**Direcciones IP**

Las MAC son direcciones físicas, las IP son direcciones lógicas.

El número telefónico sería equivalente a la dirección lógica, mientras que el número de serie que tiene el teléfono es equivalente a la dirección MAC.

Para comunicarse uno necesita el número lógico, y no necesita el número de serie.

**IPv4 (version 4)**

Es el IP que está actualmente en uso, pero ya quedo escasa para lo que se quiere hacer. Es un IP de 32 bits, lo que significa que puede haber 2^32 direcciones distintas.

Tiene que ser universal, osea la identificación es única y no puede haber dos máquinas con igual direcciones IP.

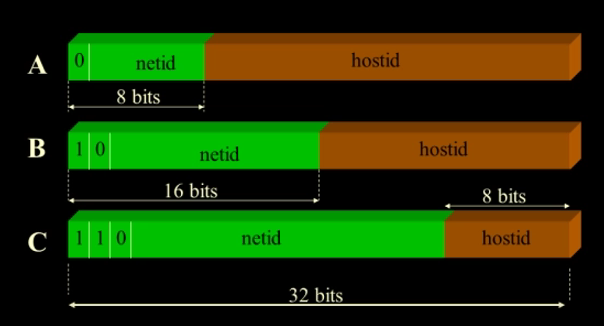
Esta dirección de 32 bits nos indica como identificador:

* El nombre.
* La dirección, ósea donde está.
* La ruta. Osea cómo llegar.

Se dividen en dos partes:

* Red
* Host

**Direcciones clase A, clase B y clase C**



* Si el primer octeto corresponde a la red, es Clase A.
* Si corresponden los 2 primeros octetos a la red, es Clase B.
* Si corresponden los 3 primeros octetos a la red, es Clase C.

Los de **Clase A** siempre empiezan con 0, lo que nos deja 7 bits de los 8 que tenemos para la dirección de red.

2^7 = **128** (osea puede haber 128 redes Clase A)

En la **Clase B** vemos que están viejos el 1 y 0 para identificarlo, lo que nos deja 14 bits de los 16 para las direcciones.

2^14 = **16,384** (osea puede haber 16,384 redes Clase B)

El de la **Clase C** tiene 1 1 0 como identificadores (3 bits), lo que nos deja 21 de los 24 bits para direcciones.

2^21 = **2,097,152** (osea puede haber 2,097,152 redes Clase C)

Y para identificar los host, tenemos la inversa

**Clase A** 24 bits osea (2^24 = **16,777,216** cantidad de máquinas que puede tener)

Clase B 16 bits osea (2^16 = **65,536** cantidad de máquinas que puede tener)

Clase C 8 bits osea (2^8 = **256** cantidad de máquinas que puede tener)

Pero esto no es COMPLETAMENTE cierto por que hay ciertas direcciones que no se pueden usar.

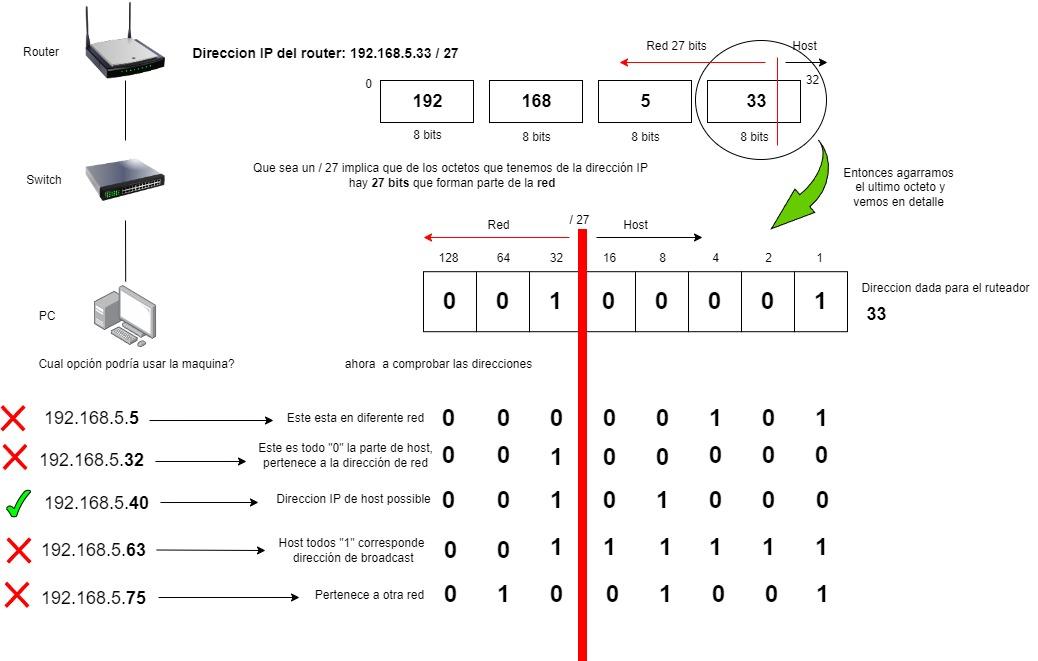
En definitiva, que sea clase A, B, o C nos da la cantidad de máquinas que una red puede tener y cuántas redes hay. Por ej: en clase hay poca cantidad de red y enorme cantidad de máquinas.

Las direcciones NO especifican una computadora, si no una conexión a la red.

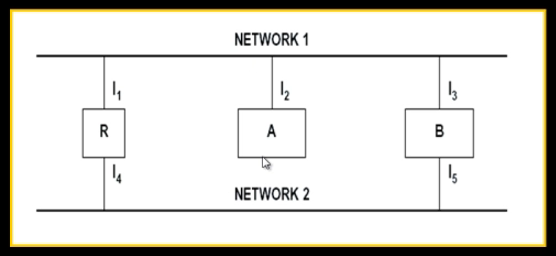
Cuando la parte marrón (hostid), osea la máquina dentro de la red toma todo ‘0’ significa que nos estamos refiriendo a la red, y no a una máquina.

Y cuando toma valores de todo ‘1’ osea 255 nos estamos referido a la dirección de difusión, que habla a todas las redes de esa máquina.

El ruteo se utiliza el netid (parte de red en verde).



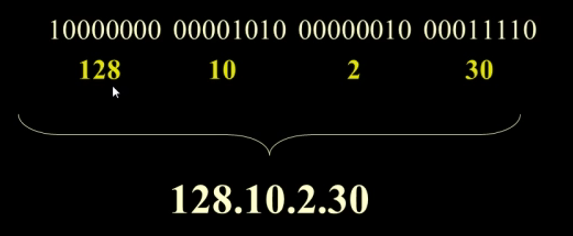
**Computadoras Multi homed**



Tenemos Red1 y Red2, la máquina A está conectada con Interfaz 2 a la red 1, mientras que la máquina B es multihome ósea está conectada con la Red1 y Red2.

Por otro lado, tenemos el router conectado a las dos redes, pero que está puesto para permitir el pasaje de paquetes entre las redes.

**Notacion (ejemplo de dirección IPv4)**

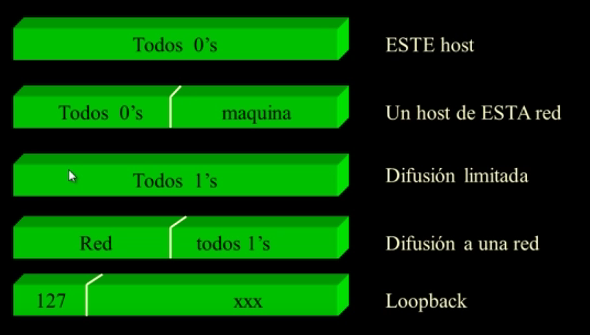


32 bits y para facilitar la lectura se pone en decimal (128.10.2.30)

Como vemos que es de 128 osea Clase B y también porque lo vemos en binario (empieza con 1 y 0)

Por lo tanto, la parte de RED es 128.10 y la parte de HOST es 2.30

**Direcciones especiales**



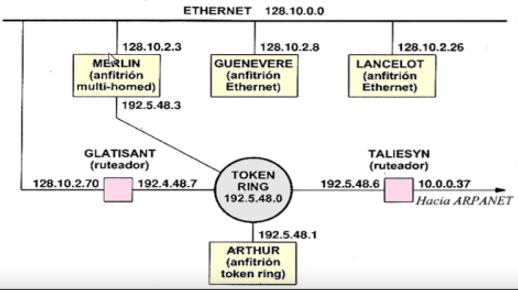
**Todos 0 -** se refiere al HOST en la que estamos parados en ese momento.

