T.5 CAPA DE APLICACIÓN

1.Protocolos a nivel de aplicación

Objetivo básico

Los protocolos a nivel de aplicación definen cómo los procesos de aplicación, que se ejecutan en sistemas diferentes, se envían mensajes entre sí y cuáles

Protocolos propietarios y protocolos de dominio público

Los protocolos de aplicaciones de red de gran número de empresas son propietarios

Los protocolos relacionados con Internet son de dominio público. Se definen en documentos llamados Requests for Comments (RFCs)

- [RFC 2821]: SMTP (Simple Mail Transfer Protocol)
- [RFC 2616]: HTTP (HyperText Transfer Protocol)

2.Requisitos de la aplicación de red

Muchos aspectos que se pueden considerar:

-Fiabilidad de la transmisión: Fiable / no fiable

-Ancho de banda (bandwitdh): Requisitos mínimos / flexible

-Requisitos de latencia (timing): Sensibles / no sensibles a la latencia

Seguridad

Aplicación	Pérdida de datos	Ancho de banda	*Timing*
Transferencia de ficheros	No	Flexible	No
Correo electrónico	No	Flexible	No
Páginas Web	No	Flexible (varios Kbps)	No
Audio/video en tiempo real	Sí	Audio: kbps – 1Mbps Video: 10kbps – 5 Mbps	100s de msegs
Mensajería instantánea	No	Flexible	Pocos segs

3.Modelos de aplicaciones

Dos modelos básicos

Cliente / servidor

Peer to Peer (P2P)

Modelo cliente / servidor

Las aplicaciones se componen de dos entidades: clientes y servidores:

Los servidores son entidades que ofrecen servicios

Los clientes solicitan los servicios de los servidores

Modelo P2P: No existen distinción de roles entre las entidades de la aplicación.

4.Direccionamiento

Puertos:

Vienen identificados por números naturales

Muchos números están asignados a aplicaciones de red concretas

Fichero /etc/services

Servicio	Puerto	Puerto (seguro)	Protocolo
ftp-data	20/tcp	989/tcp	File Transfer [Default Data]
ftp	21/tcp	990/tcp	File Transfer [Control]
ssh	22/tcp		Acceso remoto seguro
telnet	23/tcp	992/tcp	Telnet
smtp	25/tcp	465/tcp	Correo electrónico
dns	53/udp		Resolución de nombres
dhcp	67/udp	-	Configuración dinámica
http	80/tcp	443/tcp	World Wide Web HTTP
pop3	110/tcp	995/tcp	Recuperación de e-correo
imap	143/tcp	993/tcp	Recuperación de correo

5.Servicios ofrecidos por transporte

Dos protocolos de transporte: TCP (Transmission Control Protocol), UDP (User Datagram Protocol)

Servicios TCP: Servicio orientado a la conexión. Servicio de transporte fiable

UDP: Servicio sin conexión. Servicio de transmisión no fiable

Ni TCP ni UDP garantizan tasas mínimas de transmisión

Ejemplos de aplicaciones y protocolos

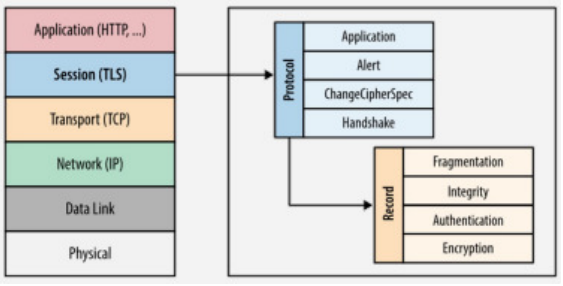
Aplicación	Protocolo de aplicación	Protocolo de transporte
Correo electrónico	SMTP [RFC 2821]	TCP
Terminal remoto	Telnet [RFC 854]	TCP
Acceso a la Web	HTTP [RFC 2616]	TCP
Transferencia de ficheros	FTP [RFC 959]	TCP
Multimedia (*streaming*): Vídeo almacenado Tiempo real	Propietario DASH WebRTC	TCP o UDP TCP UDP
Telefonía por Internet (VoIP)	Propietario	Típicamente UDP
Configuración dinámica	DHCP [RFC 2131]	UDP
Resolución de nombres	DNS [RFC 1035]	UDP

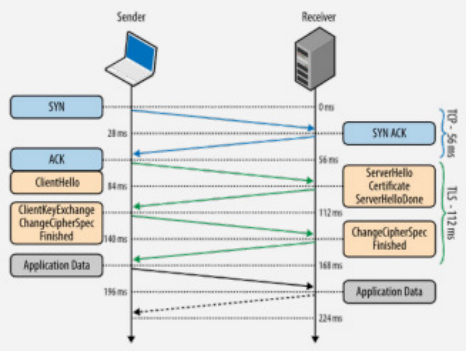
6.Seguridad

Ni TCP ni UDP ofrecen seguridad en la transmisión de datos
Secure Socket Layer (SSL): Añade aspectos de seguridad sobre la capa de transporte
SSL proporciona:
-Integridad de datos: los datos no pueden ser modificados externamente (es capaz de detectar si hubo cambios) <- Hashing
-Encriptación: los datos no son legibles por entidades externas a la conversación
-Autenticación: identificación de que los mensajes lo ha enviado el equipo que dice enviarlo

SSL / TLS

Inicialmente SSL se diseño para conexiones seguras en HTTP, pero evolucionó a TLS (TCP y extendido a UDP como DTLS) como capa intermedia entre transporte y aplicación y actualmente tiene un uso general (correo, multimedia, acceso remoto…)





7.DNS: Preguntas

¿Qué significa el acrónimo DNS? ¿Para qué sirve? ¿Cuándo se usa?
Domain Name System (Sistema de nombres de dominio)
Convierte nombres a direcciones IP (y también al revés!)
Uso:
Herramientas estándar: ping
Herramientas específicas: nslookup, dig
No es una conversión unívoca:
Una IP -> múltiples nombres: alias, diferentes contenidos
Un nombre -> múltiples IPs: equilibrado de carga / CDN
¿Es un servicio centralizado o distribuido? ¿por qué debe ser así?
Distribuido (los servidores solo almacenan parte de la BD)
Centralizado == cuello de botella, lento y difícil de manejar
Perspectiva histórica:
-ARPANET: 1969
- Poco equipos
- No necesidad de nombres

-Tabla de búsqueda de hosts: 1971-72

- Empieza a haber más equipos y se usan nombres
- Fichero local con la equivalencia entre nombre e IP (por equipo)
- ¡Todavía existe y es lo primero que se consulta!

-RFCs DNS: 1983-87

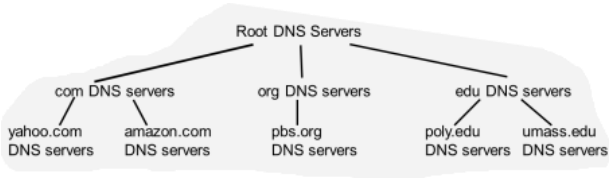
- Bases de datos distribuida

-Seguridad/privacidad: 2005 pero principalmente 2017+

- DNSSEC (2005), DNS over DTLS (2017), DNS over HTTPS (2018)

¿Cuántos y cuales tipos de servidores existen? ¿En qué orden se usan?

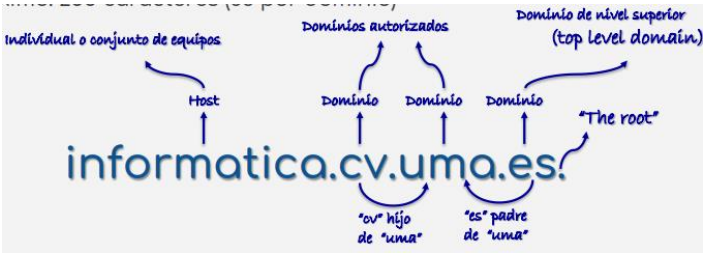
Servidores raíz (root)
Servidores top-level domain (TLD)
Servidores DNS autorizados (ANS)
Servidores locales (LNS)



Ejemplo: www.amazon.com.
-Consultamos nuestro servidor local (LNS)
-LNS solicita a root encontrar el TLD de .com
-LNS solicita al TLD .com obtener el ANS amazon.com
-LNS pregunta al ANS amazon.com la IP de www.amazon.com

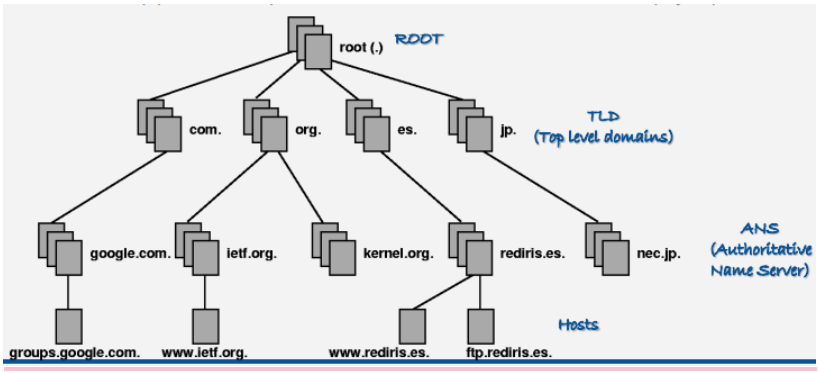
DNS: Hosts, dominios, FQDN

FQDN: Fully Qualified Domain Name:
-Nombre completo con equipo y dominio
-Máximo: 255 caracteres (63 por dominio)



