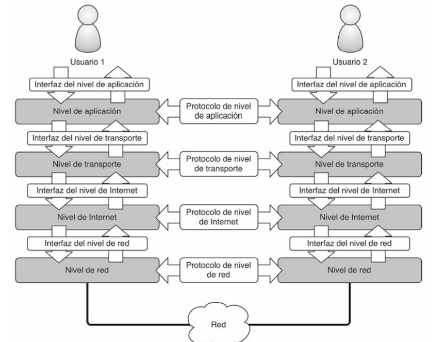
***Todos los sistemas tienen una estructura y una función:*** La estructura está formada por los componentes que forman el sistema y la función es para lo que está pensado.

***La función indica lo que hace el sistema, pero no de qué manera.*** El conjunto de mecanismos que proporciona hace posible el acceso a la función, denominada ***servicio.*** Un sistema puede proporcionar uno o más servicios, según su función. La interfaz del servicio es el punto de contacto entre el sistema y el usuario.

……………………………………………………………………………………………………………

En las comunicaciones entre elementos de un sistema distribuido se hacen uso de varios servicios distintos, que cooperan entre sí para conseguir sus objetivos. La ***pila de protocolos IP*** es un c***onjunto de sistemas independientes / capas*** que realizan entre sí una ***tarea compleja.*** Cada uno de estos ofrece un servicio específico mediante un protocolo de comunicación. Un servicio en red es cualquiera que se ubique en un nivel de dicha pila.



El nivel más alto es el de aplicaciones, estas ofrecen una interfaz de servicio y disponen de un protocolo que gestiona las comunicaciones entre ellas. (Página web, email, etc…)

Un protocolo de nivel de aplicación es el conjunto de reglas que gestionan la interacción entre los elementos de una aplicación distribuida, este es fundamental. En una aplicación cliente/servidor son sistemas independientes el cliente del servidor y cada uno cuenta con su estructura y función. El protocolo define cómo se interactúa con el cliente y es la interfaz de servicio del servidor.

El modelo cliente/servidor es el más usado en el desarrollo de aplicaciones distribuidas. Los mecanismos de transmisión de mensajes sockets son la tecnología base de este modelo. Para programar una aplicación cliente servidor se deben definir los siguientes ***aspectos:***

1. ***Funciones del servidor:*** Hay que responder una serie de preguntas: 1. ¿Cuál es la función básica del servidor?, 2. ¿El servicio que se da es rápido o lento?, 3. ¿Puede el servicio resolverse con una petición y respuesta o requiere del intercambio de múltiples mensajes?, 4. ¿Debe el servidor atender a varios clientes a la vez? - Respuestas: 1. Dar la hora, 2. Rápido, 3. Intercambio único, 4. Atiende a varios clientes.
2. ***La tecnología de comunicación*** puede ser socket stream o socket datagram según las características y requisitos especificados. El primero suele usarse para intercambiar varios mensajes mientras que el segundo es mejor para las aplicaciones sencillas.
3. ***Un protocolo de nivel de aplicación*** es el conjunto de reglas que gobiernan la interacción entre los elementos de una aplicación distribuida. En un cliente/servidor, este define el formato de los mensajes, además de las posibles secuencias en los que estos se envían y reciben o la disponibilidad de los mismos. Para ello se pregunta: 1. ¿Quién lo envía?, 2. ¿Cuándo se envía? y 3. ¿Qué contiene? - Cuando se define, debe especificarse de manera clara todas las secuencias posibles de intercambio de mensajes que pueden ocurrir en el sistema y cómo este reacciona ante ellos. La definición de este es fundamental para evitar comportamientos inesperados.

En un cliente/servidor se llama ***sesión*** a la secuencia de mensajes que se intercambian hasta el cierre de la conexión. Estos se realizan mediante peticiones y respuestas, si la interacción es sencilla no es importante controlar la sesión. Hay dos tipos de protocolos:

1. ***Protocolos sin estado:*** La secuencia de mensajes no es importante, ya que no afecta al resultado. El servidor responde las peticiones una a una.
2. ***Protocolos con estado:*** La secuencia de mensajes es importante y afecta al resultado. El servidor debe almacenar información intermedia durante la sesión, esta se conoce como estado de la sesión. Se debe conocer dicho estado para resolver las peticiones correctamente.

La mayor parte de servidores están pensados para atender a múltiples clientes a la vez. Estos funcionan de manera que cuando se está atendiendo a un cliente y se recibe una ***nueva petición:***

1. La nueva petición no debe interferir con la que está en proceso.
2. Esta debe ser atendida lo antes posible, incluso de manera simultánea a la anterior.

Para que esas dos condiciones se cumplan, la mayoría de servidores implementan una ***ejecución multihilo.*** Un servidor multihilo es aquel en el que a ***cada cliente*** se le atiende en un ***hilo de ejecución independiente.*** Cuando un cliente envía una petición, el servidor instancia un hilo exclusivo para dicha petición.

