# LAB 3 - TASK 1 (EXTERNAL RESOURCE) (0.0 / 1.0 points)

# Lab 3 - Task 1

In this task, we will get an overview about the transport layer simulation and measure the throughpu and stop protocol.

## INSTRUCTIONS

The MATLAB code in the window below simulates the stop and wait protocol. Note that the simulatio network layer is different from that in the physical layer where we are only concerned with the one-v transmission from the transmitter to the receiver. In the network layer, the receiver also sends ackn (ACK) packets back to the transmitter, which when received by the transmitter will affect its next act result, we need to take the "time dimension" into consideration when simulating the behavior in the layer.

In our simulation, we assume a discrete-time model where we divide time up into different slots, and such as packet transmissions, happen within one slot. Slots are indexed by integer time indices. Fo the transmitter transmits a packet at time index **1** and the transmission delay is 4, then the packet w the receiver at time index **5**. Similarly, the parameter "timeout" is also set to be an integer number. this is similar to what we did with slotted Aloha in the network layer.

Let's now look at the code and see how the simulation works.

The code first initializes a text message (variable **tx_msg**), which we can think of as a message at t application layer that is passed to the transport layer. It then defines several simulation parameter

1. **p_loss** indicates the probability that a packet is lost on its way between sender and receiver. In is determined by what happens at the network, link and physical layers in the nodes along the path to receiver.

2. We assume the transmission delay between the transmitter and receiver follows a discrete unifor distribution between two values **d_min** and **d_max**. In practice, this is also determined by what hap lower layers, say, for example, routing by the network layer.

3. **time_out** represents the waiting time before the sender retransmits the same packet if it does n acknowledgement for that packet.

Then, the code converts the text message to a bit sequence (variable **tx_bs**) and divides the bit se a list of packets (**sender_packet_list**) using the function **packetize**. In this lab, each packet conta character. As a result, the message 'Hello World!', which has 12 characters, will be converted to 12 Each packet consists three parts, namely, the sequence index of the packet (starting from 1), the t of packets in the list and the data (the 8 bit ASCII code for the character). Since each of these thre bits long, the packet is 24-bits long in total. After that, the code initializes the list of received packet (**receiver_packet_list**). This list will eventually contain every received packet in the order it was r

since the current implementation of the receiver does not handle duplicated and/or out of order pa

Then, the code initializes the transport layer using the function **start_transport** and initializes the s stop-and-wait protocol using the function **sender_stopwait**. The main loop of the simulation follow iteration of the while loop executes the operations of the sender, receiver and the underlying netwo time slot. These are implemented by the functions **sender_stopwait(), receiver_stopwait()**, and **step_transport()**, respectively.

1. The sender function, **sender_stopwait()**, checks whether the timeout has been reached and/o acknowedgement has been received. If so, it decides whether and which packet to send to the rece Otherwise, it does nothing. The sender function returns two values. The first one is a Boolean varia true when all packets of the list have been successfully transmitted (**done**). The second value tells number of packets that have been successfully transmitted and acknowledged so far (**noSentPacl**

2. The receiver function, **receiver_stopwait()**, sends an acknoweldgement (ACK) to the sender w receives a packet and adds the packet to the matrix **receiver_packet_list**. The content of the AC simply the index of the received message. Note that this function does not handle duplicated or out packets. You will handle that in Task 3.

3. The function **step_transport** simulates the transmission of packets through the network for one and increments the simulation time.

At the end of the loop, we convert the list of packets in the matrix **receiver_packet_list** into a text (**rx_msg**). If you run the code, you will observe that some characters are repeated because this fu not deal with duplicated packets that may occur in the stop and wait protocol. We will handle this in

Finally, we plot a figure to show the traffic in the network at the sender and receiver for the first 20( Green lines in the figure represent packets sent by the sender. Red lines represent the acknowled by the receiver. We can observe that some characters have to be sent several times due to the pa timeouts.

Your task is to revise the code to compute the throughput of the stop-and-wait protocol and save th inside the variable **throughput**. Note that the throughput is defined as the ratio between the numb successful transmission/acknowledgements and the number of time slots needed to complete those transmissions. It has units of packets per time slot. We have provided a function **getSimTime**, whic total simulation time. 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.

## Your Solution

Reset MATLAB Documentation (https://www.mathworks.

```
50
51 % % % % Revise the following code   % % % %
52 %RTT = length(rx_msg) / getSimTime();
53 %disp(RTT);
54
55 %throughput = RTT + ((p_loss/(1-p_loss) * time_out));
56 throughput = noSentPackets / getSimTime();
57 % % % % Do not change the code below % % % %
```

