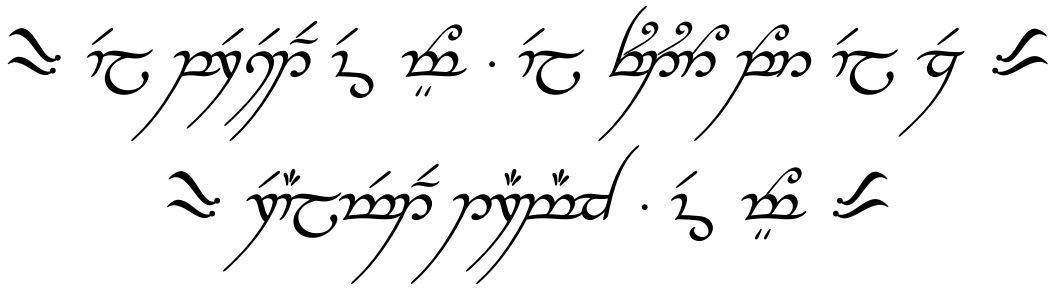
Escuela Técnica Superior de Ingeniería Informática

Asignatura:
Redes de Computadores

Autor:
Fernando José Mateos Gómez

Ultima Modificacion: **12 de Octubre del 2021**



Indice

1. Tema 1: Redes de Computadores e Internet

1.1. Internet

Internet comprende al conjunto de *Hardware* y *Software* que abarca a la red. Está formado por enlaces de conexión que alcanzan a los hosts, o sistemas terminales, dispositivos que se conectan a la Internet y ejecutan aplicaciones en red. La podemos denominar como la "red de redes", una gran red que abarca al resto de redes que se enlazan entre si, por lo que tiene dos propiedades fundamentales:

- Es pública y accesible por todo el mundo.
- Es poco jerárquica, al estar todo entrelazado no hay una red que se encuentre por encima de otra en la estructura.

Las redes se conectan mediante enlaces, hay millones de dispositivos, que pueden transportan la información mediante un medio físico o no(Fibra, Cobre, Satélite, etc), a una velocidad a la denominaremos *tasa de transmisión*, o conocida también como ancho de banda, y usarán los *routers* como medio para enviar paquetes, bloques de datos.
La red está regida por protocolos y estándares(*RFC*, "Request For Comments", *IETF*, "Internet Engineering Task Force"), con el fin de ofrecer una serie de *servicios*, **servicios de comunicación** para enviar datos a través de la red a un destinatario o **aplicaciones distribuidas**, aquellas consumidas por los hosts.

1.2. Protocolos

Para controlar el tráfico en la red, disponemos de una serie de algoritmos que se ejecutan en Software, estos son los denominados "protocolos". Estos definen el formato y orden de los mensajes en el tráfico entre las entidades de la red, y las medidas a tomar a corde a las situaciones. Podemos destacar el protocolo **TCP**, **IP** o el **Ethernet**.

1.3. Equipos, Redes de Acceso, Medios Físicos

Debemos de dividir este problema en 3 partes:

- **Frontera de la Red**: Aplicaciones.
- **Redes de Acceso**: Cableados y Medios de Acceso.
- **Núcleo de la Red**: Routers e Internet.

1.3.1. Frontera de la Red

Los Hosts ejecutan programas en la "Frontera" a través de un modelo, como por ejemplo el ***Cliente/Servidor***, en el cual el cliente solicita al servidor, es decir, le realiza una petición y este le responde, o el modelo ***Peer to Peer***, que no usa servidores y todos los equipos en la red reciben la respuesta y la petición (no se usa demasiado).

1.3.2. Redes de Acceso

Con el fin de conectar un host a un router, en la "frontera", deberemos usar redes internas para acceder a este dispositivo. Son las redes domésticas, institucionales y móviles las que aprovechan esto, por lo que hay que protegerlas (decidir si es compartida o privada) y disponer de un medio físico de transmisión de la propia red.

