WWW (HTTP)

- Belfiori Jonatan
- Cardozo Griselda
- Logioco Pablo

Sumario:

- Introducción a HTTP
- Conexiones persistentes y no persistentes
- Formato del mensaje HTTP
- Mensajes HTTP
  - Petición
  - Respuesta
- Métodos
- Autorización y cookie
- HTTPS

Introducción a HTTP

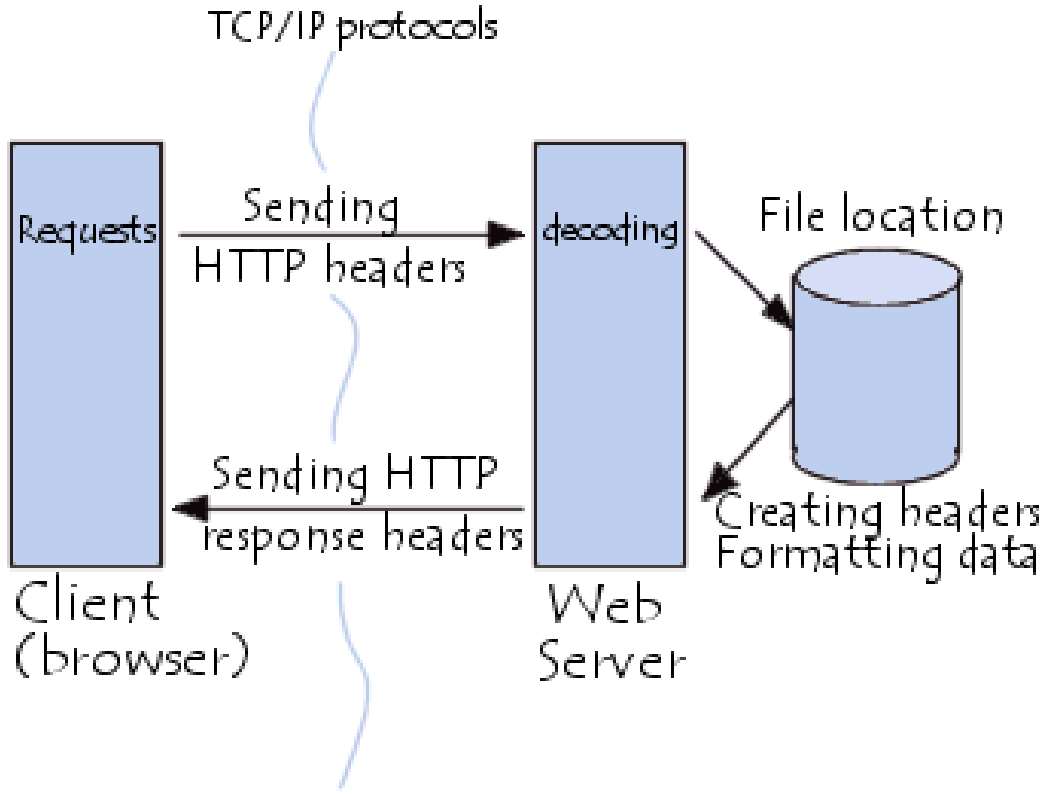
(***HTTP**, HyperText Transfer Protocol*)

HTTP es el conjunto de reglas para la transmisión y recepción de documentos hipertexto. Se diseñó específicamente para el World Wide Web: es un protocolo rápido y sencillo que permite la transferencia de múltiples tipos de información de forma eficiente y rápida.

Está implementado en dos programas: un programa cliente y un programa servidor. Los programas cliente y servidor, se ejecutan en sistemas finales diferentes, conversan entre sí intercambiando mensajes. HTTP define la estructura de estos mensajes y cómo se realiza el intercambio entre el cliente y el servidor.

