**Redes y Comunicaciones - Apunte para parcial**

**Práctica 1**

Una red está formada por un conjunto de equipos (host) conectados por medios de cables, ondas, o cualquier otro método de transporte de datos. El principal objetivo es compartir información (archivos), recursos (discos, impresoras) y servicios (accesos a bases de datos, correo electrónico).

Internet es una red de redes, en la que participan desde grandes sistemas, hasta modelos personales.

Las primeras redes fueron de poco alcance, y su objetivo era comunicar a las computadoras de una empresa, o campus de universidad. Con el avance de internet, empezó a ser necesario interconectar estas redes. Esto fue un problema, porque no había un estándar sobre cómo conectar estas computadoras, ni qué reglas debían cumplir para formar parte de la red. Para esto, surgieron protocolos. También, por ejemplo, era un problema conectar computadoras con distintos sistemas operativos.

Un protocolo es un conjunto de reglas que especifica el intercambio de datos u órdenes durante la comunicación entre entidades que forman parte de una red. Permite la comunicación y están implementados en los componentes.

Muchos protocolos están definidos en los RFC; también muchos documentos RFC se usan como estándares aunque realmente no lo sean.

Para solucionar los problemas de compatibilidad también se pensó en un modelo en capas, siendo el mas conocido, el modelo OSI. La idea es que cada capa abstrae de tareas a las siguientes capas, y cada capa tiene una tarea bien definida. Además, se divide la complejidad en partes mas chicas y manejables; los cambios en una capa no afectan a las demás si la interfaz se mantiene.

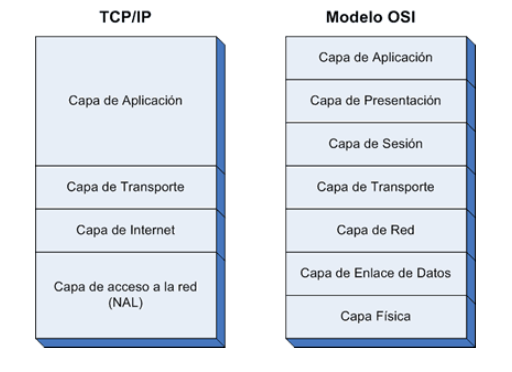
En general, la arquitectura cliente servidor es la mas usada.

Respecto a los canales de comunicación para dos computadoras, tenemos redes conmutadoras de paquetes, y redes conmutadoras de circuitos:

* Conmutación de circuitos: se establece un camino físico entre los medios de comunicación, previo a la conexión entre usuarios. Este camino está reservado solo para comunicación entre los dos extremos hasta que termine la comunicación. Ej: líneas telefónicas.
* Conmutación de paquetes: el emisor divide los mensajes a enviar en un número arbitrario de paquetes del mismo tamaño, donde adjunta una cabecera, dirección de origen, y dirección destino, así como datos de control que luego serán transmitidos por diferentes medios de conexión entre nodos temporales hasta llegar a su destino, donde los paquetes son reensamblados. Ej: Internet.

A las empresas proveedoras de internet se las conoce como ISP (Internet Service Provider). Distintas empresas usan distintos tipos de conexiones (por ejemplo fibra óptica o cable coaxial). Normalmente nos dan un modem, que se encarga de transformar las señales para que viajen las señales por el medio.

El modelo en capas implementado hoy en dia (TCP/IP) está basado en el modelo OSI, pero no es igual:

****

La idea es que cada capa va a agregar información adicional (información de cabecera) al PDU recibido de la capa superior, formando así el PDU correspondiente a la capa receptora, antes de que los datos sean enviados. En cada capa, un paquete tiene una cabecera, y un campo de carga útil (normalmente tiene el paquete de la capa superior)

Esto de “agregar información adicional” es conocido como “encapsulación”. La idea es que si una capa X del lado emisor agrega información, la misma capa X del lado receptor es la que se encarga de “desencapsular” esa información y usarla.

Nota: A partir de ahora se empieza a explicar la funcionalidad de cada capa.

**Práctica 2 - Capa de aplicación - HTTP.**

El PDU de esta capa es el “Dato”.

La capa de aplicación ofrece a las aplicaciones la posibilidad de acceder a los servicios de las demás capas, y define los protocolos que utilizan las aplicaciones para intercambiar datos. Normalmente, un usuario interactúa con una aplicación, que a su vez actúa con la capa de aplicación.

Los protocolos de la capa de aplicación vistos en la materia son:

HTTP: (normalmente usa el puerto 80) es un protocolo orientado a transacciones, y sigue el esquema de cliente - servidor (petición - respuesta). Al cliente que efectúa la petición se lo conoce como “agente de usuario”. La información transmitida es un “recurso”, identificada por URL o URI. Pone en contacto a la computadora con los servidores que brindan internet.

HTTPS: (normalmente usa el puerto 443) es un protocolo basado en HTTP, pero con el objetivo de ser una versión mas segura del mismo.

* Otros que no forman parte de esta práctica:

|  |  |
| --- | --- |
| Nombre | Puerto |
| TFTP | 69 |
| FTP | 20 y 21 |
| SMTP | 25 |
| DNS | 53 |
| SSH | 22 |
| POP3 | 110 |
| DHCP | Puerto origen 68 (cliente). Puerto destino 67 (servidor). |
| IMAP | 143 |

**A nivel de capa de aplicación, los paquetes solo cuentan con datos propios a la aplicación que está usando el cliente.**

Los agentes de usuario son aplicaciones clientes usadas por protocolos de red (normalmente los que acceden a WWW). Es una cadena de texto que es enviada al servidor, formando parte de la petición HTTP, y tiene información como por ejemplo el nombre de la aplicación, versión de la misma, sistema operativo y lenguaje.

Cuando le hacemos una petición al servidor por una página web, normalmente nos responde con un HTML. El HTML es procesado por nuestro navegador, y puede necesitar realizar mas peticiones al servidor en caso de encontrar etiquetas <link> y/o <img>. Haciendo todas las peticiones que sean necesarias, nuestro navegador debería mostrarnos, por ejemplo, una página que tenga contenido CSS, JS, etc, para los cuales tuvo que hacer mas peticiones.

Usando el comando curl (herramienta para transferir datos desde o hacia un servidor, usando algún protocolo de capa de aplicación), podemos ver los headers, los cuales nos brindan información como:

Cabeceras del request:

* User-agent
* Host
* Accept

Cabeceras del response:

* Date
* Server (por ejemplo, apache)
* Last-Modified (última modificación de la página). Sirve para no traer información de manera innecesaria, en caso de que ya tenga la versión mas reciente en mi caché. Ahorra recursos.
* Etag: también sirve para ahorrar recursos. Ayuda a validar la caché.
* Accept-Ranges
* Content-Length
* Content-Type.

También podemos ver códigos de respuesta, los cuales se clasifican en:

* 2XX: peticiones correctas.
* 3XX: redirecciones.
* 4XX: errores del cliente
* 5XX: errores del servidor.

Con el ejemplo del navegador que consulta una página HTML, vimos que es posible que un navegador tenga que hacer mas de una petición para traernos el resultado completo (la web con sus estilos demás cosas). Abrir la conexión y cerrarla por un solo recurso (HTTP VERSIÓN 1.0) tiene sus desventajas. Para solucionar eso, existe HTTP VERSIÓN 1.1, lo cual nos permite tener conexiones que queden abiertas por mas tiempo. Esto permite a un navegador, por ejemplo, hacer las peticiones a un servidor de un HTML, CSS y JS habiendo abierto la conexión solo una vez y por ende, ahorra recursos. También es importante destacar que esto puede ser inseguro, y que además, no siempre vamos a necesitar realizar mas de una petición y por ende, dejariamos la conexión abierta de forma innecesaria.

HTTP tiene distintos métodos para solicitar o enviar información. Destacamos GET y POST. Normalmente GET se usa para obtener recursos cuando nuestra solicitud no va a hacer modificaciones en el servidor, ya que los posibles parámetros viajan en la URL y están visibles. POST se usa cuando una petición sí puede provocar cambios en el servidor. En este caso, los parámetros viajan en el cuerpo de la petición.

**Práctica 3 - Capa de aplicación - DNS**

DNS es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como internet o una red privada. Su función mas importante es traducir nombres “recordables” en direcciones IP, para poder direccionar y localizar equipos mundialmente. Su segunda función mas importante es localizar servidores de correo electrónico.

DNS posee un gran número de servidores organizados de forma jerárquica y distribuidos alrededor del mundo. Ninguno de estos responde a todas las correspondencias de todos los host (están repartidas). Tenemos 3 tipos de servidores:

* DNS Raíz: hay 13 de este tipo (etiquetados de la A a la M).
* Servidores de dominio de nivel superior (TLD): responsables del nivel superior. Hay 3 tipos:
  + gTLD: propósitos particulares, como .com, .org, .net, .gov, etc.
  + ccTLD: correspondientes a países, como .ar, .cl, .es, etc.
  + ARPA TLD: dominio especial, usado por los protocolos para resolver direcciones a nombres.
  + Autoritativos: autorizado a responder la IP para los nombres de dominio que tiene alojado localmente.

También existe el servidor DNS local, el cual actúa como proxy reenviando las consultas a la jerarquía de servidores DS. Mantiene una caché con los nombres que resuelve, para que ante futuras solicitudes por ciertos nombres de DNS, no deba salir a la red a buscar la dirección IP en cuestión. También puede ser un servidor autoritativo.

