Le but de ce projet et d'implémenter un client pour un protocole d'échange de fichiers pair à pair.

1 Principe général

1.1 1ère Partie

Ce projet doit implémenter un programme qui participe à un réseau d'échanges de fichiers pair-à-pair. Le protocole utilisé est décrit dans ce sujet. À chaque instant le programme est connecté à un ensemble d'autres programmes (pair) implémentant le même protocole auquel il peut :

— Envoyer un message textuel sous forme d'une chaîne de caractères.
— Demander la liste des pairs connus.
— Demander la liste des fichiers que le pair possède.
— Demander à recevoir un fragment d'un des fichiers qu'il possède.

Le programme devra lui-même répondre aux mêmes requêtes. Le programme devra écouter sur un port arbitraire pour recevoir de nouvelles connexions et être capable de se connecter à d'autres clients. Le programme devra posséder un ensemble de fichiers qu'il partage.

L'ensemble E des pairs connu contient les pairs pour lesquels le programme connais le port sur lequel ce pair accepte de nouvelle connexions. Le protocole permet à un pair de transmettre le numéro de port sur lequel il attend des connexions.

1.2 2ième Partie

Une interface utilisateur simple (graphique avec Swing ou en ligne de commande) devra permettre de consulter la liste des pairs connectés. Et devra permettre de faire une recherche dans la liste de leurs fichiers disponible ainsi que de télécharger ces fichiers.

Cette interface ne sera pas notée sur son esthétisme. Elle devra être exécutée dans un thread indépendant de la première partie.

2 Protocole

Le protocole utilisé est construit par dessus TCP/IP. Une connexion à un pair est établie soit en effectuant un "connect" à une adresse connue ; soit en recevant une demande de connexions. Dans les deux cas, le même protocole est utilisé. Toutes les données sont transmises octet par octet en "big endian" (le défaut pour les ByteBuffers). Les entiers (int) utilisent 4 octets, les entiers long (long) utilisent 8 octets, les chaînes de caractères sont encodées en "UTF-8" et sont précédées d'un entier encodant leurs tailles en octets.

Le protocole consiste en un ensemble de messages commençant par un identifiant (ID) codé par un octet

— (ID=1) [1, STRING] Message de texte. Uniquement informatif.
— (ID=2) [2, INT] Déclaration du port en écoute. Le programme recevant ce message peut ajouté la source du message et le numéro de port à la liste des pairs connus.
— (ID=3) [3] Demande de la liste des pairs connus. Le programme doit répondre avec un message de type 4.
— (ID=4) [4, INT, [INT, STRING]*] Renvoi de la liste E des pairs connues, le premier entier est le nombre de pairs n. Chaque pair est décrit par le numéro de port sur lequel il

attend des connexions et son URL sous forme de chaîne de caractères. Le nombre de pairs envoyé doit être égal à n.

— (ID=5) [5] Demande de la liste des fichiers disponibles. Le programme doit répondre avec un message de type 6.

— (ID=6) [6, INT, [STRING, LONG]*] Renvoi la liste des fichiers disponibles, le premier entier est le nombre de fichiers n. Chaque fichier est décrit par son nom sous forme d'une chaîne de caractères et par sa taille.

— (ID=7) [7, STRING, LONG, LONG, INT] Demande d'un fragment d'un fichier. La chaîne de caractères est le nom du fichier, le premier entier long est la taille totale du fichier. Le second entier long est la position du début du fichier demandé, l'entier est la taille du fragment demandé. La taille maximale autorisée pour les fragments est de 65536 octets une demande plus longue est considérée comme insatisfiable. Le programme doit répondre avec un message de type 8.

— (ID=8) [8, STRING, LONG, LONG, INT, BLOB] Renvoi d'un fragment de fichier. Le début de ce message est le même que la demande d'un fragment. Le dernier élément BLOB est un bloc de n octets où n est la taille du fragment demandé.

— (ID>=64 & ID<128) [ID, INT, BLOB] Message d'extension. L'entier spécifie un nombre d'octets n. BLOB est une suite de n octets. Ce type de message permet d'étendre le protocole avec des messages additionnels. Ces messages peuvent toujours être ignorés.

— Toute autre valeur pour un identifiant est un message mal formé.

Le protocole ne contient pas de système d'erreur. Si un programme reçoit un message auquel il ne peut répondre (message mal formé ou requête insatisfiable) il doit fermer la connexion, il peut envoyer un message texte de type 1 avant la déconnexion pour expliquer l'erreur.

3 Extensions

Vous pouvez implémenter le téléchargement de fichier depuis plusieurs pairs simultanément pour améliorer le debit (comme BitTorrent). Dans ce cas, vous pouvez supposer que deux fichiers de même nom et de même taille sont les mêmes.

Vous pouvez étendre le protocole pour implémenter des fonctionnalités supplémentaires en utilisant des messages d'extension.

4 Évaluation

Le projet doit être réalisé en Java. Plusieur serveur avec une implémentation de référence tournent a l'adresse :

— `prog-reseau-m1.lacl.fr` sur le port 5486

Votre projet devra être capable de se connecter à ces serveurs et de récupérer les fichiers présents. Attention, le port 5486 est bloqué sur certain wifi (Etudiants-Paris12, eduroam).

Votre implémentation du protocole doit utiliser les concepts vus en cours notamment la sérialisation. Il peut utiliser des entrées-sorties bloquantes ou non bloquantes. Une explication des classes utilisées pour modéliser le protocole est attendue à la soutenance.

Le projet doit être réalisé en binôme, et soumis grâce à un répertoire git. Chaque binôme devra s'inscrire dans un groupe sur Eprel avant le jeudi 14 mars 2019. Suite à cette inscription, vous recevrez un email pour avec les identifiants pour vous connecter sur `git-etudiants.lacl.fr`.

Vous devrez effectuer des commits réguliers sur ce répertoire ($\approx$ 1 par semaine). La version finale devra contenir un fichier “README” avec les instructions pour compiler et exécuter votre projet en ligne de commande.