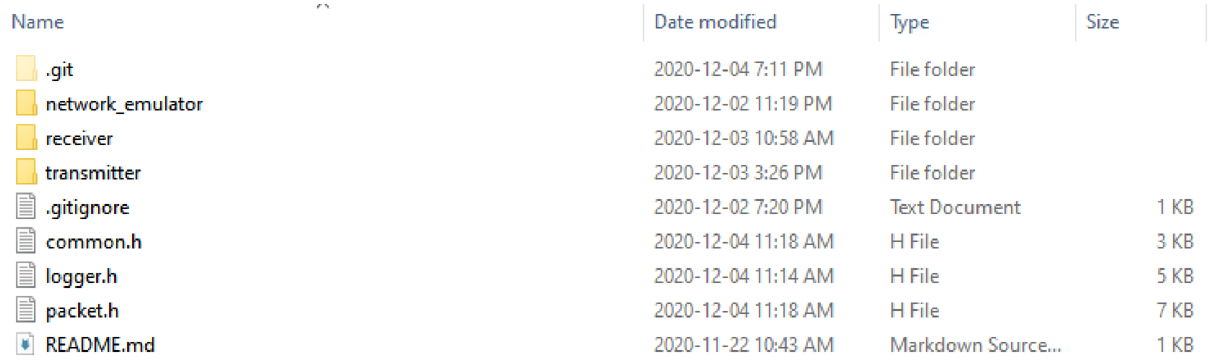
Final Project User Manual

Background

1. The transmitter and receiver were compiled on Fedora, Manjaro, and WSL environments
    a. gcc required to compiled
2. QT creator was used to compile DEBUG and RELEASE versions of the emulator on Windows
    a. QTCharts installation required
3. You should have the following directory structure once you clone the git repo

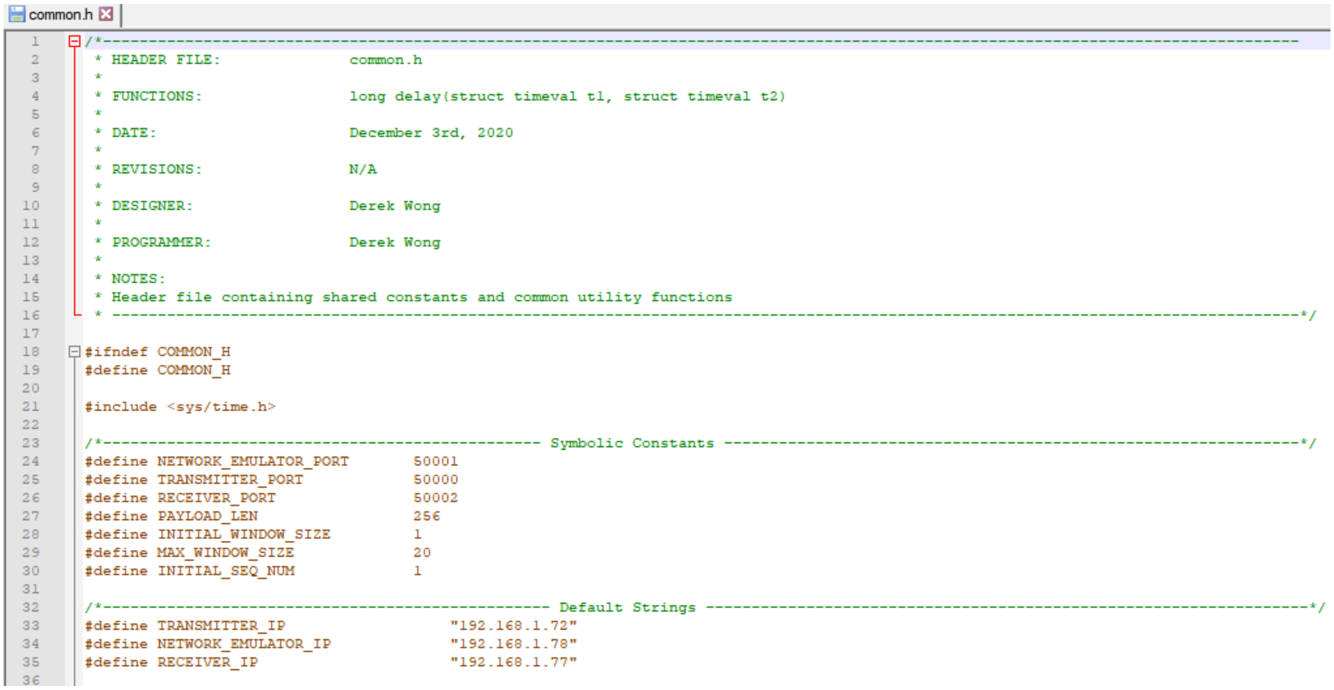


Name	Date modified	Type	Size
.git	2020-12-04 7:11 PM	File folder	
network_emulator	2020-12-02 11:19 PM	File folder	
receiver	2020-12-03 10:58 AM	File folder	
transmitter	2020-12-03 3:26 PM	File folder	
.gitignore	2020-12-02 7:20 PM	Text Document	1 KB
common.h	2020-12-04 11:18 AM	H File	3 KB
logger.h	2020-12-04 11:14 AM	H File	5 KB
packet.h	2020-12-04 11:18 AM	H File	7 KB
README.md	2020-11-22 10:43 AM	Markdown Source...	1 KB

