ECE 578 - Project 1

Tianyang Chen - tianyangchen@email.arizona.edu

SID: 23369298

Xiao Han – xiaoh@email.arizona.edu

SID: 23361457

October 2016

1. Introduction

In this report, we focus on the performance of the 802.11 DCF in different network topologies, and investigating the correlations between the performance and network behaviors (λ). There are two different topologies we need to analyze. First, exposed terminal, in which all network nodes are within the same collision domain and the communications between different nodes are concurrent, is shown in Figure 1(A). We refer to this topology as Scenario A. Second, hidden terminal, in which the nodes are on the same lines and the sending nodes are in the different collision domains, is shown in Figure 1(B). We refer to this topology as Scenario B. Finally, we will utilize the CSMA/CA and CSMA Virtual Carrier Sense respectively to these topologies and compare the performance under the different conditions.

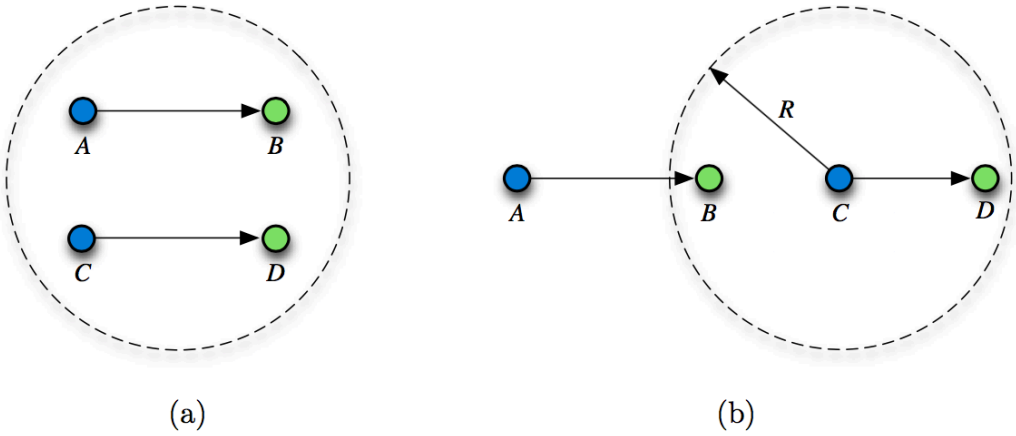


Figure 1

In this project, Xiao Han deals with part of exposed terminal programming and Tianyang Chen deals with hidden terminal programming, then Xiao Han and Tianyang Chen worked together to debug and verify the final models and the report. Tianyang Chen analyzes the outcomes of Throughput and Fairness Index, Xiao Han analyzes outcomes of collision and delay. For the project report, Tianyang Chen finished Simulation development, throughput and Fairness Index of Graphs of report, and Xiao Han finished Introduction, Collision and Delay part.

2. Simulation development

Here we use MATLAB to do the simulation.

2.1 Develop the mathematical model

Here, we use the duration of slots as the unit of time, which means 1 second can be separated into 500,00 slots. And in each slot, we updated the status of node A and node C. We use different numbers to represent different states of the node, which were shown in the following table.

Status	Number
idle	0
DIFS	1
CW	2
DATA	3
SIFS	4
ACK	5
RTS	6
CTS	7

2.2 algorithm

Here we implement some approximation. The SIFS duration is 10 μs, which equals to 0.5 slot. We rounded it to 1 slot. And the 30 bytes ACK that equals to 2 slots, for the convenience of calculation, we rounded it to 1 slot.

2.2.1 CSMA/CA

```
initialization;
begin;
for t=1:50000
    if (sensing DIFS time)
        node(t)=1;
        else if (CW counting down)
```

```
            node(t)=2;
            else if (sending data)
                node(t)=3;
                else if (Waiting SIFS)
                    node(t)=4;
                    else if(ACK)
                        node(t)=5;
                        else collision;
    end if;
end for;
end;
```

2.2.2 VCS

```
initialization;
begin;
for t=1:50000
    if (sender node is freeze)
        invoke the exponential backoff mechanism;
        else if (sending RTS)
            node(t)=6;
            sensing collision;
            else if (waiting for SIFS before CTS)
                node(t)=4;
                else if (receiver node is freeze)
                    invoke the exponential backoff mechanism;
                    else if (receiving CTS)
                        node(t)=7;
                        else if (waiting for SIFS before DATA)
                            node(t)=4;
                            else if (sending data)
```

```
                node(t)=3;
                else if (Waiting SIFS)
                    node(t)=4;
                    else if(ACK)
                        node(t)=5;
    end if;
end for;
end;
```

3. Graph for each of the simulated scenarios

3.1 Throughput

For this project, our equation for throughput is shown below.