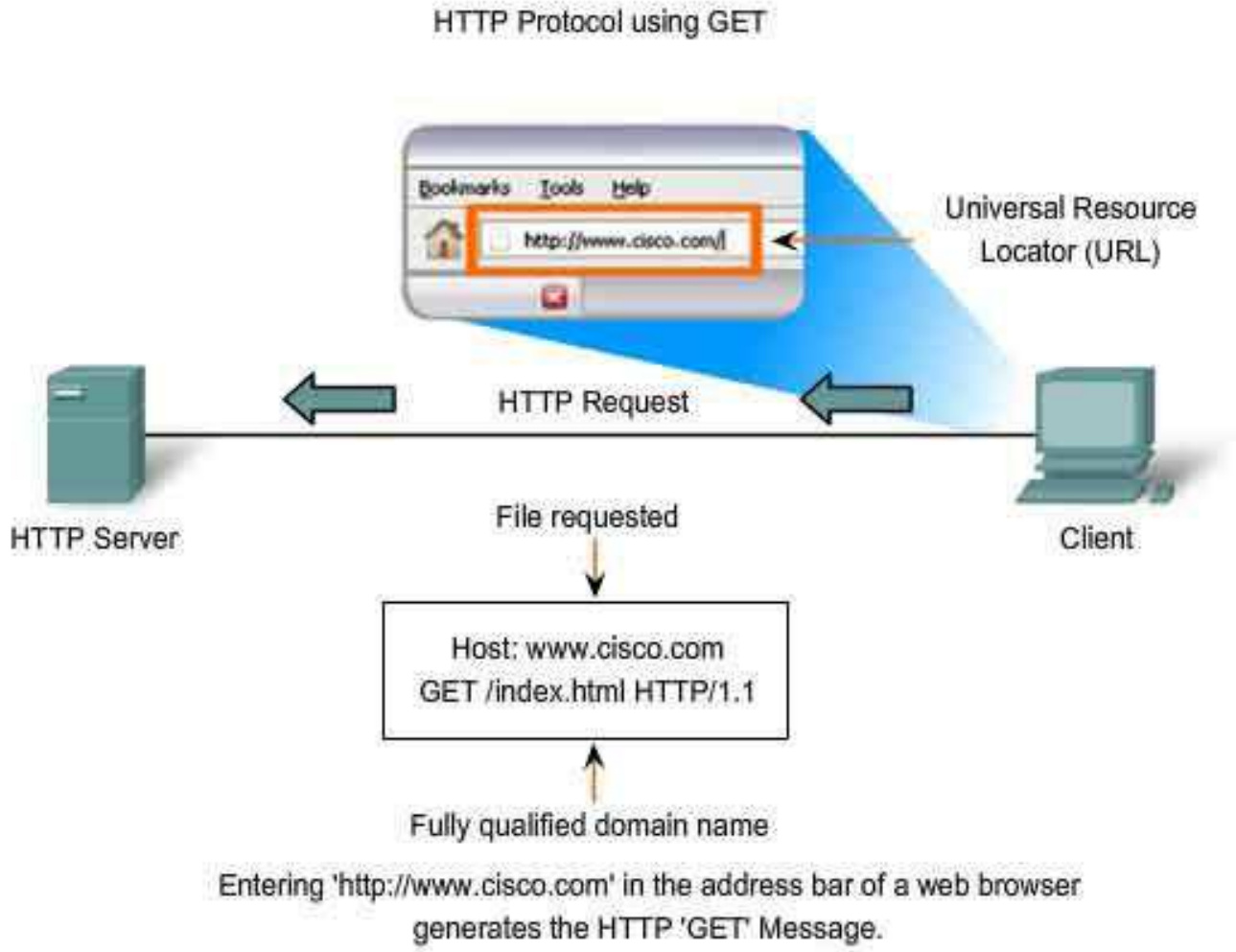
Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

1 - Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo "Location " del cliente Web. El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.

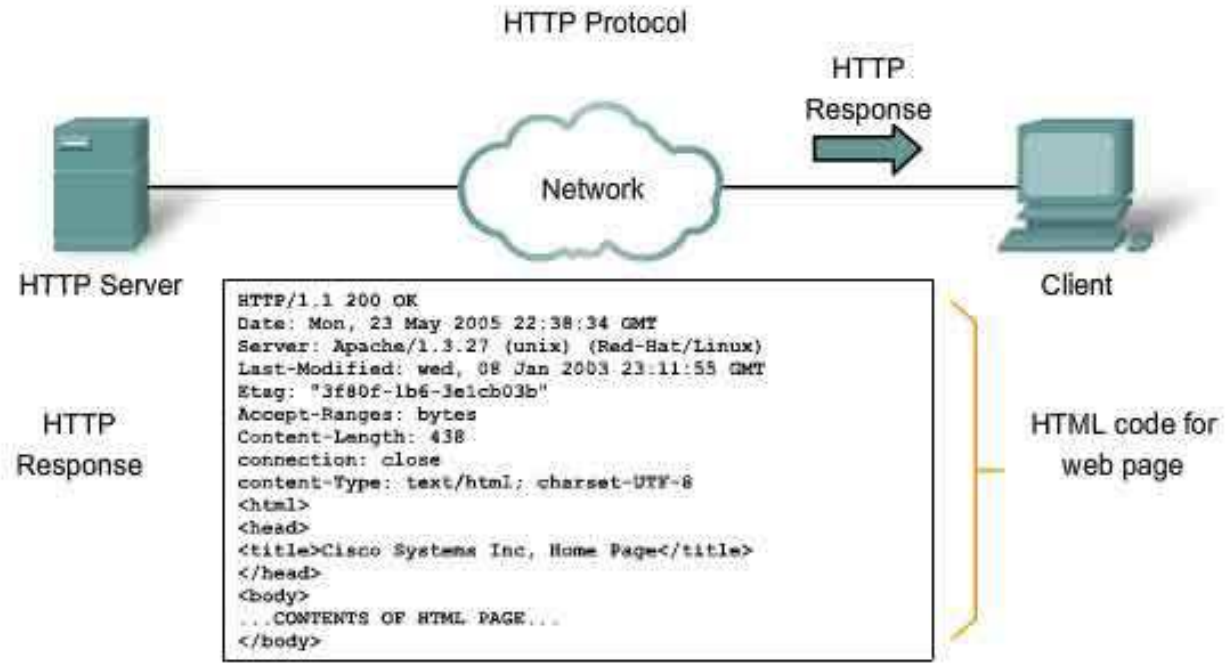


Ejemplo: http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html

2 - Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente. Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,...), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleado y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor.



3 - El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información. Se cierra la conexión TCP.



In response to the request, the HTTP server returns code for a web page.

Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML y cuatro para las imágenes.
En la actualidad se ha mejorado este procedimiento, permitiendo que una misma conexión se mantenga activa durante un cierto periodo de tiempo, de forma que sea utilizada en sucesivas transacciones. Este mecanismo, denominado HTTP Keep Alive, es empleado por la mayoría de los clientes y servidores modernos. Esta mejora es imprescindible en una Internet saturada, en la que el establecimiento de cada nueva conexión es un proceso lento y costos

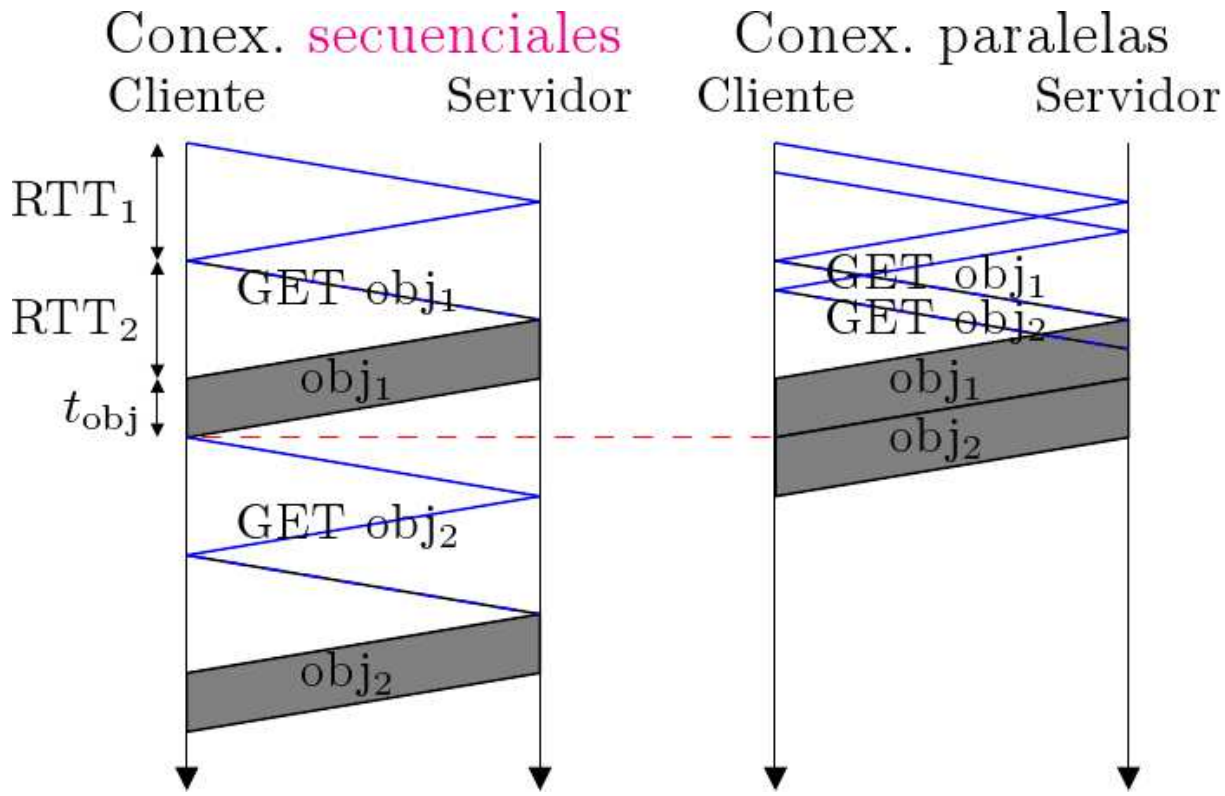
Conexiones persistentes y no persistentes

Cuando el cliente establece una conexión HTTP con un servidor Web, ésta puede ser de 2 tipos: no persistente o persistente. El HTTP/1.0 utiliza sólo conexiones no persistentes y el HTTP/1.1 puede utilizar además conexiones persistentes.

En una conexión no persistente, para cada objeto Web transferido se establece una conexión TCP independiente. En una conexión persistente, durante la misma conexión (establecida por un par cliente/servidor) se pueden transmitir muchos objetos Web, lo que generalmente ahorra recursos en el servidor (memoria y CPU) y en la red (ancho de banda).

Ambos tipos de conexiones pueden ser además secuenciales o paralelas (pipelining). Estas últimas permiten ahorrar tiempo si transmitimos varios objetos Web porque el cliente puede realizar una nueva petición antes de recibir la respuesta de una previa.

