CAPÍTULO 5
EL PROTOCOLO DE INTERNET (IP)

Luis Marrone

Nivel de Red

Comenzamos esta historia desde el momento en que necesitamos acceder a una página Web y llegamos hasta ver el intercambio de mensajes que se produce entre el navegador en nuestra PC y el servidor donde reside la página Web que queremos acceder.

Nos queda entonces averiguar como pueden esos mensajes ir y venir entre nuestra PC y el servidor. La respuesta, en una primera aproximación, está en que ambos actores están conectados o forman parte de una red. Pues bien, ¿qué es una red?.

No hay una única respuesta, Si le queremos dar un enfoque descriptivo diremos que es un conjunto de nodos interconectados. Pero no nos dice nada más. Tal vez tendremos un mayor grado de información y de comprensión de la misma si pensamos en la funcionalidad que brinda. Justamente con eso volvemos al párrafo anterior. Los mensajes pueden ir y venir entre los actores por el simple hecho de estar conectados. Encontramos así la funcionalidad de toda red, la de interconexión de sus integrantes.

Esa interconexión quedará instanciada por medio de un protocolo, como ya hemos visto en otras oportunidades y que en este caso será un protocolo de red, que integra el Nivel de Red del modelo TCP/IP que adoptamos.

Efectivamente la función primordial de un nivel de red es la de dar conectividad a sus miembros. Esa conectividad se puede lograr como en el caso de una comunicación telefónica. Discamos el número del destino y si el destino nos atiende y nos reconoce puede que acepte la llamada o simplemente corte. Otra forma de lograrla es como en le caso del correo postal. Escribimos una carta, la introducimos en un sobre en el que escribimos por lo menos el destino y lo arrojamos al buzón más cercano. Eso es todo, no sabemos si llegó la carta a destino o si el destino la leyó.

Justamente esas dos analogías se corresponden con funcionalidades diferentes que podemos encontrar en una red. El caso de la comunicación telefónica se corresponde con el de un servicio con conexión. Le avisamos al destino que queremos conectarnos con él. Si acepta recién ahí comenzamos a intercambiar datos. Debemos establecer una sesión previa al intercambio de datos. Algo así como lo que pasaba en TCP, ¿no?

En el otro caso decimos que la red da un servicio sin conexión. Enviamos el mensaje directamente. Cada mensaje lo enviamos en un sobre, de modo que pueden seguir diferentes rutas para llegar al mismo destino. Cada mensaje es independiente de los anteriores, no hay una secuencia, pueden llegar a destino en distinto orden al que los ingresamos en la red. Los protocolos que implementan este tipo de servicio

se los suele calificar como "best effort" por cuanto dada la independencia de cada mensaje hace que también no se pueda implementar dentro del protocolo funcionalidades de control que lo hagan más confiable.

Independientemente que el protocolo que implemente este nivel sea con conexión o sin ella debe definir el intercambio entre el usuario y la red y entre los nodos de la red. Esta implementación resulta crítica por cuanto generó dos paradigmas de red.

Si el protocolo hace pública solamente la especificación de los mensajes Usuario-Red y la de la interface Nodo-Nodo es privada decimos que llevará a un paradigma de Red Cerrada. Mientras que si es pública tanto la interface Usuario-Red como la de Nodo-Nodo estaremos en presencia de una Red Abierta. La ventaja de una Red Abierta es que queda garantizada la interoperabilidad de los componentes provenientes de diferentes fabricantes, no generando de esta forma clientes cautivos.

Si bien la primera red de datos propuso una arquitectura de red abierta, su implementación estaba basada en una plataforma como UNIX, relegada en esos momentos, comienzos de la década del 70, a entornos académicos y/o de proyectos especiales, ausente en el campo comercial. En parte eso hizo que en los comienzos se difundiera la otra propuesta, la de red cerrada. Con el devenir del tiempo, al adquirir una mayor difusión plataformas basadas en Unix hicieron que el modelo de red abierta con las ventajas mencionadas comenzase a ganar terreno. Es así que en la actualidad es el modelo que prima a la hora de implementar un protocolo de red.

Protocolo IP

El protocolo IP surgió de un proyecto del Departamento de Defensa de los Estados Unidos llamado DARPA en 1969. Dentro del mismo proyecto surgieron también otros protocolos que conforman el conjunto de protocolos TCP/IP. Debido a esto muchas veces se hace referencia al protocolo TCP/IP cuando en realidad es un conjunto de protocolos. De ese conjunto ya vimos en el capítulo anterior TCP.

En 1983 el nuevo conjunto de protocolos TCP/IP fue adoptado como estándar y finalmente se convirtió en el más usado en redes y el protocolo estándar de internet. En particular IP incialmente especificado en [Pos81a] forma parte junto con otros protocolos auxiliares y recomendaciones del Estándar 5 como vemos en la Figura 5.1.

Fuera de este estándar están los Estándares 37 y 38 que especifican sendos protocolos auxiliares, ARP [Plu82] y IARP [FMMT84] que veremos en un próximo capítulo.

El protocolo IP al cual hacemos referencia es IPv4, versión 4, formando parte del Estándar 5. Ocurre que con el correr del tiempo y la difusión y crecimiento exponencial de Internet surgieron problemas con IPv4 como:

- Agotamiento de las direcciones
- Necesidad de Calidad de Servicio, (QoS)
- Mejora de performance $---->$ Reducción de overhead

entre otras cosas.

Para dar una solución integral a estas principales cuestiones es que se decidió desarrollar una nueva versión. En diciembre de 1993 la IETF publica la RFC 1550 donde invita a propuestas de IPng (Next

Number	Title	Authors	Date	More Info	Status
RFC 791 part of STD 5	**Internet Protocol**	J. Postel	September 1981	**Errata**, Obsoletes RFC 760, Updated by RFC 1349, RFC 2474, RFC 6864	Internet Standard
RFC 792 part of STD 5	**Internet Control Message Protocol**	J. Postel	September 1981	**Errata**, Obsoletes RFC 777, Updated by RFC 950, RFC 4884, RFC 6633, RFC 6918	Internet Standard
RFC 919 part of STD 5	**Broadcasting Internet Datagrams**	J.C. Mogul	October 1984		Internet Standard
RFC 922 part of STD 5	**Broadcasting Internet datagrams in the presence of subnets**	J.C. Mogul	October 1984		Internet Standard
RFC 950 part of STD 5	**Internet Standard Subnetting Procedure**	J.C. Mogul, J. Postel	August 1985	Updates RFC 792, Updated by RFC 6918	Internet Standard
RFC 1112 part of STD 5	**Host extensions for IP multicasting**	S.E. Deering	August 1989	Obsoletes RFC 988, RFC 1054, Updated by RFC 2236	Internet Standard

Figura 5.1: Estándar 5 - IETF - ISOC

Generation). Se llega a una primera versión especificada en la RFC 1883 de diciembre de 1995 como IPv6. ¿Por qué no IPv5?, ocurre que en octubre de 1990 se publica con estatus experimental la RFC 1190: Experimental Internet Stream Protocol, Version 2 (ST-II), con datagramas similares a los de IPv4 en los que el header tiene un campo de versión 5. Continuando con IPv6, digamos que adquiere la mayoría de edad en julio de 2017 con la RFC 8200 [DH17], Estándar 86. Abordaremos más en detalle un análisis de IPv4 y daremos al final del capítulo características distintivas de IPv6.

