Andrew Camps
Derek Paris
October 10, 2017
ECE 478 Project 1
Loukas Lazos

The Distributed Coordination Function of 802.11

As a team Derek Paris and Andrew Camps studied the performance of multiple access protocols in a wireless setting. The project studied two different topological orderings of nodes, and two different setups of Carrier Sensing Multiple Access (CSMA) protocols, all at different frame rate distributions. To complete this project as a team we decided to meet up on multiple occasions and work on creating a simulation for each test. We worked together to list out the many possible edge cases, then design four separate simulations which would be able to calculate all of the different metrics required by this study. After working out much of the needed logic, Andrew began to code the simulations for the first few scenarios. After the framework for the simulation program was developed Derek and Andrew peer programmed many of the more logically intense edge cases. Derek spend most of the time debugging while Andrew was coding the main simulations. Once all simulation code was complete and tested, Andrew produced the needed data and created the figures in the report while Derek styled and wrote some of the descriptions for each figure.

In order to test all of the different scenarios in the project, our team decided that we would develop this simulation using C/C++. The program was developed to output data that could be used to accurately compare the difference in performance metrics for two topologies, multiple access protocols and rate sets. The two topologies studied in this simulation were nodes with parallel transmissions in the same collision domain (Topology A) and a topology where two nodes transmitting to a common receiver that are hidden to one another (Topology B). Both of these topologies were tested with two different multiple access protocols which included CSMA with collision avoidance and CSMA with collision avoidance using virtual carrier sensing. To do this, a standardized testing method was used. The simulation consisted of a single main.cpp file which, when compiled and ran, outputted average calculated performance metrics for a given rate set of all test scenarios. Two main data outputs were used meaning the program was run twice, getting the data for both node A and node C having the same rate set and node A having a doubled rate set. Given these 1:1 or 2:1 ratio rate sets for each node, the program was run and all

four simulation scenarios were conducted 500 times with different generated Poisson-distributed traffic upon each loop. In each of the 500 loops, node A generated a Poisson-distribution and node C generated a separate distribution given one of the four frame rates from the rate set. With these individual distributions for each node, all four simulation scenarios were run with the same distributions. This ensured that all scenarios could be compared with consistent frame distributions for each node. Scenario functions themselves were passed the individual node data distributions where a main clock was used to drive a scenario simulation for 10 seconds. Scenarios took into account all possible cases where packets were ready to transmit and collisions could potentially happen. After all four scenarios ran, the performance metrics were stored in a running average, and the next iteration test occurred. Average data was then output at the end of the program. This data for all scenarios included: average throughputs, collisions and fairness indexes. See below for the graphical representation of each simulation

Test Result Figures and Explanations

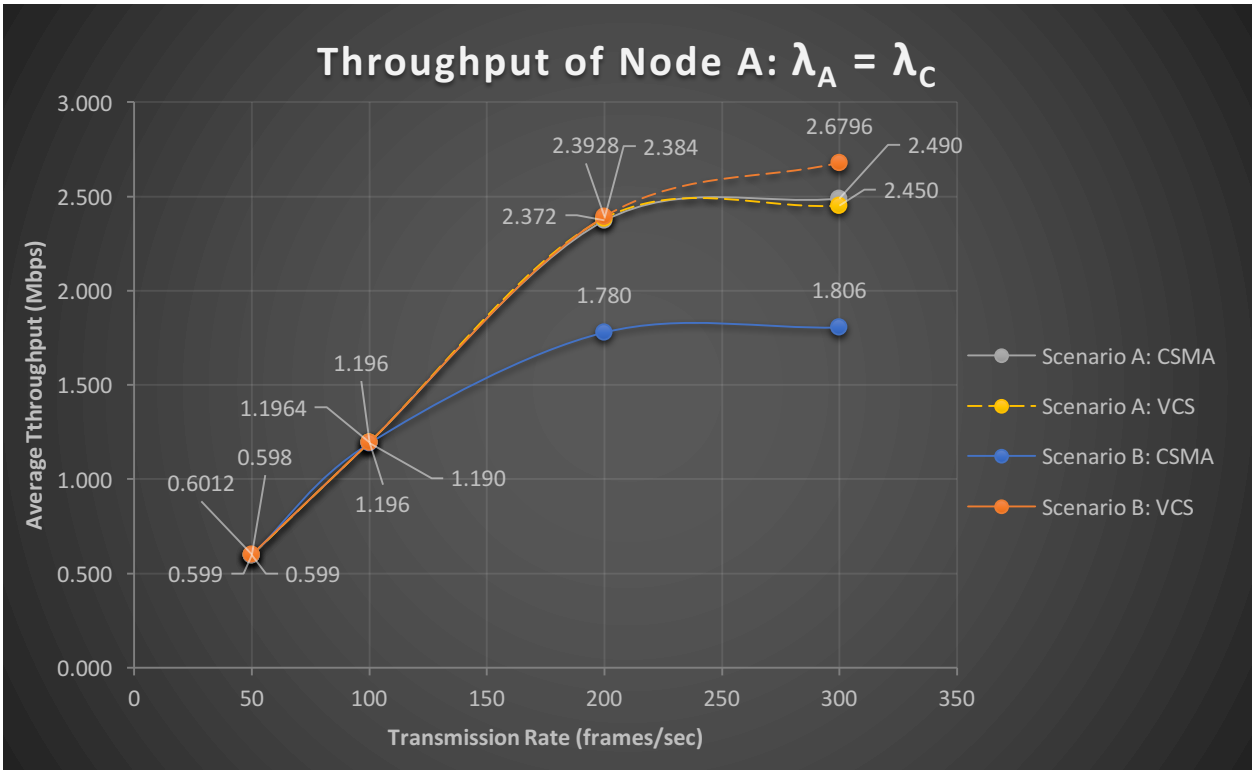

Figure 1: Average Throughput of Node A ($\lambda = \lambda_A = \lambda_C$)