Getting Started

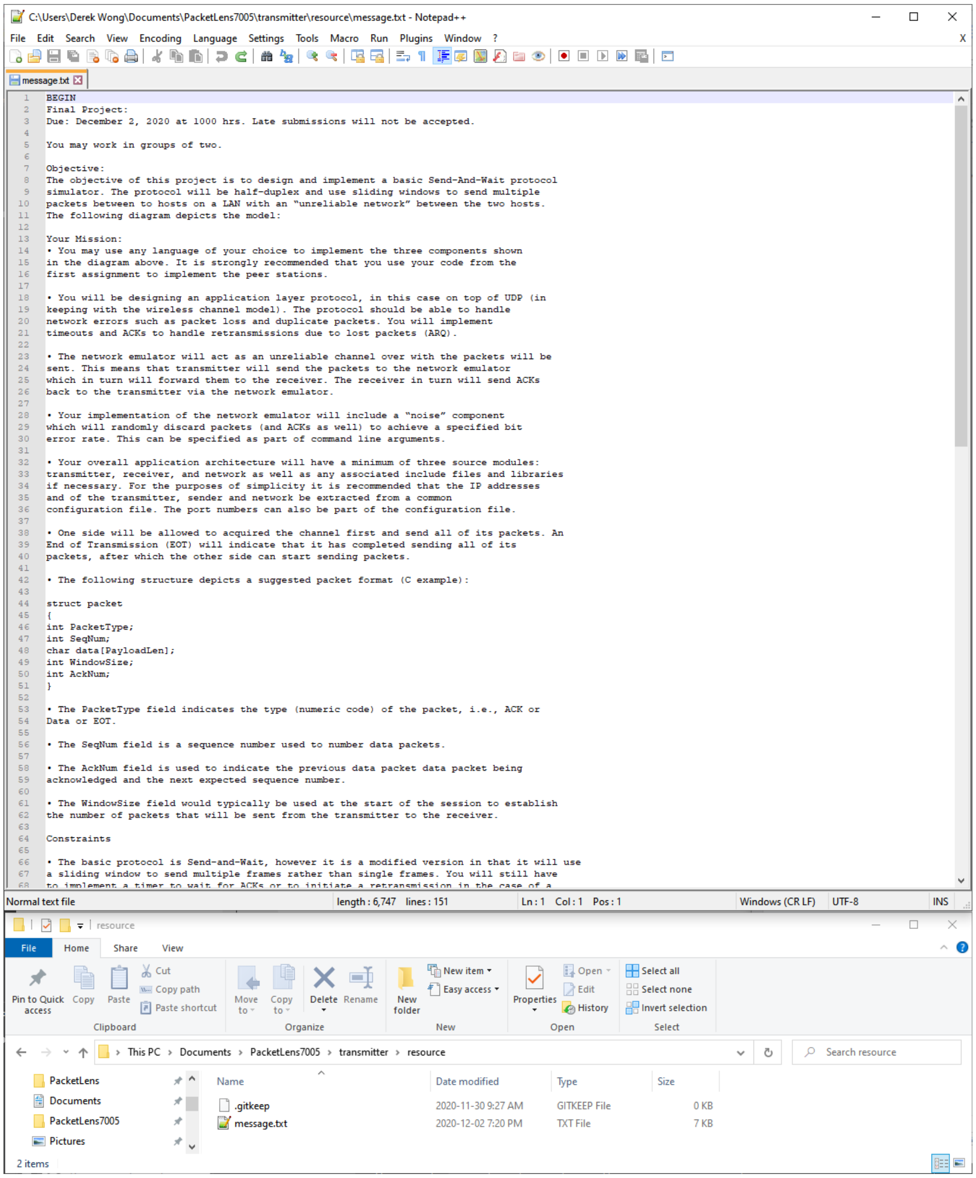
To get the transmitter, network emulator and receiver applications working, do the following:

