

Introducción a las Redes de Computadoras

Redes y Comunicaciones

- Preguntar:
 - Genéricamente: Qué es una Red ?
 - Aunque seguramente no surja naturalmente la pregunta, sino la cuestión sería más específica:
 - Qué es Internet ?
 - Qué es la WEB ?
 - Qué pasa cuando navego por Internet, Accedo a Facebook, o miro un video en Youtube ?
- Optaremos por un enfoque de lo general a lo particular preguntándonos: **Qué es una red de computadoras / ordenadores?**

Qué es una red de computadoras/ordenadores ?

- Análisis del punto de vista sistémico:
- **Red de Computadoras:** un grupo de computadoras/dispositivos interconectados.
- **Objetivo:** compartir recursos: dispositivos, información, servicios.
- El conjunto **computadoras, software de red, medios y dispositivos de interconexión** forma un sistema de comunicación.
- 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 un Sistema de Comunicación

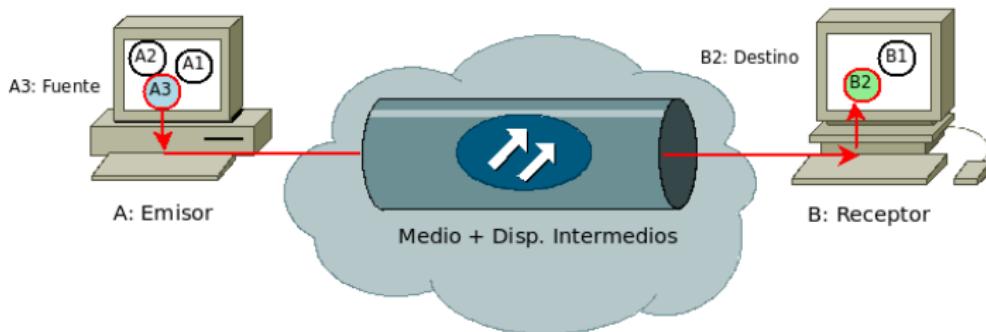


Figure: Sistema de Comunicaciones.

Componentes de una Red

- Fuera del punto de vista sistémico podemos ver un gran numero de componentes:
 - Computadoras, en el modelo de Internet: Hosts (PCs, laptops, servidores).
 - Routers/switches, Gateways, AP (Access Points).
 - NIC (placas de red), Modems.
 - Vínculos/ enlaces: conformados por:
 - Medios: cables, fibras ópticas, señales electromagnéticas, antenas, interfaces, etc.
 - Programas: Browsers, Servidores Web, Clientes de Mail, Servidores de Streaming.
 - Etc...
- Las componentes de la red deben interactuar y combinarse a través de reglas.

Protocolos

Protocolo: El conjunto de conductas y normas a conocer, respetar y cumplir no sólo en el medio oficial ya establecido, sino también en el medio social, laboral, etc.

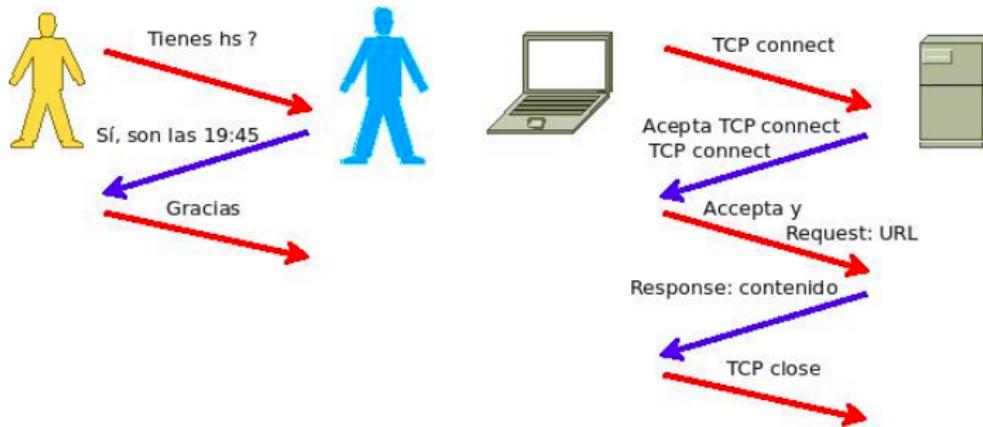


Figure: Protocolo

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 las componentes.

- A principios de los 80', las compañías comenzaron a implementar redes propias (privadas y cerradas).
- Primeras **Redes Propietarias**.
- Consecuencia: Cada red tenía sus especificaciones propias (protocolos).
- Resultados: Incompatibilidad. La comunicación entre redes era muy difícil, evolución más lenta, carencia de estándares.
- Complejidad de modelos.

- La cantidad de componentes de red a interactuar genera complejidad, se requiere una organización de las mismas.
- Se requieren **Modelos de Organización**.
- Modelo en Capas: **Layering**, divide la complejidad en componentes reusables.
 - Reduce complejidad en componente más pequeñas.
 - 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)

- Necesidad de desarrollar componentes estándares de red.
- Resultado: La ISO (International Standard Org.) crea el modelo OSI en 1984.
- Basado en los modelos de red (en capas):
 - DECNET (Digital).
 - SNA (IBM).
 - TCP/IP (DoD USA - Dept. of Defense USA).
- Modelo abierto y estándar.
- Modelo dividido en 7 (siete) capas.
- Modelo de Referencia.

Modelo OSI (Cont'd)

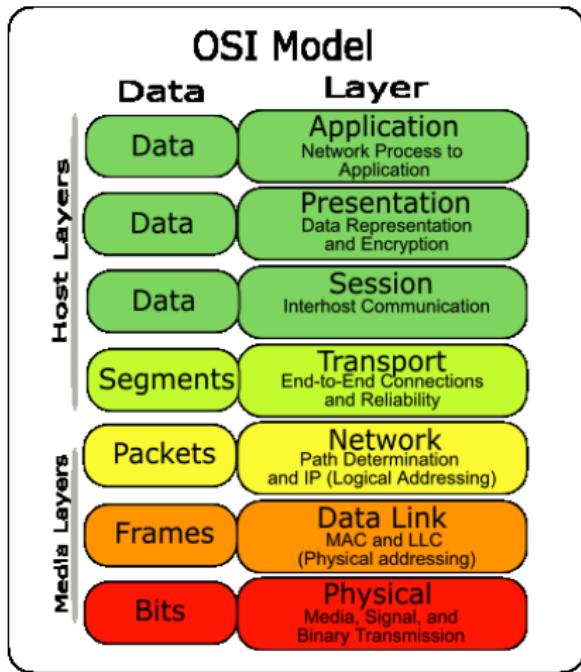


Figure: Modelo OSI

Modelo OSI (Cont'd)

- Modelo en Capas: capa ofrece servicios a la capa superior, usa servicios de la capa inferior, mediante interfaz.

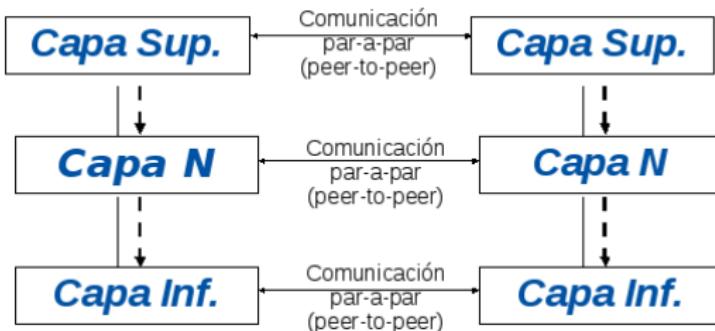


Figure: Comunicación en capas

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 (OSI)

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

Presentación/Representación (6): formato de los datos.

Sesión (5): mantener track de sesiones de la aplicación.

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

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

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

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

Ejemplos de Implementaciones por Capa (OSI)

Aplicación (7): Telnet, HTTP, DNS, FTP, DNS, NCP, NDS, X.400.

Presentación/Representación (6): Postscript, JPEG, PNG, TIFF, MPEG, ZIP, XDR, ASN, HTML, CharSets(ASCII, ISO-8859-1, UTF-8, EBDIC).

Sesión (5): RPC de NFS, SQL, NetBIOS.

Transporte (4): TCP, UDP, SPX, ISO-TP.

Red (3): IP, ICMP, OSPF, IPX, CLNP, IS-IS.

Enlace de Datos (2): Ethernet, 802.3, 802.11, PPP, HDLC.

Física (1): RJ-45, EIA/TIA-568C, V.24, V.35, G.703(TEL), G.652(FO) RS-232, DOCSIS(Coax).

- 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 5 (cinco) capas:
 - Capa de Aplicación (Process/Application).
 - Capa de Transporte o Host-to-Host.
 - Capa de Internet o Internetworking.
 - Capa de Enlace(Link Layer).
 - Capa de Física.
- Por simplicidad algunos autores hablan de 4 capas, agrupando a la Capa de Enlace y Capa física en una sola capa que llaman Capa de acceso a la Red.

Modelo TCP/IP (Cont'd)

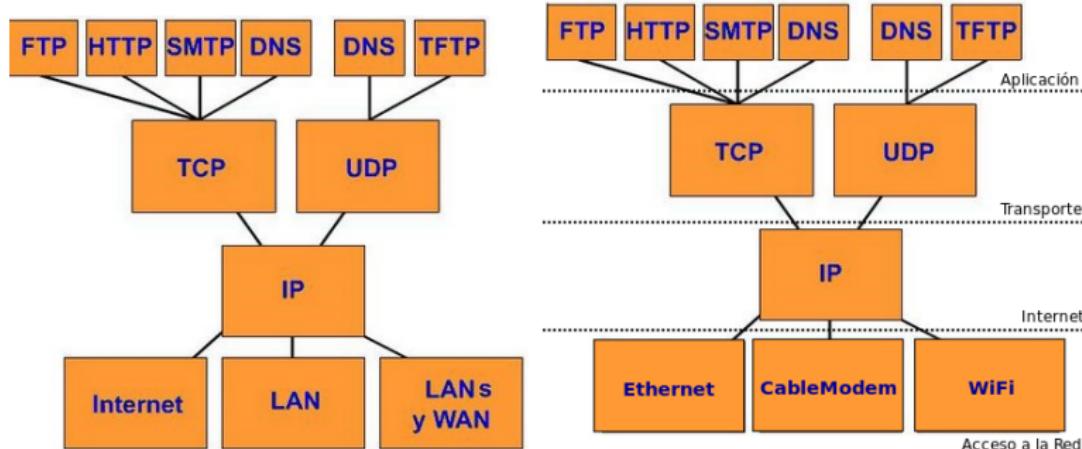


Figure: Modelo TCP/IP

Comparación: OSI vs. TCP/IP

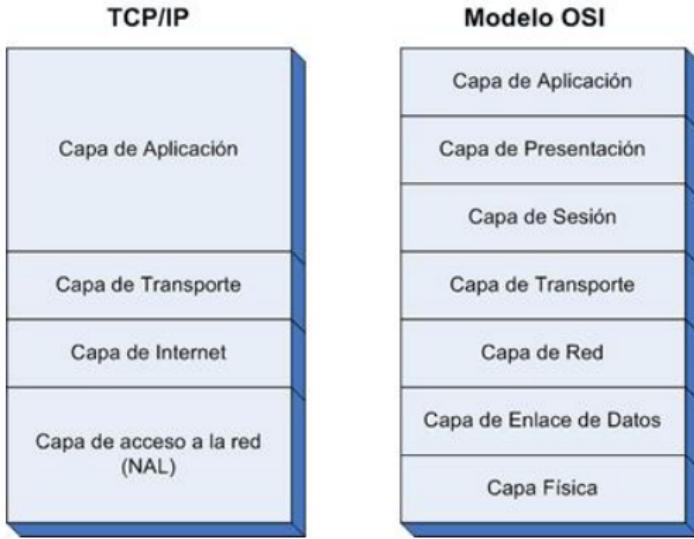


Figure: Modelo OSI vs TCP/IP

- Similitudes:

- Ambos se dividen en capas.
- Ambos tienen capas de aplicación, aunque incluyen servicios distintos.
- Ambos tienen capas de transporte similares.
- Ambos tienen 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.

Comparación: OSI vs. TCP/IP

- 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 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” de referencia, teórico, aunque hay implementaciones.

Encapsulamiento

- Cada capa define su PDU: Protocol Data Unit.

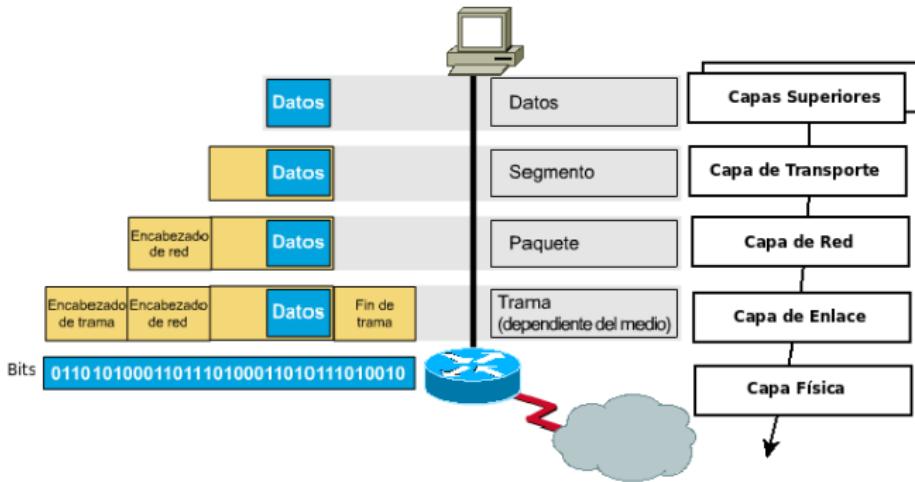


Figure: Encapsulamiento y PDUs

Dispositivos y Capas

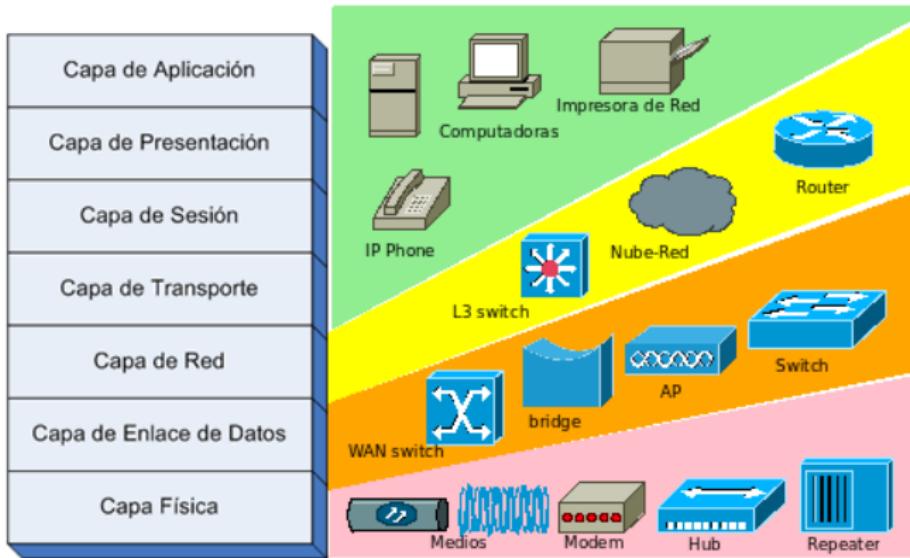


Figure: Dispositivos y Capas

Comunicación entre Capas Peer-Peer

- Cada capa usa el servicio de la de abajo.
- Cada capa se comunica con la capa del otro extremo.