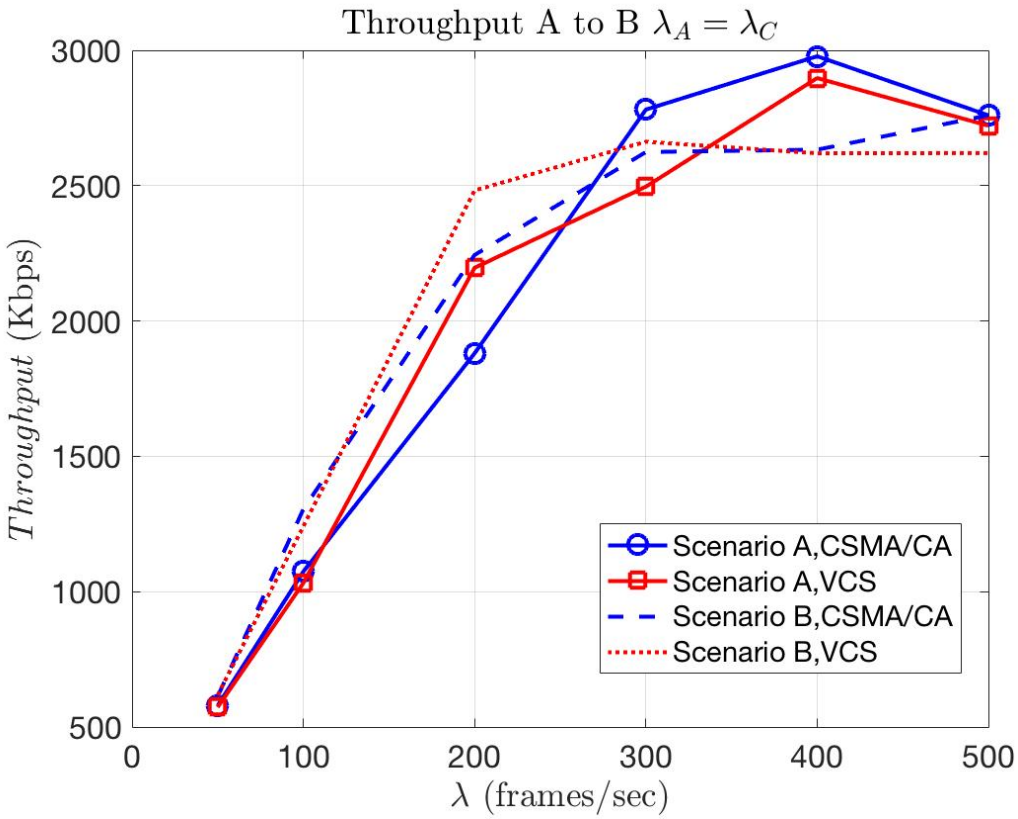
$$T = \frac{(Number\ of\ frames\ sent\ successfully)\times(Data\ frame\ size)\times 8}{Last\ frame\ received - First\ frame\ transmitted}$$

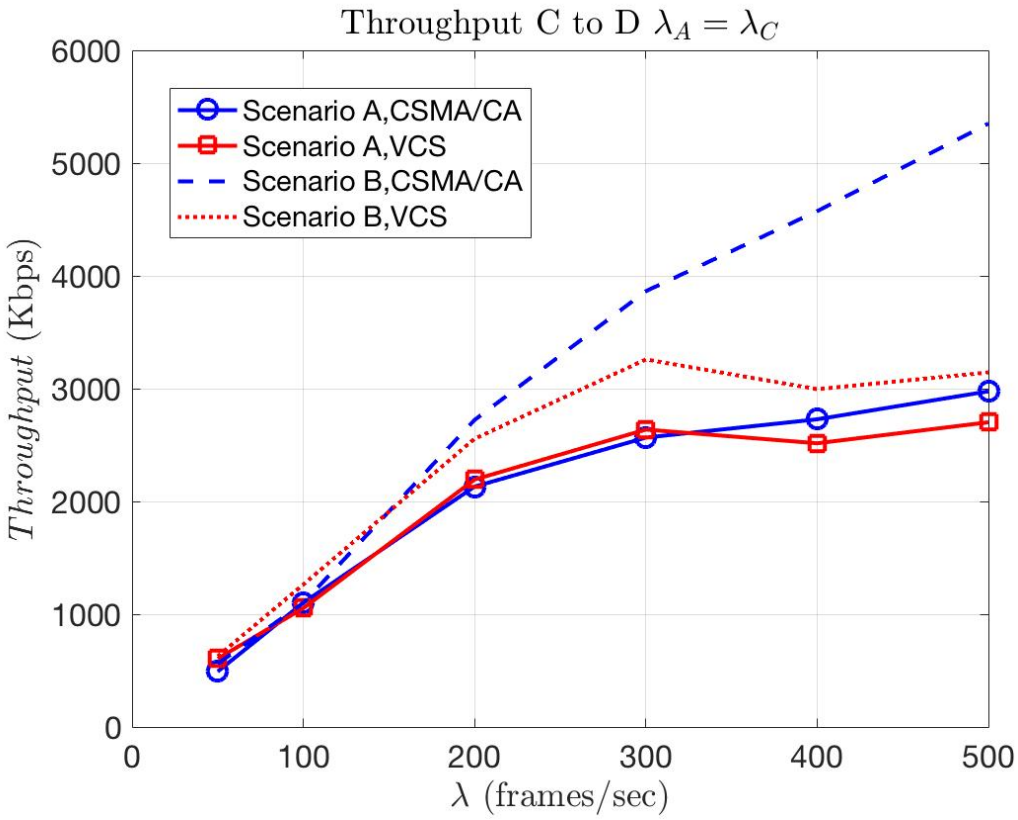
We assume that we take one frame into consideration only if it's fully transmitted and the sender receives an acknowledgement. If the sender doesn't receive an acknowledge or only send some parts of this frame at the end of simulation time, we wouldn't count this frame into this equation.

