UNIVERSITATEA TEHNICA
DIN CLUJ-NAPOCA
CENTRUL UNIVERSITAR NORD DIN BAIA MARE
FACULTATEA DE INGINERIE

DEPARTAMENTUL DE CALCULATOARE

Programul de studii: Calculatoare

APLICAȚIE P2P DE PARTAJARE FIȘIERE

PROIECT: SISTEME DISTRIBUITE

Autor: **Pop Lucas**

2026

Cuprins

1. Introducere

1.1. Context general

Aplicația de partajare de fișiere peer-to-peer (P2P) a fost dezvoltată ca o soluție descentralizată pentru transferul de fișiere între utilizatori conectați la aceeași rețea locală. Spre deosebire de arhitecturile tradiționale client-server, unde un server central gestionează toate resursele și transferurile, modelul P2P permite fiecărui participant (peer) să acționeze atât ca client, cât și ca server.

Această abordare elimină dependența de un punct central, crescând scalabilitatea sistemului. Aplicația oferă o interfață grafică simplă prin care utilizatorii pot descoperi automat alți participanți în rețea, pot vizualiza fișierele partajate de aceștia și pot inițiа descărcări directe.

1.2. Obiective

Scopul principal al proiectului este de a crea un sistem funcțional de partajare a fișierelor într-o rețea locală, bazat pe o arhitectură P2P pură, fără un server central de coordonare.

Obiectivele specifice ale aplicației sunt:

- **Descoperire automată a nodurilor (peers):** Implementarea unui mecanism de auto-descoperire a participanților activi în rețea.
- **Partajare de fișiere:** Permiterea utilizatorilor să partajeze fișiere dintr-un director local specificat.
- **Transfer de fișiere:** Facilitarea descărcării fișierelor de la un peer la altul.
- **Interfață utilizator intuitivă:** Oferirea unei interfețe grafice pentru vizualizarea nodurilor active, a fișierelor disponibile și pentru gestionarea descărcărilor.
- **Notificări în rețea:** Anunțarea automată a tuturor participanților atunci când un nou fișier este adăugat în directorul partajat.
- **Robustețe:** Sistemul trebuie să gestioneze dinamic adăugarea și plecarea nodurilor din rețea.

1.3. Lista MoSCoW

Pentru a prioritiza funcționalitățile în cadrul dezvoltării, am utilizat metoda MoSCoW.

Must have	Should have	Could have	Won't have
Descoperire peer prin UDP Multicast	Afișare progres descărcare	Căutare fișiere după nume	Autentificare utilizatori
Partajare director local	Gestionare erori de rețea	Pauză/Reluare descărcări	Criptare transferuri
Descărcare fișiere prin TCP	Notificare la finalizarea descărcării	Chat între peer-i	Suport pentru rețele WAN
Interfață grafică (JavaFX) pentru vizualizare peer-i și fișiere			Sistem de reputație
Anunțare fișiere noi			

Tabela 1: Lista MoSCoW pentru aplicația P2P.

2. Arhitectura și Componente

Arhitectura aplicației este una descentralizată, specifică sistemelor P2P. Fiecare instanță a aplicației rulează independent și comunică direct cu celelalte instanțe (peer-i) din rețeaua locală. Nu există un server central care să medieze comunicarea sau să stocheze informații.

Aplicația este construită în Java și este compusă din trei module logice principale:

- **Modulul de Rețea:** Responsabil pentru descoperirea nodurilor și transferul de fișiere.
- **Modulul de Date:** Definește structurile de date utilizate în comunicare.
- **Modulul de Interfață Utilizator (UI):** Oferă interfața grafică pentru interacțiunea cu utilizatorul.

2.1. Modulul de Rețea

Acesta reprezintă nucleul aplicației și este format din componente specializate pentru diferite tipuri de comunicare.

2.1.1. NodeDiscoveryService

Această componentă este responsabilă pentru descoperirea și monitorizarea celorlalți peer-i din rețea.

