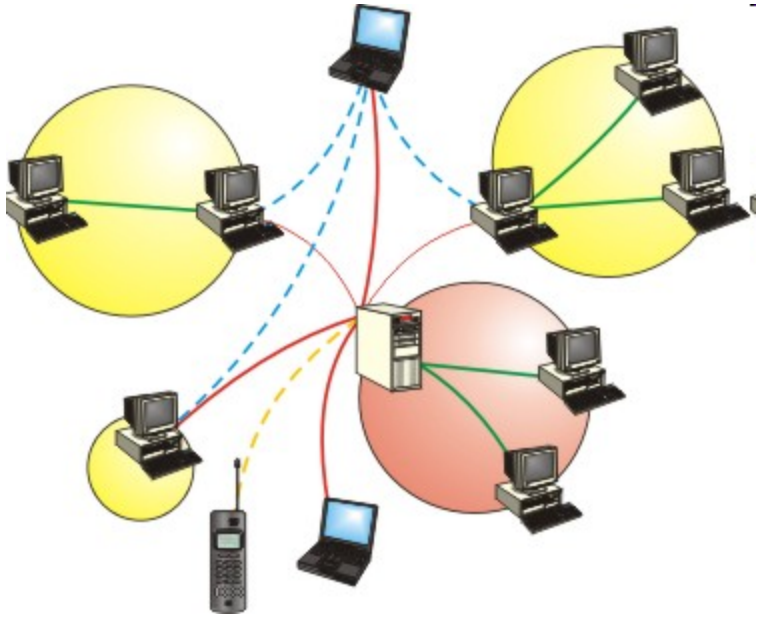
Programmation Socket

Introduction

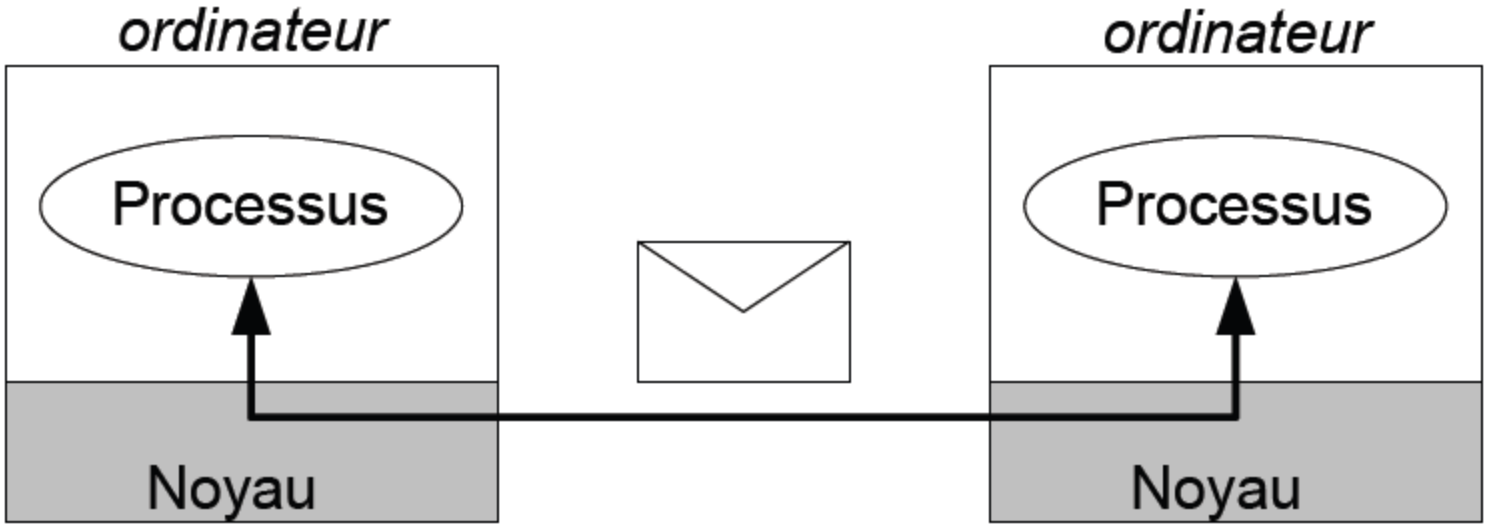
- Réseau



- Pourquoi réseaux?
  - besoin d'échanger des informations de manière simple et rapide entre machine

