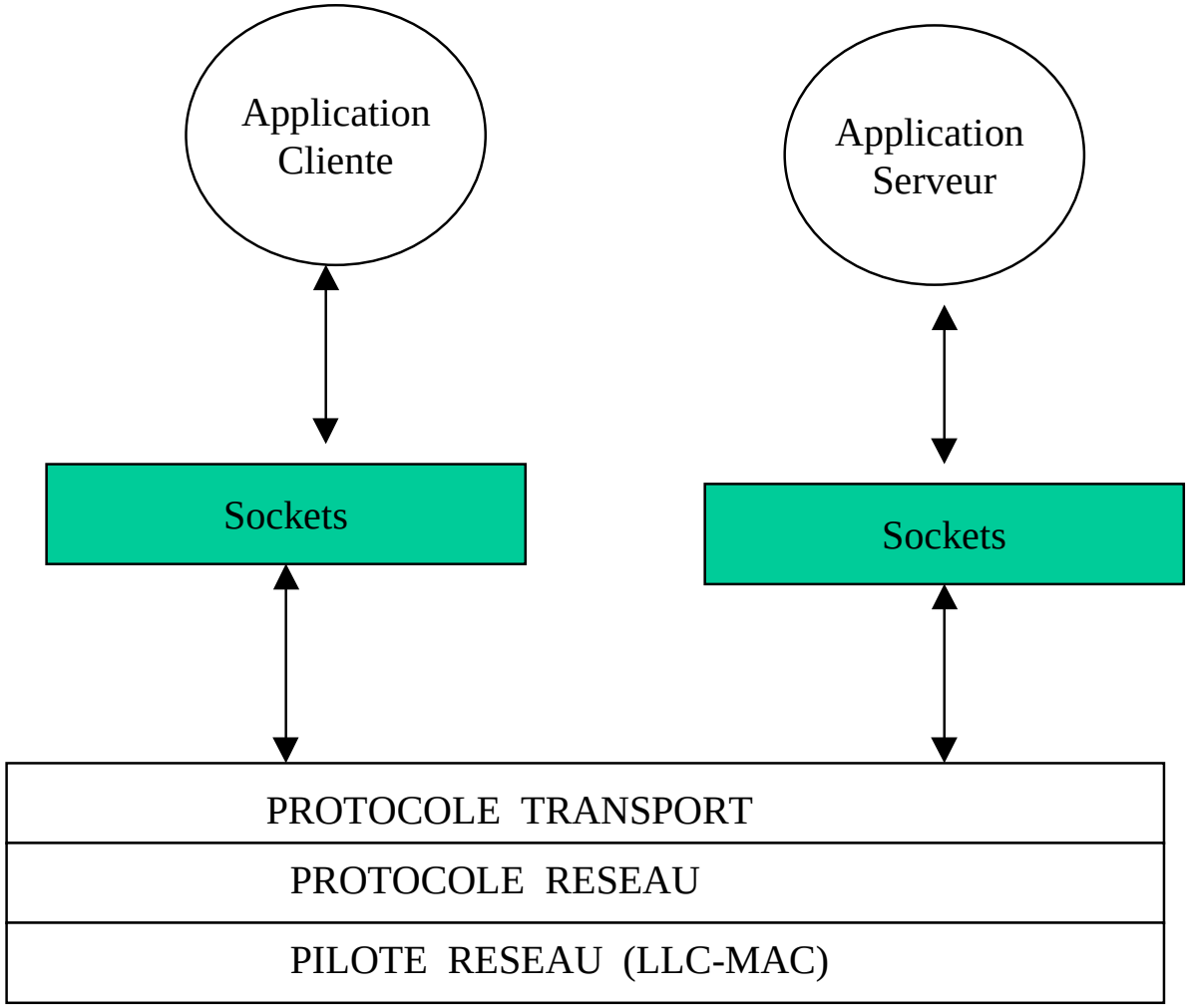
Les Sockets/1

La version Berkeley 4.2 d'Unix a été la première à inclure TCP/IP dans le noyau du système d'exploitation et à proposer une interface de programmation de ces protocoles : les *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

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Les Sockets/2



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Les Sockets/3

Pour établir une communication vers une machine distante, il faut :
1.s'attribuer (ou laisser le soin au système d'attribuer) un numéro de port

2. déterminer l'adresse Internet de la machine avec laquelle les échanges vont s'effectuer

3. connaître ou déterminer le numéro de port du service distant.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Les Sockets/4

Pour créer une socket, on doit préciser :