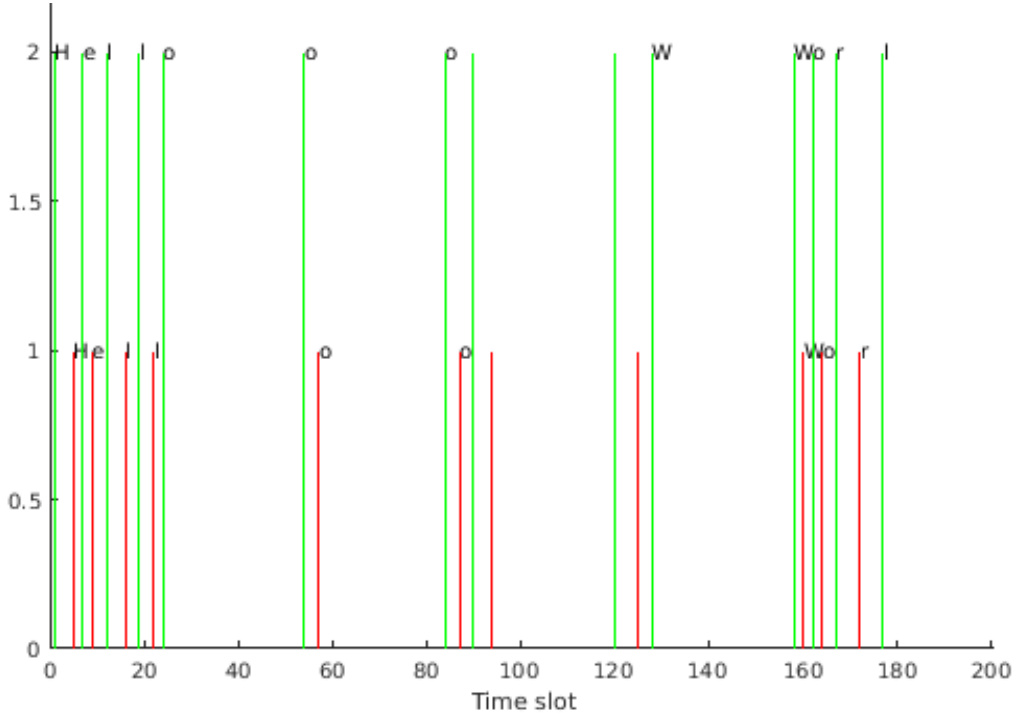
## Assessment Tests: Passed

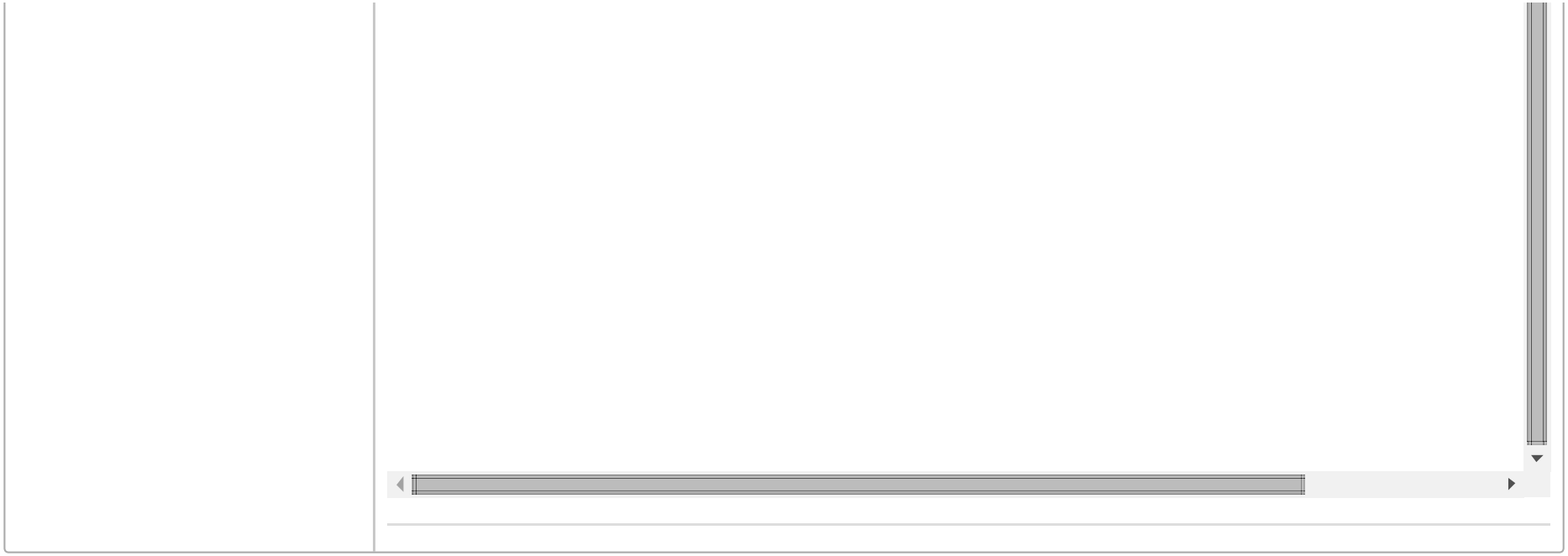
✔ Is the problem unchanged?

✔ Is the throughput correct?

## Output

```
Simulation Time: 259
Number of packets sent correctly: 12
The received message is: Helloo  Worlld!
Throughput: 0.046332 packets per slot
```


2
1.5
1
0.5
0
0
20
40
60
80
100
120
140
160
180
200
Time slot




Download on the
App Store


ANDROID APP ON
Google play