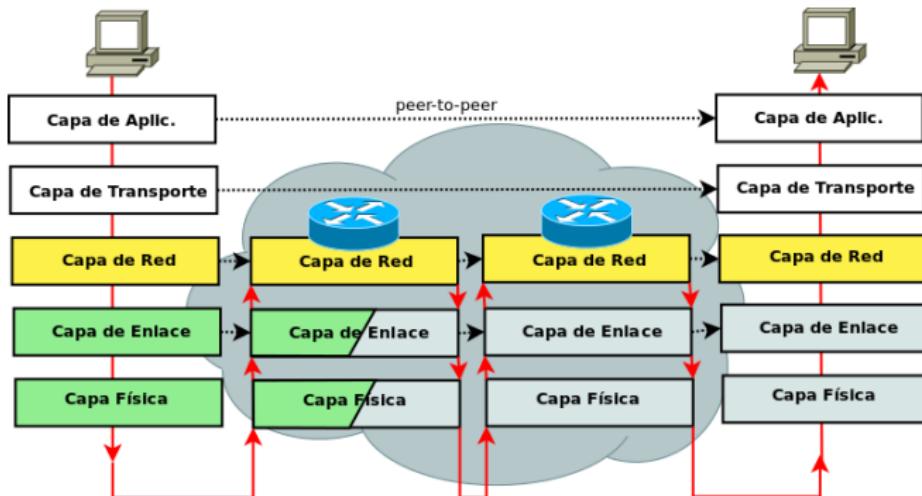


Figure: Comunicación Peer-Peer

Comunicación entre Capas Peer-Peer(Cont'd)

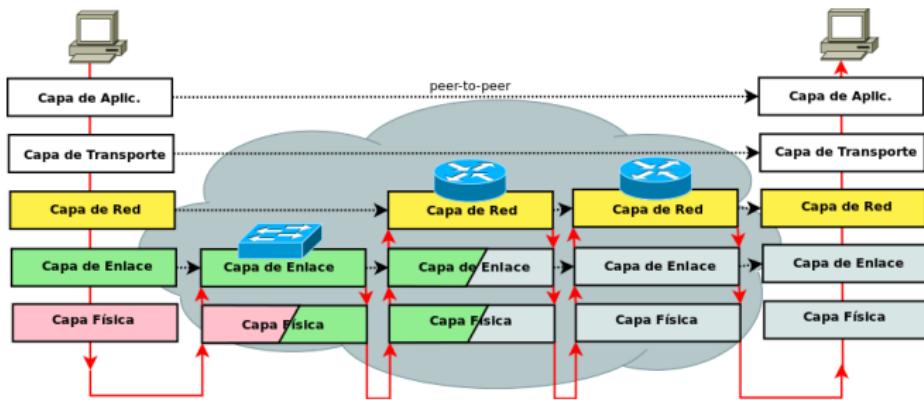


Figure: Comunicación Peer-Peer

- Diferentes clasificaciones de acuerdo a diferentes aspectos.
- Se pueden mencionar:
 - Clasificación por cobertura, distancia, alcance.
 - Clasificación por acceso abierto o privado.
 - Clasificación por topología física.
 - Clasificación por tipo de conexión/medio.
 - Etc.

Clasificación por Cobertura

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: 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 AN, NFC), ...

Internet: red pùblica global, tecnologìa TCP/IP.

Intranet: red privada que utiliza la tecnologìa 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 pùblica como Internet.

Clasificación Física de Redes

- Redes de Conmutación de Circuitos.
- Redes de Conmutación de Tramas/Paquetes.
 - Servicios Orientados a Conexión. Circuitos Virtuales.
 - Servicios NO Orientados a Conexión. Datagramas.

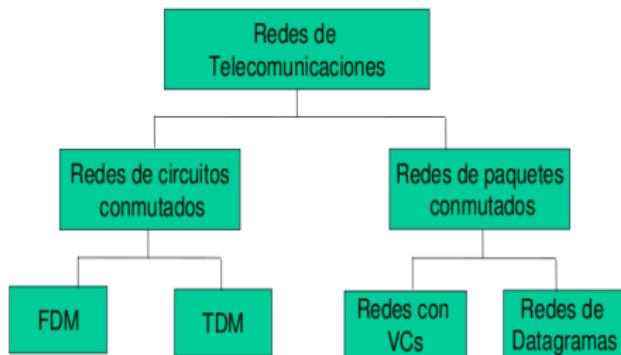


Figure: Clasificación de Redes

- 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

- Modelo de forma de reloj de arena (hourglass):

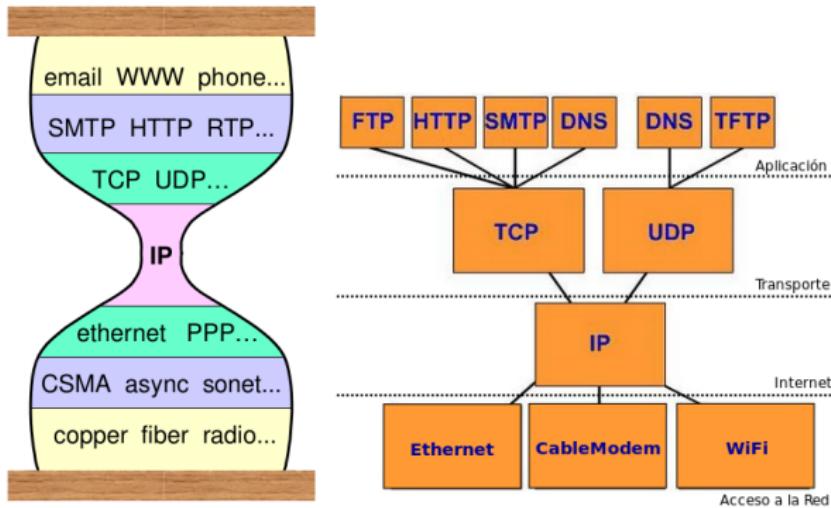


Figure: Modelo hourglass.

Modelo Simplificado de Internet

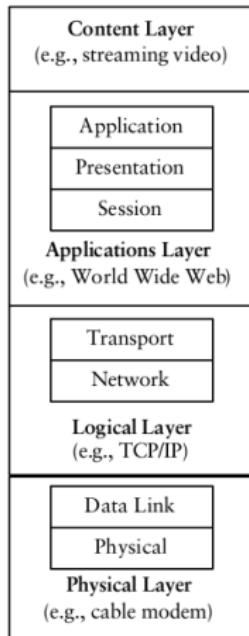


Figure: Modelo Simplificado de Internet de 4 capas

Qué es Internet ? (Cont'd)

Qué computadoras: PCs, Mainframes, Celulares, Laptops,
Handhelds, Supercomputadoras, autos, heladeras, etc

...

Computadoras Especiales: routers y switches (sucesores de IMPs
-Interface Message Processors-).

Qué medios: cobre, fibra óptica, wireless, satélites, etc.

Qué información: de todo !!!!! (de forma digital).

Objetivos/Historia de Internet

Inicios de 1960': Red militar para la guerra fría ?? (aún no existía TCP/IP). Packet Switching Theory: paper de Kleinrock, usar paquetes en lugar de circuitos en 1960. ARPANET: RAND Corp, Leonard Kleinrock del MIT trabajan sobre la red, BBN implementa IMPs.

Primera vez On-Line 1969: Conectaba las Universidades: Stanford (SRI), Utah, UCLA, UCSB (UC Santa Barbara).

Nuevo protocolo LAN 1973: Ethernet, Bob Metcalfe en Xerox PARC.

Cambio a TCP/IP 1983: Desde NCP a TCP/IP. Vinton Cerf y Robert E. Kahn.

Luego, NSFNET 1985: Red Científica e Investigación, Usada por las Universidades.

Continuando 1988: Comienza como negocio, nuevas oportunidades.

Hoy 2020: Tele-trabajo, *Clases Virtuales*, Redes Sociales,



- Categoría **STANDARD TRACK**. RFC maturity levels.
 - Proposed Standard:** no se requiere implementaciones. Se asigna RFCnnnn.
 - Internet Standard (STD):** existen implementaciones y significante experiencia operacional. Se retiene el RFCnnnn y se agrega STDxxxx.

- Otras Categorías: “**Off-track**” .

INFORMAL/EXPERIMENTAL: otro proceso, se publica como Internet Draft, pero se coloca en otra Cat.

BCP (Best Current Practices): otro proceso.

HISTORIC (STD obsoletas): las RFCs se van actualizando o se pueden declarar obsoletas por otras.

FYI: (For Your Information): como INFORMATIONAL.

RFC (Request for Comments) (Cont'd)

- Algunos ejemplos: <http://www.rfc-editor.org/rfcxx00.html>.
 - RFC 791: IP, STD 5. 1981.
 - RFC 792: ICMP. STD 5. 1981.
 - RFC 793: TCP, STD 7. 1981.
 - RFC 768: UDP, STD 6. 1980.
 - RFC 854: TELNET, STD 8. 1983.
 - RFC 1035: DNS, STD 13. 1987.
 - RFC 2460 (desde 1998), luego 8200: IPv6, STD 86. 2017.
 - RFC 1945: HTTP 1.0, Informational. 1996.
 - RFC 2616: HTTP 1.1, aun Std Track. 1999.
 - RFC 5735, BCP 153: Special Use IPv4 Addresses. 2010.
 - RFC 5721, Experimental: POP3 Support for UTF-8. 2010.
 - RFC 1149, Experimental: Standard for the transmission of IP datagrams on avian carriers. 1 April 1990. April Fools' Day.
 - RFC 1267, Historic: BGP Border Gateway Protocol 3, 1991, obsoleta por RFC 4271: BGP-4.
 - RFC 1983, (Informational) FYI 18: Internet Users' Glossary. 1996.

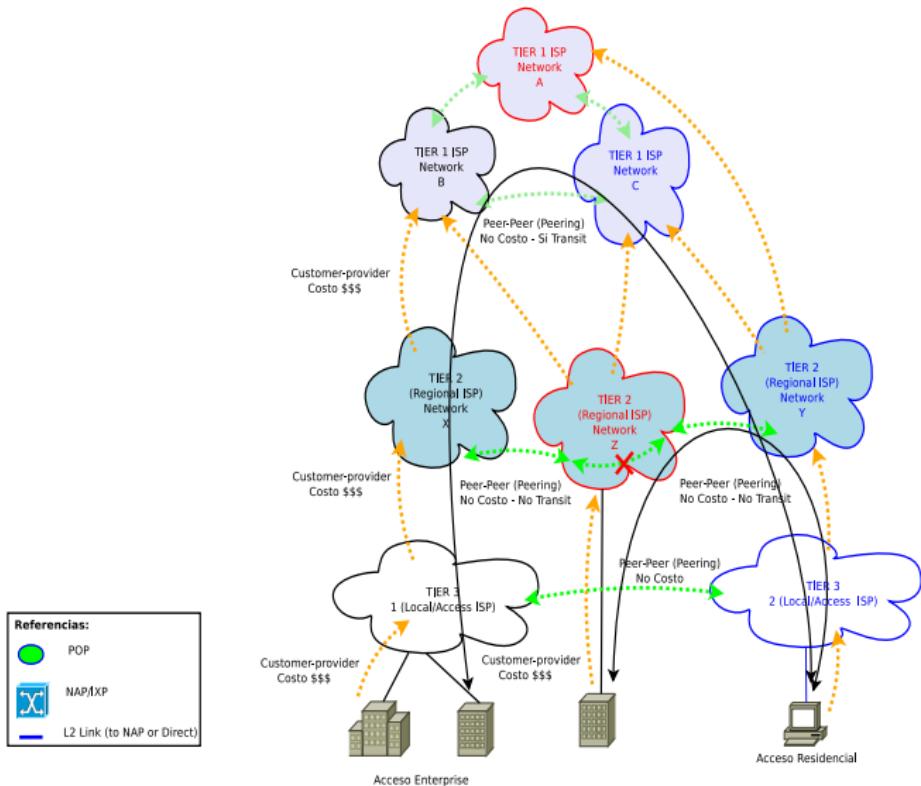
- Estructura en Jerárquica, en Tiers.
- **Capa de Acceso (Edge):** Acceso Residenciales, Acceso de Organizaciones.
- **Capa de núcleo (Core):** dividida en diferentes niveles.
 - Proveedores Regionales (Regional ISPs).
 - Proveedores Nacionales.
 - Proveedores Internacionales.
 - Proveedores Internacionales en el Tier 1.

- Diferentes tecnologías de última milla (acceso) y de redes locales.
- Despliegue local y de acceso de las personas y las organizaciones.
- Ver capa física y enlace modelo OSI.
- Últimos eslabones en la jerarquía.

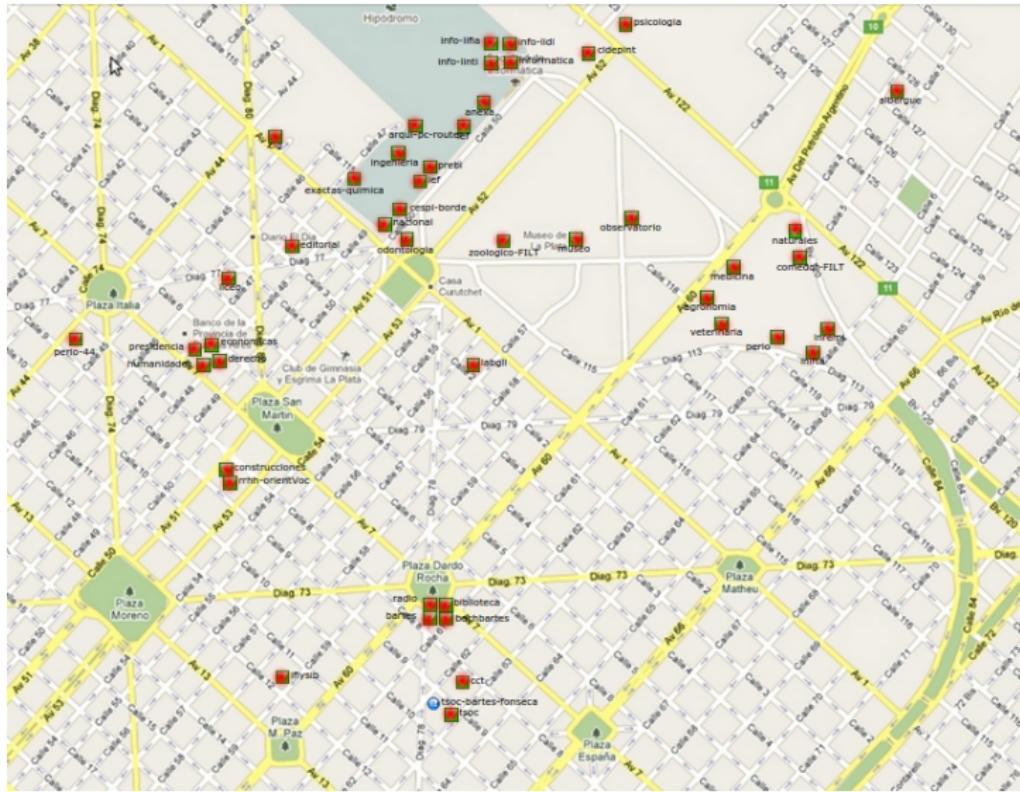
Estructura de Internet (Core)

- Tecnologías de fibra óptica, Cobre, Satélites.
- Se interconectan mediante POPs (Point Of Presence) con Proveedores.
- Entre proveedores se interconectan mediante NAPs (Network Access Point) o conexiones Directas.
- Actualmente los NAPs se los llama IXPs (Internet Exchange Point)

Estructura de Internet (Core Cont'd)



Ejemplo de la Cobertura de la UNLP



- Concepto de protocolo y modelos en capas.
- Modelo OSI, Modelo TCP/IP, PDUs.
- Comunicación peer-to-peer.
- Clasificación de redes.
- Modelo TCP/IP en modelo de Internet.
- Estructura de Internet.
- Estándares de Internet RFCs.

- Kurose/Ross: Computer Networking (6th Ed).
- Andrew S. Tanenbaum. Computer Networks (5th Edition).
- Willam Stallings. Data & Computer Communications (8th Edition).
- Wikipedia <http://www.wikipedia.org>.
- <http://www.rfc-editor.org/overview.html>.
- <ftp://ftp.rfc-editor.org/in-notes/rfc2026.txt>.
- <http://www.isoc.org/internet/history/brief.shtml>.
- <http://isoc.org/wp/ietfjournal/?p=454>.
- Internet ...

