UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
CENTRUL UNIVERSITAR NORD DIN BAIA MARE

Facultatea de Inginerie
Departamentul de Inginerie Electrică, Electronică și Calculatoare

SISTEM DISTRIBUIT DE TRANSFER FIȘIERE PENTRU REȚELE LOCALE

Proiect la disciplina:
Sisteme Distribuite

Autor:
Titea Dan Ruben

Program de studii:
Calculatoare

Anul: IV

Baia Mare
2026

Cuprins

1 Introducere

1.1 Scopul documentației

Acest document descrie arhitectura, implementarea și funcționarea unui sistem de partajare a fișierelor (file-sharing) de tip Peer-to-Peer (P2P), bazat pe specificațiile protocolului BitTorrent, adaptat pentru operarea în rețele locale (LAN) fără necesitatea unui tracker centralizat.

1.2 Domeniul de aplicare

Sistemul acoperă crearea fișierelor .torrent, descoperirea automată a nodurilor în rețea (Peer Discovery), transferul de date fragmentat (chunks/pieces) și verificarea integrității datelor prin hash-uri SHA-1.

1.3 Definiții, acronime și termeni

- **Peer**: Un nod în rețea care participă la transferul fișierelor.
- **Seeder**: Un peer care deține fișierul complet și îl oferă altora.
- **Leecher**: Un peer care descarcă fișierul (nu îl are complet).
- **Swarm**: Grupul de peer-i care partajează același fișier (identificat prin info_hash).
- **Piece**: O diviziune logică a fișierului (implicit 256 KB în acest sistem).
- **Block**: O subdiviziune a unei piese, unitatea efectivă de transfer (16 KB).
- **Info Hash**: Amprenta digitală SHA-1 a metadatelor fișierului, folosită pentru identificare unică.

2 Context și motivație

2.1 Problema rezolvată

Transferul fișierelor mari într-o rețea locală necesită adesea configurarea unor servere dedicate (FTP, SMB) sau cunoașterea adreselor IP ale participanților. Acest sistem elimină configurarea manuală.

2.2 De ce P2P / BitTorrent

Arhitectura P2P permite scalabilitatea: cu cât sunt mai mulți utilizatori care descarcă un fișier, cu atât viteza de descărcare potențială crește, deoarece fiecare peer devine și sursă de upload.

2.3 Scenarii de utilizare

- **Birouri / Companii:** Distribuirea rapidă a kit-urilor de instalare, a imaginilor ISO sau a backup-urilor mari către toți angajații simultan, fără a bloca serverul central.
- **Laboratoare Școlare:** Profesorul trimite materiale didactice voluminoase către 30 de stații de lucru simultan.
- **LAN Party / Home Use:** Transferul de fișiere multimedia între laptopuri și desktopuri fără a folosi internetul.

3 Prezentare generală a sistemului

Sistemul este o aplicație Java multi-threaded care funcționează simultan ca Server (ascultă cereri) și Client (inițiază conexiuni). Nu există un punct unic de eșec (SPOF).

3.1 Funcționalități principale

- **Generare Torrent**: Crearea fișierelor .torrent din fișiere locale.
- **Discovery**: Descoperirea peer-ilor prin Multicast UDP (înlocuiește Tracker-ul clasic).
- **Download Paralel**: Descărcarea pieselor de la mai mulți peer-i simultan.
- **Integritate**: Verificarea automată a fiecărei piese descărcate folosind SHA-1.
- **Resume**: Capacitatea de a relua descărcări întrerupte (gestionată prin Bitfield și fișiere persistente).

3.2 Limitări cunoscute

- Funcționează optim doar în LAN (Multicast nu este rutat pe internetul public).
- Nu implementează criptarea traficului.

4 Noțiuni teoretice

4.1 Rețele peer-to-peer

Spre deosebire de modelul Client-Server, în P2P fiecare nod este egal. Resursele (stocare, procesare, bandă) sunt partajate direct între noduri.