Introduction

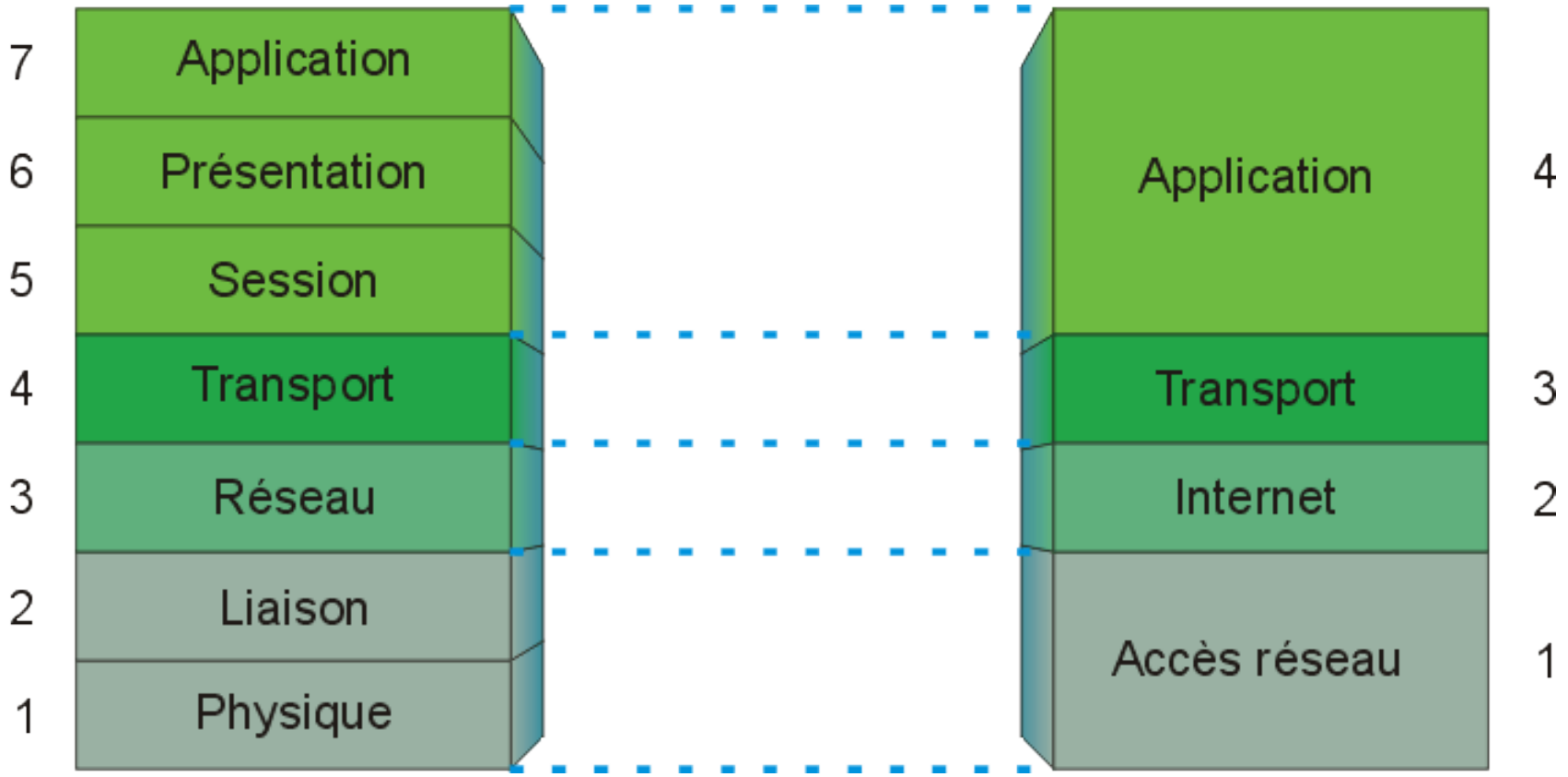
- Communication par envois des messages



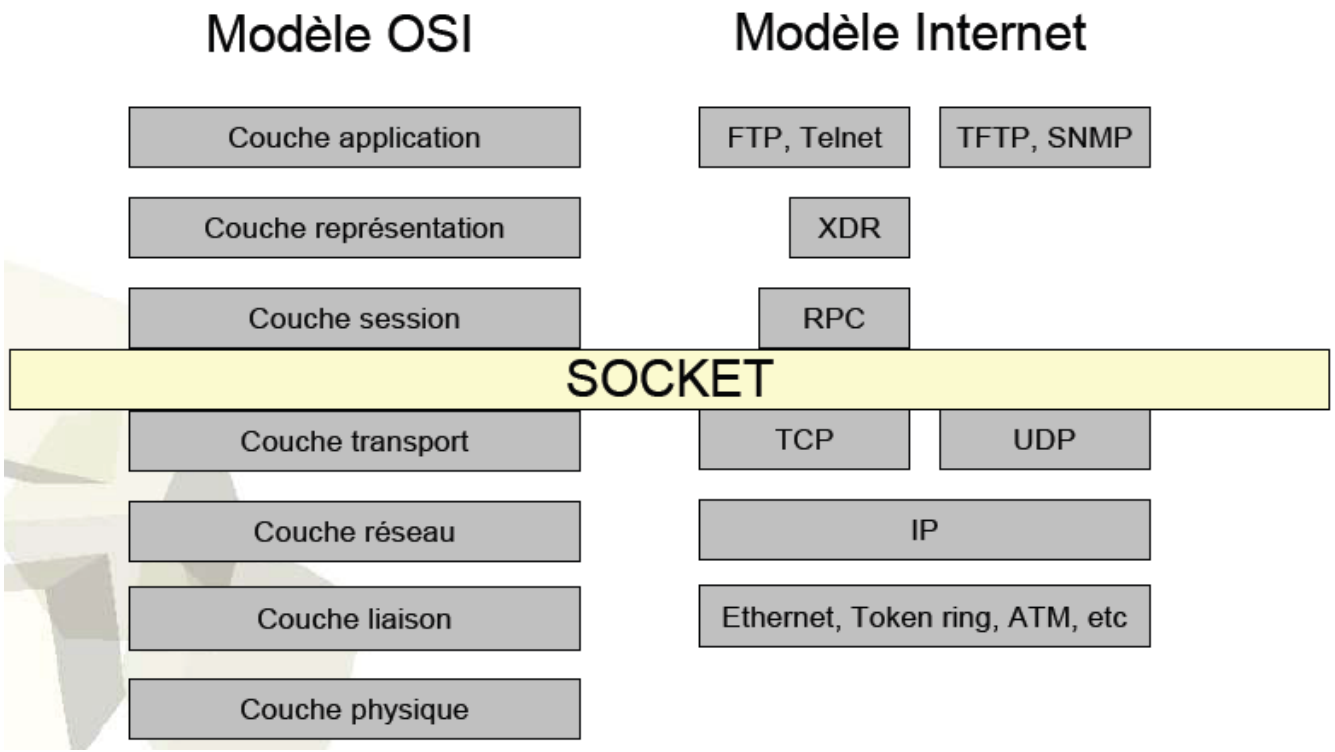
Introduction

- Une normalisation de réseaux
  - Pour pouvoir échanger les informations sans problèmes entre les entités
  - Modèle OSI : *Open System Interconnection*
  - Modèle Internet

Introduction



Introduction



Socket = Mécanisme d'interface de programmation

- permet aux programmes d'échanger des données

Introduction

- Service
  - Une machine serveur peut héberger plusieurs services:
    - Un serveur web (HTTP)
    - Un serveur de fichiers (FTP)
    - Un serveur de messagerie (SMTP et POP3)
    - …
  - Comment le serveur connaît-il le service souhaité par une machine cliente qui se connecte ?
    - ➢ Les ports

