EEE3093S - Extra Credit Assignment Submission

Samson Okuthe
OKTSAM001

October 17, 2025

Contents

1 Task 1: Web Server

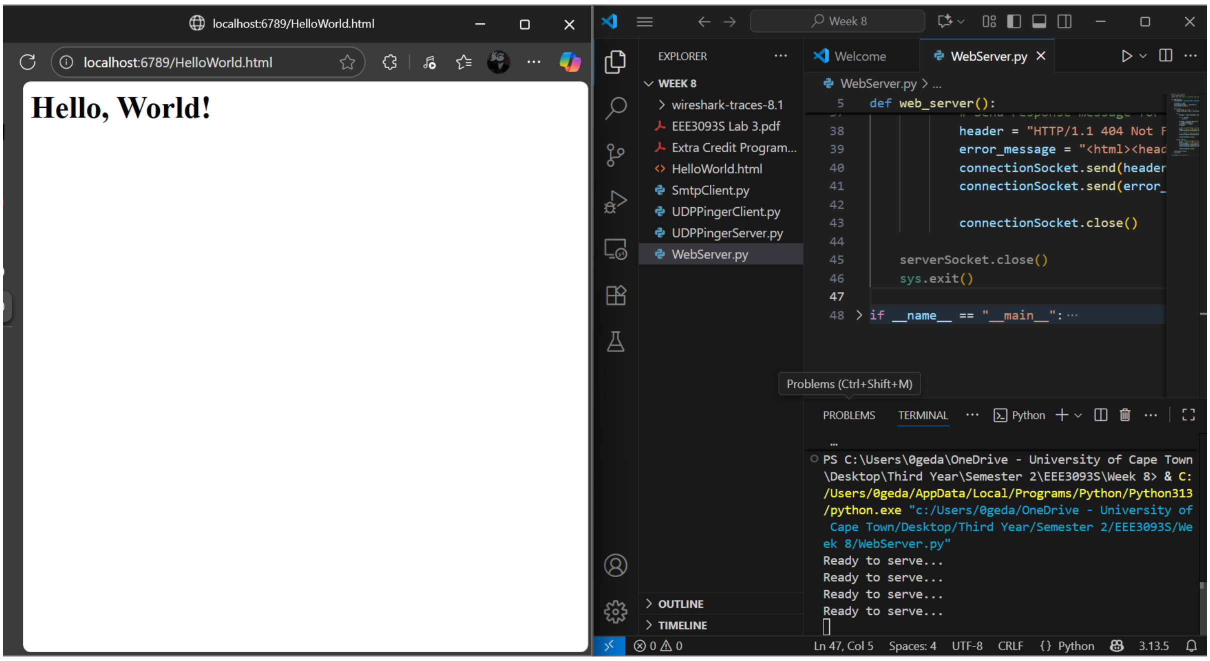
1.1 Task 1 Web Server Code

Here is the complete Python implementation for the simple TCP web server.

```python
#import socket module
from socket import *
import sys # In order to terminate the program

def web_server():
    serverSocket = socket(AF_INET, SOCK_STREAM)

    serverPort = 6789
    serverSocket.bind(('', serverPort))
    serverSocket.listen(1)

    while True:
        print('Ready to serve...')
        connectionSocket, addr = serverSocket.accept()

        try:
            message = connectionSocket.recv(1024).decode()

            if not message:
                continue

            filename = message.split()[1]
            f = open(filename[1:])
            outputdata = f.read()
            f.close()

            # Send one HTTP header line into socket
            header = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"
            connectionSocket.send(header.encode())

            # --- CHANGE 1: Send the entire file content at once ---
            connectionSocket.send(outputdata.encode())

            connectionSocket.close()

        except IOError:
            # Send response message for file not found
            header = "HTTP/1.1 404 Not Found\r\n\r\n"
            error_message = "<html><head></head><body><h1>404 Not Found</h1></body></html>\r\n"
            connectionSocket.send(header.encode())
            connectionSocket.send(error_message.encode())

            connectionSocket.close()

    serverSocket.close()
    sys.exit()

if __name__ == "__main__":
    web_server()
```

Listing 1: WebServer.py - A simple HTTP server

1.2 Demonstration Screenshots

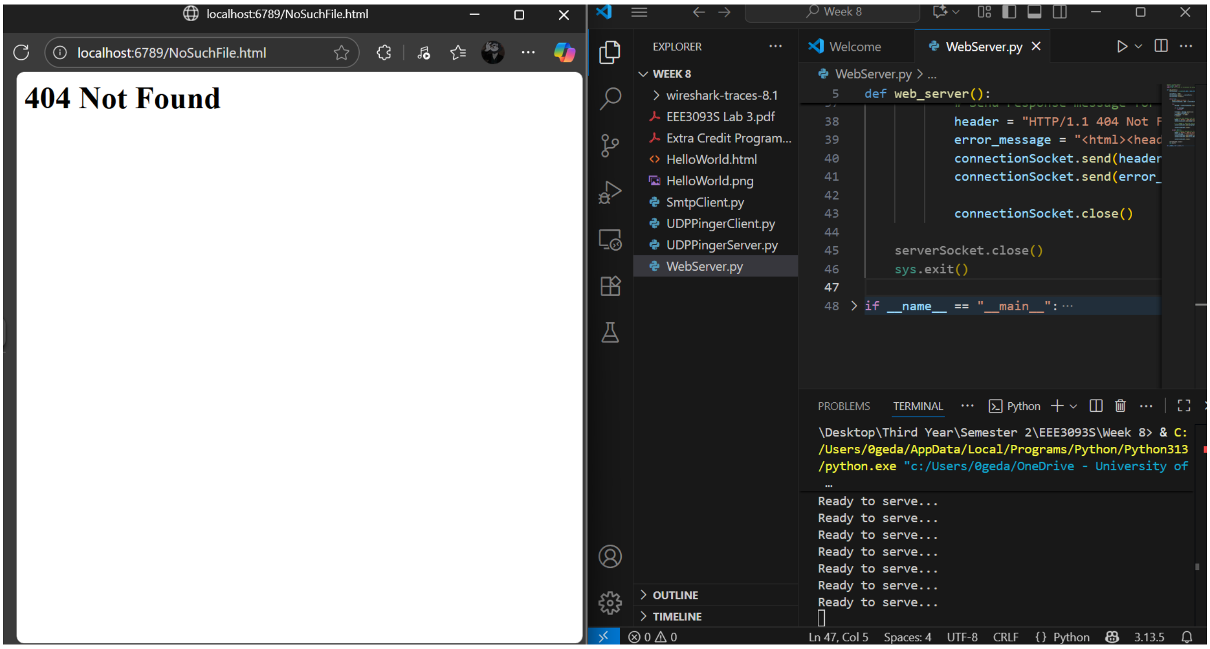


Hello, World!

```
def web_server():
            header = "HTTP/1.1 404 Not F
            error_message = "<html><head
            connectionSocket.send(header
            connectionSocket.send(error_

            connectionSocket.close()

    serverSocket.close()
    sys.exit()

if __name__ == "__main__":···
```

```
PS C:\Users\0geda\OneDrive - University of Cape Town
\Desktop\Third Year\Semester 2\EEE3093S\Week 8> & C:
/Users/0geda/AppData/Local/Programs/Python/Python313
/python.exe "c:/Users/0geda/OneDrive - University of
 Cape Town/Desktop/Third Year/Semester 2/EEE3093S/We
ek 8/WebServer.py"
Ready to serve...
Ready to serve...
Ready to serve...
Ready to serve...
```

Figure 1: The browser successfully displays the `HelloWorld.html` file.



404 Not Found

```
def web_server():
            header = "HTTP/1.1 404 Not F
            error_message = "<html><head
            connectionSocket.send(header
            connectionSocket.send(error_

            connectionSocket.close()

    serverSocket.close()
    sys.exit()

if __name__ == "__main__":···
```

```
\Desktop\Third Year\Semester 2\EEE3093S\Week 8> & C:
/Users/0geda/AppData/Local/Programs/Python/Python313
/python.exe "c:/Users/0geda/OneDrive - University of
…
Ready to serve...
Ready to serve...
Ready to serve...
Ready to serve...
Ready to serve...
Ready to serve...
Ready to serve...
```

Figure 2: The server correctly sends a ”404 Not Found” error.

2 Optional Exercises

2.1 Exercise 1: Multithreaded Web Server

The single-threaded server was modified to handle multiple simultaneous client requests by creating a new thread for each incoming connection.

