ECE/CS 438: Communication Networks **Fall 2022**

Machine Problem 4

Handed Out: Nov 15^{th}*,* 2022 *Due: Dec* 7^{th}*, 23:59, 2022*

TA: Yu-Lin Wei

This machine problem tests your understanding of the random backoff and medium access in a wired network.

1 The Scenario

In this assignment, you will develop a toy simulator that evaluates the performance of simplified CSMA protocols in a wired network. In this toy simulator, start with two nodes only – nodes A and B – and assume each node always has packets to transmit to its counterpart (later you will increase the number of nodes). To transmit a packet, a node picks an integer random number from the range [0, R) and counts down as long as the channel is idle. If the channel gets busy (because say node B has already started transmitting, or due to a collision with another transmission), node A freezes its countdown and waits till the channel is idle again. Once B finishes and the channel is idle, node A resumes countdown; node B picks a new random number and also starts counting down at the same time. In the actual assignment we will have more than 2 nodes and all nodes will perform the same operation.

Now let's see what happens when a collision occurs. Only nodes that were transmitting at the time of the collision doubles the value of R (unless specified otherwise) and also increments their collision count. When a node's collision count reaches a maximum value of M, the node gives up and drops this packet. It then resets R to its minimum specified value, resets the collision count to zero again, and attempts a fresh new transmission.

An input file will provide you the following configuration parameters: number of nodes N (not just 2), the packet length L, the minimum value of R (and how it increments), the value of M, and the total simulation time T. Your goal is to evaluate the network utilization, as described later.

2 Simplifying Assumptions

To keep the code simple, here is a list of assumptions you can make:

1. This simulation entirely takes place within one program, just like mp3.
2. Assume that clocks at all nodes are perfectly synchronized, which means all the nodes have access to a global clock.
3. Assume that the transmission time of data packets are in multiples of clock ticks (e.g., a packet might last for say 10 clock ticks); also assume that the propagation delays between all nodes are zero.
4. Assume that collisions are the only reason for packet corruption – if there is no collision, the packet is always received correctly; also assume no ACKs are sent by the receiver.
5. Assume that any processing (such as picking random numbers) takes zero time.
6. To fix the output and allow autograding, please use a pseudo number generator to set the backoff time.

$$backoff = mod(nodeID + ticks, R) \tag{1}$$

where $nodeID$ and $ticks$ both start from 0, and R is the backoff window.

7. At the start of each tick, please check if there are packets being transmitted or if multiple packets are colliding at that tick. If there is no transmission or collision (i.e., channel is idle), then all the nodes that have packets to transmit count down by one tick.

3 Hints on How to Simulate

Now, here are hints on how you can develop such a toy simulator. Imagine that the clock is a global integer variable which increments perhaps in a `for` loop. This simulates the passage of time. Now for every increment of the clock, all events that happen at that time can be inside the `for` loop. Now say two nodes are two structures that have their own backoff values, e.g., `nodeA.backoff` and `nodeB.backoff`. The operation of counting down can be simulated by decrementing the `nodeA.backoff` and `nodeB.backoff` variable in each iteration of the loop. Of course, before counting down the channel needs to be sensed, which can be simulated by a global flag, say `channelOccupied`. Once a node reaches zero, say A, it can begin transmission essentially by *setting* the `channelOccupied` flag. Node B does not decrement its counter in this iteration since the `channelOccupied` flag is already set.

If the packet lasts for say 10 clock ticks, then 10 iterations of the for loop occur without any significant activity. After that, node A pretends that the packet has been transmitted, and hence, *resets* the `channelOccupied` flag, and assigns a new random number to `nodeA.backoff`. In the next iteration, both nodes can decrement their backoff values, simulating the notion of a resumed count-down. *Note that no real transmission of the packet is necessary.*

Of course, the above is a sketch and some details are missing. For instance, you need to determine how collisions can be simulated and accordingly count how many packets are getting transmitted correctly. You also need to carefully design the order of events—in a given iteration of the for loop, node A should not decrement its backoff value if node B is going to set the `channelOccupied` flag later. Put differently, you need to check if there is any node who might set the `channelOccupied` flag—if no node does so, only then can all nodes decrement their backoff.

4 Input Formats

Your program will be run using the following command: `./csma inputfilename`
The input file will contain five items delimited by newlines. Each line starts with a character denoting the parameter, followed by the value(s) of the parameter.
Note that we have swapped the M and R rows compared to the sample input in the released code, so that you can know the length of R after reading M.

Sample Input:
```
N 4
L 2
M 6
R 4 8 16 32 64 128
T 10
```

This means that the number of nodes N is 4, the packet size L is 2 clock ticks, the initial random number range is $[0,R = 4)$ and R is increased to 8, 16, 32 and so on for each consecutive collision. The maximum retransmission attempt M is 6, implying that after the 6th collision the node should drop the packet and reset R to the minimum value 4. Finally, T denotes the total time of simulation in terms of clock ticks.

Figure 2 shows the back-off values of each node at the start of each tick. Green slot denotes a successful transmission, and orange slot indicates there is a collision. In both of these cases, other nodes will pause the count down (marked as dark grey slots). Note that a node will set its countdown using the pseudo number generator Eq. 1.

Node/T	init	0	1	2	3	4	5	6	7	8	9
N0	0	0	0	2	1	1	1	0	7	6	6
N1	1	1	1	1	0	0	2	1	1	0	2
N2	2	2	2	2	1	1	1	0	1	0	11
N3	3	3	3	3	2	2	2	1	1	0	4

Figure 2: Back-off values at the start of each tick for every node.

For example, at the start of tick 5, node 1 will set its backoff to $mod(1+5,4) = 2$. At the start of tick 7, node 0 will set its backoff to $mod(0+7,8) = 7$

5 Output Formats

Write the link utilization rate (round to 2 decimal place) to a file called "output.txt"

Sample Output:
```
0.40
```

Note that the link utilization rate is (the number of slots where a packet is transmitted without collision) divided by T.

6 Notes

All the notes for the previous MPs still apply. We are not repeating those here for brevity.

New information:

1. Your project must include a Makefile whose default target makes executables called csma

2. Command line format: `"./csma inputfile"`
3. Please submit C/C++ code.
4. Please strictly follow the input/ output format.

Submission instructions are same as for mp3. Tests generally take 10-15 minutes, and there may be a queue of students. You can see where you are in the queue at `http://cs438fa22.csl.illinois.edu:8080/queue/queue_mp4.html`. This is a UIUC-private IP.

Do refer to previous MP instructions for other notes.