R3.05 – Programmation Systèmes

Sockets - Une micro introduction

C. Raïevsky

IUT A Valence
Université Grenoble Alpes
Département Informatique

2023

Socket réseau

Extrémité d'un canal de communication entre processus

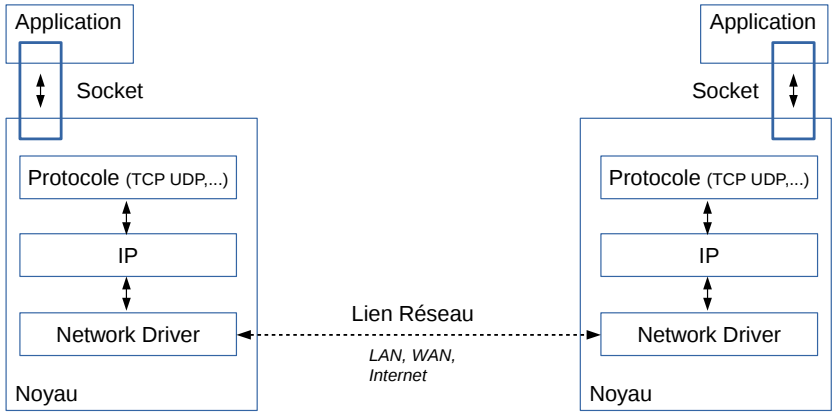
À quoi ça sert ?

- **Internet**
- Communication Inter Processus – *Inter Process Communication (IPC)*
- Architectures client-serveur

Qu'est-ce que c'est ?

L'objet manipulé par les programmes utilisateur pour communiquer via le réseau

Vue générale



Limite conceptuelle entre les processus utilisateurs et les couches réseaux

Caractéristiques

Identification

Sur le réseau, un socket est identifié par :

- Une adresse locale composée de :
  - Une adresse IP
  - Un numéro de port
- Un protocole de transport (TCP, UDP, etc.).

Pour le noyau et les applications :

Un identifiant unique de type entier – *socket descriptor*

Caractéristiques

Types

Deux principaux types de sockets

- Connecté (*stream*) – TCP, SCTP
- Non connecté (*datagram*) – UDP

Autres types :

- *Raw sockets* dans certains équipements, pas de protocole de transport
- *Unix Domain Sockets* : communication inter processus locale

Architecture Client-Serveur

Un des modèle d'interaction inter-processus les plus répandu

Le serveur :

- "Écoute" sur un port – attend une (des) connexion(s)
- Répond aux requêtes des clients
  - Après établissement d'une connexion (mode connecté, TCP)
  - Une par une (mode déconnecté, UDP)

Le client :

- Se connecte au serveur (en mode connecté)
- Envoie ses requêtes
- Traite les réponses du serveur

Doit connaître l'adresse et le port sur lequel le serveur écoute

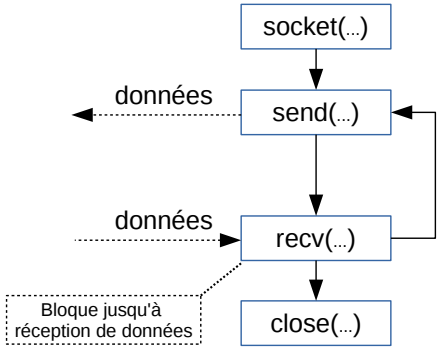
Algorithmes Génériques

- Client UDP
- Serveur UDP
- Client TCP
- Serveur TCP

Client UDP

Côté **client**

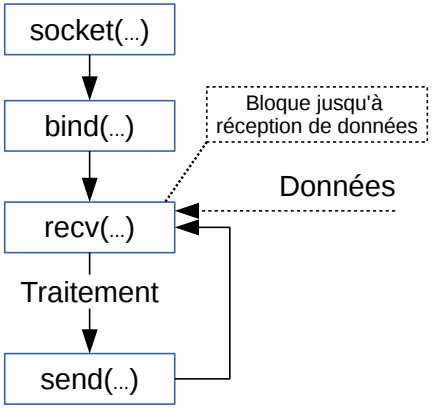
- `socket(...)` : création
- `send(...)` : envoie de données
- `recv(...)` : réception de données
- `close(...)` : fermeture



