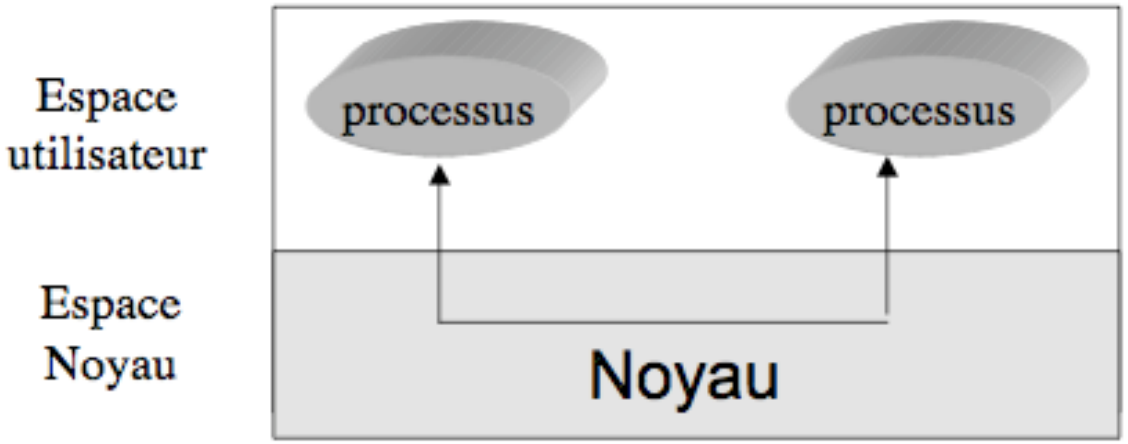
OS Réseaux et Programmation Système - C4

Rabie Ben Atitallah

Rabie.benatitallah@univ-valenciennes.fr

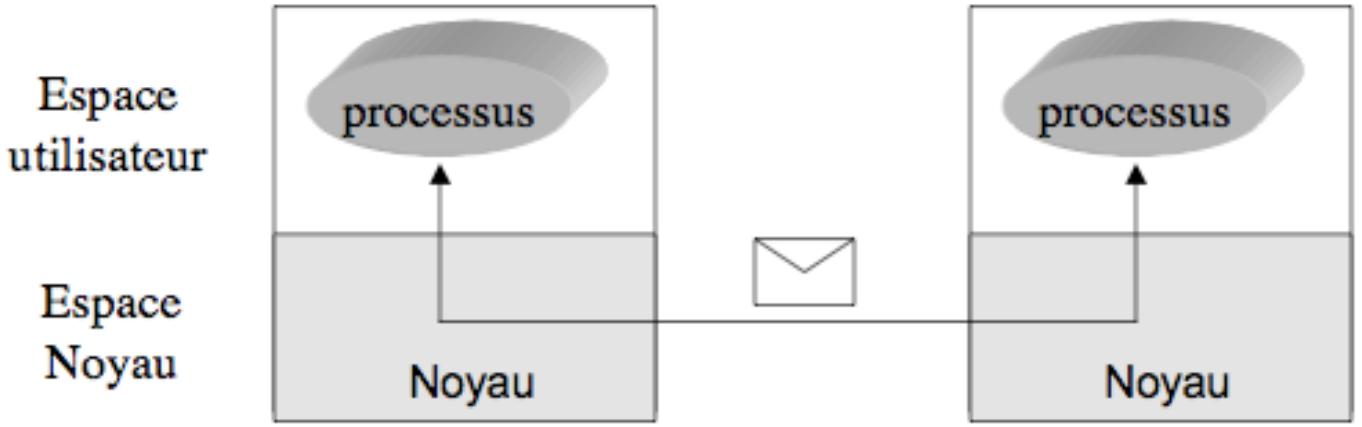
Sockets

Communications dans les systèmes centralisés



Les communications supposent
l 'existence d 'une mémoire partagée
Exemples : les pipes d 'Unix, les IPCs système V

