Computer Networking - Lab 6

Srinjoy Som	220123074
Shubham Kumar Jha	220123081
Yash Singhal	220123072
Priya Sharma	220101081

PART A.

CSMA/CD (Carrier Sense Multiple Access with Collision Detection) is a network protocol used primarily in Ethernet (wired) networks to manage access to a shared communication medium and prevent data collisions.

Working of CSMA/CD:

Step 1: Check if the sender is ready to transmit data packets.

Step 2: Check if the transmission link is idle.
The sender must keep checking if the transmission link/medium is idle. For this, it continuously senses transmissions from other nodes. The sender sends dummy data on the link. If it does not receive any collision signal, this means the link is idle at the moment. If it senses that the carrier is free and there are no collisions, it sends the data. Otherwise, it refrains from sending data.

Step 3: Transmit the data & check for collisions.
The sender transmits its data on the link. CSMA/CD does not use an 'acknowledgement' system. It checks for successful and unsuccessful transmissions through collision signals. During transmission, if the node receives a collision signal, transmission is stopped. The station then transmits a jam signal onto the link and waits for random time intervals before it resends the frame. After some random time, it again attempts to transfer the data and repeats the above process.

Step 4: If no collision was detected in propagation, the sender completes its frame transmission and resets the counters.

a.

(i) CSMA/CD is not effective in wireless scenarios because:

1. Collision Detection is Difficult in Wireless Networks

In a wired Ethernet network, a device can easily detect a collision by comparing the data it is sending with what it receives on the wire. If there's a mismatch, it knows a collision has occurred.

However, in wireless networks, a device cannot listen to the channel and transmit at the same time effectively. This is because full-duplex communication (simultaneous transmission and reception) is not practical on a single wireless channel. A device's own transmission is usually much stronger than incoming signals, effectively drowning out any potential collision signals.

As a result, collision detection (CD) is not feasible in wireless networks.

2. The Hidden Node Problem

In a wireless network, two devices (let's call them A and C) might be out of range of each other but both within range of a common access point (B). If A and C transmit simultaneously, a collision will occur at B, even though A and C cannot detect each other's transmissions due to being out of range.

CSMA/CD cannot solve this issue because devices A and C do not know that they are causing interference at B.

3. Signal Interference and Fading

Wireless signals are susceptible to interference, attenuation, and fading due to physical obstacles, distance, and environmental factors. These issues make it difficult for a device to reliably sense whether the channel is clear, which is essential for carrier sensing in CSMA/CD. Even if a device senses that the channel is clear and starts transmitting, signal quality can degrade rapidly, causing undetectable collisions.

(ii) CSMA/CD collision detection in a shared medium.

1. When a device starts transmitting data, it continuously listens to the network to detect if another device has started transmitting simultaneously.
2. If the signal that the device sends differs from what it receives on the network, it means a collision has occurred (i.e., multiple devices sent data simultaneously, causing signal interference).
3. When a collision is detected, the devices involved stop transmitting immediately and send a **jam signal** to inform other devices that a collision has occurred.

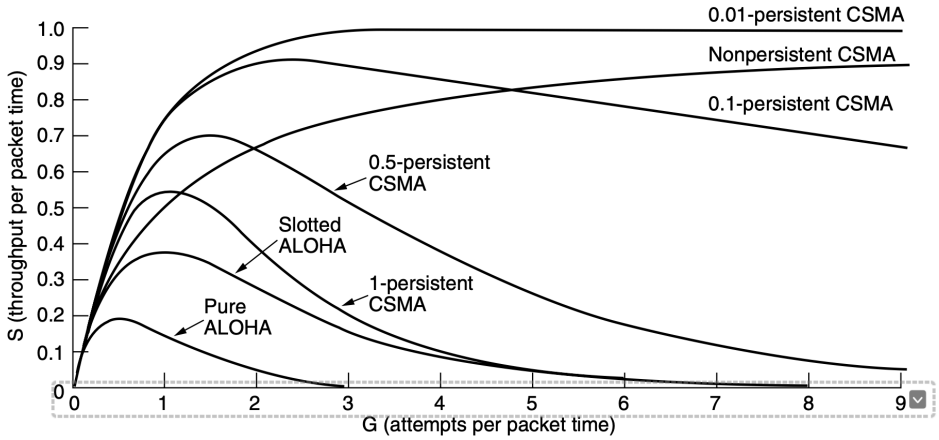
4. After sending the **jam signal**, all devices involved in the collision wait for a random time interval before attempting to retransmit. This is where the **exponential backoff algorithm** comes into play.

