HACETTEPE
University

DEPARTMENT OF COMPUTER ENGINEERING

ASSIGNMENT 2
HUBBM Computer Networking

Name : Muhammed Said
Surname : Kaya
Student Number : 21627428
Section : 02

Subject : Stack, Queue, and Dynamic Memory Allocation
Programming Language : C
Advisor : Selim Yılmaz

1.INTRODUCTION / AIM

In this experiment , People are expected to gain knowledge on C language including read files, arrays , stack and queue implementation , recursion and dynamic memory allocation.

2.PROBLEM

Design a simple version of network communication between peers within a network.

3.HOW TO SOLVE PROBLEM (Summary of Code)

1. structs.h

There are structures named client and frame . Every client have two frame queues for income and outgoing.

2. main.c

First of all , The clients.dat is opened in createClients function and according to number of client , a client array ,that named AllClients, is malloced dynamically and informations(ID , IP , Mac) are assigned to them.

For second part , every client has a two dimensional array is named routingTablesArray. After that routingtables.dat is opened in createRoutingTables function and destinationClient and it's neighbor are assgined to client's routingTablesArray.

For third part , There is a three dimensional array is named commands and commands.dat is opened in createCommands in function. Then , According to commands size (like in example 8) commands is malloced .After that with fgets Every column are read and step by step malloc and realloc are done for commands. Also for example commands[1][0] has an information about how many word are there in that column. This number is used in the rest of code.

The last one for main function**,** There is a function name workSimulator and in this function according to comparision of commands , some functions works (createRightRoute , Message , createQueue and send). Also there are functions for print part (printMessage , printShowFrameInfo)

3. workSimulator.c

- **createRightRoute.c** : This function works by recursive . Firstly, when the command is like that MESSAGE C E …. , Clients named backClient and destinationClient . in backClient's routingTable destinationClient is found and it is checked with it's neighbor .If they are not equal for example in C's routing table E's neighbor is B . So anymore backClient is B and destinationClient is E. This function Works until the in backClient's routing table , destinationClient and destinationClient's neighbor are equal. Also returns an array includes client. Like C B D E.

- **Message.c** : This function by using commands array that is three dimensional , strip the "#" with stripTheMessage function and When the commands[i][1] is MESSAGE, it collects char arrays to a a single char array with for loop.End of this function returns an char array like char *message.

- **createQueues.c** : In this function , Firstly , with LenOfMessage and sizeDivision(argv[4]) , numberOfLoop is getting and C's outQueue is malloced according to this number and in for loop , every frame has three dimensional char array and it is malloced by four then information like ID , Message .. assigned to layers with using push function and frame's tailOfLayer is increases . After the assigning layers . That frame with using a function named Enqueue is assigned to C's OutQueue of tailOfOutArray . Before Enqueue tailOfOutArray increases. Then prints the information. Lastly , a log is created . (Every Client has an logArray but it is dynamic. I mean they are malloced on runtime). C's logArray is malloced by 1 and that log assigned to this array and numberofLogs increases.

- **send.c** : This function works by recursive . There are clients backClient , nextClient ,destinationClient as arguments. Funtion works until the destinationClient's ID is equal to nextClient'ID or for fail situation I mean there is no connect between C and E . There is three condition first one is equal , second one is fail situation , third one is function works again.

  For; Successfully comunicate (Like C B D E)

  Firstly , nextClient's inQueue is malloced.

  Then helper_Send_Out_To_In (auxiliary function) works and step by step frames in backClient's outQueue are coppied by using with a function named copyFrame. Also helper_Send_Out_To_In function works by recursive until the headOfOutArray is equal to tailOfOutArray.Then Add_Log_Out_To_In function works and nextClient's logArray is malloced and a LogEntry is created. Also when these funtion works step by step frame's in backClient's outQueue are freed.

  Then helper_Send_In_To_Out (auxiliary function) works and step by step frames nextClient's inQueue are coppied by using with a function named copyFrame. Also helper_Send_In_To_Out function works by recursive until the headOfInArray is equal to tailOfInArray.Then Add_Log_In_To_Out funtion works and LogEntry is created and assigned. Also when these function works step by step frame's in nextClient's inQueue are freed.
  After that if destinationClient's ID is equal to nextClient's ID , again nextClient's InQueue is malloced . Then Adding a log and helper_Send_Out_To_In funtion works and frames are in E's InQueue right now .

  For fail situation; (Like C B D)

  Exactly between C and B frames are send then number of RoutingClientsArray is going to be same with an counter . So the situation will be failed. Then D's inQueue is malloced and adding a log with Add_Log_Out_To_In . Then helper_Send_Out_To_In function works . Because of there is no connection between D and E , frames are in D's inQueue right now.Lastly Add_Log_Fail function works.