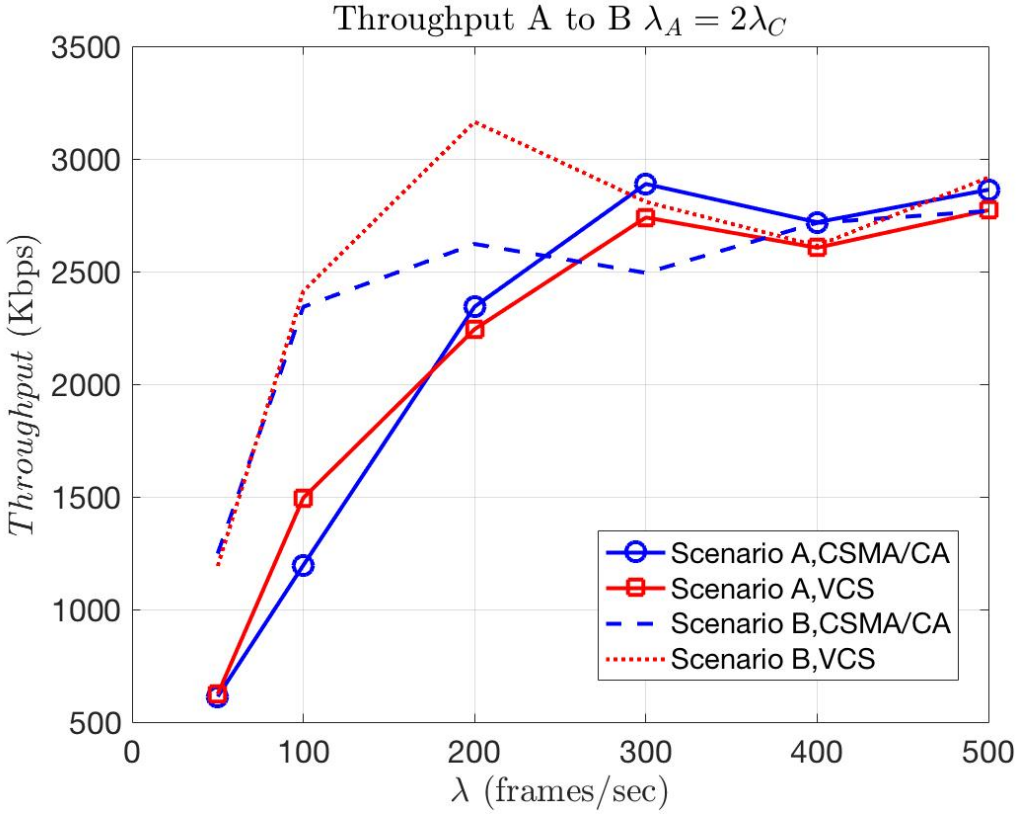
In scenario A, $\lambda_A = \lambda_C$ and $\lambda_A = 2\lambda_C$. First, with the increment of λ, throughput is larger than before. However, the increment of throughput would stop, or even decrease when λ is larger than 300 frames/sec. We deem that there would be more collisions when the λ becomes larger. Second, throughput isn't improved much when we enable CTS/RTS. This is easy to understand, because CTS/RTS was used to sense the state of the channel, which is either busy or idle. But scenario A has all four nodes in one domain, they can sense each other, which has the same effect as CTS/RTS.

