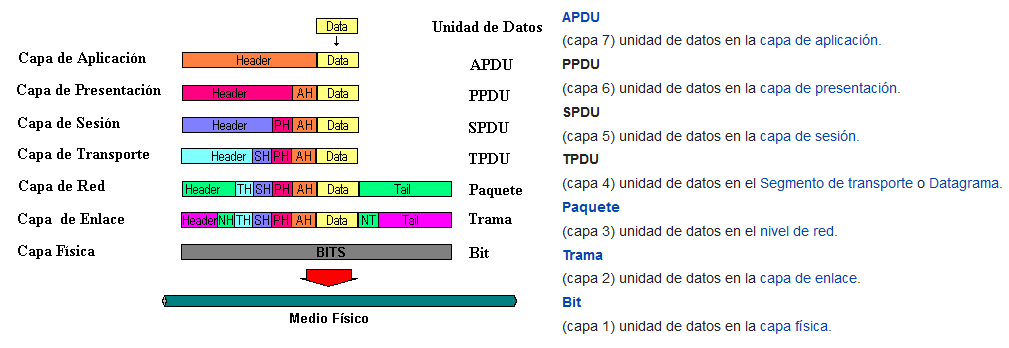
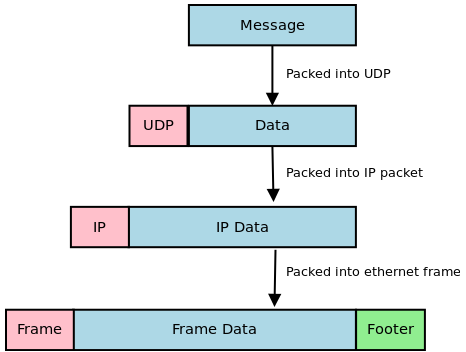
**Capas OSI:**



**Hexadecimales Conocidos:**

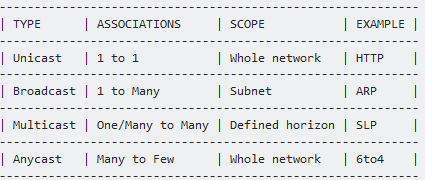
192 = C0 = 1100 0000; 168 = A8 = 1010 1000; 172 = AC = 1010 1100; 10 = 0A = 0000 1010

**Unicast**: único receptor. (Llamada telefónica)

**Multicast**: muchos receptores. (Teleconferencia). Corresponde a las direcciones clase D, y tiene un rango reservado de direcciones IPv4 que va desde la 224.0.0.0 y a la 239.255.255.255. MAC: 01:00:5E:...

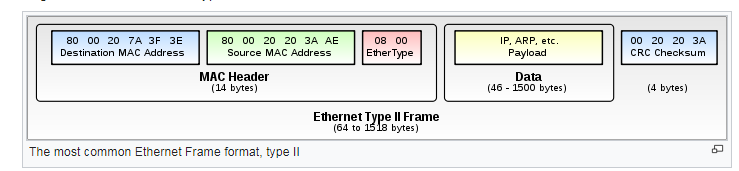
**Broadcast**: se envía a todos los receptores (Radio). MAC: FF.FF.FF.FF. IP x.x.x.255.

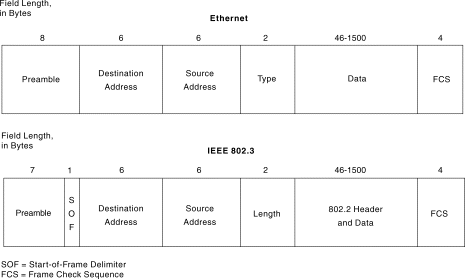
**Anycast**: se envía a cualquiera y lo toma alguno.





**Ethernet (Trama/Frame)**





**Preámbulo (Preamble)**: **El campo no es mostrado por el Wireshark**

Este campo contiene 62 bits de 1 y 0 en forma alternada finalizando con dos bits de 1 (en total 8 bytes), permitiendo ajustar los tiempos de ambas tarjetas (computadoras o equipos de comunicaciones) para tener una transmisión digital sincronizada.

**Dirección de destino(Destination Address)**

Corresponde a la dirección Ethernet (6 bytes) de la tarjeta Ethernet de destino de la trama a transmitir. Si esta dirección se compone enteramente de 1, entonces significa que es un mensaje Broadcast (mensaje para todas las estaciones de la red local). Los primeros 3 bytes de esta dirección están normados por la IEEE y cada fabricante de tarjetas Ethernet le corresponde un único trió.

**Dirección de origen (Source Address)**

Corresponde a la dirección Ethernet (6 bytes) de la tarjeta Ethernet que envía la trama a transmitir.

**Tipo de protocolo (Type)**  
Este campo indica el tipo de protocolo que está ocupando el formato de la trama Ethernet versión II. En otras palabras, diferencia los distintos tipos de protocolos de capas superiores que puedan ocupar Ethernet.

IPv4 tiene valor 0x0800,

IPv6 tiene valor 0x08DD,

ARP tiene valor 0x0806,  
RARP (Reverse ARP) 0x8035,

802.1Q tiene valor 0x8100,

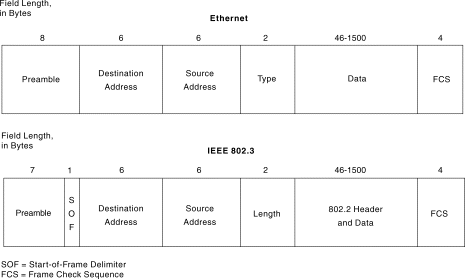
IPX tiene 0x8137.

Todos los valores son asignados por la IEEE en el RFC 1700 y poseen valores mayores de 0x05DC (1500 decimal). En este caso el protocolo es IP 0x0800 en hexadecimal y el paquete IP viene contenido en el campo DATA de esta trama Ethernet II.

