XC

DOSSIER FINAL

MIRIAM

Índice

4. TCP/UDP

5. Aplicaciones de Red

1.1 Modelo OSI (Open System Interconnection)

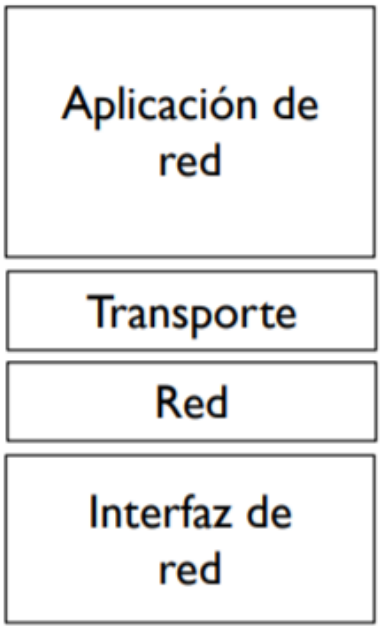
Este modelo es uno de los más importantes utilizados para referirnos a las conexiones en la red. Como podemos observar en la imagen, se divide en capas para facilitar la comunicación entre redes.
Se creó después de que en todos los sitios con conexión a la red se usará un sistema distinto, por lo tanto necesitábamos un sistema que permitiera estandarizar las conexiones y de esta forma simplificar todas las conexiones y permitir que las redes se pudiesen entender entre sí. Importante saber que este modelo es un modelo conceptual por lo tanto no es el que se utiliza a efectos prácticos(se utiliza el modelo TCP/IP).

Ejemplo de cada capa:

- **Capa 1:** La capa física es la primera capa del modelo. Mediante esta capa definimos como viajan los bits sobre un enlace de datos conectando los nodos de red para formar finalmente una red
- **Capa 2:** La capa de enlace es la segunda capa del modelo, se encarga de transferir los datos entre los nodos que pertenecen al mismo segmento de red.
- **Capa 3:** La capa de red es la tercera capa del modelo. Es la responsable del reenvío de paquetes, incluido el enrutamiento a través de routers intermedios.
- **Capa 4:** La capa de transporte proporciona los medios para transferir secuencias de datos de longitud variable desde una fuente a un host de destino, mientras se mantiene la calidad de las funciones de servicio.
- **Capa 5:** La capa de sesión proporciona el mecanismo para abrir, cerrar y administrar una sesión entre los procesos de la aplicación del usuario final.
- **Capa 6:** La capa de presentación es responsable del formato y la entrega de información a la capa de aplicación para su posterior procesamiento o visualización.
- **Capa 7:** Una capa de aplicación es una capa de abstracción que especifica los protocolos de comunicaciones compartidos y los métodos de interfaz utilizados por los hosts en una red de comunicaciones.

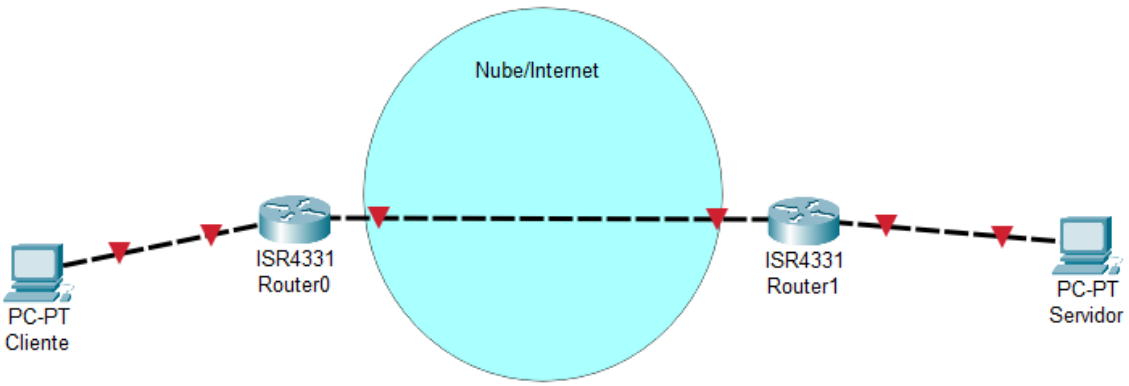
1.2 Modelo TCP/IP

Este es el modelo que se utiliza en todo Internet, previamente había diversos modelos basados en OSI, pero finalmente se estandarizó el uso del modelo TCP/IP. Se puede apreciar que es un modelo más simplificado del modelo OSI. En este caso las capas 1 y 2 del modelo OSI se agrupan para formar la capa de interfaz de red y la capa 5, 6 y 7 del modelo OSI se agrupan para formar la capa de aplicación de red.



1.3 Paradigma cliente/servidor

Este principio se describe de la forma siguiente: para que un cliente pueda acceder a un servicio necesita acceder a un servidor que puede estar a un número N de redes que ha de atravesar el cliente para llegar a ese servicio, de esta forma, el servidor proporciona el servicio que ha pedido el cliente



2.1 Introducción IP

Protocolo utilizado por todo internet cuya función principal es la comunicación entre los dispositivos a través de redes.

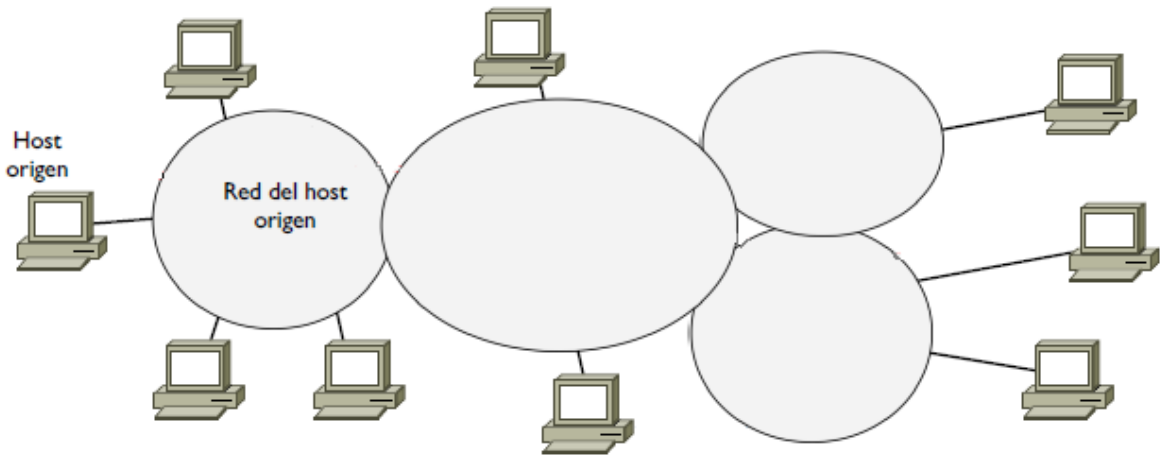
La unidad que utiliza es **paquete** o **datagrama**.

Principales características:

- **Best effort:** Mejor esfuerzo para realizar la entrega de los paquetes, se entregarán los paquete de la mejor forma posible, por lo tanto no se puede asegurar un QoS(calidad de servicio)
- **Stateless**: Protocolo sin estado, esto indica que la conexión no mantiene el estado de la conexión(por ejemplo las sesiones), por lo tanto cada conexión se trata de manera independiente.
- **Conectionless**: No orientado a conexión, se pueden enviar los paquetes sin tener en cuenta si el dispositivo final está disponible o no.

Tipos de datagrama en la comunicación IP según destino

- **Unicast**: datagrama enviado a solo un destinatario
- **Broadcast**: datagrama enviado a todos los integrantes de una red
- **Multicast**: datagrama enviado a múltiples destinatarios que pueden estar en más de una red.

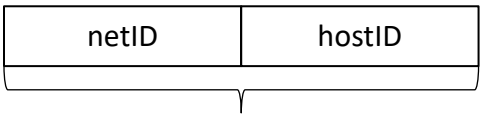


2.2 Componentes de una dirección IP

Las IP's se descomponen en 32 bits, y para hacerlas más entendibles, se dividen en 4 octetos. También tenemos que saber (y es muy importante) que las IP's tienen dos partes:

- **hostID**: para identificar a un host
- **netID**: identificador de red.

La parte alta de la dirección identifica la red y la parte baja identifica al host dentro de la red, esto dependerá totalmente del tipo de IP que tengamos, si se trata de una dirección con clases estándar y una dirección con máscara variable.



32 bits → 4 octetos en decimal separados por un punto

La @IP identifica una interfaz conectada a una red (se asigna a interfaces de hosts y routers), las @IP con mismo netID pertenecen a la misma red.

Número de @IP en una red = $2^n - 2$ (una para red y una para broadcast)

- Un hostID con todos los bits a 0 identifica la dirección de red
- Un hostID con todos los bits a 1 identifica la dirección de broadcast.

Para **reconocer** la parte netID y hostID de una @IP, tenemos dos maneras:

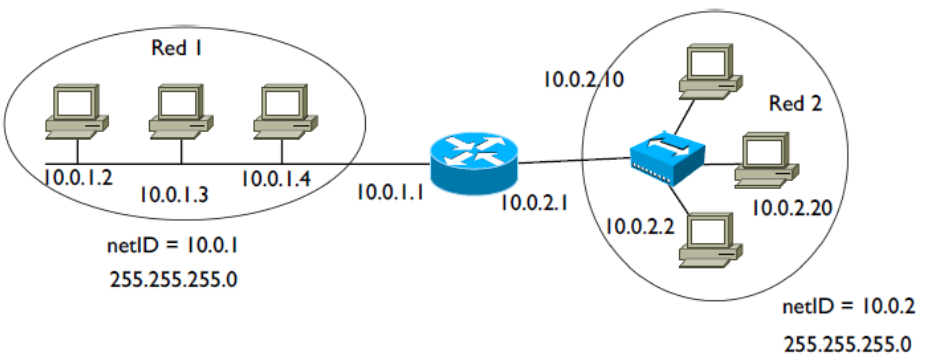
2.2.1 Máscaras: modelo sin clases llamado CIDR(Classless InterDomain Routing)

netID	hostID
11111111.11111111.11111111.	00000000

Una máscara de red se utiliza para identificar la red en una dirección IP, hay dos formas de representarlas

La primera forma de representarla es escribir la dirección de la siguiente forma:
•192.168.1.0/24 → El /24 nos indicará la máscara de red que tenemos.

La segunda forma de representarla es escribiendo la máscara poniendo todos los bits de la máscara a 1. La máscara /24 en este caso se representaría de la siguiente forma:
•255.255.255.0



Para **reconocer** la parte netID y hostID de una @IP, tenemos dos maneras:

2.2.2 Clases: Si no se asigna una máscara a una @IP, por defecto se consideran las clases para saber el número de bits del netId y hostID

Classe	netid (bytes)	hostid (bytes)	Codification	range
A	1	3	0xxxx···x	0.0.0.0 ∼ 127.255.255.255
B	2	2	10xxx···x	128.0.0.0 ∼ 191.255.255.255
C	3	1	110xx···x	192.0.0.0 ∼ 223.255.255.255
D	-	-	1110x···x	224.0.0.0 ∼ 239.255.255.255
E	-	-	1111x···x	240.0.0.0 ∼ 255.255.255.255

NOTA: Las clases D se utilizan para multicast y las E están reservadas, ambas se identifican solo con tener los 4 bits más significativos de forma 1110 y 1111 respectivamente(no se pueden usar como @IP de interfaces).
Para identificar el resto de clases es tan simple como mirar el bit más significativo. Las clases A siempre tendrán el bit más significativo a 0, las clases B lo tendrán a 1 y las clases C lo tendrán de la forma 110.

2.2.3 Direcciones especiales

Siempre que realicemos un direccionamiento de IP's tenemos que recordar que tenemos dos direcciones que siempre se han de reservar, una es la de broadcast y otra es la de red.

Dirección de broadcast: Utilizada para enviar mensajes en broadcast(a toda la red). Ejemplo:192.168.1.255/24 es una dirección de broadcast(es lo mismo que 192.168.1.11111111/24)

Dirección de red: dirección para identificar la red, se forma poniendo toda la hostID a 0. Ejemplo: 192.168.1.0/24 es una dirección de red(es lo mismo que: 192.168.1.00000000/24)

Otras direcciones especiales:

Direcciones para la **configuración de DHCP**: 0.0.0.0 y 255.255.255.255, direcciones utilizadas al encender un equipo y realizar la configuración mediante DHCP.

Dirección es **loopback**: Dirección utilizada para la comunicación entre procesos dentro del mismo equipo. Se utiliza la dirección 127.x.x.x.

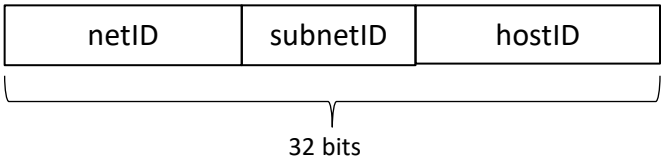
2.2.4 Direcciones privadas

Direcciones IP reservadas para poder ser reutilizadas en redes locales, los rangos van de la siguiente forma:

- Clase A: 10.0.0.0 – 10.255.255.255
- Clase B: 172.16.0.0 – 172.31.255.255
- Clase C: 192.168.0.0 – 192.168.255.255

2.3 Subnetting

El subnetting es un mecanismo mediante el cual dividimos una dirección de red en otras más pequeñas(subnets) para poder aprovechar la IP y ajustarla a nuestras necesidades.
Cuando hacemos subnetting, se extiende el netID ocupando más bits, es decir, se usan bits del hostID para el subnetID.

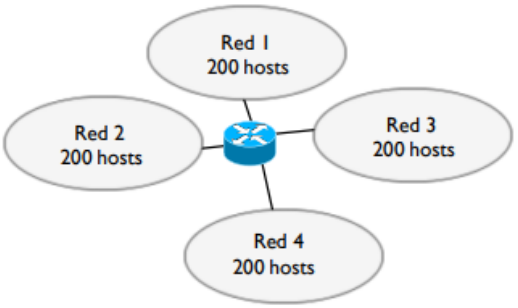


Hay 2 maneras

2.3.1 Máscara Fija

Se han de coger bloques enteros, el mismo número de host para cada red

Ejemplo: Dada la dirección 192.168.0.0/22, queremos obtener 4 redes de 200 hosts cada una.



