Sockets bloquantes

Servicios definidos en la capap de transporte

La capa de transporte del modelo OSI (y su equivalente en el modelo TCP/IP) es el responsable de;

- Encarrilamiento extremo a extremo (end to end)
- Multiplexacion (Diferentes puertos para diferentes servicios)
- Control de flujo y, si aplica, correccion de errores reenvio de paquetes

En el stack TCP/IP los servicios mas utilizados en la capap de transporte son:

- TCP (Transmission Control Protocol): Proporciona un servicio orientado a conezion, confiable (Con verificacion de entrega, control de congestion, reensamblado etc.)
- UDP (User Datagram Protocol): Proporciona un servicio no orientado a conexion, con entrega "best effort" (sin garantias de llegada, sin control de flujo avanzado)

Los sockers son el mecanismo de programacion que permite a las aplicaciones en distintos hosts (o en el mismo host, usando otras familias de direcciones) Intercambiar datos haciendo uso de estos protocolos de transporte.

En el caso de sockets bloqueantes, las llamadas de lectura (por ejemplo, `recv()`, `read()`) con escritorua por ejemplo `send()` write() se ejecutan de forma que la funcion ejecutada no regresa (no completa) hasta que la operacion haya terminado:

Cuando se realiza una peticion en un socket bloqueante, la ejecucion queda suspendida hasta que llega la cantidad de datos solicitada (o hasta que el protocolo indique que hay datos disponibles para leer).

En las conexiones orientadas a conexion (tcp), esto significa que el receive se desbloquea cuando llegan datos o cuando se cierra la conexion de forma ordenada.

En sockets de tipo datagrama (udp), la llamada a lectura se bloquea hasta que se recibe algun datagrama.

1.2 Modelo cliente - servidor

Este modelo describe como se relacionan las aplicaciones en una red:

1) Servidor
    - Suele esperar (listen) en una direccion o puerto concretos
    - Acepta conexiones (en el caso de tcp) o recibe datagramas (en el caso de UDP)
    - Generalmente, el servidor se mantiene en un bucle bloqueante:
        1) Para TCP: Primero llama al listen() y luego `accept()` y se queda bloqueado hasta que un cliente inicia la conexion.

2) Para UDP: Suele esperar mediante `recvfrom()` la llegada de datagramas de un puerto especifico.

2) Cliente

- Inicia la comunicacion con el servidor especificando su direccion y puerto.
- En TCP, se realiza una llamada `connect()` que establece la conexion.
- En UDP, no existe una "conexion" en el mismo sentido, pero se puede usar `connect()` logicamente para asociar la direccion de destino; o simplemente enviar datagramas con `sendto()`.

Con sockets bloqueantes, ambas partes (cliente y servidor) dependen de que las llamadas bloqueantes se completen. Por ejemplo:

- El servidor se bloquea esperando una nueva conexion con `accept()`
- El cliente se bloquea en la llamada a `connect()` hasta que la conexion es establecida o falla.
- Una vez establecida la conexion (TCP), ambas partes pueden usar `recv()` o `send()` de forma bloqueante.

Si uno de los extremos no envia datos o no responde, la funcion bloqueante permanece esperando (a menos que ocurra un timeout o una senal de error)

1.3 Conexiones en el dominio de internet

Cuando se habla de dominio de internet (Tambien llamado familia de direcciones AF_INET o AF_INET6 para ipv6), se utilizan direcciones IP y numeros de puerto para identificar:

- Direccion IP (por ejemplo: 190.168.0.10 en IPv4 o fe80::d4a8:64ff:fe8c:20e3 en IPv6)
- Puerto (un entero de 16 bits, por ejemplo, 80 para HTTP)

Este dominio se distingue principalmente dos tipos de sockets de transporte: TCP y UDP.

1.3.1 TCP (Transmition Control Protocol)

- Orientado a conexion: Se establece primero una conexion (conocido como handshake de tres pasos) antes de intercambiar datos
- Garantiza la entrega de los datos o notifica de forma fehaciente si no es posible entregarlos
- Stream device: No hay delimitacion de mensajes dentro del flujo, el receptor debe distinguir la frontera de la informacion segun el protocolo o la logica de aplicacion.
- Sockets bloqueantes en tcp:
  - El lado servidor se bloquea en `accept()` esperando conexiones

  - El lado cliente se bloquea en `connect()` hasta que establece la conexion.
  - Las operaciones de `recv()` se bloquean hasta que haya datos disponibles; `send()` puede bloquearse si el boofer del sistema esta lleno o la red esta saturada.

1.3.2 UDP (User Datagram Protocol)

- No orientado a conexion: No requiere establecer una conexion previa; se envian y reciben datagramas de forma independiente.
- No confiable: no hay garantia de entrega ni control de flujo. Los paquetes pueden llegar desordenados, duplicados o perderse.
- Mensajes/datagramas: cada `sendto()` o `recvfrom()` manejan un datagrama completo.
- Sockets bloqueantes en UDP:
  - El servidor simplemente llama a vint() para asociar un puerto y luego esperar datagramas (generalmente con `recvfrom()`)
  - El cliente envia datagramas con `sendto()`
  - Tanto `recvfrom()` como `sendto()` pueden bloquearse si el sistema esta gestionando la operacion. Sin embargo, en la practica, `sendto()` raramente se bloquea mucho tiempo, mientras que `recvfrom()` si esperara hasta que llegue un datagrama o se produzca un error o timeout.