El servidor Web. Arquitectura y funcionamiento

ÍNDICE

INTRODUCCIÓN

¿Qué es un servidor?

Un servidor es un tipo de software que suministra servicios a los usuarios o terminales que lo solicitan. Por ejemplo, en una típica arquitectura cliente-servidor, el cliente podría ser un ordenador que realiza peticiones de información a través de un programa de correo (Outlook Express por ejemplo) y, el servidor le entrega los datos en forma de correos electrónicos en respuesta a su solicitud.
Hay que destacar el hecho de que la palabra servidor identifica tanto al programa como a la máquina en la que dicho programa se ejecuta. Existe, por tanto, cierta ambigüedad en el término.

¿Y un servidor Web?

Un servidor Web es un programa que sirve datos en forma de páginas Web, hipertextos o páginas HTML (HyperText Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos.
La comunicación de estos datos entre cliente y servidor se hace por medio un protocolo*, concretamente del protocolo HTTP.
Con esto, un servidor Web se mantiene a la espera de peticiones HTTP, que son ejecutadas por un cliente HTTP; lo que solemos conocer como un navegador Web.
A modo de ejemplo: al teclear http://www.cnice.mec.es en un navegador, éste realizará una petición HTTP al servidor que tiene asociada dicha URL**. El servidor responde al cliente enviando el código HTML de la página; el navegador cuando recibe el código, lo interpreta y lo muestra en pantalla.

El cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página. El servidor se encarga de transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

***Protocolo**: conjunto de reglas que gobiernan el intercambio de datos entre entidades dentro de una red. Es el lenguaje común "que utilizan" los ordenadores para "hablar" y entenderse entre sí. Existen muchos tipos de protocolos cada uno con sus reglas bien definidas, como por ejemplo: FTP, POP3, SMTP, ICMP, etc.

Protocolo HTTP: una de las características del protocolo HTTP es que no es permanente, es decir, cada operación HTTP implica una conexión con el servidor, que es liberada al término de la misma. Por ejemplo, un documento HTML con 10 imágenes son necesarias 11 conexiones distintas (10 imágenes más la página HTML en sí).
Además, carece de estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.
A partir de la versión 1.1 del protocolo HTTP, se pueden habilitar conexiones persistentes (permiten enviar más objetos con un menor número de conexiones).

**** Realmente la petición de una página Web se realiza en dos pasos:
Primero, el navegador solicita como cliente DNS la traducción de una URL (por ejemplo http://www.mec.es) a una IP y segundo, una vez que ha recibido la traducción del servidor DNS, se **realiza la petición HTTP** al servidor que tenga la IP concreta.
Fijémonos que si ponemos la IP en vez de la dirección en el navegador, también funciona.

FUNCIONAMIENTO DE UN SERVIDOR WEB

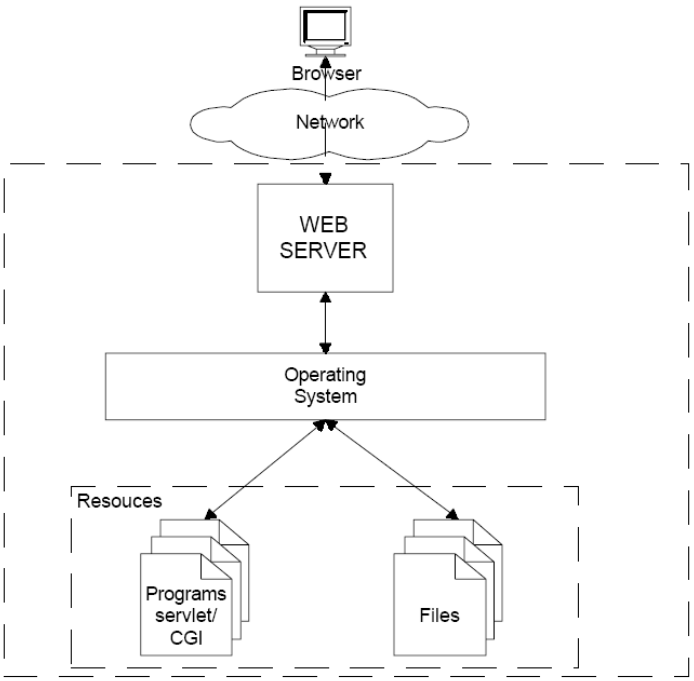

Figura 1

La figura superior muestra la interacción entre un servidor Web y el resto del entorno. El servidor es el responsable de proporcionar el acceso a los recursos solicitados que están bajo el control del sistema operativo.
Estos recursos pueden ser:

- Estáticos, como páginas HTML o texto y,
- Dinámicos, como por ejemplo CGI's. Estos programas son ejecutados por el servidor. Digamos que es la parte inteligente del servidor.

