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. This is called the hidden terminal problem, as the senders have restricted communication. 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 every slot of the scenario we are simulating. Our project contains a Station class, which represents a 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 slot. If a station 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. If there was no collision and the transmitting station receives an acknowledgement, the number of successful transmissions increases.

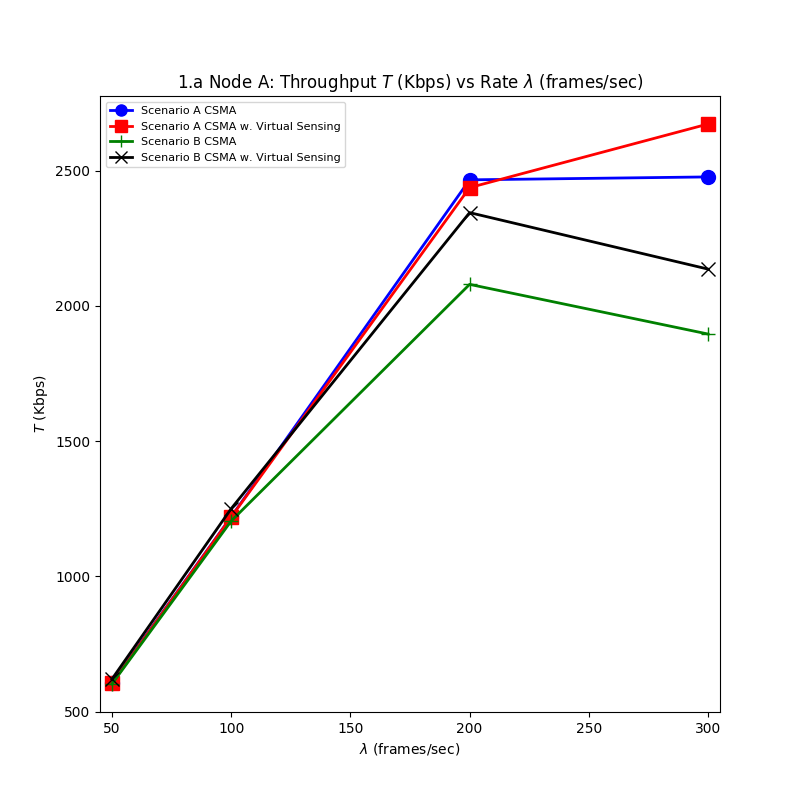
We generate ten graphs throughout our simulation. All graphs have four lines, representing Scenario A without VCS, Scenario A with VCS, Scenario B without VCS, and Scenario A with VCS. The first two graphs are the throughput of node A and node C versus lambda when their lambda values are the same. The second two graphs are the throughput of node A and node C versus the lambda of node C when the framerate of A is twice that of node C. The next two graphs show the number of collisions for node A and node C versus lambda when their lambda value is the same. The two graphs after that are the number of collisions of node A and node C versus the lambda of node C when the framerate of A is twice that of node C. The last two graphs represent the fairness of the scenario versus lambda when the lambda values are the same, and when the lambda value of node A is twice the lambda value of node C. The last two graphs are to represent the fairness of the scenario by calculating the Fairness Index (FI) where:

.

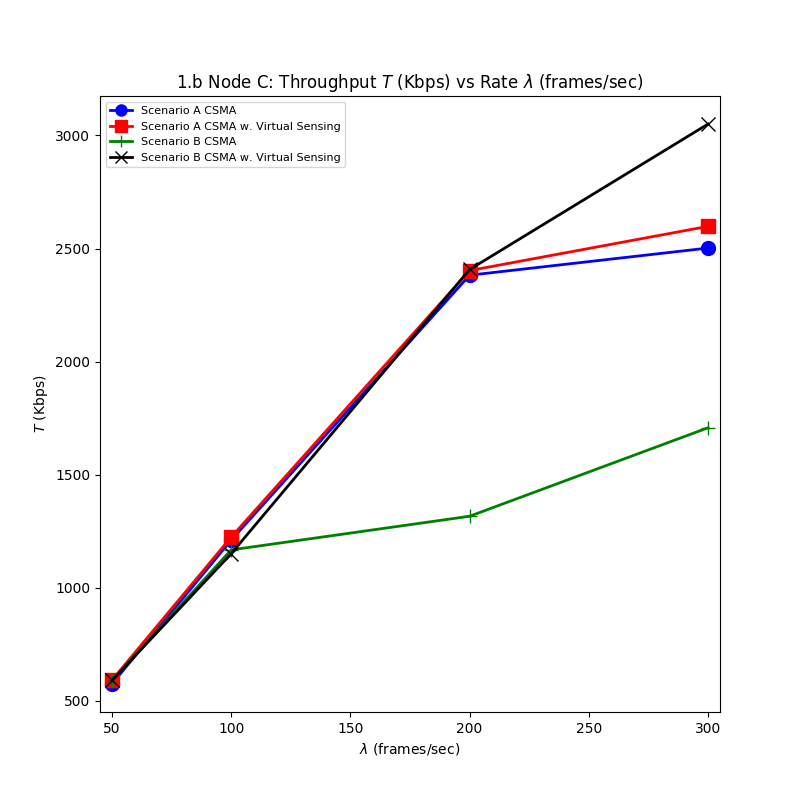
The simulation can be run by navigating to the directory of our simulation and running the command “python driver.py”. The script will go through the eight combination of lambda values, the two topologies, and with VCS turned on and off. It will then save all the ten figures required for the report in the directory of driver.py as PNGs. The figures were generated using the python module matplotlib to generate MATLAB like plots. This simulation was developed using Python 2.7, it is unknown if it will work with other Python versions.

