Bataille Navale - TP2

Katleen Blanchet

katleen.blanchet@gmail.com

9 mai 2017

Sommaire

Architecture Client/Serveur

- Un client et un serveur sont deux hôtes, par exemple deux ordinateurs, qui communiquent entre eux par le biais de **sockets**. Le client envoie communément des requêtes au serveur, qui peut y répondre car il a accès à une grande base de données. Les sockets utilisent l'adresse **IP** de l'hôte et un **port** pour établir une connexion unique et gérer les flux entrant et sortant. La connexion se base sur les protocoles **TCP**/UDP. Le protocole TCP est fiable, i.e. que les paquets transmis sont attendus et un accusé de réception est envoyé à l'expéditeur. C'est celui que nous utiliserons. Le mode UDP est utilisé pour le streaming notamment.
- Pour connaître votre IP, tapez *ip addr show* dans un terminal. Adresse après inet en 147.xxx.

Librairie ZeroMQ

Pourquoi ZeroMQ ?

- Echanges simplifiés
- Documentation très détaillée et grande communauté
- API développée pour de nombreux langages

Zguide

- Documentation disponible : http ://zguide.zeromq.org/

Vérification de l'installation de la librairie

- Disponible avec Python3 uniquement sur vos ordinateurs
- Taper 'python3' dans un terminal
- Taper la commande 'import zmq'

1. Requête-Réponse synchrone I

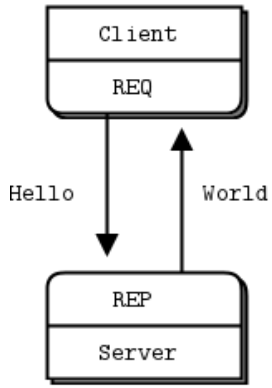

FIGURE – Requête-Réponse (extraite de zguide)

1. Lancer les codes disponibles au début du zguide (hwserver, puis hwclient)
2. Le code du client continue-t-il de tourner à la fin de son exécution ? Si oui, que pouvez-vous rajouter pour arrêter le code proprement ? Mêmes questions pour le code du serveur (même si le while empêchera l'arrêt du code pour l'instant).

1. Requête-Réponse synchrone II

3. Comment la connexion est-elle effectuée suivant le type de socket ?
4. Comment sont envoyées les requêtes et les réponses ? Respectent-elles un ordre ?
5. Que se passe-t-il si vous envoyez plusieurs requêtes sans attendre la réponse ?
6. Modifier le code du client pour que des requêtes puissent être tapées dans le terminal (fonction input()). Ajouter la possibilité d'arrêter la boucle du serveur (avec 'q' par exemple).
7. Que se passe-t-il si vous lancez plusieurs clients en même temps ?

2. Requête-Réponse asynchrone I

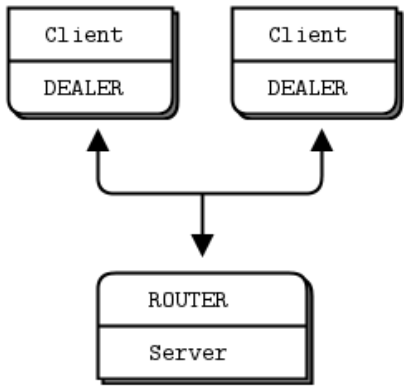

FIGURE – Requête-Réponse asynchrone (extraite de zguide)

- Les sockets ROUTER et DEALER fonctionnent de façon asynchrone, i.e. qu'elles n'ont pas besoin de respecter l'ordre requête-réponse, mais peuvent envoyer plusieurs requêtes ou plusieurs réponses à la suite.
- Elles permettent d'avoir un meilleur contrôle sur les identités de l'expéditeur et du destinataire du message.

2. Requête-Réponse asynchrone II

ROUTER

- Les messages sont envoyés et reçus avec un entête avant le message : [ID Expéditeur/Destinataire][Ø][Message]
- Réception de message : $addr, empty, msg = server.recv\_multipart()$
- Envoi de message : $server.send\_multipart([addr, b'', b"msg"])$

DEALER

- Les messages sont envoyés et reçus avec un caractère nul avant le message : [Ø][Message]
- Réception de message : $empty, msg = client.recv\_multipart()$
- Envoi de message : $client.send\_multipart([b'', b"msg"])$

Les noms (identités) de l'expéditeur et du destinataire peuvent être choisis avec la commande : $socket.setsockopt(zmq.IDENTITY, b'identity')$. S'ils ne sont pas indiqués, ils seront générés automatiquement.

2. Requête-Réponse asynchrone III

1. Reprendre le code précédent et le modifier pour remplacer les REP/REQ par des ROUTER/DEALER.
2. Configurer l'identité du client.
3. Modifier le code du serveur pour envoyer des réponses/requêtes personnalisées aux clients.
4. Lancer le serveur et deux clients.

Jeu : Intégration du Client/Serveur

1. Finir le jeu de base
2. Choisir la bonne structure de client/serveur en respectant le format détaillé page suivante
3. Intégrer le client/serveur au jeu

Jeu : Intégration du Client/Serveur

Format d'échange à respecter

- Tous les échanges s'effectuent en bits.
  - conversion d'un string $s = "text"$ en bits : $s.encode()$ ou $b'text'$
  - conversion d'une chaîne de bits b en une chaîne string : $b.decode()$
- Le serveur envoie :
  - 'NAME' pour obtenir le nom du joueur
  - 'BOATS' pour obtenir la position des bateaux du joueur
  - 'CELL' pour obtenir la cellule visée par le joueur
  - 'END' quand la partie est terminée
- Le client envoie :
  - 'PLAY' pour débuter une partie
  - son nom
  - sa liste de bateaux sous cette forme (sans espace) : $porteAvion : J6J10, croiseur : C1F1, etc.$
  - la cellule visée sous cette forme : $A1$