Redes y comunicaciones

Resumen teorías - Capa de aplicación

Clase 1 - Introducción

Red de computadoras

Grupo de computadoras/dispositivos interconectados

El conjunto computadoras, software de red, medios y dispositivos de interconexión forma un sistema de comunicación

Objetivo

Compartir recursos: dispositivos, información, servicios

Ejemplos

Red de la sala de PCs, red universitaria, Internet

Componentes de un sistema de comunicación

- Fuente (software)
- Emisor/transmisor (hardware)
- Medio de transmisión y dispositivos intermedios (hardware)
- Procesos intermedios que tratan la información (software y hardware)
- Receptor (hardware)
- Destino (software)
- Otros: protocolos (software), información, mensaje transmitido (software)
- Señal de información, materialización del mensaje sobre el medio (hardware?)

Componentes de una red

Fuera del punto de vista sistémico podemos ver un gran número de componentes:

- Computadoras, en el modelo de Internet: hosts (PCs, laptops, servidores)
- Routers/switches, gateways, AP (Access Points)
- NIC (placas de red), módems
- Vínculos/enlaces conformados por:
    - Medios: cables, fibras ópticas, señales electromagnéticas, antenas, interfaces, etc.

- Programas: browsers, servidores web, clientes de mail, servidores de streaming
- Etc.

Los componentes de la red deben interactuar y comunicarse a través de reglas

Protocolo

Un protocolo define el formato, el orden de los mensajes intercambiados y las acciones que se llevan a cabo en la transmisión y/o recepción de un mensaje u otro evento.

Protocolo de red

Conjunto de reglas que especifican el intercambio de datos u órdenes durante la comunicación entre las entidades que forman parte de una red

Permiten la comunicación y están implementados en los componentes

Modelo en capas (Layering)

Divide la complejidad en componentes reusables

Reduce la complejidad en componentes más pequeños

Las capas de abajo ocultan la complejidad a las de arriba (abstracción)

Las capas de arriba utilizan servicios de las de abajo (interfaces, similar a APIs)

Los cambios en una capa no deberían afectar a las demás si la interfaz se mantiene

Facilita el desarrollo, evolución de las componentes de red asegurando interoperabilidad

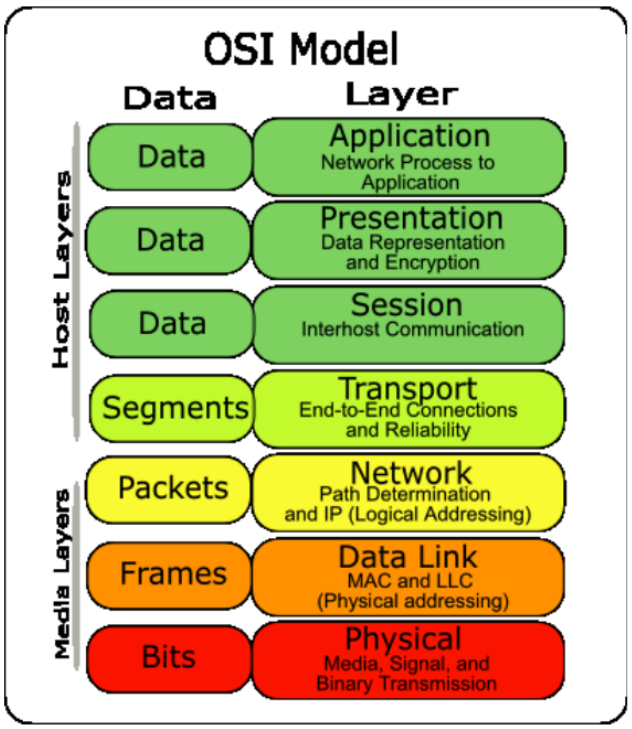
Facilita aprendizaje, diseño y administración de las redes

Modelo OSI (Open System Interconnection)

Modelo abierto y estándar

Modelo dividido en 7 (siete) capas

Modelo de Referencia



Capas de host (host layers): 7,6,5,4, proveen envío de datos de forma confiable

Capas de medio (media layers): 3,2,1, controlan el envío físico de los mensajes sobre la red

Funcionalidad por capa

7. Aplicación: servicios de red a los usuarios y a procesos, aplicaciones

6. Presentación/Representación: formato de los datos

5. Sesión: mantener track de sesiones de la aplicación

4. Transporte: establecer y mantener canal “seguro” end-to-end (applic-to-applic)

3. Red (3): direccionar y rutear los mensajes host-to-host. Comunicar varias redes

2. Enlace de Datos: comunicación entre entes directamente conectados. Comunicar una misma red. Acceso al Medio

1. Física: transportar la información como señal por el medio físico. Características físicas. Información binaria, digital

Modelo TCP/IP

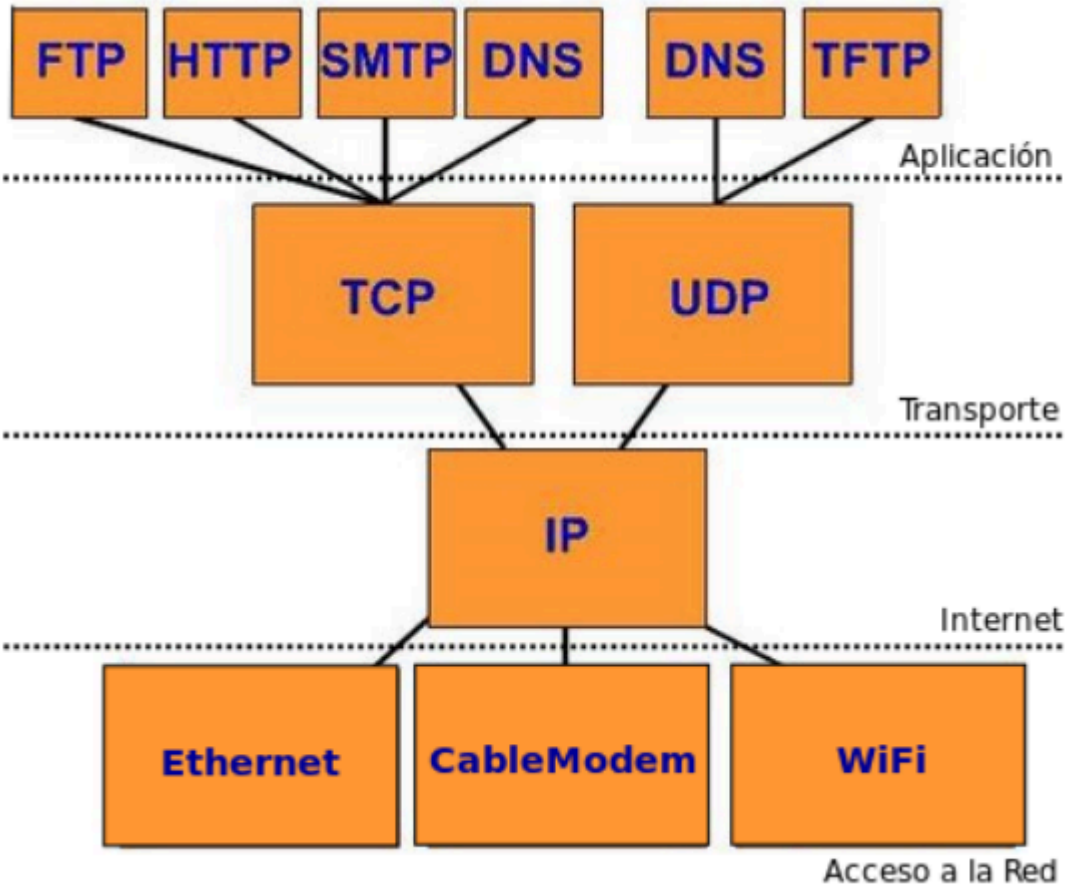
Modelo que se convirtió en estándar

¿Qué protocolos se encuentran en Internet?:

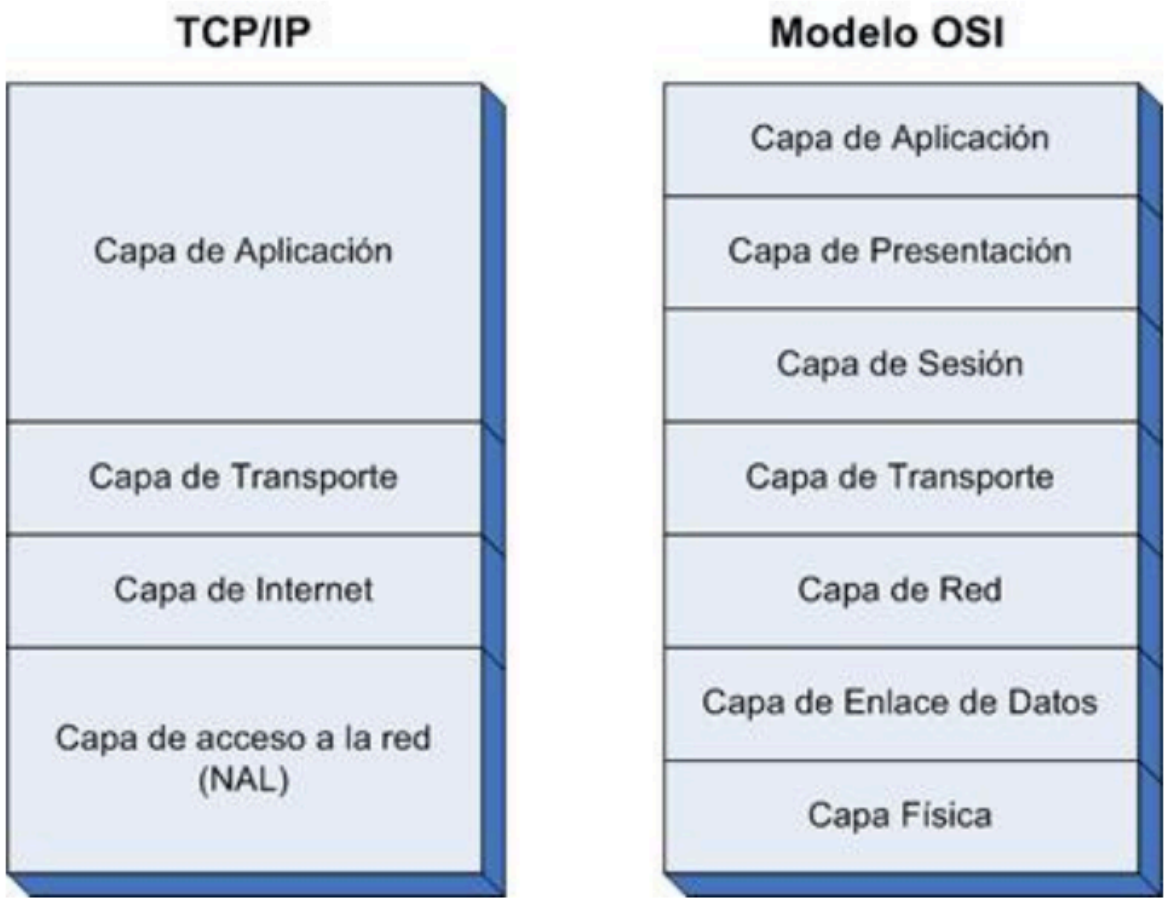
- Modelo Abierto
- Varios protocolos de nivel de enlace: ethernet, 802.3, PPP, HDLC, Frame-Relay, 802.11a/b/g/n/ac, MPLS, DOCSIS, GPON, etc. (No definidos por TCP/IP)
- Protocolos propios de Internet y transporte (núcleo): ARP, IP, ICMP, TCP, UDP, OSPF, BGP, etc.
- Protocolos de aplicaciones: DNS, HTTP, FTP, SSH, SMTP, IMAP, etc.
- API abierta para generar nuevos protocolos

Modelo de 4 capas:

- Capa de aplicación (process/application)
- Capa de transporte o host-to-host
- Capa de Internet o Internetworking
- Capa de acceso a la red (NAL)



Comparación OSI y TCP/IP



Similitudes

- Ambos se dividen en capas
- Ambos tienen capas de aplicación, aunque incluyen servicios distintos
- Ambos tienen capas de transporte similares
- Ambos tienen una capa de red similar pero con distinto nombre
- Se supone que la tecnología es de conmutación de paquetes (no de conmutación de circuitos)
- Es importante conocer ambos modelos

