Resumen unidad III

Cuestiones de Java

Receptor (sockets)

Java proporciona una clase, *InetAddress*, que representa las direcciones Internet. Los usuarios de esta clase se refieren a los computadores por sus nombres de host en el servicio DNS.

API UDP

Java proporciona dos clases: DatagramPacket y DatagramSocket.

DatagramPacket: esta clase proporciona un constructor que crea una instancia compuesta por: `una cadena de bytes que almacena el mensaje, la longitud del mensaje la dirección Internet, el número de puerto local`

Las instancias de DatagramPacket podrán ser transmitidas entre procesos cuando uno las envía, y el otro las recibe.

La clase DatagramSocket proporciona varios métodos que incluyen los siguientes:

- *send y receive*: estos métodos sirven para transmitir datagramas entre un par de conectores. Ambos pueden lanzar una excepción IOException.
- *setSoTimeout*: permite establecer un tiempo de espera límite. Cuando se fija un límite, el método receive se bloquea durante el tiempo fijado y después lanza una excepción `InterruptedIOException`.
  • *connect*: utilizado para conectarse a un puerto remoto y dirección Internet concretos, en cuyo caso el conector sólo podrá enviar y recibir mensajes de esa dirección.

API TCP

La interfaz Java para los streams TCP está consituida por las clases: ServerSocket y Socket.

- *ServerSocket*: está diseñada para crear un conector en el puerto de servidor que escucha las peticiones de conexión de los clientes. Su método `accept` toma una petición connect de la cola, o si la cola está vacía, se bloquea hasta que llega una petición. El resultado de ejecutar accept es una instancia de Socket, un conector que da acceso a streams para comunicarse con el cliente.
- *Socket*: el cliente utiliza un constructor para crear un conector, especificando el nombre DNS de host y el puerto del servidor. Este constructor no sólo crea el conector asociado con el puerto local, sino que también se conecta con el computador remoto especificado en el puerto indicado.

Comunicacion entre procesos

Caracteristicas

Se basa en dos operaciones: y envía y recibe. Implica la comunicación de datos desde el proceso emisor al receptor y puede implicar además la sincronización de los dos procesos.

A cada destino de mensajes se asocia una cola. Procesos emisores producen mensajes que serán añadidos a las colas remotas. Procesos receptores eliminarán mensajes de las colas locales.

La comunicación entre los procesos emisor y receptor puede ser:

Síncrona: los procesos receptor y emisor se sincronizan con cada mensaje. En este caso, tanto envía como recibe son operaciones bloqueantes (el emisor espera la respuesta y el receptor se bloquea hasta que llega el mensaje).

Asíncrona: la operación ENVÍA es NO BLOQUEANTE, emisor puede continuar tan pronto como el mensaje haya sido copiado al búfer local. La transmisión del mensaje se lleva a cabo en PARALELO. La operacion RECIBE puede tener variantes bloqueantes y no bloqueantes.

En la variante no bloqueante el proceso receptor sigue con su programa después de invocar la operación recibe, la cual proporciona un BUFER que será llenado en un segundo plano, pero el proceso DEBE SER INFORMADO por separado de que su búfer ha sido llenado, ya sea por el método de encuesta o mediante una interrupción.

En un entorno que soportan múltiples hilos, el recibe bloqueante tiene pocas desventajas, ya que puede ser invocado por un hilo.

La comunicación no bloqueante parece ser más eficiente, pero implica una complejidad extra. Por estas razones, los sistemas actuales no proporcionan la forma no bloqueante de recibe.

En los protocolos Internet, los mensajes son enviados a direcciones construidas (Direccion_Internet, Puerto_local).

`Puerto local` es el destino de un mensaje dentro de un computador, especificado como un nro. entero.

- Los procesos pueden utilizar múltiples puertos para recibir mensajes.
- Generalmente, los servidores hacen públicos sus números de puerto para que sean utilizados por los clientes.
- Si el cliente utiliza una dirección Internet fija para referirse a un servicio, entonces ese servicio debe ejecutarse siempre en el mismo computador para que la dirección se considere válida..

Sockets

UDP y TCP utilizan la abstracción de sockets para la comunicación entre procesos. La comunicación entre procesos consiste en la transmisión de un mensaje entre un conector de un proceso y un conector de otro proceso.

Cada socket está asociado con un protocolo particular, ya sea UDP o TCP.

Los procesos pueden usar el mismo socket para enviar y recibir mensajes. Cada computadora tiene (2^16) de números de puertos posibles para ser usados para recibir mensajes.

Cualquier proceso puede hacer uso de múltiples puertos para recibir mensajes, pero un proceso no puede compartir puertos con otros procesos en la misma computadora (excepto los que usan IP multicast).