Una consulta recursiva obliga al servidor DNS a responder una solicitud con un error o respuesta de éxito. Normalmente este tipo de consultas le hace el cliente a su sv DNS local, el cual se tiene que poner en contacto con otros servidores DNS de la jerarquía, para responder a la solicitud (normalmente arranca por un sv raíz).

A cada servidor le pregunta por toda la dirección, por ejemplo “[www.google.com](http://www.google.com)”, y cada servidor va a responder como mejor sabe, desglosando la dirección de derecha a izquierda. Estas consultas a otros servidores normalmente son iterativas, lo cual obliga a responder a un servidor lo mejor que puede (puede no responder nada útil). Un servidor puede darle una lista de opciones a nuestro servidor DNS local, y él elige con qué servidor se pone en contacto para seguir consultando.

Para la traducción se usa un módulo llamado “resolver”, el cual es un conjunto de rutinas encargado de resolver los nombres a solicitud del cliente. La idea sería que cada máquina tiene su propio módulo Resolver, y cada aplicación que quiera resolver un nombre de dominio, le pida al resolver que lo resuelva. El resolver es quien va a comenzar el proceso de resolución de nombre de dominio a IP. El servidor DNS local debe conocer al menos al servidor DNS raíz para empezar a buscar.

DNS usa distintos tipos de registros, entre los mas comunes están:

A y AAAA: dirección IP real de un dominio. AAAA es IPv6.

MX: lista de servidores de correo para un dominio.

PTR: opuesto al registro “A”. Resuelve una IP a un dominio.

SRV: “servicio”. Se lo usa para la definición de un servicio TCP en el que opera el dominio.

NS: “servidor de nombres”. Me indica los servidores autoritativos para un dominio.

En la práctica usamos el comando DIG, el cual es un comando para realizar consultas DNS.

De la salida de DIG, tenemos que tener en cuenta:

* El flag rd indica que la solicitud fue recursiva.
* El flag ra nos indica que el servidor puede responder recursivamente.
* El flag aa nos indica que la respuesta fue autoritativa (es decir, nos respondió un servidor DNS autoritativo).

Podemos ver números antes de la palabra IN. Estos números son el “TTL” (TIME TO LIVE), el cual le dice a los servidores locales cuanto tiempo se debería cachear el registro. Cuando hacemos una solicitud a un servidor no autoritativo, obtenemos ese número decrementandose. Si el request se lo hacemos a un servidor autoritativo, obtenemos el número seteado para el determinado registro.

También podemos ver que el orden de los servidores no es el mismo. Esto nos lleva a la conclusión de que se realiza un balance de carga, para no saturar a un determinado servidor, de todos los disponibles que puede tener.

Para los registros MX podemos ver que hay números al lado de nombres de dominio como el siguiente: 10 alt1.gmail-smtp-in.l.google.com. (esto es un servidor de mail de google). Este número hace referencia a la prioridad. A menor número, mayor prioridad. Las aplicaciones de mail van a intentar conectarse siempre al servidor de mail de mayor prioridad para dejarles nuestros mails, o recibirlos. Si no está disponible, la solicitud la toma otro servidor, el cual espera a que el de mayor prioridad esté disponible. Si se vence un determinado tiempo y el servidor de mayor prioridad no responde, entonces responde al cliente un mensaje de error.

Antes de la existencia de DNS se usaban archivos de texto plano para hacer las relaciones entre IP y nombres de dominio. Esto es imposible de mantener, ya que la cantidad de dominios de internet es muy grande, y además las IP suelen ir variando. Cuando esto se hizo insostenible fue que surgió DNS.

HTTP se basa en DNS, ya que las solicitudes HTTP son con nombres de dominio, y no con direcciones IP (los recursos son identificados con URI).

**Práctica 4 - Capa de aplicación - Correo electrónico**

Con el fin de enviar o consultar mails, un usuario hace uso de un agente de usuario para correo electrónico, que le permita comunicarse con su servidor de correo. Este agente de usuario conoce los siguientes protocolos:

SMTP: Normalmente se usa para enviar mensajes, sea desde el cliente a su servidor de correo, como también desde un servidor de correo a otro. Usa el puerto 25. Tiene limitaciones para la recepción de mails, y por ende, para recibir se usan POP3 o IMAP.

POP3: comienza cuando el agente abre una conexión hacia el servidor de correo en el puerto 110. Tiene 3 fases: autorización, transacción y actualización. Durante la fase de autorización, el agente de usuario envía el username y la contraseña. Durante la segunda fase, el agente de usuario recibe mensajes, o puede marcar correos para borrado. En la última fase actualizan los cambios. Esta última fase empieza luego de usar el comando “QUIT”.

Tiene 2 maneras de trabajar: descarga-borrado o descarga-almacenamiento. La mas normal es descarga-borrado. Por esto, se dice que POP3 consume menos recursos. Los mails son descargados al host del usuario y eliminados del servidor. Por ende, luego el cliente puede accederlos sin conexión a internet. Lo malo es que solo va a poder accederlos desde ese host.

IMAP: mas complejo que POP3. Cada mensaje se asocia con una carpeta (cuando un mensaje llega, normalmente va a INBOX). El usuario puede cambiar los mensajes de carpeta, leerlos, eliminarlos, etc. IMAP permite obtener solamente partes de mensajes, como solo el asunto por ejemplo, para ahorrar recursos.

IMAP mantiene los mensajes en el servidor. Esto es bueno cuando necesitamos conectarnos desde mas de un host. Lo malo es que siempre vamos a necesitar internet para conectarnos, y por ende, genera mayor gasto de recursos, ya que almacena los mails en estructuras configurables.

Aclaraciones generales:

Los protocolos de correo electrónico trabajan con el protocolo TCP de capa de transporte. Por ende, antes de comenzar a trabajar, se tiene que establecer la conexión y demás cosas que lleva a cabo TCP.

Si un usuario “Fran” (usuario de gmail) quiere mandar un mail a un usuario “Gas” (usuario de outlook), entonces Fran va a estar ejecutando una aplicación de correo electrónico que va a conocer a los servidores de mails de gmail. La aplicación va a hacer una solicitud DNS por el registro A del servidor de gmail de mayor prioridad, y enviar dicho mail. Luego, el servidor de gmail tiene que enviar el correo al servidor de outlook. Para esto, tiene que hacer una consulta DNS por el registro MX del dominio de outlook. Es esperable que como resultado tenga una lista de correos, de la cual va a elegir el de mayor prioridad, y va a hacer otra consulta DNS para saber la IP de dicho servidor. Una vez obtenida va a enviarle el mail. Luego, “Gas” va a conectarse a su servidor de correo usando POP3 o IMAP para poder leer su correo.

**Práctica 5 y 6 - Capa de transporte - Parte I y Parte II**

El PDU de esta capa es el segmento.

La capa de transporte es responsable de la transferencia de extremo a extremo en general de los datos de aplicación. Se busca lograr una comunicación lógica entre procesos, lo que quiere decir que es como si un los host que ejecutan los procesos estuvieran conectados directamente.

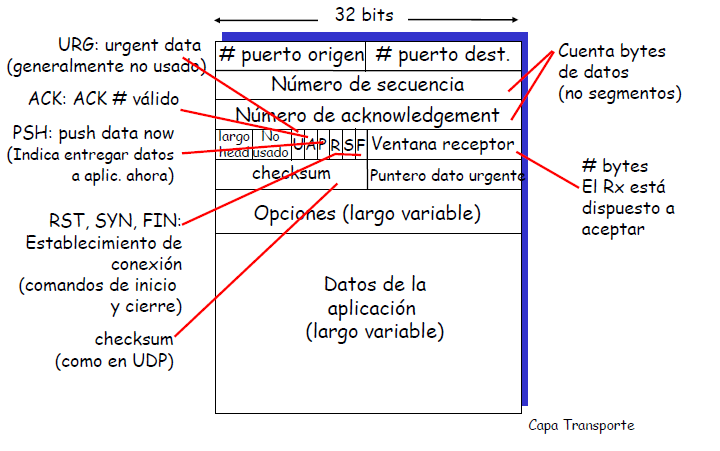
Esta capa permite segmentar los datos, y brinda el control necesario para reensamblar las partes.

Sus responsabilidades son:

* Seguimiento de la comunicación individual entre aplicaciones en los host origen y destino.
* Segmentación de datos y control de cada porción.
* Reensamble de segmentos en flujos de datos de aplicación.
* Identificación de las diferentes aplicaciones (los datos de las distintas aplicaciones que están ejecutándose en la misma computadora no deben mezclarse).

En la capa de transporte mayormente se usan TCP y UDP. A nivel de capa de transporte es que se agregan tanto el puerto origen como el puerto destino.

Cabecera TCP:



Cabecera UDP:

## 

Un proceso va a tener asignado uno o mas sockets (puertas por las que pasan los datos). En el host origen, se lleva a cabo el proceso de multiplexación, lo cual implica reunir todos los “fragmentos” de información de los distintos sockets asignados a un proceso en particular (en caso de que tenga mas de uno), y prepararlos para enviarlos. La aplicación que usa el usuario solamente se encarga de dejar los datos en el socket, y la capa de aplicación hace el resto.

Como una computadora puede ejecutar mas de un proceso, es lógico que además los paquetes se identifiquen por número de puertos, para no mezclar información enviada o recibida por distintos procesos.

En el receptor, se usa la información de puerto destino para decidir a qué socket entregar la información (demultiplexación).

Los números de puerto del 0 al 1023 son los “puertos bien conocidos”, y son empleados por los protocolos de aplicación bien conocidos. Estos son, normalmente, usados en el campo “puerto destino”, ya que se espera que un protocolo de capa de aplicación esté corriendo en dicho puerto.