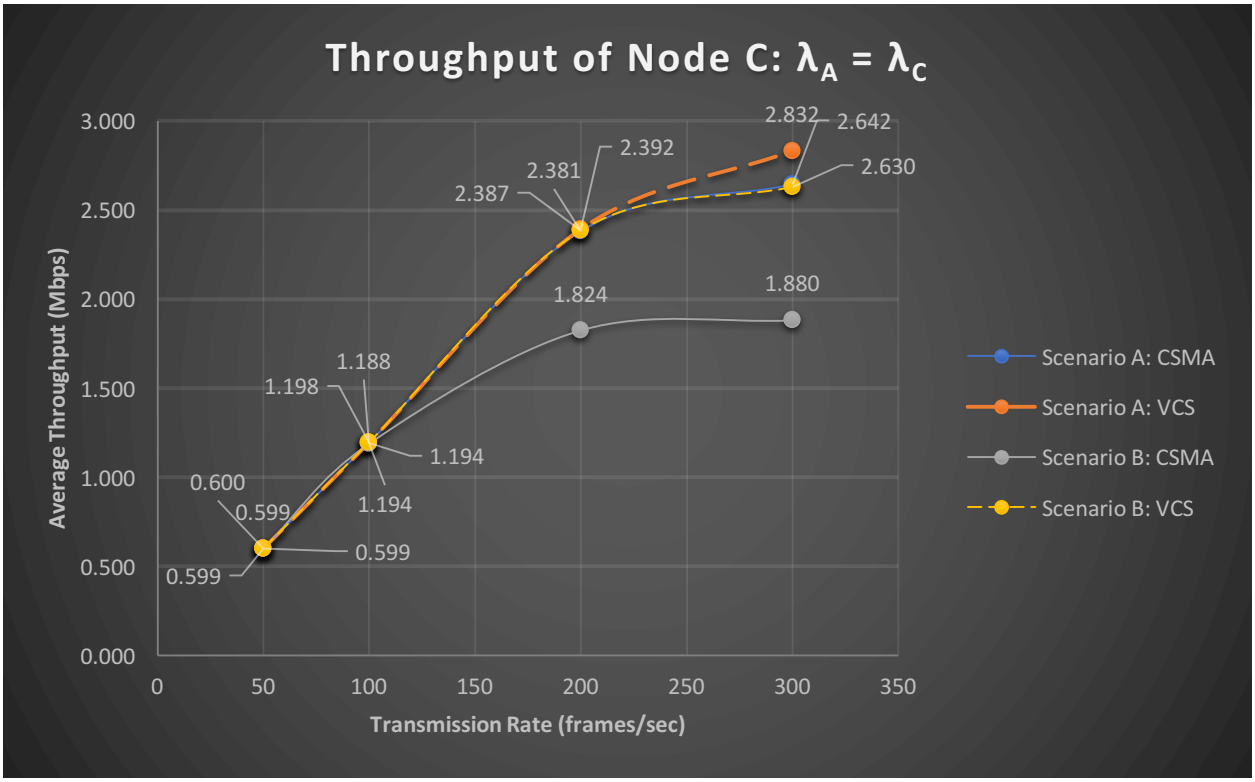

Figure 2: Average Throughput of Node C ($\lambda = \lambda_A = \lambda_C$)

Figures 1 and 2 shown above provide a representation of the average throughput of node A and node C respectively between both topologies, tested with both protocols while $\lambda = \lambda_A = \lambda_C$. Each of the points on the colored lines are an average of 500 tests between four different frame rates. The figures show that as the frame rates increased in the scenario combinations, there is a steady rise in the throughput for both nodes transmitting. As the frame rate increased further, the throughputs started to level out. To explain this, when starting at a low frame rate for both nodes, there is never a large backlog of packets ready to send. Node A and node C are able to send a packet easily and with lots of space between packet arrivals. There is also not much traffic from each node competing with one another. With this lower frame rate, there are few packets in total to send as well so the throughput starts out small with this being calculated by the total amount of data sent over the period of time (10 second simulation). As the rate increases, however, the throughput rises for both nodes until the rate reaches a point where the nodes begin to backlog incoming packets and the throughput steadies out.

The first two scenarios in figure 1 and figure 2 as well as the last, all have a similar throughput which means that there were few collisions or less collisions than the scenario that

started to drop in throughput shown in both figures. The amount of collisions and how this affects throughput will be discussed in later figures. However, when nodes A and C are in the same collision domain in Scenario A they are able to sense when one another are each sending packets. This inherently caused less collisions between the nodes allowing for a higher throughput as the packet rate increased. With the hidden terminal problem in Scenario B the CSMA with virtual carrier sensing (VCS) allowed for the hidden nodes A and C to communicate to each other. This is very similar to nodes in Scenario A, thus having a similar throughput. The nodes do this by each sending out a RTS packet and receiving a CTS packet from the receiver when it is good to start transmitting the actual data. The only time period when data can collide is when the RTS packets are being sent. Scenario B without VCS levels out at a lower throughput value. As the frame rate increases more and more collisions occur because there is no way for nodes A and C to talk to one another. Thus, many collisions occur as the nodes send their data packets blindly to the reviver node causing a lower throughput at these higher frame rates.