Los medios físicos transportan información en forma de bits, a través de un enlace físico, estos pueden ser:

- **No Guiados**: La señal se transporta por medio de ondas electromagnéticas, que pueden ser orientadas fijas o programadas para ser unidireccionales u omnidireccionales. Tendremos que tener en cuenta, los problemas que afecten a las ondas a la hora de trabajar con esta clas de conectores(interferencias, reflexión,...)

    - **Guiados**: La señal se transporta a través de un medio sólido (par trenzado, coaxial, fibra óptica, etc)
        - *Par Trenzado*: Formado por dos cables de cobre aislados, y cuya categoría indica la velocidad de transporte.
        - *Cable Coaxial*: Es bidireccional ya que en función del tipo puede ser de "banda base", unidireccional, o de "banda ancha",multidireccional.
        - *Fibra Óptica*: Aprovecha la energía lumínica y la velocidad de la luz para transmitir información. No requiere de repetidores que amplien la señal, a no ser que sean distancias muy largas
    - *Microondas*: Tiene alta direccionalidad.
    - *WLAN*: Es omnidireccional
    - *Satélite*: Tiene un retraso de $270ms$ pero como no nos conectamos directamente, no llega a ser un problema, en cortas distancias.

A la vez podemos dividir a las redes en dos categorías:

- **Acceso Fijo**: Pueden ser cableadas o usando cableado telefónico, como **ADSL**.
- **Acceso Movil**.

De entre los tipos de redes podemos destacar:

- Modems.
- DSL, "Digital Subscriber Line".
- HFC, "Hybrid Fiber Cable".
- FTTH, "Fiber To The Home"

1.3.3. Componentes típicos en una Red de Acceso

Como un caso particular, en las redes domésticas, la conexión a internet se realiza a través de un modem que se conecta al ISP, proveedor de servicios de internet, y este transmitirá una señal a un router, con una ip pública única, a la que se conectarán el resto de dispositivos de la red a través de una conexión NAT.

1.4. Conmutación de Paquetes/Circuitos y la Arquitectura de Internet

El núcleo de la red se conforma por una red entrelazada, una malla, de routers conectados entre si que transmiten datos a través de la red mediante:

- **Conmutación de circuitos**: Circuito dedicado, con la red telefónica. Usan el protocolo "Peer to Peer", y se comunican a través de llamadas de la red telefónica. Los recursos no se comparten y tiene un buen rendimiento, debido a que cada usuario se le reserva un ancho de banda, puede ser una división en función de:

    - La Frecuencia(**FDM**): Del ancho de banda total, a cada uno se le asigna una frecuencia en específica, e invariable.

- El Tiempo(**TDM**): Del ancho de banda total, cada usuario tiene un tiempo en el que puede realizar sus peticiones, es invariable.

- **Conmutación de paquetes**: Los datos se envian en forma de paquetes. Los paquetes de los usuarios se comparten en la red, cada paquete con un ancho de banda preestablecido. Además los recursos que se usan son los mínimos e indispensables, si hay más recursos que demanda a satisfacer se genera una *congestión*, que tendrá que esperar en una cola de paquetes, que se liberará con el tráfico gradual de cada paquete de uno en uno, *store and forward*, o si los paquetes de varios usuarios no tienen un patrón temporal fijo, el ancho de banda se compartirá, *multiplexación estadística*.

  En el caso del "store and forward", considerando L como el tamaño de los paquetes, en *bit* y R el tiempo que tarda cada enlace, en *bps*, podemos calcular el retardo relacionando $\frac{L}{R}$ como una relación inversamente proporcional.

  Este es el tipo de conmutación que aprovecha la Internet.

En definitiva podemos concluir que el uso de una ''conmutación de paquetes'' permite a más usuarios usar la red.