2.1.1 Multithreaded Web Server Code

```python
from socket import *
import sys
import threading

# This function will handle a single client connection.
# It will run in its own separate thread.
def handle_client(connectionSocket, addr):
    print(f"Accepted connection from {addr}")
    try:
        message = connectionSocket.recv(1024).decode()
        if not message:
            connectionSocket.close()
            return

        filename = message.split()[1]
        f = open(filename[1:])
        outputdata = f.read()
        f.close()

        # Send HTTP OK header and the file content
        header = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"
        connectionSocket.send(header.encode())
        connectionSocket.send(outputdata.encode())

    except IOError:
        # Send 404 Not Found response
        header = "HTTP/1.1 404 Not Found\r\n\r\n"
        error_message = "<html><head></head><body><h1>404 Not Found</h1></body></html>"
        connectionSocket.send(header.encode())
        connectionSocket.send(error_message.encode())

    finally:
        # Close the connection with this specific client
        print(f"Closing connection with {addr}")
        connectionSocket.close()

def main():
    serverSocket = socket(AF_INET, SOCK_STREAM)
    serverPort = 6789
    serverSocket.bind(('', serverPort))
    serverSocket.listen(5) # Listen for up to 5 connections

    print(f"Server is ready and listening on port {serverPort}")

    while True:
        # Main thread waits for a new connection
        connectionSocket, addr = serverSocket.accept()

        # Create a new thread to handle this client
        client_thread = threading.Thread(target=handle_client, args=(connectionSocket, addr))
        client_thread.start()

if __name__ == "__main__":
    main()
```

Listing 2: WebServer_Threaded.py

2.2 Exercise 2: HTTP Client

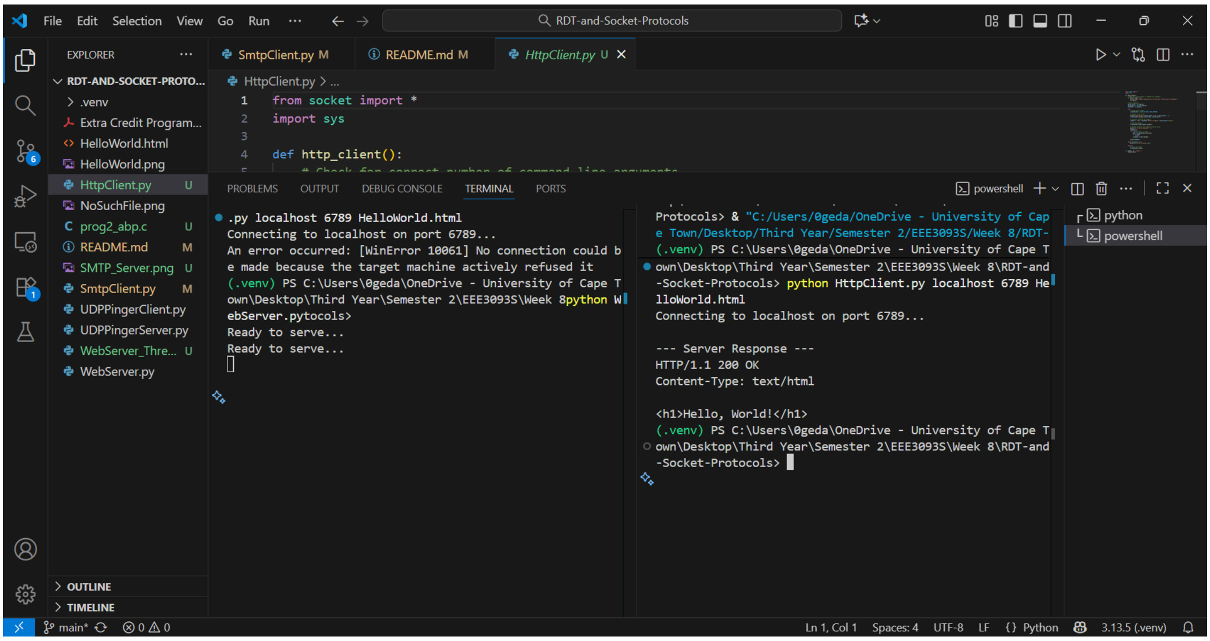
This is a command-line HTTP client that sends a GET request to a specified server.

2.2.1 HTTP Client Code

```python
from socket import *
import sys

def http_client():
    # Check for correct number of command-line arguments
    if len(sys.argv) != 4:
        print("Usage: python HttpClient.py <server_host> <server_port> <filename>")
        sys.exit()

    # Parse arguments
    server_host = sys.argv[1]
    server_port = int(sys.argv[2])
    filename = sys.argv[3]

    try:
        # Create a TCP socket
        clientSocket = socket(AF_INET, SOCK_STREAM)

        # Connect to the server
        print(f"Connecting to {server_host} on port {server_port}...")
        clientSocket.connect((server_host, server_port))

        # Construct the HTTP GET request
        request = f"GET /{filename} HTTP/1.1\r\nHost: {server_host}\r\n\r\n"

        # Send the request
        clientSocket.send(request.encode())

        # Receive and print the response from the server
        print("\n--- Server Response ---")
        response = ""
        while True:
            # Receive data in chunks
            data = clientSocket.recv(1024)
            if not data:
                break
            response += data.decode()

        print(response)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        # Close the socket
        clientSocket.close()

if __name__ == ’__main__’:
    http_client()
```

Listing 3: HttpClient.py

2.2.2 HTTP Client Demonstration



```python
from socket import *
import sys

def http_client():
```

```
.py localhost 6789 HelloWorld.html
Connecting to localhost on port 6789...
An error occurred: [WinError 10061] No connection could b
e made because the target machine actively refused it
(.venv) PS C:\Users\0geda\OneDrive - University of Cape T
own\Desktop\Third Year\Semester 2\EEE3093S\Week 8python W
ebServer.pytocols>
Ready to serve...
Ready to serve...
```

```
Protocols> & "C:/Users/0geda/OneDrive - University of Cap
e Town/Desktop/Third Year/Semester 2/EEE3093S/Week 8/RDT-
(.venv) PS C:\Users\0geda\OneDrive - University of Cape T
own\Desktop\Third Year\Semester 2\EEE3093S\Week 8\RDT-and
-Socket-Protocols> python HttpClient.py localhost 6789 He
lloWorld.html
Connecting to localhost on port 6789...

--- Server Response ---
HTTP/1.1 200 OK
Content-Type: text/html

<h1>Hello, World!</h1>
(.venv) PS C:\Users\0geda\OneDrive - University of Cape T
own\Desktop\Third Year\Semester 2\EEE3093S\Week 8\RDT-and
-Socket-Protocols>
```

Figure 3: Demonstration of `HttpClient.py` fetching a page from the running `WebServer.py`.