- **Tehnologie:** Utilizează **UDP Multicast** pentru a trimite și a primi mesaje de "broadcast" într-un mod eficient. Toți peer-ii se alătură unui grup multicast specific (`230.0.0.1`) și ascultă pe un port comun (`9876`).
- **Funcționare:**
  1. La pornire, serviciul începe să trimită periodic mesaje de anunț (`PEER_ANNOUNCE`) în grupul multicast. Aceste mesaje conțin informații despre peer-ul curent (ID unic, port pentru transfer fișiere).
  2. Simultan, ascultă pentru mesaje de la alți peer-i. Când un mesaj de anunț este primit, informațiile despre peer-ul respectiv sunt adăugate sau actualizate într-o listă locală de peer-i descoperiți.
  3. Implementează un mecanism de timeout: dacă un peer nu mai trimite mesaje de anunț pentru o perioadă de timp (`PEER_TIMEOUT_SECONDS`), este considerat inactiv și eliminat din listă.
  4. De asemenea, gestionează notificările de tip `FILE_ADDED`, permițând unui peer să anunțe întreaga rețea atunci când un nou fișier devine disponibil pentru partajare.

2.1.2. FileServer

Fiecare peer rulează o instanță de `FileServer`, care acționează ca un server pentru descărcări.

- **Tehnologie:** Utilizează un **ServerSocket TCP** care ascultă pe un port dedicat, alocat dinamic la pornirea aplicației. Portul este comunicat celorlalți peer-i prin `NodeDiscoveryService`.
- **Funcționare:** Așteaptă conexiuni de la alți peer-i. Când un peer se conectează pentru a descărca un fișier, `FileServer` citește numele fișierului solicitat, îl localizează în directorul partajat și îl transmite clientului sub formă de flux de octeți (byte stream).

2.1.3. FileClient

Această componentă este responsabilă pentru inițierea și gestionarea descărcării unui fișier de la un alt peer.

- **Tehnologie:** Utilizează un **Socket TCP** pentru a se conecta la `FileServer`-ul peer-ului de la care se dorește descărcarea.
- **Funcționare:** Când utilizatorul alege să descarce un fișier, `FileClient` obține adresa IP și portul `FileServer`-ului țintă din informațiile de peer descoperite. Se conectează la acel server, trimite numele fișierului dorit, apoi primește fluxul de octeți și îl salvează local într-un director de descărcări.

2.2. Modulul de Date (Model)

Acest modul conține clasele POJO (Plain Old Java Object) care definesc structura datelor schimbate între peer-i. Mesajele sunt serializate în format JSON folosind biblioteca **Gson**.

- **PeerInfo:** Stochează informațiile despre un peer: un ID unic (`UUID`), adresa IP, portul pentru descoperire și portul pentru transferul de fișiere.
- **FileInfo:** Reprezintă un fișier partajat, conținând numele fișierului, dimensiunea și ID-ul peer-ului care îl deține.
- **P2PMessage:** O clasă generică pentru toate mesajele din rețea. Conține un tip de mesaj (ex: `PEER_ANNOUNCE`, `FILE_ADDED`), informații despre expeditor (`PeerInfo`) și, opțional, informații despre un fișier (`FileInfo`).

2.3. Modulul de Interfață Utilizator (UI)

Interfața grafică este construită folosind **JavaFX**, un framework modern pentru aplicații desktop în Java.

- **Tehnologii:** FXML pentru definirea structurii UI-ului în mod declarativ și CSS pentru stilizare.
- **MainController:** Clasa care leagă logica de business de interfața grafică. Gestionează evenimentele declanșate de utilizator (click-uri pe butoane, selecții în liste) și actualizează UI-ul în mod dinamic.