Introducción a Capa de Aplicación

Redes y Comunicaciones

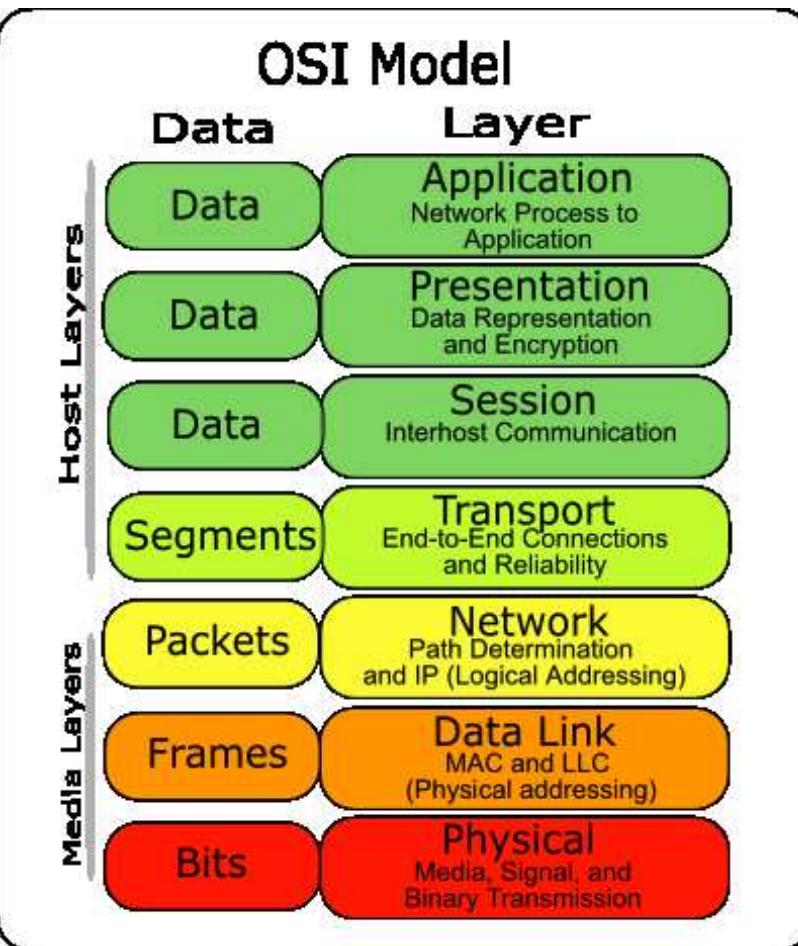
Funciones de la Capa de Aplicación

- Provee servicios de comunicación a los usuarios (Capa 8 ;) y a las aplicaciones, incluye las aplicaciones mismas.
- Existe modelo de comunicación Machine to machine (M2M), no hay usuarios (personas).
- Interfaz con el usuario -User Interface (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 principio end-2-end.
- Qué cubre?: Se verá el enfoque orientado a Internet, en el cual 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.

Modelo OSI



Capa de Sesión

- Administra las conversaciones/diálogos entre las aplicaciones.
- Podría proveer mecanismos transaccionales o de sincronización: COMMIT, CHECKPOINT, ROLLBACK.
- Maneja actividad.
- Informar Excepciones.
- 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.

Capa de Sesión (Consideraciones)

- Básicamente la Capa de sesión se podría ver como un invento de la ISO.
- Ninguna de las redes existentes tenían esta capa (salvo algunos servicios de modelo OSI aparecen en SNA).
- 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, EBDIC, 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 como debe ser el diálogo (intercambio de mensajes). Que mensajes se deben intercambiar.
- Ejemplo concreto: Protocolo HTTP y sus implementaciones mediante servidores WEB y browsers (navegadores).

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 como debe ser el diálogo (intercambio de mensajes). Que 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 Cliente/Servidor.
- Modelo Peer to Peer (P2P).
- Modelo Híbrido.

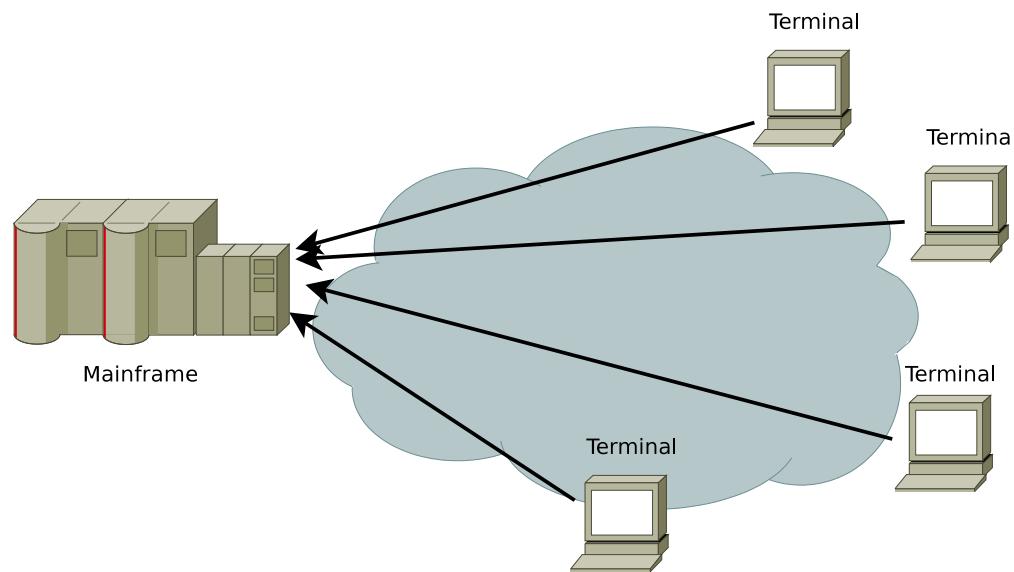
Modelo de Mainframe Centralizado

- Modelo de Carga concentrada.
- El cliente es “tonto” (dumb) solo corre la comunicación 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.
- Cliente ejecuta en el mainframe.
- Ejemplo: Sistema SNA con Mainframe IBM S/370 y terminales

3270 (las terminales verdes del antiguo sistema de alumnos !!!).

- Servidor de Terminales: X11, VNC, Cytrix Metaframe, VMWARE PCoIP, LTS, Virtualización.

Modelo de Mainframe Centralizado (Cont'd)



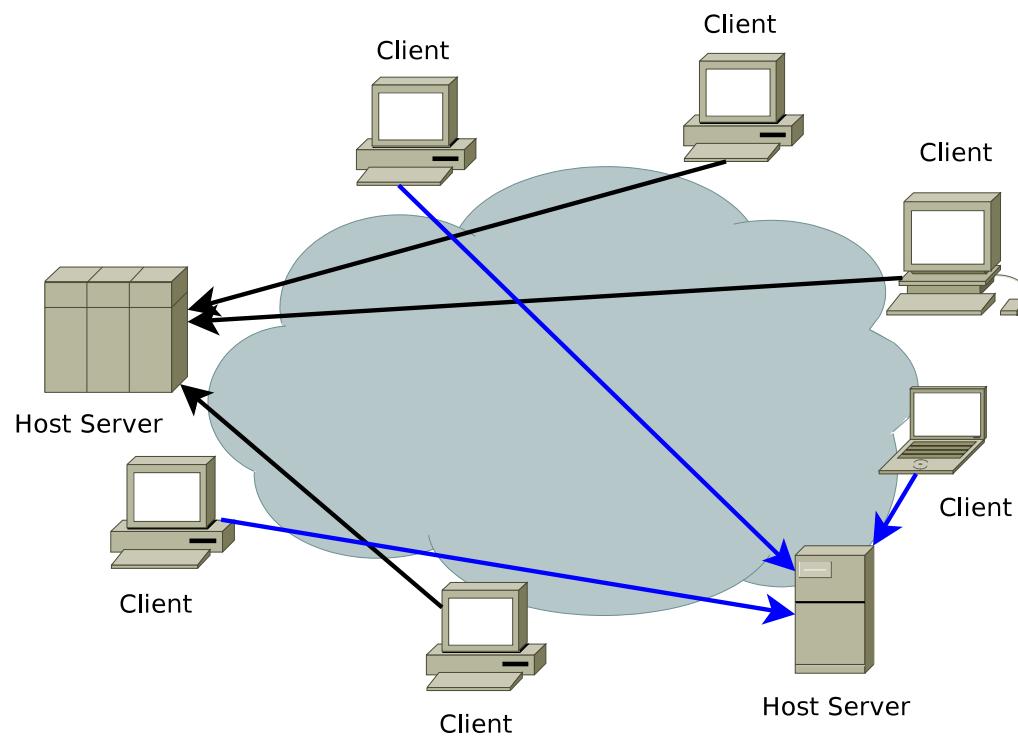
Modelo de Mainframe Centralizado (Cont'd)



Modelo de 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 servicio esperando de forma pasiva la conexión.
- Cliente se conectan al servidor y se comunican a través de este.
- Ejemplo: File Server vía NFS o FTP.
- Moldeo asimétrico 1 a N, M a N (donde $M < N$).

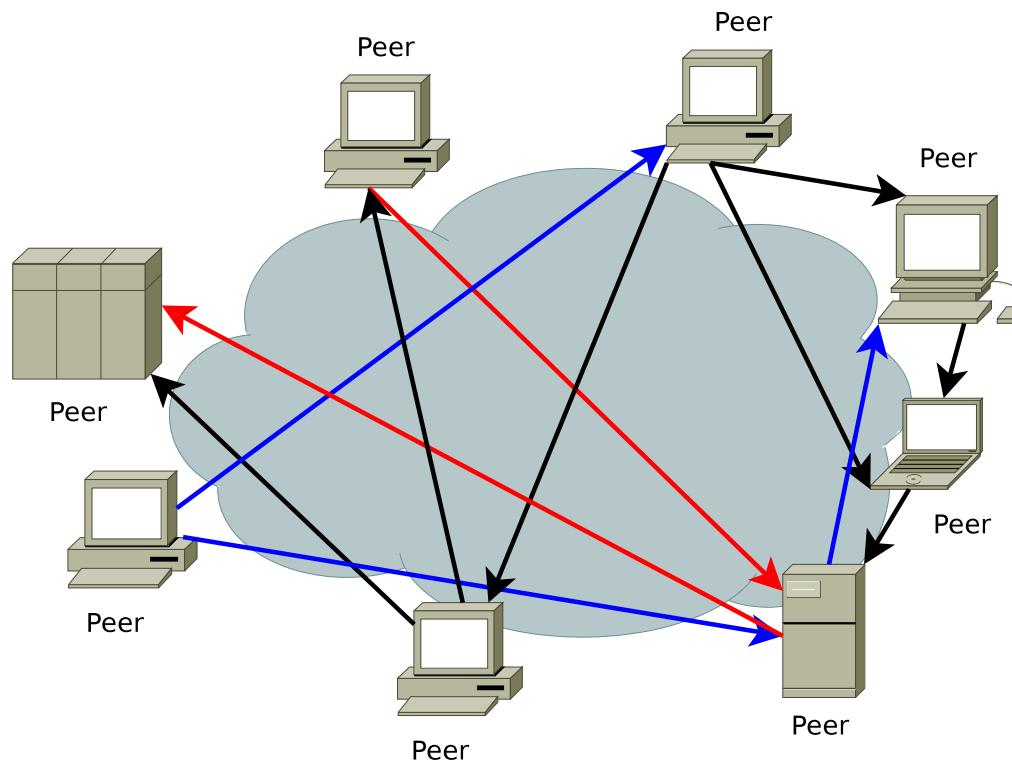
Modelo de Cliente/Servidor (Cont'd)



Modelo de Peer-to-Peer

- 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 de Peer-to-Peer (Cont'd)



Modelo de Peer-to-Peer Híbrido

- Modelo de Carga compartida y distribuida.
- Los peers (participantes) pueden cumplir rol de cliente, servidor o ambos en un instante.
- Existen diferentes tipos de nodos con diferente 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.
- Moldeo asimétrico M a N.

BitTorrent

- Protocolo para el intercambio de archivos grandes de forma masiva como peer-to-peer (P2P).
- Desarrollado originalmente por un programador, Bram Cohen (software libre).
- Cuando alguien quiere compartir un archivo genera un archivo .TORRENT.
- A diferencia de otros P2P no tiene índices de búsquedas, si existen WEBS que permite localizar archivos: Mininova, Thepiratebay, Ktorrents, TorrentReactor.

BitTorrent (Cont'd)

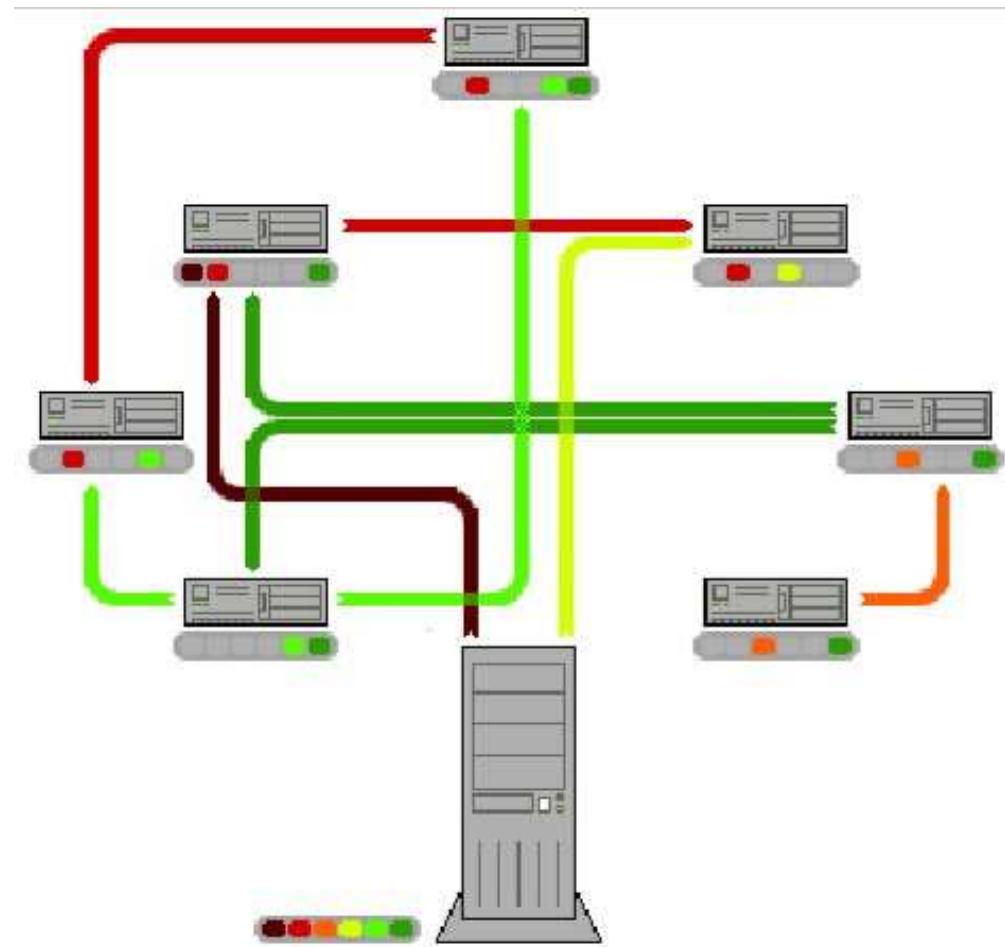
- .TORRENT. Dirección del TRACKER para unirse al grupo de PEERs (más información de archivo).
- Desde el TRACKER se obtiene PEERs: SEEDs (contienen archivo completo) y LEECHERs (contienen archivo parcial). El TRACKER se actualiza con el nuevo PEER.
- La comunicación con el TRACKER habitualmente se hace sobre HTTP, o podría ser sobre FTP.
- Los archivos se dividen en chunks, cada uno puede ser descargado de un PEER diferente.
- Se conecta con otros PEERs y comienza la descarga. Cada chunk se comparte con otros PEERs. Utiliza el port 6881 y escanea hacia arriba.