4.2 Principiile BitTorrent

Fișierul este împărțit în bucăți de dimensiuni fixe numite Pieces (de obicei 256KB - 1MB). Fiecare piesă este verificată printr-un hash SHA-1. Piesele sunt descărcate într-o ordine non-secvențială, ceea ce permite unui nod să înceapă să ofere date altora (upload) chiar înainte de a termina propria descărcare

4.3 Modele de distribuție a fișierelor

Sistemul folosește modelul "Tit-for-Tat" (în principiu), încurajând utilizatorii să ofere date pentru a primi date, deși implementarea curentă este permisivă (unchoke optimistic).

5 Protocolul BitTorrent și extensiile implementate

5.1 Componentele protocolului standard

Sistemul respectă fluxul TCP standard BitTorrent:

1. **Handshake**: Schimb de info_hash și peer_id.
2. **Bitfield**: Schimbul hărții pieselor deținute (o secvență de biți 1 și 0).
3. **Interested/Unchoke**: Negocierea stării conexiunii.
4. **Request/Piece**: Cererea și transferul efectiv de date.

5.2 Adaptări față de protocolul clasic

Sistemul înlocuiește Tracker-ul HTTP/UDP (serverul central care ține lista de IP-uri) cu un mecanism multicast.

5.3 Mecanismul de peer discovery descentralizat prin multicast

5.3.1 Descriere generală

Aplicația implementează clasa **LocalDHT** care simulează comportamentul unui Distributed Hash Table, dar transportul se face prin Multicast UDP.

5.3.2 Diferențe față de tracker/DHT

Un tracker clasic este un server care ține o listă de IP-uri. Un DHT global (Kademlia) folosește o structură de arbore logic. Această implementare(multicast) trimite interogări ("Cine are fișierul cu hash X?") către o adresă de multicast (239.192.1.1). Toți peer-ii din rețea ascultă această adresă și răspund dacă au informația.

5.3.3 Avantaje și limitări

- **Avantaj**: Zero configurare; funcționează instant în LAN; nu necesită IP-uri statice sau servere externe.
- **Limitare**: Traficul multicast nu trece de routere (de obicei), limitând funcționarea la subrețeaua locală.

5.4 Codificarea Bencode

5.4.1 Ce este Bencode?

Bencode este formatul de serializare (codificare a datelor) utilizat exclusiv de protocolul BitTorrent pentru stocarea metadatelor în fișierele .torrent și pentru comunicarea mesajelor în rețeaua DHT. A fost ales pentru simplitatea sa și pentru faptul că este "binary-safe" (poate manipula șiruri de bytes brute, cum ar fi hash-urile SHA-1, fără probleme de encoding).

5.4.2 Tipuri de date suportate

Bencode suportă doar patru tipuri de date structurale. Formatul este recursiv, ceea ce înseamnă că listele și dicționarele pot conține oricare dintre aceste tipuri.

Șiruri de caractere (Byte Strings):

- **Format**: <lungime>:<conținut>
- **Descriere**: Un număr care indică lungimea, urmat de două puncte și șirul efectiv.
- **Exemplu**: 4:spam (șirul "spam").
- **Notă**: Hash-urile SHA-1 sunt stocate ca șiruri de 20 de bytes (ex: 20:<date_binare>).

Numere întregi (Integers):

- **Format**: i<număr>e
- **Descriere**: Delimitate de literele i (start) și e (end).
- **Exemplu**: i42e (numărul 42), i-3e (numărul -3).

Liste (Lists):

- **Format**: l<conținut>e
- **Descriere**: O secvență ordonată de elemente Bencode, delimitate de l și e.
- **Exemplu**: l4:spami42ee (o listă care conține string-ul "spam" și numărul 42).

Dicționare (Dictionaries):

- **Format**: d<conținut>e
- **Descriere**: O colecție de perechi cheie-valoare, delimitate de d și e. Cheile trebuie să fie obligatoriu șiruri de caractere (byte strings).
- **Exemplu**: d3:bar4:spam3:fooi42ee (reprezintă "bar": "spam", "foo": 42).

