UT 12.
HERRAMIENTAS Y PROTOCOLOS DE RED

Sistemas informáticos
CFGS DAW

Aarón Martín
Bermejo
a.martinbermejo@edu.gva.es
2022/2023
Versión:230313.1617

Licencia

Nomenclatura

A lo largo de esta unidad se utilizarán diferentes símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

ÍNDICE

UT 12. HERRAMIENTAS Y PROTOCOLOS DE RED

1. I NTRODUCCIÓN

Como ya hemos estudiado antes, las redes informáticas son una parte enorme de los pilares de nuestra actual sociedad, altamente tecnologizada. Comunicaciones, finanzas, trabajo a distancia... casi todos los aspectos de nuestra vida cotidiana están conectados a alguno de ellos.

Por eso hemos estudiado las características de las redes, las clasificaciones, las arquitecturas... En este tema vamos a profundizar en cómo las redes de ordenadores han modelado la comunicación para hacerla eficiente, segura y fiable en condiciones extremas como alta disponibilidad, gran estrés, infraestructuras cambiantes o que fallan, etc. Esos modelos de las comunicaciones se denominan **protocolos** en redes informáticas.

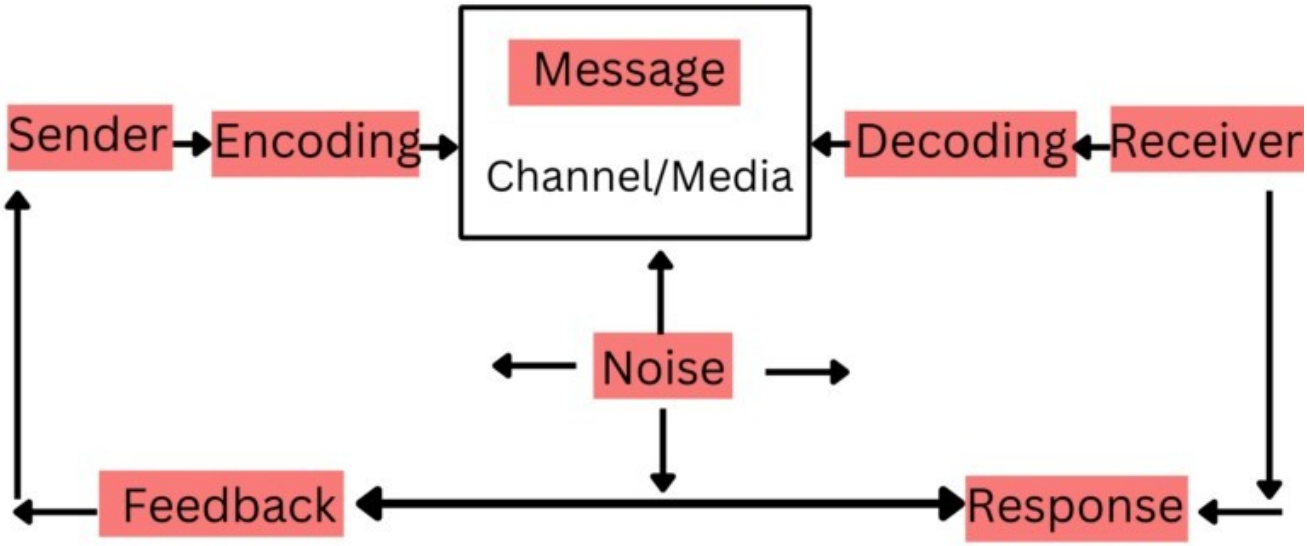
Además, cubriremos algunas herramientas con el fin de construir un conjunto de herramientas para trabajar con esos protocolos y bajo redes informáticas.

2. P ROTOCOLS

En las redes informáticas, una definición de protocolo puede ser un conjunto de reglas que especifican cómo deben formatearse los mensajes y cómo deben enviarse, recibirse y procesarse. Por lo tanto, tenemos dos partes de la definición del protocolo:

1. Formato de los mensajes
2. Cómo tratar los datos

El objetivo de un protocolo de red informática es modelar la comunicación entre ordenadores. La comunicación se basa siempre en los siguientes elementos:



1. *Elementos de comunicación: https://clearinfo.in/blog/elements-of-communication-process/*

En la figura podemos ver estos elementos básicos:

1. Emisor: el que envía la información
2. Receptor: el que recibe la información
3. Mensaje: la información que se transmite

Estos son los tres elementos básicos de la comunicación y, obviamente, sin ellos no habrá comunicación. Pero hay más elementos en la figura que son extremadamente importantes en la comunicación, pero especialmente en los ordenadores en red:

- Canal/Medio: a través de dónde o qué se transmite el mensaje. Ejemplos de canales son el aire, un cable, el papel...
- Codificación/Descodificación: cómo se codifica el mensaje y cómo se descodifica. Ejemplos de codificaciones pueden ser:
  - Cualquier lenguaje humano: español, inglés, lenguaje de signos, expresión corporal...
  - Lenguajes" de máquina: binario, hexadecimal, ASCII...