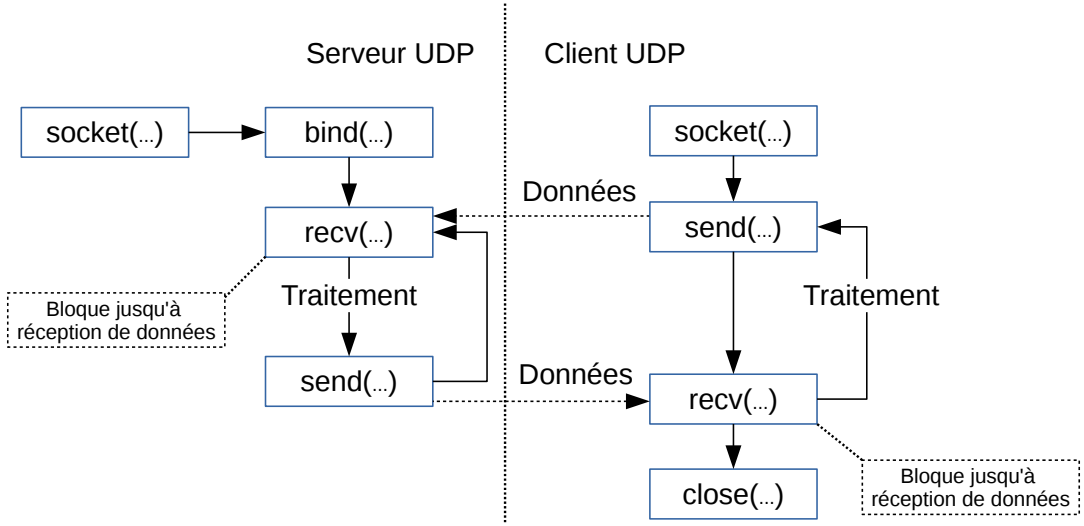
Serveur UDP

Côté **serveur**

- `socket(...)` : création
- `bind(...)` : association à une adresse
- `recv(...)` : réception de données
- `send(...)` : envoie de données



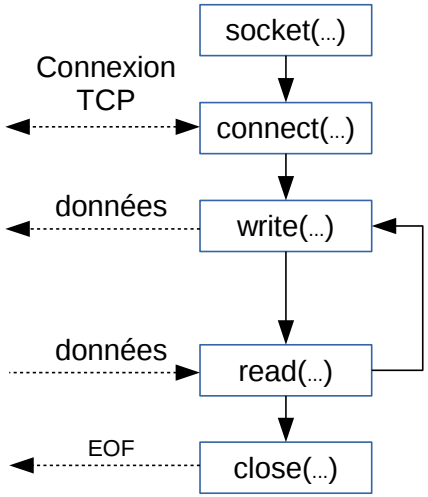
Client-Serveur UDP



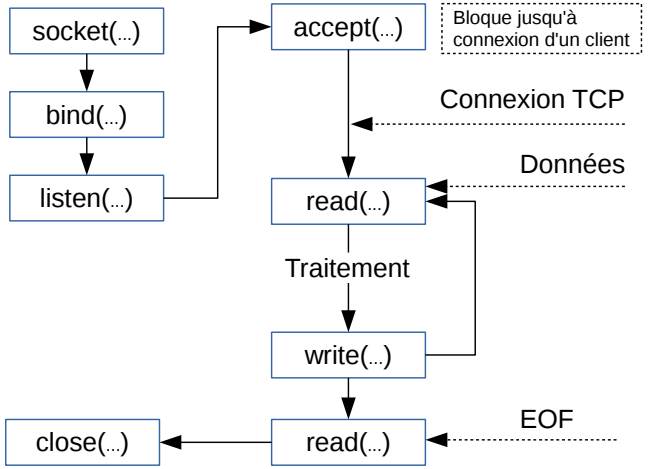
Client TCP

Côté **client**

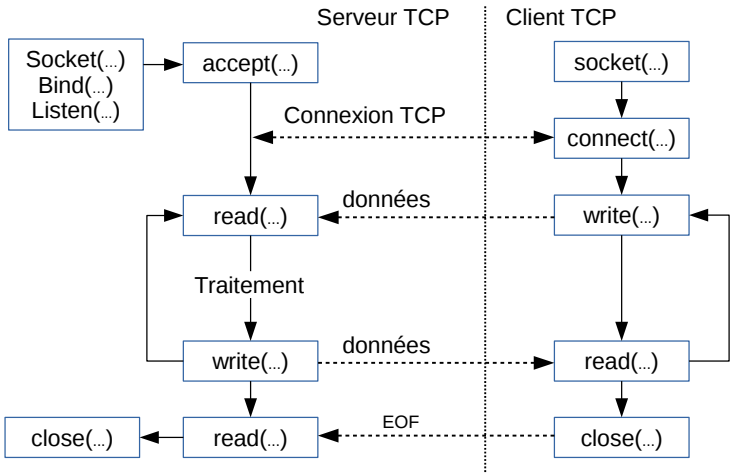
- `socket(...)` : création
- `connect(...)` : connection TCP
- `write(...)` : envoie de données
- `read(...)` : réception de données
- `close(...)` : fermeture de la connexion



Serveur TCP

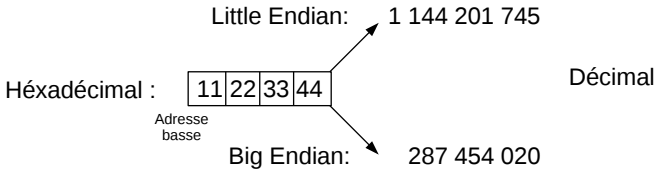


Client-Serveur TCP



Ordre des octets

Big-Endian – Little-Endian



- Internet utilise l'encodage "Big Endian"
- Suivant l'architecture (x86, arm, powerpc) le cpu (*host*) utilise l'un ou l'autre

Fcontions de conversion

- *host to network* → `htons, htonl` (16 et 32 bits)
- *network to host* → `ntohs, ntohl` (16 et 32 bits)

Références – Exemples

www.cs.dartmouth.edu/∼campbell/cs60/socketprogramming.html
https://en.wikipedia.org/wiki/Berkeley_sockets