5.4.3 Importanța în calculul Info Hash

O caracteristică critică a Bencode este că dicționarele trebuie să aibă cheile sortate lexicografic (alfabetic). Acest lucru asigură unicitatea reprezentării binare: același obiect de date va produce întotdeauna exact aceeași secvență de bytes. Această proprietate este vitală pentru calculul info_hash-ului. Dacă ordinea cheilor s-ar schimba, hash-ul SHA-1 al dicționarului info ar fi diferit, iar fișierul nu ar mai fi recunoscut în rețeaua BitTorrent.

5.5 Fișierul .torrent

5.5.1 Structura fișierului torrent:

Fișierul .torrent este un dicționar serializat (Bencoded) care conține:

- **name**: Numele fișierului.
- **piece length**: Dimensiunea unei piese.
- **pieces**: Un șir lung de bytes care reprezintă concatenarea hash-urilor SHA-1 (20 bytes fiecare) pentru toate piesele.
- **length**: Dimensiunea totală a fișierului.

5.5.2 Crearea unui fișier torrent

1. Se citește fișierul sursă.
2. Se calculează numărul de piese.
3. Se citește fiecare piesă și se aplică SHA-1.
4. Se concatenează hash-urile și se construiește dicționarul Bencode.

5.6 Mesaje și tipuri de pachete (TCP)

Mesajele sunt serializate TLV (Type-Length-Value) sau structuri fixe:

- **KeepAlive** (len=0)
- **Choke** (id=0), **Unchoke** (id=1)
- **Interested** (id=2), **Not Interested** (id=3)
- **Have** (id=4, payload=index)
- **Bitfield** (id=5, payload=bitset)
- **Request** (id=6, payload=index, begin, length)
- **Piece** (id=7, payload=index, begin, block)

5.7 Strategii de selecție a pieselor

Codul implementează o strategie de tipul "First Available" sau iterativă. Verifică ce piese are peer-ul conectat (prin BitSet) și care lipsesc local, apoi le solicită.

6 Arhitectura sistemului

Arhitectura este modulară, separând logica de rețea (TCP/UDP) de logica de stocare (File I/O).

6.1 Componente principale

6.1.1 Clientul BitTorrent (PeerNode)

Clasa centrală care este orchestratorul. Inițializează serverul, clientul, "DHT"-ul și gestionează lista de torrenți activi.

6.1.2 Server (PeerServer)

Ascultă pe un port TCP pentru conexiuni externe. Când un alt peer se conectează, instanțiază o conexiune și începe negocierea (Handshake).

6.1.3 Client (PeerClient)

Inițiază conexiuni TCP către peer-ii descoperiți prin "DHT". Gestionează un pool de fire de execuție pentru a descărca de la mai mulți peer-i simultan.

6.1.4 Manager de fișiere (FileManager)

Se ocupă de persistența datelor.

- Folosește RandomAccessFile pentru a scrie piese în ordine aleatorie direct pe disc.
- Menține un BitSet (harta pieselor) pentru a ști ce s-a descărcat.
- Validează hash-ul SHA-1 la scrierea pe disc.

6.2 Fluxuri de date

- **Descoperire**: LocalDHT primește IP:Port pentru un info_hash.
- **Conectare**: PeerClient deschide socket TCP.
- **Negociere**: Schimb de Handshake și Bitfield.
- **Transfer**: Cereri (Request) -> Răspunsuri (Piece) -> Scriere Disc (FileManager).

7 Modelul de date

7.1 Structuri interne

Aplicația folosește structuri concurente (ConcurrentHashMap) pentru a gestiona conexiunile și starea peer-ilor, asigurând thread-safety într-un mediu multithreaded.

7.2 Reprezentarea pieselor

O "Piesă" (Piece) este unitatea logică de verificare (hash). Un "Bloc" (Block) este unitatea de transfer prin rețea (16KB). O piesă este compusă din mai multe blocuri. FileManager asamblează blocurile înainte de a valida piesa completă.

7.3 Gestionarea stării download/upload

Starea este menținută prin BitSet. Un bit setat pe 1 la indexul i înseamnă că piesa i este descărcată, verificată și disponibilă pentru upload.

8 Protocolul de Descoperire (Network Discovery)

În absența unui server central, aplicatia utilizează un DHT (Distributed Hash Table) Local simplificat, bazat pe protocolul UDP.

