Modelo Cliente Servidor – Interfaz Sockets

Introducción y Fundamentos

Carlos Taffernaberry
carlos.taffernaberry@um.edu.ar

Modelo Cliente/Servidor–Socket
Bibliografía:

Internetworking with TCP/IP

Principles,Protocols and Architectures

4th. Edition – Douglas Comer

Computer Networks 5th. Edition – Andrew Tanenbaum

Unix Network Programming 3rd. Edition – Richard Stevens

Modelo Cliente/Servidor–Socket

Programa:

Modelo Cliente Servidor

- Filosofía, y características
- Persistencia, puertos y concurrencia

Interfaz Socket

- API estandar, funciones para su uso

Modelo Cliente/Servidor–Socket
Introducción:

Servicios sobre TCP/IP:

- Programas de aplicación basados en el uso cooperativo de Internet.
- Interacción entre programas de distintas computadoras. (IPC)
- Soporte de aplicaciones distribuidas
- Interacción de múltiples sistemas independientes y distintos.

Modelo Cliente/Servidor–Socket

Distintos modelos de comunicación:

Modelo Centralizado

- Problemas escalabilidad / interoperabilidad

Modelo Cliente / Servidor

- Flexibilidad / interoperabilidad / escalabilidad

Modelo Peer to Peer

- Cliente y servidor simultaneamente.
- Información distribuida / robustez
- Problemas seguridad / anonimato

Modelo Cliente/Servidor–Socket

Cliente servidor

- Terminología

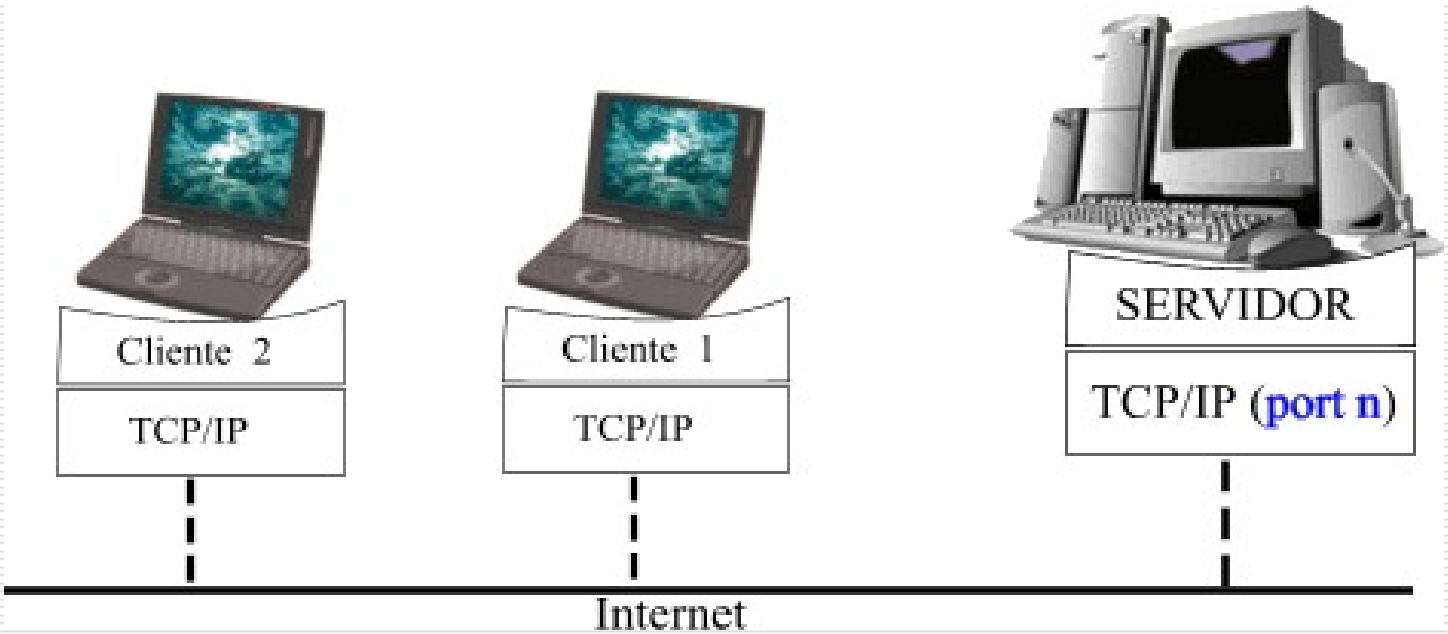
Servidor: **Aplicación** accesible a través de la red que cuenta con datos o información a la espera requerimientos por parte de los Clientes.