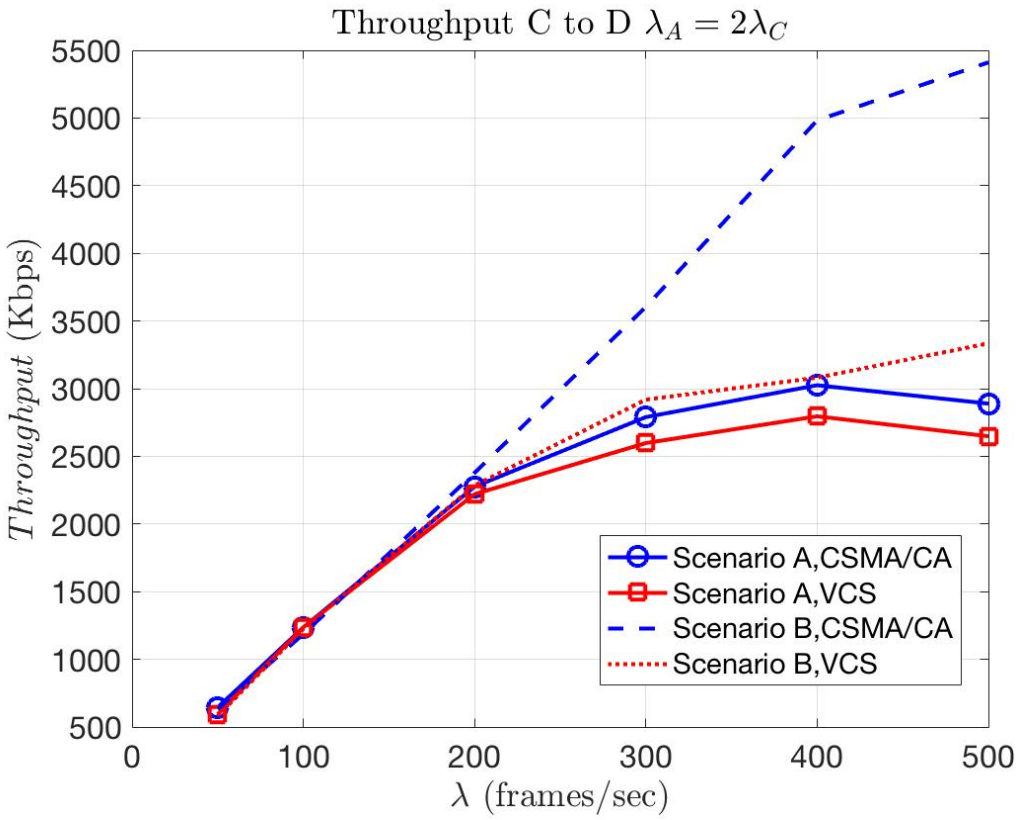
In scenario B, $\lambda_A = \lambda_C$ and $\lambda_A = 2\lambda_C$. The curve of node A looks similar no matter CTS/RTS was enabled or not. It will also increase with the growth of λ, and will become stable when λ is larger than 300 frames/sec. Here, we assume 50 frames were

transmitted per second, it's not a large number, so the improvement after enable VCS isn't obvious. If we assume more frames generated per second, this improvement might be obvious. As for node C, the throughput became much lower after enable VCS, and one interesting thing is that the curve became similar to the curve in scenario A after enable VCS. This is because when using CSMA/CA, there's no restriction on transmission from C to D. After implement VCS, it seems C can sense the state of the hidden node A, because after B sending CTS, node C will defer from transmission and avoid collision.




Throughput C to D $\lambda_A = \lambda_C$
Scenario A,CSMA/CA
Scenario A,VCS
Scenario B,CSMA/CA
Scenario B,VCS
Throughput (Kbps)
λ (frames/sec)
0
1000
2000
3000
4000
5000
6000
0
100
200
300
400
500


Throughput A to B $\lambda_A = 2\lambda_C$
Scenario A,CSMA/CA
Scenario A,VCS
Scenario B,CSMA/CA
Scenario B,VCS
Throughput (Kbps)
λ (frames/sec)
500
1000
1500
2000
2500
3000
3500
0
100
200
300
400
500



3.2 Collision

At the beginning, it's very important to analyze that where and when collisions would happen in the topologies and scenarios. Let we discuss this.

First, in scenario A. The collision would happen at Node B and D when Node A and Node C transmits frames concurrently if we don't utilize Virtual Carrier Sense. It depends on the size of frames and the contention window selected by each Node. When we utilize Virtual Carrier Sense, collisions would happen at Node B and D when Node A and Node C broadcasts RTS to Node B and D concurrently, then Node A and Node C would double contention window and select it randomly again.

Second, in scenario B. Under the condition of no Virtual Carrier Sense, collisions are very possible to happen because of hidden terminal topology. If Node A is transmitting frame, Node C wouldn't freeze the contention window because Node C can't listen up to Node A, means that Node C doesn't know Node A is transmitting. So, Node C would

transmit its frames when CW=0, then collisions would happen at Node B. However, there wouldn't have collisions if Node B sends an ACK to Node A at the same time that Node C sends an RTS to Node D, even though they are transmitted simultaneously. When we utilize Virtual Carrier Sense, collisions would happen when Node A and C broadcast RTS to Node B concurrently.

Now, look at these simulation results.

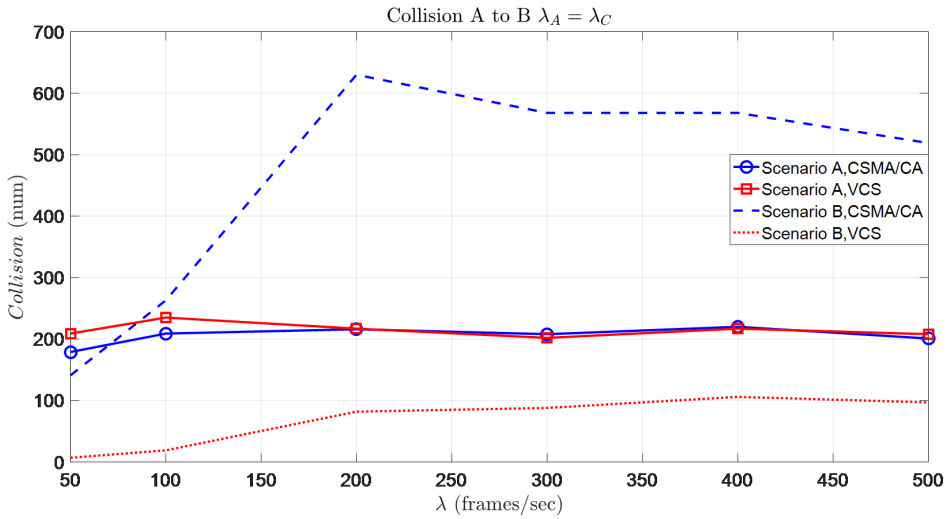
First, we focus scenario A and scenario B when $\lambda_A = \lambda_C$.