BitTorrent (Cont'd)

- Cuando baja trabaja de forma aleatoria o “rarest first algorithm”, trata de bajar los chunks con menos números de copias.
- Por cada archivo puede haber como máximo 4 PEERs remotos descargando. Los peers se rankean, se deja bajar de aquellos que mantienen una buena relación de bajada y subida con el PEER.
- Se seleccionan PEERs aleatoriamente cada determinado tiempo.
- BitTorrent puede trabajar con NAT, si el cliente se conecta (outbound connection) a otro PEER que tiene IP pública y esta conexión se usa para download y upload. Es conveniente habilitar port forwarding.

- Se puede restringir download y upload rate.

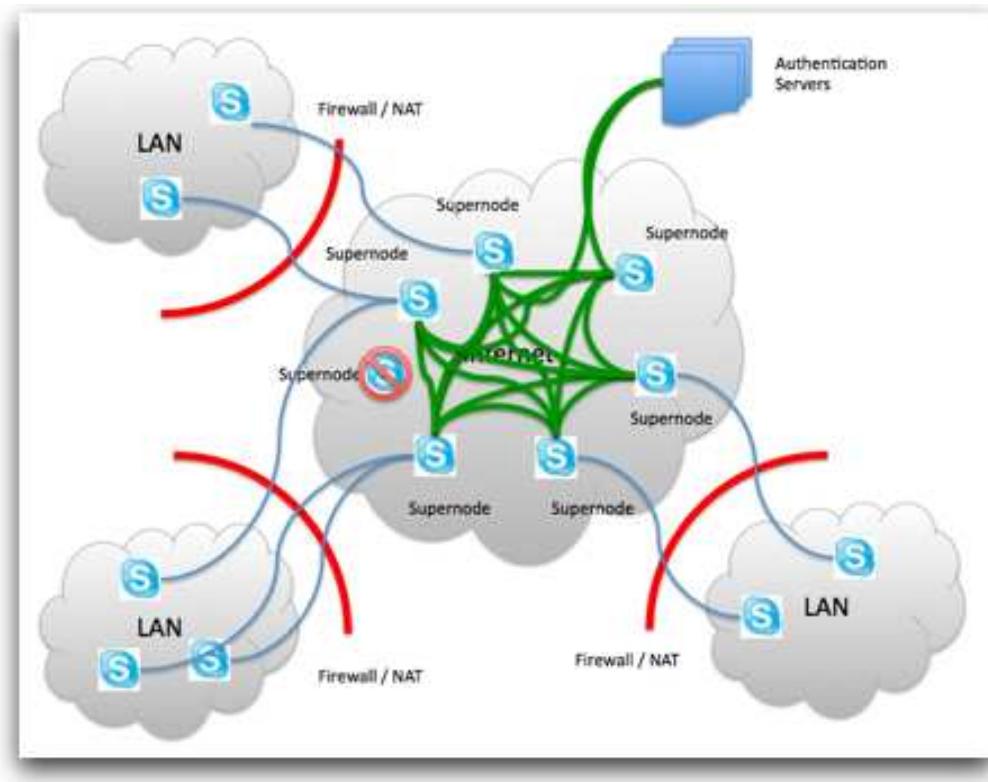
BitTorrent (Cont'd)



Skype

- Protocolo propietario que implementa VoIP.
- Nodos clientes pueden ser solo nodos o Super-nodos.
- Super-nodos actúan como directorios.
- Estos nodos hace de STUN, permitiendo conexión de computadoras detrás de NAT.
- Los super-nodos crean P2P Overlay networks (Una red sobre otra red).
- Corren el software tradicional de los clientes, pero sobre la Internet pública.
- Super-nodos se conectan entre si. Los Super-nodos utilizan servicios externos de autenticación.
- Existen Mega Super-nodos que pertenecen a Skype.

Skype (Cont'd)



Requerimientos de Aplicaciones

<u>Aplicación</u>	<u>Pérdidas</u>	<u>Bandwidth</u>	<u>Sensible a Time</u>
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: seguridad, tiempo de respuesta, confiabilidad, optimizar ancho de banda, etc...
- Diferentes Alternativas de Transporte.

- Transport Control Protocol (TCP).
- User Datagram Protocol (UDP).
- Otras alternativas: Stream Control Transmission Protocol (SCTP),
Reliable User Datagram Protocol (RUDP).

Direccionamiento de Procesos

- Las aplicaciones son implementadas por los SO como procesos.
- Conceptos de IPC (Inter-Process Communication).
- Para que un proceso reciba un mensaje, éste debe tener un identificador único.
- Identificador de host no suficiente, se agrega identificador de proceso (independiente del SO) número de puerto.
- Servicio de multiplexación provisto por nivel de transporte.
- Accedido mediante la API de **Sockets BSD**, Winsocks de Microsoft o TLI/XTI de AT&T.

Fuentes de Información

- Kurose/Ross: Computer Networking (6th Edition).
- Andrew S. Tanenbaum. Computer Networks (4th Edition).
- Wikipedia <http://www.wikipedia.org>.
- BitTorrent: <http://www.bittorrent.org>.
- Understanding Today's Skype Outage: Explaining Supernodes, Dan York, Dec 2010.
- Slides de Kurose/Ross Computer Networking 6ta edición.
- Internet ...

Protocolo HTTP(hasta 1.1) y WEB-Cache

Redes y Comunicaciones

Historia

- WWW (red de sistemas de hipertexto inter-linkeados accesibles vía Internet).
- Desarrollada en 1990 por Tim Berners-Lee en el CERN.
 - Desarrolla el protocolo HTTP y el lenguaje HTML.
- 1993: el primer cliente/browser GUI: Mosaic.
- 1994: Netscape Navigator 1.0.
- Desarrollo de servidores, por ejemplo 1995: Apache Server.
- Buscadores, Indexadores son parte importante del servicio.
- Modelo anterior de interacción: servidor produce, cliente consume.
- Nuevos Modelos: Web 2.0, Tim O'Reilly en 2004, modelo de interacción entre usuarios, consumen y producen mediante servicios especiales: blogs, redes sociales, wikis, multimedia, etc.

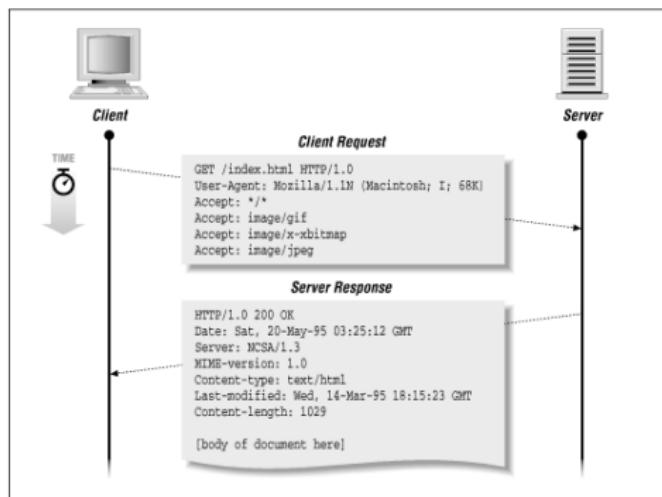
Elementos WEB

- Recurso u Objeto HTTP: 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 impl. ej.
`urn:isbn:0132856204, urn:ietf:rfc:2616`.
- Objetos pueden ser página web (web page), imágenes JPEG, PNG, GIF, Java Applet, archivos de multimedia: MP3, AVI, etc.
- Páginas web: archivo HTML que incluye vínculos o directamente otros objetos.

Funcionamiento de HTTP

- Modelo cliente/servidor, Request/Response (sin estados - 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.
- Clientes (llamados browsers o navegadores): Firefox, IE, Opera, Safari, Chrome.
- Servidores: Apache Server, MS IIS, NGINX , Google GWS, Tomcat.

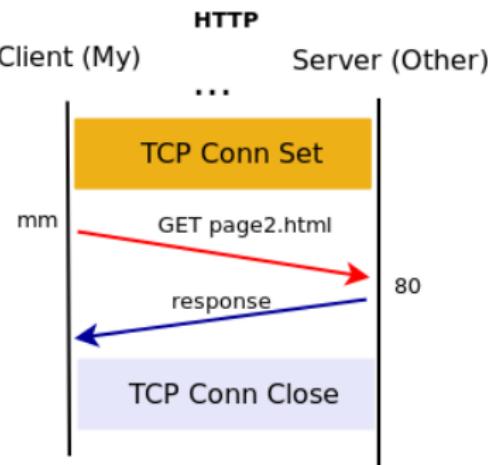
Esquema de HTTP



Versión HTTP 0.9

- Primera versión de HTTP fue la 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.

Diagrama de Pasos HTTP



Versión HTTP 0.9 (cont'd)

- 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>

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.
 - Define códigos de respuesta.
 - 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.

Request HTTP 1.0

<Method> <URI> <Version>

[<Headers Opcionales>]

<Blank>

[<Entity Body Opcional>]

<Blank>

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

DELETE, LINK, UNLINK

Response HTTP 1.0

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

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

Métodos HTTP/1.0

GET: obtener el documento requerido. Puede enviar información, pero no demasiada. Es enviada en la URL. Formato ?var1=val1&var2=val2.... Limitación de tamaño de URL por parte de las implementaciones. NO espera recibir datos en body.

HEAD: idéntico a GET, pero sólo requiere la meta información del documento, por ejemplo, su tamaño. Usado por clientes con caché.

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.

Métodos HTTP/1.0 (Cont'd)

PUT: usado para reemplazar un documento en el servidor. En general, deshabilitada. Utilizado, por ejemplo, por protocolos 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.

GET/Response HTTP 1.0

```
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
```

GET/Response HTTP 1.0 (Cont'd)

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

HEAD/Response HTTP 1.0

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

HTTP/1.1 200 OK
Date: Mon, 21 Apr 2008 00:40:25 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

POST/Response HTTP 1.0

```
POST /index.html HTTP/1.0
User-Agent: telnet/andres (GNU/Linux)
Host: estehost.com
Accept: */
Content-Type: text/plain
Content-Length: 10
```

1234567890

```
HTTP/1.1 200 OK
Date: Mon, 21 Apr 2008 00:37:22 GMT
Server: Apache/2.2.4 (Ubuntu)
Last-Modified: Mon, 21 Apr 2008 00:18:14 GMT
ETag: "a3b36-1f-91d5d80"
Accept-Ranges: bytes
```

POST/Response HTTP 1.0 (Cont'd)

Content-Length: 31

Connection: close

Content-Type: text/plain

<HTML>

<H1> HOLA </H1>

...

</HTML>

GET HTTP/1.0 con Host Virtuales

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

```
? ./mozilla http://www.uno.test
```

```
? ./mozilla http://www.dos.test
```

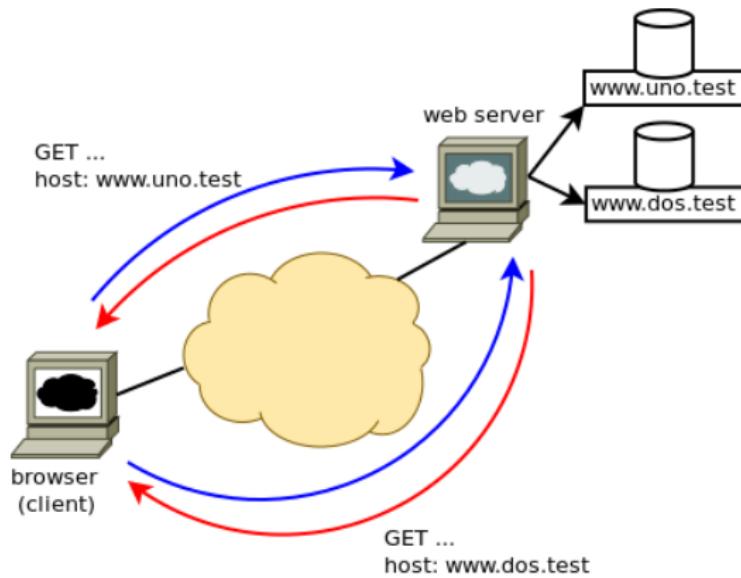
```
? host www.uno.test
```

```
www.uno.test has address 192.168.0.2
```

```
? host www.dos.test
```

```
www.dos.test has address 192.168.0.2
```

Virtual Hosts



Autenticación HTTP

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

Autenticación HTTP (Cont'd)

Authorization Required

This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.

Apache/2.2.9 (Debian) DAV/2 SVN/1.5.1 PHP/5.2.6-1+lenny3 with Suhosin-Patch mod_ssl/2.2.9 OpenSSL/0.9.8g Server at

Port 443



HTTP/1.0 con conexiones persistentes

- En HTTP/1.0 no se contemplaron las conexiones persistentes por default.
- A partir de HTTP/1.1 [RFC-2068], si.
- En HTTP/1.0 se pueden solicitar de forma explícita.

GET /index.html HTTP/1.0

Connection: Keep-Alive

User-Agent: telnet/andres (GNU/Linux)

Host: estehost.com

Accept: */*

HTTP/1.1 200 OK

Date: Tue, 22 Apr 2008 01:31:56 GMT

Server: Apache/2.2.4 (Ubuntu)

Last-Modified: Mon, 21 Apr 2008 00:18:14 GMT

HTTP/1.0 con conexiones persistentes (Cont'd)

```
ETag: "a3b36-1f-91d5d80"
Accept-Ranges: bytes
Content-Length: 31
Keep-Alive: timeout=15, max=10
Connection: Keep-Alive
Content-Type: text/html
```

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

```
GET /index.html HTTP/1.0
Connection: Keep-Alive
```

...

Connection closed by foreign host.

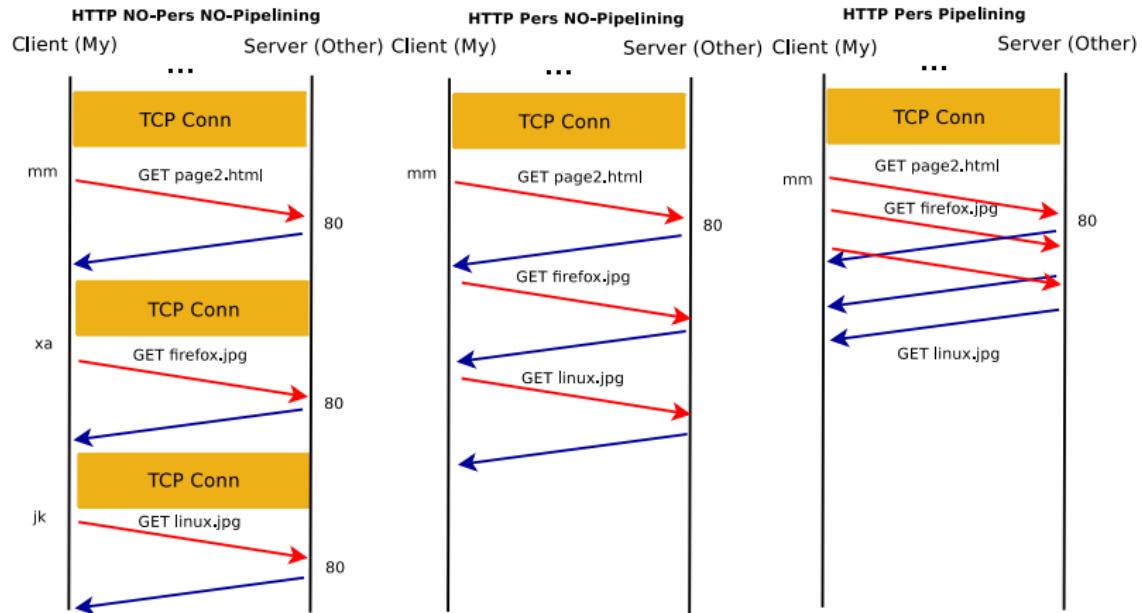
Versión HTTP 1.1

- 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.
- Pipelining, mejora tiempo de respuestas.

Pipelining HTTP 1.1

- No necesita esperar la respuesta para pedir otro objeto HTTP.
- Solo se utiliza con conexiones persistentes.
- Mejora los tiempos de respuestas.
- Sobre la misma conexión se debe mantener el orden de los objetos que se devuelven.
- Se pueden utilizar varios threads para cada conexión.
 - Sin pipelining: $1RTT + FT$ por cada objeto, n objetos $nRTT + nFT$.
 - Con pipelining óptimo: n objetos: $1RTT + nFT$.