Cliente: **Aplicación** que solicita algún tipo de información a un Servidor y espera por su respuesta.

Protocolo: acuerdo entre ambas aplicaciones de como realizar la comunicación (ej. RFC's)

Modelo Cliente/Servidor–Socket

Cliente servidor



Cliente envia requerimiento ---------->

<---------- Servidor envía respuesta

Modelo Cliente/Servidor–Socket

Cliente servidor

- Características Servidor

Espera requerimientos, procesa y responde.

Siempre ejecutandose y generalmente concurrente.

Se implementan como programas en la capa de aplicación (resultan portables a todo sistema con TCP/IP)

Well Known ports. Por que ? (/etc/services)

Usualmente aplicaciones complejas.

Debe evaluar autorización de clientes, acceso a bases de datos, validar el requerimiento, etc.

Modelo Cliente/Servidor–Socket
Cliente servidor

- Características Cliente

Se ejecuta solo para hacer requerimientos.

Puerto local aleatorio sin usar gralmente. asignado por SO.

Debe conocer IP servidor (o su nombre).

Debe conocer Puerto servidor (o el servicio).

Inicia la comunicación.

Aplicación gralmente menos compleja.

Modelo Cliente/Servidor–Socket
Interfaz Socket

- Como usan los programas de usuario la red ?