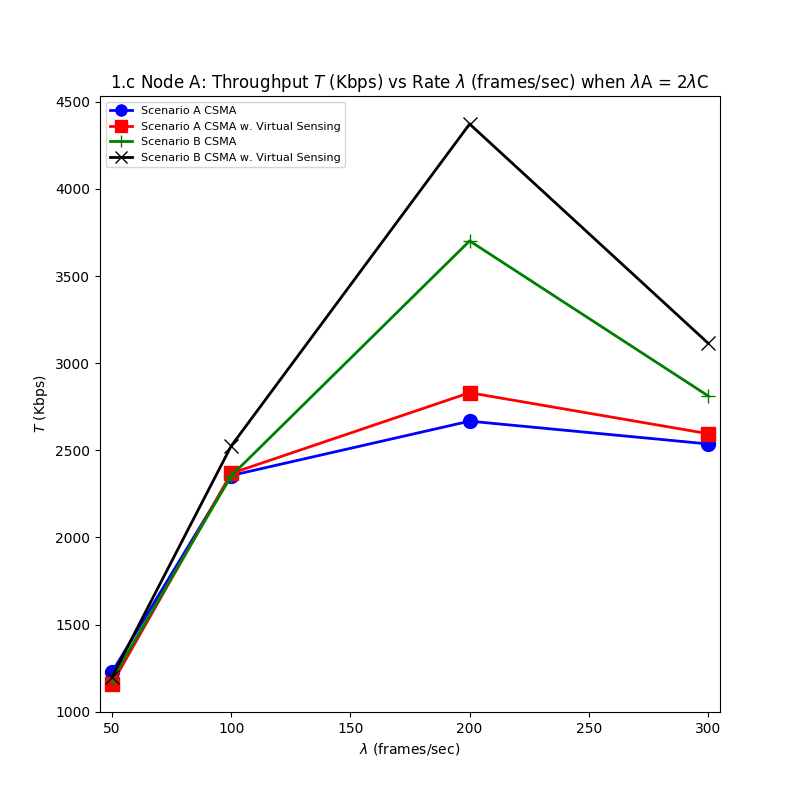
**Graphs**

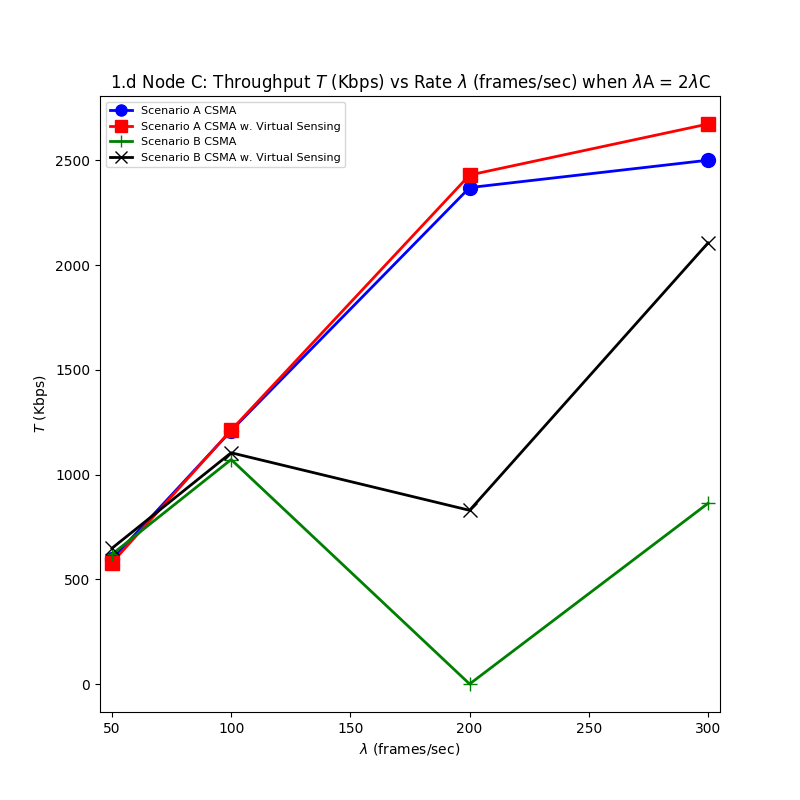
Each of these graphs have four lines, each representing one of the four Scenarios. The blue line is for Scenario A without VCS, red line is for Scenario A with VCS, green line is for Scenario B without VCS and the black line is for Scenario B with VCS.

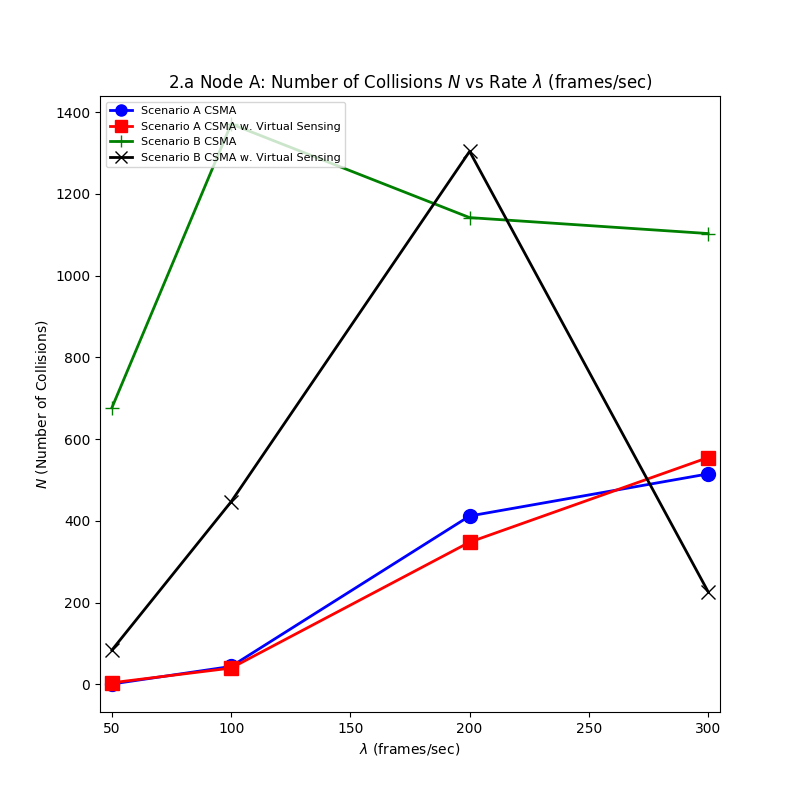


Graph 1.a - In this graph we see the throughput of node A in each of the four scenarios as lambda changes. In this graph, the lambda values of node A and C are the same. This figure demonstrates how the throughput (in Kbps) changes as lambda increases. Generally, as lambda increases, or the amount of 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.

