

Libros de **Cátedra**

Paradigma TCP/IP

Luis Marrone (coordinador)

FACULTAD DE
INFORMÁTICA

e
exactas


EDITORIAL DE LA UNLP



UNIVERSIDAD
NACIONAL
DE LA PLATA

Paradigma TCP/IP

Luis Marrone
(coordinador)

Facultad de Informática



ÍNDICE GENERAL

1. Escenario	3
2. La cercanía con el usuario (HTTP)	8
Paradigmas de Aplicaciones	8
Servicio WEB - HTTP	11
Acceso a una página WEB	13
3. Traduciendo nombres a direcciones (DNS)	22
Introducción	22
Historia	22
DNS en acción	23
El requerimiento de DNS	23
Espacio de nombres, FQDN	25
Jerarquía de DNS, servidores de dominios	28
La resolución de la consulta	30
El requerimiento y la respuesta completa	34
Otros Detalles	34
Consultas recursivas e iterativas	34
Respuestas autoritativas	36
Mecanismos de sincronización ente servidores	36
4. Conexión de Extremo a Extremo (Transporte)	38
Conexión de Extremo a Extremo (Transporte)	38
UDP (User Datagram Protocol)	38
TCP (Transport Control Protocol)	40
5. El protocolo de Internet (IP)	51
El protocolo de Internet (IP)	51
Protocolo IP	52
IPv4	54
Direcciones IP	55
Subredes	56
Fragmentación	59
Algo de gestión	60
IPv6	64

6. Constructores de caminos (Ruteo)	68
Introducción	68
Historia	71
Protocolos de ruteo dinámico	71
Estructura de Internet	72
Ejemplo de ruteo en la Red	78
Dentro de la red de acceso	78
Ruteo dentro de la red del ISP	78
Ruteo entre los ISP y el núcleo de Internet	79
7. Enlace: Ethernet, ARP y Switching	86
Introducción	86
Ethernet	87
ARP	90
Caso Práctico	91
Switching	95

ÍNDICE DE FIGURAS

2.1. Nivel de Aplicación en TCP/IP	10
2.2. Portal del Cern - 1989	11
2.3. Escenario	14
2.4. Captura acceso Web	15
2.5. Captura GET	16
2.6. Captura Fin del GET	17
2.7. Nuevo GET	17
2.8. Respuesta Negativa	18
2.9. GET Condicional	18
2.10. GET Condicional-Respuesta	19
2.11. Mensaje HTTP	20
2.12. HTTPS	20
3.1. Captura de auto-configuración por DHCP del cliente, asignación de servidor de DNS.	24
3.2. Captura del diálogo de DNS entre cliente y servidor local de DNS.	25
3.3. Diálogo dentro del sistema DNS entre cliente y servidor local.	26
3.4. URL y FQDN (Nombre completo de dominio).	26
3.5. Ejemplos de nombres de dominio.	28
3.6. Distribución de los ROOT Servers en 2015 (Fuente: <i>www.root-servers.org</i>).	29
3.7. Delegación de los sub-dominios en la topología de test.	31
3.8. Captura del diálogo de DNS entre local y los servidores autoritativos.	32
3.9. Diagrama de interacción entre el DNS local y los servidores autoritativos.	32
3.10. Diagrama alternativo de interacción entre el DNS local y los servidores autoritativos.	33
3.11. Captura del diálogo de DNS entre local y el autoritativo de unlp.edu.ar.	33
3.12. Diagrama de interacción entre el cliente, el DNS local y los autoritativos.	34
3.13. Flags que distinguen método recursivo de iterativo.	35
4.1. Datagrama - UDP	39
4.2. Muestra 1	40
4.3. Triple Handshake - TCP	41
4.4. Encabezamiento - TCP	41
4.5. Muestra 2	42
4.6. Muestra 3	43
4.7. Desconexión - TCP	43

4.8. Muestra 4	44
4.9. Muestra 5	45
4.10. Gráfico time sequence (Stevens) 1	46
4.11. Muestra 6	46
4.12. Gráfico variación ventana de recepción	47
4.13. Gráfico time sequence (Stevens) 2	48
4.14. Muestra 7	48
4.15. Gráfico de Round Trip Time	49
5.1. Estándar 5 - IETF - ISOC	53
5.2. Red IP	53
5.3. Requerimiento	58
5.4. Datagrama Fragmentado	60
5.5. Ping-1	62
5.6. Ping-2	63
5.7. Ejecución de traceroute-1	63
5.8. Ejecución de traceroute-2	64
5.9. Ejecución de traceroute-3	65
5.10. Datagrama IPv6	65
5.11. Captura Datagrama IPv6	66
6.1. Tareas realizadas por los routers en sus respectivos planos.	69
6.2. Ruteo IGP vs EGP.	73
6.3. Ejemplo de AS de tránsito, stub o multihome.	75
6.4. Diagrama de la relación entre los sistemas autónomos.	77
6.5. Captura de mensaje de OSPFv2 entre routers del mismo AS.	80
6.6. Diagrama de las distintas redes y los sistemas autónomos (AS).	80
6.7. Captura de mensaje de BGPv4 entre routers de diferentes AS, tier 1.	81
6.8. Camino tomado por el datagrama IP hasta llegar a destino.	84
7.1. Dominios de Colisión y de Broadcast	87
7.2. Trama Ethernet II/IEEE 802.3	87
7.3. Configuración interface eth0 de <i>n11client</i>	92
7.4. Tabla ruteo del host <i>n11client</i>	92
7.5. Configuración interface eth1 del router <i>n10</i>	92
7.6. ARP Request	93
7.7. ARP Reply	93
7.8. Mensaje DNS Query	94
7.9. ARP Request - Host origen y destino en la misma LAN <i>n11client</i>	95

PREFACIO

Muchos se preguntarán: “¿Hacía falta un libro sobre IP cuando es el protocolo base de una red con casi 50 años de vida y con una bibliografía extensísima tanto en inglés como en castellano? ”

Nuestra vasta experiencia en la enseñanza de estos temas y el escenario particular desde marzo del 2020 que nos obligó por la actividad no presencial a un mayor seguimiento de los temas y a una mayor carga de actividades nos dijeron que sí. Justamente esos dos factores afianzaron la idea que hacía falta desentrañar tabúes que se presentan en estos temas y que no logran desentrañarse luego de un recorrido por la bibliografía usual que se establece para estos cursos.

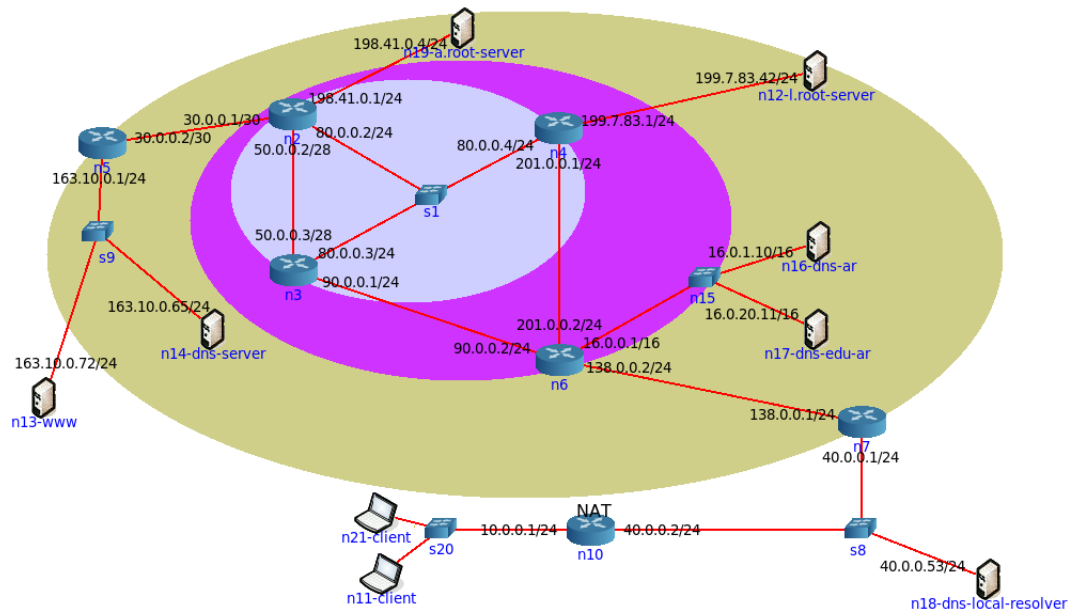
Es así que planteamos un recorrido por los temas que integran el modelo TCP/IP y que aparecen en forma explícita o implícita cuando un usuario quiere acceder a una página Web. Por supuesto no podía ser otra que la de la UNLP.

Comenzamos planteando un escenario tipo a modo de contextualizar la actividad que vamos a desplegar y comenzamos el recorrido desde el usuario accediendo a su navegador, por lo que según el paradigma TCP/IP arrancamos con el nivel de aplicaciones o servicios, tomando como ejemplo el servicio de acceso WEB. Continuamos con el auxilio dado por DNS para seguir luego con el nivel de transporte con sus dos protocolos emblemáticos, UDP y TCP.

Pasamos luego al Protocolo de Internet (IP) y sus auxiliares, dedicando el siguiente capítulo al gran auxilio dado por los protocolos de ruteo. Culminamos el viaje con los elementos más cercanos al hardware y el nivel de enlace en general.

El lector encontrará entonces todos los participantes del recorrido capítulo a capítulo con una explicación somera de conceptos teóricos e inmediatamente se encontrará con capturas de tráfico analizadas a modo de ejemplo. Dichas capturas se muestran en figuras con un análisis completo y que para un mayor estudio están disponibles en el repositorio oficial de la UNLP (<https://sedici.unlp.edu.ar/>).

Las mismas se obtuvieron por medio de la herramienta Wireshark en un escenario virtualizado por medio de CORE (Common Open Research Emulator), <https://github.com/coreemu/core>, que aquí presentamos:



Queremos aprovechar este espacio para agradecer a nuestros alumnos, depositarios indiscutibles de este trabajo, cuyas dudas y angustias nos alentaron a llegar a este punto. Vaya también nuestro agradecimiento a la UNLP por darnos esta oportunidad y por la confianza depositada en nosotros.

Los autores

CAPÍTULO 1

ESCENARIO

LUIS MARRONE

Como anticipamos en el prefacio les proponemos como escenario para derribar esos mitos y tabúes de TCP/IP un simple acceso a `www.unlp.edu.ar`. Suponemos que la conocen, ¿no?

Para llevar adelante ese cometido tenemos que disponer de un navegador web como comúnmente se le llama que para ser precisos (vayámonos acostumbrando a esto), no es otra cosa que un programa cliente. Cliente porque su desarrollo está basado en el paradigma cliente-servidor. Más allá de sus diferencias en cuanto a velocidad, funcionalidad, servicios, seguridad, etc., todos ellos, llámense Firefox, Chrome, Internet Explorer, Safari, Opera, etc., se basan en este paradigma.

Este paradigma es con el que se desarrollaron las primeras aplicaciones o servicios a través de Internet. Luego aparecieron otros como “peer-to-peer”, que en la actualidad compite en intensidad de tráfico con cliente-servidor.

Vayamos elaborando el reparto de todos los actores necesarios para completar nuestro sencillo acceso a la página web de la Universidad. Hasta ahora estamos:

Actores

- 1 Programa-Servicio-Aplicación
-

El texto de este actor será `www.unlp.edu.ar` que incluye en el nombre de la página que queremos acceder la ubicación del servidor, el otro integrante del paradigma de programación de la aplicación. En el mundo de TCP/IP la ubicación de este actor está dada por lo que llamamos la dirección IP. Puede que el navegador no la conozca, es decir, no la tenga disponible en su memoria. En ese caso, llamará a un actor de reparto que le va a conseguir la dirección IP que no tiene. Este actor de reparto es la aplicación/servicio **DNS** [Moc87], [Moc89] y [THKS03]. Lo llamamos actor de reparto por cuanto es un servicio auxiliar. Nosotros, usuarios, difícilmente querremos acceder directamente a este servicio. Se incrementa el reparto:

Actores

- 1 Programa-Servicio-Aplicación
 - 2 DNS (Domain Name Service)
-

Ya sabemos la página que queremos acceder y su dirección IP. Tendremos ahora que enviarle un mensaje al servidor para pedirle el acceso y que nos envíe esa página. Por lo pronto construiremos un mensaje con ese pedido tal que lo comprenda el servidor. Para esto, ese mensaje tendrá un formato definido por un protocolo en particular. No existe un único protocolo. En nuestro ejemplo tomaremos **HTTP**, HyperText Transfer Protocol. [BLFF96]

Tenemos que conseguir que ese pedido llegue al servidor. El programa servidor reside en un host o dispositivo de hardware generalmente destinado a ejecutar ese programa y que estará recibiendo múltiples pedidos como el nuestro. No está de más invitarlo a que nos atienda. Vamos a avisarle que queremos solicitarle información que tiene disponible y que es de nuestro interés. Esperamos que acepte. Para poder cumplir con todo eso vamos a recurrir al envío de un mensaje definido por un protocolo **TCP**, (Transmission Control Protocol) [Pos81c], que contendrá al mensaje anterior de **HTTP**. Este protocolo llevará un control estricto del envío, detectando errores y generando retransmisiones en esos casos. En otras palabras, nos garantiza la recepción de lo que envía.

Ya que mencionamos varias veces a un protocolo es hora de definirlo, por lo menos protocolos que son de nuestro interés:

protocolo: Conjunto de reglas y tipos de mensajes que permiten el intercambio de información entre dispositivos lejanos y con un nivel de confiabilidad y seguridad definido ad-hoc.

Nuestro reparto sigue creciendo:

Actores

- 1 Programa-Servicio-Aplicación
- 2 DNS (Domain Name Service)
- 3 HTTP
- 4 TCP

En otras ocasiones, el mensaje que necesitamos transferir es el que corresponde a tráfico de voz o de video que pertenecen a una clase particular de tráfico que es el de tiempo real. Ese tráfico requiere un envío inmediato y no necesita de un control tal que frente a un pérdida lo retransmitamos. Una retransmisión, por ejemplo, generaría un ruido en nuestros oídos. Es más, por su naturaleza enviamos el mensaje directamente, no dialogamos con el servidor previamente como en los otros casos. Por eso, en ese tipo de tráfico de tiempo real, necesitamos acudir a otro protocolo con una funcionalidad que se ajusta a esas características. El protocolo es UDP (User Datagram Protocol), [Pos80]. Estas características hacen que sea utilizado, también, por el DNS, por lo que lo hace un actor importante para nuestro análisis.

Actores

- 1 Programa-Servicio-Aplicación
- 2 DNS (Domain Name Service)
- 3 HTTP
- 4 TCP-UDP

Así como queremos conectarnos con el servidor de la Universidad, también, pretendemos tener una amplia versatilidad en las conexiones. Poder conectarnos con servidores tanto próximos como lejanos. En otras palabras, independizarnos de la distancia a la que nos encontremos del servidor. En ese caso vamos a recurrir a un servicio dado por una red como Internet y, para ello, a nuestro mensaje TCP habremos de encapsularlo en otro mensaje que nos permita disponer de esos servicios de red. Lo encapsularemos en el protocolo **IP** [Pos81a] y [DH17], que tiene dos versiones vigentes, IPv4 e IPv6, presentes en la red de redes como Internet. Los actores siguen creciendo:

Actores	
1	Programa-Servicio-Aplicación
2	DNS (Domain Name Service)
3	HTTP
4	TCP-UDP
5	IP

Podemos resumir en una palabra los servicios de red que mencionamos: “*conectividad IP*”.

conectividad IP: Propiedad del protocolo IP que permite el intercambio de datagramas entre usuarios de redes IP interconectadas. Para lograrlo el protocolo IP necesita solo conocer las direcciones IP origen y destino de los usuarios

Esas direcciones IP que identifican a los usuarios de una red IP y que como recordarán, con el mismo formato identificamos a una red IP, son parámetros de configuración entre otros. Configuración que la hacemos manualmente o estáticamente como suele decirse en la jerga local de IPv4 o dinámicamente como en IPv6.

También podemos recurrir a un servicio como **DHCP** [Dro97] y [HFZT10] que nos entrega la dirección IP y toda la información que necesitamos para lograr esa conectividad.

Para finalizar con este breve pantallazo de IP vale la pena recordar que es un protocolo sin conexión, los datagramas, así se los llama a los mensajes de IP, son independientes, es como un servicio de correo postal simple. Para que el mensaje llegue lo incluyo en un sobre con la dirección destino e introduzco el sobre en el buzón más cercano. No sé si el mensaje llegó a destino. Con IP ocurre lo mismo. Por eso aquello de *best effort* que lo caracteriza.

Pero contamos con un protocolo auxiliar como **ICMP** [Pos81b] y [CD06] que como vamos a ver brinda algo de gestión a las redes IP.