**Parte de RED es 0 -** Está refiriéndose a un HOST que está en la RED que estamos parados.

**Todos 1 -** Difusion.

**Parte de HOST todo 1 -** Se refiere a la difusión dentro de la red indicada.

**Comienza con 127 -** Dirección de Loopback.



Tenemos la red Ethernet arriba de clase B y tiene 3 máquinas y como la red es de 128.10, por lo tanto todas las máquinas conectadas a esa red empiezan en 128.10.

La multi homed (MERLIN) tiene una parte de red 128.10 conectada a la Ethernet, y una parte de red 192.5.48 conectada a la token ring.

El token ring es una Clase C, por lo tanto, los 3 primeros octetos son los de red.

El ruteador GLATISANT tiene 2 interfaces, una conectada a cada red. Tiene un error en el IP **192.4.48.7**, el 4 debería ser un 5

**IPv6 (version 6)**

Direcciones más grandes de 128 bits, surgió porque en 3 de febrero 2011 se terminaron las direcciones IP versión 4.

**Ventajas más evidentes:**

* Gran espacio de direcciones (Evita el uso de NAT)
* Más rápida, menos procesamiento
* Mejoras en la seguridad e integridad de datos.

**Forma de representar los 128 bits:**

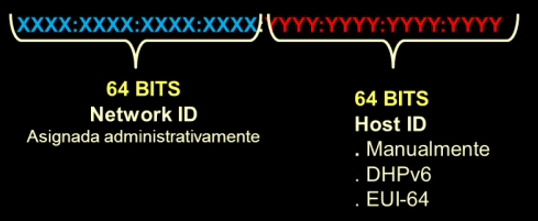
* 8 grupos de 16 bits en hexa

**Ejemplo de IP versión 6:**



Se está haciendo omisión de ceros, osea en ves de poner muchos ceros seguidos en el caso de que haya, se pone los ‘**:**’ nomás.

**Unicast**



64 bits de parte de la RED, y 64 bits de parte del HOST.

Las **Y** se pueden asignar manualmente o en forma automática.

**Tipos de direcciones IPv6:**

* Unicast
* Multicast
* Anycast

**Protocolo ARP (Address Resolution Protocol)**

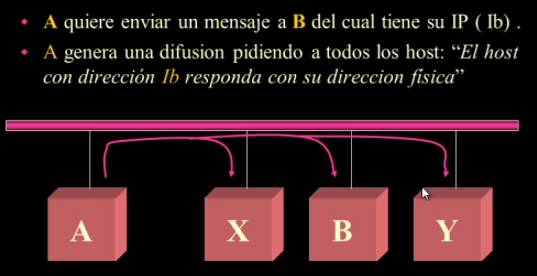
Vamos a usar este protocolo únicamente para la IP versión 4.

Nosotros queremos comunicarnos con una máquina a la que conocemos su dirección lógica, pero no conocemos la dirección MAC.

En las redes, hace falta conocer la dirección MAC, es decir tenemos que tener un mecanismo que convierta la dirección IP en dirección MAC. El protocolo ARP se encarga de esto.

Entonces, el ARP permite que la computadora A encuentre la dirección de Hard de la computadora B.

Técnica: Se emite un pedido por broadcast y se obtiene la respuesta.



B responde enviando su dirección física. Entonces ahora la máquina A ya tiene la MAC de máquina B.

Una vez que maquina A recibe el MAC, lo guarda en cache para no estar pidiendo cada vez. Entonces se puede decir que la tabla ARP se encuentra en el caché.

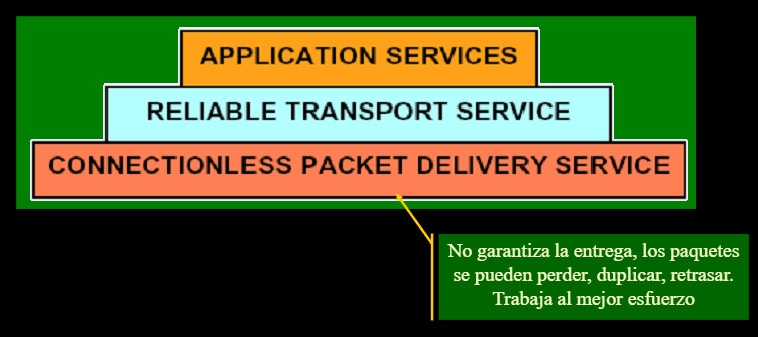
**Protocolo RARP (Reverse Address Resolution Protocol)**

Es un protocolo antiguo que trabaja en forma inversa al ARP.

Averigua nuestra propia IP, en base a nuestra MAC preguntamos nuestra IP, luego uno o más servidores nos van a contestar y nos van a dar nuestra dirección IP.

**Itinerario 3 - Protocolo ICMP**

**Capas conceptuales**



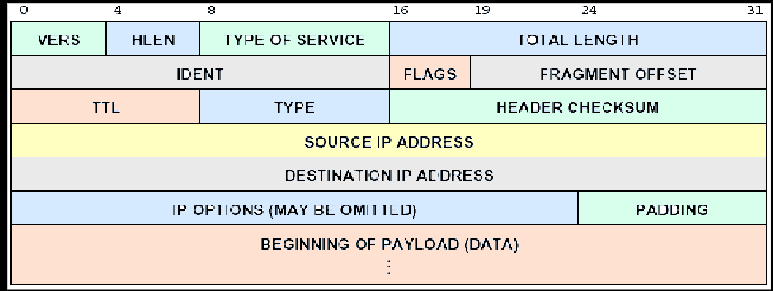
El protocolo IP es un protocolo no confiable. Esto significa que el IP hace el mejor esfuerzo para que el paquete llegue al destino pero no me garantiza que no se va a duplicar, perder o llegar. Para garantizar estos, hay que poner alguien arriba que lo controle “RELIABLE TRANSPORT SERVICE”.

**Paquete internet**

* Conocido como “Datagrama”.
* Servicio NO orientado a la conexión
  + No arma un camino antes de la transmisión.
* Header (Encabezado) + Payload (Datos)



**Formato IPv4**

****

**VERS** - Versión utilizada del protocolo IP (osea el numero 4 en binario “**0100**”)

**HLEN** - Longitud del encabezado medido en palabras de 32 bits (en este caso 5 filas de 32 bits como mínimo, en este caso “**0101**”)

**TOTAL LENGTH** - Longitud total medida en octetos de header + datos de payload (como son de bit 16 a 32, entonces tenemos **2^16** octetos como máximo en un datagrama.

TTL - Tiempo de vida, tiempo que se le permite al paquete estar en la red.

