Raport Tema 2 - Part2Part (A)

Rotariu Cosmin - Andrei, IIA4

December 2020

1 Introducere

În această **fișă de raport** voi prezenta anumite detalii ce descriu funcționalitățile proiectului pe care am ales să îl implementez, proiect numit **Part2Part**.

Cerința proiectului constă în crearea unei aplicații de tip peer-to-peer ce permite partajarea fișierelor între clienții conectați în acel moment la server. Pentru comunicație va trebui să folosesc socket-uri, iar programul server va trebui să fie capabil să servească simultan mai multe cereri de transfer provenite de la clienți.

2 Tehnologii utilizate

Voi implementa o aplicație **server-client** de tip **TCP concurent**, folosind **socketi** pentru comunicare, și, implicit **thread-uri** pentru a asigura conectarea mai multor clienți la server și pentru a-i servi simultan.

Avantajul TCP-ului[1] față de UDP[1] ar fi faptul că TCP[1] furnizează un flux sigur de date între două sau mai multe calculatoare aflate în rețea și asigură stabilirea unei conexiuni permanente între acele calculatoare pe parcursul comunicării, având siguranța că informațiile trimise de un proces vor fi recepționate corect și complet la destinație, fără pierdere.

Pentru fiecare client care se va conecta, serverul TCP va crea câte un thread[3], după cum am menționat puțin mai sus, și astfel se va facilita servirea simultană a celor conectați în rețea.
Se va putea, astfel, comunica între server și client, și, implicit, între mai mulți clienti conectați în același timp la rețea.

Îmi propun să utilizez și o bază de date de tip SQLITE[7] pentru a stoca informații despre fișierele existente în sistemele clienților conectați la rețea, mai exact numele fișierului, detalii despre el, și adresa clientului unde se poate găsi

acel fișier, așadar la fiecare cerere, serverul va ști la ce client se află fișierul căutat.

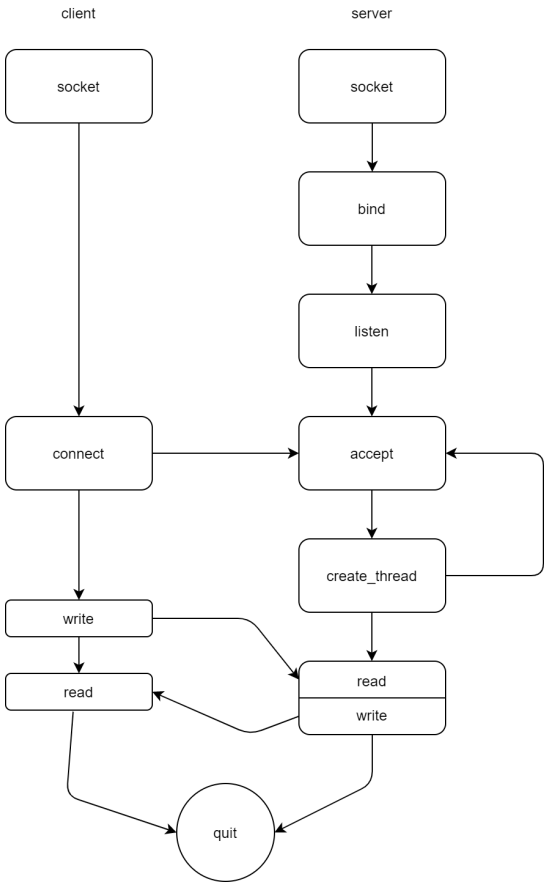

Figure 1: Diagramă TCP concurent cu thread-uri

3 Arhitectura aplicației

Descrierea diagramei de mai jos: În urma introducerii de către un client a comenzii ce face referire la intenția de a descărca un anumit fișier, serverul analizează baza de date și îi transmite clientului care a făcut request, adresa clientului țintă ce deține fișierul căutat. Eventual, clientul mai poate cere serverului să i se perimtă să partajeze și el un fișier în acea rețea.
Pentru a funcționa aplicația, este necesar ca minim 2 clienți să fie conectați la server.

