CISC7002 Computer Communications and Networks Final Report

Sun Tianhao MC251242

Abstract

This report simulates carrier sense multiple access with collision detection (CSMA/CD). We first introduce the carrier sense multiple access (CSMA) which is the basis of the CSMA/CD. Then we introduce CSMA/CD. We describe in detail the algorithm design of CSMA/CD and perform simulation experiments. Finally, we present some methods to improve CSMA/CD. Code can be found on: https://github.com/stickice/CSMA-CD-Simulation.

1 Introduction

1.1 CSMA

Carrier sense multiple access (CSMA) [1] is derived from the ALOHA protocol [2]. The main idea of CSMA is to constantly sense the channel before sending data, i.e. carrier sense. if the channel is busy and other data is being transmitted, the data transmission is delayed. If the channel is idle, the data is transmitted. CSMA has three types:

1. 1-persistent sensing

   Always sensing until idle.

   Sending data immediately after the channel is sensed to be idle.

2. non-persistent sensing

   There is a period between two senses.

   Sending data immediately after the channel is sensed to be idle.

3. p-persistent sensing

   Always sensing until idle.

Sending data with a certain probability. The probability p will influence the performance. When $p = 1$, p-persistent sensing degenerates to 1-persistent sensing.

However, CSMA can not completely avoid collisions.

1.2 CSMA/CD

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) [3] is an algorithm proposed in the IEEE 802.3 standard [4] in 1985. CSMA/CD inherits the CSMA design and adds a collision detection function. Algorithm 1 shows the process of CSMA/CD. Table 1 shows some parameters defined by the IEEE 802.3 standard.

Algorithm 1: CSMA/CD Algorithm

```
Data: Initialize N nodes;
Initialize distances[N]=[distances between nodes];
Initialize transmission speed, propagation speed, packets size;
while True and always sense channel do
    if channel is idle then
        send data and sense channel;
        if detect collision then
            abort and send jam signal;
            update collision;
            delay transmission by exponential backoff algorithm;
            continue;
        else
            set node.collision = 0;
            transmission done;
            wait IFG time;
        end
    else
        wait until transmission done;
        continue;
    end
end
```

Maximum Number of Nodes	30
Transmission Speed (bit/s, B/s)	10Mbit/s = 1.25MB/s
10BASE2 Coax Cables Maximum Length (m)	185m
Interframe Gap(IFG) (B)	20B(including the 8B packet preamble)

Table 1. Some parameters defined by IEEE 802.3

The binary exponential backoff algorithm can be formulated as follows:

$$backoff\_time = t \times r \tag{1}$$

where $t = 2\tau$, t is the basic backoff time and $r \in R$ is a random number, calculated by:

$$k = retransmission\_times \ if \ retransmission\_times < 10 \ else \ 10 \tag{2}$$

$$r = random\_sample(\{0, 1, \ldots, 2^{k-1}\}) \tag{3}$$

2 Experiments

2.1 Experimental Environment

The experimental environment is shown in Table 2.

Name	version
Linux	Ubuntu 20.04 LTS
Python	3.9
CPU	Intel Core i7

Table 2. Experiment environment.

2.2 Experiment Parameters

According to the IEEE 802.3 standard, Table 3 shows the experimental parameters. The node number takes different values according to the experiments. To simulate the real network situation, we set that each node sends a random number of packets. We set the transmission speed and propagation speed according to IEEE 802.3 standards. For all experiments, we simulate the network for $10s$.

Parameters	Value	Comment
Number of Nodes	{10, 20, 30, random number}	Different numbers are used in different experiments.
Distances Between Nodes (m)	random number ($< 185m$)	Distances between adjacent nodes.
Number of Packets	random number	Packets that every node need to transmit.
Packet Size (B)	1500B	Packet size.
Transmission Speed (B/s)	1.25MB/s	Transmission speed.
Propagation Speed (m/s)	1.95×10^8m/s	Propagation speed.
Running Time (s)	10s	Simulation time.