Comparación entre TCP y UDP:

|  |  |  |
| --- | --- | --- |
|  | **TCP** | **UDP** |
| **Confiabilidad** | Es un servicio **fiable** Garantiza que los datos transmitidos por el proceso emisor sean entregados al proceso receptor, correctamente (Continuará reenviando un segmento hasta que la recepción del mismo haya sido confirmada por el destino) y en orden. | Es un servicio **no fiable** que no garantiza que los datos enviados por un proceso lleguen intactos (o que ni siquiera lleguen) al proceso de destino.  La comprobación de errores es responsabilidad de la aplicación. |
| **Multiplexación** | Multiplexación sin conexión. Dos segmentos TCP entrantes con direcciones IP de origen o números de puerto de origen diferentes (con la excepción de un segmento TCP que transporte la solicitud original de establecimiento de conexión) serán dirigidos a dos sockets distintos. | Multiplexación orientada a la conexión. Si dos segmentos UDP con diferentes sockets de origen (Direccion IP y/o Nro. Puerto) poseen la misma dirección IP destino y el mismo nro. de puerto de destino, entonces los dos segmentos se enviarán al mismo proceso de destino a través del mismo socket de destino. |
| **Orientado a la conexión** | Se dice que TCP está **orientado a la conexión** porque antes de que un proceso de la capa aplicación pueda comenzar a enviar datos a otro, los dos procesos deben primero “establecer una comunicación” entre ellos; es decir, tienen que enviarse ciertos segmentos preliminares para definir los parámetros de la transferencia de datos que van a llevar a cabo a continuación. Casi siempre es una conexión punto a punto, es decir, entre un único emisor y un único receptor. | Es un protocolo **sin conexión.** No tiene lugar una fase de establecimiento de la conexión entre las entidades de la capa de transporte emisora y receptora previa al envío del segmento. Por tanto, UDP no añade ningún retardo a causa del establecimiento de una conexión. |
| **Controles de congestión** | **Si**. Los mecanismos de control de congestión de TCP evitan que cualquier conexión TCP inunde con una cantidad de tráfico excesiva los enlaces y routers existentes entre los hosts que están comunicándose.  TCP se esfuerza en proporcionar a cada conexión que atraviesa un enlace congestionado la misma cuota de ancho de banda del enlace. Esto se consigue regulando la velocidad a la que los lados emisores de las conexiones TCP pueden enviar tráfico a la red. | El tráfico UDP **no** está regulado. Se puede transferir tan rápido como se desee. |
| Utilización de puertos | El socket está definido para cada extremo de la conexión. Es decir, por los cuatro campos (IPOrigen, PuertoOrigen, IPDestino, PuertoDestino).  Esto quiere decir que por cada socket pueden intercambiar datos únicamente dos procesos (conexión punto a punto) | El socket está definido por el número de puerto y la dirección IP del destino.  Muchos clientes pueden enviar datos por un mismo socket.  En particular, la capa de transporte asigna un número de puerto comprendido en el rango de 1024 a 65535 que actualmente no esté siendo utilizado en ese host por ningún otro puerto UDP |

UDP no chequea que los datos hayan llegado bien. Eso a lo sumo lo hace la aplicación (usa el checksum de UDP).

TCP es orientado a la conexión. Esto quiere decir que se establece una conexión antes de intercambiar datos. Esto se llama saludo de tres vías:

**Paso 1: no lleva carga útil.**

Un cliente TCP comienza el enlace de tres vías enviando un segmento con el señalizador de control SYN (Sincronizar números de secuencia) establecido, indicando un valor inicial en el campo de número de secuencia del encabezado. Este valor inicial para el número de secuencia, conocido como número de secuencia inicial (ISN), se elige de manera aleatoria y se utiliza para comenzar a rastrear el flujo de datos desde el cliente al servidor para esta sesión.

**Paso 2: no lleva carga útil.**

El servidor TCP necesita reconocer la recepción del segmento SYN del cliente para establecer la sesión de cliente a servidor. Para hacerlo, el servidor envía un segmento al cliente con el señalizador ACK establecido indicando que el número de acuse de recibo es significativo. Con este señalizador establecido en el segmento, el cliente interpreta esto como acuse de recibo de que el servidor ha recibido el SYN del cliente TCP. El servidor manda su propio SYN. (Tenemos el flag ACK en 1, y el ACK de confirmación de secuencia recibida)

**Paso 3: puede o no llevar carga útil.**

Por último, el cliente TCP responde con un segmento que contiene un ACK que actúa como respuesta al SYN de TCP enviado por el servidor. Pueden no existir datos de usuario en este segmento. El valor del campo número de acuse de recibo contiene uno más que el número de secuencia inicial (es decir, ACK = SEQ\_SV + 1) recibido del servidor. Una vez establecidas ambas sesiones entre el cliente y el servidor, todos los segmentos adicionales que se intercambien en la comunicación tendrán establecido el señalizador ACK.

Una vez completados estos 3 pasos, los hosts cliente y servidor pueden enviarse segmentos que contengan datos el uno al otro.

Uso de flags en el saludo de tres vías:

ACK o “acknowledge” (1 bit): Si está activo entonces el campo con el número de acuse de recibo es válido (si no, es ignorado). Es un mensaje que se envía para confirmar que un mensaje o un conjunto de mensajes han llegado.

SYN o “synchronize” (1 bit): Activa/desactiva la sincronización de los números de secuencia.

Se usa para sincronizar los números de secuencia en tres tipos de segmentos: petición de conexión, confirmación de conexión (con ACK activo) y la recepción de la confirmación (con ACK activo).

RST o “reset” (1 bit): Si llega a 1, termina la conexión sin esperar respuesta.

Es un bit que se encuentra en el campo del código en el protocolo TCP, y se utiliza para reiniciar la conexión. Un ejemplo práctico de utilización es el que realiza un servidor cuando le llega un paquete a un puerto no válido: este responde con el RST activado.

Además de establecer conexión para enviar los datos, después hay que cerrar la conexión. Para esto, en lugar de usar los flags SYN / ACK, se usan los flags FIN / ACK. FIN y SYN no llevan datos. A modo de reconocimiento se toma como si fuera un byte.

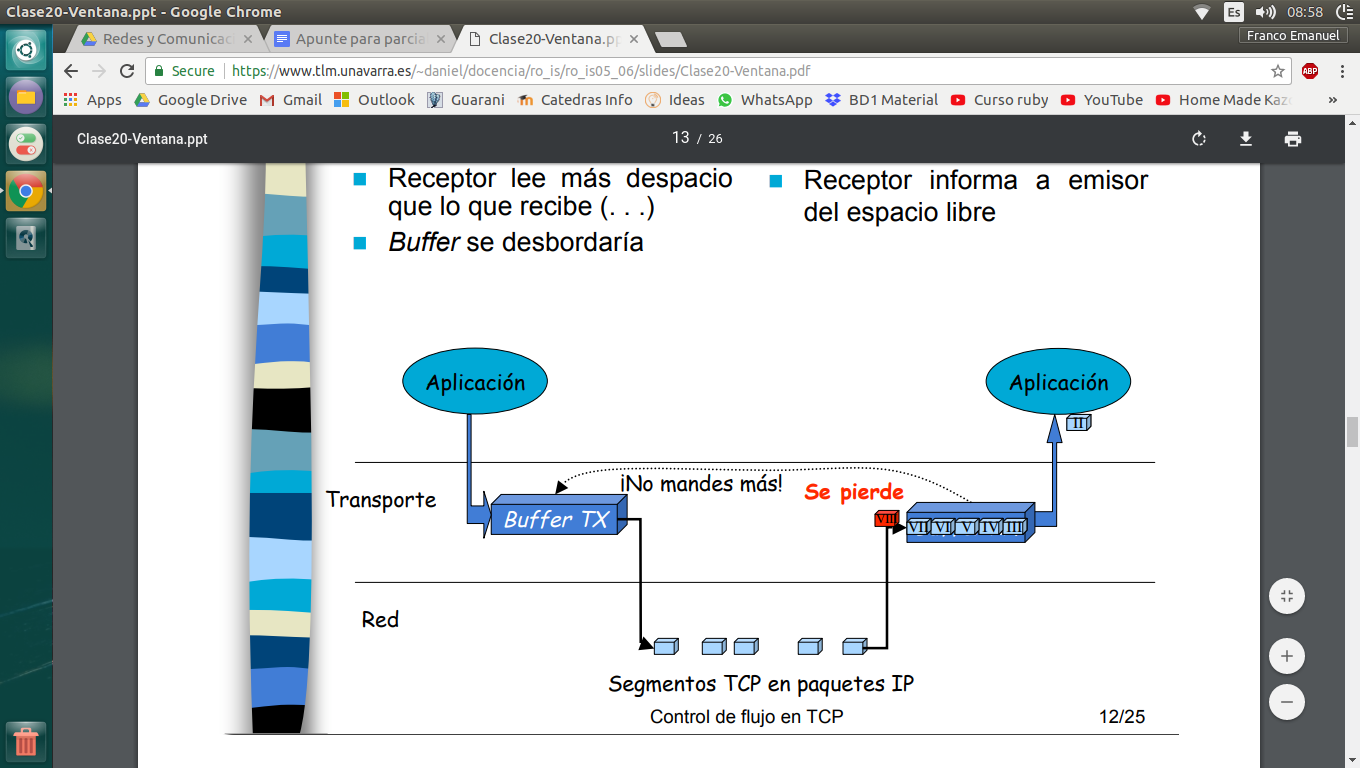
El proceso de cierre de conexión puede tener 3 o 4 pasos:

A diferencia del saludo de tres vías, las dos partes tienen que cerrar sesión. Ambas partes van a mandar un FIN (osea, cliente y servidor), y tiene que haber un ACK para las dos partes.

