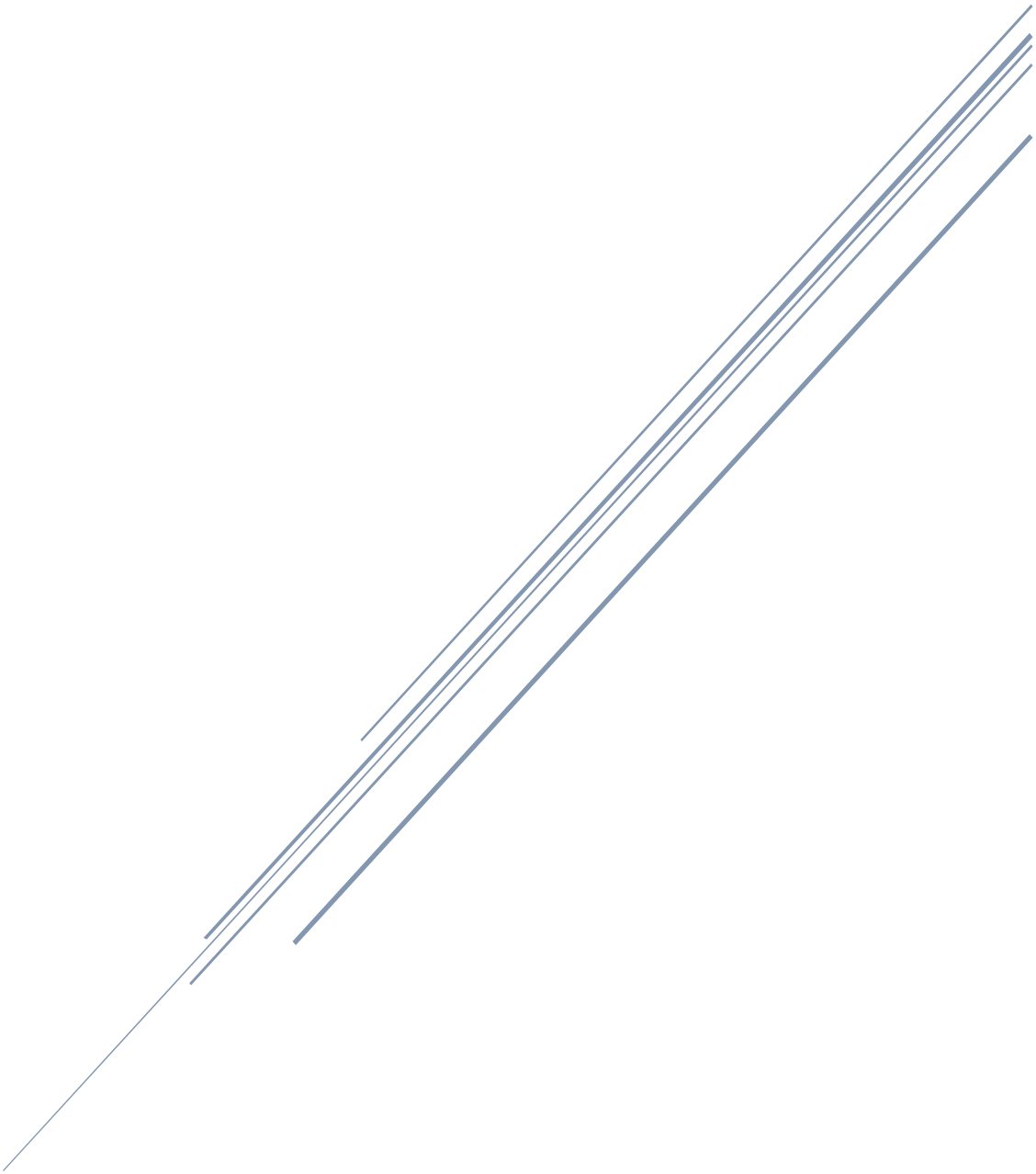
PROGRAMMING ASSIGNMENT 2

21604552 – Emre Hancı



Hacettepe University - Department of Computer Engineering
BBM203 Software Lab.

-What was the problem?

