

## Capa de Aplicación – Resumen

PDU ⇒ Mensaje

La capa de aplicación es la capa superior en la estructura de la red y es donde se encuentran las aplicaciones y sus protocolos. En Internet, esta capa abarca protocolos como HTTP, que se utiliza para solicitar documentos web, SMTP, que se utiliza para el correo electrónico, y FTP, que se utiliza para la transferencia de archivos. Además, en esta capa se lleva a cabo la conversión de nombres comprensibles por humanos a direcciones IP de 32 bits a través del Sistema de Nombres de Dominio (DNS), que es un protocolo de capa de aplicación.

Un protocolo de la capa de aplicación define cómo los procesos de una aplicación, que se ejecutan en distintos sistemas terminales, se pasan los mensajes entre sí. En particular, un protocolo de la capa de aplicación define:

- Los tipos de mensajes intercambiados; por ejemplo, mensajes de solicitud y mensajes de respuesta.
- La sintaxis de los diversos tipos de mensajes, es decir, los campos de los que consta el mensaje y cómo se delimitan esos campos.
- La semántica de los campos, es decir, el significado de la información contenida en los campos.
- Las reglas para determinar cuándo y cómo un proceso envía mensajes y responde a los mismos.

### Funciones

- Define el formato de los mensajes: Existen protocolos que trabajan de forma binaria, por ejemplo usando ASN y otros en forma textual ASCII como HTTP.
- Define la semántica de cada uno de los mensajes.
- Define como debe ser el dialogo (intercambio de mensajes). Que mensajes se deben intercambiar.
- Ejemplo concreto: Protocolo HTTP y sus implementaciones mediante servidores WEB y browsers (navegadores).

### Comunicación

Si dos procesos deben comunicarse:

- En la misma maquina: Cuando los procesos se ejecutan en el mismo sistema terminal, pueden comunicarse a través de sistemas de comunicación interprocesos que aplican reglas establecidas por el sistema operativo del sistema terminal.

- En distintas maquinas: Dos procesos de sistemas terminales diferentes se comunican intercambiando mensajes a través de una red de computadoras. Un proceso es el emisor, crea y envía mensajes, y el otro proceso es el receptor, recibe estos mensajes y puede responder con mensajes propios.

### Modelos de Comunicación de Aplicaciones

- **Modelo Mainframe (dumb client):** Modelo de **carga centralizada**. Es un sistema donde **todo el procesamiento ocurre en una computadora central** (mainframe, el servidor) y los terminales (clientes) solo corren la comunicación y la interfaz física con el usuario. El mainframe es el que decide cuando le da el control al cliente. El mainframe maneja el dialogo de las comunicaciones
- **Modelo Cliente/Servidor:** Modelo de **carga compartida**. El **cliente solicita servicios o recursos a un servidor central, que los proporciona**. El servidor corre servicio esperando de forma pasiva la conexión, mientras que el cliente se conecta al servidor y se comunica a través de este. **La idea inicial es que el cliente pone procesamiento de interfaz y el servidor el resto del procesamiento.**
- **Modelo Peer to Peer (P2P):** Modelo de **carga compartida y distribuida**. Una arquitectura de red donde los participantes (peers) **pueden actuar como clientes y servidores al mismo tiempo** para compartir recursos entre ellos. Es un sistema **escalable en cuanto a rendimiento pero no en cuanto a la administración.**
- **Modelo Híbrido:** Modelo de carga compartida y distribuida. Al igual que P2P los participantes pueden actuar como clientes y servidores al mismo tiempo para compartir recursos entre ellos, la diferencia recae en que existen diferentes tipos de nodos con diferentes roles. Hay nodos centrales donde se registra la información y al resto de los nodos. Es un sistema escalable en cuanto a rendimiento y *supongo que* mejor en administración que P2P puro.

### Requerimientos

Cada aplicación puede tener diferentes requerimientos: seguridad, tiempo de respuesta, confiabilidad, optimizar ancho de banda. De acuerdo a estos requerimientos estará asociada a determinado protocolo transporte.

### User Agent

Es una interfaz entre el usuario y la aplicación de red.

### HTTP

**HTTP** se implementa mediante dos programas, **cliente y servidor** (modelo cliente/servidor) que se comunican entre sí intercambiando mensajes HTTP. **HTTP define la estructura de estos mensajes y cómo el cliente y el servidor intercambian los mensajes.** HTTP define **cómo los clientes web solicitan páginas web a los servidores y cómo estos servidores web transfieren esas páginas a los clientes.** Utiliza **TCP** como su **protocolo de transporte.**

El cliente HTTP primero inicia una conexión TCP (puerto 80 como default, el cliente escoge cualquier puerto no privilegiado) con el servidor. Esto asegura que los mensajes se transmitan sin pérdida de datos, sin que HTTP se preocupe por pérdidas. HTTP es un protocolo sin estado.

HTTP Trabaja sobre texto ASCII, permite enviar información binaria con encabezados MIME.

- Clientes (llamados browsers o navegadores): Firefox, IE, Opera, Safari, Chrome.
- Servidores: Apache Server, MS IIS, NGINX, Google GWS, Tomcat.

## HTTP VS HTML

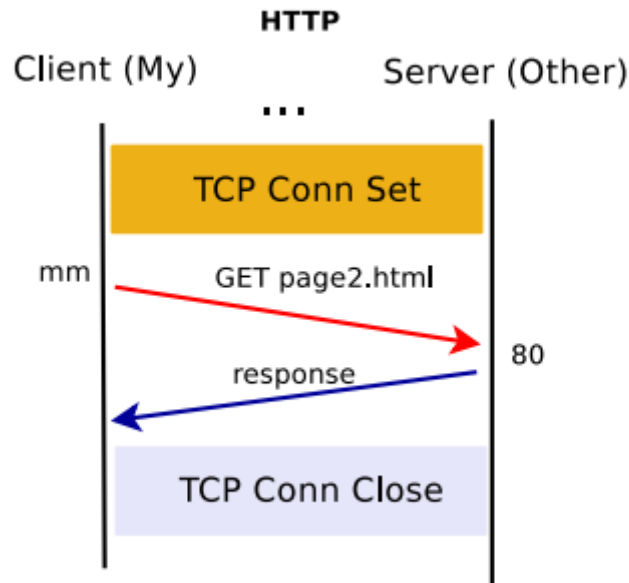
HTML es el lenguaje de marcado utilizado para crear la estructura y el contenido de las páginas web. La mayoría de las páginas web están constituidas por un archivo base HTML y varios objetos referenciados. Por ejemplo, si una página web contiene texto HTML y cinco imágenes JPEG, entonces la página web contiene seis objetos: el archivo base HTML y las cinco imágenes. El archivo base HTML hace referencia a los otros objetos contenidos en la página mediante los URL de los objetos. Cada URL tiene dos componentes: el nombre de host del servidor que alberga al objeto y el nombre de la ruta al objeto.

El navegador web (cliente) es el que interpreta y renderiza el código HTML para mostrar la página web al usuario y el servidor web es el encargado de enviar los archivos HTML. El contenido HTML se encuentra en la entidad de respuesta, y es lo que el servidor envía al cliente como parte de la respuesta a la solicitud HTTP.

El envío y recibimiento de este mismo se hace a través de mensajes respetando el protocolo HTTP.

## HTTP 0.9

- Nunca se estandarizó
- Pasos para obtener un documento:
  1. Establecer la conexión TCP
  2. HTTP Request vía comando GET.
  3. HTTP Response enviando la página requerida.
  4. Cerrar la conexión TCP por parte del servidor.
  5. Si no existe el documento o hay un error directamente se cierra la conexión.



- Solo una forma de Requerimiento.
- Solo una forma de Respuesta.
- Request/Response sin estado.

Request ::= GET <document-path> <CR><LF>

Response ::= ASCII chars HTML Document.