Es cualquier código que permite al emisor representar la información codificándola con algún sistema y al receptor entenderla decodificando la información mediante el mismo sistema que el emisor o algún otro sistema.

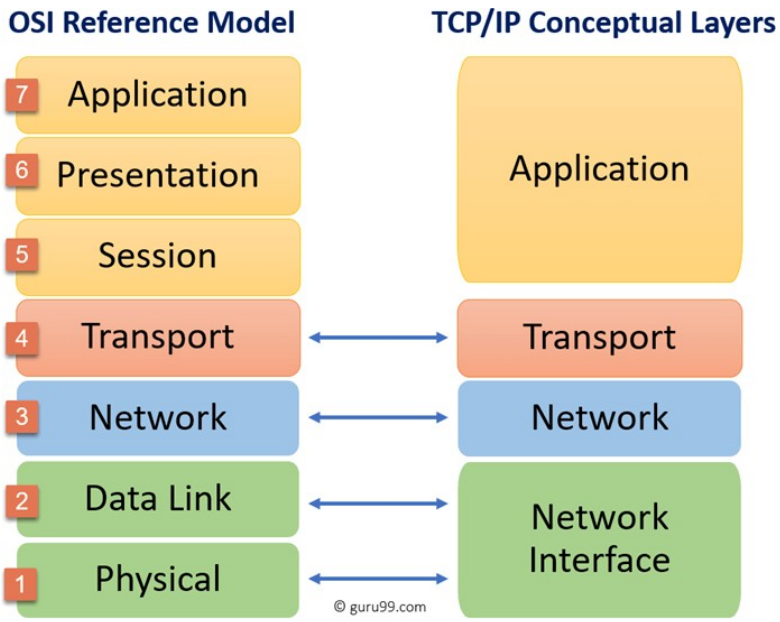
- Respuesta/Feedback: es la información que el receptor envía de vuelta al emisor. Puede ser una respuesta al mensaje originalmente enviado "hola - adiós" o puede ser cualquier otro tipo de respuesta o retroalimentación como el acuse de recibo del mensaje.
  mensaje.
- Ruido: el gran enemigo de la comunicación. El ruido se refiere a cualquier interferencia que afecte a la comunicación causando problemas en ella. Puede tratarse de ruido físico que no permite que dos personas se oigan o de interferencias de radio,
  saturación de las radiofrecuencias Wifi, etc.

Estos elementos específicos son importantes en cualquier comunicación, pero son de extrema importancia en la computación en red, ya que deben resolverse de forma que los ordenadores puedan procesarlos eficazmente.

Los protocolos de las redes informáticas tratarán de especificar cómo deben abordar los ordenadores estos elementos de comunicación para poder enviar información.

Hay que estandarizar esos protocolos para que los ordenadores puedan enviar y recibir información de forma que se entienda. Exactamente igual que en el lenguaje humano, si cada persona tuviera su propio lenguaje nadie podría entender nada.

Eso es lo que OSI y TCP/IP intentan resolver. Recordar qué eran OSI y TCP/IP y qué niveles definen:



Diferentes protocolos intervienen en las distintas capas de ambos modelos para resolver los problemas específicos que requieren. En esta unidad vamos a centrarnos en

3. DNS

Domain Name System o DNS es el sistema que permite traducir de nombres de dominio[1] a IPs. Mientras que los ordenadores se comunican entre sí utilizando IPs como forma de dirigirse los unos a los otros, para los humanos recordar cada número IP de cada sitio web o servicio web que queremos utilizar sería insoportable. Ésa es la razón de ser del DNS.

> 📢 ¿Crees que accederás tanto a aules si necesitaras recordar que su IP es https://213.0.87.122/ED

Para pedir algo utilizando un nombre de dominio y poder saber a qué IP apunta ese dominio, es necesario realizar algunos pasos para cumplir con la búsqueda DNS o resolución DNS:

1. En primer lugar, su ordenador buscará en el **archivo hosts**. El archivo hosts es un archivo de texto plano que contiene información sobre cómo los nombres de host se asignan a las IPs en su ordenador. En Linux el archivo hosts se almacena en /etc/hosts y en windows dentro de c:\Windows\System32\Drivers\etc\hosts. Normalmente contiene la redirección a localhost:

```
127.0.0.1   localhost
127.0.1.1   aamarber-MS-7996
```

   Se puede editar para contener más hosts y es súper útil especialmente para desarrollar, pero también para configurar manualmente la resolución de nombres de dominio.