* Tiene 3 pasos en caso de: El primero manda FIN. El segundo manda el ACK correspondiente a dicho FIN, y también manda su propio FIN. Finalmente el primero manda un ACK, correspondiente al FIN del paso 2.
* Tiene 4 pasos en caso de: El primero manda FIN. El segundo manda el ACK correspondiente a dicho FIN. Cuando está listo para cerrar la sesión, manda su propio FIN. Finalmente el primero manda un ACK, correspondiente al FIN del paso 3.

El fin lo puede arrancar tanto el sv como el cliente. Si el cliente manda FIN, cuando llega el mensaje, la capa de transporte lo desempaqueta, ve que hay un pedido de cierre de conexión, entonces pasa la información a la aplicación subyacente. Cuando la aplicación está lista, recién ahí le dice a la capa de transporte que también solicite el fin de conexión. Va a depender de la aplicación (que tan rápido actúe), ver si en el segundo paso se manda solo ACK o también se manda FIN.

TCP usa un buffer, el cual puede usar para ir almacenando los paquetes que se envían, como los que llegan.



Esto ayuda también a garantizar el orden, ya que podría darse la siguiente situación:

1. A manda dos segmentos a B. Seg = 526 length = 200 y Seg = 726 = 220.
2. Seg 726 llega primero a B.
3. B guarda en el buffer dicho segmento, pero avisa que todavía está esperando al Seg 526 (manda ack = 526)
4. A puede decidir reenviar Seg = 526 length = 200, o enviar en partes mas chicas para intentar que no se pierdan, o mas grandes…
5. Cuando a B le llega toda la información de Seg 526 que estaba esperando, manda ACK 946, porque ya tenía en su buffer Seg 726.
6. Si los dos segmentos que A manda, llegan bien, B podría optar por solo mandar un ACK 946.

Control de flujo en TCP:

La idea es no desbordar el buffer del lado receptor de la red. El emisor tiene que adaptar la velocidad para no desbordar el buffer y no perder paquetes. El receptor tiene que mandar, en cada envío, el valor de su ventana de recepción (espacio libre en el buffer), y entonces el emisor puede limitar el número de bytes no reconocidos (ventana de emisión) al tamaño de la ventana de recepción. Si se llena el buffer del receptor, la ventana de recepción es 0.

En caso de llenar el buffer, el emisor tiene que enterarse cuando puede seguir mandando carga útil al receptor. Para esto va a mandar una serie de segmentos con bytes de prueba para detectar cuando hay espacio en el buffer del receptor (la aplicación ya tomó bytes que estaban en el buffer).

Control de congestión en TCP:

El objetivo es intentar transmitir a la máxima velocidad posible sin congestionar la red, para perder el menor número de paquetes posible y que no haya que retransmitir. Se busca la taza justo por debajo del límite de congestión.

Se hace de forma descentralizada: cada emisor decide en base a la recepción de ACKs. Si recibe ACK es que todo va bien, y puede aumentar la velocidad. Si hay segmentos perdidos, disminuye la velocidad.

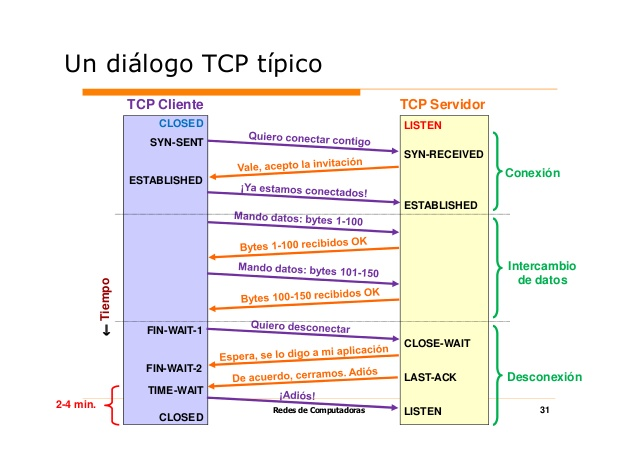
El emisor puede aumentar o disminuir su velocidad, con la ventana de emisión. No confundir ventana de congestión con ventana de recepción (esta se usa para control de flujo). La ventana del emisor es VentanaCongestion / RTT, donde VentanaCongestion = “paquetes que puede transmitir”, y RTT “Round-Trip -Time”.

Se reduce la velocidad cuando se pierden paquetes.

TCP tiene estados:

CLOSED : No hay conexión activa ni pendiente.  
LISTEN: El servidor espera una llamada.  
SYN RCVD: Llegó una solicitud de conexión; espera ACK.  
SYN SENT: La aplicación comenzó a abrir una conexión.  
ESTABLISHED: Estado normal de transferencia de datos.  
FIN WAIT 1: La aplicación dijo que ya terminó.  
FIN WAIT 2: El otro lado acordó liberar.

TIMED WAIT: Espera a que todos los paquetes mueran.  
CLOSING: Ambos lados intentaron cerrar simultáneamente.  
CLOSE WAIT: El otro lado inició una liberación.  
LAST ACK: Espera a que todos los paquetes mueran.



Si llega un segmento TCP a un host que no tiene ningún proceso esperando en el puerto destino de dicho segmento (osea, no está en estado Listen) no se podrá efectuar la conexión. Ningún socket estará esperando el segmento con el SYN. Se va a enviar una respuesta con el flag RST puesto a 1.

Si llega un segmento UDP con el número de puerto destino que no se corresponde con un socket UDP activo, el host envía un datagrama ICMP especial (puerto inalcanzable).

Cabe destacar que en caso de éxito, TCP responderia SA, y UDP no respondería nada.

Multicast es un método de transmisión uno a muchos, similar al broadcast, salvo que el multicast se envía a un grupo específico, y el broadcast a todos los nodos de la red. Normalmente se usa UDP, ya que las conexiones TCP son punto a punto. Además, la comunicación sería muy costosa, ya que habría que hacer el saludo de tres vías con cada host receptor.

Es posible iniciar mas de una conexión desde el mismo host origen hacia el mismo host destino, porque las aplicaciones en el host origen usan puertos distintos. Entonces, el servidor puede responder sin posibilidad de que la información se mezcle.

El método de **Parada y espera** (Stop-and-wait) es un tipo de protocolo ARQ para el control de errores en la comunicación entre dos hosts basado en el envío de tramas o paquetes, de modo que una vez se envía un paquete no se envía el siguiente paquete hasta que no se recibe el correspondiente ACK (confirmación de la recepción) y en caso de recibir un NACK (rechazo de la recepción) se reenvía el paquete anterior.

En un protocolo **GBN (Go-Back-N, Retroceder N)**. El emisor puede transmitir varios paquetes (si están disponibles) sin tener que esperar a que sean reconocidos, pero está restringido a no tener más de un número máximo permitido, N, de paquetes no reconocidos en el canal.

El rango de los números de secuencia permitidos para los paquetes transmitidos pero todavía no reconocidos puede visualizarse como una ventana de tamaño N sobre el rango de los números de secuencia. Cuando el protocolo opera, esta ventana se desplaza hacia adelante sobre el espacio de los números de secuencia. Por esta razón, N suele denominarse tamaño de ventana y el propio protocolo GBN se dice que es un protocolo de ventana deslizante.

La recepción del ACK de un segmento, confirma a todos los anteriores pendientes -acuse de recibo acumulado-. El vencimiento del timer para un segmento implica la retransmisión de todos los segmentos siguientes en la ventana. La ventana se desplaza a medida que se reciben ACK's. El receptor envía el ACK sólo del segmento recibido correctamente de mayor número de secuencia en orden -es decir, que no le falte ninguno-. Puede generar acuses duplicados. No precisa buffers, ya que descarta los segmentos fuera de orden y duplicados (al descartar re envía el ACK del mayor número de secuencia que recibió correctamente y en orden). Por esto último, es simple de implementar en el emisor y en el receptor.

Un único paquete erróneo podría hacer que el protocolo GBN retransmitiera una gran cantidad de paquetes, muchos de ellos de forma innecesaria. A medida que la probabilidad de errores en el canal aumenta, el canal puede comenzar a llenarse con estas retransmisiones innecesarias.

Como su nombre sugiere, los protocolos de repetición selectiva evitan las retransmisiones innecesarias haciendo que el emisor únicamente retransmita aquellos paquetes que se sospeche que llegaron al receptor con error (es decir, que se perdieron o estaban corrompidos). Esta retransmisión individualizada y necesaria requerirá que el receptor confirme individualmente qué paquetes ha recibido correctamente. De nuevo, utilizaremos una ventana de tamaño N (debe ser menor o igual a la mitad del tamaño del espacio de números de secuencia) para limitar el número de paquetes no reconocidos y en circulación en el canal. Sin embargo, a diferencia de GBN, el emisor ya habrá recibido mensajes ACK para algunos de los paquetes de la ventana. El receptor de SR confirmará que un paquete se ha recibido correctamente tanto si se ha recibido en el orden correcto como si no. Los paquetes no recibidos en orden se almacenarán en el buffer hasta que se reciban los paquetes que faltan (es decir, los paquetes con números de secuencia menores).

