Programmation Socket en Python

Application console clients / serveur

Rappel sur les notions fondamentales

Problème : communication entre ordinateurs

Le problème se décompose en deux sous-problèmes :

- Adressage : Identification d’un ordinateur et d’un programme distant
- Transport : Envoi et réception des données

Problème : communication entre ordinateurs

Le problème se décompose en deux sous-problèmes :

- **Adressage : Identification d'un ordinateur et d'un programme distant**
- Transport : Envoi et réception des données

Les machines ont des adresses IP et des noms d'hôtes (**hostnames**) et les programmes et services ont des **numéros de port**.

Problème : communication entre ordinateurs

Le problème se décompose en deux sous-problèmes :

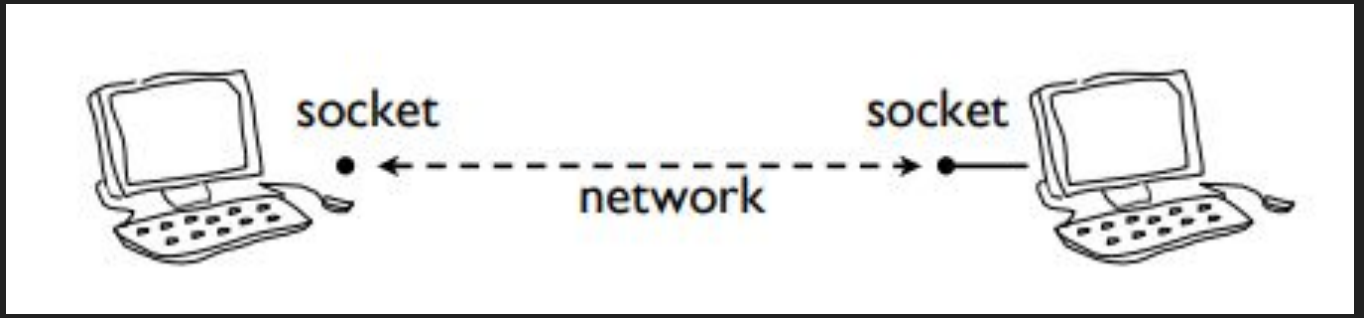
- Adressage : Identification d’un ordinateur et/ou d’un programme distant
- **Transport : Envoi et réception des données**

Transport des données

Il existe deux protocoles de communications :

- TCP (Transmission Control Protocol) : protocole orienté connexion
  - Orienté connexion
  - Fiable
  - Plus lent.
- UDP (User Datagram Protocol) : protocole sans connexion
  - Sans connexion
  - Moins fiable
  - Plus rapide

Les Sockets



- Une Socket désigne l’extrémité d’un canal de communication bidirectionnel.
- En connectant deux sockets ensemble, on peut faire passer des données entre processus.
- Une socket est donc une interface entre les applications des utilisateurs et la couche transport.

Position des sockets dans le modèle OSI

Modèle des sockets	Modèle OSI
Application utilisant les sockets	Application Présentation Session
UDP/TCP	Transport
IP/ARP	Réseau
Ethernet, X25, ...	Liaison Physique

Python Socket API : l’essentiel (coté serveur)

- création : **`s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)`**
  - la famille d’adresses : **socket.AF_INET** = adresses Internet de type IPv4 ;
  - le type du socket : **socket.SOCK_STREAM** = protocole TCP.
- liaison au port et à l'hôte : **`s.bind((host, port))`**
- mettre le socket à l’écoute : **`s.listen()`**
- accepter une connexion venant du client : **`connexion_client, adresse_client = s.accept()`**

Python Socket API : l’essentiel (coté client)

- création
- connexion au serveur : `s.connect((host, port))`

Envoi et réception des données

- `s.recv()`
- `s.send()`

Présentation de l’application

Présentation de l’application : chatroom multi clients

- Application de chat client/ serveur
- Utilisation de l’API socket de python
- Communication TCP
- Utilisation de threads pour la gestion indépendante des clients
- Deux programmes communiquant :
  - server.py
  - client.py

Codes python

Simulation