MyFileTransferProtocol(B)

Botez Serban-Mihai

Facultatea de Informatica
Grupa A1 serbanbotez@fenrir.uaic.ro

Abstract. Acest raport are ca scop descrierea proiectului si abordarea tehnicilor de rezolvare folosite. Raportul contine un numar de 6 sectiuni cu nume sugestive pentru informatiile prezentate. Unele sectiuni prezinta si subsectiuni in care se prezinta anumite detalii individuale. Capitolul "Introducere" contine descrierea proiectului intr-un limbaj natural in timp ce capitolul "Tehnologii utilizate" se prezinta ce tip de server s-a folosit. "Arhitectura" aplicatiei contine detalii si diagrame specifice aplicatiei, in timp ce in capitolul "Detalii de Implementare" contine protocolul la nivel aplicatie si cod relevant proiectului. Ultimele doua capitole contin metode de imbunatatire si bibliografia folosita.

Keywords: FileTransfer · Client-Server arhitecture · Concurent server

1 Introducere

Proiectul MyFileTransferProtocol cere implementarea unei aplicatii client/server ce permite conectarea mai multor clienti la server, unde li se va pune la dispozitie o lista de comenzi ce permit autentificarea, listarea, crearea, stergerea si mutarea directoarelor, adaugarea, citirea, stergerea si mutarea fisierelor. De asemenea, va trebui implementat un mecanism de autorizare de tip whitelist/blacklist pentru conturile utilizatorilor si un mecanism de transmitere securizata a parolei la autentificare.

2 Tehnologii utilizate

Aplicatia este de tip client/server. Limbajul de programare folosit este C. Ca sistem de build se va utiliza Visual Studio Code. Pentru criptarea parolei se foloseste AES encryption in libraria OpenSSL. Pentru compilare se va folosi : gcc client.c -o client -lssl -l crypto pentru client si gcc server.c -o server -lssl -l crypto pentru server.

Serverul este de tip TCP, fiind bazat pe conexiune intre clienti si server si care permite transferul de date fara pierdere de informatii. Prin folosirea protocolului TCP, se asigura o calitate maxima a serverului, fiind astfel posibil detectarea erorilor la transmiterea datelor sau la conexiunea dintre client si server.

La acest tip de proiect nu se poate folosi protocolul UDP deoarece este nevoie de un transfer de date sigur intre client si server. De asemenea, la acest proiect

este necesara detectarea erorilor pentru o experienta cat mai buna a clientului, lucru care nu se poate realiza intr-un server UDP, deoarece nu prezinta conexiune pentru comunicare cu clientul.

3 Arhitectura aplicatiei

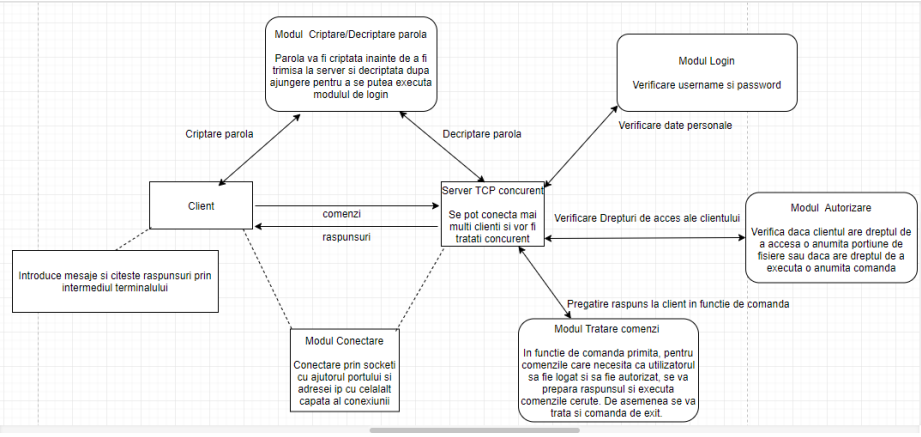
3.1 Concepte Implicate

Server Serverul este unul TCP, concurent, transferul de data facandu-se prin streaming de date.
Crearea socketului pentru a adresa clientii se realizeaza prin apelul functiei socket, dupa care se pregateste structura de date de tip sockaddr_in care contine portul, adresa ip si campul sin_family care este mereu setat la AF_INET.
Apelul functiei bind are ca scop atasarea socketului la adresa locala specificata mai sus.
Pregatirea socketului pentru a astepta clientii se face prin apelul functiei listen al carui parametru reprezinta cati clienti pot fi maxim in coada de asteptare.
Realizarea unei conexiuni cu un client specific se face prin apelul functiei accept, care va returna un nou descriptor de socket specific clientului care va fi folosit in functiile read si write.
Concurenta serverului se realizeaza utilizandu-se primitiva fork si consta in crearea unui nou proces copil de fiecare data cand un client este acceptat de catre server.
Serverul va comunica cu clientul prin intermediul primitivelor read si write care utilizeaza descriptorul de socket al clientului care a fost returnat in urma apelului accept.
Inchiderea unidirectionala a conexiunii se realizeaza prin apelul functiei close peste descriptorul de socket al clientului.

