BTN415 Lab 2

Networking Concepts: Transport Layer and Application Layer

In this lab, you will create your own methods to simulate a TCP protocol, including its three way handshake for opening connections, its four way handshake to close connections, as well as guaranteeing that the transmitted message arrives at the client without being lost or corrupted.

LEARNING OUTCOMES

Upon successful completion of this lab, you will have demonstrated the ability to:

- Understand how the TCP and UDP protocols work
- Understand basic fields in a TCP packet header
- Work with unreliable transmission mediums
- Work with random numbers

IN CLASS

Clone/Download the Visual Studio solution space from the Class Github archive (Lab2).

The provided source code introduces an unreliable channel that is used to transmit data from an UDP server, character by character, to an UDP client. You should create a struct called **tcp_packet**, and three new classes called **tcp_node**, **tcp_client**, and **tcp_server** in order to be able to send and receive information using the TCP protocol.

Your struct and new classes should have the following definitions:

```
struct tcp_packet{
    int source_port, dest_port, seq_number;
    bool acknowledge, checksum;
    string data;
};

class tcp_node {
    protected:
        string data;
        int port, current_index, data_size;
    public:
        tcp_packet rec;
        tcp_packet send_packet(int);
        int get_data_size() {return this->data_size;}
        int get_current_index() {return this->current_index;}
        void update_current_index() {this->current_index++;}
        void  request_transmission(tcp_node  *transmitter,  channel  *
wireless);
        void   send_signal(tcp_node   *dest,   tcp_node   *transmitter,
channel *wireless, string data);
        void print_data(ofstream&) {cout << this->data << endl;}
        bool three_way_handshake(tcp_node *, channel *);
        bool four_way_handshake(tcp_node *, channel *);
};

class tcp_server : public tcp_node {
    public:
        tcp_server(int, string);
};

class tcp_client : public tcp_node {
    public:
    tcp_client(int);
};
```

Also, you will need to add an overloaded **tcp_packet transmit(tcp_packet);** method to the **channel** class, so that it will have the following definition:

```
class channel{
     private:
        int prob_lost_data, prob_corrupt_data;
    public:
        channel(int, int);
        tcp_packet transmit(tcp_packet);
        udp_packet transmit(udp_packet);
};
```

The constructors for the TCP client and server classes should be identical to the ones for the UDP classes. Also, the **send_packet()** method should be nearly identical to the one for the UDP class. It should only change the return data type from **udp_packet** to **tcp_packet**. Finally, to help you, a method called **send_signal()**, which sends a packet containing user defined strings is provided at the end of the **transport.cpp** file (commented).

You should create the following methods:

tcp_node::request_transmission()

This method should do the following:

1. Sends a packet from the client to the server with "HTTP REQUEST" in its data field. *Hint: use the **send_signal()** method.*

2. Sends a packet from the server to the client with "ACK HTTP REQUEST" in its data field. *Hint: use the **send_signal()** method.*

3. Send the data in the server to the receiver, using one packet per character. *Hint: use the **send_packet()** method.*

4. Sends a packet from the client to the server with "ACK HTTP RESPONSE" in its data field. *Hint: use the **send_signal()** method.*

5. *Print "Data Successfully received" on the screen.*

tcp_node::three_way_handshake()

This method should do the following:

1. Send a packet from the client to the server with "SYN" in its data field. *Hint: use the **send_signal()** method.*

2. *Send a packet from the server to the client with "SYN/ACK" in its data field. Hint: use the **send_signal()** method.*

3. *Send a packet from the client to the server with "ACK" in its data field. Hint: use the **send_signal()** method.*

4. *Return **true**.*

tcp_node::four_way_handshake()

This method should do the following:

1. Send a packet from the client to the server with "FIN" in its data field. *Hint: use the **send_signal()** method.*

2. *Send a packet from the server to the client with "ACK/FIN" in its data field. Hint: use the **send_signal()** method.*

3. *Send a packet from the server to the client with "FIN" in its data field. Hint: use the **send_signal()** method.*

4. *Send a packet from the client to the server with "ACK/FIN" in its data field. Hint: use the **send_signal()** method.*

5. *Return **true**.*

channel::transmit(tcp_packet)

This method should be nearly identical to the **channel::transmit(udp_method)**, with two fundamental differences: In case the packet is lost by the channel, the **acknowledge** field should be set to **false**. Also, in case the packet is received, but it is corrupted, the checksum fields should be set to **false**.

TAKE HOME

Change your code so that, instead of initializing the server with “A cool message to be transmitted”, your server should get the contents of a text file. Also, change the **send_packet()** method so that it sends blocks of 64 characters per packet.

SUBMISSION REQUIREMENTS

Once you have completed your lab create and upload the following files:

- Create a single ZIP file that contains all your source code files (*.h and *.cpp)
- The output.txt file generated by the lab
- Any additional information you feel necessary for me to mark your lab