2. Si el dominio no se encuentra en el archivo hosts, se comprobará la **caché del navegador** por si ese dominio se resolvió anteriormente y se puede reutilizar esa información.

3. Entonces, si no se encuentra ni en el archivo hosts ni en la caché del navegador, se llama al DNS Resolver para resolver la IP del host. El resolver DNS buscará en su caché por si hubiera una petición anterior para resolver ese nombre de dominio específico. Si no es así, redirigirá al **servidor de nombres raíz** correspondiente en función del nombre de dominio**.**

4. **Los servidores de nombres raíz están en la** parte superior de la jerarquía DNS, pero no son los que tienen la información sobre cómo traducir un dominio a una IP. En su lugar, tienen la información sobre dónde buscarla en el servidor **de nombres de TLD.**

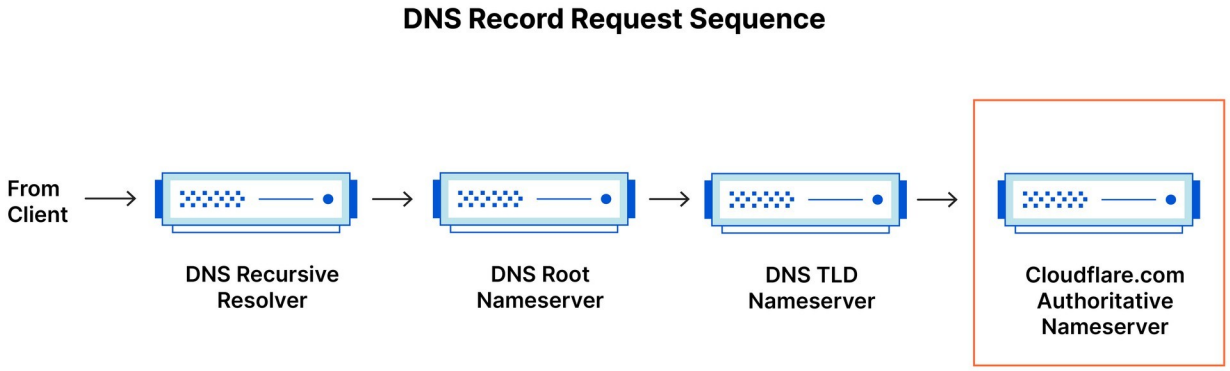
5. **Los servidores de nombres TLD** se encargan de gestionar la información de los **dominios de primer nivel (TLD)** como .com, .net, .org… y contienen la información del dominio de primer nivel específico **Servidor de nombres**

autorizado

6. El **servidor de nombres autoritativo** es el último paso en la búsqueda DNS y son los servidores que tienen toda la información sobre un dominio, incluida su IP. Será

1 https://developer.mozilla.org/en-US/docs/Glossary/Domain_name

el servidor que responderá a la búsqueda DNS en el paso final si ningún otro paso pudo hacerlo.



Cloudflare: whatis DNS[2]

Aunque parezca bastante complejo, ese proceso se realiza millones y miles de millones de veces al día con extrema rapidez, y es uno de los pilares de la web moderna.

Pero una cosa importante relacionada con este "complejo" proceso es que, como tiene esos pasos y cada uno necesita hablar con diferentes servidores con sus propias cachés, cambiar la IP de un nombre de host puede ser un proceso que lleve hasta 24 horas.

