Cairo University **Computer Networks**

Faculty of Engineering **Fall 2023**

Term Project
Data Link Layer Protocols Simulation

In this project, you will develop, simulate and test data link layer protocols between two nodes that are connected with a noisy channel, where the transmission is not error-free, packets may get corrupted, duplicated, delayed, or lost, and the buffers are of limited sizes.

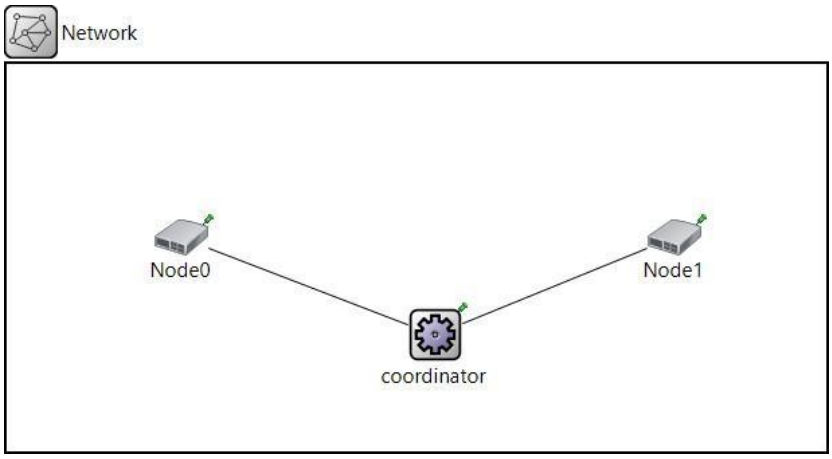

Figure 1: The design of the system's network

The system network topology is shown in figure1. It consists of one pair of nodes [Node0, Node1], and one coordinator that is connected to the pair.

In this project, the pair of nodes would communicate and exchange messages using the **Selective Repeat** algorithm with noisy channel and sender window of size WS, receiver window of size WR=3(default), using **Byte Stuffing** as a framing algorithm, and **Checksum** as an error detection algorithm.

System inputs

1. Each node has a list of messages to send, and each node reads its list of messages from a different input text file; namely 'input0.txt' for Node0 and 'input1.txt' for Node1.
2. Each message starts in a new line, and there is a 4-bits binary prefix before each message. These 4-bits represent the possibility of [*Modification, Loss, Duplication, Delay*] that would affect this message. For example, "1010 Data Link" means that the message "Data Link" will have a modification to one of its bits while sending, and will be sent twice. Figure2 includes an example of the input file.
3. Error will be on the message itself, not the other control bits (Checksum, stuffed bytes, flags, etc.) therefore **apply Checksum on the original message** then apply framing on the message payload only.

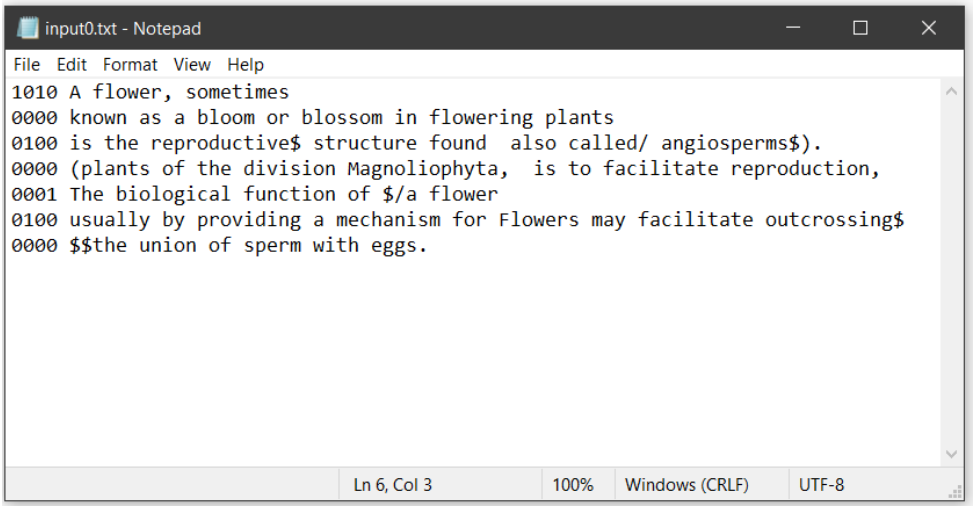

Figure 2: Example of the input file

4. Errors

Errors are put in a binary format (4 bits), where each bit location indicates the presence of a certain error type.

