Mohamed Sakkari

Unité d'Enseignement : Développement d'applications

Cours: Développement d'applications réparties

Plan du cours

- **Chapitre I**. Rappel sur les sockets
- **Chapitre II**. Architectures client / serveur
- **Chapitre III**. Intergiciels orientés objets (CORBA)
- **Chapitre IV.** Intergiciels orientés messages
- **Chapitre V**. Problèmes fondamentaux de la répartition

TP

- Tp1 :Développement d’une application client/serveur JAVA RMI

  (partie 1,2 et 3)

- Tp2 : Développement d’une application bancaire utilisant la technologie CORBA
- Tp3 : Découverte de JMS ; utilisation en mode publication/abonnement.

Chapitre I

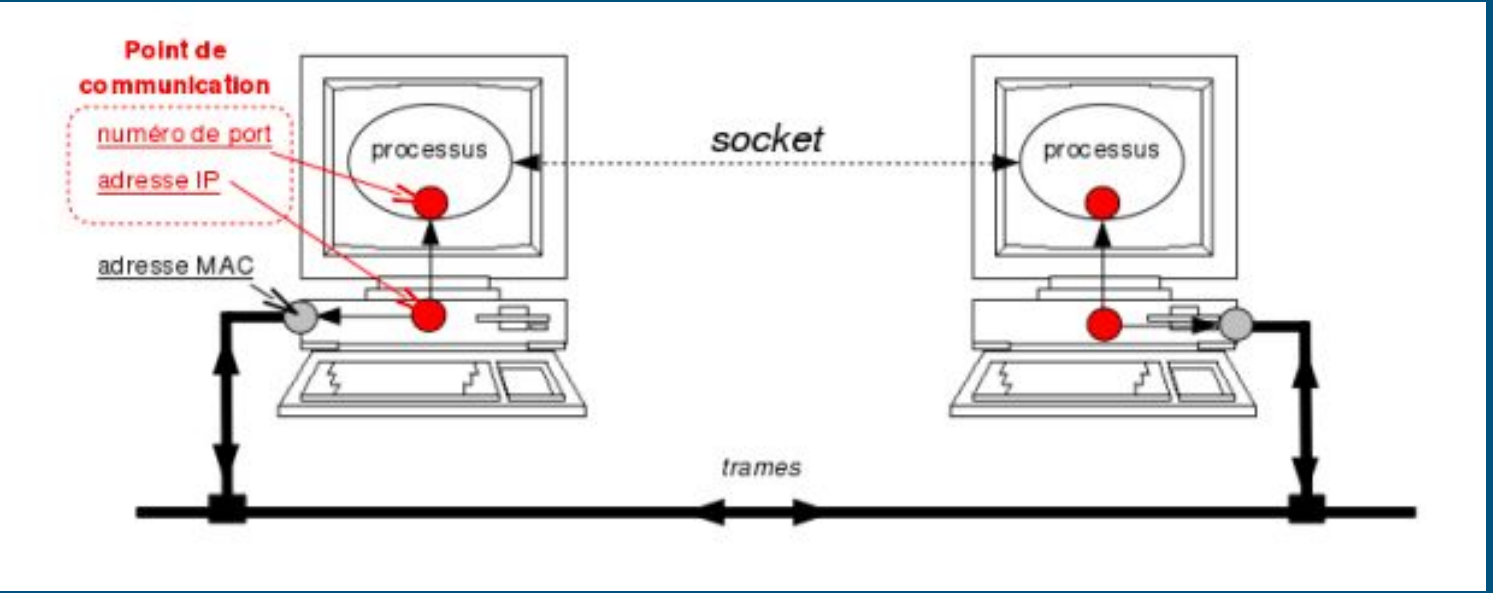
Rappel sur les sockets

I. Introduction aux sockets

Définition :

Socket : (interface) mécanisme de communication permettant d'utiliser l'interface de transport (TCP, UDP). Elle est créée par l'application et contrôlée par le système..

I. Introduction aux sockets



I. Introduction aux sockets

Les sockets sont

- une API (Application Program Interface), interface avec les couches réseau.
- un point de communication par lequel un processus peut émettre ou recevoir des données
- Une socket est communément représentée comme un point d'entrée initial au niveau TRANSPORT.