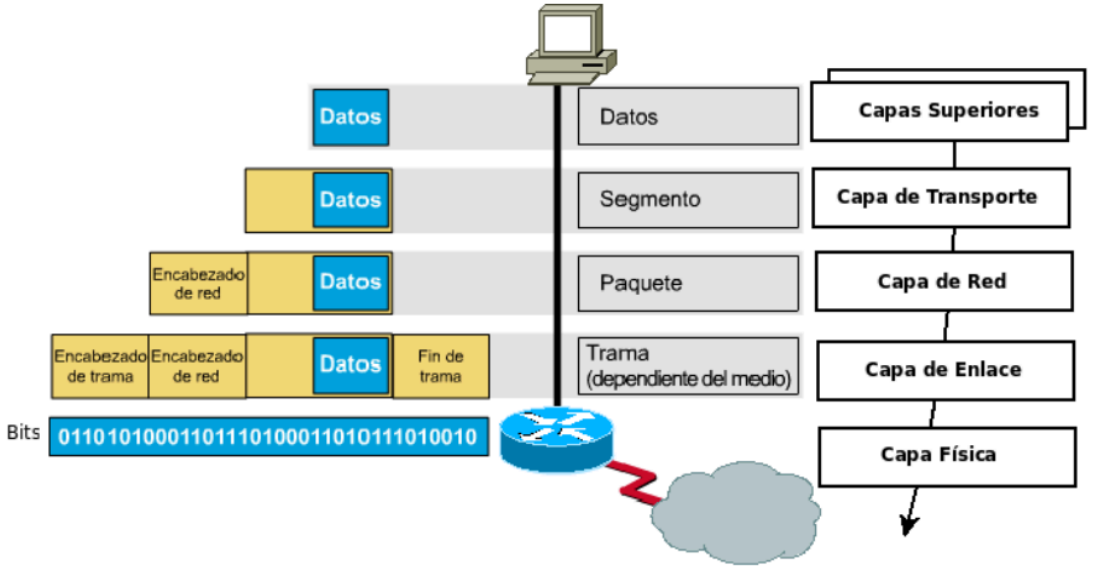
Diferencias

- TCP/IP combina las funciones de la capa de presentación y de sesión en la capa de aplicación

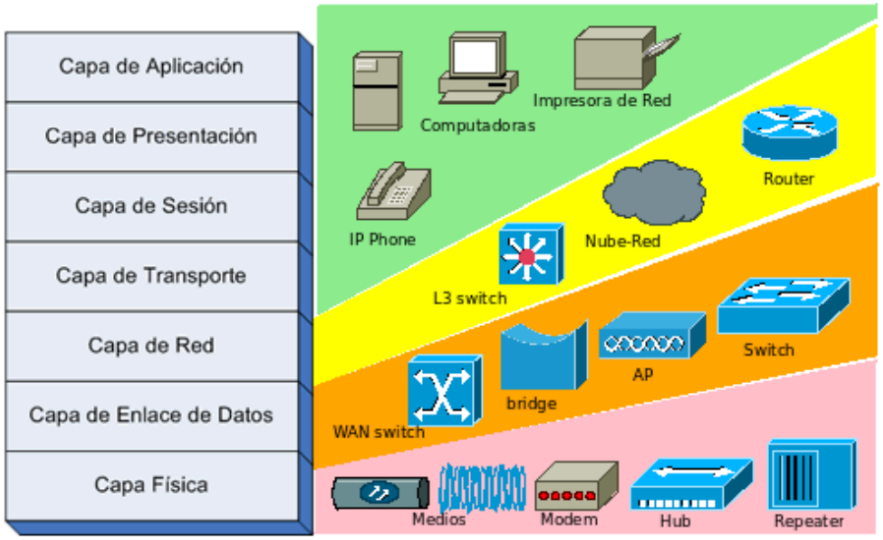
- TCP/IP combina la capas de enlace de datos y la capa física del modelo OSI en una sola capa
- TCP/IP es más simple porque tiene menos capas
- Los protocolos TCP/IP son los estándares en torno a los cuales se desarrolló Internet, de modo que la credibilidad del modelo TCP/IP se debe en gran parte a sus protocolos
- El modelo OSI es un modelo más bien de referencia, teórico, aunque hay implementaciones

Encapsulamiento

Cada capa define su PDU (Protocol Data Unit)



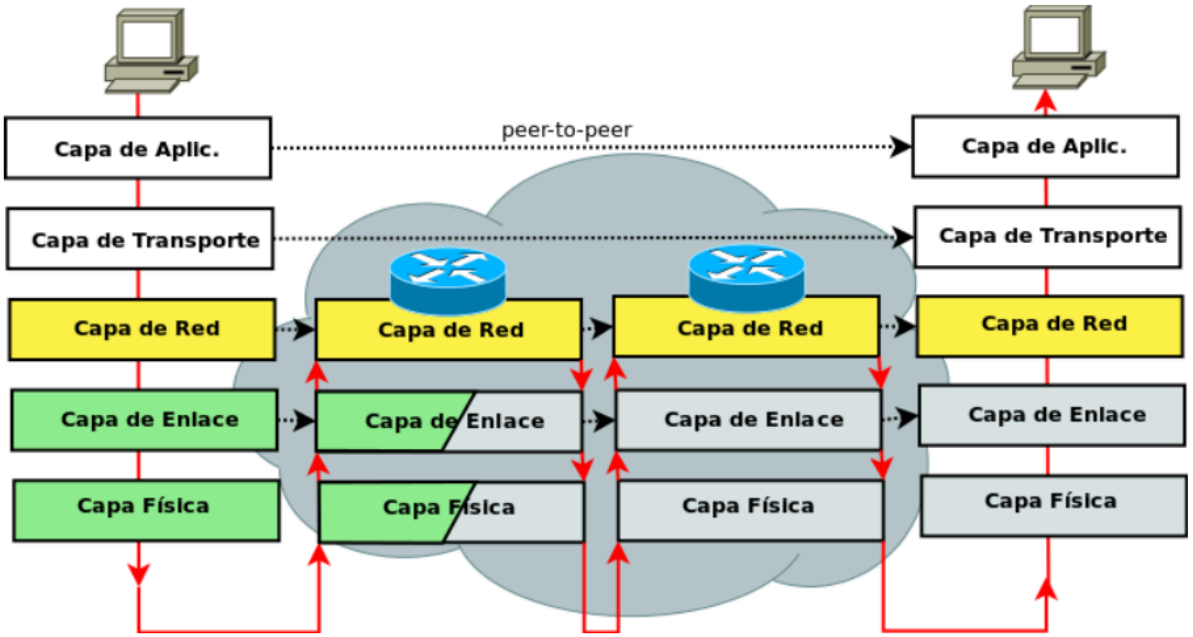
Dispositivos y capas



Comunicación entre capas peer-peer

Cada capa usa el servicio de la de abajo

Cada capa se comunica con la capa del otro extremo



Clasificación de redes

Diferentes clasificaciones de acuerdo a diferentes aspectos

Se pueden mencionar:

- Clasificación por cobertura, distancia, alcance:
    - LAN: (Local Area Network):
        - Red de cobertura local:
        - Ethernet, Wi-Fi
    - MAN: (Metropolitan Area Network):
        - Red de cobertura metropolitana, dentro de una ciudad
        - MetroEthernet, MPLS, Wi-Max
    - WAN: (Wide Area Network):
        - Red de cobertura de área amplia
        - Geográficamente distribuida
        - PPP, Frame-Relay, MPLS, HDLC, SONET/SDH
    - SAN: (Storage Area Network)
        - Red de almacenamiento
        - iSCSI, Fibre Channel, ESCON
    - PAN (Personal Area Network):
        - Red de cobertura personal
        - Red con alcance de escasos metros para conectar dispositivos cercanos a un individuo
        - Bluetooth, IrDA, USB
    - Otros términos: CAN (Controller Area Network o Campus Area Network), NAN (Near-me Area Network, NFC), ...
- Clasificación por acceso abierto o privado:
    - Internet: red pública global, tecnología TCP/IP
    - Intranet: red privada que utiliza la tecnología de Internet
    - Extranet:
        - Red privada virtualizada sobre enlaces WAN
        - Internet. Intranet con acceso de usuarios remotos
        - VPN (Virtual Private Network) IPSec, PPTP, SSL, OpenVPN, L2TP
        - Una intranet mapeada sobre una red pública como Internet
- Clasificación por topología física:
    - Redes de conmutación de circuitos
    - Redes de conmutación de tramas/paquetes:

        - Servicios orientados a conexión. Circuitos virtuales (VCs)
        - Servicios no orientados a conexión. Datagramas
    - Clasificación por tipo de conexión/medio

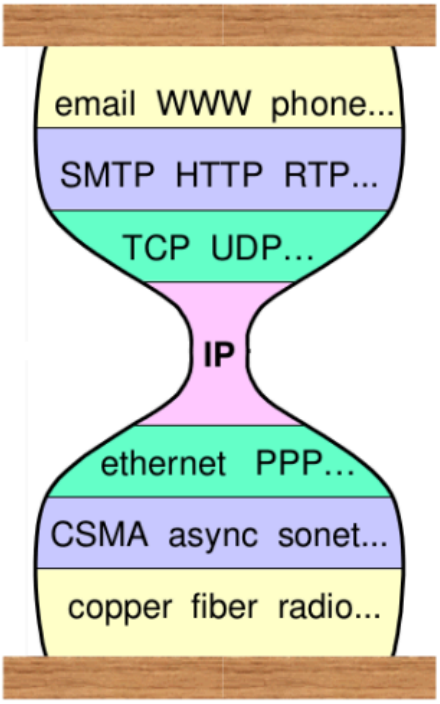
Internet

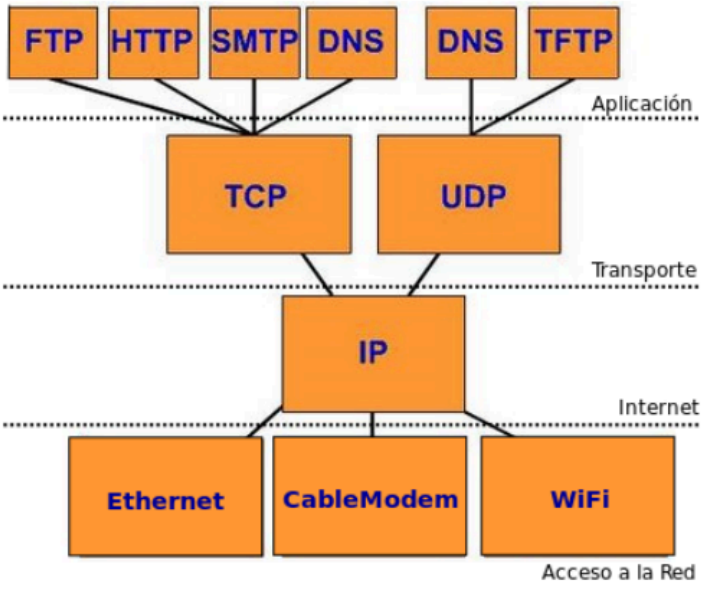
Es una red de redes de computadoras, descentralizada, pública, que ejecutan el conjunto abierto de protocolos (suite) TCP/IP