In this assignment our advisors expect that to teach us “How can I use Dynamic-Queue struct?” and “How can I use Dynamic-Stack struct?” with a simple implementation of TCP-IP protocol implementation program. This program is about clients and sending data with their connections. While sending data we need to create specific layers, which are Physical Layer, Network Layer, Transport Layer and Application Layer. In this program there are three file, maximum message size, outgoing port and incoming port as arguments.
First file is about client, in this file the first line is give us total client number and it is going like this;
“<Name> <IP> <MAC>” This is the format of a client data.
Second file is about routing data, in this file there is always (number of clients – 1) * (number of clients) data, and this is like that;
“<Receiver client> <Client you need to pass frames>” If there is no way for the sending frames between two nodes second variable of format have to been “-”
Last file is for command and process program correctly. File has 5 different type of commands which are;
<MESSAGE> in this command we need to create frames which will sent to between sender and receiver nodes, in this command there is a case like that: “If sender node has no route for sending data to receiver.” My program can handle it like that: There is no next receiver therefore the mac address of frames should be null. If mac address needs to be null then my codes not need to create frames. As a summary if message command given with two nodes which are has no route for the reach them, my program will not create frames.
<SHOW_FRAME_INFO> this command is for the peek the stack which is in the out-queue or in-queue.
<SHOW_Q_INFO> this command is for the display how many frames on the given queue.
<SEND> this command is the main of the program, after this message processed correctly, every little thing gonna be alright. When program process this command, all the frames which are waiting on the out-queue of given with commands start to moving on next clients in-queue, when next clients gets the frames, it need to check them, if it is the receiver client, it will exit from the recursive function, if not it will find a route for move all the frames, if there is a way for the moving it will change their sender and receiver mac address, after all send them to next stops in-queue. If there is no way for the moving all message have dropped.
<PRINT_LOG> This command will print all the logs of given client. If there is no log then program will not print anything.
OTHER type command should be print “This is not valid command.”

-The Data Structs, I Used

Layer;

```
struct Layer{
    char *LayerType;
    char *Area1;
    char *Area2;
    char *Area3;
};
```

I create this type struct for storage all the layers as one type, because a stack only contains one type variable (as I known). LayerType will hold the type of layer,
Area1, Area2 and Area3 will hold the information of that layer.

Stack;

```
struct Stack{
    struct Layer Layers[4];
    int top;
};
```

I create this type struct for doing stack implementation. Layer is a two-dimensional static array because total layer will never change.

Queue;

```
struct Queue{
    struct Stack Stacks;
    struct Queue *Next;
};
```

I create this type struct for doing dynamic queue implementation using linked list.

Log;

```
struct Log{
    int LogID;
    char* Message;
    int NumberOfFrames;
    int NumberOfHops;
    char* Sender;
    char* Receiver;
    char* Activity;
    char* Succes;
    char* YMDDate;
};
```

I create this type struct for making log records.

Log-Queue;

```
struct LogQueue{
    struct Log Logs;
    struct LogQueue *Next;
};
```

I create this type struct for storage logs dynamically.

Clients;

```
struct Clients {
    char* Name;
    char* Ip;
    char* Mac;
    int totalLog;
    struct LogQueue *LogsFront;
    struct LogQueue *LogsRear;
    struct Queue *IQueueFront;
    struct Queue *IQueueRear;
    struct Queue *OQueueFront;
    struct Queue *OQueueRear;
};
```

I create this type struct for hold client's information.

Routing;

```
struct Routings{
    char *Name;
    char *Destination;
    char *NextStop;
};
```

I create this type struct for hold all the routing area easily.

-The explanation of my functions

Functions getting data from files;

-int getTotalClient(char filePath);*

this function returns an integer variable which is hold the count of clients from clients file.

*-void getClientData(struct Clients *ClientsData, char* filePath);*

this function returns nothing, this function get the information about clients from clients file and assign those information to the ClientsData variable which is created, function called block.

*-void getRoutingData(struct Routings *RoutingsData, struct Clients *ClientsData, char* filePath, int totalClient);*

this function returns nothing, this function get the information about routing from routing file and assign those information to the RoutingData variable but it needs to know which routes for which clients, there it needs to know clients information and total number of clients.

-int getTotalCommands(char filePath);*

this function returns nothing, this function get the count of commands from commands file.

Functions doing process on clients data;

*-struct Clients *searchClients(struct Clients *ClientsData, char* SearchingClient, int totalClient);*

this function returns a client pointer by searching a client on the clients' data variable.

