Infrastructure de partage de fichiers

L'objectif du TP est de développer une petite infrastructure de partage de fichiers textes et binaires de type peer to peer. Les infrastructures de ce type (Gnutella, Freenet, Kaaza, ...) offrent aux usagers la possibilité de se joindre à une communauté. Non seulement une telle infrastructure permet d'accéder à des ressources distantes mais elle permet aussi de partager ses propres ressources en les offrant à la communauté.

1 Fonctionnalités

Les utilisateurs pourront accéder aux fichiers des autres utilisateurs et rendre leurs fichiers accessibles. Dans un but de simplification, la mise à disposition de fichiers se fera au niveau d'un unique répertoire sur la machine de chaque utilisateur. Ainsi, il n'est pas demandé d'assurer la gestion d'arborescence de répertoires et les primitives de navigation associées : tout fichier déposé dans le répertoire de partage sera disponible aux autres usagers. Le logiciel demandé regroupe donc la gestion du chargement et de la mise à disposition de fichiers. Pour faciliter l'accès aux fichiers disponibles dans l'environnement, une petite interface graphique permet de visualiser les nœuds de l'environnement, pour un nœud donné les fichiers exportés, et de charger un fichier présent sur un nœud distant.

2 Architecture

De par la réalisation en Java RMI, l'environnement est architecturé autour d'un registre RMI et d'un nœud racine. La création d'une communauté se fait par l'activation d'un registre et l'activation d'un nœud considéré comme racine. L'ajout d'un nœud dans la communauté revient donc à rechercher auprès du registre RMI le nœud racine, de lui demander les nœuds disponibles, et de s'enregistrer auprès de ces nœuds y compris la racine.

Un nœud doit gérer les informations suivantes :
- la liste des nœuds qu'il connaît
- la liste des fichiers disponibles localement

Un nœud peut être structuré de la manière suivante :
- un objet accessible à distance permettant à un nœud de fournir la liste de ses fichiers et leur contenu, et la possibilité de s'en faire connaître
- un objet fournissant une abstraction du système de fichiers (lister, lire et écrire des fichiers)
- une interface graphique permettant de parcourir les nœuds et leur contenu facilement
- un contrôleur gérant les interactions entre l'interface graphique et le système de fichiers local et les nœuds distants.

Enfin un nœud doit gérer la disparition d'un nœud connu autre que la racine. La panne de la racine ne permettant pas l'intégration de nouveaux nœuds dans le système, les nœuds restant ne pourront fonctionner qu'en autarcie. Il est donc concevable de choisir l'arrêt complet du système (tous les nœuds) lorsque la racine rencontre un problème.

3 Pour aller plus loin

Quelles sont les limitations des choix actuels de mise en œuvre ? Que proposez vous pour améliorer l'architecture ? Plutôt que présenter les fichiers par nœud, les présenter dans leur ensemble en masquant leur localisation ou en proposant plusieurs localisations au moment du téléchargement. Proposer une mise en œuvre de la recherche de fichiers par nom avec jokers, en donnant un fichier l'environnement recherche sur tous les nœuds connus si le fichier est disponible. Quels sont les problèmes soulevés par une recherche répartie de l'information ?