1. Update **TRANSMITTER_IP** , **NETWORK_EMULATOR_IP**, **RECEIVER_IP** constants in common.h file to match the IP of the machine each application is running on.
2. (Optional) Update **TRANSMITTER_PORT**, **NETWORK_EMULATOR_PORT**, **RECEIVER_PORT** constants to change the port each application will use to send/receive datagrams



```
/*------------------------------------------------------------------------------------------------------------------------------
 * HEADER FILE:                common.h
 *
 * FUNCTIONS:                  long delay(struct timeval t1, struct timeval t2)
 *
 * DATE:                       December 3rd, 2020
 *
 * REVISIONS:                  N/A
 *
 * DESIGNER:                   Derek Wong
 *
 * PROGRAMMER:                 Derek Wong
 *
 * NOTES:
 * Header file containing shared constants and common utility functions
 * ------------------------------------------------------------------------------------------------------------------------------*/

#ifndef COMMON_H
#define COMMON_H

#include <sys/time.h>

/*-------------------------------------------------- Symbolic Constants ---------------------------------------------------------------*/
#define NETWORK_EMULATOR_PORT       50001
#define TRANSMITTER_PORT            50000
#define RECEIVER_PORT               50002
#define PAYLOAD_LEN                 256
#define INITIAL_WINDOW_SIZE         1
#define MAX_WINDOW_SIZE             20
#define INITIAL_SEQ_NUM             1

/*--------------------------------------------------- Default Strings ----------------------------------------------------------------*/
#define TRANSMITTER_IP                  "192.168.1.72"
#define NETWORK_EMULATOR_IP             "192.168.1.78"
#define RECEIVER_IP                     "192.168.1.77"

```

3. Upload a file name **message.txt** into the **../transmitter/resource** directory



C:\Users\Derek Wong\Documents\PacketLens7005\transmitter\resource\message.txt - Notepad++

```
BEGIN
Final Project:
Due: December 2, 2020 at 1000 hrs. Late submissions will not be accepted.

You may work in groups of two.

Objective:
The objective of this project is to design and implement a basic Send-And-Wait protocol
simulator. The protocol will be half-duplex and use sliding windows to send multiple
packets between to hosts on a LAN with an “unreliable network” between the two hosts.
The following diagram depicts the model:

Your Mission:
• You may use any language of your choice to implement the three components shown
in the diagram above. It is strongly recommended that you use your code from the
first assignment to implement the peer stations.

• You will be designing an application layer protocol, in this case on top of UDP (in
keeping with the wireless channel model). The protocol should be able to handle
network errors such as packet loss and duplicate packets. You will implement
timeouts and ACKs to handle retransmissions due to lost packets (ARQ).

• The network emulator will act as an unreliable channel over with the packets will be
sent. This means that transmitter will send the packets to the network emulator
which in turn will forward them to the receiver. The receiver in turn will send ACKs
back to the transmitter via the network emulator.

• Your implementation of the network emulator will include a “noise” component
which will randomly discard packets (and ACKs as well) to achieve a specified bit
error rate. This can be specified as part of command line arguments.

• Your overall application architecture will have a minimum of three source modules:
transmitter, receiver, and network as well as any associated include files and libraries
if necessary. For the purposes of simplicity it is recommended that the IP addresses
and of the transmitter, sender and network be extracted from a common
configuration file. The port numbers can also be part of the configuration file.

• One side will be allowed to acquired the channel first and send all of its packets. An
End of Transmission (EOT) will indicate that it has completed sending all of its
packets, after which the other side can start sending packets.

• The following structure depicts a suggested packet format (C example):

struct packet
{
int PacketType;
int SeqNum;
char data[PayloadLen];
int WindowSize;
int AckNum;
}

• The PacketType field indicates the type (numeric code) of the packet, i.e., ACK or
Data or EOT.

• The SeqNum field is a sequence number used to number data packets.

• The AckNum field is used to indicate the previous data packet data packet being
acknowledged and the next expected sequence number.

• The WindowSize field would typically be used at the start of the session to establish
the number of packets that will be sent from the transmitter to the receiver.

Constraints

• The basic protocol is Send-and-Wait, however it is a modified version in that it will use
a sliding window to send multiple frames rather than single frames. You will still have
to implement a timer to wait for ACKs or to initiate a retransmission in the case of a
```