GET /hello.html <CR><LF>

GET / <CR><LF>

## HTTP 1.0

- Versión de HTTP 1.0 estándar [RFC-1945]
- Define formato basado en HTTP 0.9
  - o Se debe especificar la versión en el requerimiento del cliente.
  - o Para los Request, define diferentes métodos HTTP.
  - o Define **códigos de respuesta**.
  - o Admite repertorio de caracteres, además del ASCII, como: ISO-8859-1, UTF-8, etc.
  - o **Admite MIME** (No solo sirve para descargar HTML e imágenes).
  - o **Por default NO utiliza conexiones persistentes.**
  - o Todo se basa en HEADER (básicamente permiten al cliente y al servidor enviar información adicional junto a una petición o respuesta, son un par clave:valor)

*Formato Request*

<Method> <URI> <Version>  
[<Headers Opcionales>]  
<Blank>  
[<Entity Body Opcional>]  
<Blank>

<Method HTTP 1.0> ::= GET, POST, HEAD, PUT,  
DELETE, LINK, UNLINK

#### *Formato Response*

<HTTP Version> <Status Code> <Reason Phrase>  
[<Headers Opcionales>]  
<Blank>  
[<Entity Body Opcional>]

#### *Ejemplo*

GET /index2.html HTTP/1.0  
User-Agent: telnet/andres (GNU/Linux)  
Host: estehost.com  
Accept: \*/\*

HTTP/1.1 200 OK  
Date: Mon, 21 Apr 2008 00:28:51 GMT  
Server: Apache/2.2.4 (Ubuntu)  
Last-Modified: Mon, 21 Apr 2008 00:18:14 GMT  
ETag: "a3b36-1f-91d5d80"  
Accept-Ranges: bytes  
Content-Length: 31  
Connection: close  
Content-Type: text/plain

```
<HTML>
<H1> HOLA </H1>
...
</HTML>
```

## Métodos HTTP

- ✧ GET: obtener el documento requerido. Puede enviar información, pero no demasiada. Es enviada en la URL del requerimiento. Con el formato ?var1=val1&var2=val2.... La cantidad e información está restringida al tamaño de la URL. En general, 256 bytes.
- ✧ HEAD: idéntico a GET, pero solo requiere la metainformación del documento, por ejemplo, su tamaño. No obtiene el documento en sí.
- ✧ POST: hace un requerimiento de un documento, pero también envía información en el body. Generalmente, usado en el fill-in de un formulario HTML(FORM). Puede enviar mucha más información que un GET.
- ✧ PUT: usado para reemplazar un documento en el servidor. En general, deshabilitada. Utilizado, por ejemplo, por protocolos para compartir archivos y carpetas montados sobre HTTP, como WebDAV [WDV].
- ✧ DELETE: usado para borrar un documento en el servidor. En general, deshabilitada. También, puede ser utilizada por WebDAV.
- ✧ LINK, UNLINK: establecen/des-establecen relaciones entre documentos.

Los mensajes (operaciones) PUT, DELETE, LINK, UNLINK son pasados directamente a la aplicación si el servidor web los soporta. PUT/DELETE No se implementan directamente en el servidor. Se deben agregar como una extensión CGI para manejarlos. La diferencia de estos dos con POST es que el archivo que se indica puede no existir y el script de procesamiento es el mismo.

## Ejemplos de respuestas HTTP/1.0

HTTP/<version> 200 OK

HTTP/<version> 301 Moved Permanently

HTTP/<version> 400 Bad Request

HTTP/<version> 403 Access Forbidden

HTTP/<version> 404 Not Found

HTTP/<version> 405 Method Not Allowed

HTTP/<version> 500 Internal Server Error (CGI Error)

HTTP/<version> 501 Method Not Implemented

## Multiplexación

Mediante el parámetro Host se pueden multiplexar varios servicios sobre un mismo host. Esto quiere decir que permite que un único servidor web responda a solicitudes para múltiples dominios o servicios en el mismo host

## Autenticación

HTTP/1.0 contempla autenticación con WWW-Authenticate Headers. El servidor, ante un requerimiento de un documento que requiere autenticación, enviara un mensaje 401 indicando la necesidad de autenticación y un Dominio/Realm. El navegador solicitara al usuario los datos de user/password (si es que no los tiene cachedos) y los enviara en texto claro al servidor. El servidor dará o no acceso en base a esos valores. Para los siguientes requerimientos, el navegador usara los valores que tiene almacenados para el Realm solicitado.

## Conexión no persistente vs Conexión persistente

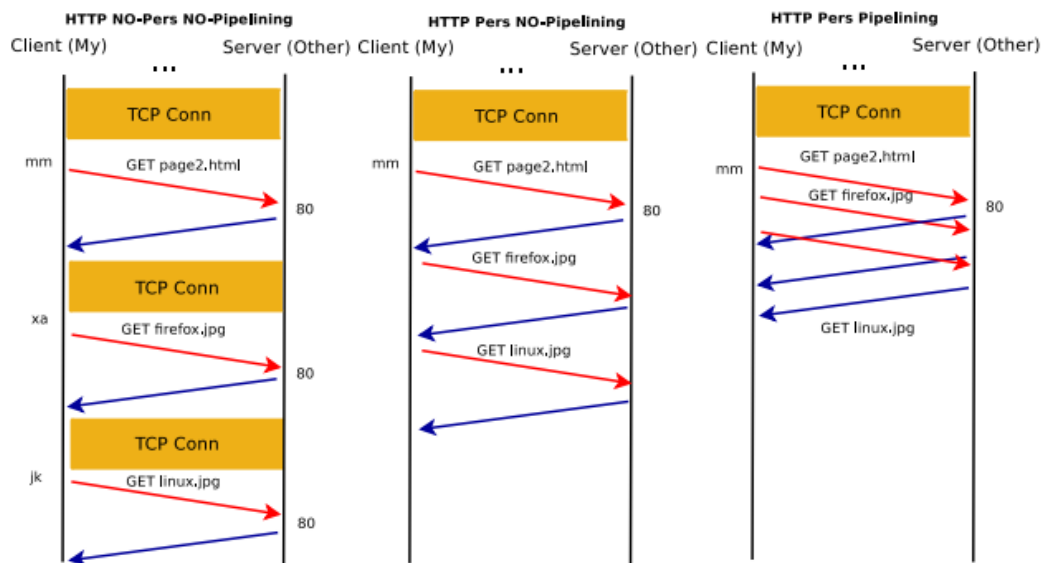
Cada recurso se solicita por separado al servidor. Si se tiene una conexión no persistente, por cada recurso se abre y cierra la conexión TCP. Esto implica una sobrecarga asociada a la apertura y cierre de conexiones, lo que conduce a una carga más lenta de las páginas web.

En cambio, si se tiene una conexión persistente, no se abre y cierra la conexión TCP por cada transacción.

Sin embargo, hay un problema asociado con la conexión persistente, especialmente cuando se trata de recursos grandes o de larga duración. Si un recurso se mantiene en la conexión durante un período prolongado, podría bloquear la conexión y retrasar la obtención de otros recursos. Esto se conoce como el problema de head-of-line blocking. Para abordar este problema, los navegadores modernos utilizan técnicas como la multiplexación y el pipelining para permitir la transferencia de múltiples recursos a la vez. Aun así Pipelining requiere que los responses sean enviado en el orden solicitado, HOL posible

## HTTP 1.1

- HTTP/1.1, 1997 con la [RFC-2068] se actualiza con [RFC-2616].
- Nuevos mensajes HTTP 1.1: OPTIONS, TRACE (utilizada para debugging) , CONNECT (utilizada para generar conexiones a otros servicios montadas sobre HTTP).
- Conexiones persistentes por default.
- Pipelining, mejora tiempo de respuestas.
  - o No necesita esperar la respuesta para pedir otro objeto HTTP.
  - o Solo se utiliza con conexiones persistentes.
  - o Mejora los tiempos de respuestas.
  - o Sobre la misma conexión de debe mantener el orden de los objetos que se devuelven.
  - o Se pueden utilizar varios threads para cada conexión



## Redirects

Los redirect son mensajes que se utilizan para enviar al user-agent (browser) a otra ubicación debido a que el recurso ha sido movido. Se realiza un Redirect temporal 302, indicando la nueva URL/URI, donde se encuentra la respuesta. La respuesta debe darse en el campo Location y/o en un HTML corto

El user-agent no debería re-direccionarlo salvo que el usuario confirme, aunque varios browser lo hacen. Moved Permanently 301, se indica que cualquier acceso futuro debe realizarse sobre la nueva ubicación (mejora Indexadores). Se pueden generar problemas con Cookies.