FTP (File transfer program) es un protocolo de transferencia de archivos. Se ejecuta por encima de TCP. En una sesión FTP típica, el usuario está sentado frente a un host (el host local) y desea transferir archivos a/o desde un host remoto. Para que el usuario pueda acceder a la cuenta remota, debe proporcionar una identificación de usuario y una contraseña. Una vez proporcionada esta información de autorización, el usuario puede transferir archivos desde el sistema de archivos local al sistema de archivos remoto, y viceversa.

**FTP ACTIVO**

Dicho modo es el habitual por defecto. De esta forma, **se establecen dos conexiones distintas.** En primer lugar, se establece una conexión desde el cliente, hacia el puerto 21 del servidor. A través de esta misma conexión, mediante un comando PORT se indica al servidor cuál será el puerto del cliente que se colocará a la escucha de datos.

De esta forma, si nos disponemos a descargar un fichero, es el servidor quien inicia la transferencia de los datos, desde su puerto 20 al puerto que se le ha sido indicado.

**FTP PASIVO**

En modo pasivo, es siempre el programa cliente quien inicia la conexión con el servidor. Se abre primeramente una conexión de control desde un puerto superior a 1023 del cliente, al puerto 21 del servidor. El cliente pasa a modo pasivo enviando el comando PASV y pidiendo un puerto abierto al servidor. Tras recibir la respuesta, será el cliente quien establezca la conexión de datos con el servidor a través del puerto que le ha sido enviado.

Protocolos de capa de aplicación sobre protocolos de capa de transporte (normalmente):

Paquetes UDP son de menores tamaño. Paquetes UDP no pueden ser mayor y 512 bytes. Por tanto, cualquier aplicación debe transferir datos mayor que 512 bytes TCP en lugar de requerir.

|  |  |
| --- | --- |
| **Capa de aplicación** | **Capa de transporte** |
| HTTP | TCP |
| HTTPS | TCP |
| TPTP | UDP |
| FTP | TCP |
| SMTP | TCP |
| IMAP | TCP |
| POP3 | TCP |
| IMAP | TCP |
| DNS | TCP para transferencia de zona. UDP para consultas de nombre o inversión. |
| DHCP | UDP |

**Práctica 7 - Capa de Red - Direccionamiento.**

El PDU de esta capa es “Paquete”. El router es solo de la capa de red (no implementa las capas superiores a la capa de red). Switch capa de enlace ->> mover.

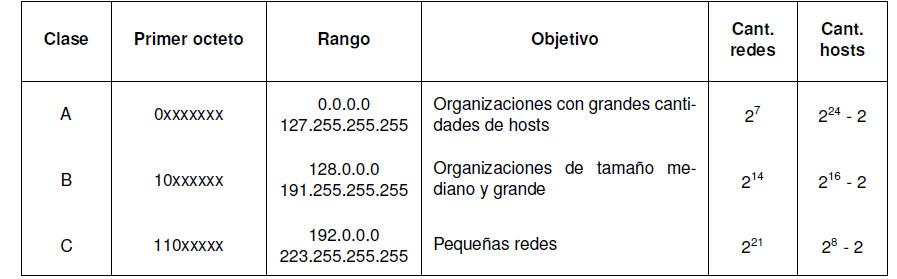
La capa de red transporta paquetes desde un host emisor a un host receptor. Esto lo hace con las siguientes funciones (a través del router):

* Reenvío (forwarding): cuando un paquete llega al enlace de entrada de un router, este tiene que pasar el paquete al enlace de salida apropiada.
* Enrutamiento (routing): se determina la ruta o camino que deben seguir los paquetes a medida que fluyen de un emisor a un receptor. Los algoritmos que calculan estas rutas se conocen como algoritmos de enrutamiento.

Un router realiza estas funciones examinando la cabecera del paquete entrante, y usa ese valor para indexarlo dentro de la tabla de reenvío del router. El algoritmo de enrutamiento determina los valores que se introducen en las tablas de reenvío de los routers.

La capa de red proporciona un servicio de mejor esfuerzo, lo cual quiere decir que la entrega de paquetes no está garantizada, ni tampoco el orden.

Direcciones IP y clases:

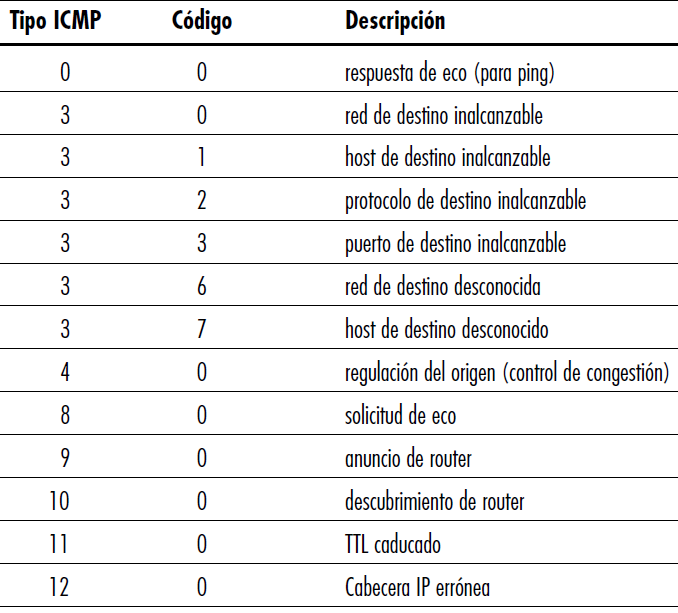
****

Inicialmente se asignaban direcciones basadas en clases a las redes. Sin embargo, esto agotó rápidamente tanto a las redes de clase A como a las redes de clase B. También es destacable la diferencia entre la cantidad de host direccionables entre las distintas redes, lo cual implica que por ejemplo, para una red, una dirección de red de clase C puede ser chica, pero una de clase B puede ser extremadamente grande, desperdiciando así muchas direcciones.

Esto dió origen a las subredes (concepto de subnetting). La idea es reducir al mínimo el desperdicio de direcciones IP. La idea es que si una red desperdicia muchas direcciones IP entonces la misma sea dividida en N subredes mas pequeñas que aprovechen mejor el espacio de direccionamiento.

El concepto de máscara indica en una dirección IP qué bits son de red, y qué bits son de host. Con el uso de las redes con clases, la máscara estaba implícita en la dirección de la clase, pues se conocía a priori los bits para red y los bits para host. Cuando se creó el concepto de subredes también se les asoció una máscara de subred, que resultó de utilizar algunos “bits de host” para crear subredes y de esta manera poder obtener varias subredes con menos hosts cada una.

ICMP es un protocolo importante de la capa de red, utilizado para intercambiar información entre hosts. El uso mas típico es informes de error. Los mensajes ICMP están justo encima de IP (son transportados dentro de datagramas IP).



Con el comando traceroute podemos trazar una ruta desde un host a cualquier otro host del mundo. Usa UDP, y un número de puerto destino poco probable. La idea es usar el TTL, el cual nos indica la cantidad de saltos que puede realizar un paquete. El cliente enviará muchos paquetes, cada cual con un valor de TTL mayor al anterior, con la idea de que el paquete esté un paso mas cerca del destino. Cada vez que el paquete pasa por un router, el router decrementa el valor de TTL. Si llega a 0, entonces el router descarta el datagrama y envía un mensaje al cliente con un mensaje ICMP tipo 11, código 0. Cuando el paquete llegue a destino, tendrá su TTL en 0, y es esperable que el puerto destino no coincida con un puerto UDP en estado Listen. En ese caso, el host destino enviará un mensaje ICMP tipo 3, código 3, y el host cliente sabrá que efectivamente el paquete llegó a destino.

*ifconfig* («configuración de interfaz») es un programa que permite configurar o desplegar numerosos parámetros de las interfaces de re[d](https://es.wikipedia.org/wiki/Tarjeta_de_red) residentes en el núcleo, como la dirección IP (dinámica o estática), o la máscara de red. Si se llama sin argumentos suele mostrar la configuración vigente de las interfaces de red activas, con detalles como la dirección MAC o el tráfico que ha circulado por las mismas hasta el momento. Las interfaces de red en Linux se suelen denominar eth (eth0, eth1, etc.).

*Route* es una herramienta de línea de comandos disponible tanto en Microsoft Windows como en GNU/Linux. Nos permite manipular las tablas de enrutamiento de nuestro sistema.

Un router suele tener mas de una interfaz de red. Una computadora normal tiene al menos dos interfaces (una de loopback y una con la que se accede a internet).

**Para el resto de la práctica, ver la guia de subnetting.**

**Trabajo Práctico 8 - Capa de red - Ruteo.**

Los routers tienen mas de una interfaz de salida, las cuales van a estar conectadas a una red. A su vez, esta red puede tener dispositivos como hub o switches, los cuales a su vez pueden tener distintas redes conectadas.

Los routers tienen tablas de enrutamiento, las cuales contienen la información necesaria para que el router tome las decisiones de reenvío de paquetes hacia la red de destino, utilizando la información de asociación entre la red y el siguiente salto. Esta tabla está en la memoria RAM del router.

Según lo visto en la materia, las tablas tienen 4 campos:

|  |  |  |  |
| --- | --- | --- | --- |
| Destino | Gateway | Máscara de red | Interfaz de salida |
| Es la red a la cual quiero llegar. La ruta por defecto es la 0.0.0.0, y es a donde se mandan los paquetes que no tienen ruta específica, lo cual normalmente se usa para internet. | Es el lugar al que tengo que enviar el datagrama para llegar a ese “destino”. Normalmente van direcciones de interfaces de router que me conectan con una determinada red. El destino 0.0.0.0 me indica que estoy en la misma red a la cual quiero enviarle información (no hace falta ningún intermediario). | Máscara de la dirección destino. | Placa de red por la que sale el paquete. En un router hay varias, en una computadora hay menos. |

En la tabla se ponen las redes conectadas directamente, y las formas de llegar a otras redes. Lo que hacemos en la práctica es enrutamiento estático.