Communications dans les systèmes répartis

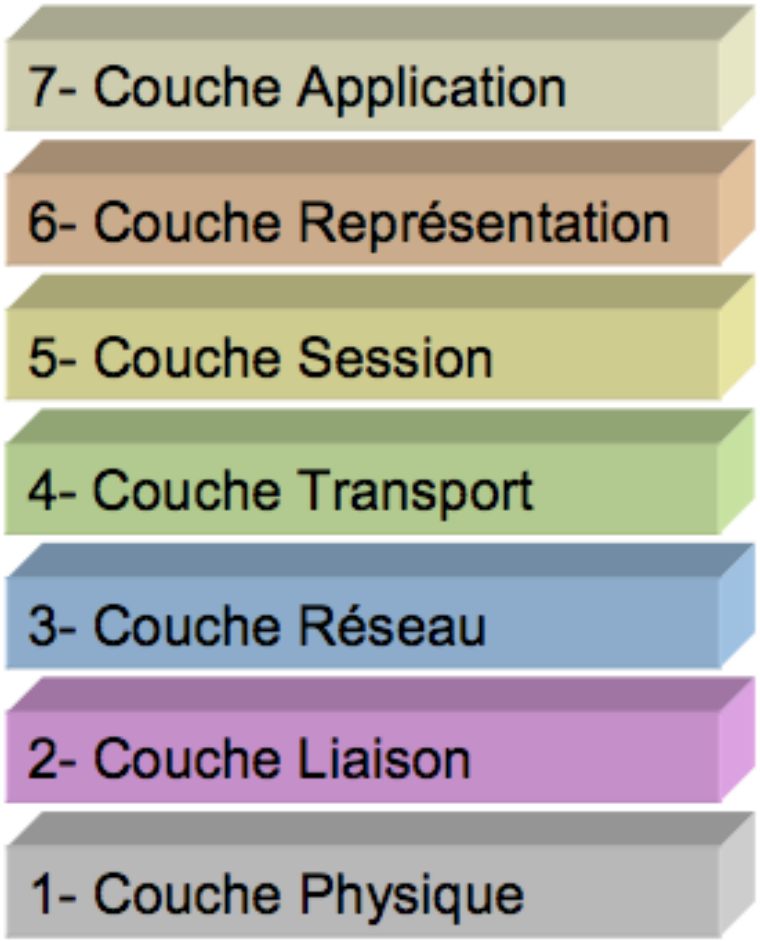


Les communications par envoi de message

Protocole de Communication

- La mise oeuvre des communications entre systèmes nécessite des protocoles de communication
- Protocole = un ensemble d'accords sur
  - la représentation des bits
  - détection de la fin d'un message
  - acheminement des messages
  - représentation des nombres, des caractères
  - Etc…
  - Il faut un accord à plusieurs niveaux, depuis les détails de bas niveau de la transmission des bits jusqu'à ceux de plus haut niveau de la représentation des données

Modèle OSI (Open System Interconnexion)



Un ensemble de services standard : terminal virtuel, transfert de fichiers, ...

Codage des informations sous une forme standard

Gestion complète d 'une session de communications entre 2 utilisateurs.

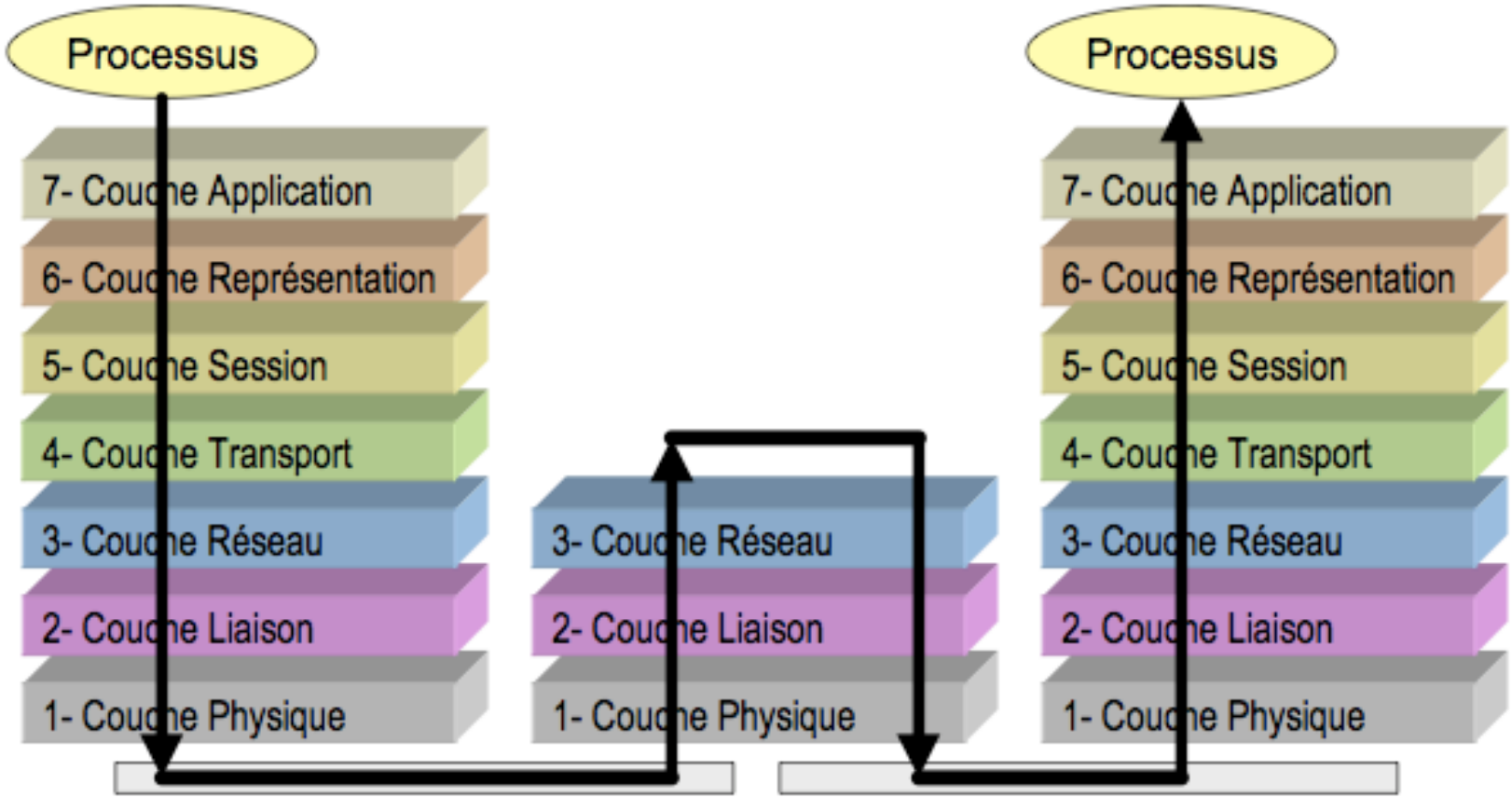
Transport de l 'information entre deux processus (Multiplexage)

