Sockets

Programación en Red.

Interproccess Comunication

- Mecanismos que permite la comunicación entre procesos que se ejecutan sobre la misma máquina.
  - Pipes, messages queues, semáforos, shared memory.
- Se da a través del paso de mensajes.
- Un proceso denominado fuente, envía un mensaje (secuencia de bytes).
- Otro proceso denominado destino, recibe el mensaje y lo procesa.
- Aspectos como la sincronización de los dos procesos deben ser tenidos en cuenta.
- La comunicación puede ser sincrónica o asincrónica.

Arquitectura Cliente/Servidor

- Considerado uno de los principales patrones arquitectónicos de interacción entre las aplicaciones que usan un red de datos
- Comunicación asimétrica:
  - Cliente envía una petición y el server envía una respuesta.
- Server operando como un daemon:
  - Un nombre y puerto bien conocido.
  - Espera a que lo contacten.
  - Procesa la petición y envía la respuesta.
- Cliente:
  - Inicia el contacto y espera por la respuesta.

Modelo Cliente/Servidor

- Un servidor es un programa que ofrece un servicio sobre una red.
- Se puede considerar como una extensión natural de IPCs.
- En este sentido, un servidor acepta una petición entrante, ensambla una respuesta y la envía al cliente.
  - El servidor puede desarrollar una tarea simple o compleja acorde a la petición del cliente.
- Un proceso se denomina cliente, cuando este envía una petición al servidor y espera por una respuesta.
- Un servidor siempre se ejecuta antes con el fin de que se pueda dar la interacción con el cliente.
- Una vez el cliente recibe la respuesta, usualmente finaliza.

Servidores: Secuencial vs Concurrente.

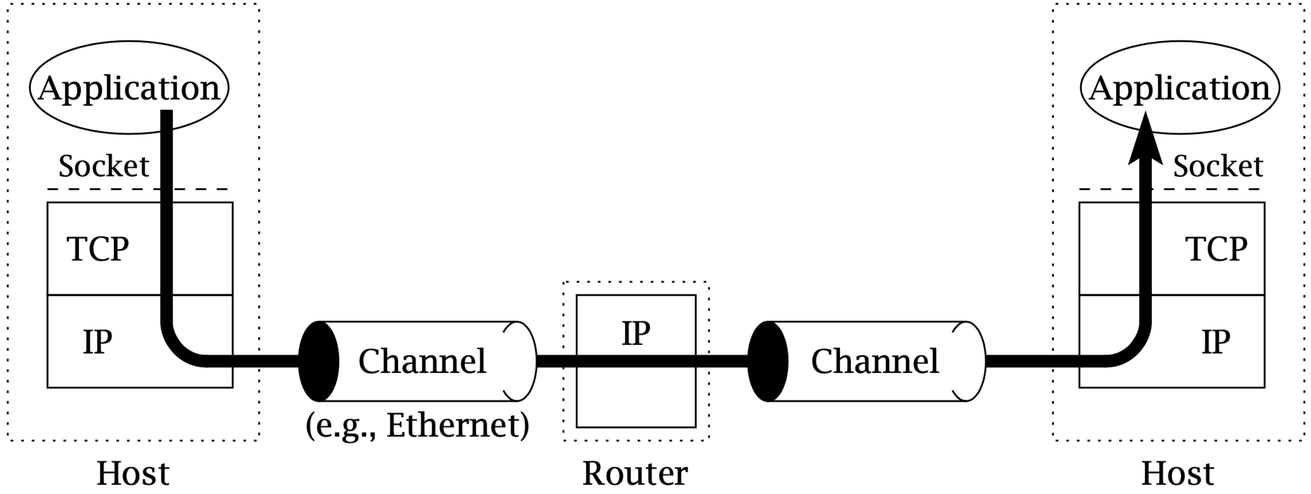
- Un proceso servidor secuencial es aquel que procesa una petición al tiempo (desempeño lento…)
- De esta forma, antes de aceptar una nueva petición envía la respuesta de la petición que esta procesando.
- ¿Qué sucede si una nueva solicitud llega mientras el servidor esta procesando una petición?
  - Las peticiones entrantes pueden ser ubicadas en una cola.
  - Servidores concurrentes: un servidor puede atender múltiples peticiones de manera simultánea.
    - La mayoría de los servidores en producción implementan la concurrencia.