## CGI y Javascript

- CGI (Common Gateway Interface): aplicación que interactúa con un servidor web para obtener información de los requerimientos generados por el cliente y así poder responder. Todo el procesamiento de CGI es realizado del lado del servidor: server-side, el cliente solo genera los datos y los envía para que luego sean procesados.
- Javascript: client-side script (ejecuta del lado del cliente). Usa modelo de objetos DOM, permiten extensiones como AJAX (Asynchronous JavaScript And XML). Este hace requerimientos particulares y no necesita recargar toda la página.

## Cookies

Las cookies, definidas en [RFC 6265] son un mecanismo que permite a las aplicaciones web del servidor "manejar estados". Habitualmente son utilizadas para:

- Autenticación por aplicación.
- Carritos de compras (shopping carts).



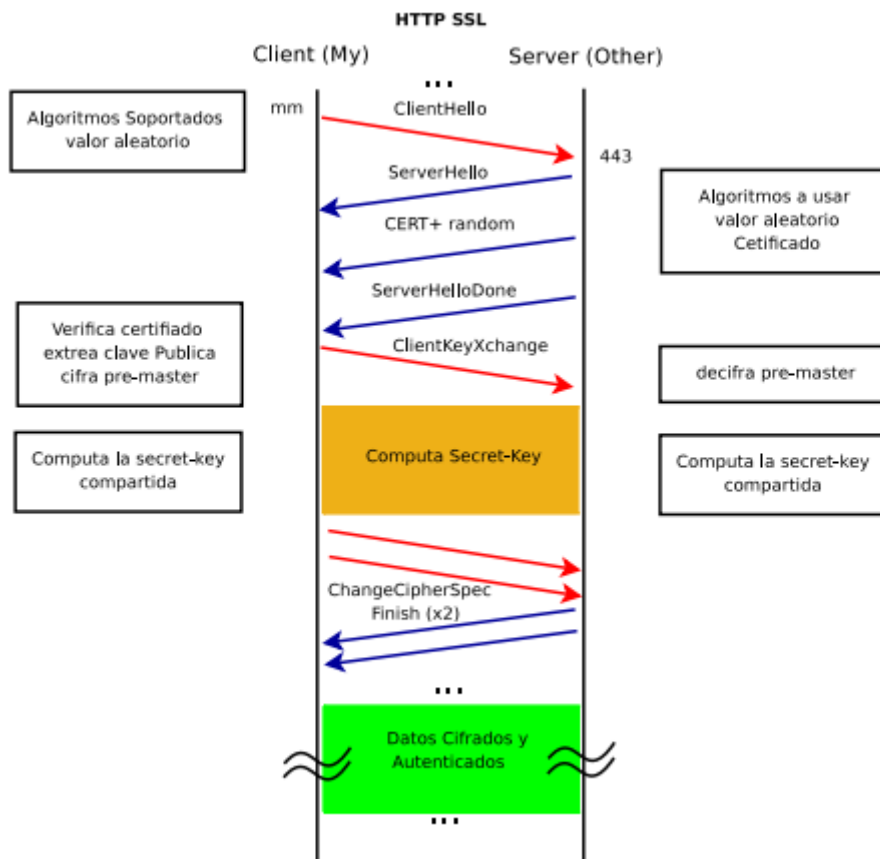
- Preferencias, recomendaciones.
- Estado de sesión de usuario (user session state, e.g. web-email).

### ¿Cómo funcionan?

El cliente hace un request, el servidor retorna un recurso (un objeto HTTP, como una página HTML) indicando al cliente que almacene determinados valores por un tiempo (esta cookie es introducida al cliente mediante el mensaje en el header Set-Cookie: mensaje que indica un par nombre,valor e incluye fecha de expiración). Después el cliente en cada requerimiento luego de haber almacenado la Cookie se la enviara al servidor con el header Cookie:. El servidor puede utilizarlo o no y el servidor puede borrarlo. Las cookies se almacenan por site y puede haber varias por c/u.

## HTTPS

Utiliza el port 443 por default. Hay una etapa de negociación previa, después de esto se cifra y autentica todo el mensaje HTTP (incluso el header).



## Web-Cache

Sirve para "Proxiar" y Chachear recursos HTTP.

Objetivos:

- Mejorar tiempo de respuesta (reducir retardo en descarga).

- Ahorro recursos de la red.
- Balance de carga, atender a todos los clientes.

Se solicita el objeto, si está en cache y esta “fresco” se retorna desde allí (HIT). Si el objeto no esta o es viejo se solicita al destino y se cachea (MISS). Se puede realizar control de acceso.

Del lado de cliente, los web browser tienen sus propias cache locales. Los servidores agregan los headers Last-Modified: date ETag: (entity tag) hash y hay requerimientos condicionales desde los clientes: If-Modified-Since: date If-None-Match: hash. El servidor puede responder 304 Not Modified (si por ejemplo se solicita If-Modified-Since: date y no esta modificado) o 200 OK (si sí lo esta).

Del lado del servidor, los cache como servers funcionan como Proxy. Son servidores a los clientes y clientes a los servidores web. Los instalan ISP o redes grandes que desean optimizar el uso de los recursos. En general corren sobre UDP.

Protocolos de comunicación entre web-cache servers:

- ICP (Internet Cache Protocol).
- (HTCP) Hyper Text Caching Protocol.

## HTTP 2

Es un reemplazo de como HTTP se transporta, no es un reemplazo completo, se conservan métodos y semánticas. Es un protocolo binario en lugar de ASCII y la mayoría está montado en TLS

### Problemas con HTTP 1.1

- Pipelining requiere que los responses sean enviado en el orden solicitado, HOL posible.
- POST no siempre pueden ser enviados en pipelining.
- Demasiadas conexiones generan problemas, control de congestión, mal uso de la red.
- Muchos requests, muchos datos duplicados (headers)

### Solución HTTP 2

- Multiplexa varios request en una petición en lugar de ser una secuencia ordenada y bloqueante.

HTTP 2 permite a los servidores “pushear” datos a los clientes (servidor enviar recursos adicionales al cliente antes de que este los solicite de manera explícita)

HTTP hace uso del **stream**, que es un canal virtual bidireccional que opera dentro de una única conexión TCP. Cada solicitud y respuesta se intercambia a través de un stream separado. Cada stream posee un ID y una prioridad asignada. Los mensajes HTTP 2 (solicitudes y respuestas) se transmiten utilizando estos streams. Los mensajes HTTP 2 se dividen en **frames** que se envían a través del mismo stream.

Básicamente, los streams son identificados y desglosados en frames, que constituyen la unidad de comunicación más pequeña. Estos frames se transmiten y se vuelven a ensamblar en el destino. Existen varios tipos de frames, como los de encabezado, de datos y de prioridad. Estos frames se combinan para formar las solicitudes y respuestas que se intercambian en los streams. Todos los streams coexisten en una misma conexión.

Frame types: HEADERS, DATA, PUSH\_PROMISE, WINDOW\_UPDATE, SETTINGS,

El mismo stream puede ser usado para llevar diferentes mensajes.

- Los streams dentro de una misma conexión tienen flow-control individual.
- Los streams pueden tener un weight (prioridad).
- Los streams pueden estar asociados de forma jerárquica, dependencias.

### Headers en HTTP/2

No se codifican más en ASCII. Surgen nuevos pseudo-headers que contiene información que estaba en el método y otros headers.

Por ejemplo: HEAD /algo HTTP/1.1 se reemplaza con HTTP 2:

:method: head

:path: /algo

:scheme: https o http

:authority: www.site.com reemplaza al headerHost:.

Para las respuestas: :status: códigos de retornos 200, 301, 404, etc.