Lastly, a point that is worth touching on are the throughputs between nodes A and C. If the values are compared, each node starts out with close to the same throughput at lower frame rates, but they begin to separate as the rates increase. This will be touched on more in depth when talking about the fairness index in the below figures.

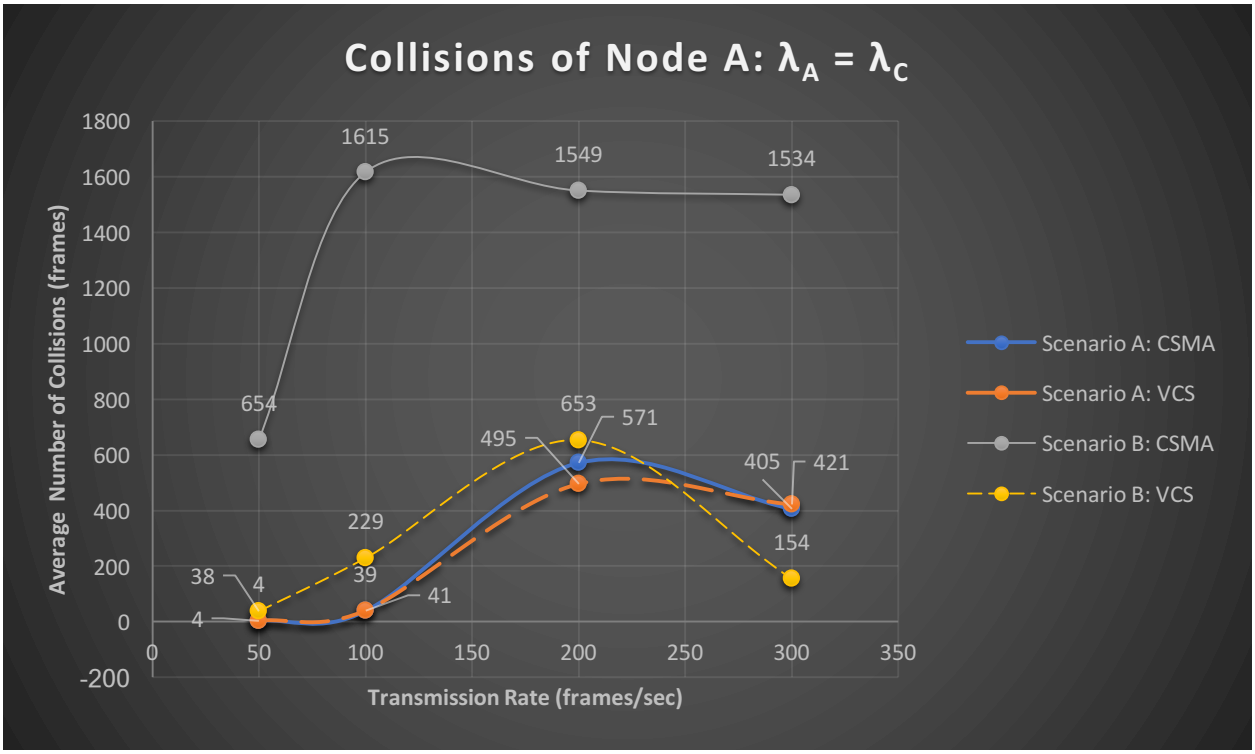

Figure 3: Average Collisions of Node A ($\lambda = \lambda_A = \lambda_C$)

