VirtualSoc

Lazurcă Samuel-Ionuț B3

Facultatea de Informatică Iași

1 Introducere

VirtualSoc este un proiect de tip server/client ce presupune simularea unei rețele sociale.

Utilizatorii vor fi de două tipuri : utilizator obișnuit si administrator.

Pentru a avea acces la anumite funcții trebuie ca utilizatorul să fie logat.

Utilizatorii logați vor avea un profil și o pagină proprie ce va conține postările lor și care va putea fi publică sau privată. Această pagină va putea fi vizualizată de ceilalți utilizatori.

Utilizatorii nelogați nu vor avea pagina proprie și singurul lucru pe care aceștia îl pot face e să vizualizeze doar postările publice ale celorlalți.

Utilizatorii logați vor putea publica diferite postări pe pagina lor și aceste postări pot fi văzute de un grup de utilizatori specificat de autorul postării.

Utilizatorii logați vor putea comunica intre ei, această comunicare fiind de 2 tipuri : comunicare între doi utilizatori și comunicare între mai mulți utilizatori(grup).

2 Tehnologii folosite

În implementarea acestui proiect se va folosi la nivelul transport protocolul TCP deoarece nu ne putem permite ca informațiile să nu ajungă la client iar TCP ne oferă această siguranță deoarece este mai sigur decât UDP din cauza 3-way-shakehanding-ului.

Limbajul de programare folosit este C++ deoarece oferă posibilitatea unei organizări mai bune a codului în clase si obiecte.

Deasemenea se utilizează o bază de data SQLite[1].

3 Arhitectura aplicației

Aplicația va conține un singur server ce va suporta cererile mai multor clienți. Serverul va comunica deasemenea cu o bază de date SQLite[5].

3.1 Arhitectura clientului

Clientul va primi input-ul de la tastatură sub forma unei comenzi astfel că există necesitatea proiectării unor comenzi scurte și neambigue.

Clientul se va conecta printr-un socket la server și va trimite server-ului comanda primită.

În același timp clientul asteaptă răspuns din partea server-ului. Însă ceea ce citește clientul de la server nu este doar răspunsul la o singură comandă ci din când în când clientul va putea primi actualizări ale datelor de la server (cum ar fi ultimele mesaje dintr-o conversație, ultimele postări, cereri de prietenie etc.)

Prin urmare, pentru a nu condiționa ca primirea unui mesaj de la server să se faca doar atunci când scriem explicit o comandă vom folosi două thread-uri[4].

Într-un thread clientul citește de la tastatură și trimite serverului comanda și în celălalt thread clientul citește ce primește de la server și pune în practică acțiunea dictată de server.

Acest lucru este foarte folositor mai ales atunci când clienții comunică între ei și trebuie ca fiecare client să aibă posibilitatea să trimită și să primească oricât de multe mesaje și în orice ordine.

Comenzile sunt următoarele:

show posts from $< username >$ -afișează postarile lui username
create_account $< username >< password >< pagetype >< usertype >$
login $< username >< password >$
logout
send friend request $< type\_of\_friend >< username >$
my posts -afișeaza conținutul paginii personale
friend_requests
accept friend request from $< username >$
my friends list
post $< content >< type\_of\_post >$
write $< username/groupname >$
enter group $< groupname >$
exit_conv
create_group $< groupname >$
exit_group $< groupname >$
add_member $< username >$
quit

Pseudocod folosit ca exemplu pentru a arăta funcționarea clientului :

```
void* readFromConsoleSendToServer(void* ptr)
{
        ...
        while(true)
        {
                cin>>cmd;
                if(cmd == "exit") break;
                write(sd, cmd, strlen(cmd));
        }
}
void readFromServer()
```

```
{
        ...
        while(true)
        {
                read(sd, raspuns, 100);
                process(raspuns);
        }
}
int main()
{
        ...
        pthread_create(& thread1,NULL,readFromConsoleSendToServer,NULL);
        readFromServer();
}
```

