



HKUSTx: ELEC1200.3x A System View of Communications: From Signals to Packets (Part 3)



Bookmarks

- ▶ Pre-course Materials
- ▶ Topic 1: Course Overview
- ▶ Topic 2: The Link Layer
- ▶ Topic 3: The Network Layer
- ▶ Topic 4: Routing
- ▶ Topic 5: The Transport Layer
- ▼ **Topic 6: Reliable Transfer Protocols**


Topic 6: Reliable Transfer Protocols > 6.4 Lab 3: Transport Layer > Lab 3 - Task 2




Bookmark

LAB 3 - TASK 2 (EXTERNAL RESOURCE) (1.0 points possible)


6.1 Stop-and-Wait Protocol

Week 3 Quiz due Feb 15, 2016 at 15:30 UTC 


6.2 Throughput of Stop-and-Wait

Week 3 Quiz due Feb 15, 2016 at 15:30 UTC 

6.3 Sliding Window Protocol

Week 3 Quiz due Feb 15, 2016 at 15:30 UTC 

6.4 Lab 3: Transport Layer

Lab due Feb 15, 2016 at 15:30 UTC 

- MATLAB download and tutorials

Lab 3 - Task 2

In this task, you will learn about the process of encapsulation. The transport layer takes messages application layer, divides them into segments and adds header information to the segments. In this task, we are mainly concerned with the problem of reliable data transfer, so we will use a simplified header that contains information about the sequence number of the segment and the total number of segments that comprise the message. If we were concerned about multiplexing/demultiplexing, we would also include information about source/destination ports.

INSTRUCTION

The MATLAB code in the window below is similar to the code in Task 1, but you will implement the function `packetize()`. As shown in Task 1, this function converts the message bit sequence (`tx_bs`) into a list of packets.

In this case, the application wishes to send the text message 'PACKET', which is contained in the variable `tx_msg`. It first generates the bit sequence `tx_bs` from the text message `tx_msg` using the function `text2bitseq()`, which we studied in Part 1. Each character in the text message is represented using its ASCII code. Thus, if the text message has `n` characters, then `tx_bs` will be `n*8` bits long.

The function `packetize()` from Task 1 converts `tx_bs` to a list of `n` packets, where each packet contains one character from the message. In this task, your job is to implement this function.

Each packet contains a 16 bit header and 8 bits of data. Thus, each packet is 24 bits long. The 16 bit header consists of an 8 bit sequence number and an 8 bit number encoding the number of packets in the message. The sequence numbers run from 1 to `n`. The 8 bit data is taken from the bit sequence `tx_bs`. The first packet contains the first 8 bits of `tx_bs`, the second packet contains the second 8 bits, and so on. For example, if the first 8 bits of `tx_bs` encode the letter 'P' and there are `n = 6` characters in the text message, the list of packets will be

```
[0000 0001 0000 0110 0101 0000]
```

where extra spaces have been inserted in between sets of four bits to improve readability. The ASCII value of 'P' is 80 in decimal.

Your task is to create the list of packets for the given message and save it inside the variable

send_packet_list, which is an **n**-by-24 element matrix where each of the **n** rows contains one packet. The first row contains the first packet and so on. In the initial code, **send_packet_list** contains only one packet. The data is the ASCII code for 'P' in binary, and the message from the application layer was assumed to be one character.

In order to convert the index of the packet and the total number of packets into binary vector, you will use the function **dec2binvec**. This function takes as input a vector of **k** decimal numbers and returns a **k** by 8 matrix where each row is the 8 bit binary representation of one element of the input vector. Note that the size of the packet, the total number of packets in the list, should be the same for all packets.

The remainder of the code is the same as that in task 1. 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.com>)

```
54 display(['Numb. of packets sent correctly: ' num2str(noSentPackets)])
55 % todo: manage duplicated and/or unordered data
56 rx_msg = reconstruct_msg(receiver_packet_list)
57
58 % plot the traffic in the network
59 if noSentPackets>0
60     transmission_display();
61 end
```