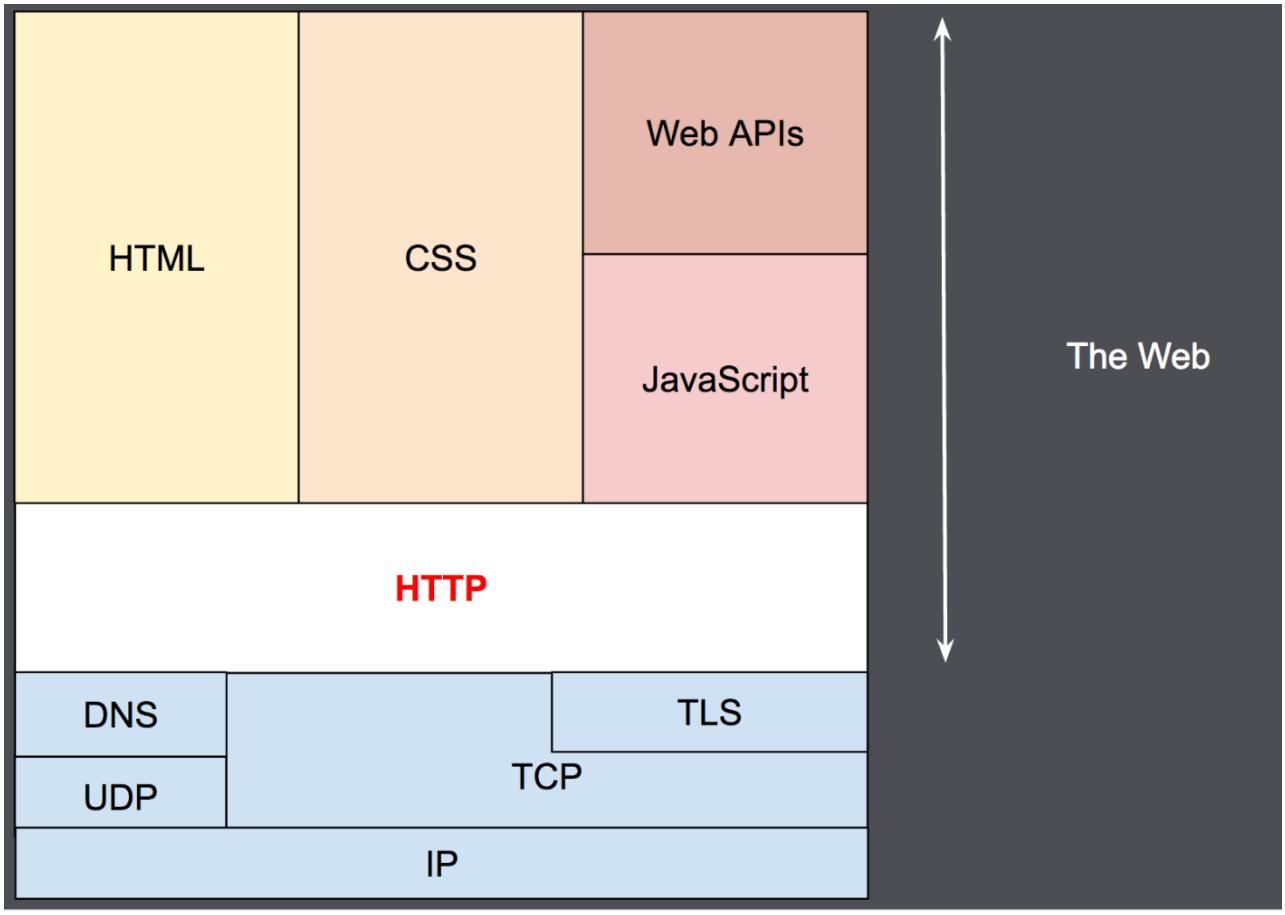
> ⚡ Recuerda ese retraso cuando necesites cambiar el servidor de nombres autoritativo de tu dominio, por ejemplo

2 https://www.cloudflare.com/learning/dns/what-is-dns/

4. H TTP

HTTP es un protocolo que pertenece a la capa de aplicación del modelo TCP/IP y está pensado para obtener recursos a través de la red. Es un protocolo cliente servidor, lo que significa que el cliente es el que inicia la comunicación y el servidor le responde. Es básico para el intercambio web de datos.

En HTTP se intercambian mensajes individuales. Los mensajes del cliente se llaman "peticiones" y los del servidor "respuestas", por razones obvias.



https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

HTTP se apoya en diferentes protocolos que están por debajo de él, tanto de la misma capa como de diferentes capas obviamente. Especialmente importante es TCP y su versión encriptada TLS, ya que la comunicación HTTP se realiza normalmente a través de TCP. Puede hacerse a través de UDP, pero no es el protocolo de comunicación por defecto a utilizar para enviar datos HTTP.

HTTP ha evolucionado desde su creación en 1989-1991 por Tim Berners Lee hasta nuestros días. La primera versión estándar fue HTTP/1.1 en 1997 y durante 15 años fue el estándar con algunas extensiones. Esa cantidad de tiempo sin ser modificado radicalmente significa que HTTP era extremadamente estable y bien definido y que

puede responder a la

necesidades de la web desde hace mucho tiempo. En mayo de 2015, HTTP/2 se estandarizó con algunas características importantes para adaptarse al uso actual de la web y, en enero de 2022, ya lo utilizaba el 46,9% de los sitios web.[3] HTTP/3 apareció en junio de 2022 con la característica principal de que utiliza QUIC (otro protocolo de transporte) en lugar de TCP.

4.1 Mensajes HTTP

Como hemos hablado al hablar de comunicación, una parte importante de ella es la codificación y decodificación de los mensajes. Por tanto, HTTP especifica esa codificación de los mensajes.

En HTTP, hay dos tipos de mensajes: **peticiones** y **respuestas**.