DNS: Servicios y estructura

Cada nodo tiene ciertos registros (equivalencia entre nombre y dirección IP) y sabe a quién consultar sus subdominios (hijos)



Un mismo organismo (servidor) puede tener almacenado varios nodos del árbol (zona)

DNS: Servidores de nombre raíz

Servidor de nombre raíz (root name server): Almacena las IPs de todos los servidores TLD

DNS: Top level domains

Dominios de nivel superior (TLD):
-Geográficos (ccTLD): .es, .fr, .jp …
-Genéricos (gTLD): .com, .net, .name… (2012+ Nuevos – genéricos: .app, .ninja, .dev, .duck, .rip, .coffee, .expert… – marcas: .google, .zara, .java, .hbo …, – en español: .soy, .abogado, .futbol, .juegos …)
-Infraestructura (iTLD): .arpa (se usa en DNS reverso 12.30.222.56 => 56.222.30.12.in-addr.arpa)
-Problemática: acortadores (goo.gl es de Groenlandia?, bit.ly es de Libia?), ambigüedades (twitch.tv es de Tuvalu?)
-Gestión gTLD: compañías, paises, DNS Registry, ICANN…

DNS: Servidores autorizados

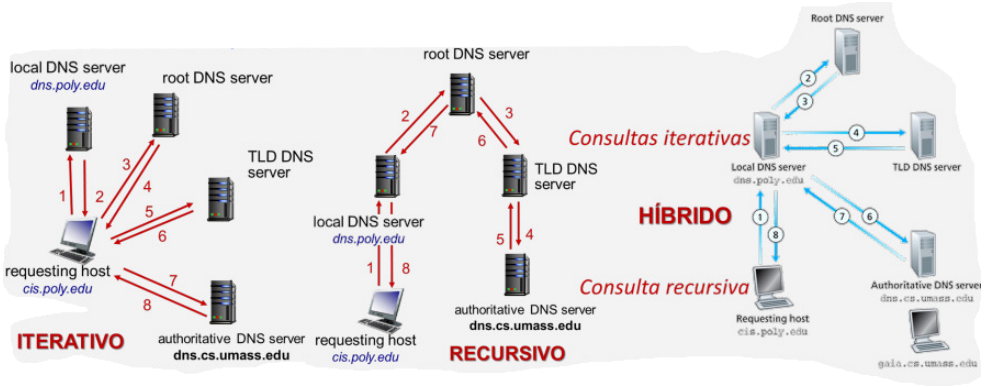
Servidores DNS autorizados (ANS):

-Servidores DNS propios de una organización que proporcionan traducciones autorizadas de los hosts de su organización.
-Puedes ser mantenidos por la propia organización o por un proveedor de servicios (registrar).
-Al registrar un dominio de este nivel se indica la localización del ANS y se informa al TLD oportuno

Según el vídeo es un servicio muy usado y debe ser eficiente, ¿qué protocolo de transporte usaría? ¿TCP o UDP?
UDP
En el vídeo la URL (www.example.com.) se resuelve a la dirección IP 192.168.1.1, ¿cree que eso es normal / correcto?
Esa IP es privada y salvo que sea un DNS interno nunca debe resolverse a una dirección privada

Protocolo DNS
Dos tipos de mensajes: Petición (Request), Respuesta (Response)
Los mensajes DNS van en datagrama UDP: Existen variantes cifradas sobre DTLS, TCP/TLS o sobre HTTPS. El puerto por el que DNS recibe peticiones es el 53
Hay diferentes tipos de registros:
-A: Nombre => IPv4
-AAAA: Nombre => IPv6
-CNAME: Nombre => Nombre canónico
-NS: Nombre => Servidor autorizado
-PTR: IP => Nombre

¿En que orden y cómo se consultan a los servidores?

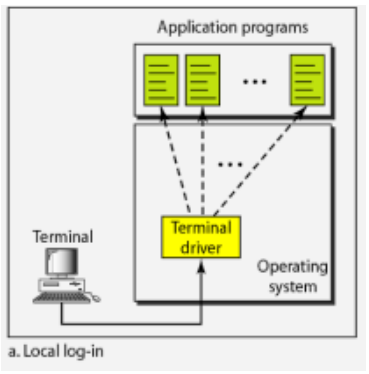


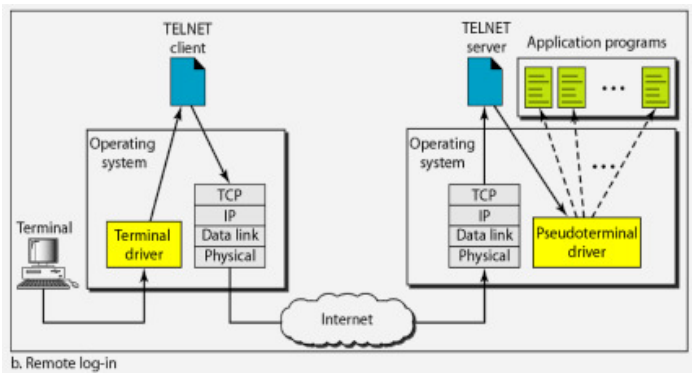
¿Cuál se indica en el vídeo? ¿Son obligatorios todos los saltos?
Híbrido (es el mas habitual). Las consultas previas son almacenadas y son utilizadas para reducir el número de consultas

DNS: cacheado y actualización
Cuando un servidor DNS aprende un nombre, lo almacena en cache
-Las entradas de la cache son limpiadas periódicamente
-Los servidores TLD están normalmente cacheados
    - No es necesario consultar los servidores root (información mínima)

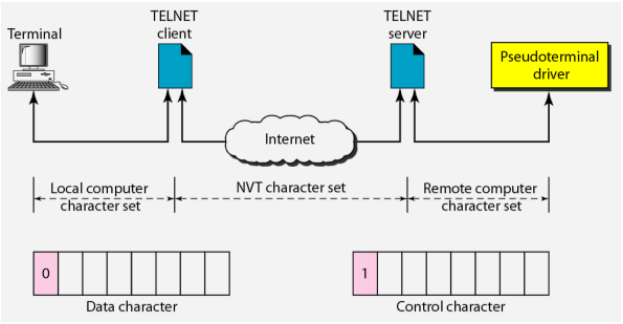
Las entradas memorizadas pueden estar desfasadas
-Cada entrada tiene un tiempo de vida (TTL) tras el cual expira
Los mecanismos para la actualización y notificación están recogidos en el RFC 2136:
-Es un aspecto muy problemático
-Entradas incorrectas impiden acceder al equipo
-Entradas falseadas (DNS Poisoning) podrían provocar ataques de phishing

Terminal Remoto: Telnet
Es una aplicación cliente -servidor de propósito general que permite el establecimiento de una conexión con un sistema remoto de forma que se puedan ejecutar comandos en el sistema remoto .
Protocolo/puerto: TCP/23





Problemas debido a la heterogeneidad de los sistemas
Ej: fin de fichero -> Windows: Ctrl + z, Linux: Ctrl + d
Los caracteres viajan a través de la conexión en un conjunto de caracteres universal denominado Caracteres de terminal virtual de red (NVT)



Telnet vs SSH

Telnet:

-Desarrollado en 1969
-Sólo sirve para acceder en modo terminal
-Problema de seguridad: contraseñas viajan como texto plano:
    Cualquiera que espíe el tráfico de la red pueda obtener los nombres de usuario y contraseñas, además de la información

SSH:

-Desarrollado en 1995
-Protocolo similar pero con comunicaciones encriptadas
-Requiere más recursos del ordenador
-Cifra la información antes de transmitirla
-Autentica la máquina a la cual se conecta
-SSH suele trabajar en el puerto 22 (tcp)

Otros usos Telnet:

El telnet puede utilizarse para establecer conexiones TCP a diferentes puertos: telnet dirección puerto
Permite realizar ciertas pruebas y comprobaciones:
- Comprobar si un determinado puerto TCP está “abierto”.
- Probar un servicio que se está desarrollado (si es “modo texto”)
- Ciertos servicios lo utilizan (Interactuar con el emulador de Android)

World Wide Web

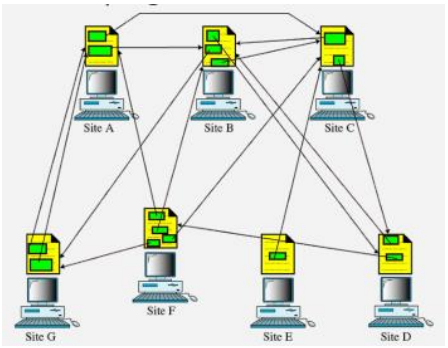
La World Wide Web (o Web) es un repositorio de información diseminada por todo el mundo y enlazada entre sí
    Iniciado por el CERN para gestionar y acceder a los recursos distribuidos requeridos por los investigadores de este centro
Es un servicio distribuido (cliente/servidor)
    Cliente es un navegador que accede a un recurso