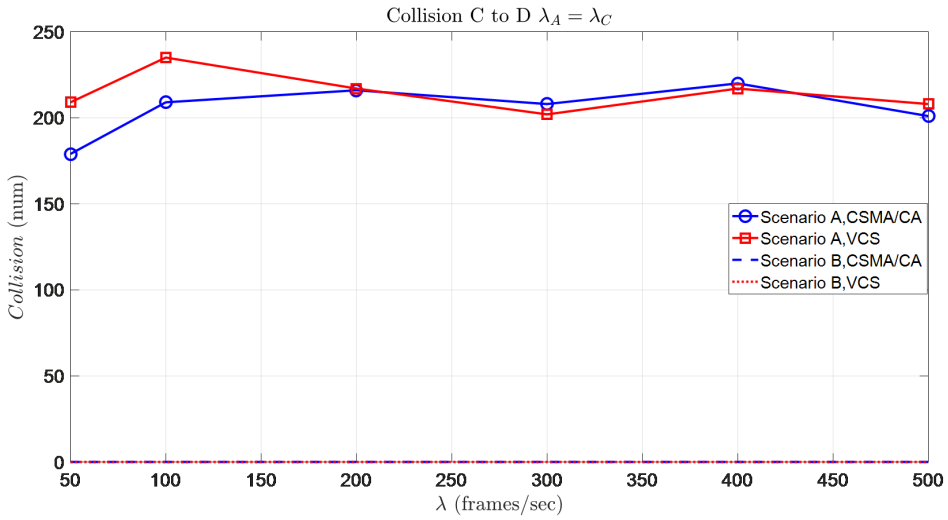
In scenario A, collisions would only happen when Node A and Node C are transmitting frames or RTS simultaneously. So, the performance of this protocol doesn't change too much. The numbers of collisions at Node A and Node C are approximately 200.

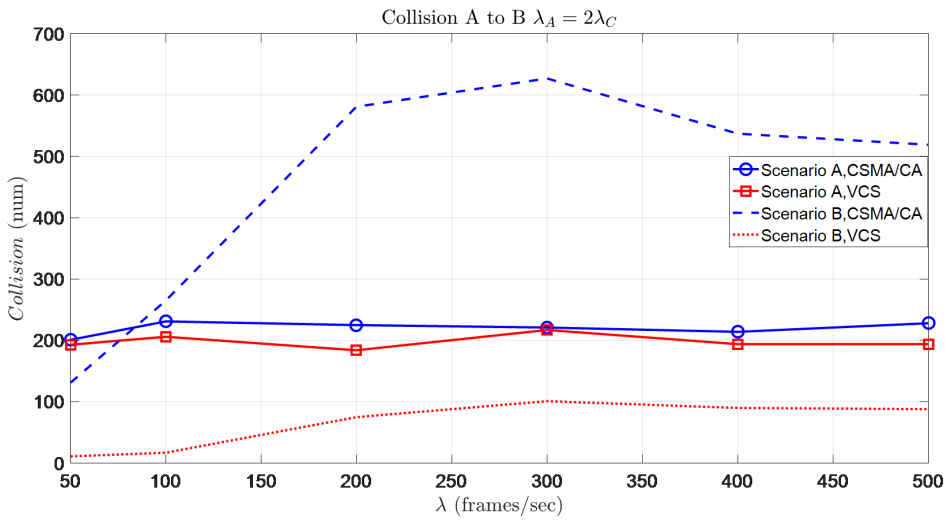
In scenario B, as we have mentioned above, collisions are very possible to happen if we don't utilize Virtual Carrier Sense in hidden terminal protocol. So, you can see there are almost 600 collisions. The number of collisions is much lesser when we utilize RTS/CTS which is designed to solve the problems caused by hidden terminal. Due to the reason that there is no collision would happen at Node D, so the number of collisions are all zero in second picture.