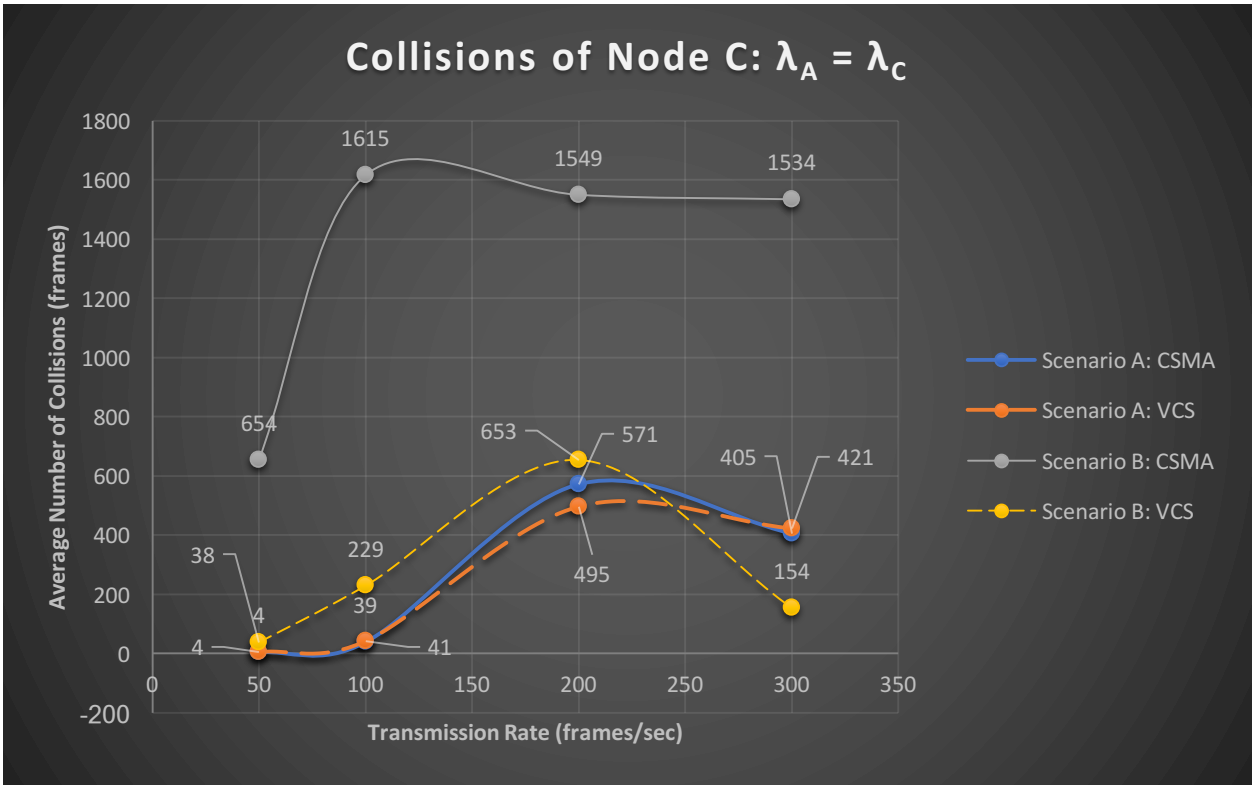

Figure 4: Average Collisions of Node C ($\lambda = \lambda_A = \lambda_C$)

Figures 3 and 4 show the average number of collisions over 500 tests between node A and node C respectively. The frame rate sets are still the same as figures 1 and 2 with $\lambda = \lambda_A = \lambda_C$. Before digging into the graphs above, you will notice that the graphs for node A and C are identical. This is due to the way our team defined a collision. We said that a collision could only be caused by another transmitting node and when one node's data collides with another node's data, they both have a collision counter that is updated. One node cannot have a collision without the other node involved, therefore each node's count of collisions is identical.

We can turn back to analyzing the graphs above and similar to figures 1 and 2, there are three scenario combinations that seems to follow a similar pattern when it comes to collisions as well. These three scenarios correspond the same three in figures 1 and 2. The overall pattern of these lines also follow a similar flow to the above figures. At lower rates, there are a lower number of collisions due to the large spread between packet arrivals and there are less packets to collide with. At higher rates, the number of collisions increase until a peak number of collisions are reached. Once this value is hit, as the rates increase, more packets arrive sooner and sooner causing a backlog to be built up. When this occurs, packets begin to start transmitting at the same time and in order for a collision to occur, the nodes need to have the same back off time. This is also where one node will start to lock out another, because one node might consistently have a lower back off time. This concept is explained in a later figure.

Another point is that these three lines seem to have a bit more variation in the amount of collisions between topologies and protocols used. Just comparing these three lines, both scenario A tests (blue and orange) stay consistently together in number of collision. However, in topology B (yellow) there are more collisions in the beginning due to the hidden terminal setup. Due to the nodes sending the RTS packets blindly there are a higher number of collisions.

Lastly, what is most noticeable with these graphs is the grey line in Scenario B. As you can see this line shows that Scenario B using CSMA with no VCS has a large amount of collisions. This is directly responsible to the lower throughput shown in figures 1 and 2. These large amounts of collisions are due to the hidden terminals and the way CSMA avoids collisions. With the nodes A and C hidden from each other, no one node can take control of the medium like the other three scenarios. CSMS without VSC doesn't use a packet in the beginning to tell the receiver that a node is about to transmit, so it can tell other nodes not to. Both nodes just simply transmit when their back off counter finishes. This causes many more collisions to occur

at the receiver which is shown in the figures. This problem is solved with VCS added with the drastic improvement of collisions.