5. Types of errors:

- No error ☐☐☐☐
- Delay ☐☐☐☑
- Duplication ☐☐☑☐
- Loss ☐☑☐☐
- Modification ☑☐☐☐
- Mixed error any 2 or more error indicators mixed with each other.

6. Table 1 contains the details of the errors and their priorities **(Check the delays in the system for better understanding):**

Error code [*Modification, Loss, Duplication, Delay]*	Effect
0000	No error
0001	Delay For example, using the default delays (listed below in the section “Delays in the system”): @t=0 read line @t=0.5 end processing time @t=5.5 the message is received.

0010	Duplication For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=1.5 version 1 of the message is received. @t=1.6 version 2 of the message is received.
0011	Make two versions of the message and add to their sending time the delay error. For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=5.5 version 1 of the message is received. @t=5.6 version 2 of the message is received.
0100	Loss For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=0.5 read the next line.
0101	Loss For example, using the default delays:

	@t=0 read line @t=0.5 end processing time @t=0.5 read the next line.
0110	Loss of the both messages For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=0.5 read the next line.
0111	Loss of both messages For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=0.5 read the next line.
1000	Modification For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=1.5 the modified message is received.
1001	Modification and delay For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=5.5 the modified message is received.

1010	Modification and Duplication For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=1.5 version 1 of the modified message is received. @t=1.6 version 2 of the modified message is received.
1011	Modification and Duplication and Delay For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=5.5 version 1 of the modified message is received. @t=5.6 version 2 of the modified message is received.
1100	Loss For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=0.5 read the next line.
1101	Loss For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=0.5 read the next line.
1110	Loss for both messages For example, using the default delays:
	@t=0 read line @t=0.5 end processing time @t=0.5 read the next line.
1111	Loss for both messages For example, using the default delays: @t=0 read line @t=0.5 end processing time @t=0.5 read the next line.

Table 1: Error codes and their meanings

7. Coordinator

The coordinator starts working in the initialization stage, its main job is to assign choose which node of the pair should start, and when to start in seconds. The coordinator gets this information from an input file “coordinator.txt”.
It will contain one line contains [“*Node_id” “starting_time”*] for the starting node, and *Node_id=[0,1].*

After the coordinator sends the initialization messages, the starting node should start reading its messages from its file on the **specified starting time**, and the receiver will respond back as will be described later. The messages between the peer don’t pass through the coordinator.

Although at any session, only one file containing the messages is read and the other will not be used, we keep both files in case the coordinator would choose any one of the nodes to start freely in later sessions or runs of the program.

8. Parameters

Parameters that are set in the .ini file:

- The sender window of size **WS**, by default=3
- The receiver window of size **WR**, by default=3
- The sender's timeout interval **TO** in seconds for the Selective repeat protocol, by default=10
- The sender's and receivers' processing time **PT** for each frame, by default=0.5
- The channel's transmission delay **TD** for any frame, by default=1.0
- The channel's error delay **ED** for any frame, by default=4.0
- The channel's duplication delay **DD** before sending the second version, by default=0.1
- ACK/NACK frame (**No Errors will occur to these frames)**.

Delays in the system

We have several delays in the system as follows.

1. Processing delay, and this happens at both pairs for any frame to be sent, and it takes a delay = **PT** as set from the .ini file with default **value 0.5**
2. Transmission delay, and this happens at both ways of the channel while sending any frame, and it takes a delay = **TD** as set from the .ini file with default **value 1**
3. Error delay, and this happens at if an error delay to be introduced to any frame, and it takes a delay = **ED,** as a result, the frame will have a total sending delay = **TD + ED**. This is also set from the .ini file. with default **value 4** Note that the delayed message can cause out of order delivery of messages. I.e., delayed messages don't lock the sender and prevent him from send the next frame on time. The delay happens in the transmission channel and should not stop the sender from continuing.

