Resumen Capa de Aplicación - 2024

Funciones de la Capa de Aplicación TCP/IP

- Provee servicios de comunicación a los usuarios y a las aplicaciones, incluye las aplicaciones mismas.
- Las aplicaciones que usan la red y los protocolos que implementan las aplicaciones pertenecen a esta capa.
- Interfaz con el usuario (UI) u otras aplicaciones/servicios.

Componentes de la Capa de Aplicación TCP/IP

- Los elementos de esta capa son:
    - Programas que corren en diferentes plataformas, en la mayoría de los casos en los nodos finales (end-systems) no en el núcleo de la red, y se comunican entre sí.
    - Los protocolos que implementan.
- Cubre a las capas de Aplicación, Presentación y Sesión del modelo OSI.

Funciones de la Capa de Sesión

- Administra las conversaciones entre las aplicaciones.
- Maneja la actividad e informa las excepciones.
- Está integrada en las aplicaciones de red.
- No es obligatoria, podría estar ausente.

Consideraciones

- Es muy delgada, es decir, posee pocas características/funcionalidad.
- Generalmente no es utilizada.

Funciones de la Capa de Presentación

- Se encarga de la compresión y descompresión de los datos, junto con el cifrado y descifrado de los mismos.
- Integrada en las aplicaciones de red mismas.

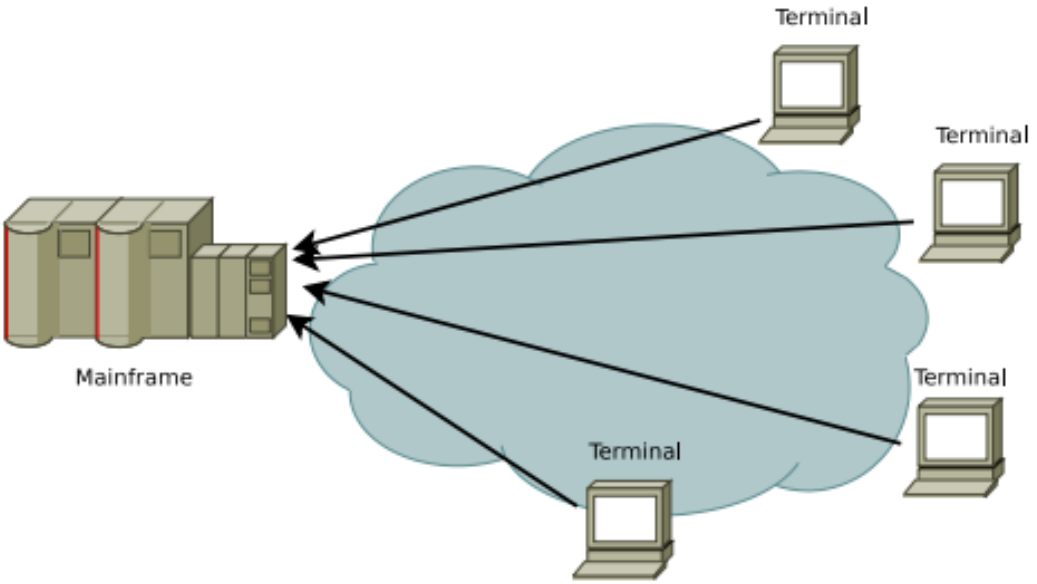
Funciones de la Capa de Aplicación

- Define el formato (sintaxis y semántica) de los mensajes. Existen protocolos que trabajan con formatos específicos de mensajes. A su vez, define cómo debe ser el intercambio de mensajes.

Modelos de Comunicación de Aplicaciones

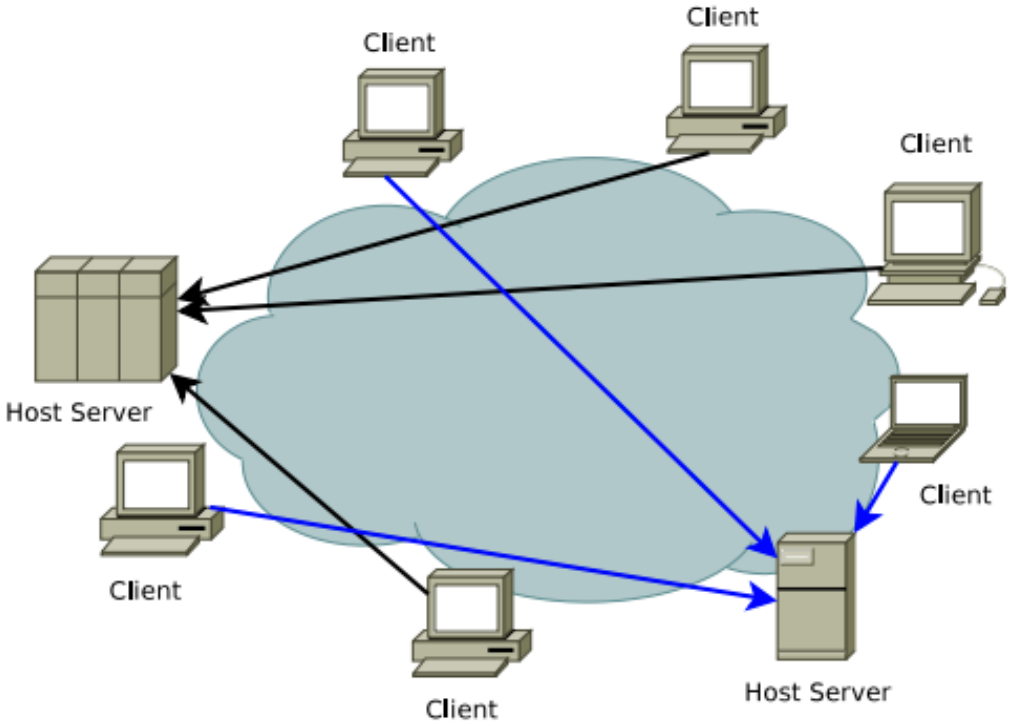
Modelo Mainframe (Dumb Client)

- Modelo antiguo que resurgió por tener thin-clients.
- En el modelo puro, el mainframe es el que decide cuando le da el control al cliente y además es el encargado de manejar el diálogo de las comunicaciones.
- El cliente es “tonto”, solo corre la comunicación y la interfaz física con el usuario mientras que el servidor pone todo el procesamiento.



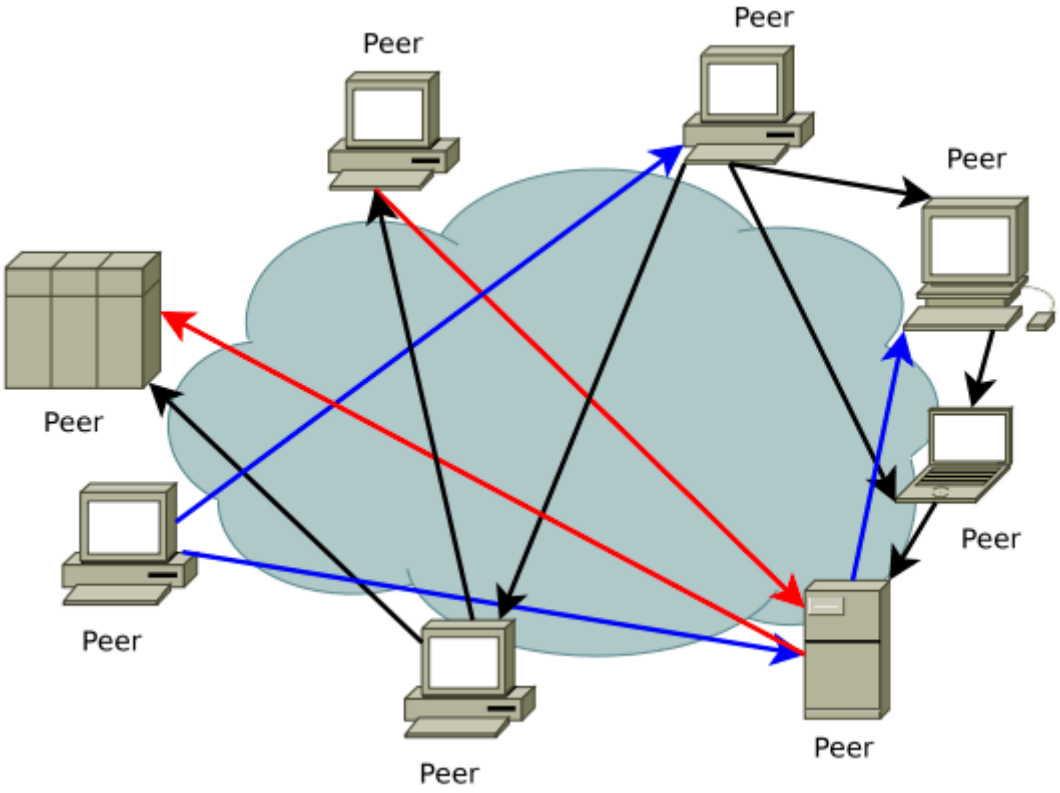
Modelo de Cliente/Servidor

- El cliente se conecta al servidor y se comunica a través de este haciendo pedidos y el servidor corre el servicio esperando de forma pasiva la conexión y el pedido de datos/requerimientos.
- La carga de trabajo se ve más repartida entre el cliente y el servidor.



Modelo Peer-to-Peer

- En este modelo ya no existe el rol de cliente o servidor. Existen pares que pueden cumplir rol de cliente o de servidor o de los dos en algún instante.
- La distribución de la carga de trabajo es igualitaria.
- Es más difícil de organizar ya que la organización deja de estar del lado del servidor, ahora todos pueden hacerlo.
- Es un modelo asimétrico N a N.
- Pueden existir híbridos de este modelo donde haya nodos centrales que sí cumplan un rol específico, por ejemplo un nodo de registro de información de los demás nodos o de qué nodos están conectados.
    - Los modelos híbridos son más escalables que los puros.



Requerimientos de Aplicaciones

- Están más relacionados con la Capa de Transporte. Dependiendo de los requerimientos de la Capa de Aplicación (seguridad, eficiencia, confiabilidad, etc) se eligen los protocolos más adecuados para la Capa de Transporte.

Protocolo HTTP

- Protocolo que sigue el Modelo Cliente/Servidor, sin estado, es decir, por cada requerimiento que hace el cliente, el servidor no almacena información sobre el estado de la relación con el cliente.
- Es un protocolo que corre sobre TCP y que usa el puerto 80 por default.

Conexiones Persistentes

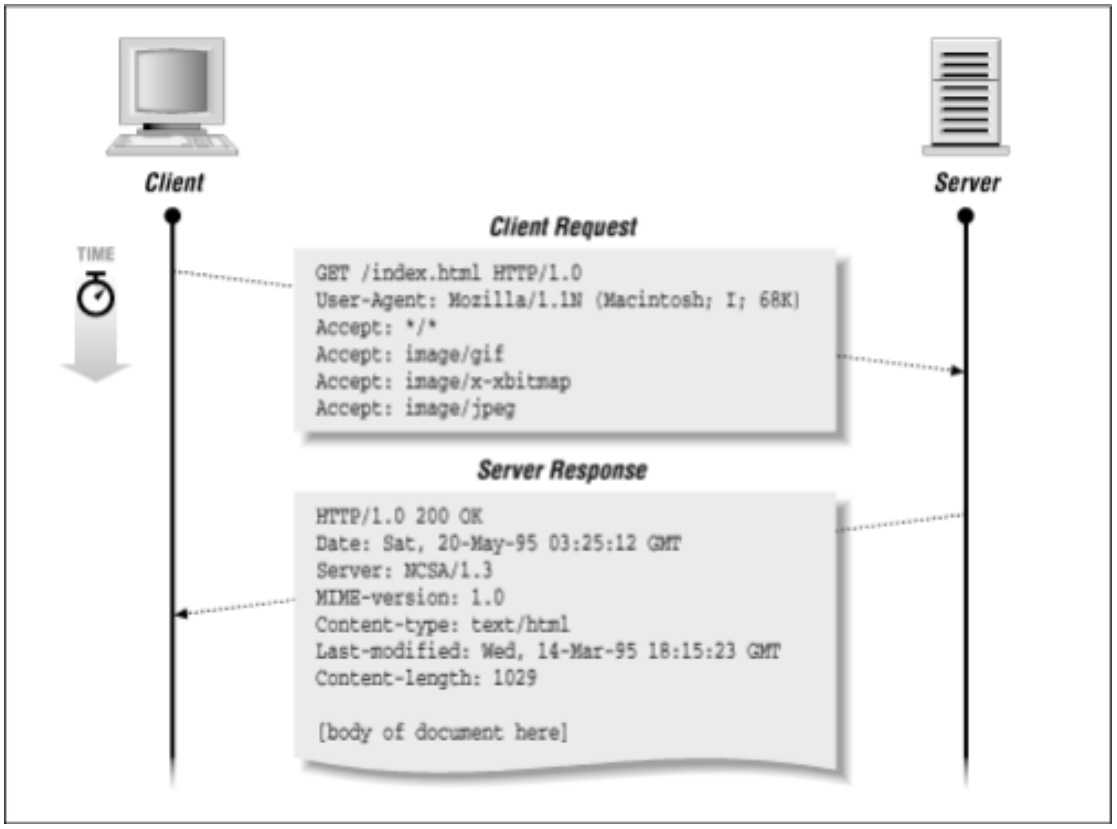
- En una conexión persistente, la conexión TCP se mantiene abierta después de que se haya completado una solicitud, permitiendo que múltiples recursos sean transferidos sin tener que abrir nuevas conexiones. Esto permite que varias solicitudes y respuestas se envíen a través de la misma conexión.

Conexiones No Persistentes

- En una conexión no persistente, el cliente establece una conexión TCP al servidor, solicita un solo recurso, el servidor envía la respuesta, y luego la conexión se cierra.

Para cada recurso adicional que el cliente necesita, se debe abrir una nueva conexión.