1.4.1. Estructura del internet

Internet tiene un punto de acceso, "NAP" (una red de alta velocidad como el *Ethernet*), que se ramificará en suminstradores de la red, "NSP", situado en la capa 1, y almacenará información referente a links generales que requieren el resto de proveedores, se conectan entre si también.

Bajando de nivel, a la capa 2, nos encontramos con los distrubidores de servicio regional, "RSP", que le proporcionará el servicio final a la capa más cercana a los hosts, nivel 3, los "ISP", que intercambiarán con la "NAP" rutas y puntos de accesos, que se encargará de recordar.

1.5. Rendimiento

1.5.1. Retardos

Los paquetes se encolan en el buffer del router y mediremos la tasa de llegada de paquetes respecto a la salida, "apilandolos". Es decir, mientras haya *buffer libre*, los paquetes se encolan, y evidentemente generan un cierto retardo, pero si está el *buffer completo*, se genera una pérdida y el flujo de paquete se detiene hasta que se libere el buffer.

Para calcular el retardo total podemos considerar la siguiente expresión:

$$\boxed{d_{nodal} = d_{proc} + d_{cola} + d_{trans} + d_{prop}}$$

Los describiremos a continuación:

- $\mathbf{d_{nodal}}$: Se refiere al retardo total por cada nodo.

- $\mathbf{d_{proc}}$: Es el *procesamiento nodal*, se encarga de comprobar errores a nivel de bit y determina el enlace de salida. Se mide en μs.

- $\mathbf{d_{cola}}$: Es el *retardo de cola*, indica el tiempo de espera antes de ser transmitido por el enlace de salida, y su valor varía en función de como de completo esté el buffer. Se mide en μs o en ms. Su expresión es la relación directamente proporcional entre la tasa media de paquetes de llegada, a medido en "paquetes por segundo", y el tamaño

del paquete, L medido en bits. Además de una relación inversamente proporcional entre este producto y el ancho de banda del enlace R, *bps*:

$$\boxed{d_{cola} = a\frac{L}{R} = a \;\; d_{trans}}$$

Tiene 3 casuísticas a analizar:

- Si $d_{cola} \sim 0$, el retardo de cola es pequeño.
- Si $d_{cola} = 1$, el retardo de cola es muy grande.
- Si $d_{cola} > 1$, entonces llegan más paquetes de lo que se puede liberar, por lo que el retardo sería infinito.

- **d_{trans}**: Es el *retardo de transmisión*, indica una relación inversamente proporcional entre la longitud del paquete L, en bits, y el ancho de banda del enlace R, *bps*. Se mide en μs o en ms, y su expresión es:

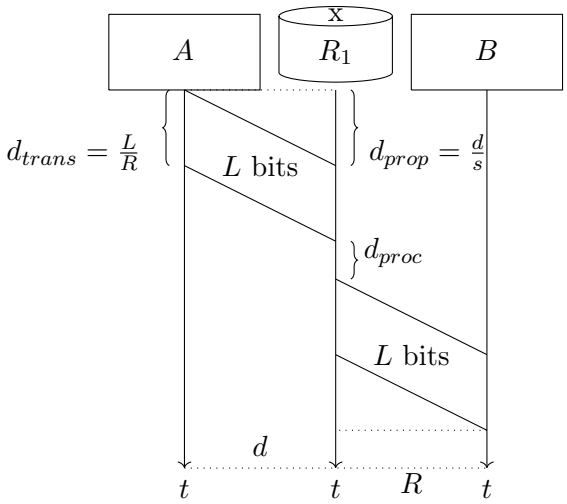
$$\boxed{d_{trans} = \frac{L}{R}}$$

Es decir, expresa el tiempo que se tarda en enviar algo en base a los recursos del sistema y de la red.

- **d_{prop}**: Es el *retardo de propagación*, indica una relación entre la longitud del medio físico, d medido en m, por el que viaja el paquete y la velocidad de propagación que tenga el medio, s que idealmente medirá aproximadamente 2x10^8 m/s. Su expresión es la siguiente:

$$\boxed{d_{prop} = \frac{d}{s}}$$

Es decir, expresa el tiempo que se tarda en enviar algo en base al medio y la distancia entre los emisores y receptores.