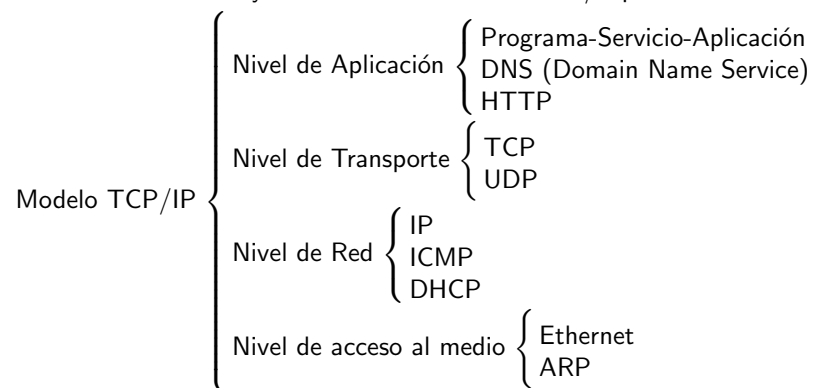
¿Cómo accedemos a Internet? Tenemos que llegar de nuestro equipo de cómputo a “La Red”. Independientemente de la ubicación del equipo, este formará parte de una red LAN que lo conectará al router de salida/ingreso a Internet de nuestro lugar de trabajo o a un port del modem-switch-router que nos instaló nuestro proveedor de Internet, ISP, en nuestro hogar. La LAN logra que el mensaje IP llegue al dispositivo de salida/acceso a Internet. El mensaje IP (datagrama) tendrá que encapsularse en un mensaje del protocolo de LAN, típicamente Ethernet [Bog76]. Hará falta conocer la dirección Ethernet del dispositivo de salida, que si no la conoce nuestro equipo acudirá a protocolos auxiliares como **ARP** [Plu82] para que obtenga esa dirección Ethernet. Completamos así el reparto:

Actores	
1	Programa-Servicio-Aplicación
2	DNS (Domain Name Service)
3	HTTP
4	TCP-UDP
5	IP-ICMP-DHCP
6	Ethernet-ARP

Este es el escenario propuesto con sus actores principales. Nuestras actividades nos llevan en muchas ocasiones a analizar el comportamiento/performance de cada uno de ellos así como su conjunto y, también, producto de ese análisis puede que necesitemos modificar su diseño para optimizar la performance.

Actividades estas que suelen llevarse a cabo de una mejor manera si contamos con un modelo de ese escenario.

En nuestro caso, como sabemos, el modelo más comúnmente empleado es el modelo TCP/IP. Recordamos también como se distribuyen esos actores en los niveles/capas del modelo:



Aprovechamos para recordar que el nombre proviene del protocolo del nivel de red como IP y de uno de los protocolos del nivel de transporte como TCP. **TCP/IP no es un protocolo.** Para llevar a cabo un correcto análisis y no generar confusión alguna es importante que seamos precisos. Por ello esa aclaración.

REFERENCIAS

- [BLFF96] T. Berners-Lee, R. Fielding y H. Frystyk. «Hypertext transfer protocol – HTTP/1.0», 1996.
- [Bog76] Robert Metcalfe; David Boggs. «Ethernet: Distributed packet switching for local computer networks». *Communications of the ACM*, vol. 19 nº 7, páginas 395–405, 1976.
- [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.
- [Dro97] R. Droms. «Rfc 2131: Dynamic host configuration protocol», 1997.
- [HFZT10] T. Huth, J. Freimann, V. Zimmer y D. Thaler. «Rfc-5970: Dhcpv6 options for network boot», 2010.
- [Moc87] P. V. Mockapetris. «Rfc 1035: Domain names: Implementation specification», 1987.
- [Moc89] P. V. Mockapetris. «Rfc-1101: Dns encoding of network names and other types», 1989.
- [Plu82] David C. Plummer. «Rfc 826: An ethernet address resolution protocol (arp)», 1982.
- [Pos80] Jon Postel. «Rfc-768: User datagram protocol (udp)», 1980.
- [Pos81a] Jon Postel. «Rfc-791: Internet protocol (ip)», 1981.
- [Pos81b] Jon Postel. «Rfc-792: Internet control message protocol (icmp)», 1981.
- [Pos81c] Jon Postel. «Rfc-793: Transmission control protocol (tcp)», 1981.
- [THKS03] S. Thomson, C. Huitema, V. Ksinant y M. Souissi. «Rfc 3596: Dns extensions to support ip version 6», 2003.

CAPÍTULO 2

LA CERCANÍA CON EL USUARIO (HTTP)

LUIS MARRONE

Paradigmas de Aplicaciones

Las aplicaciones o servicios en el mundo de TCP/IP no son otra cosa que programas que satisfacen necesidades de los usuarios como:

- Acceder a un equipo de cómputo remoto
- Transferir un archivo
- Acceder a una página Web
 - Hacer una transacción bancaria
 - Hacer compras
 - Inscribirse en un curso
 - Sustener una reunión de trabajo o una clase
 - Escuchar música
 -
- Enviar/recibir un correo electrónico
- Mantener una conversación telefónica

Terminamos la lista aquí para darle un fin. No es exhaustiva ni mucho menos.

Habíamos mencionado en el capítulo anterior que a la hora de programar estas aplicaciones el desarrollador normalmente se va a encontrar con dos paradigmas disponibles, el de *cliente-servidor* y el de *peer-to-peer*.

Paradigma Cliente-Servidor

- Es un caso más de comunicación entre procesos
- Los servidores se implementan como programas de aplicación
- Resultan así transportables a todo sistema que soporte comunicaciones TCP/IP
- Generalmente tienen un hardware (computador) dedicado a ellos. Así es que se hace referencia a la máquina como servidor

Aprovechamos para tener presentes características del cliente y del servidor.

Servidor

- Invocado automáticamente en el arranque de la máquina
- Espera pasivamente la llegada de peticiones de clientes
- Puede gestionar peticiones simultáneas de varios clientes
- En la misma máquina pueden estar funcionando varios servidores de diferentes Servicios
- Se suele llamar también "servidor" a la máquina donde se ejecuta el programa servidor. Inclusive se hace un uso extensivo del término para identificar equipos de altas prestaciones y que no necesariamente obedecen a este paradigma.
- Dispone de un port bien conocido y/o reservado por el IANA¹

Cliente

- Invocado por el usuario
- Inicia el contacto con el servidor
- Puede comunicarse con:
 - varios servidores alternativamente
 - varios servidores simultáneamente
 - el mismo servidor concurrentemente
- Dispone de un port efímero dado por el sistema operativo

Independientemente de cual se adopte una vez más tendremos un intercambio de mensajes acorde con la estructura adoptada por la aplicación y también ese mensaje será encapsulado en el Nivel de Transporte según nuestro modelo planteado de TCP/IP. Recordemos que el nivel de transporte adoptado (protocolos TCP o UDP) dependerá de la naturaleza del tráfico propio de la aplicación y/o de los requerimientos de la misma.

Sucintamente podemos decir que si la aplicación requiere confiabilidad y no un tiempo de respuesta comprometido normalmente se lo encapsulará en TCP. Si, por el contrario, el tráfico que genera es de

¹Organismo que asigna número de ports a las aplicaciones

tiempo real y/o no requiere un alto grado de confiabilidad, entonces lo encapsulará en UDP. Veremos más detalles en el capítulo dedicado al Nivel de Transporte.

Para concluir veamos una figura en la que representamos algunas aplicaciones bajo este paradigma con los port bien conocidos asociados y su ubicación en el modelo de TCP/IP

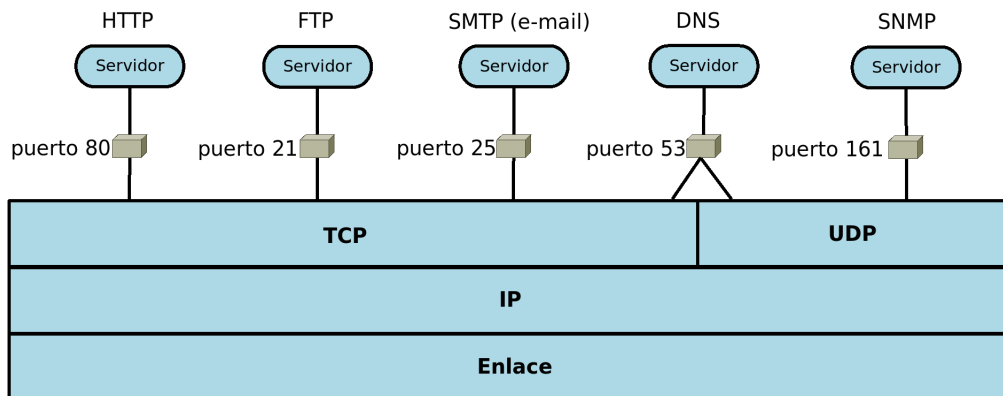


Figura 2.1: Nivel de Aplicación en TCP/IP

El otro modelo, *peer-to-peer*, presenta estas características:

Paradigma Peer-to-Peer

- Compartir recursos.
- Procesamiento distribuido.
- Procesos colaborativos.
- Es un caso más de comunicación entre procesos.
- Red de usuarios.
- Dualidad cliente-servidor en cada nodo.

Con este paradigma se han desarrollado numerosas aplicaciones que contribuyen a un porcentaje importante en el tráfico presente en Internet. Presentan algunos problemas de seguridad y de viabilidad en presencia de redes privadas virtuales y NAT que de alguna manera se verán disminuidos con la penetración de IPv6 en una red como Internet. Nosotros nos vamos a ocupar del análisis de aplicaciones basadas en el paradigma Cliente-Servidor, como es el caso de las que utilizan a *http* como protocolo de aplicación.

Antes de incursionar en *http* en detalle tengamos presente características comunes de las aplicaciones desarrolladas bajo este paradigma.

Ante todo tengamos en cuenta que será un software residente en el equipo del cliente/servidor y, como tal, tendrá que interactuar con el sistema operativo. Además, se requiere que la aplicación pueda dar su servicio entre cliente y servidor con diferentes sistemas operativos en cada uno de ellos.

La solución llegó con la implementación de una terminal virtual que brinda una interfaz entre el SO y la aplicación propiamente dicha y logra la independencia del SO en cuanto a su funcionalidad. Esa terminal virtual fue implementada en la que podemos decir fue la primera aplicación basada en ese paradigma cual es el caso de Telnet. Terminal que en mayor o menor grado la tendremos presente en todas las aplicaciones del tipo mencionado anteriormente.

Dado el alcance y extensión del presente libro es que decidimos ver en detalle uno de los servicios más populares como es el caso del servicio WEB mediante el protocolo HTTP [BLFF96]. Mencionamos, siempre dentro del paradigma cliente-servidor, otros servicios como:

Telnet: acceso remoto a otro dispositivo y que de alguna manera dio pie al desarrollo del resto de las aplicaciones.

FTP: transferencia de archivos entre dispositivos con diferentes plataformas.

SMTP: servicio de correo electrónico.

SNMP: gestión de la red.

⋮

Siempre dentro del mismo paradigma pero con carácter de auxiliares por cuanto mejoran la performance y funcionalidad de otros servicios:

DHCP: completa la configuración de un nodo IP automáticamente.

DNS: resuelve nombres de recursos a direcciones IP y viceversa.

Como adelantamos en varias ocasiones nos ocuparemos luego en detalle de DNS.

Servicio WEB - HTTP

Para comenzar no podemos dejar de incluir la Figura 2.2 que representa la primer página Web como la conocemos actualmente provista por un servidor del CERN², lugar donde nació este servicio. Esta

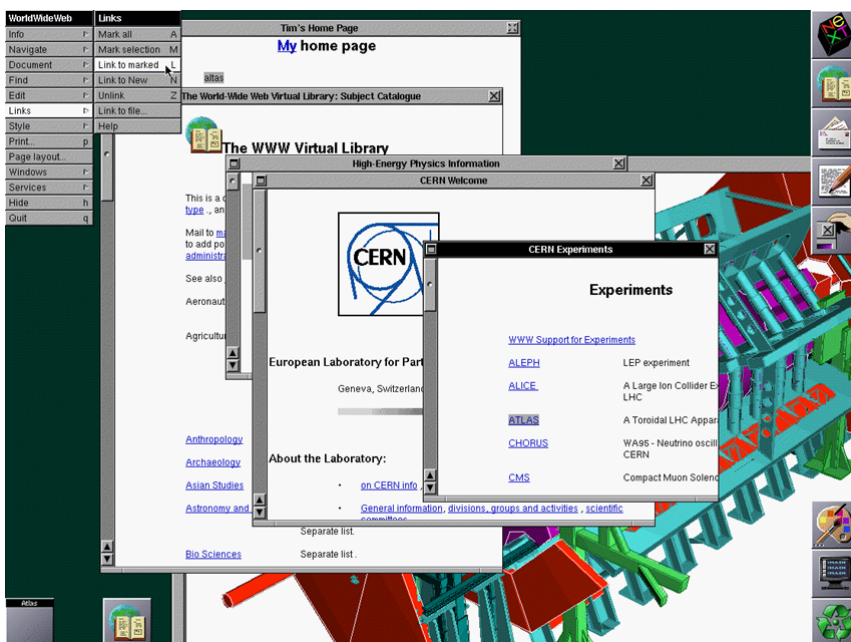


Figura 2.2: Portal del Cern - 1989

aplicación entrega a pedido del cliente información disponible en el servidor bajo un formato particular

²Conseil Européen pour la Recherche Nucléaire-Consejo Europeo para la Investigación Nuclear

y que se presenta en el cliente como la vemos en la Figura 2.2. El port bien conocido reservado para el servidor es el 80. El pedido será entregado bajo un mensaje con una estructura especificada por el protocolo *http*.

El usuario que pone en acción al lado cliente del servicio para acceder a la página/información del servidor va a llamar a un programa conocido como Web browser o navegador.

La página web que quiere acceder es en realidad un archivo de formato HTML³ que a su vez normalmente consta de objetos que pueden ser:

- archivo HTML
- imagen JPEG
- applet Java applet
- archivo de audio
- ...

Referenciados acorde con las reglas del lenguaje HTML. El formato de referencia básico está dado por una URL(Universal Resource Locator):

$$\underbrace{http}_{\text{protocolo}} : \underbrace{//www.linti.unlp.edu.ar}_{\text{host_name}} / \underbrace{archivos/banner_linti.jpg}_{\text{path_name}}$$

Los objetos HTML podrán estar disponibles desde el propio navegador y para acceder a otro tipo de información el navegador acude a visores externos adaptados a cada formato. No son otra cosa que los conectores o *plug-ins*. Pero no solo con este servicio accedemos a información. Actualmente es un servicio interactivo y bidireccional a través del cual desarrollamos la mayoría de las actividades por Internet (sobre todo en estos años tan particulares bajo pandemia, 2020-2021):

- Transacciones bancarias
- Compras de todo tipo
- Correo Electrónico
- Trámites
- Reservas
- ...

Veamos en primer término lo que ocurre al acceder a la página de la UNLP.

1. Tenemos que conocer su nombre: *www.unlp.edu.ar*
2. En la barra de direcciones de nuestro navegador introducimos el url anterior.
3. El navegador podrá acordarse de accesos anteriores y nos traerá lo que tiene almacenado en nuestra computadora. Si no va a acceder a esa página.

³Hyper Text Markup Language

4. Como el mensaje de acceso del protocolo *http* se encapsula en TCP habrá que establecer una sesión TCP con el servidor ubicado en *www.unlp.edu.ar*.
5. como todo segmento TCP se encapsulará en IP y por lo tanto necesitamos conocer la dirección IP de nuestro destino.
6. Aquí se detiene el protocolo *http* dado que comúnmente desconocemos las direcciones IP de los servidores. Se llama a un “resolver” que es un procedimiento que llama a otro servicio dentro de este paradigma TCP/IP llamado DNS. Dicho servicio actúa como auxiliar de *http* dado que realiza un mapeo de nombres bajo el formato *url* a direcciones IP. Es algo parecido en ARP entre las direcciones IP y las direcciones MAC. Sobre el particular servicio nos extenderemos más adelante.
7. De una manera u otra conseguida la dirección IP del servidor se podrá establecer la sesión y enviando el mensaje *http* correspondiente nuestro navegador nos mostrará el contenido de la página de *www.unlp.edu.ar*

Como comentamos anteriormente la página tendrá un contenido acorde con las estructuras del lenguaje *HTML*. Estructuras que permiten un contenido diverso tanto como archivos, imágenes y locaciones de otras páginas web o locaciones dentro del mismo servidor.

En este caso se deberá establecer una sesión *http* por cada acceso. Teniendo en cuenta la nueva versión del protocolo, la *http1.1*, no hará falta si es la misma locación. También se han mejorado los mecanismos de acceso en pos de una mejor performance, pero no nos ocuparemos de ellos en esta instancia.

Como mecanismos auxiliares no podemos dejar de mencionar las “cookies” que dotan de algo de memoria al acceso a páginas frecuentes y nos permiten el envío de información a los servidores dándole una característica de servicio interactivo no contemplado en sus inicios y que, por otra parte, generó un vuelco importante en cuanto a frecuencia de uso de este servicio coincidente con el incremento de tráfico en Internet debido al mismo.

Acceso a una página WEB

Vamos a ver en detalle particularidades del protocolo *http* a través de capturas realizadas por el “sniffer” Wireshark al acceder a la página *www.unlp.edu.ar*

El escenario planteado es el de la Figura 2.3

En este escenario el nodo *n10* accede al nodo *n13-www* que tiene asignado para su servicio web el nombre o *URI*: *www.unlp.edu.ar*. Analizamos el tráfico que generó este pedido de acceso a través de la captura realizada por el *Wireshark*, *n10-nat-http-firefox.pcapng*. Como los accesos son frecuentes a esta página el navegador *Firefox* conoce la dirección IP del servidor, 163.10.0.72. Puede que el navegador no conociera la dirección IP, entonces acudiría al auxilio del servicio *DNS* para obtenerlo. El análisis de dicho servicio lo veremos ampliamente en otras secciones. En una primera visión de la captura observamos, Figura 2.4.

Queremos aprovechar para un pequeño comentario respecto de las capturas de Wireshark. Van a ver que algunas líneas o párrafos aparecen entre corchetes “[,]”. Tengan en cuenta que no forman parte

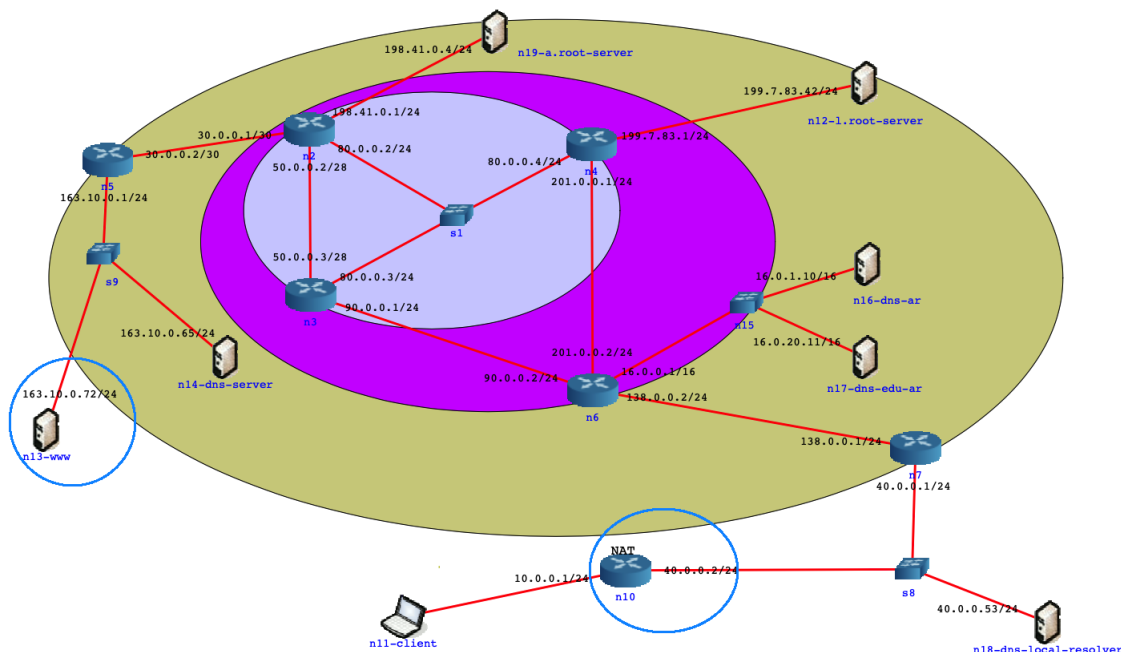


Figura 2.3: Escenario

de la trama capturada, es una interpretación/aclaración que corre por cuenta de los desarrolladores de la herramienta.

Si analizamos en detalle la trama sombreada en azul vemos que nos indica que es un mensaje del protocolo HTTP encapsulado en un segmento TCP. Comprobamos también que el servidor tiene asignado el port bien conocido 80 y, en este caso, el cliente, nodo 10, obtuvo del sistema operativo el port efímero 39796. El protocolo HTTP en su primera versión 1.0 está especificado en [BLFF96]. En este protocolo una vez que se obtenía la página se cerraba la sesión TCP.

Por lo tanto, para extraer objetos dentro de la página era necesario establecer sendas sesiones TCP. Esto generaba una no buena performance por el overhead resultante. Al poco tiempo aparece una nueva versión, la 1.1, que está especificada en [FGM⁺99]. La misma fue superada por:

- RFC 7230, RFC 7231, RFC 7232, RFC 7233, RFC 7234, RFC 7235.

y actualizada por:

- RFC 2817, RFC 5785, RFC 6266, RFC 6585.

Analicemos en detalle el contenido de este mensaje tomando un sector de la captura como vemos en la Figura 2.5

Este mensaje está solicitando el objeto *index2.html* a través de lo que llamamos uno de los métodos de HTTP, GET.

La primer línea de este método contiene su nombre: Request Method; Request URI, el objeto deseado, y Request Version, la versión de HTTP, en este caso 1.1. La presentación de Wireshark no muestra el fin de la línea, pero si analizamos a nivel de byte, en la última ventana veremos un $(0d0a)_{16}$, (representamos la codificación de $\backslash r$ y $\backslash n$ tal como lo hace Wireshark; siendo dígitos hexadecimales deberían estar en mayúscula), o sea retorno de carro y line feed. Herencia de la terminal virtual de Telnet. La segunda línea:

The image shows a Wireshark network traffic capture. The top toolbar includes icons for file operations, search, and display filters. The main pane is divided into three sections: a packet list, packet bytes, and packet details.

Packet List:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000	10.0.0.50	host163-10-0-72.cespi.unlp.e...	TCP	74	39796 → http(80) [SYN] Seq=0 Win=64240 [TCP CHECKSUM INCORRECT] Len=0 MSS=1460 SACK_PERM=1 Tsval=578256730 TSecr=0 WS=128
2	0.0000707	host163-10-0-72.ces...	10.0.0.50	TCP	74	http(80) → 39796 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 Tsval=3381107168 TSecr=578256730 WS=128
3	0.0000949	10.0.0.50	host163-10-0-72.cespi.unlp.e...	TCP	66	39796 → http(80) [ACK] Seq=1 Ack=1 Win=64256 [TCP CHECKSUM INCORRECT] Len=0 Tsval=578256730 TSecr=3381107168
4	0.2438880	10.0.0.50	host163-10-0-72.cespi.unlp.e...	HTTP	432	GET /index2.html HTTP/1.1
5	0.2438896	host163-10-0-72.ces...	10.0.0.50	TCP	66	http(80) → 39796 [ACK] Seq=1 Ack=367 Win=64896 Len=0 Tsval=3381107412 TSecr=578256974
6	0.2440253	host163-10-0-72.ces...	10.0.0.50	HTTP/XML	509	HTTP/1.1 200 OK
7	0.2440752	10.0.0.50	host163-10-0-72.cespi.unlp.e...	TCP	66	39796 → http(80) [ACK] Seq=367 Ack=444 Win=63872 [TCP CHECKSUM INCORRECT] Len=0 Tsval=578256974 TSecr=3381107412

Packet Details:

- Frame 4: 432 bytes on wire (3456 bits), 432 bytes captured (3456 bits) on interface vetha.1.bc, id 0
- Ethernet II, Src: 36:f0:9f:a9:c7:a2 (36:f0:9f:a9:c7:a2), Dst: 00:00:00:aa:00:1a (00:00:00:aa:00:1a)
- Internet Protocol Version 4, Src: 10.0.0.50 (10.0.0.50), Dst: host163-10-0-72.cespi.unlp.edu.ar (163.10.0.72)
- Transmission Control Protocol, Src Port: 39796 (39796), Dst Port: http (80), Seq: 1, Ack: 1, Len: 366
- Hypertext Transfer Protocol
 - GET /index2.html HTTP/1.1
 - [Expert Info (Chat/Sequence): GET /index2.html HTTP/1.1]
 - Request Method: GET
 - Request URI: /index2.html
 - Request Version: HTTP/1.1
 - Host: www.unlp.edu.ar
 - User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:73.0) Gecko/20100101 Firefox/73.0
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
 - Accept-Language: es-AR,es;q=0.8,en-US;q=0.5,en;q=0.3
 - Accept-Encoding: gzip, deflate
 - Connection: keep-alive
 - Upgrade-Insecure-Requests: 1

Figura 2.4: Captura acceso Web

```

▶ Frame 4: 432 bytes on wire (3456 bits), 432 bytes captured (3456 bits) on interface vetha.1.bc,
▶ Ethernet II, Src: 36:f0:9f:a9:c7:a2 (36:f0:9f:a9:c7:a2), Dst: 00:00:00_aa:00:1a (00:00:00:aa:00:
▶ Internet Protocol Version 4, Src: 10.0.0.50 (10.0.0.50), Dst: host163-10-0-72.cespi.unlp.edu.ar
▶ Transmission Control Protocol, Src Port: 39796 (39796), Dst Port: http (80), Seq: 1, Ack: 1, Len
▼ Hypertext Transfer Protocol
  ▼ GET /index2.html HTTP/1.1\r\n
    ▶ [Expert Info (Chat/Sequence): GET /index2.html HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /index2.html
      Request Version: HTTP/1.1
      Host: www.unlp.edu.ar\r\n
      User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:73.0) Gecko/20100101 Firefox/73.0\r\n
      Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
      Accept-Language: es-AR,es;q=0.8,en-US;q=0.5,en;q=0.3\r\n
      Accept-Encoding: gzip, deflate\r\n
      Connection: keep-alive\r\n
      Upgrade-Insecure-Requests: 1\r\n
      \r\n

```

Figura 2.5: Captura GET

Host:www.unlp.edu.ar\r\n, el host del recurso solicitado. Aquí si Wireshark nos muestra el fin de línea.

La tercera:

```
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:73.0)
```

Gecko/20100101 Firefox/73.0\r\n, es el agente/cliente que originó el request.

Seguimos:

```
Accept: text/html,application/xhtml+xml,+
```

application/xml;q=0.9,image/webp,*/*;q=0.8\r\n, tipos de media que se aceptan en la respuesta.

Accept-Language: es-AR,es;q=0.8,en-US;q=0.5,en;q=0.3\r\n, es obvio, no?. Aclaremos, también que para la línea anterior el parámetro q es una suerte de factor de preferencia que se le da al medio o lenguaje indicado.

Accept-Encoding: gzip, deflate\r\n, a modo de mejorar la performance acepta que el recurso venga en cualquiera de los dos esquemas de compresión, gzip o deflate.

Connection: keep-alive\r\n, es un tanto obvio caracterizar Connection con “keep-alive” porque estamos en HTTP1.1 con característica persistente. Se definió para poder cerrar la conexión pese a que se estaba indicando 1.1, caracterizando a Connection con “close”.

Upgrade-Insecure-Requests: 1\r\n, permite al agente de usuario actualizar las solicitudes a priori, inseguras por seguras antes de recuperarlas.

\r\n, así finaliza el mensaje. Otra herencia más de la terminal virtual de Telnet.

Retomamos la Figura 2.6 y en la trama 6 vemos el fin de la transferencia del recurso solicitado, index2.html con el detalle indicado en la Figura 2.6:

Igual que en el caso anterior la primer línea contiene tres campos del response dado por el servidor, la versión de HTTP; el código 200, todo OK como en Telnet; indicando el significado del código, OK. Si vemos la ventana de bytes veremos que finaliza con 0d 0a. Siguiendo con el análisis:

Wed, 14 Apr 2021 11:41:50 GMT\r\n, indica fecha y hora en que se completó la entrega del recurso.

```

▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 200 OK\r\n
    ▶ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
      Date: Wed, 14 Apr 2021 11:41:50 GMT\r\n
      Server: Apache/2.4.29 (Ubuntu)\r\n
      Last-Modified: Wed, 14 Apr 2021 11:39:55 GMT\r\n
      ETag: "b8-5bfed34b49b84"\r\n
      Accept-Ranges: bytes\r\n
    ▶ Content-Length: 184\r\n
      Keep-Alive: timeout=5, max=100\r\n
      Connection: Keep-Alive\r\n
      \r\n

```

Figura 2.6: Captura Fin del GET

Server: Apache/2.4.29 (Ubuntu)\r\n, características del servidor.

Last-Modified: Wed, 14 Apr 2021 11:39:55 GMT\r\n, fecha y hora en que fue actualizado el recurso.

ETag: "b8-5bfed34b49b84"\r\n, tag empleado en caso de haberse requerido el recurso con algún condicionamiento. Accept-Ranges: bytes\r\n, la unidad del rango solicitado.

Content-Length: 184\r\n, no hace falta aclarar.

Keep-Alive: timeout=5, max=100\r\n, permite al servidor dar pistas sobre cómo se puede usar la conexión para establecer un tiempo de espera y una cantidad máxima de solicitudes.

Connection: Keep-Alive\r\n, ya la conocemos.

Finaliza el mensaje de response al request del GET como en el caso anterior con línea \r\n

Continuando con la misma captura analicemos ahora la trama 8 donde n10 hace un GET de una imagen. Mostramos en la Figura 2.7 el detalle del message del método GET para este caso.

```

▼ Hypertext Transfer Protocol
  ▼ GET /imgs/logo.png HTTP/1.1\r\n
    ▶ [Expert Info (Chat/Sequence): GET /imgs/logo.png HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /imgs/logo.png
      Request Version: HTTP/1.1
      Host: www.unlp.edu.ar\r\n
      User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:73.0) Gecko/20100101 Firefox/73.0\r\n
      Accept: image/webp,*/*\r\n
      Accept-Language: es-AR,es;q=0.8,en-US;q=0.5,en;q=0.3\r\n
      Accept-Encoding: gzip, deflate\r\n
      Connection: keep-alive\r\n
      Referer: http://www.unlp.edu.ar/index2.html\r\n
      \r\n

```

Figura 2.7: Nuevo GET

Como vemos hay campos conocidos del GET anterior. Como novedad tenemos:

Referer: http://www.unlp.edu.ar/index2.html\r\n, que permite al cliente especificar, para beneficio del servidor, la dirección (URI) del recurso del que se obtuvo el Request-URI. Lo podemos ver como una evidencia del carácter persistente de HTTP1.1.

El recurso solicitado es un archivo png por lo que es de esperar una alta transferencia de datos. Efectivamente analizando la captura encontramos que recién en la trama 203 finaliza la descarga del archivo solicitado. Nuevamente tenemos un mensaje de fin de transferencia con código 200 previamente analizado, prácticamente igual al de la Figura 2.6.

Con el objeto de analizar otros casos vamos a la trama 284 donde encontramos un nuevo GET, similar al analizado anteriormente. Nos interesa la respuesta obtenida en la trama siguiente, 285 con el detalle del mensaje HTTP que vemos en la Figura 2.8.

```

▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 404 Not Found\r\n
    ▶ [Expert Info (Chat/Sequence): HTTP/1.1 404 Not Found\r\n]
      Response Version: HTTP/1.1
      Status Code: 404
      [Status Code Description: Not Found]
      Response Phrase: Not Found
      Date: Wed, 14 Apr 2021 11:41:50 GMT\r\n
      Server: Apache/2.4.29 (Ubuntu)\r\n
    ▶ Content-Length: 277\r\n
      Keep-Alive: timeout=5, max=97\r\n
      Connection: Keep-Alive\r\n
      Content-Type: text/html; charset=iso-8859-1\r\n

```

Figura 2.8: Respuesta Negativa

El recurso solicitado no se encontró y por eso el servidor envía un mensaje con código 404 que corresponde a esa situación.

Para completar el análisis de esta captura vamos a la trama 593 con un GET particular que detallamos en la Figura 2.9 Observamos unas cuantas novedades:

```

▼ Hypertext Transfer Protocol
  ▼ GET /index2.html HTTP/1.1\r\n
    ▶ [Expert Info (Chat/Sequence): GET /index2.html HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /index2.html
      Request Version: HTTP/1.1
      Host: www.unlp.edu.ar\r\n
      User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:73.0) Gecko/20100101 Firefox/73.0\r\n
      Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
      Accept-Language: es-AR,es;q=0.8,en-US;q=0.5,en;q=0.3\r\n
      Accept-Encoding: gzip, deflate\r\n
      Connection: keep-alive\r\n
      Upgrade-Insecure-Requests: 1\r\n
      If-Modified-Since: Wed, 14 Apr 2021 11:39:55 GMT\r\n
      If-None-Match: "b8-5bfed34b49b84"\r\n
      Cache-Control: max-age=0\r\n
      \r\n

```

Figura 2.9: GET Condicional

`Upgrade-Insecure-Requests: 1\r\n`, envía una señal al servidor expresando la preferencia del cliente por una respuesta encriptada y autenticada.

`If-Modified-Since: Wed, 14 Apr 2021 11:39:55 GMT\r\n`, si el recurso no se ha modificado desde entonces, la respuesta será un 304 sin ningún contenido.

`If-None-Match: "b8-5bfed34b49b84"\r\n`, el servidor devolverá el recurso solicitado, con un estado

200, solo si no tiene un ETag que coincida con los datos. Cuando la condición falla, el servidor debe devolver el código de estado HTTP 304 (No modificado).

Cache-Control: max-age=0\r\n, Indica que el cliente está dispuesto a aceptar una respuesta cuya edad no sea mayor que el tiempo especificado en segundos.

Si se fijan en la trama 594 verán la respuesta del servidor con un mensaje con código 304 debido a que el recurso no fue modificado en el rango. Incluimos la Figura 2.10.

```

▶ Frame 594: 245 bytes on wire (1960 bits), 245 bytes captured (1960 bit
▶ Ethernet II, Src: 00:00:00_aa:00:1a (00:00:00:aa:00:1a), Dst: 36:f0:9f
▶ Internet Protocol Version 4, Src: host163-10-0-72.cespi.unlp.edu.ar (1
▶ Transmission Control Protocol, Src Port: http (80), Dst Port: 39796 (3
▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 304 Not Modified\r\n
    ▶ [Expert Info (Chat/Sequence): HTTP/1.1 304 Not Modified\r\n]
      Response Version: HTTP/1.1
      Status Code: 304
      [Status Code Description: Not Modified]
      Response Phrase: Not Modified
      Date: Wed, 14 Apr 2021 11:41:54 GMT\r\n
      Server: Apache/2.4.29 (Ubuntu)\r\n
      Connection: Keep-Alive\r\n
      Keep-Alive: timeout=5, max=95\r\n
      ETag: "b8-5bfed34b49b84"\r\n
      \r\n

```

Figura 2.10: GET Condicional-Respuesta

¿Qué es eso de Cache-Control?. Pues bien, en los casos en que se accede repetidamente a una página, para evitar realizar el acceso como el que vimos, las páginas accedidas se suelen almacenar en la memoria del navegador, de modo que antes de disparar el acceso verifica si está disponible. Para evitar información no válida/vencida se almacena en la "Cache" con la fecha de acceso.

A modo de resumen nos parece oportuno presentarles la estructura general de un mensaje HTTP, Figura 2.11:

Completamos algunos detalles con un cuadro del resto de los métodos de HTTP:

HEAD: Similar a GET, pero sólo pide las cabeceras HTTP.

POST: Envía datos a una URL para que el recurso en esa URI los gestione.

PUT: Pone un recurso en la dirección especificada en la URL. Exactamente en esa dirección. Si no existe, lo crea, si existe lo reemplaza.

DELETE: Elimina el documento referenciado en la URL.

TRACE: Rastrea los intermediarios por los que pasa la petición.

OPTIONS: Averigua los métodos que soporta el servidor.