Esquema de HTTP



HTTP 0.9

- Primera versión de HTTP que nunca se estandarizó.
- Pasos para obtener un documento:
    - Establecer conexión TCP.
    - HTTP request vía comando GET.
    - HTTP response enviando la página requerida.
    - Cerrar la conexión TCP por parte del servidor.
    - Si no existe el documento u ocurre un error, se cierra la conexión.
- Solo se podía hacer las request con GET y las response eran ASCII en documentos HTML.

HTTP 1.0

- Versión estandarizada con un proceso basado en HTTP 0.9.
- Define códigos de respuesta.
- Para los request, define diferentes métodos HTTP (GET, POST, HEAD, PUT, DELETE, LINK y UNLINK).

- Para los response, admite repertorios de caracteres además del ASCII como response, también admite MIME (Multipurpose Internet Mail Extensions).
- Por defecto no usa conexiones persistentes.

Método GET

- Se usa para obtener el documento requerido.
- Puede enviar información pero no demasiada, y esta es enviada en la URL la cual genera limitaciones por su tamaño.
- No espera recibir datos en el body de la petición.

Método HEAD

- Idéntico a GET, pero solo requiere la meta información del documento.

Método POST

- Hace un requerimiento de un documento pero también envía información en el body de la petición.
- Puede enviar más información que GET y se usa normalmente para formularios HTML.

Método PUT

- Usado para reemplazar un documento en el servidor. En general, deshabilitada.

Método DELETE

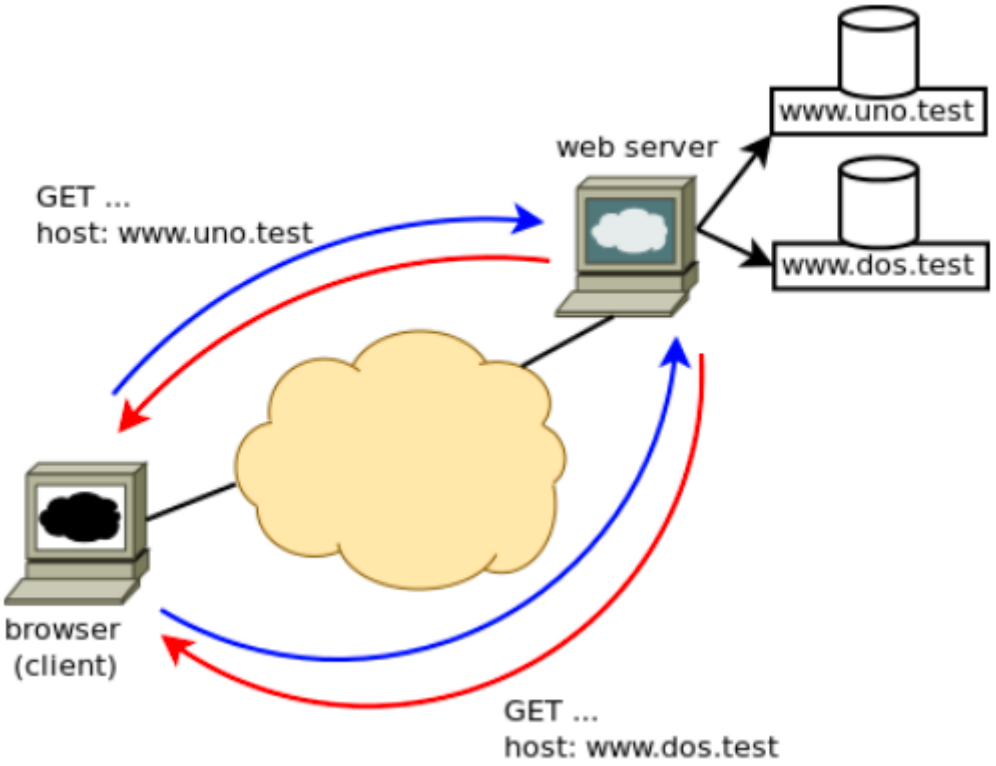
- Usado para borrar un documento en el servidor. En general, deshabilitada.

Métodos LINK y UNLINK

- Establecen y eliminan relaciones entre documentos.

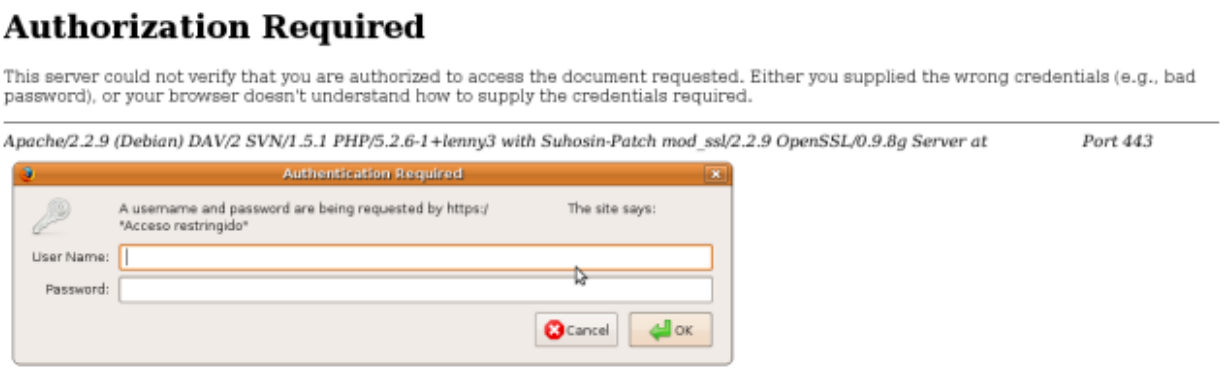
Hosts Virtuales

- Mediante el parámetro Host del encabezado HTTP podemos acceder a diversos sitios webs que están en un mismo Host, es decir, podemos multiplexar varios servicios sobre un mismo host.



Autenticación HTTP

- En HTTP 1.0 se realizaba la autenticación usando Headers.
- Mediante el uso de los encabezados del protocolo HTTP 1.0 un cliente se puede autenticar con un usuario y clave.
- El servidor pide la autenticación a través de un mensaje con código 401 indicando la necesidad de autenticación y un Dominio, luego el navegador le solicita al usuario sus datos de autenticación y los envía al servidor en texto claro, a partir de estos valores el servidor dará o no acceso.

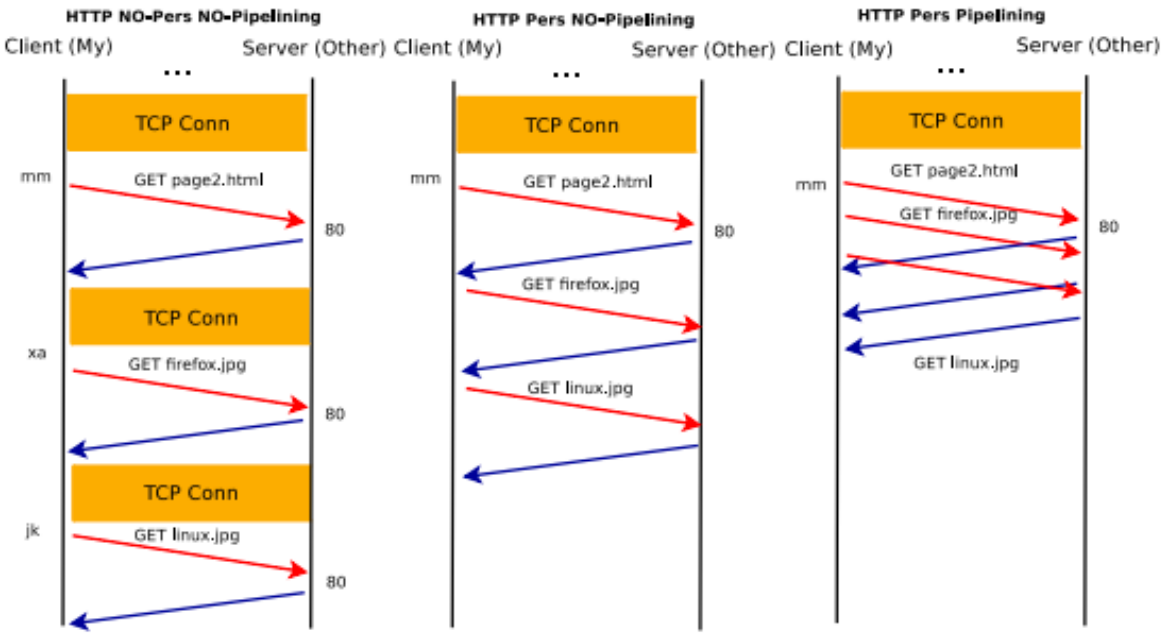


Problemas

- No persiste la conexión. Esto hace que se produzca un Overhead de aperturas y cierres de conexiones TCP para poder transportar los requerimientos, por lo tanto, se reduce la eficiencia al ser más lento.

HTTP 1.1

- Versión de HTTP estandarizada que usa conexiones persistentes por omisión y pipelining para mejorar el tiempo de respuestas.
- Permite el pedido y respuesta de requerimientos de forma paralela.



TRACE

- Mensaje utilizado para debugging.

CONNECT

- Mensaje usado para generar conexiones a otros servicios montados sobre HTTP.

Pipelining

- No se necesita esperar la response del servidor para poder pedir otro objeto HTTP, esto hace que se mejoren los tiempos de respuesta.
- Solo se utiliza con conexiones persistentes y sobre la misma conexión se debe mantener el orden de los objetos que se devuelven.

Problema

- Puede ocurrir que se generen Head Of Line Blockings, ya que si se solicita un objeto muy grande con un tiempo de obtención muy lento y luego se solicitan objetos más chicos y rápidos de obtención, al tener que mantener el orden de petición para la devolución de los objetos, el primero bloquea a los demás disminuyendo la eficiencia.

Solución

- Los navegadores generan múltiples conexiones para enviar los objetos sin que se generen Head Of Line Blockings.

Ventaja

- Al usar múltiples conexiones para solicitar objetos, cada uno viaja de manera independiente lo que lo hace más rápido.

Desventajas

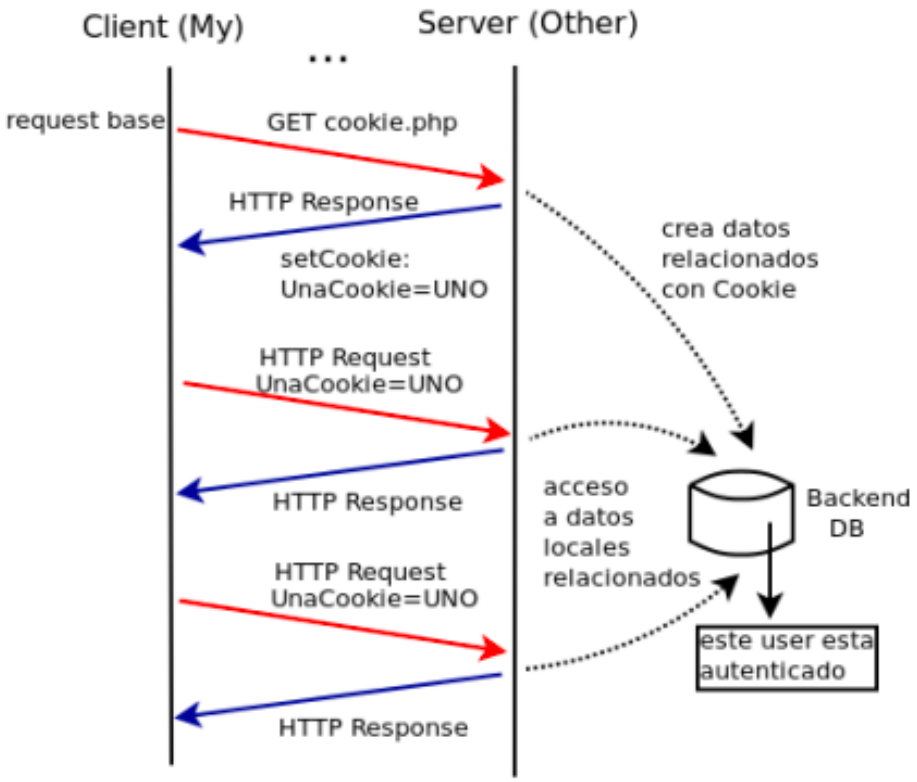
- Se genera una mayor sobrecarga en el uso de los recursos del Sistema Operativo.
- Si se utiliza TLS/SSL se genera una sobrecarga de los mismos ya que cada conexión va a manejar de forma independiente los parámetros de TLS/SSL, generando una mayor carga en el cálculo necesario en el lado del cliente como en el lado del servidor.
- Al haber varios requests se genera una duplicación de los datos de los Headers de las solicitudes HTTP.
- Necesita un control de congestión y del uso de la red.

Redirects

- Cuando un objeto se mueve, el servidor puede contestar con un mensaje de redirect ya sea con código 302 que indica la nueva ubicación del objeto y requiere la confirmación del usuario, o con código 301 que indica que el objeto se movió permanentemente a la nueva ubicación.
- Puede generar problemas con las Cookies.