Integra diferentes protocolos de un nivel más bajo: Internetworking

Modelo de Internet

En forma de reloj de arena



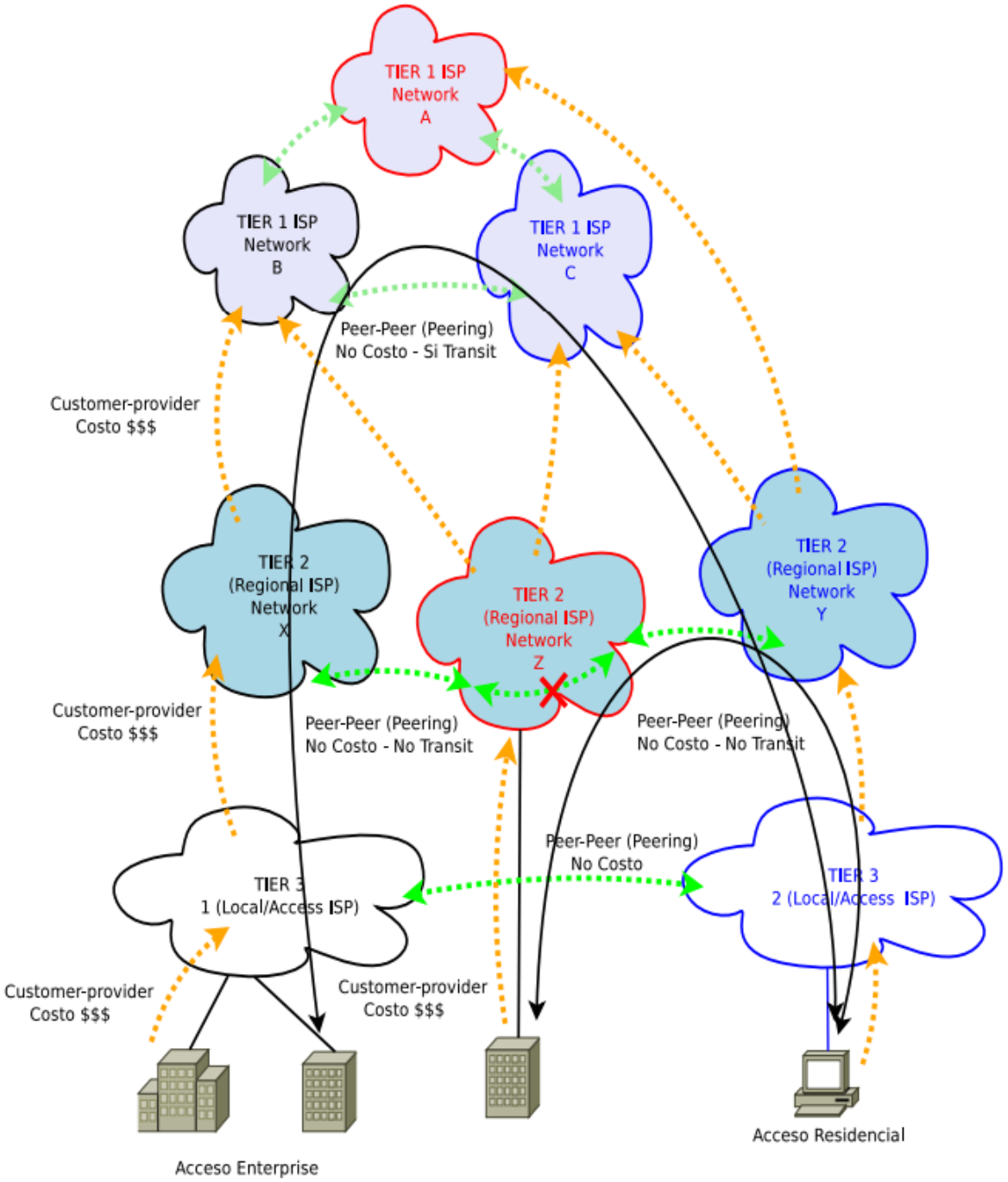


Estructura de Internet

Estructura en jerarquía, en tiers

- Capa de acceso (edge):
    - Acceso residenciales

    - Acceso de organizaciones
- Capa de núcleo (core) dividida en diferentes niveles:
    - Proveedores regionales (regional ISPs)
    - Proveedores nacionales
    - Proveedores internacionales
    - Proveedores internacionales en el tier 1



Clase 2 - Introducción a la capa de aplicación

Funciones de la capa de aplicación

Provee servicios de comunicación a los usuarios (Capa 8 😉) y a las aplicaciones, incluye las mismas

Existe modelo de comunicación machine to machine (M2M), no hay usuarios (personas)

Interfaz con el usuario (UI) u otras aplicaciones/servicios

Las aplicaciones que usan la red pertenecen a esta capa

Los protocolos que implementan las aplicaciones también

Existen aplicaciones que no son de red que deben trabajar con aplicaciones/servicios para lograr acceso a la red

Componentes de la capa de aplicación

Elementos de capa de aplicación: programas que corren en (diferentes) plataformas y se comunican entre sí y los protocolos que implementan

Las aplicaciones en la mayoría de los casos corren en los nodos finales (end-systems), no en el núcleo de la red (más fácil el desarrollo y uso), siguen el principio end-2-end

¿Qué cubre?: se verá el enfoque orientado a Internet, en el cuál la capa de aplicación integra:

- Capa de aplicación propiamente dicha del modelo OSI
- Capa de presentación del modelo OSI
- Capa de sesión del modelo OSI

Capa de sesión

Administra las conversaciones/diálogos entre las aplicaciones

Podría proveer mecanismos transaccionales o de sincronización: commit, checkpoint, rollback

Maneja la actividad

Informa excepciones

Un ejemplo concreto: RPC (Remote Procedure Call) en NFS

A menudo las facilidades del lenguaje de acceso a bases de datos: SQL podría verse como un ejemplo

Integrada en las aplicaciones de red mismas

Podría estar ausente

Básicamente la Capa de sesión se podría ver como un invento de la ISO (será OSI? aunque dice ISO en la ppt 😣)

Ninguna de las redes existentes tenían esta capa

La capa de sesión es muy “delgada”, con relativamente pocas características

En general no se utiliza

Capa de presentación

Conversión y codificación de datos a codificaciones comunes, ej.: ASCII, EBCDIC, charset ISO-8859-1, UTF-8, Unicode16

Compresión y descompresión de datos

Cifrado y de-cifrado de datos

Define formatos y algoritmos para esto: JPEG, MPEG, LZW, AES, DES, IDEA

Ejemplo concreto: XDR (External Data Representation) en NFS

Integrada en las aplicaciones de red mismas

Capa de aplicación

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 cómo debe ser el diálogo (intercambio de mensajes). Qué mensajes se deben intercambiar

Ejemplo concreto: protocolo HTTP y sus implementaciones mediante servidores web y browsers (navegadores)

Modelos de comunicación de aplicaciones

Modelo mainframe (dumb client)

Modelo de carga concentrada

El cliente es “tonto” (dumb) solo corre la comunicacion y la interfaz física con el usuario (ej. terminal)

El servidor pone todo el procesamiento

Modelo antiguo que resurge con thin-clients

Modelo puro: el mainframe es el que decide cuando le da el control al cliente

Modelo puro: el mainframe maneja el diálogo de las comunicaciones

El cliente ejecuta en el mainframe

Modelo cliente/servidor

Modelo de carga compartida.

Idea inicial: el cliente pone procesamiento de interfaz

El servidor pone el resto del procesamiento

Existen modelos en varios tiers (2 tiers, 3 tier o multi-tier)

El servidor corre un servicio esperando de forma pasiva la conexión

Los clientes se conectan al servidor y se comunican a través de este

Ejemplo: File Server vía NFS o FTP

Modelo asimétrico 1 a N, M a N (donde M<N)

Modelo peer to peer (P2P)

Modelo de carga completamente compartida y distribuida

Los peers (participantes) pueden cumplir rol de cliente, servidor o ambos en un instante

Sistema escalable en cuanto a rendimiento

Sistema no escalable en cuanto a administración

Ejemplo: redes legadas para compartir archivos:

- Novell Lite
- Microsoft Windows for Workgroup basado en LAN Manager (sobre NetBEUI)
- Algo más actual:
    - Gnutella

- Bittorrent (servicio de file sharing totalmente P2P)

Modelo asimétrico N a N.

Modelo peer to peer híbrido

Existen diferentes tipos de nodos con diferentes roles

Hay nodos centrales donde se registra la información y al resto de los nodos

Sistema escalable en cuanto a rendimiento

Sistema más escalable que P2P puro?

Ejemplos: eDonkey (y sus variantes aMule, eMule), Napster, IM, Skype

Modelo asimétrico M a N

Requerimientos de aplicaciones

Aplicación	Pérdidas	Bandwidth	Sensible a Time
file transfer	no	Flexible	no
e-mail	no	Flexible	no
Web documents	no	Flexible	no
real-time audio/video	tolerante	audio: 5kbps-1Mbps video:10kbps-5Mbps	si, 100's msec
stored audio/video	tolerante	Igual al de arriba	si, pocos secs
interactive games	tolerante	Pocos Kbps	si, 100's msec
instant messaging	no	flexible	Si y no

Cada aplicación puede tener diferentes requerimientos de: seguridad, tiempo de respuesta, confiabilidad, optimizar ancho de banda, etc.

Esto implica diferentes alternativas de transporte

Clase 3 - Introducción a HTTP

WWW (red de sistemas de hipertexto inter-linkeados accesibles vía Internet)

Elementos web

Recurso u objeto HTTP por ej. web page

Referenciado por una URI (Uniform Resource Id): URL (Uniform Resource Location) o URN (Name)

Formato de URL: protocol://[user:pass@]host:[port]/[path]. Ejemplo:

- http://www.NN.unlp.edu.ar:8080/dir/index.html

URN: no indica ubicación, solo identifican categorías, poco implementadas. Ejemplo:

- urn:isbn:0132856204, urn:ietf:rfc:2616

Los objetos que se transmiten pueden ser páginas web (web page), imágenes, archivos de multimedia, etc.

Funcionamiento de HTTP

Modelo cliente/servidor, request/response (sin estado - stateless)

Protocolo que corre sobre TCP (requiere protocolo de transporte confiable), usa el puerto 80 por default

El cliente escoge cualquier puerto no privilegiado

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

Versión 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 Request

Solo una forma de Response

Versión HTTP/1.0

Versión de HTTP/1.0 estándar [RFC-1945]

Define formato, proceso basado en HTTP 0.9:

Se debe especificar la versión en el requerimiento del cliente

Para los Request, define diferentes métodos HTTP (GET, POST, PUT, HEAD, DELETE, LINK, UNLINK)

Define códigos de respuesta (1xx, 2xx, 3xx, 4xx, 5xx)

Admite repertorio de caracteres además del ASCII, como: ISO-8859-1, UTF-8, etc.

Admite MIME (no solo sirve para descargar HTML e imágenes)

Por default no utiliza conexiones persistentes (pero podría hacerlo con el header Connection: keep-alive)

Hosts virtuales

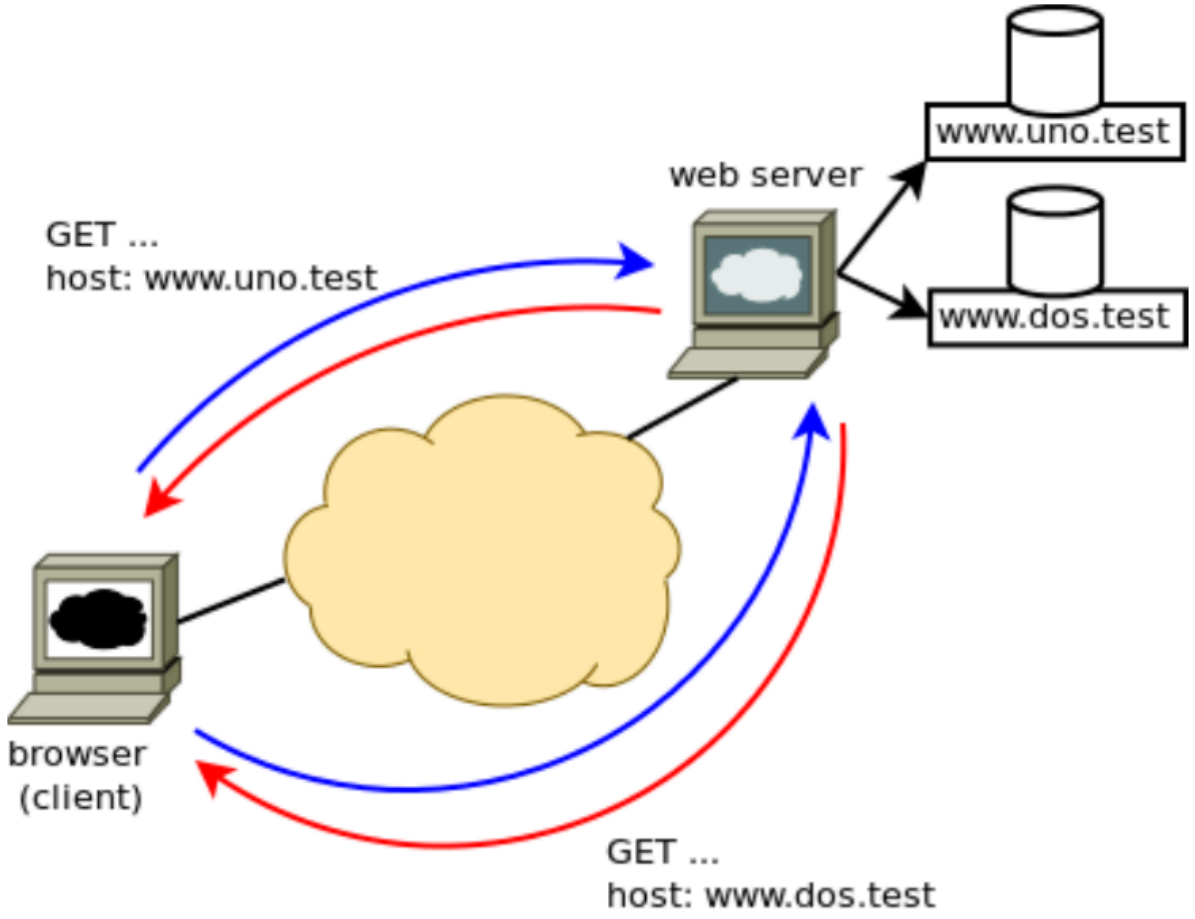
Mediante el parámetro Host se pueden multiplexar varios servicios sobre un mismo host