Pipelining HTTP 1.1 (Cont'd)



TRACE y CONNECT HTTP 1.1

- TRACE utilizada para debugging.
- El servidor debe copiar el requerimiento tal cual.
- El cliente puede comparar lo que envía con lo que recibe.
- CONNECT utilizada para generar conexiones a otros servicios montadas sobre HTTP.
- Proxy-Agent genérico.

Ejemplo de CONNECT

CONNECT 127.0.0.1:25

HTTP/1.0 200 Connection Established

Proxy-agent: Apache/2.2.4 (Ubuntu)

220 khartum ESMTP Postfix (Ubuntu)

HELO otrohost.com

250 khartum

QUIT

221 2.0.0 Bye

...

Redirects HTTP

- Redirect temporal 302, indicando la nueva URL/URI.
- El user-agent no debería re-direccionarlo 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 Scripts y JavaScript

- Necesidad de dinamismo para generar aplicaciones.
- 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 o módulos propios del web-server.

CGIs Scripts y JavaScript (Cont'd)

- 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 interfaz de programación, API fácil de utilizar.

CGIs Scripts y JavaScript (Cont'd)

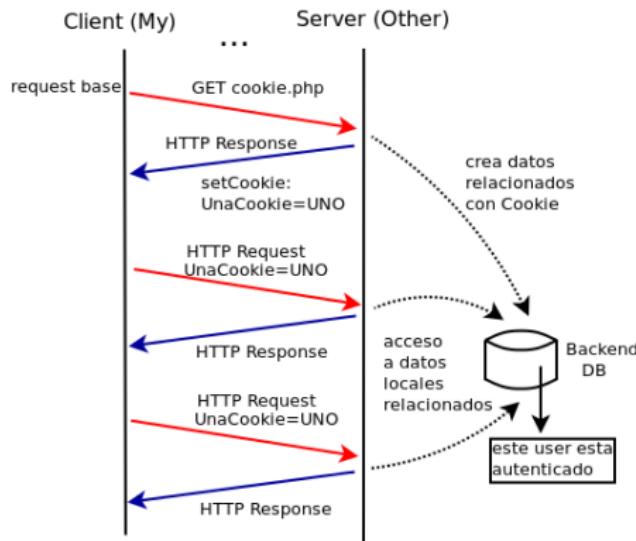
- Server-Side to Server-Side Script:

- Permiten 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

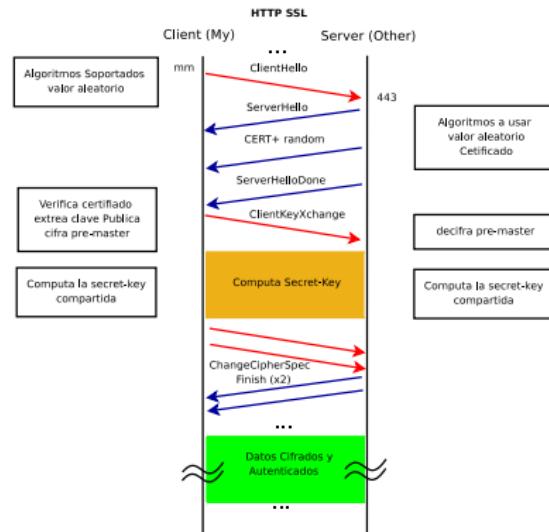
- Mecanismo que permite a las aplicaciones web del servidor “manejar estados”.
- El cliente hace un request.
- El servidor retorna un recurso (un objeto HTTP, como una página HTML) indicando al cliente que almacene determinados valores por un tiempo.
- La Cookie es introducida al cliente mediante el mensaje en el header Set-Cookie: mensaje que indica un par (nombre,valor).
- El cliente en cada requerimiento luego de haber almacenado la Cookie se la enviará al servidor con el header Cookie:.
- El servidor puede utilizarlo o no.
- El servidor puede borrarlo.
- Esta información puede ser usada por client-side scripts.

Ejemplo de Cookie



HTTPS (HTTP sobre TLS/SSL)

- Utiliza el port 443 por default.
- Etapa de negociación previa.
- Luego se cifra y autentica todo el mensaje HTTP (incluso el header).



WEB-Cache

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

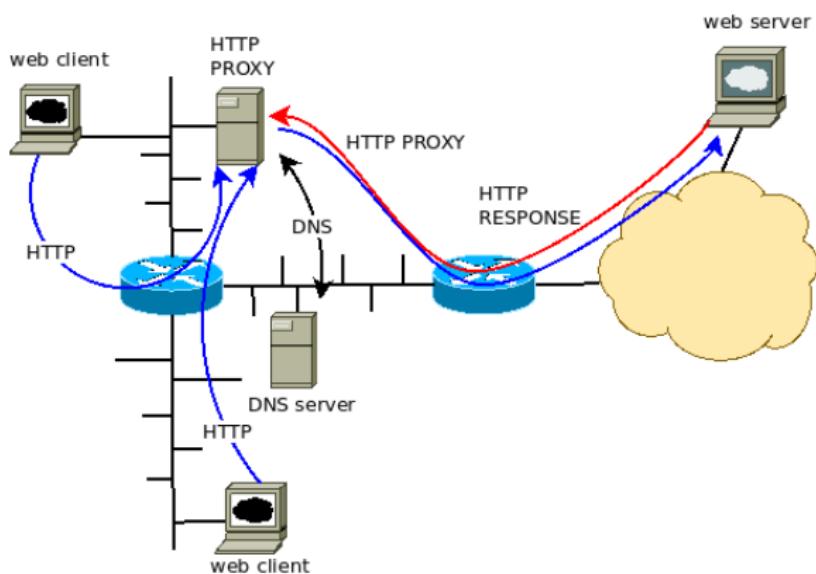
WEB-Cache (Cont'd)

- Cache del lado del cliente (privada).
- Los web browser tienen sus propias cache locales.
- Los servidores agregan headers:
 - Last-Modified: date
 - ETag: (entity tag) hash
- Requerimientos condicionales desde los clientes:
 - If-Modified-Since: date
 - If-None-Match: hash
- Respuestas de los servidores:
 - 304 Not Modified.
 - 200 OK.

WEB-Cache (Cont'd)

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

WEB-Cache (Cont'd)



- [Stevl2] TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, (2nd. Ed). 2011. Kevin R. Fall, W. Richard Stevens.
- [KR] Computer Networking: A Top-Down Approach, Addison-Wesley, (6th Edition). 2012. Kurose/Ross.
- [LX] The Linux Home Page: <http://www.linux.org/>.
- [Siever] Linux in a Nutshell, Fourth Edition June, 2003. O'Reilly. Ellen Siever, Stephen Figgins, Aaron Weber.
- [RFC-793] <http://www.rfc-editor.org/rfc/rfc793.txt>. TCP Transmission Control Protocol (Jon Postel 1981 USC-ISI IANA).
- [HTTP0.9] The Original HTTP as defined in 1991.
<http://www.w3.org/Protocols/HTTP/AsImplemented.html>.
- [RFC-1738] <http://www.faqs.org/rfcs/rfc1738.html>. Uniform Resource Locators (URL). (Berners-Lee, T., Masinter, L., and M. McCahill, CERN, Xerox PARC, University of Minnesota, 1994).
- [RFC-1866] <http://www.faqs.org/rfcs/rfc1866.html>. Hypertext Markup Language - 2.0. (Berners-Lee (MIT/W3C) , D. Connolly, 1995).
- [HTML30] <http://www.w3.org/MarkUp/html3/CoverPage>. Raggett, D., "HyperText Markup Language Specification Version 3.0", September 1995. (Available at
- [HTML401] <http://www.w3.org/TR/html401> Raggett, D., et al., "HTML 4.01 Specification", W3C Recommendation, December 1999.
- [XHTML1] <http://www.w3.org/TR/xhtml1>. "XHTML 1.0: The Extensible HyperText Markup Language: A Reformulation of HTML 4 in XML 1.0", W3C Recommendation, January 2000.
- [RFC-1945] <http://www.faqs.org/rfcs/rfc1945.html>. Hypertext Transfer Protocol – HTTP/1.0. (T. Berners-Lee (MIT/LCS) , R. Fielding (UC Irvine) , H. Frystyk (MIT/LCS) , 1996).
- [RFC-2068] HTTP/1.1. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., and T. Berners-Lee, "Hypertext Transfer Protocol HTTP/1.1", RFC 2068, 1997.
- [RFC-2616] <http://www.faqs.org/rfcs/rfc2616.html>. HTTP/1.1. (R. Fielding (UC Irvine) , J. Gettys (Compaq/W3C) , J. Mogul (Compaq) , H. Frystyk (W3C/MIT) , L. Masinter (Xerox) , P. Leach (Microsoft) , T. Berners-Lee (W3C/MIT), 1999)
- [RFC-2246] <http://www.ietf.org/rfc/rfc2246.txt>. The TLS Protocol Version 1.0. (Dierks, C. Allen (Certicom), 1999).

- [CGI1.1] <http://www.w3.org/CGI/>. The WWW Common Gateway Interface Version 1.1.
- [PUTs] Referencias al método HTTP/1.0 PUT. <http://www.apacheweek.com/features/put.html>. <http://www.w3.org/Amaya/User/Put.html>. <http://www.w3.org/Library/Examples/>.
- [APACHE] <http://httpd.apache.org/docs/2.0/server-wide.html>
- [MOZ] Mozilla Firefox. <http://www.mozilla.com>.
- [WCPP] <http://www.oreilly.com/openbook/webclient/> Web Client Programming with Perl. Clinton Wong, 1997.
- [OpenSSL] OpenSSL project: <http://www.openssl.org/>.
- [WDV] WebDAV: <http://www.webdav.org/>. RFC-4918.
- [BR1] <http://marketshare.hitslink.com>.
<http://www.nationmaster.com>.
<http://www.w3counter.com/globalstats.php>.
http://en.wikipedia.org/wiki/Usage_share_of_web_browsers.
- [SR1] <http://news.netcraft.com>.
- [CAIDAtnotb98] The nature of the beast: recent traffic measurements from an Internet backbone. K Claffy, caida, kc@caida.org. Greg Miller and Kevin Thompson, MCI/vBNS, gmiller,kthomp@mci.net. 1998.
- [YT] <http://www.youtube.com/>.
- [COM05] Ethereal, Wireshark. Autor original Gerald Combs, 2005.
<http://www.ethereal.com/>.
<http://www.wireshark.org/>.
- [SOAP] W3C Recommendation (27 April 2007). SOAP Version 1.2 Part 0: Primer (Second Edition). W3C.
<http://www.w3.org/TR/2007/REC-soap12-part0-20070427/>.
- [COOKIE] Cookies Spec. http://curl.haxx.se/rfc/cookie_spec.html.

HTTP/2 y HTTP/3

Redes y Comunicaciones (2024)

Qué es 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:
 - RFC7540: Hypertext Transfer Protocol version 2.
 - RFC7541: HPACK - Header Compression for HTTP/2.

Problemas con HTTP/1.0, HTTP/1.1

- Un request por conexión, por vez, muy lento.
- Alternativas (evitar HOL):
 - Conexiones persistentes y pipelining.
 - Generar conexiones paralelas.
- Problemas:
 - Pipelining requiere que los responses sean enviado en el orden solicitado, HOL posible.
 - POST no siempre pueden ser enviados en pipelining.
 - Demasiadas conexiones genera problemas, control de congestión, mal uso de la red.
 - Muchos requests, muchos datos duplicados (headers).

Diferencias principales con HTTP/1.1

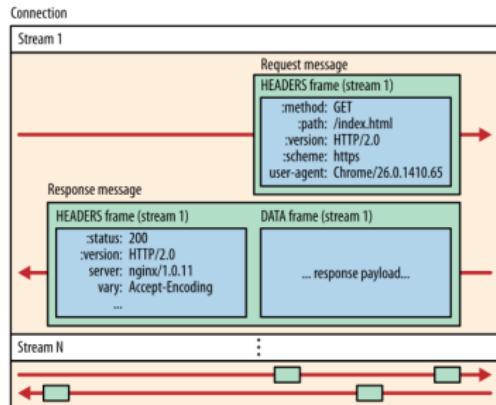
- Protocolo binario en lugar de textual(ASCII), binary framing: (más eficiente).
- Multiplexa varios request en una petición en lugar de ser una secuencia ordenada y bloqueante.
- Utilizar una conexión para pedir/traer datos en paralelos, agrega: datos fuera de orden, priorización, flow control por frame.
- Usa compresión de encabezado.
- Permite a los servidores “pushear” datos a los clientes.
- La mayoría de las implementaciones requieren TLS/SSL, no el estándar.

HTTP/2 mux stream, framing

- Puede generar una o más conexiones TCP. Trata de aprovechar las que tiene establecidas.
- Un stream es como una sub-conexión (una “conexión” http2 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 http2”).
- Los streams transportan mensajes.
- Los mensajes http2 (Request, Response) se envían usando un stream.
- Los mensajes http2 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 msj.

HTTP/2 mux stream, framing (cont.)

- 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.

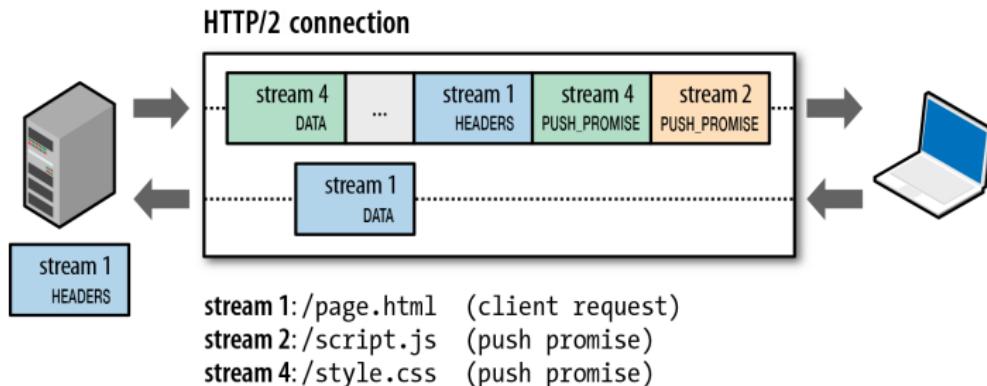


fuente: <https://web.dev/performance-http2/#streams,-messages,-and-frames>

HTTP/2 mux stream, framing (cont.)

The screenshot shows an active session in Wireshark titled "http2_native.pcapng". The packet list pane displays 21 packets, with 8 displayed. The details pane shows the transmitted data for each stream, including SETTINGS, WINDOW_UPDATE, and HEADERS frames. Stream 1 (HTTP/2) contains 152 SETTINGS[0], 100 SETTINGS[0], 76 SETTINGS[0], 369 HEADERS[1]: 200 OK, DATA[1] (text/plain), 75 SETTINGS[0], 93 HEADERS[3]: GET /humans.txt, and 124 HEADERS[2]: 404 Not Found, DATA[2] (text/plain). Stream 2 (HTTP/1.1) contains 100 HEADERS[1]: 200 OK, DATA[1] (text/html), 100 HEADERS[2]: 200 OK, DATA[2] (text/html), and 100 HEADERS[3]: 200 OK, DATA[3] (text/html).

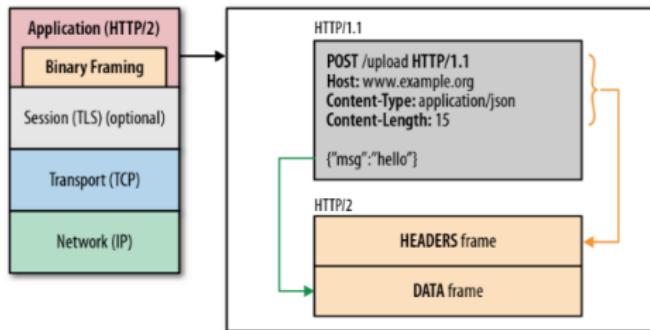
HTTP/2 mux stream, framing (cont.)



fuente: <https://docs.google.com/presentation/d/1r7QXGYOLCh4fcUq0jDdDwKJWNqWK1o4xMtYpKZCJYjM/present?slide=id.p19>

HTTP/2 mux stream, framing (Cont.)