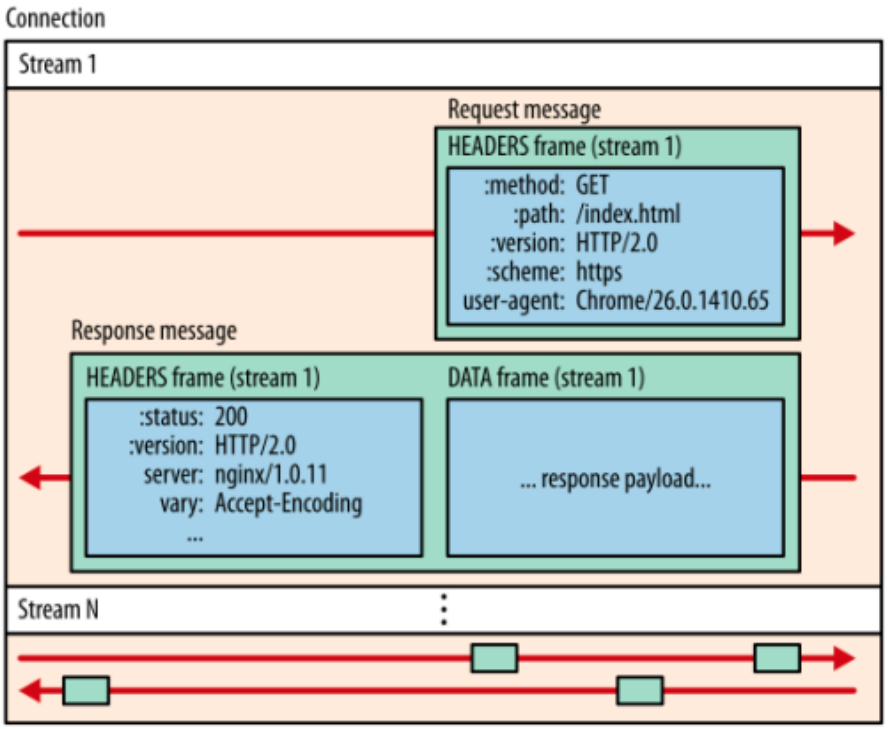
Cookies

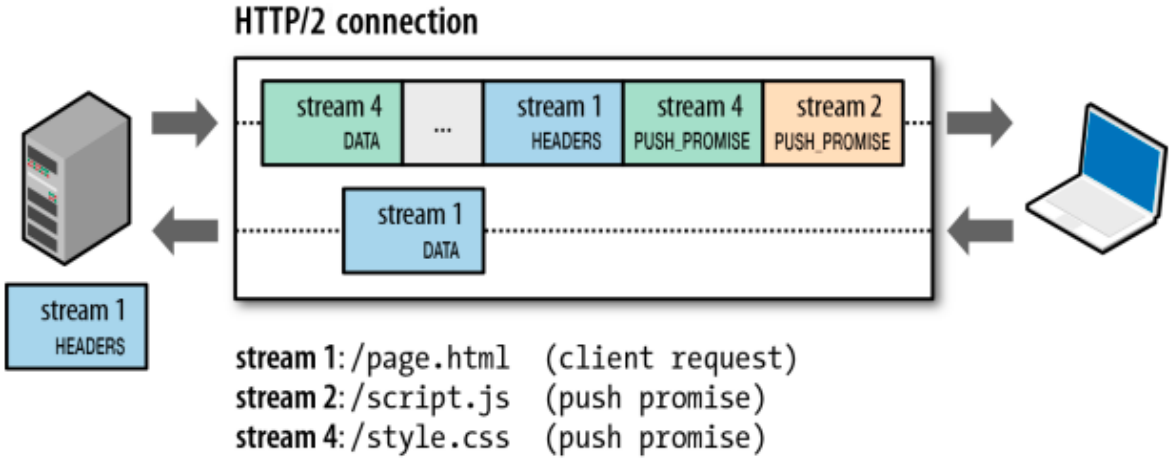
- Mecanismo que permite a las aplicaciones web del servidor “manejar estados”.
- La Cookie se introduce al cliente mediante el mensaje “Set-Cookie” en el header, indicando un par (nombre, valor). Luego de que el servidor almacena la Cookie, cada vez que el cliente haga una petición la Cookie va a ser enviada en el header de la misma.
- El servidor puede usarla o no, como también borrarla.



HTTP 2.0

- Protocolo lanzado para poder mejorar la eficiencia en cómo fluyen los datos a través de la red en comparación con HTTP 1.1. (Problema de múltiples conexiones).
- No es un reemplazo del protocolo completo, solo una mejora en el aspecto mencionado antes.
- Consiste en manejar menos conexiones donde en cada una se pueden mandar varios requerimientos que pueden viajar de forma independiente sobre esa conexión. Los requerimientos se dividen en Streams de datos codificados en binario, bidireccionales que actúan como una sub-conexión, cada Stream tendrá su prioridad (generando un orden por prioridad) y un ID, y a su vez, este Stream, se verá dividido en Mensajes y cada uno de estos Mensajes se verá dividido en Frames que guardan porciones del Mensaje, cada uno de estos Frames tendrá un encabezado fijo de 9 bytes, datos y una identificación que diga a qué tipo de Frame pertenece (HEADERS, DATA, PUSH_PROMISE, WINDOW_UPDATE, SETTINGS, etc).





Diferencias con HTTP 1.1

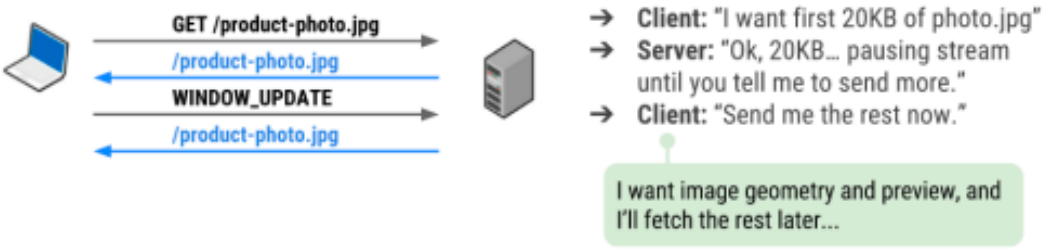
- Protocolo binario en lugar de textual (con ASCII) que es más eficiente desde el punto de vista de la cantidad de datos y del parsing de los mensajes.
- Multiplexa varios request en una petición en lugar de ser una secuencia ordenada y bloqueante.
- Agrega un flow control por frames.
- Usa compresión de encabezado.
- Permite a los servidores “pushear” datos a los clientes.
- La mayoría de las implementaciones pasan a estar montadas en TLS/SSL aunque el estándar no lo requiera.

HEADERS

- Se mantienen casi todos los HEADERS de HTTP 1,1 pero no se codifican más en ASCII.
- Surgen nuevos pseudo-headers que contiene información que estaba en el método y otros headers.

Priorización y Flow-Control

- Los streams dentro de una misma conexión tienen flow-control individual. Pueden tener un weight/prioridad y se pueden asociar/priorizar de forma jerárquica, dependencias, etc.



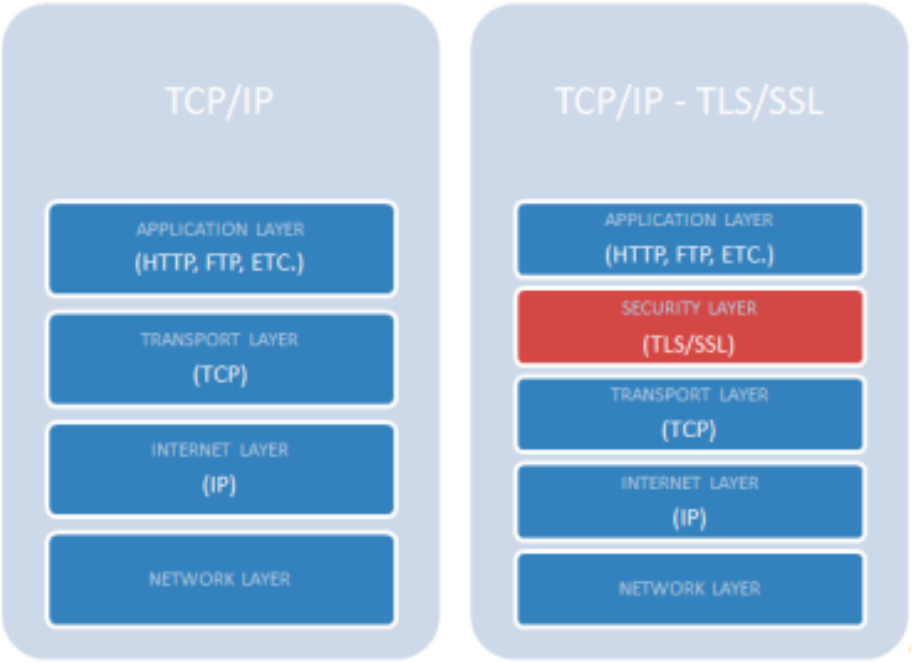
Push Promises

- El cliente solicita un recurso y el servidor le envía el recurso solicitado y ofrece/promete el envío de otros recursos que posiblemente el cliente pueda llegar a requerir.

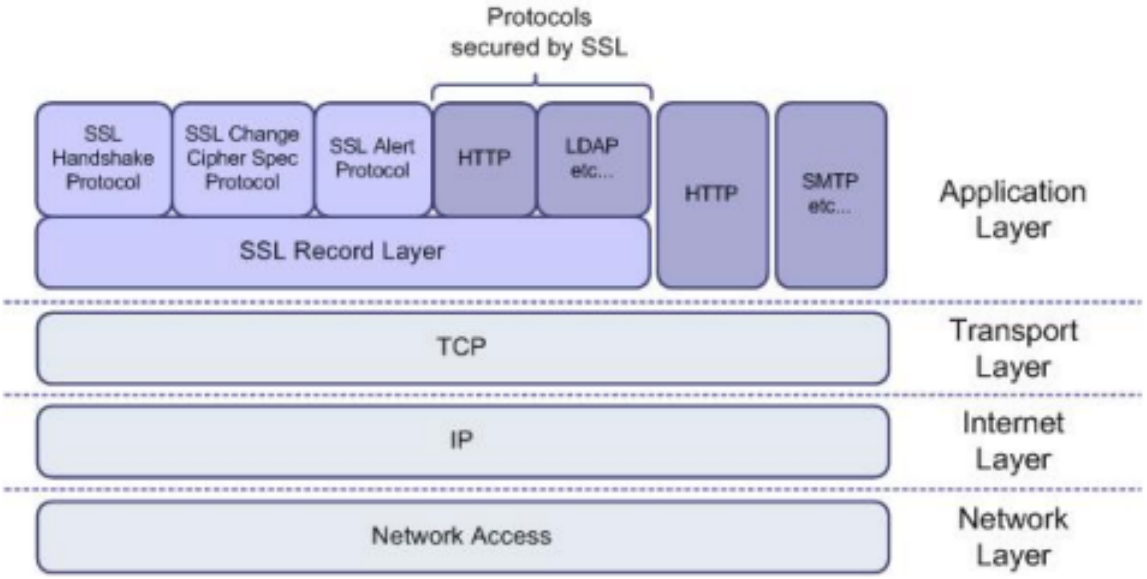
Otras Características

- Se puede generar un Upgrade Header durante la conexión:
    - Negociamos a través de TLS/SSL que por ejemplo, la conexión inicie con HTTP 1.1 pero si los 2 extremos lo soportan se genere un Upgrade a un protocolo HTTP 2.0.

TLS/SSL

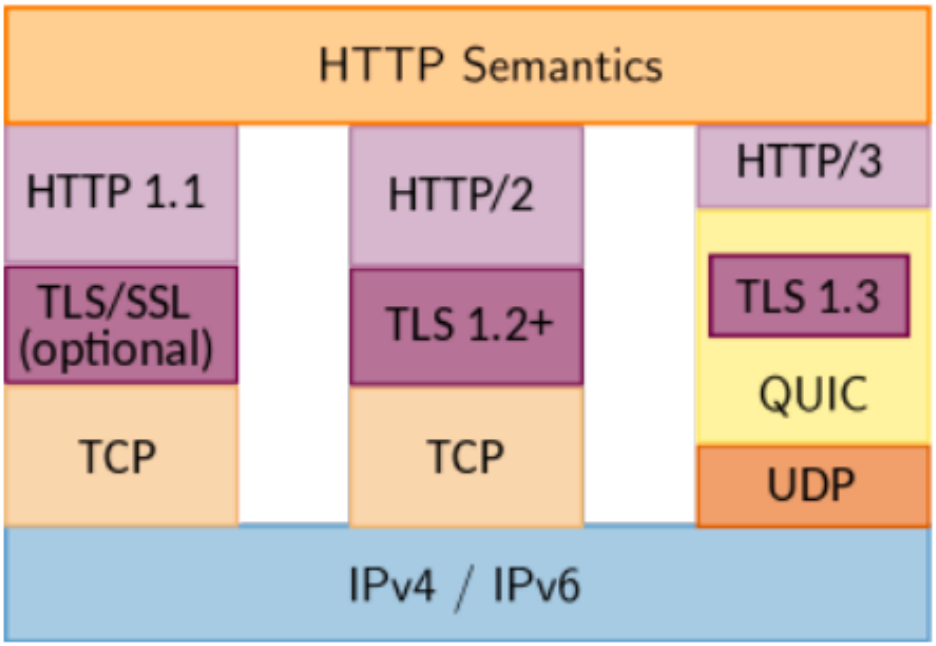


Network layer = Network Access Layer = Network Interface Layer = Link + Física



HTTP 3.0

- Protocolo lanzado con el objetivo de transportar HTTP sobre el nuevo protocolo QUIC (Quick UDP Internet Connections) de Google para mejorar el rendimiento de las aplicaciones web y reducir la latencia, en reemplazo del protocolo TCP.
- Al igual que HTTP 2.0 se conservan métodos y semántica de HTTP 1.1.



Diferencias Principales de HTTP 3.0

- Reduce la latencia en conexiones.
- No depende del manejo de la conexión y controles de TCP, evitando HOL.
- QUIC genera:
    - Nuevos números de secuencia y re-cifra las retransmisiones, permitiendo mejor detección de pérdidas y medición de RTT (Round-Trip Time o Tiempo de Ida y Vuelta).
    - Paquetes cifrados de forma individual, no por conexión/bytestream.
    - Facilidades de movilidad, tiene ID de conexión independiente de IPs y ports.

Desventajas de QUIC

- **Muchos middle-boxes filtran UDP, salvo port 53 y NAT.**
    - La mayoría de Firewalls y Routers tienden a filtrar solo el tráfico UDP en puertos específicos, como el puerto 53 utilizado para DNS. Esto puede dificultar la adopción de QUIC, ya que requiere que el tráfico UDP sea permitido a través de la red.
- **Protocolo de transporte QUIC: implementado usualmente en User-space vs Kernel-space.**
    - QUIC se implementa generalmente en el user-space en lugar del kernel-space. Esto significa que el código de QUIC se ejecuta como parte de las aplicaciones de usuario y no como parte del sistema operativo. La ventaja de esto es que permite una mayor flexibilidad y rapidez en la actualización y desarrollo del protocolo, sin necesidad de modificar el núcleo del sistema operativo. Sin embargo, puede tener un impacto en el rendimiento, ya que las operaciones en el espacio de usuario suelen ser más lentas que en el espacio del kernel.