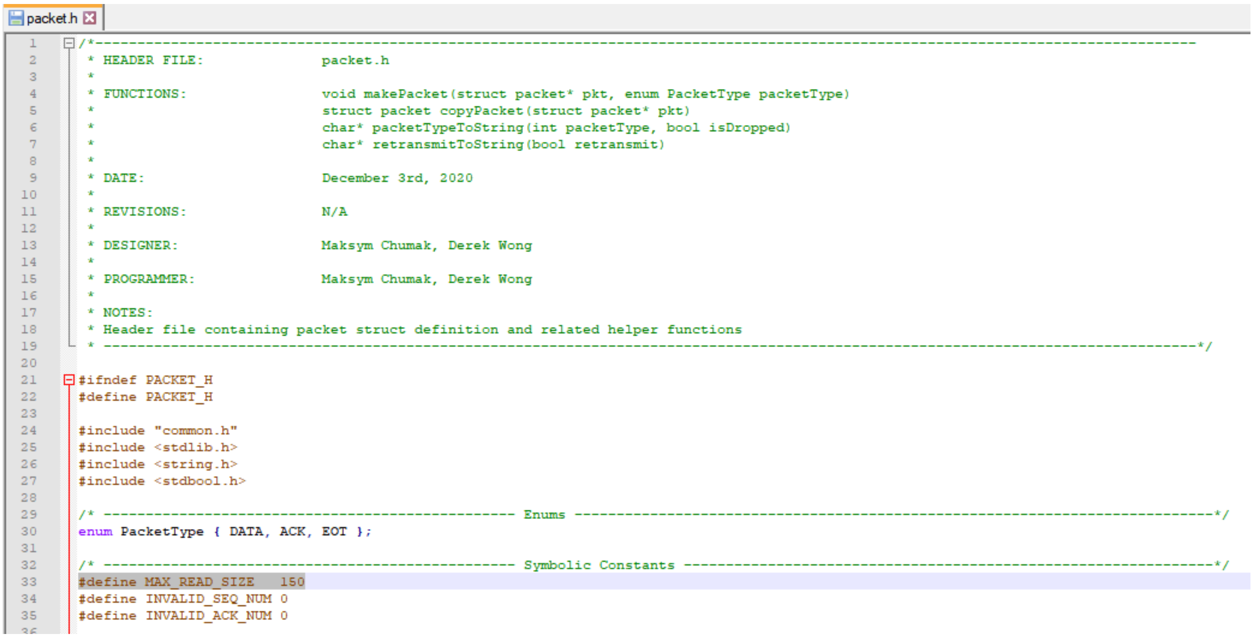
Normal text file | length : 6,747 lines : 151 | Ln : 1 Col : 1 Pos : 1 | Windows (CR LF) | UTF-8 | INS

This PC › Documents › PacketLens7005 › transmitter › resource

Name	Date modified	Type	Size
.gitkeep	2020-11-30 9:27 AM	GITKEEP File	0 KB
message.txt	2020-12-02 7:20 PM	TXT File	7 KB

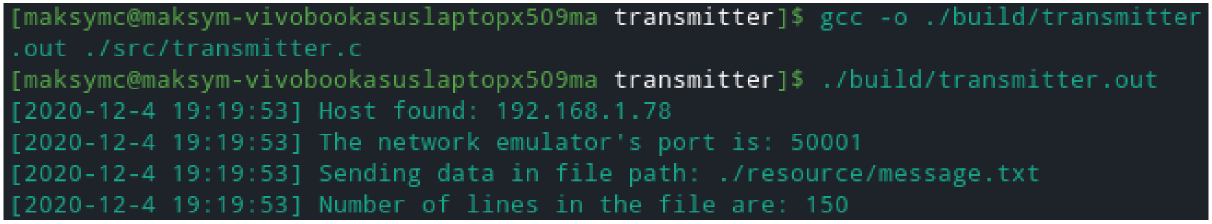
2 items

4. Ensure MAX_READ_SIZE in **packet.h** matches the number of lines in the **message.txt** file if you want the entire file contents to be transferred



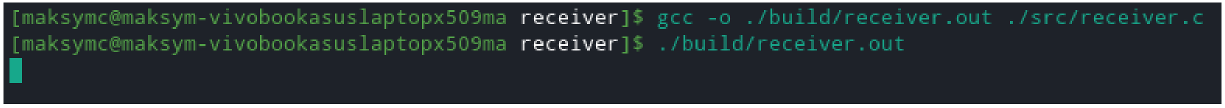
```
/*---------------------------------------------------------------------------------------------------------------------------
 * HEADER FILE:              packet.h
 *
 * FUNCTIONS:                void makePacket(struct packet* pkt, enum PacketType packetType)
 *                           struct packet copyPacket(struct packet* pkt)
 *                           char* packetTypeToString(int packetType, bool isDropped)
 *                           char* retransmitToString(bool retransmit)
 *
 * DATE:                     December 3rd, 2020
 *
 * REVISIONS:                N/A
 *
 * DESIGNER:                 Maksym Chumak, Derek Wong
 *
 * PROGRAMMER:               Maksym Chumak, Derek Wong
 *
 * NOTES:
 * Header file containing packet struct definition and related helper functions
 * -------------------------------------------------------------------------------------------------------------------------*/

#ifndef PACKET_H
#define PACKET_H

#include "common.h"
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

/* ---------------------------------------------- Enums -------------------------------------------------------------------*/
enum PacketType { DATA, ACK, EOT };

/* ---------------------------------------------- Symbolic Constants ------------------------------------------------------*/
#define MAX_READ_SIZE   150
#define INVALID_SEQ_NUM 0
#define INVALID_ACK_NUM 0
```