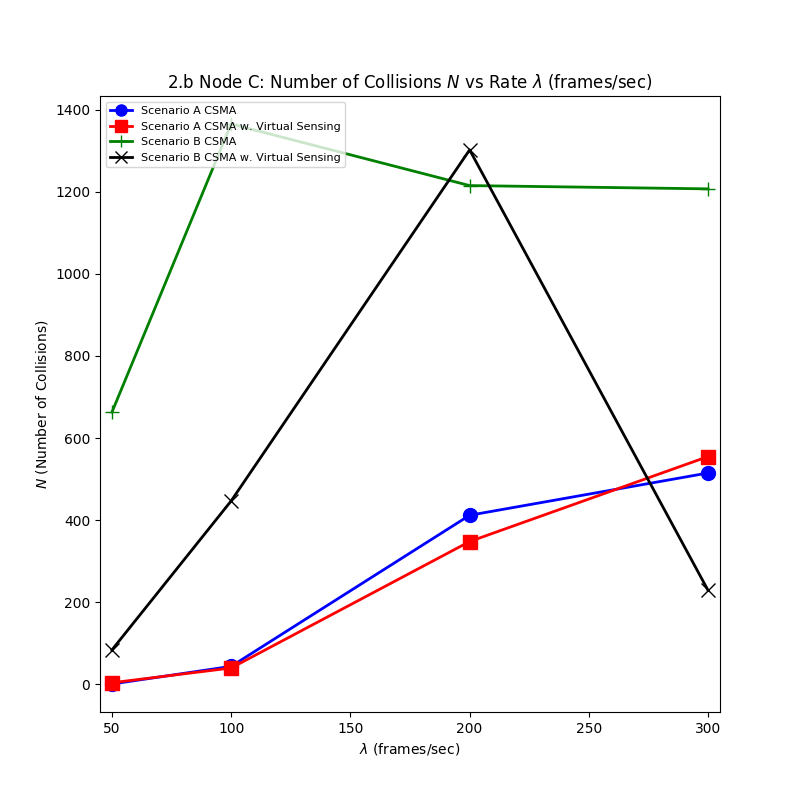
   
Graph 1.b - In this graph we see the throughput of node C in each of the four scenarios as lambda changes. In this graph, the lambda values of node A and C are the same. This figure demonstrates how the throughput (in Kbps) increases as lambda increases. In this graph, we see very similar results to 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.

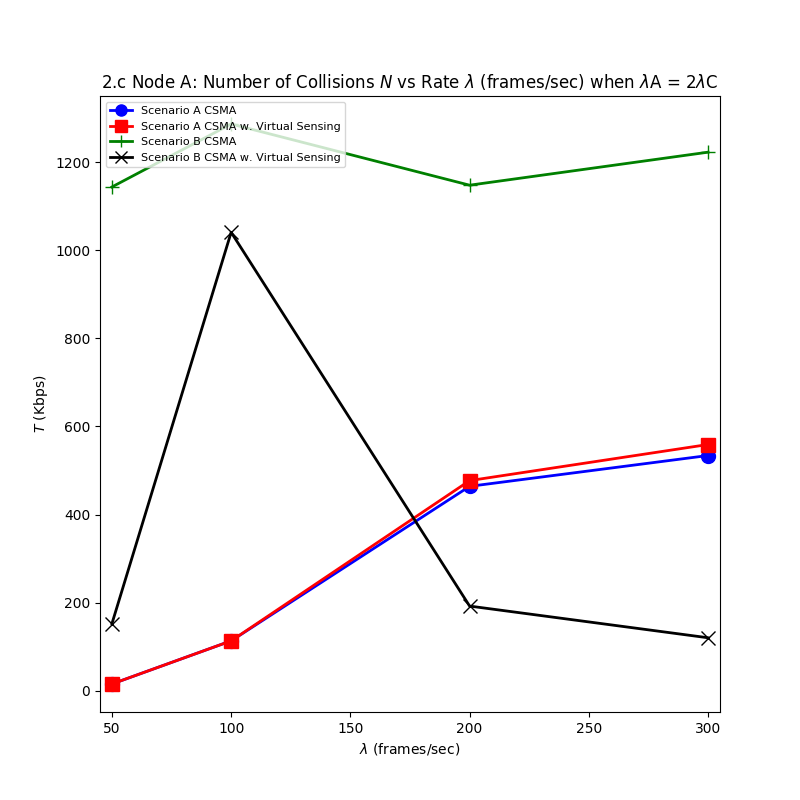
  
Graph 1.d - In this graph we see the throughput of node C in each of the four Scenarios as lambda changes. In this graph, the lambda values of node A are double those of node C. This figure demonstrates how the throughput (in Kbps) increases as the lambda value of station C change. This graph is related to Graph 1.c because the addition of the throughputs at the respective lambdas and scenario should not add up to more than the bandwidth of the medium (6 Mbps), which is seen to be correct. Scenario A with and without VCS are similar than that seen in Graph 1.c because the topology of Scenario A doesn’t add much bias to the station with the higher framerate. The throughput of Scenario B is shown to be lower than it was in Graph 1.C, this is expected because node A had and sent more information which made node C not be able to send as much. This is expected as the hidden terminal topology will be biased to the node with the higher framerate.