4.1.1 SOLICITA

El formato de las solicitudes es el siguiente:

1. **Línea de inicio**: Los mensajes HTTP son enviados por el cliente para iniciar una acción en el servidor. Contienen tres elementos

   1.1. **Método HTTP**: verbo o sustantivo que describe la acción a realizar. GET, POST, PUT, DELETE, OPTIONS son ejemplos de métodos HTTP válidos.

   1.2. **Destino de la solicitud**: normalmente, la URL, pero también puede ser una IP (y/o un puerto) o cualquier otra forma de dirigirse a un servidor.

   1.3. **Versión HTTP**: tanto para especificar la estructura del resto del mensaje como la versión HTTP esperada de la respuesta.

   Algunos ejemplos de la línea de salida podrían ser:

   GET www.google.es HTTP/1.1

   POST /registro_usuario HTTP/2.0

   GET 127.0.0.1:8081/index.html HTTP/1.1

2. **Cabeceras: el** formato de las cabeceras es siempre una cadena que no distingue mayúsculas de minúsculas seguida de un doble punto. Existen muchas, pero pueden clasificarse en dos grupos generales:

   2.1. Cabeceras de la petición: especifican la petición como indicando cuál es el origen de la petición (host: localhost), el user-agent o la codificación que el cliente aceptará como respuesta (accept: text/html).

   2.2. Cabeceras de representación: describen el mensaje, por ejemplo la codificación aplicada o la longitud de su contenido.

3 https://w3techs.com/technologies/details/ce-http2

3. **Cuerpo:** opcional, ya que algunos métodos de petición no necesitan cuerpo (como GET). Son los datos que se envían al servidor si es necesario, como los datos para POST o PUT.

4.1.2 RESPUESTAS

El formato de las respuestas es el siguiente

1. **Línea de estado**: contiene la primera línea de la respuesta HTTP:
   1.1. Versión del protocolo, como en la solicitud.
   1.2. **Código de estado:** indica el éxito o fracaso de la solicitud y normalmente el porqué, especificando información al respecto. Los códigos más comunes son:
      200: éxito
      404: no encontrado
      401: no autorizado
      Todos los códigos de estado HTTP se especifican en la norma RFC 911 0[45] pero a veces se utilizan códigos de estado personalizados.
2. Cabeceras: las cabeceras http de las respuestas siguen el mismo formato que en las peticiones, una cadena sin distinción entre mayúsculas y minúsculas seguida de un doble punto ":" y el valor de la cabecera. Al igual que en las solicitudes, las cabeceras de respuesta HTTP pueden clasificarse en dos categorías:
   1.1. Cabeceras de respuesta: cabeceras como "servidor" o "edad".
   1.2. Cabeceras de representación: al igual que en la solicitud, describen el mensaje, por ejemplo la codificación aplicada o la longitud de su contenido.
   2.2. **Cuerpo**: algunas respuestas no contienen cuerpo, porque la respuesta sólo requiere un código de estado. Pero en la mayoría de los casos el cuerpo de la respuesta se rellena con la información solicitada en la petición.

4.1.3 Mensajes HTTP/1.x vs HTTP/2

Los mensajes HTTP bajo HTTP/1.x se escriben en texto plano (sin ningún tipo de codificación) y no se dividen de ninguna manera. Eso significa que los mensajes HTTP/1.x bot solicitudes y respuestas se envían íntegramente en texto plano, lo que tiene la ventaja de que se pueden leer fácilmente, pero presentan grandes desventajas:

- No se pueden dividir fácilmente para poder multiplexarlos, ya que están escritos en texto plano.