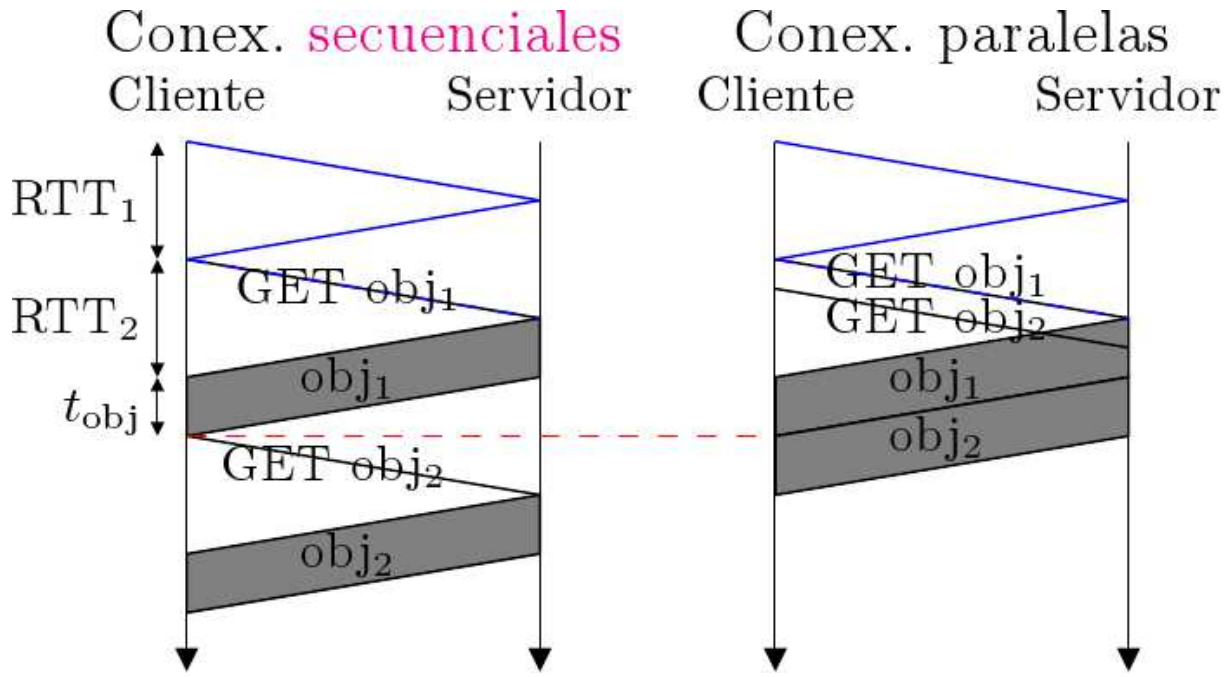
Conexiones no persistentes



RTT (Round-Trip Time) es el tiempo de ida y vuelta de un mensaje de longitud despreciable y t_{obj} es el tiempo de transmisión del objeto Web. En cada conexión TCP, el tiempo de establecimiento de conexión TCP necesita un RTT (RTT_1). El segundo RTT (RTT_2) se emplea en solicitar el objeto.

Conexiones persistentes

(Keep Alive)



El ahorro del establecimiento de una conexión TCP para cada conexión Web reduce el tiempo en el caso de las conexiones secuenciales. El tiempo de respuesta de las conexiones paralelas persistentes es el mismo que el de las conexiones no persistentes, aunque la carga para el servidor suele ser menor.

Formato del Mensaje HTTP

Un HTTP URL combina en una dirección simple los cuatro elementos básicos de información necesarios para recuperar un recurso desde cualquier parte en la Internet:

- El protocolo que se usa para comunicar, o enviar datos
- El servidor con el que se comunica,
- El puerto de red en el servidor para conectarse,
- La ruta al recurso en el servidor (por ejemplo, su nombre de archivo).

En su forma más simple, un HTTP URL tiene el siguiente formato:

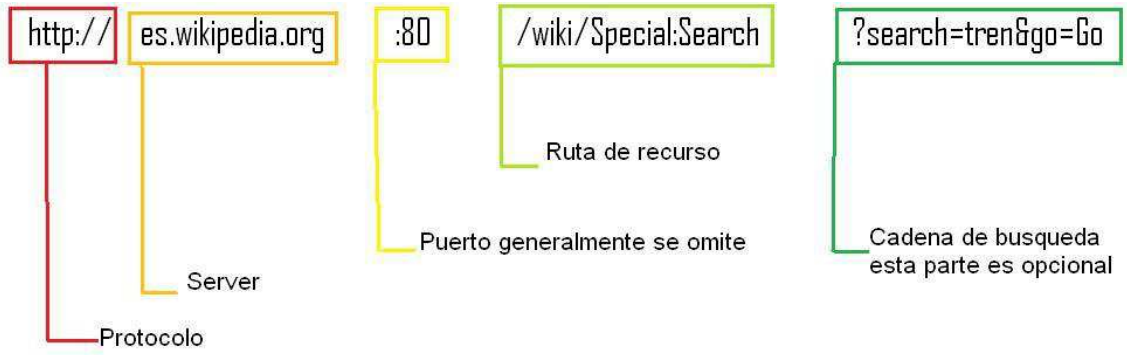
http://<host>:<port>/<path>

El número de puerto predeterminado para HTTP es 80. Si se omite la porción <path>, el URL apunta al recurso del primer nivel, es decir a la página de inicio.

Ejemplo:

http://es.wikipedia.org:80/wiki/Special:Search?search=tren&go=Go

Donde:



Mensajes HTTP

Un mensaje http consiste en una petición de un cliente al servidor y en la respuesta del servidor al cliente.
Las peticiones y respuestas pueden ser simples o completas. La diferencia es que en las peticiones y respuestas completas se envían cabeceras y un contenido. Este contenido se pone después de las cabeceras dejando una línea vacía entre las cabeceras y el contenido. En el caso de peticiones simples, sólo se puede usar el método GET y no hay contenido. Si se trata de una respuesta simple, entonces ésta sólo consta de contenido. Esta diferenciación entre simples y completas se tiene para que el protocolo HTTP/1.0 pueda atender peticiones y enviar respuestas del protocolo HTTP/0.9.