Para hacernos una idea de redes basadas en IP les presentamos el siguiente esquema, Figura5.2

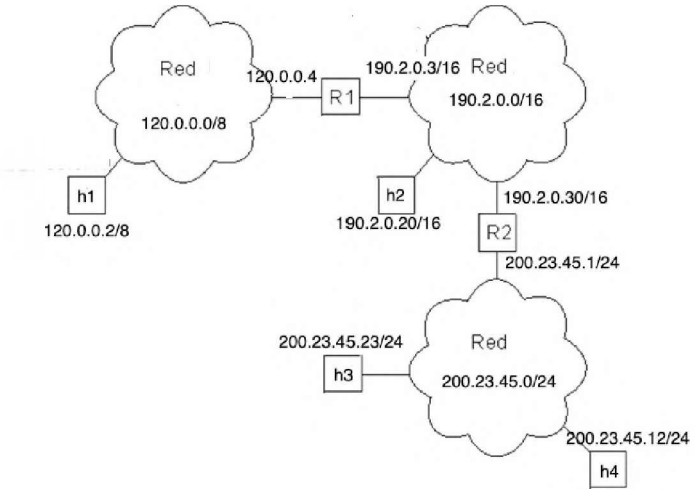


Figura 5.2: Red IP

Los equipos identificados con h son los usuarios o hosts de las redes, identificados por esas direcciones numéricas. Vemos que las redes también se identifican con el mismo formato que las direcciones de los hosts y como IP se definió entre otras cosas para interconectar redes vemos el elemento que materializa

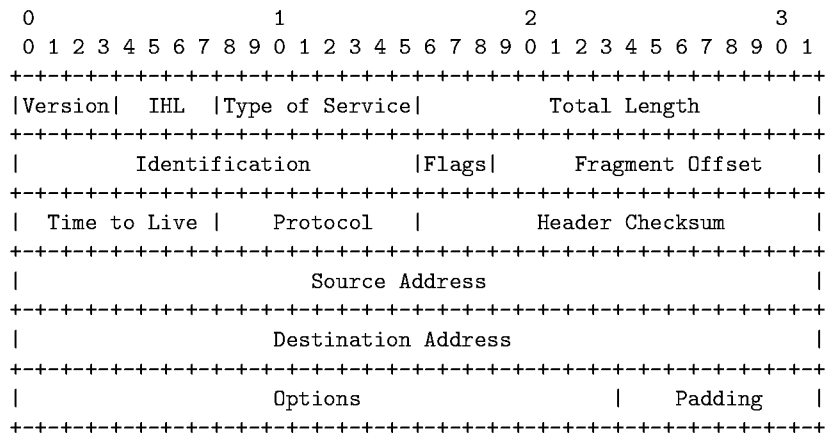
esa interconexión que es el Router, R1 y R2 en la figura, que en [Pos81a] lo denomina Gateway.

IPv4

La característica primordial de este protocolo es que se trata de uno sin conexión. Los mensajes son independientes, no hay una secuencia en el envío. La independencia y las características de su envío a través de la red hace que puedan seguir caminos diferentes y por lo tanto arriben al destino en un orden diferente al que fueron enviados. No hay control de errores, solo una verificación por suma binaria del encabezado. Características estas del protocolo que lo resumen a un protocolo "best effort".

Como mencionamos anteriormente al ser un protocolo de nivel de red garantiza la conectividad entre los miembros de la red o de las redes interconectadas. Es conectividad se logra a través del mecanismo de despacho ("forwarding") de los datagramas que realizan los routers en base a la dirección destino del datagrama que reciben y el contenido de su tabla de ruteo.

La estructura de datos o Protocol Data Unit (PDU) de IPv4 es el datagrama:



Como vemos está ordenado en palabras de 32 bits. En realidad es el encabezado. A continuación del Padding vienen los datos con un número entero de bytes.

En una descripción somera encontramos:

Version: 4

IHL: Longitud de la cabecera. ¡Cuidado! está dado en palabras de 32 bits.

Type of Service: Destinado al soporte de Calidad de Servicio (QoS).

Total Length: Longitud completa del datagrama en bytes.

2da. Palabra: Destinada a controlar la fragmentación del datagrama.

Time to Live: Inicialmente el tiempo de vida del datagrama luego cantidad de saltos que podía tener el datagrama antes de ser eliminado.

Protocol: Identifica el contenido del campo de datos o payload.

Header Checksum: Verificación del encabezado.

4ta y 5ta. palabras: Direcciones origen y destino.

Options-Padding: Está presente si el datagrama contiene opciones.

Para la descripción detallada de cada uno de los campos remitimos a [Pos81a] y sus actualizaciones.

Veamos la mezcla de unidades:

IHL: en unidades de palabras de 32 bits.

Total Length: en bytes.

Fragment Offset: Está dado en múltiplos de 8 bytes.

Los quisimos destacar para que recuerden las diferentes unidades de las magnitudes de esos campos. Con respecto a las "Options" tengan en cuenta que están ordenadas al byte, pero como el encabezado está en palabras de 32 bits por eso aparece un campo de "Padding" para que con un relleno se llegue a las unidades necesarias.

La segunda palabra del encabezado identifica si el datagrama es parte de otro datagrama (fragmentado) y también va a utilizarla el destino para recomponer el datagrama a partir de los fragmentos.

Las opciones completan la funcionalidad del protocolo aunque son raramente utilizadas, y sus prestaciones fueron incorporadas a las opciones de TCP.

Como una red IP es un caso de red abierta la conectividad será lograda por medio de IP solamente que residirá tanto en los hosts (origen y destino) como en los routers. en los hosts se deberá entonces activar el stack de IP y configurarlo. Para ello se requiere fijar:

- Su dirección IP.
- La dirección IP del Default Gateway. El router al cual derivará los datagramas que tienen por destino una red diferente a la que pertenece.
- La dirección IP del servidor de DNS.

Direcciones IP

El protocolo IP permite direccionar a un dispositivo mediante direcciones unicast; a un grupo de ellos con direcciones multicast, independientemente de la red a la que pertenezcan y a todos los integrantes de una red, con direcciones broadcast.