El servicio de acceso está distribuido entre numeroso sitios
- Cada sitio almacena uno o mas documentos (páginas Web)
- Los navegadores permiten recuperar y visualizar páginas Web
- Una página Web puede contener un enlace a otras páginas o recursos (del mismo u otros sitios) – hipertexto

Arquitectura

El cliente quiere visualizar una información almacenada en el sitio A
- Envía una petición a través de su navegador ○ La petición incluye la dirección del sitio y el recurso -> URL
- El servidor encuentra el documento y lo envía al cliente

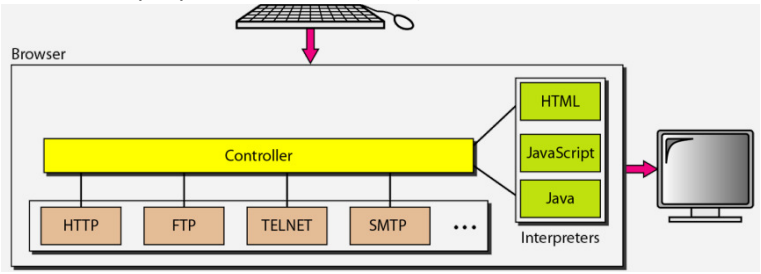
La página web contiene una referencia a páginas de otros sitios
- La referencia incluye la URL
- El cliente envía otra petición al nuevo sitio y recupera la página



Navegador (cliente)

Hay diversos fabricantes que proporcionan navegadores que interpretan y visualizan documentos Web
- Explorer/Edge, Firefox, Chrome, Safari, Opera

Cada navegador consta de tres partes:
- Un controlador (consigue la información del usuario)
- Un protocolo (accede al servidor y recupera el documento)
- Intérpretes (muestra de forma apropiada el documento)



URL

El Localizador de Recursos Uniforme (URL) es un estándar que identifica y especifica cualquier tipo de información en Internet
- HTTP lo utiliza para el acceso de los documentos

El URL define 4 elementos:
- El protocolo: Protocolo o aplicación cliente/servidor utilizado para recuperar el recurso
- El equipo (host): Dirección IP o nombre de dominio de la máquina donde se está el recurso
- El puerto: Información opcional. Normalmente se toma el puerto por defecto (80/443) del protocolo
- El camino (path): Camino completo para localizar el recurso en la estación indicada (directorios…)

Páginas web

Las páginas Web
- Es un fichero HTML que incluyen referencias a otros objetos :
  - Ficheros HTML
  - Elementos multimedia (imagines, audio, video)
  - Código (ej, Javascript)
- HTML (Hyper Text Markup Language)
  - Es un texto anotado con etiquetas (término entre <…>)

El protocolo HTTP

HTTP (HyperText Transfer Protocol)

Protocolo de capa de aplicación para acceso de documentos Web
Modelo Cliente/servidor:
- Puerto: 80 (TCP)

Protocolo sin estado:
- El servidor no mantiene información de las peticiones previas del cliente

HyperText Transfer Protocol: Transferencia de ficheros para la web
Puerto: 80 (http) y 443 (https)
Protocolo: TCP
Protocolo sin estado (toda la información debe ir en cada petición/respuesta => Cookies)

Peticiones:
Tipo (GET, POST, DELETE, PUT…), recurso solicitado y versión
Cabeceras (información sobre la petición: formato, host…)
Recurso

Respuesta:
Versión, código (1xx, 2xx, 3xx, 4xx , 5xx) y explicación (texto)
Cabeceras (información sobre la respuesta: formato, tamaño…)
Recurso

HTTP Secure (HTTPS)
HTTP sobre TLS
- Puerto: 443

HTTP: Mensajes

Petición HTTP: Del cliente al servidor
Respuesta HTTP: Del servidor al cliente
Formato (textual):
Línea inicial: Petición, Respuesta
Cabeceras
Línea en blanco
Datos

Petición: Métodos HTTP

Método	Acción
GET	Solicitud de un documento a un servidor
HEAD	Solicitud de información sobre un documento (no el documento en sí mismo)
POST	Envío de información al servidor para que sean procesadas por el recurso
PUT	Envío de un documento del servidor al cliente
DELETE	Elimina un recurso del servidor
CONNECT	Se utiliza para saber si se tiene acceso a un host y en los proxy
OPTION	Solicitud de algunas opciones disponibles

HTTP/0.9: GET
HTTP/1.0: GET, POST, HEAD
HTTP/1.1: GET, POST, HEAD, PUT, PATCH, DELETE

Respuestas HTTP: Tipos

Mensajes de respuesta (estado) y contenido (si lo hay) :
1xx : Confirmación preliminar
2xx : Confirmación
3xx : Se necesitan más acciones del cliente
4xx : Error en la petición
5xx : Error en el servidor
Ejemplo :
101 Switching
200 OK

301 Moved Permanently
400 Bad Request
404 Not Found
451 Unable For legal Reasons
505 HTTP Version Not Supported

HTTP: Cabeceras

Formato requerido para la respuesta:
    Accept, Accept-Charset, Accept-Encoding, Accept-Language…
Formato de los datos enviados:
    Content-Encoding, Content-Language, Content-Length, Content-Type…
Eficiencia:
    Cache-control, Expires, If-modified-since…
Control del protocolo:
    Authorization, Upgrade, Strict-Transport-Security (HSTS), Forwarded, Connection…
Información adicional:
    Date, User-Agent, Server, Host, X-…
"Sesiones":
    Set-Cookie, Cookie

HTTP: Documentos web

Tres tipos de documentos:
Estáticos
    ○ Recurso fijo del servidor
Dinámicos
    ○ El documento se crea en el servidor cuando llega una petición
    ○ CGI, Servlets, lenguajes scripts (PHP, JS, Python, Java…)
Activos
    ○ Programas que se ejecutan en el lado del cliente
    ○ El servidor envía un documento activo que se ejecuta en el lado de cliente
    ○ JavaScript, Silverlight, Flash, Applets de Java
Documentos web (HTML): Contienen múltiples objetos: Código Javascript, CSS, imágenes, vídeos, etc.

Petición HTTP: Envío de datos

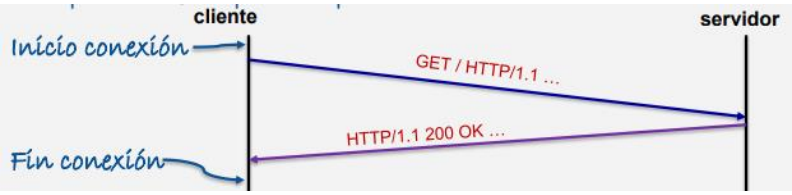
GET: (Idempotente) Los datos se codificación la URL
POST: Los datos se envían en el cuerpo de la petición
Además de la forma de envío existen notables diferencias:
GET: cacheable, se puede añadir a favoritos, se mantiene en el historial del navegador…
POST: longitud "ilimitada", permite datos binarios…

HTTP/1.1: Problemas

El estándar es muy grande:
    HTTP 1.0 (RFC 1945) – 60 páginas (1996)
    HTTP 1.1 (RFC 2616) – 176 páginas (1999)
    Reemplazado por 6 RFCs (7230 – 7235) – 200+ páginas
    Partes opcionales / implementación incompleta: Por ejemplo pipelining
Eficiencia:
    Sobrecarga de las cabeceras
    Gran impacto de la latencia (rtt):
        ○ Uso ineficiente de las conexiones TCP
        ○ Al menos un rtt por petición
        ○ Head-of-line blocking

HTTP: Uso ineficiente de conexiones TCP

Cada petición/respuesta por conexión TCP

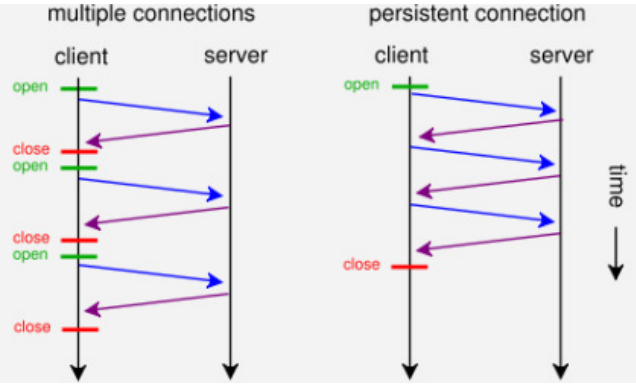


Problemas:

- Inicio/fin de la conexión (6/7 mensajes)
- Control de congestión (slow start)

HTTP: Mejora: persistencia

Utilizar la misma conexión para varios recursos:



HTTP/1.0 con la opción: Connection: Keep-Alive

HTTP/1.1 se activa por defecto

Se cierra con Connection: Close

HTTP/2 al rescate!