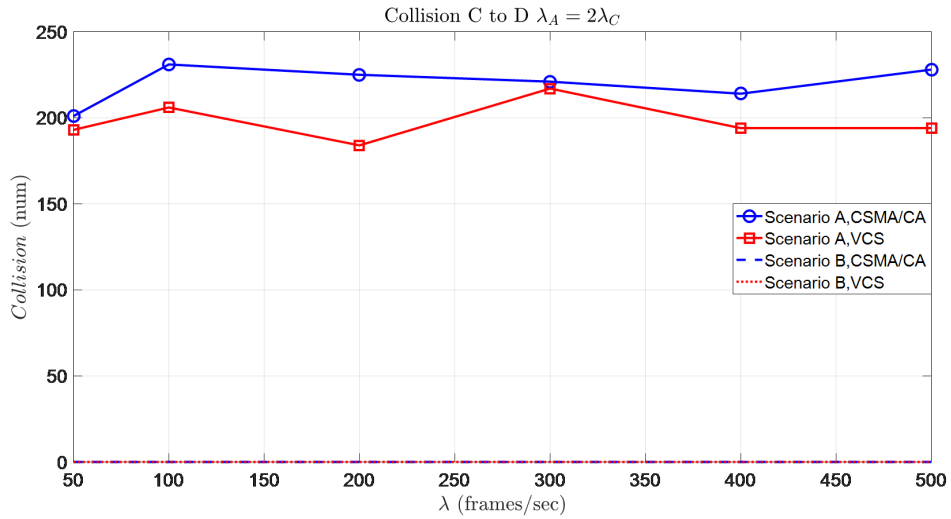
Second, scenario A and scenario B when $\lambda_A = 2\lambda_C$.

The performance of scenario A with RTS/CTS is really improved when we doubleλ_A.


Collision A to B $\lambda_A = \lambda_C$
Collision (num)
λ (frames/sec)
Scenario A,CSMA/CA
Scenario A,VCS
Scenario B,CSMA/CA
Scenario B,VCS

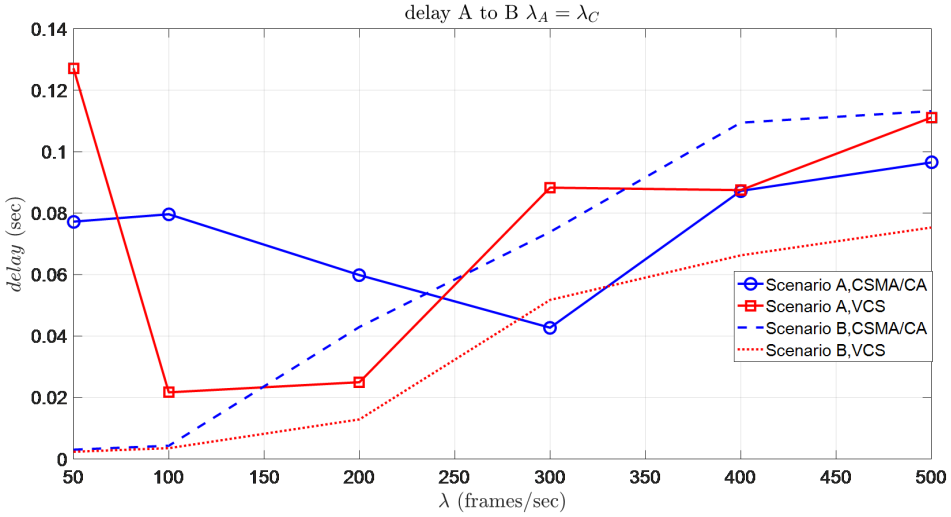

Collision C to D $\lambda_A = \lambda_C$
Collision (num)
λ (frames/sec)
Scenario A,CSMA/CA
Scenario A,VCS
Scenario B,CSMA/CA
Scenario B,VCS

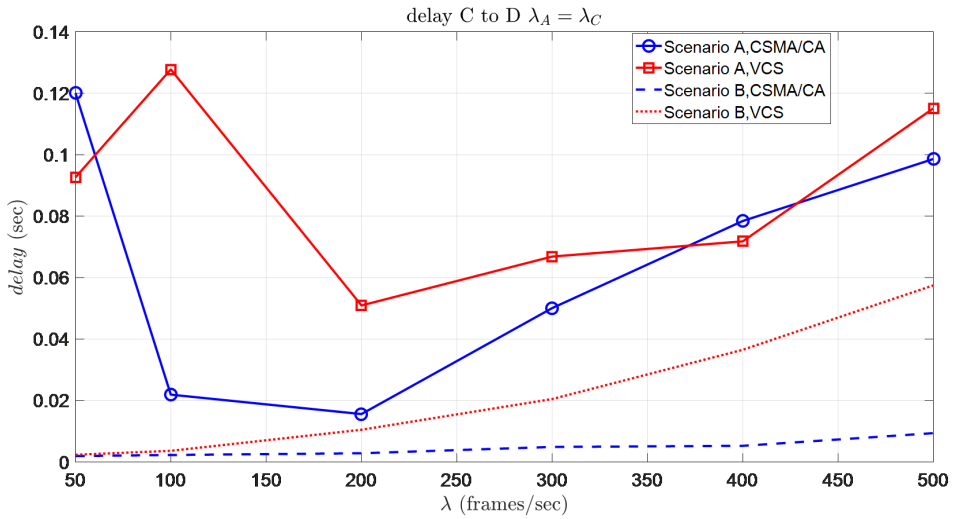

Collision A to B $\lambda_A = 2\lambda_C$
Collision (num)
λ (frames/sec)
Scenario A,CSMA/CA
Scenario A,VCS
Scenario B,CSMA/CA
Scenario B,VCS