CGI: Common Gateway Interface o Pasarela de Interface Común. Es un protocolo de comunicación entre un servidor Web y una aplicación externa para ofrecer contenido dinámico a las páginas Web. Un ejemplo muy sencillo sería un típico formulario HTML que enviamos con una serie de datos. El formulario lleva incluido un campo llamado *action*, que describe la ruta a un programa en el sistema. Al pulsar el botón de "*envío de datos*", el servidor utilizando el protocolo CGI, arranca dicho programa y le pasa los datos que han llegado con el formulario. Finalmente el programa termina enviando al servidor (a través de la interface CGI) los datos solicitados, que a su vez el mismo servidor envía estos datos al cliente en forma de página HTML, en la que se informa de las tareas realizadas.

Arquitectura

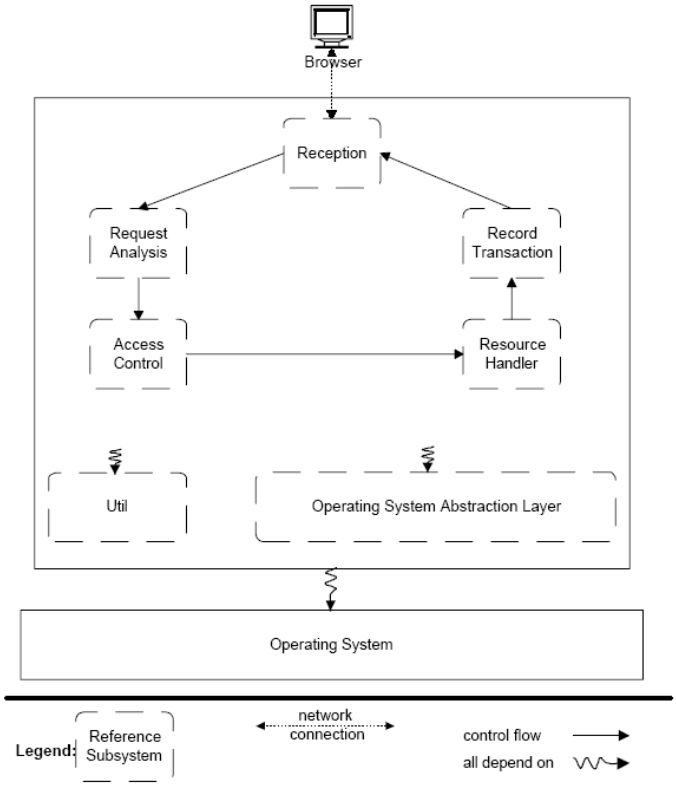

Figura 2

Vemos en el gráfico superior una arquitectura habitual de un servidor Web, dividido en dos capas:

- **Capa servidor**. Esta capa contiene cinco subsistemas, que son los responsables de implementar la funcionalidad de un servidor Web. Subsistemas:

  Subsistema de recepción: representa la primera "línea de ataque" y su labor consiste en esperar las peticiones HTTP de los clientes que llegan por la red. También, analiza las peticiones y determina las capacidades de los navegadores (tipo de navegador, compatibilidad, etc.). Este subsistema contiene la lógica necesaria para manejar múltiples peticiones.
  Analizador de peticiones: encargado de traducir la localización del recurso de la red al nombre del archivo local. Por ejemplo, la solicitud del recurso

http://www.mec.es se traduce al fichero local */var/www/webfiles/index.html*.
Control de acceso: sirve para autentificar y permitir el acceso.
Manejador de recursos: este subsistema es el responsable de determinar el tipo de recurso solicitado; lo ejecuta y genera la respuesta.
Registro de transacción: se encarga de registrar todas las peticiones y su resultado.

- **Capa soporte**. Esta capa actúa como una interface entre el sistema operativo y el servidor Web y, entre los propios subsistemas de la capa superior. Subsistemas:

  Util: contiene funciones que son utilizadas por el resto de subsistemas.
  Capa abstracta del Sistema Operativo (OSAL):este subsistema encapsula el funcionamiento específico del sistema operativo para facilitar la portabilidad del servidor Web a diferentes plataformas.

Tipos de servidores Web

Servidores basados en procesos

Este diseño es el predecesor de todos los demás. Se basa en la obtención de paralelismo mediante la duplicación del proceso de ejecución.
Existen varios diseños basados en procesos. El más simple es en el que el proceso principal espera la llegada de una nueva conexión y en ese momento, se duplica creando una copia exacta que atenderá esta conexión. Sobre esta opción de diseño caben optimizaciones importantes, como las que incluyó *Apache* con la técnica de pre-fork.
Técnica pre-fork. Consiste en la creación previa de un grupo de procesos y su mantenimiento hasta que sea necesaria su utilización.
Las principales ventajas de este diseño residen en su simplicidad de implementación y su seguridad.
La gran desventaja de este diseño es el bajo rendimiento. La creación o eliminación de un proceso son tareas pesadas para el sistema operativo y consumen una gran cantidad de tiempo.