- le domaine de travail : Unix (AF_UNIX), TCP/IP (AF_INET), etc.
- le type de protocole de communication :
  - **SOCK_DGRAM** : envoi de messages sous forme de datagrammes (exemple UDP dans le domaine AF_INET).
  - **SOCK_STREAM** : envoi de flux d'octets (exemple TCP dans le domaine AF_INET).
  - **SOCK_RAW** : il permet (moyennant des droits système) l'accès à des protocoles de plus bas niveau, comme IP dans le domaine AF_INET.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Utilisation des Sockets en mode connecté/1

CLIENT

1. crée une socket
2. se connecte 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.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

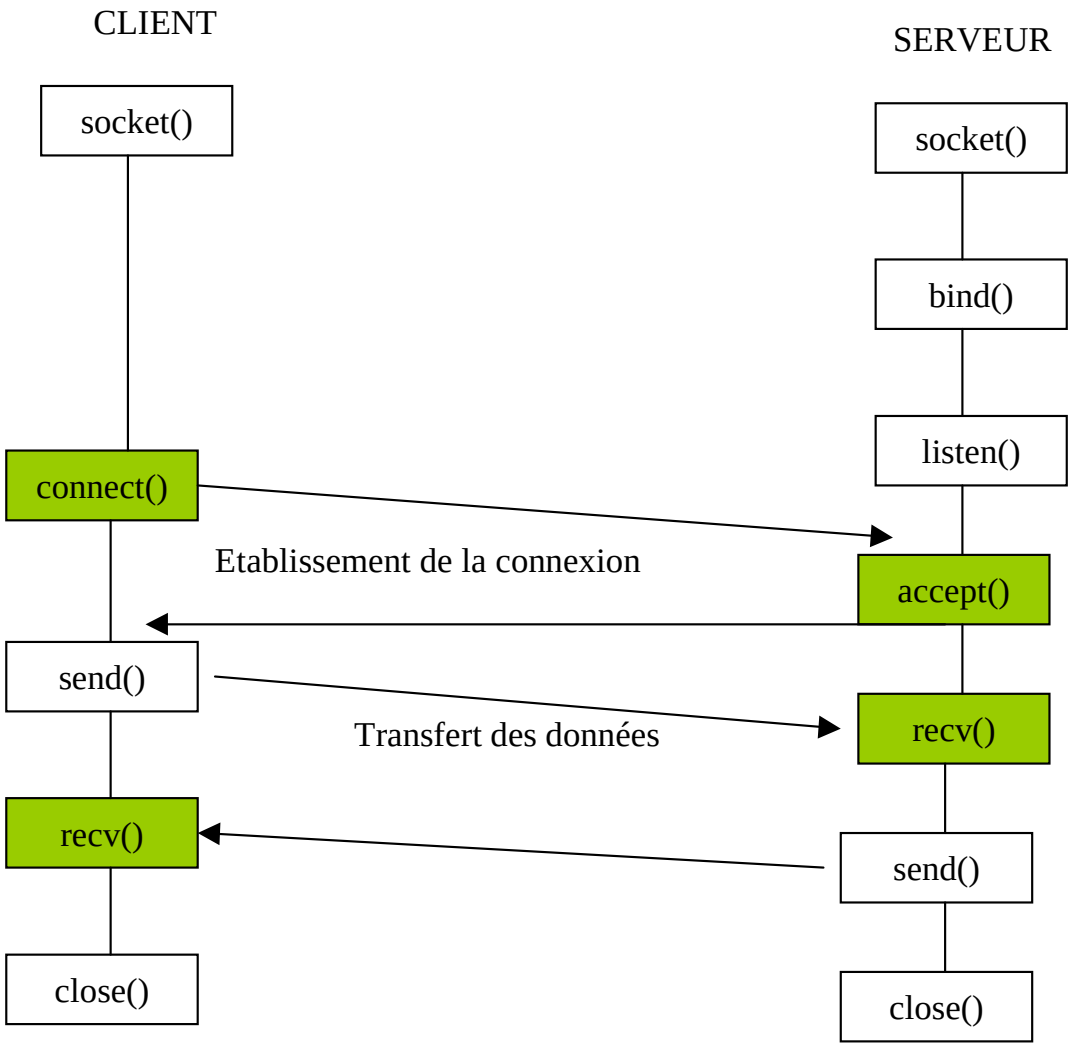
Utilisation des Sockets en mode connecté/2

SERVEUR

1. crée une socket
2. associe une adresse socket (son adresse Internet et le numéro de port choisi) au service : “ binding ”
3. se met à l’écoute des connexions entrantes
4. Pour chaque connexion entrante :
   a. accepte la connexion (une nouvelle socket est ainsi créée avec les mêmes caractéristiques que la socket d’origine)
   b. lit ou écrit sur la nouvelle socket
   c. ferme la nouvelle socket.