8.1 Mecanismul Multicast

- **Query (Interogare)**: Când un nod dorește un fișier, trimite un pachet UDP către grupul multicast conținând Info Hash-ul căutat.
- **Response (Răspuns)**: Orice nod din rețea care participă activ la acel torent (are fișierul complet sau parțial) recepționează interogarea. Dacă Info Hash-ul local coincide cu cel cerut, nodul răspunde direct (Unicast) solicitantului cu propriul său IP și Port TCP.

Acest mecanism permite crearea dinamică a unui Swarm (Roi) – o listă temporară de IP-uri care participă la transferul aceluiași fișier.

9 Fluxul de Funcționare (Cum funcționează)

9.1 Partajarea unui fișier (Publishing)

Când un utilizator dorește să trimită un fișier:

1. Aplicația analizează fișierul și îl împarte virtual în piese egale.
2. Se generează o "amprentă digitală" unică (Hash) pentru fiecare piesă.
3. Se creează un fișier .torrent care poate fi trimis colegilor (prin email, chat, stick USB).
4. Nodul începe să "anunțe" în rețea că deține acest fișier și este gata să îl livreze.

9.2 Descărcarea unui fișier (Downloading)

Când un utilizator deschide un fișier .torrent:

- **Căutarea**: Aplicația strigă în rețea: "Cine are fișierul cu această amprentă?".
- **Conectarea**: Calculatoarele care au fișierul (sau părți din el) răspund. Aplicația inițiază conexiuni directe cu ele.
- **Negocierea**: Aplicația schimbă "hărți" cu partenerii pentru a vedea cine ce bucăți are.

Transferul Paralel:

- Cere Piesa 1 de la Calculatorul A.
- Cere Piesa 2 de la Calculatorul B.
- Cere Piesa 3 de la Calculatorul C.

Verificarea: Pe măsură ce piesele sosesc, sunt verificate matematic. Dacă o piesă este coruptă, este ștearsă automat și cerută din nou de la altcineva.

10 Testare

Pentru a valida funcționalitatea sistemului am efectuat un test de integrare într-o rețea locală (LAN), utilizând două PC-uri diferite. Acest test a verificat capacitatea de descoperire automată prin Multicast, stabilirea conexiunii TCP și transferul corect al datelor.

- **PC A (Seeder):** Laptop cu IP-ul 192.168.137.1.

  Rol: Inițiază partajarea unui fișier pdf de 3MB.
- **PC B (Leecher):** Desktop cu IP-ul 192.168.137.239.

  Rol: Descarcă fișierul de la PC A.

Pasul 1: Inițializarea Seeder-ului (PC A)

Pe primul PC a fost pornită aplicația și s-a selectat fișierul pentru partajare. Aplicația a generat fișierul .torrent, a calculat hash-urile pieselor și a început să anunțe disponibilitatea fișierului prin mesaje multicast.

