I2181-B
LINUX
APPELS SYSTÈME

LES SOCKETS

MODÈLE OSI

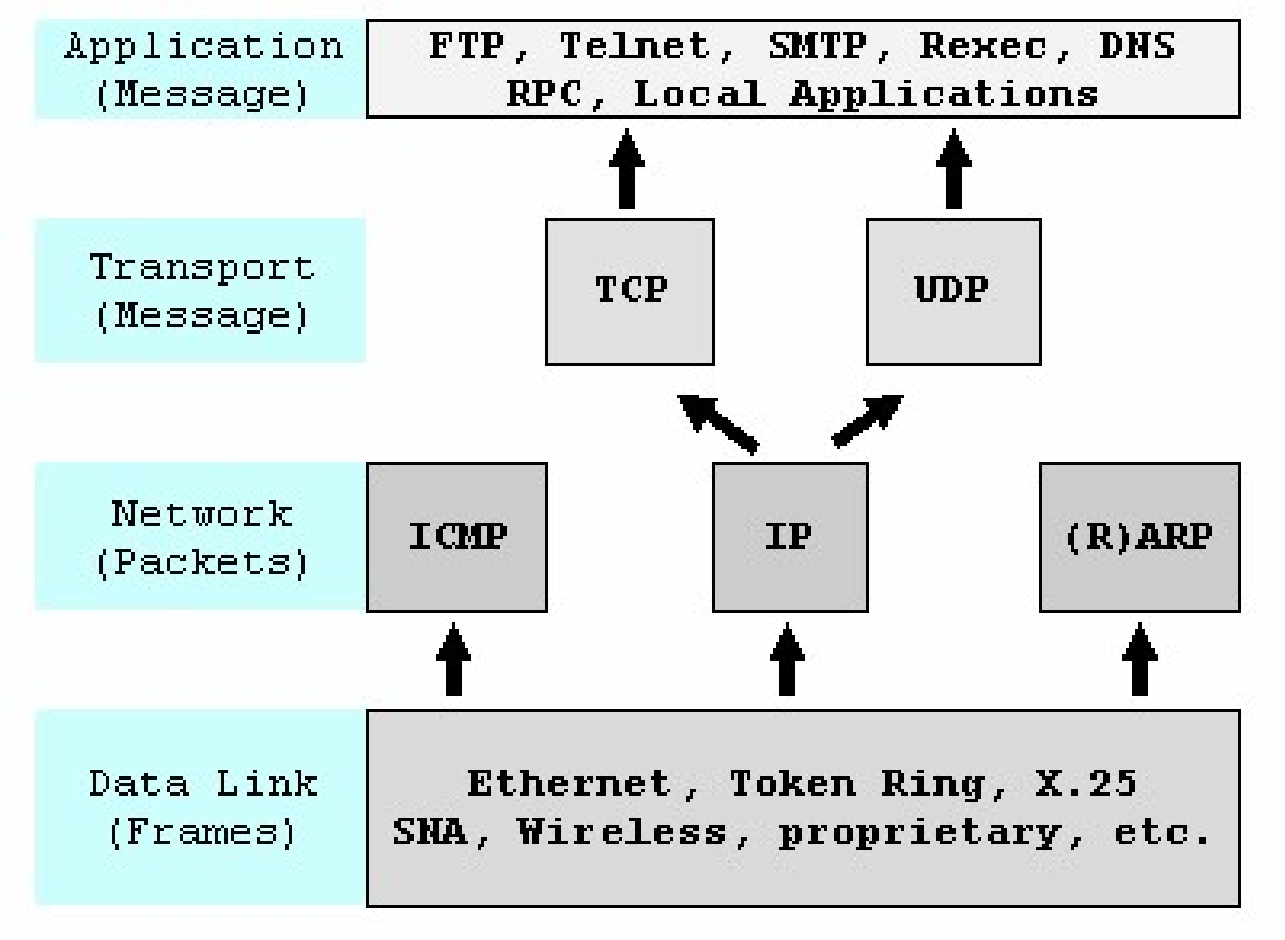