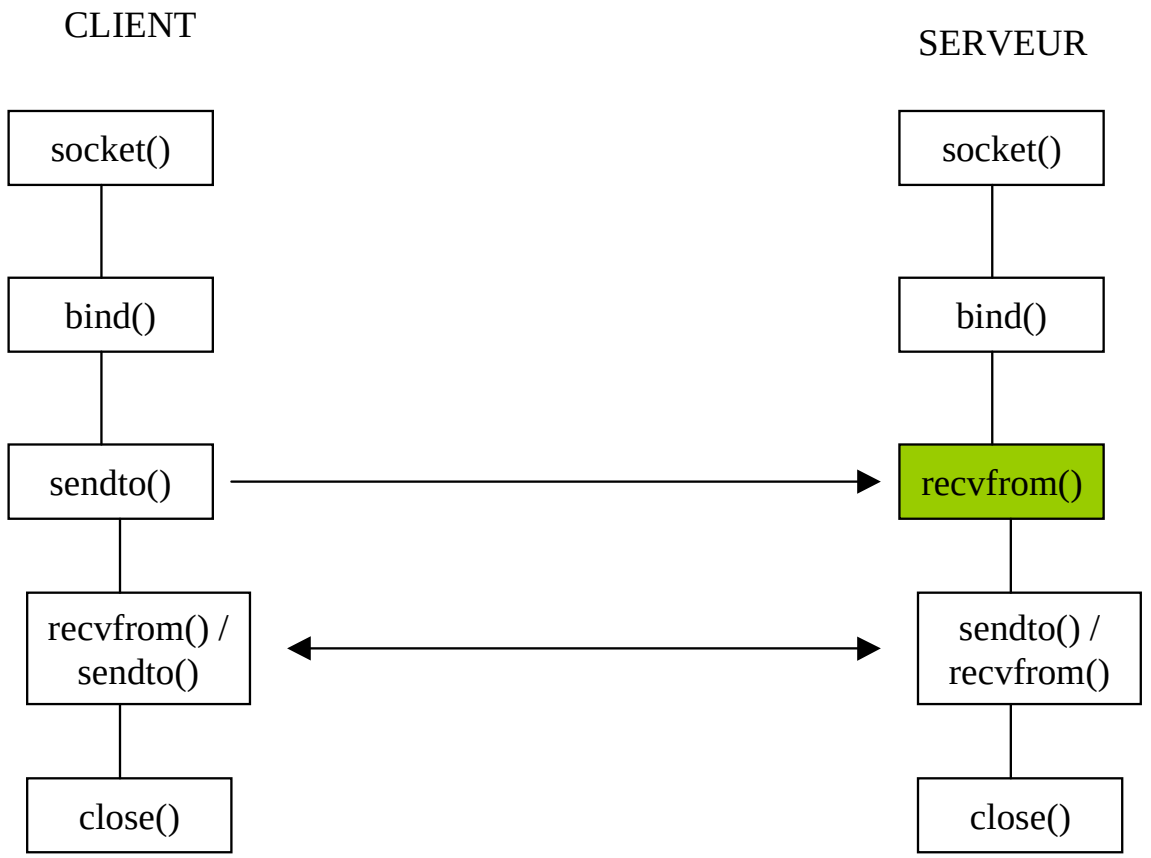
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Le modèle client/serveur en mode TCP



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Le modèle client/serveur en mode UDP



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Adressage dans le domaine AF_INET

Structures définies dans le fichier <netinet/in.h> :

```
struct   in_addr  {
u_long   s_addr;
}

struct   sockaddr_in  { /* in comme Internet  */
short  sin_family;/* AF_INET  */
u_short   sin_port;/* numéro du port en ordre réseau  */
struct  in_addr   sin_addr;/* adresse Internet de
            la machine en ordre réseau */
char   sin_zero[8];     /* inutilisé      */
}
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Fichier d’inclusion

AF_INET

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>
#define PORT  5001 /* option propre à l’application  */
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitive `socket` (TCP et UDP)

Création d'une socket

```
int socket (domaine, type, protocole);
      int domaine;       /* AF_INET ou AF_UNIX  */
      int  type;/* SOCK_STREAM (TCP) ou SOCK_DGRAM (UDP)*/
      int protocole;     /* IPPROTO_TCP ou IPPROTO_UDP
                         équivalent si = 0*/
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitive `bind` (TCP et UDP)