1.5.2. Pérdidas

El buffer está asociada a un enlace con una capacidad finita, por lo que es común que mientras el buffer se completa o llegue a su máxima capacidad, haya pérdida de paquetes, "drops", que serán descartados y enviados a un nodo anexo con el fin de intentar enviarlos nuevamente.

Se expresa en "bits / segundo" y la podemos medir de dos formas:

- Instantaneamente, medimos los paquetes en un tiempo determinado.
- Promedio, medimos la tasa media de paquetes en un periodo de tiempo.

En la teoría la tasa de transferencia entre varios terminales es:

$$\boxed{min(R_C, R_S, \frac{R}{N})}$$

Siendo R_C el canal con el cliente y R_S el canal con el servidor, por lo que mientras ambos tengan una relación proporcional, el flujo de datos que pase por cada uno de los canales será el mismo, y no se producirán muchos cuellos de botella. Además de $\frac{R}{N}$) ser el cociente entre el ancho de banda y el numero de conexiones.
Sin embargo en la realidad R_C o R_S se vuelven un cuello de botella, ya que la tasa de transferencia depende del tráfico existente, a pesar de la velocidad de transmisión.

1.6. Capas de Protocolos y Modelos de Servicio

Con el fin de organizar una red, para mayor rehusabilidad y mantenimiento, debemos de crear una arquitectura de red modulada que permita un tráfico eficiente.

En cualquier Modelo, se utiliza la misma lógica, posteriormente veremos los dos modelos más usados.
Crearemos tantas capas como servicios querramos implementar, cada una es independiente, sin embargo aprovechará la información proveniente de la capa inmediatamente anterior a esta (independientemente de como esté viajando la información).
En los extremos de cada capa existen "entidades", que recibirán la información proporcionada por las funciones generadas por cada capa, denominados "servicios". El acceso a estos servicios se realiza a través de la interfaz *SAP*, "Service Access Point".
En cada nivel se usa un determinado protocolo, para comunicarse con las "entidades" de su mismo nivel y enviar la información a la capa siguiente. Además por cada protocolo indicamos las reglas de intercambio de mensajes y su formato, *PDU*, "Protocol Data Unit".

1.6.1. Organización

Ya sabemos que el mensaje entre cada capa se denomina "PDU", esta información solo podrá leerse por aquellas capas situadas al mismo nivel, se denominan "N-PDU", pero en la realidad la información pasará a través de "SAP" tales que comunicarán una capa con otra, encapsulando la información hasta llegar a otra capa, donde irán desencapsulandolas.

1.6.2. Proceso de Encapsulación y Fragmentación

1. La información se encapsula, "PDU".
2. Usando un "ICI", *Interface Control Information*, de la capa siguiente, es decir, usamos "N-ICI", completamos la información del "(N+1)-PDU" para especificar el mensaje. Esta agrupación la pasamos por el "N-SAP" para la siguiente capa, usando un "N-IDU".
3. La nueva capa, la denominaremos "N", extraerá la información de "N-ICI" del "SAP" y responderá acorde a esta. A su vez, esta información nos permitirá formar un "PCI" para este nivel actual "N-PCI", *Protocol Control Information*.
4. Hemos separado la información, del "SAP", en un "N-PCI" y el resto en varios "N-SDU", *Service Data Unit*, a este proceso de este nivel, lo llamamos **fragmentación**

5. Juntamos el "N-PCI" y los "N-SDU" y formamos nuevamente un "PDU", pero para esta capa.

6. El proceso se repetirá hasta que no queden más capas.

1.6.3. Proceso de Desencapsulación y Reensamblaje

Es el mismo que el anterior, pero a la inversa, en vez de ir de un nivel alto a uno inferior, vamos de una capa de bajo nivel a una de alto.