Puede haber compresión de encabezados SPDY/2 propone usar GZIP. GZIP + cifrado, tiene “bugs” utilizados por atacantes. Se crea un nuevo compresor de Headers: HPACK. H2 y SPDY, soportados en la mayoría de los navegadores

### Negociación de protocolo

Con HTTP 1, si tanto el cliente como el servidor soportan el mismo protocolo, se puede negociar hacer un “upgrade”. Hay un Upgrade Header. Connection: Upgrade, HTTP2-Settings Upgrade: h2c|h2.

En HTTP 2 (y HTTP 3) se puede negociar con ALPN (extensión de TLS) entre el cliente y el servidor el protocolo de aplicación que utilizarán durante la comunicación segura. En la negociación ALPN, el servidor envía una lista de protocolos de aplicación que admite. Los clientes pueden seleccionar uno de los protocolos ofrecidos, como "h2" para HTTP/2, "h3" para HTTP/3 y "http/1.1" para HTTP/1.1.

## HTTP 3

HTTP 3 utiliza QUICK(UDP) como protocolo de transporte.

## DNS

El servicio de DNS (Domain Name System) tiene como objetivo principal traducir nombres de dominio a direcciones de Internet (direcciones IP) y, de esta forma, lograr una abstracción de las direcciones de red utilizadas internamente por los protocolos. Esto permite ubicar a un dispositivo por su nombre sin importar cual es la dirección IP que tiene asignada en ese momento. Otra ventaja es que las personas no necesitan recordar las direcciones IP.

*Mapeo de nombre a dirección.*

Funciona como un sistema distribuido de forma jerárquica, a través de dominios, sub-dominios y nombres finales, con un conjunto de servidores a lo largo del mundo. Cada servidor tiene la responsabilidad de mantener una parte dentro de la jerarquía de nombres.

Tiene un Modelo cliente/servidor, Request/Response (También hay dialogo entre los servidores, uno pasa a ser el cliente temporalmente). Corre sobre UDP y TCP en el puerto 53. No Trabaja sobre texto ASCII. Si el mensaje supera los 512 bytes se utiliza TCP, e.g. zone transfer (EDNS permite mayor cantidad de datos)

### Aspectos y Elementos de DNS

- Espacio de nombres (sintaxis y zonas, dominios) (mencionado arriba).
- Procedimiento de Delegación y Arquitectura.
- Base de datos distribuida y Servidores.
- Define las componentes y el protocolo para su comunicación.
- Procedimiento de búsqueda/resolución (Protocolos).

Se encarga de definir un nombre de dominio FQDN: lista de etiquetas (labels) separadas por puntos.

- ➔ Los nombres de los dominios se leen desde el nodo/etiqueta de la izquierda hasta la raíz del árbol (el punto), estructura jerárquica con sub-nombres (niveles).
- ➔ La sintaxis jerárquica refleja la delegación de autoridad.
- ➔ No son case-sensitive, cada etiqueta Máximo 63 chars.
- ➔ Max etiquetas 127, nombre max. 255 chars, acepta valores internacionales, UTF-8, Unicode

### Organización

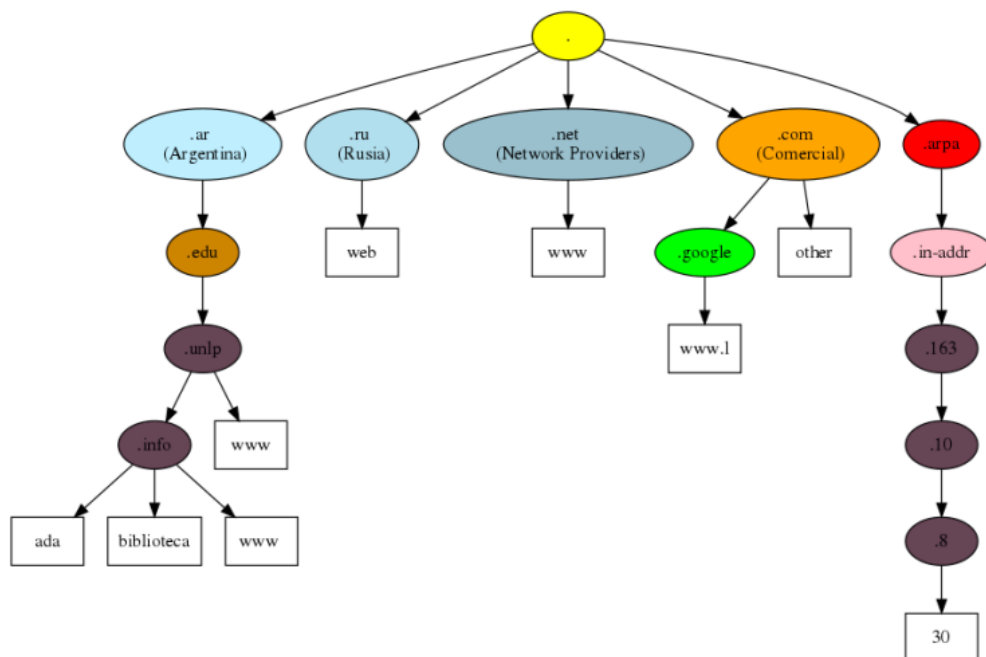
Es un sistema distribuido pero regido por organizaciones. Tiene una organización mediante dominios, sub-dominios y host o servicios (jerárquico).

Los nombres se delegan a países y a registrars, direcciones IP no.

Se tiene básicamente una “base de datos distribuida”. Esto permite:

1. Mantener gran cantidad de información.
2. Control de la información está distribuido (delegación entre zonas).
3. Tolerancia a fallos y escalable.
4. Tener un modelo de acceso altamente cacheable.

### Esquema de nombres



### Root Servers

Los root servers son los encargados de proporcionar las direcciones IP de los Top Level Domains. Es el punto de inicio (Bootstrap). Hay en la actualidad 13 ROOT Servers distribuidos en todo el mundo, 7 de los cuales trabajan con redundancia y las réplicas están distribuidos geográficamente. Proporcionan redundancia, combinada con Ruteo Anycast. Este

Básicamente lo que hace es delegar a todos TLD. No debería permitir recursivas.

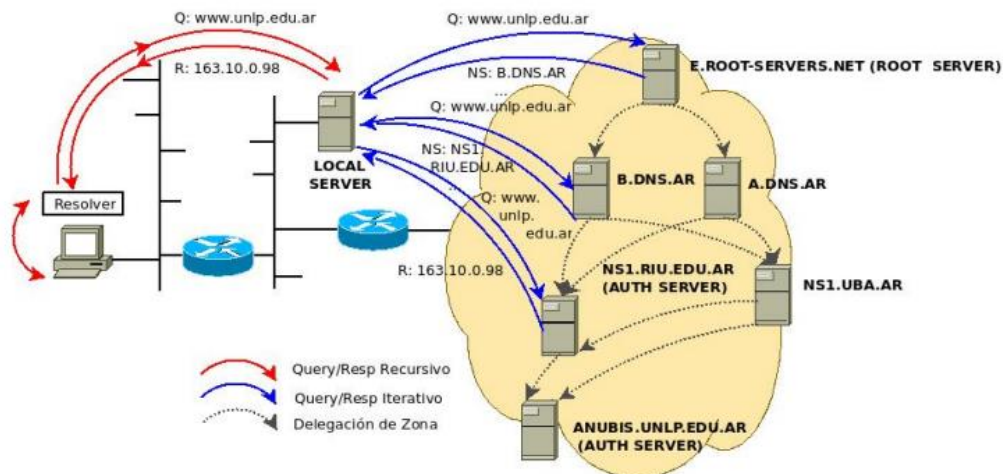
El que voy a terminar utilizando es el que me da menor tiempo de respuesta.

### TLDs (Top Level Domains)

Los TLDs (la parte más alta de la jerarquía luego de la raíz) se podrían clasificar en 3 grupos:

- **gTLDs, Generic TLDs**: Son dominios con **propósitos particulares**, de acuerdo a diferentes actividades políticas definidas por el ICANN (Unsponsored TLD) o definidas por otra organización (Sponsored TLD). Son los dominios que básicamente se utilizan para identificar categorías amplias y distintos tipos de sitios web. Algunos ejemplos de gTLDs incluyen **.com**, **.org**, **.net** y **.info**.
- **ccTLD Country-Code TLDs**: contienen dominios **delegados a los diferentes países del mundo**.
- **.ARPA TLD**: es un dominio especial, usado internamente para resolución de reversos.

