REȚELE DE CALCULATOARE
PROIECT VIRTUAL STUDY GROUP
RAPORT TEHNIC

Medeleanu Daria 2A2

Facultatea de Informatica Iași

1 Introducere

1.1 Viziunea generală și obiectivele

Proiectul Virtual Study Group își propune exemplificarea unui server TCP/IP concurent însoțit de o interfață grafică ușor de folosit în care utilizatorii vor avea la dispoziție cele 3 funcționalități principale: colaborarea la proiecte, partajarea de materiale, conversația prin mesaje în timp real. Utilizatorul ce a completat procesul de login (respectiv de register și login) va avea pus la dispoziție o pagină de start unde va putea vizualiza grupurile de utilizatori disponibile, toti utilizatorii logati, de asemenea având posibilitatea și de a crea un nou grup, de a intra în chatul unui grup din cele disponibile, de a-si schimba parola contului si de a-și da logout. Datele privind utilizatorii, grupurile din care fac parte dar și despre fisiere vor fi stocate într-o bază de date.

2 Tehnologii Aplicate

Proiectul în versiunea curentă este implementat cu ajutorul limbajului C. Pentru realizarea comunicării dintre server și client am folosit TCP, serverul creează pentru fiecare client un thread separat în care este citit inputul de la clienti. Am ales protocolul de comunicare TCP care este orientat-conexiune deoarece vizeaza oferirea calității maxime ale serviciilor, fără pierderea informațiilor, controlând fluxul de date fiind astfel potrivit în cadrul unei aplicații de tip chat și de transmitere a fișierelor. UDP-ul este potrivit pentru aplicațiile ce au nevoie de transmitere video/audio, viteza fiind cea care primează. Am ales folosirea thread-urilor (programe aflate în execuție fără spațiu de adresă proprie) deoarece oferă eficiență prin mecanismul de funcționare și anume execuția paralelă a funcțiilor definite. Aplicația afiseaza comenzile disponibile in momentul logarii. În plus, pentru stocarea datelor cu privire la utilizatori, grupuri și fișiere transmise voi folosi SQLite3 deoarece procesul de integrare în aplicație nu este unul dificil, iar aceasta platformă oferă toate funcționalitățile (interogări) necesare pentru aplicația de față.

3 Structura Aplicației

Serverul permite conectarea mai multor clienți simultan. Pentru fiecare client acceptat, se va deschide un nou thread ce va asculta cu ajutorul primitivei read inputul transmis de către client și îi va face display în cadrul serverului pentru a vedea input-ul procesat, comunicarea fiind asigurată cu ajutorul socket-urilor. Comenzile disponibile dintre pentru realizarea funcționalităților sunt: REGISTER, LOGIN, LOGOUT, CHANGE-PASSWORD, GET-LOGGED-USERS, SEE-ALL-GROUPS, JOIN-GROUP, SEE-FILES, UPLOAD-FILE, DOWNLOAD-FILE, CREATE-GROUP, EXIT-GROUP.