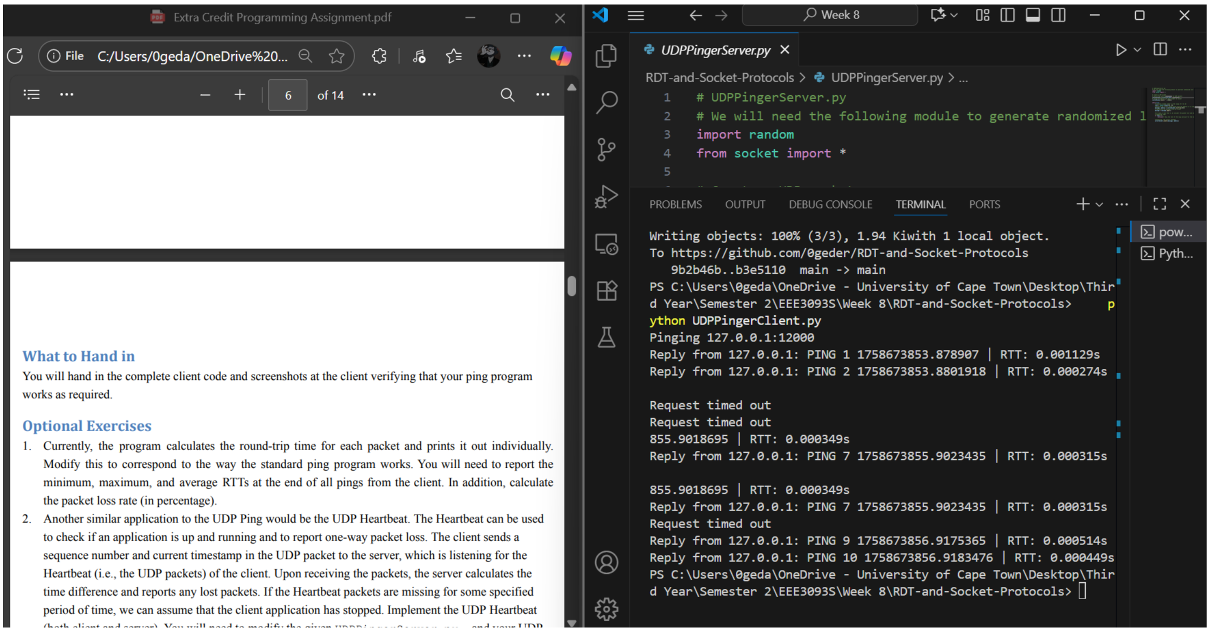
3 Task 2: UDP Pinger

3.1 UDP Pinger Client Code

```python
import time
from socket import *

def pinger_client():
    # Server details
    server_host = ’127.0.0.1’  # localhost
    server_port = 12000

    # Create a UDP socket
    clientSocket = socket(AF_INET, SOCK_DGRAM)

    # Set a timeout of 1 second for the socket
    clientSocket.settimeout(1)

    print(f"Pinging {server_host}:{server_port}")

    # Send 10 pings
    for sequence_number in range(1, 11):
        # Get the current time as a float
        start_time = time.time()

        # Format the message
        message = f’Ping {sequence_number} {start_time}’

        try:
            # Send the message to the server
            clientSocket.sendto(message.encode(), (server_host, server_port))

            # Wait to receive the reply from the server
            modifiedMessage, serverAddress = clientSocket.recvfrom(1024)

            # Get the time when reply was received
            end_time = time.time()

            # Calculate Round Trip Time (RTT)
            rtt = end_time - start_time

            # Print the response and RTT
            print(f’Reply from {serverAddress[0]}: {modifiedMessage.decode()} | RTT: {rtt:.6f}s’)

        except timeout:
            # If a ’timeout’ exception occurs, the packet was lost
            print(’Request timed out’)

    # Close the socket
    clientSocket.close()

if __name__ == ’__main__’:
    pinger_client()
```

Listing 4: UDPPingerClient.py

3.2 Demonstration Screenshot


Extra Credit Programming Assignment.pdf
File C:/Users/0geda/OneDrive%20...
6
of 14
Week 8
UDPPingerServer.py
RDT-and-Socket-Protocols > UDPPingerServer.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
pow...
Pyth...

What to Hand in

You will hand in the complete client code and screenshots at the client verifying that your ping program works as required.

Optional Exercises

1. Currently, the program calculates the round-trip time for each packet and prints it out individually. Modify this to correspond to the way the standard ping program works. You will need to report the minimum, maximum, and average RTTs at the end of all pings from the client. In addition, calculate the packet loss rate (in percentage).
2. Another similar application to the UDP Ping would be the UDP Heartbeat. The Heartbeat can be used to check if an application is up and running and to report one-way packet loss. The client sends a sequence number and current timestamp in the UDP packet to the server, which is listening for the Heartbeat (i.e., the UDP packets) of the client. Upon receiving the packets, the server calculates the time difference and reports any lost packets. If the Heartbeat packets are missing for some specified period of time, we can assume that the client application has stopped. Implement the UDP Heartbeat

```python
# UDPPingerServer.py
# We will need the following module to generate randomized l
import random
from socket import *

```

```
Writing objects: 100% (3/3), 1.94 Kiwith 1 local object.
To https://github.com/0geder/RDT-and-Socket-Protocols
   9b2b46b..b3e5110  main -> main
PS C:\Users\0geda\OneDrive - University of Cape Town\Desktop\Thir
d Year\Semester 2\EEE3093S\Week 8\RDT-and-Socket-Protocols>      p
ython UDPPingerClient.py
Pinging 127.0.0.1:12000
Reply from 127.0.0.1: PING 1 1758673853.878907 | RTT: 0.001129s
Reply from 127.0.0.1: PING 2 1758673853.8801918 | RTT: 0.000274s

Request timed out
Request timed out
855.9018695 | RTT: 0.000349s
Reply from 127.0.0.1: PING 7 1758673855.9023435 | RTT: 0.000315s

855.9018695 | RTT: 0.000349s
Reply from 127.0.0.1: PING 7 1758673855.9023435 | RTT: 0.000315s
Request timed out
Reply from 127.0.0.1: PING 9 1758673856.9175365 | RTT: 0.000514s
Reply from 127.0.0.1: PING 10 1758673856.9183476 | RTT: 0.000449s
PS C:\Users\0geda\OneDrive - University of Cape Town\Desktop\Thir
d Year\Semester 2\EEE3093S\Week 8\RDT-and-Socket-Protocols> 
```

Figure 4: Terminal output showing the UDP client handling successful replies and timeouts.

