CSMA/CD

1. **Introduction**

Although comparing to ALOHA and Slotted ALOHA, CSMA has greatly improved the efficiency of transmission. However, a user who is sending a frame needs time, which is twice of the propagation time, to capture the medium. During this small time interval, other users who happen to have packets to send may regard the medium as idle and begins to transit and thus destroys both packets. And with the increasing of propagation time caused by longer distance or lower transmission rate, this kind of fault will have larger effect and reduces the efficiency rapidly.

In order to minimize the influence of problem mentioned above, CSMA/CD (collision detection) is invented. It tells users whether their packets are having collisions with other packets or not. If true, both the users stop transmission to save time and wait for another chance.

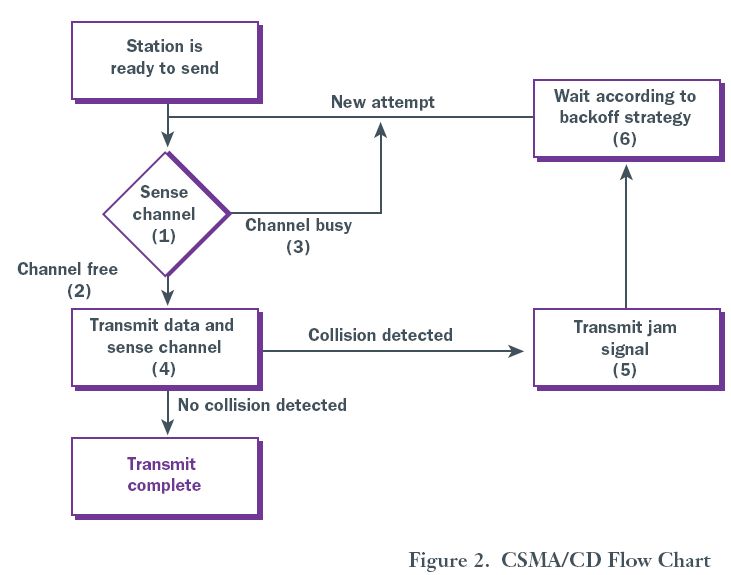
1. **The Principle of CSMA/CD**

At a given moment, user A has a packet to send. Before starting transmission, it checks whether the medium is idle or not. If the medium is busy, the user waits a random time and then tries again. If the medium is idle, the user starts to transmit, and keeps sensing the medium. Once a collision signal is detected, rather than finishing sending the packet, the station stops sending immediately and waits for the next attempt.

For choosing the random time, the protocol uses the Binary Exponential Backoff Algorithm. According to the algorithm, time is divided into discrete slots whose length is equal to the worst-case round-trip propagation time on the ether. After the first collision, the station waits either 0 or 1 slot time before another try. After the second collision, the station chooses either 0, 1, 2 or 3 slots randomly and waits that number of slot time. Thus, by the same rule, after **k**th collision, the number of time slots to wait is chosen randomly from the interval 0 to 2^**k**-1.

1. **Simulation Design**

* Simulation Platform: C++
* Form of Result: Numerical Data
* Initial Setup:
* The propagation time is assumed to be the smallest time unit in the entire design and set to be 1 time slot
* The “medium” is set to be a station with 1 time slot delay to all other stations
* The duration of the simulation is in the unit of time slots
* The states of all users during the whole process are saved in a large array and the size of it is the number of users times the duration.
* Basic Algorithm:



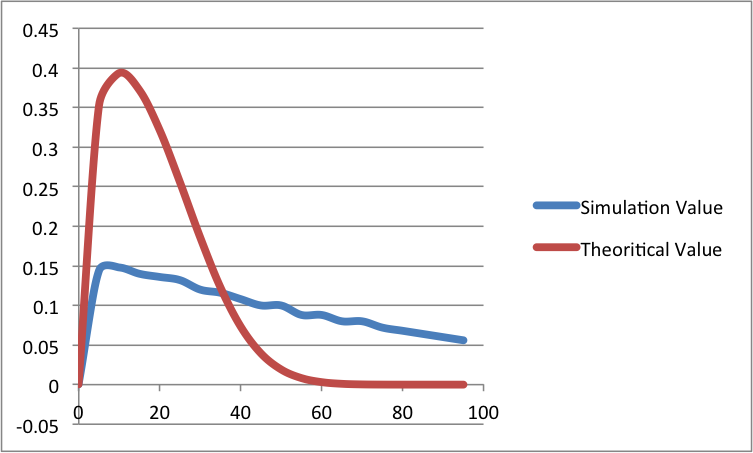
* The Simplified Procedure of the Program:

1. The program user sets the number of stations, the probability of a station having a frame to send and the duration of the program.
2. The program forms a new array and fills it with numbers other than 0 and 1.
3. Using the function named “setFrame” to make the array initialized with 0s and 1s. If the frame time is set to be 4, then every 4 consecutive 1s at a position of the array means a certain station at that time will have a frame to send.
4. The program set a “for” loop that repeats the value of duration times. In each loop, it contains another “for” loop, which repeats the number of stations times. In each small loop, the program checks the state of one station.
5. If the station has something to send and the medium is idle, then the station begins to send and the data it is sending will be received by the medium in the next larger loop.
6. If the station has something to send and the medium is busy but has no collisions, it first checks whether it sent data in the last two loops. If true, it keeps sending. If false, it waits a random number of slots based on the Binary Exponential Backoff Algorithm (BEBA). The act “Waiting” is realized by postponing the value in the array by a given slots. Values, which are moved out of bound, are just discarded and the newly - produced empty slots are filled by 0s.
7. If the station has something to send and the medium is busy and has collisions, it does the same backoff strategy as in the previous part.
8. In all other states, the station does nothing.
9. After each small loop, the program checks the state of the medium and report the change by changing values of several parameters. These parameters will be checked at the first of next large loop in order to simulate the one-slot delay.
10. Another array with comparatively small size will be used to record the number of successively transmitted frames of each station.
11. **Experiment Data**
12. For the original case, the frame time is set to be 4 slots. The whole process duration is 1000 slots, and the number of users is 10. The experiment data[[1]](#footnote-1) is in the following lists. (Since the theoretical values are measured without using BEBA, so in this case, all stations wait a random time from 0 to 1024 slots)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P value | 0.05 | 0.10 | 0.15 | 0.20 | 0.25 | 0.30 | 0.35 | 0.40 | 0.45 | 0.50 |
| Transmitted frames | 36 | 37 | 35 | 34 | 33 | 30 | 29 | 27 | 25 | 25 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P value | 0.55 | 0.60 | 0.65 | 0.70 | 0.75 | 0.80 | 0.85 | 0.90 | 0.95 | 1.00 |
| Transmitted frames | 22 | 22 | 20 | 20 | 18 | 17 | 16 | 15 | 14 | 13 |

The figure of the performance of this situation is shown below, including the theoretic value at this situation. [[2]](#footnote-2)

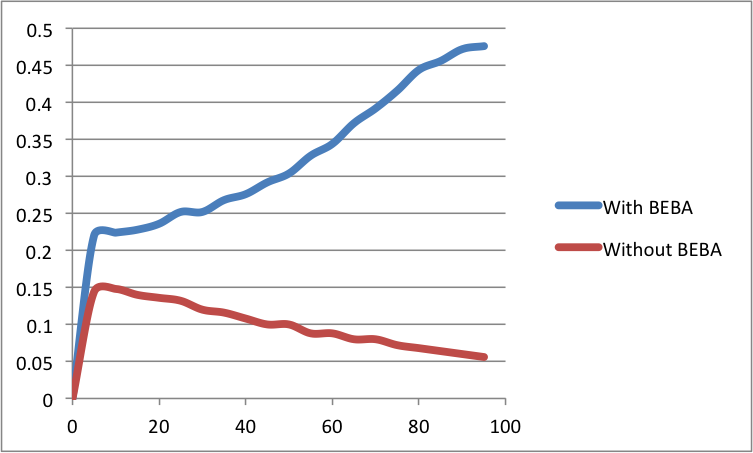
****

1. To show the influence of BEBA, the program simulates the same situation as in the first case both with BEBA and without it. The data of the one with BEBA is shown in the following list.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P value | 0.05 | 0.10 | 0.15 | 0.20 | 0.25 | 0.30 | 0.35 | 0.40 | 0.45 | 0.50 |
| Transmitted frames | 55 | 56 | 57 | 59 | 63 | 63 | 67 | 69 | 73 | 76 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P value | 0.55 | 0.60 | 0.65 | 0.70 | 0.75 | 0.80 | 0.85 | 0.90 | 0.95 | 1.00 |
| Transmitted frames | 82 | 86 | 93 | 98 | 104 | 111 | 114 | 118 | 119 | 119 |

The figure of the performance of this situation is shown below, comparing to the performance of the one without Binary Exponential Backoff Algorithm..