Objetivos:

- Aumentar la eficiencia
- Arreglar el pipelining y eliminar el head-of-line blocking
- No requerir múltiples conexiones
- No romper la web ya existente

Basado en el protocolo SPDY de Google

Aprobado el 18 de febrero de 2015:

- HTTP/2 (RFC 7540)
- HPACK: Compresión de la cabecera (RFC 7541)

Dos implementaciones:

- HTTP/2 sobre TLS (h2)
- HTTP/2 sobre TCP plano (h2c) (La mayoría de los navegadores/servidores no darán soporte)

La idea es mantener la semántica de HTTP/1.1:

- Mismos métodos (GET, PUT, …), cabeceras, casos de uso
- Mantiene el modelo cliente/servidor

Novedades:

- Binario (envío de “tramas”).
- Compresión de las cabeceras.
- Una sola conexión:
  - Múltiples de flujos (multiplexación de flujos):
    - Múltiples recursos por conexión.
    - Se pueden limitar (al menos se recomienda el uso de 100).
    - Cierre de flujos sin cerrar conexión (RST_STREAM).
  - Prioridades/dependencias en flujos y cambios dinámicos:

Recursos “fuera de orden”

○ Server Push: envío de información por el servidor sin ser solicitada

HPACK: Compresión de cabeceras

Uso de tablas de cabeceras:

Se crea tabla con cabeceras habituales (estática)

Se crea tabla con cabeceras usadas previamente (dinámica)

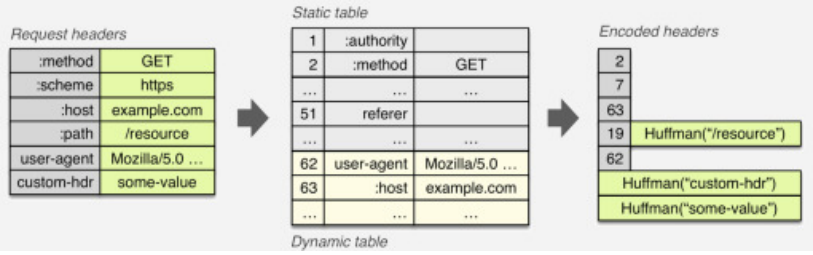
Se envían los índices de esas tablas

Los valores/cabeceras nuevas se codificación con Huffman:

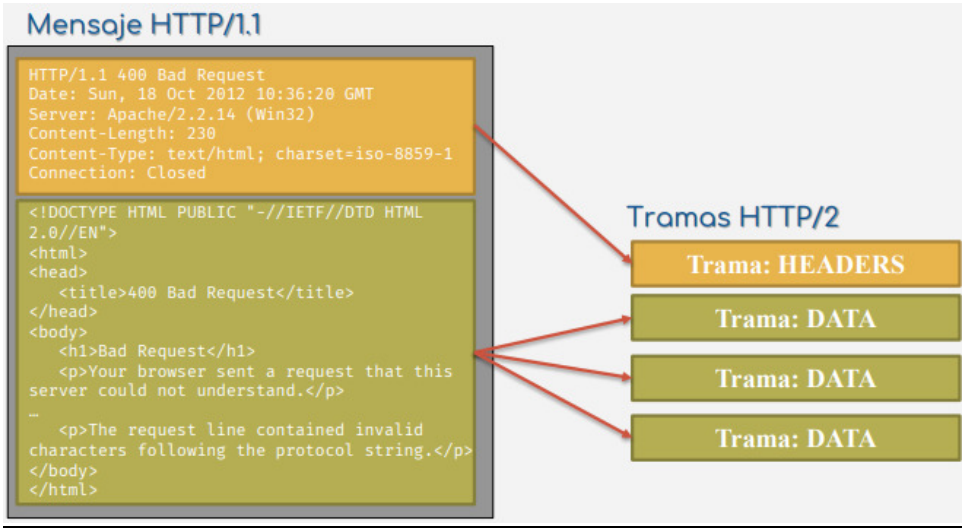
Los caracteres más usados usan menos bits. Ejemplos:

‘e’ se codifica como 00101

‘\’ se codifica como 1111111111111110000



HTTP/2: Comunicaciones



Tramas: unidad mínima de transferencia

Flujo: una petición/respuesta

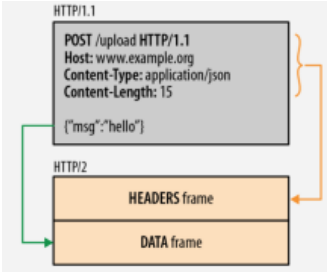
Compuesto por una o varias tramas

Una única conexión TCP :

Está formada por múltiples flujos

Prioridad (dinámica) y dependencia entre flujos

Se pueden mezclar tramas de diferentes flujos



HTTP/2: Trama

Formada por 9 bytes fijos y una parte dependiendo del tipo:

Bit	+0..7	+8..15	+16..23	+24..31
0	Length		Type	Flags
32	R	Stream Identifier		
...	*Frame Payload*			

Length: Longitud de la trama (sin incluir los 9 bytes iniciales)

Type: Tipo de trama

Flags: definidos atendiendo al tipo de trama

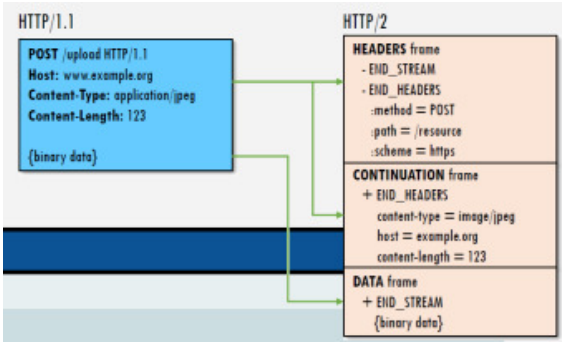
R: Reservado (0)

Stream Identifier: Identificador de flujo (petición/respuesta):
- 0: reservado para control de la conexión en general
- 1: para la conexión HTTP/1.1 inicial (si la hay)
- Iniciados por el cliente: impares
- Iniciados por el servidor: pares

Frame Payload: Datos (depende del tipo de trama)

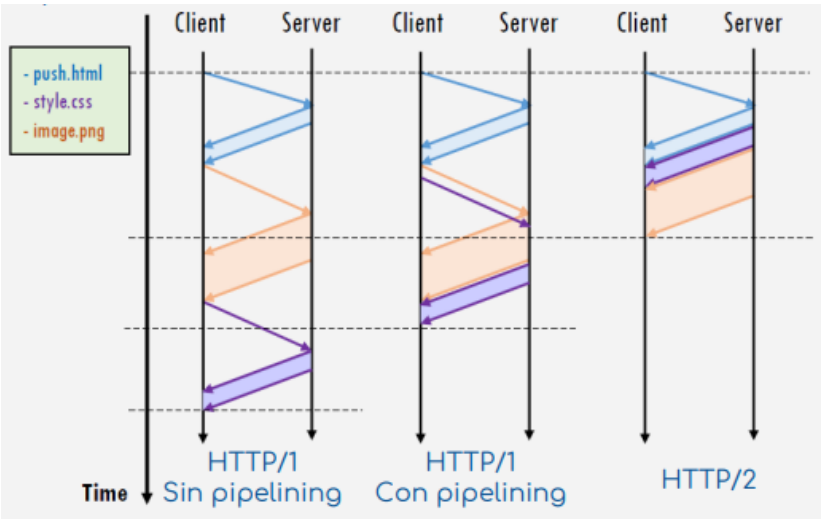
HTTP/2: Tipos de trama

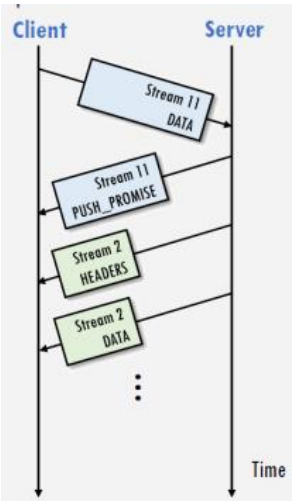
Tipo	Descripción
DATA	Cuerpo HTTP
HEADERS	Cabeceras
PRIORITY	Prioridad de un flujo
RST_STREAM	Finalización de un flujo
SETTINGS	Parámetros de configuración de la conexión (solo flujo 0)
PUSH_PROMISE	Envío de datos (sin haber sido pedidos)
PING	Para medir el rtt y que el otro extreme está active
GOAWAY	Informa al otro extreme que no cree más flujos en esta conexión
WINDOW_UPDATE	Control de flujo (flujo concreto o conexión global – flujo 0)
CONTINUATION	Continúa enviado cabeceras



HTTP/2: Server Push

El servidor puede enviar datos al cliente que sabe que va a necesitar sin pedirlos mediante la trama PUSH_PROMISE

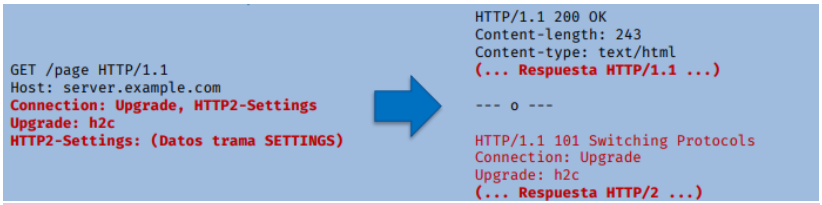




HTTP/2: Interacción con HTTP/1.1

-Implementación h2 (sobre TLS): Durante el proceso de autenticación TLS usa APLN (Application Layer Protocol Negotitation) para fijas http/2