Transport de l 'information entre deux points du réseau (Routage)

Transport, sans erreur entre deux points, des blocs de données (trames)

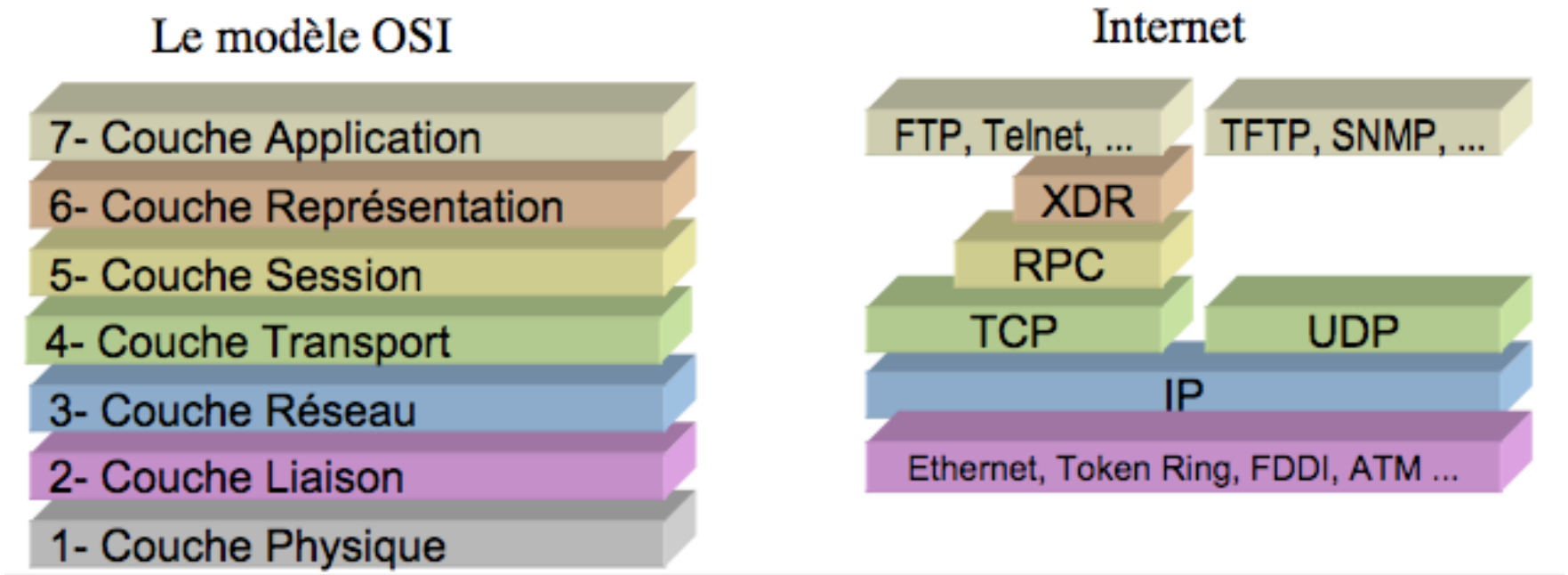
Transport de l 'information comme une suite de bits

