph, the lambda values of node A are double those of node C. This figure demonstrates how the throughput (in Kbps) increases as the lambda value of station C change. This graph is related to Graph 1.c because the addition of the throughputs at the respective lambdas and scenario should not add up to more than the max throughput of the medium, which is seen to be correct. Scenario A shows that in this case the difference in the lambda values does not matter much when it comes to the throughput as comparing the throughputs to those in Graph 1.c show that they are about the same. The throughput of Scenario B is shown to be lower than it was in Graph 1.b, this is expected because node A had and sent more information which made node C not be able to send as much.



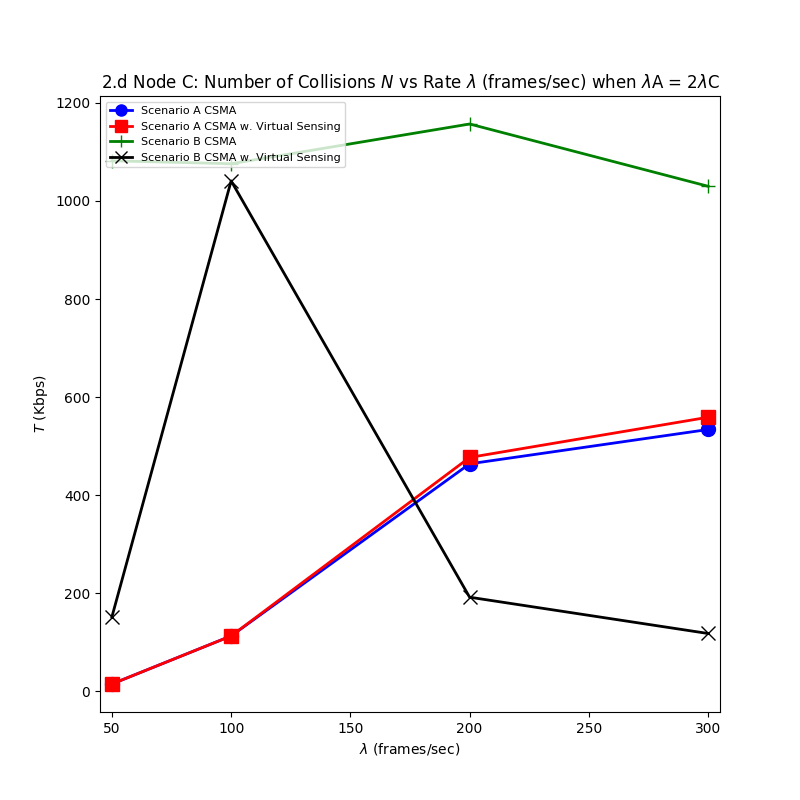
Graph 2.a - In this graph we see the collisions of node A in each of the four scenarios as lambda changes. In this graph, the lambda values of node A and C are the same. The y-axis demonstrates how the collisions as the lambda values change. The lines of Scenario A show that these two have similar results. When the value of lambda is small then there will not be enough information to send for the medium to always be busy, which means that when stations want to transmit the medium will most likely be free and without any collisions. The topology of Scenario A also lends itself to a small number of collisions because any time one of the stations wants to send, then the other sending stations will be frozen faster than in Scenario B. preventing other sending stations from causing collisions. The trend seen is that the number of collisions increase, in what seems like a logarithmic manner, as the lambda values increase so that more information wants to be sent. Scenario B without virtual carrier sending on average has the highest number of collisions because of the hidden terminal one station is not able to communicate with the other station that information is being send. Scenario B with virtual carrier sensing has more collisions than Scenario A because it takes longer to freeze the other sending station and there is a higher chance that stations will choose backoff values that will cause collisions. On average, Scenario B with virtual carrier sensing has less collisions than without virtual carrier sensing because it is able to stop other stations from transmitting.