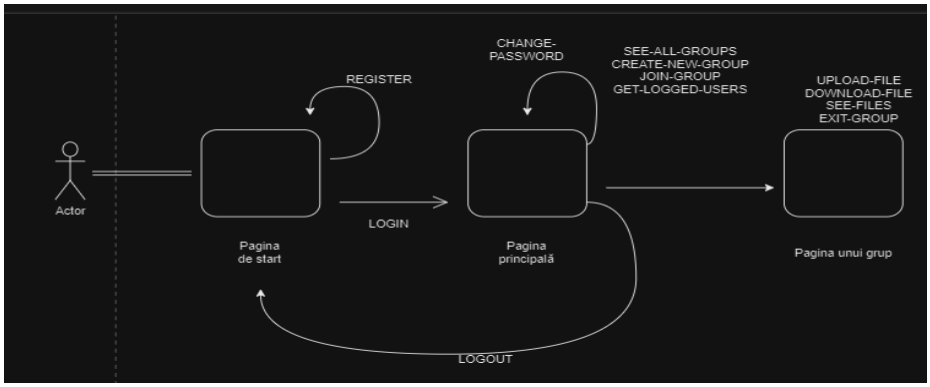

Fig. 1. Structura aplicației

4 Aspecte de Implementare

Descrierea protocolului de comunicare:
1. Register - înregistrarea unui utilizator nou
2. Login - Intrarea în cont a unui utilizator existent
3. Change-Password - Schimbarea parolei, doar în cazul în care utilizatorul este logat
4. Logout - Ieșire din cont
5. See-All-Groups - Display la toate grupurile din aplicație
6. Create-Group - Crearea unui grup nou
7. Get-Logged-Users - Display la utilizatorii online din grup
8. See-Files - Display la fișierele existente pe grup
9. Upload-File - Încărcarea unui fișier pe server
10. Join-Group - Intrare in fereastra grupului selectat
11. Download-File - descarcarea fisierului selectat
12. Exit-Group - revenire la fereastra de dupa logare

```
// acceptarea clientilor, creeaza un thread nou pentru fiecare client
        while(true){
                struct sockaddr_in clientAddress;
                int clientAddressSize = sizeof(clientAddress);
                int clientSocketFD = accept(sd,(struct sockaddr *)&clientAddress

                struct ClientiAcceptati* clientSocket = (struct ClientiAcceptati
                clientSocket->address = clientAddress;
                clientSocket->acceptatiSocketFD = clientSocketFD;
                clientSocket->acceptatSucces=clientSocketFD>0;
                clientSocket->idClient=nrClienti;
                strcpy(clientSocket->username,"");
        strcpy(clientSocket->currentWindow,"main");
                clienti[nrClienti++]=*clientSocket;

                pthread_t tid;
                pthread_create(&tid, NULL, primitSiPrintatData, clientSocket->ac
        }

//functie ce ce ocupa cu gestionarea inputului
void primitSiPrintatData(int socketFD){
        char buffer[1024]="";
        int bytesRead = 0;
        while(1){
                if( (bytesRead = read(socketFD, buffer, sizeof(buffer)-1)) == -1
                        printf("[server]Eroare la citire din client\n");
                        break;
                }
                if(bytesRead>0){
                        buffer[bytesRead]='\0';
                        printf("[server]%s\n",buffer);
                        fflush(stdout);
                        for(int i=0;i<nrClienti;i++)
                                if(clienti[i].acceptatiSocketFD==socketFD){
            printf("[server] window %s\n", clienti[i].currentWindow);
            if(strcmp(clienti[i].currentWindow,"main")==0 || strcmp(clie
                printf("[server] a detectat comanda\n");
                //displayComenziDisponibile(clienti[i].currentWindow, so
                detectareComenzi(buffer, clienti[i].idClient, socketFD);
            } else { //sunt in grup
                printf("[server] a detectat mesaj\n");
                if(strcmp(buffer,"upload-file\n") == 0 || strcmp(buffer,
                    // s-a scris o comanda in cadrul grupului
                    printf("s-a scris comanda in cadrul grupului\n");
                    //displayComenziDisponibile(clienti[i].currentWindow
```

```
                                detectareComenziDinGrup(buffer, clienti[i].idClient,
                            }
                            if((strcmp(buffer,"exit-group\n") == 0) || strcmp(buffer
                                printf("Ai scris exit-group\n");
                                if(strcmp(buffer,"exit-group\n")!=0){
                                    clienti_logati[nrClientiLogati] = clienti[nrClie
                                    trimiteMesajulPrimitLaCeilalti Clienti(buffer, so
                                } else {
                                    strcpy(clienti[clienti[i].idClient].currentWindo
                                }
                            }
                        }
                                    }
                    }
                    if(bytesRead<0){
                            break;
                    }
            }
            close(socketFD);
}


//functie ce detecteaza comenzi
int detectareComenziDinGrup(char* buffer, int clientIndex, int socketFD, char* g
    printf("Buffer in detectareComenziDingrup este %s\n", buffer);
    if((strcmp(buffer,"upload-file\n")==0)){
        printf("Ai scris upload-file\n");
        uploadFile(groupName, socketFD);
    }else if((strcmp(buffer,"see-files\n")==0)){
        printf("Ai scris see-files\n");
        seeFilesFromGroup(groupName,socketFD);
    }else if((strcmp(buffer,"download-file\n")==0)){
        printf("Ai scris download-file\n");
        downloadFile(groupName,socketFD);
    }
}
```

5 Concluzii

Pentru îmbunătățirea soluției as fi putut adauga o structură în care fiecare utilizator poate să își propune obiective pe care le poate marca drept efectuate sau în curs de efectuare.

6 Referințe Bibliografice

https://profs.info.uaic.ro/~computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf
https://profs.info.uaic.ro/~computernetworks/files/4rc_NivelulTransport_Ro.pdf
https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c