- **Funcționalități UI:**
  - O listă a peer-ilor descoperiți în rețea.
  - O listă a tuturor fișierelor partajate de peer-ii activi.
  - Butoane pentru a iniția descărcarea unui fișier selectat.
  - Un log de stare care afișează mesaje despre evenimentele din rețea (peer nou, peer pierdut, descărcare finalizată).
- **Thread-Safety:** Deoarece operațiunile de rețea rulează pe fire de execuție separate, actualizările UI-ului se fac folosind `Platform.runLater()` pentru a asigura că modificările componentelor grafice au loc pe firul de execuție JavaFX Application Thread, prevenind astfel erori de concurență.

3. Logica de Funcționare

Fluxul de operare al aplicației, din momentul pornirii, este următorul:

1. **Inițializare:** La pornirea aplicației (`P2PFileShareApp`), se inițializează componentele principale:
   - Se pornește `FileServer` pe un port TCP disponibil.
   - Se pornește `NodeDiscoveryService`, care primește ca parametru portul alocat pentru `FileServer`.
   - Se încarcă interfața grafică JavaFX (`MainWindow.fxml`) și se injectează serviciile de rețea în `MainController`.
2. **Descoperire Peer-i:**
   - `NodeDiscoveryService` începe să trimită mesaje `PEER_ANNOUNCE` prin UDP multicast la intervale regulate.
   - Simultan, ascultă mesajele de la alți peer-i. Când un peer nou este descoperit, `MainController` este notificat printr-un callback și adaugă peer-ul în lista afișată în UI.
   - Dacă un peer devine inactiv, este eliminat din lista din UI.
3. **Partajare și Anunțare Fișiere:**
   - Utilizatorul are un director local desemnat pentru partajare (`P2P-Shared`).

- Când un fișier nou este adăugat în acest director (simulat în aplicație), `NodeDiscoveryServic` trimite un mesaj `FILE_ADDED` prin multicast.
- Toți peer-ii activi primesc această notificare. `MainController` de pe fiecare peer actualizează lista globală de fișiere disponibile în UI.

4. **Descărcare Fișier:**
   - Un utilizator (Peer A) selectează un fișier din lista din UI, care este partajat de un alt utilizator (Peer B).
   - Peer A apasă butonul de descărcare. `MainController` de pe Peer A inițiază un `FileClient`.
   - `FileClient` (de pe Peer A) se conectează la `FileServer` (de pe Peer B) folosind adresa IP și portul cunoscute.
   - Clientul trimite numele fișierului dorit. Serverul citește fișierul de pe disc și îl trimite înapoi prin socket-ul TCP.
   - Clientul primește datele și le salvează într-un director local (`P2P-Downloads`).
   - La finalizare, un mesaj este afișat în log-ul din UI.
5. **Oprire:** Când utilizatorul închide aplicația, metoda `stop()` este apelată pe toate serviciile. `NodeDiscoveryService` părăsește grupul multicast și închide socket-ul, iar `FileServer` își închide portul de ascultare.

4. Concluzii

Proiectul demonstrează cu succes implementarea unei aplicații de partajare de fișiere P2P funcționale, bazată pe o arhitectură complet descentralizată. Prin utilizarea tehnologiilor Java standard (TCP/UDP Sockets, JavaFX) și a unor biblioteci consacrate (Gson), s-a realizat un sistem robust și eficient pentru rețele locale.

Obiectivele principale, precum descoperirea automată a nodurilor, transferul direct de fișiere și notificările în timp real, au fost atinse. Arhitectura modulară, cu o separare clară a responsabilităților între componentele de rețea, date și UI, face codul ușor de înțeles, întreținut și extins.

Ca direcții viitoare, aplicația poate fi îmbunătățită prin adăugarea de funcționalități precum căutarea de fișiere, afișarea progresului descărcărilor sau suport pentru reluarea transferurilor întrerupte. Cu toate acestea, versiunea curentă reprezintă o fundație solidă și o demonstrație practică a principiilor sistemelor distribuite de tip peer-to-peer.

LINK GITHUB