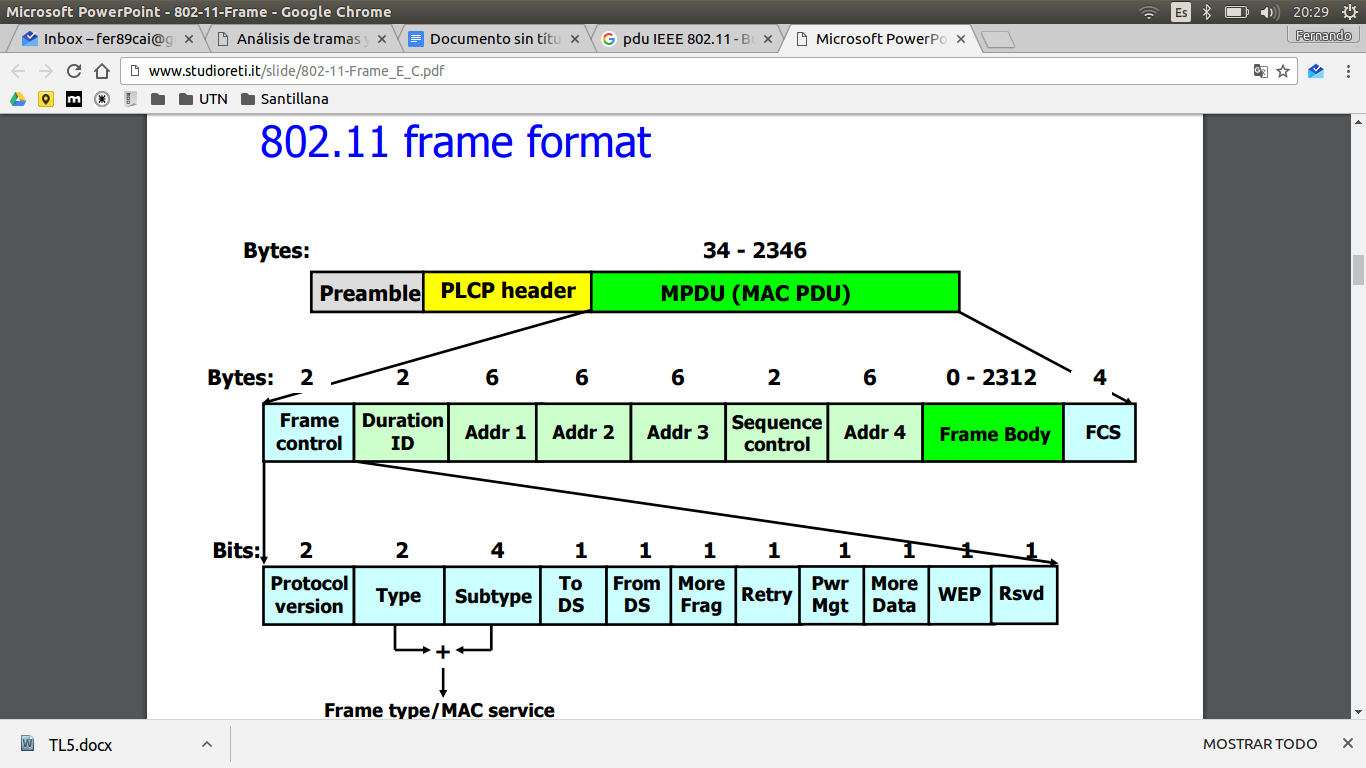
**FCS: El Campo no es mostrado por Wireshark**

Secuencia de verificación de trama, utilizada por la NIC para identificar errores durante la transmisión. El valor lo computa la máquina de envío, abarcando las direcciones de t ama, campos de datos y tipo. El receptor lo verifica

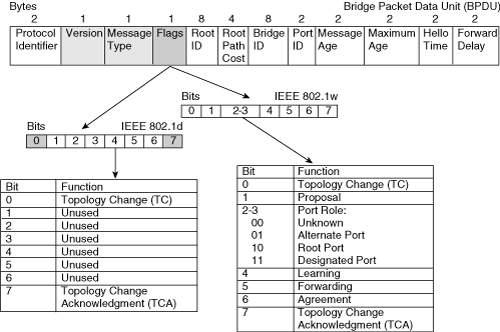
**IEEE 802.3 (Trama/Frame)**



**IEEE 802.11 Wireless LANs (Trama/Frame)**

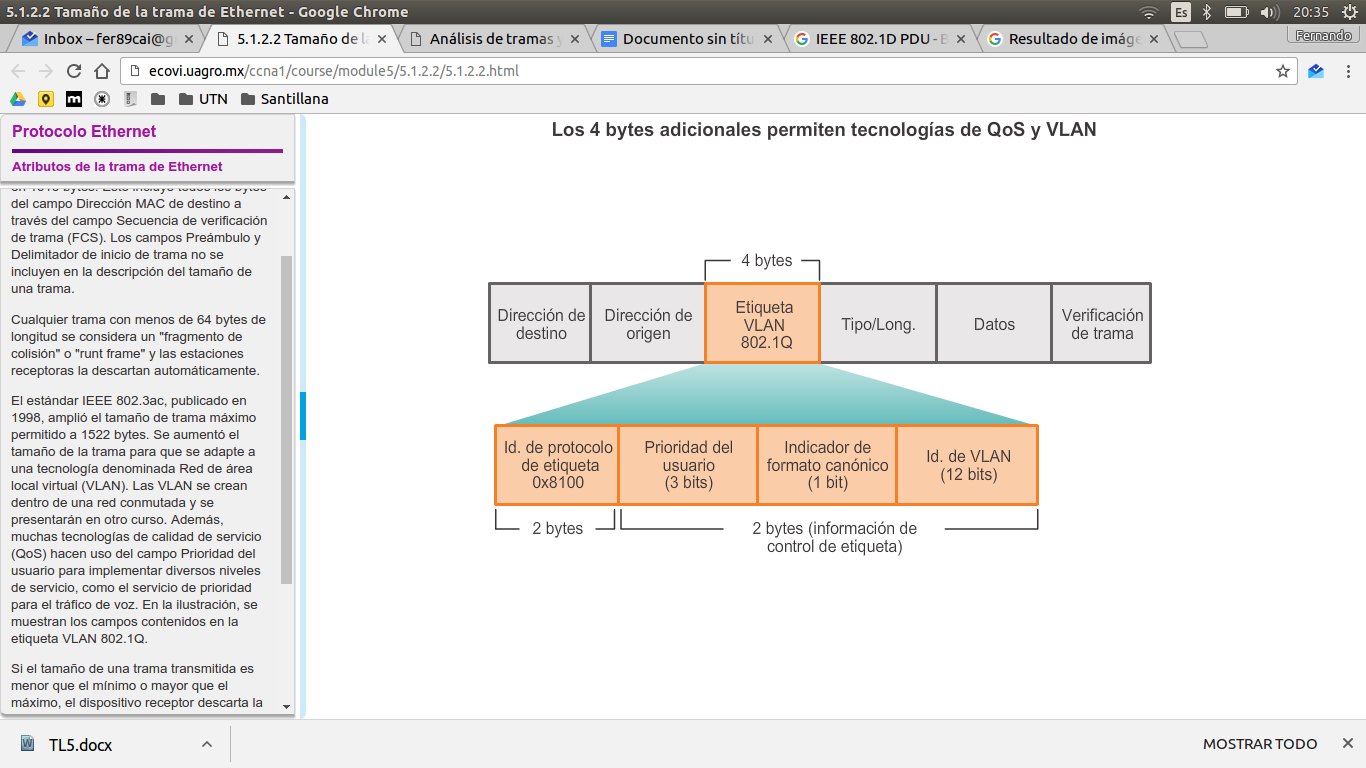


**IEEE 802.1D Spanning Tree Protocol**

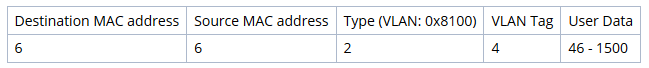


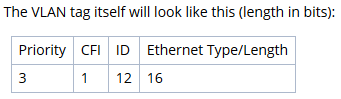
**VLAN - IEEE802.1Q**

Dentro de una trama Ethernet se la identifica con el tipo de protocolo 8100



Así lo ve Wireshark (El Tag es la información de control + Tipo/Long):

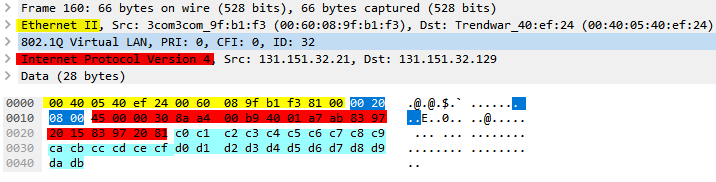




* **Priority:** the user's priority of this packet (ranges from 0 to 7)
* **Canonical Format Identifier (CFI):**
* Usually 0 (canonical format, bytes Big Endian, bits Little Endian).

