Wireless Network Project Report

Simulation of CSMA/CA performance in Python

Group 4

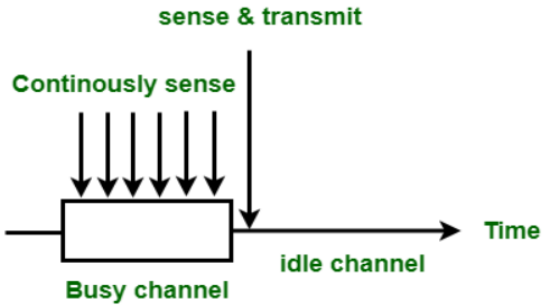
Sagar Suman (2019197), Arpit Kumar(2019153), Puneet Kumar(2019081)

1. Introduction

What is CSMA?

Carrier sense multiple access (CSMA) is a network protocol used to coordinate the transmission of data between multiple devices and to manage access to a shared communication channel.

The idea is that before a device sends data, it listens for other devices that might be transmitting data on the same channel and checks channel activity in its vicinity. If it detects any ongoing transmission, it waits for the transmission to complete before attempting to transmit its own data. This helps prevent collision for cases when multiple devices try to transmit data simultaneously, reducing interference and data loss and improving efficiency.



[Ref.: Link]

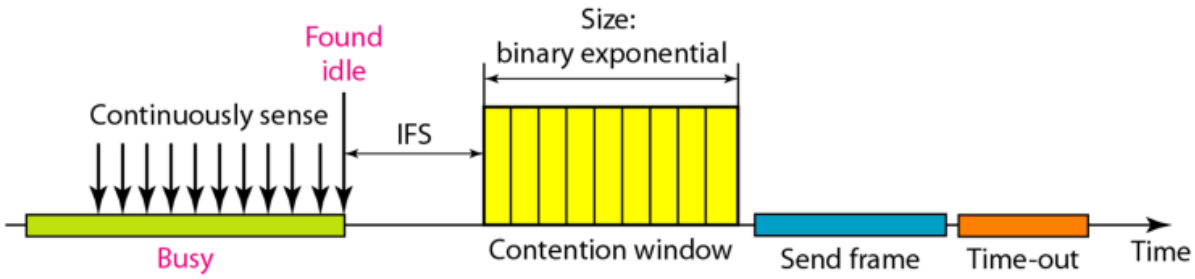
CSMA Variations

CSMA/CD - CSMA with collision detection. In CSMA, multiple devices can sense that the channel is idle in their vicinity and may possibly transmit data simultaneously causing a collision. When this collision occurs in a wired network (such as Ethernet), the colliding devices can detect the interference and stop transmission from their end in order to ensure that further collisions don't continue.

CSMA/CA - CSMA with collision avoidance in wireless networks. To reduce the likelihood of collisions and avoid collisions as much as possible, 3-4 major techniques are used -

- Interframe Spacing (IFS) - Immediate transmission of data is deferred even if the channel is found idle and the device has to wait a fixed amount of time equal to IFS assigned to it before proceeding to the next step.

- Contention Window - The device chooses a random no. of slots as it wait-time before it can transmit the data. The Contention window is the amount of time divided into slots. This contention window's size increases according to binary exponential backoff strategy(generally doubles every re-attempt). By increasing the window size after every failed-transmission-attempt (i.e. when channel is getting busy due to packet collisions etc.), further collisions are avoided by making the transmitting devices backoff from using the channel for a while.



[Ref.: Link]

- Acknowledgement - In order to ensure that the receiver has received the data, positive acknowledgement and timer timeout methods can be applied.

- RTS/CTS mechanism - Ready to send and clear to send mechanism can also be included to allow the devices to reserve the channel and avoid transmitting at the same time. When a device wants to transmit data, it first sends an RTS signal to the receiving device. If the receiving device is available, it sends a CTS signal back to the transmitting device. The CTS signal indicates that the receiving device is ready to receive data and reserves the channel for the transmitting device.

Why CSMA/CA?

- For Wireless cases, detection of collision is not possible and thus techniques like CSMA/CD etc. can not be used
- Reduced Collisions are ensured by using a variety of techniques like IFS, contention window, and RTS/CTS etc.
- Relatively simplistic protocol that require low-cost and low-power solutions
- CSMA/CA is designed to be fair to all devices on the network, regardless of their location or transmission power.