Table 3. Parameter settings according IEEE 802.3.

2.3 Experiments and Results

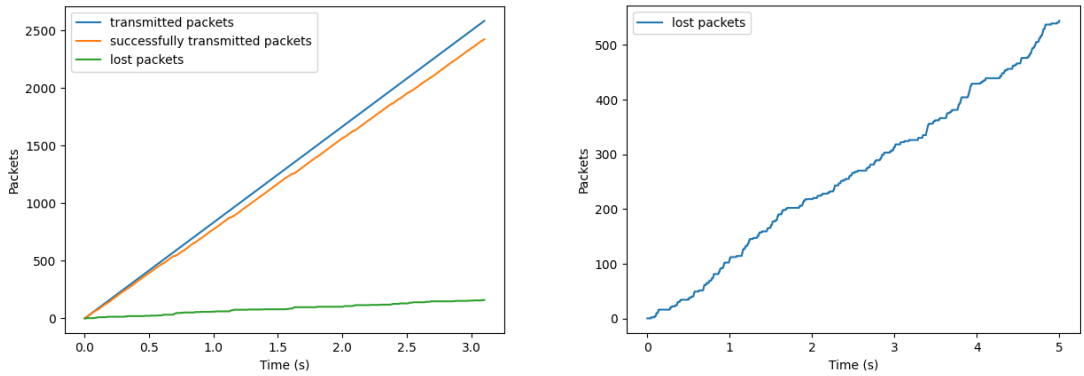
2.3.1 Simulation of General Condition

Table 4 shows the experimental parameters. To simulate the general situation, we set the random number of nodes (number of nodes$\leq$ 30).

Parameters	Value
Number of Nodes	random number
Distances Between Nodes (m)	random number ($< 185m$)
Number of Packets	random number
Packet Size (B)	1500B
Transmission Speed (B/s)	1.25MB/s
Propagation Speed (m/s)	1.95×10^8m/s
Running Time (s)	10s

Table 4. Parameter settings of general condition experiment.

Figure 1(a) shows all the packets sent by the node, successfully sent packets, and the lost packets due to collisions over time, we only show the 3s results. Figure 1(b) further shows the lost packets due to collisions over time. The results show that although CSMA/CD can guarantee the majority of data transmission, it cannot completely avoid collisions.



(a) (b)

Fig. 1. The running results of the general situation.

2.3.2 Simulation of Different Node Conditions

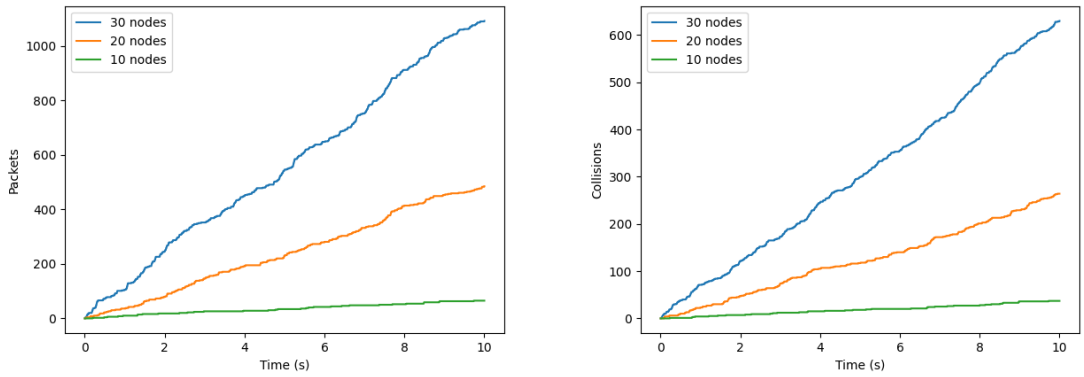
Table 5 shows the experimental parameters. This experiment was run with the number of nodes 10, 20, 30, respectively, to compare the effect of different numbers of nodes on CSMA/CD.

