INDICE

1.- Introducción.

Una red de ordenadores o red informática la podemos definir como un sistema de comunicaciones que conecta ordenadores y otros equipos informáticos entre sí, con la finalidad de compartir información y recursos.

Mediante la compartición de información y recursos en una red, los usuarios de la misma pueden hacer un mejor uso de ellos y, por tanto, mejorar el rendimiento global del sistema u organización.

Las principales ventajas de utilizar una red de ordenadores las podemos resumir en las siguientes:

Reducción en el presupuesto para software y hardware.

Posibilidad de organizar grupos de trabajo.

Mejoras en la administración de los equipos y las aplicaciones.

Mejoras en la integridad de los datos.

Mayor seguridad para acceder a la información.

Mayor facilidad de comunicación.

Pues bien, los servicios en red son responsables directos de estas ventajas.

Un servicio en red es un software o programa que proporciona una determinada funcionalidad o utilidad al sistema. Por lo general, estos programas están basados en un conjunto de protocolos y estándares Una clasificación de los servicios en red atendiendo a su finalidad o propósito puede ser la siguiente:

Administración/Configuración. Esta clase de servicios facilita la administración y gestión de las configuraciones de los distintos equipos de la red, por ejemplo: los servicios DHCP y DNS. Acceso y control remoto. Los servicios de acceso y control remoto, se encargan de permitir la conexión de usuarios a la red desde lugares remotos, verificando su identidad y controlando su acceso, por ejemplo Telnet y SSH.

De Ficheros. Los servicios de ficheros consisten en ofrecer a la red grandes capacidades de almacenamiento para descongestionar o eliminar los discos de las estaciones de trabajo, permitiendo tanto el almacenamiento como la transferencia de ficheros, por ejemplo FTP.

Impresión. Permite compartir de forma remota impresoras de alta calidad, capacidad y coste entre múltiples usuarios, reduciendo así el gasto.

Información. Los servicios de información pueden servir ficheros en función de sus contenidos, como pueden ser los documentos de hipertexto, por ejemplo HTTP, o bien, pueden servir información para ser procesada por las aplicaciones, como es el caso de los servidores de bases de datos.

Comunicación. Permiten la comunicación entre los usuarios a través de mensajes escritos, por ejemplo email o correo electrónico mediante el protocolo SMTP.

A veces, un servicio toma como nombre, el nombre del protocolo del nivel de aplicación en el que está basado. Por ejemplo, hablamos de servicio FTP, por basarse en el protocolo FTP.

En esta unidad, verás ejemplos de cómo programar en Java algunos de estos servicios, pero antes vamos a ver qué son los protocolos del nivel de aplicación.

2.- Protocolos de comunicaciones del nivel de aplicación.

En la actualidad, el conjunto de protocolos TCP/IP constituyen el modelo más importante sobre el que se basa la comunicación de aplicaciones en red. Esto es debido, no sólo al espectacular desarrollo que ha tenido Internet, en los últimos años (recuerda que TCP/IP es el protocolo que utiliza Internet), sino porque también, TCP/IP ha ido cobrando cada vez más protagonismo en las redes locales y corporativas.

Dentro de la jerarquía de protocolos TCP/IP la capa de Aplicación ocupa el nivel superior y es precisamente la que incluye los protocolos de alto nivel relacionados con los servicios en red que te hemos indicado antes.

La capa de Aplicación define los protocolos (normas y reglas) que utilizan las aplicaciones para intercambiar datos. En realidad, hay tantos protocolos de nivel de aplicación como aplicaciones distintas; y además, continuamente se desarrollan nuevas aplicaciones, por lo que el número de protocolos y servicios sigue creciendo.

En la siguiente imagen puedes ver un esquema de la familia de protocolos TCP/IP y su organización en capas o niveles.

Pila de protocolos TCP/IP.

A continuación, vamos a destacar, por su importancia y gran uso, algunos de los protocolos estándar del nivel de aplicación:

DNS. Protocolo para traducir direcciones de red.

FTP. Protocolo para la transferencia de ficheros.