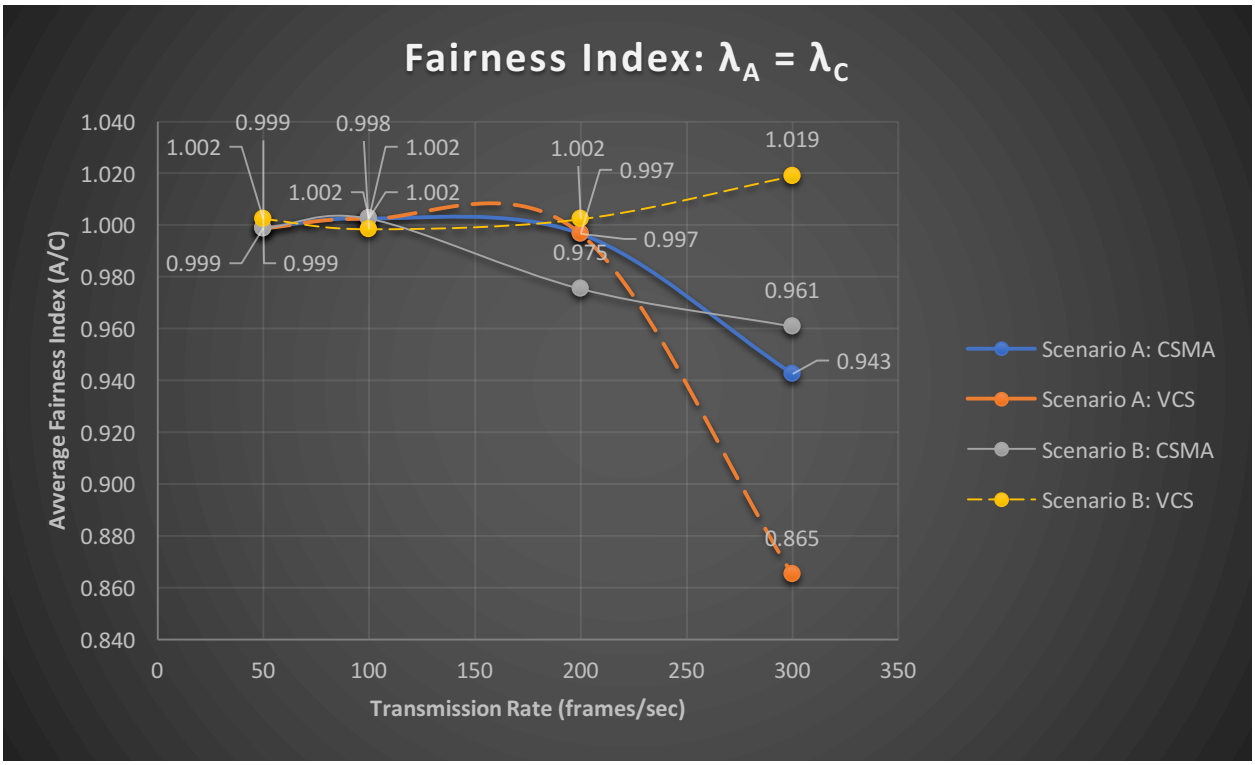

Figure 5: Fairness Index ($\lambda = \lambda_A = \lambda_C$)

Figure 5 shows the average fairness index (FI) for the same tests as above. This was defined as the amount of time that node A was sending data or had control of the medium over the amount of time that node C had control. As you can see on the graph the lines stay consistent for the most part at lower frame rates, but start to vary at the higher rates. This varied upon each run of the program with either some of the end points increasing or decreasing. What should happen at all the rates is that the FI should be more or less flat (FI = 1 approximately). For the most part this is true from rates 50 – 200, but it begins to taper off at the higher rates.

If we were to run more than 500 iterations, say 1000, the higher rates would begin to move closer to FI = 1. They taper because with random packet distributions sometimes one node tends to lock another out. This occurs when there are so many collisions and each node's backoff time grows so large, that as soon as one node transmits successfully and resets its backoff time, it can rapidly transmit packets while the other node is stuck backing off. On some runs of the program with

500 iterations, node A would get more transmission time resulting in an FI > 1 and other time node C would get more runtime resulting in an FI < 1. This is why in figures 1 and 2 that node A and C have a slightly different throughput at high rates. This, however, will not be true when each of the nodes have different frame rates, shown in the next set of figures.

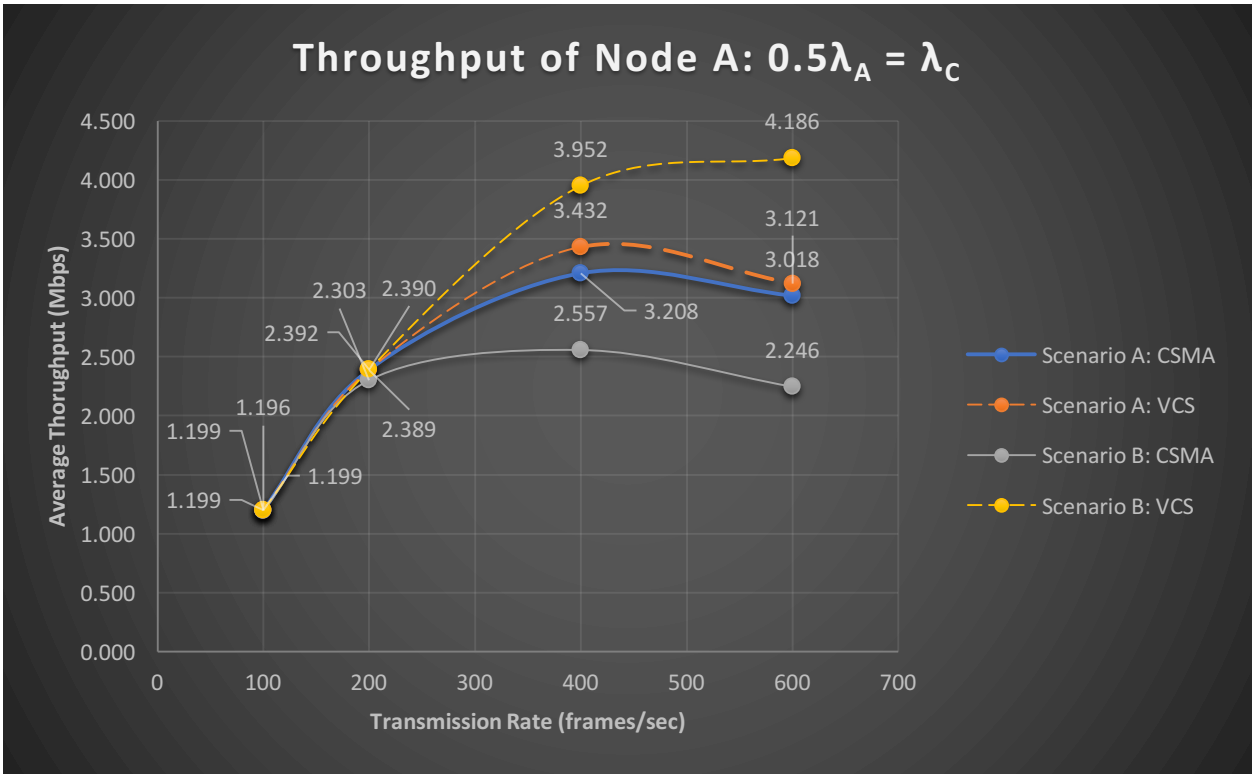

Figure 6: Average Throughput of Node A ($\lambda = 0.5\lambda_A = \lambda_C$)