- **UDP facilita algunos ataques de seguridad.**
    - UDP, el protocolo subyacente de QUIC, es más susceptible a ciertos tipos de ataques de seguridad en comparación con TCP. Por ejemplo, los ataques de suplantación de direcciones (IP spoofing) y los ataques de amplificación (amplification attacks) son más fáciles de realizar con UDP debido a su naturaleza sin conexión y la falta de mecanismos de control de flujo y congestión.

HTTPS

- Sirve para agregar seguridad cifrando la información a la hora de enviarla.
- El protocolo HTTPS provee un canal de comunicación seguro en el que viaja información cifrada de extremo a extremo entre el cliente y el servidor.
    - El cliente puede verificar la autenticidad del sitio web.
    - El servidor puede requerir autenticación del cliente.

Aspectos de Seguridad que se deben garantizar

Autenticidad

- Debemos poder garantizar que nos comunicamos con quien dice ser el que nos envía un mensaje.

Confidencialidad

- Debemos garantizar que solo personas autorizadas puedan acceder al contenido de un mensaje.

Integridad

- Debemos garantizar que solo las personas autorizadas puedan modificar el contenido del mensaje que enviamos, de forma que se mantenga íntegro.

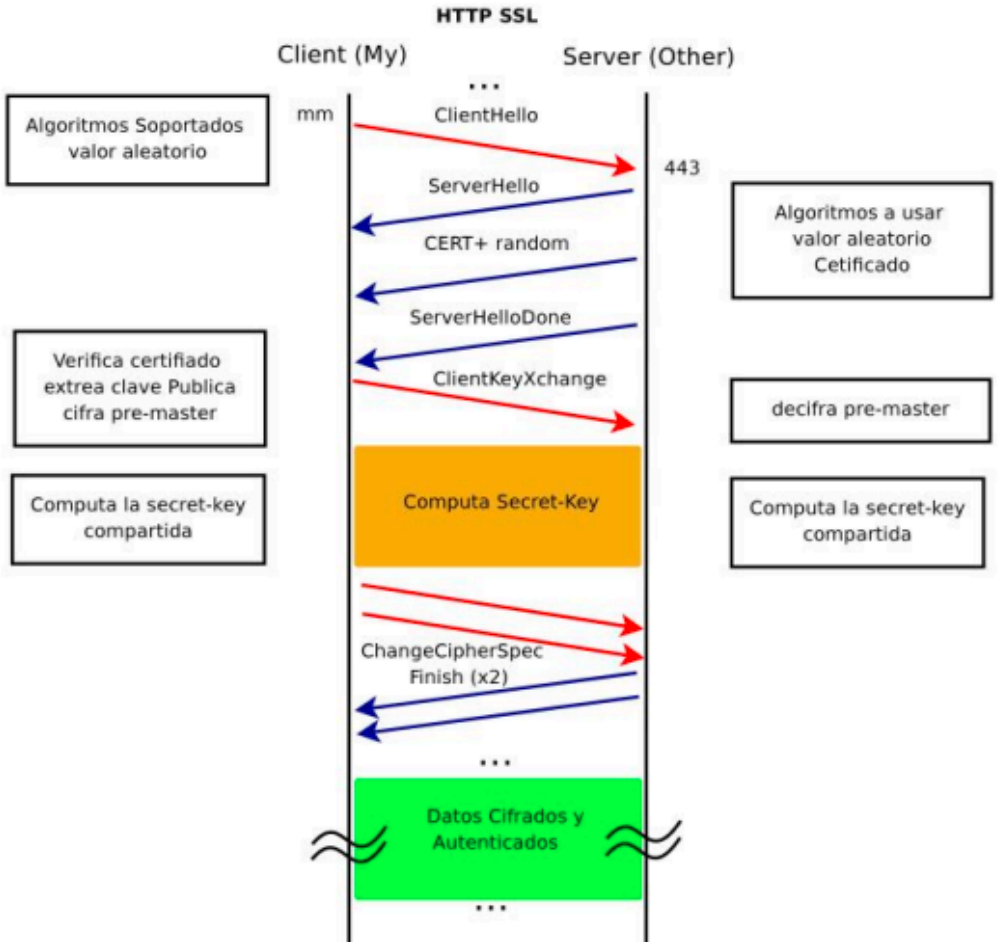
Navegar en un sitio web seguro

Certificado Digital

- Documento que provee datos de la identidad del sitio web y que está firmado por una autoridad de certificación CA que es de confianza y es quien asegura que el par de claves pertenece a determinado sitio web.
- Tiene un periodo de validez.
- Posee la clave pública del sitio web, la privada la posee el servidor.

Proceso de cifrado de datos con HTTPS

1. Se realiza un Handshake, es decir, una negociación entre los algoritmos de seguridad que soportan tanto el cliente como el servidor.
2. El servidor envía el certificado digital al cliente, se realiza un proceso de verificación de la firma del certificado para evaluar la seguridad del contenido del mismo y el cliente termina recibiendo el certificado y junto con él la clave pública del servidor.
3. El cliente usa la clave pública del servidor, crea un canal seguro con el servidor y comparte una clave compartida “clave de sesión” que se cifra junto con la clave del servidor usando criptografía asimétrica, esta clave de sesión es la que se va a usar para descifrar los datos.
4. El servidor desde su lado descifra la clave de sesión con su clave privada usando algoritmos de criptografía simétrica.
5. Se hace envío de toda la información requerida cifrando esta misma con la clave de sesión. Cuando la clave de sesión expira, se repite el proceso para crear una nueva y así poder enviar información de forma segura.



Problemas cuando usamos HTTPS

- Al acceder a un sitio web se pueden presentar problemas que no permiten al navegador asegurar la identidad del sitio visitado. Cuando ocurre esto, el navegador

alerta al usuario, pero es importante entender que el problema es que no se puede asegurar la identidad del sitio visitado.

Formas de identificar un Host

- Dentro de contextos diferentes se debe hallar la mejor manera de identificar a una Host.

Por nombre del Host

- Un nombre de Host como por ejemplo “www.google.com” o “www.facebook.com” son mnemónicos y por lo tanto entendidos por las personas.
- Proporcionan poca o ninguna información acerca de la ubicación del Host dentro de Internet.
- Al poder contener caracteres alfanuméricos de longitud variable, podrían ser difíciles de procesar por los routers.

Por IP

- Una dirección IP consta de 4 bytes y sigue una rígida estructura jerárquica.
    - El aspecto de una dirección IP podría ser por ejemplo “121.7.106.83” donde cada punto separa un byte expresado en notación decimal con valores comprendidos entre 0 y 255.
- Son jerárquicas porque al explorar la dirección de izquierda a derecha, se obtiene información más específica acerca del lugar en el que está ubicado el Host dentro de Internet, es decir, en qué red de la red de redes está.
- Los routers prefieren esta forma de identificación.

Protocolo DNS - Domain Name System

- Es una base de datos distribuida implementada en una jerarquía de servidores DNS y un protocolo de la capa de aplicación que permite a los hosts consultar la base de datos distribuida.
    - Se ejecuta entre sistemas terminales que están en comunicación utilizando el paradigma Cliente/Servidor.
    - Se basa en un protocolo de transporte subyacente extremo a extremo para transferir los mensajes DNS entre los sistemas terminales en comunicación.
    - Usa una base de datos distribuida para mantener una gran cantidad de información, generar delegación y porque una base de datos de este estilo es más tolerante a fallos y más escalable. Tiene un modelo de acceso a la información altamente cacheable.
- Los servidores DNS suelen ser máquinas UNIX que ejecutan el software BIND (Berkeley Internet Name Domain).
- El protocolo DNS se ejecuta sobre UDP y TCP y utiliza el puerto 53.

    - El cliente escoge cualquier puerto no privilegiado.
- No trabaja sobre texto ASCII.

Función principal

- Tiene como función principal ser un servicio de directorio que traduce los nombres de host en direcciones IP.
- Por ejemplo, si un navegador que se ejecuta sobre un host de usuario solicita el URL "www.unaEscuela.edu/index.html" para que el host del usuario pueda enviar un mensaje de solicitud HTTP al servidor "www.unaEscuela.edu" el host del usuario debe obtener la dirección IP de "www.unaEscuela.edu". Esto ocurre de la siguiente manera:
    1) La propia máquina cliente ejecuta el lado del cliente de la aplicación DNS.
    2) El navegador extrae el nombre del host "www.unaEscuela.edu" de la URL y pasa ese nombre al lado del cliente de la aplicación DNS.
    3) El cliente DNS envía una consulta que contiene el nombre del host a un servidor DNS.
    4) El cliente DNS recibe una respuesta que incluye la dirección IP correspondiente al nombre del host.
    5) Una vez que el navegador recibe la dirección IP del servidor DNS, puede iniciar una conexión TCP con el proceso servidor HTTP localizado en el puerto 80 en esa dirección IP.
- Desde el punto de vista de la aplicación (en el caso anterior el navegador) que se ejecuta en el host del usuario, DNS es una caja negra que proporciona un servicio de traducción simple y directo.

Problema

- Añade un retardo adicional, en ocasiones, sustancial a las aplicaciones de Internet que lo utilizan.

Solución

- La dirección IP deseada a menudo se almacena en caché de un servidor DNS "próximo", lo que ayuda a reducir el tráfico de red DNS y el retardo medio del servicio DNS.

Funciones adicionales

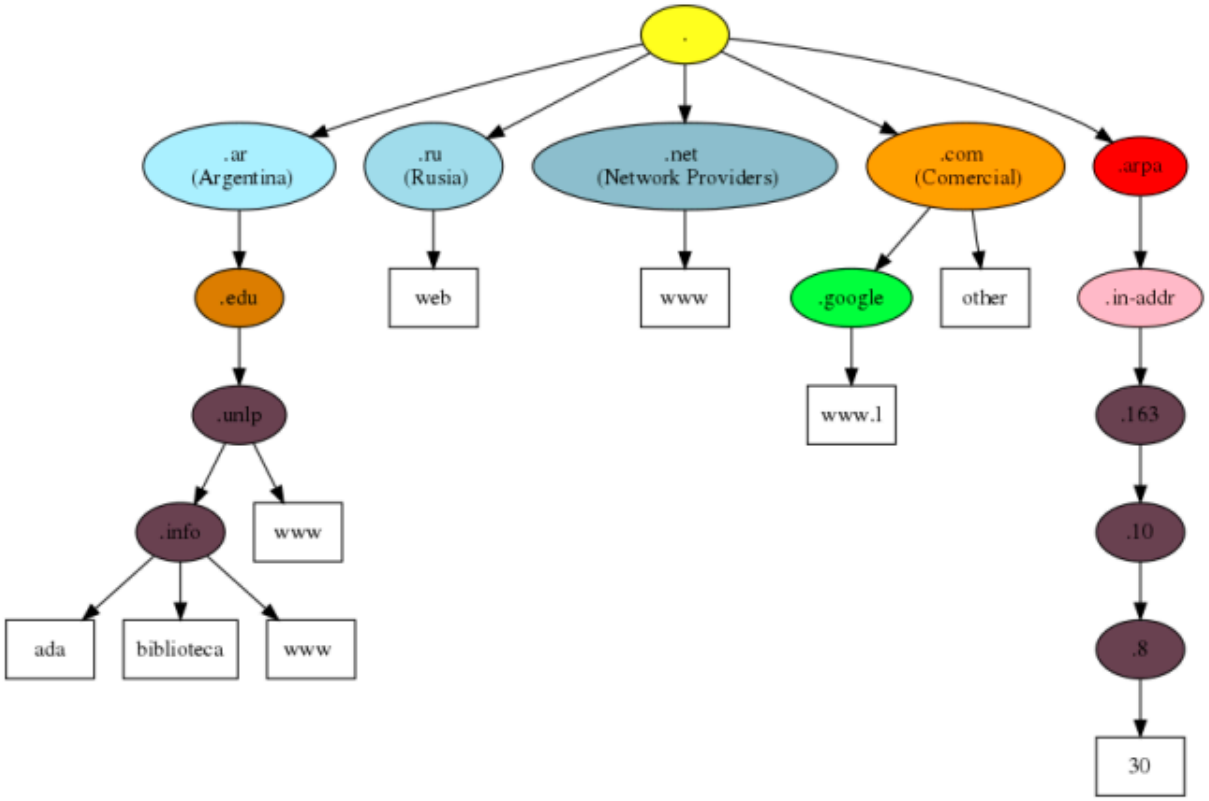
- **Alias de host**
    - Un host con un nombre complicado puede tener uno o más alias que son más mnemónicos que el nombre de host canónico (nombre original). Una aplicación puede invocar DNS para obtener el nombre de host canónico para un determinado alias, así como la IP del host.
- **Alias del servidor de correo**

    - Una aplicación de correo puede invocar al servicio DNS para obtener el nombre de host canónico para un determinado alias, así como la IP del host.
- **Distribución de Carga**
    - DNS también se emplea para realizar la distribución de carga entre servidores replicados, como los servicios web replicados (un mismo sitio web que hace una gran carga de trabajo puede estar replicado en varios servidores, ejecutándose en sistemas terminales distintos y teniendo cada uno una IP diferente). Para los servicios web replicados hay asociado un conjunto de direcciones IP con un único nombre de host canónico. La base de datos DNS almacena ese conjunto de IP 's. Cuando un cliente hace una consulta DNS sobre un nombre relacionado con ese conjunto de direcciones, el servidor responde con el conjunto completo pero rota el orden de las direcciones en cada respuesta. Dado que normalmente un cliente envía su mensaje de solicitud HTTP a la IP que está primera en el conjunto. La rotación hace que se distribuya la carga de trabajo y disminuye el tráfico entre los servidores replicados.