1.6.4. Tipos de Arquitectura

Depende del conjunto de funciones que querramos implementar, será necesario. Para eso se han creado distintas arquitecturas, aquí veremos dos:

- **TCP/IP**: Es la que utiliza Internet y describe funciones, servicios y protocolos.
- **Modelo de referencia OSI**: Es la que se usa en base al estandar "ISO" pero solo describe funciones y servicios.

1.6.5. TCP/IP

Se divide en 5 capas:

- **Aplicación**: Soporta las aplicaciones de red. Sirve de interfaz con el usuario final. Envia "A-PDU", mensajes.
- **Transporte** Transferencia de datos extremo a extremo entre procesos. Envia "T-PDU", segmentos.
- **Red**: Direccionamiento y enrutado de datagramas de origen a destino. Envia "R-PDU", datagramas.
- **Enlace**: Transferencia de datos entre elementos de red "ceranos". Envia "E-PDU", tramas
- **Física**: Transferencia de bits.

1.6.6. OSI

Se divide en las mismas capas que el "TCP/IP" pero añadiendo dos capas más entre la capa de *aplicación* y *transporte*:

- **Presentación**: Permite la interpretación de los datos y los codifica.
- **Sesión**: Se encarga de la sincronización, restauración y checkeo de datos.

La capa de **Aplicación**, se encargaría de recopilar y hacer lo que hacen estas dos capas extras, pero internet también realiza estas funciones.

1.6.7. Implementación

Las capas las podemos dividir en nivel *Software* y *Hardware*, ya que a medida que vamos bajando de nivel, la complejidad aumenta.

- **Software**:
    - *Programas*: Aplicación

    - *Sistema Operativo*: Transporte y Red
- **Hardware**:
- Enlace y Física

2. Tema 2: Capa de Aplicación

2.1. Creación de Aplicaciones en Red

Crear una aplicación en red no es más que desarrollar un programa que se ejecutan en distintos dispositivos, sistemas finales.
Estos programas se comunican a través de la red, como los navegadores.

No debemos de ahcer programas para el núcleo de la red, ya que su finalidad es encaminar los paquetes de la forma más rápida y eficiente posible, por lo que no necesitamos ningún programa extra.

2.2. Arquitecturas

2.2.1. Arquitectura Cliente-Servidor

Tenemos evidentemente dos componentes:

- **Servidor**: Siempre está encendido y posee una IP fija, un identificador único, ya que si varía nunca sabríamos la dirección del servidor. Agruparlos creamos *granjas de servidores*
- **Cliente**: Se conectan con el *servidor*. Los clientes no se comunican entre ellos directamente.

2.2.2. Arquitectura Peer To Peer

No están siempre encendidos, y sus sistemas finales se comunican entre si de forma aleatoria, debido a esto sus ip cambian cada vez que se comunican entre ellos.
Son escalables, pero no de muy facil mantenimiento.

2.2.3. Arquitectura Híbrida

Tenemos un servidor centralizado que nos permite conocer la IP del usuario de forma inmediata, conectandose a una conexión "peer to peer".

2.3. Implementación

Aprovechará los servicios de la capa de *Transporte*.

2.3.1. Definicion

Esta capa define:

- *Mensaje a intercambiar.*
- *Sintaxis del mensaje*: Número de Campos, espacios, etc...
- *Semántica del mensaje*: Qué significa cada campo.
- *Reglas sobre como responder y recibir los mensajes*, los ICP.

2.3.2. Comunicación entre procesos

Los procesos son programas que se ejecutan en un dispositivo e implementa un protocolo. Hay dos tipos de comunicación:

- **Mensajes PDU**: Los equipos se comparten la información usando mensajes PDU.
- **Comunicación entre procesos**: Hay dos o más procesos que se ejecutan paralelamente en un equipo, los proporcionan el SO.