SMTP. Protocolo que permite transferir correo electrónico. Recurre al protocolo de oficina postal POP para almacenar mensajes en los servidores, en sus versiones POP2 (dependiente de SMTP para el envío de mensajes) y POP3 (independiente de SMTP).

HTTP. Protocolo de transferencia de hipertexto.

Telnet. Protocolo que permite acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora remota mediante un intérprete de comandos.

SSH. Protocolo que permite gestionar remotamente a otra compuatora de la red de forma segura.

NNTP. Protocolo de Transferencia de Noticias (en inglés Network News Transfer Protocol).

IRC. Chat Basado en Internet (en inglés Internet Relay Chat)

2.1.- DNS y resolución de nombres.

Todas las computadoras y dispositivos conectados a una red TCP/IP se identifican mediante una dirección IP, como por ejemplo 117.142.234.125.

En su forma actual (IPv4), una dirección IP se compone de cuatro bytes sin signo (de 0 a 254) separados por puntos para que resulte más legible, tal y como has visto en el ejemplo anterior. Por supuesto, se trata de valores ideales para los ordenadores, pero para los seres humanos que quieran acordarse de la IP de un nodo en concreto de la red, son todo un problema de memoria.

El sistema DNS o Sistema de Nombres de Dominio es el mecanismo que se inventó, para que los nodos que ofrecen algún tipo de servicio interesante tengan un nombre fácil de recordar, lo que se denomina un nombre de dominio, como por ejemplo www.todofp.es.

DNS es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada.

El objetivo principal de los nombres de dominio y del servicio DNS, es traducir o resolver nombres (por ejemplo www.dominio.es) en las direcciones IP (identificador binario) de cada equipo conectado a la red, con el propósito de poder localizar y direccionar estos equipos dentro de la red.

Sin la ayuda del servicio DNS, los usuarios de una red TCP/IP tendrían que acceder a cada servicio de la misma utilizando la dirección IP del nodo.

Además el servicio DNS proporciona otras ventajas:

Permite que una misma dirección IP sea compartida por varios dominios.

Permite que un mismo dominio se corresponda con diferentes direcciones IP.

Permite que cualquier servicio de red pueda cambiar de nodo, y por tanto de IP, sin cambiar de nombre de dominio.

2.2.- El protocolo FTP.

El protocolo FTP o Protocolo de Transferencia de Archivos proporciona un mecanismo estándar de transferencia de archivos entre sistemas a través de redes TCP/IP.

Las principales prestaciones de un servicio basado en el protocolo FTP o servicio FTP son las siguientes:

Permite el intercambio de archivos entre máquinas remotas a través de la red.

Consigue una conexión y una transferencia de datos muy rápidas.

Sin embargo, el servicio FTP adolece de una importante deficiencia en cuanto a seguridad:

La información y contraseñas se transmiten en texto plano. Esto está diseñado así, para conseguir una mayor velocidad de transferencia. Al realizar la transmisión en texto plano, un posible atacante puede capturar este tráfico, acceder al servidor y/o apropiarse de los archivos transferidos.

Este problema de seguridad, se puede solucionar mediante la encriptación de todo el tráfico de información, a través del protocolo no estándar SFTP usando SSH o del protocolo FTPS usando SSL. De encriptación ya hablaremos más adelante, en otra unidad de este módulo.

¿Cómo funciona el servicio FTP? Es una servicio basado en una arquitectura cliente/servidor y, como tal, seguirá las pautas generales de funcionamiento de este modelo, en ese caso:

El servidor proporciona su servicio a través de dos puertos:

El puerto 20, para transferencia de datos.

El puerto 21, para transferencia de órdenes de control, como conexión y desconexión.

El cliente se conecta al servidor haciendo uso de un puerto local mayor de 1024 y tomando como puerto destino del servidor el 21.

Las principales características del servicio FTP son las siguientes:

La conexión de un usuario remoto al servidor FTP puede hacerse como: usuario del sistema (debe tener una cuenta de acceso), usuario genérico (utiliza la cuenta anonymous, esto es, usuario anónimo), usuario virtual (no requiere una cuenta local del sistema.