2. Problem Statement

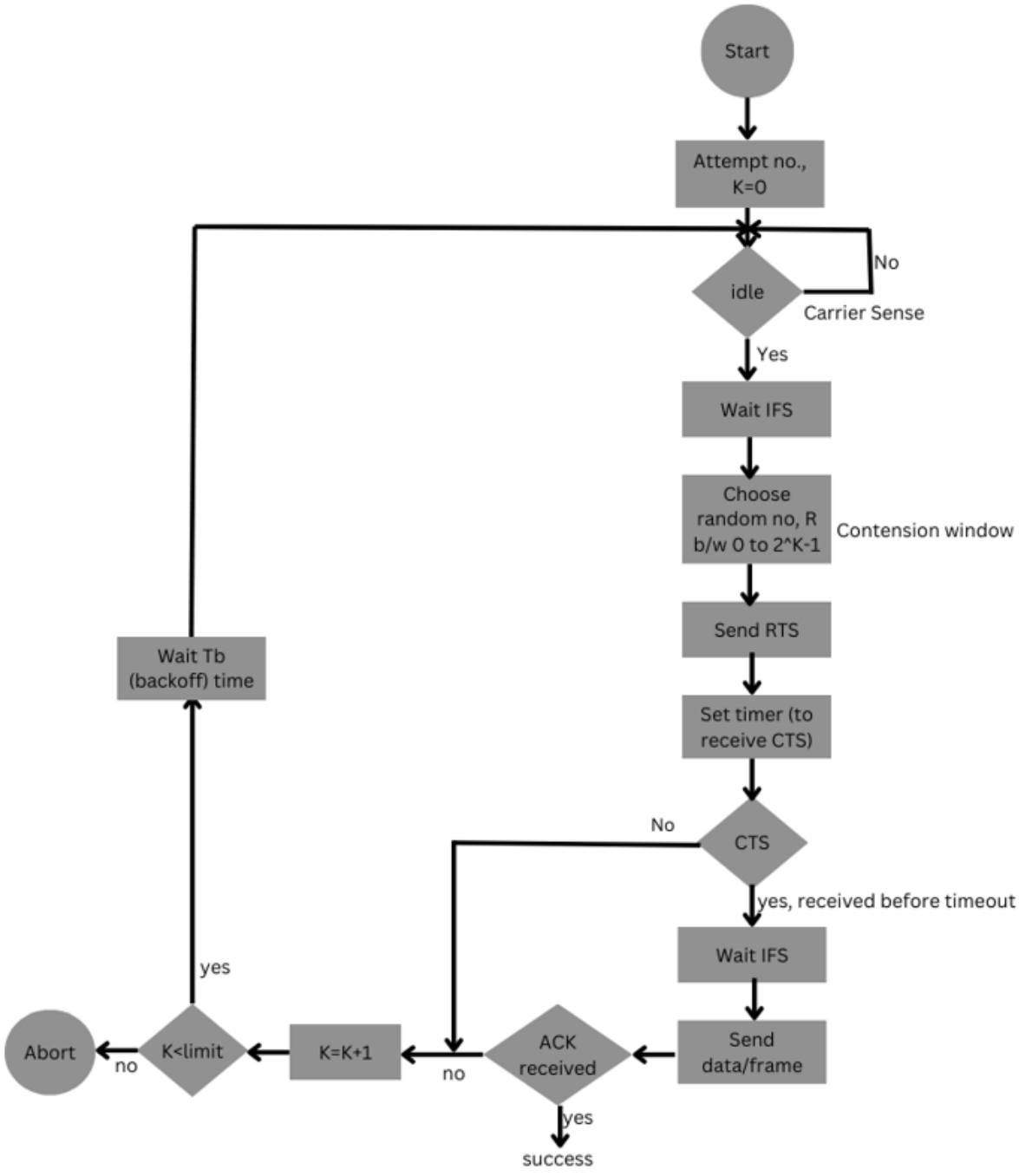
We have to simulate the performance of CSMA/CA of different networks in Python.