Servidores basados en hilos (Threads)

Este tipo de diseño hoy en día es mucho más común que el basado en procesos. Los conceptos básicos respecto al funcionamiento de un servidor basado en procesos son aplicables también a este modelo.
Las principales diferencias de los dos modelos reside en el propio concepto de hilo*.
La ventaja es que la creación de un hilo no es tan costosa como la de un proceso. Varios hilos de un mismo proceso pueden compartir datos entre ellos, ya que comparten el mismo espacio de memoria.
El modelo de servidor basado en hilos hereda muchas de las características de los servidores basados en procesos, entre ellas la de la simplicidad en su diseño e implementación. Por otro lado, el compartir el espacio de memoria implica un riesgo de seguridad que no tienen los servidores basado en procesos.

*__Hilos y procesos__.
Proceso: se puede definir como una ocurrencia o instancia de un programa en ejecución. Además, un proceso es propietario de una serie de recursos como: un espacio de direcciones en memoria, ficheros, hilos, etc..

Hilo: siguiendo con la definición anterior, un proceso totalmente aislado es un proceso inerte, es decir, para que un proceso sea capaz de hacer algo, el proceso debe ser propietario de al menos un hilo (thread). El hilo es el responsable de ejecutar el código contenido en el espacio de direcciones del proceso. De hecho, un proceso puede contener varios hilos y todos ellos ejecutando código "simultáneamente" en el espacio de direcciones del proceso y compartiendo recursos comunes.
Al compartir todos los hilos de un proceso la misma zona de memoria, si un hilo toca una variable, todos los demás hilos del mismo proceso verán el nuevo valor de la variable.
Si no hay hilos ejecutando código en el espacio de direcciones del proceso no hay ninguna razón para que el proceso continúe existiendo y el sistema destruirá automáticamente el proceso y su espacio en memoria.

Servidores basado en sockets no bloqueantes o dirigidos por eventos

Estos servidores basan su funcionamiento en la utilización de lecturas y escrituras asíncronas sobre sockets*.
Normalmente, estos servidores utilizan una llamada al sistema que examine el estado de los sockets con los que trabaja. Cada sistema operativo implementa una o más funciones de examen de sockets.
El objetivo de estas funciones es inspeccionar el estado de un grupo de sockets asociados a cada una de las conexiones.
La ventaja de este diseño es principalmente su velocidad.
Su principal desventaja es que la concurrencia es simulada; es decir, existe un sólo proceso y un sólo hilo, desde el cual se atienden todas las conexiones.

*__Socket__: no son más que puntos o medios de comunicación entre dos aplicaciones que permiten que un proceso hable (emita o reciba información) con otro proceso estando los dos en distintas máquinas. Lo vemos mejor con un dibujo:

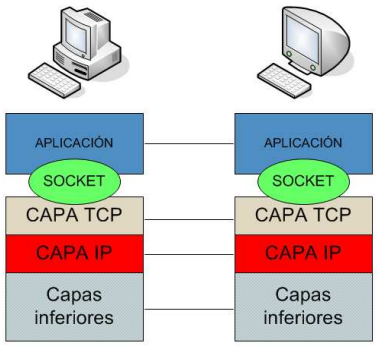

Figura 3

Si extrapolamos el concepto a la comunicación entre personas, un socket es al sistema de comunicación entre ordenadores lo que un teléfono es al sistema de comunicación entre personas: un punto de comunicación entre dos agentes (procesos o personas respectivamente) por el cual se puede emitir o recibir información.

Servidores implementados en el kernel

Este diseño es un poco especial. Se trata de un intento de acelerar la velocidad de un servidor Web mediante el movimiento de su código de espacio de usuario a espacio de kernel.
En teoría este modelo se muestra muy eficiente, pero de cara al mundo real, los problemas e inconvenientes son muy grandes. Hay que tener en cuenta que cualquier problema que se produzca a nivel de kernel puede ocasionar la caída de todo el sistema completo.

REFERENCIAS

- Apartados *"¿Qué es un servidor?"* e *"¿Y un servidor Web?"* - **Wikipedia** - http://es.wikipedia.org/

- Apartado *"Protocolo"* - **Comunicaciones y Redes de Computadores – Autor:** William Stallings - 6ª Edición - Prentice may.

- Apartado *"Protocolo HTTP"* – **Aprenda Servlets de Java como si estuviera en segundo** – Autores: Javier García de Jalón, José Ignacio Rodríguez y Aitor Imaz – Sa Sebastián, Abril 1999.

- Apartados *"Funcionamiento de un servidor Web"* y *"Arquitectura"* - **A Reference Architecture for Web Servers** - Autores: Ahmed E. Hassan, Richard C. Holt - Software Architecture Group (SWAG) -Dept. of Computer Science - University of Waterloo - Waterloo, Ontario - CANADA.

- Apartado *"Tipos de servidores Web"* – Proyecto Cherokee: Diseño, implementación y aspectos de rendimiento de servidores Web – Autor: Alvaro López Ortega.

- Apartado *"Hilos y procesos"* - **Introducción a los Hilos – Autor:** Jesús M. Vegas Hernández - Dpto. Informática - Universidad de Valladolid.