HACETTEPE UNIVERSITY DEPARTMENT OF COMPUTER ENGINEERING
BBM203

Prepared by Burak YILMAZ
Number : 21627868
E-Mail : burak040898@gmail.com
Subject : Assignment 2

MAIN GOAL

In this experiment, our aim is design a simple version of network communication between peers within a network; that is, to implement a highly simplified computer networking protocol family similar to the Internet protocol suite

PROBLEM

In this problem, we need to write a C code that we should use stacks and queues for providing the communication between computers and also use dynamic memory allocation for avoiding the wasted memory. When a computer gets the message we also have to enable its log_info.

SOLUTION

I used different kind of structures for example a client structure for the clients and for each layer I use 4 different structures and it goes like that. Each layer contains its features. I put each layer in a stack by considering the rules of stack and the order of the stacks on the Pdf. When four layer come together it becomes a frame. So I put the frames to the related clients queue by considering incoming queue or outcoming queue. And after the commands the frames move between related clients that we have to find the next hop(related client) by reading the routing file.

ALGORITHM

Firstly, I read three input files. For the clients data, I create clients and put the data inside them. After that I store the routing data in an 1D array to reach the data easily. I moved on with commands data for each command I create a function. In message command, I create the layers as a stack and put them to the outcoming queue of sender client. For show frame info and show frame are basically really similar to each other. In these functions, I check the related client incoming or outcoming queue. If their rear indicator equals to their front indicator that means queue is empty so there is nothing to show. Otherwise it shows the correct frame and what is inside that frame.

For the send command, the frames move from one client to another client by considering the neighbor of the sender client. It performs recursively. When the sender client equals to receiver client it stops. And while sending the messages for the related clients I enable their log features. For print log command ıt shows the log features. And in case of an invalid command It warns the user that you entered a wrong command.

FUNCTIONS/METHODS/STRUCTS

```
typedef struct application_layer{
    char *sender_id;
    char *receiver_id;
    char *message_chuck;
}application_layer;

typedef struct transport_layer{
    char *sender_port_number;
    char *receiver_port_number;
}transport_layer;

typedef struct network_layer{
    char *sender_ip_address;
    char *receiver_ip_address;
}network_layer;

typedef struct physical_layer{
    char *sender_mac_address;
    char *receiver_mac_address;
}physical_layer;

typedef struct log_info{
    char real_time[24];
    char *message;
    char *sender_id_1;
    char *receiver_id;
    char *activity;
    char *success;
   // int number_of_hops;
    int number_of_frames;
}log_info;

typedef struct stack{
    void *stack[4];
    int top;
}stack;
```

```
typedef struct queue{
    stack **queues;
    int rear;
    int front;
}queue;

typedef  struct client{
    char  *client_name;
    char  *client_ip;
    char  *client_address;
    queue *outgoing_buffer;
    queue *incoming_buffer;
    log_info *logInfo;
    int hops;
} client;
```

I used these structs for storing the data.

void client_initializer(client**,**int**,**char***,**int**);

-**This function is like a constructor for some values of the clients. For example rear indicator and front indicator of each client.**

void message(client,char*,char*,char*,char*,char*,stack*,int,char*);**

-This function for the command message. It creates the layers and put them into a stack and then put each frame to the related clients queue(incoming or outgoing).At the end of the function it prints each layer information.

void show_frame(client,char*,char*,int,int,int);**

-**It shows the asked frame info. If the queue is empty or user asked for wrong frame(never created before) there is nothing the show.**

void show_q(client,char*,char*,int);**

-It shows the total frames inside the outgoing queue or incoming queue. If the queue is empty there is nothing to show.

void send(client,char*,char*,int,char*,int,char*,char*);**
-It works recursively. It stops when the sender client equals to receiver client. Frames move from the outgoing queue to the incoming queue between the related hops by changing the address of the message by considering the neighbors of the clients.

void print_log(client,char*,int,char*,char*);**
-It prints the log information of asked client.

void* pop(stack *stack1);
-It pop the above layer from the current frame and returns it.

void push(stack *stack1,void*);

-It pushes the layer to the stack.

void enqueue(queue*,stack*);
-It put each frame to the related clients incoming or outgoing queue.

stack* deque(queue *item);

-It dequeue the frame from the queue.

client clients_reader(char *,int*);**

-It reads the clients data from the file and creating the clients and store their values.

char *neighbour_finder(char *,int);

-It reads the routing data and store the data inside an array and returns it.

void free_memory(client **, int);
-It frees the allocated memory.

void commands_reader(char *,int*,client,int,char*,char*,char*,int);**
-It reads the commands file and by looking the commands determine which if else statement are going to use.

int transmit_finder(int,client,int, const char*,char);**
-It finds the next hop(neighbor) of the given client and returns its index.