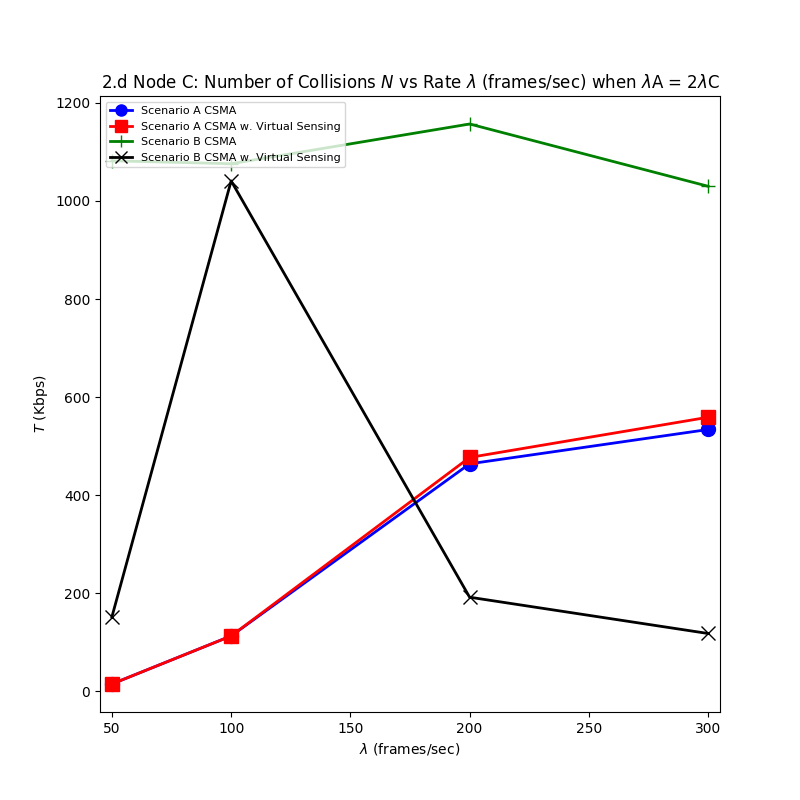
Graph 2.a - In this graph we see the collisions of node A in each of the four scenarios as lambda changes. In this graph, the lambda values of node A and C are the same. This figure demonstrates how the number of collisions increase as lambda increases. This figure shows that Scenario A with and without VCS have a similar number of collisions for all lambda values tested. 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 which will result in no collision. 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 as they are all in each other’s collision domain. The trend for the number of collisions for Scenario A as lambda increases is logarithmic. Scenario B without VCS on average has the highest number of collisions because of the hidden terminal restricting the communication between the two transmitting stations. Scenario B with VCS 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 back off values that will cause collisions. On average, Scenario B with VCS has less collisions than without VCS because it is able to stop the other transmitting station from sending a packet.



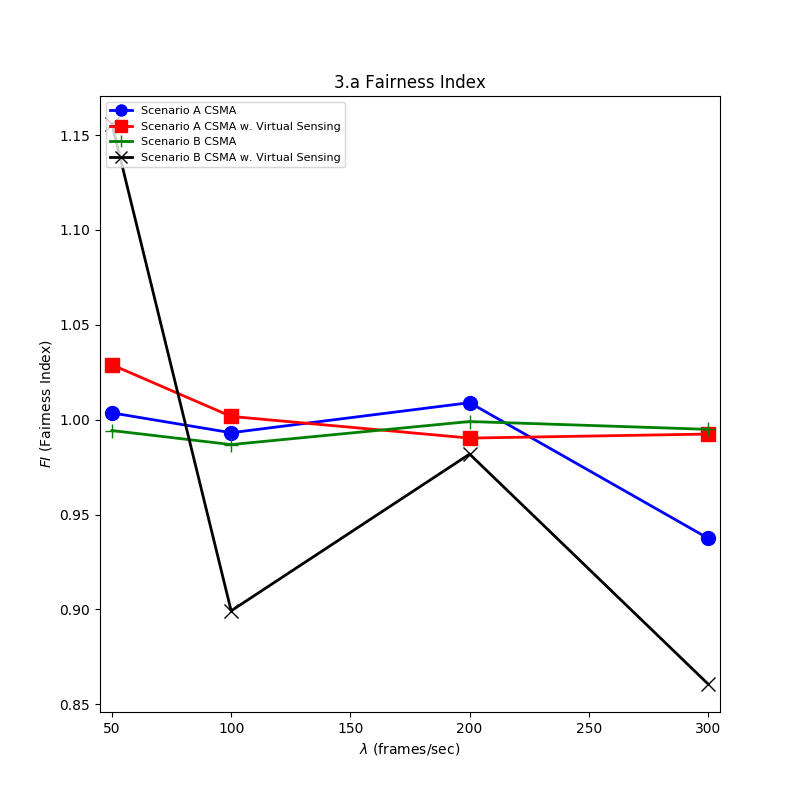
Graph 2.b - In this graph we see the collisions of node C in each of the four scenarios as lambda changes. In this graph, the lambda values of node A and C are the same. This figure demonstrates how the number of collisions increase as the lambda value increases. This graph is similar to Graph 2.a because we only have two sending stations so whenever one station is having a collision the other station has a collision because they are colliding with each other. In Scenario B without VCS, it is possible for one station to get multiple collisions while the other station only has one, if it attempts to resend a packet multiple times while one station is already transmitting. This is a rare phenomenon as the station would have to pick a very small back off value in order to attempt to send a packet multiple times while the other station is still sending one packet.