Exponential backoff algorithm for preventing further collisions.

The exponential backoff algorithm is used to control the retransmission of packets after a collision is detected. The idea is to reduce the chances of repeated collisions by making each device involved wait for a random amount of time before retrying.

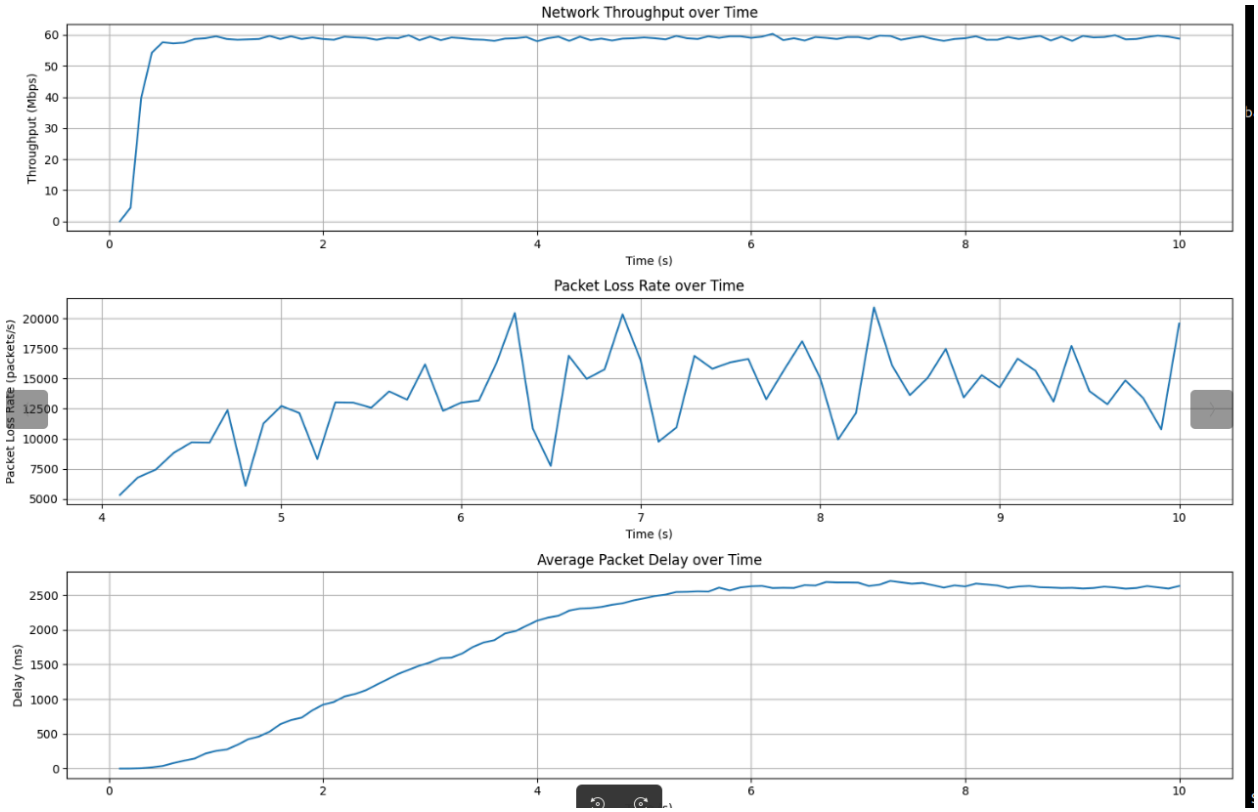
1. After detecting a collision, each device involved waits for a random amount of time before attempting to retransmit. The wait time is measured in units called **slot times**.
2. The **slot time** is the time it takes for a signal to travel from one end of the network to the other and back (i.e., round-trip time).
3. Each device picks a random number between 0 and 1 for the first retransmission attempt and waits for that many slot times.
4. If a collision occurs again on the next transmission attempt, the range doubles. For the second attempt, the random wait time is chosen between 0 and 3 slot times.
5. This process continues, with the range doubling each time a collision is detected (i.e., 0 to 7 slot times for the third attempt, 0 to 15 slot times for the fourth attempt, etc.).
6. The maximum backoff limit is typically reached after **10 retransmission attempts** (with a maximum range of 0 to 1023 slot times). If it still cannot transmit successfully after **16 attempts**, the transmission is aborted.