Como necesitamos 4 redes, necesitamos 2 bits para poder crear las redes. ¿Y para los hosts cuanto necesitamos? Si necesitamos 200 hosts sabemos que $2^7 = 128$, ¡No llega! Pero sabemos que $2^8 = 256$, este direccionamiento si da, habrá IP's que no estemos utilizando pero no nos importa ya que ganamos flexibilidad gracias a este método.
De esta forma tenemos que nuestra IP original se dividirá de la siguiente forma

*192.168. 000000 **00** . 00000000*
subnetID hostID

Lo siguiente ya es realizar el subnetting, para esto tenemos que recordar que siempre que asignamos una red, tenemos que tener 2 direcciones reservadas. Una para la dirección de red y otra para la dirección broadcast. Con esto dicho, el subnetting quedaría de la siguiente forma:

Red 1
- *Dirección de red: 192.168.0.0/24*
- *Dirección de broadcast: 192.168.0.255*
- *Direcciones asignables: 192.168.0.1 - 192.168.0.254*

Red 2
- *Dirección de red: 192.168.1.0/24*
- *Dirección de broadcast: 192.168.1.255*
- *Direcciones asignables: 192.168.1.1 - 192.168.1.254*

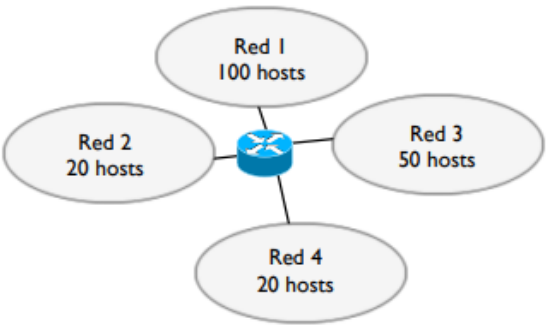
Red 3
- *Dirección de red: 192.168.2.0/24*
- *Dirección de broadcast: 192.168.2.255*
- *Direcciones asignables: 192.168.2.1 - 192.168.2.254*

Red 4
- *Dirección de red: 192.168.3.0/24*
- *Dirección de broadcast: 192.168.3.255*
- *Direcciones asignables: 192.168.3.1 - 192.168.3.254*

2.3.2 Máscara Variable (VSLM: Variable Length subnet Mask)

Se usa cuando no tenemos el mismo número de hosts en las diferentes subredes. Conviene empezar siempre con la red de mayor tamaño e ir en orden decreciente.

Ejemplo: Disponemos de la dirección 192.168.0.0/24 para realizar el subnetting y se nos pide crear 4 redes, pero esta vez serán de 100 hosts, 50 hosts, 20 hosts y 20 hosts cada red.



Red 1 (100 hosts)
Necesitamos 7 bits(128) de hostID para los 100 hosts.
La división de bits quedará de la siguiente forma:
*192.168.0. **0** 0000000*
subnetID hostID
Por lo tanto:
- *Dirección de red: 192.168.0.0/25*
- *Dirección de broadcast: 192.168.0.127*
- *Direcciones asignables: 192.168.0.1 - 192.168.0.126*

Red 2 (50 hosts)
Necesitamos 6 bits(64) de hostID para 50 hosts.
La división de bits quedará de la siguiente forma:
*192.168.0. **10** 000000*
subnetID hostID
Por lo tanto:
- *Dirección de red: 192.168.0.128/26*
- *Dirección de broadcast: 192.168.0.191*
- *Direcciones asignables: 192.168.0.129 - 192.168.0.190*

Red 3(20 hosts)
Necesitamos 5 bits(32) de hostID para 20 hosts.
La división de bits quedará de la siguiente forma:
*192.168.0. **110** 00000*
subnetID hostID
Por lo tanto:
- *Dirección de red: 192.168.0.192/27*
- *Dirección de broadcast: 192.168.0.223*
- *Direcciones asignables: 192.168.0.193 - 192.168.0.222*

Red 4(20 hosts)
Necesitamos 5 bits(32) de hostID para 20 hosts.
La división de bits quedará de la siguiente forma:
192.168.0. **111** 00000
subnetID hostID
Por lo tanto:
- Dirección de red: 192.168.0.224/27
- Dirección de broadcast: 192.168.0.255
- Direcciones asignables: 192.168.0.225 - 192.168.0.254

2.4 Tablas de enrutamiento/encaminamiento

Estas tablas sirven para guardar la información referente a cómo se llega a otras redes desde nuestra red. Es una base de datos que tiene cada router y host y contiene la información necesaria para alcanzar todos los destinos. Tienen el siguiente formato

Adquisición	Destino	Máscara	Gateway	Interfaz

Los campos se describen de la siguiente forma:

- Adquisición: Indica el mecanismo/protocolo que se ha utilizado para adquirir la ruta. Ej: S para rutas estáticas puestas manualmente, R para rutas aprendidas por RIP, C para rutas conectadas directamente, O para rutas aprendidas por OSPF...
- Destino y Máscara: Indica donde se accede mediante la entrada de la tabla. Se define mediante la dirección IP y su máscara en formato decimal(255.255.255.0 por ejemplo). Si estos valores son 0.0.0.0 indica que es la ruta por defecto(cualquier dirección que no esté en la tabla se irá por aquí)
- Gateway: Siguiente salto para llegar al destino. En caso de ser 0.0.0.0 indica que estamos conectados directamente
- Interfaz: Interfaz por la que se sale hacia el destino.

2.5 Protocolo ARP (Address Resolution Protocol)

Protocolo utilizado para poder asociar una IP con una MAC, se utiliza para poder realizar correctamente la comunicación a nivel local.

Las @IP son asignadas por un administrador, mientras que las @MAC son fijas.

Los routers y hosts almacenan estas resoluciones en una tabla ARP, cada resolución tiene una duración (normalmente 5 o 20 min). Al principio las tablas están vacías.

@IP	@MAC	Duración

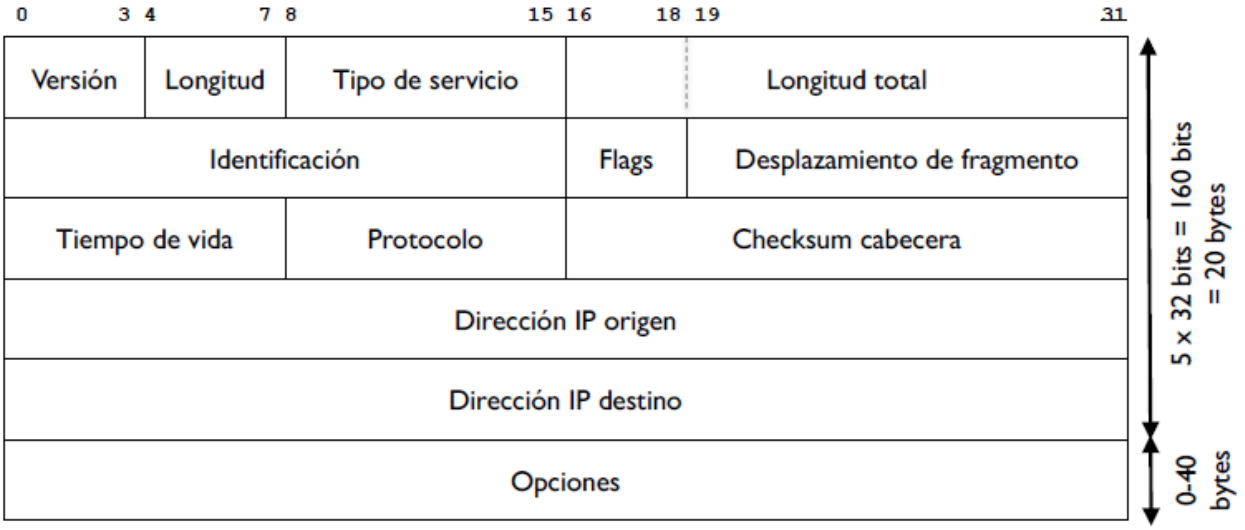
Funcionamiento: Cuando un equipo quiere conocer la dirección de otro equipo, ha de enviar un mensaje ARP en broadcast(**ARP Request: FF:FF:FF:FF:FF:FF**) para preguntar quién tiene la dirección X, una vez hecho esto, el equipo responderá indicando su MAC, mediante un **ARP Reply** en unicast indicando al equipo origen que él tiene la IP X, de esta forma se asocia la MAC con la IP X en la tabla ARP del equipo que solicitó en primera instancia saber la dirección asociada el equipo destino.

Notas:

ARP solo funciona a nivel loca (las tramas no cruzan redes)l, en otras redes se encarga el router de realizar los ARP correspondientes

Existen dos tipos especiales de ARP:
- **Reverse ARP** sirve para asociar una MAC a una IP(al revés que el ARP normal)
- **ARP gratuito** que sirve para detectar si una IP está duplicada(se envía un mensaje ARP con la misma dirección que ya tenemos asignada).

2.6 Cabecera IP

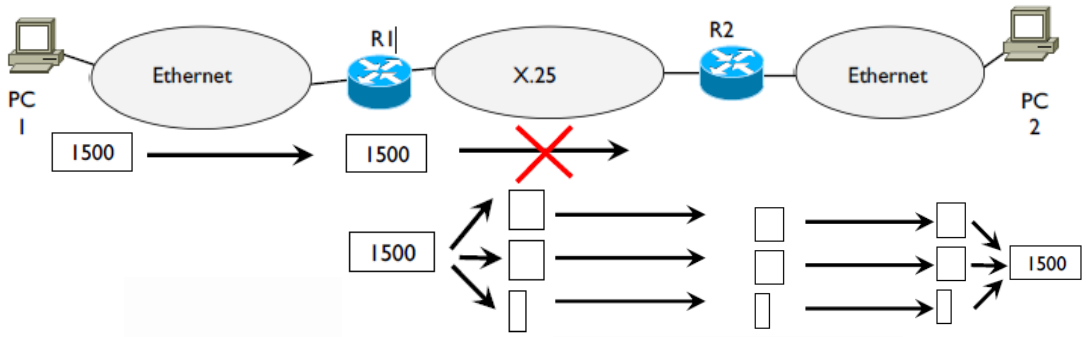


- **Versión**: campo para indicar si se trata de v4 o v6 de IP
- **Longitud**: Longitud de la cabecera IP (min. 20 bytes, max. 60 bytes)
- **Tipo de servicio**: Utilizado para indicar la calidad de servicio (en Internet no se usa)
- **Longitud total:** Longitud total de todo el datagrama(incluyendo la cabecera) (max. 2^{16} -1 = 65535 bytes)
- **Identificación**: Identificador del paquete, se utiliza solo cuando se realiza fragmentación el paquete original para poder saber a qué paquete pertenecen los fragmentos.
- **Flags**: Campo de 3 bits usado para indicar si hay fragmentación:
  - Bit 0: Siempre a 0
  - Bit 1: Si es 1 indica que se no puede fragmentar, si es 0 indica que si se puede fragmentar (DF)
  - Bit 2: Indica si es el último fragmento. (MF)
- **Desplazamiento de fragmento (OFFSET):** Cuando un paquete esté fragmentado indica la posición dentro del paquete.
- **Tiempo de vida(TTL):** Número máximo de saltos entre routers que puede hacer el paquete a lo largo de la red. Cuando este campo llegue a 0, el paquete se descarta. (normalmente = 64, max. 256)
- **Protocolo**: Indica el protocolo de capas superiores.
- **Checksum cabecera:** Utilizado para detectar paquetes corruptos/dañados.
- **Dirección IP origen**: Origen del paquete
- **Dirección IP destino**: Destino del paquete
- **Opciones:** Campo de opciones, poco utilizado.

2.7 Fragmentación

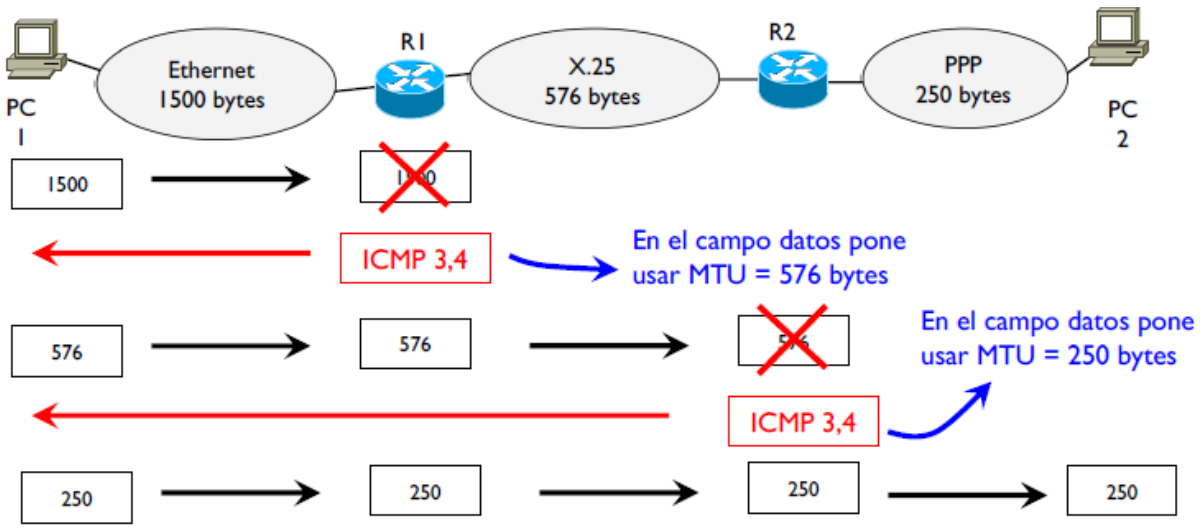
Se utiliza el MTU(maximum unit transfer), cada router define su MTU, de esta forma, si nos llega un paquete con un MTU superior al esperado, tenemos que fragmentarlo, para finalmente en su destino unir el paquete de nuevo. El MTU indica el máximo de bytes que se pueden encapsular en una trama.

Ethernet	1500 bytes
WiFi	2312 bytes
PPP	250 bytes
X.25	576 bytes



2.7.1 MTU Path Discovery

Para evitar la fragmentación de paquetes IP, se puede utilizar un mecanismo conocido como MTU path discovery, consiste en enviar un paquete que no se pueda fragmentar, de esta forma se nos devolverá el tamaño máximo que podemos atravesar, así hasta llegar al destino, de esta forma sabremos el tamaño máximo que podemos enviar en un paquete IP.



2.8 Protocolo ICMP (Internet Control Message Protocol)

Protocolo utilizado para intercambiar mensajes de estado y/o error entre dos terminales. A pesar de que se encapsule en un paquete IP puede ser generado tanto por la capa de red, transporte o aplicación. Para evitar la creación de bucles en la red, un mensaje ICMP de error no generará otro nunca.