If set to 1, this generally indicates that MAC addresses in the frame are in non-canonical format for Ethernet (bits Big Endian), i.e. Token Ring and FDDI MAC address order.

* More precisely, on Ethernet and on FDDI without source routing, i.e. when the RII bit in the frame's source MAC Address field is 0, this indicates an Embedded Routing Information Field (E-RIF) of two octets of more then follows the VLAN tag which itself has a Non-canonical Format Indicator that will definitively say whether the MAC addresses are in canonical order or not. On Token Ring and FDDI with source routing, this always indicates non-canonical / native TR and FDDI MAC address format.
* The CFI being set indicates that the frame originated on a Token Ring (IEEE 802.5) or FDDI segment.
* In IEEE 802.1ad and ah, this was replaced with a Drop Eligible Indicator (DEI) bit, and recent versions of 802.1Q (2011 and later) adopt this as well, rolling up 802.1ad and ah, since Token Ring and FDDI are now pretty rare support for bridging them was dropped.
* **ID:** the ID of the VLAN (group) to which this packet belongs



**Ethernet.**

**VLAN**:

**Prioridad|CFI|ID:** 0020 (000|0|**0000 0010 0000**) → 32

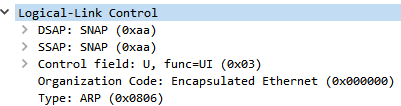
**Protocolo:** IPV4 (0800)

**IPv4**.

**Data**

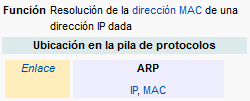
**LLC (Control de Enlace Lógico) - Parte Superior Capa de Enlace (La inferior es MAC).**

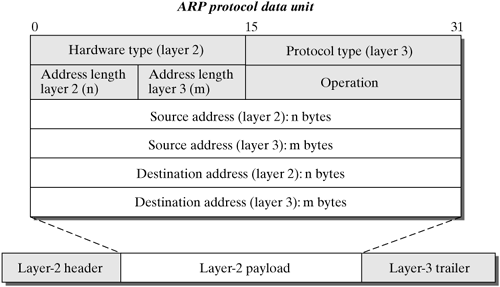
En algunas tramas de Virtual LAN aparece LLC. En ese caso el Length es 0x0024.

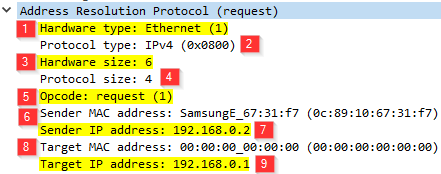


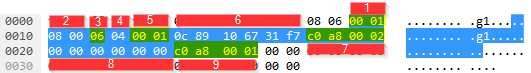


**ARP (Protocolo de Resolución de Direcciones )(Capa de Enlace)**









**Ethernet.**

**ARP**:

**Tipo Hardware**: 1 (Ethernet)

**Tipo de Protocolo**: 0800 (IPv4)

**Tamaño Hardware**: 6

**Tamaño Protocolo**: 4

**Operación**: 1 (Request)

**Emisor MAC**: 0C.89.10.67.31.F7

**Emisor IP**: C0.A8.00.02 (192.168.0.2)

**Receptor MAC**: 00.00.00.00.00.00

**Receptor IP**: C0.A8.0.01 (192.168.0.1)

**Opcode**:

Request (1): 00 01

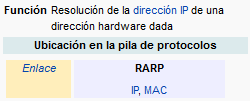


Reply (2): 00 02 (No tiene padding).

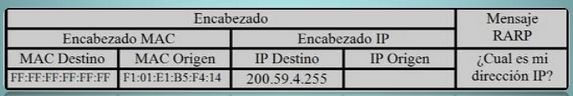


**RARP (Protocolo de Resolución de Direcciones Inverso)**

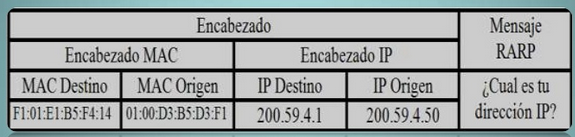
El protocolo de resolución de direcciones inverso (en inglés Reverse Address Resolution Protocol, RARP) es un protocolo de comunicaciones utilizado para resolver la dirección IP de una dirección hardware dada (como una dirección Ethernet).



Petición:

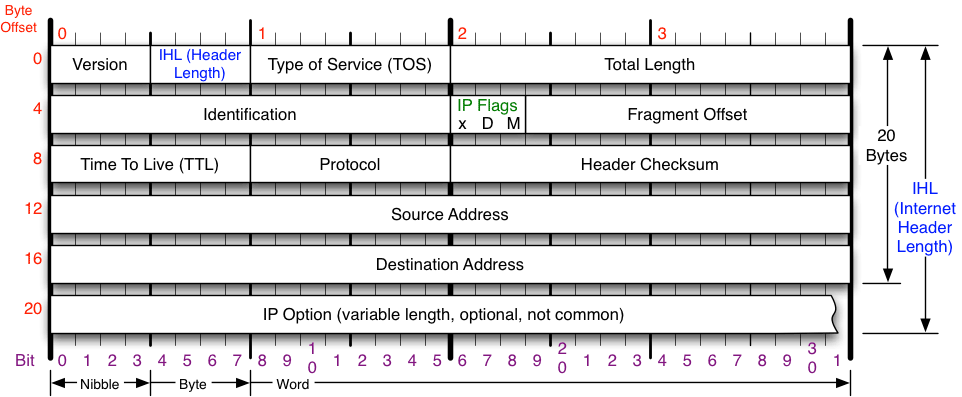


Respuesta:



**IPv4**





* **Versión** (4 bits): es la versión del protocolo IP que se está utilizando (actualmente se utiliza la versión 4 *IPv4*) para verificar la validez del datagrama. Está codificado en 4 bits.
* **Longitud del encabezado** o *IHL* por *Internet Header Length (Longitud del encabezado de Internet)* (4 bits): es la cantidad de palabras de 32 bits que componen el encabezado (Importante: el valor mínimo es 5). Este campo está codificado en 4 bits. Se debe multiplicar por 4 para tener los bytes.
* **Tipo de servicio** (8 bits): indica la forma en la que se debe procesar el datagrama.