Parameters	Value
Number of Nodes	10, 20, 30
Distances Between Nodes (m)	random number ($< 185m$)
Number of Packets	random number
Packet Size (B)	1500B
Transmission Speed (B/s)	1.25MB/s
Propagation Speed (m/s)	1.95×10^8m/s
Running Time (s)	10s

Table 5. Parameter settings of different node conditions.

Figure 2(a) shows lost packets over time for different numbers of nodes. Figure 2(b) demonstrates the number of collisions over time for different numbers of nodes. The two figures have the same trend, proving that the simulation is running correctly. We can see that as the number of nodes increases, the number of collisions and lost packets also gradually increases, which is an intuitive idea. Although the number of nodes increases only three times from 10 to 30, the number of collisions and lost packets increases tens of times. It indicates that the

impact of network topology changes on network performance grows exponentially. Although more nodes can achieve greater throughput, they also lead to more data loss, which needs to be considered in practical applications.



(a) (b)

Fig. 2. The running results of different node conditions.

3 Some Possible Improvements

3.1 Using Advanced Network Devices

Using better network devices is a direct improvement. In IEEE 802.3, the network bandwidth is defined as 1.25 MB/s. Increasing the network bandwidth can improve the performance of CSMA/CD effectively. Figure 3 shows the changes in the number of lost packets after increasing the bandwidth. Other than changing the bandwidth, the other parameters of the experiment do not change.

The two curves for the larger bandwidths (6.25MB/s, orange one and 12.5MB/s, green one) are shorter compared to the curve for the original bandwidth (blue one) because the larger bandwidth transmits all the data in a shorter time. The number of lost packets is greatly reduced by increasing the bandwidth. The figure shows that the simplest way to improve performance is to use better network devices to get larger bandwidths (or other larger

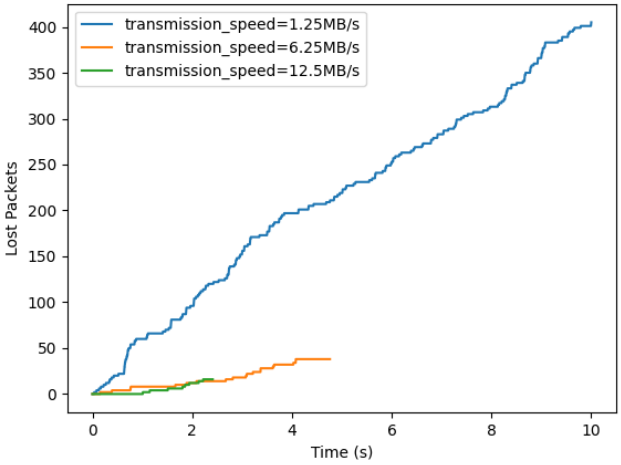


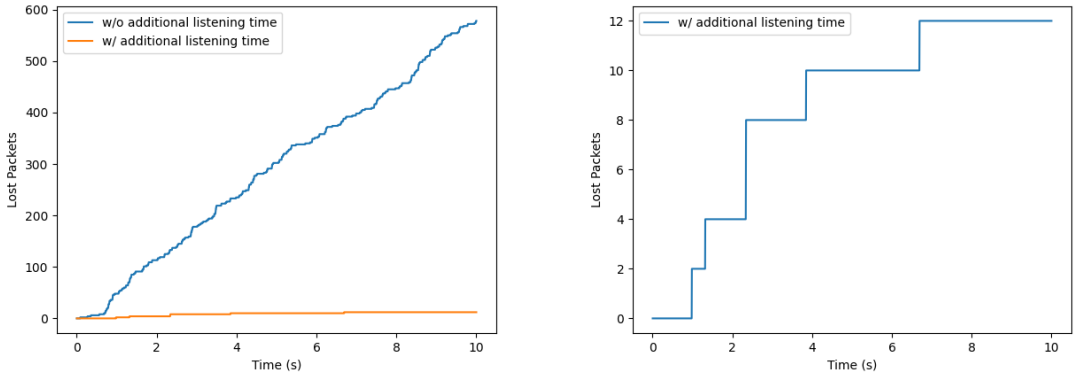
Fig. 3. The running results of improving the bandwidth.

properties). However, in most cases, this is a cost-effective method because using advanced network devices to replace old ones is expensive.