El enrutamiento estático es viable cuando hay pocas máquinas en la red. Cualquier cambio requiere intervención humana.

El enrutamiento dinámico se usa para grandes redes. A priori, no requiere intervención humana.

Las máquinas conectadas a redes tienen tablas de ruteo, las cuales tienen al menos dos entradas:

* Entrada con un destino 0.0.0.0 y un gateway en particular, que indica a donde mando los paquetes cuando el destino no está en mi tabla de ruteo.
* Entrada con un destino particular y el gateway 0.0.0.0, que indica que ya estoy en la red con la cual me quiero comunicar.

DHCP es un protocolo que permite que un equipo conectado a una red pueda obtener su configuración de red (principalmente, pero no únicamente eso), en forma dinámica (sin intervención especial).

Se necesita un servidor DHCP que distribuya las direcciones IP. Este equipo será la base para todas las solicitudes DHCP por lo cual tiene una IP fija. El sistema de comunicación básico es BOOTP (trama UDP).

Para comunicarse con el servidor DHCP, el equipo envía un paquete especial de transmisión (dirección destino 255.255.255.255, el cual es un broadcast global. Esto lo hace porque no conoce su red. Por el contrario, cuando el cliente sí conoce su red, los broadcast que realiza (por ejemplo por ARP son id\_de\_red + [todos unos]), con información adicional como el tipo de solicitud, puertos, etc, a través de la red local. Cuando el servidor DHCP recibe el paquete, contesta otro paquete en forma de broadcast (el cliente no tiene dirección IP, asique no es posible conectar con él).

Los tipos de paquetes DHCP son:

* DHCP**DISCOVER** (para ubicar servidores DHCP disponibles)
* DHCP**OFFER** (respuesta del servidor a un paquete DHCPDISCOVER, que contiene los parámetros iniciales)
* DHCP**REQUEST** (Si tenés más de un servidor DHCP, todos contestan. El cliente con el request elige con cual se queda.)
* DHCP**ACK** (respuesta del servidor que contiene los parámetros y la dirección IP del cliente)
* DHCP**NAK** (respuesta del servidor para indicarle al cliente que su concesión ha vencido o si el cliente anuncia una configuración de red errónea)
* DHCP**DECLINE** (el cliente le anuncia al servidor que la dirección ya está en uso)
* DHCP**RELEASE** (el cliente libera su dirección IP)
* DHCP**INFORM** (el cliente solicita parámetros locales, ya tiene su dirección IP)

En DHCP se usa el término “concesión”. Las IP no son asignadas para siempre a los host, sino que tienen un tiempo límite. Cuando se está próximo a vencer el tiempo, el cliente puede solicitar una extensión con una DHCPREQUEST; también puede pasar que sea el servidor quien se ponga en contacto con el cliente, con una DHCPNAK para consultarle si desea extenderla. Si no se llega a buen puerto, la dirección IP va a estar nuevamente disponible.

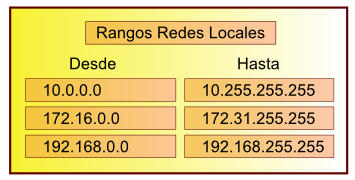
NAT es un router que se comporta de cara al exterior como un único dispositivo con una dirección IP única. Todo el tráfico que sale del router doméstico hacia internet tiene una dirección IP de origen igual, y todo el tráfico que entra en él tiene una dirección destino única. El router NAT obtiene su dirección IP de DHCP. NAT tiene una tabla de traducciones almacenada, para traducir y direccionar con éxito los paquetes que llegan del exterior, al host destino dentro de la red a la que pertenece el router NAT. Incluye los números de puerto y las direcciones IP en las entradas en la tabla.

Suponga que un usuario de la red doméstica que utiliza el host con la dirección 10.0.0.1 solicita una página web almacenada en un servidor web (puerto 80) con la dirección IP 128.119.40.186. El host 10.0.0.1, asigna el número de puerto de origen (arbitrario) 3345 y envía el datagrama a la LAN. El router NAT recibe el datagrama, genera un nuevo número de puerto de origen, 5001, para el datagrama, sustituye la dirección IP de origen por su dirección IP de la red WAN 138.76.29.7, y sustituye el número de puerto de origen original 3345 por el nuevo número de puerto de origen 5001. Al generar un nuevo número de puerto de origen, el router NAT puede seleccionar cualquier número de puerto de origen que actualmente no se encuentre en la tabla de traducciones NAT. (Observe que, puesto que la longitud del campo número de puerto es de 16 bits, el protocolo NAT puede dar soporte a 60.000 conexiones simultáneas utilizando la única dirección IP WAN del router). En el router, NAT también añade una entrada a su tabla de traducciones. El servidor web, que afortunadamente no es consciente de que el datagrama entrante que contiene la solicitud HTTP ha sido manipulado por el router NAT, responde con un datagrama cuya dirección de destino es la dirección IP del router NAT y cuyo número de puerto de destino es 5001. Cuando este datagrama llega al router NAT, éste indexa la tabla de traducciones NAT utilizando la dirección IP de destino (138.76.29.7) y el número de puerto de destino (5001) para obtener la dirección IP (10.0.0.1) y el número de puerto de destino (3345) apropiados para el navegador de la red doméstica.

¿Para qué sirve?

* Sirve para ahorrar direcciones IPv4 públicas.
* Para transformar direcciones privadas en direcciones públicas. De esta forma es posible conectar un host de una red privada con otro host de otra red privada (a través del router NAT).
* Se puede utilizar el router como intermediario (da una sensación de “seguridad”, aunque no siempre es verdad).

La RFC 1918 se relaciona con NAT, porque NAT permite transformar direcciones privadas en públicas, esta RFC especifica rangos de direcciones privadas por clases (IPv4):



**Práctica 9 - Capa de Red - IPv6**

IPv6 comenzó a desarrollarse para sustituir a IPv4, sobretodo por el agotamiento de direcciones IPv4 (por ejemplo, en IPv4 fue necesario usar NAT para ahorrar direcciones; las tablas de ruteo eran muy grandes y generaban congestión en los routers porque requería demasiado procesamiento)..

IPv6 tiene un mayor espacio de direcciones. Son 128 bits. Se utiliza el sistema hexadecimal en grupos de 16 bits, separados por “:”.

* 0 al inicio se pueden obviar.
* Ceros contiguos se pueden eliminar con “::”. Solo se puede hacer esto una vez.
* No se usa máscara.
* No hay división en clases.

IPv6 tiene una cabecera de 40 bytes simplificada. Algunos campos de IPv4 se eliminaron o hicieron opcionales. La cabecera de longitud fija permite un procesamiento mas rápido. En IPv4 el tamaño de cabecera era variable, y para decir el tamaño se necesitaba el campo “header length”, el cual en IPv6 fue eliminado por ser innecesario.

Comparación de cabeceras de IPV4 e IPV6



En IPv6 no tenemos campo checksum. La tarea de verificar que los datos llegan bien se delegó a la capa de transporte (en UDP va a tener que ser obligatoria). El objetivo es reducir el tiempo de procesamiento; en IPv4, por el campo TTL (el cual cambiaba en cada salto), la suma de comprobación necesitaba ser calculada en cada router. Ahora esto se eliminó, y será mas rápido. El campo TTL de IPv4 fue reemplazado por el campo “límite de saltos”.

ICMP ha sido redefinido para IPv6. Además de reorganizar las definiciones de tipos y códigos ICMP existentes, ICMPv6 también añadió nuevos tipos y códigos requeridos por la nueva funcionalidad de IPv6, entre los que se incluyen el tipo “Paquete demasiado grande” y el código de error “Opciones IPv6 no reconocidas”. Además, ICMPv6 incluye la funcionalidad del Protocolo de gestión de grupos de Internet (IGMP, Internet Group Management Protocol) que se emplea para gestionar el modo en que un host se une a un grupo de multidifusión y lo abandona, anteriormente era un protocolo separado de ICMP en IPv4.

Una computadora puede calcular su dirección IPv6 a partir de su MAC.

La dirección MAC es un identificador de 48 bits (6 bloques de dos caracteres hexadecimales (4 bits)) que corresponde de forma única a una tarjeta de red. Es única para cada dispositivo.

Existen distintas categorías de direcciones IP en IPv6. Una de estas categorías es las link-local. Una dirección IPv6 calculada a partir de una MAC es una de tipo link-local, y solo se pueden usar para comunicaciones dentro de una subred local. Los routers no enrutan paquetes con direcciones de enlace local. En IPv6 las direcciones de enlace local son necesarias para el funcionamiento de varios componentes del protocolo.

Las direcciones de enlace local IPv6 tienen la forma: FE80::X/64.

Para convertir las MAC en IPv6, se usa el proceso EUI-64, el cual consta de los siguientes pasos:

Las MAC tienen 48 bits. Los primeros 24 son OUI (identificador único de organización), y los últimos 24 son identificador de dispositivo.

* Paso 1: Dividir la dirección MAC entre el OUI y el identificador de dispositivo
* Paso 2: Insertar el valor hexadecimal FFFE, que en formato binario es: 1111 1111 1111 1110. Esto representa los restantes 16 bits para completar la dirección de 64 bits. (La MAC tiene 48 + 16 = 64).
* Paso 3: Convertir los primeros dos valores hexadecimales del OUI a binario e invertir el bit U/L (séptimo bit) En este ejemplo, el 0 en el bit 7 se cambia a 1.

