|  |
| --- |
| University of North carolina at greensboro |
| SIMULATIONS OF BACKOFF PROCESS |
| Principles of Wireless Networks (CSC 568) Project Report |
|  |
| **Submitted by: Junjun Ruan Jie Hou Neha Singh** |
| **12/2/2014** |

|  |
| --- |
|  |

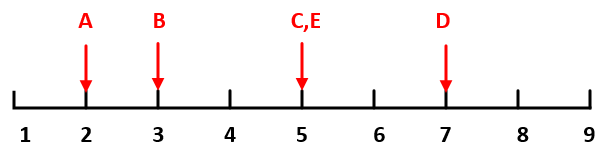
**Introduction**

The simulation of the backoff process in multi-hop wireless networks can be quite complex. The main objective of this project is to simulate the backoff problem in medium access control in wireless network and the motivation for constructing simulations is to investigate the backoff process. Here we will see the change in contention window with different number of active users and different time slots of each package length.

**System Model**

We model the system with time-slotted transmission and investigate the change in best contention window with highest throughput also known as optimum contention window, CW\* with the different number of active user nodes. The contention on wireless channels is modeled by comparing the chosen delays. Each node randomly chooses an integer delay, 1, 2, … , CW, when it has a chance to send a new transmission or a retransmission. The delay is added to current time to schedule its next transmission.

When more than one node chooses the same time slot to send packets, then collision will occur and transmissions will fail. Therefore the backoff process is needed to reschedule immediately by choosing a new randomly backoff number. A successful sender will do the same as well and the successful transmission increases by 1 and if there is a collision then the total tries increases by 1.For many time, we assume data packets to be exactly one time slot in length. What we need to pay attention is: when data packets take more than one time slot, the system remains the same, but the collision problem gets a little bit more complex when we implement it. Figure 1 indicates the model when data packets take 5 time slots. In the figure, we investigate the packet C at time 5. The packet occupies time slots 5, 6, 7, 8 and 9. Any package within time slots 5-9 will collide with C (in our figure, packet E and D will collide with C). The packet C can also collide with packets before it since all packets take 5 time slots. So packet starts at time slot 1-4 will collide with C (in our figure, packet A and B will collide with C).



**Figure 1: Collision model when L = 5, current time = 5**

We also consider the min/max fairness of each node which is the smaller throughput divided by the largest throughput where one node have different contention window than all other nodes with same contention window values. We examine the change in fairness with the change in contention window.

**Approach and Program Modules**

**Part 1:** In order to find out the relationship between the optimum contention window (CW\*) with the number of user nodes (N), we need to simulate different CW values for a fixed N, and pick up CW\* that get the maximum throughput. Here, each data packet is exactly one time slot in length. We choose an N which is a number of user nodes from N\_array = [20:20:500], and for each N, choose a contention window CW from CW\_array = [20:10:1000]. In the beginning, randomly assign each node with a transmission time from 1 to CW using:

transmission\_time = 1 + floor ( CW\_array(iCW) \*rand(1,N\_array(iN)))

When start to transmit packets, let the minimum transmission time as current time, i.e.

current\_time = min(transmission\_time)

For those nodes whose transmission time is current time, they start to transmit packets. Then check whether the current transmission is a success or not. If more than one node send packets at the same time then collision will occurs, otherwise, it will be a successful transmission and total\_success will increase by one, i.e.

If num\_of\_senders==1

total\_success = total\_success+1;

After this, update all the nodes which have just send packets and make them ready for next transmission by choosing a delay from 1 to CW, and then adding that delay to the current time, no matter whether the current transmission is a successful transmission or a collision, i.e.

transmission\_time(senders) = current\_time + 1 + floor(CW\_array(iCW) \*

rand(1, num\_of\_senders))

Repeat this process and for better results repeat this till current time is not less than 10000\*max(CW\_array).

Then compute the throughput, S i.e.

S(iCW) = total\_success/current\_time

Repeat this whole process for different contention window and then the contention window with the maximum throughput will be the optimum contention window for particular N. Then repeat this process for different user nodes N.

