



## 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

Final Exam &gt; Final Exam &gt; Section 3: Transport Layer



Bookmark

## Section 3 Question 1

(4/4 points)

Consider a network running a reliable transport protocol where the receiver sends acknowledgements for packets it has received to the sender. Suppose that

1. Packets sent by the sender are 40 kilobytes long
2. The transmission rate of the slowest link between sender and receiver is 120 megabytes per second (in both directions)
3. The time to send and process acknowledgements is negligible.
4. The round trip time for a packet to be sent and the corresponding acknowledgement to be received by the sender is 4ms.
5. The retransmission time out is 5ms.

Give your answers to the problems below in units of packets per second to the nearest integer, e.g. 4000 packets/sec.

Suppose there is no packet loss. What is the maximum throughput of the system if the stop and wait protocol is used?




► Topic 7: The Application Layer

► Topic 8: Course Review

▼ Final Exam

**Final Exam**

Final Exam due Feb 22, 2016  
at 15:30 UTC 

► MATLAB download and tutorials

250

Suppose that there is no packet loss. What is the throughput of the system if the sliding window protocol is used and the size of the sliding window is 8?

2000



2000

Suppose that there is no packet loss. What is the maximum throughput of the system if the sliding protocol is use and you are free to choose the sliding window size?

3000



3000

Suppose that the round trip loss probability is 0.05. What is the throughput of the system if the stop and wait protocol is used?

235



235

*You have used 1 of 1 submissions*

Section 3 Question 2

(2/2 points)

Consider a network running the stop and wait protocol. Suppose that the sender wants to send three packets, labelled P1, P2, and P3. If the packets are received by the receiver, then the receiver sends corresponding acknowledgements, labelled as A1, A2 and A3. Suppose that the sender and receiver have correct implementations of the stop and wait protocol. Suppose also that no packets/acknowledgements are lost in the network, except for the first sending of P2 and the first sending of acknowledgements A2 and A3.

What is the sequence of packets/acknowledgements sent/received by the sender

- ☐ P1,A1,P2,P2,A2,P3,P3,A3
- ☒ P1,A1,P2,P2,P2,A2,P3,P3,A3 ✓
- ☐ P1,A1,P2,A2,P2,A2,P3,A3,P3,A3
- ☐ P1,A1,P2,A2,P3,A3

What is the sequence of packets/acknowledgements received/sent by the receiver

- ☐ P1,A1,P2,A2,P2,A2,P2,A2,P3,A3,P3,A3
- ☐ P1,A1,P2,A2,P3,A3

☐ P1,A1,P2,P2,P2,A2,P3,A3,P3,A3

☒ P1,A1,P2,A2,P2,A2,P3,A3,P3,A3 ✓

*You have used 1 of 1 submissions*

## MATLAB INSTRUCTIONS

The MATLAB code in the window below is similar to the code of Lab 3 - Task 4. But, instead of the stop and wait protocol, you have to implement sender side of the sliding window protocol. Note that, with the sliding window protocol, the sender will send several packets without waiting for an ACK signal. The list of unacknowledged packets is known as the window, and the maximum number of packets that can be sent without an acknowledgement is known as the window size.

Once the sender has sent out the maximum number of packets it is allowed to, it waits either for an acknowledgement, or for a time-out. If an acknowledgement is received and it corresponds to one of the packets in the window, then the sender removes that packet from the window and is free to send another packet. If the acknowledgement does not correspond to one of the packets in the window, the sender ignores the acknowledgement. Since packets in the window may have been sent at a different times, the sender maintains a list of times that each packet was sent. If the time since a packet

was sent exceeds a retransmission timeout period, then the sender re-sends the packet, and leaves it inside the window. It must also update its record of when that packet was sent.

The code below contains an incorrect implementation of the sliding window protocol. In particular, there is an error in how it handles acknowledgements. Your task is to correct this error.

The window size is contained in the variable **win\_size**, which is set equal to 4.