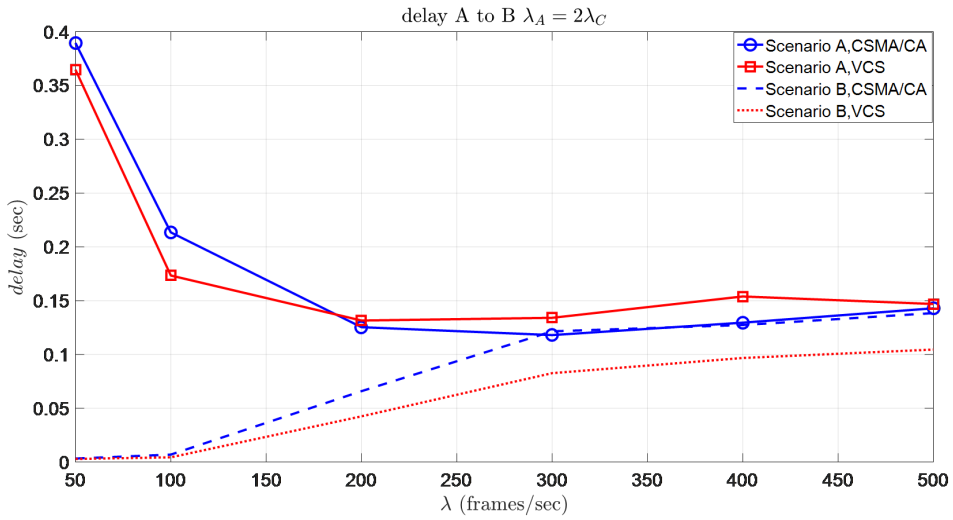


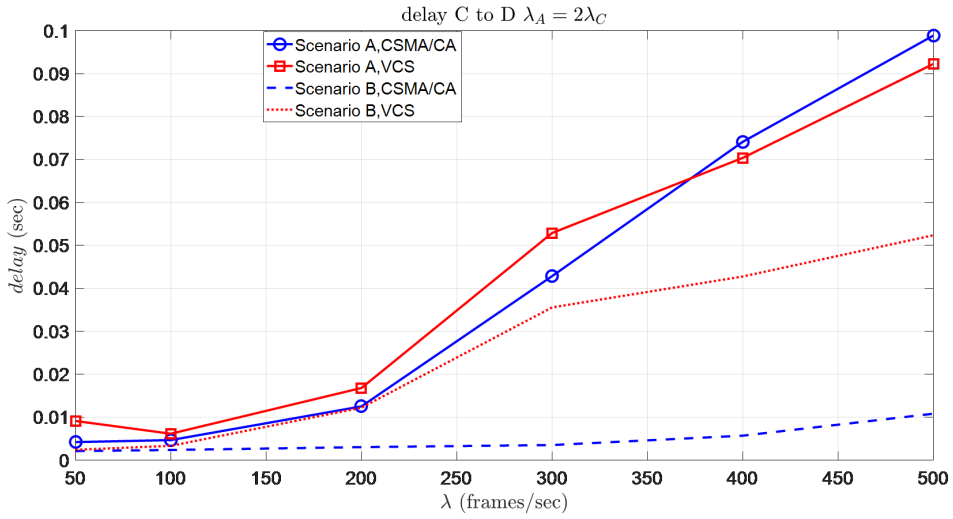
3.3 Delay

In this project, Delay is calculated as the time from the sending at the sender (Node A and Node C) to the time of receipt of that packet at the receiver. In the code, we calculate Delay via the difference between the number of slots cost to complete all transmissions and to receive frames transmitted successfully. So, there would be a delay time only if the receiver receives packets.