Nos ocuparemos de las direcciones unicast. Podemos verlas en la Figura 5.2. El formato se corresponde con una estructura jerárquica de direcciones que nos permite identificar la red y el host dentro de cada red. La desventaja es que perdemos direcciones, cosa que no ocurriría en un esquema plano. Tengamos en cuenta que en este no tenemos la capacidad de identificación del jerárquico.

Más allá de la estructura jerárquica se definieron clases para contemplar redes de diferentes capacidades. Así tenemos, tomando como referencia la Figura 5.2.

Clase A: La dirección de h1. El primer campo, 120 identifica la red y los tres campos restantes a h1 propiamente. ¿Como sabemos que es una clase A?. Porque el bit más significativo del primer campo está en **0**. La red también queda identificada con el mismo formato, con los campos de host en 0, 120.0.0.0/8. /8 en la notación actual es lo que se llama prefijo e identifica la cantidad de bits que determinan la red.

Clase B: La dirección de h2. Los dos primeros campos, 190.2 identifican la red y los dos restantes a los hosts. ¿Cómo sabemos que es una clase B? porque los dos bits más significativos del primer campo están en **10**. Vemos entonces que el prefijo correspondiente será /16

Clase C: La dirección de h3. Con el mismo criterio los tres bits más significativos del primer campo están en **110**.

Las direcciones broadcast también tienen el formato de las unicast con la particularidad que todos los bits que identifican al host están en **1**, por ejemplo en el caso de nuestra red 120.0.0.0/8, será 120.255.255.255/8.

Con esto último vemos que efectivamente en toda red perdemos dos direcciones para asignarlas a un host, la que identifica a la red y la de broadcast.

¿Qué pasa con las demás posibilidades?.

Clase D: bits más significativos del primer campo en **1110**. es la clase para identifcar los grupos multicast en los que no hay una identificación de red y host, sólo el grupo multicast.

Clase E: bits más significativos del primer campo en **1111** es una clase experimental.

Algo más. Dentro de las clases A, B y C se definen direcciones privadas que no pueden asignarse a redes que se quieran integrar a una red pública como Internet. Son estas:

Direcciones Privadas
Clase A: 10.0.0.0/8 a 10.255.255.255/8
Clase B: 172.16.0.0./16 a 172.31.255.255/16
Clase C: 192.168.0.0/24 a 192.168.255.255/24

Recomendamos consultar [RMKdG96] para mayores detalles.

Subredes

Máscara fija - Máscara variable. A mediados de la década de los noventa comienza un crecimiento exponencial de usuarios. De 45 millones en 1996 llegamos a 4.910 millones en 2021[1]. Con ese crecimiento surge un problema grave, el agotamiento de las direcciones. Ya comentamos la pérdida experimentada debido al esquema jerárquico de direcciones y en menor grado con la definición de direcciones privadas.

En los orígenes se había definido un procedimiento para un mejor aprovechamiento de las direcciones [MP85], que ya a comienzos del 2000 adquiere una difusión en parte obligada.

Este procedimiento popularizado como "subnetting" no es otra cosa que generar a partir de una red, subredes, manteniendo el mismo esquema de direcciones , identificador de red y host. Si bien se habla de subredes, esto es más que nada por el hecho que se subdivide una red en otras, pero esas subredes son manejadas por IP como cualquier red. Para ello se define la máscara que determina la cantidad de bits que ahora van a identificar a una red o subred.

La máscara está especificada bajo el formato de una dirección IP, cuatro campos decimales, separados por punto y en el rango de 0 a 255. Como la máscara indica la cantidad de bits en 1 que identifican a la red entonces la máscara de una red Clase A sin subnetting será 255.0.0.0, dado que los cuatro primeros bits identifican a una red Clase A. Para dar otro ejemplo la de una Clase B será 255.255.0.0.

[1] https://www.internetlivestats.com/internet-users/

Con este procedimiento, a partir de una red Clase C podremos subdividirla en redes definiendo una máscara que nos permite tomar bits del campo de hosts e incorporarlos a los que definen la red; claro a costa de tener menos hosts por red.

Por ejemplo si disponemos de la red 200.23.45.0 y necesitamos dividirla en 5 redes, entonces tomaremos, para un mejor manejo de las direcciones los 3 bits más significativos del último campo. Tener en cuenta que el único campo disponible para subdividir es el de los hosts.

La máscara será entonces 255.255.255.192 dado que el último campo tiene los 3 bits más significativos en 1. Supongamos que para identificar una nueva red fijamos los 3 bits en 010. Entonces esa nueva red quedará identificada por 200.23.45.64. Recordar que 64 en binario es **0100 0000**. Aprovechamos y decimos que la dirección broadcast de esa red es 200.23.45.95.

Un llamado de atención, si nos presentan la dirección 200.23.45.64 y no aclaran que está "subneteada", entonces diremos que es una dirección de host de 200.23.45.0. Es por ello que las direcciones deben especificarse acompañadas de la máscara correspondiente para evitar la ambigüedad mencionada.

Para especificar una dirección en forma más compacta se definió el prefijo que indica directamente la cantidad de unos que representan a la red. En nuestro ejemplo sería:

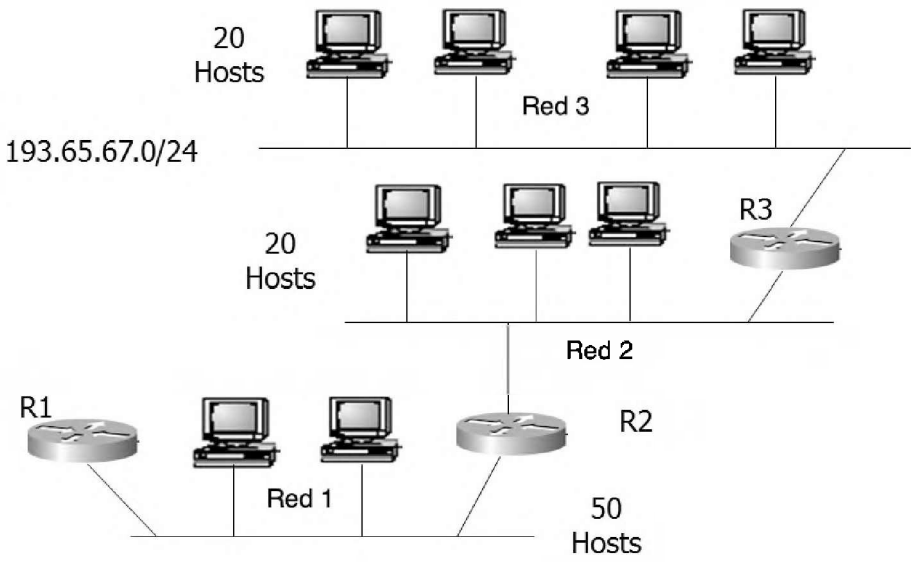
Notación Moderna
Prefijo: /27
Red: 200.23.45.64/27
Broadcast: 200.23.45.95/27

Este procedimiento permite generar redes acorde con la capacidad necesaria en cada red. Sin subnetting todas las redes con menos de 254 hosts deberían ser /24. Si necesitamos una red de 36 hosts entonces podemos generar una red /26 y no una /24, donde desperdiciaríamos 218 direcciones. A la hora de configurar redes con subnetting no nos olvidemos de considerar los ports de los Routers de la red que ocupan una dirección IP como cualquier host. Algo más a tener en cuenta, En la [MP85] se indicó que los bits de extensión o subred no podían estar todos en 0 o todos en 1 por ambigüedades con la dirección de red original y el broadcast también de esa red. Luego esa ambigüedad fue resuelta por los Routers y se aceptan esas asignaciones.

Pero a la hora de optimizar las direcciones falta algo por hacer. Consideremos el requerimiento indicado en la Figura 5.3, atendiendo a los requerimientos de la [MP85]:

Si abordamos la solución como en el caso anterior veremos que no hay solución posible. Para definir tres redes necesitamos tomar 3 bits del campo de host dela red /24. Con 3 bits nos quedan 30 direcciones por red y en una de ellas necesitamos asignar 52.

La solución fue hacer la división de redes con diferentes máscaras, lo que se dio en llamar *VLSM*(Variable Length Subnet Mask). Es decir se va a ajustar la máscara acorde con la capacidad requerida en cada



Dirección disponible : 193.65.67.0/24

Red 1: 52 direcciones. Recuerden los ports de los Routers

Red 2: 22 direcciones.

Red 3: 21 direcciones

Figura 5.3: Requerimiento

red. Para mayores detalles consultar [BP87] y [PM95]. Entonces tendríamos la siguiente asignación:

Dirección disponible : 193.65.67.0/24.

Red 1: 193.65.67.64/26.

Red 2: 193.65.67.128/27.

Red 3: 193.65.67.160/27.

La mayor cantidad de errores que se producen en este tipo de diseño la de asignar direcciones de red que están incluidas en otras y por lo tanto son en realidad direcciones de host. En nuestro ejemplo al asignar dirección a la Red 1 con los bits en **01** con /26 las asignaciones para Red 2 y Red 3 con /27 hace que los dos bits más significativos del último campo no pueden ser **01**.

Para terminar con el subnetting diremos que los router debieron soportar el VLSM, por lo cual se actualizaron los protocolos de ruteo de modo que al publicar un ruta incluyeran la máscara/prefijo correspondiente.

Con el crecimiento de Internet apareció un nuevo actor, el ISP (Internet Service Provider), que por ejemplo puede necesitar 1500 direcciones. Ese número requiere de una Clase B, pero que difícilmente esté alguna disponible. En su lugar se le dan 8 Clases C contiguas

$$194.32.136.0/24 - -194.32.143.0/24$$

El rango de direcciones considerado tiene los primeros 21 bits iguales por lo que de alguna manera constituyen un prefijo y entonces podemos identificar el bloque como:

$$194.32.136.0/21$$

Lo curioso es que 194 nos delataría una clase C, pero ocurre que el prefijo es /21 cuando, por lo que vimos antes, deberíamos esperar /24 o mayor. ¿Qué pasa?

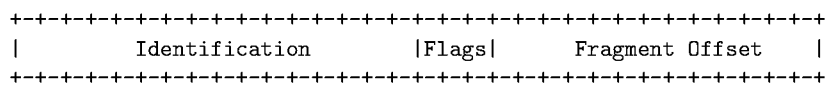
Se acabaron las clases. Para poder identificar correctamente a una dirección basta con tener en cuenta el prefijo. Con esta consideración y con el soporte por parte de los Routers de forwardear esa dirección surge el CIDR, "**C**lassless **I**nter-**D**omain **R**outing". Si lo vemos desde el punto de vista de las direcciones se lo llama Supranetting. En el caso de Subnetting avanzamos por derecha sobre los bits de hosts, con Supranetting avanzamos por izquierda sobre los bits de red. Tiene una especificación inicial en la RFC 1338 pero damos como mejor referencia la RFC 4632 [FL06].

Fragmentación

En la 5 comentamos que la segunda palabra del datagrama estaba destinada al control de la fragmentación. ¿Por qué se fragmentan los datagramas?. Tengamos en cuenta que IP fue pensado para la interconexión de redes y por lo tanto desde el origen hasta el host destino el datagrama puede pasar por diferentes redes. Esas redes pueden trabajar con una longitud máxima de PDU diferente y por lo tanto el router a la hora de "forwardear" un datagrama se encuentra con que la longitud supera al MTU de la red a la que lo tiene que forwardear. Entonces lo fragmenta y encargamos al destino que rearme el rompecabezas.

¿Por qué el destino?. Recordemos que IP es best effort y por lo tanto los datagramas fragmentos pueden seguir caminos diferentes y por lo tanto puede ocurrir que no todos los datagramas pasen por un mismo Router. El destino es el único, que de no perderse ningún fragmento los recibirá a todos ellos.

Veamos la segunda palabra del datagrama:



El campo Identification de 16 bits contiene bits aleatorios dados por el Sistema operativo cuando se genera el datagrama, de modo que todos los fragmentos tendrán ese mismo identificador y será usado por el destino para separar las piezas de un mismo rompecabezas, es decir de un mismo datagrama. Por seguridad también tendrá en cuenta las direcciones IP.

El campo Flags de tres bits contiene un primer bit sin definir, el segundo **D**, que estando en 1 impide que el datagrama sea fragmentado y el tercero **M**, que si está en 1 indica que hay más fragmentos.

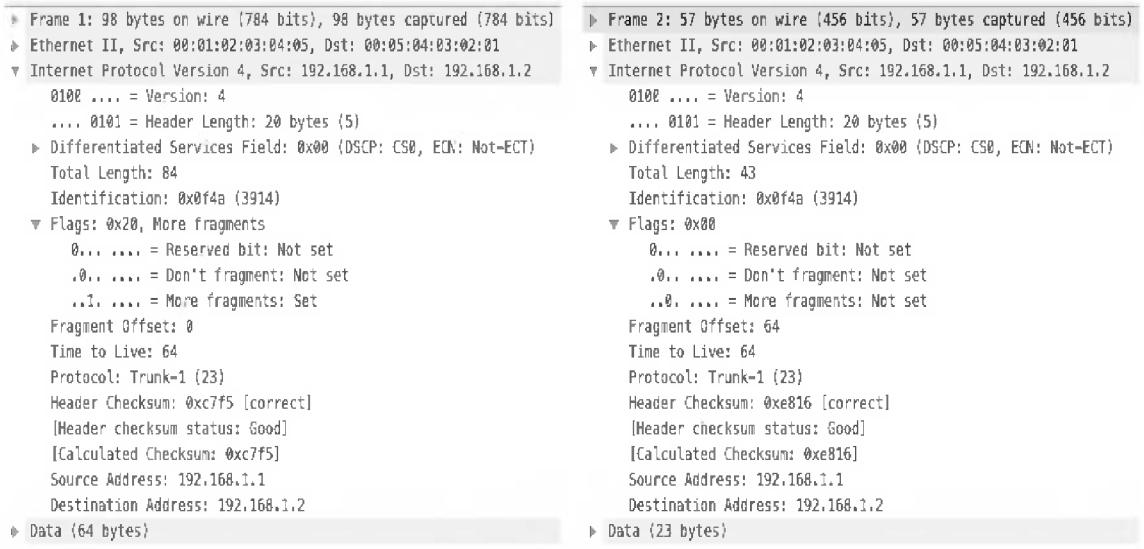
El campo Fragment Offset nos indica en múltiplos de 8 bytes la posición relativa del payload de ese fragmenteo en el payload del datagrama original. Campo necesario dado que los fragmentos pueden llegar desordenados.

¿Cómo se da cuenta el destino que recibe un fragmento?. Si es el primero el Offset estará en 0 y el Flag **M** estará en 1. Si es el último entonces el Offset será distinto de 0 y el flag **M** estará en 0. Si es un fragmento pero no el primero ni el último entonces el Offset será distinto de 0 y el flag **M** estará en 1.

¿Cómo sabe que no es un fragmento?. Porque el Offset y el Flag **M** estarán en 0.

¿Qué pasa si el Router recibe un datagrama que debe fragmentar pero el flag **D** está en 1?, pues simplemente lo descarta y si está habilitado el protocolo ICMP que veremos en sesión próxima, enviará al origen un mensaje indicando la causa por la que lo descartó.

Completemos el análisis de la fragmentación mediante la captura de dos tramas de la Figura 5.4



(a) Primera Trama (b) Segunda Trama

Figura 5.4: Datagrama Fragmentado

En la primera trama vemos en el campo de Flags que el flag **M** está en 1 y el Fragment Offset en 0, lo que os indica que es un datagrama fragmentado.

En la segunda, observando los mismos campos vemos que el flag **M** está en 0 y el Fragment Offset en 64.

Por último teniendo los dos datagramas el mismo ID y las direcciones IP no cabe duda que se trata de los fragmentos de un mismo datagrama.

Si nos interesa la longitud del payload del datagrama original podemos extraer del primer fragmento que su payload es de 64 bytes (80 bytes de longitud total del datagrama - 20 bytes del encabezado). 64 es múltiplo de 8 como debe ser la longitud de los fragmentos, salvo el último por supuesto.

Del segundo fragmento obtenemos 23 bytes. En este segundo fragmento observamos que el Fragment Offset es 64, en realidad el Wireshark nos da su contenido en bytes. En realidad su contenido es 8 dado que la unidadl del Fragment Offset es un múltiplo de 8 bytes.

El payload del datagrama original es entonces de 87 bytes.

Algo de gestión

Con el objeto de proveer algo de gestión y control de errores a un protocolo "best effort" como IP es que se definió un protocolo auxiliar. Ese protocolo es ICMP (Internet Control Message Protocol), especificado inicialmente en RFC 792 [Pos81b]. Del momento que provee gestión a IP debe entonces poder atravesar redes, por tal motivo se lo encapsula en IP con protocol type:1. Esta es su estructura:

IP Header	
Type	} 8 bits
Error Code	} 8 bits
Checksum	} 16 bits
Parámetros	} Variable
Información	} Variable

Los campos de Type y Error Code codifican el mensaje que transporta. Los mensajes reportan errores o son mensajes que pueden ser utilizados en una gestión de mayor nivel como veremos más adelante. Entre los que reportan errores tenemos por time to live excedido, destino inalcanzable, error de parámetros en el datagrama que imposibilitan su procesamiento.

El Checksum es igual al de IP. Según el mensaje tendrá campos adicionales clasificados como Parámetros. El campo final de Información estará presente cuando el mensaje requiera de datos adicionales o en el caso que sea un mensaje de error por datagrama descartado contendrá el encabezado y los primeros 64 bytes del datagrama descartado.

Cuando el Time to Live de un datagrama llega a cero se generará un mensaje de *time to live exceeded in transit* que se enviará a la IP origen del datagrama descartado.

Por razones obvias no se generan mensajes ICMP sobre datagramas que transportaban mensajes ICMP. Vale aclarar también que no se generan mensajes ICMP por descarte de un datagrama debido a error de verificación del Checksum.

Los mensajes propios de ICMP, no debido a errores se suelen emplear en el desarrollo de herramientas que facilitan la gestión de redes IP. Las más utilizadas son ping y traceroute.

Ping es un comando implementado a base de dos mensajes ICMP, echo request y echo replay. Consiste en el envío de un mensaje de echo request con datos de prueba. Cuando este mensaje llega a destino, este lo devuelve bajo el mensaje ICMP echo replay. De esta manera se puede verificar la presencia activa del destino, la ruta activa, ida y vuelta al destino y el round trip time correspondiente. Dependiendo de la implementación se puede definir la longitud de los datos a transmitir, el time to live del datagrama que lo transporta, enviar uno solo o una ráfaga, etc. Normalmente finaliza el comando con una estadística de mensajes transmitidos, recibidos, perdidos y round trip time.

Traceroute también es un comando que releva las rutas por la que pasa un datagrama para llegar a destino. Este consiste en enviar inicialmente un datagrama con datos con el TTL = 1 de modo que cuando llegue al default gateway se decremente y al ser 0 se descarta. Si ICMP está habilitado en el entonces enviará al origen un mensaje de tiempo excedido en tránsito. Cuando llega se registra la IP que lo envía y ahí se tiene el primer salto de la ruta. Luego se vuelve a enviar el datagrama con TTL = 2 para registrar el segundo salto y así hasta llegar al origen. ¿Cómo sabe que llega al origen?, por dos motivos. Uno porque en el mensaje de error estará la IP del que lo envió y también se envían datos en ese datagrama de modo que al llegar al origen el mensaje de error generado sea diferente.

Veamos la captura *n11_ping_traceroute* tomada por el Wireshark en nuestro escenario de pruebas, donde el nodo n11 ejecuta *ping* y *traceroute* al servidor web que vimos en el Capítulo Aplicaciones.

