Groupe Tuaillon Pierre, Pierre Laeticia, Milanesio Alexandre

Pour envoyer un message : Pour envoyer un message l'émetteur utilise la fonction forwardMessage qui prend en paramètre l'id de l'émetteur ainsi que le message. Le message n'est pas affiché s'il est envoyé 2 fois d'affilés. Le message est ensuite ajouté dans le tampon du salon pour l'afficher aux nouveaux arrivants.

Pour la transmission du message, on cherche le successeur qui fait partie du même salon. On crée une socket auprès de celui-ci. Puis on passe par l'OutputStream de la socket et Writer pour lui transmettre le message. Et enfin on l'ajoute à la liste des chaines stockées du ChorPeer.

Pour la réception, nous disposons d'un Thread service client qui gère la réception du message via l'inputStream. On ajoute à notre interface graphique. Et la transmission continue jusqu'à la dernière chaîne stockée identique.

Pour les communications externes, nous souhaitions utiliser le format JSON pour gérer les requêtes GET/POST. Nous avons importé la librairie externe javax et décrit la procédure de lecture et d'écriture de ce format dans la classe ComJsonReader mais ne nous n'avons pas implémenté ce format dans le projet.

Pour rejoindre l'anneau : On récupère l'instance annuaire de la classe Client. Si celle-ci n'existe pas, le client arrivant se propose en tant que ChordPeer référent. Sinon, il recupère le ChordPeer référent et apppel la méthode joinMainChord pour initialiser ses références (avec une potentielle màj de la maxkey). Le nouveau client dispose d'un ServerSocket avec un thread pour la méthode accept().

Pour quitter l'anneau : Lors de la fermeture de la fenêtre, leaveMainChord est appelé pour garder le système dans un état cohérent (s'il est la référence de l'annuaire, son successeur le devient ; s'il est la maxkey, on remplace par celle en dessous ; on met à jour les références de son successeur et prédécesseur). Puis, un message est envoyé aux autres clients restants pour qu'ils soient alertés de la déconnexion. Pour gérer les déconnexions intempestives (méthode non implémentée), nous aurions souhaité utiliser un thread (voir classe VerificationBadDeco) qui demanderait à chaque noeud de "ping" (méthode isReachable de inetAddress) son successeur pour tester s'il est toujours connecté. Si celui-ci n'est plus connecté, on cherche le prochain successeur qui l'est pour mettre à jour les références.

Fonctionnement des salons : Les échanges de messages se font uniquement entre membres du même salon. Il existe un salon par défaut (Default) sur lequel les nouveaux arrivants se connectent. Les clients peuvent changer de salon en écrivant le nom de celui-ci dans l'espace approprié de l'interface. A ce moment, joinChatRoom est appelé : si le salon existe déjà (référence à la key indiquée), il est renvoyé et utilisé, sinon, le client crée un nouveau salon. En arrivant sur le salon, les anciens messages envoyés avec succès sur le salon (qui sont stockés dans le tampon de celui-ci) sont

parcourus et affichés sur l’interface avec readChatRoom. Pour obtenir la liste des salons, getChatRoomsList parcours tous les noeuds en les interrogeant sur leur salon. Les salons sont définis par leur nom et une key (le nom en hashCode).