El acceso al sistema de archivos es más o menos limitado, según el tipo de usuario qe se conecte y sus privilegios. Por ejemplo, el usuario anónimo solo tendrá acceso al directorio público que establece el administrador por defecto.

El servicio FTP soporta dos modos de conexión: modo activo y modo pasivo.

Modo activo. Es la forma nativa de funcionamiento (esquema ampliable de la derecha).

Se establecen dos conexiones distintas: la petición de transferencia por parte del cliente y la atención a dicha petición, iniciada por el servidor. De manera que, si el cliente está protegido con un cortafuegos, deberá configurarlo para que permita esta petición de conexión entrante a través de un puerto que, normalmente, está cerrado por motivos de seguridad.

Modo pasivo. El cliente sigue iniciando la conexión, pero el problema del cortafuegos se traslada al servidor.

2.3.- Los protocolos SMTP y POP3.

El correo electrónico es un servicio que permite a los usuarios enviar y recibir mensajes y archivos rápidamente a través de la red.

Principalmente se usa este nombre para denominar al sistema que provee este servicio en Internet, mediante el protocolo SMTP o Protocolo Simple de Transferencia de Correo, aunque por extensión también puede verse aplicado a sistemas análogos que usen otras tecnologías.

Por medio de mensajes de correo electrónico se puede enviar, no solamente texto, sino todo tipo de documentos digitales.

El servicio de correo basado en el protocolo SMTP sigue el modelo cliente/servidor, por lo que el trabajo se distribuye entre el programa Servidor y el Cliente. Te indicamos a continuación, algunas consideraciones importantes sobre el servicio de correo a través de SMTP:

El servidor mantiene las cuentas de los usuarios así como los buzones correspondientes.

Los clientes de correo gestionan la descarga de mensajes así como su elaboración.

El servicio SMTP utiliza el puerto 25.

El protocolo SMTP se encarga del transporte del correo saliente desde la máquina del usuario remitente hasta el servidor que almacene los mensajes de los destinatarios.

El usuario remitente redacta su mensaje y lo envía hacia su servidor de correo.

Desde allí se reenvía al servidor del destinatario, quién lo descarga del buzón en la máquina local mediante el protocolo POP3, o la consulta vía web, haciendo uso del protocolo IMAP.

2.4.- El protocolo HTTP.

El protocolo HTTP o Protocolo de Transferencia de Hipertexto es un conjunto de normas que posibilitan la comunicación entre servidor y cliente, permitiendo la transmisión de información entre ambos. La información transferida son las conocidas páginas HTML o páginas web. Se trata del método más común de intercambio de información en la World Wide Web.

HTTP define tanto la sintaxis como la semántica que utilizan clientes y servidores para comunicarse. Algunas consideraciones importantes sobre HTTP son las siguientes:

Es un protocolo que sigue el esquema petición-respuesta entre un cliente y un servidor.

Utiliza por defecto el puerto 80

Al cliente que efectúa la petición, por ejemplo un navegador web, se le conoce como agente del usuario (user agent).

A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL).

Por ejemplo: http://www.iesalandalus.org/organizacion.htm

Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

El funcionamiento esquemático del protocolo HTTP es el siguiente:

El usuario especifica en el cliente web la dirección del recurso a localizar con el siguiente formato: http://dirección[:puerto][ruta], por ejemplo: http://www.iesalandalus.org/organizacion.htm

El cliente web descompone la información de la URL diferenciando el protocolo de acceso, la IP o nombre de dominio del servidor, el puerto y otros parámetros si los hubiera.

El cliente web establece una conexión al servidor y solicita el recurso web mediante un mensaje al servidor, encabezado por un método, por ejemplo GET /organizacion.htm HTTP/1.1 y otras líneas.

El servidor contesta con un mensaje encabezado con la línea HTTP/1.1 200 OK, si existe la página y la envía, o bien envía un código de error.

El cliente web interpreta el código HTML recibido.