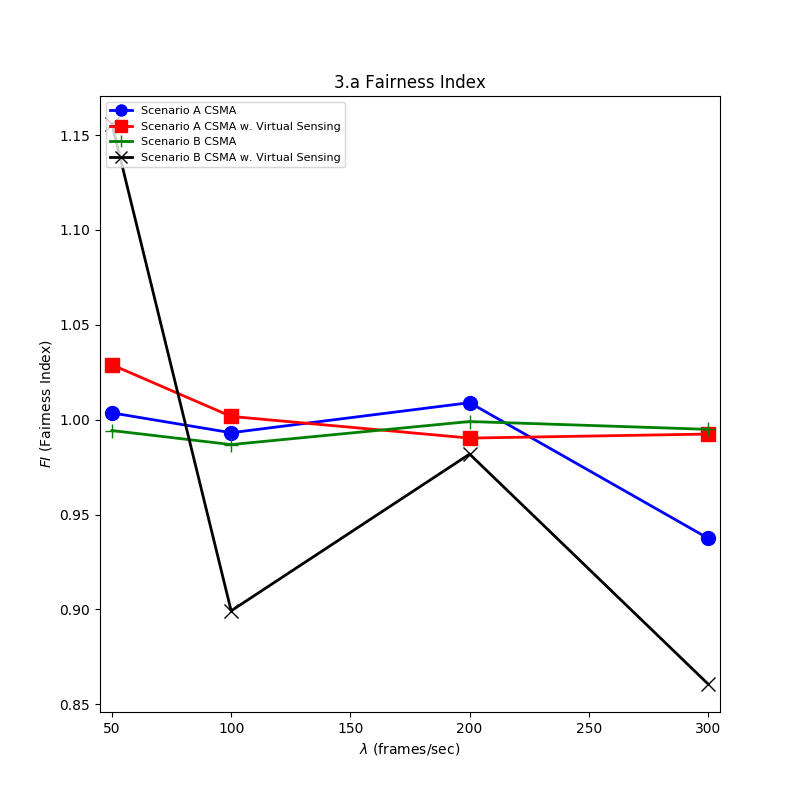
Graph 2.b - In this graph we see the collisions of node A in each of the four scenarios as lambda changes. In this graph, the lambda values of node A and C are the same. The y-axis demonstrates how the collisions as the lambda values change. This graph is almost the same as graph 2.a. This is because we only have two sending stations so whenever one station is having a collision, it needs another station that it will collide with. One scenario in which one station could get more collisions than the other is if there is a collision at the end of a packet sending, the original packet will be corrupted causing a collision, the packet that was trying to send will also be a collision but this is not known until the packet is done transmitting and there is no ack. The original station might try to transmit again as soon as possible and it may be that it is while the second station is still transmitting.