5. Navigate to the sub directory specific for transmitter and receiver and execute:
    a. **gcc -o ./build/<application_name>.out ./src/<application_name>.c**
    b. Transmitter



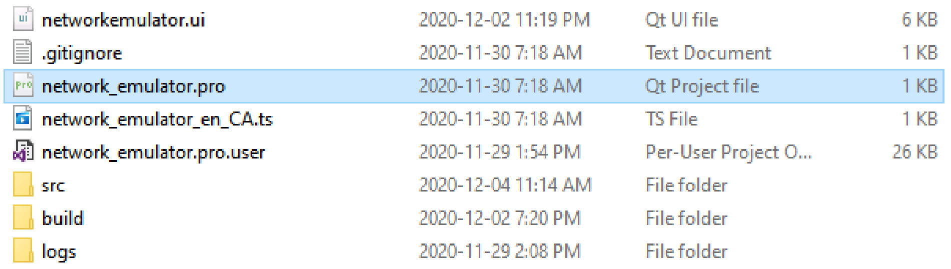
```
[maksymc@maksym-vivobookasuslaptopx509ma transmitter]$ gcc -o ./build/transmitter
.out ./src/transmitter.c
[maksymc@maksym-vivobookasuslaptopx509ma transmitter]$ ./build/transmitter.out
[2020-12-4 19:19:53] Host found: 192.168.1.78
[2020-12-4 19:19:53] The network emulator's port is: 50001
[2020-12-4 19:19:53] Sending data in file path: ./resource/message.txt
[2020-12-4 19:19:53] Number of lines in the file are: 150
```

    c. Receiver



```
[maksymc@maksym-vivobookasuslaptopx509ma receiver]$ gcc -o ./build/receiver.out ./src/receiver.c
[maksymc@maksym-vivobookasuslaptopx509ma receiver]$ ./build/receiver.out
```

6. Logs can be found in **../<application_name>/logs/out.log**
7. Navigate to ../network_emulator/ directory and double click on **network_emulator.pro**

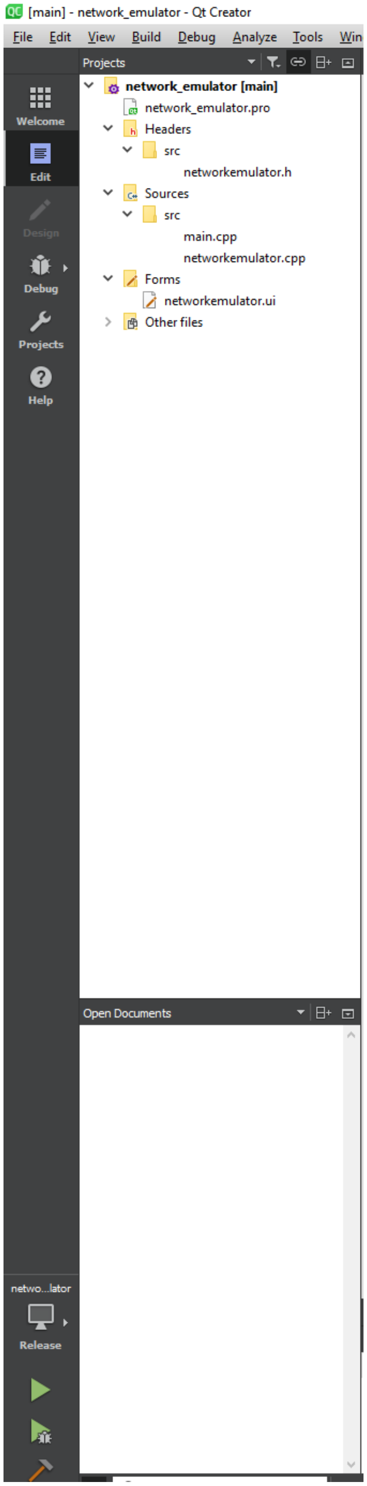


networkemulator.ui	2020-12-02 11:19 PM	Qt UI file	6 KB
.gitignore	2020-11-30 7:18 AM	Text Document	1 KB
network_emulator.pro	2020-11-30 7:18 AM	Qt Project file	1 KB
network_emulator_en_CA.ts	2020-11-30 7:18 AM	TS File	1 KB
network_emulator.pro.user	2020-11-29 1:54 PM	Per-User Project O...	26 KB
src	2020-12-04 11:14 AM	File folder	
build	2020-12-02 7:20 PM	File folder	
logs	2020-11-29 2:08 PM	File folder	

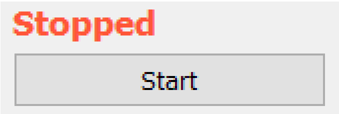
    a.

8. Click on the start button to build and run the application

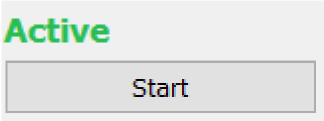
a.