Y dos tipos de procesos:

- **Cliente**: Proceso que inicia la comunicación.
- **Servidor**: Proceso que espera a ser contactado.

2.4. Sockets

Los sockets son los puntos de acceso a través de los cuales se accede al servicio, para que el nivel de aplicación pueda enviar mensajes al nivel de transporte, es decir, es un "SAP".

Cada protocolo de aplicación se identifica con un número de puerto, que es usado para identificar el proceso cliente y servidor.
Es decir, un socket queda identificado por una dirección IP y un número de puerto.

2.5. LocalHost

Localhost, con una dirección IP particular, 127,0,0,1, aunque puede variar, permite identificar al dispositivo final del sistema, el que esté usando una persona en ese momento.
Puede probar aplicaciones en red, para un dispositivo final, sin necesidad de estar conectado a la red, y además de comunicarse con procesos dentro de ese equipo sistema final, usando sockets.

2.6. Perdida de Datos

Dependiendo de la aplicación, podemos tolerar un porcentaje de perdida de información, un *tiempo de retardo* corto para ser efectiva o una *tasa de transferencia* con una pérdida minima.
Evidentemente debemos de mantener una seguridad en el envio del mensaje, lo que puede provocar, si se viola, una perdida de la información.

2.7. Servicios

2.7.1. TCP

- **Orientado a Conexión**: Requiere de un acuerdo entre los procesos cliente y servidor antes de iniciar la transferencia.
- **Transporte Fiable**: Solo entre procesos *emisor* y *receptor*.
- **Control de Flujo**: El *emisor* no satura al *receptor*
- **Control de Congestión**: Se evita la creción de cuellos de botella, controlando un uso equitativo del *ancho de banda*.
- No **provee** de sincronización, ancho de banda o seguridad.

2.7.2. UDP

- **Transporte Ligero**: No está orientado a la conexión y no es seguro entre procesos entre *emisor* y *receptor*
- No **provee** de un acuerdo previo entre procesos, seguridad, sincronización o control del flujo. Se usa para telefonía movil.

2.8. DNS

Es un protocolo de la capa de *aplicación* que se encarga de generar identificadores que se encargen de diferenciar un sitio de otro, usando una *base de datos distribuida*, la cual permitirá el acceso rápido a través de una jerarquía, en función de la frecuencia de búsqueda de estas direcciones, es decir, usa alias para relacionar las direcciones de las páginas con las búsquedas.

2.8.1. Funcionamiento

- Enviamos una "DNS-PDU" al servidor DNS, la cual la reciribirá si hay un receptor, a través de un canal no protegido, UDP.
- El servidor enviará una respuesta al cliente de vuelta, si no hubiera un receptor anteriormente, se enviará tantas veces como sea necesario

Dentro de sus funcionalidades encontramos, la traducción de IP a alias o la asociación de un alias o varios a una IP. Por desgracia no podemos hacerla centralizada, la base de datos DNS, ya que no permitiría su escabilidad y daría problemas en caso de caida.
Hemos dicho que usa UDP como medio de transporte, normalmente usando el puerto 53.

2.8.2. Memoria Caché

Al igual que un ordenador, los servidores DNS, guardan en memoria de acceso rápido direcciones y alias que se usen con frecuencia, de forma que no es necesario buscarlas nuevamente, ya que se encuentran en un acceso fácil. En caso de no encontrarla, mandan la petición a otros servidores por encima de este, hasta que alguno lo encuentre, o no haya más servidores, en cuyo caso, se realizará la búsqueda inmediatamente.