3. Solution

3.1. Approach

For the simulation and implementation component, we first tried to come up with an algorithm that accurately depicts CSMA/CA and constructed a flowchart for the same as shown below. Let's go over the algorithm step-by-step:

At start, attempt no. K=0. For each consecutive attempt of transmission, K increases by 1. CSMA is applied by sensing if the channel is idle in the device's vicinity. If it is, instead of directly attempting to transmit, we wait for IFS time, even after that if the channel is still idle we wait for amount of time-slots in contention windows (which consists of R slots of fixed time where R is some random number ranging from 0 to 2^K-1). After that, the device sends a ready-to-send signal to the receiver and if the clear-to-send signal is obtained before timeout the channel is reserved and the transmitter device finally waits for some time and sends the data. If the transmitter receives an acknowledgement back from the receiver, then it means data is successfully transmitted. If not, then variable K is increased by 1 (for next attempt). If the no. of attempts exceeds a specified limit, then this process is aborted. If not, then we try to repeat the process again from the start after a certain amount of random backoff time, as shown.



Implementation Algorithm/Flowchart

3.2. Implementation details

The implementation consists of these major steps in Python using **simpy**:

- Defining the network topology: Determining the total no. of nodes, and connections between nodes etc.
- Defining the simulation parameters: Determining duration of simulation, type of traffic etc.
- Implementation of CSMA/CA algorithm (using the discussed flowchart)
- Simulation of network traffic: traffic between nodes in the network is generated
- Collection of results: Collisions etc. are detected using the simulation and performance of the network based on metrics such as throughput, packet loss and delay etc is calculated. Performance of networks under different conditions is observed and analyzed.

3.3. Results

RESULT 1

Output of the CSMA/CA Simulation without using RTS/CTS mechanism is as follows:

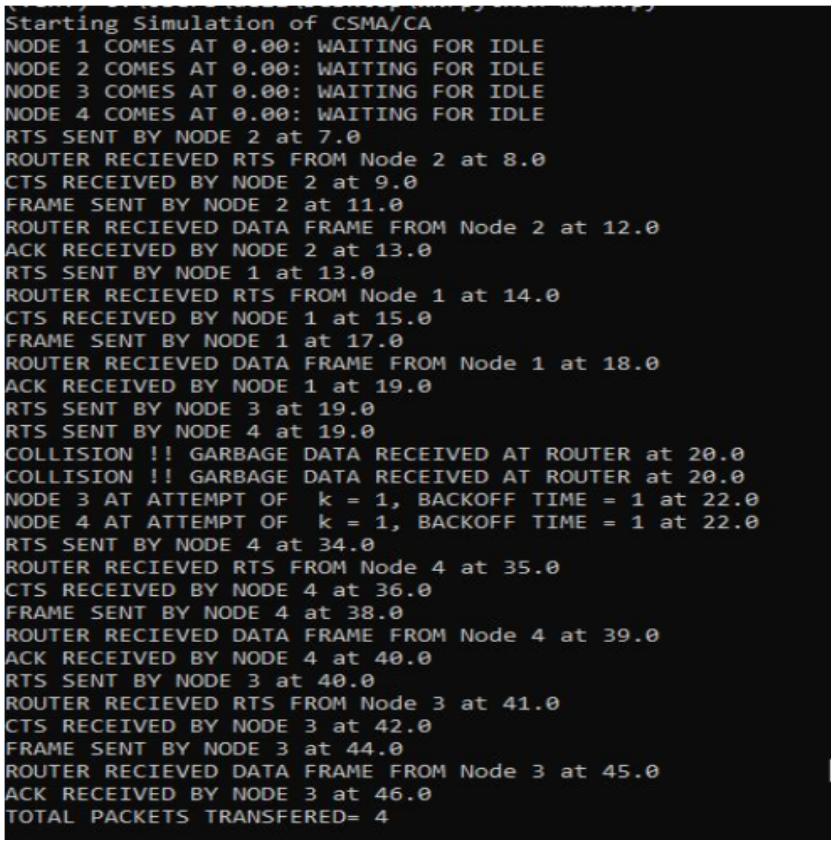
```
Starting Simulation of CSMA/CA
NODE 1 COMES AT 0.00: WAITING FOR IDLE
NODE 2 COMES AT 0.00: WAITING FOR IDLE
NODE 3 COMES AT 0.00: WAITING FOR IDLE
NODE 4 COMES AT 0.00: WAITING FOR IDLE
FRAME SENT BY NODE 1 at 5.0
FRAME SENT BY NODE 2 at 5.0
FRAME SENT BY NODE 4 at 6.0
FRAME SENT BY NODE 3 at 6.0
COLLISION !! GARBAGE DATA RECEIVED AT ROUTER at 6.0
COLLISION !! GARBAGE DATA RECEIVED AT ROUTER at 6.0
COLLISION !! GARBAGE DATA RECEIVED AT ROUTER at 7.0
COLLISION !! GARBAGE DATA RECEIVED AT ROUTER at 7.0
NODE 1 AT ATTEMPT OF  k = 1, BACKOFF TIME = 0 at 8.0
NODE 2 AT ATTEMPT OF  k = 1, BACKOFF TIME = 0 at 8.0
NODE 4 AT ATTEMPT OF  k = 1, BACKOFF TIME = 0 at 9.0
NODE 3 AT ATTEMPT OF  k = 1, BACKOFF TIME = 1 at 9.0
FRAME SENT BY NODE 2 at 19.0
FRAME SENT BY NODE 3 at 20.0
FRAME SENT BY NODE 4 at 20.0
ROUTER RECIEVED PACKET FROM Node 2 at 20.0
COLLISION !! GARBAGE DATA RECEIVED AT ROUTER at 21.0
COLLISION !! GARBAGE DATA RECEIVED AT ROUTER at 21.0
ACK RECEIVED BY NODE 2 at 21.0
FRAME SENT BY NODE 1 at 22.0
ROUTER RECIEVED PACKET FROM Node 1 at 23.0
NODE 3 AT ATTEMPT OF  k = 2, BACKOFF TIME = 2 at 23.0
NODE 4 AT ATTEMPT OF  k = 2, BACKOFF TIME = 1 at 23.0
ACK RECEIVED BY NODE 1 at 24.0
FRAME SENT BY NODE 3 at 34.0
ROUTER RECIEVED PACKET FROM Node 3 at 35.0
ACK RECEIVED BY NODE 3 at 36.0
FRAME SENT BY NODE 4 at 37.0
ROUTER RECIEVED PACKET FROM Node 4 at 38.0
ACK RECEIVED BY NODE 4 at 39.0
TOTAL PACKETS TRANSFERED= 4
```

Here, assumptions are that node takes 1 sec to send data to the router. The acknowledgment by the Router also reaches the nodes in 1 sec. Node waits for another 1 sec in case ACK isn't received before proceeding to the next phase (i.e. incrementing attempt no. K by 1 and going to the backoff time stage).

Explanation of Output:
Here, we have a total of 4 nodes and each node is sending 1 packet to the router. At time 0.0 second, the channel is idle and so the nodes do carrier sensing in their vicinity and all of the nodes come to the conclusion that channel is idle at t=0.0 sec. So, they go through the whole waiting process (in flowchart i.e. wait for IFS and contention window) before sending the data frame directly as no RTS/CTS mechanism is involved. Based on this, node 1 and 2 send their data frame at t=5 sec. And nodes 3 and 4 at t=6 sec. Now, for nodes 1 and 2, their packets should reach router at t=5+1= 6sec. However, since both node 1 and 2 packets reach at same time, there is a collision between them at t=6sec. Similarly for nodes 3 and 4 their packets collide at router at t=6+1=7 sec. The nodes 1 and 2 wait for acknowledgement by router but no acknowledgment comes in 1 sec due to collision. Thus, nodes 1 and 2 wait for another 1 second (t=7+1=8) before assuming that since no ACK came, there must have been a collision (Similar is the case for nodes 3 and 4 delayed by a sec). So, they try to send the data again and increment their attempt no. K by 1. The backoff time is decided based on random(0,2^K-1)= random(0,2^1-1)= random(0,1). For nodes 1, 2, 3 backoff time randomly decided came out to be 0 secs and for node 4 it came out to be 1 sec. So, nodes 1,2 wait for 1 sec after t=8 and then go through the same process of DIFS and contention window. The randomness by contention window is different for nodes this time and node 2 sends its data at t=19sec, nodes 3 and 4 send their data frame at t=20 sec. Now the data by node 2 is received successfully by the router and ack is received by node 2 at t=20. Also again there is a collision b/w nodes 3 and 4 **(Note:** for 1 collision, 2 print statements are printed not to be confused with 2 collisions). Meanwhile, node 1 also reattempts to send its data packet and sends the frame at t=22sec and receives the ACK by t=22+1+1 = 24 sec. Meanwhile node 3 and 4 tries their 2nd attempt in sending the data (k=2). Backoff-time is randomly chosen in range(0,2^2-1) = range(0,3). For node 3 it is 2 sec and for node 4 it is 1 sec. So, node 3 goes through the waiting process again and sends the data frame successfully at t=34 sec after finding the channel idle and receives the ACK for the same at t=34+1+1=36sec. Node 4 also reattempts and sends the data frame successfully at t=37 and receives a positive acknowledgement of the same at t=37+1+1=39 sec.