? host www.uno.test

www.uno.test has address 192.168.0.2

? host www.dos.test

www.dos.test has address 192.168.0.2



HTTP/1.1

1997 con la [RFC-2068] se actualiza con [RFC-2616]

Nuevos mensajes HTTP 1.1: OPTIONS, TRACE, CONNECT

Conexiones persistentes por omisión (Connection: keep-alive no es necesario)

Implementa pipelining, lo cual mejora el tiempo de respuestas

Pipelining

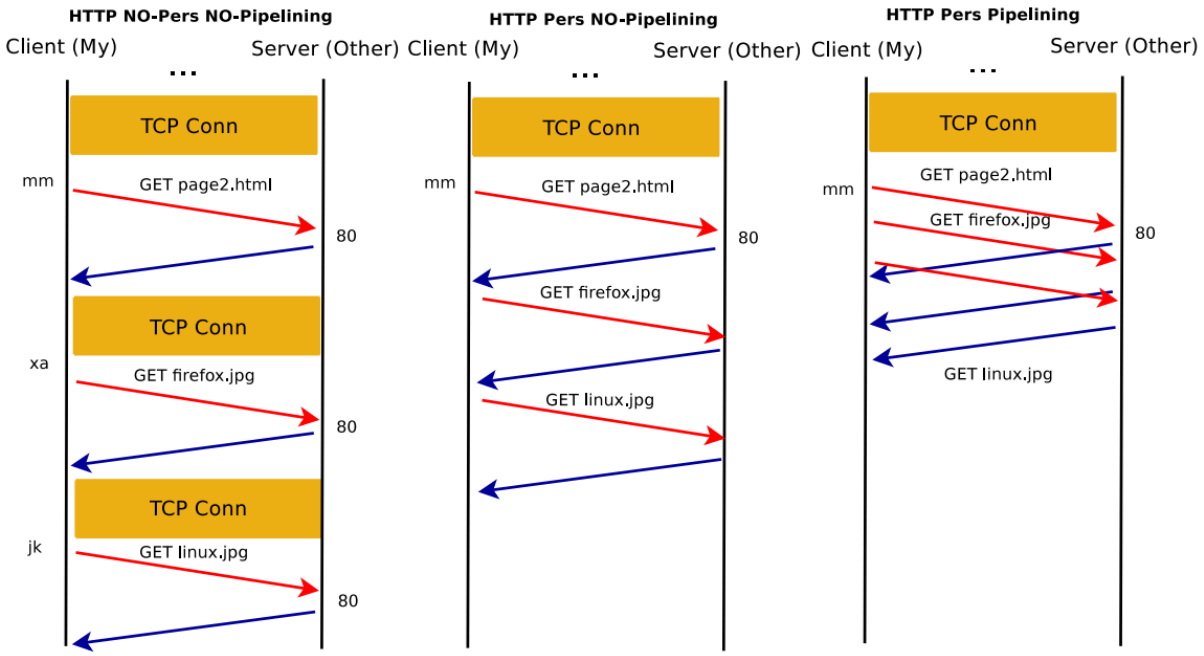
No necesita esperar la respuesta para pedir otro objeto HTTP

Solo se utiliza con conexiones persistentes

Mejora los tiempos de respuesta

Sobre la misma conexión se debe mantener el orden de los objetos que se devuelven

Se pueden utilizar varios threads para cada conexión



Trace

Es un método utilizado para debugging

El servidor debe copiar el requerimiento tal cual

El cliente puede comparar lo que envía con lo que recibe

Connect

Es un método utilizado para generar conexiones a otros servicios montados sobre HTTP

Proxy-Agent genérico

Redirects HTTP

Redirect temporal 302:

- Se indica la nueva URL/URI
- El user-agent no debería redireccionar salvo que el usuario confirme

Moved Permanently 301:

- Se indica que cualquier acceso futuro debe realizarse sobre la nueva ubicación (mejora indexadores)

- Se pueden generar problemas con Cookies

CGIs

Server-Side Script:

- Ejecuta del lado del servidor
- CGI (Common Gateway Interface): aplicación que interactúa con un servidor web
- CGIs leen de STDIN (POST) o de variables de entorno (GET): query string datos de usuario
- Escriben en la STDOUT response
- Deben anteponer el content-type en el header
- POST permite enviar más datos
- Lenguajes de scripting más flexibles y seguros: PHP, ASP, JSP
- Implementados como CGIs, dentro de módulos propios del web-server

JavaScript

Client-Side Script:

- Ejecuta del lado del cliente, en el browser
- JavaScript estándar W3C
- Usan modelo de objetos DOM (Document Object Model)
- Otros lenguajes JScript, VBScript
- Permiten extensiones como AJAX (Asynchronous JavaScript And XML):
    - AJAX hace requerimientos particulares y no necesita recargar toda la página
    - Parseo XML para comunicarse
- Existen numerosos frameworks que encapsulan esta funcionalidad brindando una API fácil de utilizar

Server-Side to Server-Side Script:

- Permiten la comunicación entre servidores
- Modelo de “objetos” y servicios distribuidos
- Conjunto de convenciones para implementar RMI (Remote Method Invocation) sobre HTTP (u otro protocolo de texto)

- Previo XML-RPC
- SOAP (Simple Object Access Protocol)
- Web-Services
- REST

Cookies

Mecanismo que permite a las aplicaciones web del servidor “manejar estados”

El cliente hace un request

El servidor retorna un recurso (un objeto HTTP, por ejemplo una página HTML) indicando al cliente que almacene determinados valores por un tiempo

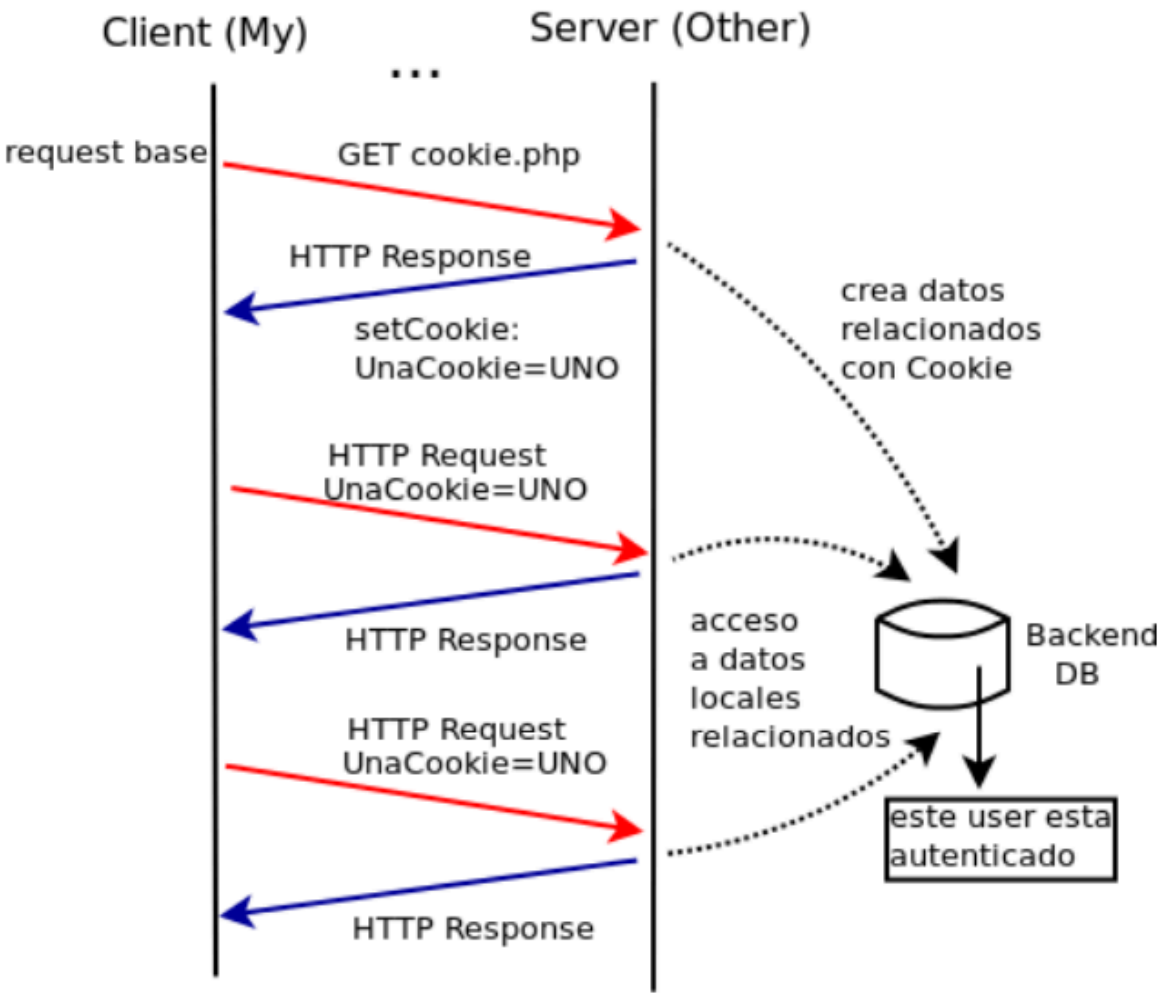
La Cookie es introducida al cliente mediante el mensaje en el header Set-Cookie que indica un par (nombre,valor) (ej.: Set-Cookie: session=session_id)

El cliente en cada requerimiento luego de haber almacenado la Cookie se la enviará al servidor con el header Cookie (ej.: Cookie: session=session_id)

