# Redes

1. **¿Qué es un puerto?**

Un puerto proporciona una interfaz física específica entre los dispositivos. También se refiere a un punto de conexión entre redes en forma de nodo en una red en la que los paquetes de datos se dirigen hacia la dirección de destino final. Existen dos tipos de puertos: Físico y Virtual.

Virtual: Las suites de protocolos de Internet como UDP o TCP se basan en puertos virtuales como puntos de conexión para el intercambio de información y la transmisión de datos. Los datos viajan desde un puerto en el dispositivo inicial y se dirigen hacia el extremo receptor de la línea.

Físico: Un puerto físico permitía la conexión de componentes de hardware internos o dispositivos externos a la unidad de procesamiento principal del ordenador.

1. **¿Como están formados los endpoints?**

Permite a los administradores de los servicios de seguridad de los endpoints configurar politicar de proteccion, por ejemplo, el acceso a determinados sitios web que podrian contener algun virus o software malicioso.

1. **¿Qué es un socket?**

Un socket, es un método para la comunicación entre un programa del cliente y un programa del servidor en una red, se define, por tanto, como el punto final en una conexión.

1. **¿A qué capa del modelo TPC/IP pertenecen los sockets? ¿Por qué?**

En la actualidad existen varios tipos de socket y cada uno por lo regular se asocia a un tipo de protocolo, por ejemplo:

SOCK\_STREAM: está asociado al protocolo TCP, este brinda seguridad en la transmisión de datos, seguridad en la recepción, en la integridad y en la secuencia, entre otros.

SOCK\_DGRAM: está asociado al protocolo UDP, e indica que los paquetes viajarán en tipo datagramas, el cual tiene una comunicación asíncrona.

Los sockets permiten implementar una arquitectura cliente-servidor o peer to peer.

1. **¿Cómo funciona el modelo cliente-servidor con TCP/IP Sockets?**

El modelo cliente/servidor funciona cuando los usarios invocan la parte cliente de la aplicación,que construye una solicitud para ese servicio y se la envia al servidor de la aplicación que usa TCP/IP como transporte. El servidor recibe una solicitud, reaqliza el servicio requerido y devuelve los resultados en forma de una respuesta.

1. **¿Cuales son las causas comunes por la que la conexión entre cliente/servidor falle?**

Las causas comunes por la que la conexión entre cliente/servidor falle son: Error 2200: El cliente no recibio una respuesta del servidor en un tiempo determinado, esto sucede solo si un limite de tiempo fue especificado. Errror 2300: El servidor cerro la conexcion. Error 2310: El servidor se ha caido mientras intentaba procesar la conexión con el cliente. La conexión se cerro. Error 2315: El servidor recibio una conexión con el cliente y devolvio una respuesta del tipo non-IIOP que el cliente no puee procesar.

1. **Diferencias entre sockets UDP y TCP**

La principal diferencia entre ambos es que el UDP necesita que le entregemos paquetes de datos que el usuario debe construir, mientras el TCP admite bloques de datos (cuyo tamaño puede ir desde 1 bytes hasta muchos K bytes, dependiendo de la implementación) que serán empaquetados de forma transparente antes de ser transmitidos.

Existe además otra diferencia importante. Tanto los paquetes de datos UDP como los segmentos TCP (este es el nombre que reciben los paquetes TCP) pueden perderse (muy rara vez llegan al destino correcto con errores). Si un paquete se pierde el UDP no hace nada. Por el contrario, si un segmento se pierde el TCP lo retransmitirá, y este proceso durará hasta que el segmento ha sido correctamente entregado al host receptor, o se produzca un número máximo de retransmisiones.

Finalmente, en aplicaciones en tiempo real es necesario también tener en cuenta una cosa. En el UDP controlamos qué datos viajan en cada paquete. En el TCP esto no es posible porque el *empaquetamiento*es automático. De hecho, el TCP espera un tiempo prudencial a tener bastantes datos que transmitir antes de enviar un segmento ya que esto ahorra ancho de banda. Si es importante que los datos tarden el mínimo tiempo posible en llegar al receptor el UDP es la mejor opción. En este sentido se dice que el UDP tiene una menor latencia que el TCP.

1. **Diferencia entre synchoronous and asynchronous sockets?**

Uno de los primeros problemas que encontrará al desarrollar su aplicación es la diferencia entre conexiones síncronas (bloqueantes) y asíncronas (no bloqueantes). Siempre que realice alguna operación en un socket, es posible que no pueda completarse inmediatamente y devolver el control a su programa. Por ejemplo, una lectura en un socket no puede completarse hasta que el servidor haya enviado algunos datos.Si no hay datos esperando ser leídos, puede suceder una de dos cosas: la función puede esperar hasta que se hayan escrito algunos datos en el socket, o puede regresar inmediatamente con un error que indica que no hay datos para leer.

En el Socket síncrono o de bloqueo. En otras palabras, el programa se "bloquea" hasta que se satisfaga la solicitud de datos. Cuando el servidor escribe algunos datos en el socket, la operación de lectura se completará y se reanudará la ejecución del programa. En cambio se denomina socket asíncrono o sin bloqueo y requiere que la aplicación reconozca la condición de error y maneje la situación de manera apropiada.

Lerner Andrés – Roldan Nicolas.