## Resolución de nombres



## Tipos de Servidores

- **Servidor autoritativo**: servidor con una zona o sub-dominio de nombres a cargo. Podría sub-delegar si se tratase de un dominio del cual no está a cargo.
  - o **Respuesta autoritativa**: Una respuesta autoritativa **es aquella dada por el servidor que tiene la autoridad sobre el nombre que se está consultando**. Este **responde directamente desde su base de datos de nombres**, sin subdelegaciones ni cacheo de direcciones. **Caso contrario**, si se realiza esto último, **se trata de una Respuesta NO Autoritativa**
- **Servidor Local/Resolver Recursivo**: es un servidor que es consultado dentro de una red. mantiene cache. Puede ser Servidor Autoritativo. Permite recursivas "internas". También llamado Caching Name Server.
  - o Al Resolver se lo podría considerar como un agente encargado de resolver los nombres a solicitud del cliente (dentro de mi maquina). Se puede tener un Stub/Dumb Resolver que no realiza ninguna forma de caching y deja que el encargado de esto sea el Servidor Local o un

resolver activo, llamado Smart Resolver, que funciona en cada equipo como si fuese un Servidor Local, realizando caching u ofreciendo funcionalidades extras. Este suele hacer consultas recursivas.

- **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:** interactúan directamente con el sistema de DNS exterior. Son DNS proxies de otros DNS internos.

Servidor Primario y Secundario: en los servidores DNS hay uno que es el servidor primario, la razón de que sea así es para simplificar la configuración de los servidores autoritarios, evitando configurar a cada servidor de un mismo dominio de forma independiente. En lugar de esto, se configura aquel que es primario y el resto de los servidores se sincronizan con este. Esto a su vez garantiza la consistencia de los datos DNS.

La copia de la base de datos de nombres de un servidor primario a uno secundario se llama "transferencia de zona". Esto permite mantener la consistencia entre los servidores de una zona de dominio.

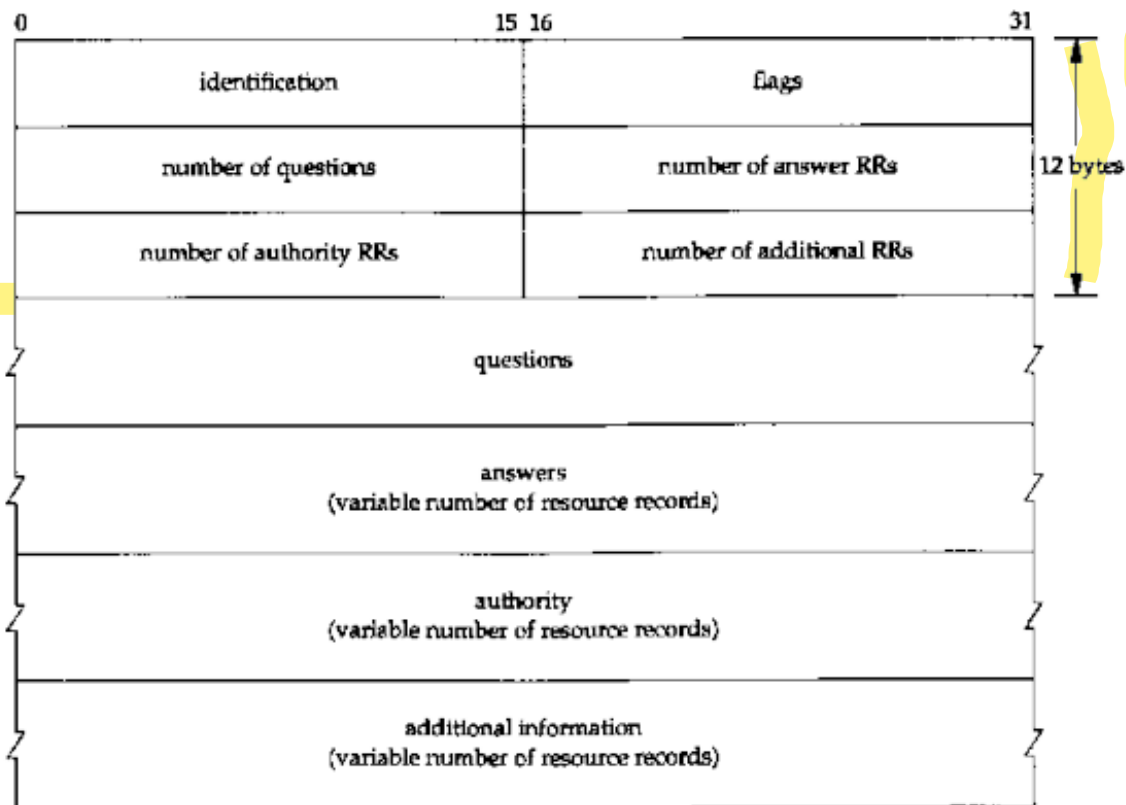
#### Consulta recursiva vs iterativa

- En una consulta recursiva, el servidor DNS consultado se encarga de resolver toda la consulta en nombre del cliente, realizando consultas adicionales si es necesario y proporcionando una respuesta completa.
- En una consulta iterativa, el servidor DNS consultado proporciona una respuesta parcial con información sobre dónde buscar más detalles, y el cliente debe seguir el proceso de consulta paso a paso para obtener la respuesta completa.
  - o Si consulto de forma recursiva a alguien que responde de forma iterativa, me responderá de forma iterativa.

#### ¿Cómo funciona la resolución de nombres?

El cliente (navegador) consulta primero al resolver privado de forma recursiva, si esta no está cacheada, el resolver privado delegará al local server (sigue siendo recursivo). Si no está cacheada este consultará de forma iterativa con el root server. Después de ahí habrá una delegación de zonas (siempre con consultas iterativas) hasta finalmente dar con el servidor autoritativo del dominio que se está consultando, el servidor local obtendrá la dirección IP (la cachea), se la dará al resolver privado (la cachea) y este al cliente.

## Estructura mensajes



El mensaje de DNS tiene un encabezado fijo y un cuerpo variable

### Identificador

valor aleatorio generado por el cliente y copiado por el servidor en la respuesta. Valor entre 0 y 64K-1. Usado por el cliente para asociar las consultas con las respuestas. Además, el cliente puede usar diferentes números de puertos UDP para agregar entropía a las consultas. También se lo conoce como Transaction ID.

### Flags

Se codifican varias indicaciones. Indica si es una consulta o una respuesta (QR), si es autoritativa o no (AA), si la respuesta está truncada o no (TC) -no entra en un mensaje UDP, deberá usarse TCP-, si es recursiva o no (RD) habilitada por el cliente, recursión permitida o no (RA) habilitada por el servidor, código de operación, si es estándar una notificación, un update, etc.

### Contadores varios

De cantidad de consultas, respuestas, autoritativas o no que lleva el mensaje.

### Cuerpo



Respuestas y/o consultas, más datos adicionales. Dentro de las preguntas y respuestas se pueden trabajar con diferentes tipos de información.

### Tipos de registros DNS

Un servidor de DNS almacena la información con la que trabaja en una base de datos (DB) Dentro de la base de datos, la información se organiza en registros llamados Resource Record (RR). Cada uno de estos registros puede guardar diferente tipo de información.