-Implementación h2c (sobre TCP plano): Usa la cabecera Update



Si ya sabe que usa HTTP/2, el cliente puede empezar con:
0x505249202a20485454502f322e300d0a0d0a534d0d0a0d0a… PRI * HTTP/2.0\r\n\r\nSM\r\n\r\n + trama SETTINGS

Correo electrónico

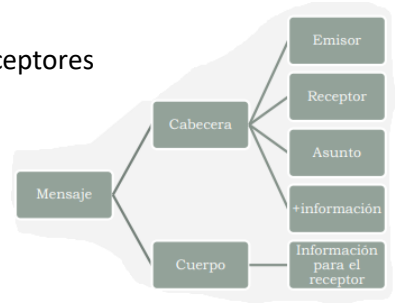
Uno de los servicios más populares de Internet

Hay tres componentes involucrados en el envío y recepción de correo electrónico
- Agentes de usuario (UA)
- Agentes de transferencia de mensajes (MTA)
- Agentes de acceso a mensajes (MAA)

Partes de un mensaje:

Sobre: Contiene las direcciones electrónicas del emisor y los receptores

Mensaje:



Dirección de correo electrónico :

Agente de usuario

Ofrece servicios para el envío o recepción de un mensaje

Programa que compone, lee, responde y envía mensajes

También incluye y gestiona buzones (uno de entrada y otro de salida)

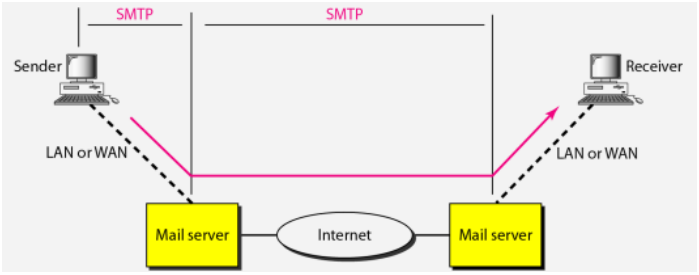
Ejemplos: Outlook, Netscape Messenger, Thunderbird, Mail

Agente de transferencia de mensajes (SMTP):

Para enviar un correo electrónico, se debe tener un cliente MTA y para recibir un correo electrónico debe tener un servidor MTA

El protocolo entre el cliente y el servidor MTA es SMTP (protocolo sencillo de transferencia de correo electrónico)

Puerto TCP/25 (o TCP/587 con SSL)



Mensajes en SMTP

Mensajes: COMANDO [parámetro]* (= espacio y es \r\n)

`HELO dominio`	Saludo (cambiado en ESMTP)
`MAIL FROM: <origen>`	Indica el origen del correo
`RCPT TO: <destino>`	Indica el destino (puede usarse múltiples veces)

`DATA`
`Mensaje`
`.`
	Contenido del mensaje (toda la información que deba recibir el cliente de correo). Acaba con una línea en blanco con un punto (`<CRLF>.<CRLF>`)

`RSET`	Resetea el envío del correo
`NOOP`	No hace nada (se usa de prueba o medir tiempos)
`QUIT`	Mensaje de despedida

Respuestas en SMTP

Respuestas: Si el comando requiere transferir datos se crea conexión de datos. Se envía código de respuesta: XYZ <sp> Explicación <crlf>. X = tipo Y = función Z = detalle

1YZ	Respuesta preliminar positiva	(Vamos bien pero **debo** hacer más)
2YZ	Se completó correctamente	(Todo fue bien)
3YZ	Respuesta intermedia positiva	(Vamos bien pero **debes** hacer más)
4YZ	Respuesta negativa temporal	(Algo **hiciste** mal)
5YZ	Respuesta negativa permanente	(Algo **hice** mal)

Este tipo de respuestas se usa en muchos servicios (HTTP, FTP…)

Ejemplo SMTP (C=cliente y S=servidor)

```
S:      220 sol10.lcc.uma.es Simple Mail Transfer Service Ready
C:      HELO sol10.lcc.uma.es
S:      250 sol10.lcc.uma.es
C:      MAIL FROM: <gabriel@lcc.uma.es>
S:      250 OK
C:      RCPT TO: <rusman@lcc.uma.es>
S:      250 OK
C:      RCPT TO: <mercedes@lcc.uma.es>
S:      550 No such user here
C:      DATA
C:      Subject: Reunión de RySD
C:      To: rusman@lcc.uma.es
C:      From: gabriel@lcc.uma.es
C:      Content-Transfer-Encoding: 8bit
C:      Hola,
C:      ¿Quedamos a las 19:00 en mi despacho?
C:      Gabriel
C:      .
S:      250 OK
C:      QUIT
S:      221 sol10.lcc.uma.es Service closing transmission channel
```

Sobre
Cabecera
Cuerpo
Mensaje
Acaba con un solo punto en una línea

Extended SMTP (ESMTP)

SMTP se extendió con algunos comandos:

-EHLO (vs HELO): El servidor responde informando de las extensiones soportadas

-STARTTLS: Cambia a formato encriptado usando TLS

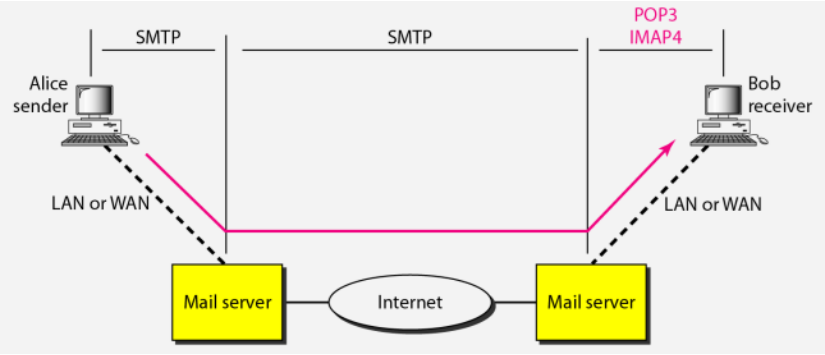
-SIZE: Indica el tamaño máximo aceptado (servidor) o el tamaño estimado del correo (cliente)

-AUTH: Autenticación del usuario:

- AUTH PLAIN: Tras enviar este comando el usuario debe enviar su nombre de usuario y clave juntos codificado en Base64
- AUTH LOGIN: Tras enviar este comando el servidor pedirá el nombre de usuario y posteriormente la contraseña (todo lo enviado, cliente y servidor, en Base64)
- AUTH CRAM-MD5: El servidor envía un reto para que el cliente envíe su contraseña cifrada

Recepción de correos electrónicos

Agente de Acceso a mensajes: POP3 e IMAP4: Se encarga de extraer los mensajes del buzón/servidor. Los mensajes van del servidor al cliente



Acceso al correo electrónico

-POP3 (protocolo de la oficina de correo):

Sencillo pero de funcionalidad limitada

El cliente se instala en la maquina del usuario

Protocolo TCP / 110

Dos modos de funcionamiento: borrado y mantenimiento

-IMAP4 (protocolo de acceso a correo de Internet):

Protocolo TCP / 143

Más complejo y potente que POP (usuarios nómadas)

Permite gestionar el buzón del servidor (consultar/buscar sin descargar, crear carpetas, marcar leídos…)

-Basados en la Web (Webmail tipo Gmail, Hotmail…):
Acceso al correo desde el navegador web
Combina el uso de SMTP con HTTP

Mensajes y Respuestas en POP3

Mensajes POP3 (implementación reducida)

```
USER<SP>name<CRLF>
PASS<SP>string<CRLF>
STAT<CRLF>
LIST [<SP>msg] <CRLF>
RETR<SP>msg<CRLF>
DELE<SP>msg<CRLF>
QUIT<CRLF>
```

Respuestas (ejemplos)

```
+OK<CRLF>
-ERR<CRLF>
```

Ejemplo POP3 (C=Cliente y S=Servidor)

```
S: <wait for connection on TCP port 110>
C: <open connection>
S:    +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C:    USER rdo
S:    +OK rdo
C:    PASS rdo123
S:    +OK
C:    STAT
S:    +OK 2 320
C:    LIST
S:    +OK 2 messages (320 octets)
S:    1 120
S:    2 200
S:    .
C:    RETR 1
S:    +OK 120 octets
S:    <the POP3 server sends message 1>
S:    .
C:    DELE 1
S:    +OK message 1 deleted
C:    RETR 2
S:    +OK 200 octets
S:    <the POP3 server sends message 2>
S:    .
C:    DELE 2
S:    +OK message 2 deleted
C:    QUIT
S:    +OK dewey POP3 server signing off (maildrop empty)
C: <close connection>
```

Acaba con un solo punto en una línea

File Transfer Protocol – FTP

Protocolo que permite transferir ficheros entre equipos Se basa en el modelo cliente/servidor y usa el protocolo TCP
Los ficheros se organizan en una jerarquía de directorios Controla el acceso de usuarios (autorización)
Dos modos de transferencia:

- ASCII: Los datos se envían convertidos a ASCII de 8 bits (Tipo A)
- Binario (o Imagen): Los datos se envían sin conversión (Tipo I)