Theoretical Throughput vs Time graph:



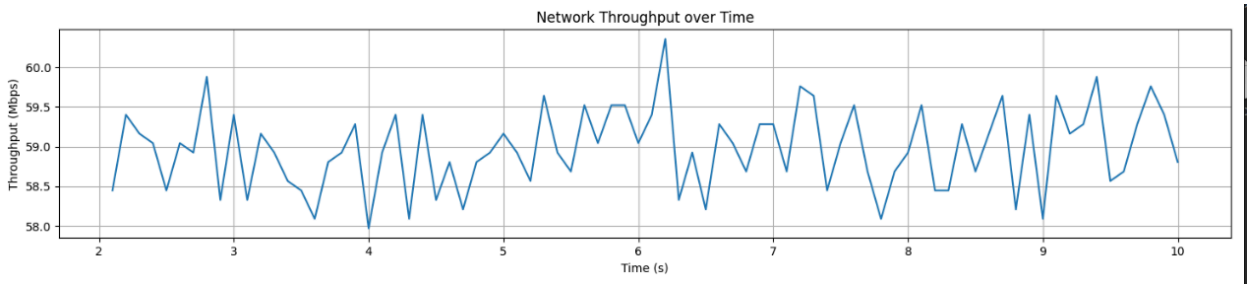
In ns3 to simulate the environment we have writtten the simulator code and added it in the scratch folder of the ns-3.36 folder which recedes on the source folder after we download ns-3using bake(there are other methods like using tarball and manually from github) . So in scratch file we simulated a network with 5 nodes and we used the csma helper class which is used to create channel with csma channel .then we set the data rate and delay as the question specified .We also make a new class that calculates metrics like packets dropped and throughput and delay at each interval of 0.1 seconds .

And later we plot the graph of these three data .



We can see that the throughput increases from 0 initially and then oscillates around a value according to the congestion in the medium synced with the packet loss .

We can take a closer look at the oscillation then it shows



.

PART B:

CSMA/CA:

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is a network protocol for carrier transmission that operates in the Medium Access Control (MAC) layer. In contrast to CSMA/CD (Carrier Sense Multiple Access/Collision Detection) that deals with collisions after their occurrence, CSMA/CA prevents collisions prior to their occurrence.

The algorithm of CSMA/CA is:

1. When a frame is ready, the transmitting station checks whether the channel is idle or busy.
2. If the channel is busy, the station waits until the channel becomes idle.
3. If the channel is idle, the station waits for an Inter-frame gap (IFG) amount of time and then sends the frame.
4. After sending the frame, it sets a timer.
5. The station then waits for acknowledgement from the receiver. If it receives the acknowledgement before expiry of timer, it marks a successful transmission.
6. Otherwise, it waits for a back-off time period and restarts the algorithm.

RTS/CTS:

In wireless networks, devices (nodes) do not always have visibility of each other due to obstacles or distance. This can lead to the hidden node problem. When two devices (e.g., Node A and Node C) are both in range of a central access point (Node B) but not in

range of each other. If Node A and Node C transmit simultaneously, a collision occurs at Node B, even though Nodes A and C are unaware of each other's transmissions.

To address this issue, the RTS/CTS protocol is used as a handshake mechanism to reserve the channel before transmitting data, reducing the chances of collisions.

Working:

Request to Send (RTS):
The sender (e.g., Node A) sends a short RTS frame to the receiver (e.g., Access Point B). This frame includes information such as the duration of the intended transmission.