Image de :
https://www.tenouk.com/Module39.html

MODÈLE OSI & SOCKETS

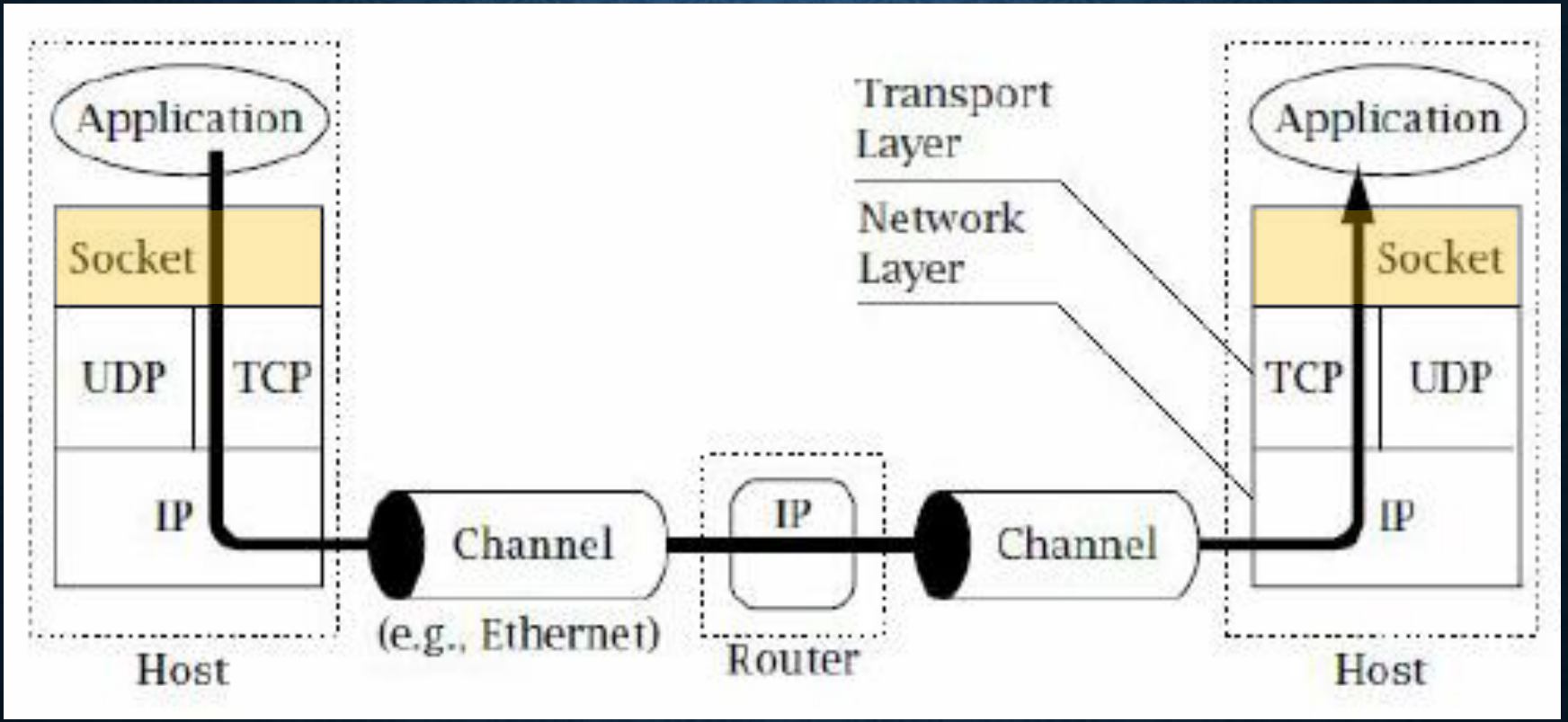
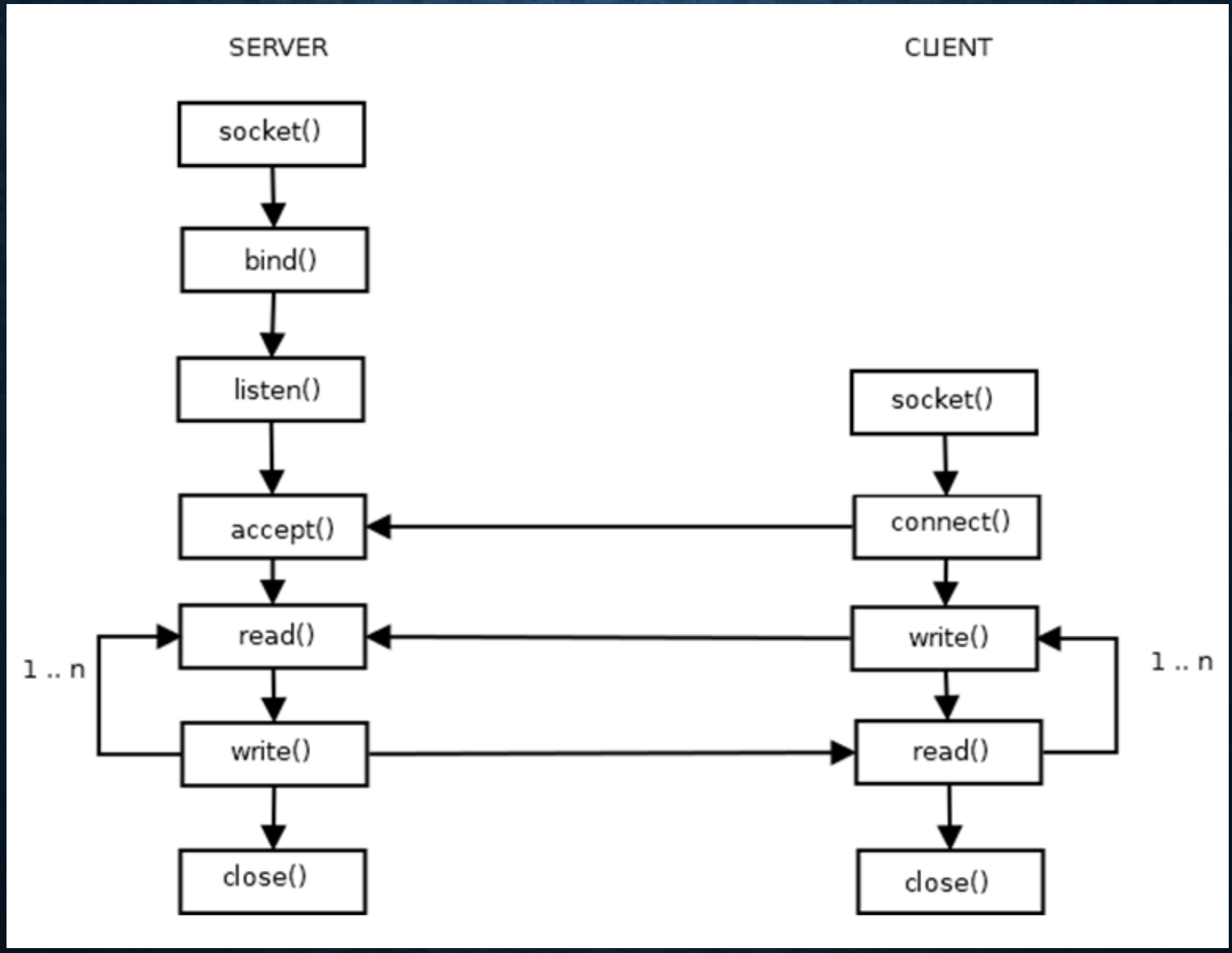