RESULT 2

Output of the CSMA/CA Simulation on using RTS/CTS mechanism is as follows:

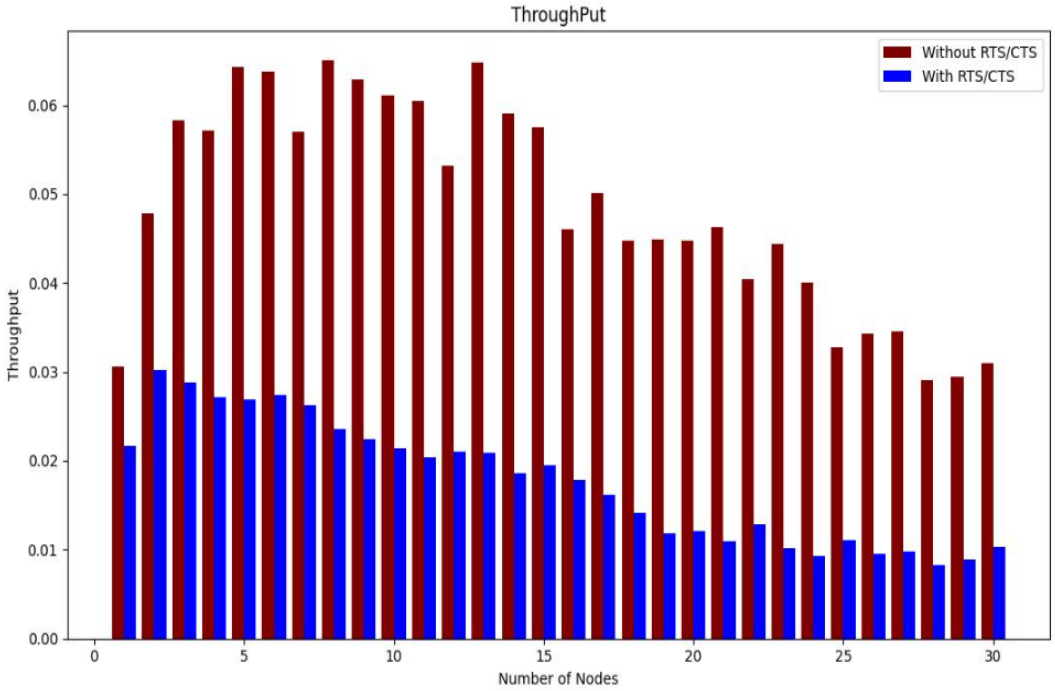


Explanation of output:

All nodes sense the channel as idle at t=0 sec. At t=7 sec, RTS is sent by node 2 (after completion of its waiting DIFS time, contention windows etc.) and received by router at t=7+1=8sec and CTS by router reaches node 2 at t=9sec. SO the channel gets received for node 2 at this time and node 2 sends data frame (t=11), and receives positive ack by router for the same (at t=11+1+1=12 sec). After this, the process goes in a similar fashion for node 1 (from t=t=13 to t=19). However, we see that for next time, node 3 and node 4 collision occurs in their RTS and thus they retry their attempt at sending packet and goes through their backoff time. Node 4 again sends RTS successfully at t=35 and this time it gets CTS back and successfully sends its data at t=38 sec and receives positive ACK at t=40. Finally the same process happens successfully for node 3.