Elementos de DNS

FQDN - Fully Qualified Domain Name

- **Definiciones:**
    - Lista de labels separadas por puntos.
    - Nombre de dominio completo que especifica la ubicación exacta de un host en DNS. Incluye tanto el nombre del host como el nombre del dominio, y puede ser asignado a una dirección IP.
- Se leen desde el nodo/etiqueta de la izquierda hasta la raíz del árbol (el punto), estructura jerárquica con sub-nombres (niveles).



TLDs (Top Level Domains)

- Se pueden clasificar en 3 grupos.

gTLDs (Generic TLDs)

- Contienen dominios con propósitos particulares, de acuerdo a diferentes actividades.
- Estos son los más comunes y no están vinculados a ninguna ubicación geográfica específica. Ejemplos incluyen .com, .net, y .org.
- Se utilizan para una amplia variedad de propósitos y son ideales para sitios web globales.

Unsponsored TLDs (uTLDs)

- Son dominios de nivel superior que no tienen una organización patrocinadora específica que establezca las reglas y políticas para su uso. En cambio, operan bajo

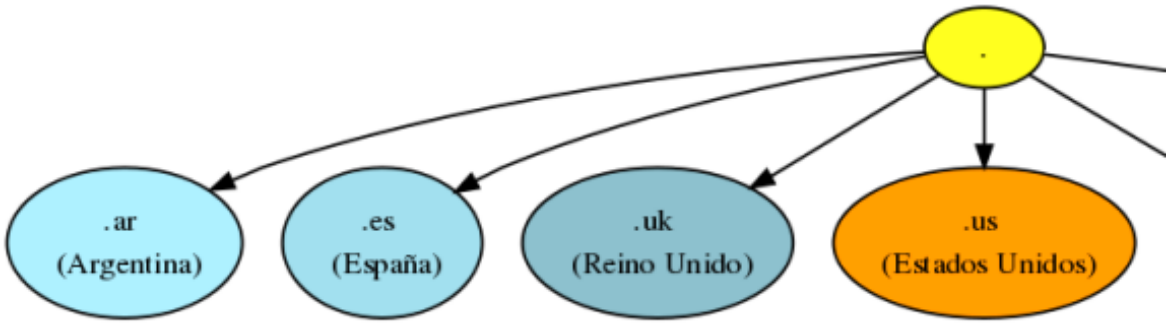
políticas establecidas por la comunidad global de Internet a través del proceso de ICANN (Internet Corporation for Assigned Names and Numbers).

Sponsored TLDs (sTLDs)

- Son dominios de nivel superior que tienen una organización patrocinadora específica que representa a una comunidad particular y establece las reglas y políticas para su uso. Estas comunidades pueden estar basadas en criterios étnicos, geográficos, profesionales, técnicos u otros conceptos temáticos.

ccTLDs (Country-Code TLDs)

- Estos TLDs están asociados a países o territorios específicos.
- Ejemplos incluyen .ar para Argentina, .us para Estados Unidos, y .uk para Reino Unido.
- Son útiles para sitios web con un enfoque local, ya que pueden mejorar la visibilidad en motores de búsqueda locales.



.ARPA (Address and Routing Parameter Area) TLD

- Es un dominio especial, usado internamente para resolución de reversos, es decir, convertir direcciones IP en nombres de dominio **(in-addr.arpa).**
    - Se utiliza principalmente para fines técnicos relacionados con la infraestructura de Internet. Como por ejemplo que el Servidor valide que la IP que está generando una conexión es válida, es decir, validar que esa IP es de un host reconocido por la red en la cuál está corriendo o pertenece.
        - Si la conexión no se valida correctamente el servidor elige si rechaza la conexión o si la deja pasar.
    - Hace que el proceso sea un poco más lento ya que por cada conexión el Servidor tendría que hacer esa validación.
    - Este servicio no se utiliza en el protocolo HTTP. En protocolos como SMTP (emails) o en conexiones locales con SSH si se utiliza.

Registros DNS

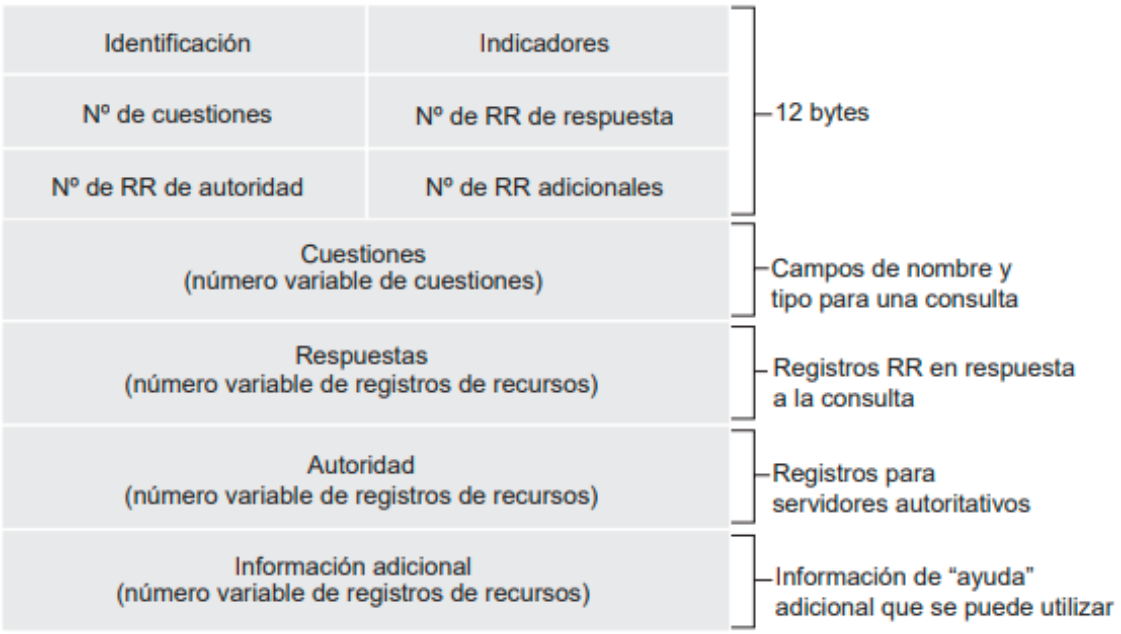
- Los servidores DNS que implementan conjuntamente la base de datos distribuida DNS almacenan los **registros de recursos (RR),** incluyendo los que proporcionan las correspondencias entre nombres de host y dirección IP.
- Un **registro de recurso** está formado por los siguientes 4 campos:
    - (Nombre, Valor, Tipo, TTL).
        - **TTL** → Tiempo de vida del registro de recurso; determina cuándo un registro debería ser eliminado de una caché.
        - El significado de **Nombre** y **Valor** depende del campo **Tipo**.

Valores según Tipo

- Si el **Tipo = A** → **Nombre** es un nombre de host y **Valor** es la dirección IP correspondiente a dicho nombre.
    - Un registro de este tipo proporciona la correspondencia estándar entre nombre de host-dirección IP.
- Si el **Tipo = NS** → **Nombre** es un dominio y **Valor** es el nombre de host de un servidor DNS autoritativo que sabe cómo obtener las direcciones IP de los hosts del dominio.
    - Este registro se utiliza para enrutar las consultas DNS a lo largo de la cadena de consultas.
- Si el **Tipo = CNAME** → **Nombre** es un alias y **Valor** es un nombre de host canónico correspondiente al alias especificado por **Nombre.**
    - Este registro puede proporcionar a los hosts que hacen consultas el nombre canónico correspondiente a un nombre de host.
- Si el **Tipo = MX** → **Valor** es el nombre canónico de un servidor de correo que tiene un alias dado por **Nombre.**
    - Los registros MX permiten a los nombres de host de los servidores de correo tener alias simples.
- Si un servidor DNS es autoritativo para un determinado nombre de host, entonces el servidor DNS contendrá un **registro de tipo A** para el nombre de host. (Incluso aunque el servidor DNS no sea autoritativo, puede contener un **registro de tipo A** en su caché.) Si un servidor no es autoritativo para un nombre de host, entonces el servidor contendrá un **registro de tipo NS** para el dominio que incluye el nombre de host; también contendrá un **registro de tipo A** que proporcione la dirección IP del servidor DNS en el campo Valor del **registro NS.**

Mensajes DNS

- Los mensajes pueden ser de 2 clases: mensajes de respuesta DNS o mensajes de consulta DNS. Ambas clases usan el mismo formato de mensaje.



Formato de los mensajes DNS

- Los primeros 12 bytes constituyen la **sección de cabecera**, la cual contiene una serie de campos.
    - El primero de estos campos es un número de 16 bits que identifica la consulta. Este **identificador** se copia en el **mensaje de respuesta** a la **consulta**, lo que permite al cliente establecer las correspondencias correctas entre las respuestas recibidas y las consultas enviadas.
    - En el campo **Indicadores** se incluyen una serie de indicadores. Un **indicador consulta/respuesta** de 1 bit informa de si el mensaje es una consulta (0) o una respuesta (1). Un **indicador autoritativo** de 1 bit se activa en un mensaje de respuesta cuando un servidor DNS es un servidor autoritativo para un nombre solicitado. El **indicador recursión-deseada**, también de 1 bit, se activa cuando un cliente (host o servidor DNS) desea que el servidor DNS realice una recursión cuando no disponga del registro. En un mensaje de respuesta, el campo de recursión-disponible de 1 bit se activa si el servidor DNS soporta la recursión.
    - En la **cabecera** también se incluyen cuatro campos **"número de"**, que indican el número de apariciones de los cuatro tipos de secciones de datos que siguen a la cabecera.
- La sección **cuestiones** contiene información acerca de la consulta que se va a realizar. Esta sección incluye (1) **un campo de nombre** que contiene el nombre que se va a consultar y (2) **un campo de tipo** que especifica el tipo de cuestión que se plantea acerca del nombre.

- En una **respuesta** de un servidor DNS, la sección **respuestas** contiene los registros del recurso para el nombre que fue consultado originalmente.
- La sección **autoridad** contiene registros de otros servidores autoritativos.
- La sección **información adicional** contiene otros registros útiles.

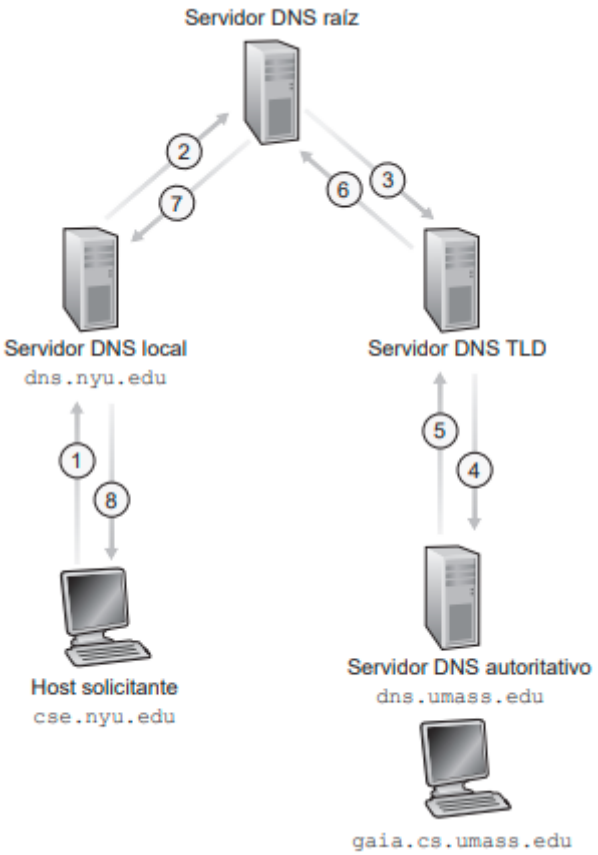
Mensaje de Respuesta DNS

- En estos mensajes viajan uno o más registros de recursos.

Mensaje de Consulta DNS

Consulta Recursiva

- En una consulta recursiva, el cliente DNS solicita a un servidor DNS que resuelva completamente el nombre de dominio. Si el servidor DNS no tiene la respuesta en su caché, se encargará de consultar a otros servidores DNS en nombre del cliente hasta obtener la respuesta completa. Luego, el servidor DNS devuelve la respuesta final al cliente.



Cadena de Consultas Recursivas

Consulta Iterativa

- En una consulta iterativa, el cliente DNS solicita a un servidor DNS que le proporcione la mejor respuesta posible basada en la información que tiene. Si el servidor DNS no

tiene la respuesta completa, devolverá una referencia a otro servidor DNS que pueda tener más información. El cliente DNS entonces repetirá el proceso con el nuevo servidor DNS, y así sucesivamente, hasta obtener la respuesta completa.

Diferencia Principal

- En una **consulta recursiva**, el **servidor DNS** hace todo el trabajo de búsqueda, mientras que en una **consulta iterativa**, el **cliente DNS** sigue las referencias proporcionadas por los **servidores DNS** hasta encontrar la respuesta.

Tipos de servidores DNS

- DNS utiliza un gran número de servidores para abordar el problema del escalado. Estos servidores se organizan de forma jerárquica y están distribuidos en todo el mundo. Ningún servidor DNS dispone de todas las correspondencias de todos los host de Internet. En su lugar, estas correspondencias se distribuyen por los servidores.

Servidores DNS raíz

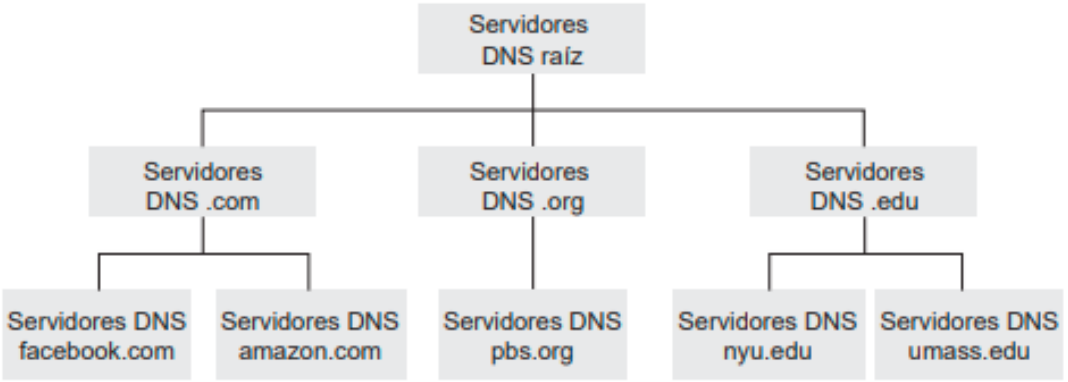
- Servidores que proporcionan las direcciones IP de los servidores TLD. No deberían permitir recursivas.
- Existen unos 400 servidores de nombres raíz distribuidos por todo el mundo.
- Trece organizaciones diferentes se encargan de gestionarlos.