Clienții vor avea posibilitatea de a alege:

- să descarce un fișier din lista documentelor partajate de către ceilalți clienți (listă pusă la dispoziție serverului de către baza de date)
- să distribuie și ei un fișier în acea rețea

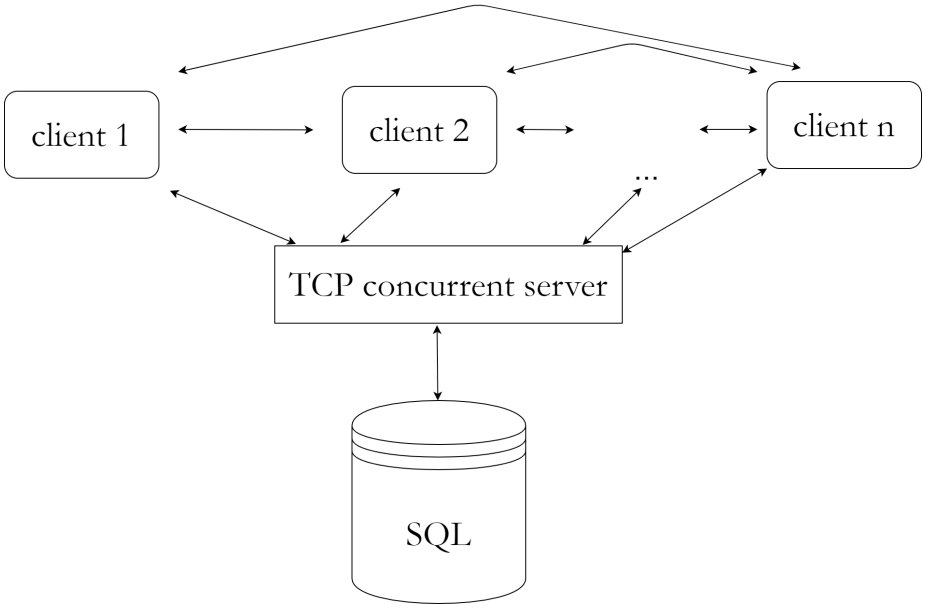

Figure 2: Arhitectura aplicației

Serverul:

- primește numele unui fișier de la un client și eventual alte criterii de căutare
  - accesează baza de date, caută fișierul respectiv
  - trimite clientului adresa deținătorului
- îi oferă posibilitatea unui client de a partaja un fișier cu întreaga rețea

4 Detalii de implementare

Pentru implementarea acestui protocol de comunicare pentru **Part2Part**, am creat un fisier *server.c* și unul *client.c*.
Comunicarea dintre *server* și *client* se realizează prin *socketi de comunicare* și *thread*-uri. Am ales să folosesc *thread*-uri în loc de $fork$-uri deoarece *thread*-urile din motive de performanță și deoarece se recepționează mai rapid mesajele dintre *server* și *client*.

Astfel, *clientul* se conectează la *server* și i se afișează un meniu cu diferite comenzi pe care le poate executa.
Meniul va avea 3 comenzi: **"download"**, **"share"** și **"exit"**.

- Dacă clientul va alege **prima** comandă, și anume *download*, serverul îi va cere prin funcția $get\_filename\_from\_user()$ să introducă de la tastatură numele fișierului pe care dorește să îl descarce din rețea, și, eventual, anumite criterii după care serverul poate găsi mai ușor acel fișier. Serverul va prelua datele de la client, va interoga baza de date cu ajutorul funcției *call_db*() și va obține adresa userului care deține fișierul căutat. Apoi va întoarce un răspuns pozitiv clientului care a făcut request prin funcția *send_path_adress*, alături de adresa respectivului user, sau un răspuns negativ, caz în care fișierul nu a fost găsit în baza de date. Mai departe, clientul se va putea conecta la clientul target folosind comanda *connect* $< adresa >$ și va putea descărca fișierul dorit.
- Dacă clientul va alege a **doua** comandă, și anume *share*, serverul îi va solicita, prin funcția $get\_filename\_from\_user()$ să introducă numele fișierului pe care dorește să îl partajeze în rețea, acel fișer fiind astfel vizibil și pentru ceilalți clienți conectați la rețea în acel moment. După primirea datelor de la client, serverul va uploada în baza de date numele fișierului, alături de adresa clientului deținător cu ajutorul funcției *upload_to_db*(), apoi va întoarce un mesaj pozitiv, caz în care uploadarea s-a efectuat cu succes, iar în caz contrar, va întoarce un mesaj care indică faptul că fișierul există deja în baza de date.
- Dacă clientul va alege a **treia** comandă, și anume *exit*, se va deconecta de la server.