Datos del mensaje ICMP

Tipo	Código	Checksum
Datos ICMP		

- **Tipo**: Indica el tipo de mensaje ICMP
- **Código**: Junto con el tipo indica que mensaje ICMP estamos tratando
- **Checksum**: Comprobación de errores
- **Datos ICMP**: Varía dependiendo del tipo de mensaje ICMP

Type	Code	query/error	Name	Description
0	0	query	echo reply	Reply an echo request
3	0	error	network unreachable	Network not in the RT.
	1	error	host unreachable	ARP cannot solve the address.
	2	error	protocol unreachable	IP cannot deliver the payload
	3	error	port unreachable	TCP/UDP cannot deliver the payload
	4	error	fragmentation needed and DF set	MTU path discovery
4	0	error	source quench	Sent by a congested router.
5	0	error	redirect for network	When the router send a datagram by the same interface it was received.
8	0	query	echo request	Request for reply
11	0	error	time exceeded, also known as TTL=0 during transit	Sent by a router when `--TTL=0`

La imagen presentada nos indica las combinaciones de Tipo y Código para cada tipo de mensaje ICMP.

2.9 Protocolo DHCP (Dinamic Host Configuration Protocol)

Protocolo utilizado para realizar la configuración inicial de red de los equipos cuando entran en la red de forma automática(sin necesidad de tener que configurar nada manualmente). Puede dar más cosas que una @IP(máscara de red, Gateway, ….)

Usa UDP como transporte, el **servidor** utiliza el **puerto 67** y el **cliente** el **puerto 68**. Se basa en el paradigma cliente-servidor.

El servidor y el cliente deben estar en la misma red y en un misma red pueden haber más de un servidor DHCP.

La asignación puede ser:
- Automática: sin límite de tiempo
- Dinámica: durante un periodo de tiempo determinado por el servidor, para mantener la IP por más tiempo, hay que renovar el tiempo de expiración antes de que termine.
- Manual: el servidor asigna @IP a determinadas @MAC, previamente configuradas manualmente por el administrador.

El funcionamiento funciona mediante los mensajes DHCP siguientes: DHCPDISCOVER, DHCPOFFER, DHCPREQUEST, DHCPACK.

2.10 Mecanismo NAT (Network Address Translation)

Mecanismo utilizado para poder realizar la traducción de IP privadas a IP públicas. ¡**ATENCIÓN,** se trata de un **mecanismo**, no de un protocolo!

Existen distintos tipos de NAT (que se pueden combinar):

• NAT Estático (SNAT)

Se utiliza de forma que se mapea uno a uno las direcciones, es decir, una IP pública se asocia con una IP privada. Se usa para servidores que están en la red interna y deben ser accesibles desde Internet.

• NAT Dinámico (DNAT)

A diferencia del NAT estático, ahora tendremos una pool de direcciones públicas que utilizaremos para mapear de forma dinámica según demanda a las direcciones privadas. Se suele usar para clientes que empiezan una comunicación con otro host.

• PAT: Port Address Translation(PNAT)

La más utilizada en hogares ya que nos hace ahorrar muchas direcciones públicas. Su uso se basa en que solo se utiliza una dirección pública en una red. Para que múltiples IP's privadas puedan salir, se utiliza la dirección pública junto con un puerto para diferenciarlas.

- Puertos efímeros;: ≥ 1024
- Puertos Well-known: < 1024

2.11 Protocolos de enrutamiento

Protocolos utilizados para definir las tablas de enrutamiento de forma dinámica. Las tablas se construyen mediante la métrica entre las distintas redes, facilitando la administración de las mismas y ahorrando trabajo.

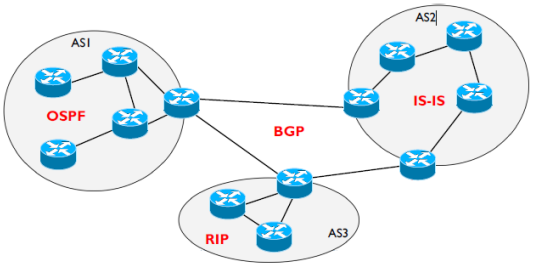
Para definir las tablas de enrutamiento de forma dinámica. Necesitamos:

- **Algoritmo de encaminamiento:** según la información y las métricas definidas, decide las entradas en la tabla.
  - estado de enlace: camino más corto con otras métricas
  - vector distancia: camino más corto en #saltos
- **Protocolo de encaminamiento:** coordina y define el contenido
  - RIP: vector distancia
  - BGP: vector distancia
  - OSPF: estado de enlace

Para diferenciar si es interno o extreno:
- IPG: Internal Gateway Protocol
- EPG: External Gateway Protocol

Como puede resultar obvio, en el encaminamiento no se puede utilizar solo 1 protocolo de enrutamiento debido a motivos prácticos(por ejemplo RIP solo admite un máximo de 15 saltos), por lo tanto se han de utilizar diversos protocolos de enrutamiento para unir los distintos puntos de internet. Cabe destacar que internet está dividido en **sistemas autónomos (AS)**

- Sistemas autónomos: los sistemas autónomos se identifican con un número y son grupos grandes de redes que tienen una política interna propia de encaminamiento(un ISP es un AS). Cada AS tiene un #limitado de routers.



2.11.1 RIP(Routing Information Protocol)

Protocolo utilizado con vector de distancia, envía mensajes periódicamente para actualizar las tablas de enrutamiento en cada router, estos mensajes contendrán las redes que sabe el router que las envía, de esta forma, cada router podrá actualizar su información mediante los mensajes que le lleguen del router origen.

La métrica en RIP se indica con un número, siendo 1 el mínimo(conectado directamente) y 15 el máximo(el 16 denota infinito, es decir, el enlace se ha caído). Cada salto indica el número de redes que hay entre cada red. Usa puerto 520 (origen y destino)

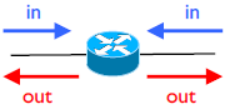
- **SPLIT-HORIZON**: Transmite por una interfaz solo aquellas entradas de la tabla que el router no ha aprendido por la red de esa interfaz
- **POISON REVERSE**: Si hay un fallo en una interfaz de un router, éste detecta el fallo y pone métrica 16
- **TRIGGERED UPDATE**: El router envía un mensaje para notificar el problema.

Usa @IP multicast para distribuir actualizaciones

Tiempo de convergencia: es el tiempo que se tarda en rellenar la tabla

2.12 ACL (Access Control List)

Listas de acceso, se utilizan como método de seguridad, pudiendo filtrar los paquetes que entran y salen de una red.
El router necesita implementar funciones de Firewall y el control el router se hace mediante las ACL.. Para poder diseñar las ACL's tenemos que tener en cuenta que tipo de ACL queremos diseñar, estas pueden ser de entrada o salida



Un ejemplo de ACL podría ser:

access-list 102 deny tcp any any eq 22

En este caso, estamos denegando el tráfico mediante ssh(puerto 22) a cualquier usuario que se intente conectar por ssh.
En las tablas de exámenes normalmente se siguen las siguientes columnas:

- Protocolo
- @IP origen y destino
- Puerto origen y destino
- acción(denegar o aceptar)

Tipos:
- Estandard: filtran sólo por la cabecera IP
- Extendida: además filtran por la cabecera TCP

Al final de todas las ACL hay una regla implícita que deniega todo como política de seguridad.

Se aconseja aplicar una primera ACL lo más próximo posible a la zona que se quiere proteger

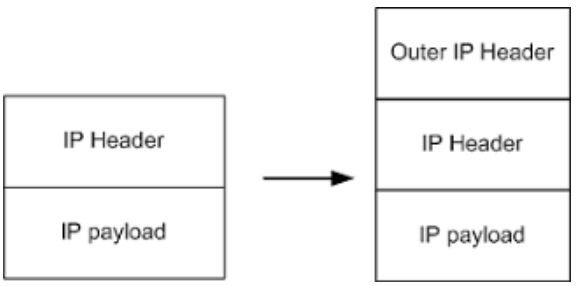
Puertos importantes:
- ssh → 22 TCP
- telnet → 23 TCP
- FTP → 20/21 TCP
- http → 80 TCP
- https → 443 TCP
- DHCP → 67/68 UDP
- DNS → 53 UDP

Cualquier puerto >1024 es un puerto no conocido y por tanto es el utilizado por cualquier cliente.

2.13 VPN (Virtual Private Networks)

Una VPN es una red virtual privada que se crea mediante un túnel con el cual podemos conectarnos desde cualquier ubicación a la red de la empresa. Un túnel es un enlace virtual que une dos routers, para poder enviar paquetes entre los túneles, se ha de crear una red entre ambos.
Este túnel entonces encapsulará el paquete IP origen dentro de otro paquete IP con los datos de la red del túnel, este proceso se hace mediante un protocolo llamado IPinIP.

NOTA A la hora de proceder con la fragmentación, hay que tener en cuenta que al encapsular un paquete dentro de otro, tendrá una cabecera extra de tamaño, por lo tanto el paquete tendrá un tamaño de: tamaño del paquete+20.



3.1 LAN's

En este tema nos enseñarán cómo funcionan las redes a nivel local. Ahora estamos hablando sobre la capa 1 del modelo TCP/IP(Interfaz de red). Trabajaremos a nivel local, es decir, estaremos siempre sobre la misma red trabajando cuando hablemos de LAN. Estas nubes pueden ser:

- Redes de área local (Local Area Network, LAN)
- Redes de área extendida (Wide Area Network, WAN) }
- Redes de área metropolitana (Metro Area Network, MAN)

En XC nos centraremos en las redes de área local(LAN). A diferencia del nivel 3, ahora trabajaremos a nivel de trama, que es el nombre de la PDU utilizada en el nivel 1, sin embargo, dentro de este nivel, tenemos otras subcapas que estudiaremos, qué son las siguientes:

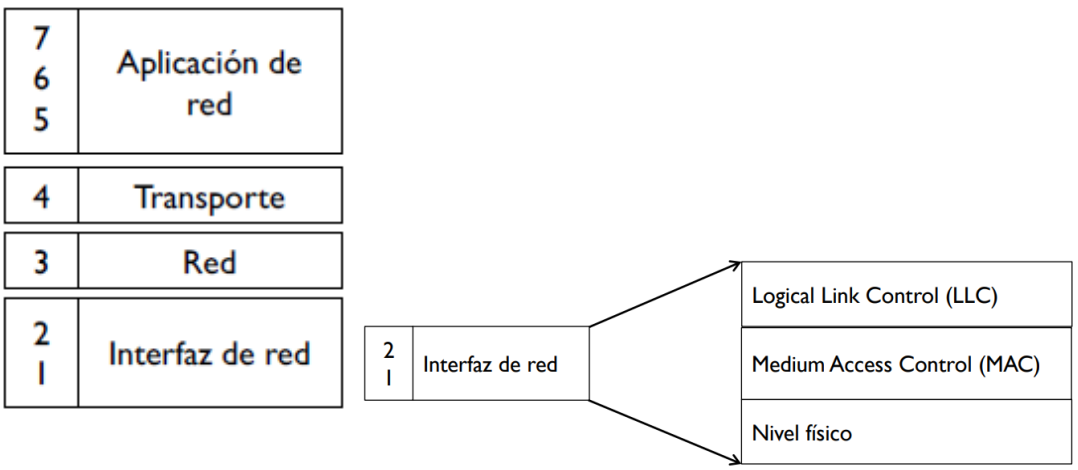
- **LLC(IEEE 802.2)**: Es el mismo en todos los estándares de MAC.
  Sirve como interfaz entre los niveles superiores y todos los protocolos MAC que pueda tener por debajo

- **MAC(IEEE 802.x)**: Distintos protocolos encargados de repartir el uso de la red local en el medio.

En XC los dos principales que se estudian son el 802.11 que es el protocolo para definir el WiFi y el 802.3 que es el protocolo Ethernet.

Concepto clave
- **Colisión**: Una colisión ocurre cuando dos dispositivos intentan enviar datos a la vez en una red ethernet.
- **Dominio de colisión**: Segmento de la red donde pueda haber una colisión

3.1.1 Arquitectura

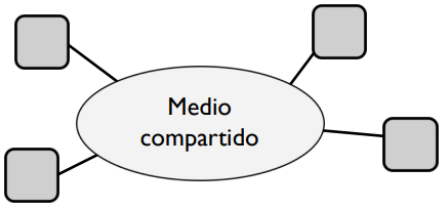


802.1: LAN/MAN architecture.
802.2 Logical Link Control (LLC)
802.3 Ethernet
802.4 Token Bus
802.5 Token Ring
802.8 FDDI
802.11 WiFi: Wireless LANs.
802.15 Personal Area Networks or short distance wireless networks (WPAN)
802.15.1 Bluetooth
802.15.4 low data rate and low cost sensor devices
802.16 WiMAX: broadband Wireless Metropolitan Area Networks.

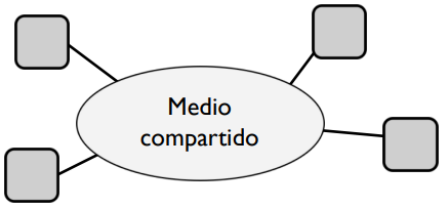
3.1.2 Acceso al medio compartido

Una LAN es una red de acceso múltiple en un medio compartido. Características

- Las estaciones están conectadas a un medio de transmisión común que no permite más de una transmisión a la vez
- Si una estación transmite una trama, todas las otras reciben, pero solo una se queda con la trama



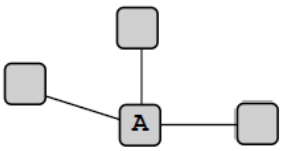
- Si dos o mas estaciones transmiten a la vez, se crea una colisión

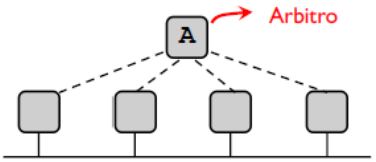


3.1.3 Protocolos MAC

Se define para regular el acceso al medio compartido. Ha de ser **distribuido**, **eficiente** y **equitativo**. Hay 2 tipos

- Acceso centralizado: Una estación y otro dispositivo regulan el acceso de cada estación





- Acceso distribuido: No hace falta un arbitro, cada estación decide por su cuenta
    - Estático
    - Paso de testigo: funciona mediante un token para evitar las colisiones en la red, solamente la terminal que tenga el token podrá transmitir, por lo tanto solo podrá viajar un paquete a la vez en la red. Se utiliza en token ring por ejemplo.
    - Aleatorio: No hay un token, por lo tanto existe una probabilidad de colisión, para evitar esto, se utiliza un tiempo de backoff en el cual se espera para volver a reenviar la trama (usa CSMA/CD para las colisiones)

3.2 Ethernet

- Estándar IEEE 802.3
- Usa protocolo MAC llamado CSMA/CD
- Tipo distribuido y aleatorio
- Topología bus y estrella

Tipos de conexiones

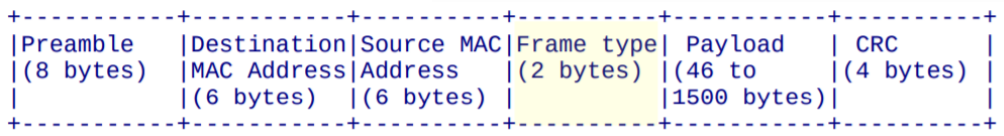
Velocidad de transmisión ⟵ X base Y ⟶ Varios significados
- Distancia máxima
- Topología
- Tipo de transmisor
- Etc.