Se cierra la conexión.

En el siguiente enlace dispones de un ejemplo de diálogo entre cliente y servidor utilizando el protocolo HTTP. Más adelante, estudiaremos algunos detalles más de este protocolo para poder programar un servicio web básico.

Ejemplo de diálogo entre cliente y servidor usando el protocolo HTTP.

HTTP es un protocolo sin estado, lo que significa que no recuerda nada relativo a conexiones anteriores a la actual. Algunas aplicaciones necesitan que se guarde el estado y para ello hacen uso de lo que se conoce como cookie.

2.5.- Telnet

Mucho antes de que existieran las computadoras de escritorio con interfaces gráficas sofisticadas, las personas utilizaban sistemas basados en textos que eran simplemente terminales conectadas físicamente a una computadora central. Una vez que las redes estuvieran disponibles, las personas necesitaban acceder en forma remota a los sistemas informáticos de la misma manera en que lo

hacían con las terminales conectadas en forma directa.
Telnet se desarrolló para satisfacer esta necesidad. Telnet se remonta a principios de la década de los setenta y se encuentra entre los servicios y protocolos de capa de aplicación más antiguo dentro del grupo TCP/IP. Telnet proporciona un método estándar de emulación de dispositivos de terminal basados en texto en la red de datos. El protocolo y el software del cliente que implementa el protocolo comúnmente se definen como Telnet.
Y como consecuencia, una conexión que utiliza Telnet se llama Sesión o conexión de terminal virtual (VTY). En lugar de utilizar un dispositivo físico para conectar al servidor, Telnet utiliza software para crear un dispositivo virtual que proporciona las mismas funciones que una sesión terminal con acceso a la Interfaz de línea de comandos (CLI) del servidor.
Para admitir conexiones al cliente Telnet, el servidor ejecuta un servicio llamado daemon de Telnet. Se establece una conexión de terminal virtual desde un dispositivo final utilizando una aplicación del cliente Telnet. La mayoría de los sistemas operativos incluye un cliente de Telnet de la capa de aplicación. En una PC de Microsoft Windows, Telnet puede ejecutarse desde la entrada del comando. Otras aplicaciones de terminal comunes que ejecutan clientes de Telnet son HyperTerminal, Minicom y TeraTerm.
Una vez establecida una conexión Telnet, los usuarios pueden realizar cualquier función autorizada en el servidor, como si utilizaran una sesión de línea de comandos en el servidor mismo. Si están autorizados, pueden iniciar y detener procesos, configurar el dispositivo e inclusive cerrar el sistema.
Aunque el protocolo Telnet admite autenticación de usuario, no admite el transporte de datos encriptados. Todos los datos intercambiados durante una sesión Telnet se transportan como texto sin formato por la red. Esto significa que los datos pueden ser interceptados y entendidos fácilmente.
Si la seguridad es un problema, el protocolo Shell seguro (SSH) ofrece un método seguro y alternativo para acceder al servidor. SSH proporciona la estructura para un inicio de sesión remoto seguro y otros servicios de red seguros. Además proporciona mayor autenticación que Telnet y admite el transporte de datos de sesión utilizando cifrado. Como una mejor práctica, los profesionales de red deberían siempre utilizar SSH en lugar de Telnet, cada vez que sea posible.

2.6.- SSH

SSH (Secure SHell, en español: intérprete de órdenes segura) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos, y también puede redirigir el tráfico de X para poder ejecutar programas gráficos si tenemos un Servidor X (en sistemas Unix y Windows) corriendo.
Además de la conexión a otros dispositivos, SSH nos permite copiar datos de forma segura (tanto ficheros sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a los dispositivos y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.

3.- Programación con URL.

La programación de URL se produce a un nivel más alto que la programación de sockets y ésto, puede facilitar la creación de aplicaciones que acceden a recursos de la red.

Una URL, Localizador Uniforme de Recursos, representa una dirección a un recurso de la World Wide Web. Un recurso puede ser algo tan simple como un archivo o un directorio, o puede ser una referencia a un objeto más complicado, como una consulta a una base de datos, el resultado de la ejecución de un programa, etc.