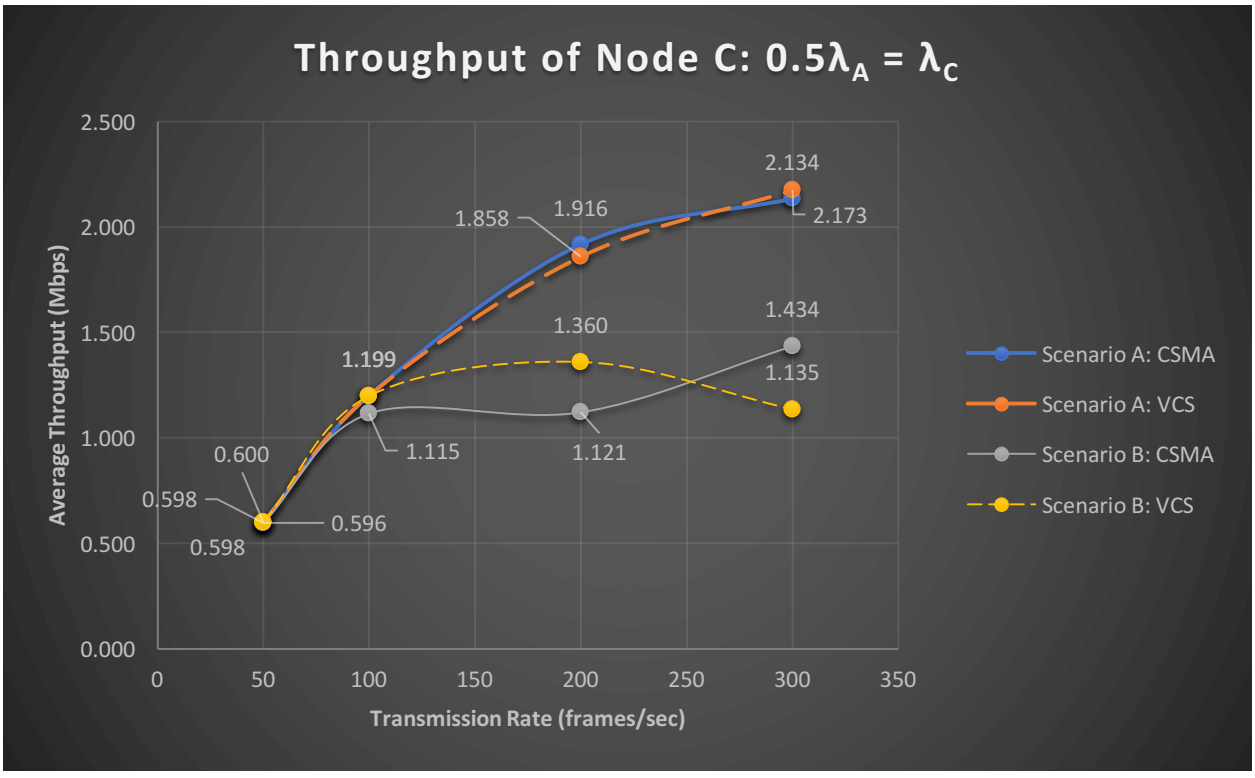

Figure 7: Average Throughput of Node C ($\lambda = 0.5\lambda_A = \lambda_C$)

Figures 6 and 7 again show an average throughput between nodes A and C. However, this set of test data had a modification to the frame rate ratio between node A and node C. In these figures, what is interesting are the differences in throughput values between node A and node C. With node A have a doubled frame rate compared to node C, we would expect that node A would be able to transmit more packets and have a higher throughput. When you compare figures 6 and 7 you can see this is definitely true. Immediately at the first frame rate test, node A has nearly a double throughput compared to node C. As the rates increase the doubled throughput seems to taper off, but A still has a higher throughput than node C at all rates.

Another thing to notice is that at higher frame rates node A seems to be more affected by the increase in frame rate of C. In some of the scenarios when the frame rate is at the maximum, throughput actually starts to decrease. In figure 7 for node C at the end, some of the throughputs continue to rise whereas in figure 7 they begin to decrease. This is due to the competition that C starts to have with A. Node C begins to block node A out more and more, unlike node A consistently blocking node C out due to the doubled packet rate. However, the line patterns should continue to be similar to the line patterns of figures 1 and 2, which is true.