Communication

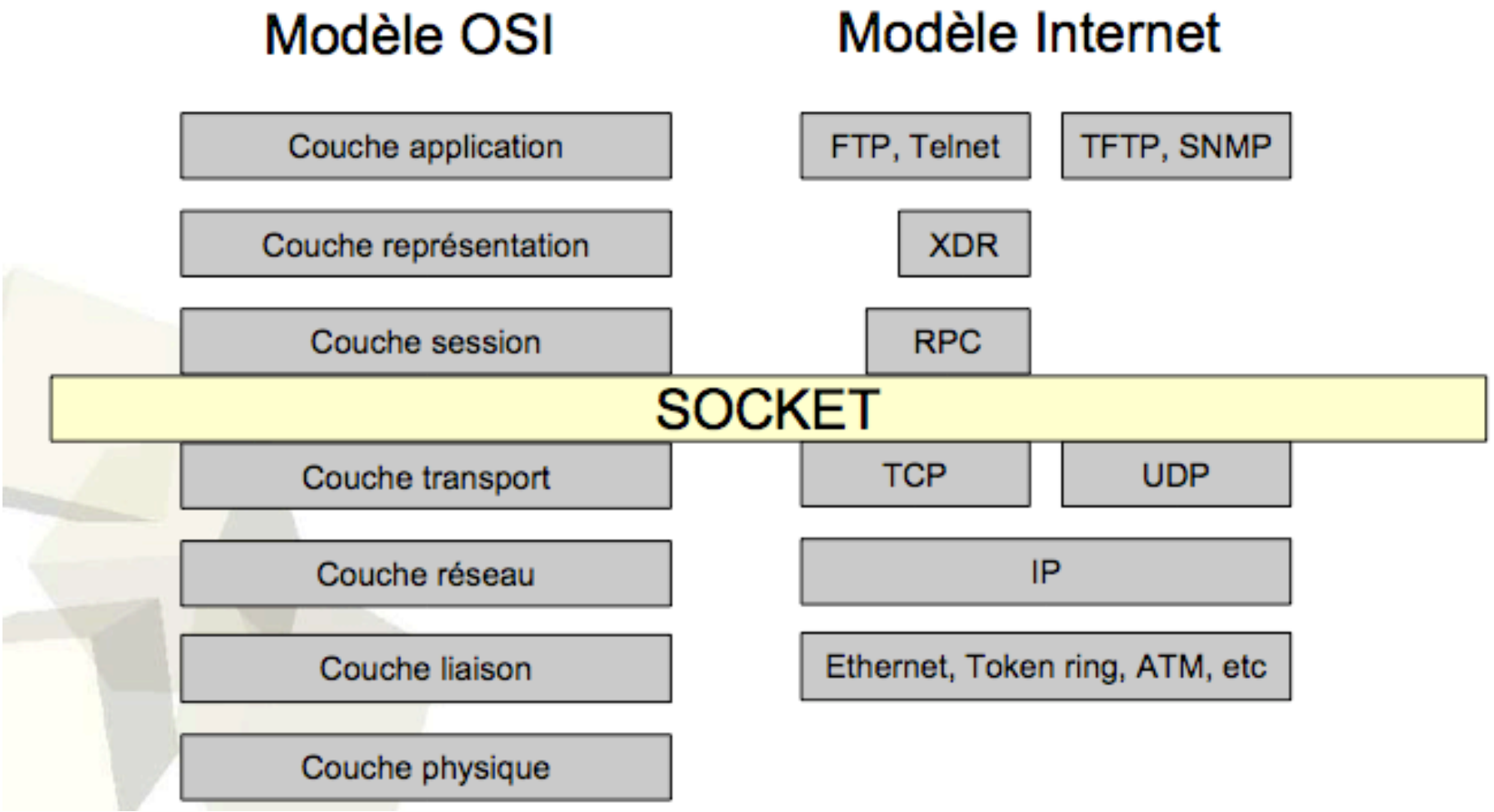


Routage entre plusieurs
supports de communication

IP Généralités



Les sockets


Modèle OSI
Modèle Internet
Couche application
Couche représentation
Couche session
Couche transport
Couche réseau
Couche liaison
Couche physique
FTP, Telnet
TFTP, SNMP
XDR
RPC
SOCKET
TCP
UDP
IP
Ethernet, Token ring, ATM, etc