3.2 Adding Additional Waiting Time

CSMA/CD is based on 1-persistent CSMA, which means that CSMA/CD continuously senses the channel and sends frames as soon as the channel is idle. We reference carrier sense multiple access with collision avoidance (CSMA/CA) [5] to improve this part. CSMA/CA waits for IFS (Inter Frame Space) time when the channel is idle. Then CSMA/CA generates new waiting time $R \times slots$ using the binary exponential backoff algorithm. The data is transmitted if no collision occurs during the waiting time $R \times slots$. The CSMA/CD algorithm is improved by adding additional waiting time R generated by the binary exponential avoidance algorithm after sensing the channel idle. If no collision occurs during the wait time R, the frame is sent.

Figure 4(a) shows the curves of lost packets over time before and after increasing the waiting time R. Figure 4(b) shows the curve of lost packets over time separately. We can see that there is a huge reduction in lost packets after adding the additional wait time R, proving that the approach is effective.



(a) (b)

Fig. 4. The running results of adding additional waiting time.

3.3 Adding the ACK Technique

We can also reduce collisions by adding the **ACK** technique. CSMA/CD waits for an inter-frame gap (IFG) time after finishing sending data and then continues to sense the channel for the next transmission. Based on this, we add the **ACK** technique. After the source node sends data to the destination node, if the destination node receives data successfully, it will send an **ACK** message to the source node. If the source node does not receive the **ACK**, it will retransmit the data.Algorithm 2 shows the process of CSMA/CD with the **ACK** technique.

Algorithm 2: Improved CSMA/CD Algorithm

```
Data: Initialize N nodes;
Initialize distances[N]=[distances between nodes];
Initialize transmission speed, propagation speed, packets size;
while True and always sense channel do
    if channel is idle then
        R time: generated by binary exponential backoff algorithm;
        wait R time;
        send data and sense channel;
        if detect collision then
            abort and send jam signal;
            update collision;
            delay transmission by exponential backoff algorithm;
            continue;
        else
            set node.collision = 0;
            transmission done;
            while True do
                if receive ACK and no timeout then
                    break;
                else
                    if no time out then
                        continue;
                    else
                        goto send data and sense channel;
                    end
                end
            end
            wait IFG time;
        end
    else
        wait until transmission done;
        continue;
    end
end
```

4 Conclusion

In this report, we simply simulate CSMA/CD. We show the performance of CSMA/CD in different network topologies and analyze the experimental results. According to the drawbacks of CSMA/CD, we propose several improvements. We demonstrate that our proposed methods can improve the performance of CSMA/CD in certain environments. The experimental code can be found on: https://github.com/stickice/CSMA-CD-Simulation.

References

1. S. S. Lam, "A carrier sense multiple access protocol for local networks," *Computer Networks (1976)*, vol. 4, no. 1, pp. 21–32, 1980.
2. N. Abramson, "The aloha system: Another alternative for computer communications," in *Proceedings of the November 17-19, 1970, fall joint computer conference*, pp. 281–285, 1970.
3. F. A. Tobagi and V. B. Hunt, "Performance analysis of carrier sense multiple access with collision detection," *Computer Networks (1976)*, vol. 4, no. 5, pp. 245–259, 1980.
4. I. . Committee *et al.*, "Ieee standards for local area networks: Carrier sense multiple access with collision detection (csma/cd) access method and physical laver specifications," 1985.
5. A. Colvin, "Csma with collision avoidance," *Computer communications*, vol. 6, no. 5, pp. 227–235, 1983.