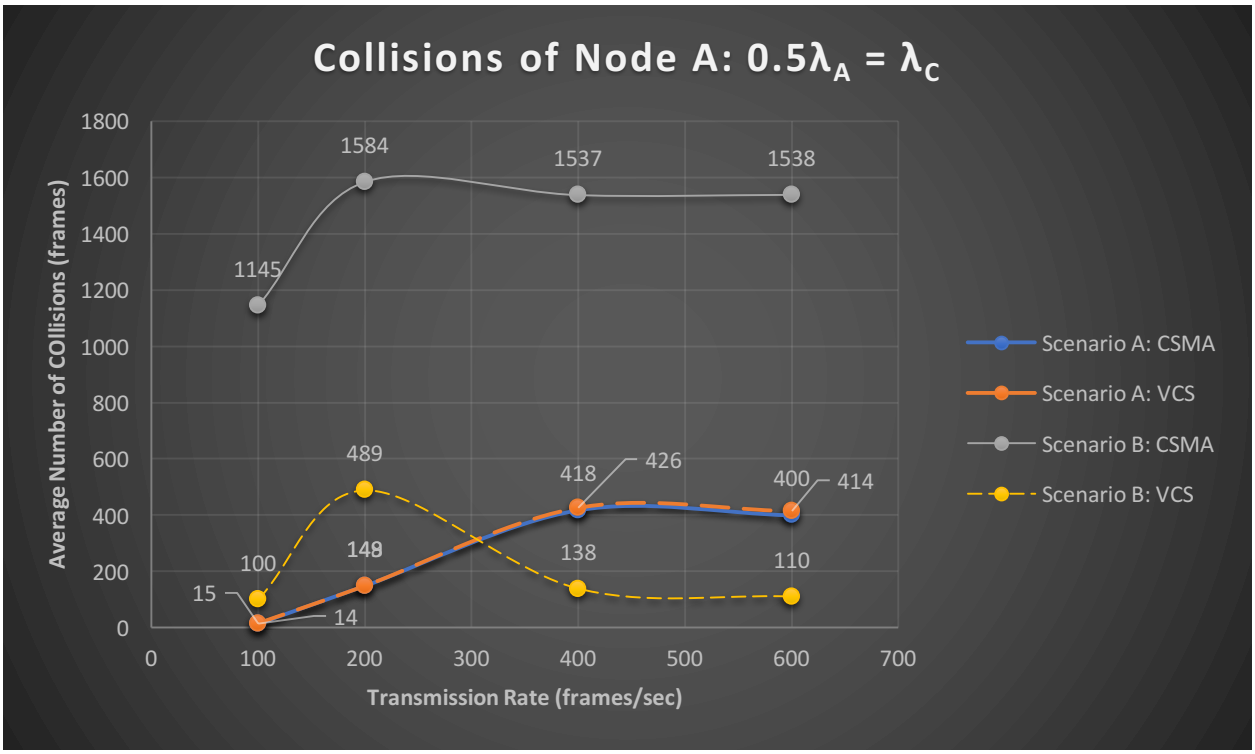

Figure 8: Average Collisions of Node A ($\lambda = 0.5\lambda_A = \lambda_C$)

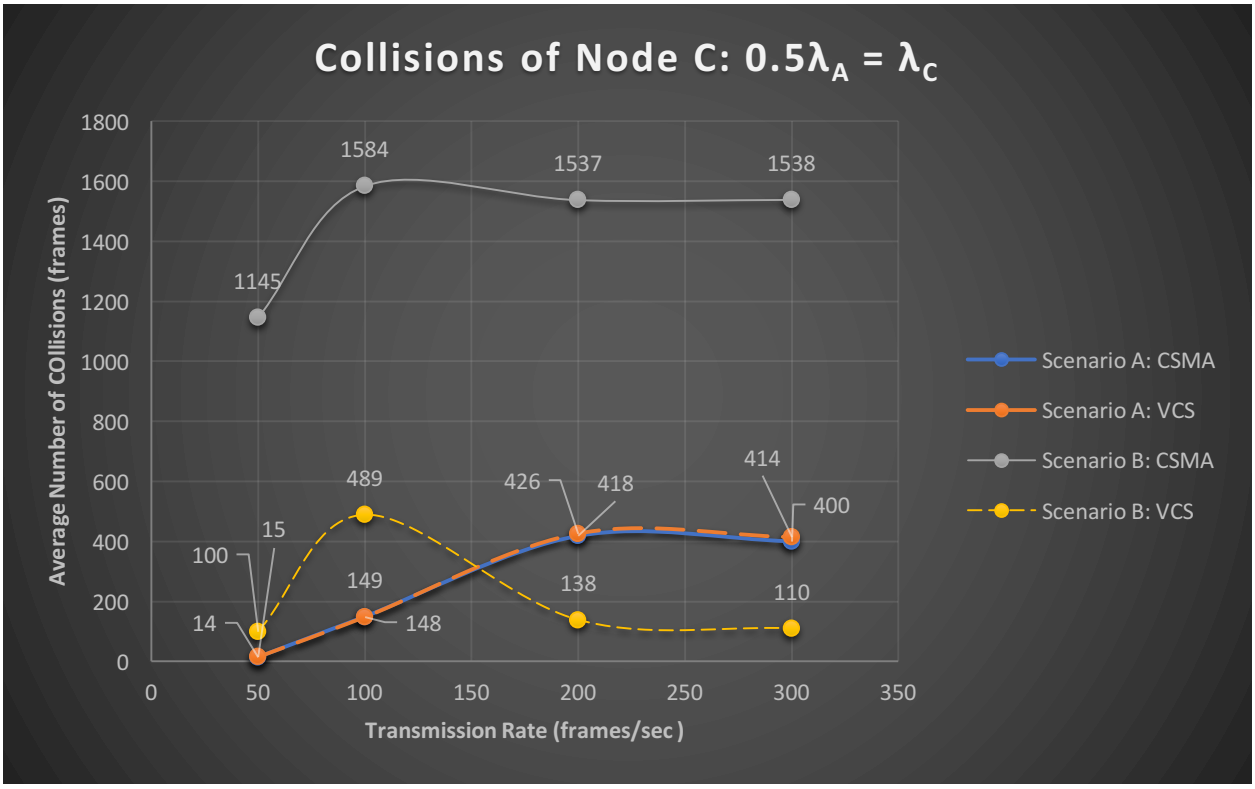

Figure 9: Average Collisions of Node C ($\lambda = 0.5\lambda_A = \lambda_C$)