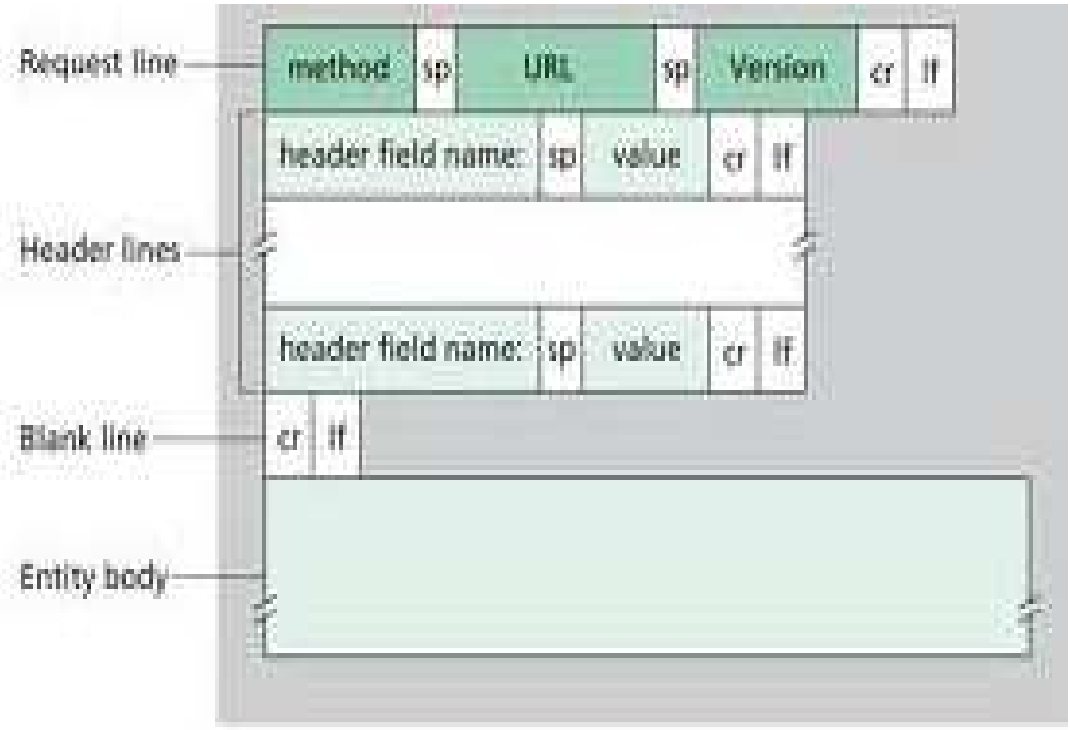
El diálogo con los servidores HTTP se establece a través de mensajes formados por líneas de texto, cada una de las cuales contiene los diferentes comandos y opciones del protocolo. Sólo existen dos tipos de mensajes, uno para realizar peticiones y otro para devolver la correspondiente respuesta.

Mensaje de respuesta	Mensaje de peticion
Resultado de la solicitud Cabeceras de la respuesta (línea en blanco) Información opcional	Comando HTTP + parámetros Cabeceras del requerimiento (línea en blanco) Información opcional

La primera línea del mensaje de solicitud contiene el comando que se solicita al servidor HTTP, mientras que en la respuesta contiene el resultado de la operación, un código numérico que permite conocer el éxito o fracaso de la operación. Después aparece, para ambos tipos de mensajes, un conjunto de cabeceras (unas obligatorias y otras opcionales), que condicionan y matizan el funcionamiento del protocolo.

La separación entre cada línea del mensaje se realiza con un par CR-LF (retorno de carro más nueva línea). El final de las cabeceras se indica con una línea en blanco, tras la cual se pueden incluir los datos transportados por el protocolo, por ejemplo, el documento HTML que devuelve un servidor o el contenido de un formulario que envía un cliente**.**

Mensaje de petición



Formato general de un mensaje de petición

La línea de petición tiene 3 campos:

- el campo de método: que puede tomar distinto valor como GET, POST y HEAD
- el campo de URL
- el campo de versión de HTTP

Las cabeceras más comunes:

- **Accept** : Los tipos MIME que prefiere el navegador.
- **Accept-Charset :** El conjunto de caracteres que espera el navegador.
- **Accept-Encoding** : Los tipos de codificación de datos (como gzip) para que el navegador sepa como decodificarlos. Los servlets pueden chequear explícitamente el soporte para gzip y devolver páginas HTML comprimidas con gzip para navegadores que las soportan.
- **Accept-Language :** El idioma que está esperando el navegador, en el caso de que el servidor tenga versiones en más de un idioma.
- **Authorization :** Información de autorización, usualmente en respuesta a una cabecera WWW-Authenticate desde el servidor.\
- **Connection** Sí un servlet obtiene un valor Keep-Alive aquí, u obtiene una línea de petición indicando HTTP 1.1 (donde las conexiones persistentes son por defecto), podría ser posible tomar ventaja de las conexiones persistentes, ahorrando un tiempo significante para las páginas Web que incluyen muchas piezas pequeñas. Para hacer esto,necesita enviar una cabecera Content-Length en la respuesta, que es fácilmente conseguido escribiendo en un ByteArrayOutputStream, y preguntando por el tamaño antes de escribir la salida.