El servidor puede utilizarlo o no

El servidor puede borrarlo

Esta información puede ser usada por client-side scripts



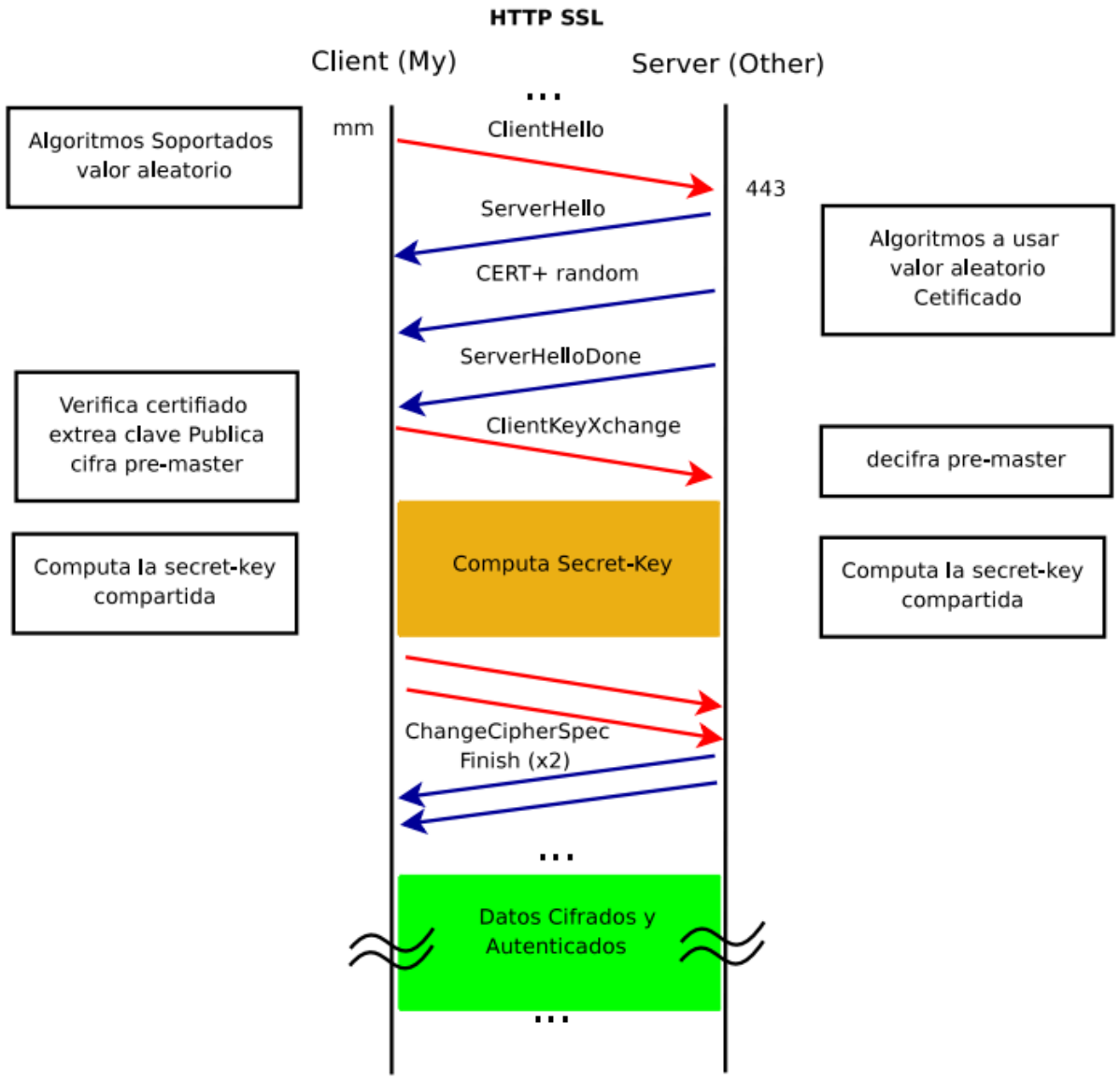
HTTPs

Es HTTP sobre TLS/SSL

Utiliza el port 443 por default

Tiene una etapa de negociación previa

Luego se cifra y autentica todo el mensaje HTTP (incluso el header)



A diferencia de HTTP, el protocolo de navegación HTTPS provee un canal de comunicación seguro entre el cliente y el servidor

La seguridad que ofrece HTTPs implica que:

- La información viaja de manera cifrada de extremo a extremo
- El cliente puede verificar la autenticidad del sitio visitado
- Opcionalmente el servidor puede requerir autenticación del cliente

Certificados digitales

Los certificados digitales, son reconocidos como válidos, porque fueron emitidos por una autoridad de certificación en la que confiamos

La autoridad de certificación es quien asegura que el par de claves pertenece a determinado sitio web

Un certificado digital, está firmado por la autoridad de certificación

Por defecto nuestros navegadores vienen con un conjunto de autoridades de certificación en la que se confía

Web cache

"Proxiar" y cachear recursos HTTP

Objetivos:

- Mejorar tiempo de respuesta (reducir retardo en descarga)
- Ahorro de BW (recursos de la red)
- Balance de carga, atender a todos los clientes

Se solicita el objeto, si está en cache y está "fresco" se retorna desde allí (cache hit)

Si el objeto no está o es viejo (cache miss) se solicita al destino y se cachea

Se puede realizar control de acceso

Caché del lado del cliente (privada)

Los web browser tienen sus propias cache locales

Los servidores agregan headers:

- Last-Modified: date
- ETag (entity tag): hash. Es una etiqueta de entidad que representa una versión única del recurso. Si no existe, significa que el contenido cambió y por ende cambió su ETag

Requerimientos condicionales desde los clientes:

- If-Modified-Since: date
- If-None-Match: hash. Se usa para comparar ETags

Respuestas de los servidores:

- 304 Not Modified
- 200 OK

Los cache como servers funcionan como proxy (shared cache)

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

Existen:

- Proxys no-transparentes
- Proxys transparentes
- Proxy en jerarquía o mesh (ICP y HTCP)
- CDN (Content Delivery Network), funcionan por DNS

Clase 4 - HTTP/2 y HTTP/3

HTTP/2

Reemplazo de cómo HTTP se transporta

No es un reemplazo del protocolo completo

Se conservan métodos y semántica

Base del trabajo protocolo desarrollado por Google SPDY/2

Definido en:

- RFC 7540: Hypertext Transfer Protocol version 2
- RFC 7541: HPACK - Header Compression for HTTP/2

Problemas con HTTP/1.0

Un request por conexión, por vez, muy lento

Alternativas (evitar HOL):

- Conexiones persistentes y pipelining
- Generar conexiones paralelas

Problemas con HTTP/1.1

Pipelining requiere que las responses sean enviadas en el orden solicitado, HOL posible

HOL = Head Of Line blocking (si hay un requerimiento grande delante de uno pequeño, como estos deben llegar en orden, hay que esperar que finalice el grande para enviar el pequeño)

El HOL se puede mejorar generando múltiples conexiones (esto implica más carga al OS y más cantidad de conexiones TLS, lo que también implica más carga)

POST no siempre pueden ser enviados en pipelining

Demasiadas conexiones generan problemas, control de congestión, mal uso de la red

Muchas requests, muchos datos duplicados (headers)

Diferencias entre HTTP/2 y HTTP/1.1

Protocolo binario en lugar de textual(ASCII), binary framing: (más eficiente)

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

Utilizar una conexión para pedir/traer datos en paralelos, agrega: datos fuera de orden, priorización, flow control por frame

Usa compresión de encabezado

Permite a los servidores “pushear” datos a los clientes, es decir, enviar información de manera anticipada (si piden el html le mandan también el js y el css porque lo van a necesitar)

La mayoría de las implementaciones requieren TLS/SSL, pero no es el estándar

HTTP/2 mux stream, framing

Puede generar una o más conexiones TCP. Pero trata de aprovechar las que tiene establecidas

Un stream es como una sub-conexión (una “conexión” HTTP/2 dentro de una conexión TCP)

Un stream tiene un ID y una prioridad (alternativa) y son bidireccionales

Sobre una conexión TCP multiplexa uno o más streams (“conexiones HTTP/2”)

Los streams transportan mensajes

Los mensajes HTTP/2 (Request, Response) se envían usando un stream

Los mensajes HTTP/2 son divididos en frames dentro del mismo stream

Un frame es una porción de mensaje: header fijo+payload variable (unidad mínima)

El mismo stream puede ser usado para llevar diferentes mensajes

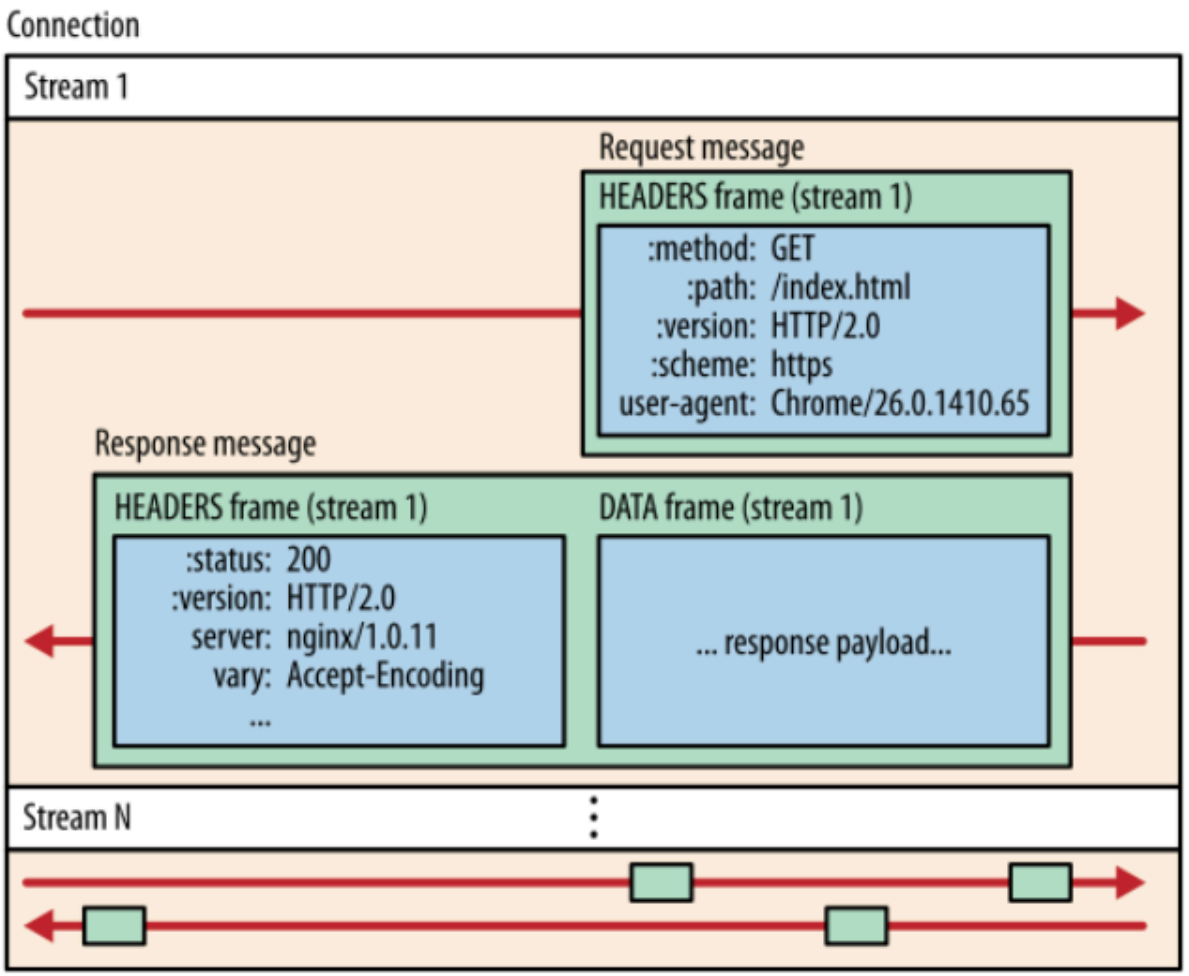
Los mensajes están compuestos por frames, que podrían ser de diferentes tipos

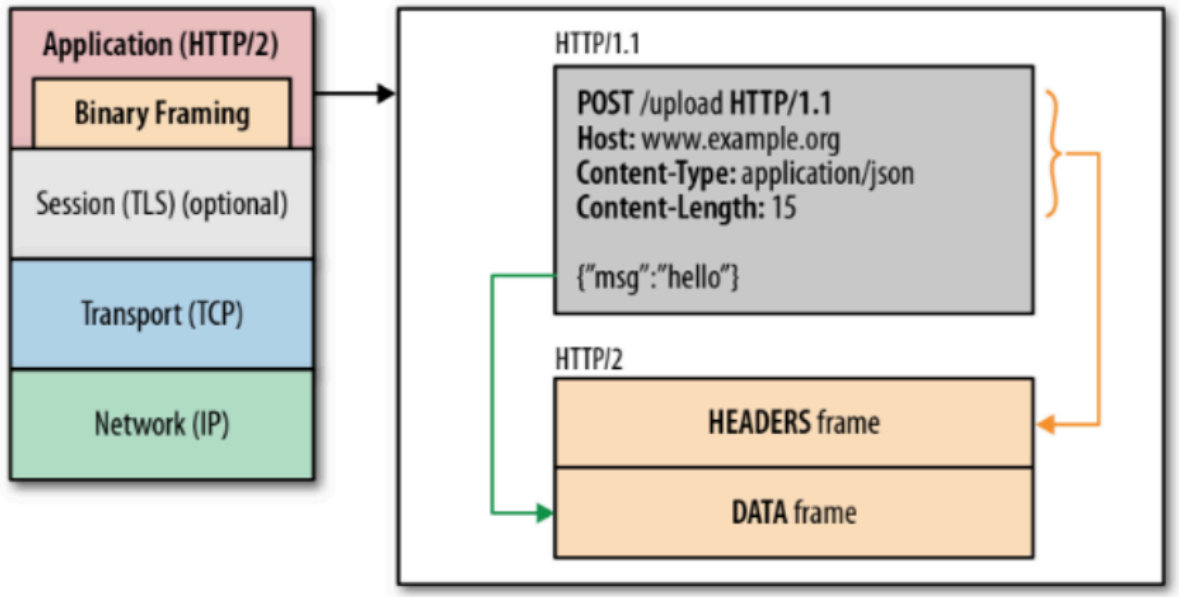
Los streams van en una misma conexión

Los streams son identificados y divididos en frames

Frame types: HEADERS, DATA, PUSH_PROMISE, WINDOW_UPDATE, SETTINGS, etc.