- **A:** mapean un nombre de dominio a una dirección IPv4. Pueden existir varios registros (A) con el mismo nombre

```
berlin.cities.org.      IN      A      172.20.1.100
brasilia.cities.org.    IN      A      172.20.1.5
paraguil-br0.cities.org. IN      A      172.20.1.1
```

- **NS:** indican los servidores de nombre autoritativos para una sub-dominio. A partir de esto, se puede lograr una delegación de sub-dominios. No hay prioridad, todos los servidores tienen la misma precedencia. Los NS los tiene el servidor de nombre por encima de mi (el que delega a mi) y también el propio NS (junto con las direcciones obviamente).

```
; ## ZONA RAIZ
cities.org.      IN  NS berlin.cities.org.
cities.org.      IN  NS brasilia.cities.org.
; ## ZONA delegada
trees.cities.org. IN  NS brasilia.cities.org.
trees.cities.org. IN  NS berlin.cities.org.
trees.cities.org. IN  NS oak.trees.cities.org.
; ## GLUE RECORD ##
oak.trees.cities.org. IN  A 192.168.40.1
```

- **MX:** indican para un nombre de dominio cuáles son los servidores de mail SMTP encargados de recibir los mensajes para ese dominio. El servidor de mail SMTP que envía el mensaje deberá consultar, vía el servicio de DNS, cuáles son los servidores SMTP receptores para el dominio dado. Se asignan prioridades para servidores del mismo dominio. Se puede también lograr un balance de cargas entre distintos servidores SMTP

```

gmail.com  IN      MX      5  gmail-smtp-in.1.google.com.
gmail.com  IN      MX     10  alt1.gmail-smtp-in.1.google.com.
gmail.com  IN      MX     10  alt2.gmail-smtp-in.1.google.com.
gmail.com  IN      MX     50  gsmt147.google.com.
gmail.com  IN      MX     50  gsmt183.google.com.

```

- **CNAME:** mapean un nombre de dominio a otros nombres. Hacen el mapeo del alias de un dominio su nombre canónico (vendría a ser el nombre original)

Ambos ftp y www-> te llevan a berlin.cities

```

ftp.cities.org.  IN      CNAME  berlin.cities.org.
www.cities.org.  IN      CNAME  berlin.cities.org.

```

- **PTR:** mapean direcciones IP a nombres de dominio. Son el inverso de los registros (A). Trabajan en el dominio especial in-addr.arpa
- **SOA:** se utilizan para proporcionar información autoritaria sobre una zona de dominio, lo que incluye información sobre la administración y configuración de esa zona. Solo se admite un registro SOA por zona. Permite que servidores autoritarios de la misma zona se puedan sincronizar.

```

cities.org.  IN      SOA      berlin.cities.org.
                                root.berlin.cities.org. (
                                2008092901 ; ## Serial
                                604800    ; ## Refresh
                                86400     ; ## Retry
                                2419200   ; ## Expiry
                                604800 ) ; ## Neg Cache TTL

```

- Serial: Formato YYYYMMDDSS permite saber en qué fecha se creó la actualización. Con cada cambio en un mismo día, el número de versión (SS) aumenta en una cifra. Al día siguiente cambia el número de serie y el número de versión vuelve a ponerse a 00.
- Refresh: indica cada cuanto tiempo los servidores secundarios deben refrescar desde el primario. La RFC-1912 recomienda entre 1200 a 43200 segundos.
- Retry: Si falla en refrescar, este campo indica el tiempo que se tiene que esperar para volver a hacerlo.
- Expiry: el tiempo que deja de ser autoritativa una zona. Luego de este tiempo, el servidor secundario debe chequear el Serial de la SOA del maestro y, si cambio, realizar una Transferencia de Zona
- TTL de cache negativa: Esto quiere decir que si se preguntó por un valor en donde el servidor autoritativo respondió que no lo tiene el cliente no volverá a preguntar por ese nombre de dominio durante la cantidad de segundos indicados ahí después de recibir la respuesta negativa del servidor autoritativo.

- **AAAA:** mapean un nombre de dominio a una dirección IPv6.

berlin.cities.org. IN AAAA 2001:db8:1234:4567::100  
 brasilia.cities.org. IN AAAA 2001:db8:1234:4567::5

- **TXT:** Son registros que mapean de un nombre de dominio a información extra asociada con el equipo que tiene dicho nombre, por ejemplo pueden indicar finalidad, usuarios, etc. No son utilizados habitualmente. Se los puede ver en uso asociando una clave pública, utilizando por ejemplo IPsec con un esquema de Opportunistic Encryption
- **SRV:** se utilizan para asociar servicios o recursos a nombres de dominio.

### TTL

Es asignado por el servidor autoritativo de un dominio. Este indica cuánto tiempo debe almacenarse en caché una pieza de información antes de que deba considerarse obsoleta o caduca.

### FTP

FTP (File Transfer Protocol) conforma el grupo de los protocolos más viejos de la Internet aun utilizados. Existió antes de TCP/IP, ejecutaba sobre NCP. Es un protocolo para copiar archivos completos, a diferencia de otros protocolos que brindan acceso a archivos: NFS, CIFS. Tiene un modelo cliente/servidor, command/response y corre sobre TCP (requiere protocolo de transporte confiable)

Los clientes FTP no requieren interfaz gráfica.

Soportado por los browsers/clientes Web mediante la URI: <ftp://...>

### Dos conexiones TCP

- Conexión de Control (Out-Of-Band Control) port 21 para los comandos
- Conexión para la transferencia de datos.

El cliente escoge cualquier puerto no privilegiado, ( $n > 1023$ ) y genera conexión de control contra el puerto 21 del servidor. El servidor recibe los comandos por dicha conexión y responde/recibe por la conexión de datos aquellos que lo requieran. La conexión de datos se crea y se cierra bajo demanda. El estado de cada operación se transmite por el canal de control.

Versiones de FTP seguras:

- FTPS, FTP over SSL/TLS.

## Formatos

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

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

## Comandos

- ✧ RETR: obtener un archivo desde el servidor. A nivel de interfaz de usuario el comando que lo inicia es el get.
- ✧ STOR: envía un archivo al servidor. A nivel de interfaz de usuario el comando que lo inicia es el put.
- ✧ LIST: envía una petición de listar los archivos del directorio actual en el servidor. A nivel de interfaz de usuario el comando que lo inicia es el ls o dir.
- ✧ DELE: comando para borrar un archivo en el servidor.
- ✧ SIZE, STAT: obtiene información de un archivo en el servidor.
- ✧ CD, PWD, RMD, MKD: cambia de directorio, obtiene el dir. actual, borra y crea dir.

## Modo Activo

- Conexión de control: port 21.
- Conexión de datos: port 20.
- El servidor de forma activa se conecta al cliente para generar la conexión de datos.

## Modo Pasivo

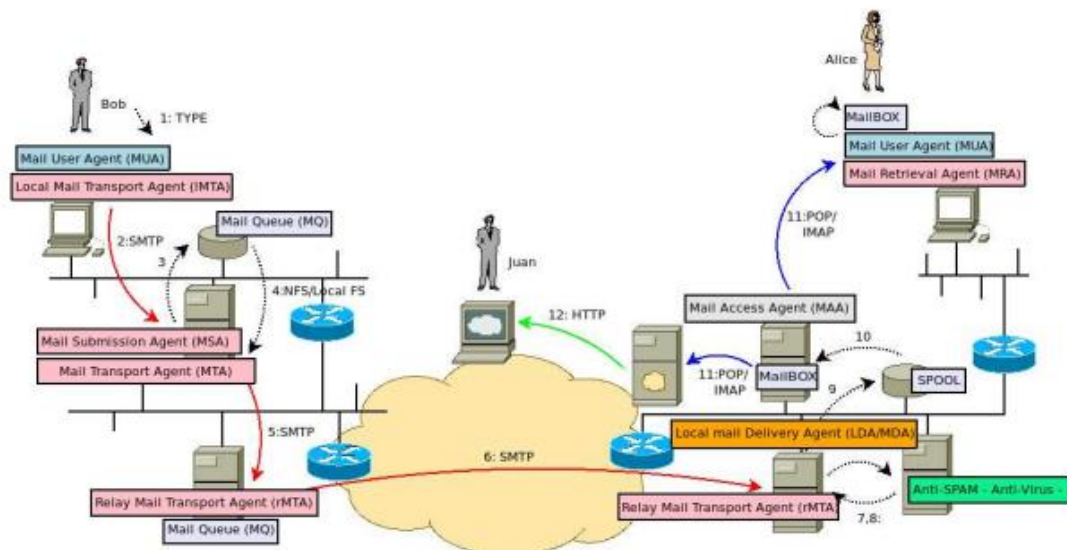
- Conexión de control: port 21.
- Conexión de datos: port no privilegiado.
- El servidor de forma pasiva indica al cliente a que nuevo puerto debe conectarse. La conexión de datos la abre el cliente.  
El cliente elige el puerto de conexión, siempre que no sea un puerto no privilegiado