10base5	10 Mbit/s	500 m
10baseT	10 Mbit/s	Topología estrella
100baseTX	100 Mbit/s	Topología estrella, full duplex
1000baseLX	1000 Mbit/s	Fibra óptica, laser
10GbaseCX4	10 Gbit/s	Infiniband

Por ejemplo:

10baseT indica que se trata de una conexión que va a 10Mbps, si hubiese un 100, serían 100Mbps, en cambio si hubiese una G(10G), indicaría que se trata de una conexión de 10GBps

Trama Cabecera



Preámbulo: Usada para la sincronización del reloj

MAC destino y MAC origen: Identifica la dirección MAC de los dispositivos origen y destino

Longitud: Indica la longitud de la trama, como puede ser de tamaño variable, se indica en este campo(va de 0 a 1500)

Payload: Los datos del paquete encapsulado dentro de la trama.

CRC: Utilizado para el cálculo de redundancia (detección y control de errores)

Concepto half-duplex full-duplex

Concepto utilizado en las redes Ethernet para referirnos a la capacidad de un medio:

- **Half-dúplex:** Una estación puede o transmitir o recibir, nunca las dos cosas a la vez
- **Full-dúplex:** Una estación puede transmitir y recibir al mismo tiempo

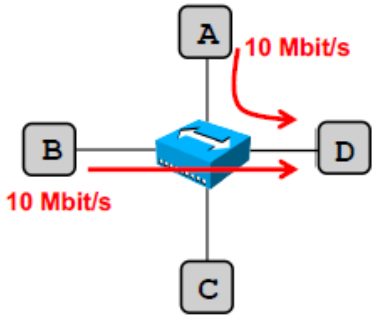
Actualmente prácticamente todas las conexiones son full-duplex

3.3 CSMA/CD: Carrier Sense Multiple Access With Control Detection

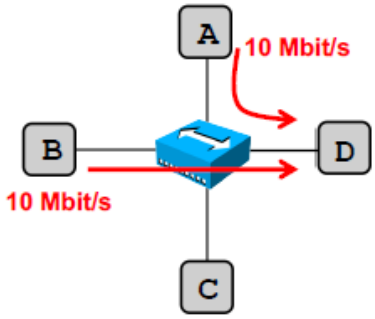
Sistema utilizado en las redes Ethernet para detectar las colisiones.

- CSMA: Siempre antes de iniciar una transmisión de datos(Tx), la estación verificará si otra estación está transmitiendo ya los datos (si está ocupado espera que se libere)
- CD: La estación también buscará en caso de que haya iniciado la transmisión al mismo instante que otra estación. En caso de detectar que hay otra estación transmitiendo, esperará un tiempo de backoff, mediante el cual permitirá a la otra estación enviar los datos que estaba enviando en un principio.

Ejemplo: Si A y B quieren transmitir a D al mismo tiempo



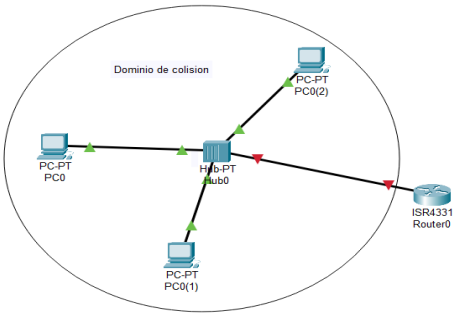
Ejemplo: Si A y B quieren transmitir a D al mismo tiempo. Suponiendo una eficiencia del 70% en una 10 base T



3.4 Ethernet conmutada

Como hemos visto, existe el problema que al comunicarnos dentro de una red solo puede transmitir un equipo a la vez. ¿Cómo solucionamos este problema? Principalmente mediante el uso de **switches**, aunque en XC también explican lo que son los bridges.

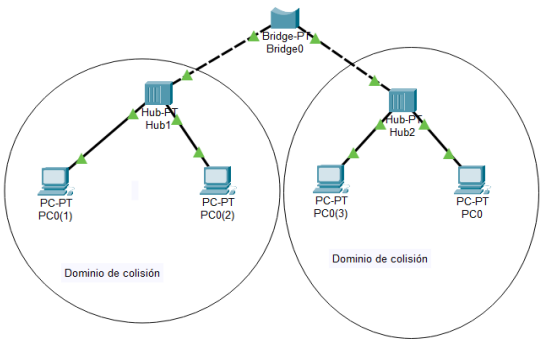
El esquema que tenemos ahora mismo es el siguiente cuando queremos representar una red Ethernet.



De esta forma, como podremos observar, solo un equipo puede transmitir a la vez, lo cual puede llevar a ser un gran inconveniente en una red local demasiado grande.

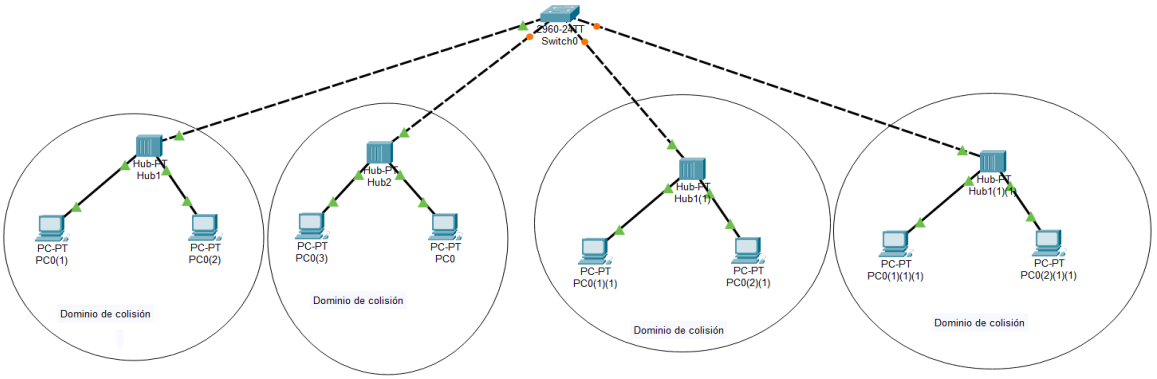
Bridges(puentes)

Un puente de red es un dispositivo que divide una red en segmentos. Cada segmento representa un dominio de colisión independiente, por lo que se reduce el número de colisiones en la red. Cada dominio de colisión tiene su propio ancho de banda independiente, por lo que un puente también mejora el rendimiento de la red.



Switch

Un conmutador de red es un dispositivo que conecta dispositivos en una LAN. Un conmutador es esencialmente un puente de red multipuerto y realiza las mismas funciones básicas que un puente, pero a velocidades mucho más rápidas y con muchas características adicionales. Cada puerto de un conmutador está en un dominio de colisión independiente y puede ejecutarse en full-duplex, lo que significa que los hosts conectados a algún puerto de un switch pueden transmitir al switch al mismo tiempo que el switch les transmite.



Funcionamiento:

- Cuando se recibe una trama con una dirección de origen que no está en la tabla, se agrega a la tabla MAC del switch.
- Si se recibe una trama con una dirección de destino que no esté en la tabla o se trate de una trama broadcast o multicast se copia la trama en todos los buffers de transmisión de los otros puertos.
- Si recibe una trama dirigida a otra estación desde el mismo puerto, se descarta esa trama.

En cuanto a los mensajes en broadcast, si no se usan VLAN's, tenemos que saber que un mensaje en broadcast se enviará por todos los puertos de un switch, la única forma de segmentarlo es mediante un router.

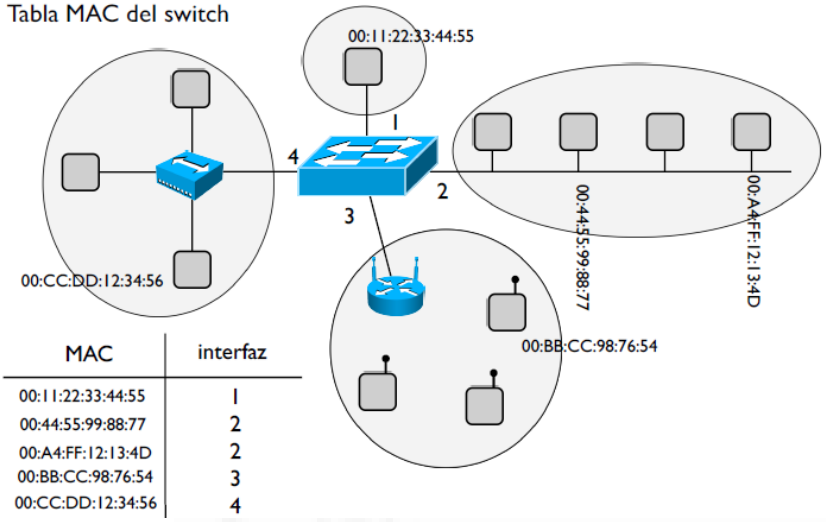
Control de flujo: Consiste en adaptar la velocidad a la que el interruptor recibe las tramas y la velocidad a la que el conmutador puede enviarlas.

Los conmutadores realizan el **control de flujo** de datos(ratio en el cual se envían y reciben tramas) mediante dos mecanismos dependiendo si se trata de una conexión half duplex o full duplex:

- **Jabber signal** (half duplex): El switch envía una señal al puerto que deben reducirse, de modo que CSMA vea que el medio está ocupado.
- **Pause frames** (full duplex); El switch envía tramas de pausa especiales. Estas tramas tienen un número entero (2 bytes) que indica el número de intervalos de tiempo (512 bits) que las NIC que reciben la trama deben estar en silencio.

Ejemplo:

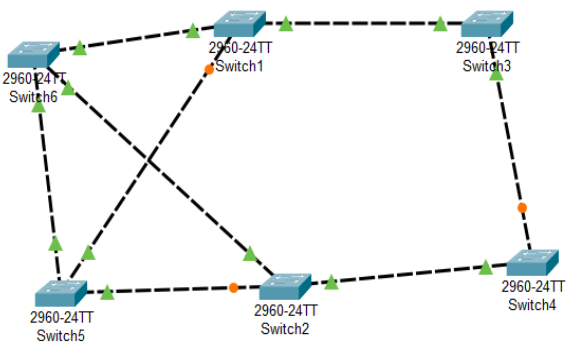
- Tabla MAC del switch

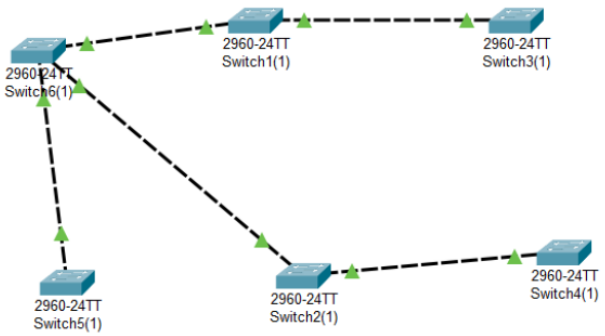


MAC	interfaz
00:11:22:33:44:55	1
00:44:55:99:88:77	2
00:A4:FF:12:13:4D	2
00:BB:CC:98:76:54	3
00:CC:DD:12:34:56	4

3.5 Spanning Tree Protocol (STP)

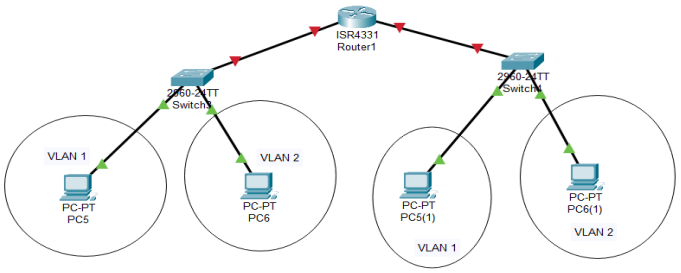
De cara al diseño correcto de interconexiones entre switches, hay que tener en cuenta que tenemos que tener una topología libre de bucles, esto se consigue mediante el **protocolo STP**(Spanning Tree Protocol). Este protocolo se encarga de conseguir una topología libre de bucles, dejando los posibles enlaces que pudieran causar algún conflicto en la red como enlaces de repuesto en caso de que algún otro enlace cayese.





3.6 Virtual LAN (VLAN)

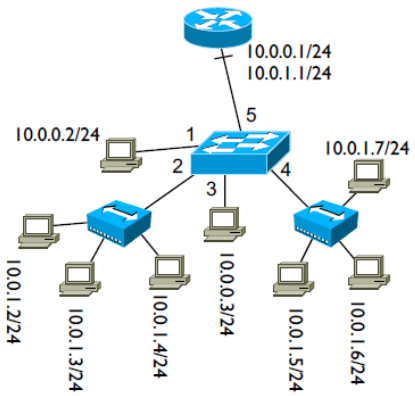
Las VLAN son un mecanismo que permite crear dominios de broadcast lógicos que pueden abarcar un único switch o varios, independientemente de la proximidad física. Esta función es útil para reducir el tamaño de los dominios de broadcast o para permitir que grupos o usuarios se agrupen lógicamente sin la necesidad de estar ubicados físicamente en el mismo lugar. Hay que tener en cuenta que cada puerto del switch indicará a que VLAN pertenece.