Hemos decidido comentar en forma extensa capturas correspondientes al protocolo HTTP 1.1 por cuanto es ampliamente soportado por la mayoría de los servidores. De todos modos debemos mencionar que existe una nueva versión del protocolo. Está disponible HTTP 2 [BPT15]. En pocas palabras

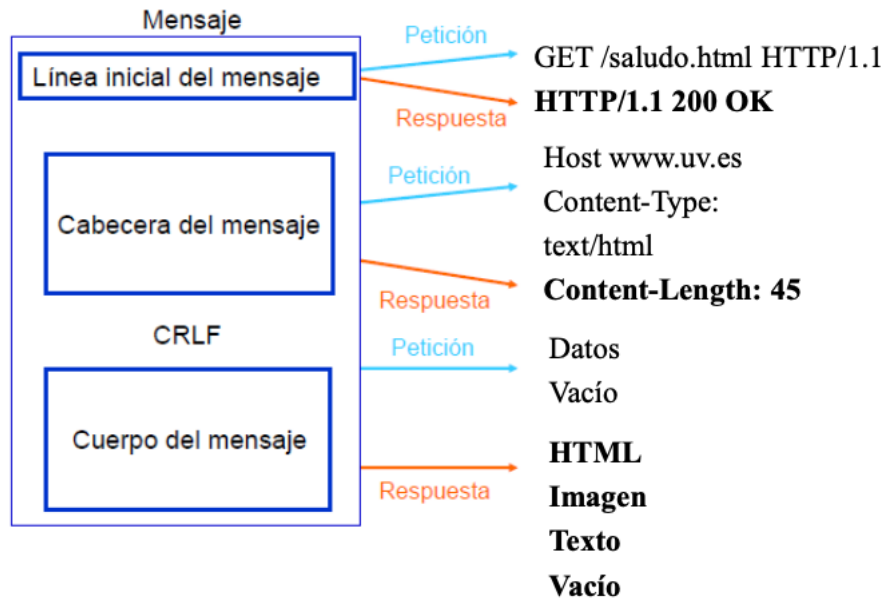


Figura 2.11: Mensaje HTTP

presenta un menor overhead con compresión de los encabezados, requests múltiples en paralelo y un mayor grado de seguridad.

Para finalizar, si se requiere plena seguridad en el acceso entonces se plantea este modelo, Figura 2.12: Como la misma RFC referenciada lo dice HTTPS [Res00] no es otra cosa que HTTP montado

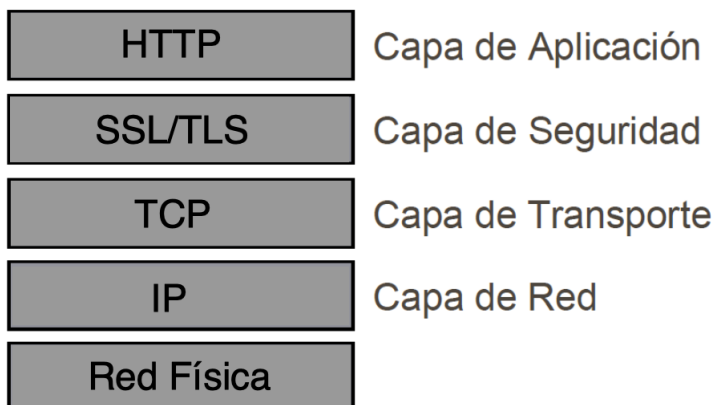


Figura 2.12: HTTPS

sobre SSL⁴ o en su actualización por TLS⁵. El port bien conocido dado por el IANA es el 443.

⁴Secure Socket Layer

⁵Transport Layer Security

REFERENCIAS

- [BLFF96] T. Berners-Lee, R. Fielding y H. Frystyk. «Hypertext transfer protocol – HTTP/1.0», 1996.
- [BPT15] Mike Belshe, Roberto Peon y Martin Thomson. «Hypertext transfer protocol version 2 (HTTP/2)», mayo de 2015.
- [FGM⁺99] Roy T. Fielding, Jim Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach y Tim Berners-Lee. «Hypertext transfer protocol - HTTP/1.1», jun de 1999.
- [Res00] Eric Rescorla. «HTTP over TLS», mayo de 2000.

CAPÍTULO 3

TRADUCIENDO NOMBRES A DIRECCIONES (DNS)

ANDRES BARBIERI

En este capítulo se mostrarán más detalles sobre el sistema de DNS, actor necesario para que el requerimiento HTTP realizado por el usuario tenga éxito, obteniendo así la información contenida en la URL: `http://www.unlp.edu.ar`. El texto no entrará en los detalles de cuestiones como el formato específico del mensaje de DNS y todos los tipos de registros que define. Para esto se puede consultar la bibliografía [FS11], [KR12] y [LA06].

Introducción

El servicio de DNS (Domain Name System) tiene la particularidad que, en la mayoría de los casos, no es utilizado directamente por los usuarios o aplicaciones, sino que funciona como un apoyo al resto de los servicios de Internet. Podríamos decir que no existen user-agents para este sistema, aunque sí varias herramientas como pueden ser: `dig(1)`, `nslookup(1)` y `host(1)`. Su objetivo principal es el de traducir nombres de dominio a direcciones de Internet (direcciones IP) y, de esta forma, lograr una abstracción de las direcciones de red utilizadas internamente por los protocolos del stack TCP/IP. Esto permite ubicar a un dispositivo, o un nodo en la red, por su nombre sin importar cual es la dirección IP que tienen asignada en ese momento. Otra ventaja es que los usuarios personas, no necesitan recordar las direcciones IP. Para IPv4 (32 bits), recordar cuatro números de a lo sumo tres dígitos no parece tan complicado, el problema surge cuando se deben recordar muchos de estos valores y el caso parece empeorar con la necesidad de recordar direcciones IPv6 (128 bits).

El núcleo del DNS es un sistema distribuido de forma jerárquica, el cual está conformado por servidores ubicados a lo largo y ancho de todo el mundo. Cada servidor tendrá la responsabilidad de mantener parte de la información dentro de la jerarquía del espacio de nombres. Para mayor escalabilidad y tolerancia, los servidores, por lo general, tienen réplicas funcionando de forma activo-activo. Este sistema es apoyado por servidores intermedios o de cache, generalmente locales, a los cuales se suman los clientes como miembros.

Historia

El hecho de utilizar nombres en lugar de direcciones data de 1973, cuando todo se mantenía en un archivo global: `HOSTS.TXT`, que era soportado por el SRI (Stanford Research Institute, hoy SRI International). Dentro del SRI, esta tarea dio lugar a la creación del NIC (Network Information Center). El servicio funcionaba de forma completamente centralizada. El archivo `HOSTS.TXT` estaba disponible para “bajar” utilizando el protocolo FTP y las modificaciones al mismo eran solicitadas vía e-mail.

Cada determinado tiempo se juntaban las solicitudes de actualizaciones, se cambiaba el archivo y se generaba una nueva versión disponible al público. El sistema centralizado aseguraba que el NIC asignara las direcciones de forma consistente, pero tenía el problema de no ser escalable ni tolerante a fallas.

Para 1980, este método era muy difícil de mantener por lo que empezaron las investigaciones para reemplazarlo. El encargado de diseñar el nuevo sistema fue Paul Mockapetris, del USC (University of Southern California) Information Sciences Institute, y para fines de 1983, se generaron los documentos RFC-882[Moc83a] y RFC-883[Moc83b], que luego fueron reemplazados, en 1987, por RFC-1034[Moc87a] y RFC-1035[Moc87b]. El sistema diseñado por Mockapetris trabaja de forma distribuida, pero concentra el comienzo de las búsquedas en un conjunto de servidores raíces y luego delega de forma jerárquica. Existen varias RFC definidas dentro del marco del IETF que aportan más detalles y enriquecen el protocolo, como pueden ser sobre la seguridad con DNSSEC[AAL⁺05a][AAL⁺05b].

La primera implementación de un servidor de DNS fue realizada en 1984 en la Universidad de Berkeley y ejecutaba sobre Unix BSD. Más tarde, al ser re-escrita por otros estudiantes de la misma universidad, fue nombrada BIND (Berkeley Internet Name Domain). En la actualidad, hay diferentes versiones del servidor DNS mantenidas por el ISC (Internet Systems Consortium), que fue fundado por Paul Vixie en 1994. Además, ha sido portado a una gran cantidad de plataformas.

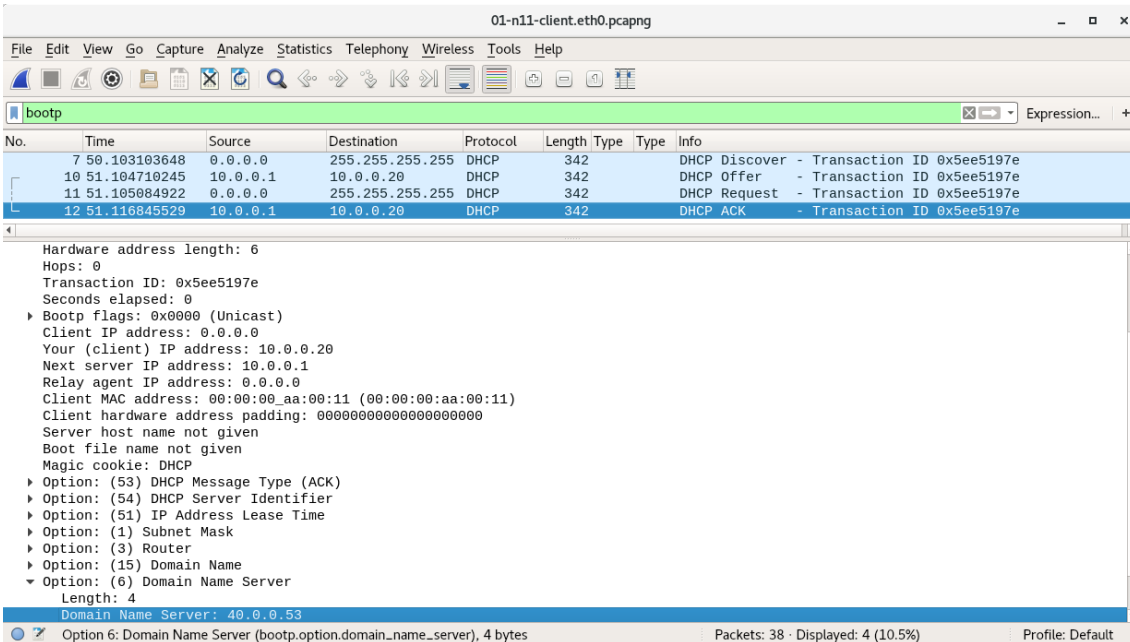
DNS en acción

El requerimiento de DNS

Desde el host *n11-client* el usuario ha “tipeado” en su navegador, o más precisamente aplicación de user-agent, la URL `http://www.unlp.edu.ar`. La aplicación deberá encargarse de encaminar el requerimiento HTTP hacia el servidor (podría estar implementado de distintas formas, como podría ser: por un servicio montado sobre un dispositivo físico, un sistema embebido, una máquina virtual, etc.) en la red que sirve ese recurso web. El servidor será alcanzado por su identificador único, la dirección IP mediante la cual se conecta a Internet. La aplicación que utiliza el usuario contactará al sistema de DNS para solicitarle la traducción del nombre de dominio: *www.unlp.edu.ar*, contenido dentro de la URL, a su identificador IP. La URL además del nombre de dominio, puede contener otras características como el protocolo, el recurso dentro del dominio, etc. Podría estar formada de la siguiente forma: `http://www.unlp.edu.ar:8080/index.html`.

El funcionamiento del DNS se realiza en un esquema cliente/servidor, siendo el cliente, en este caso la aplicación, que requiere la resolución de nombres. El código (instrucciones de máquina) del cliente de DNS se lo agrupa en un módulo de software llamado **Resolver** (en un intento de castellanización podríamos llamarlo “Resolvedor”). El **Resolver** se lo podría considerar como un agente encargado de solucionar las solicitudes del cliente utilizando la infraestructura de DNS. En este caso lo consideramos como parte del user-agent mismo, el que usa el usuario para solicitar el recurso (i.e. el navegador o browser). En el ejemplo la herramienta utilizada es `curl(1)`.

El **Resolver** deberá contactar al **Servidor de DNS local** asignado al host cliente para solicitarle esta traducción de nombre de dominio a dirección IP. El servidor de DNS local es contactado directamente por su dirección IP, que el host cliente obtuvo mediante su configuración, para el ejemplo la configuración se



No.	Time	Source	Destination	Protocol	Length	Type	Type	Info
7	50.103103648	0.0.0.0	255.255.255.255	DHCP	342			DHCP Discover - Transaction ID 0x5ee5197e
10	51.104710245	10.0.0.1	10.0.0.20	DHCP	342			DHCP Offer - Transaction ID 0x5ee5197e
11	51.105084922	0.0.0.0	255.255.255.255	DHCP	342			DHCP Request - Transaction ID 0x5ee5197e
12	51.116845529	10.0.0.1	10.0.0.20	DHCP	342			DHCP ACK - Transaction ID 0x5ee5197e

```

Hardware address length: 6
Hops: 0
Transaction ID: 0x5ee5197e
Seconds elapsed: 0
  Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 10.0.0.20
  Next server IP address: 10.0.0.1
  Relay agent IP address: 0.0.0.0
  Client MAC address: 00:00:00_aa:00:11 (00:00:00:aa:00:11)
  Client hardware address padding: 00000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
  Option: (53) DHCP Message Type (ACK)
  Option: (54) DHCP Server Identifier
  Option: (51) IP Address Lease Time
  Option: (1) Subnet Mask
  Option: (3) Router
  Option: (15) Domain Name
  Option: (6) Domain Name Server
    Length: 4
    Domain Name Server: 40.0.0.53
Option 6: Domain Name Server (bootp.option.domain_name_server), 4 bytes
Packets: 38 · Displayed: 4 (10.5%)
Profile: Default

```

Figura 3.1: Captura de auto-configuración por DHCP del cliente, asignación de servidor de DNS.

realizó de forma automática mediante el protocolo DHCP[Dro97] (ver. figura 3.1 del archivo de captura 01-n11-client.eth0.pcapng).

Usando el protocolo de DNS, el cliente solicitará a su DNS local y esperará la traducción. El formato de los mensajes y este diálogo se puede ver en la figura 3.2 del archivo de captura 01-n11-client.eth0.pcapng tomada en la interfaz del dispositivo cliente. En la misma se observa que se utiliza el protocolo de transporte UDP (User Datagram Protocol) y el port usado por el servidor de DNS local es el 53. Se ve que se consulta por el FQDN (Full Qualified Domain Name), en castellano *Nombre Completo de Dominio*, *www.unlp.edu.ar* y se recibe como respuesta la *IP: 163.10.0.72*. Entre el mensaje número 19, requerimiento, y su respuesta el mensaje número 20, hay aún detalles no revelados, que el sistema local de DNS debió encargarse de solucionar. El diálogo aquí descrito corresponde al identificado en la figura 3.3 entre *n11-client* y *n18-dns-local-resolver*.

En la estructura del mensaje, sin entrar en los detalles, se puede observar que lleva un identificador del mensaje: campo *Transaction ID*. Este campo, en conjunto con los números de ports, sirve para asociar la respuesta con la consulta correspondiente.

Parece importante aclarar al lector que el código del **Resolver** ejecutando en el user-agent, generalmente no se implementa como un servicio activo, sino como un conjunto de rutinas encapsuladas (software) en una biblioteca de funciones que se link-editan conjuntamente con el código de la aplicación. En los sistemas Unix o basados en sus principios, en este caso GNU/Linux, el `resolver(3)(5)` es un conjunto de funciones incluidas en la biblioteca/librería de funciones estándares **C library (libc)**. Este tipo de resolver no realiza forma alguna de caching y se lo suele llamar **Stub/Dumb Resolver**. El encargado del caching, en estos casos, es el **Servidor de DNS local**, que a veces, también se lo suele llamar **Resolver**. Existen otras implementaciones que tienen un resolver activo, conocidos como **Smart Resolver**, funcionando en cada equipo como si fuese un **Servidor de DNS local**. Estos permiten realizar caching u ofrecer funcionalidades extras, como, por ejemplo, manejo de estadísticas o selección de los servidores con

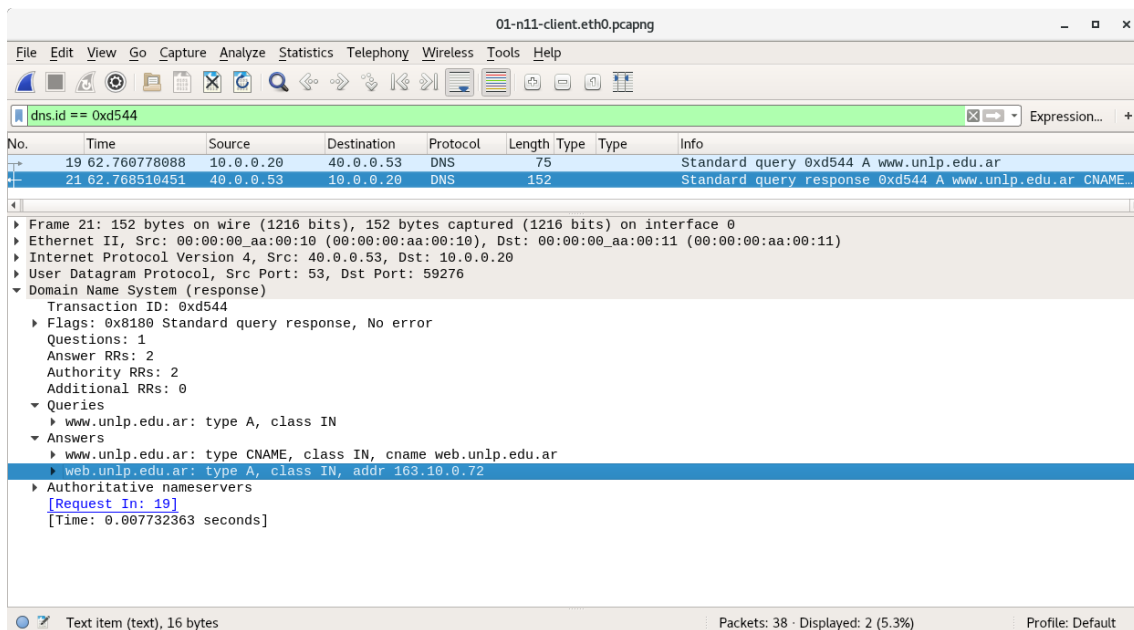


Figura 3.2: Captura del diálogo de DNS entre cliente y servidor local de DNS.

menor RTT. En la actualidad, también existen servidores de DNS públicos funcionando como **Servidor de DNS local**, que se encuentran distribuidos en la Internet, a menudo replicados, como es el caso de los **Public DNS**. Estos son servidores de DNS globales utilizables como locales. Mas detalles se podrán encontrar en URLs como: https://en.wikipedia.org/wiki/Public_recursive_name_server.