We maintain a list of unacknowledged packets as a row vector called **window**, which may have variable size, depending upon the number of unacknowledged packets. However, the maximum size of the vector **window** should be **win\_size**. Each element of **window** contains the sequence number of a packet that was sent into the network but is still unacknowledged. This vector is initialized to be empty.

We also maintain the list of times that the packets were sent as a row vector called **time\_sent**. This vector should have exactly the same size as **window**, since corresponding positions in the vectors correspond to the same packet. Each element of **time\_sent** contains the time that the packet was sent, i.e. **time\_sent(k)** contains the last time that packet **window(k)** was sent, where  $k$  can run from **1** to **win\_size**. This vector is initialized to be empty.

We also maintain the sequence number of the last packet sent in the scalar variable **last\_sent**. The current time in the simulation is maintained by the integer variable **t**. These are both initialized to zero.

Each iteration of the sender proceeds as follows:

The sender first checks whether it has received an acknowledgement. If it has and the corresponding packet is in the window, it should remove the packet from the window. This means that the length of the variable **window** in the code will decrease. Remember also that **time\_sent** and **window** should have the same time and that corresponding positions refer to the same packet. In the code, the variable **ind** refers to the position of the acknowledged packet inside the sliding window (**ind** varies from 1 to **win\_size**). The variable **ind** is empty if the acknowledged packet is not in the sliding window. You can remove an element of a vector by setting that element equal to the empty matrix.

The sender then checks whether the retransmission time out has expired for any of the packets, if it has, then it resends the packet and resets the entry of the vector **time\_sent** for that packet. If no time-outs have expired, then the code checks to see whether the number of packets in the sliding window is less than **win\_size**. If so, it sends the next packet that it is waiting to be sent (**last\_sent+1**) and adds that information to **window** and **time\_sent**. In our slotted implementation of the transport layer, only one packet can be sent in a time slot.

The remaining code executes the receiver and makes one step through the transport layer simulation. The simulation terminates after all packets have been sent and acknowledged, or the maximum number of iterations has expired. The code then plots the activity in the network, similar to the way seen in Lab 3 - Task 4.

In the initial code, the acknowledgement is not handled correctly, so the sender keeps resending the same four packets after the timeout. To enable you to visualize the system operation better, we have assumed no packet loss and constant delay. If you have

interest, you might try changing these and the window size, but only change these parameters after you have scored the points, as the grader assumes that you are using the values of **p\_loss**, **d\_min** and **d\_max** given in the initial code (0, 5 and 5).

Please, revise the the code between the lines:

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

and

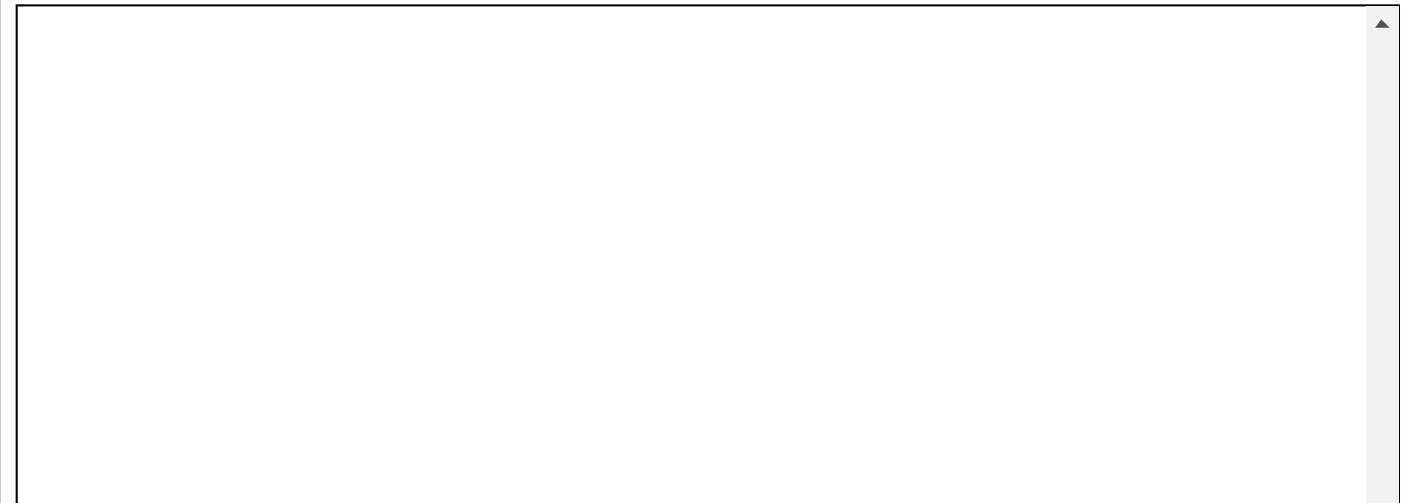
% % % % Do not change the code below % % % %