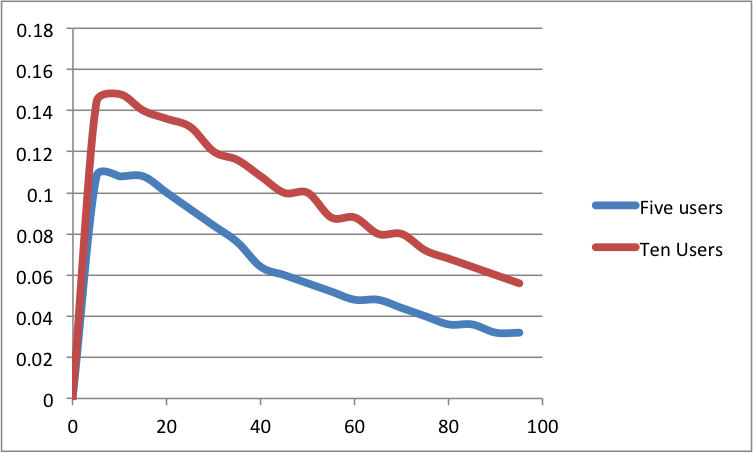


1. In order to find out the relationship between performance and number of users, the first case is tested again by changing the number of users to be 5. The experiment data is in the following lists.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P value | 0.05 | 0.10 | 0.15 | 0.20 | 0.25 | 0.30 | 0.35 | 0.40 | 0.45 | 0.50 |
| Transmitted frames | 25 | 28 | 27 | 26 | 21 | 21 | 18 | 18 | 16 | 14 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P value | 0.55 | 0.60 | 0.65 | 0.70 | 0.75 | 0.80 | 0.85 | 0.90 | 0.95 | 1.00 |
| Transmitted frames | 14 | 11 | 11 | 10 | 10 | 9 | 9 | 8 | 8 | 8 |

The figure of the performance of this situation is shown below, comparing to the performance of the one with ten users.

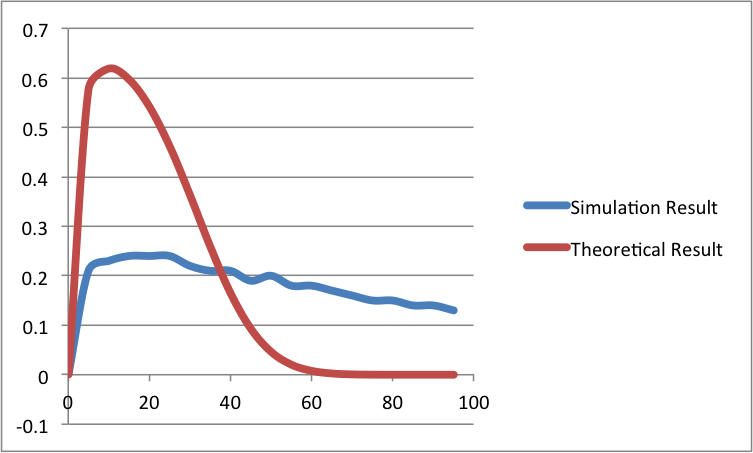


1. To see the efficiency change due to the ratio of propagation time and frame time, a new simulation is done with the set used in the first one except that the frame time is set to be 10. The whole process is 1000. The experiment data is in the following lists.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P value | 0.05 | 0.10 | 0.15 | 0.20 | 0.25 | 0.30 | 0.35 | 0.40 | 0.45 | 0.50 |
| Transmitted frames | 21 | 23 | 24 | 24 | 24 | 22 | 21 | 21 | 19 | 20 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P value | 0.55 | 0.60 | 0.65 | 0.70 | 0.75 | 0.80 | 0.85 | 0.90 | 0.95 | 1.00 |
| Transmitted frames | 18 | 18 | 17 | 16 | 15 | 15 | 14 | 14 | 13 | 13 |

The figure of the performance of this situation is shown below, including the theoretic value at this situation.



1. **Data Analysis**
2. From the figures of sample 1 and sample 4, which show the difference between experimental and theoretical results, it can be inferred by both curves that the highest transmission efficiency happens when n = 1/p, and G = 1. However, there are some obvious errors between two results. One of them is that the simulation cannot reach as high efficiency as in theory. The most important reason may be that in the process of calculating the theoretical result, it doesn’t take the influence of random waiting time into account because the theoretical result is just based on the probability of just one attempt per frame time. So with different scope of random waiting slots, the result may differ a lot.

Another big difference is that according to the theory, the efficiency at the point p = 1 should be zero. But in the simulation, the value is a positive integer. It is quite reasonable to see that because since the number of slots ranges from 0 to a number larger than half of the entire duration. If some of stations just wait a few slots while others wait much more time, then these stations would have enough time to finish their transmission. But this kind of transmission will cause a very large delay, which means that it is not suitable for many heavy-traffic users.

Between the two figures, we can see that the one with smaller A (propagation time divided by frame time) has better performance, which is the same as illustrated in the textbook.

1. In the second simulation, which compares the program both with and without BEBA. From the figure, it is clear that the one with BEBA performs much better than the one without BEBA. And instead of having bad performance, the one with BEBA has a better efficiency when p increases. Thus as a conclusion of the result of simulation, it is better to choose BEBA as the algorithm of setting random waiting time when the traffic is very heavy.
2. For the third simulation, it tells that when user numbers are larger, the performance will be better. This result is expected because when calculating the maximum efficiency, an approximation is made at (1-1/n)^(n-1) when n is large enough. However, one problem emerges. In the theory, the probability of a transmission success is Ps = np(1-p)^(n-1), and the maximum value happens when n=1/p. Thus the peak value of S should happens at p = 1/n, but in the figure, the two peak values of different curves happen at the same point. This error may be caused by the scale of simulation or some little errors in the design of simulation.

1. All the data is measured by running the program for 100 times and then takes their average values. [↑](#footnote-ref-1)
2. In all figures, the x – axis is the percentage of ratio of G and user numbers, and the y – axis is in unit of S. [↑](#footnote-ref-2)