Módulo Nº 5

Interfaz de Socket

Agenda

- Definición de Socket.
- Tipos de Aplicaciones.
  - Orientadas a conexión.
  - No-orientadas a conexión.
- Tipos de Servidores:
  - Iterativos.
  - Concurrentes.
- Systemcalls.

Sockets

- Un socket define el punto de contacto por el que se trasmite/recibe información en un host dado. Esta identificado por:

  {Protocolo de Transporte, Dirección IP, Puerto}

- Una comunicación entre dos aplicaciones cooperativas, que incluyen igual protocolo de transporte, queda definida por la quíntupla:

  {Protocolo, Direcc_IP_local, Puerto_local, Direcc_IP_remota, Puerto_remoto}

- El protocolo de transporte define dos tipos de sockets:
  - ***Sockets de Flujo ("Stream Sockets"):***
    - Utilizan protocolos de transporte orientados a conexión (TCP).
    - Definirán aplicaciones Orientadas a Conexión.
  - ***Sockets de Datagrama ("Datagram Sockets"):***
    - Utilizan protocolos de transporte no-orientados a conexión como (UDP).
    - Definirán aplicaciones No-Orientadas a Conexión.

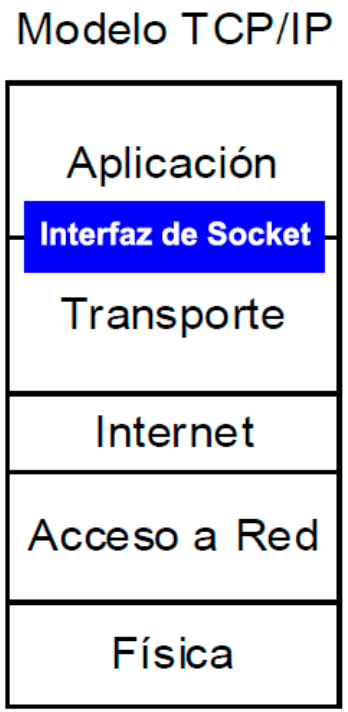
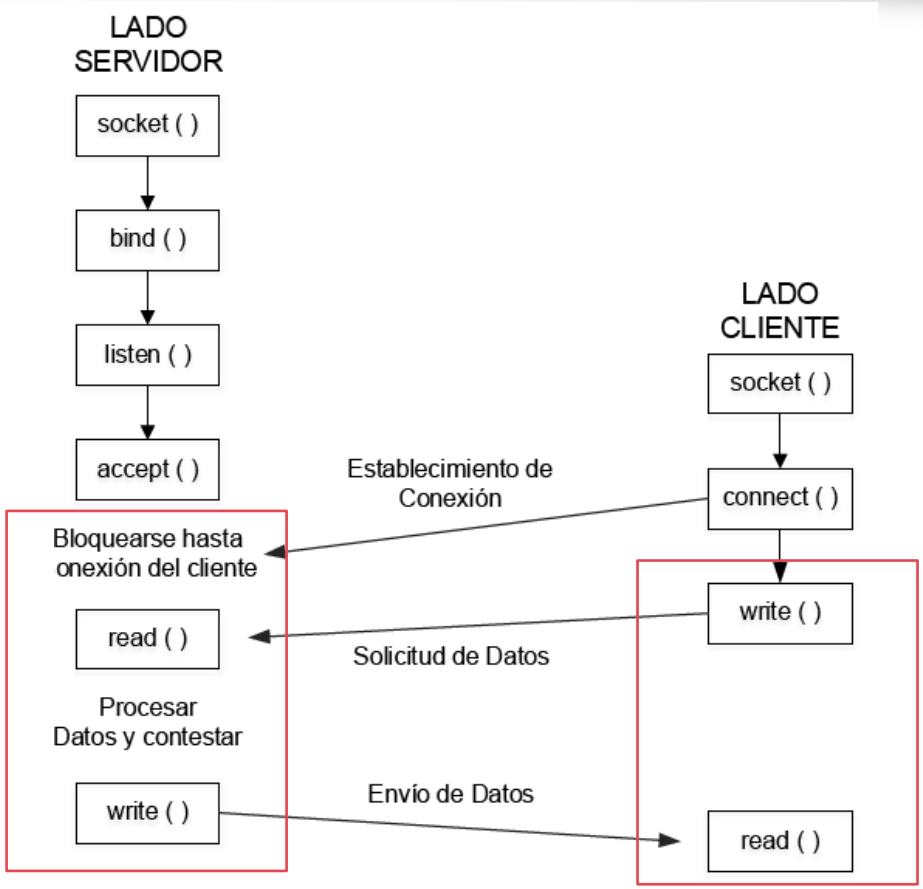


Diagrama de interacción cliente-servidor (sockets de flujo)



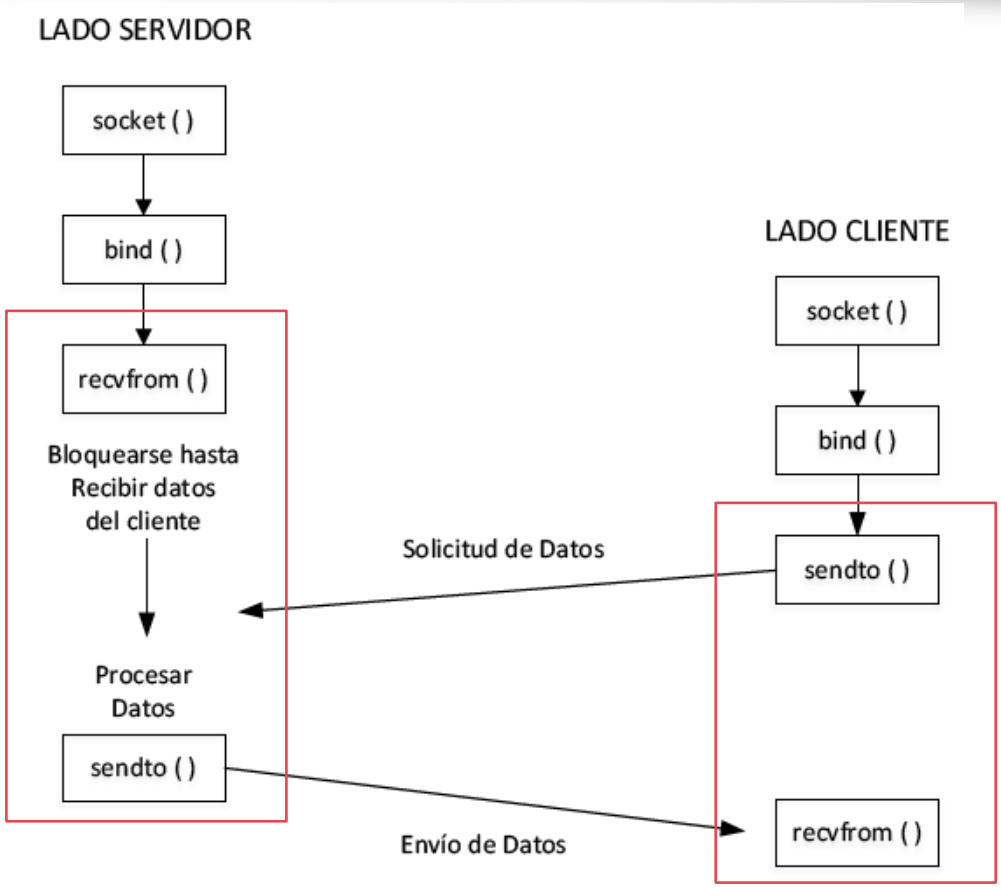
Aplicaciones orientadas a conexión: aplicación servidor

- Los pasos que se siguen son:
  1. **socket():** se crea un socket. Este paso simplemente asocia la aplicación al protocolo de transporte – TCP o UDP.
     - No asocia a la aplicación con número de puerto ni dirección IP.
  2. **bind():** se asocia el socket con la dirección IP (**local**) y el puerto (**local**).
  3. **listen():** se crea un objeto tipo cola para las solicitudes de conexión futuras.
  4. **accept():** la aplicación servidora se bloquea esperando conexiones de clientes.
  5. Cuando se establece la conexión con el cliente, el servidor realiza la tarea correspondiente.
     - En el ejemplo, lee los datos recibidos desde el cliente con la Sys call read() y luego de procesados los datos, envía un "reply" con la Sys call **write().**
  6. Continúa la comunicación hasta que una de las partes realiza un **close().**
     - La system call close() cierra el socket y lleva implícita el cierre de la conexión TCP.

Aplicaciones orientadas a conexión: aplicación cliente

- Los pasos que se sigue son:
  - **socket():** asociar la aplicación al protocolo de transporte que va a utilizar (TCP o UDP.
  - **connect():** realiza la conexión con el lado remoto. Se completa la quíntupla al asociar IP origen y destino y Puertos origen y destino.
  - Inicia la transacción de datos.
    - En el ejemplo: envía datos al servidor y espera por una respuesta de este.
    - **Nota:** observar que en el lado cliente **no** se utiliza el sys call bind().

Diagrama de interacción cliente-servidor (sockets de datagrama)



Aplicaciones no orientadas a conexión

- En este caso, la secuencia de eventos entre Servidor y Cliente es mucho más simple.

- Tanto en Servidor como en Cliente, luego de creado el socket y haber sido asociado a una dirección IP y a un puerto, inicia el intercambio de datos.
  - Las system call de envío y recepción de datos permiten especificar la dirección IP y puerto de la aplicación a la que se envía o de la cual se recibe la información.

Sys Calls de envío y recepción

- En general las system call de:

  - Envío de datos son no bloqueantes es decir que, luego de invocadas, siguen con la ejecución de la próxima línea de código.

  - Recepción de datos son bloqueantes, es decir que si se ejecutan y no tienen datos para leer, esperan el arribo de los mismos.

Servidores iterativos y concurrentes

- Iterativos:
  - Los requests de los clientes son atendidos por el servidor (propiamente dicho) en un tiempo acotado (relativamente corto).
  - Cuando el servidor termina de atender el request, vuelve a esperar por otro cliente.
  - Este tipo de servidores se utiliza cuando el procesamiento de la petición del cliente es simple y "acotada", por lo que normalmente la respuesta se realiza en forma rápida y con un intercambio mínimo de información.
  - Ejemplo de uso: servicio "time-of-day".
- Concurrentes:
  - El servidor no sabe a-priori el tiempo de atención de la solicitud del cliente.
  - El servidor crea otro proceso (concurrente) para atender la solicitud del cliente.
  - El servidor vuelve a esperar por otro cliente, creando un nuevo proceso hijo cuando se presenta.
  - Necesita de un sistema operativo multitarea para la creación y administración de tareas concurrentes.
  - Ejemplo de uso: transferencia archivos.

Quíntupla y System Calls

- La siguiente tabla muestra las system calls que completan la quíntupla de un socket.

Aplicación/Parámetro	Protocolo	Loc_Add, Loc_Port	Rem_Add, Rem_Port
Servidor Orientado a Conexión	socket( )	bind( )	listen( ), accept( )
Cliente Orientado a Conexión	socket( )	connect( )	
Servidor No-Orientado a Conexión	socket( )	bind( )	recvfrom( )
Cliente No-Orientado a Conexión.	socket( )	bind( )	sendto( )

Pautas para el Laboratorio

- Trabajo grupal: mismos grupos que para las exposiciones orales.
- Entregar por mail hasta el 18/11 a las 19:00 hs.
  - Código fuente de ambas aplicaciones e informe.
- La nota se promedia con las exposiciones orales.
  - Si no se presenta, la nota es CERO y no se puede rendir libre.
- Se recomienda leer la bibliografía.
  - Libro de Ing. Sergio D. Saade.
  - Material adicional:
    - Programación Distribuida con Interfaz de Socket en IPv6.
    - Programación en Redes TCP-IP Utilizando Interfaz de Sockets.