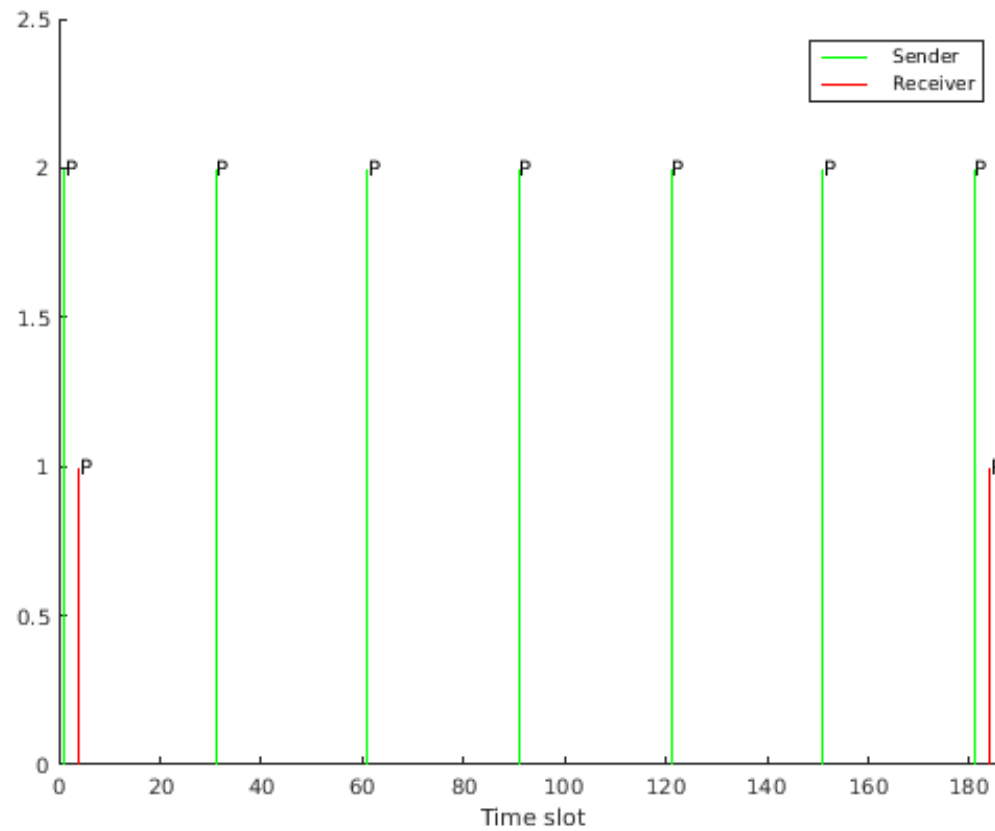
Output

Simulation Time: 187

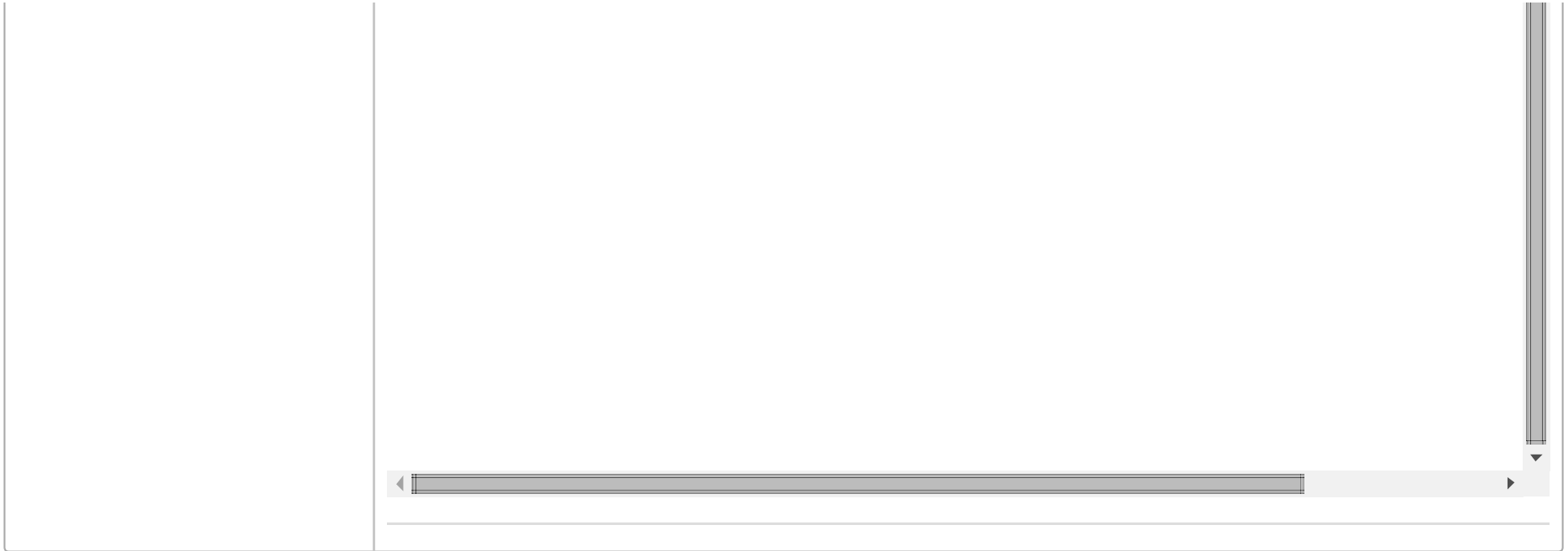
Numb. of packets sent correctly: 1

rx_msg =

PP



Assessment Tests



© All Rights Reserved



© edX Inc. All rights reserved except where noted. EdX, Open edX and the edX and Open EdX logos are registered trademarks or trademarks of edX Inc.

POWERED BY
OPENedX



