Friday, 21 December 2018

Network Simulation

Laboratory 3 - “Finite State Machines”

DEADLINE: January 11, 2019

In this lab, you will use the same topology from Lab 2, and will implement a “real” Stop&Wait protocol.
You MUST use OMNET++ finite state machines (FSM) to implement *txApp* and *rxApp* as follows.

1. *txApp* FSM: the first message will be generated by *txApp* of every node at the beginning of the simulation, after a random time T_{ia}, negative exponentially distributed with parameter 0.1s. The message is sent to node $(i+G) \bmod 25$, where i is the originator node index and G is your group number. The message must have Sequence Number (seqNo) = 0. *txApp* will set a timeout and will retransmit the message if the timeout expires. If a message “ACK(0) received” arrives from the *rxApp* in the same node, the timeout is cancelled and the transmission of a new message, with seqNo = 1 is scheduled after a random time T_{ia}, negative exponentially distributed with parameter 0.1s. The transmission of this message too is regulated by timeout and ACK (message “ACK(1) received" from rxApp). When the process is complete, after a random time, the next message will be transmitted and it will have seqNo=0 again, and so on. *txApp* must discard and ignore out-of-order ACKs (e.g., any ACK 0 received while *txApp* is waiting for ACK 1).

2. *rxApp* FSM: it starts by waiting for a data message with seqNo=0. If the data message is received, it will send it back after swapping addresses and marking it as an ACK. It will then wait for a message with seqNo=1, repeating the procedures above when the message is received. If it receives an out-of-order message, it must ~~discard and ignore it~~ ignore it (not change state) and return it to the sender. If it receives an ACK with seqNo=n, with n={0,1}, it must send an “ACK(n) received” message to *txApp* on the same node.

The *linkLayer* module will not need a FSN. Its behaviour is as follows.

- It will receive new data messages from *txApp* and it will deliver data messages and ACK messages to *rxApp* when they have reached the destination.
- It will route any message with seqNo=0 using Row/Column routing, while it will use Random Routing for any message with seqNo=1.
- It will not lose a message based on probability but it will discard any message whose hop count is larger than 30 (therefore, make sure you have a hop count field in your message definition). The hop count resets when the messages reaches its destination, whether it is a data message or an ACK (i.e., when an ACK is returned by *rxApp* it will start with hop count =0).

Let the simulation end when each node has generated 10000*G/10 messages.
Computing the delivery delay as the time needed to receive a message *including* the time taken up by its retransmission(s), collect the same statistics as a function of L as in Lab Task 1.2.

By January 11, submit a written report with:

- A diagram representing the FSM of *txApp* and *rxApp*, highlighting steady and transient states.
- The delivery delay values as a function of L as in Lab Task 1.2.
- The .ini, .ned and C++ files you wrote for each of the tasks. Be sure to exhaustively comment your code.