Streams codificados en binario y cada frame con header común fijo(9B)


Connection
Stream 1
Request message
HEADERS frame (stream 1)
:method: GET
:path: /index.html
:version: HTTP/2.0
:scheme: https
user-agent: Chrome/26.0.1410.65
Response message
HEADERS frame (stream 1)
:status: 200
:version: HTTP/2.0
server: nginx/1.0.11
vary: Accept-Encoding
...
DATA frame (stream 1)
... response payload...
Stream N


Application (HTTP/2)
Binary Framing
Session (TLS) (optional)
Transport (TCP)
Network (IP)
HTTP/1.1
POST /upload HTTP/1.1
Host: www.example.org
Content-Type: application/json
Content-Length: 15
{"msg":"hello"}
HTTP/2
HEADERS frame
DATA frame

Headers HTTP/

Se mantienen casi todos los headers de HTTP/1.1

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 en HTTP/2 con:

:method: head

:path: /algo

:scheme: https o http

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

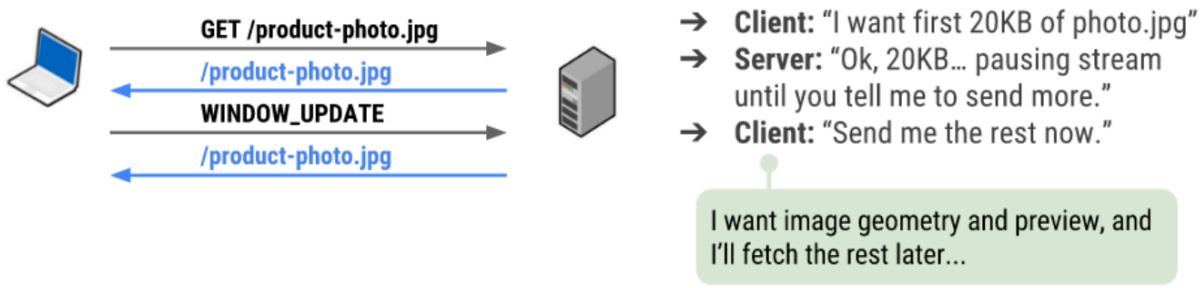
Para las respuestas: :status: códigos de retornos 200, 301, 404, etc.

HTTP/2 priorización y flow-control

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



HTTP/2 inline vs push

Cuando el cliente solicita una página, "parsea" el primer response HTML luego solicita el resto

El server puede enviar el HTML más otros datos, por ejemplo CSS o Javascript (PUSH_PROMISE)

No siempre es lo que necesita el cliente, depende de qué funcionalidad ofrece

Compresión y Soporte

Compresión de encabezados

SPDY/2 propone usar GZIP. Pero GZIP + cifrado, tiene "bugs" utilizados por atacantes

Se crea un nuevo compresor de Headers: HPACK

HTTP/2 y SPDY, soportados en la mayoría de los navegadores

Otras Características

HTTP/1.1, posibilidad de hacer un upgrade durante la conexión:

- Upgrade Header. Connection: Upgrade, HTTP2-Settings Upgrade: h2c|h2

HTTP/2: negociar el protocolo de aplicación: ALPN: Application-Layer Protocol Negotiation. Se negocia como extensión de SSL en Hello (Anteriormente NPN). Se ofrece: h2, h3, http/1.1, etc.

Posibilidad de negociar protocolo alternativo: alternative Service: alt-svc

HTTP/3

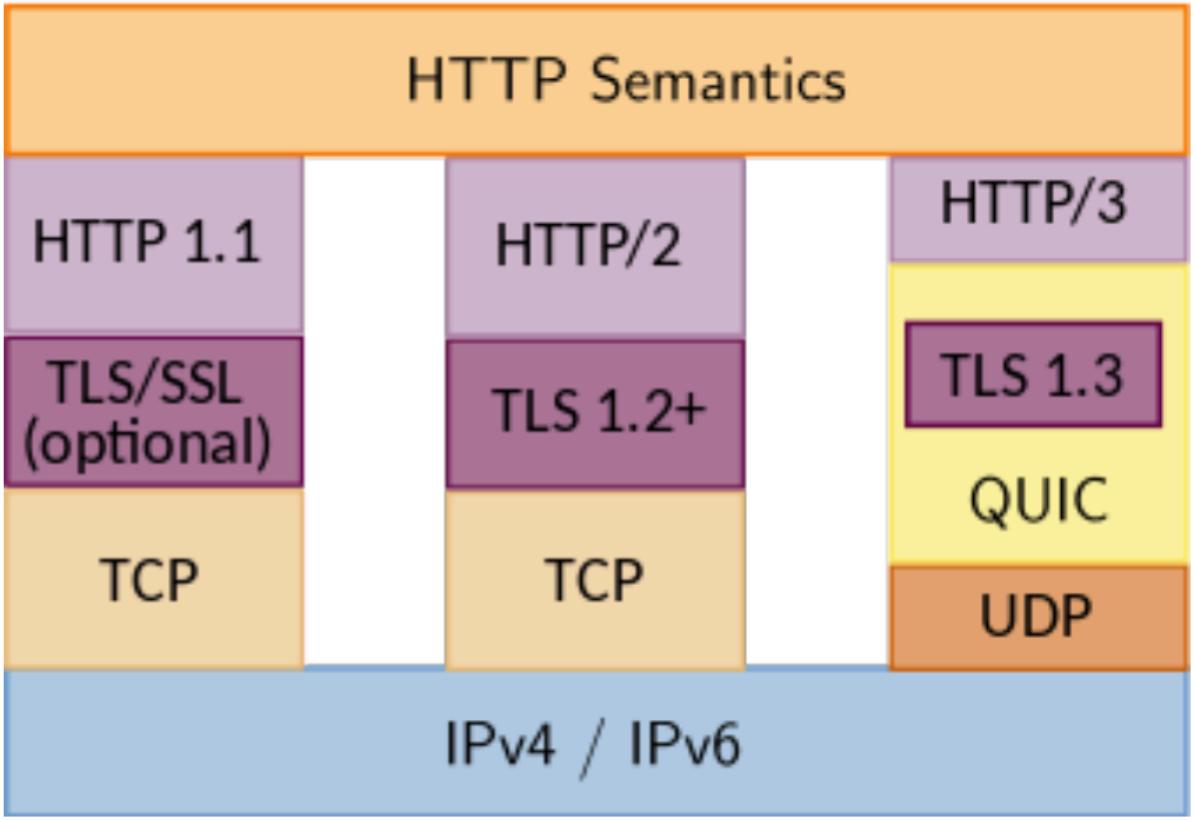
Transportar HTTP sobre nuevo protocolo: QUIC, que corre sobre UDP, default UDP:443

Igual que HTTP/2 se conservan métodos y semántica de HTTP/1.1

Base del trabajo protocolo desarrollado por Google, QUIC

Definido en:

- RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport
- RFC 9114: HTTP/3



Diferencias principales de HTTP/3-QUIC

Reducción de latencia en conexiones (2 mensajes de handshake)

No depende del manejo de la conexión y controles de TCP (evita HOL)

Loss recovery/congestion control, sobre UDP, lo puede hacer como TCP-CUBIC o de otra forma

QUIC genera nuevos números de secuencia y re-cifra las retransmisiones, permitiendo mejor detección de pérdidas y medición de RTT

QUIC tiene paquetes cifrados de forma individual, no por conexión/bytestream

QUIC facilita movilidad, tiene ID de conexión independiente de IPs y ports

Desventaja de QUIC: muchos middle-boxes filtran UDP, salvo port 53 y NAT

Protocolo de transporte QUIC: implementado usualmente en User-space vs Kernel-space

UDP facilita algunos ataques de seguridad

Clase 5 - DNS

Aspectos y Elementos de DNS

- Espacio de nombres (sintaxis y zonas, dominios)
- Procedimiento de delegación y arquitectura
- Base de datos distribuida y servidores
- Define los componentes y el protocolo para su comunicación
- Procedimiento de búsqueda/resolución (protocolos)

FQDN

Fully Qualified Domain Name

Nombre de dominio FQDN: lista de etiquetas (labels) separadas por puntos

Se leen desde el nodo/etiqueta de la izquierda hasta la raíz del árbol (el punto)

Estructura jerárquica con sobrenombres (niveles)

La sintaxis jerárquica refleja la delegación de autoridad

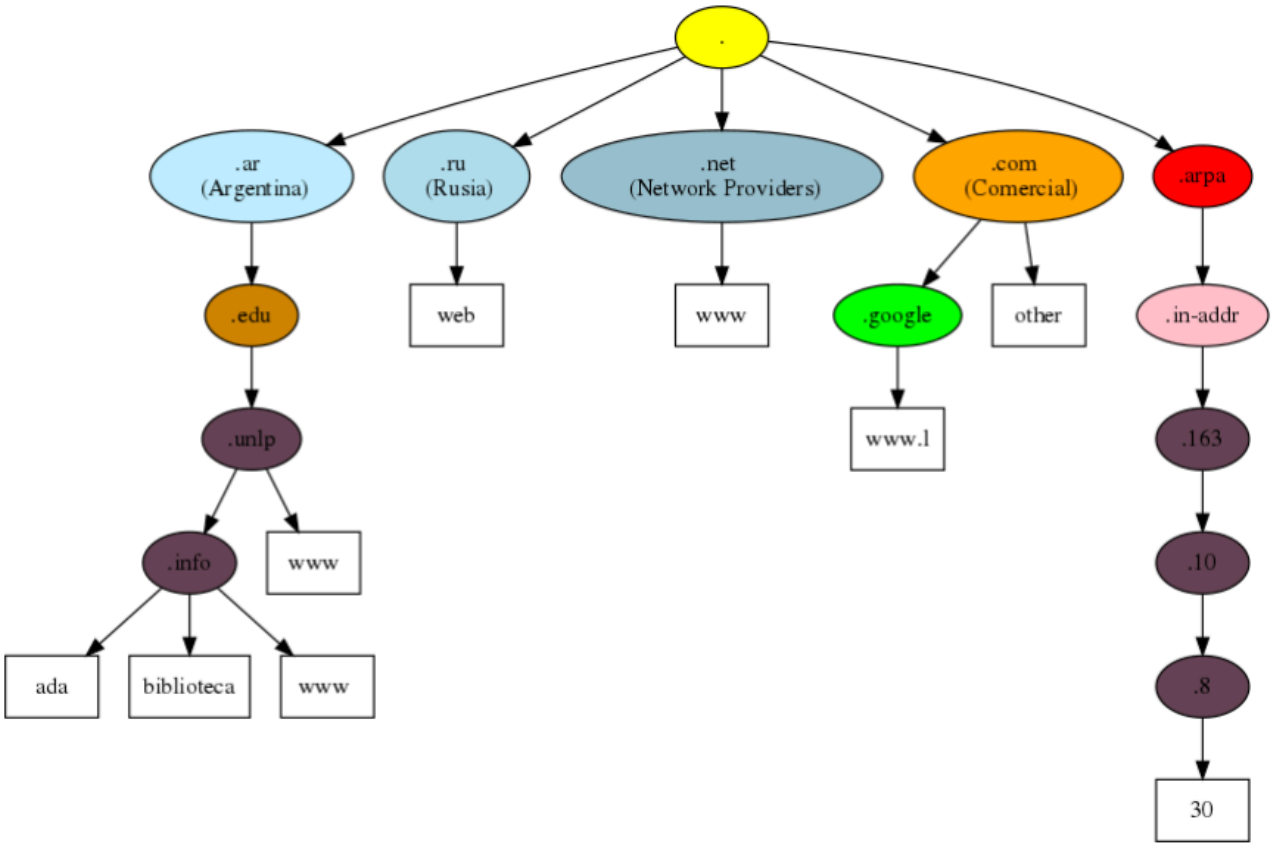
No son case-sensitive, cada etiqueta tiene máximo 63 chars. Máximo de etiquetas 127, máximo nombre 255 chars

www, biblioteca, ada (No FQDN)

www.info.unlp.edu.ar. , biblioteca.info.unlp.edu.ar. (FQDN)

ada.info.unlp.edu.ar. (FQDN)

www.info.unlp.edu.ar (Considerado FQDN)



Top Level Domains (TLDs)

Los TLDs se podrían clasificar en 3 grupos:

1. gTLDs, Generic TLDs:
    - Contienen dominios con propósitos particulares, de acuerdo a diferentes actividades
    - Políticas definidas por el ICANN: unsponsored TLD o
    - Definidas por otra organización: sponsored TLD
2. ccTLD Country-Code TLDs:
    - Contienen dominios delegados a los diferentes países del mundo
    - ISO 3166-1 alfa-2
3. .ARPA TLD: es un dominio especial, usado internamente para resolución de reversos

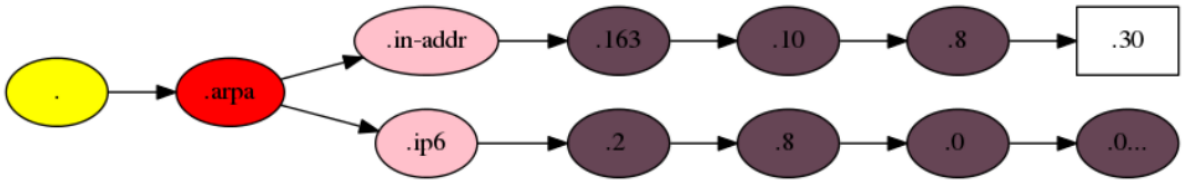
Generic TLDs actualmente

A partir de 2012 se comenzó a aceptar nuevas aplicaciones para new gTLDs

Proceso de licitación, donde hay en juego grandes sumas de dinero

Nuevos dominios registrados (500+): .academy, .casa, .beer, .bike, .futbol, .pizza, .paris, .wiki, .viajes, etc.