Les sockets

- Les sockets = API (Application Program Interface)
  - interfaces entre les programmes d'applications et les couches réseaux
  - le terme Socket désigne aussi un canal de communication par lequel un processus peut envoyer ou recevoir des données
- L'API Socket s'approche de l'API Fichier d'Unix
  - Descripteur de socket dans la table des descripteurs du processus
  - Primitives read(), write(), close(), ioctl(), fnctl(), select()
  - Les descripteurs peuvent être partagés par les descendants de son créateur

Création d'un socket (1)

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int domaine, int type , int protocole);
```

- crée un socket et retourne son numéro de descripteur dans la table des descripteurs du processus (-1 si erreur)
- cependant le socket n'est pas lié (*bounded*) à un socket distant
  - ce qui est nécessaire pour le mode connecté
  - ce qui n'est pas nécessaire pour le mode non connecté

Création d'un socket (2)

- Domaine d'adresse:
  - Internet `AF_INET`, Unix `AF_UNIX`
- Type
  - au niveau bas du protocole (exemple : datagramme IP) `SOCK_RAW`
  - en mode non connecté `SOCK_DGRAM`, `SOCK_RDM`, `SOCK_SEQPACKET`
  - en mode connecté `SOCK_STREAM`
- Protocole :
  - par défaut à 0, le système choisit le protocole
  - `IPPROTO_UDP` pour UDP avec les sockets de type `SOCK_DGRAM`, `SOCK_RDM`, `SOCK_SEGPACKET`
  - `IPPROTO_TCP` pour TCP avec un socket de type `SOCK_STREAM`

Attachement d'un socket à une adresse

- Après sa création, un socket n'est connu que du processus qui l'a créé (et de ses descendants).
- Il doit être désigné par une adresse pour pouvoir être contacté de l'extérieur (autres processus locaux ou distants).
- Primitive bind()

```
int bind(int descsock, struct sockaddr *ptlocsockaddr,int locsockaddrlen)
```

  - Associe l'adresse locale ptlocsockaddr au socket descsock
  - locsockaddrlen est la taille de l'adresse *ptlocsockaddr

Adressage des sockets (1)

- Adresse de la domaine Unix `AF_UNIX` = une entrée dans le système de fichiers (local ou par NFS)

- Adresse du domaine Internet `AF_INET` = <adresse IP de la machine , numéro de port>
  - Sous Unix, les numéros de ports < 1024 sont réservés au SU
    - Exemple de numéros réservés
      - ftp : 21/tcp
      - telnet : 23/tcp
  - La liste des services se trouvent dans le fichier /etc/services
  - La liste des exécutables servant ces services via inetd et dans /etc/inetd

Adressage des sockets (2)

- Format générique des adresses de sockets

```
struct sockaddr {
        short sa_family;        /* domaine AF_UNIX, AF_INET */
        char sa_data[14];       /* adresse */
}
```

- Type associé aux adresses dans le domaine Unix

```
#include<sys.un.h>
struct sockaddr_un {
        short sun_family;       /* domaine AF_UNIX */
        char sun_path[128];     /* chemin  */
}
```

- Type associé aux adresses dans le domaine Internet

```
#include <netinet/in.h>
struct in_addr { u_long s_addr; };
struct sockaddr_in {
        short           sin_family;             /* domaine AF_INET */
        u_short         sin_port;               /*  port de la socket  */
        struct in_addr  sin_addr;    /* N° IP de la machine format réseau */
        char            sin_zero[8]; /* 8 charactères nuls de bourrage */
};
```

Fabrication d'adresse AF_INET

- Fabrication d'une adresse destinée à être attachée localement (par bind)

```
void localaddress( struct sockaddr_in *s_loc) {
    s_loc->sin_family = AF_INET;
    s_loc->sin_port = 0;   /* le port sera alloué dynamiquement */
    s_loc->sin_addr.s_addr = INADDR_ANY;/* adresse jocker */
}                                   inet_addr("127.0.0.1");
```

- Fabrication d'une adresse éloignée

```
void remoteaddress(struct sockaddr_in *s_rem, const char *rhost, int port) {
    struct hostent *h;
    s_rem->sin_family = AF_INET;
    s_rem->sin_port = port;
    if((h=gethostbyname(rhost))==0) {fprint(stderr,  "%s : machine inconnue ",rhost); }
    bcopy((char *)h->h_addr,(char *)s_rem->sin_addr, h->h_length);
}
```

Résolution DNS (1)

- Obtenir un adresse IP à partir d'un nom DNS

