Contexte

Dans ce projet, nous avons mis en place un protocole client-serveur permettant la gestion de fichiers à distance. Le client peut s'enregistrer auprès du serveur, lister les fichiers disponibles, en lire ou en écrire à distance. La communication s'effectue à l'aide de commandes textuelles structurées (REGISTER, LS, READ, WRITE) échangées via des sockets TCP. Le protocole est conçu pour être utilisé dans un environnement local avec possibilité d'extension à plusieurs serveurs, permettant une forme de redirection en cas de demande de fichier non local.

Analyse

Hypothèses formulées :

- Chaque client doit s'enregistrer avant d'effectuer toute opération.
- Un jeton d'authentification (token) est généré pour chaque client après l'enregistrement, et il est requis pour toutes les commandes suivantes.
- Les fichiers sont listés dans un fichier `FilesListe.txt` côté serveur.
- Les fichiers du client à envoyer sont situés dans un dossier `Client_files`.
- Les fichiers reçus côté serveur sont stockés dans un dossier `Stockage_fichier`.

Limitations rencontrées :

- La spécification ne précise pas le format des jetons ni leur durée de validité.
- Aucune information sur la gestion des erreurs en cas de jeton invalide ou de fichier introuvable.
- Le protocole ne gère pas la persistance de la connexion pour plusieurs commandes successives (problème rencontré dans WRITE et READ).

Raisons et justifications :

Nous avons décidé d'utiliser un système de redirection dans le cas où un serveur ne possède pas le fichier demandé. Il consulte alors les informations IP/port associées au fichier dans son `FilesListe.txt` pour orienter le client vers un autre serveur. Toutefois, cette redirection implique une nouvelle session et donc un nouveau jeton. Cela a causé un échec lors de l'accès au fichier, car le jeton initial n'était pas reconnu par le second serveur.

Exemples de scénarios :

- Le client demande un fichier non local : le serveur redirige mais le nouveau serveur rejette la commande à cause d'un jeton inconnu.
- Lors de plusieurs commandes READ/WRITE successives, la connexion se ferme, forçant une recompilation du client.

Conception

- **Client**
  - Attributs : adresse IP, port, jeton, dossier Fichiers
  - Méthodes : register(), listFiles(), readFile(), writeFile()
- **Serveur**
  - Attributs : adresseIP, port, listeFichiers, dossierStockage
  - Méthodes : accepterConnexion(), traiterCommande(), genererToken(), envoyerFichier(), recevoirFichier(), redirigerClient()

Role que joue chaque commande :

- **REGISTER**
  - Client → Serveur : REGISTER|IP|
  - Serveur → Client : TOKEN|jeton_unique|
- **LS (list files)**
  - Client → Serveur : LS|jeton|
  - Serveur → Client : liste des fichiers depuis FilesListe.txt
- **READ**
  - Client → Serveur : READ|jeton|nom_fichier|
  - Serveur vérifie si le fichier est local :
    - Oui → envoie le fichier
    - Non → redirige vers un autre serveur
- **WRITE**
  - Client → Serveur : WRITE|jeton|nom_fichier|
  - Serveur → reçoit le fichier et le stocke dans `Stockage_fichier`

Conclusion

Ce projet nous a permis de mettre en œuvre un protocole client-serveur simple avec gestion de fichiers, communication réseau, traitement de commandes et génération de

jetons d’authentification. Nous avons appris à structurer un échange client-serveur via des sockets TCP et à gérer les flux de données entrants/sortants.
Les principales difficultés ont été liées :

- à la gestion de la connexion persistante (fermeture automatique après certaines commandes),
- aux boucles non optimales qui ralentissaient le transfert,
- à la redirection inter-serveurs, notamment à cause de jetons non reconnus.

Pour une version améliorée, il serait judicieux :

- d’ajouter une gestion centralisée des jetons (ex. : base de données partagée),
- d’assurer la persistance de la connexion pour plusieurs commandes successives,
- de mieux gérer les erreurs et les cas de redirection.

Répartition des tâches

Membre	Tâches réalisées
Kadiatou	Implémentation des commandes côté client (REGISTER, LS,
Lamarana Bah	READ, WRITE)
Mamadou	Conception du serveur, gestion du fichier `FilesListe.txt`,
Sanoussy Bah	redirection, compilation