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



fuente: <https://docs.google.com/presentation/d/1r7QXGYOLCh4fcUq0jDdDwKJWNqWK1o4xMtYpKZCJYjM/present?slide=id.p19>

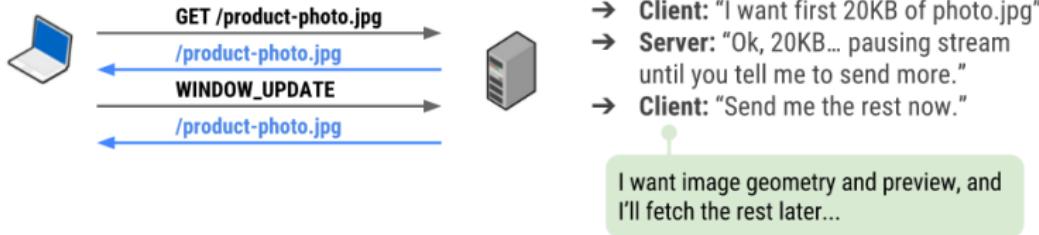
HTTP/2 HEADERS

- 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 con http2:
 - :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.



fuente: <https://docs.google.com/presentation/d/1r7QXGYOLCh4fcUq0jDdDwKJWNqWK1o4xMtYpKZCJYjM/present?slide=id.p19>

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

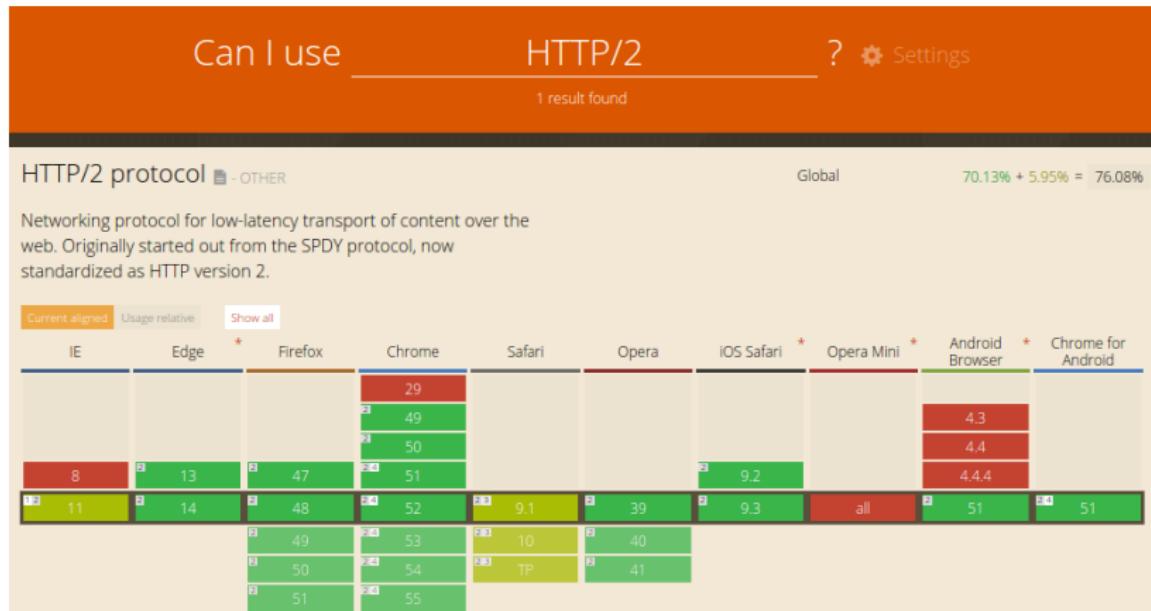


fuente: <https://docs.google.com/presentation/d/1r7QXGYOLCh4fcUq0jDdDwKJWNqWK1o4xMtYpKZCJYjM/present?slide=id.p19>

Compresión y Soporte

- 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.

Soporte en clientes para 2016



fuente: <http://caniuse.com/#search=HTTP%2F2>

Otras Características

- HTTP/1.1, posibilidad de hacer un upgrade durante la conexión:
Upgrade Header. Connection: Upgrade, HTTP2-Settings
Upgrade: h2c|h2.
- http2: 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, ...
- Posibilidad de negociar protocolo alternativo:
Alterantive Service: alt-svc.

Application-Layer Protocol Neg.

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Length	Info
4	0.000114	127.0.0.1	127.0.0.1	56	[TCP Window Update] 8443->61946 [ACK] Seq=1
5	0.000285	127.0.0.1	127.0.0.1	461	Client Hello
6	0.000328	127.0.0.1	127.0.0.1	56	8443->61946 [ACK] Seq=1 Ack=406 Win=146576
7	0.001662	127.0.0.1	127.0.0.1	1122	Server Hello, Certificate, Server Hello Done

▼ Extension: Application Layer Protocol Negotiation
 Type: Application Layer Protocol Negotiation (0x0010)
 Length: 52
 ALPN Extension Length: 50

▼ ALPN Protocol
 ALPN string length: 17
 ALPN Next Protocol: HTTP-draft-04/2.0
 ALPN string length: 8
 ALPN Next Protocol: spdy/4a2
 ALPN string length: 8
 ALPN Next Protocol: spdy/3.1
 ALPN string length: 6
 ALPN Next Protocol: spdy/3
 ALPN string length: 6
 ALPN Next Protocol: spdy/2

▼ Extension: status_request
 Type: status_request (0x0005)
 Length: 5

File: "/home/andres/docs/mvdoc...." Packets: 202 - Disp... Profile: Default

Debugging (2016)

Google - Google Chrome

Google https://www.google.com.ar/#gfe_rd=cr&ei=6XPEV9hsCcOgxg55...

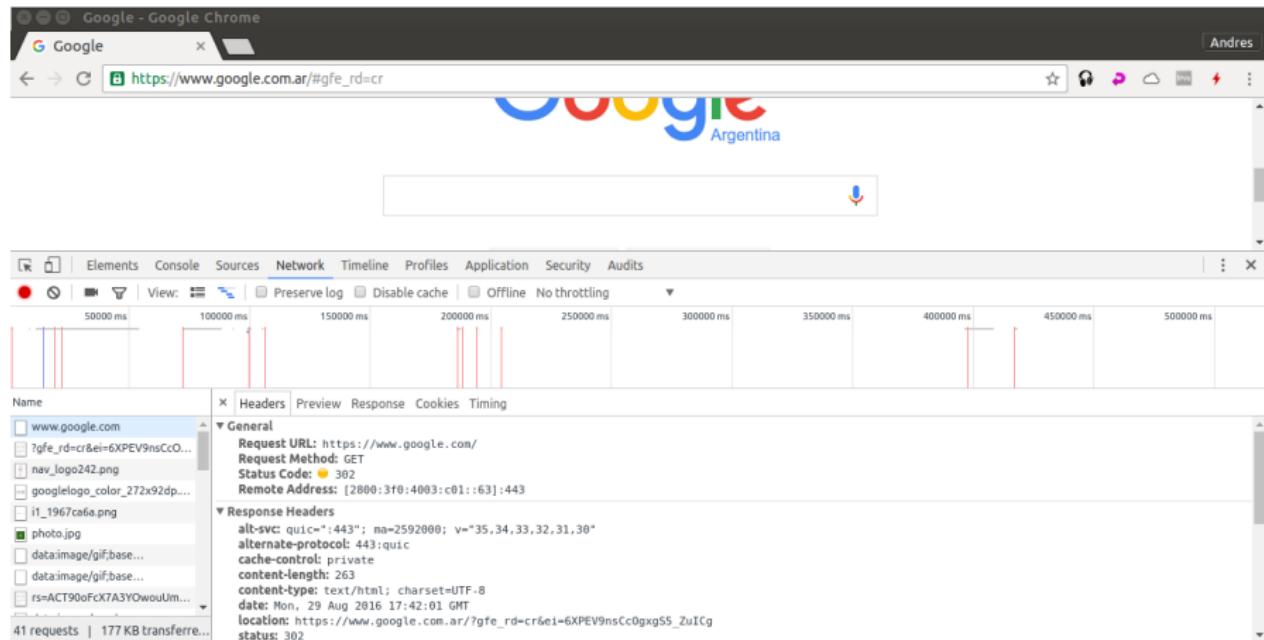
Andres

Timeline (ms): 10000 ms, 20000 ms, 30000 ms, 40000 ms, 50000 ms, 60000 ms, 70000 ms, 80000 ms, 90000 ms, 100000 ms, 110000 ms

Name	Method	Status	Protocol	Type	Initiator	Size	Time	Timeline – Start Time	4.00 s	6.00 s
www.google.com	GET	302	h2	text/html	Other	399 B	33 ms			
gfe_rd=cr&ei=6XPEV9hsCcOgxg55...	GET	200	quic/1+spdy/3	document	https://www.google...	62.7 KB	335 ms			
nav_logo242.png	GET	200	quic/1+spdy/3	png	gfe_rd=cr&ei=6X...	(from cac...)	2 ms			
googlelogo_color_272x92dp.png	GET	200	quic/1+spdy/3	png	gfe_rd=cr&ei=6X...	(from cac...)	3 ms			
i1_1967ca6a.png	GET	200	quic/1+spdy/3	png	gfe_rd=cr&ei=6X...	(from cac...)	5 ms			
photo.jpg	GET	200	quic/1+spdy/3	png	gfe_rd=cr&ei=6X...	(from cac...)	4 ms			
data:image/gif;base...	GET	200	data	gif	gfe_rd=cr&ei=6X...	(from cac...)	0 ms			
data:image/gif;base...	GET	200	data	gif	gfe_rd=cr&ei=6X...	(from cac...)	0 ms			

25 requests | 64.3 KB transferred | Finish: 7.34 s | DOMContentLoaded: 536 ms | Load: 535 ms

Debugging (Cont.)



Debugging (Cont.)

chrome://net-internals/#http

Capturing halted

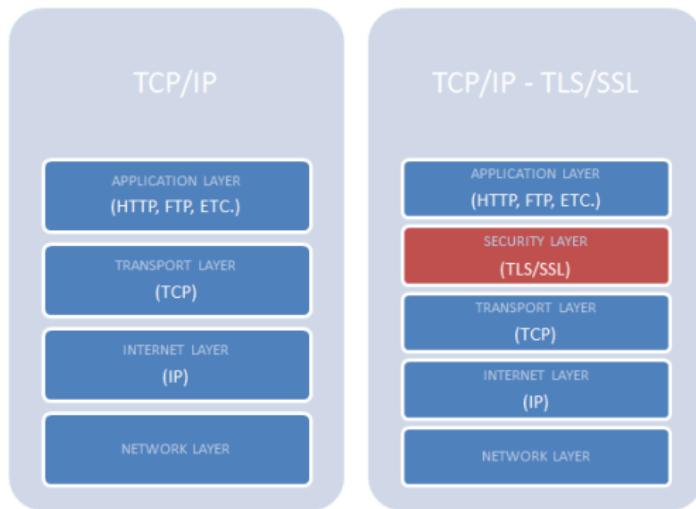
- HTTP/2 Enabled: true
- SPDY/3.1 Enabled: false
- Use Alternative Service: true
- ALPN Protocols: h2,http/1.1
- NPN Protocols: undefined

HTTP/2 sessions

[View live HTTP/2 sessions](#)

Host	Proxy	ID	Protocol Negotiated	Active streams	Unclaimed pushed	Max	Initiated	Pushed	Pushed and claimed	Abandoned	Received frames	Secure	Sent settings	Received settings
accounts.google.com:443	direct://	231	h2	0	0	100	0	0	0	0	0	true	true	true
s2.googleusercontent.com:443	direct://	370	h2	0	0	100	0	0	0	0	0	true	true	true
ssl.google-analytics.com:443	direct://	265	h2	0	0	100	0	0	0	0	0	true	true	true
ssl.gstatic.com:443	direct://	450	h2	0	0	100	0	0	0	0	0	true	true	true
www.google-analytics.com:443	direct://	339	h2	0	0	100	0	0	0	0	0	true	true	true
www.google.com:443	direct://	516	h2	0	0	100	0	0	0	0	0	true	true	true
www.google.com.ar:443	direct://	335	h2	0	0	100	0	0	0	0	0	true	true	true
accounts.google.com:443	direct://	328	h2	0	0	100	0	0	0	0	0	true	true	true
clients2.google.com:443	direct://	654	h2	0	0	100	0	0	0	0	0	true	true	true
www.googleapis.com:443	direct://	120	h2	0	0	100	0	0	0	0	0	true	true	true

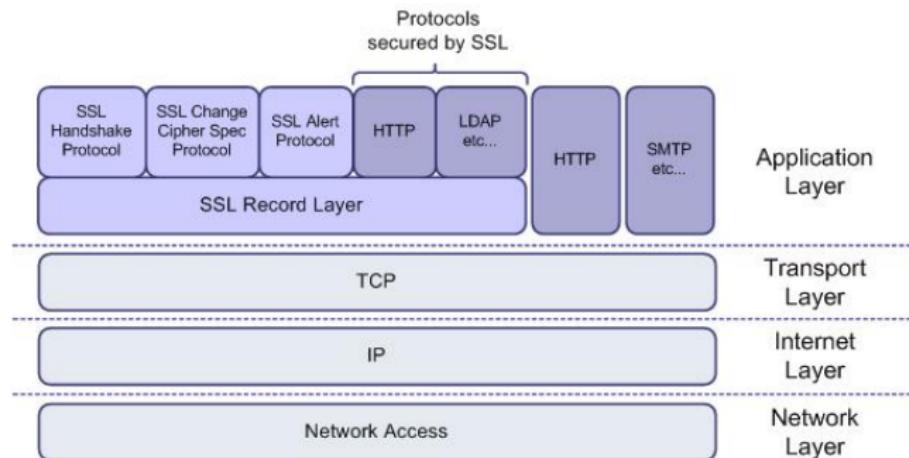
SSL/TLS



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

fuente: <https://www.simple-talk.com/dotnet/net-framework/tlssl-and-net-framework-4-0/>

SSL/TLS (Cont.)

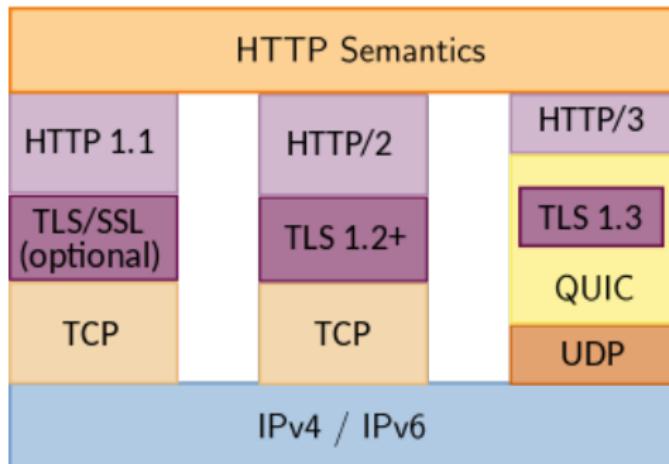


fuente: http://nicolascormier.com/documentation/bin/apache/apache2_with_ssl_tls/part1.htm

Qué es 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:
 - RFC9000: QUIC: A UDP-Based Multiplexed and Secure Transport.
 - RFC9114: HTTP/3.

HTTP/3



fuente: <https://en.wikipedia.org/wiki/HTTP/3>

Diferencias principales de HTTP/3-QUIC

- Reducción de latencia en conexiones (2 msj. 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 genere nuevos números de secuencia y re-cifra las retransmisiones, permitiendo mejor detección de pérdidas y medición de RTT.
- QUIC Paquetes cifrados de forma individual, no por conexión/bytestream.
- QUIC facilita movilidad, tiene ID de conexión independiente de IPs y ports

Diferencias principales de HTTP/3-QUIC (Cont.)

- 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.