Finally, plot the graph for N and CW\*.

**Part 2:** In this part, we still need to find out the relationship between the optimum contention window (CW\*) with the number of user nodes (N). Compared to part 1, the only difference is that each data packet has a transmission time of L time slots (L>1). In order to run the programming with different L’s (L = 5 and L = 10), we define a function: CWN = part2(L), which set L as a parameter. In this function, we choose an N which is a number of user nodes from N\_array = [20:10:200], and for each N, choose a contention window CW from CW\_array = [20:20:4000]. For each CW, create an array named transmission\_last to record the last transmission time for collision nodes, what’s more, these collisions happed from current time to current time + L – 1. Before transmission, the collision time is assigned as all 0’s.

transmission\_last = zeros(1,N\_array(iN))

Then randomly choose a next transmission time (from 1 to CW) for each node with the particular N and contention window and store it in the array called transmission\_time i.e.

transmission\_time = 1 + floor ( CW\_array(iCW) \* rand(1, N\_array(iN)) )

Initially total number of success as 0 and will gradually increase with the transmission of packets.

Current time for transmission will be the minimum transmission time from the transmission time array i.e.

current\_time = min(transmission\_time)

Next step is to determine whether the node can transmit packet successfully or not at current time, because L > 1, we need to find the nodes (senders) whose transmission time is from current\_time - L + 1 to current\_time + L – 1, which can be calculated by

senders = find( transmission\_last > 0 & transmission\_last >= current\_time - L + 1

|transmission\_time <= current\_time + L - 1)

If the number of senders is only 1, this node will be sending package successfully, we need to increase value of total success as 1; otherwise, collision will happen.

if num\_of\_senders==1

total\_success = total\_success + 1;

end

Now we have to reschedule the transmission time or retransmission for the nodes that choose transmission time from current time to current time + L – 1 to send packets, no matter they can send successfully or not.

senders\_collision = find (transmission\_time <= current\_time + L - 1)

Now the new transmission time for those nodes collided will be calculated as

transmission\_time(senders\_collision) = current\_time + L + floor(CW\_array(iCW) \*

rand(1, num\_of\_collision))

Again the current time will be the minimum of transmission time in transmission\_time array and this whole will repeat till the current time is not less than 1000 times greatest contention window.

Then we will calculate the throughput for different contention windows by using

S(iCW) = total\_success\*L/current\_time

Then repeat these steps for different N’s and find the optimum contention window with maximum throughput for each N, i.e.

[~,index] = max(S);

CWN(iN) = CW\_array(index)

And then plot the graph for N and CW\*.

**Part 3:** Next we have to examine the min/max fairness with different contention window. Here min/max fairness is the smaller throughput by one node divided by the largest throughput by one node. Here we are assuming that if a node uses a smaller or larger contention window than every other nodes present are using, than the node’s throughput will be different. Then we have to check the behavior of one node’s throughput and system’s fairness.

We have assume number of active user nodes is N = 20, the contention window for 19 nodes is CWa = 35 and the contention window for 1 node is between 5 and 50 i.e.

CW\_array = [5:1:50]

Randomly choose a node with different contention window using

N\_diff = randperm(N,1)

Choose a contention window CW and calculate the transmission time and store it in the array using i.e.

transmission\_time = 1 + floor ( CWa \* rand(1,N))

Then find the array of next transmission times for N user nodes by using

transmission\_time(N\_diff) = 1 + floor ( iCW \* rand)

Current time will be the minimum transmission time from transmission time array i.e.

current\_time = min(transmission\_time)

Then find the user nodes sending packet called senders by using

senders = find(transmission\_time==current\_time)

And find the number of user nodes sending packets at the current time by using

num\_of\_senders = size(senders, 2)

Now check whether the current transmission time slot is a success or a collision for nodes sending packets in that current time slot and for that if there is only one node sending packets in that current time then it will be successful transmission i.e.

if num\_of\_senders==1

node\_success(senders) = node\_success(senders) + 1;

end

Otherwise if there is more than one node sending packets in the current time, then collision will occur and transmission will fail.