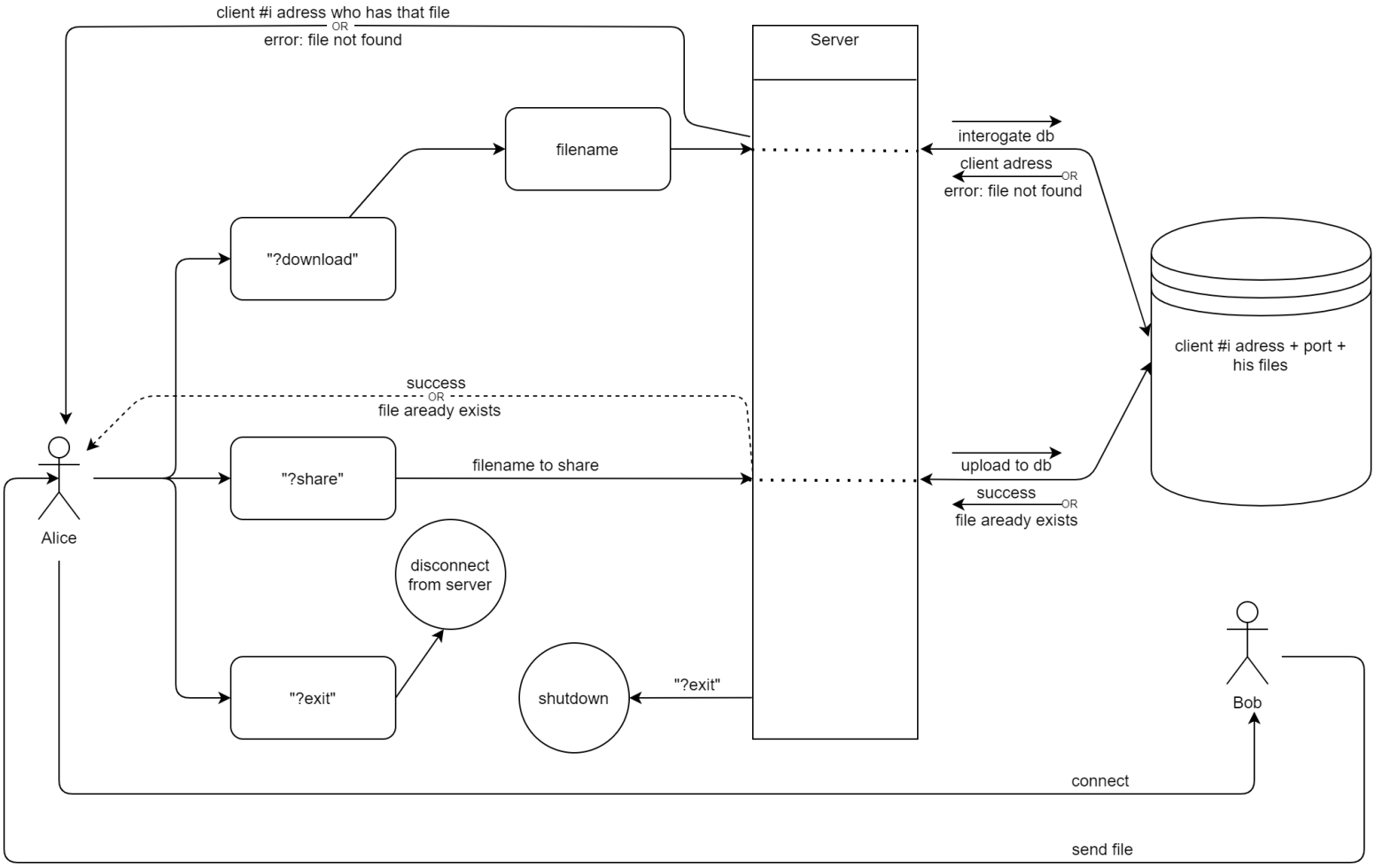

Figure 3: Use Cases

5 Concluzie

Această aplicație este utilă mai ales persoanelor care doresc să partajeze anumite date confidențiale și nu doresc să folosească software-urile deja existente pe piață. De asemenea, permite partajarea de fișiere într-un mod sigur și fără pierderi.
Soluția propusă ar putea fi îmbunătățită prin adăugarea pe viitor a unei părți de login, conturile fiind stocate într-o bază de date SQL sau fișiere JSON. În plus, aplicația ar putea conține o interfață, fiindu-le mult mai ușor clienților să interacționeze cu funcțiile programului.

References

[1] https://ftp.utcluj.ro/pub/users/civan/CPD/2_LABORATOR/06_Socket/6_SK.pdf

[2] https://app.diagrams.net/

[3] https://sites.google.com/view/fii-rc/

[4] https://profs.info.uaic.ro/ computernetworks/

[5] https://www.overleaf.com/

[6] https://notes.shichao.io/unp/

[7] https://www.sqlite.org/cintro.html