**En Wireshark aparece como Differentiated Services Field.**

* **Longitud total** (16 bits): indica el tamaño total del datagrama en bytes. El tamaño de este campo es de 2 bytes, por lo tanto el tamaño total del datagrama no puede exceder los 65536 bytes. Si se lo utiliza junto con el tamaño del encabezado, este campo permite determinar dónde se encuentran los datos.

**En Wireshark si ocupa 1500 aparece como 05DC (20 bytes de cabecera + 1480 de datos)**

* **Identificación:** Se utilizará, en caso de que el datagrama deba ser fragmentado, para poder distinguir los fragmentos de un datagrama de los de otro. El originador del datagrama debe asegurar un valor único para la pareja origen-destino y el tipo de protocolo durante el tiempo que el datagrama pueda estar activo en la red. El valor asignado en este campo debe ir en formato de red.
* **Indicadores** Actualmente utilizado sólo para especificar valores relativos a la fragmentación de paquetes. Los 3 bits (por orden de mayor a menor peso) son:

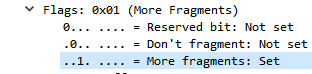
bit 0: Reservado; debe ser 0

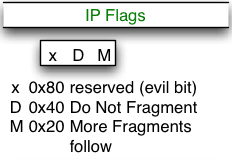
bit 1: 0 = Divisible, 1 = No Divisible (DF: Don’t Fragment)

bit 2: 0 = Último Fragmento, 1 = Fragmento Intermedio (le siguen más fragmentos) (MF: More fragments)

La indicación de que un paquete es indivisible debe ser tenida en cuenta bajo cualquier circunstancia. Si el paquete necesitara ser fragmentado, no se enviará.

**En Wireshark aparece como 0x20 cuando hay más fragmentos.**





* **Posición del fragmento** En paquetes fragmentados indica la posición, en unidades de 64 bits, que ocupa el paquete actual dentro del datagrama original. El primer paquete de una serie de fragmentos contendrá en este campo el valor 0. (Es decir, es la posición que le corresponde al reensamblar el paquete)
* **TTL** o **Tiempo de vida** (8 bits): este campo especifica el número máximo de routers por los que puede pasar un datagrama. Por lo tanto, este campo disminuye con cada paso por un router y cuando alcanza el valor crítico de 0, el router destruye el datagrama. Esto evita que la red se sobrecargue de datagramas perdidos.
* **Protocolo** (8 bits): este campo, en [notación decimal](http://es.ccm.net/contents/base/binaire.php3), permite saber de qué protocolo proviene el datagrama.
  + ICMP 1 (0x01)
  + IGMP: 2 (0x02)
  + TCP: 6 (0x06)
  + UDP: 17 (0x11)
  + IPV6: 41 (0x29)
* **Suma de comprobación del encabezado (16 bits)**: este campo contiene un valor codificado en 16 bits que permite controlar la integridad del encabezado para establecer si se ha modificado durante la transmisión. La suma de comprobación es la suma de todas las palabras de 16 bits del encabezado (se excluye el campo *suma de comprobación*). Esto se realiza de tal modo que cuando se suman los campos de encabezado (suma de comprobación inclusive), se obtenga un número con todos los bits en 1.
* **Dirección IP de origen** (32 bits): Este campo representa la [dirección IP](http://es.ccm.net/contents/ip.php3) del equipo remitente y permite que el destinatario responda.
* **Dirección IP de destino** (32 bits): [dirección IP](http://es.ccm.net/contents/ip.php3) del destinatario del mensaje.

**Fragmentación de datagramas de IP**

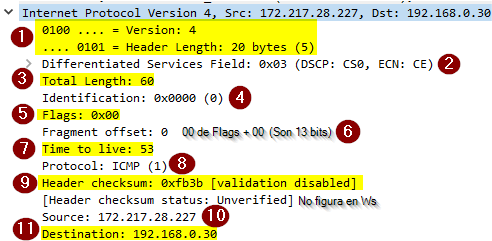
Como se ha visto anteriormente, el tamaño máximo de un datagrama es de 65536 bytes. Sin embargo, este valor nunca es alcanzado porque las redes no tienen suficiente capacidad para enviar paquetes tan grandes. Además, las redes en Internet utilizan diferentes tecnologías por lo tanto el tamaño máximo de un datagrama varía según el tipo de red.

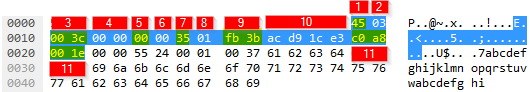
El tamaño máximo de una trama se denomina MTU (Unidad de transmisión máxima). El datagrama se fragmentará si es más grande que la MTU de la red.

|  |  |
| --- | --- |
| Tipo de red | MTU (en bytes) |
| Arpanet | 1000 |
| Ethernet | 1500 |
| FDDI | 4470 |

La fragmentación del datagrama se lleva a cabo a nivel de router, es decir, durante la transición de una red con una MTU grande a una red con una MTU más pequeña. Si el datagrama es demasiado grande para pasar por la red, el router lo fragmentará, es decir, lo dividirá en fragmentos más pequeños que la MTU de la red, de manera tal que el tamaño del fragmento sea un múltiplo de 8 bytes.

El router enviará estos fragmentos de manera independiente y los volverá a encapsular (agregar un encabezado a cada fragmento) para tener en cuenta el nuevo tamaño del fragmento. Además, el router agrega información para que el equipo receptor pueda rearmar los fragmentos en el orden correcto. Sin embargo, no hay nada que indique que los fragmentos llegarán en el orden correcto, ya que se enrutan de manera independiente.





**Ethernet.**

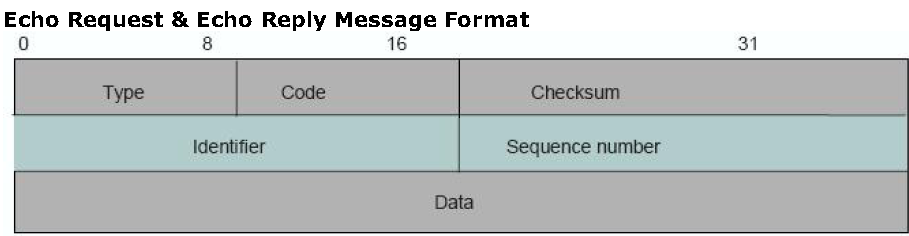
**IPv4:**

1. **Versión / Header Length:** 0x45 (0100|0101) → V: 4 ; HL: 5 (20 bytes)
2. **Tipo de Servicio**: 03
3. **Tamaño:** 0x003C (60 bytes)
4. **Identificación**: 0x0000
5. **Flags:** 0x00 (0|0|0|0 0000) (Reservado|No fragmentado|Más fragmentos|5 bits Posición de Fragmento)
6. **Posición de Fragmento**: (0 0000 del (5)) + 00000 00000 (13 bits en total)
7. **TTL**: 35(53 saltos)
8. **Protocolo:** ICMP (0x01)
9. **Header Check-Sum:** 0xFB3B
10. **IP Fuente**: AC.D9.1C.E3 (172.217.28.228)
11. **IP Destino**: C0.A8.00.1E (192.168.0.30)

**ICMP**





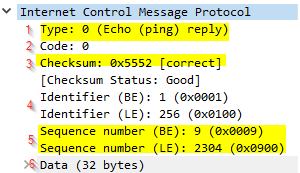


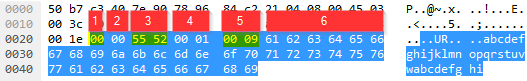
Echo Reply:Tipo 00, Código 00.

