Spécifications Techniques : Projet P2P Java Swing

Architecture Multithreading & Réseau

12 février 2026

Objectif du Projet

Développer une application Peer-to-Peer (P2P) permettant le partage et le téléchargement de fichiers en réseau local. L'application doit intégrer une interface graphique réactive et gérer plusieurs connexions simultanées grâce au multithreading.

1 Architecture Logicielle

Dans ce modèle P2P, chaque instance du programme agit comme un **nœud hybride**. Voici les classes structurantes :

1.1 Le Cœur du Système

- `PeerNode` : Classe principale de données. Elle stocke les paramètres de session (IP, Port d'écoute) et l'état des connexions actives.
- `FileService` : Gère les entrées/sorties physiques sur le disque dans le dossier `/partage`.

1.2 Le Réseau et Multithreading

Gestion des Flux

- `ServerTask` **(extends Thread)** : Boucle d'écoute infinie. Elle utilise `ServerSocket.accept()` pour détecter les pairs entrants.
- `ConnectionHandler` **(implements Runnable)** : Instanciée pour chaque nouveau client. Elle permet de lire les commandes (LIST, GET) sans bloquer le serveur.
- `ClientService` : Gère l'émission de requêtes vers d'autres serveurs (méthodes de connexion sortantes).

2 To-Do List Détaillée

Phase 1 : Infrastructure de fichiers

- Création du dossier `/partage`.

— Implémentation de la lecture de répertoire avec `java.io.File`.

- **Phase 2 : Serveur Multithread**
  - Initialisation du `ServerSocket`.
  - Mise en place du `while(true)` pour l'acceptation des connexions.
  - Transmission du socket au `ConnectionHandler`.
- **Phase 3 : Protocole de Communication**
  - **Requête LIST** : Envoyer la liste des noms de fichiers (ex : via `ObjectOutputStream`).
  - **Requête GET** : Transférer les octets du fichier par blocs (buffers) de 4096 octets.
- **Phase 4 : Interface Swing**
  - Création de la `JFrame` avec `BorderLayout`.
  - Utilisation de `DefaultListModel` pour mettre à jour la liste des fichiers à l'écran.
  - **Crucial** : Implémenter `SwingWorker` pour les téléchargements afin de ne pas geler l'UI.

3 Détails Techniques Importants

3.1 Le dossier de partage

Chaque instance doit pointer vers son propre répertoire. En local, vous pouvez différencier les dossiers par le numéro de port utilisé : `partage_8080/`, `partage_8081/`.

3.2 Éviter le gel de l'interface

Toute opération réseau doit être faite hors du *Event Dispatch Thread* (EDT).

```
// Exemple de SwingWorker pour le t l chargement
SwingWorker<Void, Integer> downloadTask = new SwingWorker<>() {
    @Override
    protected Void doInBackground() {
        // Logique de t l chargement r seau ici
        return null;
    }
};
downloadTask.execute();
```

Attention

N'oubliez pas d'appeler `socket.close()` et de gérer les `IOException` pour éviter les fuites de mémoire et les ports bloqués sur votre machine.