Las direcciones IPv6 se clasifican en: unicast, anycast y multicast.

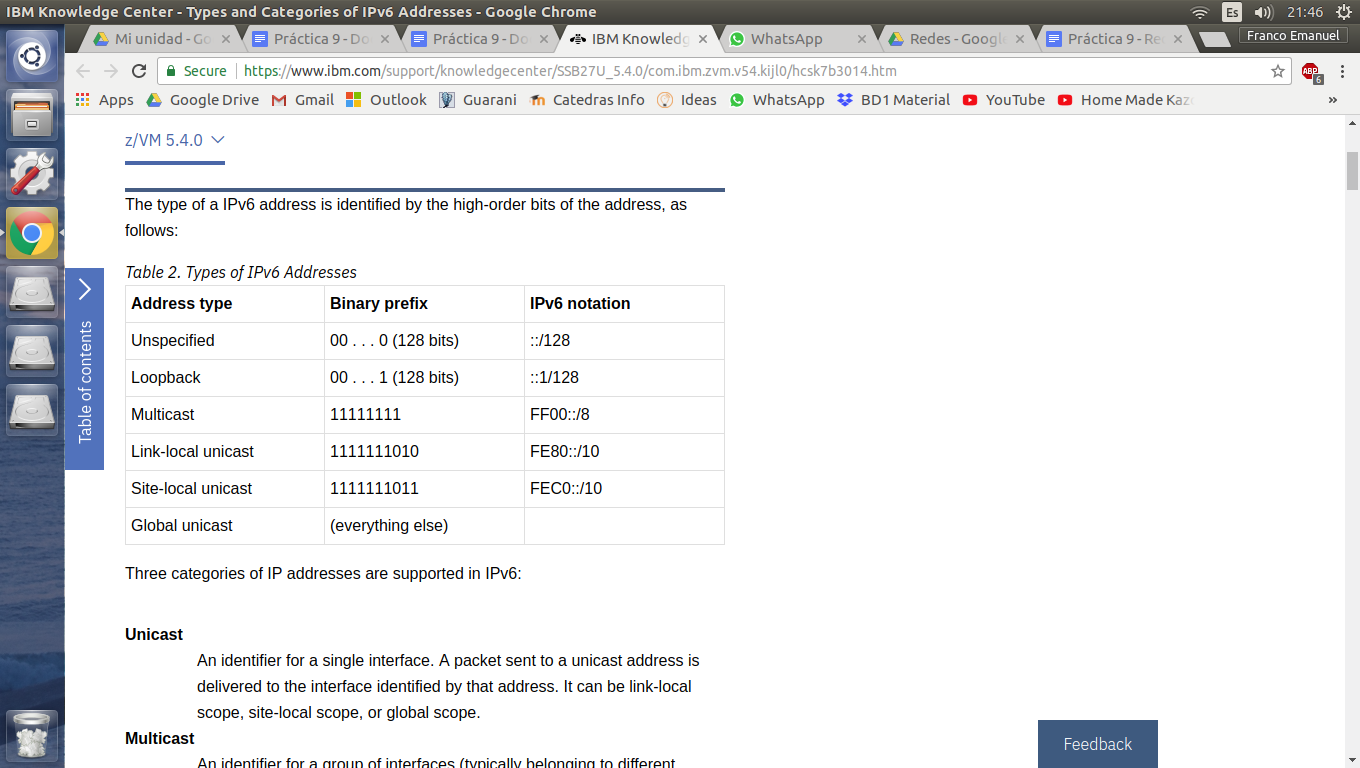
* Una dirección unicast identifica a un único interfaz de red. El protocolo de internet entrega los paquetes enviados a una dirección unicast al interfaz específico.
* Una dirección anycast es asignada a un grupo de interfaces, normalmente de nodos diferentes. Un paquete enviado a una dirección anycast se entrega únicamente a uno de los miembros, típicamente el host de menos coste. Cualquier dirección unicast puede usarse como anycast (la única diferencia, es que las anycast son varias).
* Las multicast son usadas por múltiples interfaces también. Consiguen esta dirección participando de un grupo de multidifusión (multicast) entre los routers de la red. Un paquete enviado a una dirección multicast es entregado a todos los interfaces que se hayan unido al grupo multicast correspondiente. En IPv6 no hay broadcast, se usa multicast (ff01::1, que es “all nodes”). Los primeros dos dígitos hexadecimales son FF.

Dentro de la categoría unicast, existen:

* Link local: equivalentes a IP privadas en IPv4. El prefijo es FE80::/10. No pueden ser encaminadas en routers fuera del segmento local. Los primeros 10 bits (por eso el /10) se usan para formar FF8, los siguientes bits tienen valores 0, consiguiendo que el prefijo de red sea el mismo para todas las direcciones locales y por tanto, no sea enrutable. FE80:0000:0000:0000:0000:0000:0000:0000/10 → Porción de red.

Los últimos 64 bits son la porción de nodo, la cual se forma con EUI-64 (explicado previamente).

* Site local: también equivalentes a las IPv4 privadas. Pueden ser encaminadas fuera del segmento local, pero no hacia internet. El prefijo es FEC0::/10. El procedimiento es el mismo que las link-local.
* Global: equivalentes a las públicas de IPv4. Pueden ser encaminadas a través de la red. Tienen un valor hexadecimal de 2xxx, con una máscara /3.



Para multicast, los cuatros bits menos significativos del segundo octeto de una dirección multicast (ff0*X*::) identifican el ámbito, es decir, hasta dónde se propaga el tráfico multicast. Los ámbitosdefinidos actualmente son:



La idea será que cada host pueda autoconfigurar su dirección IPv6 sin usar DHCP. Para esto, usando EUI-64 genera los 64 bits menos significativos de su dirección IP, y usar la dirección de red (enviada periódicamente por el router) como 64 bits mas significativos. La unión, me genera una dirección IPv6 de 128 bits.

**Práctica 10 y 11 - Capa de enlace - Parte I y II**

PDU de esta capa: trama.

La capa de enlace se encarga de mover las tramas nodo a nodo. Un mismo datagrama de capa de enlace puede ser transportado por diferentes protocolos de la capa de enlace en los distintos enlaces que forman la ruta.

Los servicios de esta capa son:

* Encapsulación (o entramado): En el campo de datos va el paquete de la capa de red. En la cabecera van campos a usar por la capa de enlace.
* Acceso al enlace: Usa el protocolo de acceso al medio MAC para especificar reglas a la hora de transmitir una trama a través de un enlace.
* Entrega fiable: garantiza que no se van a producir errores. No es recomendable implementarla siempre. Se recomienda en enlaces inalámbricos, ya que tienen altas tasas de error. No se recomienda en cableados, ya que la tasa de error es baja.
* Control de flujo: los nodos en los extremos de un enlace tienen una capacidad limitada de almacenamiento en su buffer de tramas. La idea es que el emisor no abrume al receptor.
* Detección de errores: los errores de bit se producen debido a la atenuación de las señales y el ruido electromagnético. Muchos protocolos de la capa de enlace usan un mecanismo de errores de bit, lo cual usa una serie de bits de comprobación de errores en la trama, que el nodo receptor usa. En general esto se implementa en hardware.
* Corrección de errores: similar a la detección de errores, pero el nodo receptor corrige.
* Semi-duplex y full-duplex: con la transmisión full-duplex, los nodos de ambos extremos de un enlace pueden transmitir paquetes al mismo tiempo. Sin embargo, con la transmisión semi-duplex, un mismo nodo no puede transmitir y recibir al mismo tiempo.

La capa de enlace le brinda servicios a la capa de red, y la capa de transporte le brinda servicios a la capa de aplicación.

Dos máquinas en una misma red se identifican por su dirección MAC, las cuales (como fue dicho antes), son direcciones de 48 bits de longitud. Se suelen expresar en notación hexadecimal. Nunca puede haber dos placas de red con la misma dirección MAC.

La dirección de broadcast de la capa de enlace es FF-FF-FF-FF-FF-FF. Se usa cuando un adaptador de red quiere que todos los demás adaptadores de la LAN reciban y procesen la trama que va a enviar.

Una colisión se produce cuando dos o mas nodos transmiten por el mismo enlace al mismo tiempo. Las señales se mezclan y por ende no quedan utilizables, así que hay que retransmitirlas.

Los dominios de colisión representan un segmento físico de una red donde es posible que las tramas puedan interferir unas con otras.

Los switches, bridges y routers dividen los dominios de colisión. Los repetidores y hubs propagan las colisiones.

El dominio de broadcast es el conjunto de todos los dispositivos que reciben tramas de broadcast que se originan en cualquier dispositivo del conjunto.

Los routers dividen los dominios de broadcast. Un dominio de broadcast representa una red. Los routers separan redes. Los repetidores, hubs, bridges y switches propagan los dominios de broadcast.

Dispositivos por capas (modelo OSI):

|  |  |
| --- | --- |
| **Dispositivo** | **Capa** |
| NIC (tarjeta de red) | Capa de enlace |
| Repetidor | Capa física |
| HUB | Capa física |
| Switch | Capa de enlace |
| Router | Capa de red |
| Bridge | Capa de enlace |

Las tablas de switches tienen dos campos, dirección MAC de la PC y PUERTO que me comunica con dicha PC. Las entradas se van a ir llenando cuando fluya el tráfico ARP. Para esto es que nos sirve saber de qué capa es cada dispositivo, nos permite saber con que datos trabaja.