Los sockets stream son los adecuados para para estos servidores multihilo. En estos, la operación accept crea un nuevo socket en el servidor que se conecta al cliente, quedando libres el hilo principal y el socket servidor.

***Aplicaciones distribuidas de uso generalizado / Protocolos de nivel de aplicación:***

1. ***Telnet:*** Este es un protocolo diseñado para proporcionar comunicación bidireccional basada en texto plano entre dos elementos de una red. Simula una conexión virtual a una terminal de texto. Forma parte de la pila de protocolos IP y usa el protocolo de transporte TCP para intercambiar mensajes entre ambos elementos. Principalmente se ha usado para conectarse remotamente a máquinas, creando una sesión de línea de comandos como en teamViewer. El problema es que las cadenas de texto se envían de forma clara por la red, sin cifrado o protección, por ello está en desuso.
2. ***SSH / Secure Shell:*** Protocolo con estado muy parecido a Telnet, más moderno en el que la información transferida está cifrada. Es recomendado para establecer sesiones de líneas de comandos a máquinas remotas mediante internet u otras redes.
3. ***FTP / File Transfer Protocol:*** Es un protocolo diseñado para la transferencia de archivos a través de una red de comunicaciones. Al igual que los anteriores, este usa el protocolo de transferencia TCP para establecer un canal de comunicación y realizar la transferencia de datos. Este protocolo permite la descarga y subida de archivos, esto lo permite estableciendo un par de conexiones:

1. Conexión de control: Envía órdenes al servidor y obtiene información.

2. Conexión de datos: Se usa para transferir archivos.

Además permite listar y navegar un árbol de directorios y cambiar las propiedades de los archivos. Es un protocolo con estado y su puerto es el 21.

1. ***HTTP / HyperText Transfer Protocol:*** Es el más importante, ya que la mayoría del tráfico que se realiza en la WWW usa este protocolo para controlar la transferencia de información entre las aplicaciones. Su función es facilitar la transmisión de documentos entre las aplicaciones de una red. Un documento de hipertexto es un texto en un código especial que mediante hiperenlaces permiten acceder a otros documentos. Este sigue el modelo de comunicación cliente/servidor. Forma parte de la pila IP y está diseñado para usar un protocolo de transporte fiable y orientado a conexión. Puede usar TCP y UDP, su puerto es el 80. Es un protocolo sin estado. Algunos servidores HTTP usan mecanismos para almacenar el estado de la sesión, el método más usado para conseguirlo consiste en almacenar pequeños fragmentos de información en la aplicación cliente / cookies.

Se conoce como una sesión HTTP a una secuencia intercambios petición respuesta entre un cliente y un servidor. El cliente inicia una sesión mediante una conexión TCP con el puerto 80 del servidor. Las respuestas del servidor contienen información sobre el estado de la petición, además de contenido adicional como páginas web almacenadas en el cuerpo del mensaje. Un servidor HTTP organiza la información que contiene usando recursos. Un recurso puede ser cualquier documento procedente del servidor, estos tienen un URL que se usa en hiperenlaces.

***HTTP tiene varias peticiones:***

1. GET: Se usa para solicitar recursos, como páginas web.
2. HEAD: Similar a GET pero la respuesta no tiene cuerpo del mensaje, por lo que no se obtienen recursos. Únicamente se obtiene información del servidor.
3. POST: Se usa para solicitar al servidor la incorporación de nuevo contenido a un recurso existente con su URL como en páginas web.
4. PUT: Se usa para solicitar la incorporación de nuevo contenido al servidor.
5. DELETE: Se usa para solicitar la eliminación de un recurso.
6. OPTIONS: Se usa para realizar una lista de peticiones que el servidor acepta sobre una URL.
7. TRACE: Se usa para solicitar al servidor que le devuelva la petición al cliente.
8. CONNECT: Se usa para convertir la conexión entre cliente y servidor en un túnel TCP/IP; esto facilita la transmisión de datos cifrados.
9. PATCH: Se usa para realizar modificaciones parciales a un recurso.

Hay peticiones que no está obligado a aceptar como POST o DELETE en páginas web que no se puedan modificar. Al menos debe aceptar GET y HEAD, además de OPTIONS si fuera posible.

Las respuestas HTTP contienen información sobre el estado de la petición. La primera línea de texto del mensaje de respuesta se llama ***línea de estado*** y contiene un código numérico y una frase explicativa. Estos códigos normalmente tienen tres dígitos y se dividen en cinco categorías:

1. Información: Códigos que empiezan por 1 y se usan para informar al cliente en general.
2. Éxito: Códigos que empiezan por 2, se usan para indicar al cliente que su petición ha sido recibida y procesada.
3. Redirección: Códigos que empiezan por 3, indican al cliente que debe realizar operaciones adicionales para completar su petición.
4. Error del cliente: Códigos que empiezan por 4, indica al cliente que se ha producido un error.
5. Error del servidor: Códigos que empiezan por 5, indican al cliente que el servidor ha experimentado un error y no se puede completar la petición.