Cualquier número de procesos puede enviar mensajes al mismo puerto.

Receptor

En procesos receptores, su conector debe estar asociado a un
puerto local y a una de las direcciones del computador.

Los mensajes enviados a una dirección de Internet y a un puerto concretos, sólo pueden ser recibidos por el proceso asociado con esa dirección y con ese puerto.

Datagramas UDP

Un datagrama se envia sin confirmacion de recepcion o reintentos. Si algo falla, el mensaje puede no llegar a su destino. Cualquier proceso que necesite enviar o recibir mensajes debe crear, primero, un conector asociado a una dirección Internet y a un puerto local.

- Un servidor enlazará su conector a un puerto de servidor.
- Un cliente ligará su conector a cualquier puerto local libre.
- El método Recibe devolverá, M:mensaje, D:dirección emisor; P:puerto emisor, permitiendo al receptor enviar la correspondiente respuesta.

Aspectos referentes:

- *Tamaño del mensaje*: el receptor debe especificar una cadena de bytes de un tamaño concreto para el mensaje recibido. Si el mensaje es demasiado grande para dicha cadena, será truncado a la llegada. La capa subyacente IP permite paquetes de hasta 216 bytes, incluyendo tanto las cabeceras como el mensaje. La mayoría de los entornos imponen una restricción a su talla de 8 kilobytes.
- *Bloqueo*: comunicación de datagramas UDP utiliza operaciones de envío, envía, no bloqueantes y recepciones, recibe, bloqueantes.
- *Puerto origen*
- *Puerto destino*
- *Tiempo limite de espera*
- *Recibe de cualquiera*: el método recibe no especifica el origen de los mensajes.

Modelo de fallos para UDP

UDP padece de las siguientes debilidades:

- *Fallos de omisión*: los mensajes pueden desecharse ocasionalmente.
- *Ordenación*: algunas veces, los mensajes se entregan en desorden.
- Las aplicaciones que utilizan datagramas UDP dependen de sus propias comprobaciones para conseguir la calidad y confiabilidad en la comunicación. Puede construirse un servicio de entrega fiable mediante la utilización de acuses de recibo.

Para algunas aplicaciones resulta aceptable utilizar un servicio que sea susceptible de sufrir fallos de omisión ocasionales. Por ejemplo el servicio DNS está implementado sobre UDP.

UDP no padece sobrecargas asociadas a la entrega de mensajes garantizada. Existen tres fuentes principales para esta sobrecarga:

1. La necesidad de almacenar información de estado en el origen y en el destino.
2. La transmisión de mensajes extra.
3. La latencia para el emisor.

Streams TCP

La API originaria de UNIX BSD 4.x, proporciona la abstracción de un flujo de bytes (stream) en el que pueden escribirse y desde el que pueden leerse datos.

La abstracción **oculta**:

- *Tamaño de los mensajes*: la aplicación puede elegir la cantidad de datos que quiere escribir o leer del stream. En el destino, los datos son proporcionados a la aplicación según los va solicitando (las aplicaciones pueden forzar le envio inmediato de datos).
- *Mensajes perdidos*: TCP implementa un esquema de confirmacion de mensajes, por lo que si el emisor no recibe dicha confirmacion en un tiempo fijado, volverá a transmitir el mensaje.
- *Control de flujo*.
- *Duplicacion y ordenamiento de mensajes*: a cada paquete se le asocia un id, lo que posibilita rechazar duplicados y reordenar los mensajes.
- *Destinos de los mensajes*: se establece una conexión antes de la comunicacion. Una establecida la comunicación. La conexión implica una petición `connect`, desde el cliente al servidor, seguida de un `accept`, desde el servidor al cliente antes de que cualquier comunicación pueda tener lugar.
- *Concordancia de datos*: los dos procesos que se comunican necesitan estar de acuerdo en el tipo de datos transmitidos por el stream.
- *Bloqueo*: los datos escritos en un stream se almacenan en un búfer en el conector destino. Cuando un proceso quiere leer, extrae los datos de la cola o se bloquea hasta que existan datos disponibles. El proceso que escribe los datos en el stream resultará bloqueado por el mecanismo de control de stream de TCP si el conector del otro lado intenta almacenar en la cola de entrada más información de la permitida.
  • *Hilos*: cuando un servidor acepta una conexión, generalmente crea un nuevo hilo con el que comunicarse con el nuevo cliente.

Modelo de fallos TCP

Se tienen garantizo la entrega incluso cuando alguno de los paquetes subyacentes se haya perdido.