```
#include <sys/types.h><sys/socket.h><netinet/in.h><arpa/inet.h><netdb.h>
struct hostent *gethostbyname(char *hostname)
```

  - retourne un pointeur sur une structure du type suivant

```
struct hostent {
    char *h_name;        /* nom canonique de la machine */
    char **h_aliases;    /* tableau des autres noms d 'alias */
    int h_addrtype;      /* type d 'adresse (AF_INET pour Internet) */
    int h_length;        /* la longueur de l 'adresse (4 en IPv4) */
    char **h_addr_list;  /* tableau des adresses de type struct in_addr */
};
```

- *D'*autres fonctions de consultation

int gethostname(char *name, size_t lg) *retourne l 'adresse IP dans la zone name*
long gethostid() *retourne l 'adresse IP de la machine locale*

Résolution DNS (2)

- Fonction de conversion

#include <sys/types.h><sys/socket.h><netinet/in.h><arpa/inet.h><netdb.h>
char *inet_ntoa(const struct in_addr in) convertit la structure en adresse lisible
unsigned long inet_addr(const char *cp) donne la forme condensée d 'un adresse

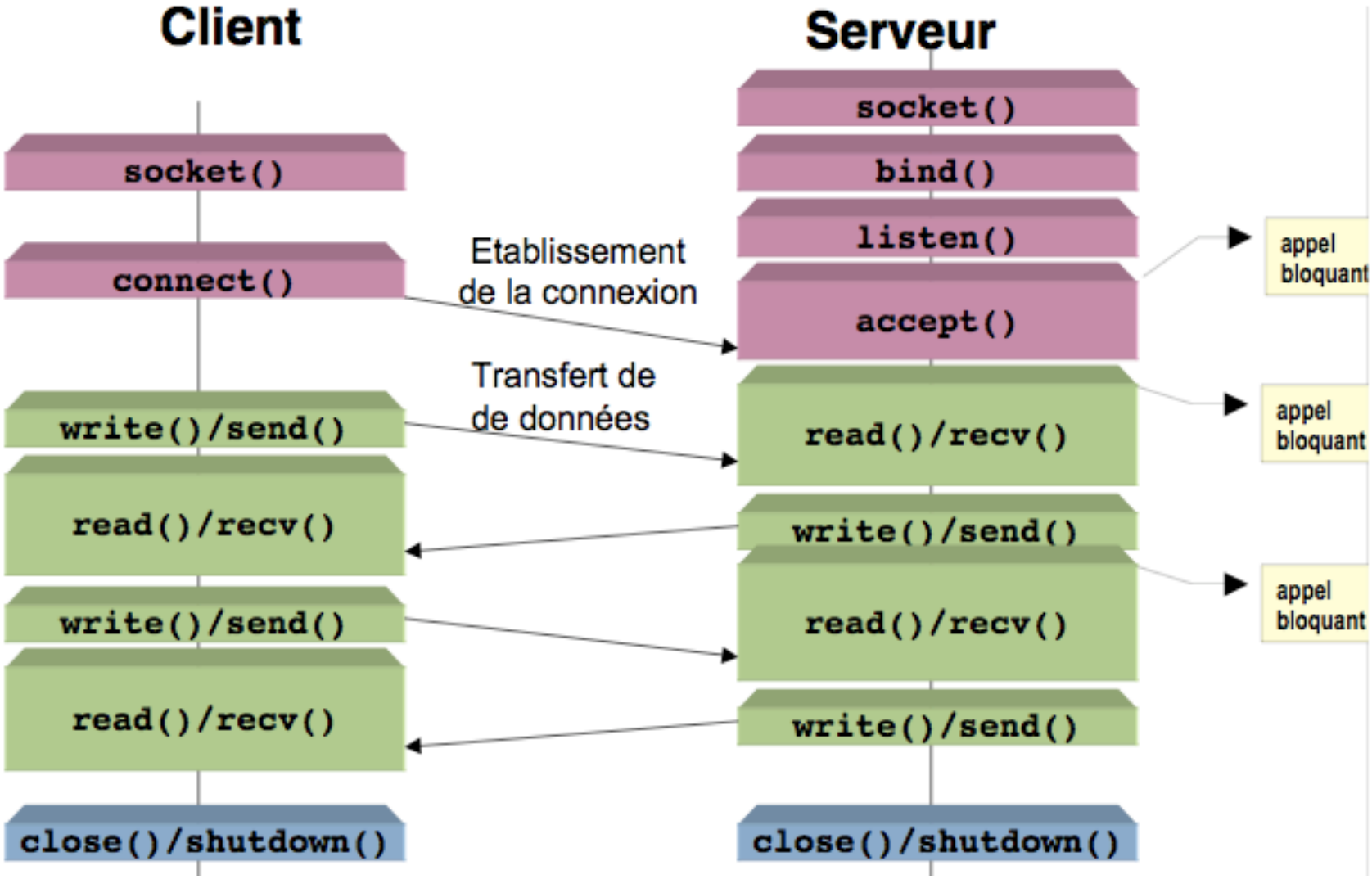
- Exemple

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
int main(int argc, char** argv[]) {
    struct hostent *res; struct in_addr * addr;
    res = gethostbyname(argv[1]);
    addr = (struct in_addr *) res->h_addr_list[0];
    printf("l 'adresse principale IP est %s\n", inet_ntoa(*addr));
    for(int i=1; addr=res->h_addr_list[i] ; i++) printf("l 'autre adresse IP est %s\n", inet_ntoa(*addr));
}
```

