UNIVERSITÉ TOULOUSE Jean Jaurès

Projet Système/Réseau

Dans le cadre d'une simulation de comportements de robots, nous souhaitons mettre en place un serveur gérant une carte contenant des ressources. Des clients se connectent sur le serveur afin de tester le comportement de leur robot (1 robot/client). Le comportement du robot consiste à explorer la carte tout en indiquant sa position et les ressources récupérées.

Première partie : rédaction de la RFC décrivant le protocole SpaceX

Dans cette partie, vous devez rédiger un document sur le modèle des RFC existantes (celle de POP3, par exemple). Il sera remis au format texte, sur le modèle des RFC et de préférence en anglais. Le format complet des RFC est décrit par la RFC 2223, notamment sa section 3a.

Cette RFC doit décrire le protocole permettant à des clients de se connecter au serveur SpaceX et au serveur de dialoguer avec ses clients via une connexion TCP.

Le serveur attend les connexions des clients, qui doivent proposer un pseudo. Si ce pseudo n'est pas déjà pris, le serveur connecte le client sous le pseudo demandé.

Une fois connecté, le client reçoit une copie de la carte qu'il peut afficher en local et qui doit être rafraîchie au fur et à mesure de la simulation. Le client demande l'ajout de son robot sur une case en particulier. Selon l'état de la case, le serveur accepte ou non l'emplacement. Une fois placé, le robot peut commencer son exploration.

L'exploration peut se faire d'une manière aléatoire, à chaque mouvement le robot doit indiquer son nouvel emplacement au serveur SpaceX, qui peut valider ou non le déplacement. Si le déplacement est validé, le serveur indique si des ressources sont disponibles sur la case où se trouve le robot.

De plus, les clients connectés peuvent envoyer des « messages de contrôle » pour contrôler leur propre robot. La RFC doit imposer les messages de contrôle suivants (dans cette liste, les termes *cmdX* seront remplacés par les mots-clés que vous aurez choisis : *cmd1* pourrait donc être *fin*, *exit* ou *quit*, par exemple). Ces termes ne sont pas sensibles à la casse (mais leurs paramètres le sont) :

- cmd1 : le client met fin à la connexion.
- cmd2 : le client reste connecté et met son robot en pause.
- cmd3 : le client remet son robot qui en pause dans l'état actif.
- cmd4 : le client demande la liste des pseudos connectés et les ressources récoltées.
- cmd5 pseudo : le client change de pseudo
- cmd6 pseudo : le client indique à pseudo qu'il désire recevoir le code de son robot, le destinataire peut alors autoriser ou non ce transfert. Si le transfert est accepté, il démarre automatiquement. Lorsque les deux extrémités se sont mises d'accord, le transfert s'effectue en peer to peer, sans passer par le serveur. Le fichier doit être considéré comme du binaire. Le protocole doit donc gérer la connexion directe entre deux clients (choix du port et du protocole de transfert, notamment).

Dans tous les cas, les réponses du serveur doivent être accompagnés de codes d'état (qu'il vous appartient de définir) afin de prévenir les clients que leur requête a été acceptée ou qu'il y a eu une erreur (et, éventuellement, le type de cette erreur). Ces codes doivent être classés en familles, selon le modèle des codes HTTP, par exemple.

En outre, votre RFC doit bien préciser les formats des réponses du serveur à chaque requête des clients. Bien entendu, il s'agit ici d'une définition minimale. Vous pouvez, par exemple, imposer un horodatage des messages, des connexions, etc. ou rajouter des messages de contrôle qui vous sembleraient utiles.

NB :

- On insiste bien sur le fait qu'il s'agit ici d'une RFC et qu'il n'est pas question de problème d'implémentation.
- Les spécifications mentionnées ci-dessus sont le « minimum syndical » : vous pouvez laisser libre-cours à votre imagination pour définir bien d'autres fonctionnalités — mais attention, vous devrez ensuite les implémenter.

Deuxième partie : Implémentation

Écrivez une implémentation d'un serveur SpaceX en Python et d'un client SpaceX avec le toolkit graphique et le langage de votre choix, conformes à la RFC que vous avez définie.

Travail à rendre

- La RFC au format .txt
- Le code source du client et du serveur et deux programmes, prêts à être lancés en ligne de Commande
- Une documentation permettant d'utiliser vos deux programmes. Évidemment, les codes sources doivent être correctement documentés (commentaires, docstrings, etc.)

Règles de travail :

- Vous rédigerez la RFC par groupes de 4 que vous enregistrerez auprès de l'intervenant de TP.
- Ce groupe de 4 se séparera ensuite en deux groupes de 2 totalement autonomes qui écriront chacun de leur côté une implémentation de cette RFC (vous aurez compris que les programmes de ces deux groupes devront donc être entièrement compatibles puisqu'ils sont censés implémenter la même RFC).
- Vous n'avez pas le droit d'utiliser une API de plus haut niveau que socket ou son équivalent dans le langage retenu.
- Le client doit être graphique (GTK ou QT, ou Swing avec Java). Il peut utiliser un fichier de configuration *spaceX.conf* stocké dans le répertoire personnel de l'utilisateur.
- Le serveur doit enregistrer dans un fichier journal toutes les commandes qu'il reçoit des clients, avec horodatage de ces messages. Le serveur doit être configurable via la lecture d'un fichier *spaceXserver.conf*, stocké dans le même répertoire que l'exécutable, qui définira son port d'écoute, le chemin d'accès de son fichier log, etc.
- Le serveur et le client doivent pouvoir s'exécuter aussi bien sous Windows que sous Linux ou MacOS.

Conseils :

- Commencez par écrire le serveur et testez-le (tests automatiques et/ou connexions «manuelles » avec telnet).
- Commencez par implémenter la RFC. N'ajoutez les éventuelles extensions que plus tard et documentez-les.