Espacio de nombres, FQDN

En párrafos anteriores se mencionó que habían detalles sin resolver. ¿Cómo obtiene el **Servidor de DNS local** la respuesta que entrega al cliente, donde indica que *www.unlp.edu.ar* mapea a la *IP: 163.10.0.72*? Para resolver esta cuestión el servidor deberá interrogar al núcleo del DNS, que, como se mencionó, es un sistema distribuido de forma jerárquica.

Para lograr esta distribución jerárquica se requiere que el espacio de nombres utilizados también así lo sea. Si se presta atención, los FQDN (ver figura 3.4) están formados por una lista de labels (etiquetas) separadas por puntos que van desde el nodo, la etiqueta más a la izquierda, hasta la raíz del árbol, el punto. De esta forma se puede pensar que la etiqueta de la derecha está enganchada en la raíz y a medida que vamos pasando por los puntos se va “subiendo” a la copa del árbol, o de otro modo, al tope de la jerarquía. Gráficamente, el árbol estaría invertido. Ver de ejemplo la figura 3.5.

Otras características que se pueden mencionar de los FQDN es que son “case-insensitive” (es decir, mayúsculas y minúsculas dan igual) y cada etiqueta puede tener como máximo 63 caracteres. La cantidad máxima de etiquetas es 127 y el nombre no puede tener más de 255 caracteres en total. Se distingue por terminar con un punto, “.”, (implícito si no lo tuviese) y se le puede agregar/concatenar, al momento de hacer la búsqueda, un espacio de dominio/sub-dominio. Si el nombre sólo tiene una etiqueta se le agrega un dominio, si tiene más eslabones se lo considera como un FQDN y solo se le agrega el punto final de forma implícita al tratarlo. Existe, también, la posibilidad de escribir nombres de dominios con juegos de símbolos no ASCII llamados Internationalized Domain Name for Applications

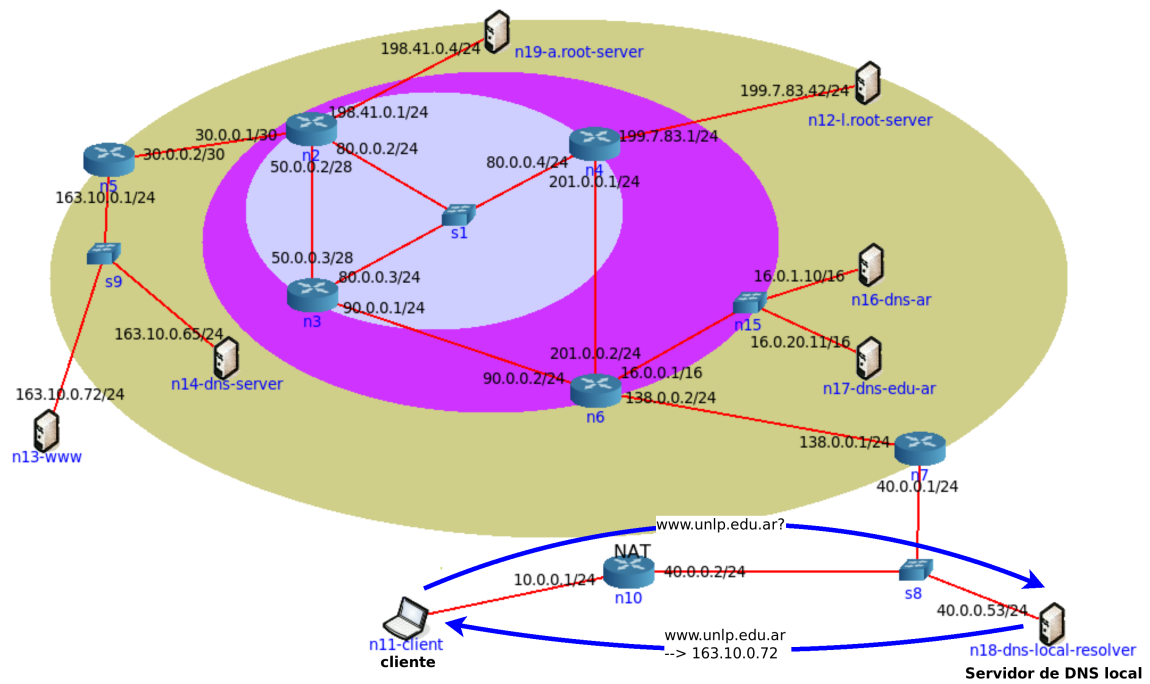


Figura 3.3: Diálogo dentro del sistema DNS entre cliente y servidor local.

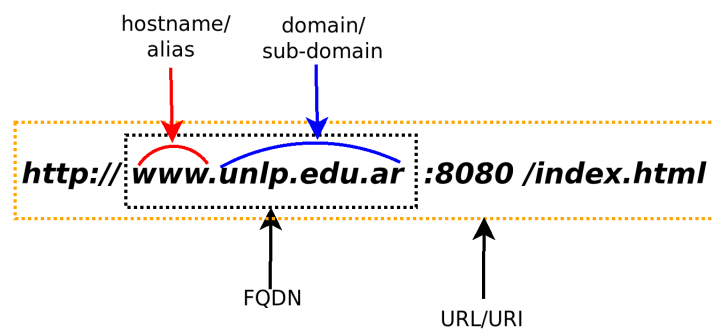


Figura 3.4: URL y FQDN (Nombre completo de dominio).

(IDNA) [Dro10]. La siguiente lista son ejemplos de nombres de dominios, el primero Non-FQDN y el resto FQDN.

- *www* (No FQDN)
- *www.unlp.edu.ar*. (FQDN)
- *www.unlp.edu.ar* (considerado FQDN)
- *web.ru*.
- *www.net*.
- *ada.info.unlp.edu.ar*.
- *www.l.google.com*.
- *other.com*.
- *30.8.10.163.in-addr.arpa*.

Dominios y sub-dominios

Para el ejemplo de *www.unlp.edu.ar* se considera que *unlp.edu.ar* es el dominio para el Non-FQDN *www*. Para el caso del dominio *unlp.edu.ar*, *unlp.edu.ar* es un sub-dominio dentro del dominio *edu.ar* y así lo será *edu.ar* dentro de *ar*. Otros ejemplos serían: *info.unlp.edu.ar* y *psico.unlp.edu.ar* son sub-dominios dentro de *unlp.edu.ar*. Un concepto similar a dominio, que a veces resulta un tanto confuso, es el de zona. Este término está más relacionado con el despliegue de los servidores y lo veremos más adelante.

TLD (Top Level Domain)

Los dominios de primer nivel en la jerarquía, los que se encuentran enganchados a la raíz, son conocidos como TLD (Top Level Domains), y están predefinidos, aunque pueden aparecer nuevos. Actualmente existen TLD genéricos conocidos como gTLD que contienen dominios con propósitos particulares, de acuerdo a diferentes actividades. Para 1980, los gTLD eran: **.com**, **.edu**, **.gov**, **.int**, **.mil**, **.net** y **.org**, pero sólo **.com**, **.net** y **.org** tenían abierto el registro, el resto estaban dedicados (Sponsored TLD). Entre 2011 y 2012 el ICANN abrió los gTLD permitiendo registrar nuevos nombre de dominio en el tope de la jerarquía. El programa indica un proceso de licitación y remate. Para 2015 había más de 500 nuevos dominios, como **.beer**, **.paris**.

De forma paralela a los gTLD existen los ccTLD (Country-Code TLD) que contienen dominios delegados a los diferentes países del mundo. Los códigos de los países están codificados en dos símbolos, habitualmente letras, y actualmente se admiten símbolos o letras de otros alfabetos. Se encuentran, por ejemplo, en este formato **.ar (Argentina)**, **.tv (Tuvalu)**, **.zw (Zimbabwe)**, **.uk (Reino Unido)**, **.ru (Rusia)**, **.us (Estados Unidos)**, etc.

También existen otros dominios con propósitos específicos como el **.ARPA.TLD** que es un dominio especial, llamado de infraestructura, usado internamente por los protocolos para resolución de reversos: de direcciones IP a nombres.

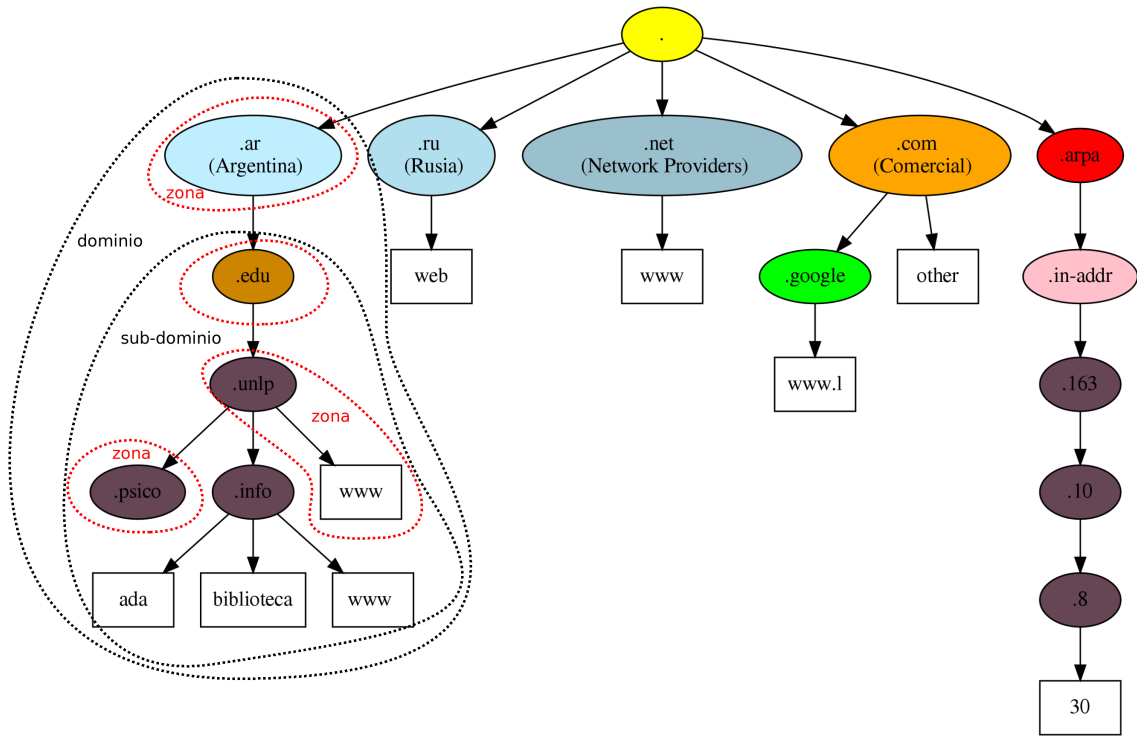


Figura 3.5: Ejemplos de nombres de dominio.

Jerarquía de DNS, servidores de dominios

Una vez organizado el espacio de nombres de forma jerárquica mediante dominios y sub-dominios se lo distribuye en los diferentes servidores que implementan la infraestructura de Internet. Esta asignación también conlleva delegar la administración.

Servidores raíces (Root name servers)

Comenzando desde la raíz, ".", es necesario asignar los servidores raíces, es decir, los servidores que dan origen a las delegaciones o distribución del sistema. Son los que conocen a que servidores delegar la responsabilidad de cada TLD. En la actualidad, existen 13 **Root Name Servers** distribuidos en todo el mundo, de los cuales varios no son servidores únicos, sino que trabajan con redundancia y las réplicas están distribuidos geográficamente. Con esta redundancia, combinada con *Anycast*, se logra que los requerimientos se atiendan según la proximidad del punto de vista de la red, logrando mayor eficiencia (i.e. mejores tiempos de respuesta). La cantidad de servidores raíces está acotada a 13 debido a una limitación inicial de 512 bytes de un mensaje de DNS sobre UDP. A continuación se listan los servidores raíces disponibles en el año 2015.

.	518400	IN	NS	A.ROOT-SERVERS.NET.	# Versign-grs.com
.	518400	IN	NS	B.ROOT-SERVERS.NET.	# ISI.edu
.	518400	IN	NS	C.ROOT-SERVERS.NET.	# Cogent.com (ANYCAST)
.	518400	IN	NS	D.ROOT-SERVERS.NET.	# UMD.edu (Univ. Maryland)
.	518400	IN	NS	E.ROOT-SERVERS.NET.	# NASA.gov
.	518400	IN	NS	F.ROOT-SERVERS.NET.	# ISC.org (ANYCAST)
.	518400	IN	NS	G.ROOT-SERVERS.NET.	# NIC.mil
.	518400	IN	NS	H.ROOT-SERVERS.NET.	# ARMY.mil
.	518400	IN	NS	I.ROOT-SERVERS.NET.	# NIC.ddn.mil (ANYCAST)
.	518400	IN	NS	J.ROOT-SERVERS.NET.	# Versign-grs.com (ANYCAST)
.	518400	IN	NS	K.ROOT-SERVERS.NET.	# RIPE.net (ANYCAST)
.	518400	IN	NS	L.ROOT-SERVERS.NET.	# ICANN.org (ANYCAST)
.	518400	IN	NS	M.ROOT-SERVERS.NET.	# WIDE.ad.jp (ANYCAST)
A.ROOT-SERVERS.NET.	3600000	IN	A	198.41.0.4	



Figura 3.6: Distribución de los ROOT Servers en 2015 (Fuente: www.root-servers.org.)

A.ROOT-SERVERS.NET.	3600000	IN	AAAA	2001:503:ba3e::2:30
B.ROOT-SERVERS.NET.	3600000	IN	A	192.228.79.201
C.ROOT-SERVERS.NET.	3600000	IN	A	192.33.4.12
D.ROOT-SERVERS.NET.	3600000	IN	A	128.8.10.90
E.ROOT-SERVERS.NET.	3600000	IN	A	192.203.230.10
F.ROOT-SERVERS.NET.	3600000	IN	A	192.5.5.241
F.ROOT-SERVERS.NET.	3600000	IN	AAAA	2001:500:2f::f
G.ROOT-SERVERS.NET.	3600000	IN	A	192.112.36.4
H.ROOT-SERVERS.NET.	3600000	IN	A	128.63.2.53
H.ROOT-SERVERS.NET.	3600000	IN	AAAA	2001:500:1::803f:235
I.ROOT-SERVERS.NET.	3600000	IN	A	192.36.148.17
J.ROOT-SERVERS.NET.	3600000	IN	A	192.58.128.30
J.ROOT-SERVERS.NET.	3600000	IN	AAAA	2001:503:c27::2:30

Esta lista se puede obtener vía una consulta de DNS o usando FTP[PR85].

En la figura 3.6 se muestra la distribución de los **Root Name Servers** en el mundo para 2015. En la actualidad, el gráfico se torna bastante más cargado de hitos en el mapa.

Cada servidor de DNS debe conocer al menos uno de esos servidores y poder alcanzarlo para comenzar a operar. En la topología de ejemplo se han definido solo 2 root servers: *n19-a.root-server* y *n12-l.root-server*. El **Servidor de DNS local** del ejemplo, *n18-dns-local-resolver*, debe contactarlos para comenzar a operar.

Servidores TLD y de niveles inferiores

La distribución del servicio se logra mediante la delegación jerárquica. Como se indicó, los servidores raíces deben conocer a los servidores responsables de cada TLD. Esto se conoce como delegación de zona de DNS. La zona sería el dominio sin los sub-dominios delegados, gráficamente, un eslabón en la cadena de delegación. La delegación consiste en saber cuales son los servidores que se encargan de resolver (o sub-delegar) los sub-dominios, llamados en la implementación del servicio como zonas. La delegación se realiza de manera **autoritativa**. Un servidor **autoritativo** tiene toda la información para una zona, puede producir cambios sobre la misma y es el que tiene la última versión. Puede existir más de un servidor autoritativo para una zona y es lo recomendable para ofrecer tolerancia a fallas y

probablemente distribución de carga. De este modo, para el ccTLD **.ar**, se podría obtener la siguiente delegación:

ar.	7172	IN	NS	c.dns.ar.
ar.	7172	IN	NS	a.dns.ar.
ar.	7172	IN	NS	e.dns.ar.
ar.	7172	IN	NS	f.dns.ar.
ar.	7172	IN	NS	ar.cctld.authdns.ripe.net.
ar.	7172	IN	NS	b.dns.ar.
ar.	7172	IN	NS	d.dns.ar.
c.dns.ar.	86400	IN	A	200.108.148.50
...				

Esto mismo sucederá con otros TLDs. Para que un TLD pueda operar debe ser cargada la delegación en los servidores raíces. Luego, de forma similar sucederá que los servidores encargados de los TLD, como el ccTLD **.ar**, tendrán que saber delegar los sub-dominios o zonas, por ejemplo, **.edu.ar**, **.com.ar**, **.org.ar**, etc. La forma de organizar cada dominio es potestad exclusiva de a quien se lo delega. En particular, en la Argentina se ha tomado una política similar que en los TLD y se han generado sub-dominios con etiquetas similares y estos se han sub-delegado generando un estructura similar a la de "arriba". Podría suceder que el dominio de nivel superior no delegue en zonas e implemente los niveles de etiquetas dentro de la misma. Bajando en la jerarquía irá sucediendo lo mismo con respecto a las delegaciones y a la creación de zonas.