- **Content-Length** (para mensajes POST, cúantos datos se han añadido)
- **Cookie**
- **Host** (host y puerto escuchado en la URL original)
- **User-Agent** (tipo de navegador, útil si el servlets está devolviendo contenido específico para un navegador)

El campo de entidad está vacío con el método GET, aunque se utiliza con el método POST.

Métodos

Un método se dice que es seguro si no provocan ninguna otra acción que no sea la de devolver algo (no produce efectos laterales). Estos métodos son el método GET y el método HEAD. Para realizar acciones inseguras (las que afectan a otras acciones) se pueden usar los métodos POST, PUT y DELETE. Aunque esto está definido así, no se puede asegurar que un método seguro no produzca efectos laterales, porque depende de la implementación del servidor.

HTTP define 8 métodos (algunas veces referido como "verbos") que indica la acción que desea que se efectúe sobre el recurso identificado. Lo que este recurso representa, si los datos pre-existentes o datos que se generan de forma dinámica, depende de la aplicación del servidor.

Comandos:

OPTIONS :

Este método representa un petición de información sobre las opciones de comunicación disponibles en la cadena petición-respuesta identificada por la URI de la petición. Esto permite al cliente conocer las opciones y requisitos asociados con un recurso o las capacidades del servidor. Devuelve los métodos HTTP que el servidor soporta para un URL específico.Esto puede ser utilizado para comprobar la funcionalidad de un servidor web mediante petición en lugar de un recurso específico.

GET:

El método GET requiere la devolución de información al cliente identificada por la URI. Si la URI se refiere a un proceso que produce información, se devuelve la información y no la fuente del proceso.

El método GET pasa a ser un GET condicional si la petición incluye las cabeceras If-Modified-Since, If- Unmodified-Since, If-Match, If-None-Match o If-Range. Estas cabeceras hacen que el contenido de la respuesta se transmita sólo si se cumplen unas condiciones determinadas por esas cabeceras. Esto se hico para reducir el tráfico en las redes.

También hay un método GET parcial, con el que se envía sólo parte del contenido del recurso requerido. Esto ocurre cuando la petición tiene una cabecera Range. Al igual que el método GET condicional, el método GET parcial se creó para reducir el tráfico en la red.

HEAD:

El método HEAD es igual que el método GET, salvo que el servidor no tiene que devolver el contenido, sólo las cabeceras. Estas cabeceras que se devuelven en el método HEAD deberían ser las mismas que las que se devolverían si fuese una petición GET.

Este método se puede usar para obtener información sobre el contenido que se va a devolver en respuesta a la petición. Se suele usar también para chequear la validez de links, accesibilidad y modificaciones recientes

DELETE:

Este método se usa para que el servidor borre el recurso indicado por la URI de la petición. No se garantiza al cliente que la operación se lleve a cabo aunque la respuesta sea satisfactoria.

POST:

El método POST se usa para hacer peticiones en las que el servidor destino acepta el contenido de la petición como un nuevo subordinado del recurso pedido. El método POST se creó para cubrir funciones como la de enviar un mensaje a grupos de usuarios, dar un bloque de datos como resultado de un formulario a un proceso de datos, añadir nuevos datos a una base de datos. La función llevada a cabo por el método POST está determinada por el servidor y suele depender de la URI de la petición. El resultado de la acción realizada por el método POST puede ser un recurso que no sea identificable mediante una URI.

PUT:

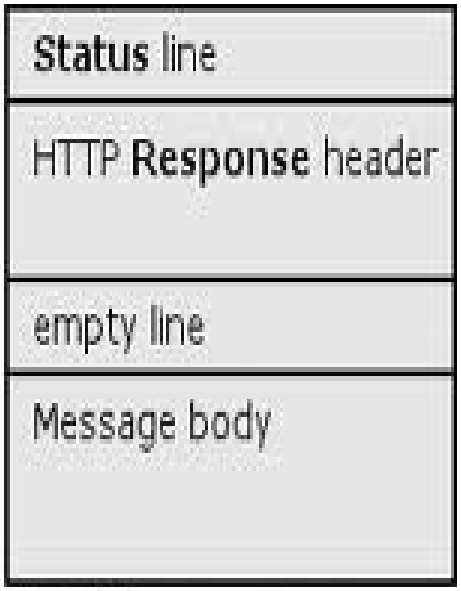
El método PUT permite guardar el contenido de la petición en el servidor bajo la URI de la petición. Si esta URI ya existe, entonces el servidor considera que esta petición proporciona una versión actualizada del recurso. Si la URI indicada no existe y es válida para definir un nuevo recurso, el servidor puede crear el recurso con esa URI. Si se crea un nuevo recurso, debe responder con un código 201 (creado), si se modifica se contesta con un código 200 (OK) o 204 (sin contenido). En caso de que no se pueda crear el recurso se devuelve un mensaje con el código de error apropiado. La principal diferencia entre POST y PUT se encuentra en el significado de la URI. En el caso del método POST, la URI identifica el recurso que va a manejar en contenido, mientras que en el PUT identifica el contenido. Un recurso puede tener distintas URI.

