OfflineMessenger

Baltag Bianca-Teodora[1]

Facultatea de Informatica, Universitatea "Alexandru Ioan Cuza Iasi"

1 Introducere

Acest raport tehnic este conceput pentru a oferi o viziune clară și riguroasă asupra conceptului de Mesagerie Offline, o aplicație client/server care să permită schimbul de mesaje între utilizatori indiferent de starea lor(online/offline). Aplicația recunoaște și implementează complet conceptele de LOGIN, LOGOUT, EXIT, RESET-PASSWORD, SIGNUP, dar și funcționalități inedite, precum: MESSAGE – ce oferă posibilitatea de a trimite mesaje și de a răspunde la acestea(reply), HISTORY – oferă istoricul mesajelor utilizatorului, conectat, cu oricare dintre cei aflați în baza de date a aplicației. Soluția propusă de mine este construită pe baza unui server TCP concurent care deservește fiecare utilizator prin crearea unui thread dedicat fiecărui client.

2 Tehnologii Aplicate

Soluția propusă de mine pentru realizarea proiectului OfflineMessanger constă în implementarea unui aplicații de tipul client/server care permite utilizatorilor online/offline să comunice prin intermediul mesajelor. Tipul de server folosit este TCP concurent, care creează un thread pentru fiecare client nou conectat la server. Această implementare permite server-ului să deservească simultan mai mulți clienți, asigurându-se totodată că orice cerere făcută de aceștia la server va fi tratată în mod concurent și nu va fi blocată sau întârziată de cererile altor ultilizatori. Am optat pentru folosirea unui server de tip TCP datorită siguranței pe care o oferă din punct de vedere al transmiterii informației, dar și a ordinii în care aceasta este transmisă. Serverul de tip TCP, adaptându-și viteză de trimitire a datelor pentru a nu supraîncărca rețeaua, iar dacă un pachet de date este pierdut, ulterior va fi retrimis. Persistența datelor: mesajele trimise către utilizatorii offline sunt stocate temporar în baza de date până la reconectarea acestora. Acest mecanism asigură continuitatea serviciului și integritatea datelor.

3 Structura Aplicației

Aplicația OfflineMessanger este bazată pe o arhitectură client-server, concepută după o organizare logică cât mai eficientă și după cunoștiintele aprofundate de-alungul semestrului în cadrului cursului "Rețele de Calculatoare".

Serverul reprezintă punctul principal, central de gestionarea a informației, comezilor, mesajelor. Acesta este responsabil de deservirea clienților într-un mod concurent, procesând comezi, stocând și transmițând mesajele offline/online dintre utilizatorii aplicației.

Clientul este responsabil de interacțiunea directă cu utilizatorul, trimițând comezi serverului și așteptând mesaje, informații din partea acestuia.

Așa cum am precizat în secțiunea anterioară, protocolul TCP(Transmission Control Protocol) a fost alegerea mea, oferind o multitudine de beneficii și siguranță aplicației mele, precum: mesajele sunt livrate în ordinea corectă, orice pachet pierdut este retrimis, iar conexiunea TCP rămâne activă până la finalul sesiunii.

Folosind concurență prin thread-uri serverul este capabil să gestioneze mai mulți clienți simultan , fiecare thread fiind atribuit unui client conectat. Astfel, reușind izolarea procesării comezilor pentru fiecare utilizator, astfel încât operațiunile utilizatorilor să nu interfereze cu ale altora.

În aplicația OfflineMessanger, stocarea datelor despre: utilizatori(datele de autentificare, flag-ul online/offline), mesaje(mesajul în sine, utilizatorul care a expediat mesajul, destinatarul acestuia și flag-ul sau la momentul respectiv), istoric(toate mesajele dintre utlizatorii aplicației) este realizată de o bază de date relațională.

Odată ce un client se conectează la server, acesta are posbilitatea de a trimite comenzi serverului, precum: LOGIN, LOGOUT, MESSAGE, HISTORY, EXIT, RESET-PASSWORD, SIGNUP.

Logarea se realizează cu: LOGIN <username> <password>, comandă și numele de utilizator sunt trimise serverului așteptând validarea.

Comanda LOGOUT trimisă de utilizator către server este validată doar dacă aceasta este logat într-un cont de utilizator.

Comanda HISTORY <username1> returnează utilizatorului istoricul convorbirilor cu <username1>

Comanda MESSAGE <username receiver> <message> este o comandă funcțională, ce reprezintă comunicarea a doi utilizatori prin intermediul serverului, indiferent dacă unul dintre aceștia este online sau offline.

Comanda EXIT realizează închiderea conexiunii cu serverul și curățarea resurselor asociate clientului.

Comanda RESET-PASSWORD <new password> <confirm newpassword>, inlociueste in baza de date vechea parola cu cea noua, comanda functioneaza doar daca utilizatorul este conectat sub numele de utilizator.

Comanda SIGNUP <username> <password> <confirm password>, trimite spre validare username-ul astfel incat sa fie unic in baza de date, dupa validare stocheaza noul utilizator in baza de date.