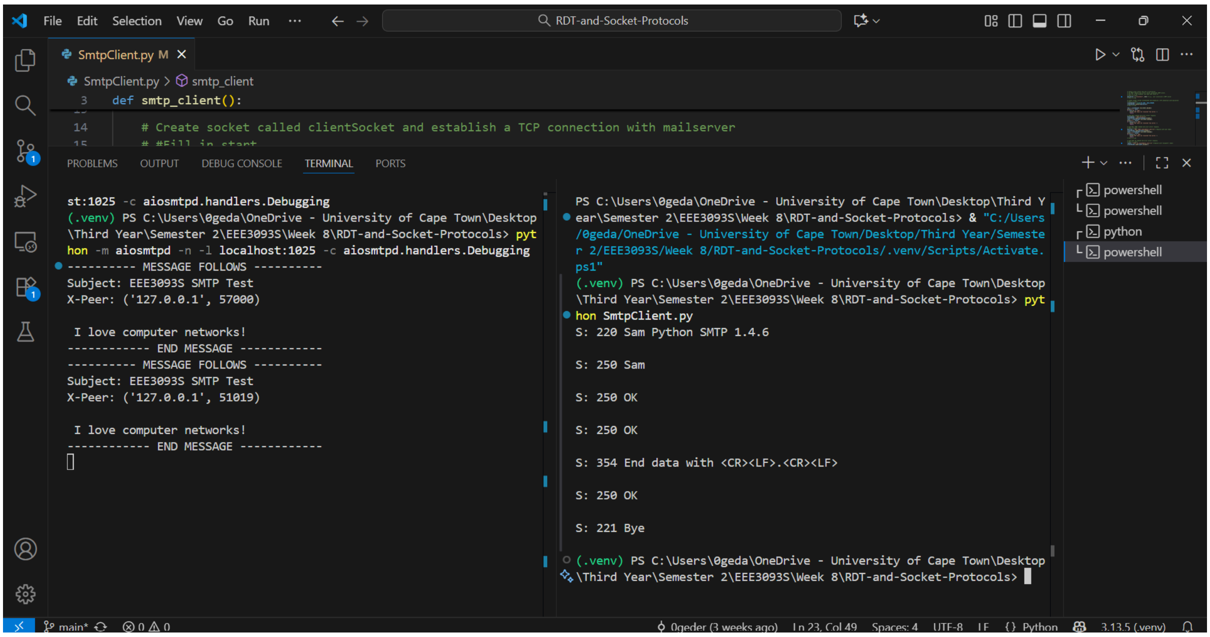
4 Task 3: SMTP Mail Client

4.1 SMTP Mail Client Code

```
from socket import *

def smtp_client():
    msg = "\r\n I love computer networks!"
    endmsg = "\r\n.\r\n"

    # Choose a mail server and call it mailserver
    # You MUST replace this with a valid, accessible SMTP server.
    # Port 25 is the standard, but many ISPs block it.
    # #Fill in start
    mailserver = ("localhost", 1025) # e.g., your university’s SMTP server
    # #Fill in end

    # Create socket called clientSocket and establish a TCP connection with mailserver
    # #Fill in start
    clientSocket = socket(AF_INET, SOCK_STREAM)
    clientSocket.connect(mailserver)
    # #Fill in end

    recv = clientSocket.recv(1024).decode()
    print("S:", recv)
    if recv[:3] != ’220’:
        print(’220 reply not received from server.’)
        return

    # Send HELO command and print server response.
    heloCommand = ’HELO Alice\r\n’
    clientSocket.send(heloCommand.encode())
    recv1 = clientSocket.recv(1024).decode()
    print("S:", recv1)
    if recv1[:3] != ’250’:
        print(’250 reply not received from server.’)
        return

    # Send MAIL FROM command and print server response.
    # #Fill in start
    mailFrom = "MAIL FROM:<samson@test.com>\r\n" # Replace with your email
    clientSocket.send(mailFrom.encode())
    recv2 = clientSocket.recv(1024).decode()
    print("S:", recv2)
    if recv2[:3] != ’250’:
        print(’250 reply not received from server.’)
        return
    # #Fill in end

    # Send RCPT TO command and print server response.
    # #Fill in start
    rcptTo = "RCPT TO:<okuthe@test.com>\r\n" # Replace with recipient’s email
    clientSocket.send(rcptTo.encode())
    recv3 = clientSocket.recv(1024).decode()
    print("S:", recv3)
    if recv3[:3] != ’250’:
        print(’250 reply not received from server.’)
        return
    # #Fill in end

    # Send DATA command and print server response.
    # #Fill in start
    dataCommand = "DATA\r\n"
    clientSocket.send(dataCommand.encode())
    recv4 = clientSocket.recv(1024).decode()
    print("S:", recv4)
    if recv4[:3] != ’354’:
        print(’354 reply not received from server.’)
        return
    # #Fill in end

    # Send message data.
    # #Fill in start
    # You can add email headers here for a proper email
    subject = "Subject: EEE3093S SMTP Test\r\n"
    clientSocket.send(subject.encode())
    clientSocket.send(msg.encode())
    # #Fill in end

    # Message ends with a single period.
    # #Fill in start
    clientSocket.send(endmsg.encode())
    recv5 = clientSocket.recv(1024).decode()
    print("S:", recv5)
    if recv5[:3] != ’250’:
        print(’250 reply not received from server.’)
        return
    # #Fill in end

    # Send QUIT command and get server response.
    # #Fill in start
    quitCommand = "QUIT\r\n"
    clientSocket.send(quitCommand.encode())
    recv6 = clientSocket.recv(1024).decode()
    print("S:", recv6)
    if recv6[:3] != ’221’:
        print(’221 reply not received from server.’)
    # #Fill in end

    clientSocket.close()

if __name__ == ’__main__’:
    smtp_client()
```

Listing 5: SmtpClient.py

4.2 Demonstration Screenshot



```
def smtp_client():
    # Create socket called clientSocket and establish a TCP connection with mailserver
```

```
st:1025 -c aiosmtpd.handlers.Debugging
(.venv) PS C:\Users\0geda\OneDrive - University of Cape Town\Desktop
\Third Year\Semester 2\EEE3093S\Week 8\RDT-and-Socket-Protocols> pyt
hon -m aiosmtpd -n -l localhost:1025 -c aiosmtpd.handlers.Debugging
---------- MESSAGE FOLLOWS ----------
Subject: EEE3093S SMTP Test
X-Peer: ('127.0.0.1', 57000)

 I love computer networks!
------------ END MESSAGE ------------
---------- MESSAGE FOLLOWS ----------
Subject: EEE3093S SMTP Test
X-Peer: ('127.0.0.1', 51019)

 I love computer networks!
------------ END MESSAGE ------------
```

```
PS C:\Users\0geda\OneDrive - University of Cape Town\Desktop\Third Y
ear\Semester 2\EEE3093S\Week 8\RDT-and-Socket-Protocols> & "C:/Users
/0geda/OneDrive - University of Cape Town/Desktop/Third Year/Semeste
r 2/EEE3093S/Week 8/RDT-and-Socket-Protocols/.venv/Scripts/Activate.
ps1"
(.venv) PS C:\Users\0geda\OneDrive - University of Cape Town\Desktop
\Third Year\Semester 2\EEE3093S\Week 8\RDT-and-Socket-Protocols> pyt
hon SmtpClient.py
S: 220 Sam Python SMTP 1.4.6

S: 250 Sam

S: 250 OK

S: 250 OK

S: 354 End data with <CR><LF>.<CR><LF>

S: 250 OK

S: 221 Bye

(.venv) PS C:\Users\0geda\OneDrive - University of Cape Town\Desktop
\Third Year\Semester 2\EEE3093S\Week 8\RDT-and-Socket-Protocols>
```

Figure 5: Output from the local SMTP debugging server, verifying receipt of the email.

5 Task 4: RDT (Alternating-Bit Protocol)

5.1 Design Document

This implementation of the Alternating-Bit Protocol (rdt3.0) uses a finite state machine for the sender (A) with two states:

1. Waiting for a message from the application layer.
2. Waiting for an acknowledgment.

The sender maintains the current sequence number (0 or 1). When a packet is sent, a timer is started. The sender will retransmit the packet if the timer expires.

The receiver (B) maintains an `expected_seqnum`. If a correct, in-order packet arrives, it is delivered to Layer 5 and an ACK for that sequence number is sent. If a corrupt or out-of-order packet arrives, the receiver discards it and resends an ACK for the last correctly received packet to inform the sender.

5.2 Complete C Code for Alternating-Bit Protocol

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* *****************************************************************
 ALTERNATING BIT AND GO-BACK-N NETWORK EMULATOR: VERSION 1.1  J.F.Kurose

   This code should be used for PA2, EEE3093S, at the University of Cape Town.
   It has been generously provided by J.F.Kurose, University of Massachusetts.
**********************************************************************/

#define TRUE 1
#define FALSE 0
#define BIDIRECTIONAL 0    /* change to 1 if you're doing extra credit */
                           /* and write a B_output routine */

/* a "msg" is the data unit passed from layer 5 (teachers code) to layer  */
/* 4 (your code).  It contains the data (characters) to be delivered */
/* to layer 5 running on the other side of the network.         */
struct msg {
  char data[20];
};

/* a packet is the data unit passed from layer 4 (your code) to layer */
/* 3 (teachers code).  Note the pre-defined packet structure, you can not */
/* change it. */
struct pkt {
   int seqnum;
   int acknum;
   int checksum;
   char payload[20];
};

/* Function prototypes for student routines */
void A_output(struct msg message);
void A_input(struct pkt packet);
void A_timerinterrupt();
void A_init();
void B_input(struct pkt packet);
void B_init();

/* Function prototypes for simulator routines */
void starttimer(int AorB, float increment);
void stoptimer(int AorB);
void tolayer3(int AorB, struct pkt packet);
void tolayer5(int AorB, char datasent[20]);
float jimsrand();


/********* STUDENTS WRITE THE NEXT SEVEN ROUTINES *********/

```

```c
// Define states for sender A
#define WAITING_FOR_CALL 0
#define WAITING_FOR_ACK  1

// Global variables for sender A
int A_state;
int A_seqnum;
struct pkt A_last_packet;
float timer_increment = 30.0; // Timeout duration

// Global variable for receiver B
int B_expected_seqnum;

/* Helper function to calculate checksum */
int calculate_checksum(struct pkt packet) {
    int sum = 0;
    sum += packet.seqnum;
    sum += packet.acknum;
    for (int i = 0; i < 20; i++) {
        sum += (unsigned char)packet.payload[i];
    }
    return sum;
}

/* called from layer 5, passed the data to be sent to other side */
void A_output(struct msg message)
{
    // If sender is not ready (still waiting for an ACK), drop the message.
    if (A_state == WAITING_FOR_ACK) {
        printf("  A_output: Sender busy. Dropping message.\n");
        return;
    }

    // Create the packet
    A_last_packet.seqnum = A_seqnum;
    A_last_packet.acknum = 0; // Not used for data packets
    memcpy(A_last_packet.payload, message.data, 20);
    A_last_packet.checksum = calculate_checksum(A_last_packet);

    // Send the packet and start the timer
    printf("  A_output: Sending packet with seq=%d\n", A_seqnum);
    tolayer3(0, A_last_packet);
    starttimer(0, timer_increment);
    A_state = WAITING_FOR_ACK;
}

/* called from layer 3, when a packet arrives for layer 4 */
void A_input(struct pkt packet)
{
    // Verify checksum and check if it's the expected ACK
    if (calculate_checksum(packet) != packet.checksum) {
        printf("  A_input: Received a CORRUPT ACK. Waiting for timeout.\n");
        return;
    }

    if (packet.acknum != A_seqnum) {
        printf("  A_input: Received a DUPLICATE ACK (ack=%d). Waiting for timeout.\n", packet.acknum);
        return;
    }

    // Correct ACK received
    printf("  A_input: Received correct ACK (ack=%d). Ready for next message.\n", packet.acknum);
    stoptimer(0);
    A_state = WAITING_FOR_CALL;
    A_seqnum = 1 - A_seqnum; // Flip the sequence number (0 -> 1, 1 -> 0)
}