Socket en mode connecté (SOCK_STREAM)

- Coté client
  - Demandeur de la connexion
  - Socket actif
- Coté serveur
  - Attente de connexion
  - Socket passif

Socket en mode connecté (SOCK_STREAM)



Primitives du mode connecté (1)

- Coté Client
  - connect() : Demande d'une pseudo-connexion vers un socket distant

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int descsocket, struct sockaddr* ptsockaddr, int lensockaddr);
```

    - *obligatoire en mode connecté*
    - *primitive bloquante*
    - la primitive enregistre l'adresse distante dans le descripteur de socket
    - Remarque:
      - tout appel à connect() annule l'appel précédent
      - si ptsockaddr==NULL, le socket n'est plus associée à un autre

Primitives du mode connecté (2)

- Coté Serveur
  - listen() : Ouverture d'un service

    int listen(int *descsocket*, int *dc*);
    - permet de déclarer un service ouvert auprès du système local
    - indique le nombre maximum de demandes de connexion mises en attente
  - accept() : Acceptation de connexion

    int accept(int s, struct sockaddr *ptremotesaddr,int * ptlenremotesaddr)
    - primitive normalement bloquante (asynchronisme avec select())
    - permet de prendre connaissance d'une nouvelle connexion
    - retourne un descripteur d'un socket de service dédié à cette connexion
    - ptremotesaddr contient l'adresse du client

Emission - Réception - Fermeture

- Emission

  ssize_t write (int *descsock*, void* *buffer*, size_t *length*)

- Réception

  ssize_t read (int *descsock*, void* *buffer*, size_t *length*)

- Fermeture

  int close(int descsocket);

Exemple, Client

```
#include <sys/types.h>
#include <sys/socket.h>
#include<string.h>
main(int argc, char *argv[]) {
int sd; int len;
struct sockaddr_un serveraddr;

sd=socket(AF_INET, SOCK_STREAM,0);

serveraddr.sun_family=AF_UNIX;
strcpy(argv[1], serveraddr.sun_path, sizeof(argv[1]));
if(connect(sd, (sockaddr *)&serveraddr, sizeof(serveraddr)) <0) exit();