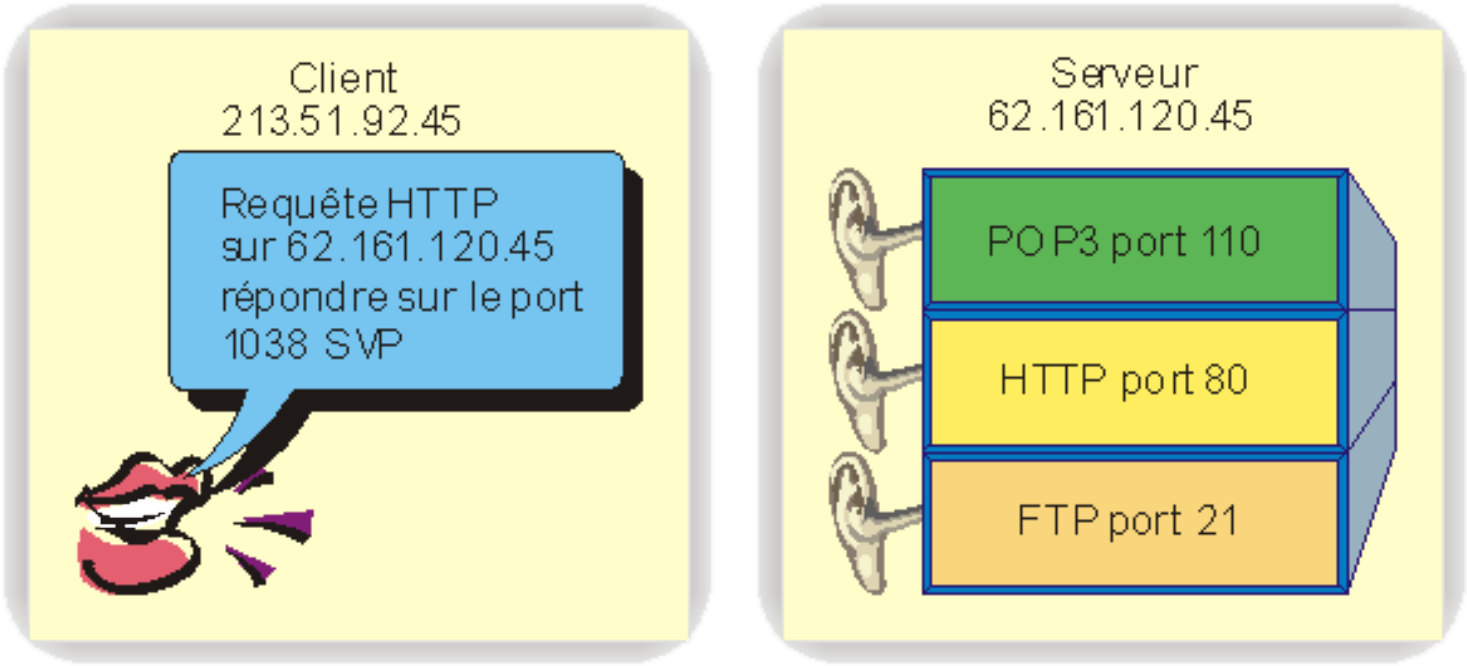
Introduction

- Socket
  - C’est la combinaison

    adresse IP:numéro de port
  - C’est une porte sur une machine à travers laquelle une application peut à la fois envoyer et recevoir des messages d’une autre application

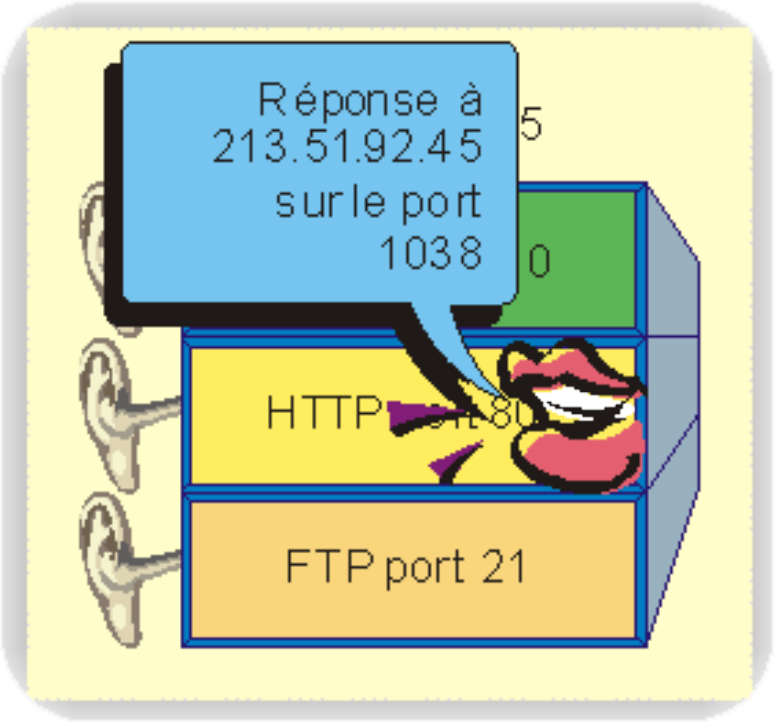
Introduction

Principe - Demande



Introduction

Principe - Réponse



Introduction

Les communications par socket se font toujours en client/serveur :

- Le programme serveur choisi un port *P* et y attend les clients
- Le programme client contacte le serveur en se connectant sur la machine du serveur et allant sur le port *P*

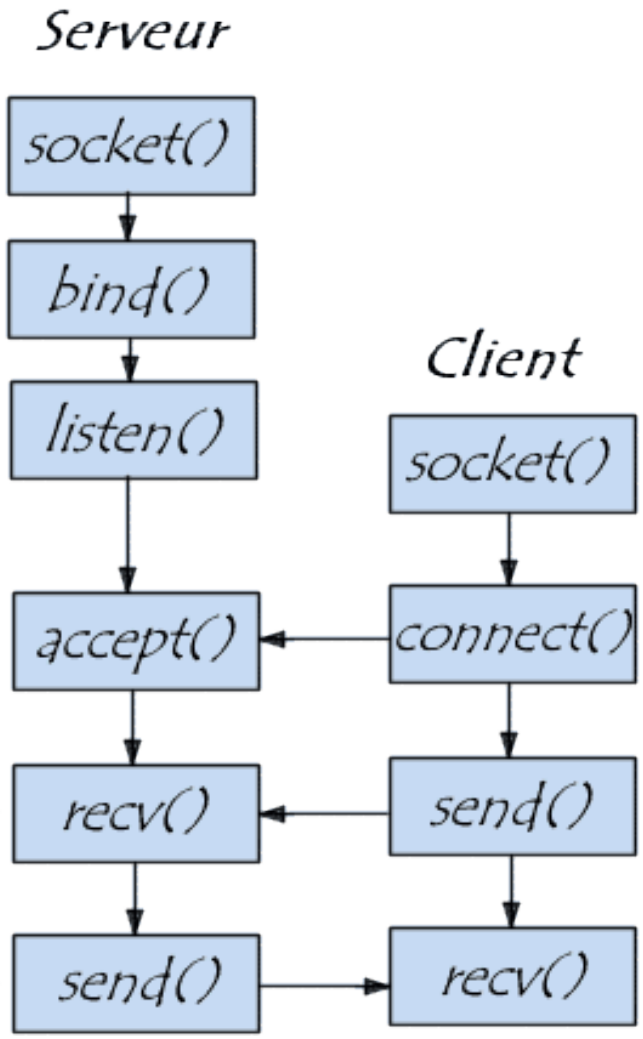
Programmation Socket

- Socket = Mécanisme d'interface de programmation
  - permet aux programmes d'échanger des données / messages

Utilisation de Socket

- Côté client (demandeur de la connection) ACTIF
  - créer une socket **socket()**
  - se connecter à un couple adresse, port **connect()**
  - lire et écrire dans la socket **read(), write(), send(), recv()**
  - fermer la socket **close()**
- Côté serveur (en attente de connection) PASSIF
  - créer une socket **socket()**
  - associer une adresse à la socket **bind()**
  - se mettre à l'écoute des connections **listen()**
  - accepter une connection **accept()**
  - lire et écrire sur la socket **read(), write(), send(), recv()**
  - ferme la socket **close()**

Etapes de réalisation



Echanges des données/messages
→ Ordre SEND/RECEIVE en fonction de protocoles/cahier de charges