Modelo Cliente/Servidor–Socket

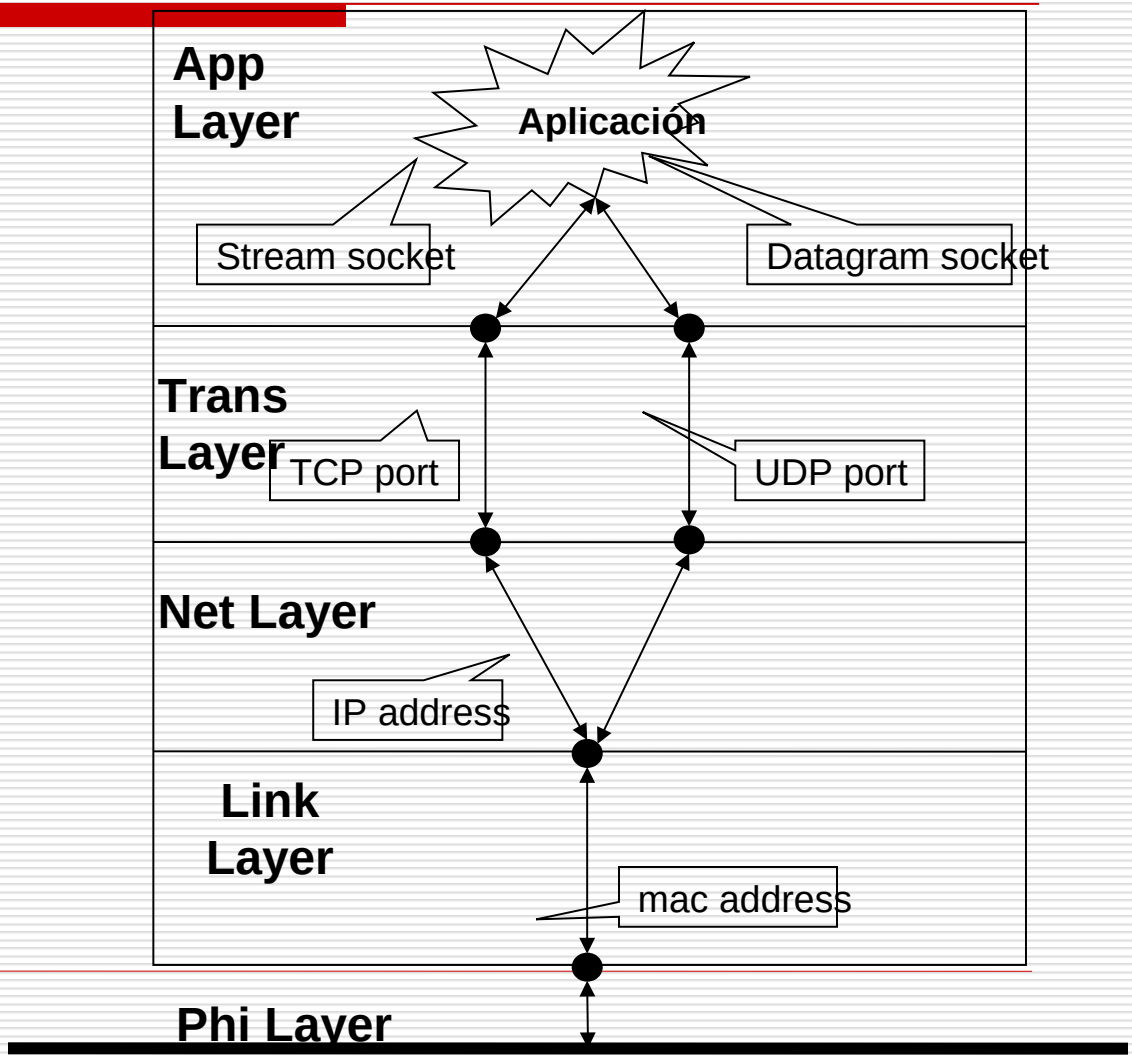
Interfaz Socket

- Introducción
  - Protocolos de red incluidos en SO.
  - Aplicaciones corren fuera del SO.
  - Necesidad de API (Application Program Interface) para acceder a protocolos de red.
  - No hay estandar definido por RFC's.
  - Socket: definido en Unix BSD y adoptado por todos los SO como estandar defacto.
  - Otras API: XTI (TLI) X/Open Transport Interface y Win Sockets.

Modelo Cliente/Servidor–Socket

Interfaz Socket

Esquema Conexión



Modelo Cliente/Servidor–Socket
Interfaz Socket

- Características
  - Sigue paradigma Unix open-read-write-close. (Todo es un archivo)
  - Usa abstracción de descriptor de archivo.
  - Forma de uso:
    - Crea un socket
    - Usa funciones específicas de socket
    - Una vez establecido escribe y/o lee
    - Finalmente Cierra el socket

Modelo Cliente/Servidor–Socket

Interfaz Socket:

- **Endpoint:**
  - Par (Dirección, Puerto)
- **Socket Pasivo:**
  - Asociado a un solo endpoint (local)
  - Ej. Server en estado listen.
- **Socket activo:**
  - Asociado a dos endpoints (local&remoto)
  - Usado para enviar/recibir datos.
  - Cliente o servidor en estado established

Modelo Cliente/Servidor–Socket

Interfaz Socket: Funciones

- **Socket()**

Abre un canal bidireccional de comunicaciones. Socket crea un punto terminal para conectarse a un canal y devuelve un descriptor. El descriptor se usará en llamadas posteriores a funciones de la interfaz.

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

- **Bind()**

Se utiliza para unir un socket a una dirección de red determinada

```
int bind(int s, const struct sockaddr *name, int namelen)
```

- **Listen()**

Sirve para que un proceso servidor indique que está disponible para recibir peticiones de conexión.

```
int listen(int s, int backlog)
```

Modelo Cliente/Servidor–Socket

Interfaz Socket: Funciones cont.

- **Accept()**

Sirve para que los procesos servidores lean las peticiones de servicio de los clientes.

```
Int accept(int s, struct sockaddr *addr, int *addrlen)
```

- **Connect()**

Sirve para que un proceso cliente inicie una conexión con un servidor a través de un socket.

```
int connect(int s, const struct sockaddr *name, int
namelen)
```

Modelo Cliente/Servidor–Socket

Interfaz Socket: Funciones cont.

- **Send()**

Las llamadas para escribir datos a un socket son: write, send, sendto y sendmsg. La llamada write es la misma que se usa para manejar archivos.
Las llamadas send, sendto y sendmsg se usan para escribir datos en un socket exclusivamente

```
ssize_t  send(int s, const void *msg, size_t len, int
flags)
ssize_t sendto(int s, const void *msg, size_t len, int
flags, const struct sockaddr *to, int tolen)
      ssize_t sendmsg(int s, const struct msghdr *msg,
      int flags)
```

Modelo Cliente/Servidor–Socket

Interfaz Socket: Funciones cont.

