

CS3210 Computer Networks Lab  
Even Sem. 2017, Prof. Krishna Sivalingam  
Lab 7: Slotted Aloha Protocol  
Due date: March 21, 2017, 11PM, On Moodle

March 7, 2017

## 1 Description

The objective of this project is to implement the Slotted Aloha protocol, as discussed in class.

**Input:** The command-line input parameters are:

1. “-N integer”: number of users sharing the channel
2. “-W integer”: specifies initial value of Collision Window size for all nodes ( $W(i)$ ; default: 2).
3. “-p double”: specifies PACKET\_GEN\_RATE ( $p$ ): probability of packet generation per unit time per node
4. “-M integer”: MAX\_PACKETS

### Protocol Operation:

Time is slotted. All packets are of fixed length, equal to one slot. This can be simulated using a while loop and a variable called *SimTime*. The variable is initialized to 0, and incremented by one in each loop instance.

Initially, all nodes are non-backlogged, i.e. do not have a packet to transmit. Each node's buffer can hold at most two packets (including the packet for which transmission is being attempted). The packet is removed from the buffer after it has been successfully transmitted.

The main steps of the program are:

1. In each time slot, every non-backlogged node will generate a packet with probability  $p$ . If there is space in the node's buffer, the packet will get added to the buffer, else the packet is dropped.

Note: This packet generation step precedes the packet transmission step.

2. A node will attempt transmission in a slot if: (i) a new packet was added to an empty buffer in the beginning of the slot; (ii) if the backoff counter goes to zero, in which case the packet at the head of the queue is re-transmitted.
3. A packet transmission is successful if only one node attempts transmission during a slot.  
Assume that there is a “centralized” view of the network, that keeps track of the number of attempted transmissions in a slot. If this number equals one, then packet success is recorded; if the number is greater than one, collision is recorded.
4. If node  $i$ 's transmission attempt is successful, then  $W(i) = \min(2, W(i) * 0.75)$ . Remove the packet from the buffer; the node becomes non-backlogged at the end of the slot, if the buffer is empty and backlogged, if there is another packet in the buffer.
5. If node  $i$ 's transmission attempt is not successful (i.e. there is collision since two or more nodes attempted transmission in the same slot), the node enters backoff state and is considered backlogged. It will generate a random number  $k$ , which is between 0 and  $(W(i) - 1)$ , and attempt retransmission in the  $k^{th}$  slot following the current slot. That is,  $k$  represents a backoff counter that is decremented after each slot; when  $k = 0$ , it will attempt retransmission in that slot. It also updates  $W(i) = \max(256, W(i) * 2)$ , to be used for the next collision occurrence.
6. Calculate: (i) mean delay per packet and (ii) average utilization per slot (throughput), i.e. number packets sent per slot.

The program terminates after MAX\_PACKETS (a command-line parameter) have been successfully transmitted combined across all nodes (OR) if the maximum retransmission attempt for any packet exceeds 10.

**Output:** On termination, the program will print the following information, in a single line to the screen:

1. Number of nodes,  $W$  and  $p$
2. Utilization: Average number of packets successfully sent per slot
3. Average Packet Delay: Time between packet generation and successful transmission.

Write a shell script (bash, perl, etc.) that will generate throughput values for varying  $N = 50$ ,  $p \in \{0.01, 0.02, 0.03, 0.05, 0.1\}$  and  $W \in \{2, 4\}$  values. Generate one graph with Utilization on the  $y$ -axis,  $p$  on the  $x$ -axis and one line for each value of  $W$ . Prepare a brief report that includes this graph and your observations based on the graph.

## 2 What to Submit

The platform for this project will be Linux and C/C++/Java. Create a tar-gz file with name: Lab7-RollNo1.tgz (e.g., Lab7-CS14B110.tgz).

The directory should contain the following files:

- Source File(s)
- Makefile and Script File  
Typing command 'make' or your script program, at the UNIX command prompt, should generate all the required executables.
- A Script file obtained by running UNIX command *script* which will record the way you have finally tested your program.
- Technical Report (as above)
- a README file containing instructions to compile, run and test your program.

### 3 Help

1. Ask questions EARLY and start your work NOW (really, no choice). Take advantage of the help of the TAs and the instructor.
2. Use random() or similar (divide random() by RAND\_MAX if needed). Make sure that you seed the random number generator before used.
3. Submissions PAST the extended deadline SHOULD NOT be mailed to the TAs. Only submissions approved by the instructor or uploaded to Moodle within the deadline will be graded.
4. Demonstration of code execution to the TAs MUST be done using the student's code uploaded on Moodle.
5. NO sharing of code between students, submission of downloaded code (from the Internet, Campus LAN, or anywhere else) is allowed. The first instance of code copying will result in ZERO marks for the Lab component of the Course Grade. The second instance of code copying will result in a 'U' Course Grade. Students may also be reported to the Campus Disciplinary Committee, which can impose additional penalties.
6. Please protect your Moodle account password. Do not share it with ANYONE. Do not share your academic disk drive space on the Campus LAN.

### 4 Grading

- Aloha Code working correctly: 70 points
- Shell script and output for varying parameter values: 20 points
- Viva Voce: 10 points