ç

TRACE:

Este método se usa para saber si existe el receptor del mensaje y usar la información para hacer un diagnóstico.

Mensaje de respuesta



HTTP Response message

La línea de status tiene tres campos:

- el campo de versión del protocolo
- el código de status
- el correspondiente mensaje de status

El código de estado es para aplicaciones de programación, mientras que el texto que lo acompaña está destinado a los lectores humanos. El primer dígito del código de estado define la categoría del resultado código.

La siguiente tabla enumera los primeros dígitos y permite ver las categorías correspondientes

Significado del valor numérico	Código de estado
Informativo - La solicitud fue recibida y se está procesando.	100-199
Éxito - La acción fue recibida con éxito, entendido y aceptado.	200-299
Redirección - Otras peticiones hay que realizar para completar la petición.	300-399
Error en Cliente - La solicitud contiene mala sintaxis y no pueden llevarse a cabo.	400-499
Error de servidor - El servidor no pudo cumplir con una solicitud aparentemente válida.	500-599

Líneas de cabecera:

- Connection-close: para indicar al cliente que va a cerrar la conexión TCP después de enviar el mensaje

- Date: indica la fecha y hora en que se creó y envío la respuesta HTTP por parte del servidor.

- Server: indica que servidor generó el mensaje

- Last-Modified : indica la fecha y hora en que el objeto fue creado o modificado por última vez.

- Content-Length: indica el número de bytes del objeto enviado.

El cuerpo de entidad es la esencia del mensaje, contiene el propio objeto pedido.

Autorización **y cookies**

El protocolo HTTP no tiene estado. Sin embargo, a menudo deseable que un sitio web identifique a los usuarios, bien porque el servidor desee restringir el acceso de los usuarios, o porque quiera servir cierto contenido en función de la identidad del usuario. HTTP proporciona dos mecanismos que ayudan al servidor a identificar a los usuarios: Autorización y cookies.

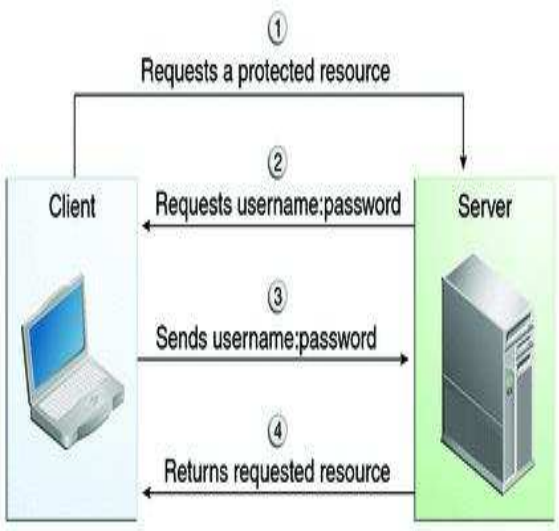
Autorización:

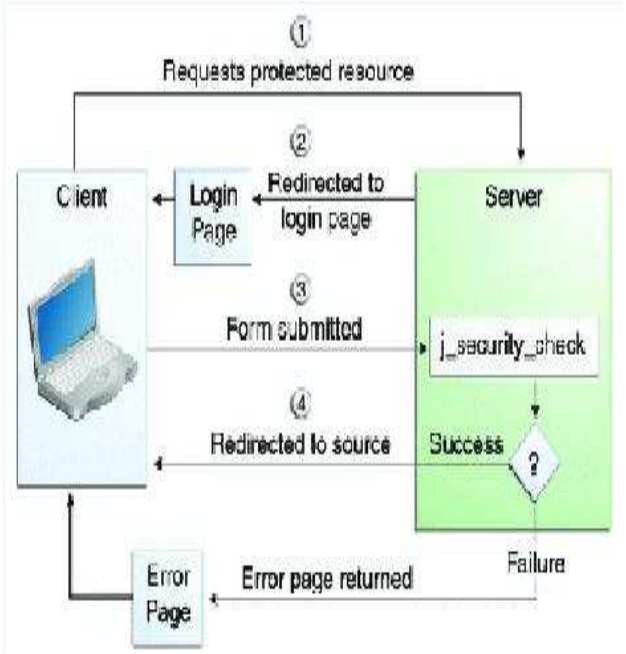
Se denomina autorización al requisito que obliga a los usuarios proporcionar un nombre de usuario y una palabra clave. La petición y recepción de autorización se realiza habitualmente por cabeceras HTTP y códigos de estatus especiales.

El protocolo HTTP incluye un mecanismo “digest accsess authentication” que permite acceder a una página web sólo cuando el usuario ha facilitado un nombre de usuario y contraseña correctos. Una vez que se han introducido los credenciales, el navegador las almacena y las utiliza para acceder a las páginas siguientes, sin pedirlas de nuevo al usuario.

Cuando se utiliza una autentificación básica pasa lo siguiente:

1 - El cliente manda un mensaje de petición pidiendo acceso a recursos protegidos.
2 – El servidor retorna una petición de usuario y contraseña.
3 – El cliente envía el usuario y la contraseña.
4 – Si es correcta el servidor autentifica al usuario y devuelve el recurso solicitado si no es correcta retorna una página de error.