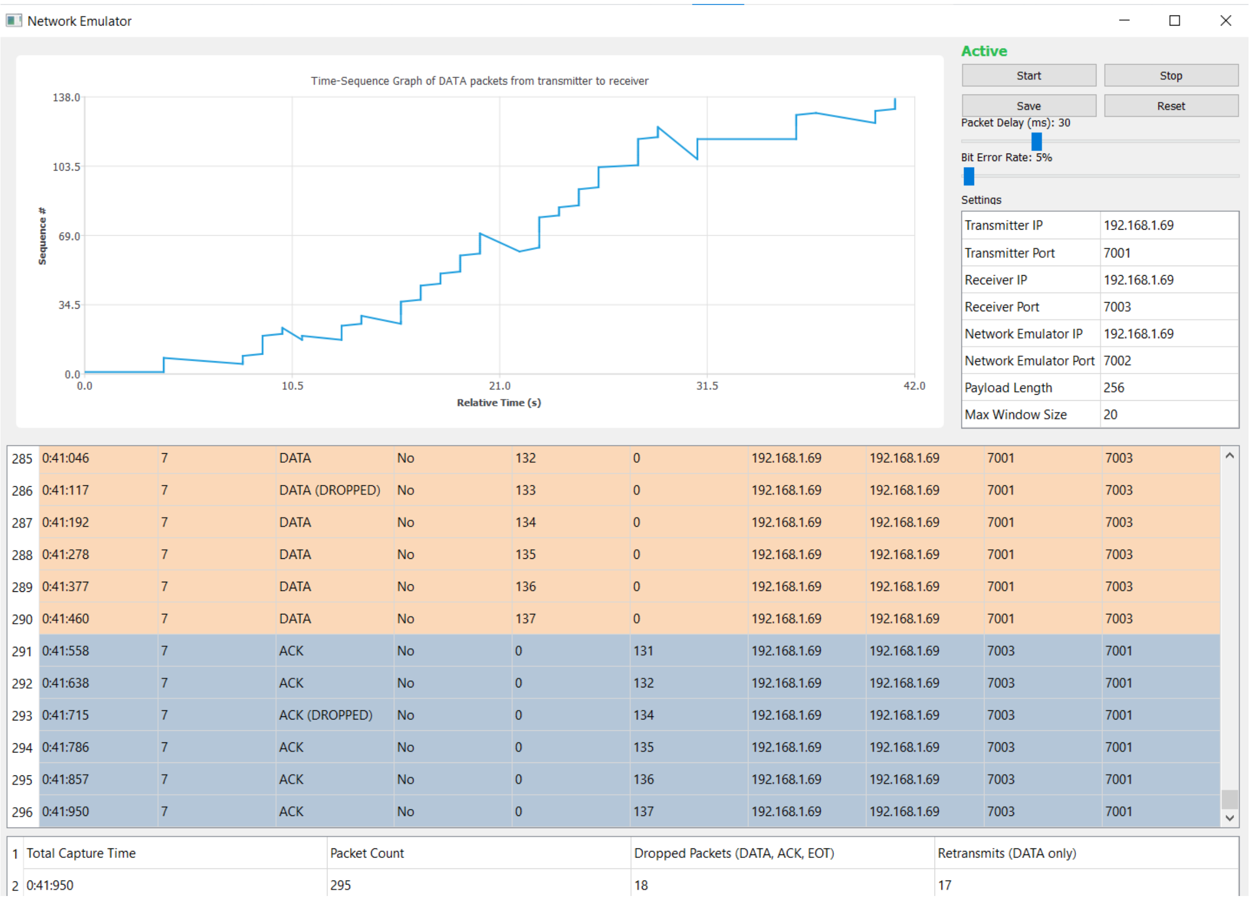
9. Once transmitter, network emulator and receiver applications are started:
   a. Click the “Start” button to launch the application, the status label will change from red “Stopped” to green “Active”



----------------------->



   b. On packet arrival the Packet Table, Summary Table and Time Sequence Graph will start populating



Packet Delay (ms): 30

Bit Error Rate: 5%

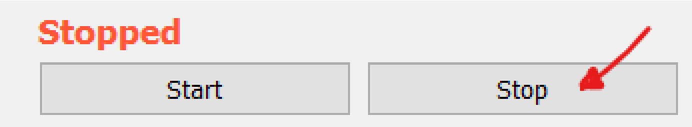
Settings

Transmitter IP	192.168.1.69
Transmitter Port	7001
Receiver IP	192.168.1.69
Receiver Port	7003
Network Emulator IP	192.168.1.69
Network Emulator Port	7002
Payload Length	256
Max Window Size	20