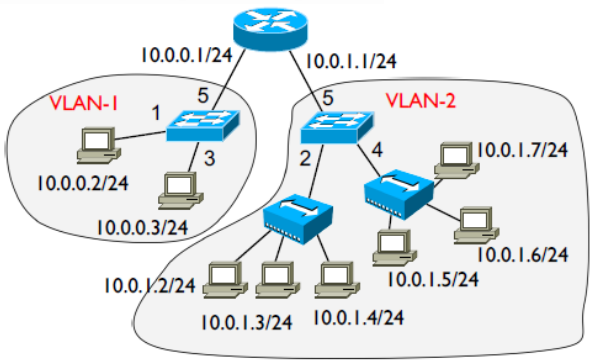


En este caso, al enviar un mensaje en broadcast, a pesar de estar separados por el router, si se envía un mensaje en broadcast desde VLAN1 desde el ordenador de la izquierda, este mensaje llegará al ordenador de la derecha. Tenemos que tener en cuenta que para disponer de esta distribución tenemos que tener un router que pueda comunicarse con las distintas VLAN's, el enlace entre el switch y el router se llamará enlace **trunk**.

NOTA: Dado que cada VLAN es un dominio de difusión diferente, por lo general, se utiliza una instanciación de STP diferente para cada VLAN. Por lo tanto, se construye un árbol STP diferente en cada VLAN.

Ejemplo:





VLAN	interfaz
VLAN-1	1, 3
VLAN-2	2, 4
trunk	5

3.7 Wifi

Protocolo utilizado para las redes inalámbricas, utilizan el protocolo 802.11x, dependiendo de la letra variará la velocidad de la conexión como se enseña a continuación:

802.11	Primera versión	1-2 Mbit/s	1997
802.11b	Comercial, WiFi	11 Mbits/s	1999
802.11g	WiFi de alta velocidad	54 Mbit/s	2003
802.11n	Mayor velocidad	hasta 600 Mbit/s	2009
802.11ac		hasta 1.3 Gbit/s	2013
802.11ax		hasta 10 Gbit/s	2019

A diferencia de las redes Ethernet, en las redes Wifi se utiliza otro método para tratar el problema de las colisiones, se utiliza el **CSMA/CA**. Este sistema funciona de forma que siempre se hace uso del tiempo de backoff, de esta forma se crea un sistema sin colisiones(CA viene de collision avoidance). Se utiliza este método ya que detectar colisiones en las redes inalámbricas puede llegar a ser muy difícil debido a la diferencia de magnitud en la transmisión y recepción respecto a las redes no cableadas.

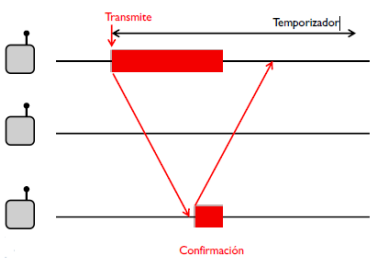
Las tramas Wifi toman la siguiente forma:

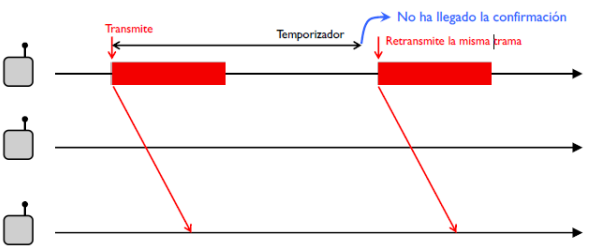
2 bytes	2 bytes	6 bytes	6 bytes	6 bytes	2 bytes	6 bytes	0-2312 bytes	4 bytes
Control de trama	Duración	MAC 1	MAC 2	MAC 3	Control de secuencia	MAC 4	Payload	CRC

Es importante notar que pueden haber hasta 4 direcciones.

3.8 CSMA/CA: Carrier Sense Multiple Access With Collision Avoidance

1. Si una estación quiere transmitir, escucha el medio
   - Si está ocupado va al punto 2).
   - Si está libre, espera un tiempo, transmite y va al 3)
2. Como el medio esta ocupado, genera un tiempo aleatorio(Backoff) y vuelve al 1)
3. Después de haber transmitido la trama entera, espera un tiempo
   - La estación espera que la estación destino le transmite una trama de vuelta que confirme que la trama se ha recibido correctamente
   - Si pasado este tiempo no recibe confirmación se vuelve al 1)





3.9 Problema del nodo oculto

El problema del nodo oculto nace cuando dos nodos que están en distinto rango de cobertura del AP entran en conflicto al enviar datos a la vez, dándose de esta forma una colisión. Para solucionar este problema se utiliza un mecanismo llamado RTS/CTS (solo se usa para unicast en Tx).

4.1 Introducción

En este tema trataremos el protocolo TCP y UDP, ambos corresponden a la capa de transporte y son los encargados de crear los canales de comunicación entre las aplicaciones(multiplexación de aplicaciones).

Esta multiplexación se realiza mediante los puertos, los puertos son identificadores de 16 bits(0-65535) y mediante este número identificamos la aplicación (esto también se ha visto en el tema de ACLs). Los puertos <1024 identifican a servicios conocidos y son conocidos como well-known ports mientras que el resto de puertos también se conocen como puertos efímeros y suelen servir para identificar clientes(PAT).
Esto funciona igual tanto para UDP como TCP.

En cuanto al tipo de modelo que utilizan, tanto TCP como UDP utilizan el modelo cliente/servidor.

Puertos importantes:
- ssh → 22 TCP
- telnet → 23 TCP
- FTP → 20/21 TCP
- http → 80 TCP
- https → 443 TCP
- DHCP → 67/68 UDP
- DNS → 53 UDP
- SMTP → 25 TCP
- RIP → 520 UDP

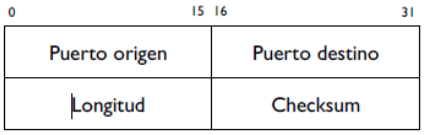
4.2 UDP (User Datagram Protocol)

Características principales de UDP

- No fiable: No hay forma de recuperar un paquete si se pierde
- No tiene recuperación de errores: es decir que no tiene ningún mecanismo que utilizando la secuencia de bytes pueda detectar un error y enviarlo en caso de que ocurra.
- No funciona con ack: Similar al punto anterior y siguiente, al no utilizar ningún tipo de ack, no puede tener ni recuperación de errores ni puede estar orientado a la conexión
- No orientado a conexión: esto quiere decir que no emplea ninguna sincronización entre el origen y el destino
- No tiene control de flujo: Como consecuencia de no tener control de flujo, los paquetes pueden llegar en cualquier orden.
- SOLO proporciona multiplexación de aplicaciones mediante puertos
- Usa el paradigma cliente-servidor
- Se usa para aplicaciones:
    - Que no necesitan fiabilidad porque envían mensajes periódicos: como pueden ser aplicaciones de comunicación como Skype, Discord o cualquier aplicación que tenga audio/vídeo ya que nos interesa mantener una conversación fluida aunque con algo menos de calidad que una llamada que se tenga que cortar(esto sería así con TCP).
    - Que son real-time

Indicar también que como UDP no tiene un buffer de transmisión como lo tiene TCP, no puede enviar bit a bit como lo hace TCP. UDP por su parte envía datagramas(bloques de bytes).

4.2.1 Cabecera UDP



- Puerto origen: identifica la aplicación origen
- Puerto destino: identifica la aplicación destino
- Longitud: tamaño del datagrama UDP, también incluye el tamaño de la cabecera.
- Checksum: control de errores en la cabecera y en el payload

4.3 Protocolo ARQ (Automatic Repeat reQuest)

Los protocolos ARQ son utilizados por el protocolo TCP como medida de control de flujo

- **Control de flujo**: El control de flujo se encarga de evitar que los buffers de Tx y Rx se sobrecarguen mediante distintos mecanismos.

- **Control de congestión**: El control de congestión se encarga de evitar sobresaturar la red, es decir, adaptando el tamaño de los datos, podemos conseguir evitar una utilización demasiado alta de la red .

Existen 3 protocolos de ARQ que se explican en XC.

- Stop & Wait
- Go back N
- Selective retransmission

Estos protocolos utilizan todos ack (acknowledgement), el ack son acuses de recibo, es decir, son segmentos que enviarán aquellos que reciban un mensaje para poder confirmar que se ha recibido correctamente este mensaje(por este motivo TCP es fiable).

Cabe destacar que la utilización de ack se utiliza con piggybacking, esto quiere decir que en vez de enviar un segmento solo con el ack, se envía el ack dentro del segmento TCP.

Estos protocolos también tienen otro campo llamado RTO(retransmission timeout) que es un temporizador que sirve para saber si se ha perdido un segmento, este temporizador se reinicia cada vez que se recibe un ack.

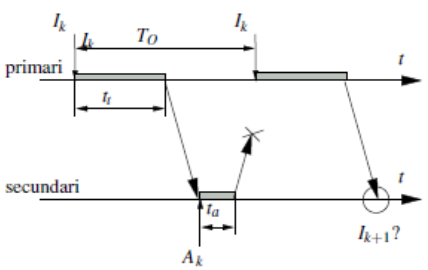
Para finalizar, también tenemos que entender que dependiendo del protocolo a utilizar, se puede utilizar una ventana de transmisión o no.

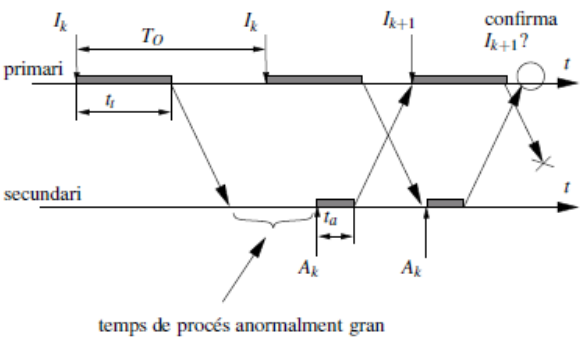
4.3.1 Stop & Wait

El protocolo de Stop & Wait es el más básico utilizado en los protocolos ARQ. Este protocolo funciona de la siguiente forma:

- El emisor envía un segmento al buffer de Tx y con esto se realiza la transmisión del segmento
- El receptor recibe el segmento y genera un ack que envía al emisor de vuelta.
- En cuanto el emisor reciba el ack, volverá al primer paso.

En caso de haber una pérdida después de superar el RTO (retransmission timeout), el segmento se reenvía

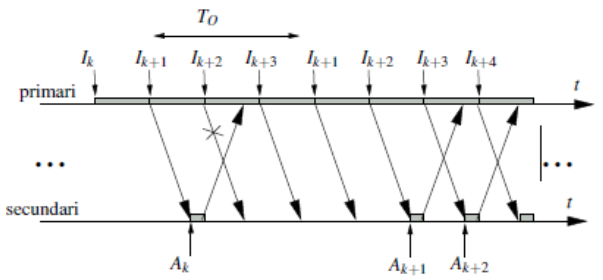




4.3.2 Go back N

Protocolo utilizado para mejorar la eficiencia del Stop & Wait.
Este protocolo se mantiene mediante una ventana, esta ventana indica el número de segmentos que podemos enviar a la vez.

- Utiliza ACK's acumulativos, esto quiere decir que el ACK del segmento K confirma los paquetes <=K. Si enviamos un ACK con número de secuencia 10, estamos confirmando todos los segmentos <=10(0,1,2,3...10).
- En caso de enviar y recibir todo correcto, avanzamos la ventana en el número que hemos asignado. Por ejemplo si tenemos una ventana de 10 segmentos y enviamos de ACK1-ACK10, podremos realizar el envio de los paquetes de ACK11-ACK20.
- En caso de haber un error en el envío o recibir una PDU fuera de orden, descarta todos los segmentos hasta que llegue el que tiene que llegar en orden.
- En caso de haber un RTO, vuelve atrás(go back) hasta el paquete que no se ha recibido correctamente y vuelve a enviar todo a partir de ese segmento.



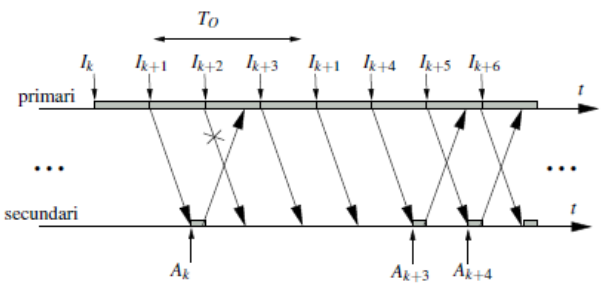
4.3.3 Retransmisión Selectiva

Este protocolo es muy similar a Go Back N, pero tiene ciertas diferencias para intentar mejorar la eficiencia.

- En este caso el emisor sólo retransmitirá si ha ocurrido un RTO.
- Para que la retransmisión selectiva funciona correctamente el receptor debe ser capaz de almacenar los segmentos fuera de orden, de esta forma, no hace falta descartar segmentos como se hace en Go Back N.

Funcionamiento

- Utiliza Ack's acumulativos, el Ack del segmento k confirma los paquetes <= k
- Si el secundario recibe una PDU (Ik) con errores o fuera de secuencia
  - Deja de enviar confirmaciones hasta que recibe correctamente la PDU que falta
  - Guarda todas las PDU que recibe con #secuencia ≠ k
- Cuando salta el temporizador, el primario retrasmite (Ik), pero no retransmite otras PDU que ya se habían enviado antes

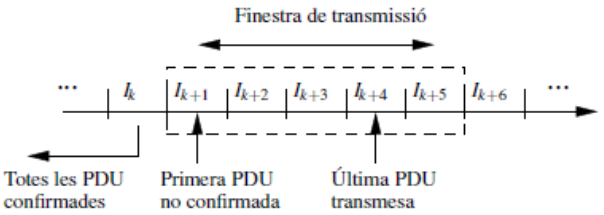


4.3.4 Protocolos con ventana

Este es un concepto que viene bien tener claro, y es entender que los protocolos de transmisión continua(Go Back N y Retransmisión selectiva) funcionan mediante ventanas. Por ejemplo, si en Go Back N queremos que se envíen paquetes de 5 en 5 es lo mismo que decir que tenemos una ventana de 5(W=5).

También podemos afirmar que Stop & Wait es un caso específico con una ventana de 1(W=1).

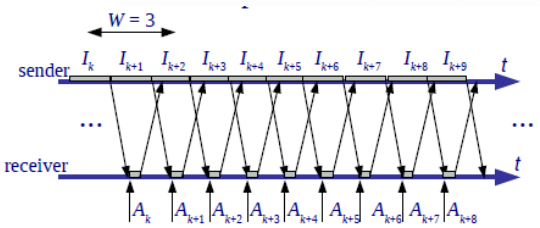
Esto nos permite también deducir que el buffer de transmisión ha de ser de máximo W para poder almacenar el burst (rafaga) de segmentos que vamos a enviar.



Ventana óptima

Definimos la ventana óptima como la ventana que nos permite conseguir la velocidad efectiva máxima utilizando el menor tamaño de ventana.

Si la ventana es muy pequeña, pasaría que no estaríamos utilizando el máximo de la velocidad efectiva, por otra parte el hecho de que sea muy grande no nos aporta ningún beneficio, ya que no podemos ir más rápido que la velocidad efectiva, por muy grande que sea la ventana.



$$W_{opt} = \left\lceil \frac{T_C}{t_t} \right\rceil$$

4.4 TCP (Transmission Control Protocol)

TCP es el protocolo orientado a conexión y fiable alternativo a UDP, está basado en el paradigma cliente-servidor.

- Orientado a conexión: A diferencia de UDP, TCP si está orientado a la conexión, esto quiere decir que antes de comenzar a transmitir la información, se encargará de realizar una conexión mutua entre cliente y servidor, este método se conoce como TWH(Three Way Handshaking).

- Fiable: Mediante distintos métodos TCP nos ofrece métodos para que en caso de que un segmento se pierda, poder recuperarlo de una forma segura y fiable.

- Control de congestión: TCP nos ofrece una solución a la congestión que pueda ocurrir en una red, hablamos de congestión cuando en una red está circulando/enviando más información de la que puede aguantar, ralentizando de esta forma la red.

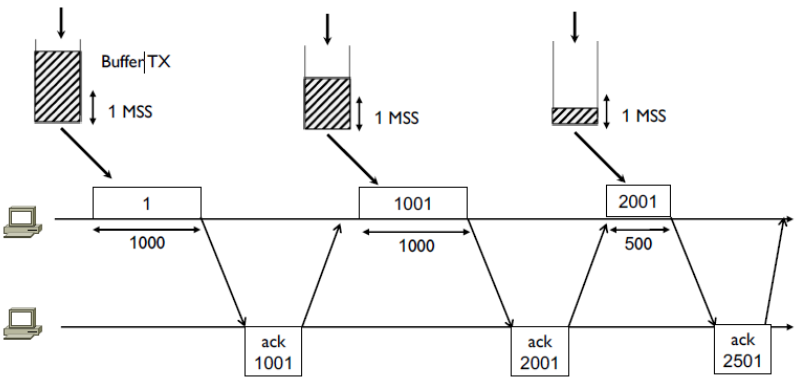
Además, también utiliza ACK en sus conexiones, de esta forma mantenemos un control sobre los segmentos.

En el intercambio entre segmentos en TCP tenemos dos ventanas, la ventana de congestión(cwnd) y la ventana anunciada(awnd). Protocolo de ventana ARQ, con ventana variable: wnd = min(awnd, cwnd)
La ventana de congestión se utiliza para el control de congestión como indica su nombre en el emisor mientras que la ventana anunciada se utiliza en el receptor para realizar el control de flujo.

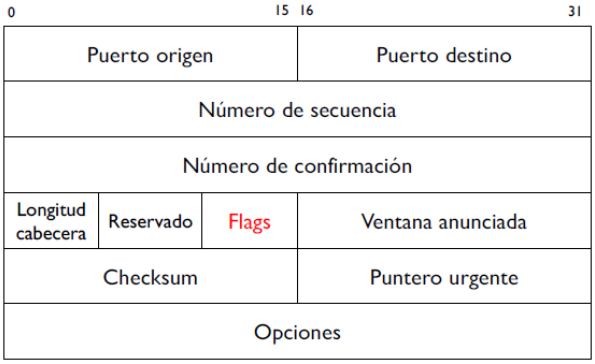
- Segmentos de tamaño óptimo(MSS)

- La unidad de información es el segmento

Normalmente se utiliza TCP:
- Aplicaciones que requieren fiabilidad: Web, ftp, ssh, telnet, mail,...



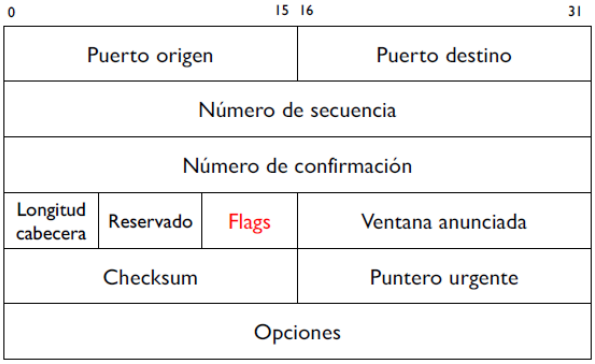
4.5 Cabecera TCP



- **Puerto origen y puerto destino**: identifican las aplicaciones (identificadas por puertos) tanto en origen como en destino.
- **Número de secuencia**: Identifica el primer byte del campo de datos.
- **Número de ACK**: Contiene el número del siguiente byte que está dispuesto a recibir.
- **Longitud de la cabecera**: Indica la longitud de la cabecera.
- **Reservado**: Campo reservado, se inicializa siempre a 0
- **Flags de TCP**: Campo de 6 bits donde cada bit señala una función específica del protocolo cuando está activo (en 1), las funciones son las siguientes:
  - URG: Existen datos urgentes que enviar, el puntero urgente indica la cantidad de datos urgentes que se encuentran en el segmento.
  - ACK: Se utiliza para reconocer los paquetes que el host ha recibido correctamente. El flag se setea si el campo del número de ACK contiene un número de ACK válido.
  - PSH: El flag PSH indica al receptor que tiene que procesar los segmentos a medida que son recibidos y que no se deben almacenar en el buffer.
  - RST: Se utiliza para terminar la conexión si el remitente RST siente que algo anda mal con la conexión TCP o que la conversación no debería existir. Puede enviarse desde el lado del receptor cuando el paquete se envía a un host en particular que no lo esperaba.
  - SYN: Se utiliza en el primer paso de la fase de establecimiento de la conexión o en el proceso TWH entre los dos hosts.
  - FIN: Se utiliza para solicitar la terminación de la conexión, es decir, cuando no hay más datos del remitente, se solicita la terminación de la conexión. Este es el último paquete enviado por el remitente. Libera los recursos reservados y finaliza la conexión.

FIN	RST
--> gracefully terminates the connection.	--> abruptly tells the other side to stop communicating.
--> Only one side of conversation is stopped.	--> The whole conversation is stopped.
--> No data loss.	--> Data is discarded.
--> Receiver of FIN keeps communicating till it wants to.	--> Receiver has to stop communication.

4.5 Cabecera TCP



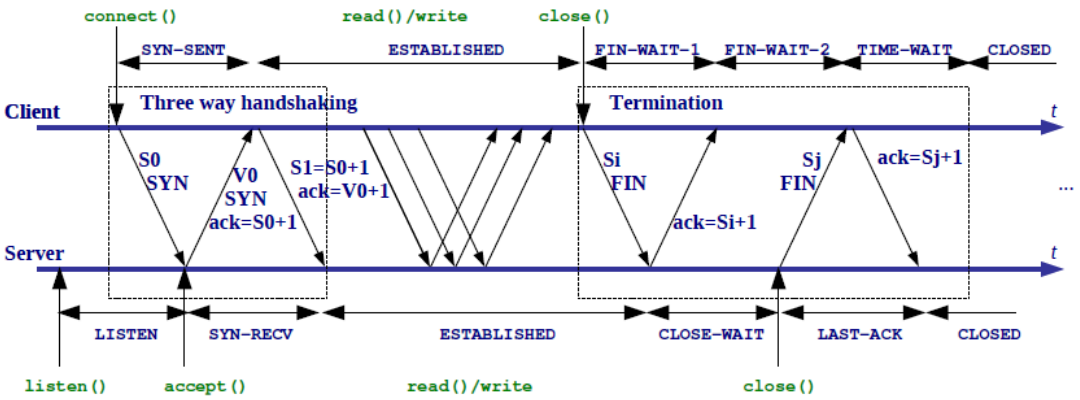
- **Ventana anunciada**: Indica cuántos bits componen la ventana de transmisión del protocolo de control de flujo por ventana deslizante.
- **Checksum**: Se utiliza para detectar errores
- **Puntero urgente**: Tiene sentido cuando el bit de control URG está activo e indica que los datos que envía el origen son urgentes. También identifica el último byte del campo de datos que es urgente.
- **Opciones**: Campo para añadir campos a la cabecera para realizar operaciones como indicar el tamaño máximo del segmento o como para indicar el factor de escalado.
    - Tamaño Máximo de Segmento (MSS): Usado en el TWH para inicializar el MSS.
    - Factor de Escala de Ventana(WSF): Utilizado en el TWH. El awnd se multiplica por $2^{\text{Window Scale}}$ (es decir, la escala de la ventana indica el número de bits para desplazar hacia la izquierda). Eso permite usar awnd de más de 2^{16} bytes.
    - Timestamp: se utiliza para calcular el tiempo de ida y vuelta (RTT).
    - Selective Ack(SACK): Permite que TCP haga transmisión selectiva.
- **Padding**: El padding se utiliza en el protocolo TCP para garantizarnos que la cabecera TCP finalice y los datos comiencen en el límite de 32 bits.

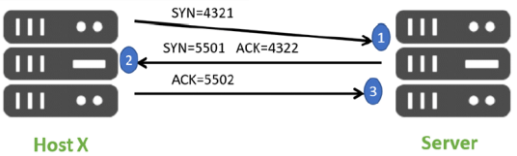
4.6 Three-way Handshaking

Paso 1 (SYN): En el primer paso, el cliente desea establecer una conexión con el servidor, por lo que envía un segmento con SYN (Sincronizar número de secuencia) que informa al servidor que el cliente desea iniciar una comunicación con el servidor, además envía el número de secuencia (este número nos sirve para saber los bytes que estamos enviando). Antes de iniciar este paso el cliente realiza una llamada del sistema connect() para iniciar la conexión, mientras que en el servidor estará la llamada listen() que significa que el servidor está escuchando.

Paso 2 (SYN + ACK): el servidor responde a la solicitud del cliente enviando un ACK con el número de secuencia que recibió+1 y con un número aleatorio como número de secuencia. ACK es la respuesta del segmento que recibió y SYN significa con qué número de secuencia es comenzarán los segmentos

Paso 3 (ACK): Finalmente, el cliente envía un ACK al servidor. El número de secuencia se establece como el ACK+1, y el ACK se establece en uno más que el número de secuencia recibido.



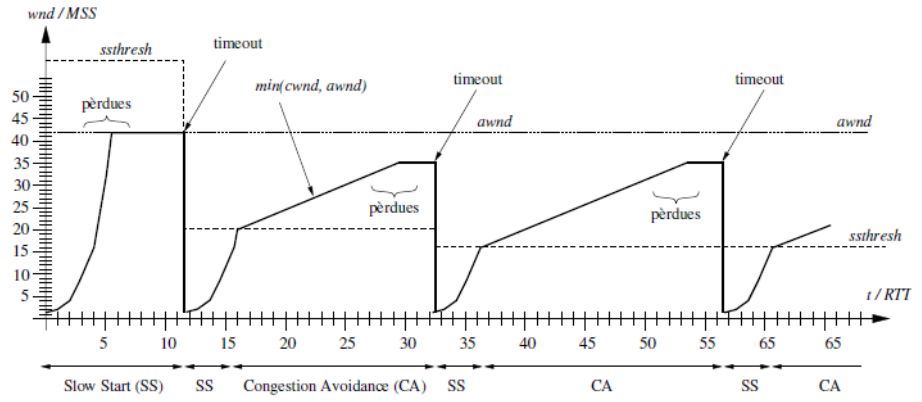


En este paso también se intercambia el **factor de escalado** del protocolo TCP. El campo de ventana anunciada tiene 16 bits (2^{16}-1) 65535 bytes. Este valor se creía suficiente, sin embargo se demostró más tarde que es un valor que no es lo suficientemente grande para el buffer de Rx, por lo que este campo nos permite aumentar el tamaño de la ventana anunciada.
El método es que se multiplica el valor de la ventana por 2 elevado al factor de escalado.
Es decir si tenemos una ventana de 3000 y tenemos un factor de escalado de 3 tendremos que hacer la siguiente operación: 3000 * 2^3 = 24000 awnd.

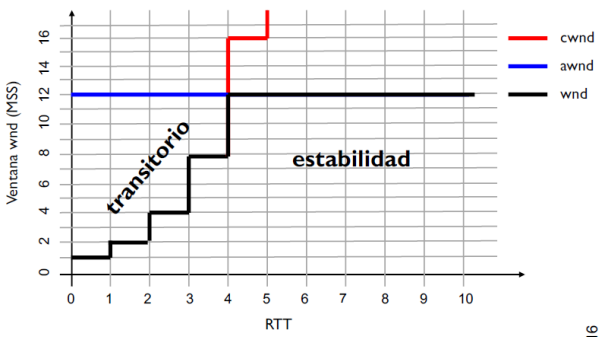
4.7 Slow Start / Congestión avoidance

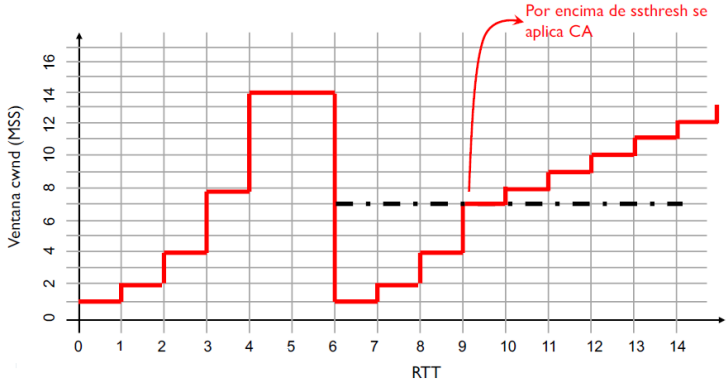
El slow start es parte de la estrategia de control de congestión utilizada por TCP en conjunto con otros algoritmos para evitar enviar más datos de los que la red es capaz de reenviar, es decir, para evitar causar congestión en la red. Este algoritmo es bastante agresivo como observaremos en las diapositivas, ya que la ventana de congestión se duplica por cada ack recibido.

```
Inicialització:
        cwnd = MSS ;
        ssthresh = infinit ;

Cada cop que es rep un ack de noves dades:
        si(cwnd < ssthresh) {
                cwnd += MSS ; /* Slow Start */
        } altrament {
                cwnd += MSS * MSS / cwnd ; /* Congestion Avoidance */
        }

Quan hi ha un timeout:
        Retransmetre snd_una ;
        cwnd = MSS ;
        ssthresh = max(min(awnd, cwnd) / 2, 2 MSS) ;
```



Ejemplo:

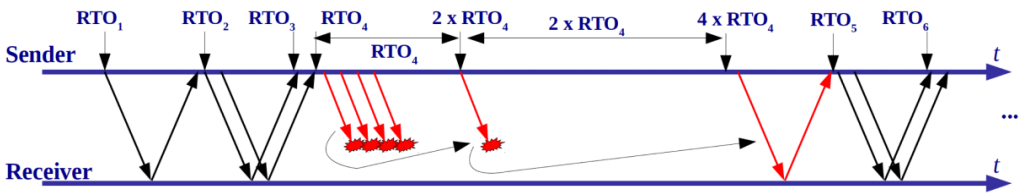




El RTO se inicializa cuando se envía el primer segmento. Cada vez que se recibe un ack nuevo y hay segmentos pendientes de confirmación, el RTO está activo y va disminuyendo con el tiempo. Cuando llega a 0, se retransmite el primer segmento pendiente de confirmación (el primero del buffer TX). Si no hay segmentos pendientes de confirmación, el RTO está desactivo.
RTO se computa con una formula:

- RTO = srtt + 4 rttvar
- Donde srtt es la media de los RTT y rttvar es la varianza

Cada vez que se pierde un segmento y salta el RTO, su valor se duplica



4.8 Tcpdump

```
12:30:37.069541 IP 147.83.34.125.17788 > 147.83.32.82.80: S 3473661146:3473661146(0) win 5840 <mss
                1460,sackOK,timestamp 296476754 0,nop,wscale 7>
12:30:37.070021 IP 147.83.32.82.80 > 147.83.34.125.17788: S 544373216:544373216(0) ack 3473661147 win 5792 <mss
                1460,sackOK,timestamp 1824770623 296476754,nop,wscale 2>
12:30:37.070038 IP 147.83.34.125.17788 > 147.83.32.82.80: . ack 1 win 46 <nop,nop,timestamp 296476754
                1824770623>
12:30:37.072763 IP 147.83.34.125.17788 > 147.83.32.82.80: P 1:602(601) ack 1 win 46 <nop,nop,timestamp 296476754
                1824770623>
12:30:37.073546 IP 147.83.32.82.80 > 147.83.34.125.17788: . ack 602 win 1749 <nop,nop,timestamp 1824770627
                296476754>
12:30:37.075932 IP 147.83.32.82.80 > 147.83.34.125.17788: P 1:526(525) ack 602 win 1749 <nop,nop,timestamp
                1824770629 296476754>
12:30:37.075948 IP 147.83.34.125.17788 > 147.83.32.82.80: . ack 526 win 54 <nop,nop,timestamp 296476755
                1824770629>
12:30:53.880704 IP 147.83.32.82.80 > 147.83.34.125.17788: F 526:526(0) ack 602 win 1749 <nop,nop,timestamp
                1824787435 296476755>
12:30:53.920354 IP 147.83.34.125.17788 > 147.83.32.82.80: . ack 527 win 54 <nop,nop,timestamp 296480966
                1824787435>
12:30:56.070200 IP 147.83.34.125.17788 > 147.83.32.82.80: F 602:602(0) ack 527 win 54 <nop,nop,timestamp
                296481504 1824787435>
12:30:56.070486 IP 147.83.32.82.80 > 147.83.34.125.17788: . ack 603 win 1749 <nop,nop,timestamp 1824789625
                296481504>
```

5.Introducción

Las aplicaciones de red son las aplicaciones que dan un servicio a través de una red que puede estar a nivel local (misma red) o en otra red distinta (Internet o una red diferente dentro de la misma organización).
Se incluyen en esta capa aquellos protocolos que hacen funcionar las aplicaciones } Sirvan para facilitar su funcionamiento, su aspecto de cara al usuario, su mantenimiento, etc. } Por ejemplo:

- RIP es una aplicación de red ya que facilita la construcción de las tablas de encaminamiento y su mantenimiento en los routers
- DHCP es una aplicación de red que permita la autoconfiguración de un host
- DNS es un sistema que facilita los usuarios ya que pueden tratar los hosts con nombres en lugar de con números (@IP)

En este tema nos centraremos en los siguientes 3 protocolos:

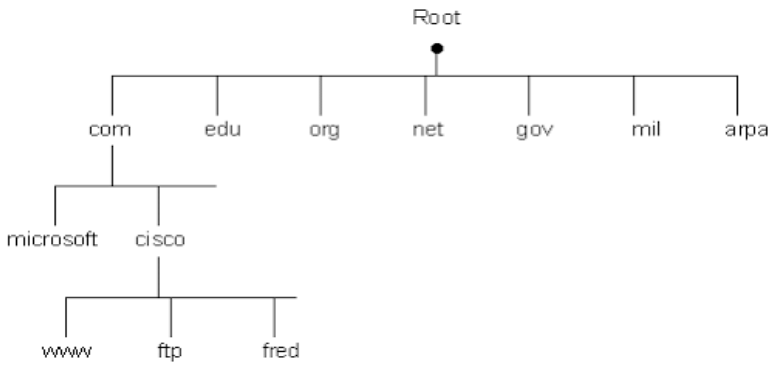
- DNS: Protocolo utilizado para traducir de IP's a nombres entendibles para las personas
- SMTP: Protocolo utilizado para enviar correos.
- HTTP: Protocolo utilizado para la comunicación web

5.1 DNS (Domain Name Server)

Es un protocolo utilizado para poder traducir las IP's a nombres entendibles. Utiliza UDP y viaja por el puerto 53

```
Haciendo ping a www.google.es [172.217.17.3] con 32 bytes de datos:
Respuesta desde 172.217.17.3: bytes=32 tiempo=12ms TTL=114
Respuesta desde 172.217.17.3: bytes=32 tiempo=12ms TTL=114
Respuesta desde 172.217.17.3: bytes=32 tiempo=12ms TTL=114
```

En este ejemplo podemos ver como al hacer ping a www.google.es realmente no hacemos un ping a google.es, sino que hacemos un ping a su dirección IP, en este caso la 172.217.17.3 .



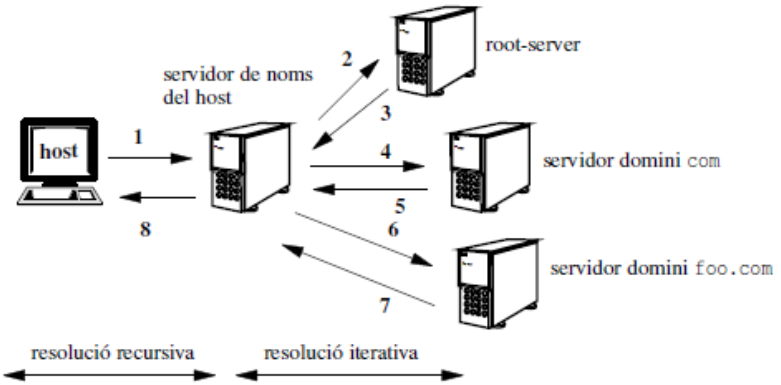
Aquí podemos ver un ejemplo de cómo están formados los distintos nombres de dominio. Siempre se empieza por el root(de este servidor hay multitud de servidores para facilitar el acceso). En el siguiente nivel tenemos lo que se conoce como TLDs(Top Level Domain), son los dominios más altos después del root y estos ya derivan en las distintas organizaciones, en este caso por ejemplo hablamos de cisco. De esta forma ya tendríamos la dirección cisco.com(realmente sería cisco.com. pero el . de root siempre es implícito y no hace falta que lo pongamos en el navegador). Ahora querríamos entrar al nodo(hostname) fred para que nos quede la dirección fred.cisco.com. Un hostname es un nombre que identifica un equipo dentro de una red.
Podemos decir entonces que estamos accediendo al dominio cisco.com y al nodo fred.

Para poder realizar esta traducción, cada vez que enviemos una petición DNS, iremos a un servidor DNS que contendrá unos registros donde estará la información de lo que estamos pidiendo, estos registros tienen el nombre de Resource Records(abreviado RR).

Por lo tanto, cada servidor de DNS estará compuesto de un conjunto de RR's donde contendrá diversa información. Estos RR's pueden ser estáticos(permanentes) o pueden ser cacheados y que desaparezcan pasado un tiempo.

Cabe destacar que cada NS se encargará de su dominio(también conocida como zona) y tendrá como mínimo 2 servidores, uno principal y uno de backup(usa TCP cuando se hacen backups). También tenemos que saber que un NS puede delegar parte de su zona a otro NS, esto se conoce como delegación de zona y la zona delegada pasa a ser llamada subzona.

¿Cómo se realiza una resolución DNS?



Para empezar, para realizar una petición DNS el cliente siempre hará una petición recursiva, esto quiere decir que delegará todo el trabajo sobre su servidor de DNS, por lo tanto, el cliente realiza la petición a su servidor y espera en respuestas la dirección IP correspondiente al nombre que ha solicitado. Por su parte el servidor DNS hará peticiones iterativas a cada NS para poder llegar finalmente a la dirección que está buscando.
Como último apunte cabe destacar la existencia de los CDN. Estos son servidores espejo que sirven para poder acceder más rápidamente a la información ya que accedemos a los servidores más cercanos que tengamos, de esta forma también creamos un balanceo de carga ya que las peticiones no acaban yendo todas al mismo servidor.

5.2 Formato mensaje DNS

```
+-----------------------------------------+
|            Header (12 bytes)           |
+-----------------------------------------+
/            Question (variable)          /
+-----------------------------------------+
/            Answer (variable)            /
+-----------------------------------------+
/            Authority (variable)         /
+-----------------------------------------+
/            Additional (variable)        /
+-----------------------------------------+
```

Header: Indica el tipo de mensaje que enviamos

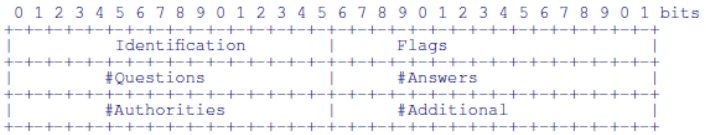
Question: ¿Qué es lo que intentamos resolver?

Answer: Respuesta a lo que intentamos resolver

Authority: Las autoridades que hemos tenido que contactar para poder llegar a la resolución que necesitábamos

Additional: Campo con información adicional que suele contener las IP's de las autoridades

5.2.1 Header



Tiene un tamaño fijo de 12 bytes

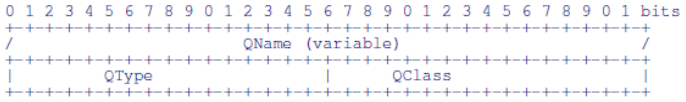
Identification: permite relacionar los mensajes de Query y Reply, es un número aleatorio de 16 bits.

Flags: algunos de ellos son:

- QR (Query-Response): 0 si es Query o 1 si es Response
- AA (Authoritative Answer): si ha respondido la autoridad o estaba en caché
- RD (Recursion Desired): indica que se desea la resolución recursiva

El resto de campos indican el número de preguntas, respuestas, autoridades y adicional que hay en el resto del mensaje

5.2.2 Question



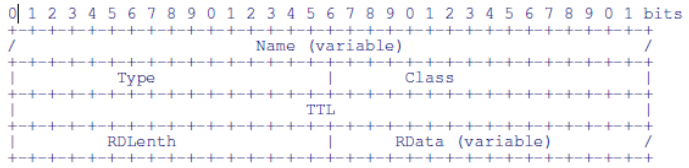
QName: indica el nombre que se quiere resolver

QType: indica el tipo de pregunta

- A: para conocer una @IP a partir de un nombre
- NS: para conocer el nombre de un NS de un dominio
- PTR: para conocer un nombre a partir de una @IP
- MX: para conocer el nombre del server de email de un dominio
- CNAME: para conocer todos los alias de un nombre

QClass: para direcciones de Internet Ipv4 es 1

5.2.3 Answer, authority y additional



Name, Type y Class: Es lo mismo que en Question

TTL: número de segundos que hay que guardar la resolución en la caché RR

RDLenth: tamaño de la resolución en bytes

RData: la resolución

RR format: `(name, value, type, ttl)`

Tipo A
- `name` es el hostname
- `value` es la @IP

Tipo CNAME
- `name` es un alias
- `value` es el nombre canonico
  p. e., `www.ibm.com` es un alias de `servereast.backup2.ibm.com`

Tipo NS
- `name` es el dominio (p.e., foo.com)
- `value` es el nombre del autoridad DSN del dominio

Tipo MX
- `name` es el dominio (p.e., foo.com)
- `value` es el nombre del servidor de correos de este dominio

Ejemplo:

```
$ORIGIN tld.
$TTL 5m

example         IN      SOA     ns1.example.tld. hostmaster.example.tld. (
                                90              ; serial
                                4h              ; refresh
                                15m             ; retry
                                8h              ; expire
                                4m)             ; negative caching TTL
                IN      NS      ns1.example.tld.
                IN      NS      ns2.example.tld.
                MX      10      mail.example.tld.
                IN      A       127.0.0.1

                IN TXT "v=spf1 a mx a:mail.example.tld a:www.example.tld ?all"

$ORIGIN example.tld.

ns1                     IN      A       127.0.0.2
ns2                     IN      A       127.0.0.3
www                     IN      CNAME   example.tld.
mail                    IN      AAAA    ::1
mail                    IN      A       127.0.0.4
```

Campos a destacar:

El campo de la izquierda es el nombre al que nos referimos, en la primera línea se refiere al nombre del dominio.
Los campos de clase pueden ser de tipo IN o MX, el primero significa indica que trabajamos en internet(este campo siempre es el mismo, antiguamente podía ir HS que significaba Hesiod, un antiguo estándar utilizado por IBM en su proyecto Athena) mientras que el segundo indica que se trata de un servidor de correo.

SOA o Start Of Authority es un campo que sirve para indicar el servidor de nombres primario(ns1.example.tld) del dominio, el correo de contacto del administrador(hostmaster.example.tld) y campos de configuración(serial, refresh, retry...).