HTTP/3 captura

*enp0s31f6

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

quic

No.	Time	Source	Destination	Protocol
21	9.202350845	[REDACTED]	2800:3f0:4002:809::2003	QUIC
22	9.202542084	[REDACTED]	2800:3f0:4002:809::2003	QUIC

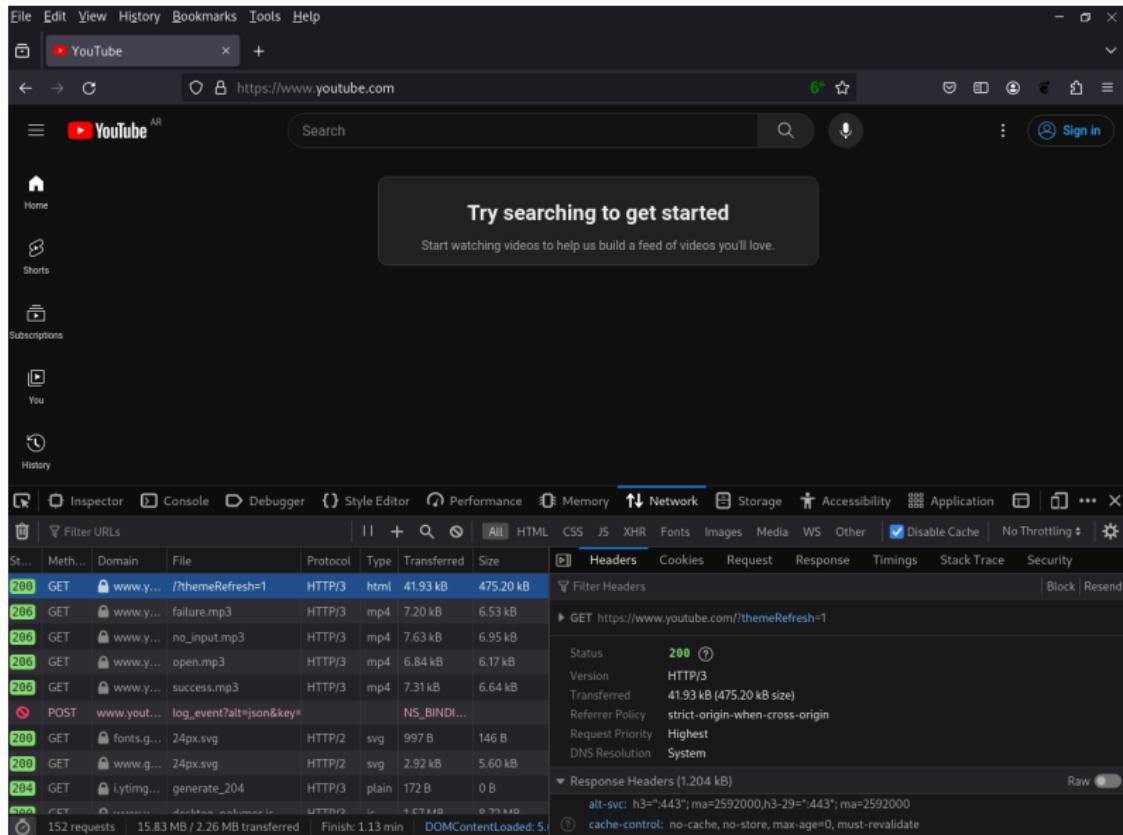
```

> Internet Protocol Version 6, Src: 2800:340:... Dst: 2800:3f0:4002:809::2003
> User Datagram Protocol, Src Port: 56546, Dst Port: 443
> QUIC IETF
  > QUIC Connection information
    [Packet Length: 1230]
    1... .... = Header Form: Long Header (1)
    .1.. .... = Fixed Bit: True
    ..00 .... = Packet Type: Initial (0)
    .... 00.. = Reserved: 0
    .... ..00 = Packet Number Length: 1 bytes (0)
    Version: 1 (0x00000001)
    Destination Connection ID Length: 8
    Destination Connection ID: 5b8c72711860d7e2
    Source Connection ID Length: 0
    Token Length: 70
    Token: 00a724a2bb3734560b9583c50a85edabdd33ef2a8f75a68ae4ec82ff9041010b80dbdf3...
    Length: 1141
    Packet Number: 1
    Payload: d7b50a500c2fdf8454cd5916cc5100508a1778eae56de71f0604326db821547415afac4d...
  > PING
    Frame Type: PING (0x0000000000000001)
  > PING
  > PING
  > PADDING Length: 18
  > CRYPTO
    Frame Type: CRYPTO (0x0000000000000006)
    Offset: 312
    Length: 746
    Crypto Data
    > TLSv1.3 Record Layer: Handshake Protocol: Encrypted Handshake Message
  > PING
  > CRYPTO
  Destination Connection ID Length (quic.dcll), 1 byte

```

Packets: 2015 · Displayed: 670 (33.3%) · Profile: Classic

HTTP/3 Debugging (2024)



The screenshot shows a browser window with the YouTube homepage loaded at <https://www.youtube.com>. The browser interface includes a menu bar (File, Edit, View, History, Bookmarks, Tools, Help), a toolbar with icons for back, forward, search, and refresh, and a status bar at the bottom.

The main content area displays the YouTube homepage with a search bar and a large button saying "Try searching to get started". On the left, there's a sidebar with links for Home, Shorts, Subscriptions, You, and History.

At the bottom, the developer tools Network tab is open, showing a list of network requests. The table has columns for St..., Method, Domain, File, Protocol, Type, Transferred, Size, Headers, Cookies, Request, Response, Timings, Stack Trace, and Security.

St...	Method	Domain	File	Protocol	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	Stack Trace	Security
200	GET	www.y...	?themeRefresh=1	HTTP/3	html	41.93 kB	475.20 kB							
200	GET	www.y...	failure.mp3	HTTP/3	mp4	7.20 kB	6.53 kB							
200	GET	www.y...	no_input.mp3	HTTP/3	mp4	7.63 kB	6.95 kB							
200	GET	www.y...	open.mp3	HTTP/3	mp4	6.84 kB	6.17 kB							
200	GET	www.y...	success.mp3	HTTP/3	mp4	7.31 kB	6.64 kB							
200	POST	www.yout...	log_event?alt=json&key=		NS_BINDI...									
200	GET	fonts.g...	24px.svg	HTTP/2	svg	997 B	146 B							
200	GET	www.g...	24px.svg	HTTP/2	svg	2.92 kB	5.60 kB							
204	GET	Lyting...	generate_204	HTTP/3	plain	172 B	0 B							
200	GET	www.yout...	getAutoplayVideoList	HTTP/3	js	4.87 kB	0.77 kB							

The bottom status bar indicates 152 requests, 15.83 MB / 2.26 MB transferred, and a total time of Finish: 1:13 min. The DOMContentLoaded value is 5. The developer tools also show expanded sections for Response Headers and Cache Control headers.

[HTTP/2] [https://http2.github.io/.](https://http2.github.io/)

[Ilya Grigorik] HTTP/2 is here, let's optimze!

[https://docs.google.com/presentation/d/1r7QXGYOLCh4fcUq0jDdDwKJWNqWK1o4xMtYpKZCJYjM/present?slide=id.p19.](https://docs.google.com/presentation/d/1r7QXGYOLCh4fcUq0jDdDwKJWNqWK1o4xMtYpKZCJYjM/present?slide=id.p19)

[HTTP/2-dev] <https://web.dev/performance-http2/#streams,-messages,-and-frames>

[HTTP/2-undertow] <https://undertow.io/blog/2015/04/27/An-in-overview-of-HTTP2.html>

[HTTP/3-cloudflare] <https://www.cloudflare.com/learning/performance/what-is-http3/>

[HTTP/3-wikipedia] <https://en.wikipedia.org/wiki/HTTP/3>

[QUIC-wikipedia] <https://en.wikipedia.org/wiki/QUIC>

[QUIC-google] <https://peering.google.com/#/learn-more/quic>

¿Qué pasa con la seguridad cuando se transmite información?

Hay algunas preguntas que podemos hacernos:

- ¿Estamos seguros que quien nos envía un mensaje es quien dice ser?
- Si alguien logra interceptar una comunicación ¿puede ver el contenido del mensaje?
- Si alguien accede al contenido del mensaje ¿puede modificarlo sin que el que lo reciba se dé cuenta?

¿Qué pasa con la seguridad cuando se transmite información?

Hay distintos mecanismos para proteger la información y garantizar:

- Autenticidad
- Confidencialidad
- Integridad

Depende de lo que queramos garantizar qué mecanismos vamos a usar!

Un ejemplo de intercambio de Información Seguro

Pensemos en un servicio de mensajería de paquetes que usa un protocolo para proteger la comunicación.

El protocolo sería:

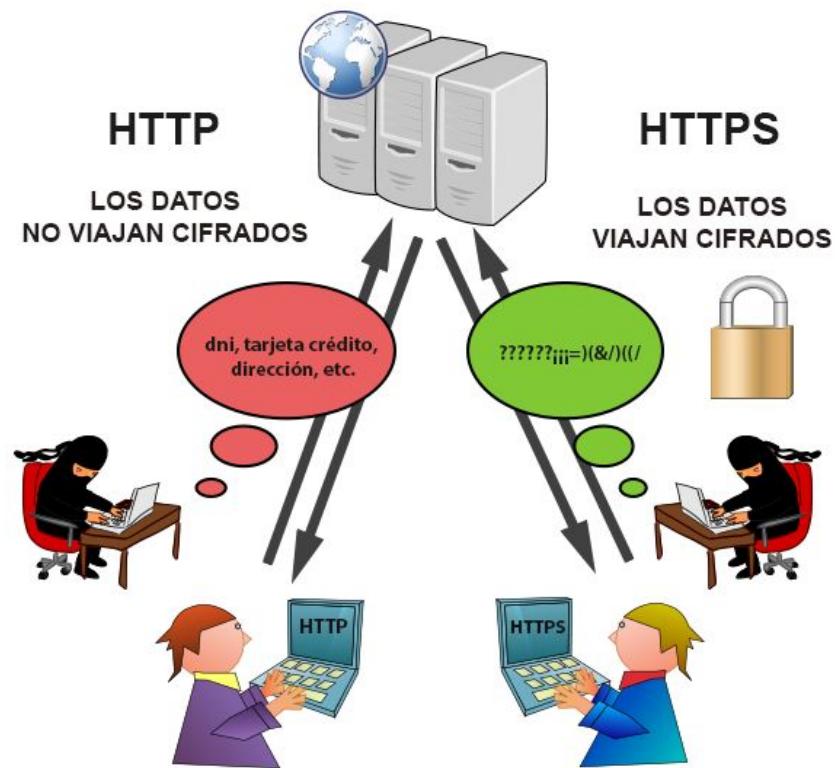
- Se usa una caja con candado para guardar los mensajes a enviar. Existe una única llave.
- Inicialmente el receptor debe hacer llegar al emisor el candado que éste debe utilizar.
- Para transmitir un mensaje secreto, el emisor guarda el mensaje en la caja y la cierra con el candado que el receptor le dio.
- El receptor recibe la caja, y luego de abrir el candado con la llave que **solo él posee** puede acceder al mensaje.



Navegando en sitios seguros

Cuando navegamos en Internet, los protocolos de comunicación más usados son HTTP y HTTPS.

Estos protocolos son los que utilizan los navegadores para conectarse a páginas web.



Navegando en sitios seguros - HTTPS

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:

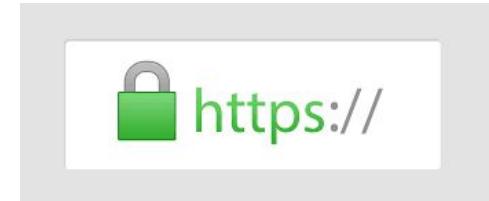
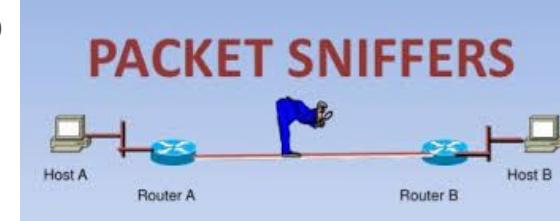
- La información viaja de manera cifrada de extremo a extremo.
- El cliente puede verificar la autenticidad del sitio visitado.
- Opcionalmente el servidor puede requerir autenticación del cliente.

Navegando en sitios seguros - HTTPS

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
Un sniffer (aplicación que permite analizar el tráfico de red con el propósito de espionar) no podrá entender los mensajes intercambiados entre el cliente y el servidor
- El cliente puede verificar la autenticidad del sitio visitado
El navegador tiene la posibilidad de utilizar la criptografía asimétrica para verificar la autenticidad del sitio visitado



Cómo se distinguen los navegadores HTTP y HTTPS

Comparación en cómo se muestra un sitio HTTP y uno HTTPS válido

A screenshot of a web browser window titled "HTTP vs HTTPS — Test this site". The address bar shows "HTTP vs HTTPS — Test this site" and "https://httpvshttps.com". The content area displays the text "HTTP vs HTTPS Test" and "Encrypted Websites Protect Our Privacy and are Significantly Faster". Below this, it says "Compare load times of the unsecure HTTP and encrypted HTTPS versions of this page. Each test loads 360 unique, non-cached images (0.62 MB total). For fastest results, run each test 2-3 times in a private/incognito browsing session." At the bottom, there is a row of green checkmarks. To the right of the content area, there is a legend with "HTTP" and "HTTPS" and a performance summary: "1.557 s" and "88% faster than HTTP".

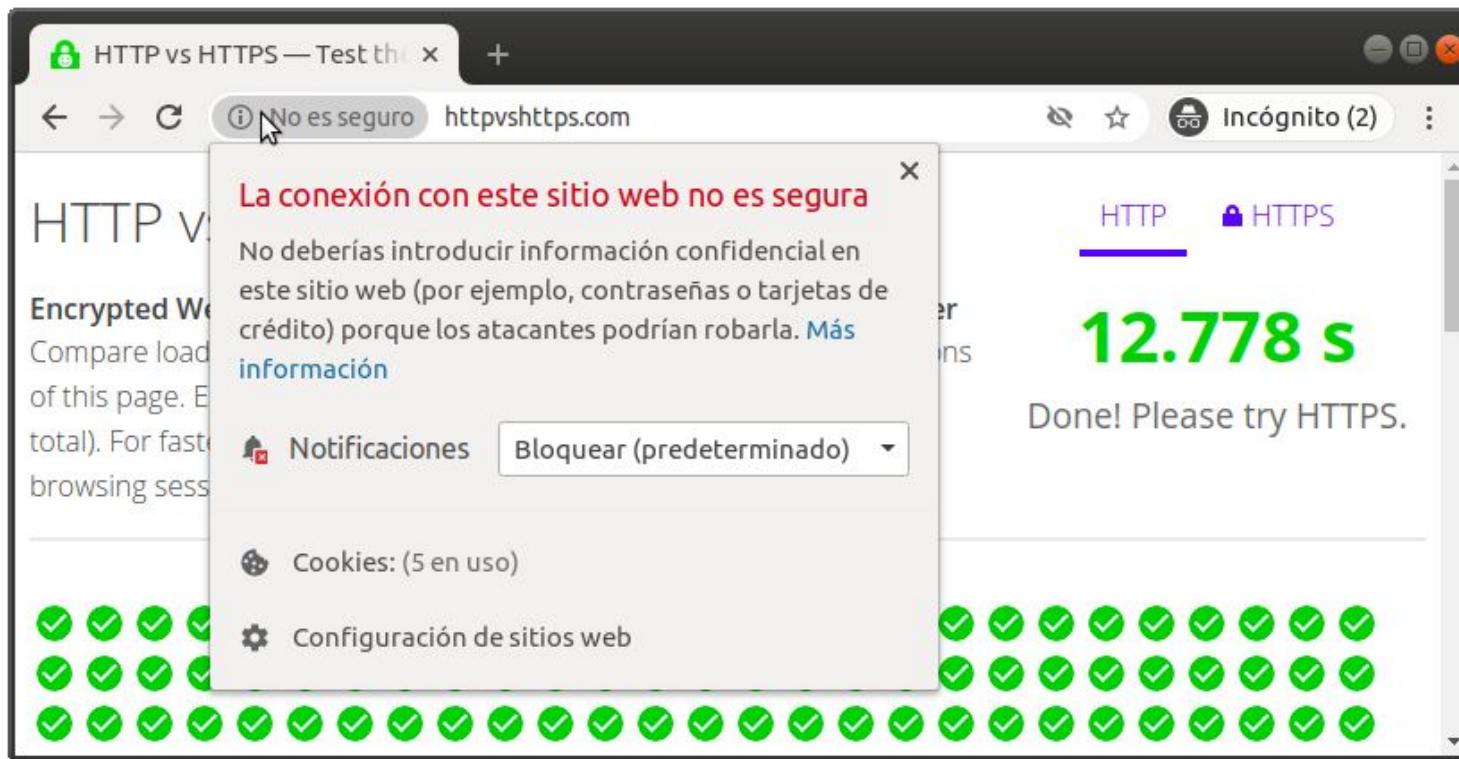
HTTP **HTTPS**

1.557 s

88% faster than HTTP



¿Qué significa ese: No es seguro?