En primer lugar la Figura 5.5 nos muestra en la primera trama la ejecución del ping con el envío de un echo request al servidor.

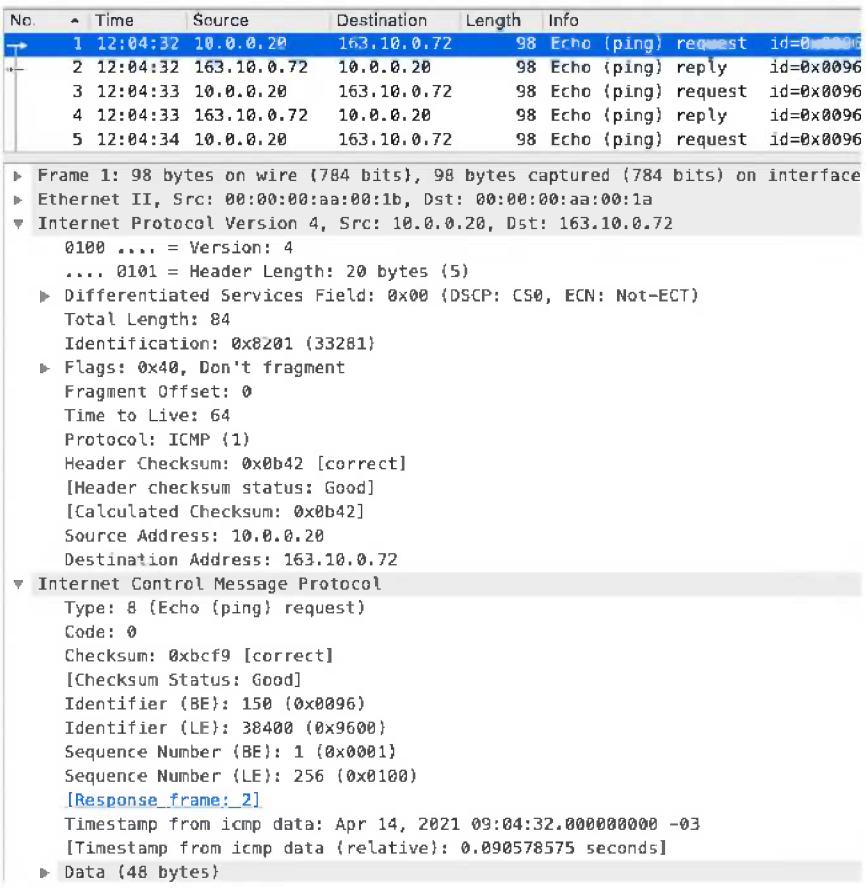


Figura 5.5: Ping-1

En la Figura 5.5 tenemos desplegada en la ventana central el datagrama IP y el mensaje ICMP de la trama 1. Podemos apreciar que el protocol type = 1 nos indica que efectivamente contiene un mensaje ICMP y yendo al mensaje vemos que type 8 y código 0 corresponden al echo request. El Wireshark decodifica el Type incluyendo ping. No es muy correcto, ping es un comando.

El Identifier y Sequence Number sirven para cotejarlos contra el echo replay recibido. El Identifier actúa como identificador de una sesión, como un port TCP o UDP. Todos los mensaje enviados desde un solo comando tendrán el mismo Identifier y lo que se incrementa es el Sequence Number. Vemos dos lecturas del Sequence Number (BE) y (LE). El contenido es el de BE, Wireshark nos está dando la posibilidad de decodificarlo como el bit más significativo a izquierda (Big Endian) o a derecha Little Endian.

Veamos en la Figura 5.6 la respuesta recibida.

Podemos apreciar el código 0-0 del echo request y la correspondencia del Identifier y Sequence Number con el datagrama anterior.

Retomando la captura *n11_ping_traceroute* a partir de la trama 15, vemos que nuestro inquieto nodo n11 envía un datagrama UDP encapsulado en un datagrama IP particular, con el TTL en 0. Se debe a que ese envío es producto de haber ejecutado el comando *traceroute* hacia el servidor Web conocido. Ese envío terminará en el default gateway que lo descartará por haber decrementado el TTL a 0 y si está habilitado ICMP enviará un mensaje a n11 de *TTL exceeded in transit*. En este caso la implementación hecha del *traceroute* hace que se repita el envío dos veces más. En la Figura 5.7 vemos el detalle de lo expuesto:

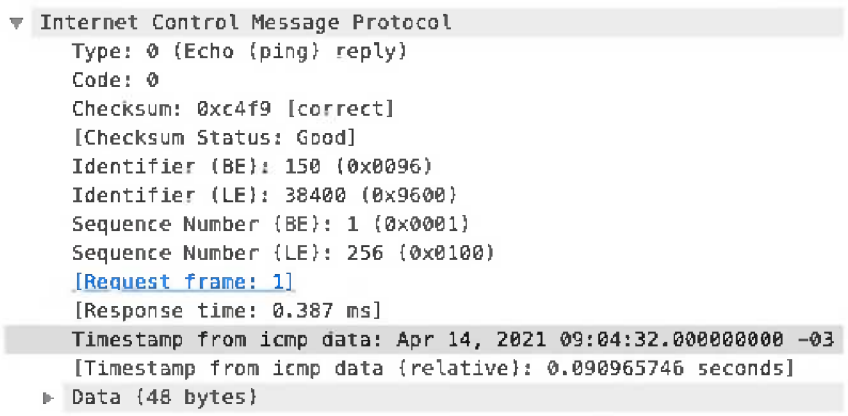


Figura 5.6: Ping-2

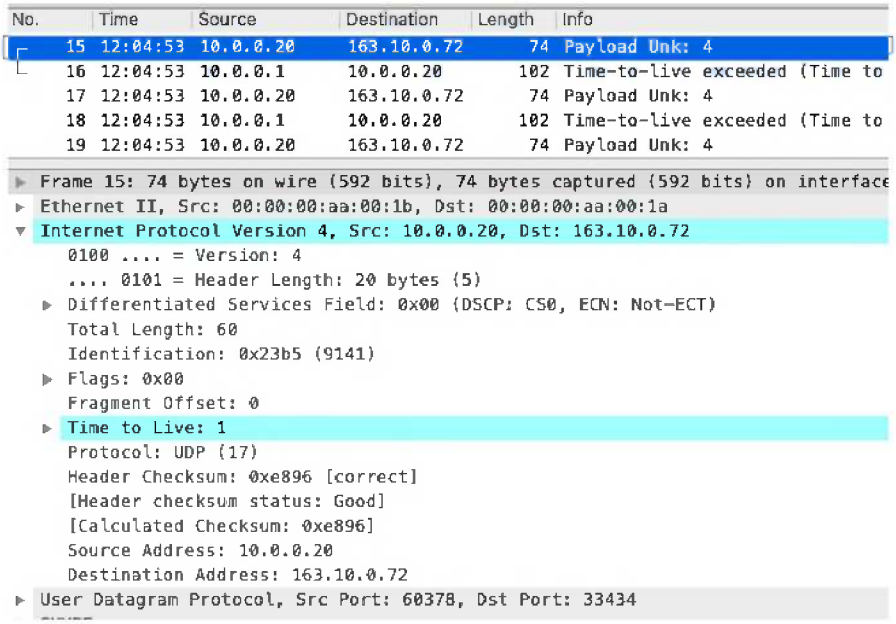


Figura 5.7: Ejecución de traceroute-1

En la trama 16 vemos la respuesta del default gateway con un mensaje ICMP que comentábamos anteriormente. Podemos verificar que en el mensaje incluye el encabezamiento del datagrama que descartó (el de la trama 15), y los primeros 64 bytes del miso. Queda detallado en la Figura 5.8

El nodo n11 continuará enviando el mismo mensaje con un incremento en el TTL y como comentamos con dos repeticiones. ¿Hasta cuando?. Volviendo a nuestra captura veremos que en la trama 51 n11 envía el mensaje con TTL = 7 y en la trama 52 recibe una respuesta diferente y por el destino buscado. De esta manera n11 se entera que el mensaje llegó a destino y que para llegar tuvo que dar 7 saltos y con cada devolución de ICMP releva la ruta para llegar a destino. En la Figura 5.9 tenemos el detalle del nuevo mensaje ICMP enviado por el destino. En el datagrama IP podemos apreciar la dirección del servidor.

Estas herramientas son muy útiles a la hora de encarar algún diagnóstico. Pero debemos tener en cuenta que la naturaleza de IP y la presencia de Calidad de Servicio en las redes involucradas pueden hacer que el camino tomado por los datagramas generados por estas herramientas sea diferente al del tráfico cuyas dificultades queremos diagnosticar. También recordemos que es necesario que ICMP esté

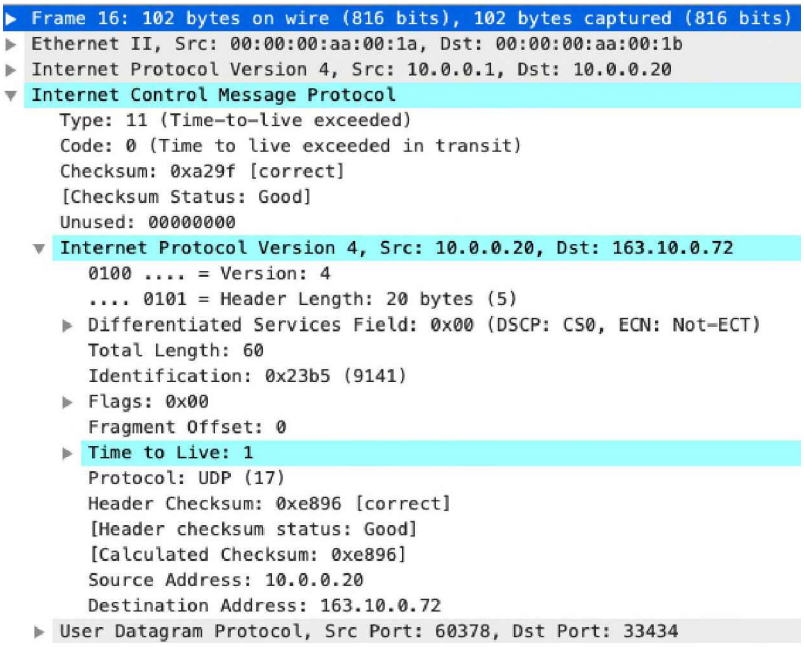


Figura 5.8: Ejecución de traceroute-2

habilitado porque por cuestiones de seguridad muchas veces se lo deshabilita.

IPv6

Protocolo que llevó bastante tiempo en comenzar a desplegarse en las redes llegando a la mayoría de edad en cuanto a su especificación en la RFC 8200 [DH17]. Por cuestiones de espacio no vamos a hacer un tratamiento como el que hicimos para IPv4, pero aquí van algunas cuestiones y comentarios acerca de IPv6.

Entre sus características principales podemos decir que fue pensado para resolver el problema del agotamiento de las direcciones, pasando de 32 bits a 128 bits de longitud.

Otra característica importante es que se buscó la reducción del overhead, aprovechando el avance tecnológico de los medios físicos. No tenemos campo de Checksum. Define un encabezado básico de longitud fija y las opciones se agregan a continuación de él quedando especificadas en el campo de Next Header, que reemplaza al de protocol de IPv4. Cada opción tendrá ahora un campo de Next Header que indicará lo que viene a continuación. Es decir en este campo aparecerán en algún caso la codificación de UDP, TCP y de todo aquello que se quiera encapsular en IPv6. Se las considera a las opciones como extensión de encabezado. Por último el campo de TTL fue reemplazado por Hop Limit, que es la cantidad de saltos que se le dan al datagrama. Semántica esta ya incorporada al TTL de IPv4.

En cuanto a características finalmente diremos que el campo de TOS de IPv4 fue reemplazado por dos campos, el de Traffic Class y Flow Label que permiten un mayor y mejor soporte de Calidad de Servicio. Presentamos la estructura del datagrama Ipv6 en la Figura 5.10

Si la comparamos contra la de iPv4 que vimos al comienzo más allá de una mayor longitud por las direcciones de 128 bits es indudable la simplicidad por tener menos campos lo que acelera su procesamiento. En cuanto a u menor tiempo de procesamiento es importante mencionar que los routers no fragmentan. Si se encuentran en la encrucijada de hacerlo por una diferencia de longitudes entonces

