Examen de « Programmation Réseau »
Licence Informatique
Université Paris Diderot

Jean-Baptiste Yunès

Mai 2012

La RFC 959 décrit un protocole de transfert de fichiers (FTP) sur canaux fiables TCP. Son fonctionnement le plus général repose sur trois entités : deux serveurs et un client. Les serveurs s'échangent des contenus de fichiers. On s'inspire de ce protocole pour en inventer un de notre cru, dans lequel le client se propose de contrôler les serveurs en leur envoyant des informations afin d'indiquer comment et dans quel sens le transfert se produit. On se propose ici d'implémenter trois entités de cette sorte séparément (le client, un serveur - appelé envoyeur - qui sera chargé d'envoyer le contenu d'un fichier et un serveur - appelé récepteur - qui sera chargé de réceptionner le contenu du fichier).
Les deux serveurs (au départ non distingués) écoutent toujours sur le port P (P sera une constante dans les différents programmes à écrire) les ordres en provenance du client.

1. pourquoi, compte-tenu de la description donnée, les deux entités serveurs sont nécessairement sur deux « machines » différentes ? Vous pouvez raffiner la notion de « machine » invoquée ici...

Le client se connecte aux deux serveurs (donc sur leurs ports P) et décide que l'un sera l'envoyeur et l'autre le récepteur. Pour se faire il (le client) enverra un message à l'un d'entre eux (serveurs) permettant de le positionner en mode récepteur. Ce message sera constitué du seul octet contenant le code ASCII du caractère `'R'`. En réponse à ce message le serveur - devenu récepteur - renverra sur le canal TCP, un numéro de port sur lequel l'envoyeur pourra se connecter pour transmettre le contenu du fichier. Le message sera simplement constitué de deux octets, codant en big-endian le numéro de port en question.

2. écrire une méthode statique Java de nom `recepteur` qui :
   - prend en paramètre une `Socket s`,
   - recherche un port libre (en faisant peut-être des essais) pour y attacher une `ServerSocket` de nom `ss`
   - en cas de réussite, renvoie au client le message adéquat (numéro de port en big-endian) sur `s` et fait appel à la méthode `lit(ss);`
   - en cas d'échec, renvoie au client le message indiquant le numéro de port le port 0 et lève une exception.

Une fois la réponse reçue - si elle est positive, *i.e.* port non-nul - , le client se connecte à l'autre serveur pour le placer en mode envoyeur. Pour cela, il lui envoie un message constitué du code ASCII du caractère `'E'`, suivi par 4 octets codant en big-endian l'adresse du récepteur, puis par deux octets codant en big-endian le numéro du port sur lequel il essaiera de se connecter pour réaliser le transfert.

3. écrire en Java une méthode statique de nom `envoyeur` qui :
   - prend en paramètre une `Socket s`,
   - en extrait les 4 octets constituant l'adresse du récepteur et les 2 octets constituant le numéro de port sur lequel le récepteur écoute,
   - créé une `Socket` de nom `st` connectée sur celle du récepteur,
   - en cas de réussite, renvoie au client l'octet `'Y'` et appelle la méthode `ecrit(st);`
   - en cas d'échec, renvoie au client l'octet `'N'` et lève une exception.

4. écrire en Java la méthode statique `lit(Socket s)` qui créé un `Thread` qui :
   - attend une connexion entrante sur `s` reçue en paramètre,
   - transfère les octets lus dans la `Socket` dans un fichier sur disque par appel à une méthode `transfert(Socket s,String nom);` qui fait le travail,
   - s'arrête proprement
5. écrire en Java la méthode statique `ecrit(Socket s)` qui créé un `Thread` qui :
   - transfère dans `s` le contenu d'un fichier lu sur disque par appel à une méthode `transfert(String nom,Socket s);` qui fait le travail,
   - s'arrête proprement
6. écrire la méthode statique `main` de la classe `ServeurTransfertFichier` qui :
   - créé une `ServerSocket` attachée sur le port P,
   - s'y place en écoute,
   - attend l'arrivée d'un octet sur la `Socket` de service
   - si l'octet est '`R`', appelle la méthode `recepteur` (de la question 2) sur la `Socket` de service ;
   - si l'octet est '`E`', appelle la méthode `enyoyeur` (de la question 3) sur la `Socket` de service ;
   - traite l'erreur (comme on peut) sinon.
7. écrire le code de la méthode statique `main` de la classe `ClientTransfertFichier` qui :
   - reçoit en paramètre `String []args`, le tableau constitué des noms Internet des deux machines sur lesquelles s'exécutent les deux serveurs (`args[0]` sera l'adresse ou le nom de l'envoyeur et `args[1]` l'adresse ou le nom du récepteur),
   - envoie la commande '`R`' au récepteur,
   - envoie la commande '`E`' avec les 6 octets qui doivent suivre, à l'envoyeur
   - quitte proprement (avec ou sans exception).
8. décrire les commandes qui doivent être exécutées afin d'obtenir un transfert de fichier (commandes lancées sur quelles machines, dans quel ordre, etc) - faites toutes les suppositions que vous voulez sur les adresses des machines, etc.
9. écrire en C le code du client qui possède les mêmes fonctionnalités que sa version Java
10. on désire modifier le tout pour permettre d'indiquer le nom du fichier à transférer lors du lancement du client. Quelles sont les modifications nécessaires :
    (a) dans le protocole/les messages ?
    (b) dans le code ?