En la topología utilizada se implementa de forma simplificada solo un TLD: **.ar** y se sirve con un solo nodo: *n16-dns-ar(a.dns.ar)* Bajo este se implementa el dominio **edu.ar** hosteado por los nodos *n17-dns-edu-ar(ns1.riu.edu.ar)* y *n14-dns-server(anubis.unlp.edu.ar)*. El sub-dominio: **unlp.edu.ar** está implementado en el nodo servidor *n14-dns-server* que es *anubis.unlp.edu.ar*. Las direcciones IP correspondientes a los nodos *a.dns.ar*, *b.dns.ar* y *ns1.riu.edu.ar* son distintas a las reales. Para *a.dns.ar* se usa 16.0.1.10 y para *ns1.riu.edu.ar* 16.0.20.11. En las figuras 3.7 se muestra como está desplegado en la topología.

La resolución de la consulta

Volvamos al **Servidor de DNS local** recibiendo la consulta del FQDN: *www.unlp.edu.ar* desde el cliente. Ahora, el servidor tiene que arreglárselas para responder y va a comenzar consultando desde la raíz del sistema. En este caso va a preguntar a algunos de los **Root Name Servers** que conoce.

En la figura 3.8 del archivo de captura 03-n18-dns-local.eth0.pcapng tomada en la interfaz del dispositivo del servidor local de DNS, *n18-dns-local-resolver*, se puede ver la respuesta del servidor raíz consultado por el DNS local. La respuesta recibida no es la traducción solicitada sino los registros de delegación, una "pista" para que el servidor local se aproxime a la respuesta. Si se continúa analizando las consultas y las respuestas que va recibiendo se puede observar que el comportamiento es el mismo, no recibe la traducción, sino los registros de delegación. Solo cuando llega al servidor autoritativo para el dominio *unlp.edu.ar* recibe la respuesta esperada.

En la figura 3.9 se muestra el diagrama de flujos/interacción entre el servidor local y los servidores que forman parte de la jerarquía de DNS en la topología de ejemplo.

1. Primero consulta a un servidor raíz, *a.root-server.net* (198.41.0.4).
2. Luego consulta a un servidor del TLD extbf.ar *a.dns.ar* (16.0.1.10).
3. Finalmente a *anubis.unlp.edu.ar* (163.10.0.65).

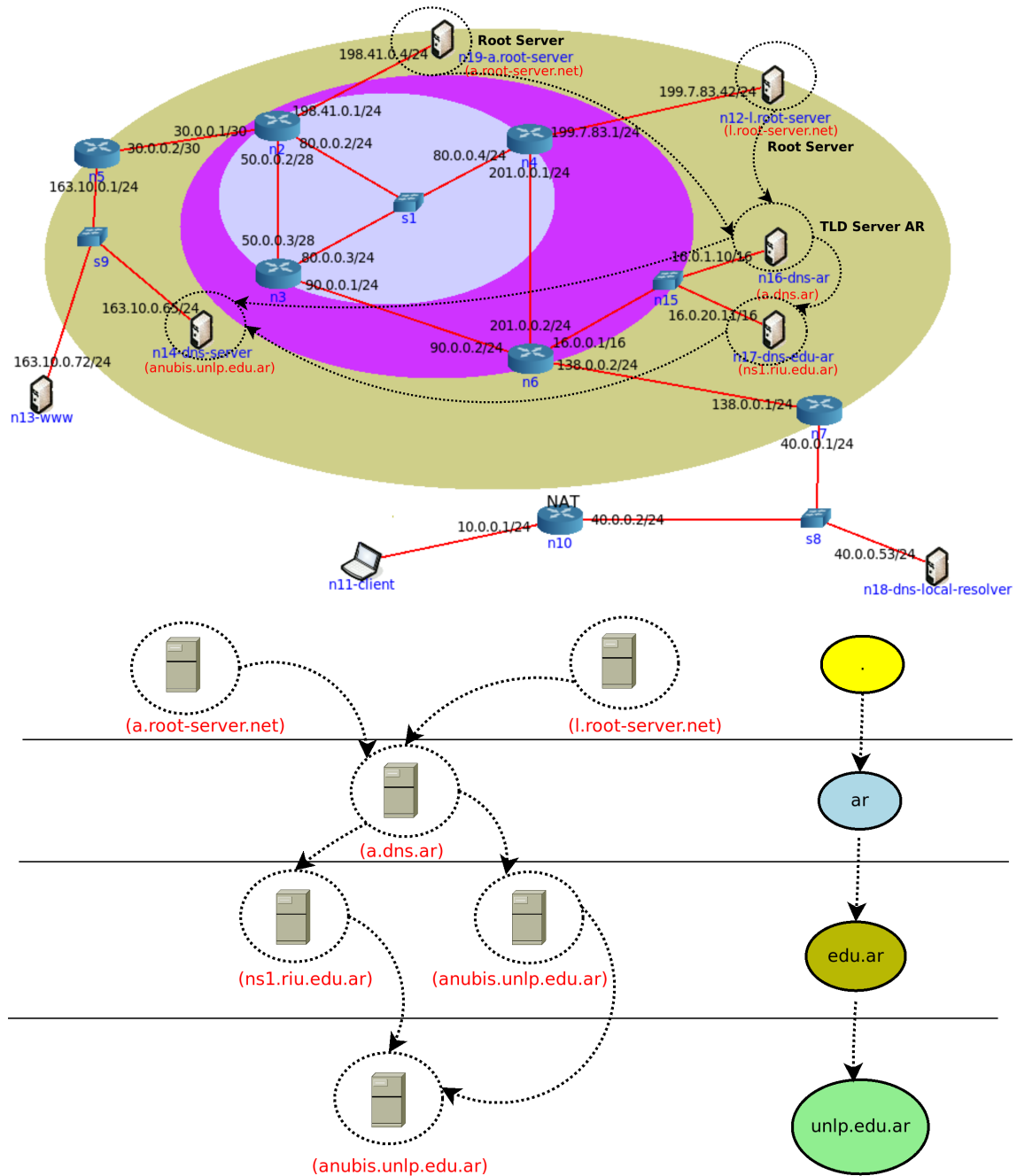


Figura 3.7: Delegación de los sub-dominios en la topología de test.

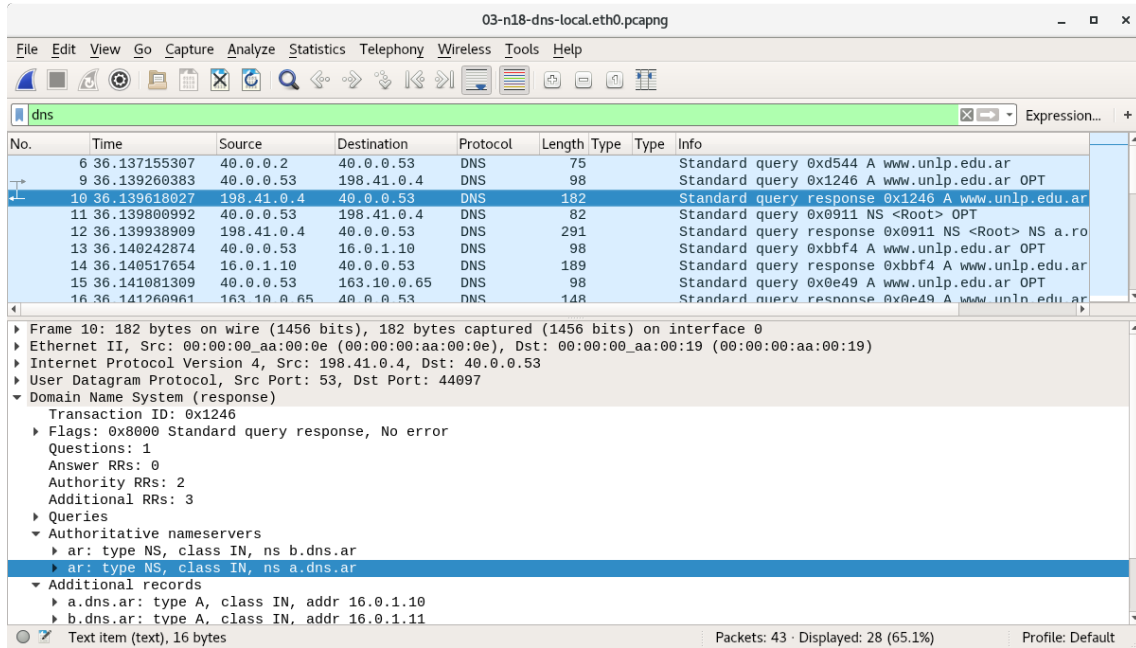


Figura 3.8: Captura del diálogo de DNS entre local y los servidores autoritativos.

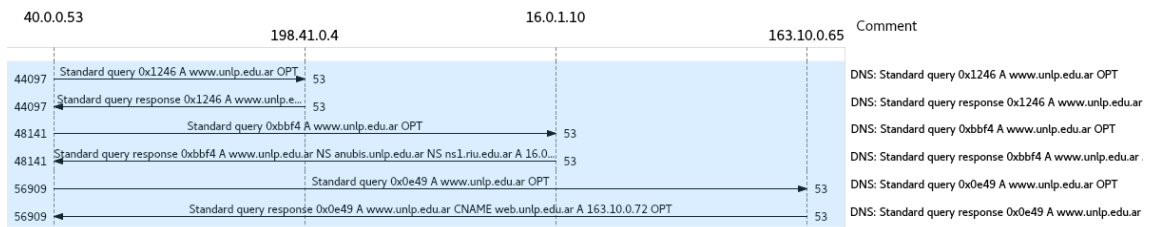


Figura 3.9: Diagrama de interacción entre el DNS local y los servidores autoritativos.

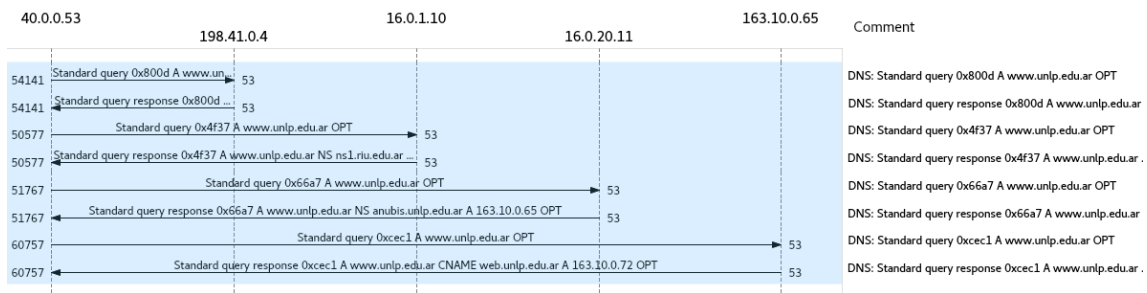


Figura 3.10: Diagrama alternativo de interacción entre el DNS local y los servidores autoritativos.

Pareciese que saltase el dominio **.edu.ar** y consulta directamente a uno de los servidores del dominio **unlp.edu.ar**. Esto no es así, ya que el servidor *anubis.unlp.edu.ar* funciona para ambos dominios y en este caso puede responder de forma autoritativa directamente con la traducción deseada. Otro posible escenario podría ser el mostrado en la figura 3.10 del archivo de captura `03-n18-dns-local.eth0-v2.pcapng`.

1. Consulta a un servidor raíz, *a.root-server.net* o *l.root-server.net*.
2. Consulta a un servidor del TLD **.ar** *a.dns.ar* (16.0.1.10).
3. Luego consulta a un servidor del sub-dominio **edu.ar**, donde podría ser *ns1.rii.edu.ar* (16.0.20.11).
4. Finalmente a *anubis.unlp.edu.ar* (163.10.0.65) de quien recibe la traducción.

Con una secuencia o la otra el servidor local ha obtenido la respuesta donde se le indica que *www.unlp.edu.ar* es un alias a *web.unlp.edu.ar* mediante un registro de tipo “**CNAME**” y que este tiene la *IP: 163.10.0.72* según el registro “**A**” devuelto.

No.	Time	Source	Destination	Protocol	Length	Type	Type	Info
6	36.137155307	40.0.0.2	40.0.0.53	DNS	75	Standard query	0xd544 A	www.unlp.edu.ar
9	36.139260383	40.0.0.53	198.41.0.4	DNS	98	Standard query	0x1246 A	www.unlp.edu.ar
10	36.139618027	198.41.0.4	40.0.0.53	DNS	182	Standard query response	0x1246 A	www.unlp.edu.ar
11	36.139800992	40.0.0.53	198.41.0.4	DNS	82	Standard query	0x0911 NS	<Root> OPT
12	36.139938909	198.41.0.4	40.0.0.53	DNS	291	Standard query response	0x0911 NS	<Root> OPT
13	36.140242874	40.0.0.53	16.0.1.10	DNS	98	Standard query	0xbbf4 A	www.unlp.edu.ar
14	36.140517654	16.0.1.10	40.0.0.53	DNS	189	Standard query response	0xbbf4 A	www.unlp.edu.ar
15	36.141081309	40.0.0.53	163.10.0.65	DNS	98	Standard query	0x0e49 A	www.unlp.edu.ar
16	36.141260961	163.10.0.65	40.0.0.53	DNS	148	Standard query response	0x0e49 A	www.unlp.edu.ar

Frame 16: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface 0
 Ethernet II, Src: 00:00:00_aa:00:0e (00:00:00:aa:00:0e), Dst: 00:00:00_aa:00:19 (00:00:00:aa:00:19)
 Internet Protocol Version 4, Src: 163.10.0.65, Dst: 40.0.0.53
 User Datagram Protocol, Src Port: 53, Dst Port: 56909
 Domain Name System (response)
 Transaction ID: 0x0e49
 Flags: 0x8400 Standard query response, No error
 Questions: 1
 Answer RRs: 2
 Authority RRs: 0
 Additional RRs: 1
 Queries
 Answers
 www.unlp.edu.ar: type CNAME, class IN, cname web.unlp.edu.ar
 web.unlp.edu.ar: type A, class IN, addr 163.10.0.72
 Additional records
 Text item (text), 16 bytes
 Packets: 43 · Displayed: 28 (65.1%) Profile: Default

Figura 3.11: Captura del diálogo de DNS entre local y el autoritativo de unlp.edu.ar.

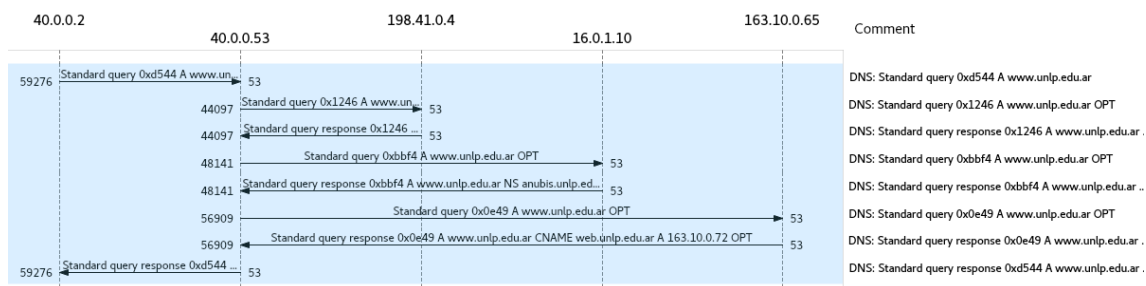


Figura 3.12: Diagrama de interacción entre el cliente, el DNS local y los autoritativos.

El requerimiento y la respuesta completa

Ahora, el **Servidor de DNS local** ha obtenido la respuesta. En la misma tiene la información solicitada por el cliente al cual le deberá entregar la información. Es la respuesta que se muestra en la figura 3.2. El diagrama de interacción completo se puede ver en la figura 3.12.

Al recibir la respuesta de su **Servidor de DNS local** el cliente ya tiene la información necesaria para construir el requerimiento HTTP y contactar al servidor web del sitio `http://www.unlp.edu.ar`, como se podrá aprender en el capítulo correspondiente.

Una cuestión que puede resultar extraña al lector, si compara la captura tomada en la interfaz del cliente con la tomada en la interfaz del servidor local, es que el requerimiento de DNS sale con la dirección IP origen 10.0.0.20 y llega al destino 40.0.0.53 con la dirección origen 40.0.0.2. Eso se debe a un proceso de traslación de direcciones llevada a cabo por el router que da servicio al cliente. Este mecanismo se llama NAT, o más precisamente NAPT [SH99], y no es recomendado por romper el principio de end-to-end [Car96]. Hoy en día es un requerimiento de facto debido a la escasez de direcciones IPv4.

Otros Detalles

Consultas recursivas e iterativas

Si se analiza el escenario y los actores se pueden observar dos formas de consultar o de resolver la petición. Una es la que utiliza el cliente con su servidor, en la cual al hacer la petición se desentiende de los detalles y espera la respuesta final. Este método se llama *Recursivo* y se puede divisar en los flags de los mensajes de DNS. La otra forma es la que utiliza el servidor local. En el segundo caso, él mismo tiene que hacer el trabajo de ir consultando paso por paso la jerarquía de DNS hasta obtener la respuesta que entregará al cliente. Este método se llama *Iterativo* y también se puede divisar en los flags. Esta comparación se puede hacer observando las figuras 3.13. En la consulta recursiva se espera que el servidor al cual se indaga resuelva la cuestión haciendo las búsquedas necesarias, o si tenemos suerte se entregará desde cache. Para el caso de las iterativas solo se espera que el servidor responda con lo que el conoce, en particular la información sobre la cual tiene autoridad. Los servidores de DNS, en la mayoría de los casos restringen las consultas recursivas a los posibles clientes finales que atenderán. En este caso, el servidor local del ISP del ejemplo no atenderá consultas recursivas de clientes que no pertenezcan a su porción de red.

The image displays two Wireshark capture windows side-by-side, illustrating DNS flags for recursive and iterative queries.