Echo Request (Ping): Tipo 08, Código 00.

Tiempo Excedido: Tipo 11 (0x0B), Código 00.

Destino Inalcanzable: Tipo 3. Código Varios.





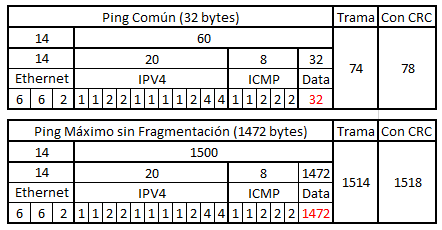
**Ethernet.**

**IPv4.**

**ICMP:**

1. **Tipo:** 0
2. **Código:** 0
3. **Checksum:** 5552
4. **Identificador:** 0001
5. **Número de secuencia:** 0009

**Datos.**

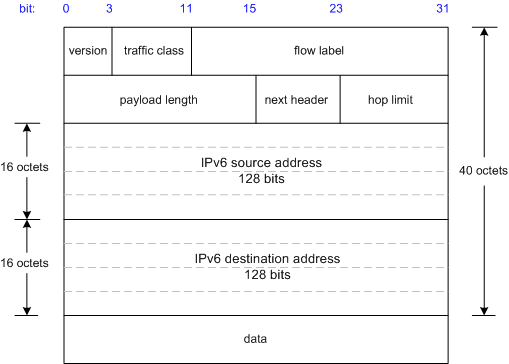


**Fragmentación:**:

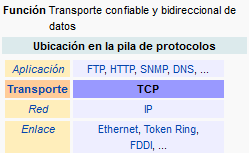
Cuando hay fragmentación, solo se tiene un paquete con cabecera ICMP o UDP, el resto de los paquetes tiene datos de 1480 bytes (+ 20 bytes de la cabecera IPV4 para completar el MTU de 1500 bytes).

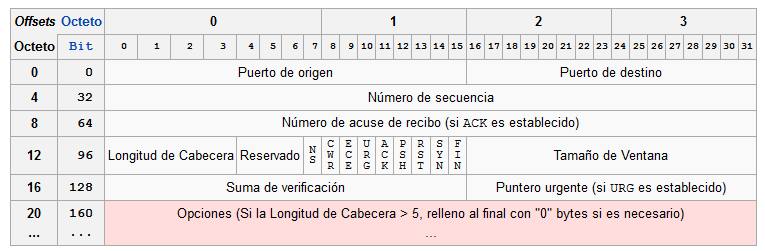
En TCP/IP debido al control de flujo, en cada paquete debe estar la cabecera TCP

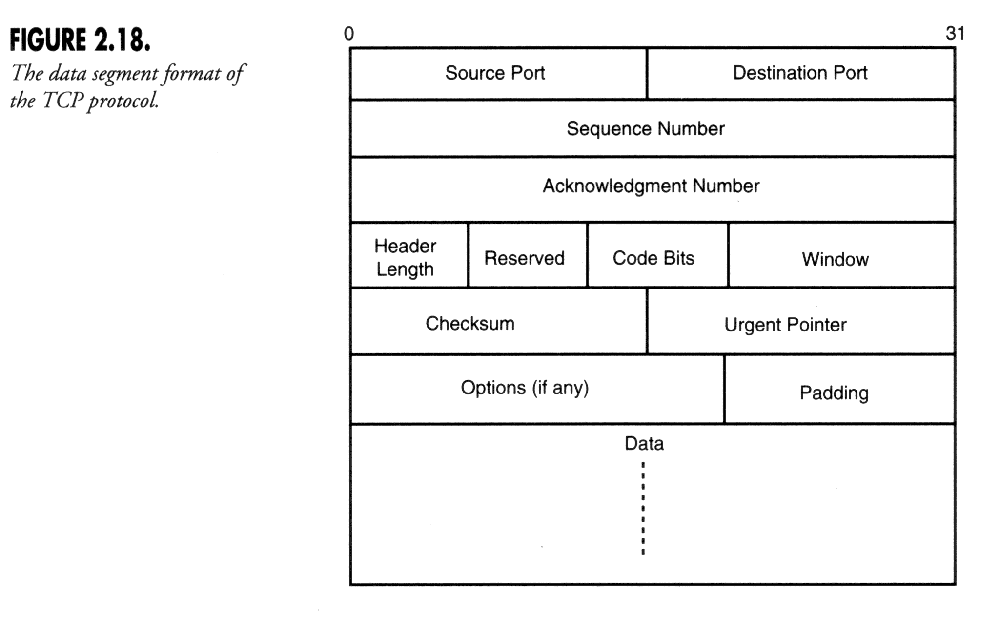
**IPv6**



**TCP (Protocolo de Control de Transmisión)**



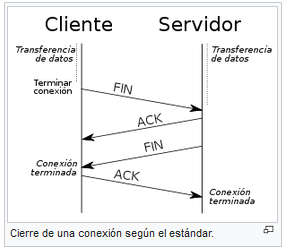
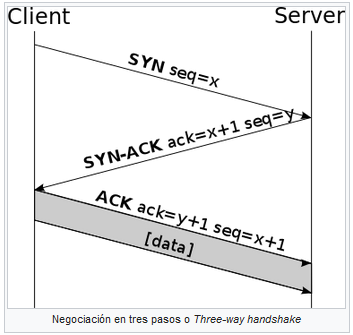