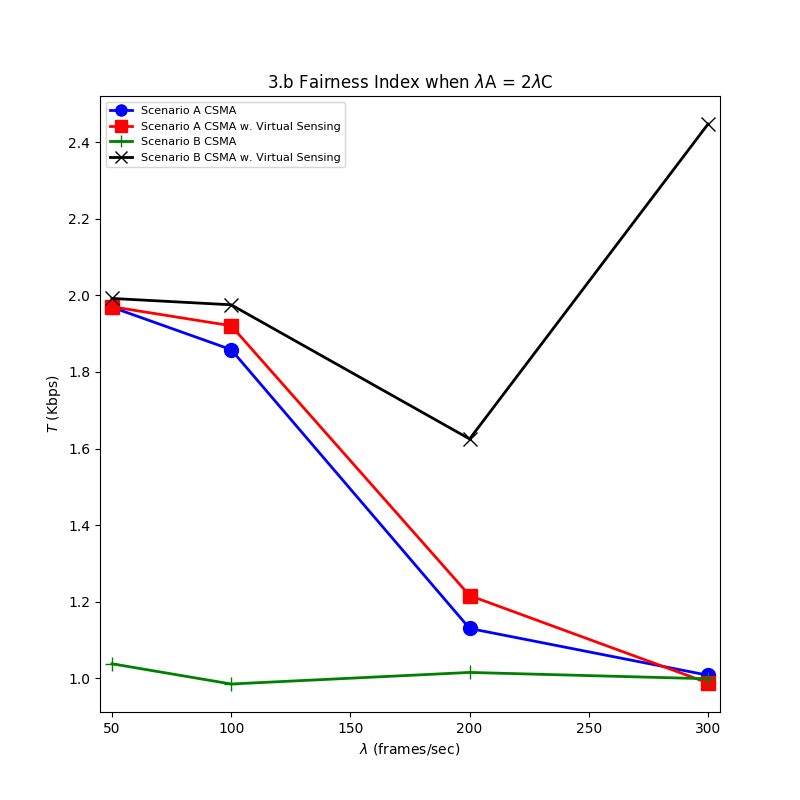
Graph 2.c - In this graph we see the collisions of node A in each of the four scenarios as lambda changes. In this graph, the lambda values of station A are double those of station C. The y-axis demonstrates how the collisions as the lambda values of node C change. The shape of this graph is similar to the shapes of graphs 2.a and 2.b, which show the collisions of a node when lambdas are the same. Although, the trends seen are the same, one difference is that in this graph, the spike in Scenario B with virtual carrier sensing is cause when the lambda of A is 200 frames/sec or as shown in the graph when the lambda of C is 100 frames/sec. This high spike just tells us that this is a lambda value which based on the parameters used causes a high number of conditions in that scenario.



Graph 2.d - In this graph we see the collisions of node C in each of the four scenarios as lambda changes. In this graph, the lambda values of station A are double those of station C. The y-axis demonstrates how the collisions as the lambda values of node C change. This graph can be seen that shows some of the same characteristics that were discussed in graphs 2.a, 2.b, and 2.c such that there is an high spike in collisions when the lambda value of C is 100 frames/sec and the graphs of 2.c and 2.d are almost the same because most likely the two stations will have about the same number of collisions.



Graph 3.a - In this graph we see the fairness index (FI) in each of the four scenarios as lambda changes. In this graph, the lambda values of node A and C are the same. The y-axis demonstrates how the fairness index as the lambda values change. This graph is meant to represent the “fairness” at which data packets are sent from either node. A fairness index of 1 indicates that both stations are sending information at the same rate. In general, the fairness index of the nodes in all Scenarios are close to 1 and for the most part only get closer to 1.



Graph 3.b - In this graph we see the fairness index (FI) in each of the four scenarios as lambda changes. In this graph, the lambda values of station A are double those of station C. The y-axis demonstrates how the fairness index as the lambda values of node C change. There are three different trends to be observed in this graph. The first is what is happening to Scenario A, at first the fairness index is at 2 which indicates that node A is sending twice more packets than node C. As the lambda value of A gets larger, then there will be more contention on the medium, causing a competition between the two nodes for sending. If there is always information to be sent then nodes will always compete for who can send next, explaining why the fairness index is at two for small lambdas and it is close to 1 for larger lambdas. The next trend to be observed is what is happening in Scenario B without virtual carrier sensing. In this scenario, there are no ways for the other station to know that the other is transmitting so there will be always be a competition for who can transmit next, causing the fairness index to always be a value close to 1. Lastly, Scenario B with virtual carrier sensing has a similar property as Scenario A in that for small lambdas the fairness index is close to 2. Then, in the last point, when the lambda of A is 600 frames/sec and the lambda of C is 300 frames/sec, the fairness index of Scenario B with virtual carrier sensing goes higher. This is because there will always be contention on the medium since both of the sending nodes have packets ready to be sent. If there are a few collisions between these two nodes then they will start choosing larger backoff values. Then when one of the nodes is ready to transmit, it is possible that the other station is frozen with a large backoff value. The first station will finish sending and the backoff counter will reset back to 4 while the other node that was frozen is still counting down from the large backoff when there was contention for the medium. This will make the node that just sent information send another packet because its backoff will reach 0 before the other node. This scenario can happen with both nodes but node A benefits more from it because it has more information to send.