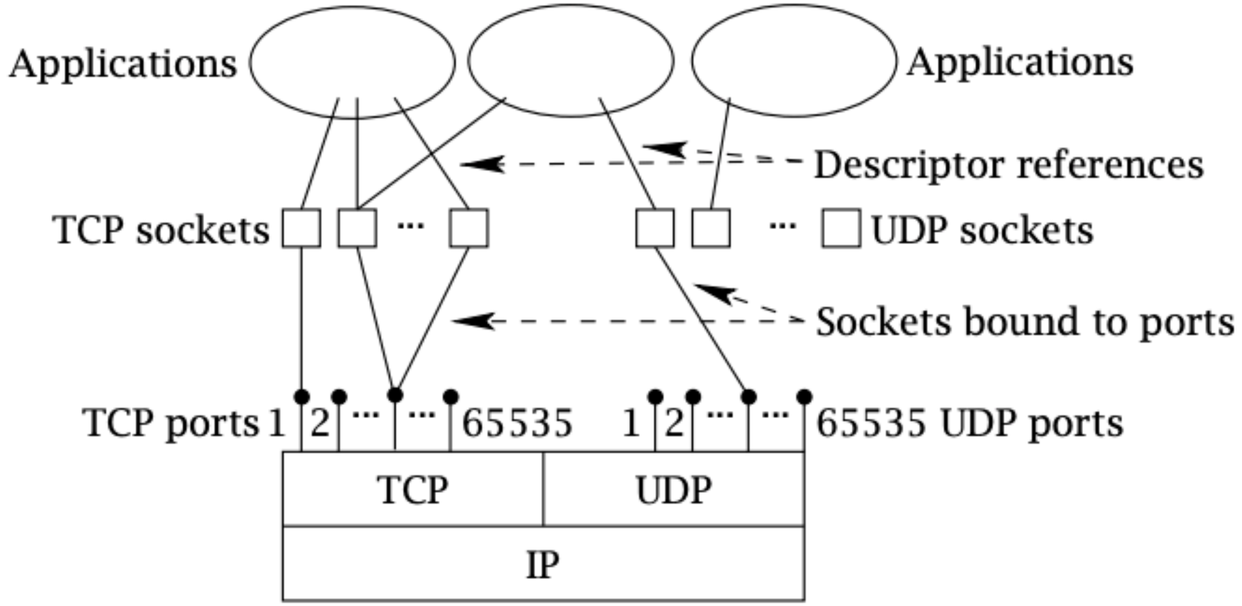
Programación en Red

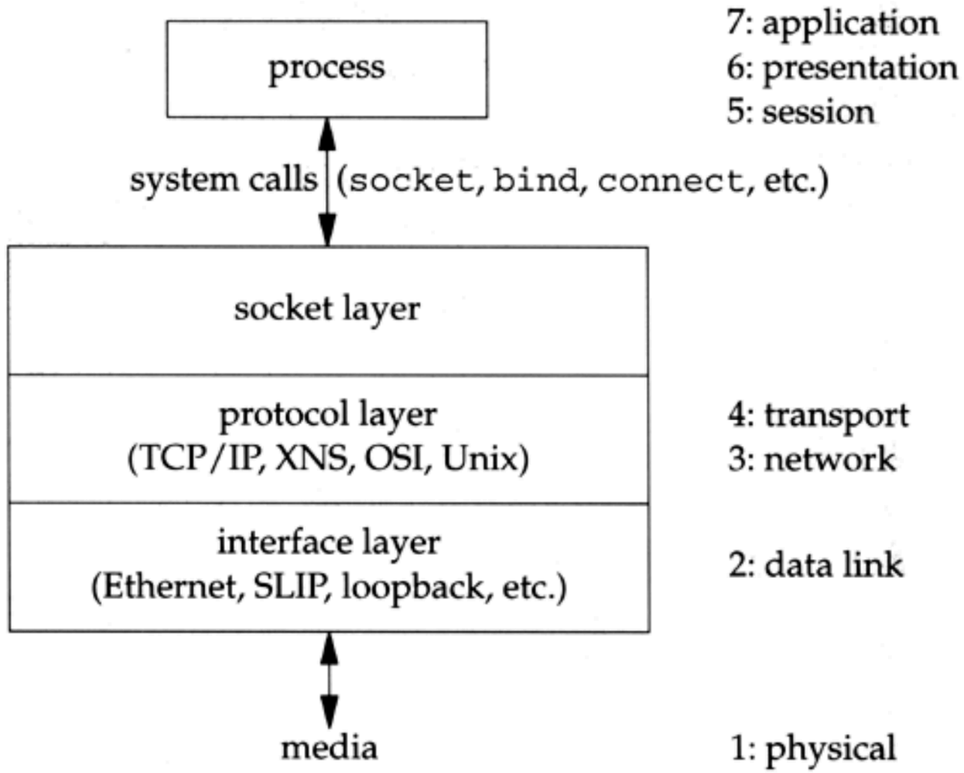
- ¿Cómo se deben comunicar dos host entre si en Internet?
  - A través de la arquitectura de red con los protocolos de la capa de Internet y de transporte.
- ¿Cómo deben interactuar los programadores con los protocolos de red?
  - A través de la API Sockets.
  - Estándar de facto para la programación en red.

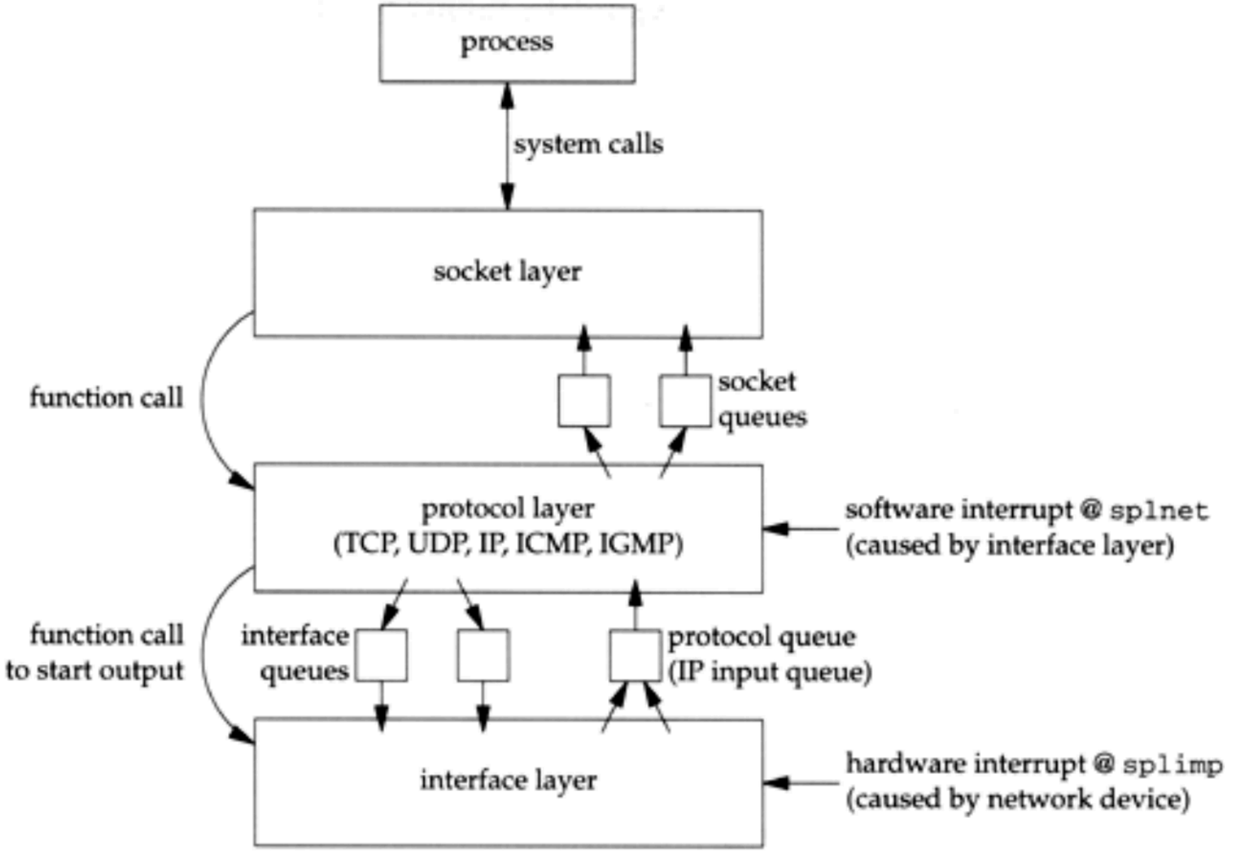
¿Qué es un Socket?

- Es una abstracción para un punto de comunicación por la cual una aplicación puede enviar o recibir datos.
- Interface entre la capa de aplicación y la capa de transporte.
- Proporciona una abstracción a los procesos de tal forma que cada proceso solo tenga que operar en términos de flujos de datos en lugar de paquetes.
- Un punto de comunicación bidireccional que se utiliza para el intercambio de datos entre procesos locales y remotos.
- Para lograr que dos procesos se comuniquen a través de un socket, se necesita una infraestructura de comunicaciones pre-existente.
    - Red de datos.



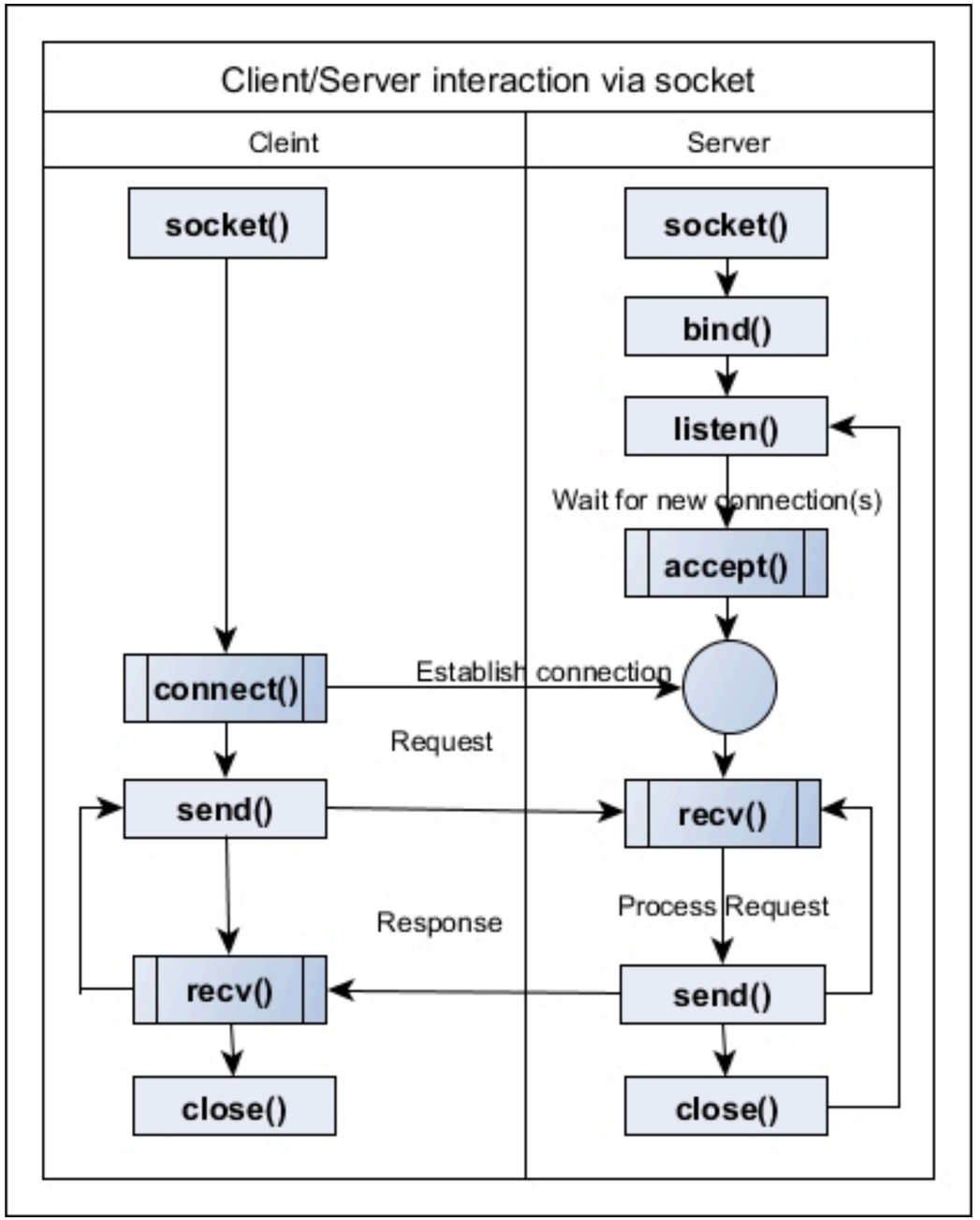



process
system calls (socket, bind, connect, etc.)
socket layer
protocol layer
(TCP/IP, XNS, OSI, Unix)
interface layer
(Ethernet, SLIP, loopback, etc.)
media
7: application
6: presentation
5: session
4: transport
3: network
2: data link
1: physical


process
system calls
socket layer
function call
socket
queues
protocol layer
(TCP, UDP, IP, ICMP, IGMP)
software interrupt @ splnet
(caused by interface layer)
function call
to start output
interface
queues
protocol queue
(IP input queue)
interface layer
hardware interrupt @ splimp
(caused by network device)

¿Cómo Funcionan los Sockets?

- Sockets son usados normalmente para la interacción cliente/servidor.
- Los clientes se conectan al servidor, intercambian información y luego se desconectan.
- Un socket, tiene normalmente un flujo de eventos:
  - Como se observar en la figura, en un modelo cliente/servidor orientado a la conexión, el servidor espera por conexiones de los clientes.
  - Para esto el servidor establece primero una dirección donde puede ser contactado por el cliente.
  - En este punto, ya el servidor puede recibir peticiones del cliente.
  - Ahora proceden a intercambiar datos.



Tipos de Sockets

- En la actualidad, existen diferentes tipos de sockets, para efectos de este curso revisaremos los sockets en el Stack de protocolos TCP/IP.
- Los principales tipos de sockets en esta arquitectura son:
  - Stream sockets. Emplean protocolo TCP y por ende ofrecen un servicio de transmisión de dataos confiable.
  - Datagram sockets. Emplean protocolo UDP y por lo tanto ofrecen un servicio de transmisión de datos basado en el mejor esfuerzo.

Actividad de Laboratorio

- Crear una aplicación cliente/servidor que se comuniquen a través de la API sockets. En este caso se utilizarán sockets TCP.
- La aplicación cliente debe ser capaz de enviar comandos y datos a la aplicación servidor (p.ej. HELO, DATA y QUIT). El servidor debe enviar un mensaje de respuesta al cliente, confirmando o indicando una respuesta.
- Ejemplo de secuencia:
    - C:HELO
    - S:100 OK\n
    - C:DATA
    - C:Input data to send: Hello world
    - S:300 DRCV\n
    - C:QUIT
    - S:200 BYE\
- Cuando se envíe un comando desconocido, el servidor debe enviar un mensaje de respuesta con código 400 indicando que no es un comando válido:
    - C:USER
    - S:400 BCMD\r\n
    - Command-description: Bad command