# **Lab 3 - Task 4**

In this task, you will implement the sender function of the stop-and-wait protocol.

**INSTRUCTIONS**

The MATLAB code in the window below is similar to that in Task 1, but you will implement the function **sender\_stopwait()**. As described in Task 1, this function sends the packets contained in the variable **send\_packet\_list** to the receiver using the stop-and-wait protocol.

In order to implement this protocol, the sender maintains several variables:

1. **num\_packets** stores the total number of packets to be sent
2. **send\_packet\_num** indicates the sequence number of the packet that is to be sent or is waiting to be acknowledged
3. if an acknowledgment has been received, **ack\_num** indicates the sequence number of the packet that has been acknowledged
4. **send** is a boolean flag indicating whether to send a packet or not.

In each iteration, the sender should first check whether to send a packet or not. There are two conditions that may trigger a transmission/retransmission:

(1) The timeout expires while the sender is waiting for the acknowledgement of the current packet to come back. The sender checks whether the timeout has expired using the command **timeoutExpired()**, which returns true if the timeout has expired, and false otherwise. If the timeout has expired, the sender should re-transmit packet number **send\_packet\_num**, by setting **send** equal to true.

(2) An acknowledgement comes back. The sender checks for an acknowledgement using the command **send\_ack = sender\_get\_ack()**. If an acknowledgement has arrived, then **send\_ack** contains the acknowledgement packet, otherwise it is empty. The acknowledgement packet contains the 8 bit sequence number of the acknowledged packet. This can be converted into a decimal sequence number **ack\_num** by the command **binvec2dec(send\_ack)**. If the acknowledgement packet's sequence number (**ack\_num**) is the same as the sequence number of the acknowledgement that the sender is waiting for (**send\_packet\_num**), then the the sender should increment **send\_packet\_num** to indicate that the next packet should be sent and set **send** equal to true to trigger a send. However, if **ack\_num** is equal to **num\_packets**, then there are no more packets to send, and the sender should trigger the simulation to stop by setting **run** equal to false.

If a packet should be sent (**send** = true), then the sender should get packet number **send\_packet\_num** from the matrix **send\_packet\_list** and transmit it using the command **sender\_send\_packet(packet)**. It should then reset the timeout timer to wait for **time\_out** iterations with the command **resetTimeout (time\_out).**

Your task is to revise the code to implement the sender. Specifically, you will implement the code to handle the case (2) as described above. Revise the code between the lines

% % % % Revise the following code % % % %

and

% % % % Do not change the code below % % % %

Please do not change other parts of the code.