/* called when A's timer goes off */
void A_timerinterrupt()
{
    printf("  A_timerinterrupt: Timeout! Resending packet with seq=%d\n", A_last_packet.seqnum);
    tolayer3(0, A_last_packet);
    starttimer(0, timer_increment);
}

/* the following routine will be called once (only) before any other */
/* entity A routines are called. You can use it to do any initialization */
void A_init()
{
    A_state = WAITING_FOR_CALL;
    A_seqnum = 0;
    printf("A_init: Sender initialized. Ready to accept messages.\n");
}


/* Note that with simplex transfer from a-to-b, there is no B_output() */

/* called from layer 3, when a packet arrives for layer 4 at B*/
void B_input(struct pkt packet)
{
    // Check if packet is corrupt OR has the wrong sequence number
    if (calculate_checksum(packet) != packet.checksum || packet.seqnum != B_expected_seqnum) {
        int last_ack = 1 - B_expected_seqnum;
        printf("  B_input: Received corrupt or out-of-order packet. Resending last ACK=%d.\n", last_ack);

        struct pkt ack_pkt;
        ack_pkt.acknum = last_ack;
        ack_pkt.checksum = ack_pkt.acknum; // Simple checksum for ACK is just the ACK number
        tolayer3(1, ack_pkt);
        return;
    }

    // Packet is correct and in order
    printf("  B_input: Received correct packet (seq=%d). Sending ACK and delivering to layer 5.\n", packet.seqnum);
    tolayer5(1, packet.payload);

    // Send ACK for the packet we just received
    struct pkt ack_pkt;
    ack_pkt.acknum = B_expected_seqnum;
    ack_pkt.checksum = ack_pkt.acknum;
    tolayer3(1, ack_pkt);

    // Flip the expected sequence number for the next packet
    B_expected_seqnum = 1 - B_expected_seqnum;
}

/* the following routine will be called once (only) before any other */
/* entity B routines are called. You can use it to do any initialization */
void B_init()
{
    B_expected_seqnum = 0;
    printf("B_init: Receiver initialized. Expecting packet with seq=0.\n");
}


/*****************************************************************
**************** NETWORK EMULATION CODE STARTS BELOW ***********
The code below emulates the layer 3 and below network environment:
  - emulates the tranmission and delivery (possibly with bit-level corruption
    and packet loss) of packets across the network
  - handles the starting/stopping of a timer, and generates timer
    interrupts (resulting in calling students timer handler).
  - generates message to be sent (passed from later 5 to 4)

THERE IS NO REASON THAT ANY STUDENT SHOULD HAVE TO READ OR UNDERSTAND
THE CODE BELOW.  YOU SHOLD NOT TOUCH, OR REFERENCE (in your code) ANY
OF THE DATA STRUCTURES BELOW.  If you're interested in how I designed
the emulator, you're welcome to look at the code - but again, you should have
```

```
to, and you definitely should not have to modify
******************************************************************/

struct event {
   float evtime;           /* event time */
   int evtype;             /* event type code */
   int eventity;           /* entity where event occurs */
   struct pkt *pktptr;     /* ptr to packet (if any) assoc w/ this event */
   struct event *prev;
   struct event *next;
 };
struct event *evlist = NULL;   /* the event list */

/* possible events: */
#define  TIMER_INTERRUPT 0
#define  FROM_LAYER5     1
#define  FROM_LAYER3     2

#define  OFF             0
#define  ON              1
#define   A    0
#define   B    1



int TRACE = 1;             /* for my debugging */
int nsim = 0;              /* number of messages from 5 to 4 so far */
int nsimmax = 0;           /* number of msgs to generate, then stop */
float time = 0.000;
float lossprob;            /* probability that a packet is dropped   */
float corruptprob;         /* probability that one bit is packet is flipped */
float lambda;              /* arrival rate of messages from layer 5 */
int   ntolayer3;           /* number sent into layer 3 */
int   nlost;               /* number lost in media */
int ncorrupt;              /* number corrupted by media */

void init();
void generate_next_arrival();
void insertevent(struct event* p);

int main()
{
   struct event *eventptr;
   struct msg  msg2give;
   struct pkt  pkt2give;

   int i,j;
   char c;

   init();
   A_init();
   B_init();

   while (1) {
        eventptr = evlist;            /* get next event to simulate */
        if (eventptr==NULL)
           goto terminate;
        evlist = evlist->next;        /* remove this event from event list */
        if (evlist!=NULL)
           evlist->prev=NULL;
        if (TRACE>=2) {
           printf("\nEVENT time: %f,",eventptr->evtime);
           printf("  type: %d",eventptr->evtype);
           if (eventptr->evtype==0)
         printf(", timerinterrupt  ");
             else if (eventptr->evtype==1)
               printf(", fromlayer5 ");
             else
       printf(", fromlayer3 ");
           printf(" entity: %d\n",eventptr->eventity);
           }
        time = eventptr->evtime;        /* update time to next event time */
        if (nsim==nsimmax)
```

```
    break;                           /* all done with simulation */
        if (eventptr->evtype == FROM_LAYER5 ) {
            generate_next_arrival();    /* set up future arrival */
            /* fill in msg to give with string of same letter */
            j = nsim % 26;
            for (i=0; i<20; i++)
               msg2give.data[i] = 97 + j;
            if (TRACE>2) {
               printf("          MAINLOOP: data given to student: ");
                 for (i=0; i<20; i++)
                  printf("%c", msg2give.data[i]);
               printf("\n");
       }
            nsim++;
            if (eventptr->eventity == A)
               A_output(msg2give);
             else
               ;
  }
          else if (eventptr->evtype ==  FROM_LAYER3) {
            pkt2give.seqnum = eventptr->pktptr->seqnum;
            pkt2give.acknum = eventptr->pktptr->acknum;
            pkt2give.checksum = eventptr->pktptr->checksum;
            for (i=0; i<20; i++)
                pkt2give.payload[i] = eventptr->pktptr->payload[i];
      if (eventptr->eventity ==A)      /* deliver packet to A */
               A_input(pkt2give);
      else B_input(pkt2give);          /* deliver packet to B */
      free(eventptr->pktptr);          /* free the memory for packet */
            }
          else if (eventptr->evtype ==  TIMER_INTERRUPT) {
            if (eventptr->eventity == A)
         A_timerinterrupt();
      else
               ;
      }
          else  {
       printf("INTERNAL PANIC: unknown event type \n");
             }
        free(eventptr);
        }

terminate:
   printf(" Simulator terminated at time %f\n after sending %d msgs from layer5\n",time,nsim);
   return 0;
}

void init()                         /* initialize the simulator */
{
  int i;
  float sum, avg;
  float jimsrand();


   printf("-----  Stop and Wait Network Simulator Version 1.1 -------- \n\n");
   printf("Enter the number of messages to simulate: ");
   scanf("%d",&nsimmax);
   printf("Enter  packet loss probability [enter 0.0 for no loss]:");
   scanf("%f",&lossprob);
   printf("Enter packet corruption probability [0.0 for no corruption]:");
   scanf("%f",&corruptprob);
   printf("Enter average time between messages from sender's layer5 [ > 0.0]:");
   scanf("%f",&lambda);
   printf("Enter TRACE:");
   scanf("%d",&TRACE);

   srand(9999);              /* init random number generator */
   sum = 0.0;                /* test random number generator for students */
   for (i=0; i<1000; i++)
      sum=sum+jimsrand();    /* jimsrand() should be uniform in [0,1] */
   avg = sum/1000.0;
   if (avg < 0.25 || avg > 0.75) {
    printf("It is likely that random number generation on your machine\n");
    printf("is different from what this emulator expects.  Please follow\n");
    printf("the advice in the assignment manual.\n");
    exit(0);
    }

   ntolayer3 = 0;
   nlost = 0;
   ncorrupt = 0;

   time=0.0;                    /* initialize time to 0.0 */
   generate_next_arrival();     /* initialize event list */
}

/****************************************************************************/
/* jimsrand(): return a float in range [0,1].  The routine below is used by */
/* hosts A and B to send packets to layer 3.                      is the RANDOM     */
/* numbers generated by rand() which returns an integer in range [0, SRT_MAX]*/
/****************************************************************************/
// float jimsrand()
// {
//   double mmm = 2147483647;   /* largest int  - MACHINE DEPENDENT!!!!!!!!   */
//   float x;                   /* individual students may need to change mmm */
//   x = rand()/mmm;            /* x should be uniform in [0,1] */
//   return(x);
// }