Consideraciones sobre una URL:
Puede referirse a sitios web, ficheros, sitios ftp, direcciones de correo electrónico, etc.
La estructura de una URL se puede dividir en varias partes:

Protocolo. El protocolo que se usa para comunicar.
Nombrehost. Nombre del host que proprociona el servicio o servidor.
Puerto. El puerto de red en el servidor para conectarse. Si no se especifica, se utiliza el puerto por defecto para el protocolo.
Ruta. Es la ruta o path al recurso en el servidor.
Referencia. Es un fragmento que indica una parte específica dentro del recurso especificado.
Por ejemplo, algunas posibles direcciones URL serían las siguientes:
http://www.iesalandalus.org/organizacion.htm, el protocolo utilizado es el http, el nombre del host www.iesalandalus.org y la ruta es organización.htm, que en este caso es una página html. Al no indicar puerto, se toma el puerto por defecto para HTTP que es el 80.
http://www.iesalandalus.org:85/organizacion.htm, en este caso se está indicando como puerto el 85.
http://www.dominio.es/public/pag.hml#apartado1, en este caso se especifica como ruta /public/pag.html#apartado, por lo que una vez recuperado el archivo pag.htm se está indicando mediante la referencia #apartado1 que interesa el aparatdo1 dentro de esa página.

4.- Programación de servidores.

Entre los diferentes aspectos que se deben tener en cuenta cuando se diseña o se programa un servidor o servicio en red, vamos a resaltar los siguientes:

El servidor debe poder atender a multitud de peticiones que pueden ser concurrentes en el tiempo. Esto lo podemos conseguir mediante la programación del servidor utilizando hilos o Threads.

Es importante optimizar el tiempo de respuesta del servidor. Esto lo podemos controlar mediante la monitorización de los tiempos de proceso y transmisión del servidor.

La clase ServerSocket es la que se utiliza en Java a la hora de crear servidores. Para programar servidores o servicios basados en protocolos del nivel de aplicación, como por ejemplo el protocolo HTTP, será necesario conocer el comportamiento y funcionamiento del protocolo de aplicación en cuestión, y saber que tipo de mensajes intercambia con el cliente ante una solicitud o petición de datos.

En los siguientes apartados vamos a ver el ejemplo de cómo programar un servidor web básico, al que después le añadiremos la funcionalidad de que pueda atender de manera concurrente a varios usuarios, optimizando los recursos.

4.1.- Programación de un servidor HTTP.

Antes de lanzarnos a la programación del servidor HTTP, vamos a recordar o conocer cómo funciona este protocolo y a sentar las hipótesis de trabajo.

El servidor que vamos a programar cumple lo siguiente:

Se basa en la versión 1.1 del protocolo HTTP.
Implementará solo una parte del protocolo.
Se basa en dos tipos de mensajes: peticiones de clientes a servidores y respuestas de servidores a clientes.

Nuestro servidor solo implementará peticiones GET.

Para crear un servidor HTTP o servidor web, el esquema básico a seguir será:

Crear un socketServer asociado al puerto 80 (puerto por defecto para el protocolo HTTP).
Esperar peticiones del cliente.
Acceptar la petición del cliente.
Procesar petición (intercambio de mensajes según protocolo + transmisión de datos).
Cerrar socket del cliente.

A continuación, vamos a programar un sencillo servidor web que acepte peticiones por el puerto 8066 de un cliente que será tu propio navegador web. Según la URL que incluyas en el navegador, el servidor contestará con diferente información. Los casos que vamos a contemplar son

los siguientes:

Al poner en tu navegador http://locahost:8066, te dará la bienvenida.

Al poner en tu navegador http://localhost:8066/quijote, mostrará un párrafo de el Quijote.

Al poner en tu navegador una URL diferente a las anteriores, como por ejemplo http://localhost:8066/a, mostrara un mensaje de error.

Recuerda detener o parar el servidor, una vez lo hayas probado, antes de volver reiniciarlo.