Now update all the nodes which have just sent packets by choosing a random delay from 1,2,......, CWa i.e.

transmission\_time(senders) = current\_time + 1 + floor(CWa \* rand(1, num\_of\_senders))

And for the node with different contention window CW, choose a delay from 1,2,.....,CW i.e.

if ismember(N\_diff,senders ) == 1

transmission\_time(N\_diff) = current\_time + 1 + floor(iCW \* rand);

end

Again current time will be the minimum transmission time from transmission time array and repeat this process till current time is not greater than 1000 times greatest contention window from contention window array i.e.

current\_time < 10000\*max(CW\_array)

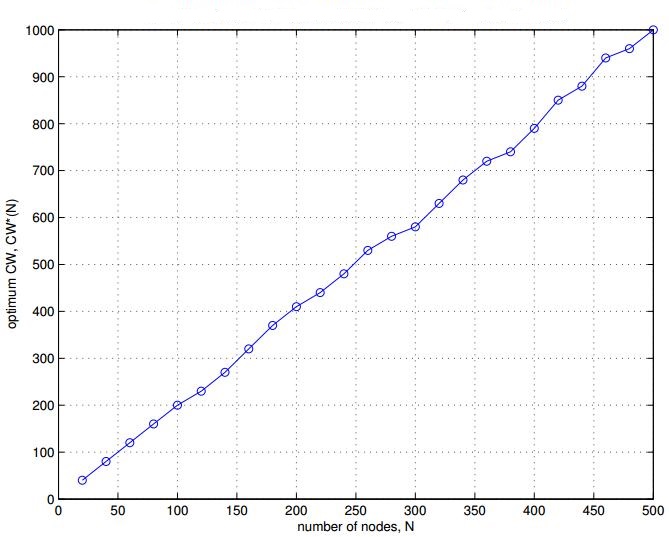
Next step is to calculate the min/max fairness using maximum throughput of one node and minimum throughput of one node i.e.

Fairness(iCW) = min(S)/max(S)

Repeat the whole process for different contention window and observe the results obtained.

**Results and Discussions**

For **part 1**, we have to investigate the change in optimum contention window with the change in number of active user nodes. So we have taken different CW’s to apply backoff process with different N’s and obtain the following relationship as shown in figure 2.



**Figure 2: Relationship between N and CW\* with L = 1**

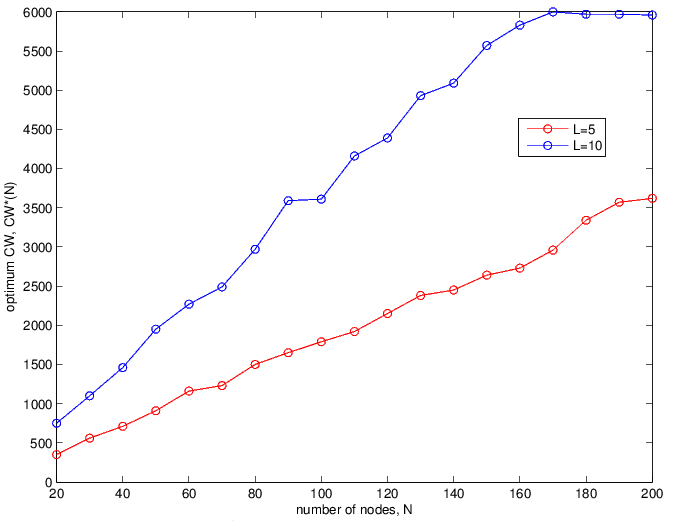
From the figure 2, we can observe that with the increasing in number of nodes, N, the optimum contention window, CW\*, is also increasing. From the figure, it is obviously that optional contention window CW\* is directly proportional to the number of nodes, N. We fit the curve and get the equation:

CW\* = 2N

Which means that the optimum contention window CW\* is twice of the nodes number.