Servidores de dominio de nivel superior (TLD)

- Servidores que proporcionan las direcciones IP de los servidores DNS autoritativos.
- Por cada dominio de nivel superior (TLD), ya sea genéricos como com, org, edu, gov, etc. O de código de país uk, fr, ca, jp, ar, etc. Existe un servidor TLD (o una agrupación de servidores).

Servidores DNS autoritativos

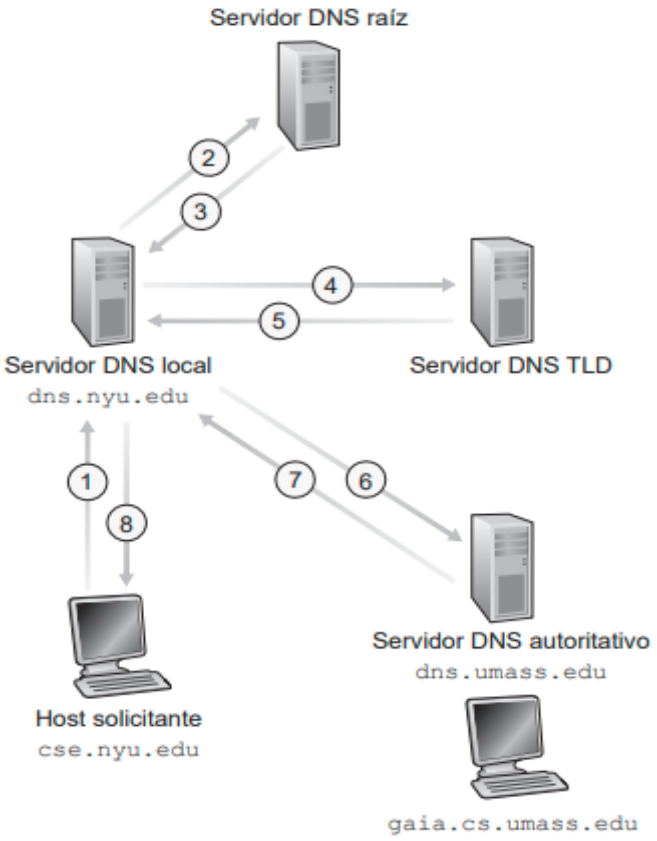
- Servidores con una zona o sub-dominio de nombres a cargo. Podrían sub-delegar.
- Todas las organizaciones que tienen hosts accesibles públicamente a través de Internet deben proporcionar registros DNS accesibles públicamente que establezcan la correspondencia entre los nombres de dichos hosts y sus direcciones IP. Un servidor DNS autoritativo de una organización alberga estos registros DNS.



Parte de la jerarquía de los servidores DNS

Servidor DNS Local/Resolver Recursivo

- Son servidores que son consultados dentro de una red. Mantienen Cache. Pueden ser Servidores Autoritativos. Permiten recursivas “internas”. También son llamados Caching Name Servers.
- Cada ISP (Internet Service Provider) tiene un servidor DNS local. Cuando un host se conecta a un ISP, este proporciona al host las direcciones IP de uno o más de sus servidores DNS locales.
- Cuando un host realiza una consulta DNS, esta se envía al servidor DNS local, que actúa como proxy, reenviando la consulta a la jerarquía de servidores DNS.
- No pertenece estrictamente a la jerarquía de servidores (esa la conforman los 3 anteriores).



Open Name Servers

- Servidores de DNS que funcionan como locales para cualquier cliente.
- Por ejemplo 8.8.8.8, 8.8.4.4, 4.2.2.2, 4.2.2.3.

Forwarder Name Server

- Servidores que interactúan directamente con el sistema de DNS exterior.
- Son DNS proxies de otros DNS internos.

Servidor Primario y Secundario

- Son solo una cuestión de implementación. Donde se modifican los datos realmente.

Protocolo FTP

- **El protocolo FTP (File Transfer Protocol) es un protocolo diseñado para la transferencia de archivos.**
- Es de los más antiguos de Internet.
- Antes de ejecutarse sobre TCP/IP se ejecutaba sobre NCP.

Objetivo

- **El protocolo FTP tiene como objetivo copiar/transferir archivos completos (ya sea desde el servidor al cliente o viceversa)**, a diferencia de otros protocolos que brindan acceso a archivos: NFS, CIFS.

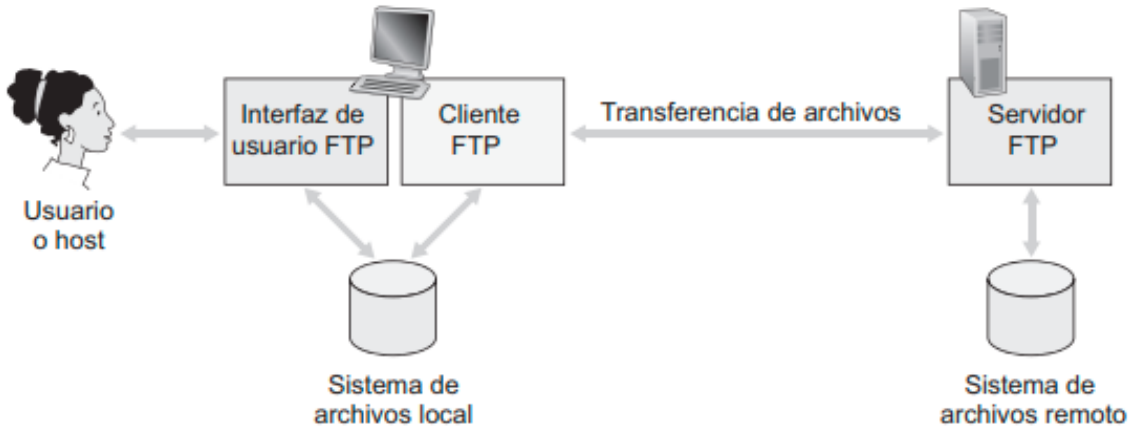
Características

- **Sigue el Modelo Cliente/Servidor.**
- Los mensajes se codifican en **ASCII estándar.**
- **No es seguro por defecto**, FTP transmite datos, incluidas las credenciales, en texto claro (ASCII), lo que lo hace vulnerable a ataques de intercepción. Por esta razón, se ha desarrollado el FTPS (FTP sobre SSL/TLS) y SFTP (Secure File Transfer Protocol), que cifran las transferencias.
    - **TCP le brinda detección de errores y por lo tanto, mayor confiabilidad.**
- **Los clientes FTP no requieren una interfaz gráfica.**

Funcionamiento

- El protocolo FTP usa 2 conexiones TCP paralelas para transferir un archivo, una **conexión para el protocolo de control (Out-Of-Band Control)** y otra **conexión para la transferencia de datos.**
- Cuando un usuario inicia una sesión FTP con un host remoto, el lado del cliente de FTP (usuario) establece en primer lugar una conexión de control TCP con el lado del servidor (host remoto) en el puerto de servidor número 21. El lado del cliente de FTP

envía la identificación y la contraseña del usuario a través de esta conexión de control. El lado del cliente FTP también envía, a través de la conexión control, comandos para modificar el directorio remoto. Cuando el lado del servidor recibe un comando para realizar una transferencia de archivo a través de la conexión de control (bien a, o desde el host remoto), el lado del servidor inicia una conexión de datos TCP con el lado del cliente. FTP envía exactamente un archivo a través de la conexión de datos y luego cierra la conexión de datos. Si durante la misma sesión el usuario desea transferir otro archivo, FTP abre otra conexión de datos. Luego, con FTP, la conexión de control permanece abierta mientras que dure la sesión de usuario, pero se crea una nueva conexión de datos para cada archivo transferido dentro de la sesión (es decir, las conexiones de datos no son persistentes).



- A lo largo de la sesión, el servidor FTP **tiene que mantener un estado del usuario**. En concreto, el servidor debe asociar la **conexión de control** con una cuenta de usuario específica y debe estar al tanto del directorio actual del usuario cuando éste se mueve por el árbol del directorio remoto. **Llevar el control de esta información de estado para cada sesión de usuario activa restringe de forma significativa el número total de sesiones que FTP puede mantener simultáneamente.**

Conexión de Control (Out-Of-Band Control)

- Se emplea para enviar información de control entre los 2 hosts, como la identificación del usuario, la contraseña, comandos para modificar el directorio remoto y comandos para “introducir” (PUT) y “extraer” (GET) archivos.
- Al usar una conexión de control separada, se dice que FTP envía su información de control **fuera de banda.**
- Se establece esta conexión en el puerto de servidor número 21.
- Busca minimizar el delay.
- Transmite el estado de cada operación.

Conexión de Datos

- Conexión que se utiliza para la transferencia de archivos.
- Se crea y se cierra bajo demanda.

- Por esta conexión se transmite exactamente un archivo.
- Se hace en cualquier puerto no privilegiado (cualquiera mayor al 1023).

Comandos

- Los comandos, del cliente al servidor, y las respuestas, del servidor al cliente, se envían a través de la conexión de control en formato ASCII de 7 bits. Por lo tanto, las personas pueden leer los comandos FTP.
- **Algunos de los comandos más comunes:**
    - **USER nombre_de_usuario**
        - Se utiliza para enviar la identificación del usuario al servidor.
    - **PASS contraseña**
        - Se utiliza para enviar la contraseña del usuario al servidor.
    - **LIST**
        - Se utiliza para pedir al servidor que devuelva una lista de todos los archivos existentes en el directorio remoto actual. La lista de archivos se envía a través de una conexión de datos (nueva y no persistente), en lugar de a través de la conexión de control TCP.
        - A nivel de interfaz de usuario el comando que lo inicia es el ls o dir.
    - **RETR nombre_de_archivo**
        - Se utiliza para recuperar (es decir, extraer) un archivo del directorio actual del host remoto. Este comando hace que el host remoto inicie una conexión de datos y envíe el archivo solicitado a través de la conexión de datos.
        - A nivel de interfaz de usuario el comando que lo inicia es el get.
    - **STOR nombre_de_archivo**
        - Se utiliza para almacenar (es decir, introducir) un archivo en el directorio actual del host remoto.
        - A nivel de interfaz de usuario el comando que lo inicia es el put.
    - **DELE**
        - Se utiliza para borrar un archivo del servidor.
    - **SIZE, STAT**
        - Se utilizan para obtener información de un archivo en el servidor.
    - **CD, PWD, RMD, MKD**
        - Se utilizan para cambiar de directorio, obtener el directorio actual, borrar y crear directorios.

Respuestas

- Existe una correspondencia uno a uno entre el comando que ejecuta el usuario y el comando FTP enviado a través de la conexión de control. Cada comando va seguido de una respuesta, enviada desde el servidor al cliente. Las respuestas son números de tres dígitos, con un mensaje opcional que sigue al número.
- **Algunas respuestas:**
    - 150 Here comes the directory listing.
    - 200 Switching to ASCII mode.

    - 200 Switching to Binary mode.
    - 200 NOOP ok.
    - 200 PORT command successful. Consider using PASV.
    - 214 The following commands are recognized.
    - 226 Directory send OK.
    - 230 Login successful.
    - 250 Rename successful.
    - 331 Username OK, password required.
    - 125 Data connection already open; transfer starting.
    - 425 Can’t open data connection.
    - 426 Failure writing network stream.
    - 452 Error writing file.
    - 500 Command not understood.
    - 504 Bad MODE command.
    - 530 Please login with USER and PASS.
    - 550 RNFR command failed.
    - 550 Failed to open file.