4. Duplication Delay **DD** for any Duplicated message as set from the .ini file with default **value 0.1**, the first version is sent as usual using the **PT,** and then, the second version is sent after the **DD** + **PT**.

System outputs

- The system should print one log file named [**output.txt**], containing the details for each message transmission from both nodes using the following format:

1.	Upon reading each line in the sender, print the following:
	At time [.. starting processing time…..], Node[id] , Introducing channel error with code =[…code in 4 bits…] .
2.	Before transmission of any data frame, print the following (even if the message will be lost later):
	At time [.. starting sending time after processing…..], Node[id] [sent/received] frame with seq_num=[..] and payload=[….. in characters after modification…..] and trailer=[…….in bits…..] , Modified [-1 for no modification, otherwise the modified bit number] , Lost [Yes/No], Duplicate [0 for none, 1 for the first version, 2 for the second version], Delay [0 for no delay , otherwise the error delay interval].
3.	For any time-out event:
	Time out event at time [.. timer off-time…..], at Node[id] for frame with seq_num=[..] *Then, upon sending all the messages in the sender window again, print a log line for each message as indicated in 2. above.*
4.	For any control frame sending:
	At time[.. starting sending time after processing…..], Node[id] Sending [ACK/NACK] with number […] , loss [Yes/No]

- Print the details for each message transmission from both nodes to the simulation console. “I.e., the same upper messages in the log file”.

Messages

Every message contains the following fields:

- Header: Data sequence number.
- Frame type: Data=0/ACK=1 /NACK=2.
- ACK/NACK number.
- Payload: the message contents after byte stuffing (in char, including flags).
- Trailer: Checksum.

You can choose the data types as you like, but when printing, use the format described in the table above in the “System outputs” section.

Framing

- Is done using the byte stuffing algorithm with starting and ending Flags of ‘#’ and the Esc is ‘/’.
- The framing is applied to the message payload only.
- There is no specific maximum transmission size for the frame, each message is represented with one line in the input file.

Error detection

- This is done using Checksum.
- checks for the payload before applying the byte stuffing.
- Checksum bits are added after the stuffed message.
- Checksum bits are 8 bits.
- A single word size is 8 bits
- The receiver Checks the checksum himself to detect if there is/isn’t any single bit error during the transmission and so decides to send ACK/NACK.
- The ACK numbers are set as the sequence number of the next expected frame.
- The NACK numbers are set as the sequence number of the erroneous frame.
- No error correction is needed in this project

The Selective Repeat protocol

- Implement the Selective Repeat protocol taking into consideration the time-out and the window size with the noisy channel.
- The coordinator reads the initialization from the ‘coordinator.txt’ file and sends the information to the pair.
- The starting node reads the input file and starts sending at the specific given time, and takes into account the four types of errors. It will send the frames within its sending window size WS with PT between frames as its processing speed.
- the sender sends data and receiver responds with control frame; “ACK” if no error and with “NACK” if there is an error, ACK is carrying the sequence number of the next frame, NACK carries the sequence number of the erroneous frame.
- Sender advances its window if it received a correct ACK at the beginning of the window, otherwise waits for this ACK.
- There is a timer on the sender side, when the timer goes off, the sender assumes that the packet got lost and retransmits the unacknowledged packet in the window (Note: the timer restarts after sliding the window).
- Each Message, whether it is Data or Control message, carries a sequence number.
- The receiver always responds for any received packet with control message (and no payload).(NOTE: You can implement a control message with no payload, via inheriting from cMessage)
- The Simulation ends when the sender node finishes sending all the messages in its input file and receives all the ACKs regarding these messages.
- There are no accumulative ACKs in this project, send a separate ACK/NACK for every frame.
- If any detail is missing from the project document about the protocol, you should use the algorithm given in the lecture as your reference, assumptions should be communicated

via classroom.

- You can use the following link for more visualization about the Selective Repeat algorithm.
- print the log file at the end of the session.

Delivery and discussion time: Wednesday of week 13

Number of team members: Maximum of four