## Comparación con HTTP

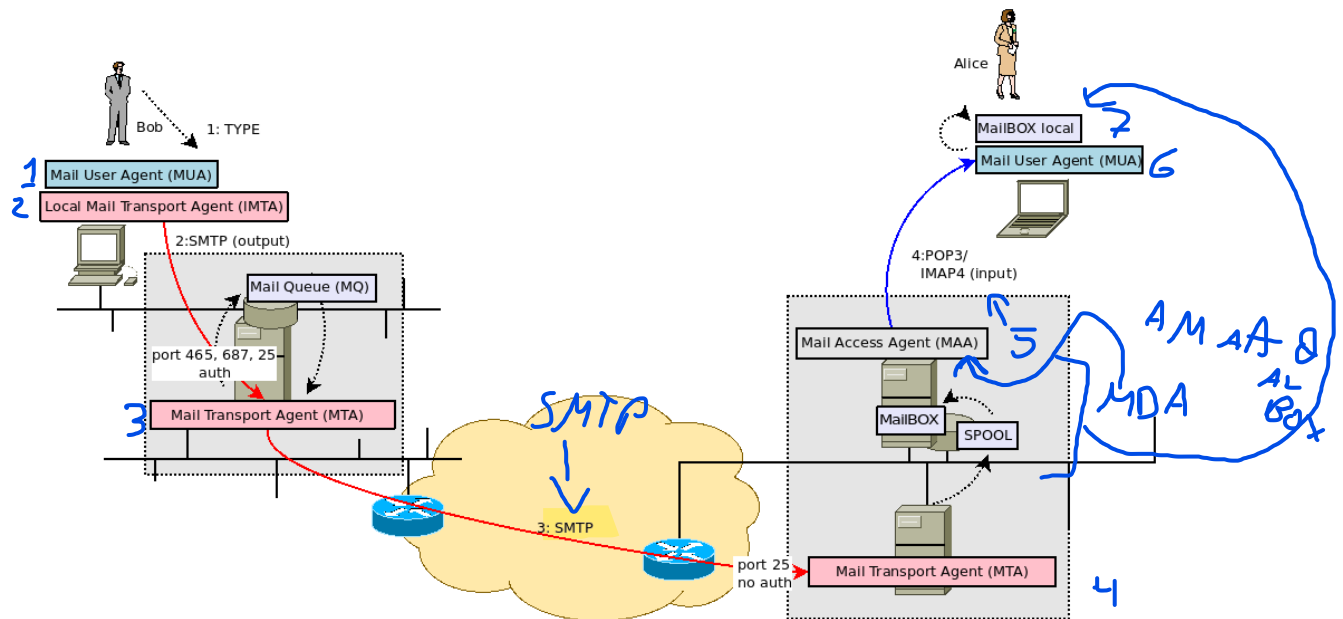
FTP	HTTP
Diseñado para Upload	Se puede con PUT, POST
Soporte de Download	Soporte de Download
Formato ASCII/binary cliente selecciona	meta-data with files, Content-Type, más flexible
No maneja Headers	Manjea Headers, mas información
FTP Command/Response, archivos pequeños más lento	Pipelining
Más complejo 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

### Arquitectura



**Simplificada:**



## Componentes

- MUA (Mail User Agent).
- MTA (Mail Transport Agent).
- MDA (Mail Delivery Agent) o LDA (Local Delivery Agent).
- MAA (Mail Access Agent).
- Otros: MRA (Mail Retrieval Agent), MSA (Mail Submission Agent).

### MUA - Mail User Agent

Cliente. provee una interfaz con usuario. Puede ser un cliente pesado o un servidor web.

Posee integrado un local MTA 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

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 y con MAA propio.

### Submission MSA - Mail Transport Agent

Servidor. Este agente esta habitualmente integrado en el MTA. Recibe el mensaje del MUA y lo pre-procesa antes de pasarlo al MTA para que haga el transporte. Agrega campos que pueden faltar, formato del header: Message-ID, To:, From:, Date: (estos campos los agrega el primero que procesa el mail, que es generalmente el MUA, pero si faltan lo agrega el siguiente).

Utiliza protocolo SMTP, ESMTP.

Habitualmente usaba el port 25, se recomienda usar 587(submission)

Ejemplos: componentes de Postfix (postdrop, pickup), Sendmail-MSA.

### MTA - Mail Transport Agent

Cliente y Servidor. Toma el e-mail desde el MSA o directamente desde el MUA (MSA integrado). Se encarga de enviar el mensaje de e-mail al servidor donde está la casilla de mensaje destino. Hay una comunicación de MTA a MTA. Almacena temporalmente el correo saliente (si no puede entregarlo al MTA correspondiente)

Entre servidores MTA se utiliza el protocolo SMTP

### MDA - Mail Delivery Agent

Servicio interno o Servidor. Se encarga de tomar los e-mails recibidos por el MTA (acepta mensajes del MTA) y llevarlos al mailbox del usuario local. Del spool al mailbox. Habitualmente define el formato del mailbox. Se integra con los protocolos POP y/o IMAP dejando los recursos disponibles al MAA o directamente al usuario.

### MAA - Mail Access Agent

Servidor. Integrado o separado del MDA. Autentica al MUA/usuario y lee los e-mails del mailbox local dejados por el MDA. Transpara los e-mails hacia el MUA o los hace accesibles a este. Se integra con el MDA. Implementa los protocolos POP y/o IMAP dejando acceder a los recursos y dialogando con el MUA

### SMTP

Para el envío de mails entre cliente y servidor se utiliza SMTP. Entre servidores de correo también se utiliza el protocolo SMTP

- Simple Mail Transport Protocol.
- Usa TCP en el puerto 25.
  - o Un MSA puede escuchar en un puerto TCP diferente a los convencionales. Hay que tener en cuenta que si se cambia el MSA hay que informar a todos los MUA del dominio que se cambió el puerto del MSA y estos tendrían que configurar el puerto.
  - o Es posible configurar un MTA para que escuche en un puerto TCP diferente al convencional. Sin embargo, hay que tener en cuenta que si se cambia el MTA se tendría que informar a todos los MTA de los otros dominios de dicho cambio, que a diferencia del cambio de puerto del MSA, es algo prácticamente imposible.
- Utiliza formato ASCII 7 bits en 8 NVT.
- Usa conexiones persistentes
- Puede o no requerir Autenticación.
- Puede o no trabajar de forma segura: SSL/TLS.



Entre MUA-MSA es posible enviar más de un correo durante una misma conexión TCP. Entre MTA-MTA si se trata de destinatarios del mismo dominio se puede enviar en una misma conexión (aunque también podría haber conexiones distintas, verdaderamente depende si el MTA lo permite o como está configurado). Si se trata de destinatarios de distintos dominios, siempre va a haber distintas conexiones, porque son dominios separados y responden a distintos servidores.

Entre los agentes se van agregando encabezados.

## Formato Mensaje

Definido el Cuerpo y el Encabezado en RFC-822, Redefinido RFC-2822.

- ➔ Envelope (Envoltorio), el usuario no lo ve, usado por MTAs, MAIL FROM:, RCPT TO:
- ➔ Header (Encabezado), meta información del mail: Subject:, From:, To:, Return-Path: X-Mailer:, X-...:
- ➔ Body (Cuerpo), separado por línea en blanco del header: Contenido del e-mail.

## MIME ((Multipurpose Internet Mail Extensions)

Como los correos fueron diseñados originalmente para el envío de texto plano y caracteres ASCII, las notas MIME (Multipurpose Internet Mail Extensions) permiten enviar correos electrónicos con contenido diverso, como texto, imágenes y archivos adjuntos (datos binarios en lugar de ASCII). Con tags especiales indica el tipo al sistema destino. Luego codifica el mensaje binario en formato que no viole US-ASCII:

Content - Transfer - Encoding : base64

Content - Type : image / jpeg

## Mensajes Multipart

Cuando un correo electrónico tiene el encabezado "Content-Type: multipart/mixed", significa que el mensaje consta de múltiples partes, algunas de las cuales pueden ser archivos adjuntos.