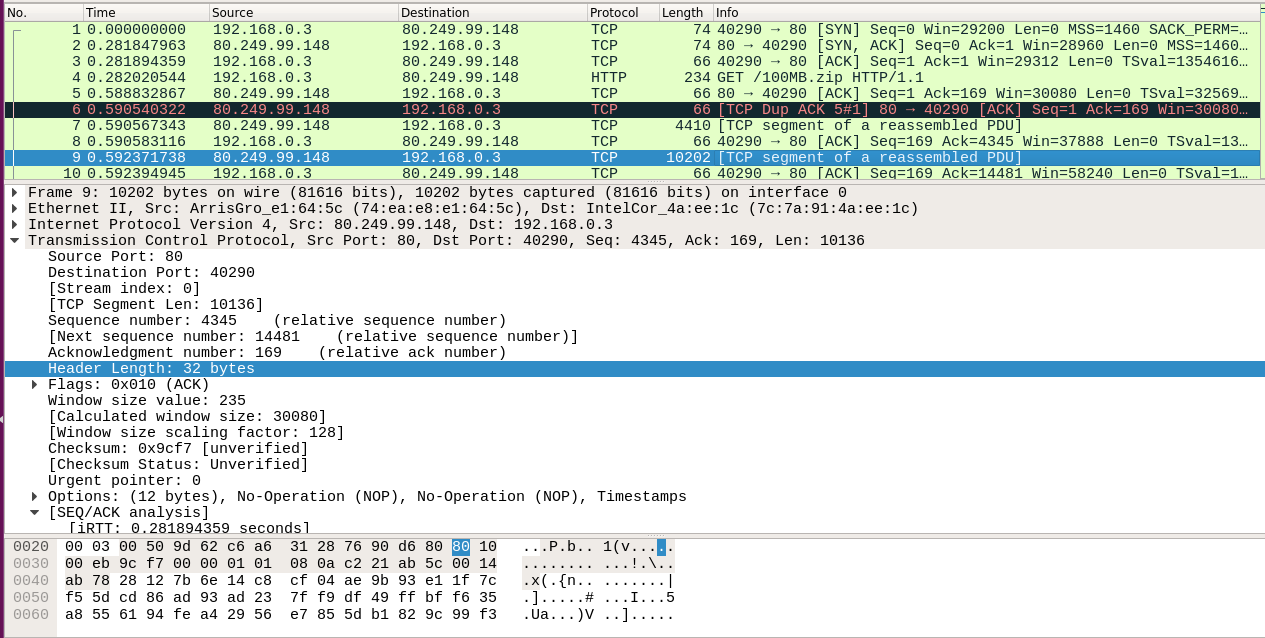
El segmento TCP está compuesto por los datos enviados desde la capa de aplicación y la cabecera añadida por el protocolo de transporte. El segmento TCP es luego encapsulado en un datagrama IP para ser enviado por la capa de red.

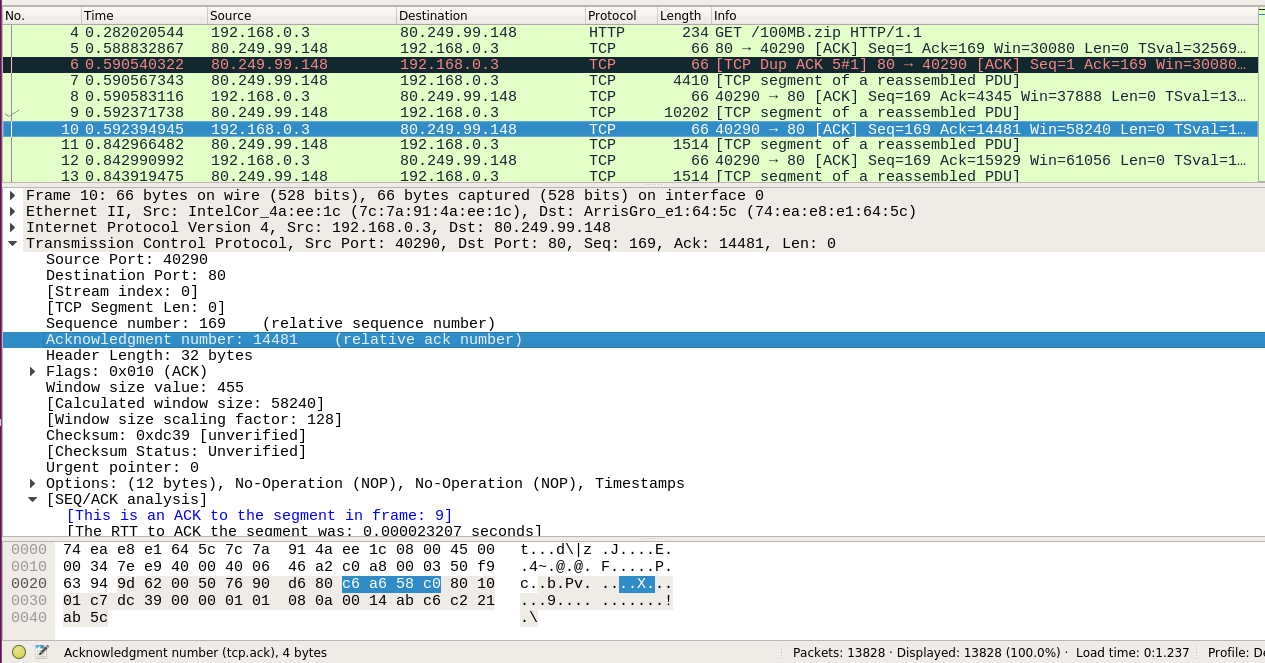
**Puerto origen** (16 bits): Identifica el puerto emisor.  
**Puerto destino** (16 bits): Identifica el puerto receptor.  
Estos dos valores identifican la aplicación receptora y la emisora, junto con las direcciones IP del emisor y receptor identifican de forma unívoca cada conexión. La combinación de una dirección IP y un puerto es llamado socket. Es el par de sockets (dirección IP + puerto del emisor y dirección IP + puerto del receptor) emisor y receptor el que especifica los dos puntos finales que unívocamente se corresponden con cada conexión TCP en internet.  
**Número de secuencia** (32 bits): Identifica el byte del flujo de datos enviado por el emisor TCP al receptor TCP que representa el primer byte de datos del segmento.  
Cuando una conexión está siendo establecida el flag SYN se activa y el campo del número de secuencia contiene el ISN (initial sequence number) elegido por el host para esa conexión. El número de secuencia del primer byte de datos será el ISN+1 ya que el flag SYN consume un número de secuencia.  
**Número de acuse de recibo** (32 bits): Contiene el valor del siguiente número de secuencia que el emisor del segmento espera recibir.  
Una vez que la conexión ha sido establecida, este número se envía siempre y se valida con el flag ACK activado. Enviar ACKs no cuesta nada ya que el campo de acuse de recibo siempre forma parte de la cabecera, al igual que el flag ACK. TCP se puede describir como un protocolo sin asentimientos selectivos o negativos ya que el número de asentimiento en la cabecera TCP significa que se han recibido correctamente los bytes anteriores pero no se incluye ese byte.  
No se pueden asentir partes selectivas del flujo de datos (suponiendo que no estamos usando la opción SACK de asentimientos selectivos). Por ejemplo si se reciben correctamente los bytes 1-1024 y el siguiente segmento contiene los bytes 2049-3072, el receptor no puede asentir este último segmento. Todo lo que puede enviar es un ACK con 1025 como número de asentimiento, al igual que si llega el segmento 1025-2048 pero con un error de cheksum.  
**Longitud de cabecera (4 bits)**: especifica el tamaño de la cabecera en palabras de 32 bits.  
Es requerido porque la longitud del campo “opciones” es variable. Por lo tanto el tamaño máximo de la cabecera está limitado a 60 bytes, mientras que sin “opciones” el tamaño normal será de 20 bytes. A este campo también se le suele llamar “data offset” por el hecho de que es la diferencia en bytes desde el principio del segmento hasta el comienzo de los datos.  
**Reservado** (3 bits): para uso futuro. Debe estar a 0.  
**Flags** (9 bits):  
NS (1 bit): ECN-nonce concealment protection. Para proteger frente a paquetes accidentales o maliciosos que se aprovechan del control de congestión para ganar ancho de banda de la red.  
CWR (1bit): Congestion Window Reduced. El flag se activa por el host emisor para indicar que ha recibido un segmento TCP con el flag ECE activado y ha respondido con el mecanismo de control de congestión.  
ECE (1 bit): Para dar indicaciones sobre congestión.  
URG (1 bit): Indica que el campo del puntero urgente es válido.  
ACK (1 bit): Indica que el campo de asentimiento es válido. Todos los paquetes enviados después del paquete SYN inicial deben tener activo este flag.  
PSH (1 bit): Push. El receptor debe pasar los datos a la aplicación tan pronto como sea posible, no teniendo que esperar a recibir más datos.  
RST (1 bit): Reset. Reinicia la conexión, cuando falla un intento de conexión, o al rechazar paquetes no validos.  
SYN (1 bit): Synchronice. Sincroniza los números de secuencia para iniciar la conexión.  
FIN (1 bit): Para que el emisor (del paquete) solicite la liberación de la conexión.  
**Tamaño de ventana o ventana de recepción** (16 bits): Tamaño de la ventana de recepción que especifica el número máximo de bytes que pueden ser metidos en el buffer de recepción o dicho de otro modo, el número máximo de bytes pendientes de asentimiento. Es un sistema de control de flujo.  
**Suma de verificación** (16 bits): Checksum utilizado para la comprobación de errores tanto en la cabecera como en los datos.  
**Puntero urgente** (16 bits): Cantidad de bytes desde el número de secuencia que indica el lugar donde acaban los datos urgentes.  
**Opciones**: Para poder añadir características no cubiertas por la cabecera fija.  
**Relleno**: Se utiliza para asegurarse que la cabecera acaba con un tamaño múltiplo de 32 bits.