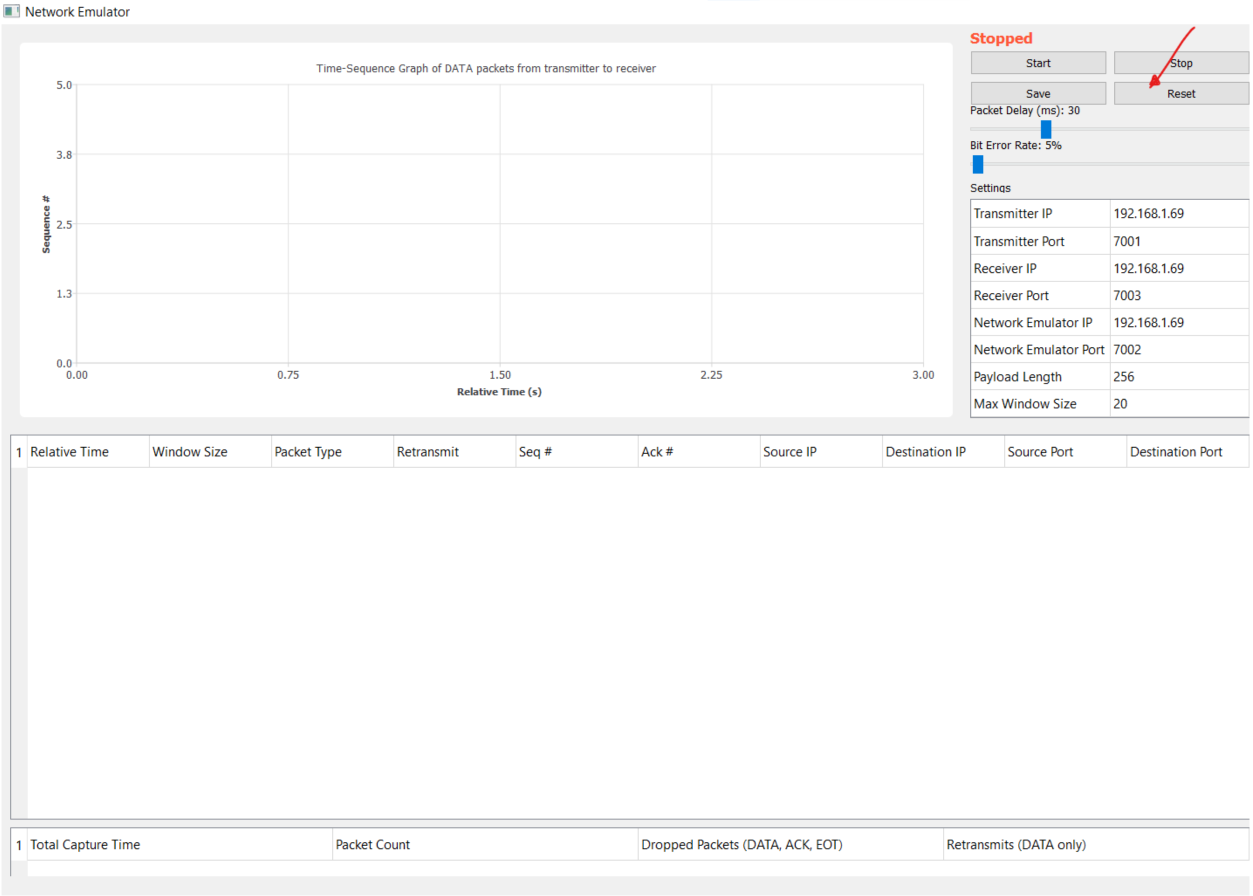
285	0:41:046	7	DATA	No	132	0	192.168.1.69	192.168.1.69	7001	7003
286	0:41:117	7	DATA (DROPPED)	No	133	0	192.168.1.69	192.168.1.69	7001	7003
287	0:41:192	7	DATA	No	134	0	192.168.1.69	192.168.1.69	7001	7003
288	0:41:278	7	DATA	No	135	0	192.168.1.69	192.168.1.69	7001	7003
289	0:41:377	7	DATA	No	136	0	192.168.1.69	192.168.1.69	7001	7003
290	0:41:460	7	DATA	No	137	0	192.168.1.69	192.168.1.69	7001	7003
291	0:41:558	7	ACK	No	0	131	192.168.1.69	192.168.1.69	7003	7001
292	0:41:638	7	ACK	No	0	132	192.168.1.69	192.168.1.69	7003	7001
293	0:41:715	7	ACK (DROPPED)	No	0	134	192.168.1.69	192.168.1.69	7003	7001
294	0:41:786	7	ACK	No	0	135	192.168.1.69	192.168.1.69	7003	7001
295	0:41:857	7	ACK	No	0	136	192.168.1.69	192.168.1.69	7003	7001
296	0:41:950	7	ACK	No	0	137	192.168.1.69	192.168.1.69	7003	7001

1	Total Capture Time	Packet Count	Dropped Packets (DATA, ACK, EOT)	Retransmits (DATA only)
2	0:41:950	295	18	17

   c. To stop the network emulator from receiving packets press the “Stop” button. The status label will update from green “Active” to red “Stopped”.