Los streams TCP **utilizan**: Checksum para detectar y rechazar los paquetes corruptos y número de secuencia para detectar y eliminar paquetes duplicados.

Con respecto a la propiedad de **validez** utilizan: timeouts y retransmisión de los paquetes perdidos.

Conexion Rota: Si la pérdida de paquetes sobrepasa un cierto límite, o la red que conecta un par de procesos en comunicación está severamente congestionada, el software TCP no recibirá confirmaciones de los paquetes enviados y declarará rota la conexión.

Cuando una conexión está rota, se notificará al proceso que la utiliza siempre que intente leer o escribir. Esto tiene los siguientes *efectos*:

- Los procesos que utilizan la conexión no distinguen entre un fallo en la red y un fallo en el proceso que está en el otro extremo de la conexión.
- Los procesos comunicantes no pueden saber si sus mensajes recientes han sido recibidos o no.

Utilizacion TCP

Muchos de los servicios utilizados se ejecutan sobre conexiones TCP, con números de puerto reservados. Entre ellos se encuentran los siguientes:

- HTTP / HTTPS: El protocolo de transferencia de hipertexto.
- FTP: El protocolo de transferencia de archivos permite leer los directorios de un computador remoto y transferir archivos entre los computadores de una conexión.
- Telnet: terminal remota.
- SMTP: El protocolo simple de transferencia de correo.

Representacion externa de datos y empaquetado

La informacion en los programas se representa como estructuras de datos mientras que la información en los mensajes consiste en secuencias de bytes. Independientemente de la forma de comunicación utilizado, las estructuras de datos deben ser aplanadas (convertido a una secuencia de bytes) antes de la transmisión y reconstruido en destino.

Las datos primitivos en la transmisión de los mensajes pueden tener valores de diferentes tipos, y no todos los computadores almacenan valores, como enteros, en el mismo orden. La representación de números en coma flotante también difiere entre las arquitecturas.

Para resolver esto se pueden utilizar dos metodos:

- Los valores se convierten a un formato convenido externo antes de la transmisión y convertido a la forma local en recepción. Si los dos ordenadores conocen el mismo tipo, la conversión a formato externo puede omitirse.
- Los valores se transmiten en formato del remitente, junto con una indicación del formato utilizado, y el receptor convierte los valores si es necesario.

Algunoas alternativas para la representación externa de datos y empaquetado:

- Representación común de datos CORBA.
- Serialización de objetos Java, (representación "aplanada" y externa de datos, para uso exclusivo de Java.).
- XML (Extensible Markup Language)
- Protocol Buffers (Google):
- JSON (JavaScript Object Notation)

Comunicacion en grupo

Multidifusión IP: se construye sobre el protocolo Internet (IP). La multidifusión IP permite que el emisor transmita un único paquete IP a un conjunto de computadores que forman un grupo de multidifusión. Los grupos de multidifusión se especifican utilizando direcciones de clase D (primeros cuatro bits son 1110 en IPv4). La pertenencia a los grupos de multidifusión es dinámica.

Los siguientes detalles son específicos de IPv4:

- *Routers multidifusión*: los paquetes IP pueden multidifundirse tanto en la red local como toda Internet. Para limitar la distancia de propagación de un datagrama de multidifusión, el emisor puede especificar el número de routers que puede cruzar; llamado tiempo de vida (time to TTL).
- *Reserva de direcciones de multidifusión*: las direcciones de multidifusión se pueden reservar deforma temporal o permanentemente. Existen grupos permanentes y sus direcciones son asignadas arbitrariamente por la autoridad de Internet en el rango 224.0.0.1 a 224.0.0.255.

Virtualizacion de red

Se ocupa de la construcción de redes virtuales diferentes sobre una red existente tal como el Internet. Cada red virtual puede ser diseñada para soportar una aplicación distribuida en particular.

```
"Una red virtual específica de aplicación puede ser construida sobre una red
existente y optimizada para esa aplicación particular, sin cambiar las características
de la Red subyacente."
```

Overlay networks

Una red de superposición es una red virtual que consta de nodos y enlaces virtuales, que se encuentran encima de una red subyacente (como una red IP) y ofrece algo que no se proporciona de otra manera:

- Un servicio adaptado a las necesidades de una clase de aplicación o de un servicio particular.
- Funcionamiento más eficiente en un entorno de red dado (ej: encaminamiento en una red ad hoc).
- Una característica adicional (ej: comunicación multicast o segura).
- De manera similar, cada red virtual tiene su propio esquema de direccionamiento particular, protocolos y algoritmos de enrutamiento, pero redefinido para satisfacer las necesidades particulares de las diferentes clases de aplicaciones.