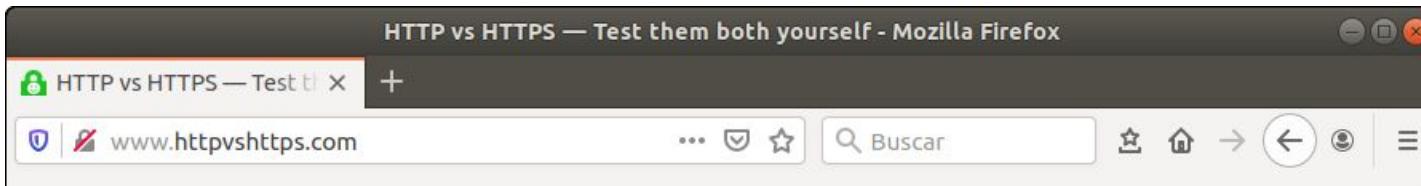


A screenshot of a web browser window titled "HTTP vs HTTPS — Test this site". The address bar shows "http://https://testthissite.com". A tooltip message "No es seguro" (Not secure) is visible over the address bar. The main content area displays a red warning message: "La conexión con este sitio web no es segura. No deberías introducir información confidencial en este sitio web (por ejemplo, contraseñas o tarjetas de crédito) porque los atacantes podrían robarla." Below this, there are dropdown menus for "Notificaciones" (Notifications) set to "Bloquear (predeterminado)" (Block) and "Cookies: (5 en uso)" (Cookies: (5 in use)). At the bottom, there is a "Configuración de sitios web" (Site settings) button. The background of the browser window features a grid of green checkmarks.



Como se distinguen los navegadores HTTP y HTTPS

Comparación en como se muestra un sitio HTTP y uno HTTPS válido



A screenshot of a Mozilla Firefox browser window. The title bar says "HTTP vs HTTPS — Test them both yourself - Mozilla Firefox". The address bar shows "HTTP vs HTTPS — Test them both yourself" and "https://www.httpvshttps.com". The page content is identical to the HTTP version but includes a "HTTPS" link under the heading and a green "1.288 s" badge indicating a faster load time. The Firefox logo is partially visible on the right side of the window.



Navegando en un sitio seguro

A screenshot of a Google search results page. The browser window title is "Google" and the address bar shows "google.com". The main content area features a colorful, whimsical illustration of various cartoon characters, including a lion, a monkey, and a bear, playing with kites and a sailboat on a beach-like setting. Below the illustration is a search bar with a magnifying glass icon and a microphone icon for voice search. At the bottom of the page, there are links for "Publicidad", "Negocios", "Sobre Google", "Cómo funciona la Búsqueda", "Buscar con Google", "Mostrar con cuenta", "Privacidad", "Condiciones", and "Preferencias".

Google

google.com

Incógnito

Gmail Imágenes

Iniciar sesión

Publicidad Negocios Sobre Google Cómo funciona la Búsqueda Buscar con Google Mostrar con cuenta Privacidad Condiciones Preferencias



Navegando en un sitio seguro

Google

google.com

Incógnito

La conexión es segura

Tu información (por ejemplo, las contraseñas o los números de las tarjetas de crédito) es privada cuando se envía a este sitio web. [Más información](#)

Ubicación: Bloquear

Notificaciones: Bloquear (predeterminado)

Certificado (válido)

Cookies: (6 en uso)

Configuración de sitios web

Gmail Imágenes Iniciar sesión

Publicidad

Privacidad Condiciones Preferencias

This screenshot shows a Google search results page for 'google.com'. A security warning overlay is displayed, stating 'La conexión es segura' (The connection is secure) and explaining that user information like passwords or credit card numbers are private when sent to this site. It includes links to 'Más información' and 'Bloquear' (Block) for location and notifications. Below the overlay, there's a cartoon illustration of children playing with paper boats on water. At the bottom, there are navigation links for 'Gmail', 'Imágenes', 'Iniciar sesión', and sections for 'Publicidad', 'Privacidad', 'Condiciones', and 'Preferencias'.



Datos del certificado del sitio visitado

The screenshot shows the 'Visor de certificados' (Certificate Viewer) for the domain *.google.com. The 'General' tab is selected. Key information displayed includes:

- Este certificado se ha verificado para los siguientes usos:** Certificado de servidor SSL.
- Enviado a**:
 - Nombre común (CN): *.google.com
 - Organización (O): Google LLC
 - Unidad organizativa (OU): <No incluido en el certificado>
- Emitido por**:
 - Nombre común (CN): GTS CA 1 O1
 - Organización (O): Google Trust Services
 - Unidad organizativa (OU): <No incluido en el certificado>
- Período de validez**:
 - Emitido el: miércoles, 15 de julio de 2020, 5:29:16
 - Vencimiento el: miércoles, 7 de octubre de 2020, 5:29:16
- Huellas digitales**:
 - Huella digital SHA-256: F4 57 71 F0 87 49 49 FE 10 C7 2F 56 9D 7B CA 4A
24 C6 9B 9B FC 2F 30 11 D8 F2 5A 5B 43 29 CE 01
 - Huella digital SHA-1: F8 94 0F 5D C6 9E 48 AB 4A D8 FA 83 93 B0 53 A1
2E 7D 4C 54

Datos de la identidad del sitio para el cual fue emitido este certificado.
***.google.com** vale para diferentes sitios en el dominio **.google.com**

Datos sobre el período de validez del presente certificado.

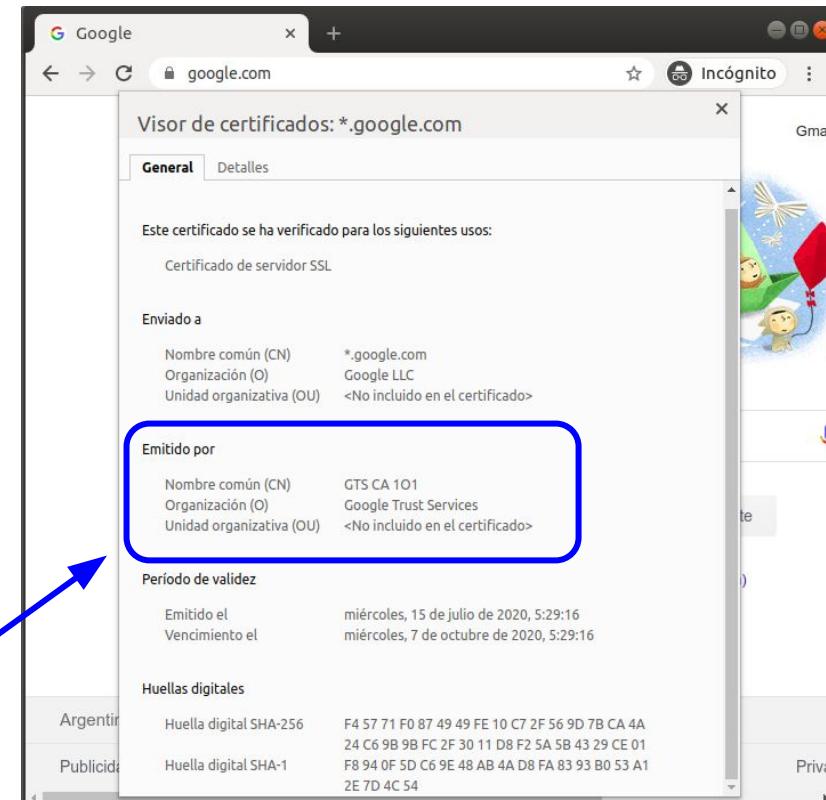
El certificado está firmado digitalmente por una CA

Los **certificados digitales**, son reconocidos como válidos, porque fueron emitido 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.

Entidad en la que confiamos!!!



Algunas Autoridades de Certificación en las que Chrome confía

Por defecto nuestros los navegadores vienen con conjunto de **Autoridades de certificación en la que se confía**.

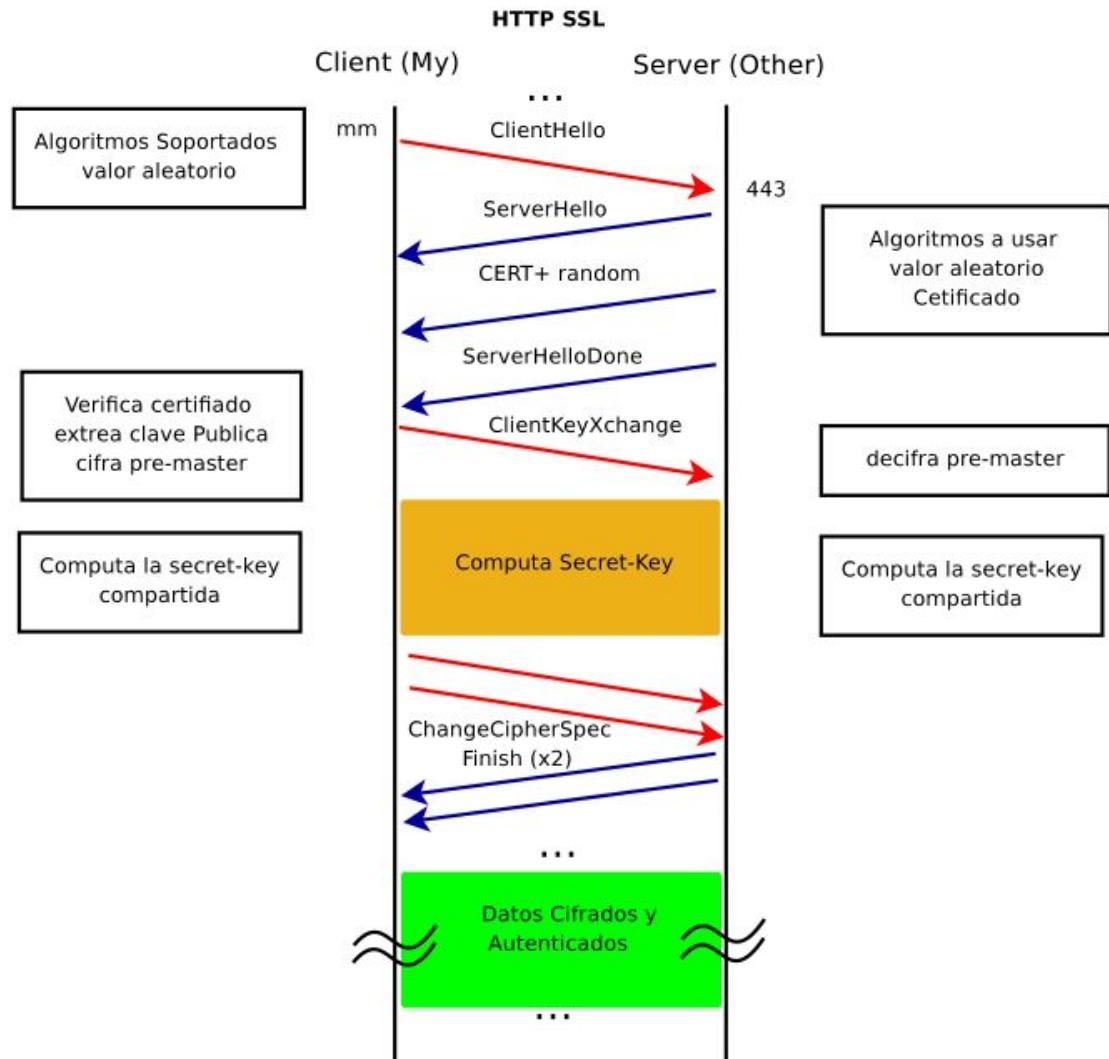
Nosotros podríamos agregar o borrar alguna **autoridad de certificación**, pero no es algo que se suela hacer.

org-E-Tuğra EBG Bilişim Teknolojileri ve Hizmetleri A.Ş.	▼
org-Entrust, Inc.	▼
org-Entrust.net	▼
org-FNMT-RCM	▼
org-GeoTrust Inc.	▼
org-GlobalSign	▲
GlobalSign	⋮
org-GlobalSign nv-sa	▼
org-GoDaddy.com, Inc.	▼
org-Government Root Certification Authority	▼
org-GUANG DONG CERTIFICATE AUTHORITY CO.,LTD.	▼
org-Hellenic Academic and Research Institutions Cert. Authority	▼
org-Hongkong Post	▼

HTTPS

HTTP sobre TLS/SSL

- Utiliza por defecto el puerto 443
- Lleva a cabo un handshake (negociación) antes del intercambio de datos.



Otros protocolos seguros

El protocolo SSL/TLS tiene multitud de aplicaciones en uso actualmente. La mayoría de son versiones seguras de programas que emplean protocolos que no lo son. Ejemplos:

- SSH utiliza SSL/TLS por debajo.
- SMTP y NNTP pueden operar también de manera segura sobre SSL/TLS.
- POP3 e IMAP4 sobre SSL/TLS son POP3S i IMAPS.

Problemas cuando usamos HTTPs

Cuando accedemos a un sitio HTTPs, se pueden presentar distintos problemas que no permiten al navegador asegurar la identidad del sitio visitado.

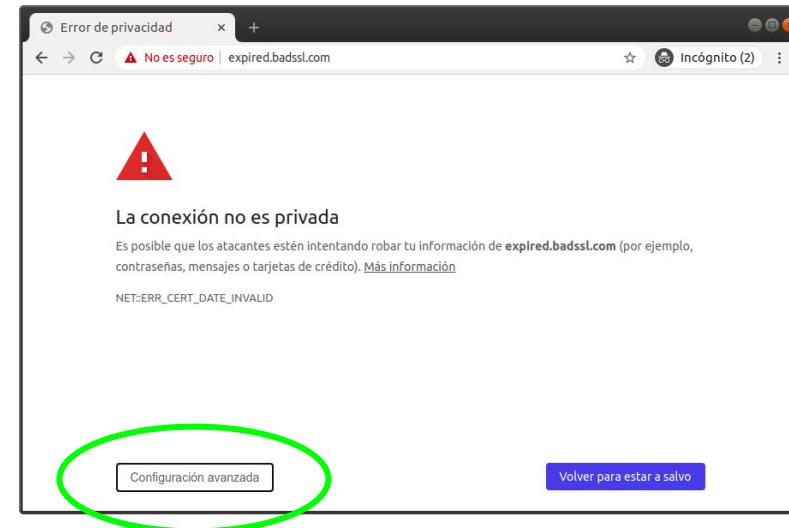
Cuando ocurre alguna de estas situaciones, el navegador alerta al usuario. Es importante que el usuario entienda que el problema es que **no se puede asegurar la identidad del sitio visitado**.

Estos problemas pueden deberse tanto a errores en la configuración del sitio como así también a ataques de phishing contra los usuarios.

Más información

En caso que no se trate de un phishing, algunas situaciones en las que podemos ver este tipo de alertas es cuando:

- El certificado está vencido.
- No coincide la URL del sitio visitado con la identidad del certificado presentado por el sitio web.
- El certificado está autofirmado.
- El certificado está firmado por una CA en la que no se confía.
- Cuidado con la fecha/hora del sistema!!



Problemas cuando usamos HTTPs

En estos dos videos pueden encontrar más información para complementar:

- <https://youtu.be/tHhFQaurGAg> (muy básico y sencillo)
- <https://youtu.be/pOeWmStBOYY> (un poquito más detallado)
- <https://www.websecurity.digicert.com/es/es/security-topics/what-is-ssl-tls-https> (un poco más de info en Digicert!)