Réalisation d'un outil pour la synchronisation de fichiers

Hakim Ferrier-Belhaouari

Master Informatique de l'Université de Poitiers 2018-2019

1 Contexte :

Un client NON-INFORMATICIEN (ex. petite PME) souhaite réaliser un outil capable de synchroniser ses fichiers sur deux périphériques différents. En particulier, il souhaite utiliser ce logiciel pour faire des duplications de sauvegarde de son travail ou pour le transfert automatique de fichier sur différents systèmes d'exploitation (Windows, Linux, Android, ...). L'hétérogénéité des machines impose une synchronisation non-symétrique afin d'éviter des problèmes de place sur les espaces de stockage. Par conséquent, le client définit comme source l'entité émettrice de la demande de synchronisation et la destination comme l'espace de stockage final. Dans ce cas, la synchronisation par défaut souhaitée par le client est d'envoyer depuis la source sur la destination tous les fichiers nouvellement créés ou modifiés et de récupérer en retour uniquement les fichiers modifiés dans la destination et existant sur la source. Bien évidemment, pour des raisons de commodité un utilisateur peut forcer l'importation d'éléments contenus dans la destination vers la source et vice-versa. De plus, pour éviter tout problème de mauvaise manipulation le client souhaiterait avoir un mécanisme d'archivage de dossier sur la destination capable de sauvegarder les 5 dernières modifications des fichiers du répertoire à condition que l'utilisateur le souhaite (sinon on gaspillerait beaucoup de place inutilement). Enfin, l'ensemble de ces informations est contenu dans un fichier de configuration que nous appellerons profil. Ce dernier contient entre autre, le nom et les adresses IP des machines, le répertoire source et le répertoire destination, la liste des fichiers acceptant le mécanisme d'archivage et toutes les informations que vous jugerez nécessaire pour la synchronisation.

2 Réalisation

1. Le logiciel se décompose en deux parties, une partie client et une partie serveur. Ces deux parties seront implantées dans des langages différents.
2. La réalisation de cet outil devra cependant, inclure un tronc commun afin que le client final puisse choisir l'implémentation de son client et du serveur tout en ayant des fonctionnalités minimales à ces demandes. (voir ci-dessous)
3. Vous proposerez au moins 1 grosse fonctionnalité innovante de votre projet (ou plusieurs petites à valider avec vos encadrants) et que vous implémenterez.

3 Tronc commun

Les informations que nous donnons ici représente le tronc commun entre tous les groupes d'étudiants et par conséquent doivent être respectées. Cependant en dehors de ces limitations, vous pouvez réaliser autant de chose qu'il vous plaira.

Pour illustrer notre exemple de protocole nous considérerons que le répertoire source (sur la source) est `/home/masource/` et le répertoire destination (sur la destination) est `/home/madestination/` et les deux machines possèdent déjà un profil compatible.

L'ensemble des communications (du tronc commun) doivent se faire via les sockets TCP/IP sur le port 23768 initié par le client bien évidemment. Vous pouvez gérer les erreurs comme vous l'entendez du moment qu'elles restent compatible avec notre protocole de base.

1. le client envoie le message texte suivant : `PROJET GL1 2018`

2. Dans ce cas le serveur répond : `OK` (sinon le serveur envoie un message ou non mais ferme la socket)
3. Le client envoie un ensemble d'information avec la balise `INFO`. En particulier le nom de la machine (ou l'adresse IP) et le nom du profil (qui pourra servir à des optimisations dans le traitement du scan). Par exemple, si la machine nommée virtualpc souhaite synchroniser le profil Photos il enverra la ligne suivante : `INFO virtualpc Photos`
4. Le client envoie le répertoire source : `SRC /home/masource`
5. Le client envoie le répertoire destination : `DEST /home/madestination`
6. Le serveur répond : `OK` si le répertoire existe et autre chose si le répertoire n'existe pas ou est indisponible (la socket se ferme alors)
7. Le serveur scanne les fichiers du répertoire et cherche les informations de taille (en octets) et date de dernière modification (en secondes) et les envoies au client. (ex : `SCAN /home/madestination/toto/foo 12345 29/09/2018 11:28:34`)
8. Le client peut envoyer les commandes suivantes :
   (a) `TIME` demande au serveur de lui envoyer l'heure sur sa machine (au format 29/09/2018 23 :58 :59)
   (b) `STOPSCAN` demande au serveur d'arrêter le scan car ce dernier en a pas besoin (le serveur répond `OK` ou `NOK`)
   (c) `RESCAN` demande au serveur d'arrêter un scan en cours et de relancer un nouveau scan (le serveur répond `OK` ou `NOK`)
   (d) `SEND /home/madestination/toto/foobar 1234` informe le serveur que le client va envoyer le fichier /home/madestination/toto/foobar (en binaire) faisant une taille de 1234 octets. Le serveur répond `OK` ou `NOK`
   (e) `RECV /home/madestination/titi/note.txt` demande au serveur de lui envoyer le fichier passé en paramètre. Le serveur vous enverra alors la ligne SEND formater comme précédemment.
   (f) `ARCHCOUNT /home/madestination/toto/foo` demande au serveur le nombre d'archive pour ce fichier
   (g) `ARCHIVE 2 /home/madestination/toto/foo` demande au serveur l'archive numéro 2 pour un fichier.
9. A la fin d'un scan le serveur envoie le message `ENDSCAN`
10. Le client ou le serveur peuvent envoyer `BYE` qui termine la communication proprement en fin de traitement ou en cas de timeout d'un des deux cotés.

4 Remarques

— On ne cherche pas à faire compliqué !
— Le `NOK` peut être plus loquace (en ajoutant un message par exemple), si vous le souhaitez mais à terme il faut fermer la socket en cours.
— Le point 6 peut se faire au « fil de l'eau » ou par lot ou d'un seul coup suivant votre implémentation.
— Attention les points 7 et 8 peuvent être en parallèle.
— La numérotation des archives, le numéro 0 est le fichier en cours et plus on monte et plus on va vers les anciennes versions (donc i est plus récent que i+1)
— Les commandes ou les traitements donnés ci-avant, sont là uniquement pour vous aider et surtout pour pouvoir tester votre application.
— Si le delta entre l'heure du client et du serveur est supérieur à 5 minutes il faut quitter l'application. On considère que notre client attendra au moins 5 minutes entre toutes ces synchro et que les machines soient à une heure raisonnable (même fuseau horaire).