NS indica un servidor de nombres

A indica una dirección IPv4 mientras que AAAA indica una dirección IPv6

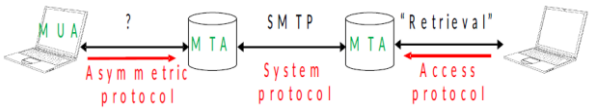
Al campo CNAME indica un alias, en nuestro ejemplo www.example.tld es un alias para example.tld

5.3 SMTP (Simple Mail Transfer Protocol)

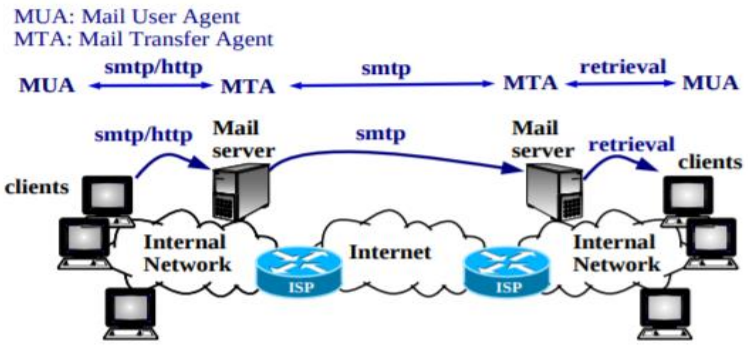
SMTP es el protocolo utilizado para el envío de correos, viaja mediante el puerto 25 por TCP. Las direcciones de correo se componen con un nombre seguido de un @ y el nombre de dominio donde se aloja el servidor de correo, ejemplo: aidan@foo.org (nombre aidan y dominio foo.org).

El protocolo tiene dos componentes esenciales que son:

- MUA: Mail User Agent, el MUA es el encargado de enviar el correo desde la máquina cliente al servidor de correos. Ejemplo: Outlook, Thunderbird, etc...
- MTA: Mail Transfer Agent, el MTA es el encargado de la comunicación entre servidores de correos. Ejemplo: Postfix, Sendmail, qman, etc...

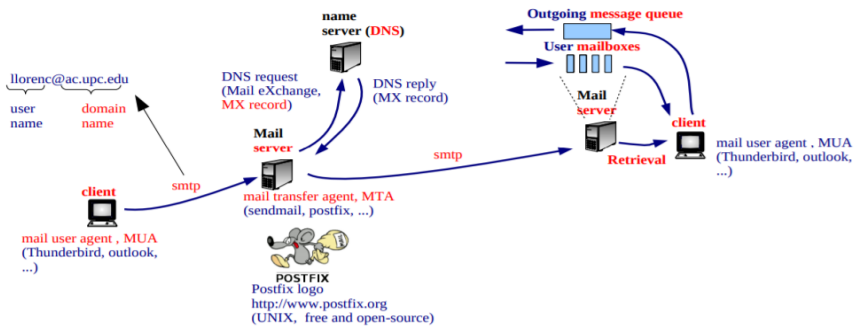


- Protocolos de Recuperación (Retrieval): IMAP, POP, HTTP



Como podemos observar en la imagen, la comunicación se realiza primera con el MUA enviando el correo al servidor MTA, este a su vez envía el correo al servidor de correos destino(en caso de que se envíe al mismo servidor de correos, este paso no sería necesario) y finalmente se entrega al cliente destino

Esquema de la comunicación con SMTP



El primer paso es que el cliente envíe el correo a su servidor de correos(el MUA se encarga de esto), una vez llegue el correo al servidor de correos, este ha de identificar a qué dirección de correo quiere enviar el correo origen. En caso de ser el mismo servidor(de aidan@foo.org a nico@foo.org por ejemplo), simplemente ha de redirigir el correo y no hace falta hacer ninguna resolución con DNS.
En caso de enviar un correo a un servidor distinto(de aidan@foo.org a nico@bar.org), el servidor de correos ha de realizar una resolución DNS(a bar.org) buscando el servidor de correo destino(RR de tipo MX), cuando tengamos la dirección del servidor destino, podemos enviar el correo al servidor destino y este finalmente se encargará de enviar el correo al usuario destinatario.

El protocolo utiliza ASCII para codificarse y funciona mediante unos comandos muy simples:

- HELO: Para identificar el cliente SMTP
- MAIL FROM: Remitente del correo
- RCPT TO: Destinatario del correo
- DATA: Mensaje del correo
- QUIT: Comando utilizado para indicar que se ha acabado el envio del correo

Ejemplo:

```
S: 220 smtp2go.com ESMTP Exim
C: HELO mydomain.com
S: 250 Hello mydomain.com
C: MAIL FROM:<sender@mydomain.com>
S: 250 Ok
C: RCPT TO:<recipient@anotherdomain.com>
S: 250 Accepted
C: DATA
S: 354 Enter message, ending with "." on a line by itself
C: Subject: sample message
C: From: sender@mydomain.com
C: To: recipient@anotherdomain.com
C:
C: Greetings,
C: Typed message (content)
C: Goodbye.
C: .
S: 250 OK
C: QUIT
S: 221 www.sample.com closing connection
```

Ejemplo de envío de correo:

Como anotación, podemos observar que el servidor responde siempre que hacemos un comando. El código más importante es el 250 ya que indica que todo ha ido correctamente en cuanto a la ejecución del comando.
Además de esto, también hay que destacar que cuando queremos acabar el contenido del correo(DATA) tenemos que poner un . y un salto de línea.

5.4 MIME (Multipurpose Internet Mail Extensions)

MIME es un mecanismo(que no protocolo) utilizado para poder realizar intercambio de información que no sea solo texto en los correos. Con esto conseguimos poder enviar ficheros tales como audios, fotos, videos, y también enviar caracteres de otros idiomas(chino, cirílico, árabe, etc...)

MIME utiliza un delimitador(boundary) para delimitar cada parte del mensaje, de esta forma tenemos distintos tipos de ficheros a enviar.

El mensaje se compone de :

- Cabecera: incluye la definición de una “palabra” que hace de frontera
- Cuerpo: puede contener datos de diferentes formatos

En la siguiente tabla se puede observar ejemplos de los tipos que se pueden utilizar en MIME

Type	Subtype	Description	File extensions
Application	postscript text	Printable postscript document TEX document	.eps, .ps .tex
Audio	midi realaudio wav	Musical Instrument Digital Interface Progressive Networks sound Microsoft sound	.midi, .mid .ra, .ram .wav
Image	gif jpeg png	Grapical Interchange Format Joint Photographic Experts Group Portable Network Graphics	.gif .jpeg, .jpg, .jpe .png
Model	vrml	Virtual Reality Modeling Language	.wrl
Text	html plain	Hypertext Markup Language Unformatted text	.html, .htm .txt
Video	avi mpeg quicktime	Microsoft audio video interleaved Moving Picture Experts Group Apple QuickTime movie	.avi .mpeg, .mpg .qt, .mov

Ejemplo:

```
To: dest@upc.edu
From: The sender <sender@isoc.org>
Subject: test
Message-ID: <5aedea3f@isoc.org>
Date: Wed, 11 Dec 2019 15:15:25
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="16A5"

This is a multi-part message in MIME format.
--16A5
Content-Type: multipart/alternative; boundary="65E8"

--65E8
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: quoted-printable

A _test_

--65E8
Content-Type: multipart/related; boundary="ECB3"

--ECB3
Content-Type: text/html; charset=utf-8
Content-Transfer-Encoding: 8bit

<html><body><p>A <u>test</u> <img src="cid:part1.29EE"></p></body> </html>

--ECB3
Content-Type: image/png
Content-Transfer-Encoding: base64
Content-ID: <part1.29EE>
Content-Disposition: inline; filename="1x1.png"
iVBORw0KGgoAAAANSUhEUgAAAAEAAAABAQMAAAAl21bKAAAAA1BMVEUAAACnej3aAAAAAXRS
TlMAQObYZgAAAApJREFUCNdjYAAAAAIAAeIhvDMAAAAASUVORK5CYII=
--ECB3--

--65E8--

--16A5
Content-Type: application/pdf
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="test.pdf"
JVBERi0xLjUKJcOkw7zDtsOfCjIMCBvYmoKPDwvTGVuZ3RoIDMgMCB0ZpbHRMQolJUVPRgo=
--16A5--
```

5.5 POP (Post Office Protocol)

Actualizado a la versión 3 (POP3). El servidor guarda en el servidor los mensajes del usuario hasta que este se conecta, en ese momento se descargan todos los mensajes del servidor al equipo del usuario y se borran del servidor.
Se puede configurar para que el servidor guarde una copia de los correos durante un cierto tiempo (puede ser infinito)
Viaja por TCP por el puerto 110.

5.6 IMAP (Internet Message Access Protocol)

A diferencia de POP3, no hace falta descargar los mensajes y además nos da la posibilidad de agrupar los correos mediante carpetas entre otras funciones típicas de un correo moderno.
Se puede acceder a los correos desde cualquier dispositivo ya que no se guardan en local.
Viaja por TCP por el puerto 143.

5.7 HTTP

Funciona igual que IMAP en cuanto al tratamiento de mensajes(no hace falta descargarlos y las funciones agregadas), la diferencia es que se accede mediante un cliente HTTP(Mozilla, Chrome, Safari, etc…)

HTTP es el protocolo utilizado en la web para realizar la comunicación en la web para poder realizar transferencia/visualización de información en la web.
Como vimos en el primer tema, está basado en el paradigma cliente/servidor. Con esto queremos decir que tendremos un cliente que quiere acceder a una web y esta web estará hospedada en un servidor que ofrecerá el servicio web a los clientes.

Métodos
HTTP funciona mediante peticiones y respuestas. Este sistema se realiza mediante métodos y estos métodos consisten en distintas acciones que puede realizar un cliente a un servidor HTTP. Los dos principales son los siguientes:

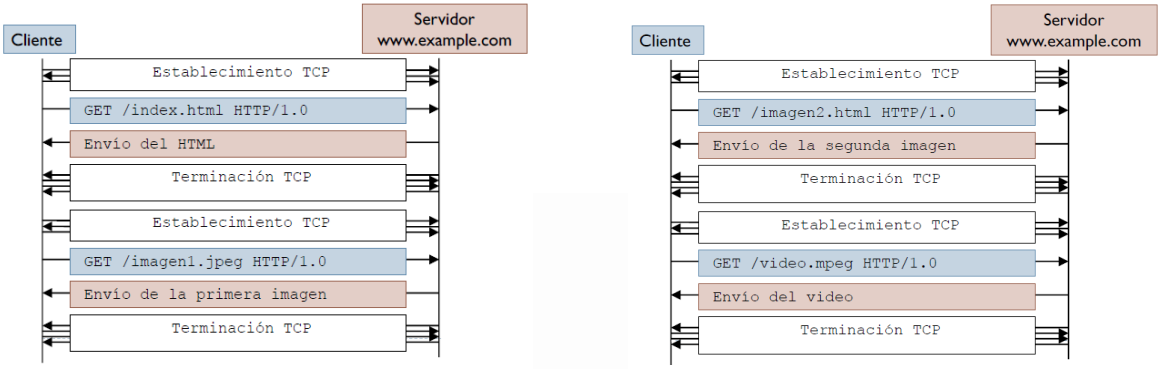
- **GET**: Método utilizado para obtener información de una página web
- **POST**: Método utilizado para poder subir información a una página web, el ejemplo más básico son los formularios.
- **HEAD**: Mismo utilización que el GET, pero en este caso no nos devuelve el cuerpo de la información, solo las cabeceras.

Otros métodos de HTTP 1.1:

- OPTIONS: Para indicar las posibles opciones aceptadas por la página web
- PUT: Para reemplazar/insertar un recurso de red/objeto.
- DELETE: Borra un recurso de la página web
- PATCH: Modifica un objeto existente
- TRACE: Funciona mediante ecos para comprobar conexión
- CONNECT: Establece un túnel, se suele utilizar para implementar SSL(conexión segura/HTTPS)

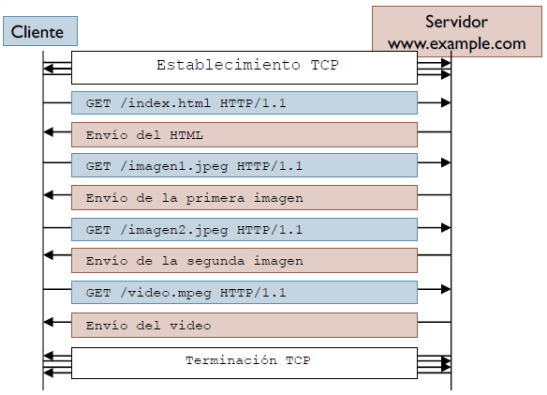
5.7.1 No persistente

Por defecto en HTTP 1.0, el servidor envía el objeto solicitado y cierra la conexión, si hay más objetos en una página, hay que abrir una conexión para cada uno.



5.7.2 Persistente

Por defecto en HTTP 1.1, el servidor mantiene la conexión abierta durante un tiempo determinado. Si hay más objetos se pueden pedir uno después del otro.



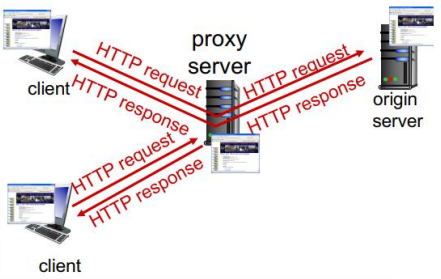
5.7.3 Persistente con pipeling

Solo en HTTP 1.1 pero uso no recomendado, el servidor mantiene la conexión abierta durante un tiempo determinado y el cliente puede pedir nuevos objetos en el momento en que encuentra nuevas referencias en el objeto que se está bajando aunque la descarga no se haya completado (transmisiones en paralelo).

5.8 Proxy

El objetivo de añadir un proxy es contestar a los clientes sin que estos accedan al servidor, se usa como intermediario y los clientes se comunican con el proxy.
El proxy accede al servidor y almacena las páginas web solicitadas y sus objetos temporalmente en su caché local
Ventajas de usar un proxy:

- Seguridad
- Rapidez
- Menor carga en los servidores
- Servidor sin acceso a Internet(@IP privada)



5.9 Charsets

Se utiliza para representar caracteres en formato electrónico.
Ejemplo:
email y web: el charset se especifica usando MIME
Content-Type: text/plain;
charset = UTF-8
Típicamente se negocia el idioma y charset a usar en los protocolos de nivel aplicación

- Unicode
  Caracteres de todos los idiomas + matemáticas + emoticonos + . . .
  Unicode (6/2017): 136.755 caracteres
  notación: U + hex. P.ej. U+0020 = ' ' (espacio en blanco)
  los primeros 128 códigos corresponden a ASCII
  codificación: Universal Transformation Format (UTF)
  Unidades de código UTF: UTF-8, UTF-16, UTF-32. Número de bits de las unidades de codificación.
  – UTF-8: de uno a cuatro unidades de código de 8 bits.
  – UTF-16: una o dos unidades de código de 16 bits.
  – UTF-32: unidades de código de 32 bits de longitud fija.

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

www.asesacademia.com

ASES

assessoria d'estudis superiors