3.2 Arhitectura serverului

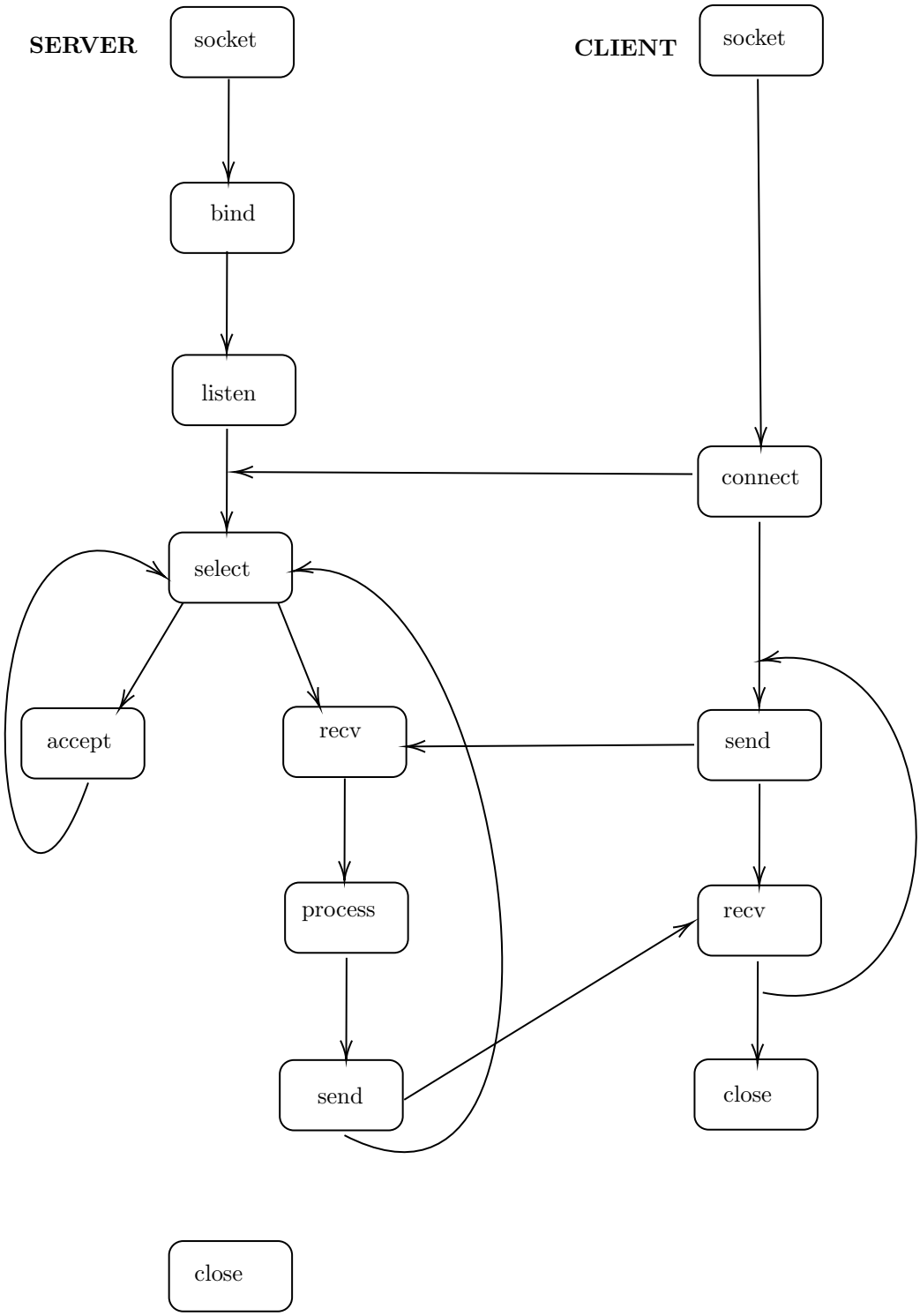
Serverul trebuie să fie concurent. Acest lucru va fi implementat prin multiplexare I/O cu select()[3].

Deasemenea serverul va trebui să interogeze, insereze sau să actualizeze o bază de date ce va conține diverse informații despre clienti cum ar fi : username, parola, relații de prietenie, postări, conversații etc.

În momentul în care un client se conectează, serverul va genera un obiect dintr-o clasă denumită Client ce va conține toate informațiile necesare despre acel client cum ar fi: username, dacă e logat în acel moment etc. dar și metode care se vor apela în urma parsării comenzilor primite de la client (pentru fiecare comandă există cel puțin o metodă în clasa Client).

Aceste obiecte vor fi ținute într-un container (ex: std::vector) iar atunci când un client se va deconecta, va fi eliminat și din container.

Modelul TCP [2] adaptat la aplicație este:


SERVER
socket
bind
listen
select
accept
recv
process
send
close
CLIENT
socket
connect
send
recv
close

3.3 Diagrama aplicației

Diagrama aplicației ar putea fi reprezentată în felul următor:

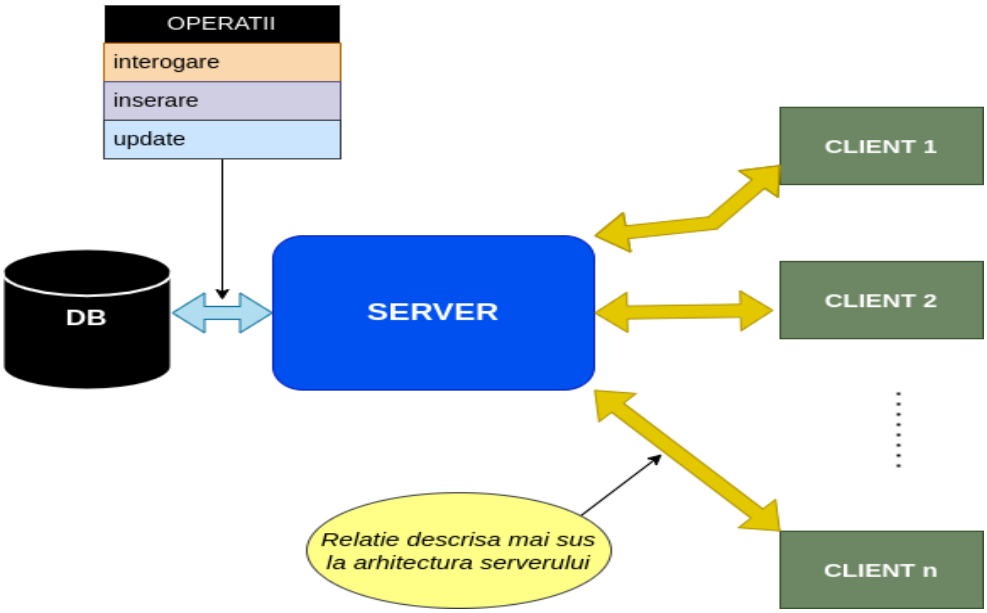


Fig. 1. Diagrama generalizată a aplicației.

4 Detalii de implementare

După cum am scris și mai sus clientul va utiliza două thread-uri (programul principal și un thread creat cu pthread_create()) iar serverul concurent va utiliza multiplexarea I/O cu select().

Implementarea constă în :

Citirea din consolă a unei comenzi (cât timp acea comanda nu e ”exit”) și trimiterea ei de către client server-ului.

Server-ul va parsa comanda și dacă e nevoie va lucra cu o bază de date în următoarele cazuri:

- informațiile despre fiecare utilizator în parte vor fi ținute într-o tabelă și acestea vor fi : id-ul, username, parola, tip utilizator si tip pagină (publică /privată);
- în momentul în care clientul cere o listă a prietenilor, se va interoga baza de date și se va obține lista care va fi trimisă apoi clientului;

- în momentul în care se face login pentru a verifica dacă perechea (username, password) se găsește în tabela de useri sau în momentul în care se face crearea unui cont nou când se verifică ca să nu mai existe deja un user cu același nume și apoi inserarea perechii (username, password) plus un id unic de identificare;
- în momentul când clientul deschide o conversație și serverul va interoga baza de date si va trimite clientului ultimile 10-15 mesaje și chiar atunci când cineva scrie un nou mesaj, acel mesaj va fi trimis destinatarului/destinatarilor dar va fi memorat și în baza de date;
- când un client postează, acea postare va fi deasemenea ținută intr-o bază de date împreună cu alte detalii despre ea cum ar fi tipurile de utilizatori care o pot vedea (toti utilizatorii, toti prietenii, doar rudele etc.);
- când un client va dori să vizualizeze postările unui alt utilizator serverul va trimite înapoi clientului doar acele postări pentru care are permisiuni (de exemplu un utilizator nu va putea vedea postările unui alt utilizator care nu îi este prieten și care sunt destinate doar prietenilor acestuia);

Tabelele din baza de date SQLite în care vom păstra informații despre utilizatori sunt:

Table 1. Tabela utilizatorilor

USERS				
ID	USERNAME	PASSWORD	PAGE_TYPE	USER_TYPE

Table 2. Tabela postărilor

POSTS			
ID_USER	CONTENT	PTYPE	NR

Table 3. Tabela relațiilor de prietenie

FRIENDS		
ID1	ID2	FRIENDSHIP_TYPE

Table 4. Tabela conversațiilor dintre 2 persoane

CONVERSATIONS		
CONV_ID	ID1	ID2

Table 5. Tabela mesajelor din fiecare conversație și un număr de ordine

MESSAGES			
CONV_ID	ID_SENDER	MESSAGGE	NR

Table 6. Tabela conversațiilor dintre mai multe persoane(grupuri)

GROUPS		
GROUP_ID	GROUPNAME	ADMIN_ID

Table 7. Tabela participanților din grupuri

GROUP_MEMBERS	
GROUP_ID	MEMBER_ID

Table 8. Tabela mesajelor dintr-un grup și numărul lor de ordine

GROUP_MESSAGES			
GROUP_ID	MEMBER_ID	CONTENT	NR

5 Concluzii

În concluzie, VirtualSoc este o aplicație ce simulează o rețea de socializare obișnuită prin intermediul comenzilor.

VirtualSoc este un proiect ce îmbină noțiuni de rețele de calculatoare și baze de date.

Îmbunătățirile care se pot face sunt în ceea ce privește gestionarea tabelelor din baza de date dar și adăugarea unor comenzi in plus.

References

1. https://sqlite.org/cintro.html
2. https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
3. https://profs.info.uaic.ro/ georgiana.calancea/rc-home.html
4. https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html
5. https://www.tutorialspoint.com/sqlite/index.html