/****************************************************************************/
/* jimsrand(): return a float in range [0,1]. A simple LCG. */
/****************************************************************************/
long random_seed = 12345; // A seed for our own random number generator

float jimsrand()
{
    // A simple linear congruential generator (LCG) to ensure consistency
    // across different systems.
    random_seed = (random_seed * 1103515245 + 12345) & 0x7fffffff;
    return ((float)random_seed / (float)0x7fffffff);
}

/********************* EVENT HANDLINE ROUTINES *******/
/*  The next set of routines handle the event list   */
/*****************************************************/

void generate_next_arrival()
{
   double x,log(),ceil();
   struct event *evptr;
    //char *malloc();
   float ttime;
   int tempint;

   if (TRACE>2)
       printf("          GENERATE NEXT ARRIVAL: creating new arrival\n");

   x = lambda*jimsrand()*2;  /* x is uniform on [0,2*lambda] */
                             /* having mean of lambda        */
   evptr = (struct event *)malloc(sizeof(struct event));
   evptr->evtime =  time + x;
   evptr->evtype =  FROM_LAYER5;
   if (BIDIRECTIONAL && (jimsrand()>0.5) )
      evptr->eventity = B;
    else
      evptr->eventity = A;
   insertevent(evptr);
}


void insertevent(p)
   struct event *p;
{
   struct event *q,*qold;

   if (TRACE>2) {
      printf("            INSERTEVENT: time is %lf\n",time);
      printf("            INSERTEVENT: future time will be %lf\n",p->evtime);
      }
   q = evlist;      /* q points to header of list in which p struct inserted */
   if (q==NULL) {   /* list is empty */
        evlist=p;
        p->next=NULL;
        p->prev=NULL;
        }
     else {
        for (qold = q; q !=NULL && p->evtime > q->evtime; q=q->next)
              qold=q;
        if (q==NULL) {   /* end of list */
             qold->next = p;
             p->prev = qold;
             p->next = NULL;
             }
           else if (q==evlist) { /* front of list */
             p->next=evlist;
             p->prev=NULL;
             p->next->prev=p;
             evlist = p;
             }
           else {     /* middle of list */
             p->next=q;
             p->prev=q->prev;
             q->prev->next=p;
             q->prev=p;
             }
         }
}

void printevlist()
{
  struct event *q;
  int i;
  printf("--------------\nEvent List Follows:\n");
  for(q = evlist; q!=NULL; q=q->next) {
    printf("Event time: %f, type: %d entity: %d\n",q->evtime,q->evtype,q->eventity);
    }
  printf("--------------\n");
}



/********************** Student-callable ROUTINES **********************/

/* called by students routine to cancel a previously-started timer */
void stoptimer(AorB)
int AorB;  /* A or B is trying to stop timer */
{
 struct event *q,*qold;

 if (TRACE>2)
    printf("          STOP TIMER: stopping timer at %f\n",time);
/* for (q=evlist; q!=NULL && q->next!=NULL; q = q->next)  */
 for (q=evlist; q!=NULL ; q = q->next)
    if ( (q->evtype==TIMER_INTERRUPT) && (q->eventity==AorB) ) {
       /* remove this event */
       if (q->next==NULL && q->prev==NULL)
          evlist=NULL;         /* remove first and only event on list */
          else if (q->next==NULL) /* end of list - there is one in front */
          q->prev->next = NULL;
          else if (q==evlist) { /* front of list - there must be event after */
             q->next->prev=NULL;
             evlist = q->next;
             }
           else {     /* middle of list */
             q->next->prev = q->prev;
             q->prev->next =  q->next;
             }
       free(q);
```

```c
       return;
     }
  printf("Warning: unable to cancel your timer. It wasn't running.\n");
}


void starttimer(AorB,increment)
int AorB;  /* A or B is trying to start timer */
float increment;
{

 struct event *q;
 struct event *evptr;
 //char *malloc();

 if (TRACE>2)
    printf("          START TIMER: starting timer at %f\n",time);
 /* be nice: check if timer is already started, if so, then  warn */
/* for (q=evlist; q!=NULL && q->next!=NULL; q = q->next)  */
   for (q=evlist; q!=NULL ; q = q->next)
    if ( (q->evtype==TIMER_INTERRUPT) && (q->eventity==AorB) ) {
      printf("Warning: attempt to start a timer that is already started\n");
      return;
      }

/* create future event for timer interrupt */
   evptr = (struct event *)malloc(sizeof(struct event));
   evptr->evtime =  time + increment;
   evptr->evtype =  TIMER_INTERRUPT;
   evptr->eventity = AorB;
   insertevent(evptr);
}


/************************** TOLAYER3 ***************/
void tolayer3(AorB,packet)
int AorB;  /* A or B is sending this packet */
struct pkt packet;
{
 struct pkt *mypktptr;
 struct event *evptr,*q;
 //char *malloc();
 float lastime, x, jimsrand();
 int i;


 ntolayer3++;

 /* simulate losses: */
 if (jimsrand() < lossprob)  {
      nlost++;
      if (TRACE>0)
  printf("          TOLAYER3: packet being lost\n");
      return;
    }

/* make a copy of the packet student just gave me since he/she may decide */
/* to do something with the packet after we return back to him/her */
 mypktptr = (struct pkt *)malloc(sizeof(struct pkt));
 mypktptr->seqnum = packet.seqnum;
 mypktptr->acknum = packet.acknum;
 mypktptr->checksum = packet.checksum;
 for (i=0; i<20; i++)
    mypktptr->payload[i] = packet.payload[i];
 if (TRACE>2)  {
   printf("          TOLAYER3: seq: %d, ack %d, check: %d ", mypktptr->seqnum,
    mypktptr->acknum,  mypktptr->checksum);
    for (i=0; i<20; i++)
        printf("%c",mypktptr->payload[i]);
    printf("\n");
   }

/* create future event for arrival of packet at the other side */
```

```c
  evptr = (struct event *)malloc(sizeof(struct event));
  evptr->evtype =  FROM_LAYER3;   /* packet will pop out from layer3 */
  evptr->eventity = (AorB+1) % 2; /* event occurs at other entity */
  evptr->pktptr = mypktptr;       /* save ptr to my copy of packet */
/* finally, compute the arrival time of packet at the other end.
   medium can not reorder, so make sure packet arrives between previous packet
   and next packet scheduled delivery time.  At the beginning, lastime is 0.0. */
  lastime = time;
/* for (q=evlist; q!=NULL && q->next!=NULL; q = q->next) */
 for (q=evlist; q!=NULL ; q = q->next)
    if ( (q->evtype==FROM_LAYER3  && q->eventity==evptr->eventity) )
      lastime = q->evtime;
 evptr->evtime =  lastime + 1 + 9*jimsrand();



 /* simulate corruption: */
 if (jimsrand() < corruptprob)  {
    ncorrupt++;
    if ( (x = jimsrand()) < .75)
       mypktptr->payload[0]='Z';   /* corrupt payload */
      else if (x < .875)
       mypktptr->seqnum = 999999;
      else
       mypktptr->acknum = 999999;
    if (TRACE>0)
  printf("          TOLAYER3: packet being corrupted\n");
    }

  if (TRACE>2)
     printf("          TOLAYER3: scheduling arrival on other side\n");
  insertevent(evptr);
}

void tolayer5(AorB,datasent)
  int AorB;
  char datasent[20];
{
  int i;
  if (TRACE>2) {
     printf("          TOLAYER5: data received: ");
     for (i=0; i<20; i++)
        printf("%c",datasent[i]);
     printf("\n");
   }

}
```

Listing 6: prog2_abp.c

5.3 Sample Output

The following is a curated sample from the simulation output, demonstrating the key functionalities of the Alternating-Bit Protocol, including error recovery from packet loss and corruption. The full log was omitted for brevity.

```
-----  Stop and Wait Network Simulator Version 1.1 --------

Enter the number of messages to simulate: 20
Enter packet loss probability [enter 0.0 for no loss]:0.3
Enter packet corruption probability [0.0 for no corruption]:0.1
Enter average time between messages from sender's layer5 [ > 0.0]:1000
Enter TRACE:2
A_init: Sender initialized. Ready to accept messages.
B_init: Receiver initialized. Expecting packet with seq=0.

... (Initial simulation events) ...
EVENT time: 530.514221, type: 1, fromlayer5 entity: 0
```

```
    A_output: Sending packet with seq=0
        TOLAYER3: packet being lost

EVENT time: 560.514221, type: 0, timerinterrupt entity: 0
    A_timerinterrupt: Timeout! Resending packet with seq=0

EVENT time: 570.260742, type: 2, fromlayer3 entity: 1
B_input: Received correct packet (seq=0). Sending ACK and delivering to layer 5.