Image de :
https://www.researchgate.net/figure/A-TCP-IP-Connection_fig1_228728809

MODÈLE OSI & SOCKETS

- Les sockets se situent juste au dessus de la couche transport et s'interfacent donc avec cette couche …
  - les sockets peuvent donc être configurés pour travailler en TCP ou UDP
  - la couche transport utilise la notion de port, c'est pourquoi un socket devra définir un port (port d'écoute ou port de connexion)
  - l'architecture employée sera une architecture client/serveur
  - le serveur définit un port d'écoute
    - les clients se connectent sur ce port d'écoute

SOCKETS


SERVER
CLIENT
socket()
bind()
listen()
accept()
read()
write()
close()
socket()
connect()
write()
read()
close()
1 .. n
1 .. n

HEADERS

```
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

SOCKET

```
int socket (int domain, int type, int protocol);
```

- Crée un socket
- Renvoie un fd
- Renvoie -1 en cas d'erreur

SOCKET

- domain
  - AF_UNIX : communication locale
  - AF_INET : communication IPv4
  - AF_INET6 : communication IPv6
  - ….
- type
  - SOCK_STREAM : communication fiable (TCP)
  - SOCK_DGRAM : communication non fiable (UDP)
  - ….
- protocol
  - Permet d'indiquer un protocole particulier à utiliser avec le *type*
  - Il n'existe qu'un seul protocole actuellement par type → paramètre obsolète
  - Toujours 0

BIND

```
int bind (int sockfd, const struct sockaddr *addr,
          socklen_t addrlen);
```

- Demande la réservation d'un port et adresse auprès du système
- Renvoie -1 en cas d'erreur

SOCKADDR / SOCKADDR_IN

```
struct sockaddr {
    u_short sa_family;
    char sa_data[14];
};

struct sockaddr_in {
    sa_family_t sin_family;
    uint16_t sin_port;
    struct in_addr sin_addr;
    char sin_zero[8];
};
```

- SOCKADDR et SOCKADDR_IN sont 2 structures équivalentes en mémoire
- On préférera utiliser SOCKADDR_IN qui est plus précise
- Pour les fonctions utilisant SOCKADDR, un cast peut être utilisé

SOCKADDR_IN

```
struct sockaddr_in {
    sa_family_t sin_family;   // famille d'adresses: AF_INET
    uint16_t sin_port;        // port dans l'ordre d'octets réseau
    struct in_addr sin_addr; // adresse Internet
    char sin_zero[8];         // (for padding, to make it same
                              //  size as struct sockaddr)
};
```

- sin_family
  - idem socket
- sin_port
  - Port dans l'ordre d'octets réseau
    - little endian vs big endian
- sin_addr
  - Adresse dans l'ordre d'octets réseau

SOCKADDR_IN

```
int inet_aton (const char *cp, struct *sin_addr);
```

- Convertit une adresse IP au format A.B.C.D en une adresse réseau compatible avec la structure sockaddr_in
  - Ex: inet_aton("192.168.1.1", &myaddr.sin_addr);

```
uint32_t htonl (uint32_t hostlong);
uint16_t htons (uint16_t hostshort);
```

- Convertit un entier ou un entier long en un entier compatible avec l'ordre des octets réseaux de la structure sockaddr_in
  - Ex: htons(8080);

LISTEN

```
int listen (int sockfd, int backlog);
```

- Marque le socket *sockfd* comme une **socket passif**, càd. comme un socket qui sera utilisé pour accepter les demandes de connexions entrantes (via le syscall *accept*)
- Backlog
  - Valeur maximale de la file d'attente pour les clients
- Renvoie -1 en cas d'erreur

ACCEPT