2.9. HTTP y Web

2.9.1. Páginas Web

Las páginas web están formadas por una fichero HTML, que contiene objetos referenciados a los elementos que componen el fichero HTML, tales como imágenes, archivos JS, etc...
Cuando enviemos al servidor una petición para recoger una página, recopilaremos tantos objetos como referencias haya en el HTML, más una más por el propio fichero HTML.
Aqui, un ejemplo de un fichero HTML:

```
<!DOCTYPE html>
<html>
        <head>
        <!-- Este es un comentario en la cabecera-->
                <title>Hello World<\title>
        <\head>
        <body>
        <!-- Este es un comentario en el cuerpo-->
                <h1>Hello World but in boddy<\h1>
```

```
                <img src="hola.jpg"\>
        <\body>
<\html>
```

2.9.2. Protocolo HTTP

Protocolo, *HyperText Transfer Protocol* a nivel de aplicación que al contrario que DNS, usa TCP para mantener un canal seguro. Usa el puerto 80 para comunicarse con el servidor, y usa la arquitectura **Cliente-Servidor** para responder a las consultas de los que accedan a sus recursos.

- **Cliente**:Navegador que solicita objetos a un servidor Web.
- **Servidor**: Dispositivo que envia objetos pedidos por el cliente.

Los servidores no guardan información de peticiones anteriores, son *sin extado* y podemos dividir las conexiones en dos tipos:

- *No persistente*: El cliente solicitará una petición al servidor, este aceptará la petición en caso de encontrarse, tras un tiempo, tras encontrar el servidor, se enviará un **mensaje de petición** por el cliente, con la petición de obtener el objeto HTML principal, y lo intentará recibir. Tras esto realizará una petición por un objeto del HTML, se le denomina **mensajes de respuesta** a aquellos mensajes que contengan el resto de objetos de la página. Finalmente se cerrará. Estos pasos se repetirán mientras aún no haya objetos transmitidos.

  $$\boxed{TR = 2RTT + TTP}$$

  Siendo RTT como *el tiempo de ida y vuelta* en una petición al servidor, el tiempo que se tarda en enviar y recibir un mensaje de petición, y TTO el tiempo de *transmisión de Bytes por objeto.*

  Podemos observar que este sistema es más lento, no permite paralelización y cargará uno a uno todos los elementos del HTML.

- *Persistente*: El servidor mantendrá abierta la conexión tras enviar la respuesta de haber encontrado el objeto HTML, y el tiempo total será la suma de RTT por cada objeto más lo que tarde en enviarse cada objeto, se le denominará a esto *solicitud de descarga del objeto.*

  $$\boxed{\sum_{i=0}^{n} RTT_n + K}$$

Los mensajes HTTP pueden ser:

- **Peticiones**: Enviadas por el cliente, formadas por un texto ASCII, y transporta HTTP-PCI para solicitar los HTTP-UD. Pueden ser GET, POST, HEAD, PUT o DELETE. HTTP-PCI está formado por la cabecera, y el HTTP-UD por la url y el cuerpo del mensaje.
- **Respuestas**: Enviada por el servidor, y transporta los HTTP-UD. HTTP-PCI está formado por la cabecera y HTTP-UD por el mensaje.

2.10. Cookies

No son más que elementos que guardan el estado del servidor, de forma que un servidor es capaz de identificar un usuario incluso si accede desde otros dispositivos.

2.11. Servidor Proxy

Los Proxy tienen la finalidad de satisfacer la petición del cliente sin involucrar al servidor web original, ya que cuenta con memoria caché que almacena diversas páginas webs.
Se configura del lado del cliente, de forma que todas las peticiones HTTP son enviadas a través de este proxy.
Si el proxy solicita una página web que está en caché, se devuelve el objeto al cliente, pero si se realiza al servidor, el proxy guarda en memoria esta página web. Sin embargo no solo el proxy tiene caché, pero el navegador también, por lo que podemos reducir enormemente el tiempo de respuesta con estas dos herramientas.

2.12. Socket

Una interfaz del equipo local, creada por una aplicación y controlada por el SO (una "puerta") por la que el proceso de aplicación puede tanto enviar como recibir mensajes de otros procesos o equipos remotos.

3. Tema 3: Capa de Transporte

4. Tema 4: Capa de Red

5. Tema 5: Capa de Enlace de Datos