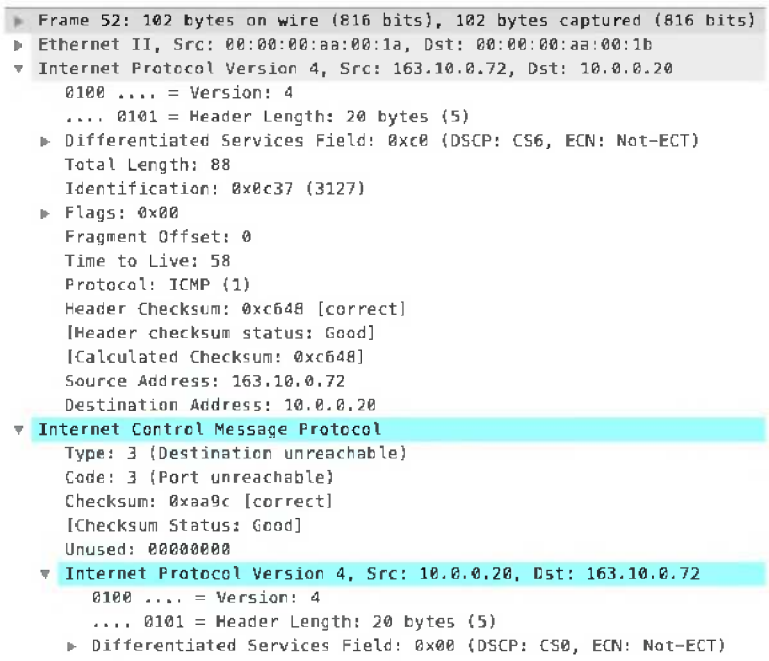


Figura 5.9: Ejecución de traceroute-3

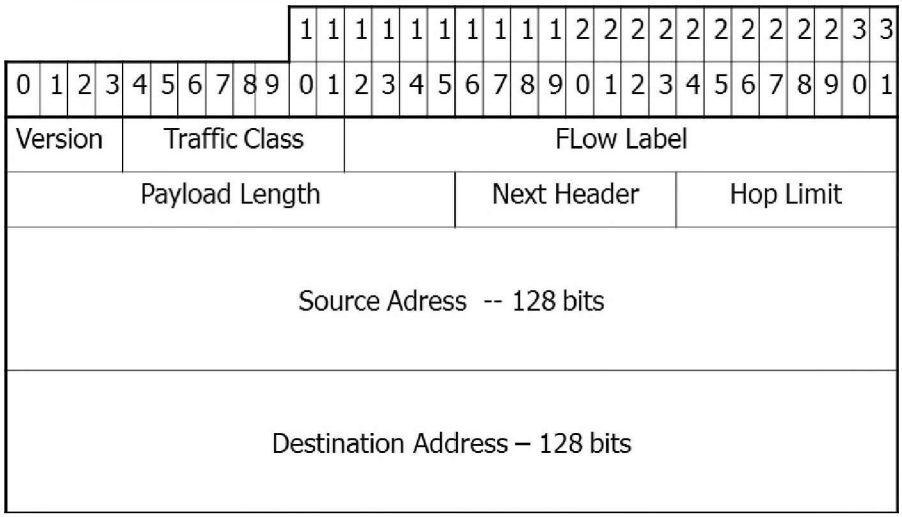


Figura 5.10: Datagrama IPv6

lo descartan. La fragmentación solo la realizan los hosts y está dada como una opción para que pueda ser reconstruido en el destino.

A continuación vemos el detalle de una captura de un datagrama IPv6 en la Figura 5.11:

Aprovechamos en primer término para comentar el formato de las direcciones. Están representadas por dígitos hexadecimales en grupos de cuatro separados por : , los :: se aplican para compactar la expresión y deben ser reemplazados por los ceros que falten para completar los 128 bits; en la dirección destino observamos un campo **ff**, los ceros a izquierda del grupo se pueden omitir.

No encontramos opciones dado que el campo de Next Header especifica el protocolo ICMPv6 [CD06].

Con la aparición del protocolo ICMPv6 podemos preguntarnos ¿por qué hay una versión 6? Eso tiene que ver con que si bien el modelo TCP/IP es un modelo de capas, hay interacción entre las capas o niveles. Retomando algunas cosas de IPv4 vimos que tiene un protocolo auxiliar al que acude en cuestiones de direccionamiento, el protocolo ARP. La estructura de ARP no soporta las direcciones de

```
Frame 3: 118 bytes on wire (944 bits), 118 bytes captured (944 bits)
Ethernet II, Src: 52:54:21:cb:97:92, Dst: 52:54:00:12:34:61
Internet Protocol Version 6, Src: fe80::5054:21ff:fecb:9792,
                             Dst: fe80::5054:ff:fe12:3461
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00
    .... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 64
    Next Header: ICMPv6 (58)
    Hop Limit: 64
    Source Address: fe80::5054:21ff:fecb:9792
    Destination Address: fe80::5054:ff:fe12:3461
Internet Control Message Protocol v6
    Type: Echo (ping) request (128)
    Code: 0
    Checksum: 0x42ff [correct]
    [Checksum Status: Good]
    Identifier: 0x2014
    Sequence: 1
    [Response In: 4]
    Data (56 bytes)
```

Figura 5.11: Captura Datagrama IPv6

IPv6 por lo que habría que definir un ARPv6. Pues bien no se ha hecho y la función equivalente de ARP para IPv6 se incorpora a ICMPv6 con los mensajes de *Neighbour Discovery* y *Neighbour Solicitation*.

En el capítulo de Aplicaciones vimos que es necesario un mapeo de nombres de recursos a direcciones IP. Para ello tenemos al servicio de DNS. Como trata de direcciones hay que modificarlo o definir un DNSv6. Por suerte por las estructuras de mensaje adoptadas no hizo falta una nueva versión, sino que se acomodó par el soporte de IPv6.

En cuanto al nivel de transporte también se vio afectado por la aparición de IPv6. Recordemos que UDP y TCP arman un pseudo encabezado para la generación y verificación del Checksum. Ese pseudo encabezado tiene en cuenta las direcciones IP, por lo que hubo que acondicionar el nivel de transporte para el soporte de IPv6.

REFERENCIAS

[BP87] T. Robert T. Braden y Jon Postel. «Requirements for internet gateways», 1987.

[CD06] A. Conta y S. Deering. «Internet control message protocol (ICMPv6) for the internet protocol version 6 (IPv6) specification», 2006.

[DH17] S. Deering y R. Hinden. «Internet protocol, version 6 (ipv6) specification», 2017.

[FL06] Vince Fuller y Tony Li. «Classless inter-domain routing (CIDR): the internet address assignment and aggregation plan», 2006.

[FMMT84] R. Finlayson, T. Mann, J. C. Mogul y M. Theimer. «A reverse address resolution protocol», 1984.

[MP85] C. Jeffrey Mogul y Jon Postel. «Internet standard subnetting procedure», 1985.

[Plu82] David C. Plummer. «Rfc 826: An ethernet address resolution protocol (arp)», 1982.

[PM95] T. Troy Pummill y Bill Manning. «Variable length subnet table for ipv4», 1995.

[Pos81a] Jon Postel. «Rfc-791: Internet protocol (ip)», 1981.

[Pos81b] Jon Postel. «Rfc-792: Internet control message protocol (icmp)», 1981.

[RMKdG96] Y. Rekhter, B. Moskowitz, D. Karrenberg y deGroot G.J. «Rfc 1918: Address allocation for private internets», 1996.