Client Pentru crearea socketului de conexiune catre server se apeleaza functia socket.
La fel ca la server, se va pregati structura de date de tip sockaddr_in.
Clientul se va conecta la server prin intermediul primitivei connect si va comunica cu serverul prin intermediul primitivelor read si write. Acestea vor utiliza descriptorul de socket al clientului pentru a putea realiza conexiunea si a citi si scrie date. Inchiderea conexiunii cu serverul se realizeaza prin apelul functiei close.

3.2 Diagrama Arhitecturala

Fig. 1. Arhitectura aplicatiei



3.3 Prezentare Module login si autorizare

Fig. 2. Modul Login

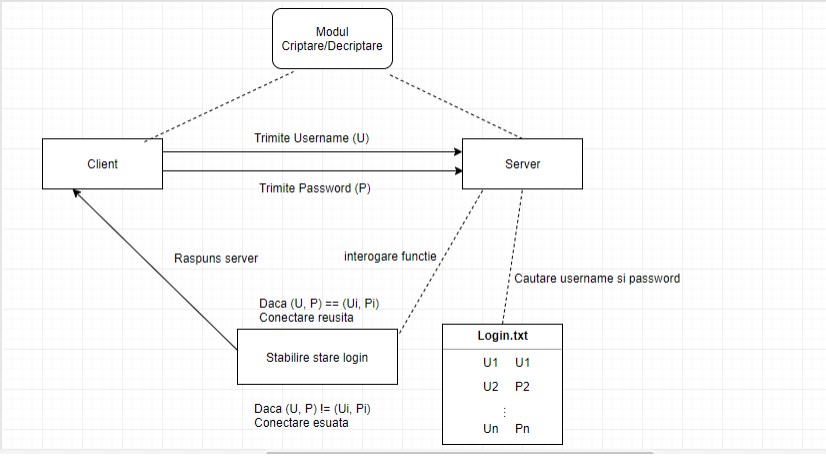
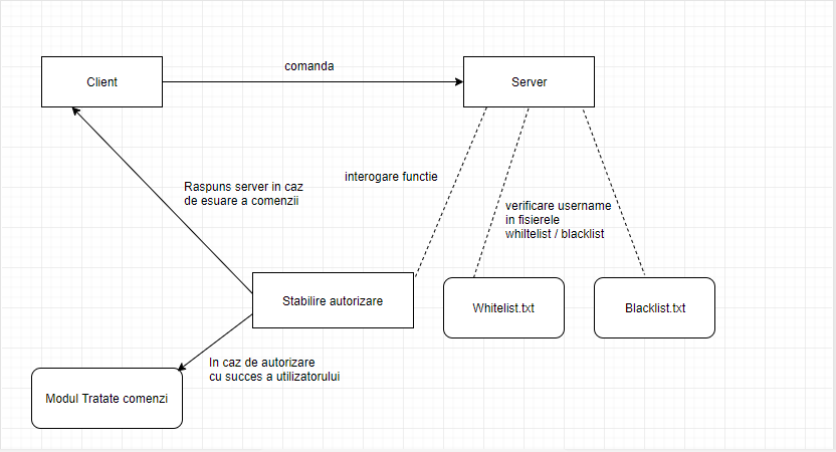