The reason is: for each N, when CW < 2N, it has less time to choose, so it has less successful sending times. Thus we get smaller throughput. When CW > 2N, although it is easier to send packet successfully, the number of node is too smaller compared to contention window, thus it waste a lot of time, and each node has less chance to schedule a next transmission. So it still get lower throughput.

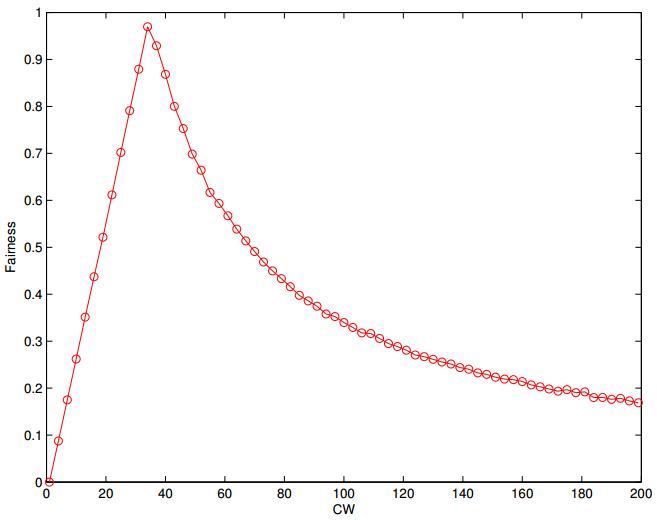
For **part 2**, we have to investigate the relationship between optimum contention window CW\* and the number of active user nodes, N with each data packet’s transmission time L = 5 and L = 10. We investigate the optimum contention window CW\* with different N by choosing different contention window and applying backoff process, and the results is shown in figure 3.



**Figure 3: Relationship between CW\* and N with different L’s**

From Figure 3, we can see the relationship between N and CW\* is almost linear. When L = 5, CW\* = 18N, and when L = 10, CW\* = 38N. The reason is as follows: when L>1, it is hard to send packet successfully, if only one node chooses to send packet in current time, no other nodes choose transmission time from current\_time - L + 1 to current\_time + L – 1, it will make a successful transmission. Actually we can find the rule from figure 2 and 3, for L = 1, only one node chooses to send packet at each 1 time slot, CW\* = 2N = 2N; for L = 5, only one node chooses to send packet during each 9 time slots will lead to successful transmission, thus CW\*= 92N = 18N; for N = 10, only one node chooses to send packet during each 19 time slots will lead to successful transmission, thus CW\*=192N=38N.

For **part 3**, we investigate the min/max fairness with one of all the nodes use a different contention window. We use total nodes number of 20. Nineteen of them use contention window CW = 35, one node chooses a different CW. We change the CW of this particular node explore the relationship with fairness.

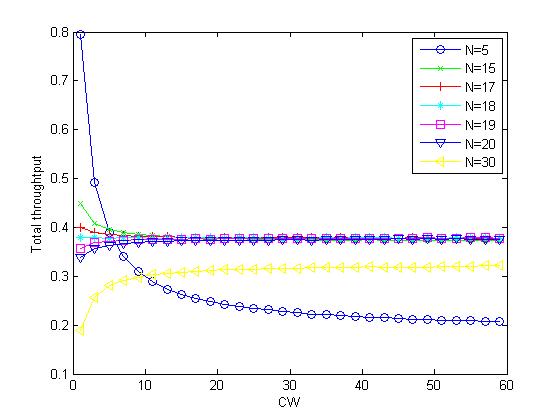


**Figure 4: Relationship between min/max fairness and CW**

We observe from figure 4 that: the fairness increase before CW=35 as CW increase; at CW=35 it reach the maximum fairness which is ≈ 1; after CW=35, the fairness decrease. We explain the result as follow:

All the 19 node are using contention window CW=35, one particular node use a different CW. When this particular node use a smaller CW than the rest 19 nodes, this particular node get more chance to send packets than others, which is unfair to other nodes. And the smaller the CW this particular node use, the more chance to send packets this node get, so more unfair the system is. When this particular node chose a bigger CW than the 19 nodes, it has less chance to send packets. And the larger the CW this particular node use, the less chance to send packets this node get, so more unfair the system is. When this particular choose the same CW with the rest of 19 nodes, every nodes have the same chance to send packets, the fairness reach to 1.