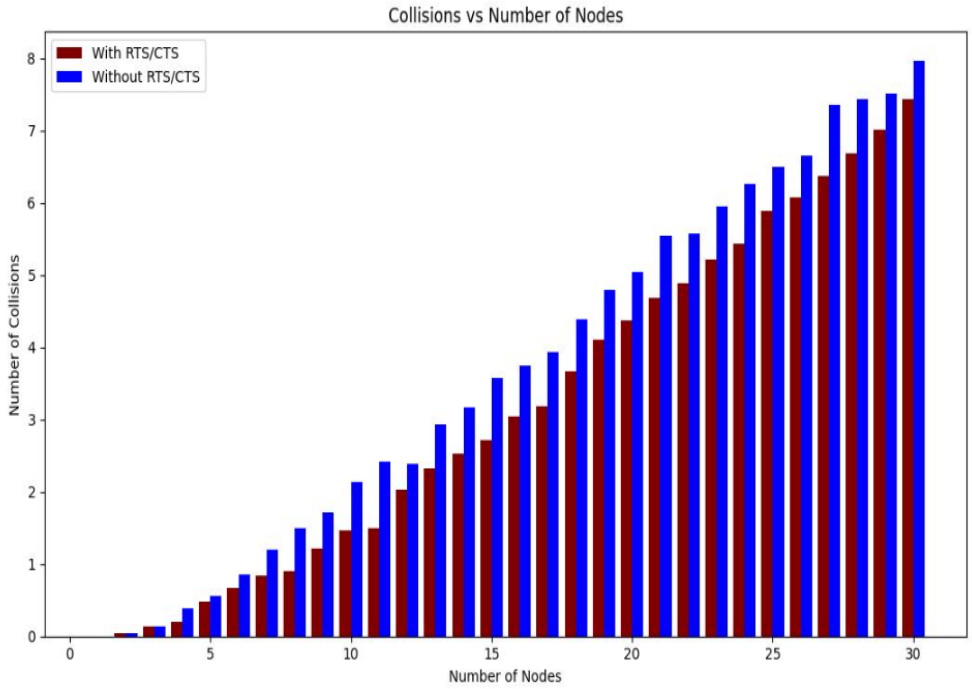
3.4. Analysis

After the above results, we ran our simulation for different conditions and scenarios for instance, by changing the total no. of nodes present in the network, by changing the coverage area of nodes in the network, by changing the presence of RTS/CTS mechanism in the algorithm, etc. and finally compared these different cases by plotting the graphs for the same.



Analysis 1→As no. of nodes increases in the network, throughput (total no. of packets transferred/total time taken) is decreasing. This is because the more the no. of nodes, more the communication and more the chances of collisions which leads to less efficient use of the channel and less throughput.

Analysis 2→ Analysis 2→In our case (i.e. sparse network), we see that the throughput is high for without RTS/CTS mechanism (than RTS/CTS) because of the additional overhead of wait-time for all the nodes when channel is reserved by RTS/CTS. However, if we plot for dense networks the RTS/CTS one should perform better as less collisions in RT/CTS leads to less completion time and better throughput.



Analysis 3→ We see that as no. of nodes increases, the no. of collisions in the network also increases due to more communication and more chances of collision.

Analysis 4→ Wee see that the no. of collisions in RTS/CTS case is lesser than no. of collisions in without RTS/CTS case, as RTS/CTS ensures the channel gets reserved and other nodes stop their transmission temporarily in order to reduce the chances of collisions.

4. Conclusion

We conclude that CSMA/CA is a powerful, relatively simplistic protocol that ensures collision avoidance in the wireless network. We also saw how RTS/CTS further improves its efficiency at the cost due to transmission of extra frames. Finally, we plotted some graphs and analyzed each of them carefully in order to back the statements above.

5. References

Research paper used for guidance:
https://www.researchgate.net/publication/278729275_Performance_Analysis_of_CSMA CA_in_Wireless_Local_Area_Network