Fig. 3. Modul Autorizare



4 Detalii de implementare

4.1 Usecase-uri

1. Un utilizator nelogat fara cont in aplicatie se conecteaza la server. Deoarece conturile pot fi adaugate doar de administratorul aplicatiei, el nu se poate loga folosind comanda "login" si deci nu poate folosi comenzile puse la dispozitie de catre server. Singurele comenzi disponibile sunt "help" care afiseaza la terminal toate comenzile disponibile la momentul curent si "exit" care sugereaza inchiderea conexiunii din client si din server.
2. Un utilizator nelogat care are cont in aplicatie se conecteaza la server. Are la dispozitie 2 tipuri de comenzi pe server: care nu necesita logarea (login, exit, help) si care necesita logarea(list, create, ...). Daca, logarea a esuat, utilizatorul se poate loga din nou. Daca logarea a avut succes, utilizatorul poate folosi comenzile ce necesita logarea.
3. Un utilizator logat incearca sa apeleze o comanda. Se va face verificarea daca utilizatorul are aprobarea pentru a folosi acea comanda sau pentru a accesa o zona din memorie. Pentru aceasta validare se vor folosi fisierele de whitelist/blacklist. Daca utilizatorul se gaseste in fisierul whitelist, i se va aproba comanda si se va returna rezultatul comenzii. Daca el se gaseseste in blacklist, un mesaj specific va fi returnat. 4. Orice utilizator(indiferent daca este logat sau nu) are acces la comenzi locale, pe partea de client.

4.2 Usecase-uri rele

Cand un utilizator trimite catre server o comanda necunoscuta, el va primi inapoi un mesaj care il notifica de faptul ca acea comanda nu este valida.

In cazul in care conexiunea la server nu se poate realiza, clientul va fi notificat printr-un mesaj la ecran, in urma valorii negative de return a functiei conect.

In cazul in care serverul nu trimite mesaje catre client, va fi afisat un mesaj, cu ajutorul valorii de return a functiei read sau write, in functie de momentul in care nu s-a putut realiza transferul. Invers, in cazul in care serverul nu poate scrie sau citi date de la client, se va apela comanda exit.

4.3 Protocol la nivelul aplicatie

Portul si adresa ip a clientului vor fi hardcodate in codul clientului.
Comenzile sunt de 2 tipuri: locale(pe partea de client) si remote.

Comenzile remote alocate sunt de 3 tipuri: pentru lucru cu foldere, pentru lucrul cu fisiere si implicite.

Comenzi implicite :

- exit : inchide conexiunea clientului cu serverul si iese din aplicatie
- login : permite logarea
- disconnect : permite delogarea utilizatorului deja logat
- list : listeaza continutul directorului curent
- help : listeaza comenzile valide care pot fi introduse
- location : listeaza locatia curenta ca si path

Comenzi specifice folderelor :

- crdir : pentru a creea un director nou
- chdir : pentru muta locatia curenta la un director valid
- deldir : pentru sterge un director selectat

Comenzi specifice fisierelor :

- mvfile : pentru a muta un fisier
- delfile : pentru a sterge un fisier
- renamefile : pentru a redenumi un fisier
- put : uploadeaza un fisier de pe calculatorul local pe server
- get : ia in directorul curent un fisier de pe server

Comenzi disponibile local :

- clocation : afiseaza locatia curenta de pe masina locala
- clist : listeaza directorul local curent
- chelp : listeaza toate comenzile disponibile local
- cchdir : schimba directorul curent
- ccrdir : creeaza local un director nou
- cdeldir : sterge un director local selectat
- cdelfile : sterge un fisier local selectat
- crenamefile : redenumeste un fisier local selectat
- cmvfile : muta un fisier local selectat la o destinatie stabilita

Clientul are ca scop citirea comenzilor de la tastatura si transmiterea acestora catre server pentru a putea fi validate, dupa care primeste raspunsul. Pentru comenzile locale, serverul nu este implicat, ele realizandu-se inainte de transmiterea mesajului catre server. Serverul primeste comenzile de la client si in caz ca sunt valide, apeleaza functii specifice care returneaza un rezultat in functie de comanda specificata.

Implicit dupa comenzile de schimbare a directorului se vor afisa .

La comunicarea dintre client si server, intai se trimite lungimea mesajului si apoi mesajul in sine. Astfel, la citire se va sti exact cate caractere trebuie citite.

Erorile vor fi detectate de fiecare daca cand o functie apelata returneaza o valoare de eroare si se va trimite catre capatul opus al conexiunii un mesaj de eroare, atunci cand inca se mai poate(serverul si clientul mai sunt conectati).

5 Concluzii

Pentru stocarea si verificarea datelor referitoare la utilizatori se poate utiliza o baza de date in loc de fisiere stocate local pe sistem.

6 Bibliografie

https://www.csd.uoc.gr/ hy556/material/tutorials/cs556-3rd-tutorial.pdf
http://man7.org/linux/man-pages/man7/ip.7.html
http://manpages.ubuntu.com/manpages/trusty/man1/tcpserver.1.html
https://about.draw.io/support/
https://devops.datenkollektiv.de/using-aes-with-openssl-to-encrypt-files.html
https://github.com/llubu/mpro/blob/master/crypt/src/aes2.c
https://linux.die.net/man/