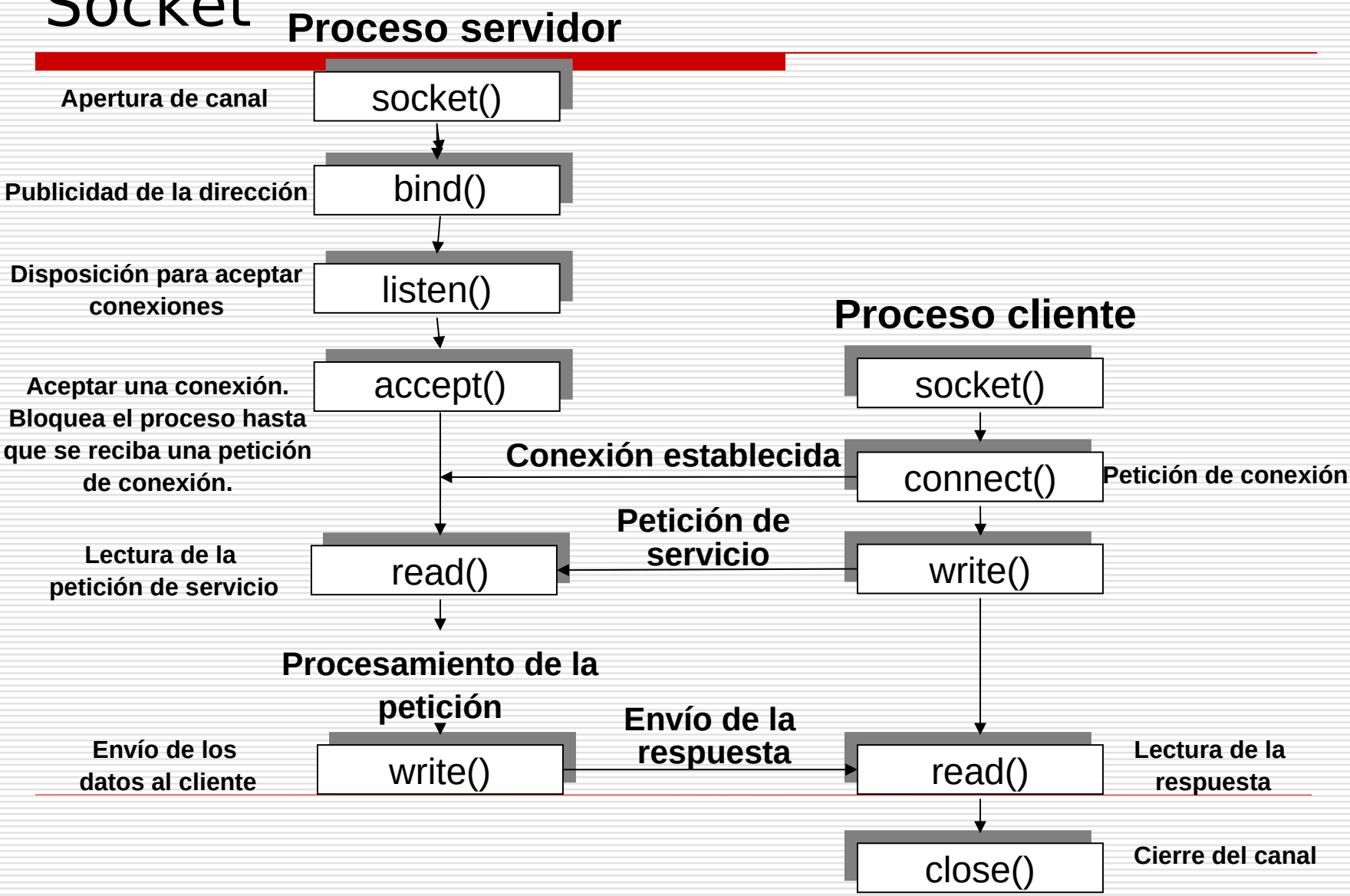
- **Rec()**

Las llamadas para leer un datos de un socket son: read, recv, recvfrom y recvmsg. La llamada read es la misma que se usa para manejar archivos.
Las llamadas recv, recvfrom y recvmsg se usan para leer datos de un socket exclusivamente.

```
ssize_t recv(int s, void *buf, size_t len, int flags)
ssize_t recvfrom (int s, void *buf, size_t len, int
flags, struct sockaddr *from,  int *fromlen)
ssize_t recvmsg (int s, struct msghdr *msg, int flags)
```

