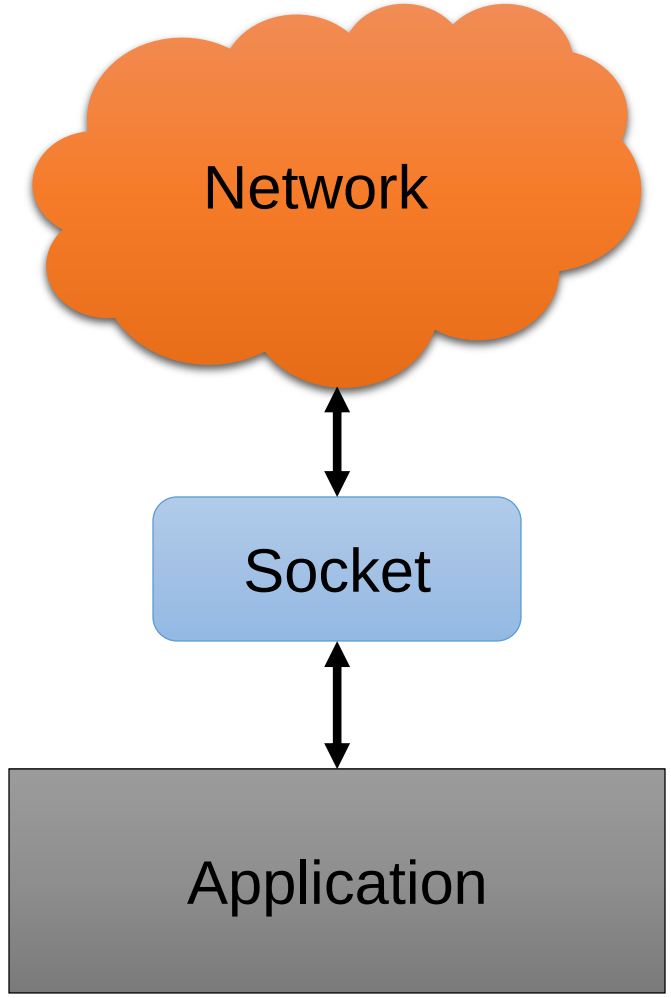
Sockets

- Provee una interface estándar para comunicar 2 procesos, tanto locales o ejecutándose en máquinas diferentes.
- Cuando un socket es creado, se debe especificar el dominio y el tipo. Dos procesos se comunican solo si sus sockets son del mismo dominio y tipo.
- Dominios principales:
  - UNIX domain: los procesos utilizan una entrada en el filesystem como dirección.
  - Internet domain: cada host tiene su IP address. Es necesario además un port sobre ese host.

Sockets en Internet domain

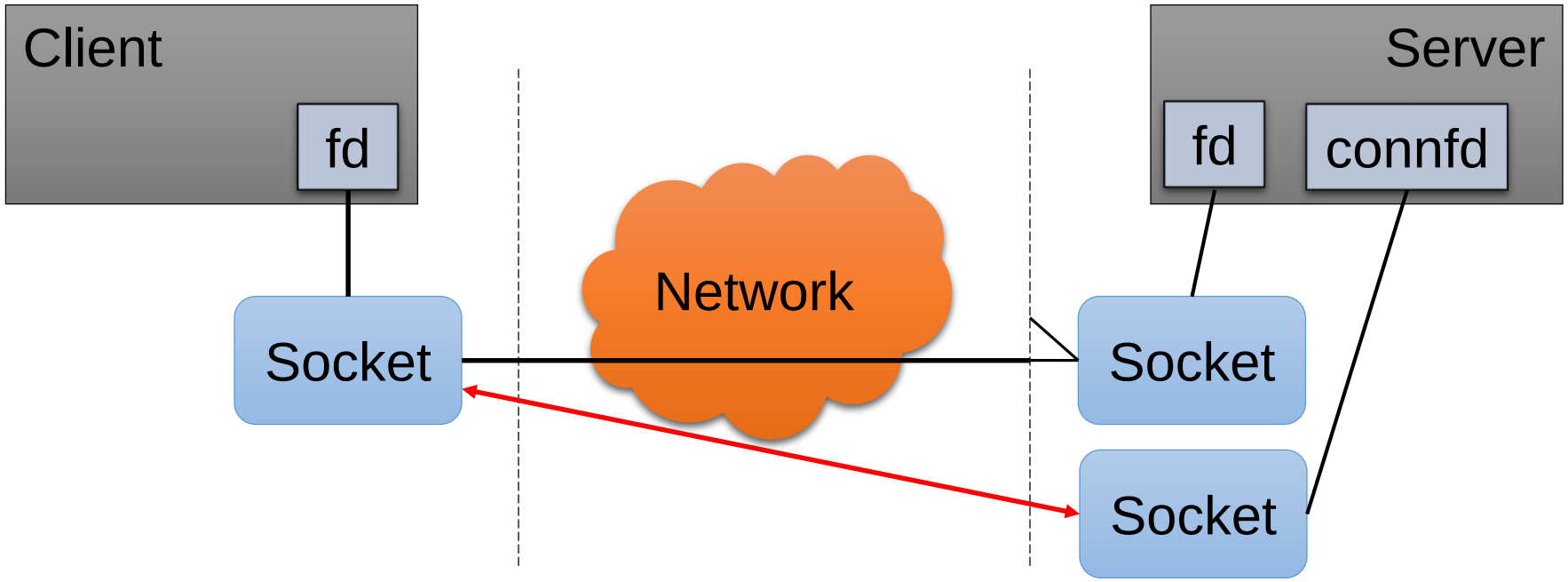
- Provee una interface estándar entre la aplicación y la red.
- 2 tipos:
  - Stream: utilizan TCP (transmission Control Protocol). Provee servicio de circuitos virtuales
  - Datagram: utilizan UDP (Unix Datagram Protocol) envío de paquetes individuales.