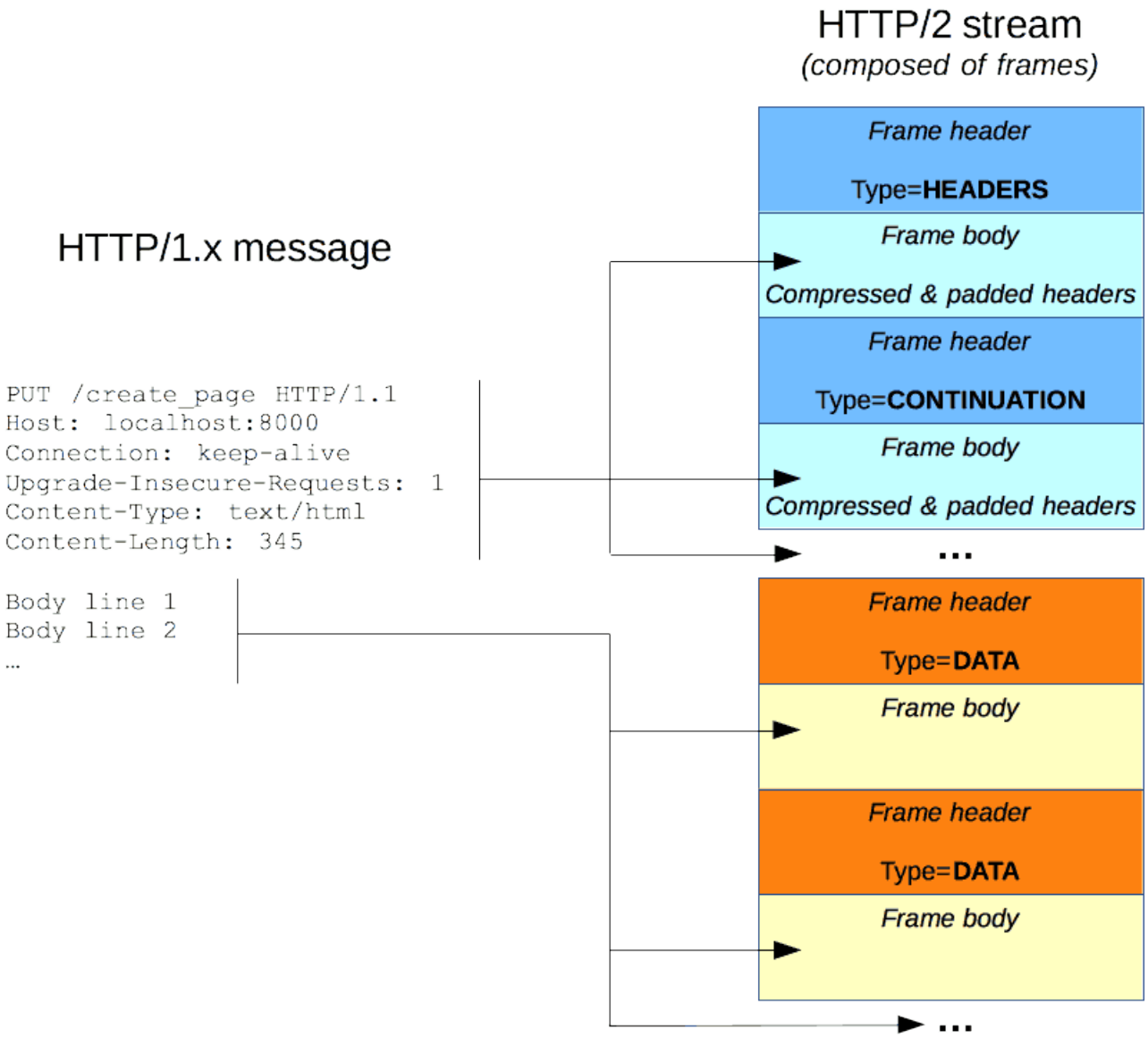
- No se puede comprimir fácilmente o, al menos, no de la forma más eficaz.

Estas dos desventajas llevan a algunos cambios en la codificación de los mensajes en HTTP/2:

4 https://httpwg.org/specs/rfc9110.html#overview.of.status.codes
5 https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

- Los mensajes HTTP/2 se envían en binario, en lugar de texto sin formato
- Los mensajes HTTP/2 se dividen en tramas y se envían en un flujo

Ambos cambios tienen por objeto mejorar el rendimiento comprimiendo y multiplexando el intercambio de información.