……………………………………………………………………………………………………………

***El protocolo de aplicación POP*** está diseñado para que las aplicaciones clientes accedan a los mensajes alojados en un servidor de correo electrónico (actualmente en su versión POP3). Este se basa en el protocolo de transporte TCP y proporciona peticiones básicas para acceso, descarga y borrado de mensajes. Este es un protocolo sin estado y su número de puerto es el 110.

***El protocolo de aplicación SMTP*** es el estándar para el envío de correo electrónico en Internet a través de POP. Este también se basa en TCP, es un protocolo sin estado y su número de puerto es 587.

***El protocolo de aplicación DHCP*** se usa para la configuración dinámica de dispositivos dentro de una red y asignación dinámica de direcciones IP.

***El protocolo de aplicación DNS*** se usa para obtener la dirección IP de una máquina mediante su nombre o URL.

***El protocolo de aplicación NTP*** está diseñado para la sincronización de relojes entre máquinas de una red.

***El protocolo de aplicación TLS*** está diseñado para incorporar características criptográficas a los protocolos de transporte como TCP.

……………………………………………………………………………………………………………

Todos estos servicios y protocolos se basan en la pila IP y usan protocolos de transporte como TCP y UDP para gestionar el envío de mensajes. Para una aplicación de alto nivel los sockets no son lo más óptimo, por lo que hay varias alternativas. Estas son una capa extra de software que se coloca entre el nivel de transporte y de aplicación. Aunque la comunicación sigue ocurriendo mediante sockets pero su uso se abstrae al programador.

***1. Invocación de métodos remotos:*** En un programa orientado a objetos la invocación de métodos es un proceso de comunicación (este se conoce por RMI). En una invocación a un método remoto existen cuatro componentes fundamentales:

1. Objeto servidor/remoto: Objeto cuyo método es invocado. Es el que recibe la petición y la procesa.
2. Objeto cliente: Objeto que invoca el método. Envía una petición al objeto servidor a través de la red, indicando el método y sus parámetros. Se recibe un mensaje del objeto servidor con el resultado.
3. Método invocado: El método y sus parámetros se convierten en un mensaje que se envía por la red al objeto servidor a modo de petición.
4. Valor del retorno: El resultado se convierte en un mensaje que se envía al cliente.

Para realizar la invocación de métodos hace falta una capa de software extra, cuya función es traducir las llamadas a métodos remotos y valores de retorno de estos a mensajes que se envían por sockets. Esta capa además añade mecanismos para crear objetos remotos y localizarlos desde otros puntos de la red. Los componentes básicos de la capa software son:

1. Stubs: Son objetos que sustituyen a los objetos remotos dentro del programa del cliente. Cuando un stub recibe la invocación de un método, construye un mensaje y lo envía por la red al servidor. Se espera a que el objeto remoto responda y luego devuelve la respuesta. Por lo que el cliente solo invoca el método del objeto stub.
2. Registro de objetos remotos: Servicio de nivel aplicación que controla todos los objetos remotos que existen en el sistema. Cuando un programa crea un objeto remoto debe anotarlo en el registro .Cuando un cliente necesite conectar con el servidor lo buscará en el registro. Si un stub quiere configurarse para conectarse a un objeto remoto, el registro le da la información necesaria.

Proceso de invocación de métodos remotos:

1. Arranque del registro de objetos remotos.
2. Creación del objeto servidor.
3. Inscripción del objeto en el registro de objetos remotos.
4. Localización del objeto remoto en el registro.
5. Invocación de métodos del objeto stub.
6. Intercambio de mensajes entre stub y objeto servidor.
7. Obtención del valor de retorno de la invocación al método remoto.

Las clases para realizar una invocación son del paquete java.rmi. Para desarrollar una de estas aplicaciones se necesita:

1. **Definición de las funciones servidor**.
2. **Selección de la tecnología de comunicaciones**.
3. **Definición del protocolo de nivel de aplicación**.
4. **Implementación de la clase del objeto servidor**
5. **Implementación de la clase del objeto cliente**.

***2. Servicios web:***

SOAP: Usa XML para el protocolo de mensajes y el contenido de estos. La descripción de interfaz de servicio de un SOAP se realiza usando el sublenguaje WDSL de XML. Estos consumen muchos recursos de red. Pueden o no tener estado.

REST: No están forzados a usar XML para representar sus mensajes e interfaz. Este es un formato más ligero y flexible.