Institut Français de Géopolitique
M2 Cyberstratégie et terrain numérique

Approche technique de l'espace numérique

Chapitre 8

Introduction à la programmation réseau

Pablo Rauzy <pr@up8.edu>
pablo.rauzy.name/teaching/aten

Introduction à la programmation réseau

- On appelle “programmation réseau” le code qui sert à communiquer entre machines/logiciels, du côté application.
- On utilise pour cela des abstractions qui permettent de ne se soucier que du minimum : les sockets.

Les sockets

- L'idée est que lire et écrire sur une connexion réseau devrait se faire de la même manière que lire et écrire dans un fichier.
- Une socket est un fichier virtuel qui sert de point d'accès au réseau.
- Grâce aux sockets, un client peut se connecter à un serveur.

- Des deux côtés, il faut commencer par ouvrir une socket.
- Pour cela, il faut lui spécifier un domaine, un type, et un protocole.
- Le domaine correspond au protocole de la couche réseau (IPv4 ou IPv6 par exemple).
- Le type va généralement correspondre au protocole de la couche transport (TCP ou UDP).
- Le protocole, dans le cas où le type n'est pas TCP ou UDP mais "RAW", indique le protocole qui va être utilisé au dessus de la couche réseau.

- Des deux côtés, il faut commencer par ouvrir une socket.
- Pour cela, il faut lui spécifier un domaine, un type, et un protocole.
- Le domaine correspond au protocole de la couche réseau (IPv4 ou IPv6 par exemple).
- Le type va généralement correspondre au protocole de la couche transport (TCP ou UDP).
- Le protocole, dans le cas où le type n'est pas TCP ou UDP mais "RAW", indique le protocole qui va être utilisé au dessus de la couche réseau.
- Pour la suite on va se mettre dans le cas le plus courant : une socket TCP.

- Une fois la socket ouverte, côté serveur, il faut lui dire, dans l'ordre :
  - de s'attacher à une adresse, càd un nom d'hôte et un numéro de port ;
  - de se mettre en mode passif pour écouter ;
  - d'attendre une connexion d'un client.
- Cela correspond aux appels à `bind`, `listen`, et `accept`.

- Du côté du client, il suffit de demander à se connecter à une adresse (toujours un nom d'hôte et un numéro de port).
- Cela correspond à l'appel `connect`.

Une fois connecté

- Une fois connectés, un client et un serveur peuvent échanger dans les deux sens.
- Il suffit de lire ou écrire dans la socket, comme si il s'agissait d'un fichier local.

- En Python, la bibliothèque standard du langage contient un module `socket` :
  - `import socket`
- Pour créer une socket TCP :
  - `sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0)`
- Du côté serveur :
  - ```
    addr = "localhost", 2345
    sock.bind(addr)
    sock.listen(8)
    client, _ = sock.accept()
    ```
- Du côté client :
  - ```
    addr = "localhost", 2345
    sock.connect(addr)
    ```
- Ensuite on peut utiliser les méthodes `send` et `recv`.

- Regardons ensemble des exemples simples d'un client et d'un serveur.