[--- Example 1: Recovery from Packet Loss ---]

EVENT time: 530.514221,  type: 1, fromlayer5  entity: 0
  A_output: Sending packet with seq=0
          TOLAYER3: packet being lost

EVENT time: 560.514221,  type: 0, timerinterrupt   entity: 0
  A_timerinterrupt: Timeout! Resending packet with seq=0

... (Log continues with more events) ...

[--- Example 2: Recovery from Packet Corruption ---]

EVENT time: 680.514221,  type: 0, timerinterrupt   entity: 0
  A_timerinterrupt: Timeout! Resending packet with seq=0
          TOLAYER3: packet being corrupted

EVENT time: 686.225220,  type: 2, fromlayer3  entity: 1
    B_input: Received corrupt or out-of-order packet. Resending last ACK=0.

... (Log continues until the end) ...

[--- Final Output ---]

Simulator terminated at time 22280.513672
 after sending 20 msgs from layer5
```

6 Task 5: RDT (Go-Back-N Protocol)

6.1 Design Document

This Go-Back-N implementation uses a sender window size of 8. The sender maintains a buffer for all unacknowledged packets, a ‘base‘ pointer for the oldest unacknowledged packet, and a ‘nextseqnum‘ pointer for the next available slot in the window. A single timer is used, which is always associated with the packet at the ‘base‘ of the window. On a timeout, all packets from ‘base‘ to ‘nextseqnum-1‘ are retransmitted. The receiver is simple: it only accepts in-order packets. If a packet arrives with the ‘expectedseqnum‘, it is delivered to Layer 5, a cumulative ACK is sent for that sequence number, and the ‘expectedseqnum‘ is incremented. All out-of-order or corrupt packets are discarded, and an ACK for the last correctly received in-order packet is re-sent.

6.2 Complete C Code for Go-Back-N

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* ******************************************************************
 ALTERNATING BIT AND GO-BACK-N NETWORK EMULATOR: VERSION 1.1  J.F.Kurose

   This code should be used for PA2, EEE3093S, at the University of Cape Town.
   It has been generously provided by J.F.Kurose, University of Massachusetts.
**********************************************************************/

#define TRUE 1
#define FALSE 0
#define BIDIRECTIONAL 0

struct msg {
  char data[20];
};

struct pkt {
   int seqnum;
   int acknum;
   int checksum;
   char payload[20];
};

void A_output(struct msg message);
void A_input(struct pkt packet);
void A_timerinterrupt();
void A_init();
void B_input(struct pkt packet);
void B_init();

void starttimer(int AorB, float increment);
void stoptimer(int AorB);
void tolayer3(int AorB, struct pkt packet);
void tolayer5(int AorB, char datasent[20]);
float jimsrand();


/********* STUDENTS WRITE THE NEXT SEVEN ROUTINES *********/
#define WINDOW_SIZE 8
#define BUFFER_SIZE 50

// Sender (A) variables
int a_base;
int a_nextseqnum;
struct pkt a_buffer[BUFFER_SIZE];
float timer_increment = 30.0;

// Receiver (B) variables
int b_expectedseqnum;

/* Helper function to calculate checksum */
int calculate_checksum(struct pkt packet) {
    int checksum = 0;
    checksum += packet.seqnum;
    checksum += packet.acknum;
    for (int i = 0; i < 20; i++) { checksum += (unsigned char)packet.payload[i]; }
    return checksum;
}

void A_output(struct msg message) {
    if (a_nextseqnum >= BUFFER_SIZE) {
        printf("  A_output: Buffer full, dropping message.\n");
        return;
    }
    if (a_nextseqnum >= a_base + WINDOW_SIZE) {
        printf("  A_output: Window is full, buffering message for later.\n");
        // Buffer the message data for when the window slides
        memcpy(a_buffer[a_nextseqnum].payload, message.data, 20);
        a_buffer[a_nextseqnum].seqnum = a_nextseqnum; // Store seqnum for later
        a_nextseqnum++; // Increment so we know we have a buffered message
        return;
    }

    // Create and store the packet in the buffer
    memcpy(a_buffer[a_nextseqnum].payload, message.data, 20);
    a_buffer[a_nextseqnum].seqnum = a_nextseqnum;
    a_buffer[a_nextseqnum].acknum = 0; // Not used
    a_buffer[a_nextseqnum].checksum = calculate_checksum(a_buffer[a_nextseqnum]);

    // Send the packet
    printf("  A_output: Sending packet with seq=%d\n", a_nextseqnum);
    tolayer3(0, a_buffer[a_nextseqnum]);

    if (a_base == a_nextseqnum) { starttimer(0, timer_increment); }
    a_nextseqnum++;
}

void A_input(struct pkt packet) {
    if (calculate_checksum(packet) != packet.checksum) {
        printf("  A_input: Received CORRUPT ACK. Ignoring.\n");
        return;
    }

    printf("  A_input: Received ACK for %d. Updating base.\n", packet.acknum);

    // Check if the ACK is for a packet within the current window
    if (packet.acknum >= a_base) {
        a_base = packet.acknum + 1;
        stoptimer(0); // Stop the old timer
        // If there are still unacknowledged packets in the window, start a new timer
        if (a_base < a_nextseqnum) {
            starttimer(0, timer_increment);
        }
    }
}

void A_timerinterrupt() {
    printf("  A_timerinterrupt: TIMEOUT! Resending window from base=%d\n", a_base);
    stoptimer(0); // Stop current timer before starting a new one
    starttimer(0, timer_increment);
    for (int i = a_base; i < a_nextseqnum; i++) {
        printf("  A_timerinterrupt: Resending packet seq=%d\n", i);
        tolayer3(0, a_buffer[i]);
    }
}

void A_init() {
    a_base = 0;
    a_nextseqnum = 0;
}

void B_input(struct pkt packet) {
    if (calculate_checksum(packet) == packet.checksum && packet.seqnum ==
    b_expectedseqnum) {
        printf("  B_input: Received correct packet (seq=%d). Delivering and sending ACK.\n", packet.seqnum);
        tolayer5(1, packet.payload);

        struct pkt ack_pkt;
        ack_pkt.acknum = b_expectedseqnum;
        ack_pkt.checksum = ack_pkt.acknum;
        tolayer3(1, ack_pkt);

        b_expectedseqnum++;
    } else {
        int last_ack = b_expectedseqnum - 1;
        printf("  B_input: Received out-of-order/corrupt packet. Resending last good ACK=%d.\n", last_ack);
        if (last_ack >= 0) { // Don't send ACK -1
            struct pkt ack_pkt;
            ack_pkt.acknum = last_ack;
            ack_pkt.checksum = ack_pkt.acknum;
            tolayer3(1, ack_pkt);
        }
    }
}

void B_init() {
    b_expectedseqnum = 0;
}


/****************************************************************
**************** NETWORK EMULATION CODE STARTS BELOW ***********
*****************************************************************/

// (The rest of this file is the exact same boilerplate simulator code as the ABP file)
// (It starts with ‘struct event‘ and ends with ‘tolayer5‘)

struct event {
   float evtime; int evtype; int eventity; struct pkt *pktptr;
   struct event *prev; struct event *next;
};
struct event *evlist = NULL;
#define TIMER_INTERRUPT 0
#define FROM_LAYER5 1
#define FROM_LAYER3 2
#define OFF 0
#define ON 1
#define A 0
#define B 1
int TRACE = 1; int nsim = 0; int nsimmax = 0; float time = 0.000;
float lossprob; float corruptprob; float lambda;
int ntolayer3; int nlost; int ncorrupt;
long random_seed = 12345;

void init();
void generate_next_arrival();
void insertevent(struct event*);

int main() {
   struct event *eventptr;
   struct msg  msg2give;
   struct pkt  pkt2give;
   int i,j;

   init();
   A_init();
   B_init();

   while (1) {
        eventptr = evlist;
        if (eventptr==NULL) goto terminate;
        evlist = evlist->next;
        if (evlist!=NULL) evlist->prev=NULL;
        if (TRACE>=2) {
           printf("\nEVENT time: %f,",eventptr->evtime);
           printf("  type: %d",eventptr->evtype);
           if (eventptr->evtype==0) printf(", timerinterrupt");
           else if (eventptr->evtype==1) printf(", fromlayer5 ");
           else printf(", fromlayer3 ");
           printf(" entity: %d\n",eventptr->eventity);
        }
        time = eventptr->evtime;
        if (nsim==nsimmax && evlist==NULL) break;
        if (eventptr->evtype == FROM_LAYER5) {
            if (nsim < nsimmax) {
                generate_next_arrival();
                j = nsim % 26;
                for (i=0; i<20; i++) msg2give.data[i] = 97 + j;
                nsim++;
                if (eventptr->eventity == A) A_output(msg2give);
            }
        } else if (eventptr->evtype == FROM_LAYER3) {
            pkt2give.seqnum = eventptr->pktptr->seqnum;
            pkt2give.acknum = eventptr->pktptr->acknum;
            pkt2give.checksum = eventptr->pktptr->checksum;
            for (i=0; i<20; i++) pkt2give.payload[i] = eventptr->pktptr->payload[i];
            if (eventptr->eventity ==A) A_input(pkt2give);
            else B_input(pkt2give);
            free(eventptr->pktptr);
        } else if (eventptr->evtype == TIMER_INTERRUPT) {
            if (eventptr->eventity == A) A_timerinterrupt();
        } else {
       printf("INTERNAL PANIC: unknown event type \n");
        }
        free(eventptr);
    }

terminate:
   printf(" Simulator terminated at time %f\n after sending %d msgs from layer5\n",time, nsim);
   return 0;
}

void init() {
   printf("-----  Go-Back-N Network Simulator Version 1.1 -------- \n\n");
   printf("Enter the number of messages to simulate: ");
   scanf("%d",&nsimmax);
   printf("Enter  packet loss probability [enter 0.0 for no loss]:");
   scanf("%f",&lossprob);
   printf("Enter packet corruption probability [0.0 for no corruption]:");
   scanf("%f",&corruptprob);
   printf("Enter average time between messages from sender's layer5 [ > 0.0]:");
   scanf("%f",&lambda);
   printf("Enter TRACE:");
   scanf("%d",&TRACE);

   ntolayer3 = 0; nlost = 0; ncorrupt = 0;
   time=0.0;
   generate_next_arrival();
}

float jimsrand() {
    random_seed = (random_seed * 1103515245 + 12345) & 0x7fffffff;
    return ((float)random_seed / (float)0x7fffffff);
}

void generate_next_arrival() {
   double x;
   struct event *evptr;
   if (nsim >= nsimmax) return;
   x = lambda*jimsrand()*2;
   evptr = (struct event *)malloc(sizeof(struct event));
   evptr->evtime =  time + x;
   evptr->evtype =  FROM_LAYER5;
   if (BIDIRECTIONAL && (jimsrand()>0.5) ) evptr->eventity = B;
   else evptr->eventity = A;
   insertevent(evptr);
}
```

```

void insertevent(struct event *p) {
   struct event *q,*qold;
   q = evlist;
   if (q==NULL) {
        evlist=p; p->next=NULL; p->prev=NULL;
   } else {
        for (qold=q; q!=NULL && p->evtime > q->evtime; q=q->next) qold=q;
        if (q==NULL) {
             qold->next=p; p->prev=qold; p->next=NULL;
        } else if (q==evlist) {
             p->next=evlist; p->prev=NULL; p->next->prev=p; evlist=p;
        } else {
             p->next=q; p->prev=q->prev; q->prev->next=p; q->prev=p;
        }
   }
}

void stoptimer(int AorB) {
 struct event *q;
 for (q=evlist; q!=NULL; q=q->next)
    if ((q->evtype==TIMER_INTERRUPT) && (q->eventity==AorB)) {
       if (q->next==NULL && q->prev==NULL) evlist=NULL;
       else if (q->next==NULL) q->prev->next = NULL;
       else if (q==evlist) { q->next->prev=NULL; evlist = q->next; }
       else { q->next->prev = q->prev; q->prev->next = q->next; }
       free(q);
       return;
     }
}

void starttimer(int AorB, float increment) {
 struct event *q;
 struct event *evptr;
 for (q=evlist; q!=NULL; q=q->next)
    if ((q->evtype==TIMER_INTERRUPT) && (q->eventity==AorB)) {
      printf("Warning: attempt to start a timer that is already started\n");
      return;
    }
   evptr = (struct event *)malloc(sizeof(struct event));
   evptr->evtime =  time + increment;
   evptr->evtype =  TIMER_INTERRUPT;
   evptr->eventity = AorB;
   insertevent(evptr);
}

void tolayer3(int AorB, struct pkt packet) {
 struct pkt *mypktptr;
 struct event *evptr, *q;
 float lastime, x;
 int i;
 ntolayer3++;
 if (jimsrand() < lossprob)  {
      nlost++;
      if (TRACE>0) printf("          TOLAYER3: packet being lost\n");
      return;
 }
 mypktptr = (struct pkt *)malloc(sizeof(struct pkt));
 *mypktptr = packet;
 if (TRACE>2)  {
   printf("          TOLAYER3: seq: %d, ack %d, check: %d ", mypktptr->seqnum, mypktptr->acknum, mypktptr->checksum);
   for (i=0; i<20; i++) printf("%c",mypktptr->payload[i]);
   printf("\n");
 }
 evptr = (struct event *)malloc(sizeof(struct event));
 evptr->evtype =  FROM_LAYER3;
 evptr->eventity = (AorB+1) % 2;
 evptr->pktptr = mypktptr;
 lastime = time;
 for (q=evlist; q!=NULL; q=q->next)
    if ((q->evtype==FROM_LAYER3 && q->eventity==evptr->eventity))
      lastime = q->evtime;
```

```c
 evptr->evtime =  lastime + 1 + 9*jimsrand();

 if (jimsrand() < corruptprob) {
    ncorrupt++;
    if ((x = jimsrand()) < .75) mypktptr->payload[0]='Z';
    else if (x < .875) mypktptr->seqnum = 999999;
    else mypktptr->acknum = 999999;
    if (TRACE>0) printf("          TOLAYER3: packet being corrupted\n");
 }
 insertevent(evptr);
}

void tolayer5(int AorB, char datasent[20]) {
  /* Do nothing */
}
```