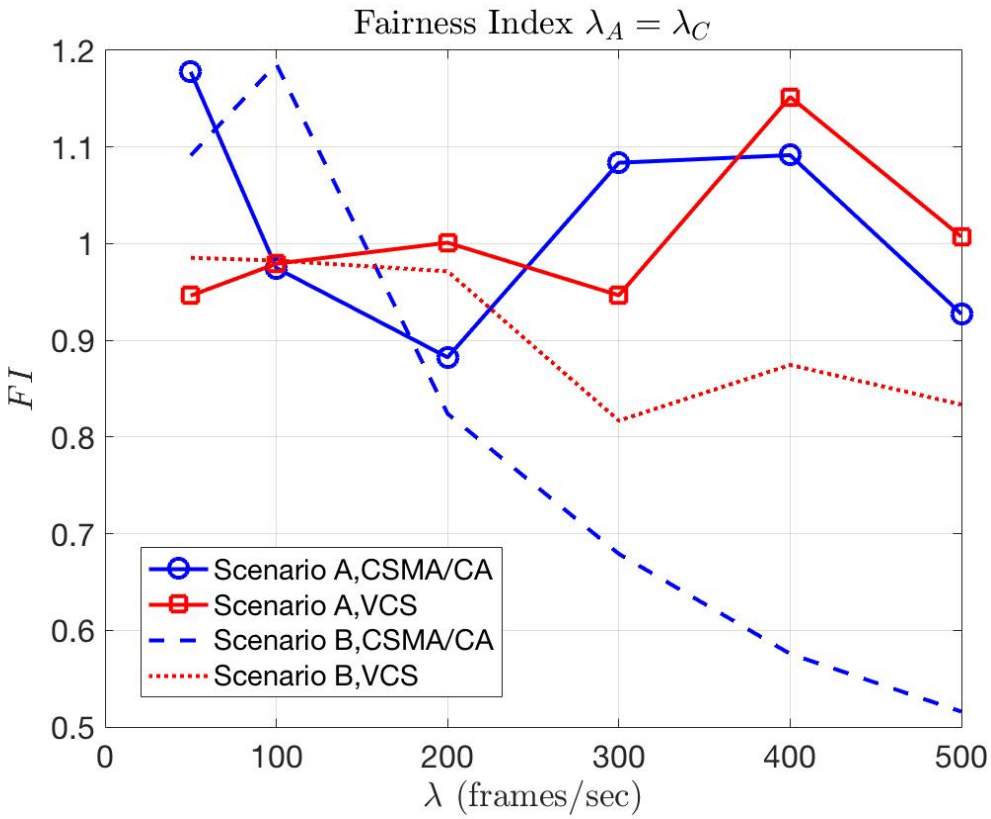


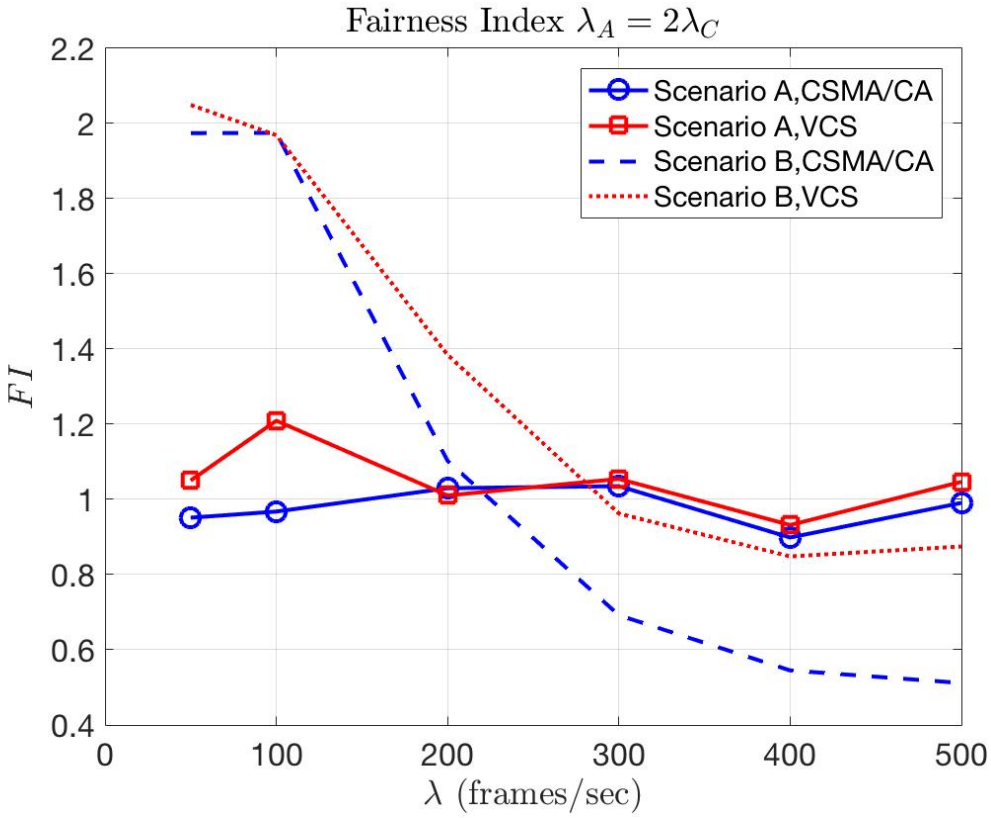
3.4 Fairness Index

The Fairness Index was used to estimate the fairness for both channels, the more the value is close to one, the more fair for both channels.

In scenario A, because of the symmetrical topology, node A and node C have the same chance to transmit frames, it's easy to understand the curve fluctuates along 1.

In scenario B, you can see an obvious down trend of both curves, and the curve of CSMA/CA decreases faster when λ increases. with the increment of λ, C has more chance to transmit frames, in the meantime, it will be easier to collide in node B, which causes FI get away from 1. However, after implement of VCS, it will improve such unfairness, and make the FI value become close to 1.




Fairness Index $\lambda_A = 2\lambda_C$
Scenario A,CSMA/CA
Scenario A,VCS
Scenario B,CSMA/CA
Scenario B,VCS
FI
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0
100
200
300
400
500
λ (frames/sec)