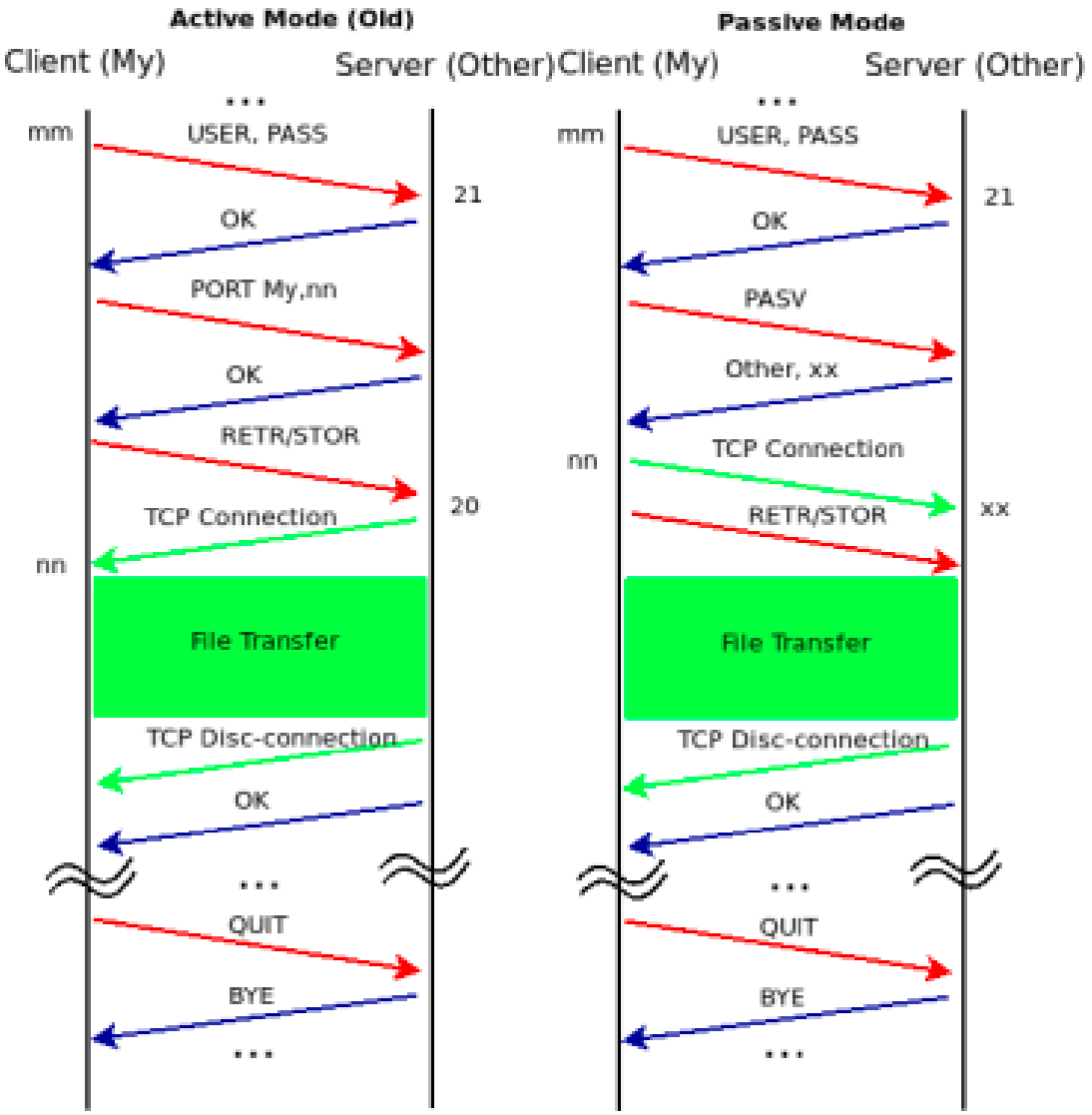
Modalidades de FTP

Modalidad Activa

- Es una **modalidad vieja** en la que cuando el cliente se conecta se establece una **conexión de control** hacia el servidor en el puerto 21 (enviando el comando PORT) y, bajo demanda, se establece un **canal para los datos** donde el servidor desde su puerto 20 se conecta al cliente.
- **El servidor de forma activa se conecta al cliente para generar la conexión de datos hacia el cliente.**
- El **modo activo** puede tener problemas cuando el cliente está detrás de un firewall o un router con NAT, ya que muchas veces estos dispositivos bloquean las conexiones entrantes que no fueron iniciadas desde el cliente.

Modalidad Pasiva

- Modalidad en la que cuando el cliente se conecta se establece una **conexión de control** hacia el servidor en el puerto 21 (enviando el comando PASV) y, **el servidor de forma pasiva responde con un número de puerto no privilegiado** y espera que sea **el cliente** quien abra una **conexión de datos** hacia el puerto indicado.
- **Es más adecuado para redes con firewalls o NAT.** En este modo, el cliente es quien inicia todas las conexiones, lo que evita problemas con firewalls que bloquean conexiones entrantes no solicitadas.
- **El cliente no necesita abrir puertos específicos para recibir conexiones del servidor, ya que es él quien inicia la conexión de datos hacia el servidor.**



Diferencias

Características	Modo Activo	Modo Pasivo
Quién inicia la conexión de datos	El servidor hacia el cliente	El cliente hacia el servidor
Puerto de origen de datos (servidor)	Puerto 20 del servidor	Puerto no privilegiado aleatorio
Comando del Cliente	PORT (indica el puerto en el que escuchará el cliente)	PASV (el servidor envía el puerto donde esperará la conexión)

Formato de Datos (Bytes)

- Debido a los diferentes tipos de plataformas, los archivos pueden ser convertidos a diferentes representaciones.

    - ASCII A NVT-ASCII.
    - EBCDIC E EBCDIC Text.
    - IMAGE I Raw binary, serie de bytes.
    - LOCAL L Raw binary, serie de bytes, usando var. byte size.
- **Es responsabilidad del cliente indicarle al servidor el tipo/formato, sino el default es ASCII, aunque hoy es más común encontrar image.**

Formato de Archivos

- Las plataformas (OS) pueden almacenar archivos en diferentes estructuras. **FTP define estructuras para transportar datos:**
    - **File F Unstructured, sequence of bytes (default).**
    - Record R Series of records.
    - Page P Series of data blocks (pages).
- **Se especifica el formato para la transferencia con el comando STRU.**

Modo de Transferencia

- **MODE** se usa para especificar una codificación adicional aplicada sobre los datos transmitidos, de forma independiente del Formato del Archivo.
    - **Stream S** stream of bytes: Si es R el formato EOF se pone como registro Si es F el formato EOF indica el cierre del stream.
    - **Block B** Archivo se envía como header más secuencia de bloques. Permite interrumpir y reiniciar.
    - **Compressed C** Datos comprimidos usando RLE: Run Length Encoding. BBBBBBNN == 6B2N.

FTP vs HTTP

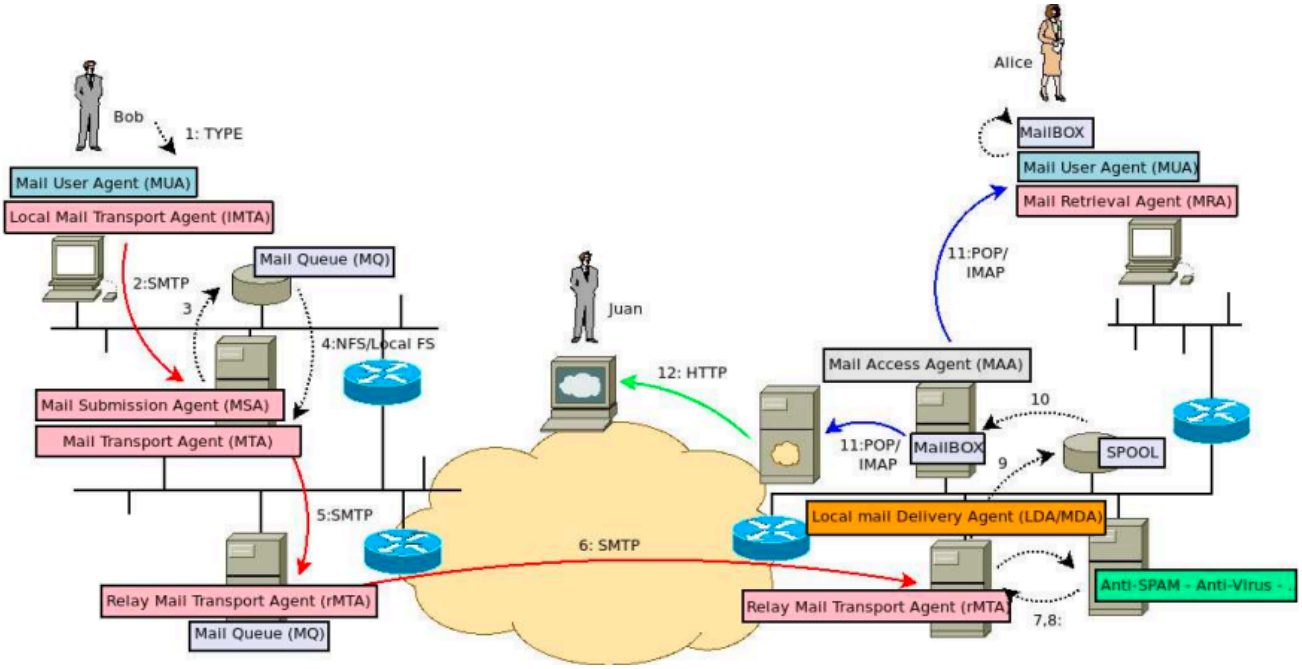
FTP	HTTP
Diseñado para Upload	Se puede con PUT, POST
Soporte de Download	Soporte de Download
Formato ASCII/binary cliente selecciona	meta-data with files, Content-Type, más flexible
No maneja Headers	Manjea Headers, mas información
FTP Command/Response, archivos pequeños más lento	Pipelininig
Más complejo con Firewalls y NAT	Más amigable con Firewalls y NATs
Dos conexiones, y modo Activo o Pasivo	Una conexión, más sencillo
Soportan Ranges/resume	Range/resume HTTP opciones más avanzadas
Soporte de Autenticación	Soporte de Autenticación
Soporte de Cifrado (problemas fwall)	Soporte de Cifrado
Soporte de Compresión RLE	Soporte de Compresión Deflate: LZ77+Huffman

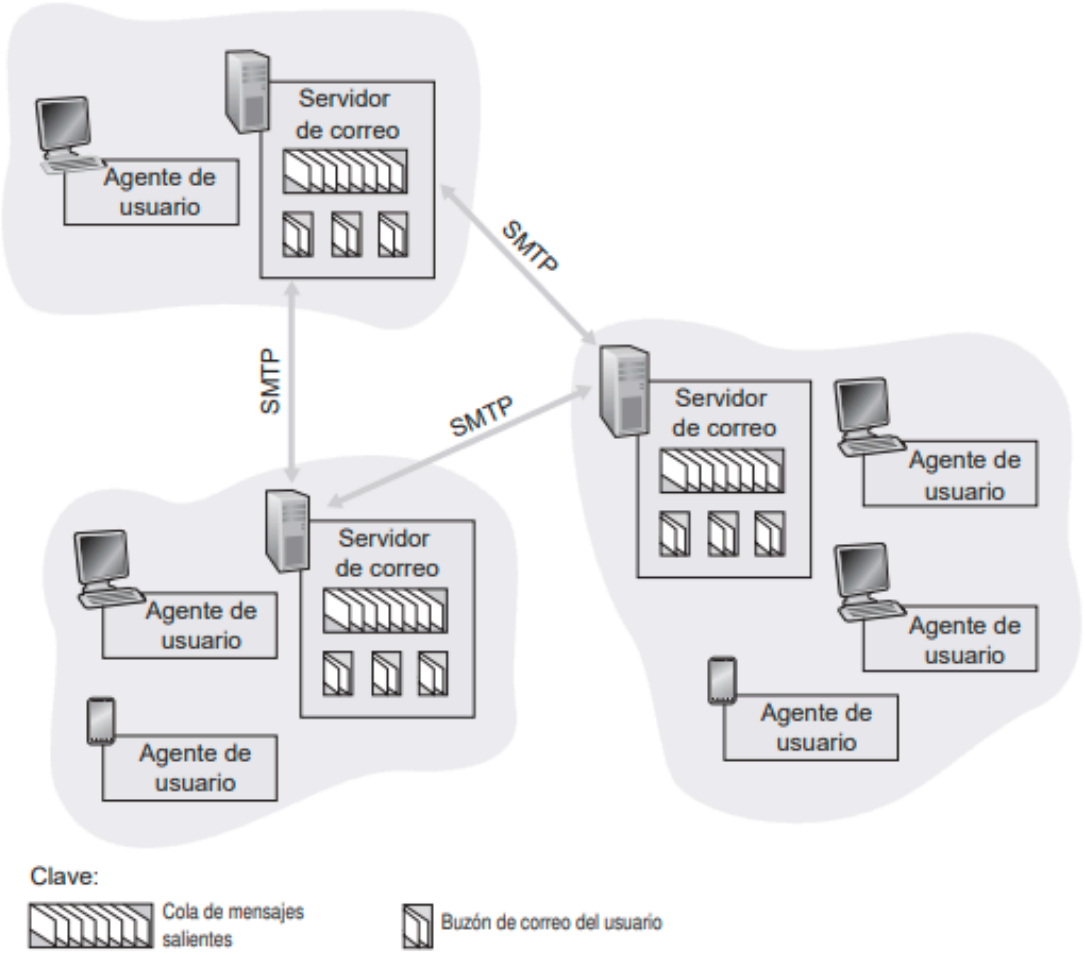
Correo electrónico en Internet

- Al igual que el servicio ordinario de correo postal, el correo electrónico es un medio de comunicación asíncrono (las personas envían y leen los mensajes cuando les

conviene, sin tener que coordinarse con las agendas de otras personas). En contraste con el correo postal, el correo electrónico es rápido, fácil de distribuir y barato.

- Es uno de los primeros servicios de Internet y utiliza conexiones vía módems.
- **Imágenes que muestran la Arquitectura del sistema de correo en Internet:**





Componentes de la Arquitectura del sistema de correo en Internet

MUA - Mail User Agent

- **Cumple el rol de cliente.**
- Es la **interfaz gráfica** que los usuarios utilizan para enviar y recibir correos electrónicos.
- **Es el software que interactúa directamente con el usuario final y permite redactar, enviar, recibir y leer mensajes.**
- Posee integrado un local MTA (Mail Transport Agent) para comunicarse con el Servidor de Mail Saliente, MTA que hace relay.
- El MTA que hace relay para el usuario pre-procesa el e-mail recibido desde el MUA con el agente MSA (Mail Submission Agent).
- Agrega la mayoría de los campos del header: Message-ID, To:, From:, Date:, Subject:, etc.
- Posee integrado un MRA para comunicarse con el Servidor de Mail Entrante, MAA (Mail Access Agent).
- **Utilizan protocolos SMTP o ESMTP, POP o IMAP. Habla con MTA para enviar correos y con el MAA para recibirlos.**

MTA - Mail Transport Agent

- **Cumple el rol de cliente y de servidor.**
- Encargado de recibir, enrutarse y enviar correos entre servidores. **Este componente se encarga del transporte de los correos electrónicos a través de Internet, comunicándose con otros servidores hasta que el mensaje llega al destinatario final.**
- Toma el e-mail desde el MSA o directamente desde el MUA (MSA integrado).
- Determina el destino del correo y lo reenvía al MTA correspondiente.
- Almacena temporalmente el correo saliente.
- Se encarga de recibir y almacenar temporalmente los mensajes para las casillas que sirve desde el MTA remoto.
- **Utiliza protocolo SMTP o ESMTP, entre servidores MTA.**

MDA - Mail Delivery Agent / LDA - Local Delivery Agent

- **Cumple el rol de Servicio interno o Servidor.**
- **Es el componente encargado de recibir los correos del MTA y entregarlos en el buzón de correo del usuario local**.
- Habitualmente define el formato del mailbox.
- Servicio de MDA puede hacer delivery remoto, cumple rol de MAA.
- Se integra con los protocolos POP y/o IMAP dejando los recursos disponibles al MAA o directamente al usuario.