Listing 7: prog2_gbn.c

6.3 Sample Output

The following is a curated sample from the Go-Back-N simulation output, run with high loss and corruption rates as specified. The full log has been truncated for clarity, but these snippets demonstrate the protocol's core recovery mechanisms.

```
-----  Go-Back-N Network Simulator Version 1.1 --------

Enter the number of messages to simulate: 30
Enter packet loss probability [enter 0.0 for no loss]:0.2
Enter packet corruption probability [0.0 for no corruption]:0.2
Enter average time between messages from sender's layer5 [ > 0.0]:10
Enter TRACE:2
A_init: Sender initialized.
B_init: Receiver initialized.

... (Initial packet transmissions) ...

[--- Snippet 1: Receiver correctly handles an out-of-order packet ---]
% The receiver is waiting for a packet but receives a corrupt or out-of-order one.
% It correctly discards the bad packet and resends an ACK for the last
% successfully received in-order packet (in this case, ACK=7).

EVENT time: 156343.875000,  type: 2, fromlayer3  entity: 1
  B_input: Received out-of-order/corrupt packet. Resending last good ACK=7.

... (Log continues with many similar events and corrupted ACKs) ...

[--- Snippet 2: Sender Timeout and Go-Back-N Retransmission ---]
% The sender's timer for the base of the window (stuck at base=0) expires.
% The protocol correctly "Goes Back N" and retransmits the ENTIRE window
% of unacknowledged packets, from seq=0 all the way to seq=29.

EVENT time: 156373.093750,  type: 0, timerinterrupt entity: 0
  A_timerinterrupt: TIMEOUT! Resending window from base=0
  A_timerinterrupt: Resending packet seq=0
          TOLAYER3: packet being corrupted
  A_timerinterrupt: Resending packet seq=1
          TOLAYER3: packet being lost
  A_timerinterrupt: Resending packet seq=2
          TOLAYER3: packet being lost
  A_timerinterrupt: Resending packet seq=3
  A_timerinterrupt: Resending packet seq=4
  A_timerinterrupt: Resending packet seq=5
  A_timerinterrupt: Resending packet seq=6
          TOLAYER3: packet being lost
  A_timerinterrupt: Resending packet seq=7
  A_timerinterrupt: Resending packet seq=8
  ... (retransmissions continue for the rest of the window) ...
  A_timerinterrupt: Resending packet seq=29
          TOLAYER3: packet being lost

... (The cycle of retransmissions and errors continues until termination) ...

[--- Final Output ---]

Simulator terminated at time 156553.093750
 after sending 30 msgs from layer5
```