ECE/CS 438: Communication Networks **Fall 2020**

Machine Problem 4

Handed Out: Nov 20^{th}*,* 2020 *Due: Dec. 6, 2020 (5:00pm)*

TA: Zhijian Yang

Abstract

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. 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 started transmitting), 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.

Upon every collision, R is doubled (unless specified otherwise) and the node that experienced the collision increases its collision count. When this 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. The 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 will be provided as an input file. Your goal is to evaluate the performance of such networks.

2 Simplifying Assumptions

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

1. To be clear, this simulation all 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 delay between all nodes in 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.

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 need 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 channel Occupied 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 File

The input file should be named input.txt and should be organized as follows. Of course, when grading this assignment, TAs may input different values for these variables.

```
N 25
L 20
R 8 16 32 64 128 256 512
M 6
T 50000
```

This means that the number of nodes N is 25, the packet size L is 20 clock ticks, the initial random number range is [0,R = 8] and R is increased to 16, 32 and so on for each consecutive collision. The maximum retransmission attempt M is 6, implying that after 6 collisions the node should drop the packet. Finally, T denotes the total time of simulation in terms of clock ticks.

5 What You Need to Submit

1. Your code, the input and output files, and a report (see details below). We have not created any starting code for you. Organize the files similar to previous MPs. Also add the report to the folder. Compress it and upload the entire zip file onto Gradescope. **Please use the Gradescope built-in function to add your partner.**

2. The output of your program should be written to an output file named `output.txt`, and organized as follows:
   Channel utilization (in percentage) ???
   Channel idle fraction (in percentage) ???
   Total number of collisions ???
   Variance in number of successful transmissions (across all nodes) ???
   Variance in number of collisions (across all nodes) ???

3. By default, assume the following parameters: `N=25, L=20, R=8 16 32 64 128 256 512, M=6, T=50000` provided to your program through an `input.txt` file. Now, plot graphs for the following scenarios where certain parameters are varying. **Important: for each simulation, run at least 100 times (with different random seeds) so that the results can converge.**
   (a) Plot how channel utilization (in percentage) varies with increasing number of nodes (i.e., N varying from 5 to 500). Channel utilization is defined as the ratio of clock ticks that were used up for correct communication to the total number of clock ticks, T.
   (b) Plot how the channel idle fraction (in percentage) varies with increasing number of nodes (i.e., N varying from 5 to 500). Channel idle fraction is defined as the ratio of unused clock ticks to the total number of clock ticks, T. Note that unused clock ticks do not include collisions.
   (c) Plot the total number of collisions with increasing number of nodes (i.e., N varying from 5 to 500).
   (d) Repeat part (a) but plot 5 curves on the same graph, each curve corresponding to different initial values of R: 1, 2, 4, 8, 16. For each of the 5 cases, let R double upon collisions.
   (e) Repeat part (a) but plot 5 curves on the same graph, each curve corresponding to different packet lengths L: 20, 40, 60, 80, 100.
   (f) Explain the shape of the curves in (d) and (e) by elaborating on how/why increasing value of N, R, and L impact channel utilization.

   Submit a pdf to Gradescope, `analysis.pdf`, with all this explained.

Please label the graphs carefully and make sure that curves and legends are legible - points will be deducted if graphs are missing information or difficult to read.

Final Note: Your project must include a Makefile whose default target makes executables called csma. Command line format: "`./csma input.txt`"