*-struct Clients searchClientsByValue(struct Clients *ClientsData, char* SearchingClient, int totalClient);*

this function returns a client by searching a client on the clients' data variable.

*-struct Clients *searchClientsByIP(struct Clients *ClientsData, char* SearchingClient, int totalClient);*

this function returns a client pointer by searching a client using its' IP-address' on the clients' data variable.

*-struct Clients *searchClientsByMac(struct Clients *ClientsData, char* SearchingClient, int totalClient);*

this function returns a client pointer by searching a client using its' Mac-address' on the clients' data variable.

*-void InitClientFrontAndRear(struct Clients *Client);*

this function initialize to null all the queue variable on the client.

Functions doing process on stack;

*-void IntoTheStack(struct Layer Layers[4],struct Stack *MessageStacks,int totalPart);*

this function inserts all layer of stack one time using push function.

*-void Push(struct Stack *MessageStacks, struct Layer Layers);*

this function inserts a layer to stack at a time.

*-void Pop(struct Stack *MessageStacks);*

this function remove top layer of the stack.

*-void Display(struct Stack *MessageStacks);*

this function print out all the layers on a stack.

*-void Init(struct Stack *MessageStacks);*

this function initializes the element "top" in the stack to -1.

*-struct Layer Peek(struct Stack *MessageStacks);*

this function returns the top layer of stack.

*-char *PeekIP(struct Stack *Stacks);*

this function returns a string which is hold the IP-address of second layer. It accesses the second layer by popping top layer of stack.

*-struct Layer PeekAppLayer(struct Stack *Stacks);*

this function returns a layer which is Application Layer of frame, it doing that using pop and push functions.

*-struct Layer PopAllStackAndReturnAppLayer(struct Stack *Stacks);*

this function popped all the layers of frame and return Application Layer the different between PeekAppLayer function this function not doing push again

*-void changeMacAdress(struct Stack *Stacks,char *Rmac,char *Nmac);*

this function change the physical layer of frame for next stop.

Functions doing process on queues;

*-void InsertToInQueue(struct Clients *Client, struct Stack *Stacks2);*

this function inserts a stack to clients' incoming queue.

*-void InsertToOutQueue(struct Clients *Client, struct Stack *Stacks2);*

this function inserts a stack to clients' outgoing queue.

*-int DisplayInQueue(struct Clients *Client);*

this function returns counts of frames which are in the incoming queue

*-int DisplayOutQueue(struct Clients *Client);*

this function returns counts of frames which are in the outgoing queue

*-struct Stack *PeekFirstFrame(struct Clients *Client);*

this function returns a pointer which is point to the given clients out-queues' first frame

*-void OutgoingToIncoming(struct Clients *Sender, struct Clients *Reciver);*

this function moves frames from out-queue to in-queue of between two nodes.

*-struct Clients *IncomingToOutgoing(struct Clients *Reciver,struct Clients *FinalClient,struct Routings *RoutingsData,struct Clients *ClientsData,int totalClient, int MaxMsgSize);*

this function moves frames from in-queue to its' out-queue.

Functions doing process on logs;

*-struct Log *CreateLog(struct Clients *Client, char* Message, int NumberOfFrames, int NumberOfHops, char* Sender, char* Receiver, char* Activity, char* Succes);*

this function returns pointer to point a log type struct which is hold all information about passing through, sent or arrived frames.

*-void InsertLog(struct Clients *Client, struct Log *Log);*

this function inserts log type struct to given clients log queue

*-void DisplayLogs(struct Clients *Client);*

this function print-out all the logs of given clients.

Other Functions;

-void printDash(int j);

this function prints-out given count of "-"

-char findRouting(struct Routings *RoutingsData,int totalClient , char* Sender, char* Receiver);*

this function find route between two nodes, if there is no route it returns "-"

*-void processingOfCommands(struct Routings *RoutingsData, struct Clients *ClientsData, char* filePathOfCommands, char* filePathOfClients, char* MaxMsgSize, char* outPort, char* inPort);*

this function gets the all information, then start process all the commands from command file

*-void MessageSendingUntilEnd(struct Clients *ClientsData, struct Clients *ReceiverClient, int totalClient, struct Routings *RoutingsData, int MaxMsgSize);*

this function send all the frames waiting on the senders' out-queue to next-stop's in-queue then check if frames arrive the receiver function stop sending, if not program firstly send all the frame out-queue of waiting clients then call itself again.