Establecimiento de la conexión (negociación en tres pasos = tres paquetes) / Fin de la conexión (2 pares de paquetes = 4)



**Transmisión de Datos:  
  
Envio de Paquete:**



**ACK**

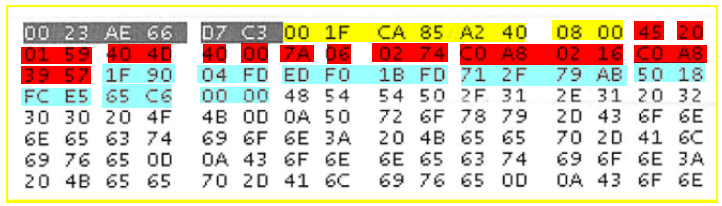
**Consideraciones:**

* **El primer envío tiene la secuencia 4345 y un largo de 10136 de datos**   
  Si bien esto no se indica en la trama, puede verse haciendo   
  10282 del campo Length -14(Head Ethernet) -20(Head IP) - 32 (Head TCP en esta trama en particular) = 10136
* La respuesta envía un ACK sobre la secuencia **14481**, que resulta de sumar la secuencia anterior (**4345**) al largo (**10136**)

**Fragmentación con IP/TCP:**

Debido al control de flujo, en cada paquete debe estar la cabecera TCP.

**Trama de Ejemplo:**



**Ethernet:**

**MAC Destino:** 00 23 AE 66 D7 C3

**MAC Fuente:** 00 1F CA 85 A2 40

**Protocolo:** IPv4(0800)

**IPv4:**

**Versión / Header Length:** 45 (0100|0101) → V: 4 ; HL: 5 (20 bytes)

**Tipo de Servicio**: 20

**Tamaño:** 01 59 (345 bytes)

**Identificación**: 404D

**Flags:** 40 (0|1|0|0 0000) (Reservado|**No fragmentado**|Más fragmentos|5 bits Posición de Fragmento)

**Posición de Fragmento**: 0 0000 00000 00000 (13 bits en total: Primero 5 de Flags + 8)

**TTL**: 7A (122 saltos)

**Protocolo:** TCP (0x06)

**Header Checksum:** 0274

**IP Fuente**: c0 a8 02 16 (192.168.2.22)

**IP Destino**: c0 a8 39 57 (192.168.57.87)

**TCP**:

**Puerto fuente:** 1F90 (8080)

**Puerto destino:** 04FD (1277)

**Número de Secuencia:** ED F0 1B FD (3991935997)

**Número de Ack:** 713F79AB (1899985323)

**Tamaño de Cabecera:** 50 (**0101** 0000) → 5\*4 = 20 bytes

**Flags**: 000|0|0|0|0|1|1|0|0|0 (5018: 0101 **0000 0001 1000**) → Ack = 1 | Psh = 1

(Res|NS|CWR|ECE|URG|ACK|PSH|RST|SYN|FIN)

**Tamaño de Ventana**: FC E5 (64741 bytes)

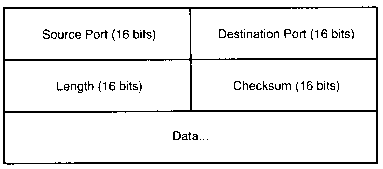
**Checksum**: 65 C6

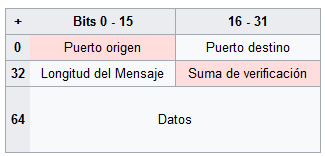
**Puntero Urgente:** 0000

**Opciones** + **Datos**

**UDP**







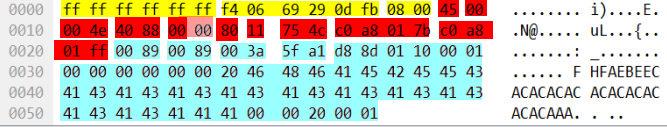
**Puerto Origen**: Opcional

**Longitud del Mensaje**: tamaño en bytes del datagrama UDP incluidos los datos (Mínimo 8 bytes).

**Suma de Verificación**: Opcional

El 0x0800 establece que el protocolo encapsulado es iPV4

El 0x11 (17) establece que el protocolo encapsulado es UDP.



**Ethernet:**

**Mac Destino:** FF.FF.FF.FF.FF.FF

**Mac Fuente:** F4.06.69.29.0D.FB

**Protocolo:** IPv4 (0x800)

**IPv4:**

**Versión / Header Length:** 45 (0100|0101) → V: 4 ; HL: 5 (\*32 bits = 20 bytes)

**Tipo de Servicio**: 0

**Tamaño:** 004E (78 bytes)

**Identificación**: 40 88

**Flags:** 0 (0|0|0|0 0000) (Reservado|No fragmentado|Más fragmentos|5 bits Posición de Fragmento)

**Posición de Fragmento**: 0 0000 00000 00000 (13 bits en total: Primero 5 de Flags + 8)

**TTL**: 80 (128 saltos)

**Protocolo**: UDP (0x11)

**Header Checksum:** 754C

**IP Fuente**: C0 A8 01 7B (192.168.1.123)

**IP Destino**: C0 A8 01 FF (192.168.1.255)

**UDP**:

**Puerto Origen**: 0089 (137)

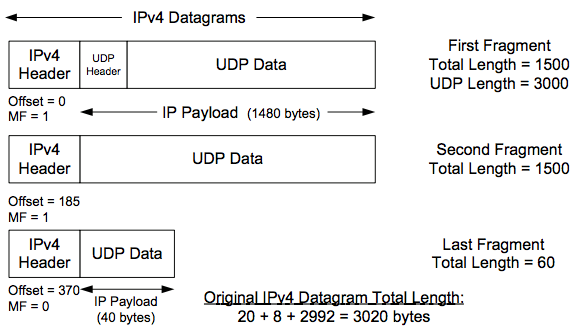
**Puerto Destino**: 0089 (137)

**Tamaño Mensaje**: 003A

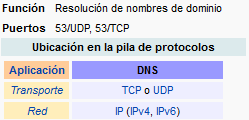
**Checksum**: 5FA1

**Datos**.

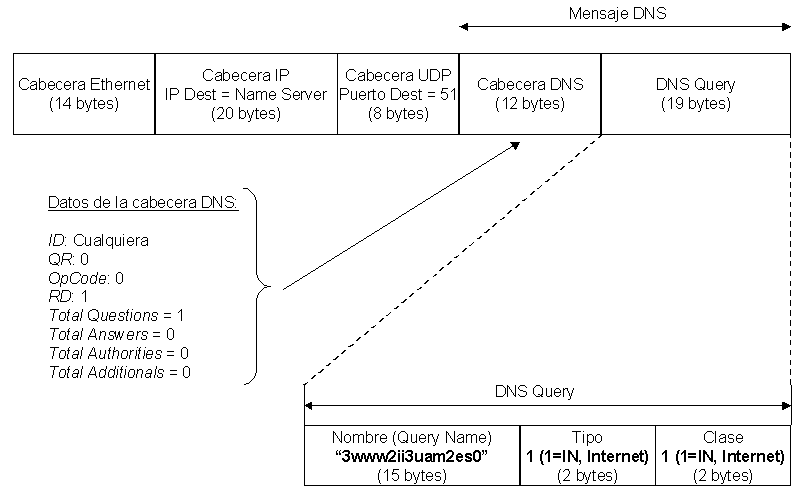
**Fragmentación IP con UDP:**



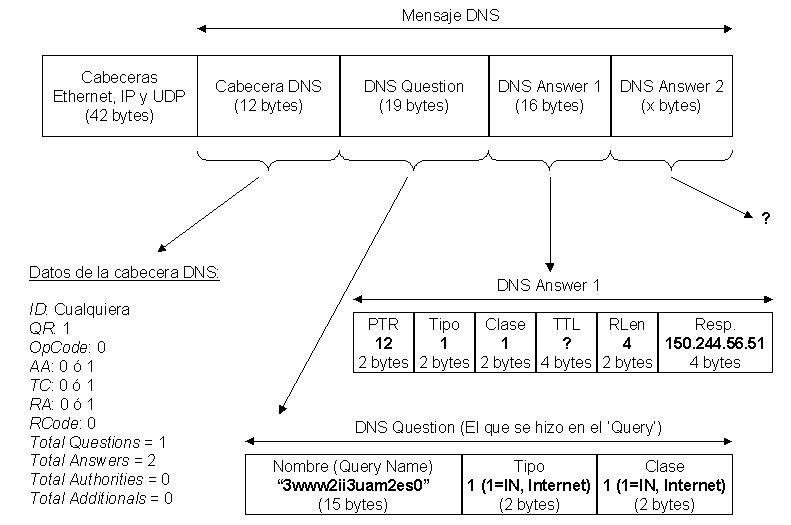
**DNS**:



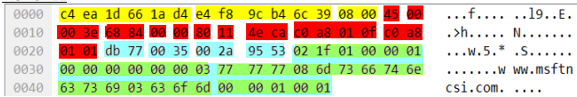
Pregunta:



Respuesta:



Ejemplo:



Ethernet → IP4 → UDP → DNS

La segunda franja verde, que empieza con 77 y termina con 6d corresponde al texto www.msftncsi.com

|  |  |
| --- | --- |
| **HTTP: Hypertext Transfer Protocol** | **HTTPS: Hypertext Transfer Protocol Secure** |
| **DHCP (Dynamic Host Configuration Protocol)** | **SNMP (Protocolo Simple de Administración de Red)** |
| **SMTP (Simple Mail Transfer Protocol)** | **POP3 (Post Office Protocol)** |
| **IMAP** (**Internet Message Access Protocol)** | **FTP (File Transfer Protocol)** |

|  |  |
| --- | --- |
| TCP |  |
| Ordenamiento y reensamble | Sí, a través del uso de números de secuencia y asentimiento, TCP puede pasar los segmentos recibidos en el orden correcto dentro del flujo de bytes a la aplicación receptora. |
| Fiabilidad | Sí. El receptor TCP utiliza los números de secuencia para reorganizar los segmentos cuyo lleguen fuera de orden y para eliminar segmentos duplicados. |
| Control de errores - Reconocimientos y retransmisiones | Durante la etapa de transferencia de datos, una serie de mecanismos claves determinan la fiabilidad y robustez del protocolo. Entre ellos están incluidos el uso del número de secuencia para ordenar los segmentos TCP recibidos y detectar paquetes duplicados, checksums para detectar errores, y asentimientos y temporizadores para detectar pérdidas y retrasos.  El receptor puede pedir la retransmisión de un paquete. Si el paquete no es notificado como recibido (ACK), el transmisor envía de nuevo el paquete. |
| Control de flujo - El modelo de ventana aplicado en TCP | TCP usa una [ventana](https://es.wikipedia.org/wiki/Ventana_deslizante) deslizante para el control de flujo. En cada segmento TCP, el receptor especifica en el campo receive window la cantidad de bytes que puede almacenar en el búfer para esa conexión. El emisor puede enviar datos hasta esa cantidad. Para poder enviar más datos debe esperar que el receptor le envíe un ACK con un nuevo valor de ventana. |
| Multiplexación | Sí, usa puertos |
| Conexión Full Duplex | Una conexión TCP es un par de circuitos virtuales, cada uno en una dirección. Sólo los dos sistemas finales sincronizados pueden usar la conexión. |

|  |  |
| --- | --- |
| UDP |  |
| Conexión | No |
| Fiabilidad | No |
| Control de Flujo | No |
| Control de Errores | No. Se detectan por el checksum |
| Multiplexación | Sí, usa puertos |