len=strlen(argv[2]); write(sd, &len, sizeof(len));
write(sd, argv[2], len);
close(sd);
}
```

Exemple, Serveur

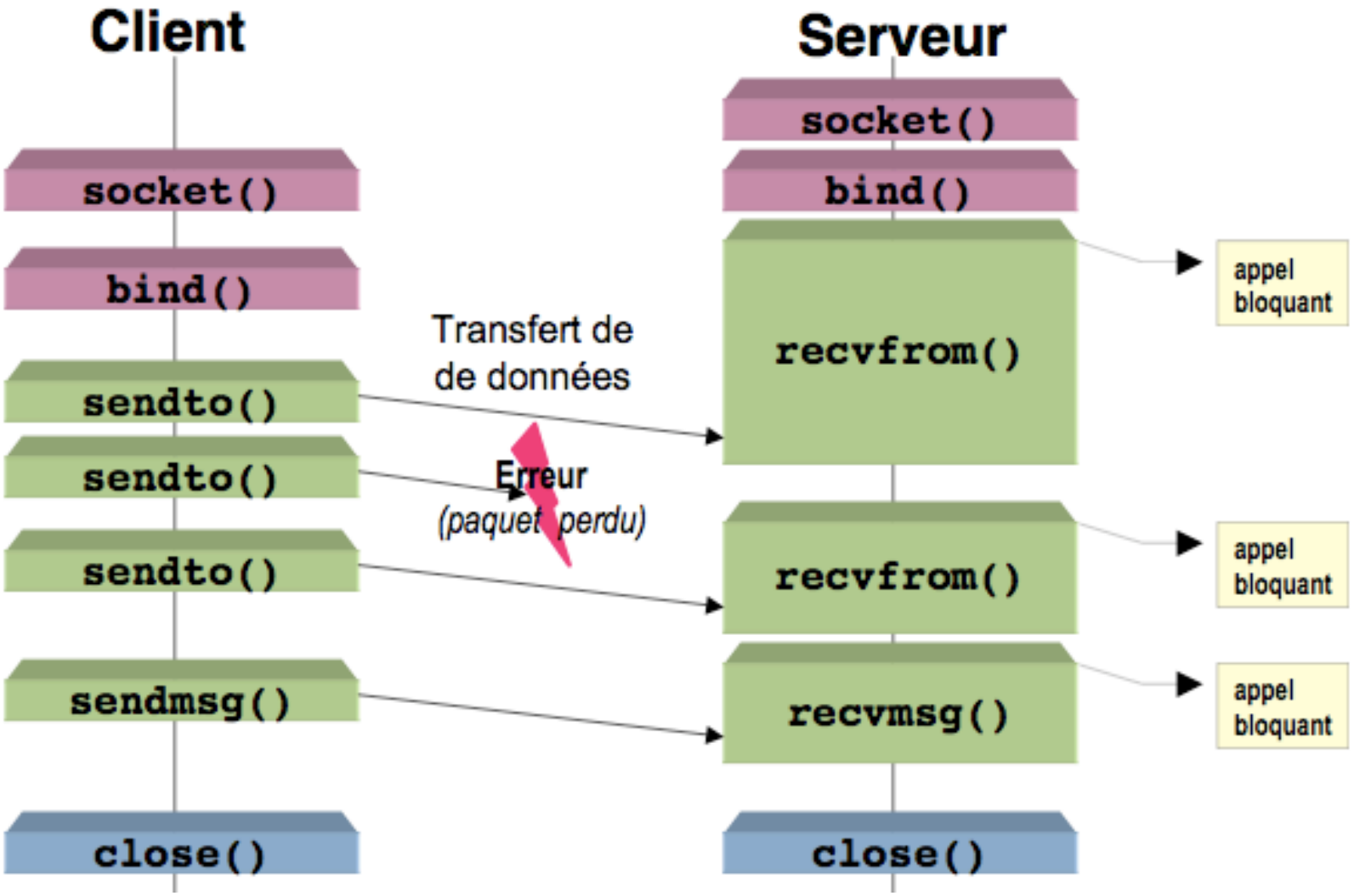
```
#include <sys/types.h>
#include <sys/socket.h>
#include<string.h>
main(int argc, char *argv[]){
    int sd, ns, nb; char buf[256]; int len;
    struct sockaddr_un serveraddr, clientaddr; int clientaddrlen;
    sd=socket(AF_UNIX, SOCK_STREAM,0);

    servaddr.sun_family=AF_UNIX;
    strcpy(argv[1], serveraddr.sun_path, sizeof(argv[1]));

    bind(sd, (sockaddr *)&serveraddr, sizeof(servaddr));
    listen(sd,1);
    ns=accept(sd,(sockaddr *)&clientaddr, &clientaddrlen);

    nb=read(ns, &len, sizeof(len)); /* les erreurs ne sont pas traitées */
    nb=read(ns, buf, len); write(1,buf,nb);
    close(ns); close(sd);
}
```

En mode non connecté (SOCK_DGRAM)



Emission - Réception

- Emission

```
int sendto(int descsock, void* buffer, size_t length, int flag,
           struct sockaddr* ptsockaddr, int lensockaddr)

int sendmsg(int descsock, struct msghdr* msghdr, int flag)
```

- Réception

```
int recvfrom(int descsock, void* buffer, size_t length, int flag,
             struct sockaddr* ptsockaddr, int* ptlensockaddr)

int recvmsg(int descsock, struct msghdr* msghdr, int flag)
```

Exemple, émetteur

```
#include <sys/types.h>
#include <sys/socket.h>
#include<string.h>
main(int argc, char *argv[]) {
int sd;
struct hostent hp;
struct sockaddr_in recvaddr;
sd=socket(AF_UNIX, SOCK_DGRAM,0);

hp=gethostbyname(argv[1]);
recvaddr.sin_family=AF_INET; recvaddr.sin_port=htons(atoi(argv[2]));
memcpy(& recvaddr.sin_addr.s_addr, hp->h_addr, hp->hp_length);

sendto(sd, argv[3], strlen(argv[3]),&recvaddr, sizeof(recvaddr));
close(sd);
}
```