TCP/IP Connection

- Conexión como stream de bytes, confiable entre 2 computadoras.
  - Comúnmente usados en topologías cliente-servidor:
    - Un server escucha en un puerto determinado.
    - El cliente se conecta a ese puerto.
    - Una vez que se estableció la conexión, cualquiera de los lados puede escribir y/o leer.
- El API de sockets representa la conexión a través de un file descriptor.

TCP/IP Connection



TCP/IP Connection

Client

```
int fd = socket(…);




connect(fd, …, …);

write(fd, data, datalen);

read(fd, buffer, buflen);

close(fd);
```

Server

```
int fd = socket(…);

bind(fd, …, …);

listen(fd, …);

connfd = accept(fd, …);

read(connfd, buffer, buflen);

write(connfd, data, datalen);

close(connfd);
```

Creación de un socket

```
#include <sys/types.h>
#include <sys/socket.h>

...
...

int fd;
fd = socket(family, type, protocol);

if( fd == -1 ) {
    // Error: unable to create socket
    // actual error in "errno"...
}
...
```

AF_UNIX -> Unix socket
AF_INET -> IPv4
AF_INET6 -> IPv6

SOCK_STREAM -> TCP
SOCK_DGRAM -> UDP

0 (no usado en internet sockets)

- Crea un socket no ligado ni conectado a la red. Puede utilizarse tanto como server o como cliente.
- man page.

Binding de un server socket

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
if( bind(fd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) == -1) {
    // Error: unable to bind
    // actual error in "errno"...
}
...
```

- Necesario para servers. Generalmente no usado en clientes dado que tipicamente no importa que puerto local es utilizado.
- man page

listen sobre un server socket

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
if( listen(fd, 3) == -1) {
    // Error: unable to listen
    // actual error in "errno"...
}
...
```

- Marca al socket como aceptando conexiones.
- man page

connect

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
if (connect(fd,(struct sockaddr *)&serv_addr,sizeof(serv_addr)) < 0) {
    // Error: unable to connect
    // actual error in "errno"...
}
...
```

- Intentará hacer una conexión sobre un socket.
- man page

accept

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
clilen = sizeof(cli_addr);
connfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
...
```

- Extrae la primer conexión de la cola de conexiones pendientes o espera si **`O_NONBLOCK`** no se impuso en el descriptor (**`fcntl`**).
- man page

send

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
ret = send(sockFd, &buff, length, flags);
...
```

- Inicia la transmisión de un mensaje desde el socket hacia su partner.
- Es equivalente al syscall write si flags == 0
- man page

sendto

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
ret = sendto(sockFd, &buff, length, flags,
(const struct sockaddr *) &destAddr, sizeof(destAddr));
...
```

- man page

sendmsg

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
ret = sendmsg(sockFd, &msgHdr, flags);
...
```

- man page

struct msghdr

```
struct iovec {                    /* Scatter/gather array items */
    void  *iov_base;              /* Starting address */
    size_t iov_len;               /* Number of bytes to transfer */
};

struct msghdr {
    void         *msg_name;       /* optional address */
    socklen_t     msg_namelen;    /* size of address */
    struct iovec *msg_iov;        /* scatter/gather array */
    size_t        msg_iovlen;     /* # elements in msg_iov */
    void         *msg_control;    /* ancillary data, see below */
    socklen_t     msg_controllen; /* ancillary data buffer len */
    int           msg_flags;      /* flags on received message */
};
```

recv

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
ret = recv(sockFd, &buff, length, flags);
...
```

- Inicia la transmisión de un mensaje desde el socket hacia su partner.
- Es equivalente al syscall write si flags == 0
- man page

recvfrom

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
ret = recvfrom(sockFd, &buff, length, flags
                (struct sockaddr *) &destAddr, &destAddrLen);
...
```

- man page

recvmsg

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
ret = recvmsg(sockFd, &msgHdr, flags);
...
```

- man page

shutdown

```
#include <sys/types.h>
#include <sys/socket.h>

...
...
ret = shutdown(sockFd, SHUT_RDWR);
...
```

- man page

setsockopt/getsockopt

```
#include <sys/types.h>
#include <sys/socket.h>

int getsockopt(int sockfd, int level, int optname,
                        void *optval, socklen_t *optlen);

int setsockopt(int sockfd, int level, int optname,
                        const void *optval, socklen_t optlen);
```

- man page