Operaciones de usuario:

- Acceder/acabar: open, quit
- Transferir: get, put, mget, mput
- Modos de transmisión: binary, ascii
- Navegar por directorios: cd, dir
- Otros: help, status…

Establece dos tipos de conexiones entre los computadores involucrados en la transferencia de ficheros:
Conexión de control (TCP/21):

- -Se mantiene abierta durante toda la sesión
- -Se dedica para el envío de los comandos y el estado de la respuesta (fue bien o no)

Conexiones de datos (TCP/20 si se usa modo activo):

- -Se crea cuando un comando requiere el envío de datos (transferencia de fichero o que genere respuesta como dir)
- -Se usa para enviar dichos datos y luego se destruye

Mensajes: COMANDO [<sp> parámetro] <crlf> (sp = espacio, crlf = \r\n)

USER <username>	Nombre del usuario que quiere conectarse
PASS <password>	Clave del usuario
QUIT	Cerrar sesión
CWD <pathname>	Cambio al directorio indicado (relativo)
PORT <host-puerto>	Abrir conexión de datos en modo activo
PASV	Abrir conexión en modo pasivo
TYPE <I o A>	Modo de transferencia de datos
RETR <pathname>	Descargar el archivo indicado
STOR <pathname>	Subir el archivo indicado
LIST <pathname>	Lista el contenido del directorio

Respuestas:

Si el comando requiere transferir datos se crea conexión de datos

Se envía código de respuesta: XYZ <sp> Explicación <crlf>

X = tipo Y = función Z = detalle

1YZ	Respuesta preliminar positiva	(Vamos bien pero **debo** hacer más)
2YZ	Se completó correctamente	(Todo fue bien)
3YZ	Respuesta intermedia positiva	(Vamos bien pero **debes** hacer más)
4YZ	Respuesta negativa temporal	(Algo **hiciste** mal)
5YZ	Respuesta negativa permanente	(Algo **hice** mal)

Este tipo de respuestas se usa en muchos servicios (HTTP, SMTP…)

Modos de creación la conexión de datos en FTP

Activo:

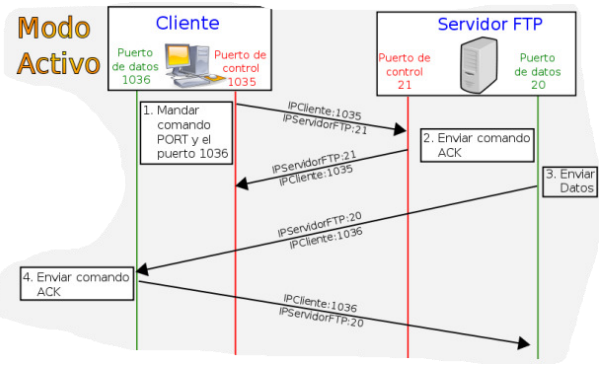
- El cliente crea el socket para la transferencia de datos.
- El cliente envía el puerto al servidor mediante un comando PORT
- El servidor se conecta (puerto 20) y establece la conexión de datos.

Pasivo:

- El cliente envía el comando PASV
- El servidor crea el socket para la transferencia de datos.
- El servidor comunica el puerto al que debe conectarse.
- El cliente se conecta a dicho puerto y se establece la conexión de datos.

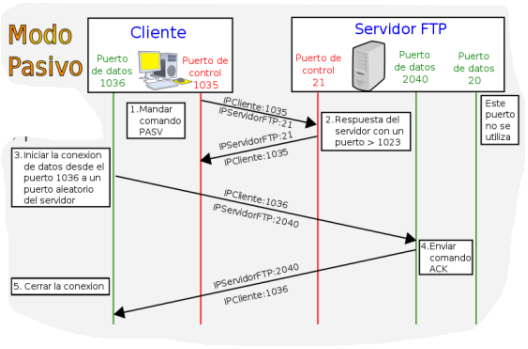
Modos activo (PORT): El cliente FTP actúa de servidor para la conexión de datos:

-El cliente crea servidor
-Envía PORT con IP, puerto
-El servidor (p. 20) se conecta
-Se envían los datos
-Se cierra la conexión



Modos pasivo (PASV): El servidor FTP actúa de servidor para la conexión de datos:

-El servidor FTP crea servidor
-Envía respuesta con IP, puerto
-El cliente se conecta
-Se envían los datos
-Se cierra la conexión



Introducción a multimedia

Definición: "El término Multimedia se utiliza para indicar que la información y los datos de una aplicación integran o pueden estar compuestos de diferentes tipos de medios (de comunicación)" Los tipos de medios de comunicación son: Texto, Imágenes (Gráficos, fotografías), Audio (Voz , Música), Vídeo.

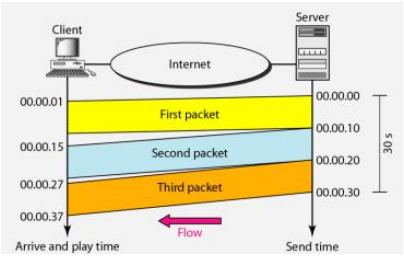
Los principales problemas son en recursos de gran tamaño que se van consumiendo a la vez que se reciben.

Streaming audio/video:

-Almacenado (Youtube, Netflix, TV a la carta, …),

-En directo (TV en directo),

-Tiempo real/interactivo (Videoconferencia, videollamada, …)



Problemas a resolver :

-Retraso en los paquetes

-Latencia variable (jitter)

-Pérdida de paquetes

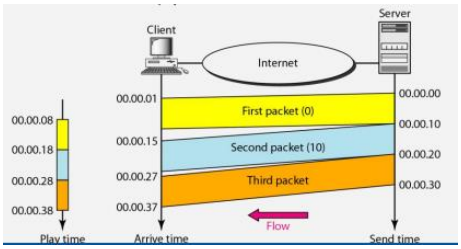
Soluciones :

-Marcas de tiempo

-Almacenamiento (buffers)

-Entrelazado

-Numeración



Caso Estudio 1: Netflix, Youtube

Los recursos multimedia ya están disponibles antes de su envío al usuario y se pueden pre-procesar

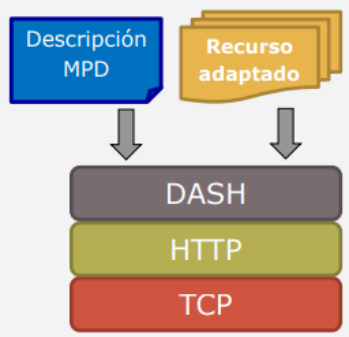
Características deseables:

-Localización del recurso conocida (URL)

-Accesibilidad universal y sencilla

-Transmisión sin errores

-Ajustar a las características del usuario



DASH (Dynamic Adaptive Streaming over HTTP):

-Transmisión: HTTP. Extendido y permitido universalmente

-Recursos multimedia adaptables:

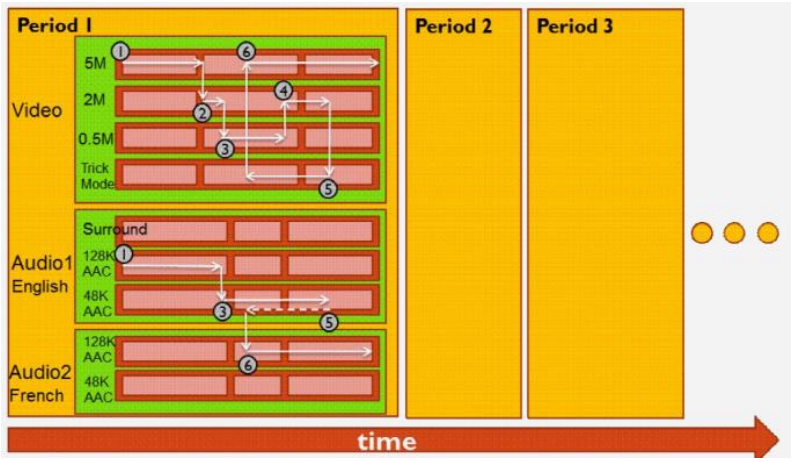
- Dividido en pequeños trozos
- Cada trozo disponible en diferentes calidades

¿Cómo se conocen los trozos disponibles?

¿Cómo se elige la calidad?