```
int accept (int sockfd, struct sockaddr *adresse,
            socklen_t *longueur);
```

- Attend sur le socket passif *sockfd* les connexions des clients
- Crée et renvoie un nouveau **socket connecté** qui permettra d'échanger des messages avec le client
- sockaddr
  - Idem que pour socket
  - Permet de récupérer l'adresse du client se connectant
  - N'est pas obligatoire
- Renvoie -1 en cas d'erreur

CONNECT

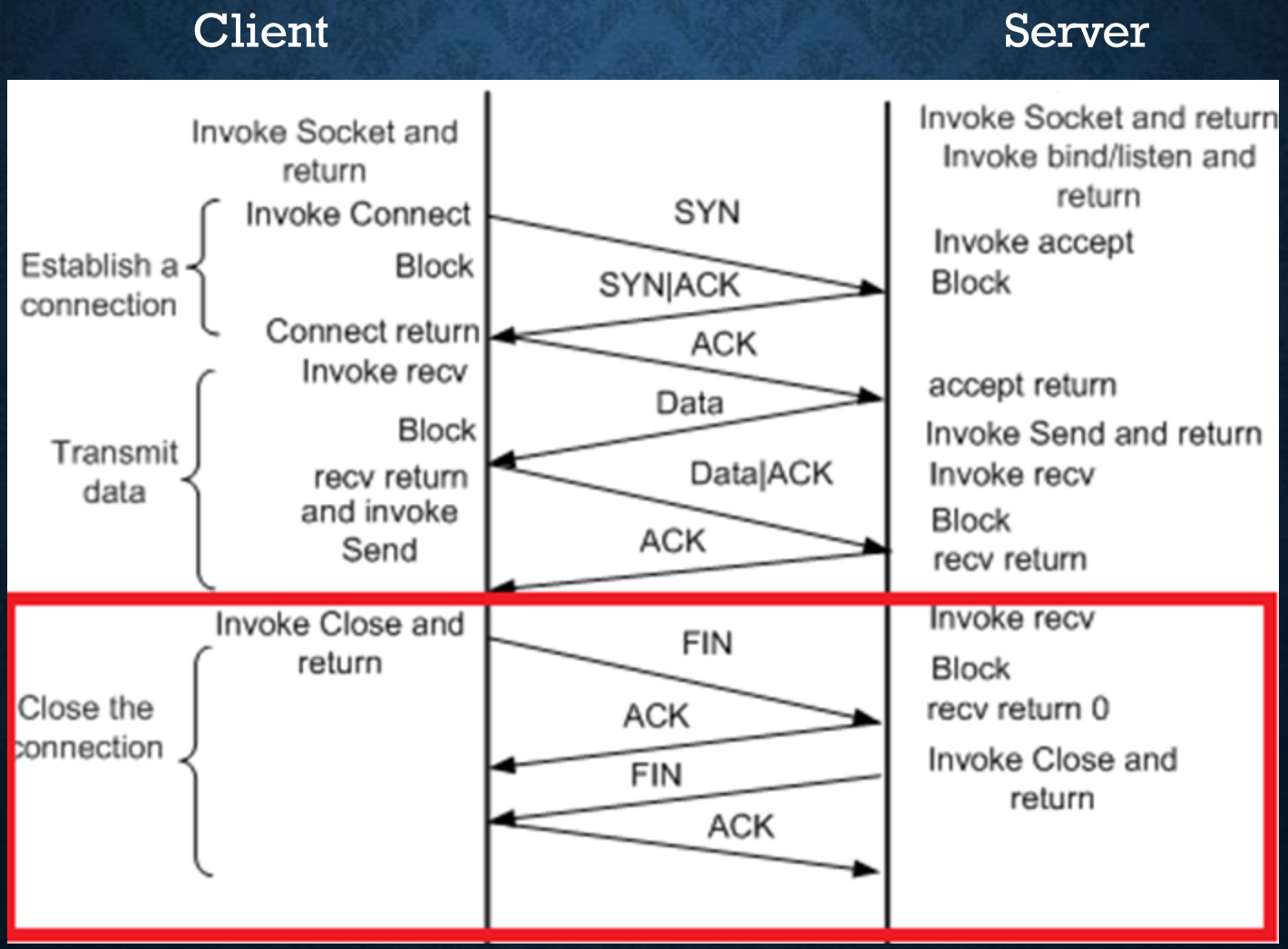
```
int connect (int sockfd,
             const struct sockaddr *serv_addr,
             socklen_t addrlen);
```

- débute une connexion sur un socket (côté client)
- sockaddr
  - Idem que pour socket
  - Permet d'indiquer l'adresse et le port du serveur auquel le client veut se connecter
  - Obligatoire
- Renvoie -1 en cas d'erreur

BIND: "ADDRESS ALREADY IN USE"

- Ce message survient quand une connexion utilisée par votre programme n'a pas été fermée correctement
  - Le système d'exploitation attend alors car il considère le port encore occupé
  - Le port sera libéré automatiquement après un certain temps (cela peut être long)
  - Souvent ceci a pour effet de ne plus pouvoir redémarrer le serveur sur le même port d'écoute

BIND: "ADDRESS ALREADY IN USE"



BIND: “ADDRESS ALREADY IN USE”

- Stratégies pour se prémunir contre ce problème
  - Fermer correctement les connexions (close)
  - En particulier n'oubliez pas de fermer proprement la connexion sur le client
  - Changer le port d'écoute du serveur
  - Utiliser la fonction `setsockopt()` avec l’option SO_REUSEADDR résout partiellement le souci
    - A faire avant le bind !

  Plus d’informations :
  https://hea-www.harvard.edu/~fine/Tech/addrinuse.html

EXEMPLE

Un exemple de communication entre un serveur et client

1. Voir `exemple17_Serveur.c`
2. Voir `exemple17_Client.c`