I. Introduction aux sockets

Couche Transport Rappel :

La couche Transport est responsable du transport des messages complets de bout en bout (soit de processus à processus) au travers du réseau. En programmation, si on utilise comme point d’entrée initial le niveau TRANSPORT, il faudra alors choisir un des deux protocoles de cette couche

I. Introduction aux sockets

Couche Transport Rappel :

- TCP (Transmission Control Protocol) est un protocole de transport fiable, **en mode connecté** .
- UDP (User Datagram Protocol) est un protocole souvent décrit comme étant non-fiable, **en mode non-connecté**, mais plus rapide que TCP.

I. Introduction aux sockets

Numéro de ports Rappel :

Un numéro de port sert à identifier un processus (l'application) en cours de communication par l'intermédiaire de son protocole de couche application (associé au service utilisé, exemple : 80 pour HTTP).

I. Introduction aux sockets

L’attribution des ports est faite par le système d’exploitation, sur demande d’une application. Ici, il faut distinguer les deux situations suivantes :

- cas d’un processus client : le numéro de port utilisé par le client sera envoyé au processus serveur.
- cas d’un processus serveur : le numéro de port utilisé par le serveur doit être connu du processus client.

I. Introduction aux sockets

Pour dialoguer, chaque processus devra préalablement créer une socket de communication en indiquant :

1. **Le domaine de communication** : ceci sélectionne la famille de protocole à employer. Il faut savoir que chaque famille possède son adressage. Par exemple, pour les protocoles Internet IPv4, on utilisera le domaine PF_INET ou AF_INET et AF_INET6 pour le protocole IPv6.

I. Introduction aux sockets

Pour dialoguer, chaque processus devra préalablement créer une socket de communication en indiquant :

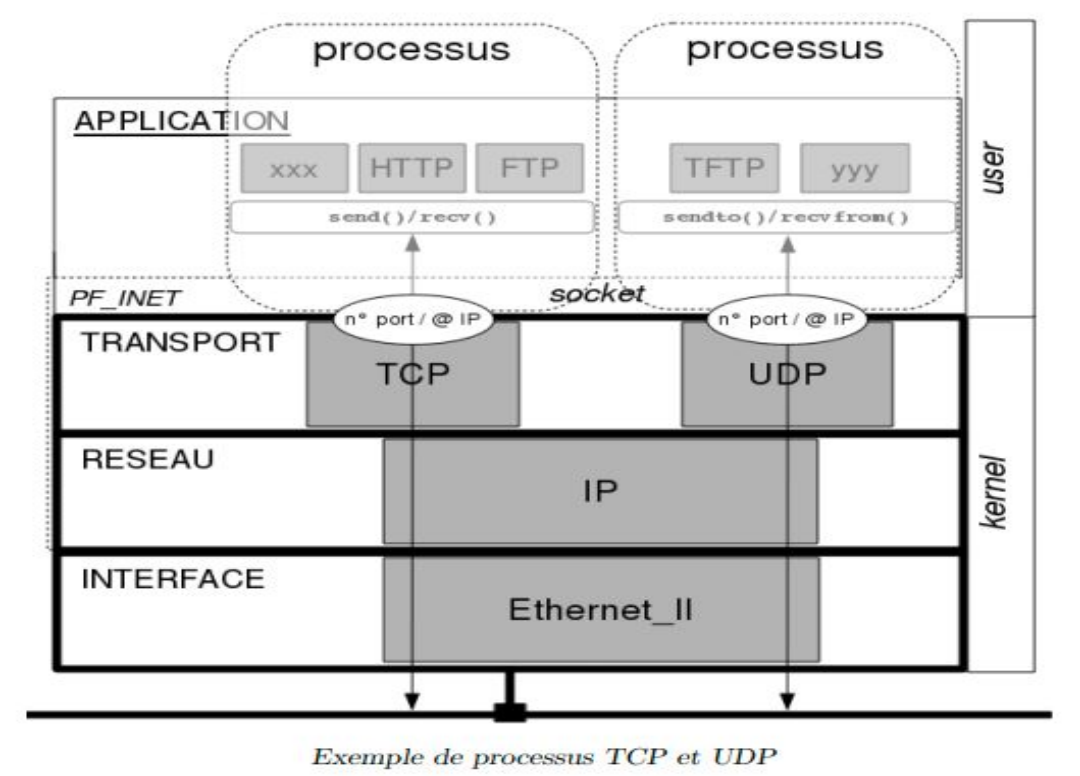
2. **Le type de socket** à utiliser pour le dialogue. Pour PF_INET, on aura le choix entre : SOCK_STREAM (qui correspond à un mode connecté donc TCP par défaut), SOCK_DGRAM (qui correspond à un mode non connecté donc UDP) ou SOCK_RAW (qui permet un accès direct aux protocoles de la couche Réseau comme IP, ICMP, ...).

I. Introduction aux sockets

Pour dialoguer, chaque processus devra préalablement créer une socket de communication en indiquant :

3. **Le protocole** à utiliser sur la socket. Le numéro de protocole dépend du domaine de communication et du type de la socket. Normalement, il n'y a qu'un seul protocole par type de socket pour une famille donnée (SOCK_STREAM →TCP et SOCK_DGRAM→ UDP).

I. Introduction aux sockets



Exemple de processus TCP et UDP

II. Socket en mode connecté (au dessus de TCP)

CLIENT

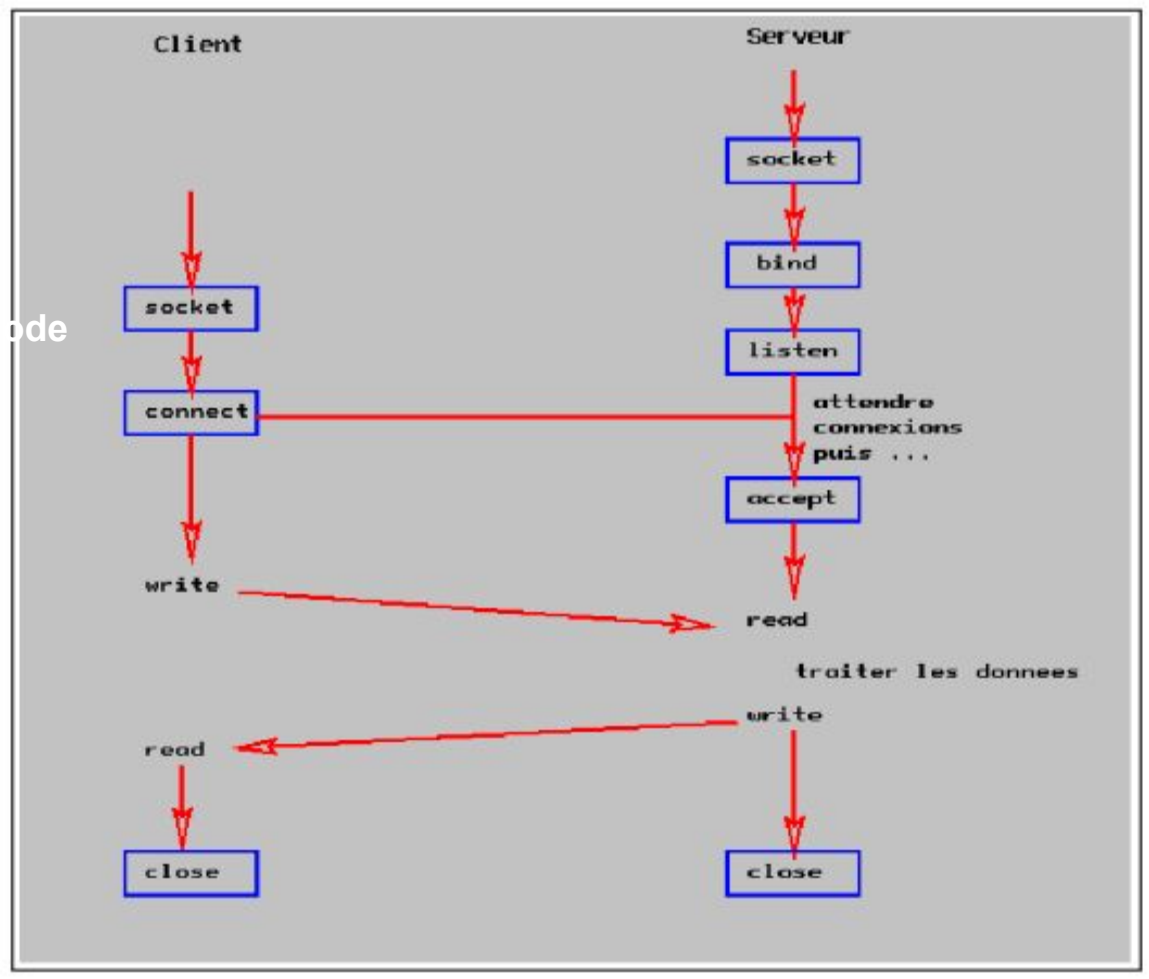
1. crée une socket
2. se connecter au serveur en donnant l'adresse socket distante (adresse Internet du serveur et numéro de port du service).Cette condition attribue automatiquement un numéro de port local au client.
3. Lit ou écrit sur la socket
4. ferme la socket.

II. Socket en mode connecté (au dessus de TCP)

SERVEUR

1. crée une socket
2. associe une adresse socket au service binding (permet de spécifier le type de communication associé au socket).
3. se met à l'écoute des connexions entrantes
4. Pour chaque connexion entrante :
   - ➔ accepte la connexion
   - ➔ lit ou écrit
   - ➔ ferme .

Le modèle client/serveur en mode TCP

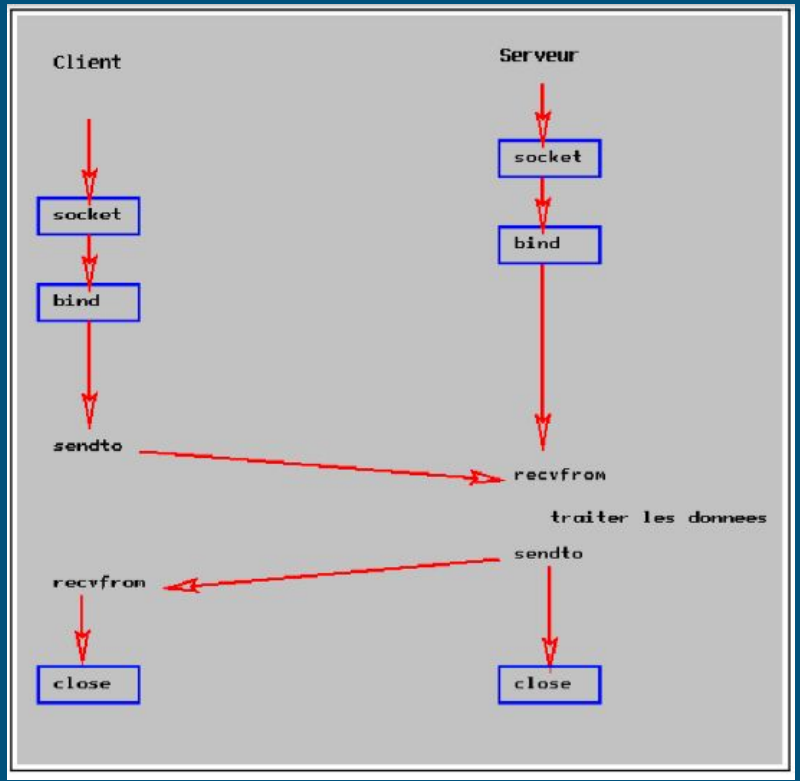


III. Socket en mode paquet (au dessus de UDP)

Les fonctions d'échanges de données sur une socket UDP sont:

- recvfrom()
- et sendto()

➔ qui permettent la réception et l'envoi d'octets sur un descripteur de socket en mode non-connecté.



III. Socket en mode paquet (au dessus de UDP)

- Une fois la socket UDP créée, le client pourrait déjà communiquer avec un serveur UDP car nous utilisons un mode non-connecté.
- Le code source d'un serveur UDP basique est très similaire à celui d'un client UDP. Évidemment, un serveur UDP a lui aussi besoin de créer une socket SOCK_DGRAM dans le domaine PF_INET, et il doit utiliser l'appel système bind() aussi.

Fin

Merci Pour Votre Attention