MAA - Mail Access Agent

- **Cumple el rol de Servidor.**
- **Es el componente que autentica al MUA y permite al mismo acceder a sus correos electrónicos almacenados en el servidor.**
- **Permite la interacción del MUA con el servidor para que el usuario pueda leer o descargar sus correos.**
- Puede estar integrado o separado del MDA.
- Implementa los protocolos POP y/o IMAP dejando acceder a los recursos y dialogando con el MUA.

MRA - Mail Retrieval Agent

- **Es el encargado de obtener el correo del servidor remoto y entregarlo al MUA.** Este componente generalmente interactúa con servidores que utilizan IMAP o POP3 para recuperar los correos y presentarlos al cliente de correo del usuario.
- **Actúa como intermediario entre el servidor de correo y el MUA para obtener mensajes.**

MSA - Mail Submission Agent

- **Cumple el rol de servidor.**
- Se encarga de aceptar correos electrónicos desde el MUA, pre-procesarlos y entregarlos al MTA para su envío. En otras palabras, e**l MSA es el componente**

responsable de procesar los correos que salen del usuario (cliente) y pasarlos al sistema de transporte de correos (MTA).
- Habitualmente está integrado en el MTA.
- Agrega campos que pueden faltar, formato del header: Message-ID, To:, From:, Date:, etc. Y le termina de dar formato al header del e-mail.
- **Utiliza protocolo SMTP, ESMTP.**
- Usaba el port 25 pero ahora se recomienda usar 587.

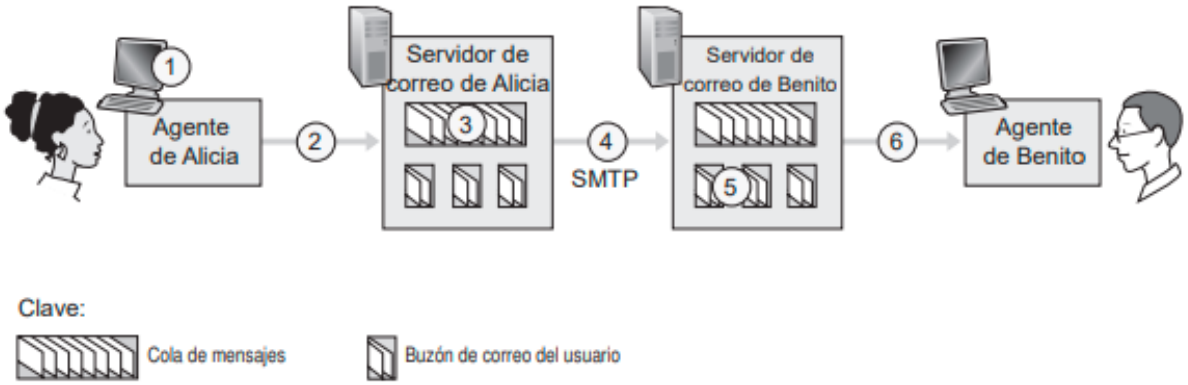
Protocolo SMTP

- **SMTP (Simple Mail Transport Protocol)** es el **protocolo principal de la capa de aplicación para el correo electrónico por Internet.**
- Utiliza el servicio de transferencia de datos fiable de **TCP** para transferir el correo desde el servidor de correo del emisor al servidor de correo del destinatario (puerto servidor 25).
- **Sigue el modelo cliente/servidor por lo tanto tiene 2 lados:**
    - **Lado del Cliente**
        - Se ejecuta en el servidor de correo del emisor.
    - **Lado del Servidor**
        - Se ejecuta en el servidor de correo del destinatario.
    - Ambos lados se ejecutan en todos los servidores de correo. Cuando un servidor de correo envía mensajes de correo a otros servidores de correo, actúa como un cliente SMTP. Cuando un servidor de correo recibe correo de otros servidores, actúa como un servidor SMTP.
- Restringe el cuerpo (no sólo las cabeceras) de todos los mensajes a formato **ASCII de 7 bits. Esto causa los siguientes problemas**
    - Requiere que los datos binarios multimedia se codifiquen a ASCII antes de ser transmitidos a través de SMTP.
    - Requiere que el correspondiente mensaje ASCII sea decodificado de vuelta a binario una vez realizado el transporte SMTP.
- Los LMTA de los MUA hablan SMTP con su servidor SMTP saliente.
- Los servidores SMTP hablan entre sí este protocolo.
- **El protocolo no usa servidores de correos intermedios para enviar correos**, es decir, si el servidor emisor se encuentra en Hong Kong y el receptor en Argentina, la conexión TCP será directa entre los 2 servidores.
- **Usa conexiones persistentes.**
- Trabaja de forma Interactiva (Requerimiento/Respuesta) o Pipeline.
- Puede o no requerir autenticación.
- Puede o no trabajar de forma segura: SSL/TLS.

Funcionamiento

- En primer lugar, el cliente SMTP (que se ejecuta en el host servidor de correo emisor) establece una conexión TCP con el puerto 25 del servidor SMTP (que se ejecuta en el host servidor de correo receptor).
    - Si el servidor no está operativo, el cliente lo intentará más tarde.

- Una vez que se ha establecido la conexión, el servidor y el cliente llevan a cabo el proceso de negociación de la capa de aplicación (al igual que las personas, que antes de intercambiar información se presentan, los clientes y servidores SMTP se presentan a sí mismos antes de transferir la información).
    - Durante esta fase de negociación SMTP, el cliente SMTP especifica la dirección de correo electrónico del emisor (la persona que ha generado el mensaje) y la dirección de correo electrónico del destinatario.
- Una vez que el cliente y el servidor SMTP se han presentado a sí mismos, el cliente envía el mensaje. SMTP cuenta con el servicio de transferencia de datos fiable de TCP para transferir el mensaje al servidor sin errores.
- El cliente repite entonces este proceso a través de la misma conexión TCP si tiene que enviar otros mensajes al servidor; en caso contrario, indica a TCP que cierre la conexión.

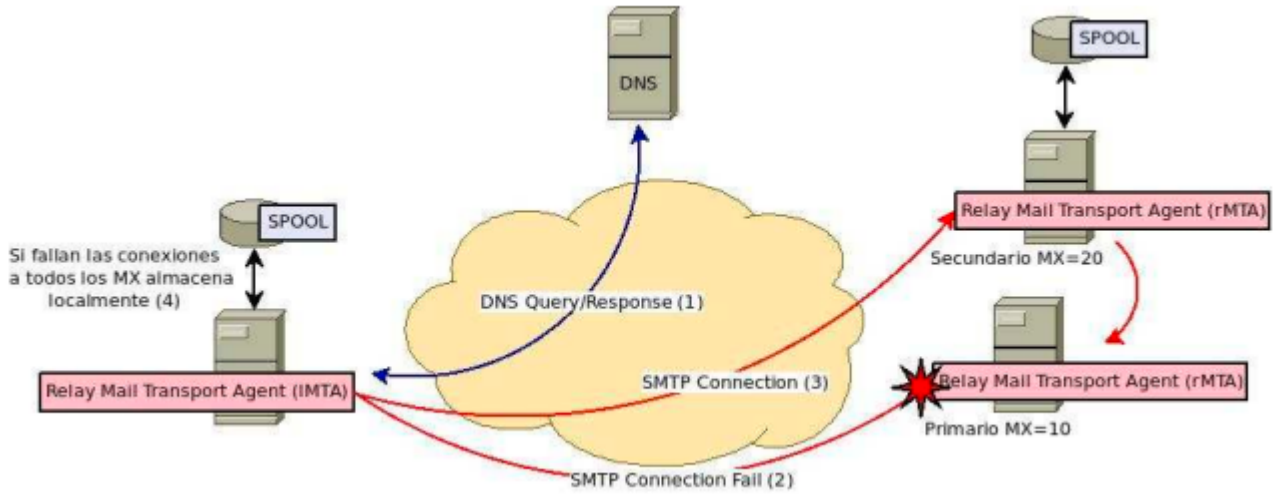


Diálogo de correos

- **El cliente en el diálogo ejecuta 5 comandos (se explican solos):**
    - HELO (abreviatura de HELLO).
    - MAIL FROM.
    - RCPT TO.
    - DATA.
    - QUIT.
- El cliente también envía una línea que **consta únicamente de un punto**, que indica el **final del mensaje para el servidor**. (En la jerga ASCII, cada mensaje termina con CRLF.CRLF, donde CR y LF se corresponden con el retorno de carro y el salto de línea, respectivamente.).
- El servidor responde a cada comando, teniendo cada una de las respuestas un código de respuesta y una explicación (opcional) en inglés.

```
S:    220 hamburger.edu
C:    HELO crepes.fr
S:    250 Hello crepes.fr, pleased to meet you
C:    MAIL FROM: <alicia@crepes.fr>
S:    250 alicia@crepes.fr ... Sender ok
C:    RCPT TO: <benito@hamburger.edu>
S:    250 bob@hamburger.edu ... Recipient ok
C:    DATA
S:    354 Enter mail, end with "." on a line by itself
C:    ¿Te gusta el ketchup?
C:    ¿Y los pepinillos en vinagre?
C:    .
S:    250 Message accepted for delivery
C:    QUIT
S:    221 hamburger.edu closing connection
```

SMTP y DNS



Formato de los mensajes de correo

- Para el formato de los mensajes es necesario definir el encabezado y el cuerpo del mismo.
- **Encabezado**
    - Todo Encabezado tiene que estar formado por una **línea de cabecera From:** y una **línea de cabecera To:**; también puede incluir una **línea Subject:,** así como otras líneas de cabecera opcionales.
- Después del encabezado se incluye una **línea en blanco.**
- Luego de la línea en blanco se incluye el **cuerpo del mensaje** (en ASCII).

Versión Extendida

- MIME (Multipurpose Internet Mail Extensions) brinda extensiones para enviar datos binarios.

- Se usan tags especiales que indican el tipo al sistema destino.
- Luego se codifica el mensaje binario en formato que no viole US-ASCII (NVT).

```
From: bob@other.test
To: alice@cities.org
Subject: Test Mail MIME
Date: Thu, 11 Aug 2005 20:20:56 -0300 (ART)
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded image .....
.........................
......base64 encoded image
```

Protocolos de acceso para correo electrónico

- **Los protocolos de acceso para correo electrónico más usados son POP3 e IMAP, ambos:**
    - Requieren autenticación.
    - Utilizan formato ASCII de 7 bits.
    - Usan TCP en los puertos servidor: 110 (POP3) y 143 (IMAP).
    - Permiten correr de forma segura sobre SSL/TLS.

POP3

- **POP3 (Post Office Protocol—Version 3)** es un protocolo de acceso a correo **simple**, por lo tanto su **funcionalidad es limitada.**
- POP3 se inicia cuando el agente de usuario (el cliente) abre una conexión TCP en el **puerto 110** al servidor de correo (el servidor). Una vez establecida la conexión TCP, POP3 pasa a través de tres fases:
    - **Autorización**
        - Durante esta fase el agente de usuario envía un nombre de usuario y una contraseña (en texto legible) para autenticar al usuario.
        - Tiene dos comandos principales: **user <nombreUsuario> y pass <contraseña>.**
    - **Transacción**
        - Durante esta fase el agente de usuario recupera los mensajes; también durante esta fase, el agente de usuario puede marcar los mensajes para borrado, eliminar las marcas de borrado y obtener estadísticas de correo.
        - Un agente de usuario que utilice POP3 suele ser configurado (por el usuario) para **“descargar y borrar”** o para **“descargar y guardar”**. La

secuencia de comandos que ejecute un agente de usuario POP3 dependerá de en cuál de estos dos modos esté operando.
        - En el modo **descargar y borrar**, el agente de usuario ejecutará los comandos **list, retr y dele**. Tiene como **problema** que no guarda los mensajes en el servidor de correo después de que se hayan descargado.
        - En el modo **descargar y guardar**, el agente de usuario deja los mensajes en el servidor de correo después de que se hayan descargado.
    - **Actualización**
        - Esta fase tiene lugar después de que el cliente haya ejecutado el comando **quit**, terminando la sesión POP3; en este instante, el servidor de correo borra los mensajes que han sido marcados para borrado.
- En una transacción POP3, el agente de usuario ejecuta comandos y el servidor devuelve para cada comando una respuesta. Existen dos posibles respuestas:
    - **+OK** (seguida en ocasiones por una serie de datos servidor-cliente), utilizada por el servidor para indicar que el comando anterior era correcto.
    - **-ERR**, utilizada por el servidor para indicar que había algún error en el comando anterior.
- **Durante una sesión POP3 entre un agente de usuario y el servidor de correo, el servidor POP3 mantiene cierta información de estado**; en concreto, mantiene la relación de los mensajes de usuario que han sido marcados para ser borrados. Sin embargo, **el servidor POP3 no conserva la información de estado de una sesión POP3 a otra.**

IMAP

- El protocolo **IMAP (Internet Mail Access Protocol)** es un protocolo de acceso a correo que ofrece muchas **más funcionalidades** que POP3 pero es más **complejo**.
- **Un servidor IMAP asociará cada mensaje con una carpeta;** cuando un mensaje llega al servidor, se asocia con la carpeta **INBOX (Bandeja de entrada**) del destinatario, el cual puede entonces pasar el mensaje a una nueva carpeta creada por el usuario, leer el mensaje, borrarlo, etc.
- Proporciona comandos para crear carpetas, mover mensajes entre carpetas, buscar mensajes que cumplan con ciertos criterios en carpetas remotas, etc.
- **Un servidor IMAP mantiene información acerca del estado a lo largo de las sesiones IMAP**, como por ejemplo, los nombres de las carpetas y los mensajes asociados con cada una de ellas.
- Dispone de comandos que permiten a un agente de usuario obtener partes componentes de los mensajes.