În orice caz(comandă valida sau nu, username valid sau nu, alte erori posibile) serverul comunică explicit clientului validarea sau posibilă eroare apărută pe parcursul procesului de utilizare, acest lucru fiind valabil pentru oricare dintre comenzile enumarate anterior.

Această strucrura modulară urmărește extinderea proiectului fără dificultăți din punct de vedere al arhitecturii de baza.

4 Aspecte de Implementare

În următoarele 3 figuri sunt prezentate unele dintre cele mai importante și semnificative secvențe de cod, utilizate pentru realizarea aplicației OfflineMessanger.

OfflineMessanger, are desigur și un protocol la nivelul aplicației, prezentat după cum urmează:

Autentificarea, reprezentată de comandă LOGIN <username> trimisă de către client. Această comandă este trimisă serverului, alături de username-ul propus de client.Serverul interpretează comanada și verifică existența acestui username în baza de date aferentă aplicației. Dacă username-ul nu este valid, sau deja un alt client s-a conectat sub numele acesta de utilizator, serverul transimite clientului un mesaj de eroare cu privire la problema apărută. În schimb, dacă clientul nu este daja logat și dacă username-ul trimis spre validare este conform, atunci serverul trimite un mesaj de confirmare al conectării și setează flagul utilizatorului din baza de date 1/online.

Trimiterea mesajelor, reprezentată de comandă MESSAGE <username receptor> <mesaj propriu-zis>, funcționează sub protocolul: verifică dacă emițătorul este conectat(obligatoriu), dacă posibilul receptor are un username valid și dacă este online/offline.

```c
pthread_t fire_thread[100]; // Thread-uri pentru clienti

while (1) {
    int client;
    dateThread *date_thread = (dateThread *)malloc(sizeof(dateThread));
    int lungime_adresa = sizeof(din);

    if ((client = accept(sd, (struct sockaddr *)&din, &lungime_adresa)) < 0) {
        perror("[server] Eroare la acceptarea conexiunii");
        continue;
    }

    date_thread->idThread = i++;
    date_thread->descriptor_client = client;

    // Creare thread pentru client
    pthread_create(&fire_thread[i], NULL, &proceseaza_client, date_thread);
}
```

Fig. 1. Conexiunea client/server și gestionarea concurenței cu ajutorul thread-urilor. Serverul primește noi clienți și creează un thread pentru fiecare conexiune, accepta o conexiune de la un client și returnează descriptorul socket-ului (client).

```c
if (strncmp(buffer, "LOGIN", 5) == 0) {
    char nume_utilizator[LUNGIME_UTILIZATOR];
    sscanf(buffer, "LOGIN %s", nume_utilizator);

    pthread_mutex_lock(&mutex_utilizatori);
    int index_utilizator = cautaUtilizator(nume_utilizator);
    if (index_utilizator != -1 && utilizatori[index_utilizator].esteOnline == 0) {
        utilizatori[index_utilizator].descriptor_socket = date_thread.descriptor_client;  // Asociere socket
        utilizatori[index_utilizator].esteOnline = 1; // Marcare ca online
        date_thread.esteAutentificat = 1;
        snprintf(raspuns, sizeof(raspuns), "Bine ai venit, %s! Te-ai autentificat cu succes.\n", nume_utiliza
    } else {
        snprintf(raspuns, sizeof(raspuns), "Utilizatorul nu a fost găsit sau este deja autentificat.\n");
    }
    pthread_mutex_unlock(&mutex_utilizatori);
}
```

Fig. 2. Autentificare și gestionarea utilizatorilor.Dacă utilizatorul este găsit și nu este deja conectat, serverul:actualizează descriptorul socket, marchează utilizatorul ca fiind online, trimite un mesaj de confirmare către client.

```
if (strncmp(buffer, "MESSAGE", 7) == 0) {
    if (date_thread.esteAutentificat == 0) {
        snprintf(raspuns, sizeof(raspuns), "Trebuie să te autentifici mai întâi.\n");
    } else {
        char destinatar[LUNGIME_UTILIZATOR], continut_mesaj[LUNGIME_MESAJ];
        sscanf(buffer, "MESSAGE %s %[^\n]", destinatar, continut_mesaj);

        pthread_mutex_lock(&mutex_utilizatori);
        int index_destinatar = cautaUtilizator(destinatar);
        if (index_destinatar != -1 && utilizatori[index_destinatar].esteOnline) {
            char mesaj_complet[1024];
            snprintf(mesaj_complet, sizeof(mesaj_complet), "[Mesaj de la %s]: %s\n",
                     utilizatori[date_thread.index_utilizator].nume_utilizator, continut_mesaj);

            write(utilizatori[index_destinatar].descriptor_socket, mesaj_complet, strlen(mesaj_complet));
            snprintf(raspuns, sizeof(raspuns), "Mesajul a fost trimis către %s.\n", destinatar);
        } else {
            adaugaMesajCoada(index_destinatar, utilizatori[date_thread.index_utilizator].nume_utilizator,
            snprintf(raspuns, sizeof(raspuns), "Mesajul a fost stocat pentru utilizatorul %s.\n", destina
        }
        pthread_mutex_unlock(&mutex_utilizatori);
    }
}
```

Fig. 3. Trimiterea Mesajelor. Verifică dacă utilizatorul este autentificat, verifică dacă destinatarul este online, dacă este online/1, mesajul este livrat imediat, dacă este offline/0, mesajul este stocat într-o coadă de mesaje pentru livrare ulterioară.