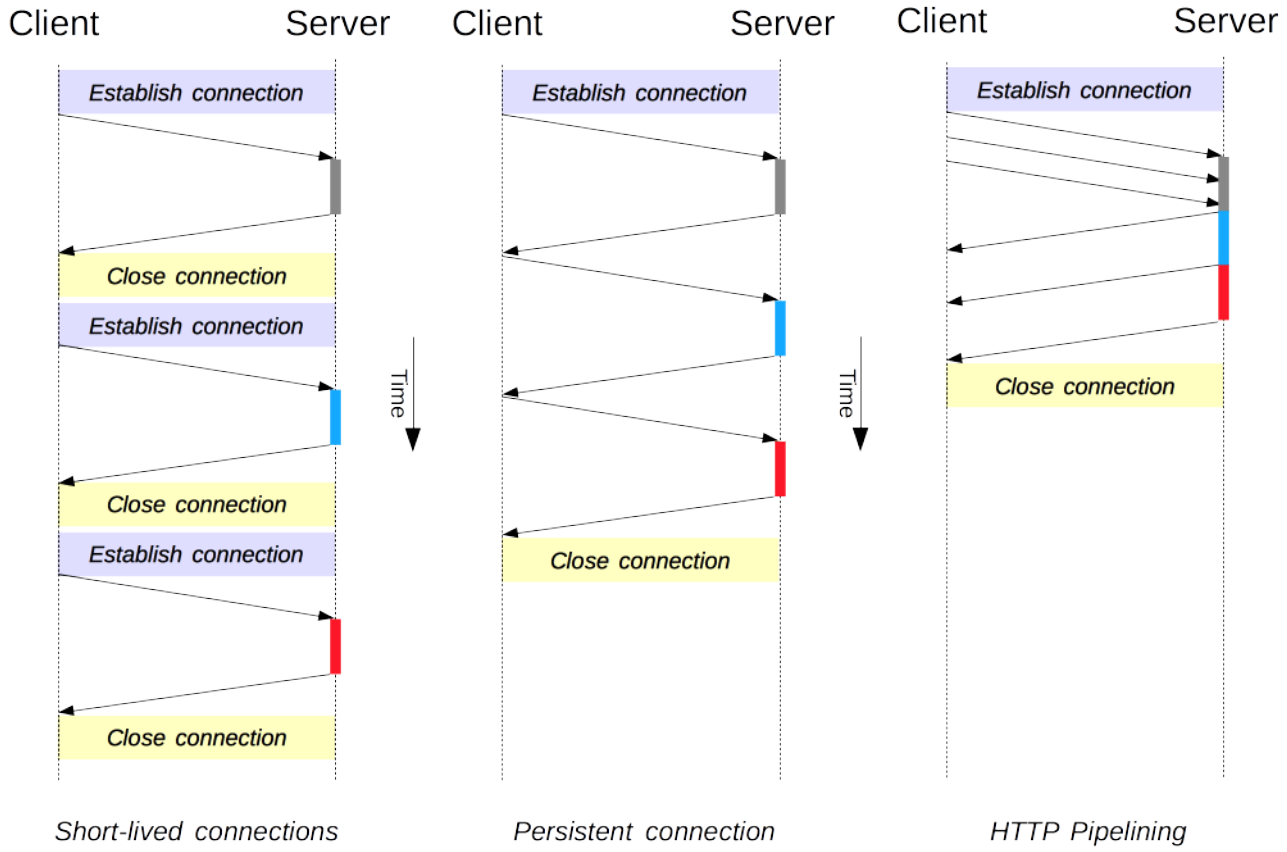
MDN: Mensajes HTTP/1.x frente a mensajes HTTP/2

4.2 Flujo HTTP

El flujo de comunicación en HTTP suele tener tres pasos simplificados:

1. El cliente establece la conexión con el servidor
2. El cliente envía la solicitud
3. El servidor gestiona la solicitud y envía una respuesta

Esos tres pasos básicos que parecen tan sencillos necesitan resolver problemas complejos debido a la gran demanda del mundo de las peticiones web. Especialmente, la gestión de la conexión es un problema, ya que establecer una conexión TCP consume recursos. Por eso existen diferentes estrategias para hacerlo:



MDN Gestión de la conexión s[6]

- Conexiones cortas: significa que cada vez que hay que hacer una petición, se crea una nueva conexión y, por tanto, hay que realizar el establecimiento de la conexión y su cierre.
- Conexiones persistentes: en lugar de establecer una nueva conexión para cada petición, las peticiones se hacen persistentes para evitar el establecimiento y cierre de las mismas. De ese modo, cuando sea necesario realizar varias peticiones pueden saltarse esos pasos.
- HTTP pipelining: en lugar de esperar a que se realice cada petición, todas se envían a través de la misma conexión y el servidor las irá respondiendo a medida que pueda.

HTTP/1.x y HTTP/2 tienen algunas diferencias en ese tema de la gestión de conexiones, optimizando ese tema al permitir que se hagan más peticiones al mismo tiempo o utilizando multiplexación en lugar de pipelining como algoritmo a gestionar.

6 https://developer.mozilla.org/en-US/docs/Web/HTTP/Connection_management_in_HTTP_1.x

4.3 HTTPS, TLS y SSL

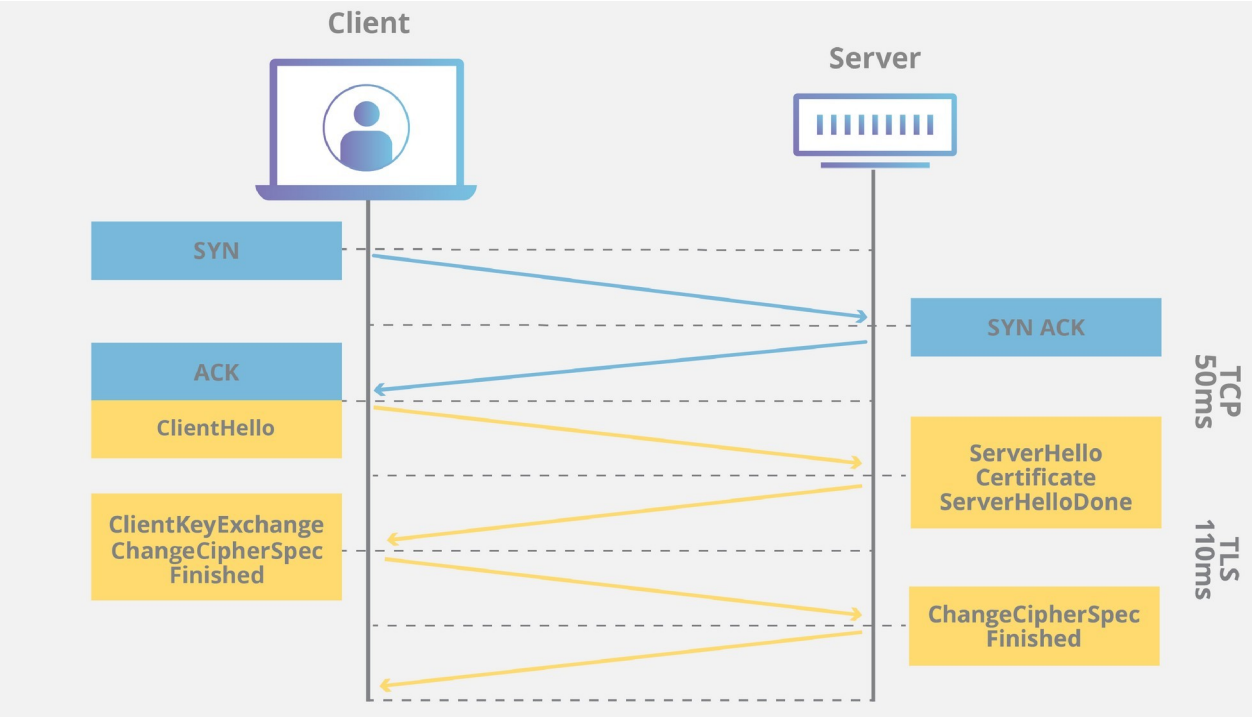
Como en casi todos los aspectos de la informática, la seguridad es un aspecto importante. Pero para la comunicación a través de redes, dado que la información enviada puede ser bastante sensible y esa información viaja a través de toneladas de ordenadores diferentes desconocidos que pueden interceptar esa información, darle una capa de seguridad es casi obligatorio.

Por eso, Netscape desarrolló SSL o Secure Sockets Layer, un protocolo de seguridad en Internet basado en la encriptación. Su objetivo es resolver problemas como garantizar la privacidad, la autenticación y el cifrado de la información enviada a través de redes, especialmente en HTTP. De hecho, lo que HTTPS significa es Hypertext Transfer Protocol **Secure** y significa que el sitio web implementa SSL/TLS.

Espere... ¿TLS? TLS o Transport Layer Security es un protocolo desarrollado por el IETF (Internet Engineering Task Force) como evolución de SSL. Y como SSL fue desarrollado por una empresa privada (Netscape) que no participó en el desarrollo del nuevo estándar, le cambiaron el nombre. Pero como TLS se basó en SSL, ambos suelen confundirse.

TLS funciona de la siguiente manera:

1. Primero se inicia un proceso de autenticación entre el cliente y el servidor. Se conoce como **handshake TLS** y funciona como especifica la siguiente figura:



TLS handshak e[7]

2. Después, como ambas partes tienen las claves para cifrar la información, toda la comunicación se cifrará entre el cliente y el servidor,

7 https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/

3. Además, todos los datos se firman para verificar que, aunque estén cifrados, mantienen su integridad y no han sido contaminados.

Para que un sitio web pueda implementar TLS necesita un certificado de seguridad. Un certificado tiene dos aspectos importantes:

1. Es como una tarjeta de identificación que garantiza que el servidor con el que estamos hablando es el que indica.
2. Contiene la **clave pública** y la **privada**.
    2.1. La clave pública está expuesta a cualquier cliente, para que pueda cifrar la información enviada al servidor y nadie pueda espiarla.
    2.2. El servidor mantiene en secreto la clave privada, que le permite descifrar la información enviada por el cliente.

También existen tres tipos de certificados de seguridad:

1. Dominio único: sólo puede utilizarse para un dominio específico.
2. Wildcard: está vinculado a un único dominio, pero puede utilizarse para subdominios como "aules.edu.gva.es" "portal.edu.gva.es", etc.
3. Multidominio: emitido para varios dominios

La mayoría de los navegadores modernos marcan los sitios web sin HTTPS como inseguros y normalmente se posicionan mucho peor en los motores de búsqueda si no se utiliza la seguridad. Así que es una obligación si creas o administras un sitio web moderno.

RECURSOS

1. https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview
2. https://www.cloudflare.com/learning/dns/what-is-dns/