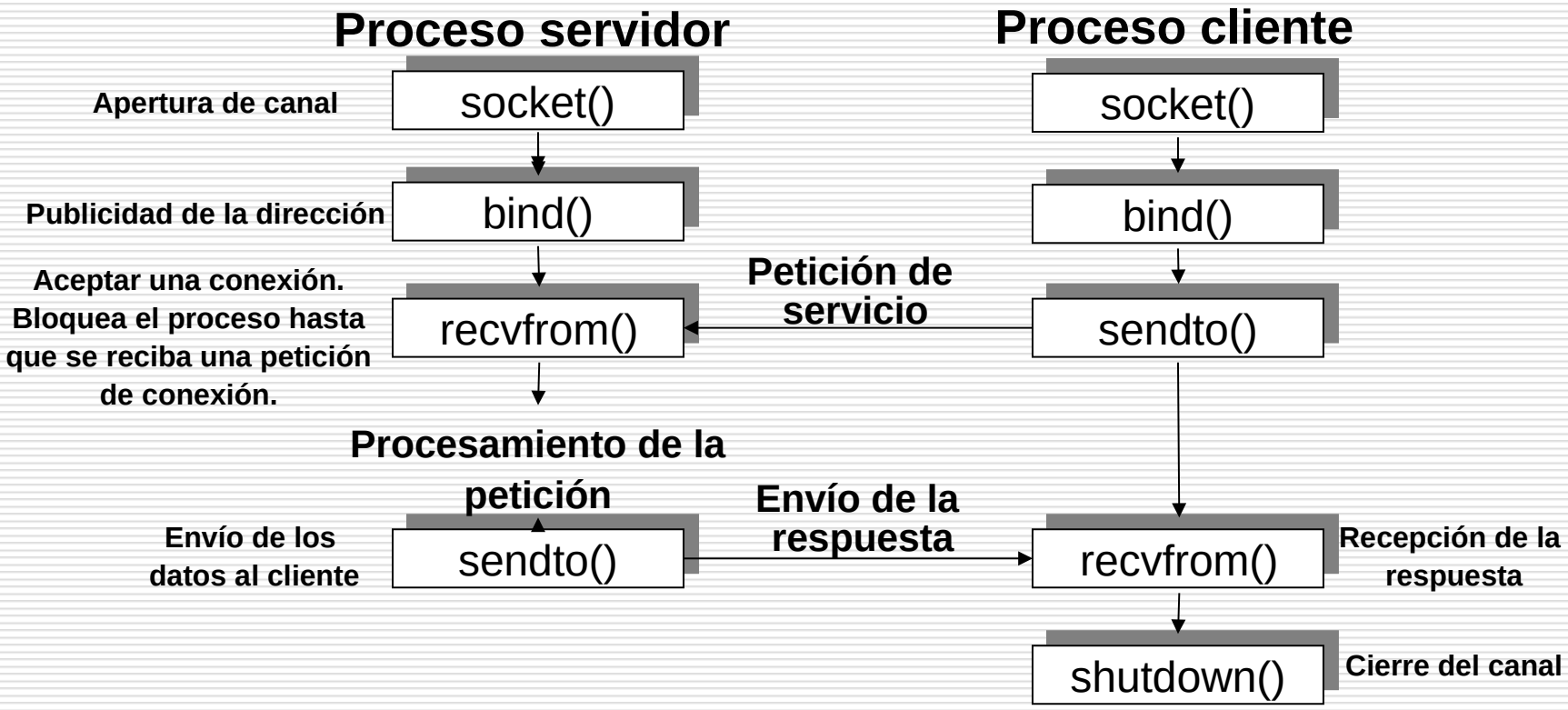
Modelo Cliente/Servidor–Socket

Socket



Modelo Cliente/Servidor–Socket

Socket udp



Modelo Cliente/Servidor–Socket

Interfaz Socket: estructuras

```
/*
 * Address families.
 */
#define pseudo_AF_XTP   19              /* eXpress Transfer Protocol (no AF) */
#define AF_COIP         20              /* connection-oriented IP, aka ST II */
#define AF_CNT          21              /* Computer Network Technology */
#define pseudo_AF_RTIP  22              /* Help Identify RTIP packets */
#define AF_IPX          23              /* Novell Internet Protocol */
#define AF_SIP          24              /* Simple Internet Protocol */
#define pseudo_AF_PIP   25              /* Help Identify PIP packets */
#define AF_ISDN         26              /* Integrated Services Digital Network*/
#define AF_E164         AF_ISDN         /* CCITT E.164 recommendation */
#define pseudo_AF_KEY   27              /* Internal key-management function */
#define AF_INET6        28              /* IPv6 */
#define AF_NATM         29              /* native ATM access */
#define AF_ATM          30              /* ATM */

#define AF_MAX          31
```

Modelo Cliente/Servidor–Socket

Interfaz Socket: Conexiones concurrentes

- Sockets pasivos & Sockets activos (endpoints)