Las tramas en una red Ethernet se envían a todos los dispositivos que conforman la red, siendo los propios aparatos los que determinan si el paquete va dirigido a ellos o no, denegando todos los paquetes que no se dirigen estrictamente al dispositivo en particular.

Todos los dispositivos de una red pueden transmitir paquetes en cualquier momento en que así se requiera, sin embargo esto puede provocar problemas cuando dos dispositivos intentar hacerlo al mismo tiempo, produciéndose colisiones. Por esta razón se creó CSMA/CD (Acceso múltiple con escucha de portada y detección de colisiones), el cual es un protocolo que permite a los dispositivos escuchar la red, para determinar si el canal y los recursos están libres. En caso afirmativo, se podrá realizar la transmisión para no colisionar con otros paquetes.

PASOS:

1. El adaptador recibe un datagrama de la capa de red y crea la trama.
2. Si el adaptador sensa que el canal está libre, éste comienza a transmitir la trama. Si éste sensa canal ocupado, espera hasta que esté libre y transmite.
3. Si el adaptador transmite la trama entera sin detectar colisión, se considera transmisión lograda.
4. Si el adaptador detecta otra transmisión mientras transmite, aborta y envía una señal de bloqueo (jam)
5. Después de abortar, el adaptador entra en backoff exponencial: después de la m-ésima colisión, el adaptador elige un K aleatorio entre {0, 1, 2, …, 2^m-1}. El adaptador espera K·512 periodos de 1 bit y retorna al paso 2

El protocolo ARP (protocolo de resolución de direcciones) se usa para encontrar la dirección MAC correspondiente a una determinada dirección IP. Surge de la necesidad de que para que una trama se coloque en los medios LAN primero es necesario que cuente con una dirección MAC destino. Para esto, el nodo primero consulta a la tabla ARP de su memoria para encontrar la dirección de la capa de enlace de datos (osea, la MAC) que se mapea con la dirección IP de destino. Si no encuentra la entrada que busca en la tabla, envía un paquete ARP request a la dirección de difusión de la red (broadcast, MAC FF-FF-FF-FF-FF-FF) que contiene la dirección IP por la que se pregunta, y se espera que esa máquina (u otra) responda un ARP reply con la dirección Ethernet que le corresponde (la respuesta es unicast). Esta respuesta se usa para registrar una nueva entrada en la tabla ARP.

Se usa entonces, para supervisar y modificar la tabla de asignaciones de direcciones IP y direcciones MAC. En la tabla ARP de cada dispositivo tenemos las asignaciones de nivel de enlace de datos (MAC) y las direcciones IP del nivel de red, logrando una correspondencia MAC → IP.

Los nodos crean sus tablas de ARP sin intervención del administrador (autoaprendizaje).

<https://geekytheory.com/redes-el-protocolo-arp>

Cuando una computadora se quiere comunicar con otra, primero realiza un proceso llamado ANDing.

**Primero, el host de origen compara (AND) su propia dirección IP con su propia máscara de subred. El resultado de ANDing es la identificación de la red en la que reside el host de origen. Entonces compara las direcciones IP de destino con su propia máscara de subred. El resultado del segundo ANDing será la red en la que está el host de destino. Si la dirección de la red de origen y la dirección de la red de destino son iguales, se pueden comunicar directamente. Si los resultados son distintos, están en diferentes redes o subredes, y necesitarán comunicarse a través de routers, o puede que no sean capaces de comunicarse.**

El proceso ANDing depende de la máscara de subred. La máscara de subred siempre son únicas. Una máscara de subred predeterminada para una red de Clase C es 255.255.255.0 o 11111111.11111111.11111111.00000000 . El host compara esta máscara con la dirección IP de origen bit a bit. El primer bit de la dirección IP se compara con el primer bit de la máscara de subred, el segundo con el segundo, y así sucesivamente. Si los dos bits son unos, el resultado de ANDing es un 1. Si los bits son un cero y un uno o ambos son cero, el resultado de ANDing es un 0. Básicamente, esto significa que una combinación de dos unos da como resultado un 1; cualquier otra cosa es un 0. El resultado del proceso ANDing es el número de red o subred en que está la dirección de origen o de destino.

Si la una PCA comprueba que una PCB con la cual quiere comunicarse no está en la misma red, necesita la MAC del default gateway para poder comunicarse.

Comprobación de errores:

* Bits de paridad:

Los **códigos de paridad** se usan en telecomunicaciones para detectar, y en algunos casos corregir, errores en la transmisión. Para ellos se añade en origen un bit extra llamado [*bit de paridad*](https://es.wikipedia.org/wiki/Bit_de_paridad) a los *n* [bits](https://es.wikipedia.org/wiki/Bit) que forman el carácter original.

Este valor del bit de paridad se determina de forma que el número total de bits *1* a transmitir sea par (código de paridad par) o impar (código de paridad impar).

Así, **para el código de paridad par, el número de unos contando el caracter original y el bit de paridad tiene que ser par.** Por lo tanto, el bit de paridad PAR será un 0 si el número total de unos a transmitir es par y un 1 para un número impar de unos.

Por el contrario, **para el código de paridad impar, el número de unos contando el caracter original y el bit de paridad ha de ser impar.** De esta forma, el bit de paridad IMPAR será un 0 si el número total de unos es impar y un 1 para un número par de unos.

Normalmente el bit de paridad se añade a la izquierda del caracter original.

* FCS

Idea: enviar los datos originales más un FCS que me permita decir si los datos llegaron correctamente. Se supone que antes de la transferencia tanto el receptor como el emisor deben pactar cuál será el Generador, secuencia de bits que se usará para dividir el msj y comprobar.

Para enviar:

1. Se obtiene la longitud del FCS, será la longitud del Generador menos 1
2. Se le agregan tantos ceros como sea la longitud del FCS al final del msj a transmitir
3. Se divide esa nueva secuencia de bits con los ceros agregados POR el Generador, haciendo esta operación en módulo a 2, por lo que no interesan los carries al restar y se usa XOR en la resta (da 1 cuando los bits que se están operando son distintos, y 0 cuando son iguales)
4. Se obtiene el resto de esa división, el cual serà el FCS.
5. Se envían los datos originales reemplazando los ceros concatenados anteriormente por el FCS.

Para verificar la info recibida:

1. Como se que me llegó un msj mas un FCS, divido todo eso por el Generador y la división me debe dejar un resto = a 0. Si no es así, hubo error.

Video explicativo: [**https://www.youtube.com/watch?v=Fz6b1t8sr10**](https://www.youtube.com/watch?v=Fz6b1t8sr10)

ARP no existe en IPv6. En su lugar se usa el multicasting, el cual funciona de la siguiente manera:

Se crea un grupo Multicast, formado por conjunto de interfaces de red.

Si se está interesado en que cierta computadora reciba los paquetes de difusión del grupo se agrega una interfaz de red, de esa forma se envía un paquete multicast al grupo X.

Ese paquete sólo llegará a aquellas computadoras que tengan su interfaz incluida en el grupo multicast X. Con ello se permite tener niveles de eficiencia de red superiores a los presentados en IPv4, lo cual se verá traducido en la disminución de los ciclos de procesamiento de CPU de las computadoras en la red local al no procesar paquetes de difusión que no van dirigidos a ellos y de la misma manera se estará eliminando el problema de las tormentas de paquetes de difusión de IPv4.

El funcionamiento del protocolo ND de IPv6 equivale a combinar los siguientes protocolos de IPv4: ARP (Address Resolution Protocol), ICMP (Internet Control Message Protocol, Router Discovery e ICMP Redirect. IPv4 carece de un protocolo general establecido y de un mecanismo para detectar la *‘alcanzabilidad’* de vecinos.

**Neighbor Discovery** (ND) es un [protocolo](https://es.wikipedia.org/wiki/Protocolo_de_red) de [IPv6](https://es.wikipedia.org/wiki/IPv6), y es equivalente al protocolo [Address Resolution Protocol (ARP)](https://es.wikipedia.org/wiki/Address_Resolution_Protocol) en [IPv4](https://es.wikipedia.org/wiki/IPv4), aunque se distingue porque también incorpora funcionalidades de [ICMP](https://es.wikipedia.org/wiki/Internet_Control_Message_Protocol).

Utiliza mensajes especiales de [ICMPv6](https://es.wikipedia.org/wiki/ICMPv6) construyendo así una manera simple para que los terminales aprendan las direcciones [IPv6](https://es.wikipedia.org/wiki/IPv6) de los vecinos de la capa de enlace. Consiste en un mecanismo con el cual un nodo que se acaba de conectar a la red, descubre la presencia de otros nodos en el mismo enlace, además de ver sus direcciones IP. Otra de las grandes funcionalidades de este protocolo es que se ocupa de mantener limpios los cachés dónde se almacena la información relativa al contexto de la red a la que está conectado un nodo. Así, cuando una ruta hacia cierto nodo falla, el router correspondiente buscará rutas alternativas. Basándose también en los mensajes ICMPv6 se permite un mecanismo de auto-configuración.

Mapea direcciones lógicas (IPv6) a direcciones de Hardware (MAC, EUI-48, EUI-64).

**Aclaraciones generales:**

En capa 4 se ponen los puertos.

En capa 3 se ponen las IP.

En capa 2 se ponen las MAC.

Protocolos de capa 2: Ethernet y ARP.

Protocolos de capa 3: IP, ICMP

Protocolos de capa 4: TCP, UDP.

Protocolos de capa 5: DNS, SMTP, etc.