Do not change other parts of the code.

---

### Section 3 MATLAB Question

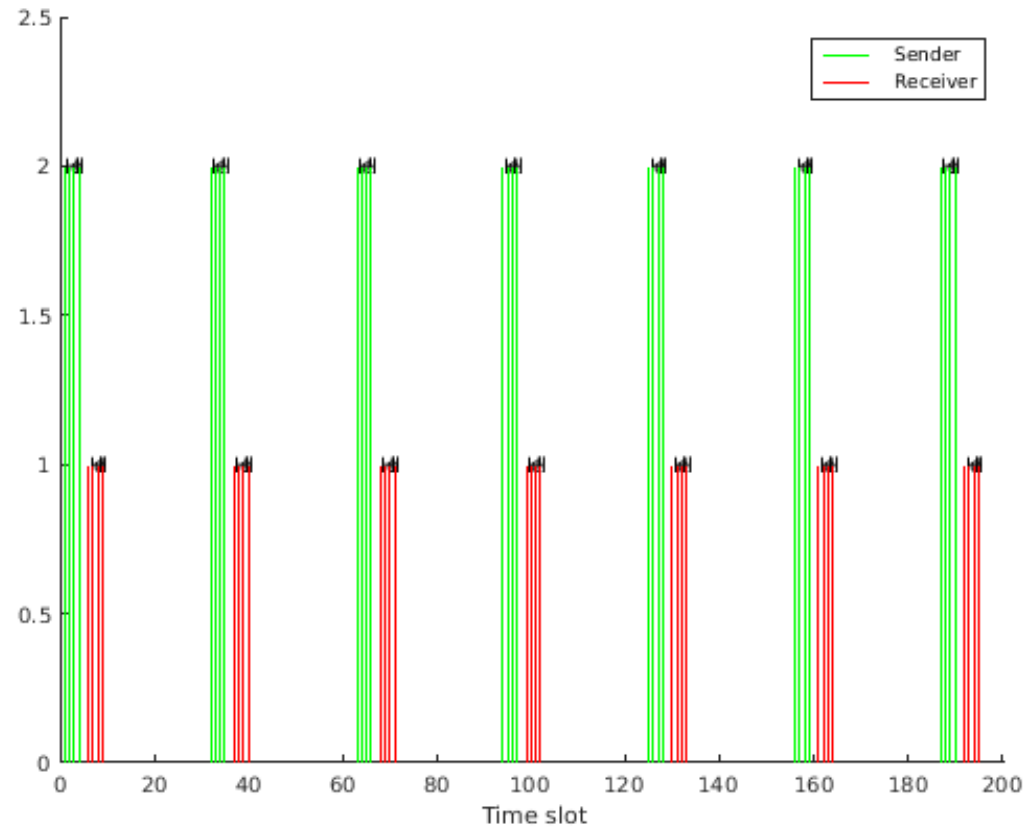
(2 points possible)



```
75 end
76
77 % execute one iteration of receiver
78 receiver_packet_list = receiver_stopwait(receiver_packet_list);
```

Unanswered

Figure 1



Maximum number of transport simulation steps (2550) reached.

rx\_msg =



Run Code

© All Rights Reserved