Reverso (.arpa)



La idea del reverso es poder validar la IP que está generando una conexión

Ralentiza el proceso, ya que para cada consulta habría una consulta reversa

En HTTP no se usaría

Por ejemplo, en SMTP sirve para chequear correos de spam, si el reverso da que la IP no se corresponde con la dirección de envío, es más probable que sea spam

Organización en el DNS

Recursos de Internet: Sistema distribuido pero regido por organizaciones

Organización mediante dominios, sub-dominios y host o servicios (jerárquico)

IANA a través del ICANN (Internet Corporation for Assigned Names and Numbers) controla el funcionamiento

Organizaciones paralelas: Open Root Server Network (ORSN), OpenNIC

ICANN homologa y delega gTLD a DNS Registrars y ccTLD a países

RIRs (Regional Internet Registers):

- American Registry for Internet Numbers (ARIN)
- RIPE NCC -Europa y parte de Asia- (RIPE)
- Asia-Pacific Network Information Centre (APNIC)
- Latin American and Caribbean NIC (LACNIC)
- African Network Information Centre (AfriNIC)

Nombres se delegan a países y a registrars, direcciones IP no

Delegación de autoridad ejemplo

ada.info.unlp.edu.ar “Ada” fue registrada por la administración de la red de la Facultad de Informática

El administrador de la Facultad obtuvo previamente la autoridad sobre el dominio “info.unlp.edu.ar” a partir de la administración de la universidad UNLP

La Universidad obtuvo autoridad sobre el dominio “unlp.edu.ar” a partir de la administración de “edu.ar”, RIU (Red Inter-universitaria)

La RIU obtuvo autoridad sobre “edu.ar” a partir de la delegación de NIC.AR que depende de alguna organización de gobierno, como la Sec. Legal y Técnica u otro ente a cargo de “.AR” (Argentina)

La administración de nombres en la Argentina, sea la Secretaría Legal y Técnica u otro ente obtuvo la autoridad delegada a partir del IANA o ICANN

Base de Datos Distribuida

Mantener gran cantidad de información

Control de la información distribuida (delegación)

Tolerante a fallos y escalable

Modelo de acceso (altamente cacheable):

- Muchas lecturas
- Pocas escrituras
- Consistencia relajada

Zona “.” (Raíz)

Punto de inicio (Bootstrap)

Actualidad: 13 ROOT Servers distribuidos en todo el mundo

7 de los cuales trabajan con redundancia y las réplicas están distribuidas geográficamente

Redundancia, combinada con Ruteo Anycast

Funcionamiento de DNS

Modelo cliente/servidor, Request/Response

También hay diálogo entre los servidores

El protocolo corre sobre UDP y TCP, puerto 53

El cliente escoge cualquier puerto no privilegiado

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)

Clientes: resolver + cualquier aplicación que requiera la resolución de nombres

En Unix el resolver es un conjunto de funciones C library (libc)

Otras implementación: Smart Resolver -> es un servidor local en cada equipo, puede hacer caching

Servidores: BIND (Berkeley Internet Name Domain/Daemon) de ISC; UNBOUND

Tipos de Servidores

- Servidor Raíz:
    - Servidor que delega a todos TLD (Top Level Domains)
    - No debería permitir recursivas
- Servidor Autoritativo:
    - Servidor con una zona o sub-dominio de nombres a cargo
    - Podría sub-delegar
    - No necesita hacer consultas adicionales para responder, ya que tiene la respuesta porque es quien define el registro
    - En el ejemplo serían ns1.riu.edu.ar y anubis.unlp.edu.ar
- Servidor Local/Resolver (no es el mismo que el resolver de la libc) 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
    - Podría ser el smart resolver
- 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:

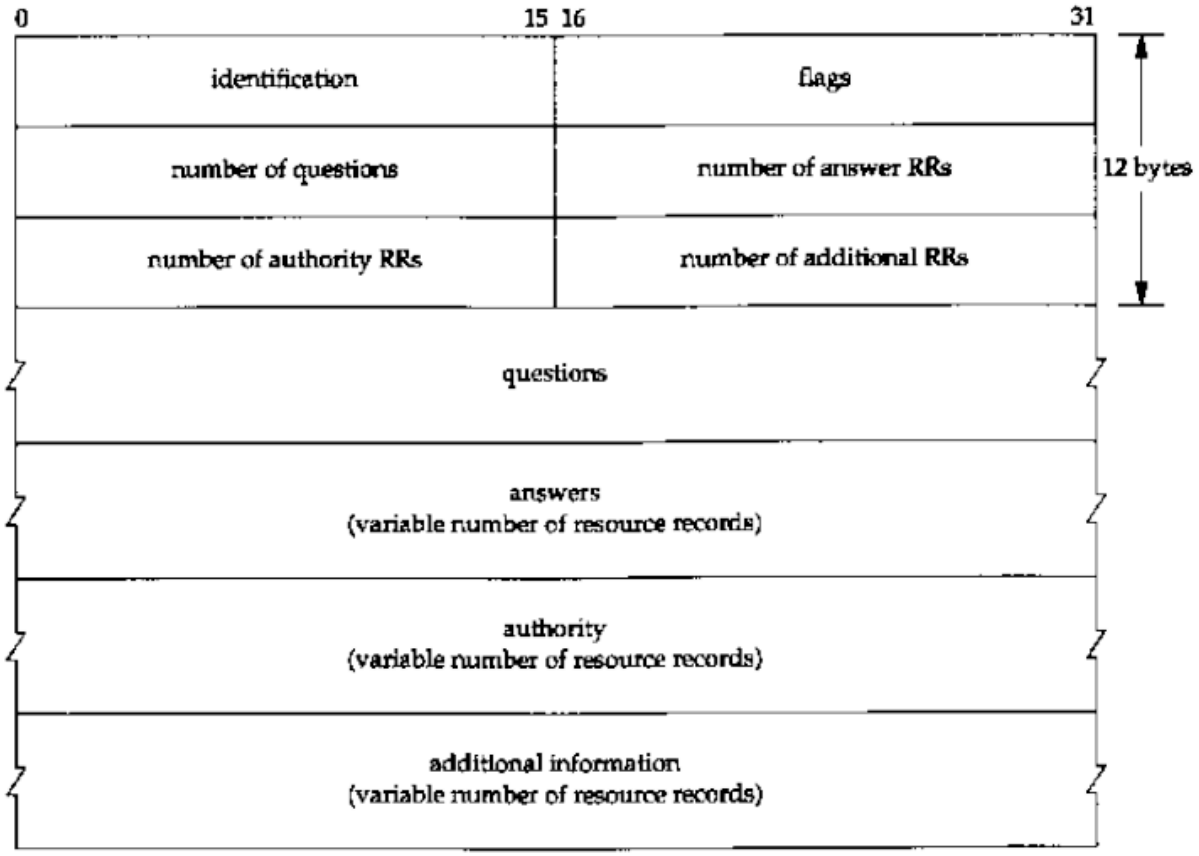
- Solo una cuestión de implementación
- Donde se modifican los datos realmente

Consultas recursivas vs iterativas

Recursivas son las consultas que se propaga sin volver al origen

Iterativas son las consultas cuya respuesta vuelve a quien hizo el requerimiento y dicho servidor volverá a realizar otra consulta, o no lo hará, dependiendo de la respuesta del servidor consultado (si la respuesta nos da lo que buscábamos, no se volverá a hacer otra consulta)

Estructura de un mensaje de DNS



Servicios y registros de DNS

Servidor de DNS almacena la información formando base de datos (DB) de RR (Resource Records)

No necesariamente es DB relacional

Registros

A/AAAA (Address): nombre → IP/IPv6

PTR (Pointer): IP → nombre

CNAME (Canonical Name): nombre → nombre

Registros HINFO (Hardware Info): nombre → info

TXT (Textual): nombre → info

MX (Mail Exchanger): nombre-dom → mail exchanger(s)

NS (Name Server): nombre-dom → dns server(s)

SOA (Start Of Authority): params. de dominio

Clase 6 - FTP

File Transfer Protocol

Conforma el grupo de los protocolos más viejos de la Internet aún utilizados

Existió antes de TCP/IP, ejecutaba sobre NCP

El protocolo original ha sufrido varias, adaptado a IP, la esencia es la misma

FTP es un protocolo para copiar archivos completos

Mensajes se codifican en ASCII estándar (de 7 bits codificados en 8)

Modelo cliente/servidor, command/response

Protocolo 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://....

Funcionamiento

Usa 2 (dos) conexiones TCP:

- Conexión de control (Out-Of-Band Control): port 21
- Conexión para la transferencia de datos: port 20 (o algún otro puerto definido por el servidor en FTP pasivo)

Cada conexión tiene requerimientos servicios diferentes:

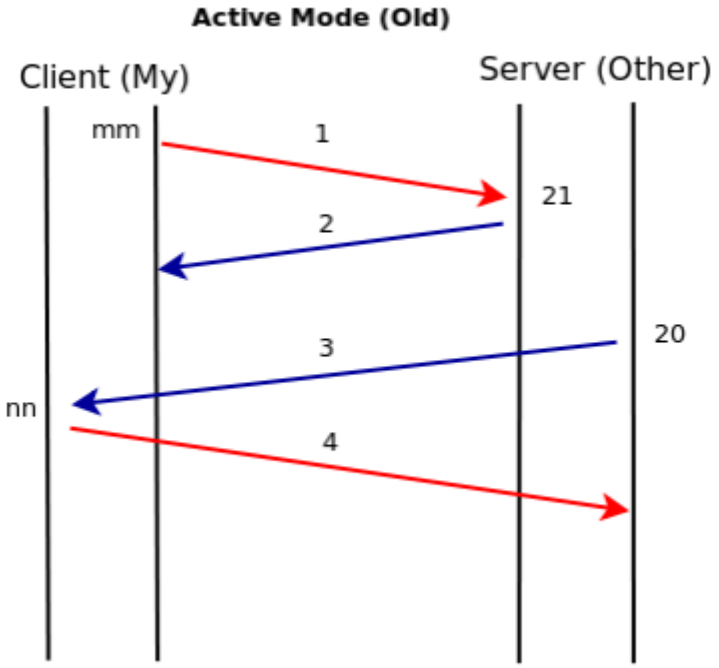
- Conexión de Control: min delay (minimizar el retraso/latencia)
- Conexión de Datos: max throughput (maximizar el rendimiento o ancho de banda. No es tan importante la latencia)