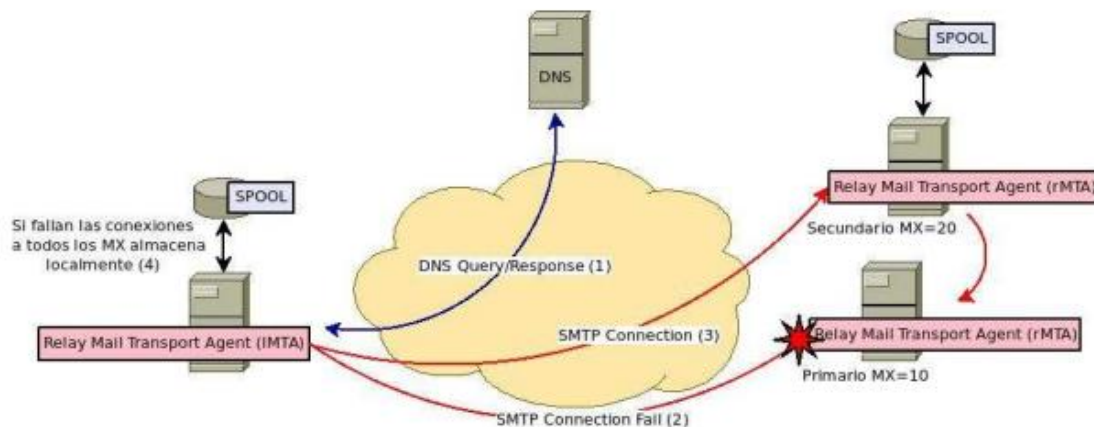
Para separar y marcar cada parte dentro de un mensaje MIME, se utiliza un identificador de límite llamado "boundary", que se establece en el encabezado "Content-Type".

Cada parte del mensaje se delimita con "--boundary", y el "boundary" actúa como un marcador para indicar dónde comienza y termina cada parte.

Debajo de cada divisor se encuentra de nuevo el encabezado Content-Type y otro llamado Content-Transfer-Encoding. Estos indican el tipo del contenido y el algoritmo usado para codificarlo y decodificarlo.



## SMTP y DNS



En el envío de correo entre MTA-MTA, el MTA local debe consultar al DNS local el MX del MTA remoto al que debe enviar el correo.

El MUA no debe hacer ninguna consulta DNS, puesto que ya tiene configurado el servidor MSA al que debe enviar los correos electrónicos (se lo da el proveedor).

## POP3 E IMAP


- POP: Post Office Protocol, RFC-1939: POPv3
- IMAP: Internet Mail Access Protocol, RFC-1730: IMAPv4.
- Usan TCP.
  - o Puertos servidor: 110 (POP) y 143 (IMAP).
- Permiten correr de forma segura sobre SSL/TLS.
- Ambos requieren autenticación. Utiliza formato ASCII 7 bits en 8 NVT.

## Comparación entre ambos protocolos

### POP3 (Post Office Protocol versión 3):

- **Simplicidad:** POP3 es un protocolo de acceso a correo extremadamente simple. Su simplicidad lo hace fácil de implementar y utilizar.
- **Descarga y Borrado:** POP3 generalmente se configura para descargar los correos electrónicos desde el servidor a la máquina local del usuario. En este modo, los correos se eliminan del servidor después de la descarga (aunque se pueden configurar para mantener una copia en el servidor).
- **No Mantiene Estado:** POP3 no mantiene información de estado entre sesiones. Esto significa que no guarda información sobre carpetas, mensajes marcados o cualquier otra información relacionada con el estado de la cuenta del usuario en el servidor.

aka bueno para los sedentarios

- 
- **Limitado para Usuarios Nómadas:** Para usuarios que desean acceder a sus correos electrónicos desde múltiples dispositivos, POP3 puede ser limitante ya que no ofrece una forma sencilla de sincronizar carpetas y correos entre dispositivos.

IMAP (Internet Message Access Protocol):

- **Funcionalidad Avanzada:** IMAP es más avanzado que POP3 y ofrece una amplia gama de funcionalidades. Permite a los usuarios organizar correos electrónicos en carpetas remotas, buscar mensajes, mover mensajes entre carpetas y realizar otras acciones avanzadas.
- **Mantiene Estado:** IMAP mantiene información de estado en el servidor. Esto significa que las carpetas, los mensajes marcados como leídos/no leídos, y otras acciones realizadas en un dispositivo se reflejan en todos los dispositivos conectados, lo que lo hace ideal para usuarios nómadas.
- **Acceso a Partes Componentes de los Mensajes:** IMAP permite a los usuarios acceder a partes específicas de los mensajes, como la cabecera o partes de un mensaje MIME. Esto es útil cuando se necesita descargar solo partes específicas de un mensaje para ahorrar ancho de banda.
- **Complejidad Adicional:** Debido a su mayor funcionalidad, IMAP puede ser más complejo de implementar tanto en el lado del cliente como en el lado del servidor en comparación con POP3.

En resumen, en POP3 los correos electrónicos se descargan del servidor al cliente y se marcan como no leídos en el cliente, mientras que en el protocolo IMAP, los correos se almacenan en el servidor y se reflejan en el cliente.

POP3 no admite la sincronización de carpetas en el servidor (están localmente), mientras que IMAP permite acceder y sincronizar todas las carpetas en el servidor.

IMAP es más completo y adecuado si se quiere una experiencia de correo electrónico más sincronizada entre múltiples dispositivos (lo que la mayoría desea) y se necesita acceder a carpetas en el servidor. Permite trabajar desde varios dispositivos y mantener una estructura de carpetas consistente. Generalmente requiere más recursos debido a su capacidad de mantener el estado de los correos electrónicos y la estructura de carpetas en el servidor. Sin embargo, la mayoría de los servidores de correo modernos están diseñados para gestionar eficientemente esta carga de trabajo adicional.

¿Qué protocolo usará nuestro MUA para enviar un correo con remitente? ¿Con quién se conectará? ¿Qué información será necesaria y cómo la obtendrá?

Usará el protocolo SMTP. Se conectará con el MSA. El MUA debe conocer el puerto SMTP que se utilizará para la comunicación con el MSA. El puerto estándar para la comunicación con el MSA en la mayoría de los casos es el puerto 587 (SMTP Submission), pero también se podría usar el puerto 25 (SMTP) si está configurado de esa manera (u otro si no es ninguno de esos). En muchos casos, se requerirá

autenticación del usuario antes de enviar el correo. Esto implica proporcionar un nombre de usuario y una contraseña válidos para el servidor MSA. Como se dijo antes en el resumen el MUA ya tiene configurado el servidor MSA al que debe enviar los correos electrónicos (se lo da el proveedor).

### QUE PUERTOS UTILIZAN LOS SISTEMAS/PROTOCOLOS QUE VIMOS?

HTTP-> puerto 80  
HTTPS-> puerto 443  
DNS-> puerto 80  
FTP-> puerto 21 control  
-> puerto 20 (modo activo)/puerto elegido por el servidor (modo pasivo)  
SMTP-> puerto 25 o 587 para TCP//465 para SSL  
IMAP-> puerto 143  
POP3-> puerto 110

### Sobre que protocolo de transmision trabajan los protocolos que vimos?

HTTP-> TCP/IP  
HTTPS-> TCP/IP + TLS/SSL  
DNS-> UDP(principalmente) + TCP/IP(para cosas extensas/ mayor seguridad)  
FTP-> Dos conexiones TCP(contro y datos).  
SMTP->TCP/IP.

### Códigos de Retorno de un servidor web

1XX-> Respuesta informativa  
100 Continue  
2XX-> Respuesta exitosa  
200 OK  
3XX-> Redirecciones  
301 Moved Permanently  
302 Found  
304 Not Modified  
4XX-> Errores del cliente  
400 Bad Request  
404 Not Found  
5XX-> Errores del servidor  
500 Internal Server Error  
502 Bad Gateway