Cookies

Las cookies son un mecanismo alternativo que utilizan los sitios para hacer un seguimiento de los usuarios.
La tecnología de cookies tiene cuatro componentes:

1. Una línea de cabecera de cookie en el mensaje HTTP de respuesta;
2. Una línea de cabecera de cookie en el mensaje de HTTP de petición;
3. Un archivo de cookie que se almacena en el sistema del usuario y que es gestionado por el navegador del usuario;
4. Una base de datos de respaldo en el sitio web.

Ejemplo:

Se solicita una pagina con el comando GET y esta en su respuesta devuelve una respuesta con el comando Set-cookie para que el navegador guarde cookies

Si el navegador ya tiene las cookies autentificadas cada mensaje al servidor va a contener la cookie. Con un mensaje como el siguiente:

Atributos de la cookies:

Caducidad:

Es un límite de tiempo en el que una de ellas puede ser usadas. Cuando las cookies han caducado, estas no son enviadas al navegador. Las condiciones que suelen tener lo servidores para dejar de enviar cookies pueden ser:

- al finalizar una sesión de usuario: por ejemplo, cuando se cierra el navegador (si esta no es persistente)
- Se ha fijado una fecha de caducidad y esta ha pasado.
- La fecha de caducidad es cambiada a una fecha anterior (por el servidor)
- esta se borra por orden del usuario.

Autentificación:

Muchos servidores o páginas web utilizan las cookies para reconocer usuarios que ya se hayan autenticado o para personalizar páginas web dependiendo de las opciones que un usuario seleccione. Por ejemplo, esto puede suceder cuando:

- El usuario escribe su nombre y contraseña, los cuales son enviados al servidor
- El servidor verifica la información proporcionada, y si es correcta devuelve una página de confirmación con una cookie, guardando así esta información en la computadora del usuario.
- Cuando el usuario visita una página la cual pertenece al servidor, este verifica la existencia de las cookies y luego comprueba si las cookies existentes son iguales a las que han sido guardadas en el servidor. Si hay coincidencias, el servidor puede identificar el usuario que solicitó la página.

HTTPS

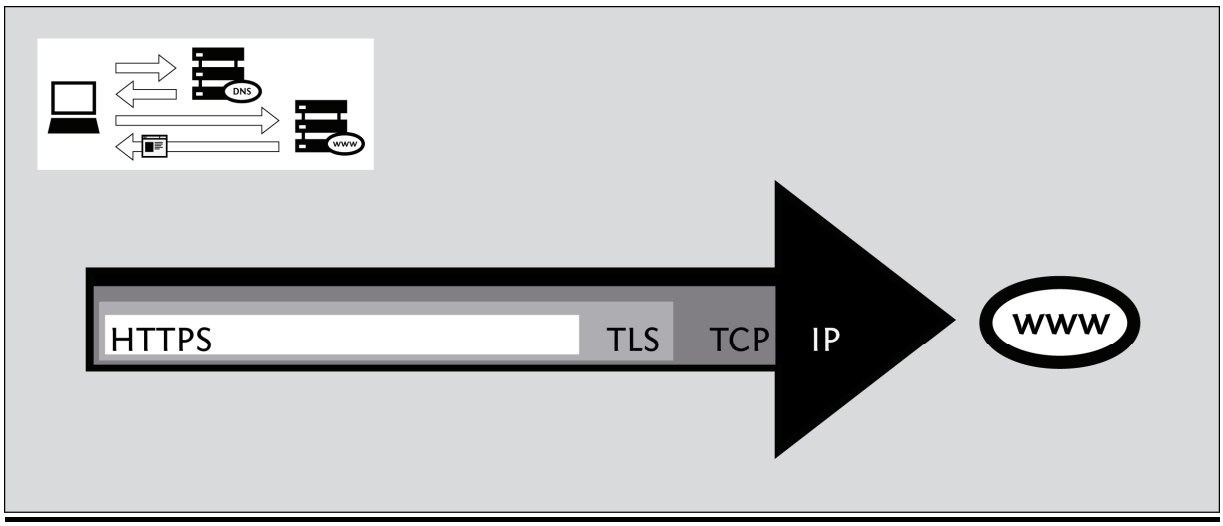
(Hypertext Transfer Protocol Secure)

El sistema HTTPS utiliza un cifrado basado en SSL/TLS* para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) . De este modo se consigue que la información sensible (usuario y claves de paso normalmente) no pueda ser usada por un atacante que haya conseguido interceptar la transferencia de datos de la conexión, ya que lo único que obtendrá será un flujo de datos cifrados que le resultará imposible de descifrar.

En el protocolo HTTP las URL comienzan con "http://" y utilizan por defecto el puerto 80, las URL de HTTPS comienzan con "https://" y utilizan el puerto 443 por defecto

- (Conexión Segura cifrada Secure Sockets Layer (SSL) - Seguridad de la Capa de Transporte(TASS

Ejemplo:



El protocolo HTPPS depende del protocolo **TSL** para realizar el cifrado de las comunicaciones para que sean privadas y no se puedan modificar durante su viaje a través de la red. El protocolo **TSL**, a su vez, depende del protocolo TCP para asegurarse que la información no se pierda accidentalmente o se corrompa durante la transmisión. Finalmente, TCP depende del protocolo IP para asegurarse de que los datos son entregados en el destino esperado.