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 de cierra bajo demanda

El estado de cada operación se transmite por el canal de control



Comandos

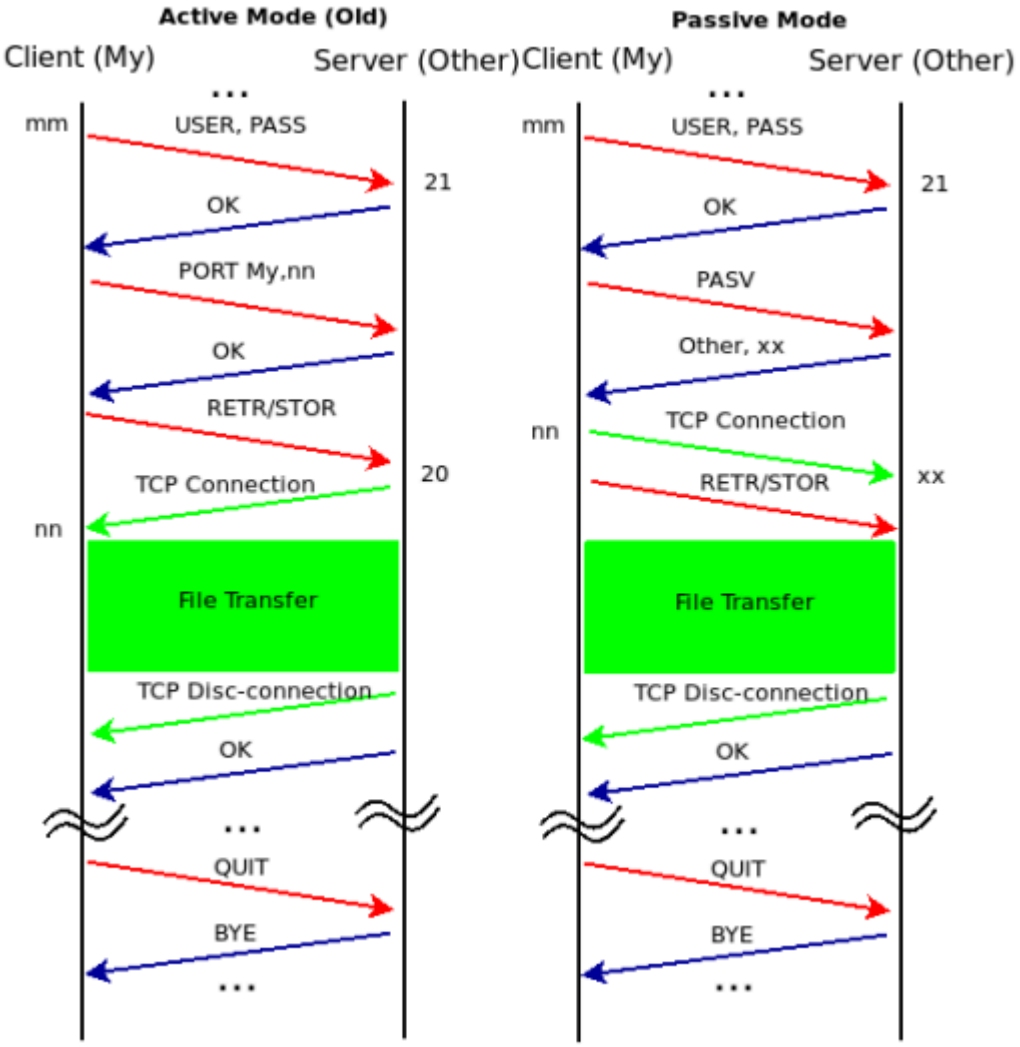
- Comandos que requieren conexión de datos:
    - RETR: obtener un archivo desde el servidor. A nivel de interfaz de usuario el comando que lo inicia es el get
    - STOR: envía un archivo al servidor. A nivel de interfaz de usuario el comando que lo inicia es el put
    - LIST: envía un petición de 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
- Comandos que no requieren conexión de datos:
    - 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 directorio actual, borra y crea dir
    - MODE: cambiar el modo
    - HELP: obtener ayuda

Modalidades de FTP

- FTP Activo (modalidad vieja):

- Conexión de control: port 21
- Conexión de datos: port 20
- Se diferencia como maneja la conexión de datos
- El servidor de forma activa se conecta al cliente para generar la conexión de datos
- PORT h1,h2,h3,h4,p1,p2
- Ej.: PORT 127,0,0,1,4,3 == 127.0.0.1:1027 (4*256)+3

- FTP 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
  - PASV 227 h1,h2,h3,h4,p1,p2
  - Ej.: 227 127,0,0,1,4,3



Formato de datos

FTP tiene funcionalidad de la capa ISO L6(representación)

Debido a los diferentes tipos de plataformas, las 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

Los tipos son:

- 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

Formato de archivos

Las plataformas (OS) pueden almacenar los archivos en diferentes estructuras

FTP define estructuras para transportar datos

Se especifica el formato para transferencia con el comando STRU

Formatos:

- File F Unstructured, sequence of bytes (default)
- Record R Series of records
- Page P Series of data blocks (pages)

Modo de transferencia

MODE se usa para especificar una codificación adicional aplicada sobre los datos transmitidos, de forma independiente del formato del archivo

Modos:

- 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:
    - El archivo se envía como header más secuencia de bloques
    - Permite interrumpir y reiniciar
- Compressed C:

    - Datos comprimidos usando RLE: Run Length Encoding. BBBBBBNN == 6B2N
    - Habitualmente no soportado

Alternativas

- Versiones de FTP seguras: FTPS, FTP over SSL/TLS
- Versiones integradas con la suite de Open-SSH:
    - SCP (Secure remote Copy)
    - SFTP (Secure FTP)
- Aplicación para compartir recursos:
    - NFS
    - SMB/CIFS
    - iSCSI
- TFTP

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

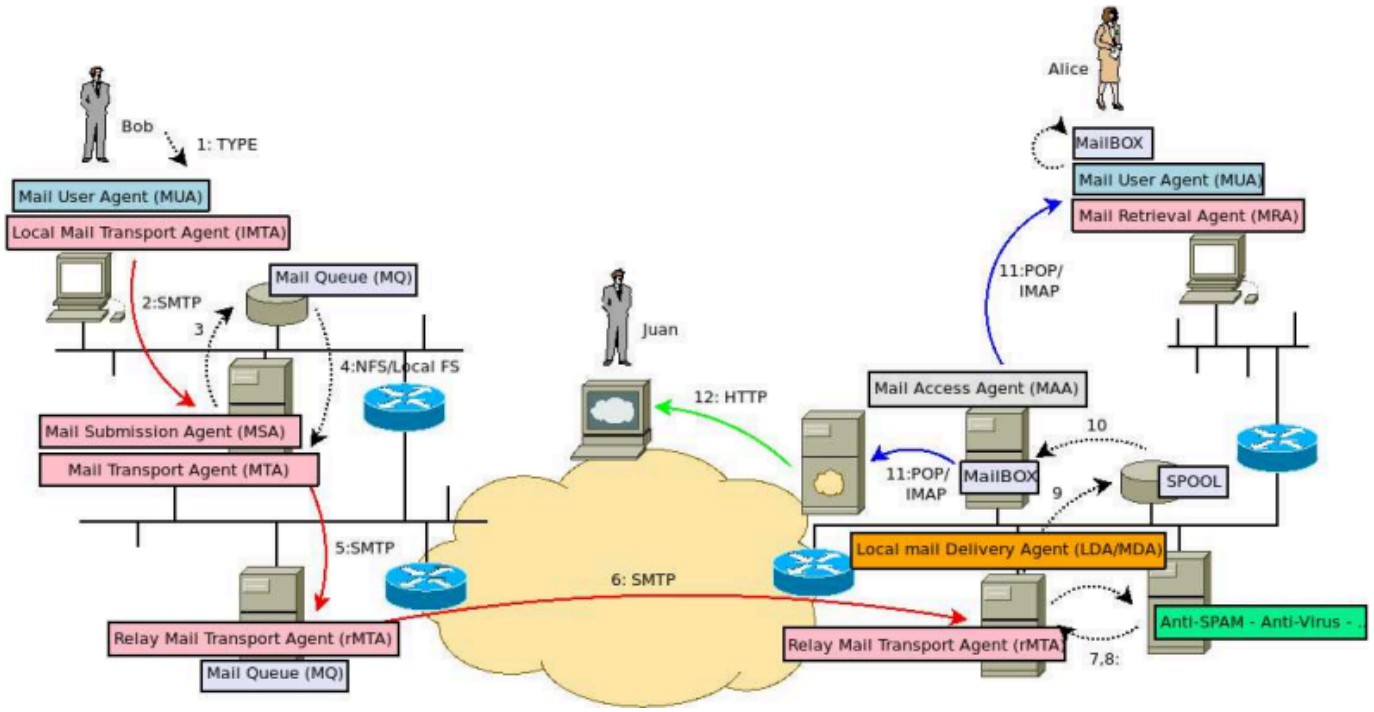
Clase 7 - Mail

Uno de los primeros servicios de Internet

Creado por Ray Tomlinson en 1971

Arquitectura

- Componentes principales:
  - 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)
- Componentes secundarios:
  - Servidor de Autenticacion externo
  - Web Mail (Front-End WWW)
  - Servidor de Anti-Virus, Anti-SPAM, Servidores de Listas, etc.



El MUA deposita el mail en el servidor de correo local, el cual se conecta mediante SMTP al servidor de mail saliente, que se va a conectar, también por SMTP al servidor

donde tiene la casilla de correo el usuario receptor. Allí se hacen controles de anti-spam, anti-virus.

Una vez el mail está en la casilla de usuario, mediante POP o IMAP se puede acceder a ver el mail (ya sea por la web o con un MUA)

Mail User Agent (MUA)

Cliente

Interfaz con usuario

Lector/Editor/Emisor local de correos (e-mails)

Posee integrado un MRA para comunicarse con el Servidor de Mail Entrante, MAA (Mail Access Agent)

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, etc.

Utilizan protocolos SMTP o ESMTP, POP o IMAP, habla con MTA y con MAA propio

Ejemplos: Eudora, MS Outlook, Mozilla Thunderbird, elm, pine, mutt

Web-mailers: Horde/IMP, Squirrel, GMAIL, Yahoo, Hotmail

Mail Submission Agent (MSA)

Servidor

Agente 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:, etc.

Termina de dar formato al header del e-mail

Utiliza protocolo SMTP, ESMTP

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

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

Mail Transport Agent (MTA)

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 esta la casilla de mensaje destino, comunicación de MTA a MTA

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

Ejemplos: Postfix, Sendmail, MS Exchange, Exim, Qmail

Mail Delivery Agent (MDA)

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

Se lo llama tambien LDA (Local Delivery Agent)

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

Ejemplos: procmail, postfix local, Sendmail, courrier, cyrus-IMAP, dovecot, sieve, fetchmail(mas un MRA), getmail(mas un MRA)

Mail Access Agent (MAA)

Servidor

Integrado o separado del MDA

Autentica al MUA/usuario y lee los e-mails del mailbox local dejados por el MDA/LDA

Transporta 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

Ejemplos de MDA/LDA: courrier, cyrus-IMAP, dovecot, sieve

SMTP

Simple Mail Transffer Protocol

Protocolo Cliente/Servidor

Utiliza fromato ASCII 7 bits en 8 NVT

Usa TCP puerto servidor: 25

Los LMTA de los MUA hablan SMTP con su servidor SMTP saliente

Los servidores SMTP hablan entre sí este protocolo

RFC-821 (SMTP), extendida por RFC-1869 (ESMTP) y Redefinida por RFC-2821

Requiere finalizar con CRLF.CRLF

Usa conexiones persistentes

Trabaja de forma Interactiva (Requerimiento/Respuesta) o Pipeline

Puede o no requerir Autenticacion

Puede o no trabajar de forma segura: SSL/TLS

Formato de un mensaje

El concepto de Envelope fue definido en RFC-821

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 informacion del mail: Subject:, From:, To:, Return-Path: X-Mailer:, X-...:

Body (Cuerpo), separado por línea en blanco del header: Contenido del e-mail

Extensiones para enviar datos binarios

MIME (Multipurpose Internet Mail Extensions): Definido en RFC-1521 y RFC-1522. Redefinidas en RFC-2045 y RFC-2046

Con tags especiales indica el tipo al sistema destino

Luego codifica el mensaje binario en formato que no viole US-ASCII (NVT)

Algunos Ejemplos: uunecode, base64, quotes-strings, etc

Protocolos de acceso a correo

Protocolos Acceso a Correo POP: Post Office Protocol, RFC-1939: POPv3

IMAP: Internet Mail Access Protocol, RFC-1730: IMAPv4

Requieren Autenticacion

Utiliza fromato ASCII 7 bits en 8 NVT

Usan TCP puertos servidor: 110 y 143

Permiten correr de forma segura sobre SSL/TLS

IMAP es más flexible permite uso de carpetas y manipulación de mensajes en el servidor