Top Window: 01-n11-client.eth0.pcapng

- Filter: `dns.id == 0xd544`
- Packet 21: Standard query response 0xd544 A www.unlp.edu.ar CNAME...
- Transaction ID: 0xd544
- Flags: 0x8180 Standard query response, No error
- Bit 1 (0x0001): Recursion available: Server can do recursive queries
- Bit 0 (0x0000): Recursion desired: Do query recursively

Bottom Window: 03-n18-dns-local.eth0.pcapng

- Filter: `dns`
- Packet 10: Standard query response 0x1246 A www.unlp.edu.ar NS...
- Transaction ID: 0x1246
- Flags: 0x8000 Standard query response, No error
- Bit 0 (0x0000): Recursion available: Server can't do recursive queries
- Bit 1 (0x0001): Recursion desired: Don't do query recursively

Figura 3.13: Flags que distinguen método recursivo de iterativo.

En el modelo de prueba también se tiene como “usuario” del servicio al servidor web, *n13-www*, o directamente *web.unlp.edu.ar*. En este caso el host que cumple las funciones de servidor http usará su propio servidor DNS local, *n14-dns-server (anubis.unlp.edu.ar)*. Este es un caso particular donde el servidor de DNS funciona como autoritativo y DNS o resolver local.

Respuestas autoritativas

Otra cuestión importante con respecto a los servidores autoritativos y las respuestas autoritativas es identificarlas. Se aclaró que los servidores autoritativos son los responsables para un dominio o zona. Estos son los encargados de mantener la información de la misma y actualizarla cuando sea necesaria. Son los que se indican con los registros “**NS**”. Las respuestas que estos brinden de los registros que poseen irán marcadas con el flag de autoritativa. Los servidores que cacheen esta información, como es el caso de el **Servidor de DNS local**, mantendrán en cache esta información mientras el tiempo de cache señalado por el autoritativo sea válido. Ante las reiteradas consultas por los mismos registros podrán entregar esta información desde la cache. De esta forma, se ahorra hacer la búsqueda por cada consulta recibida y se entregará la información almacenada con un flag que indique que dicha respuesta no es autoritativa. Para ver el comportamiento autoritativo y no autoritativo se puede ver la captura de la respuesta que entrega *anubis* al servidor local de la red del ISP, *03-n18-dns-local.eth0.pcapng*, fila 16, y la que entrega el servidor local al cliente, *01-n11-client.eth0.pcapng*, fila 19, donde se marca que no es autoritativa. Si se volviese a consultar por la misma información al servidor local, este la entregaría desde la cache hasta que venza su temporizador de almacenamiento.

Mecanismos de sincronización ente servidores

Para un mejor funcionamiento del sistema se recomienda tener 2 o más servidores autoritativos para una misma zona. Esto ofrece tolerancia a fallas y un mejor funcionamiento permitiendo balance de carga entre los mismos. Para una configuración más flexible se selecciona a uno de los servidores autoritativos como el principal(master), o primario, y el resto se los llama secundarios o slaves. El protocolo de DNS propone un mecanismo de sincronización entre primarios y secundarios llamado **Transferencia de Zona (XFR)**. Este método como transporta más información que la respuesta a una consulta típica utiliza como protocolo de transporte TCP (Transport Control Protocol). El protocolo TCP debe estar siempre disponible en los servidores y, de acuerdo a la cantidad de información a transmitir cliente-servidor pueden optar cuál utilizar.

REFERENCIAS

- [AAL⁺05a] R. Arends, R. Austein, M. Larson, D. Massey y S. Rose. «Rfc 4033: Dns security introduction and requirements», 2005.
- [AAL⁺05b] R. Arends, R. Austein, M. Larson, D. Massey y S. Rose. «Rfc 4035: Protocol modifications for the dns security extensions», 2005.
- [Car96] B. Carpenter(Ed.). «Rfc 1958: Architectural principles of the internet», 1996.
- [Dro97] R. Droms. «Rfc 2131: Dynamic host configuration protocol», 1997.
- [Dro10] R. Droms. «Rfc 5890: Internationalized domain names for applications (idna): Definitions and document framework», 2010.
- [FS11] K. Fall y R. Stevens. «Tcp/ip illustrated, volume 1: The protocols, 2nd. ed.». Addison-Wesley, 2011. ISBN: 9780321336316.
- [KR12] J. Kurose y K. Ross. «Computer networking: A top-down approach, 6th. ed.». Addison-Wesley, 2012. ISBN: 0132856204.
- [LA06] Cricket Liu y Paul Albitz. «Dns and bind (5th edition)». O'Reilly Media, Inc., 2006. ISBN: 0596100574.
- [Moc83a] P. V. Mockapetris. «Rfc 882: Domain names: Concepts and facilities», 1983.
- [Moc83b] P. V. Mockapetris. «Rfc 883: Domain names: Implementation specification», 1983.
- [Moc87a] P. V. Mockapetris. «Rfc 1034: Domain names: Concepts and facilities», 1987.
- [Moc87b] P. V. Mockapetris. «Rfc 1035: Domain names: Implementation specification», 1987.
- [PR85] J. Postel y J. Reynolds. «Rfc 959: File transfer protocol (ftp)», 1985.
- [SH99] P. Srisuresh y M. Holdrege. «Rfc 2663: Ip network address translator (nat) terminology and considerations», 1999.

CAPÍTULO 4

CONEXIÓN DE EXTREMO A EXTREMO (TRANSPORTE)

NÉSTOR CASTRO

Introducción

En este capítulo veremos el accionar de actores que podríamos suponer de reparto ya que sostienen a la aplicación, aunque está claro que los papeles principales y secundarios en protocolos de arquitectura de redes no existen debido a que la dependencia de unos con otros es tan fuerte que si falla uno de ellos se reciente toda la estructura.

Los protocolos de la capa de transporte (TCP y UDP) dan servicios a los de la capa de aplicación por lo que van a tener que “hablar” con ellos (en nuestro caso con HTTP), pero también, como usan servicios de la capa de red, deberán interactuar con el protocolo de la misma (o sea IP).

Como vamos a tener varias conexiones simultáneas a nivel de aplicaciones, los actores que veremos permitirán que las mismas se lleven a cabo al mismo tiempo ya sea en el cliente como en el servidor y lo logran interactuando con el sistema operativo que será el responsable de establecer los procesos en cada caso.

Todo lo dicho nos lleva a la idea que es necesario abrir puertas para que el flujo de información fluya desde y hacia la aplicación. Esas puertas, denominadas puertos, en los servidores estarán asociadas biunívocamente con las aplicaciones y se abrirán ni bien sean invocadas (por ej. en nuestro caso el puerto 80 asociado normalmente con la aplicación HTTP). Por otro lado, los mensajes originados por las aplicaciones deben ser transportados en segmentos que generan los protocolos que se ocupan del transporte.

Pasamos a ver a estos actores que como todos los involucrados en el escenario planteado en deben cumplir con funciones específicas.

UDP (User Datagram Protocol)

Es el protocolo más simple de los dos que veremos, es limitado en sus funciones y va a hacer lo que pueda (en realidad muy poco) para que los segmentos de datos, generalmente denominados datagramas, generados con mensajes de las aplicaciones lleguen al destino. Podríamos decir que es bastante irresponsable desde ese punto de vista y su forma de operar se denomina *best-effort* (mejor esfuerzo).

Una vez generados los datagramas en una estación podrían no llegar a destino o bien llegar desordenados, pero es tan básico el manejo de UDP que no se entera que es lo que pasó realmente. Por supuesto, su forma de operar debe tenerse en cuenta y solo será usado por aquellas aplicaciones que

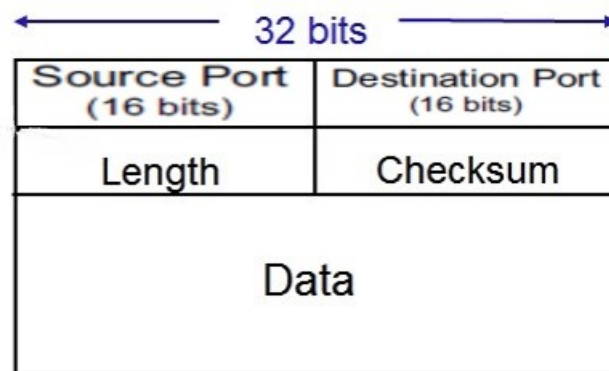


Figura 4.1: Datagrama - UDP

pueda soportar dicho comportamiento. Es por eso que se asociará un datagrama UDP con un mensaje de la aplicación entero, o sea, NO cortará un mensaje en varios datagramas ya que, como dijimos, no tiene la “inteligencia” necesaria para controlarlos y rearmar el mensaje original en el receptor.

Todo lo dicho tiene que ver con una característica muy importante que UDP no tiene y es que si bien establece una conexión, no mantiene el control de la misma, mecanismo que vamos a ver implementado en el “robusto” TCP. Como ventaja, al no realizarse dicho control hace que todo sea más simple, pero con el precio de la incertidumbre que genera no saber si llegó o no el datagrama. Esto último no es un problema para aplicaciones que no requieren tener esas certezas, a final de cuentas algunas de ellas no son tan exigentes. Por el otro lado, hay una ventaja y es que andando todo bien los retardos debido a chequeos (que está claro acá prácticamente no existen) se minimizan y algunas aplicaciones del tipo multimediales ven con buenos ojos esto ya que optimizarán sus tiempos de respuesta.

En resumen, si se utiliza UDP como protocolo de transporte y la aplicación hace algún tipo de consulta, si la misma no llega a destino repetirá la consulta hasta que reciba una respuesta y si una aplicación informa, por ej. el estado de un dispositivo, seguro que lo hará nuevamente en un tiempo breve, por lo tanto, no es tan grave.

Como vemos, como todo en la vida hay ventajas y desventajas. Lo importante es detenernos en las primeras y saber cómo sacar partido de las mismas y claramente las aplicaciones que usan UDP saben cómo hacerlo y en todo caso, y de ser necesario, serán ellas las responsables de proporcionar la confiabilidad que UDP no les brinda para el transporte de datos.

El encabezamiento del datagrama UDP, como es de esperar, no es muy complejo ni muy extenso como vemos en la Figura 4.1.

Solo con los puertos fuente y destino, el tamaño total del datagrama y un *checksum* (de uso opcional) que como su nombre lo indica sirve para chequear si ha sufrido alguna modificación el datagrama recibido, UDP realiza su tarea.

Lo que sigue es una muestra de una captura donde se usa UDP como protocolo de transporte (la denominada 03-n18-dns-local.eth0). Nos detenemos en el evento 6 que corresponde a un datagrama (son todos similares como se puede comprobar) que corresponde al primero usado para una consulta del protocolo DNS (*Domain Name Service*) como se ve en la Figura 4.2:

Vemos el puerto fuente u origen (59276) y el destino (53), el tamaño del datagrama (41 bytes) y el *checksum* (0x55b1).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	fe80::30aa:70ff:fe8...	ff02::2	ICMPv6	70	Router Solicitation from 0a:5d:97:d6:3c:6e
2	10.241257638	fe80::200:ff:feaa:19	ff02::2	ICMPv6	70	Router Solicitation from 00:00:00:aa:00:19
3	14.336273963	fe80::85d:97ff:fed6...	ff02::2	ICMPv6	70	Router Solicitation from 0a:5d:97:d6:3c:6e
4	36.137103166	00:00:00_aa:00:0f	Broadcast	ARP	42	Who has 40.0.0.53? Tell 40.0.0.2
5	36.137139864	00:00:00_aa:00:19	00:00:00_aa:00:0f	ARP	42	40.0.0.53 is at 00:00:00:aa:00:19
6	36.137155307	40.0.0.2	40.0.0.53	DNS	75	Standard query 0xd544 A www.unlp.edu.ar
7	36.139203895	00:00:00_aa:00:19	Broadcast	ARP	42	Who has 40.0.0.1? Tell 40.0.0.53
8	36.139256141	00:00:00_aa:00:0e	00:00:00_aa:00:19	ARP	42	40.0.0.1 is at 00:00:00:aa:00:0e
9	36.139260383	40.0.0.53	198.41.0.4	DNS	98	Standard query 0x1246 A www.unlp.edu.ar OPT
10	36.139618027	198.41.0.4	40.0.0.53	DNS	182	Standard query response 0x1246 A www.unlp.edu.ar NS b.dns.ar NS
11	36.139800992	40.0.0.53	198.41.0.4	DNS	82	Standard query 0x0911 NS <Root> OPT
12	36.139938909	198.41.0.4	40.0.0.53	DNS	291	Standard query response 0x0911 NS <Root> NS a.root-servers.net

```

> Frame 6: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface 0
> Ethernet II, Src: 00:00:00_aa:00:0f (00:00:00:aa:00:0f), Dst: 00:00:00_aa:00:19 (00:00:00:aa:00:19)
> Internet Protocol Version 4, Src: 40.0.0.2, Dst: 40.0.0.53
< User Datagram Protocol, Src Port: 59276, Dst Port: 53
  Source Port: 59276
  Destination Port: 53
  Length: 41
  Checksum: 0x55b1 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 0]
> Domain Name System (query)

```

Figura 4.2: Muestra 1

Como ya hemos dicho, la información de la aplicación se encapsula en el datagrama de la capa de transporte por lo que también se puede ver el contenido del campo de datos, o sea, lo relativo al protocolo DNS, en este caso.

Teniendo en cuenta todo lo dicho, UDP se utiliza principalmente con aplicaciones que tengan las siguientes características:

- Cuando el intercambio de mensajes es muy escaso en cantidad y tamaño, ej. consultas de DNS.
- Cuando la aplicación es en tiempo real y no se puede esperar confirmación de llegada (ACK) por lo que significaría dicho retardo, ej. videoconferencia, VoIP.
- Cuando los mensajes se producen regularmente y no importa si se pierden algunos, ej. SNMP.
- Cuando el medio de transmisión es altamente confiable y sin congestión (LAN), ej. NFS.
- Para el envío de tráfico de tipo *broadcast* o *multicast*.

Para más información sobre el protocolo de transporte UDP ver [Pos80].

TCP (Transport Control Protocol)

Todo lo que aparentemente planteábamos como malo o limitado en UDP, ahora en TCP serán sus características distintivas: orientado a conexión, ordenado y confiable. ¿Qué más se le puede pedir?.

Vamos por parte. Que sea orientado a conexión nos indica que antes de usar la aplicación, a nivel de capa de transporte, habrá que realizar algún tipo de arreglo entre los extremos (cliente-servidor) previo a la apertura de los puertos utilizados. En el caso de TCP, es un intercambio de segmentos básicos de triple vía (denominado también como 3-way handshake) donde los Flags del encabezamiento de estos segmentos SYN y ACK deben cumplir con una secuencia prefijada y, además, se aprovecha para intercambiar el tamaño de segmento máximo (Maximun Segment Size - MSS), el tamaño de ventana de recepción de datos (W) que ambos pueden manejar e inicializan la secuencia de datos mediante el uso de los campos Sequence Number (ISN) y Acknowledgment Number (ACK) (ver Figura 4.3 y Figura 4.4).

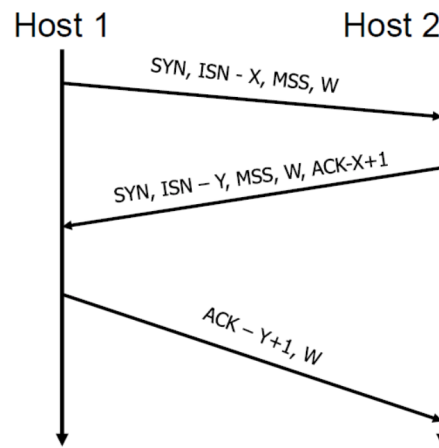


Figura 4.3: Triple Handshake - TCP

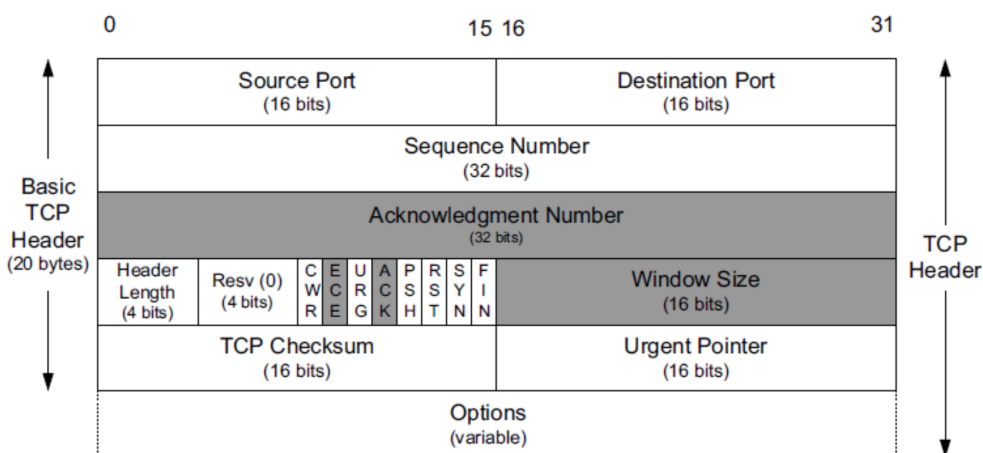


Figura 4.4: Encabezamiento - TCP

Es el cliente el que envía el primer segmento, con el bit SYN activado (o seteado), al servidor para solicitar el inicio de una conexión. El servidor puede aceptar la conexión o rechazarla. Si decide aceptarla aprovecha el segmento con el ACK para enviar su solicitud de conexión en sentido inverso activando el bit SYN, paso seguido el cliente aceptará su pedido enviando un segmento con el bit ACK activado. Si el servidor, por alguna causa, decide rechazar la conexión una vez recibido el primer segmento enviado por el cliente con el bit SYN activado lo hará mediante el envío al cliente de un segmento con el bit de Flag RST activado.

Debido a lo dicho hasta ahora sobre TCP, el encabezamiento del segmento y en comparación con el de UDP debería ser un poco más complejo y así lo es.