HEADER CHECKSUM - Complemento a 1 solo sobre el header

SOURCE IP ADDRESS - Direccion IP del origen

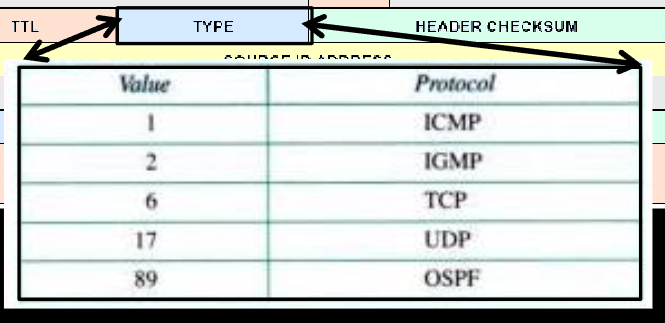
DESTINATION IP ADDRESS - Direccion IP del destino

IDENT - Todos los fragmentos igual ID (el del original)

FLAGS - El último bit indica si siguen o no más fragmentos

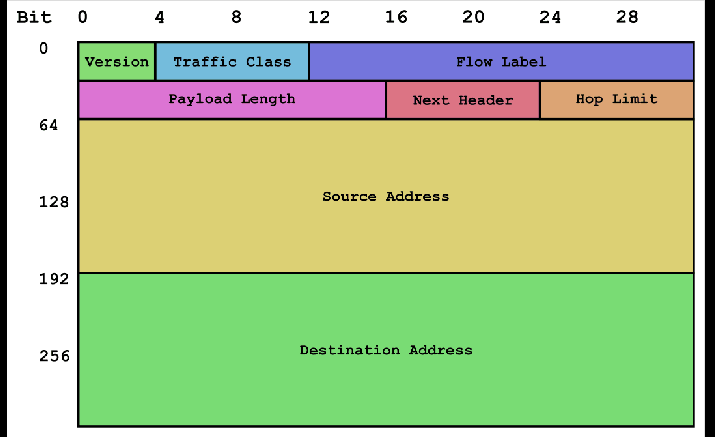
FRAGMENT OFFSET - Orden de los fragmentos

TYPE -



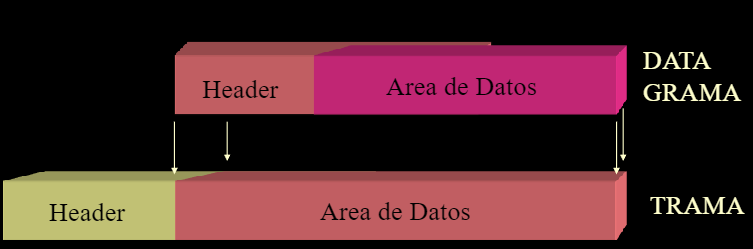


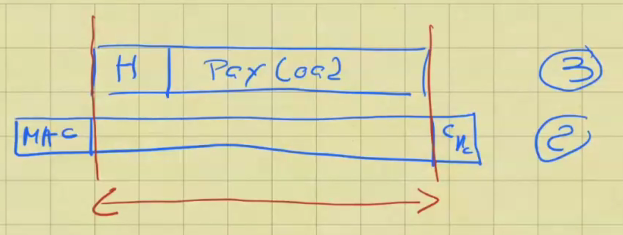
**Header IPv6 std**



FLOW LABEL - Un flujo, permite diferenciar conjuntos de paquetes que vienen viajando.

**Encapsulado IPv4**





El máximo tamaño que puede tener una trama en red ethernet es de (**MTU**) 1500 octetos.

El paquete IP que está encapsulado dentro del área de datos de la trama puede ser mucho más grande que 1500 octetos.

Esto quiere decir que capaz no entre todo en la trama, entonces hay que fraccionar el paquete en tamaños que si entre en la trama.

Cada fragmento del paquete tiene un header

0100 0101

| VERS | HLEN | TOS | TLEN |
| --- | --- | --- | --- |

0 31

La versión es 4 por ser IPv4 osea 0100

El HLEN es el tamaño del header, el header tiene un tamaño mínimo que es de 5 líneas.

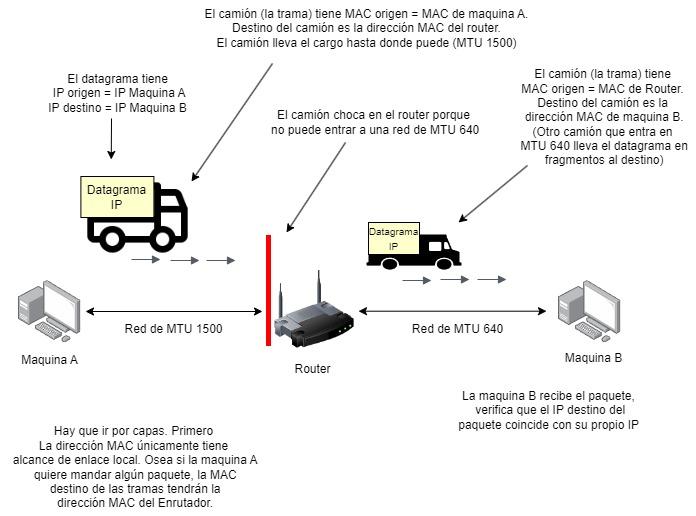
Las 5 líneas son palabras de 32 bits cada una, y la cantidad de líneas va en HLEN.

Cada palabra tiene 4 octetos. Entonces, 4 x 5 octetos son 20 octetos como máximo.

20 octetos es normalmente el número que se usa en HLEN porque es el mínimo.

Entonces el mínimo tamaño es de 20 octetos y el máximo es tener 1111 en HLEN, o sea 15 palabras de 32 bits. Y si cada palabra tiene 4 octetos, 4 x 15 = 60 octetos como máximo tamaño de HLEN.

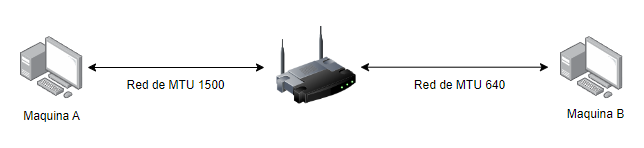
**El método camioncito**



A través de **ARP** (Address resolution Protocol), el router envía a todas sus máquinas diciendo “por favor, la Máquina B, contesteme su MAC”, La máquina B responde su MAC y el router lo guarda en una tabla.

**Fragmentacion**

Supongamos un paquete IP de máximo tamaño



El máximo tamaño posible de un paquete es de 2^16 = 65536 octetos bits.

Estos octetos están formado por header = 20 y datos = 65516

Este paquete original tiene un ID (supongamos ID1)

| 20 | 65516 |
| --- | --- |

Y tenemos que fragmentar estos en tramas de 1500 MTU

| Header | MTU 1500 |
| --- | --- |

Dentro de los 1500 MTU están los paquetes:

| 20 | 1480 |
| --- | --- |

¿Cuántos paquetes de 1480 octetos hace falta para mandar 65516 octetos de datos?

N = 65516 / 1480 → **44.26**

Osea que el paquete original de 20 octetos en header y 65516 octetos de datos se va a dividir en 44 paquetes pequeños fragmentos de 1480 octetos de datos + 20 octetos de header.

El paquete 45 sería más chico de 396 octetos.

Cada uno de estos fragmentos contienen en el header el ID del paquete original.

Entonces en el enlace 1 de 1500 MTU viajan 45 paquetes.

La máquina B sabe cuáles fragmentos pertenecen al paquete a través del ID1 que contienen los header y los junta.

Pero ahora los tiene que ordenar. Esto se hace con el offset. El offset es el número del primer octeto en referencia al octeto original.

Por ejemplo el primer octeto contiene el octeto 0 al octeto 1479, el segundo octeto contiene el del octeto 1480 al octeto 2959, etc… Entonces la primera trama tiene un offset de 0, la segunda trama tiene un offset de 1480, el tercero tiene offset de 2960, etc…

La fórmula para poder calcular el offset es (N - 1) \* Tamaño de trama (datos) donde N es igual a la posición de la trama.

Entonces sabemos que la trama 1 va primero por que (1 - 1) \* 1480 = 0 entonces tiene offset de 0 por ende va primero.

Sabemos el offset de la trama 45 (el último de 396 octetos) por que (45 - 1) \* 1480 = 65120 y como la trama contiene 396 octetos, 65120 (offset) + 396 = 65516 octetos (tamaño de datos del paquete original).

Y así se puede calcular el offset de cualquier paquete.

**Itinerario 4 - Protocolo TCP**

Ya estamos en la **capa 4 (Transporte).**

Hay dos protocolos principales:

* Transmission control protocol (TCP)
* User Datagram protocol (UDP)

Sabemos cómo sabe la máquina con que protocolo salta a partir del campo TYPE en el header del IP.

TCP o UDP se encapsula en el campo TYPE de la parte de datos en el payload del IP tanto como el IP se encapsula en la trama.

**UDP**

* Un protocolo NO orientado a la conexión. No arma un camino previo a la comunicación.
* Trabaja al “mejor” esfuerzo. Lo mismo que trabaja IP. Osea “intenta” efectuar su trabajo pero no tiene confirmación.

**TCP**

* Más lento pero más confiable. Verifica que el mensaje realmente llegó, esa verificación lo hace más lento.
* Un protocolo SI orientado a la conexión. Arma un camino antes de la comunicación.

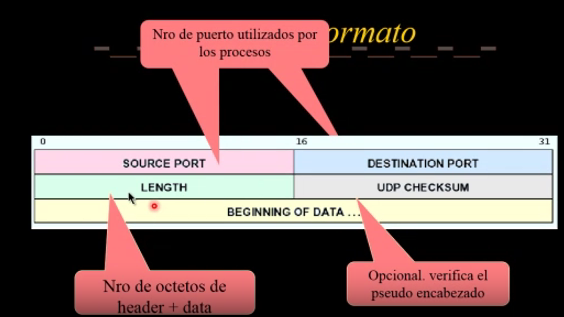


A Veces no es importante verificar que te llegue todo, por ejemplo en un video no te va a importar si te falta un pixel, pero importa que llegue a tiempo para que no se detenga la imagen del video. Por lo tanto, a veces es preferible el UDP sobre el TCP.

En este protocolo, la aplicación es responsable de corregir errores, o sea el que pidió que le llegue el mensaje tiene que decidir si no le interesa que llegue el 100%.

En caso de tener error, se **descarte** un datagrama, no se genera mensaje de error.

IP lo que hace es especificar la computadora (especificar la conexión) y decide a qué máquina o conexión va el mensaje. Una vez que llega a la conexión, no sabe que hacer adentro y a que aplicación llevarlo. El UDP decide a qué aplicación en específico van los mensajes. El UDP tiene como paradigma los **puertos.** Cada aplicación corre en un puerto en específico en el cual transmite y recibe información.



Tenemos puerto origen, y puerto destino. Estos hacen referencia a los procesos (16 bits). Osea podríamos tener 2^16 procesos distintos (aunque algunos están reservados).

El campo LENGTH que vendría a ser el número de octetos que tiene el HEADER más la DATA.

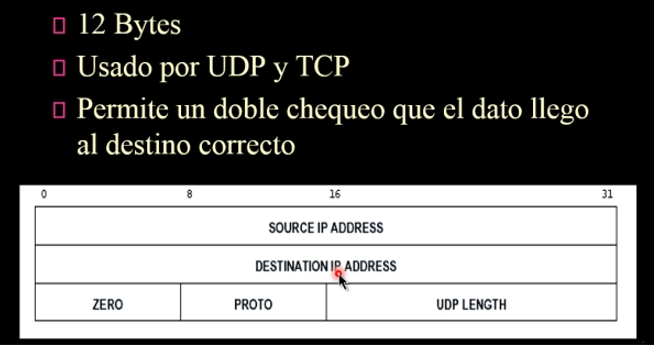
Tenemos el campo CHECKSUM que lo que hace es un control de que no haya errores y de que no se haya modificado que puede ser opcional.

A partir de ahí, comienzan los datos. En este caso tenemos 8 octetos (SOURCE PORT = 2 octetos, DESTINATION PORT = 2 octetos, LENGTH = 2 octetos y CHECKSUM = 2 octetos). Lo cual es el HEADER mínimo que tiene un segmento UDP. El de IP, si recordamos, el HEADER mínimo era de 20 octetos mientras UDP tiene 8.

**PSEUDO encabezado (PSEUDO HEADER)**

Existe tanto en TCP como en UDP.

**UDP:**

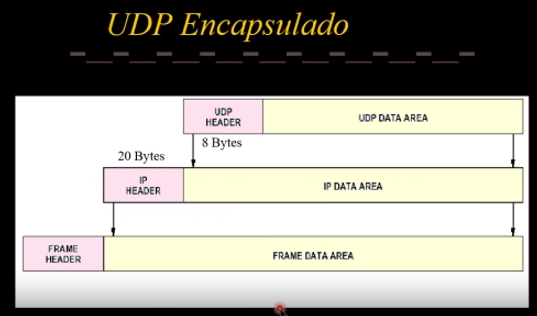


Se le pide prestado al IP algunos datos. Con esos datos se arma un PSEUDO header. Esto da una seguridad extra. Con solamente UDP, sabemos el puerto en el que estamos. Pero podríamos llegar a equivocarnos. Osea podríamos llegar al puerto correcto pero de la máquina incorrecta. Para evitar este error, se le agrega ciertos valores.

**Desventaja:**

Esto es un problema con la norma general en los modelos de capas porque el modelo de capas favorece a que las capas sean completamente independientes, y esto los hace que NO sean independientes.

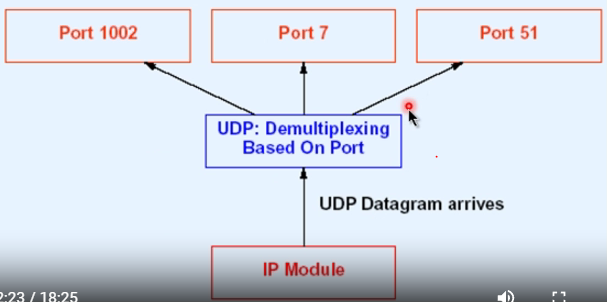
De todas maneras, es conveniente porque nos permite un doble chequeo que nos da una mayor seguridad que es realmente importante.



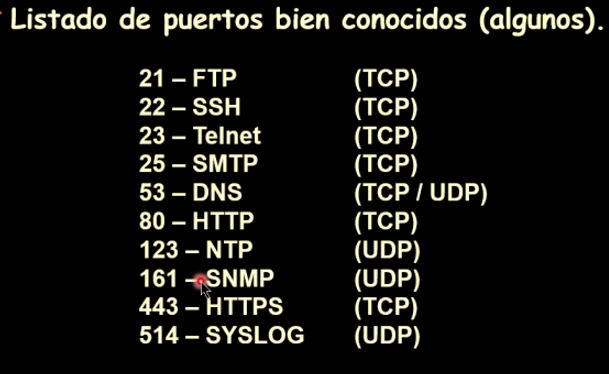
UDP se encapsula dentro del área de datos del IP, que se encapsula dentro del área de datos de la trama

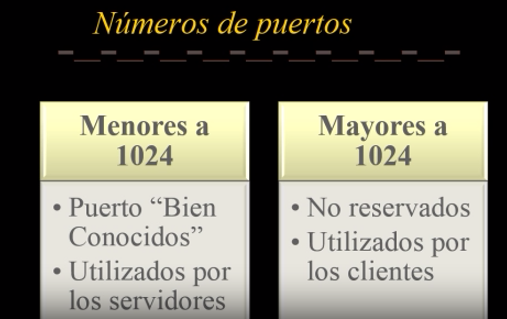
HEADER UDP - siempre 8 bytes (8 octetos)

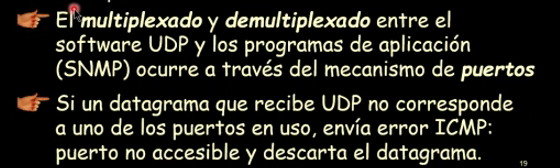
HEADER IP - minimo 20 bytes (20 octetos)



Un proceso importante que hace UDP, es el demultiplexado. Significa que dentro UDP lo que hace es diferenciar por puertos. Esta apertura es demultiplexado. En el receptor se demultiplexan lo que en el transmisor se multiplexing. Es decir, sacamos para cada proceso el puerto al que le corresponde ir.



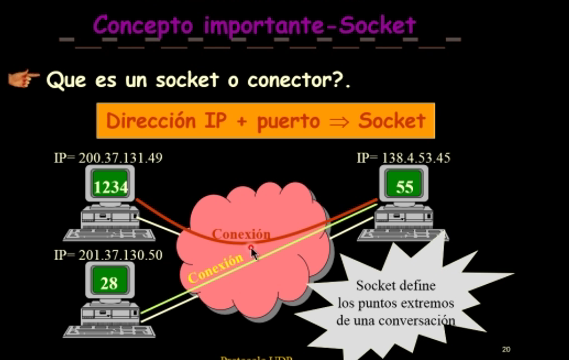




**¿Qué es un conector o socket?**

Está formado por la dirección IP (origen y destino) + el puerto (origen y destino).

No puede haber en un momento dado, 2 mensajes con igual sockets. Por ejemplo, no puede haber en un momento dado en la red, dos paquetes con el mismo origen con mismo destino, mismo puerto destino y mismo puerto origen. No esta permitido.



**El tamaño máximo que puede tener un datagrama**

Máximo tamaño = 65535 bytes

Minimo HEADER IP = 20 bytes

HEADER UDP = 8 bytes

Maximo DATA UDP =

( Máximo tamaño = 65535 bytes) - (HEADER IP + HEADER UP) =

65507 bytes o menos dependiendo el tamaño de HEADER IP.

**TCP:**

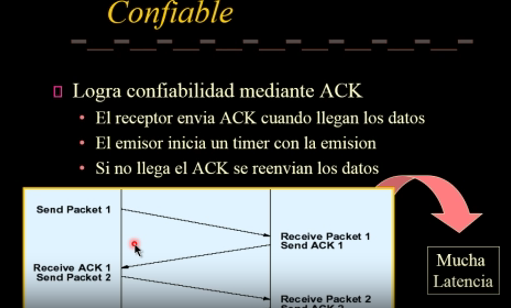


En TCP el receptor le interesa que todo el paquete está llegando, aunque lo reciba más lento. Osea hay envíos de vuelta desde el receptor al emisor avisando que necesita recibir todo el paquete con Acknowledgements de tal forma de controlar el flujo.

El protocolo TCP también corre SOBRE el IP al igual que el UDP, y ofrece un servicio de transferencia de datos eficiente y confiable. Se encapsula sobre el IP, y se consigue esta confiabilidad de por que de algún modo pide retransmisiones en el caso de que pierda algún paquete. Justamente eso es lo que lo hace más lento que la UDP.

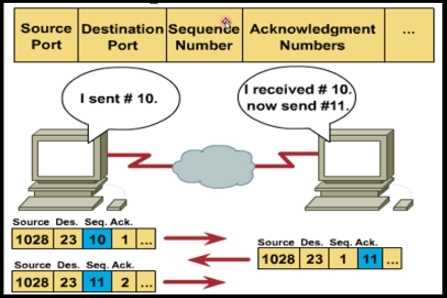
**Características de TCP**

* Orientado a la conexión
* Full Duplex
* Fiable
* Flujo de Bytes controlado
* Segmentacion
* Transmisión uno a uno

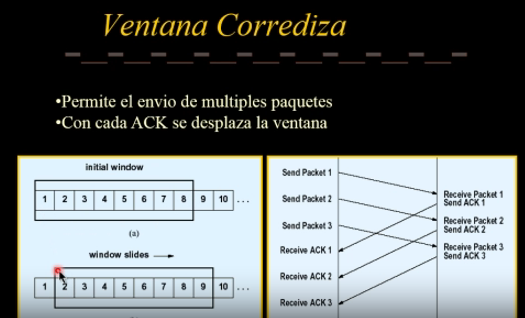


**Fases de TCP**

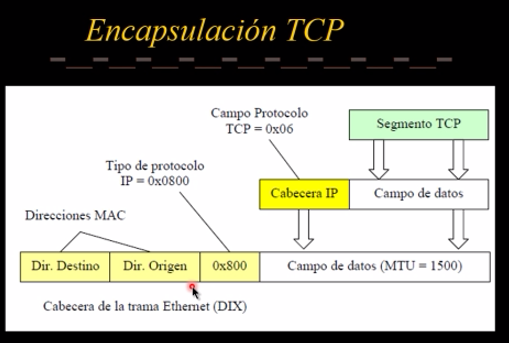
* Establecimiento de conexión (se hace el triple hashing) y una vez establecida la conexión, se fijan los parámetros de la conexión.
* Transferencia de datos
* Cierre de conexion

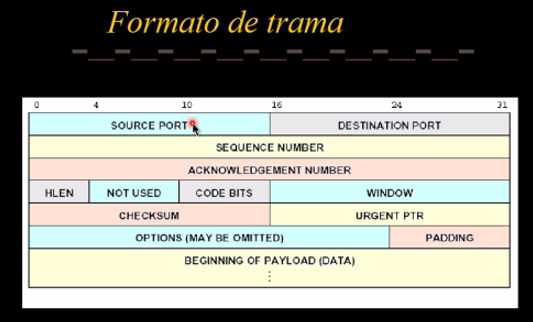


Aca en la máquina B el 1028 y el 23 son erróneos, se deben invertir.

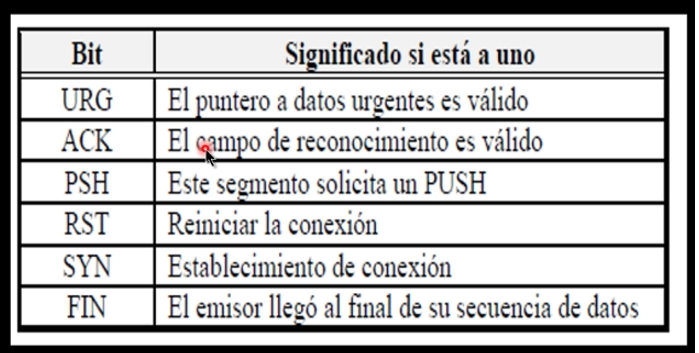


**Segmentos**





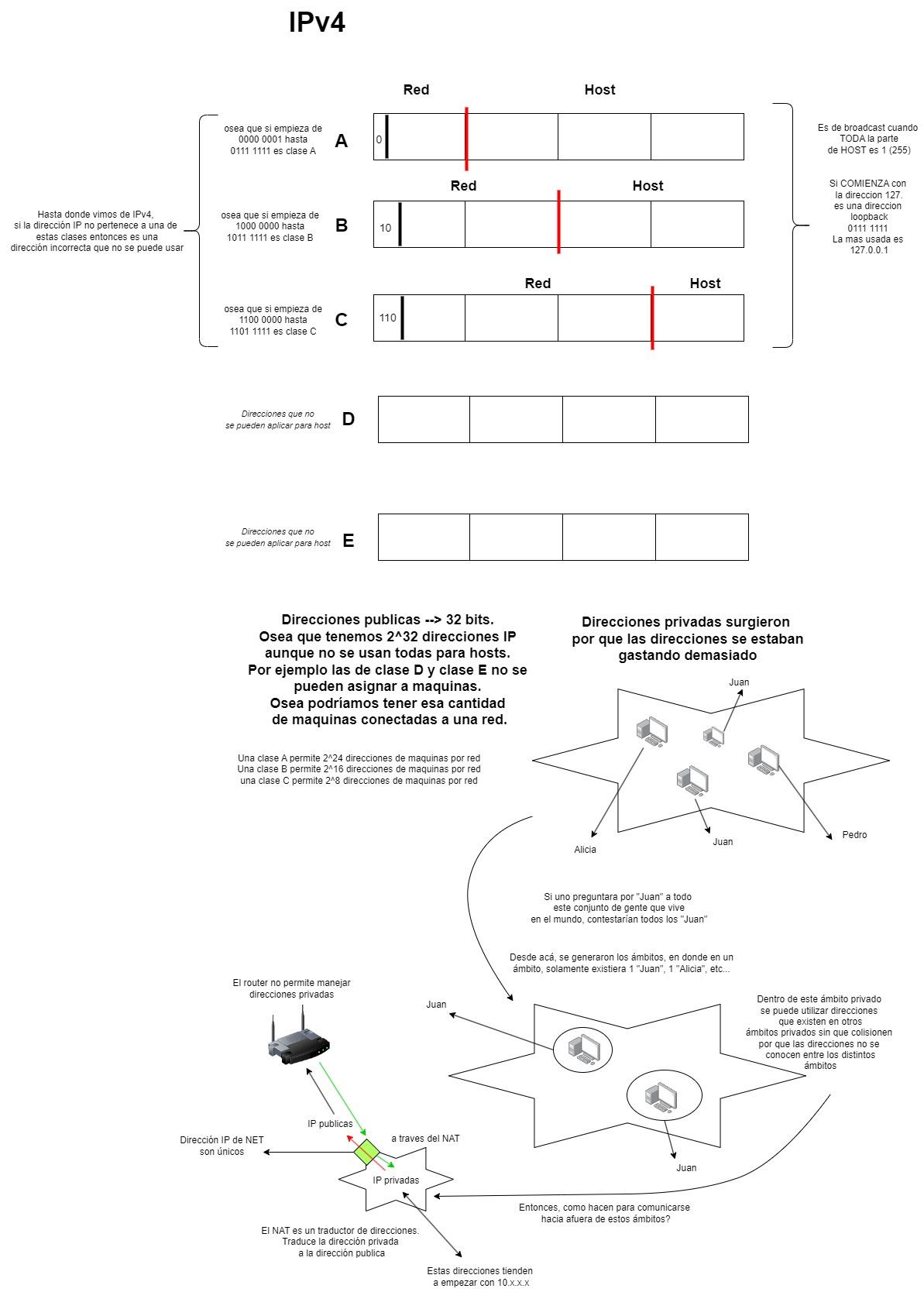
**Campos**

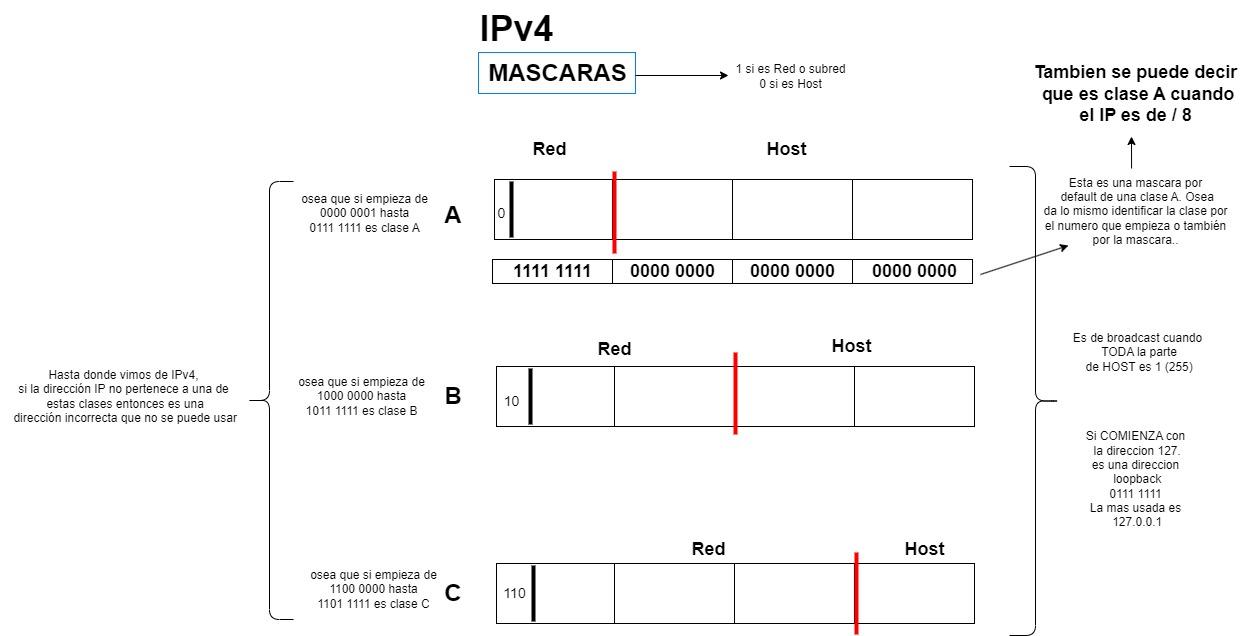


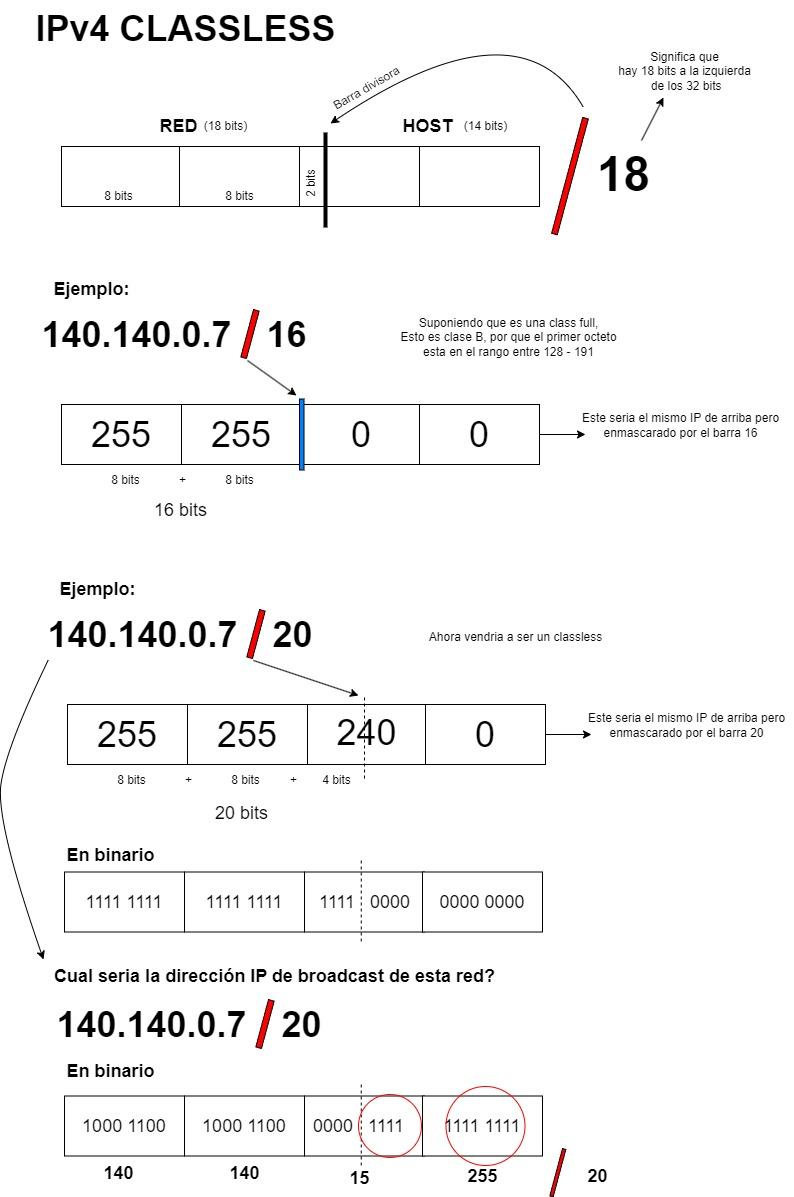
**Como hacer para que la red no esté ansiosa, y tampoco lenta**

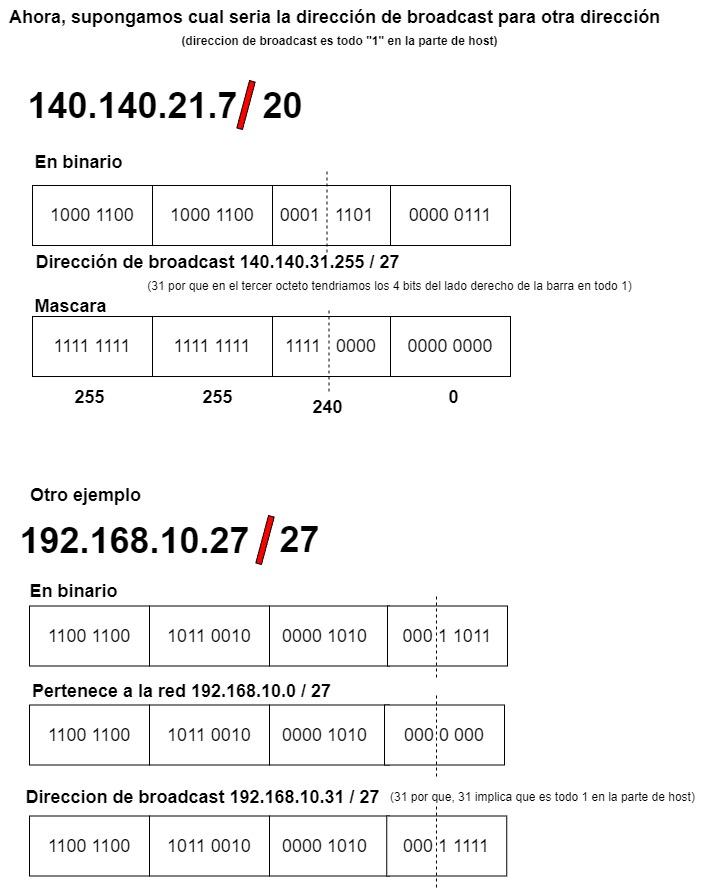


**VC 03**





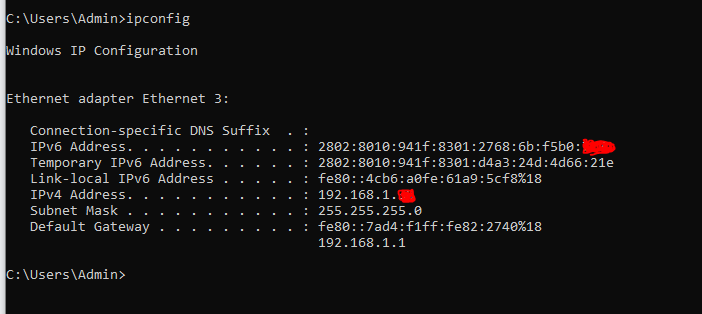




**Itinerario 5 - Videos explicativos de aspectos prácticos**

Ir a escritorio, WINDOWS + R y doy enter en “cmd”

Ingreso “ipconfig”



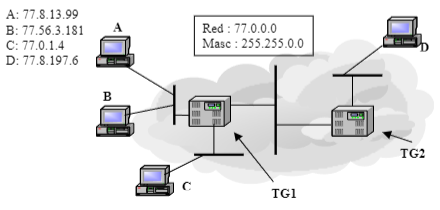
**Itinerario 6 - Subredes**

La RFC 950\* define los procedimientos estándar de división de las clases A,B y C en partes más pequeñas llamadas subredes.

**Subredes implícitas**

Método indicado en la RFC - 925

Se tienen aquí "cajas" ( TG1, TG2 ) que participan en un ARP más allá de las LAN.



Este es un procedimiento sencillo que permite que las máquinas direccionen de acuerdo alas reglas normales de IP que dicen que una estación debe ARP'ar otra estación en su misma red y usar el default Gateway para salir de su red.Dado que la máscara es 255.0.0.0 se hace ARP para cualquier máquina dentro de la red 77. este permite a una empresa usar completamente su dirección antes de necesitar otra.

**Ejemplo 1:**

Supongamos que la estación A (77.8.13.99 ) desea comunicarse con la estación B (77.56.3.181 ) de su misma red.

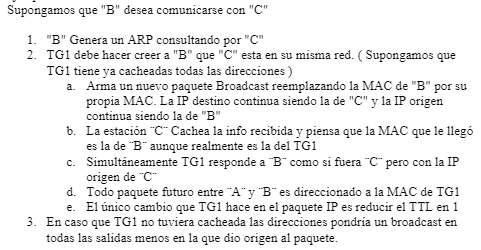
1. "A" verifica direcciones y máscara, notando que se encuentra en la misma red que el destino.

2. "A" envía un ARP consultando por "B"

3. "B" responde al ARP

4. TG1 cachea toda la info que escucha.

**Ejemplo 2:**



**Ejemplo 3:**

Supongamos que la estación "D" desea comunicarse con "A"

1. "D" genera un ARP

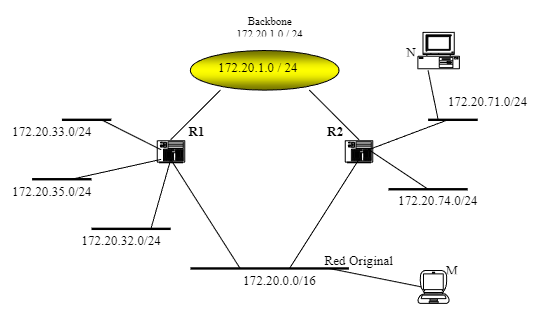
2. TG2 lo pasará a la otra interface y lo recibirá TG1 ( lo cachea )

3. TG1 conoce donde está "A" y le envía el ARP ( modificado )

4. TG1 responde a TG25. TG2 responde a "D”

**Método Proxy ARP**

Este es un método que aún se utiliza, especialmente cuando se está en medio de una conversión a subredes.Considere que una red se está "subneteando" . Durante algún tiempo parte de la red pertenece a una subred ( con la máscara que le corresponda ) y parte de la red aún estará en la red original ( con la máscara original).



En la red original es donde aún permanecen la mayoría de las máquinas, para estas máquinas la red no está subneteada ( notar /16 ). Los ruteadores R1 y R2 aíslan las subredes de la red original y permiten que los host de las subredes se comunique entre sí por el backbone de alta velocidad.

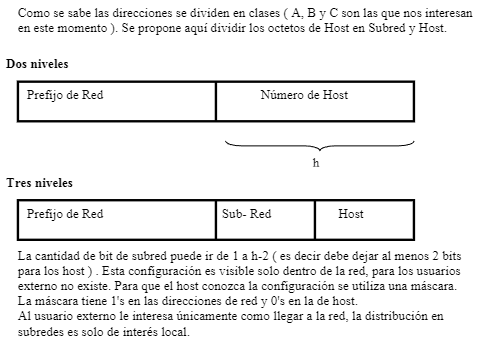
Supongamos que la máquina "M" ( 172.20.97.101 / 16 ) quiere comunicarse con lamáquina "N" ( 172.20.71.76 / 24 ).En virtud de su máscara, "M" considera que "N" está en su misma subred, por lo cual generará un ARP. Normalmente los ruteadores no permiten el paso de los broadcast por lo cual "N" no se enteraría nunca que "M" desea comunicarse.

Sin embargo R2 está recibiendo un pedido de ARP de una máquina que está buscando una dirección IP que corresponde a una red a la que R2 está directamente conectado. SiR2 tiene habilitada la opción ARP contesta a "M".R2 "mentirá" que es N . Cada vez que R2 reciba un paquete dirigido a 172.20.71.76 sedará cuenta que lo debe reenviar a su destino real, para ello arma una trama en la que mantiene la IP original pero reemplaza la dirección destino ( que era la propia - la de R2-)por la de N y pone como MAC origen la propia.