We also investigate the relationship between contention window of the particular node and the system throughput when number of nodes N is different. In this system, we still choose one node use a different CW with others, and the rest of nodes use CW = 35 as before.



**Figure 5: Relationship between CW and throughput S with different N**

From the figure 4, we observe that the particular node with different CW has different impact to the throughput of system when total nodes numbers are different. When the system has less number of nodes, such as N < 18, the small CW of the particular node enhance the system throughput. When N = 18, this particular node has no much influence to the system throughput; when N > 18, the small CW of the particular node restrains the system throughput. We explain it as following:

Obviously, small CW of the particular node with different CW bring this node more chance to send packet, which can enhance the throughput. But also, the small CW of this node make more collision with other nodes, which can restrain the throughput. So there is always a competition of them.

From figure 5, we can see the competition. When the system has less nodes, such as N<18, the small contention window bring more chance to send packets than collision, so the less the nodes number is, the higher the throughput is. When N=18, it seems the system find a balance between them, so the change of the contention window of the particular node has not much effect on the throughput. When N>18, the small contention window bring more collision than

**Conclusions**

After constructing the simulations for backoff process we observed that the relationship between optimum contention window and number of user nodes is linear and optimum contention window increases twice the number of nodes increases with time slot of each packet transmission equal to 1. When packet takes 5 time slots, optimum contention window increases 18 times.

Min/Max fairness is maximum (approx 1) for contention window equal to 35.

**Code for Part 1:**

**Part1.m**

N\_array = [20:20:500];

CW\_array = [20:10:1000];

LN = length(N\_array);

LCW = length(CW\_array);

CWN = zeros(size(N\_array)); %optimal CW for maximum throughput

for iN=1:LN

S = zeros(size(CW\_array));

for iCW=1:LCW

%choose among 1, 2, ... CW.

transmission\_time = 1+floor ( CW\_array(iCW) \* rand(1, N\_array(iN)) ); %array of next transmission times for N nodes

total\_success = 0;

total\_tries = 0;

current\_time = min(transmission\_time);

while current\_time < 10000\*max(CW\_array)

%check whether the current time slot is a success or a collision index

%of nodes who are sending in the current time slot

senders = find(transmission\_time==current\_time);

num\_of\_senders = size(senders, 2);

%when there is one sender, it's a success

if num\_of\_senders==1

total\_success = total\_success+1;

end

%only update those nodes that have just sent packets and choose a delay

%from 1, 2, ... CW.

transmission\_time(senders) = current\_time + 1+floor(CW\_array(iCW) \* rand(1, num\_of\_senders));

current\_time = min(transmission\_time);

end

S(iCW) = total\_success/current\_time;

end

%compute optimal CW for maximum throughput

[max\_s,index] = max(S);

CWN(iN) = CW\_array(index);

end

save('p1\_CWN20aaa.mat','CWN');

N\_array = [20:20:500];

plot(N\_array, CWN, 'o-');

xlabel('number of nodes, N');

ylabel('optimum CW, CW\*(N)');

grid on;

print -depsc2 p1.eps

**Code for Part 2:**

**Part2.m**

function CWN = part2(L)

N\_array = [20:10:200];

CW\_array = [20:10:6000];

LN = length(N\_array);

LCW= length(CW\_array);

CWN= zeros(size(N\_array)); %optimal CW for maximum throughput

for iN=1:LN

S = zeros(size(CW\_array));

for iCW=1:LCW

transmission\_last = zeros(1,N\_array(iN)); %array record last collision time

%choose among 1, 2, ... CW.