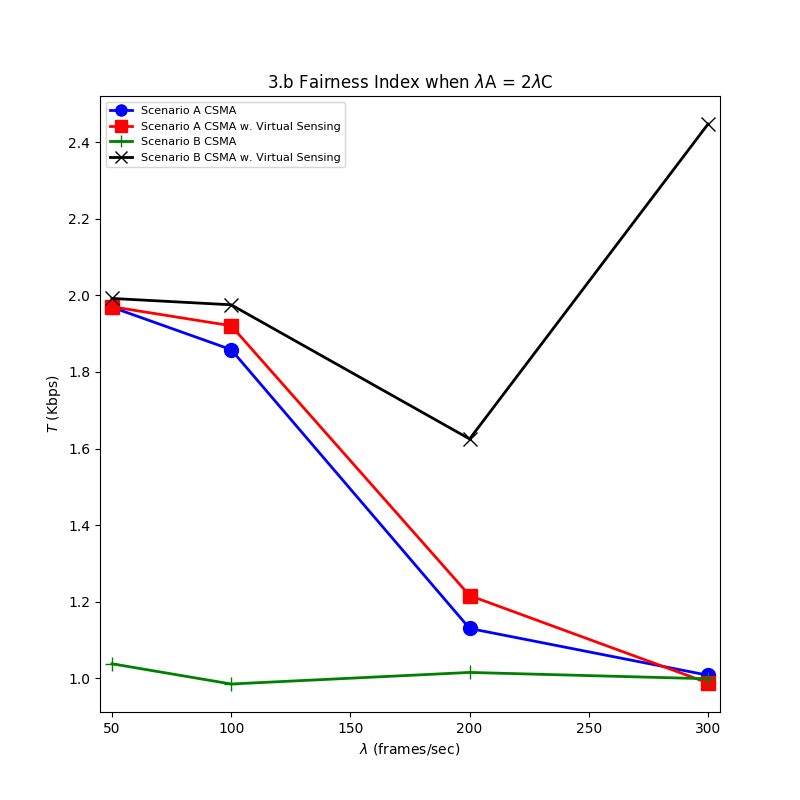
Graph 2.c - In this graph we see the collisions of node A in each of the four scenarios as lambda changes. In this graph, the lambda values of station A are double those of station C. This figure demonstrates how the number of collisions increase as the lambda value of node C increases. The shape of 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 VCS is 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 tells us that this is a lambda value which based on the parameters used causes a high number of collisions.



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. This figure demonstrates how the number of collisions increase as the lambda values of node C change. This graph shows some of the same characteristics that were discussed in Graphs 2.a, 2.b, and 2.c such that there is a high spike in collisions when the lambda value of C is 100 frames/sec and that Graphs of 2.c and 2.d are similar because the majority of the time the two stations will have 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. This figure demonstrates how the fairness index changes as the lambda values change. In general, the fairness index of the nodes in all Scenarios are close to 1. This makes sense for both Scenario A and Scenario B as the lambda values are similar, which means that there is no bias of a particular station.



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. This figure 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 in Scenario B without VCS. In this scenario, there is no way for the other station to know that the other is transmitting so there will be always be a competition for the medium, causing the fairness index to always be a value close to 1. Lastly, Scenario B with VCS 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 back off values. Then when one of the nodes is ready to transmit, it is possible that the other station is frozen with a large back off value. The first station will finish sending and the back off counter will reset back to in-between zero and four while the other node that was frozen is still counting down from the large back off when there was contention for the medium. This will make the node that just sent information send another packet because its back off 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.