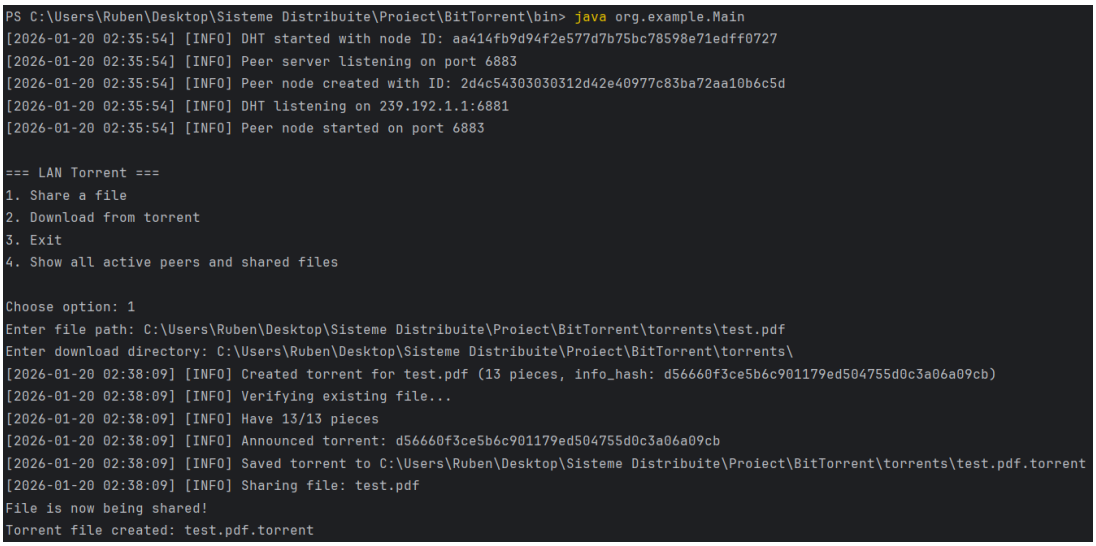

Figura 1: Seeder (Partajare fișier)

Pasul 2: Descoperirea și Transferul Datelor (PC B)

Aplicația a fost rulată pe al doilea PC și a fost încărcat fișierul .torrent generat anterior.

Aplicația trimite o cerere GET_PEERS în rețea. PC B primește un răspuns de la PC A conținând IP-ul și portul acesteia. Se inițiază conectarea, iar apoi începe să solicite piese, iar PC A le livrează.

După descărcarea tuturor pieselor, PC B a asamblat fișierul final. Hash-ul total al fișierului a fost comparat cu cel original, confirmând că transferul s-a realizat fără erori de biți.

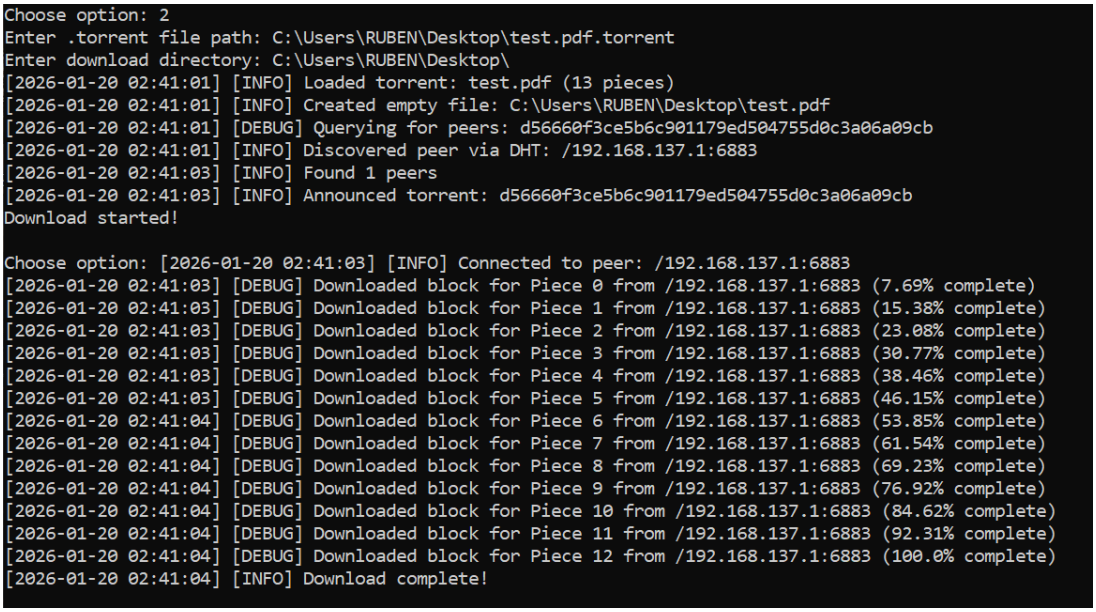

Figura 2: Leecher (Descărcare fișier)

11 Link Github

https://github.com/ruben-23/sisteme-distribuite/tree/main/Proiect/BitTorrent

12 Concluzii

Acest sistem reprezintă o implementare funcțională a nucleului protocolului BitTorrent, adaptată pentru medii locale prin înlocuirea tracker-ului centralizat cu un DHT Multicast. Arhitectura modulară permite extinderea ulterioară (ex: adăugarea criptării sau a unui DHT global Kademlia).