Binding (association) : associe une adresse locale (port) à un descripteur de socket est facultatif pour les clients TCP car il se fait à la connexion.

```
int bind  (socket, adresse, l_adresse);
      int   socket;      /* descripteur de la socket  */
      struct  sockaddr  *adresse; /*adresse socket
                         (port+adresse)  */
      int  l_adresse;    /* longueur de la structure
            adresse  */
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitive `connect` (TCP)

Connexion d'un client à un serveur

```
int connect  (socket, adresse, l_adresse);
      int   socket;     /* descripteur de la socket  */
      struct  sockaddr  *adresse; /*adresse socket
            (port+adresse)  */
      int  l_adresse;   /* longueur de la structure
      adresse  */
```

- Convertir le type `sockaddr_in` pour le domaine INET en `sockaddr`

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitives `listen` (TCP)

Mise d'un serveur à l'état d'écoute : indique que le serveur peut attendre au maximum `qlen` requêtes des clients.

```
int listen (socket,  qlen);
      int  socket;       /* descripteur socket */
      int  qlen;  /* nombre maximal de connexions
                     traitées */
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitive `accept` (TCP)

```
int accept(socket, struct sockaddr *pin, addrlen));
      int   socket;     /* descripteur retourné par
            la primitive socket  */
      struct  sockaddr  *adresse; /*initialisé à vide
            et mis à jour par accept avec le port
            et l'adresse IP de la machine où
            se trouve le client qui a fait un connect */
      int  l_adresse;/*longueur de la structure adresse*/
```

La primitive `accept`() retourne un nouveau descripteur de socket, qui sera utilisé pour l'échange de données avec le client.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitives d'envoi et de réception en mode TCP

```
ssize_t  recv(
      int  descripteur,
      void  *ptr,
      size_t  nb_caracteres,
      int  option  /* = 0, MSG_PEEK pour
         consulter sans extraire,
         MSG_OOB pour lire une donnée (1 car) urgente */
);

ssize_t  send(
      int  descripteur,
      void  *ptr,
      size_t  nb_caracteres,
      int  option /* =0 ou MSG_OOB  */
);
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitive de réception en mode UDP

```
int  recvfrom(
       int descripteur,  /* descripteur de la socket  */
       void  * message,  /* adresse de réception */
       int longueur,   /* taille zone réservée  */
       int option,   /* 0 ou MSG_PEEK */
       struct  sockaddr  *ptr_adresse,/* adresse émetteur */
       int  *ptr_longueur_adresse /* pointeur sur longueur
             zone adresse */
}
```

retourne le nombre de caractères effectivement reçus.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitive d'envoi en mode UDP

```
int  sendto(
      int descripteur,  /* descripteur de la socket  */
      void  * message,  /* adresse à envoyer */
      int longueur,     /* longueur du message  */
      int option,       /* 0*/
      struct  sockaddr  *ptr_adresse, /* adresse destinat */
      int  longueur_adresse /*longueur adresse destinat */
}
```

retourne le nombre de caractères effectivement envoyés.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitive `close` (TCP et UDP)

Fermeture de connexion

```
int  close (socket)
      int  socket;      /* descripteur de socket */
```

Le noyau essaie d'envoyer les données non encore émises avant de sortir de `close`().

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitives auxiliaires

Primitives permettant de convertir le format local (“ little indian ” ou “ big indian ”) en format réseau (toujours “ big indian ”) ou bien de faire l’opération inverse.

```
 #include  <sys/types.h>
#include  <netinet/in.h>

u-long   htonl (hostlong);   (host to network long)
      u_long   hostlong;

u-long   ntohl (hostlong);   (network to host long)
      u_long   hostlong;

u_short  htons  (netshort); (host to net short)
      u_short  netshort;

u_short  ntohs  (netshort); (net to host short)
      u_short  netshort;
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Primitive gethostbyname

Permet de construire l'adresse Internet à partir du nom d'une machine

```
#include <netdb.h>

struct  hostent  *gethostbyname (hostname)
      char  *hostname;
```

Retourne un pointeur sur une structure hostent :

```
struct  hostent  {
      char  *h_name;              /* nom de la machine  */
      char  **h_aliases;  /* liste d'alias  */
      int  h_addrtype;            /* AF_INET  */
      int  h_length;      /* 4 octets  */
      char  **h_addr_list;/* liste d'adresses IP,chaque adresse
            est donnée en adresse réseau */
};
#define  h_addr  h_addr_list[0]     //la 1ère adresse
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -