Tiny Torrent

Projet Miage Info Master 1

9 décembre 2012

1 Introduction

L'objectif de ce projet est d'implémenter une version simplifiée du protocole Bittorent. Le principe de fonctionnement est le suivant : lorsqu'un client C_1 souhaite partager une ressource (image, vidéo, etc.) il doit tout d'abord découper la ressource en morceaux (chunks), puis créer un fichier `.tinytorrent`. Ensuite, le client C_1 doit s'enregistrer au près d'un serveur nommé Tracker afin d'offrir la possibilité à d'autres clients (leechers) de connaître quels sont les clients (seeders) disposant d'une ressource.

Un client C_2 souhaitant télécharger une ressource doit, quant à lui, commencer par récupérer le `.tinytorrent` associé à la ressource dont il souhaite disposer. Cela peut se faire via divers moyens : liste de diffusion, annuaire de fichiers `.tinytorrent`, etc. Le client C_2, ayant récupéré le fichier `.tinytorrent` associé à la ressource qu'il souhaite télécharger, doit contacter le tracker dont l'URL se trouve dans le fichier tinytorrent afin de récupérer les clients possédant un ou plusieurs morceaux de la ressource recherchée. La liste retournée par le tracker contient uniquement une liste de clients et en aucun cas les morceaux gérés par les clients. Il faut donc que le client C_2 contacte chaque client retourné afin de leur demander quels sont les morceaux dont ils disposent. Une fois cette information récupérée, une stratégie de sélection est appliquée pour savoir dans quel ordre et depuis quel clients sont téléchargés les morceaux (e.g. maximiser le nombre de clients contactés, maximiser la bande passante utilisées, récupérer les morceaux les plus rares en premiers, etc.). A chaque fois qu'un nouveau morceau est récupéré, la valeur de hachage du morceau récupéré est comparée à celle du morceau original se trouvant dans le fichier tinytorrent. Si la valeur de hachage est différente, alors le morceau est corrompu et doit être retéléchargé. Dans le cas contraire, une fois que tous les morceaux ont été récupérés, ils sont fusionnés afin de recréer la ressource initiale.

2 Contraintes

1. Chaque client dispose d'un dossier partagé (*shared folder*) dans lequel doivent se trouver les ressources à partager ainsi que les fichiers intermédiaires créés ;
2. Toutes les ressources à partager sont découpées en morceaux de taille 1 Mio ;
3. Un fichier tinytorrent est créé pour chaque ressource à partager. Il est supposé que le fichier tinytorrent créé a le même nom que la ressource partagée auquel est ajouté le suffixe `.tinytorrent` ;
   - un fichier tinytorrent est associé à un seul fichier régulier (pas de répertoire) ;
   - la première ligne du fichier tinytorrent contient le nom de la ressource partagée ;

- la deuxième ligne la valeur de hachage [1] associée à la ressource partagée ;
- la troisième ligne contient l'URL du tracker utilisé pour partager la ressource ;
- les lignes suivantes contiennent la valeur de hachage associée à chacun des morceaux composant la ressource partagée : une valeur de hachage par ligne.

4. Un tracker maintient un lien entre les ressources partagées (identifiées par leur valeur de hachage) et les client disposant d'une partie ou de l'intégralité des morceaux composant la ressource ;

5. Un tracker offre une API permettant d'ajouter ou de supprimer des entrée (ressources partagées, référence client), de rechercher les clients disposant d'une ressource, et les ressources gérées par un client ;

6. Un client offre une API distante permettant de connaître les morceaux dont il dispose pour une valeur de hachage associée à la ressource recherchée ;

   - Si aucun client a les morceaux manquants, il peut soit contacter à nouveau le tracker pour avoir une liste plus à jour des clients (certains ont pu se connecter récemment), soit demander périodiquement aux clients qu'il connaît si ils n'ont pas le morçeau en question.

3 Packages, classes et interfaces

Afin de pouvoir faciliter le développement et le test des projets, il faudra impérativement suivre les conventions de nommage suivantes. Vous n'êtes pas autorisé à modifier la signature des méthodes déclarées dans les interfaces qui vous sont données ; cependant, vous pouvez en ajouter de nouvelles, si vous le souhaitez.

Packages

- `fr.unice.tinytorrent` : package par défaut du projet
- `fr.unice.tinytorrent.api` : interfaces distantes ou non
- `fr.unice.tinytorrent.exceptions` : exceptions
- `fr.unice.tinytorrent.utils` : découpeur de fichier et générateur de `.tinytorrent`

Interfaces

- `fr.unice.tinytorrent.api.RemoteClient` : interface distante du client
- `fr.unice.tinytorrent.api.Tracker` : interface distante du tracker

1. Toutes les valeurs de hachage sont supposées être calculées en utilisant la fonction de hachage SHA-1 dont le résultat est retourné sous la forme de 40 caractères hexadécimaux.

Classes

- `fr.unice.tinytorrent.ClientImpl` : implémentation du client
- `fr.unice.tinytorrent.TrackerImpl` : implémentation du tracker

Un squelette de projet Eclipse est disponible à l'adresse

`https://bitbucket.org/lp/tinytorrent-skeleton/`.

Il contient les interfaces et les classes permettant de créer les fichiers `.tinytorrent`.

3.1 Options de lancement

Un tracker doit pouvoir être démarré en spécifiant le numéro de port utilisé pour le `rmiregistry` dans lequel le tracker va s'enregistrer. Cela implique que vous démarrez le `rmiregistry` de manière programmatique via un appel à `LocateRegistry.createRegistry(int port)`.

Un exemple d'exécution d'un tracker depuis un terminal est donné ci-après :

```
java fr.unice.tinytorrent.TrackerImpl 7986
```

De la même manière un client doit pouvoir être démarré en spécifiant le chemin absolu ou relatif vers son dossier partagé :

```
java fr.unice.tinytorrent.ClientImpl /tmp/tinytorrent/client1/
```

4 À faire

Le projet se fait par groupe de 3 étudiants au maximum. Le travail demandé consiste à :

- implémenter le *Tracker* ainsi que le *Client* au travers des classes `fr.unice.tinytorrent.TrackerImpl` et `fr.unice.tinytorrent.ClientImpl` tout en respectant les contraintes définies dans la section 2. Chaque partie sera évaluée indépendamment ;
- gérer le téléchargement en parallèle des morceaux associés à une ressource ;
- prendre en compte le rafraîchissement de la liste des clients depuis ou les morceaux d'une ressource sont téléchargés (de nouveaux clients peuvent disposer des morceaux d'une ressource au cours du temps) ;
- gérer une stratégie de sélection des morceaux (par exemple en essayant de maximiser le nombre de clients depuis où sont téléchargés les morceaux) ;
- créer des scénarios permettant de tester votre implémentation ;
- écrire une documentation qui décrit vos choix d'implémentation et notamment comment vous gérez l'arrivée de nouveaux clients, quelle stratégie de sélection des morceaux vous avez choisi, etc.
- créer une archive JAR contenant les binaires de votre implémentation ;
- générer la javadoc associée au projet.

Le projet devra être rendu avant le **vendredi 11 Janvier 2013 à 23h59** via l'interface de l'environnement pédagogique Jalon, sous forme d'archive zip dont le nom sera *nom1-nom2-nom3.zip*. Il devra se dézipper dans un répertoire *nom1-nom2-nom3*. Le répertoire devra contenir :

1. le code source de votre projet ;
2. l'archive JAR contenant les binaires de votre implémentation ;
3. la javadoc associée au projet ;
4. un rapport qui décrit vos choix d'implémentation.

5 Question Bonus

Gérer une stratégie particulière de téléchargement des chunks et/ou implémenter un annuaire de fichiers tinytorrents (Registry) que l'on peut contacter pour retrouver un tinytorrent en fonction de mots clefs.