Figures 8 and 9 again show the average number of collision frames for node A and C with this rate ratio modification. Because we defined collisions the way we did, there will be no difference of collisions between node A and C. However, if we look at these figures and compare that to figures 3 and 4 with 1:1 rate ratios, we can see a slight variation in the graph patterns. Like in figures 3 and 4 at higher rates, the collisions will start to even out to a point if the rates continue to increase and the collision numbers will stay about the same. What can be seen in figures 8 and 9 is this evening out point happening sooner. This is caused by the overall total frame rate between nodes A and C being higher in this test case caused by node A having a doubled frame rate.

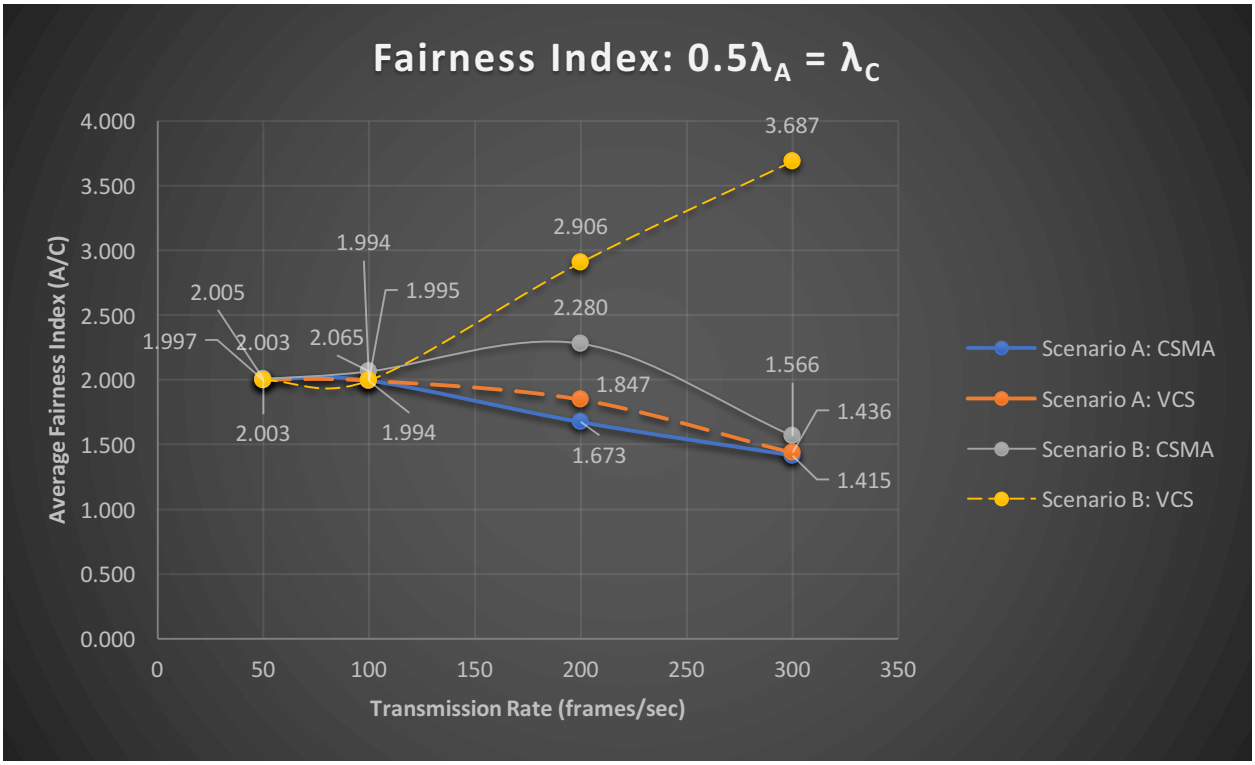

Figure 10: Fairness Index ($\lambda = 0.5\lambda_A = \lambda_C$)

Figure 10 shows the average fairness index (FI) for this 2:1 ratio of rates between A and C. With all of the rates doubled for node A someone's intuition might cause them to think that node A would have double the amount of transmission time or time blocking the medium. This is more or less true shown in the figure above. At the lower rates when there is a lot of space between incoming packet arrivals, the FI for node A is almost exactly double. Both node A and C are most likely to be able to transmit all of their packets with minimal collisions shown in the collision graphs above. However, as the rates begin to increase node C begins to take back the competition and brings down the time that A gets to transmit. This is shown with falling lines from the first three scenarios.

What is interesting is that the last scenario (yellow) with node A continues to rise in the amount of time that it has the medium. This can be explained by how VCS works with the hidden terminals. With the doubled rates of node A, node C doesn't have a chance due to the small collision window that is created with the RTS packet. As soon as node A and node C start colliding, their back offs will increase, but node A will almost always have a successful packet transmit first because of the doubled rate. With the small size of the RTS packet it will get a CTS

response after 2 slots and C will have to wait for node A to transmit all of its data. This creates many long waits while node C is waiting for its back off to decrease or with a frozen back off.