Dacă este online mesajul este trimis imediat destinatarului, în caz contrar mesajul este stocat în baza de date până la următoare sa conectarea, când îl va și restitui din baza de date.

Accesarea istoricului, opțiune reprezentată de comandă HISTORY <username2> are următorul protocol: serverul verifică mai întâi dacă clientul este conectat sub numele unui utilizator al aplicației, în caz afirmativ verifică și dacă username2 este valid, adică apare în baza de date a aplicației. Dacă cele două verificări sunt validate, serverul va trimite utilizatorului întreg istoricul conversației cu username2, stocat în baza de date.

Delogarea utilizatorului, reprezentată de comandă LOGOUT prezintă în aplicația mea următorul protocol: server-ul verifică dacă acesta este logat, în caz afirmativ modifică în baza de date flag-ul utlizatorului în 0/offline, delogandu-l. În cazul în care clientul este deja offline va primi o notificare din partea server-ului în care i se va aduce la cunoștiință faptul că acesta este deja delogat.

Ieșirea, reprezentată de comandă EXIT, închide conexiunea clientului cu serverul. Dacă clietul a fost conectat și comandă aplicată, flag-ul din baza de date al utilizatorului din care s-a făcut comandă exit va fi setat 0/offline.

Functionalitatea de sign-up verifica mai intai daca username-ul dorit este unic printre cele deja existente, apoi valideaza parola, daca aceasta contine numarul de caractere dorit, dupa care stoceaza noul utilizator in baza de date users.

Functionalitatea de resetare a parolei poate fi realizata doar daca clientul este conectat sub numele unui anumit utilizator, verifica daca parola dorita si confirmarea acesteia sunt identice, dupa care stocheaza noua parola in baza de date, in locul celei vechi.

Unele din numeroasele scenarii posibile, aferente aplicației OfflineMessanger sunt:

Conectarea și trimiterea unui mesaj, un utilizator se autentifica cu comandă: LOGIN MARIA, comandă, fiind validată de către server. Apoi dorește să trimită un anumit mesaj lui ION, folosind comandă MESSAGE ION bună Ion.Ce faci?. Dacă ION este offline, mesajul va fi stocat în baza de dată, în caz contrar ION va restitui mesajul trimis de către MARIA ([from MARIA] : bună Ion.Ce faci?).

Un utilizator deja conectat, ANA, dorește să acceseze istoricul convorbirilor cu VASILE. Aceasta va folosi comandă HISTORY VASILE, reușind astfel să acceseze toată convorbirea cu utlizatorul VASILE.

Primirea unui mesaj offline, IONUȚ îi trimite un mesaj RAMONEI, însă această nu a fost conectată la momentul respectiv. Ea, după ceva timp se va conecta, primind o notificare imediat cu mesajul primit de la IONUȚ, putând să îi și răspundă acestuia cu ajutorul funcției de reply implementate. Vrând mai apoi să se deconecteze, folosind comandă LOGOUT.

5 Concluzii

Realizarea acestui poiect , OfflineMessanger, a fost o experiență benefică pentru mine, reușind să implementez conceptele teoretice învățate de-alungul cursurilor, dar și să le înțeleg într-un mod mai profund. Pe lângă cerințele obligatorii ale proiectului, îmi doresc să aduc în completare: o interfață grafică, dar și facilități de utilizare cât mai apropiate unei aplicații pe care o utilizăm în viață noastră de zi cu zi, precum: autentificare se va face pe baza de <username> și <password>, având posibilitatea de a-ți reseta parola, de a trimite simultan mai multor utilizatori același mesaj, dar și de a crea statusuri ale utilizatorilor, cu vedere la posibilele relații unii față de ceilalți (ex. necunoscuți, prieteni , parteneri), crearea unor chat-uri unde pot participa mai mulți utilizatori ai aplicației.

References

1. Curs 7, Retele de Calculatoare , Exemplu de server TCP concurent care deserveste clientii prin crearea unui thread pentru fiecare client. Asteapta un numar de la clienti si intoarce clientilor numarul incrementat. Intoarce corect identificatorul din program al thread-ului. https://edu.info.uaic.ro/computer-networks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
2. Socket programming in C https://www.binarytides.com/socket-programming-c-linux-tutorial/
3. Socket Programming in C/C++: Handling multiple clients on server https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/
4. TCP Server-Client implementation in C https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/
5. Springer. LNCS Guidelines. Disponibil la: Springer LNCS Guidelines.