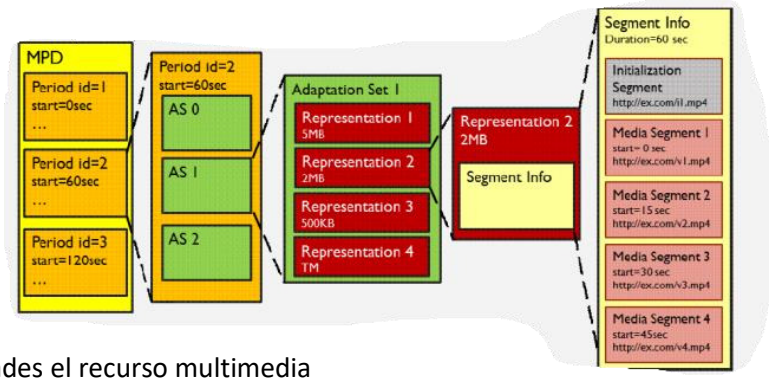
-DASH: Ejemplo de uso

Esquema básico de funcionamiento a nivel de usuario :



-DASH: Fichero MPD
DASH define:
Cómo se describe el recurso multimedia (MPD –xml– Multimedia Presentation Description)

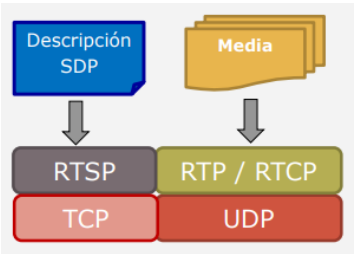
Formato de los segmentos de información
Otras características: DRM, publicidad, selección de flujos, segmentos de longitud variable, …



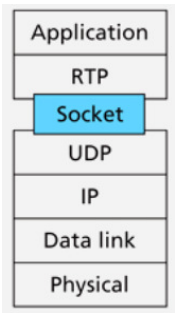
- DASH: Funcionamiento
Servidor: trocea y codifica en varios formatos/calidades el recurso multimedia
Servidor: genera el fichero MPD
Cliente: obtiene el fichero MPD usando HTTP
Cliente: analiza MPD
Cliente: usa estadísticas => segmento apropiado con HTTP
Cliente: recibe segmento y obtiene el recurso y se lo pasa al reproductor

Caso Estudio 2: Videocámara, TV online…
El recurso multimedia se va generando a la vez que se consume. No es bidireccional.
Características deseables:
- Localización del recurso conocida (URL)
- Primar la velocidad

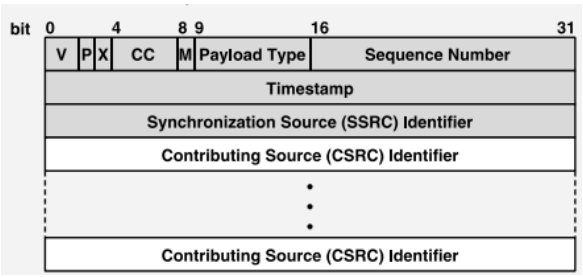
Solución propuesta:
- Transporte: UDP
- ¿Es UDP suficiente?
  - Sincronización flujos multimedia
  - Temporización, congestión…
- ¿Cómo controla el cliente la reproducción (parar, pausar, reiniciar…) y hace control de flujo?
- ¿Cómo se saben los flujos y características?



Protocolo de tiempo real (RTP)
Es un protocolo del nivel de aplicación que permite la transferencia de datos con características de tiempo real e información de control
Es un protocolo desarrollado por la comunidad de Internet y se recoge en el RFC 3550 (que reemplaza al RFC 1889)
Este protocolo está compuesto de dos protocolos que colaboran estrechamente entre sí: RTP y RTCP
Este protocolo utiliza los servicios de UDP para la transmisión de sus paquetes



El protocolo define un único paquete de datos: Paquete RTP:
- -Tipo de carga útil
- -Número de secuencia
- -Marca de tiempo
- -Identificador de fuente de sincronización (identificador de las comunicación - Coordinador)
- -Identificadores de fuente de contribución (identificador del origen de este flujo de vídeo/audio, …)



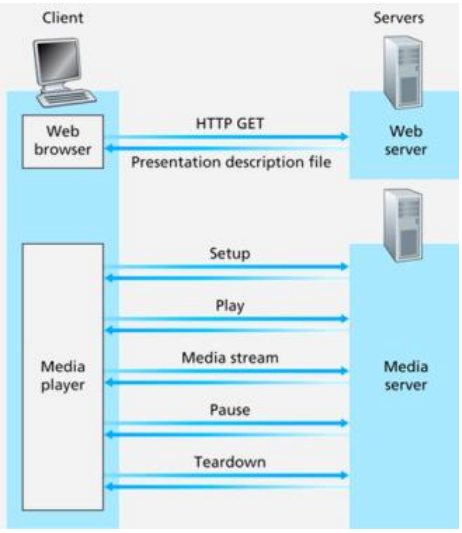
Protocolo de control de tiempo real (RTCP)
Ofrece los siguientes servicios:
Monitorización de la calidad de servicio:
- Permite adaptar la codificación a la calidad

Monitorización control de congestión
Identificación de la fuente (por si hay fallos o reinicios)
Sincronización entre flujos
    Indicación de tiempo real y su correspondiente timestamp
Información de control escalable
    Estos paquetes se envían de forma periódica a todos los nodos
    Mantienen datos para calcular estadísticas (rtt, jitter, …)
    Cambian de frecuencia para usar un máximo de 5% de todo el tráfico de la sesión (RTP)

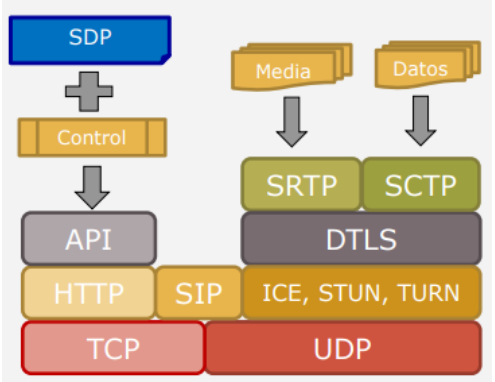
Protocolo de descripción de sesión (SDP)
Es un lenguaje que permite describir los recursos multimedia, temporización y características de conexión de una reunión:
    Líneas del tipo: <letra> = <configuración>
Ejemplo:
v=0
o=RySD 2890844526 2890844526 IN IP4 10.120.42.3
s=Meeting
c=IN IP4 10.120.42.3
…
m=audio 49170 RTP/AVP 0 8 97
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 51372 RTP/AVP 31 32
a=rtpmap:31 H261/90000

Sesión llamada "Meeting"
Creada por RySD
Los recursos están en 10.120.42.3
Dos flujos multimedia (video y audio)
- Audio con 3 posibles codificaciones y se envía por el puerto 49170/RTP
- Video con 1 sola posible codificación y se envía por el puerto 51372/RTP

Protocolo de streaming tiempo real (RTSP)
Este protocolo se utiliza para intercambiar comandos de control de la transmisión de datos de tiempo real con el servidor.
No se encarga de transmitir los datos, para eso se utiliza RTP/UDP, UDP o TCP.
Es el equivalente al "mando a distancia": PLAY, PAUSE, TEARDOWN.
Los mensajes intercambiados son de texto plano y similar a HTTP: Usa cabeceras HTTP

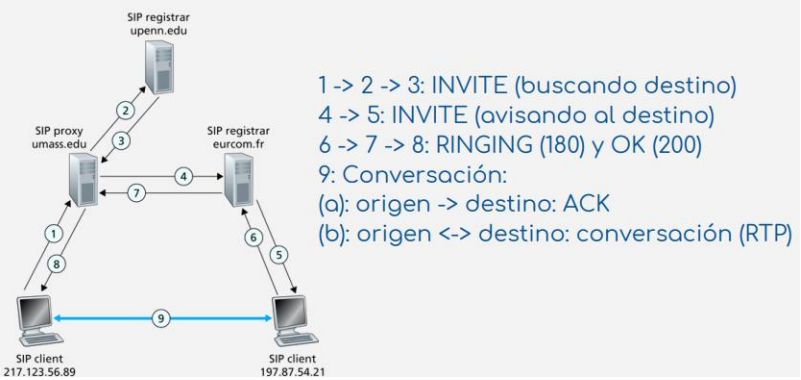


Caso Estudio 3: Discord, Meets, Teams …
El recurso multimedia se va generando a la vez que se consume. Es multidireccional (videoconferencia).
Características deseables:
    Localización de los extremos
    Comunicaciones P2P y multicast
    Primar la velocidad a la corrección
    Utilizable desde el navegador
    Transmisión (opcional) de datos fiable
    Seguridad
WebRTC:
    Base conocida: UDP, SRTP/SRTCP, SDP
    Localización de usuarios: SIP
    Usuarios como "servidor": ICE, STUN, TURN
    Más seguridad: DTLS
    Datos fiables: SCTP
    Uso web: HTTP + API Javascript

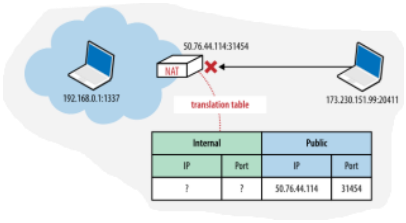


Protocolo de Inicio de la Sesión (SIP)

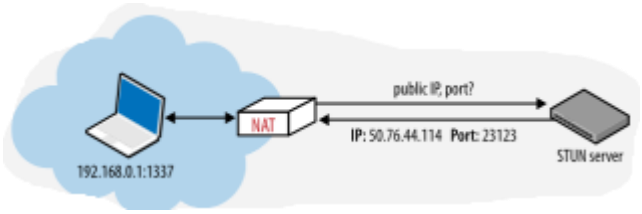
Este protocolo se utiliza para establecer conexiones multimedia entre dos usuarios (muy usado en VoIP)
No se encarga de transmitir los datos, para eso RTP/UDP, UDP o TCP
Modo texto y usa SDP para dar información sobre las características de la conexión
Mensajes: INVITE, ACK, BYE, OPTIONS, CANCEL, REGISTER
Permite localizar la IP de un usuario gracias a un sistema de registro