d. To reset all the figures to the initial state, click the “Reset” button



Stopped

Packet Delay (ms): 30

Bit Error Rate: 5%

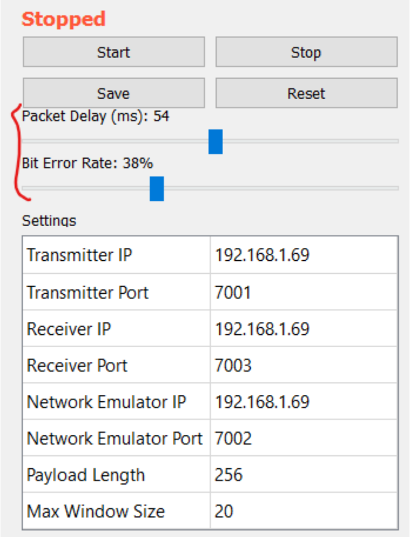
Settings

Transmitter IP	192.168.1.69
Transmitter Port	7001
Receiver IP	192.168.1.69
Receiver Port	7003
Network Emulator IP	192.168.1.69
Network Emulator Port	7002
Payload Length	256
Max Window Size	20

1	Relative Time	Window Size	Packet Type	Retransmit	Seq #	Ack #	Source IP	Destination IP	Source Port	Destination Port

1	Total Capture Time	Packet Count	Dropped Packets (DATA, ACK, EOT)	Retransmits (DATA only)

e. Packet Delay and Bit Error Rate values can be updated using the corresponding sliders



Stopped

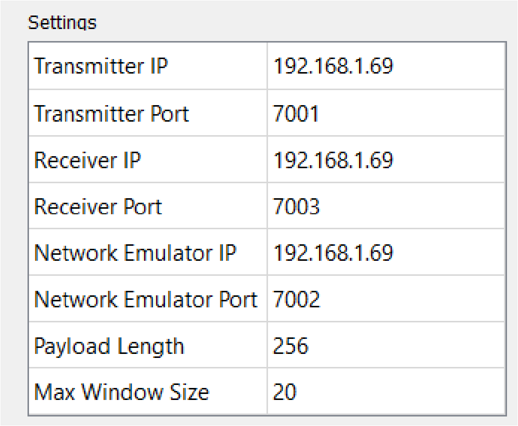
Packet Delay (ms): 54

Bit Error Rate: 38%

Settings

Transmitter IP	192.168.1.69
Transmitter Port	7001
Receiver IP	192.168.1.69
Receiver Port	7003
Network Emulator IP	192.168.1.69
Network Emulator Port	7002
Payload Length	256
Max Window Size	20

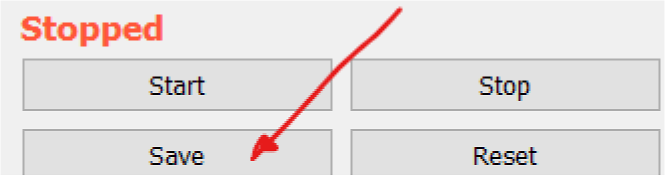
f. Applied settings are displayed in a table format



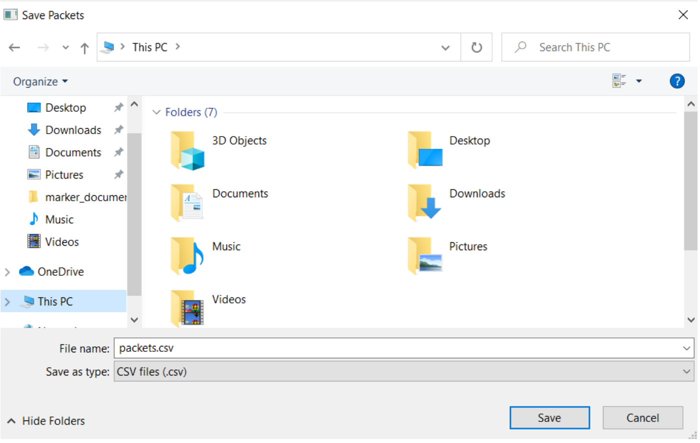
Settings

Transmitter IP	192.168.1.69
Transmitter Port	7001
Receiver IP	192.168.1.69
Receiver Port	7003
Network Emulator IP	192.168.1.69
Network Emulator Port	7002
Payload Length	256
Max Window Size	20

g. To save the data in the packets table to a .CSV file press “Save” button. The prompt window will appear asking to specify the output file location



Stopped



Save Packets

File name: packets.csv

Save as type: CSV files (.csv)

10. Once file transfer is completed, you may find the transferred file in the receiver directory **./receiver/data/message.txt**

11. Log files are available under **./<application_name>/logs/out.log**