transmission\_time = 1+floor ( CW\_array(iCW) \* rand(1, N\_array(iN)) ); %array of next transmission times for N nodes

total\_success = 0;

total\_tries = 0;

current\_time = min(transmission\_time);

while current\_time < 1000\*max(CW\_array)

%check whether the current time slot is a success or a collision index

%of nodes who are sending in the current time slot

senders = find( transmission\_last > 0 & transmission\_last >= current\_time - L + 1 |transmission\_time <= current\_time + L - 1);

num\_of\_senders = size(senders, 2);

%when there is one sender, it's a success

if num\_of\_senders==1

total\_success = total\_success + 1;

end

%shedual retransmission time for senders whose transmission

%time is from current time to current\_time + L-1

senders\_collision = find (transmission\_time <= current\_time + L - 1);

transmission\_last(senders\_collision) = transmission\_time(senders\_collision);

num\_of\_collision = length(senders\_collision);

transmission\_time(senders\_collision) = current\_time + L + floor(CW\_array(iCW) \* rand(1, num\_of\_collision));

current\_time = min(transmission\_time);

end

S(iCW) = total\_success\*L/current\_time;

end

%optimum CW of highest throughput

[~,index] = max(S);

CWN(iN) = CW\_array(index);

end

save(strcat('p2\_L',num2str(L),'.mat'),'CWN');

**part2\_plot.m**

CWN\_L5 = part2(5);

CWN\_L10 = part2(10);

plot(N\_array, CWN\_L5, 'o-r');

hold on;

plot(N\_array, CWN\_L10, 'o-b');

hold on;

xlabel('number of nodes, N');

ylabel('optimum CW, CW\*(N)');

set(gca,'YTick',[0:250:3000])

legend('L=5','L=10',0)

print -depsc2 p2\_L.eps

**Code for Part 3:**

**Part3.m**

N = 20;

CWa = 35;

CW\_array = [1:3:200];

N\_diff = randperm(N,1);%randomly choose a node with different CW

Fairness = zeros(1, length(CW\_array));

S\_total = zeros(1, length(CW\_array));%sum throughput for different CW

for iCW=CW\_array

node\_success = zeros(1,N);

S = zeros(1,N);

%choose among 1, 2, ... CW.

transmission\_time = 1 + floor ( CWa \* rand(1,N)); %array of next transmission times for N nodes

transmission\_time(N\_diff) = 1 + floor ( iCW \* rand);

node\_success = zeros(1,N);

current\_time = min(transmission\_time);

while current\_time < 10000\*max(CW\_array)

%check whether the current time slot is a success or a collision index

%of nodes who are sending in the current time slot

senders = find(transmission\_time==current\_time);

num\_of\_senders = size(senders, 2);

%when there is one sender, it's a success

if num\_of\_senders==1

node\_success(senders) = node\_success(senders) + 1;

end

%only update those nodes that have just sent packets and choose a delay

%from 1, 2, ... CWa.

transmission\_time(senders) = current\_time + 1 + floor(CWa \* rand(1, num\_of\_senders));

%for node with different CW, choose a delay from 1,2,...CW

if ismember(N\_diff,senders ) == 1

transmission\_time(N\_diff) = current\_time + 1 + floor(iCW \* rand);

end

current\_time = min(transmission\_time);

end

%record throughput of each node

S = node\_success/current\_time;

%compute min/max fairness

Fairness(iCW) = min(S)/max(S);

%compute total throughput in each CW

S\_total(iCW) = sum(S);

end

save('p3\_fairness.mat','Fairness');

save('p3\_S\_total.mat','S\_total');

figure

plot(CW\_array, Fairness(CW\_array), 'o-r');

xlabel('CW');

ylabel('Fairness');

print -depsc2 p3\_fairness.eps

figure

plot(CW\_array, S\_total(CW\_array), 'o-b');

xlabel('CW');

ylabel('Total throughtput for each each CW');

print -depsc2 p3\_S\_total.eps