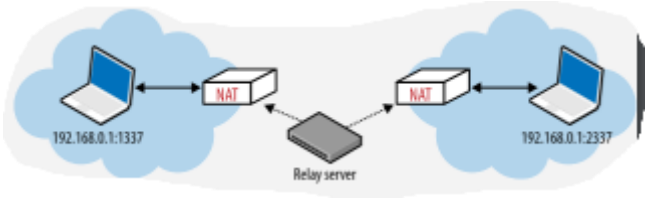
Funcionando como servidor temporal
El uso de NAT dificulta actuar de servidor
En estos sistemas queremos actuar de servidor pero temporal (no podemos configurar de forma permanente)



STUN: Nos conectamos a un servidor externo (STUN)
Permite que conozcamos la IP/puerto público
Se añade la entrada a la tabla NAT



TURN: Dependiendo de la configuración puede no ser posible: Comunicación con nodo intermedio
ICE: Permite elegir entre STUN y TURN (lo más efectivo)



Otros protocolos: DTLS y SCTP

DTLS:
Implementación de TLS sobre datagramas UDP
Ofrece encriptación, integridad y autenticación
En SRTP solo se usa al inicio para compartir información de configuración

SCTP:
Alternativa a TCP y UDP pero poco usado debido a problemas de la configuración con equipos
Usarlo sobre UDP soluciona esos problemas
Características:
- Envía mensajes (como UDP)
- Orientado a la conexión, confiable, con control de flujo y de congestión y secuenciación (como TCP)
- Selección y monitorización de múltiples flujos (como HTTP/2)
- Mensajes fuera de orden y protección contra ataques

Compartición de contenidos en redes P2P

Las redes P2P (peer-to-peer) son redes que no tienen clientes ni servidores fijos, sino un conjunto de nodos que se comportan simultáneamente como clientes y servidores respecto a los demás nodos de la red

Permiten compartir y transferir archivos (legales) de forma eficiente

-La conectividad entre nodos de la red permite mejorar el rendimiento en la conexiones y transferencias frente a redes centralizadas

-La utilidad más demandada en la compartición de archivos pero también es muy utilizada para la telefonía VoIP y streaming en general

Cualquier nodo puede iniciar, detener o completar una transacción

Los archivos se almacenan de forma distribuida entre los nodos de la red

Filosofía P2P: El que más comparta más privilegios tiene

Compartición de Archivos – Ejemplo

Alicia ejecuta su cliente P2P en su portátil

Tiene una conexión intermitente a Internet; en cada conexión obtiene una dirección IP diferente

Solicita “Hey Jude”

La aplicación muestra otros nodos “ peers ” que disponen de una copia de “Hey Jude”.

Alicia elige uno de los peers, Bob.

El fichero se transfiere y copia desde el PC de Bob al portátil de Alicia (HTTP)

Mientras Alicia descarga, otros usuarios se descargan desde el portátil de Alicia

El nodo de Alicia es tanto un cliente, como un servidor transitorio

P2P: Localización de contenidos

Una vez que se ha elegido el nodo, la descarga es directa.

Dada la conectividad intermitente de los nodos, la dificultad está en localizar y buscar contenidos (en los nodos).

Tres acercamientos:

Centralizada: Servidor de directorio centralizado

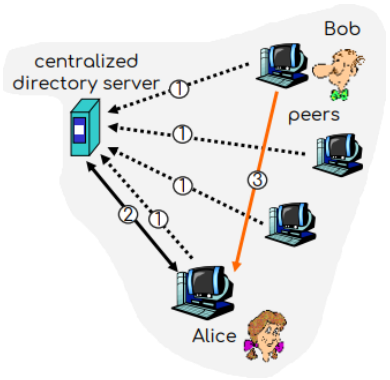
Distribuida: Búsqueda por inundación

Híbrida: Tiene nodos especiales que forma una red distribuida

-Localización de contenidos centralizada

Hay un servidor dedicado centralizado a funciones de directorio:

1. Cuando un nodo se conecta, informa al servidor central
   - Dirección IP
   - Contenido
2. Alice pregunta por “Hey Jude” y recibe los peers
3. Alice le pide el fichero a Bob La transferencia es descentralizada, pero la localización de contenido es centralizada:



Cuello de botella Único punto de fallo

- Localización de contenidos distribuida

Totalmente distribuida: No hay servidor central

Los nodos forman un grafo -> overlay network

Un arco entre los nodos X e Y indica que entre ellos existe una conexión TCP

La red overlay está formada por todos los nodos activos y los arcos entre ellos

Un arco no es un enlace físico

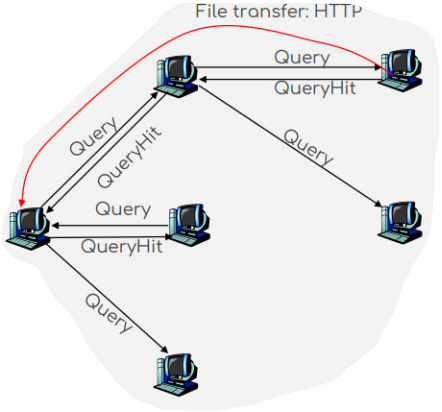
Típicamente un nodo estará conectado con <10 vecinos overlay

La localización de contenidos se realiza mediante inundación de consultas en la red overlay

- protocolo “ query ” - “ query hit” entre los nodos.

1. El mensaje es enviado a todos los nodos vecinos a través de la conexión TCP
2. Los nodos reenvían el mensaje de localización
3. Cuando el recurso se localiza en un nodo, se envía un mensaje QueryHit a través del mismo camino en dirección contraria

Escalabidad: Inundación de ámbito limitado



¿Cómo unirse a la red?

Para unirse a la red, el nodo X debe encontrar otro nodo perteneciente a la red:

1. Usar una lista de candidatos (¿fuerza bruta?)
2. Intentar de forma secuencial la conexión con al menos uno (Y)
3. X envía un mensaje de Ping Y; Y reenvía el mensaje de Ping a sus vecinos
4. Todos los nodos que reciben el mensaje de Ping responden con un mensaje de Pong
5. X puede establecer conexiones adicionales.

- Localización de contenidos híbrida

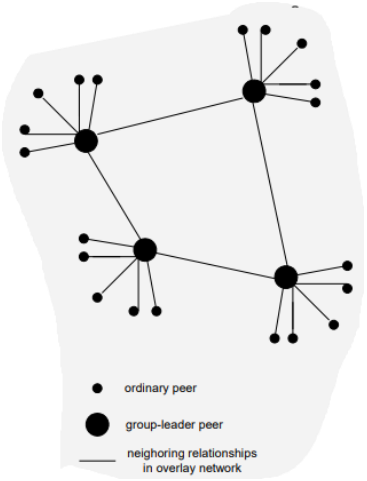
Cada nodo es un líder de grupo o es asignado a un líder de grupo:

- Conexión nodo <-> su líder
- Conexiones entre líderes

Un líder mantiene información del contenido de su grupo

Búsqueda:

- Se consulta al líder
- El líder facilita los datos de los que tienen el recurso en el grupo
- Reenvía petición a otros líderes que ofrecen también su información
- El cliente selecciona un peer



BitTorrent: Funcionamiento/terminología

Protocolo para el intercambio de archivos usando P2P

Localización de nodos

A través de un servidor (tracker) que mantiene una lista de nodos (peer) que contienen el recurso conectados a la red (swarm o enjambre)

La información para localizar un tracker viene en un fichero .torrent

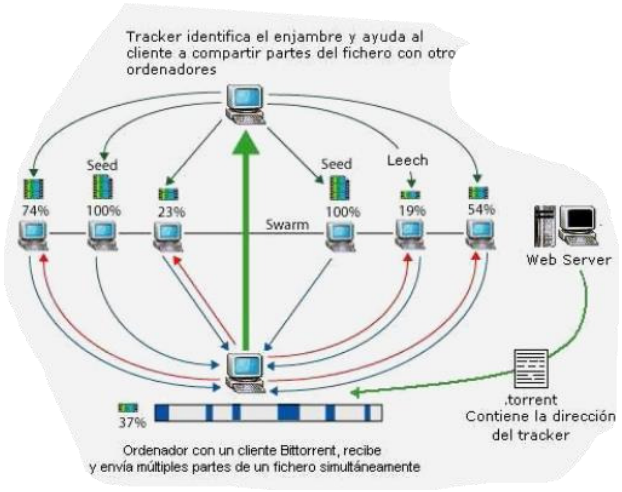
Divide los ficheros en trozos (chunks)

Los nodos ofrecen las partes que tienen al resto y solicitan las que les falten

Tipos de nodos (peers):

- Tiene el fichero completo (seed)
- Les faltan trozos (leech)

Se forma una red P2P por archivo



No existe buscador, pero si páginas “especializadas”

Se obtiene el fichero .torrent del archivo

Ahí viene información del tracker que tiene que nodos tienen el recurso

Solicitamos a los nodos las partes que nos falten

Facilitamos al resto de nodos las que tenemos

Al completarlo nos convertimos en “Seed”