Clear to Send (CTS):
If the receiver (Node B) is ready to receive and the channel is free, it responds with a CTS frame.
The CTS frame informs all other nearby devices that the channel is reserved for Node A for a specific time period.

Data Transmission:
Upon receiving the CTS, Node A starts transmitting its data.
Other devices that hear the CTS frame will refrain from transmitting until the data transfer is complete.

Acknowledgement (ACK):
After the data transmission is completed, the receiver (Node B) sends an ACK frame to confirm successful reception.
If the sender (Node A) does not receive an ACK, it assumes the transmission failed and retries using CSMA/CA.

In our code we have written a code to make a grid of 5 * 5 wireless network using wifi version 802.11g (we specified that in code) and then ran simulation once with rts cts on and in other case off then we printed the metrics in in the cases and compared

them .

```
srinjoy2004@srinjoy2004-IdeaPad-Gaming-3-15IMH05:~/source/ns-3.36$ ./ns3 run "scratch/quest2 --enableRtsCts = 1"
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/scratch/ns3.36-quest2-default
OLSR working
Flow 1
Number of packets transferred: 874
Number of packets received: 419
Throughput: 0.170003 Mbps
Average delay: 0.00845558 seconds
Packet loss rate: 52.0595%
Flow 2
Number of packets transferred: 843
Number of packets received: 428
Throughput: 0.179109 Mbps
Average delay: 0.0110119 seconds
Packet loss rate: 49.2289%
Flow 3
Number of packets transferred: 812
Number of packets received: 681
Throughput: 0.270283 Mbps
Average delay: 0.00244799 seconds
Packet loss rate: 16.133%
srinjoy2004@srinjoy2004-IdeaPad-Gaming-3-15IMH05:~/source/ns-3.36$ ./ns3 run "scratch/quest2"
[0/2] Re-checking globbed directories...
ninja: no work to do.
OLSR working
Flow 1
Number of packets transferred: 874
Number of packets received: 394
Throughput: 0.155948 Mbps
Average delay: 0.173225 seconds
Packet loss rate: 54.9199%
Flow 2
Number of packets transferred: 843
Number of packets received: 353
Throughput: 0.149648 Mbps
Average delay: 0.0761794 seconds
Packet loss rate: 58.1257%
Flow 3
Number of packets transferred: 812
Number of packets received: 582
Throughput: 0.270269 Mbps
Average delay: 0.00461921 seconds
Packet loss rate: 28.3251%
srinjoy2004@srinjoy2004-IdeaPad-Gaming-3-15IMH05:~/source/ns-3.36$
```

And we print them side by side to compare them .

Difference in throughput and packet drop rates with and without RTS/CTS:

1.

In networks with hidden nodes or high traffic, the absence of collision avoidance mechanisms can lead to more frequent collisions, reducing effective throughput. Collisions force devices to retransmit packets, wasting bandwidth and reducing overall throughput.

By reserving the channel before data transmission, RTS/CTS minimizes collisions, especially in networks with hidden nodes. This results in more successful data transmissions, which can improve effective throughput. In crowded networks where multiple devices compete for access, RTS/CTS helps manage the contention, reducing retransmissions and saving bandwidth.

2.

Sending RTS and CTS frames before each data packet adds extra control traffic to the network. This overhead can reduce overall throughput, especially for small packets or in lightly loaded networks. The additional exchange of RTS and CTS frames increases the time before data transmission, which can slow down the network and reduce throughput for real-time applications.

Without the extra RTS and CTS frames, the network can achieve higher throughput in environments with low traffic and minimal collision risk. For small data transmissions, omitting RTS/CTS reduces unnecessary delays, improving efficiency.

3.

Without collision avoidance, there is a higher chance of collisions, leading to more dropped packets. Packet drops are more frequent in crowded networks, leading to poorer performance, especially for applications that require reliable data delivery.

By reducing collisions, RTS/CTS significantly lowers the likelihood of packet drops, especially in scenarios with hidden nodes. With fewer collisions, data packets are more likely to reach their destination successfully, leading to a more stable network performance.

Link to all the files and the modified ns3 folder

https://drive.google.com/drive/folders/1YgzOlqnLJvSml1d-QrUdGUUjptoiQMhn?usp=sharing