El proceso inverso, es decir que N contesta a M, es un procedimiento normal.172.20.71.76/24 sabe que está en una subred y que el destino está en una subred distinta,por lo que el paquete es enviado al Default Gateway ( R2 - 172.20.71.1 ).Para ello N hace un ARP y averigua la la MAC de R2 y luego envía a la trama. A su vez R2 consulta su tabla para ver si tiene la MAC de M.Si R2 esta conectado diurectamente con la red de M continúa el proceso normal de delivery mediante un nuevo ARP para averiguar la MAC de M.El procedimiento se limita a los ruteadores directamente conectados entre los destinos

**Subredes Explícitas**

***( Este método DEBE ser bien conocido por el estudiante y será sobre el que se trabajará posteriormente.)***



**Máscaras**

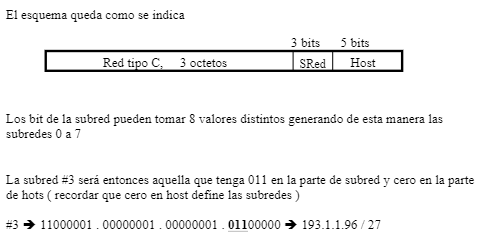
**Problema 1**

Una organización tiene una dirección de red asignada : 193.1.1.0 /24 y necesita definir 6 subredes. Cada subred debe soportar 25 Host.

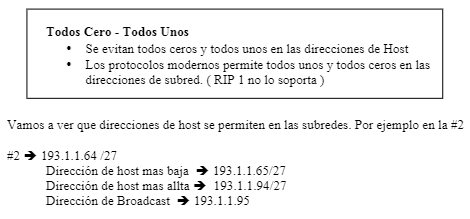
1) Determinar la cantidad de bit necesarios para definir las 6 subredes.

Se ve que se necesitan 3 bit ya que con dos solo se pueden definir 4 subredes, por lo que la máscara será : 255.255.255.224. Nos quedan aún 5 bit para host, es decir hasta 30 hosten cada subred ( 25 - 2 ). se restan 2 por las direcciones de host todas 1 y todas cero.En notación simplificada la máscara es /27.

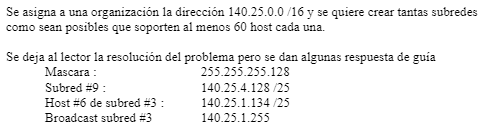
2) Definiendo cada subred



3) Direcciones de Host de cada subred



**Problema 2**



Para crear subredes a partir de la dirección IP 140.25.0.0/16 y asegurarse de que cada subred pueda soportar al menos 60 hosts, primero debemos determinar cuántos bits debemos asignar para la parte de host. Luego, restamos esos bits del prefijo de red original para obtener la nueva máscara de subred.

La dirección IP 140.25.0.0/16 tiene una máscara de subred de 16 bits, lo que significa que los primeros 16 bits se reservan para la red y los últimos 16 bits están disponibles para hosts. Para admitir al menos 60 hosts en cada subred, necesitamos al menos 6 bits para la parte de host, ya que 2^6 = 64, lo que nos da más de 60 direcciones de host por subred (64 - 2 para la dirección de red y la dirección de difusión).

Ahora, restamos esos 6 bits del prefijo de red original (16 bits) para obtener la nueva máscara de subred. La nueva máscara de subred será de 16 bits para la red original más 6 bits para hosts, lo que resulta en una máscara de 22 bits:

| 16 bits | 6 bits | 6 bits |
| --- | --- | --- |
| Red original | subredes | hosts |
|  |  | 2^6 = 64  64 - 2 = **62**  *( Restamos 2 porque todos 1 y todos 0 están reservados )* |

140.25.0.0/22

Con esta nueva máscara de subred, podemos crear subredes que admitan al menos 60 hosts cada una. Cada subred tendrá un tamaño de 2^6 = 64 direcciones de host, y las direcciones de red y de difusión se cuentan en esos 64. Por lo tanto, cada subred tendrá 64 - 2 = 62 direcciones de host utilizables.

Puedes crear varias subredes dentro del rango de 140.25.0.0/22, y cada una de ellas tendrá suficiente espacio para al menos 60 hosts.

**Subredes**

Para hacer más eficiente el aprovechamiento del espacio de direcciones es conveniente crear subredes de tamaño más ajustado en vez de un mismo tamaño para todas.Evidentemente los ruteadores deberán ser informados para poder rutear adecuadamente estas direcciones.Para poder lograr esto se debe trabajar con máscaras distintas conociéndose la operación como **VLSM ( Variable Length Subnet Masks )**

**VSLM**

Mediante VSLM logramos que el tamaño de la subred refleje exactamente lo deseado.

**Ejemplo VLSM**

Ud es el administrador de redes de una empresa con tres sucursales y una casa central unidas mediante enlaces WAN. Ver la figura:



Se le otorga la direccion 192.168.10.0/24

Se dan a continuación las sucursales y la cantidad de host necesarios en cada una.



La casa central en:



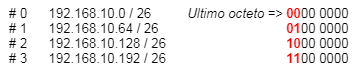
Su trabajo es asignar las subredes correspondientes de forma tal que se desperdicie la menor cantidad de direcciones posible.

1) Asigne primero el sitio con mayor cantidad de host

Ing. Berenga 60 Host ==> **6 bit de host** , **2 de direccionamiento**

6 bits de host por que 2^6 = 64 direcciones para hosts, -2 de broadcast y red, osea **62 direcciones** para hosts.

Con 2 bit de direccionamiento tenemos **4 subredes**

****

Ing Berenga. toma entonces la subred #0 192.168.10.0 / 26



Host más alto de la subred #0 seria **192.168.10.62 /26**

Broadcast de la subred #0 seria **192.168.10.63 /26**

2) El siguiente lugar en cantidad de host es Londres con 28 máquinas para lo cual hacen falta 5 bits : 2^5 -2= 30. Esto nos dejan 3 bit para direccionamiento (/ 27)

Tomamos las #1 192.168.10.64 / 26 considerada ahora como /27



Asignamos # 1,0 a Londres 192.168.10.64 / 27

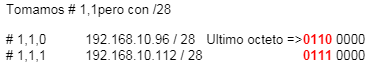


Host más bajo de subred #1,0 seria **192.168.10.65 /27**

Host más alto de subred #1,0 seria **192.168.10.94 /27**

3) Nos queda por asignar Nueva York y Tokio

Nueva York necesita 12 Host ==> 4 bits, esto nos deja 4 bits para direccionamiento.



Asignamos a Nueva York

# 1,1,0 192.168.10.96 / 28

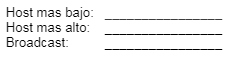


Host más bajo de la subred #1,1,0 seria **192.168.10.97 /28**

Host más alto de la subred #1,1,0 seria **192.168.10.110 /28**

Broadcast para la subred #1,1,0 seria **192.168.10.111 /28**

Tokio necesita también 12 Host, al igual que Nueva York, Podemos asignarla otra subred, la #1,1,1

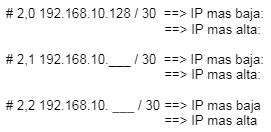


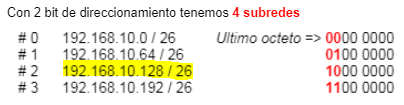
Host más bajo de la subred #1,1,1 seria **192.168.10.113 /28**

Host más alto de la subred #1,1,1 seria **192.168.10.126 /28**

Broadcast para la subred #1,1,1 seria **192.168.10.127 /28**

5) Recordando que los enlaces WAN necesitan cada uno dos direcciones IP,tenemos entonces que es suficiente con 2 bits para cada una.Para los enlaces WAN empleamos entonces la subred # 2 mediante una configuración /30 ( Complete el ultimo octeto )





El #2 pero /30. Osea **1000 0000**

*#2,0* ***1000 0000***

*IP mas bajo: 192.168.10.128 /30*

*IP mas alto: 192.168.10.131 /30*

*#2,1* ***1000 0100***

*IP más bajo: 192.168.10.132 /30*

*IP mas alto: 192.168.10.135 /30*

*#2,2* ***1000 1000***

*IP mas bajo: 192.168.10.136 /30*

*IP mas alto: 192.168.10.139 /30*

*No estoy seguro!*

**Parcial Nº1**

Multiple choice de 15 minutos (se habilita durante 30 minutos, de 17:00 a 17:30).

Después se habilita la parte práctica de VLSM la cual se pide que hagamos un VLSM con ciertos datos y lo configuremos en packet tracer que pueda pingear a todo. (20 minutos)

**Ejercicio de práctica**