Para los fines que persigue el presente libro hemos decidido focalizarnos en algunos de los campos que componen el segmento y se presentan en la Figura 4.4, puerto fuente u origen, puerto destino, número de secuencia, número de confirmación y Flags.

Quizás ya sea el momento de presentar parte de una captura, la denominada 06-n13-www-http-server.eth0, como para ir viendo lo dicho en el párrafo anterior, Figura 4.5.

En los eventos 23-25 podemos ver claramente el 3-way handshake.

Completada la etapa de conexión tenemos establecido un camino virtual en dos sentidos por el cual se enviarán los mensajes de la aplicación mediante varios segmentos que serán identificados secuencial-

No.	Time	Source	Destination	Protocol	Length	Info
22	62.768701474	40.0.0.53	10.0.0.20	DNS	141	Standard query response 0x4459 AAAA www.unlp.edu.ar CHAME web.unlp.edu.ar SOA anubis.unlp.edu.ar
23	62.77543769	10.0.0.20	163.10.0.72	TCP	74	48028 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2958869436 TSecr=0 WS=128
24	62.77573668	163.10.0.72	10.0.0.20	TCP	74	80 → 48028 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2693093558 TSecr=2958869436 WS=128
25	62.775751442	10.0.0.20	163.10.0.72	TCP	66	48028 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2958869437 TSecr=2693093558
26	62.775809588	10.0.0.20	163.10.0.72	HTTP	145	GET / HTTP/1.1
27	62.775865470	163.10.0.72	10.0.0.20	TCP	66	80 → 48028 [ACK] Seq=1 Ack=80 Win=65152 Len=0 TSval=2693093558 TSecr=2958869437
28	62.776411902	163.10.0.72	10.0.0.20	HTTP	369	HTTP/1.1 200 OK
29	62.776422989	10.0.0.20	163.10.0.72	TCP	66	48028 → 80 [ACK] Seq=80 Ack=304 Win=64128 Len=0 TSval=2958869437 TSecr=2693093558
30	62.776661484	10.0.0.20	163.10.0.72	TCP	66	48028 → 80 [FIN, ACK] Seq=80 Ack=304 Win=64128 Len=0 TSval=2958869438 TSecr=2693093558
31	62.77754742	163.10.0.72	10.0.0.20	TCP	66	80 → 48028 [FIN, ACK] Seq=304 Ack=81 Win=65152 Len=0 TSval=2693093559 TSecr=2958869438
32	62.777560755	10.0.0.20	163.10.0.72	TCP	66	48028 → 80 [ACK] Seq=81 Ack=305 Win=64128 Len=0 TSval=2958869438 TSecr=2693093559
33	76.904132696	10.0.0.20	40.0.0.53	DNS	80	Standard query 0xd73f A noexiste.unlp.edu.ar

> Frame 23: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
 > Ethernet II, Src: 00:00:00:aa:00:11 (00:00:00:aa:00:11), Dst: 00:00:00:aa:00:10 (00:00:00:aa:00:10)
 > Internet Protocol Version 4, Src: 10.0.0.20, Dst: 163.10.0.72
 * Transmission Control Protocol, Src Port: 48028, Dst Port: 80, Seq: 0, Len: 0
 Source Port: 48028
 Destination Port: 80
 [Stream index: 0]
 [TCP Segment Len: 0]
 Sequence number: 0 (relative sequence number)
 [Next sequence number: 0 (relative sequence number)]
 Acknowledgment number: 0
 1010 = Header Length: 40 bytes (10)
 > Flags: 0x002 (SYN)

Figura 4.5: Muestra 2

mente de forma tal que al arribar a destino puedan ser reordenados armando el mensaje original de la aplicación. Lo expuesto hace que TCP sea considerado confiable, lo que envía llegará en orden o en desorden y si no llega el segmento será identificado como no recibido y se enviará nuevamente por el emisor o bien sera pedido su reenvío por el receptor ya que se tendrá forma de controlar la llegada de los segmentos y como ya dijimos de reordenarlos en el caso que sea necesario.

Ahora, ¿cómo puede ser que teniendo un camino virtual a nivel transporte los segmentos puedan llegar en desorden?, recordemos que los protocolos de capa de transportes actúan por encima del protocolo de la capa de red, o sea IP (Internet Protocol) que es del tipo “best-effort”, como lo es UDP, y que en este caso genera datagramas que contienen los segmentos que arribarán a destino por distintos caminos lo que significa que los segmentos podrán llegar fuera de orden, pero por suerte para la aplicación, está TCP para volver a acomodarlos.

En el evento 29 de la Figura 4.6 podemos ver un segmento y su puerto fuente (48028), el destino (80), el numero de secuencia (80) y el número de confirmación (304) y el estado de los flags, en este caso el ACK está activado o seteado.

Hemos dicho que TCP es un protocolo orientado a conexión y por lo tanto además de su establecimiento, necesita una finalización ordenada para liberar recursos, especialmente del servidor, que estarán nuevamente disponibles para futuras conexiones.

La desconexión puede ser iniciada por cualquiera de los extremos, el equipo que desee terminar envía un segmento con el bit FIN activado y el otro equipo responde con otro con el bit ACK activado. Este último podría haber aprovechado y enviar también en el mismo segmento su pedido de desconexión con el bit FIN activado, pero si no lo hace puede seguir enviando datos, por lo que después deberá solicitar la desconexión con un segmento TCP con el bit FIN activado que será contestado por el receptor con un segmento con ACK correspondiente.

Lo expresado anteriormente patentiza el hecho de que debemos considerar que en el momento de la conexión se han establecido dos caminos uno de ida y otro de vuelta y que los mismos son independientes, es por eso que se necesita este doble cierre, tal como vemos en la Figura 4.7.

27	62.775865470	163.10.0.72	10.0.0.20	TCP	66 80 → 48028 [ACK] Seq=1 Ack=80 Min=65152 Len=0 TSval=2693093558 TSecr=2958869437
28	62.776411902	163.10.0.72	10.0.0.20	HTTP	369 HTTP/1.1 200 OK
29	62.776422989	10.0.0.20	163.10.0.72	TCP	66 48028 → 80 [ACK] Seq=80 Ack=304 Min=64128 Len=0 TSval=2958869437 TSecr=2693093558
30	62.776661484	10.0.0.20	163.10.0.72	TCP	66 48028 → 80 [FIN, ACK] Seq=80 Ack=304 Min=64128 Len=0 TSval=2958869438 TSecr=2693093558
31	62.777554742	163.10.0.72	10.0.0.20	TCP	66 80 → 48028 [FIN, ACK] Seq=304 Ack=81 Min=65152 Len=0 TSval=2693093559 TSecr=2958869438
32	62.777560755	10.0.0.20	163.10.0.72	TCP	66 48028 → 80 [ACK] Seq=81 Ack=305 Min=64128 Len=0 TSval=2958869438 TSecr=2693093559

Source Port: 48028
Destination Port: 80
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 80 (relative sequence number)
[Next sequence number: 80 (relative sequence number)]
Acknowledgment number: 304 (relative ack number)
1000 = Header Length: 32 bytes (8)
▼ Flags: 0x010 (ACK)
000. = Reserved: Not set
...0 = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
....0.. = Push: Not set
....0. = Reset: Not set
....0. = Syn: Not set
....0. = Fin: Not set
[TCP Flags:A....]
Window size value: 501
[Calculated window size: 64128]
[Window size scaling factor: 128]
Checksum: 0x7efd [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
> Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
> [SEQ/ACK analysis]
> [Timestamps]

Figura 4.6: Muestra 3

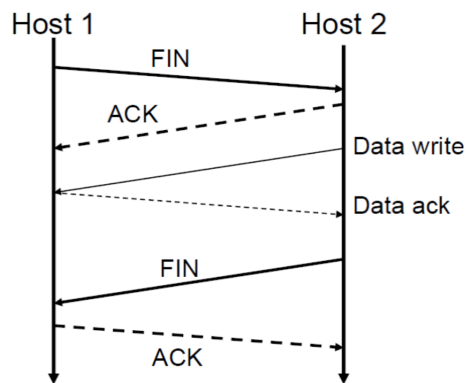


Figura 4.7: Desconexión - TCP

244	14.088978363	163.10.0.72	10.0.0.20	TCP	1514	80 → 56774	[ACK]	Seq=169417	Ack=151	Win=65024	Len=1448	TSval=2264214802	
245	14.131964872	10.0.0.20	163.10.0.72	TCP	66	56774 → 80	[ACK]	Seq=151	Ack=170865	Win=87168	Len=0	TSval=3921321486	TSe
246	14.213285144	163.10.0.72	10.0.0.20	TCP	1514	[TCP	Previous segment not captured]	80 → 56774	[ACK]	Seq=172313	Ack=151		
247	14.213323000	10.0.0.20	163.10.0.72	TCP	78	[TCP	Dup ACK 245#1]	56774 → 80	[ACK]	Seq=151	Ack=170865	Win=87168	Len=0
248	14.330865451	163.10.0.72	10.0.0.20	TCP	1514	80 → 56774	[ACK]	Seq=173761	Ack=151	Win=65024	Len=1448	TSval=2264215046	
249	14.330939783	10.0.0.20	163.10.0.72	TCP	78	[TCP	Dup ACK 245#2]	56774 → 80	[ACK]	Seq=151	Ack=170865	Win=87168	Len=0
250	14.452288582	163.10.0.72	10.0.0.20	TCP	1514	[TCP	Previous segment not captured]	80 → 56774	[ACK]	Seq=176657	Ack=151		
251	14.452326256	10.0.0.20	163.10.0.72	TCP	86	[TCP	Dup ACK 245#3]	56774 → 80	[ACK]	Seq=151	Ack=170865	Win=87168	Len=0
252	14.573121413	163.10.0.72	10.0.0.20	TCP	1514	80 → 56774	[ACK]	Seq=178105	Ack=151	Win=65024	Len=1448	TSval=2264215288	
253	14.573162769	10.0.0.20	163.10.0.72	TCP	86	[TCP	Dup ACK 245#4]	56774 → 80	[ACK]	Seq=151	Ack=170865	Win=87168	Len=0
254	14.694243494	163.10.0.72	10.0.0.20	TCP	1514	[TCP	Previous segment not captured]	80 → 56774	[ACK]	Seq=181001	Ack=151		
255	14.694303254	10.0.0.20	163.10.0.72	TCP	94	[TCP	Dup ACK 245#5]	56774 → 80	[ACK]	Seq=151	Ack=170865	Win=87168	Len=0
256	14.815293870	163.10.0.72	10.0.0.20	TCP	1514	80 → 56774	[ACK]	Seq=182449	Ack=151	Win=65024	Len=1448	TSval=2264215530	
257	14.815390464	10.0.0.20	163.10.0.72	TCP	94	[TCP	Dup ACK 245#6]	56774 → 80	[ACK]	Seq=151	Ack=170865	Win=87168	Len=0
258	14.936806095	163.10.0.72	10.0.0.20	TCP	1514	[TCP	Previous segment not captured]	80 → 56774	[ACK]	Seq=185345	Ack=151		
259	14.936883613	10.0.0.20	163.10.0.72	TCP	94	[TCP	Dup ACK 245#7]	56774 → 80	[ACK]	Seq=151	Ack=170865	Win=87168	Len=0
260	15.058012596	163.10.0.72	10.0.0.20	TCP	1514	80 → 56774	[ACK]	Seq=186793	Ack=151	Win=65024	Len=1448	TSval=2264215769	

Figura 4.8: Muestra 4

Lo dicho se puede ver en la muestra de la Figura 4.5 en los segmentos 30-32, en este caso con la variante en la cual el cliente aprovecha para enviar los bits ACK y FIN en el mismo segmento.

Pero, volvamos para atrás y analicemos qué pasa una vez establecida la conexión. Comienza el intercambio de datos en forma continua y un equipo, ya sea el cliente o el servidor, puede en un segmento además de enviar datos, confirmar datos previamente recibidos, o sea que hace un dos en uno.

Debido a las características de TCP, este protocolo utiliza diferentes métodos para detectar posibles errores que se pueden producir en la transferencia de datos:

- Checksum: similar a como lo hace UDP.
- Numeración de segmentos: para llevar correctamente el control en la recepción de los mismos. Cada segmento tendrá asociado un número de secuencia que indica su ubicación en el stream de bytes.
- Confirmaciones de tipo selectivas (SACK): además de permitir acumular segmentos para que con un único ACK se confirmen varios de ellos, TCP permite mediante los SACK hacer el pedido de un segmento determinado.
- Temporizadores: si pasa un cierto tiempo (RTO), automáticamente TCP retransmite el segmento que “aparentemente” no ha llegado al receptor.
- Se descartan los segmentos duplicados: en caso de que llegue un segmento duplicado (porque uno ha tardado más de lo normal y se ha vuelto a enviar) lo elimina.

En el caso que sea necesario retransmitir algún segmento el mecanismo a utilizar será transparente para la aplicación ya que TCP actúa en forma independiente y está preparado para solucionar el inconveniente.

En la Figura 4.8, perteneciente a la captura n11-client.eth0-tcp-fc-cc-3, podemos analizar el mecanismo que utiliza TCP en el caso de ser necesarias las retransmisiones.

Se puede ver que el cliente, debido a problemas ajenos a él, no recibe una cierta cantidad de segmentos, la captura es normal hasta el evento 245 que corresponde aun segmento en el cual el cliente con el ACK le comunica al servidor que el próximo byte a recibir es el 170865 y de esta manera además le confirma la llegada de los anteriores.

En el evento 246 tenemos el próximo segmento que arriba al cliente con número de secuencia 172313, este “analizando lo que recibió para deducir que debería haber recibido” detecta que no corresponde en

No.	Time	Source	Destination	Protocol	Length	Info
245	14.131964072	10.0.0.20	163.10.0.72	TCP	66	56774 → 80 [ACK] Seq=151 Ack=170865 Win=87168 Len=0 TSval=3921321486 TSecr=2264214802
246	14.213285144	163.10.0.72	10.0.0.20	TCP	1514	[TCP Previous segment not captured] 80 → 56774 [ACK] Seq=172313 Ack=151 Win=65024 Len=1448
247	14.213323000	10.0.0.20	163.10.0.72	TCP	78	[TCP Dup ACK 245#1] 56774 → 80 [ACK] Seq=151 Ack=170865 Win=87168 Len=0 TSval=3921321567

```

<
> Frame 247: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
> Ethernet II, Src: 00:00:00_aa:00:1b (00:00:00:aa:00:1b), Dst: 00:00:00_aa:00:1a (00:00:00:aa:00:1a)
> Internet Protocol Version 4, Src: 10.0.0.20, Dst: 163.10.0.72
v Transmission Control Protocol, Src Port: 56774, Dst Port: 80, Seq: 151, Ack: 170865, Len: 0
  Source Port: 56774
  Destination Port: 80
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 151 (relative sequence number)
  [Next sequence number: 151 (relative sequence number)]
  Acknowledgment number: 170865 (relative ack number)
  1011 .... = Header Length: 44 bytes (11)
  > Flags: 0x010 (ACK)
  Window size value: 681
  [Calculated window size: 87168]
  [Window size scaling factor: 128]
  Checksum: 0x7c29 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  > Options: (24 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, No-Operation (NOP), No-Operation (NOP), SACK
    > TCP Option - No-Operation (NOP)
    > TCP Option - No-Operation (NOP)
    > TCP Option - Timestamps: TSval 3921321567, TSecr 2264214802
    > TCP Option - No-Operation (NOP)
    > TCP Option - No-Operation (NOP)
    > TCP Option - SACK 172313-173761
  > [SEQ/ACK analysis]
  > [Timestamps]

```

Figura 4.9: Muestra 5

la secuencia de bytes, entonces en el evento 247, ver Figura 4.9, envía un ACK duplicado repitiendo el número de byte de confirmación o sea el próximo byte esperado por el cliente, se puede ver que el servidor sigue enviando otros segmentos, algunos de los cuales no llegan al cliente, mientras que éste sigue reclamando el segmento que corresponde a la secuencia por él controlada, o sea el 170865. Recién en el ACK duplicado nro. 39 el servidor le retransmite el segmento solicitado por el cliente (eventos 325-326). Es interesante ver que a pesar de que se necesitó 39 pedidos de retransmisión para resolver el inconveniente, lo que implicó un cierto tiempo perdido, el cliente fue recibiendo segmentos con secuencias posteriores al que solicitaba, los almacenó y con los mismos pudo deducir que otros no habían arribado. Posteriormente, el cliente, comienza a realizar sus ACK selectivos para solicitar las retransmisiones necesarias y así completar satisfactoriamente la secuencia de bytes enviados por el servidor.

Todo lo expuesto queda patentizado en el gráfico time sequence de Stevens donde los segmentos retransmitidos son los que están marcados en rojo y corresponden a los que se alejan de la secuencia que forman la recta que debería presentarse en caso de falta de retransmisiones, ver Figura 4.10.

De la misma forma que expresamos respecto a UDP, TCP debe ser usado para aquellas aplicaciones que necesiten sus características distintivas, si no está claro que no tiene sentido ya que el establecimiento de la conexión, el control y el reordenamiento de los segmentos consume recursos y tiempo y por lo tanto su uso debe estar plenamente justificado.

Como si fuera poco lo que hemos dicho sobre TCP, además realiza una función que hace a la buena comunicación de las partes. De la misma manera que nosotros no podemos procesar lo que nos dice una persona que habla continuamente sin parar ni para respirar, este protocolo controla el flujo de la información que podría llegar a rebasar al receptor, permitiendo que el cliente y el servidor vayan adaptando en todo momento la cantidad de información que pueden enviar y recibir, esto lo hace con el concepto de ventanas deslizantes a nivel de bytes y el uso de confirmaciones de llegada (ACK).

En la captura n11-client.eth0-tcp-fc-cc-7-full-window, de la cual se ha extraído la Figura 4.11, podemos ver el mecanismo de control de flujo que implementa TCP. Como se puede apreciar, en este

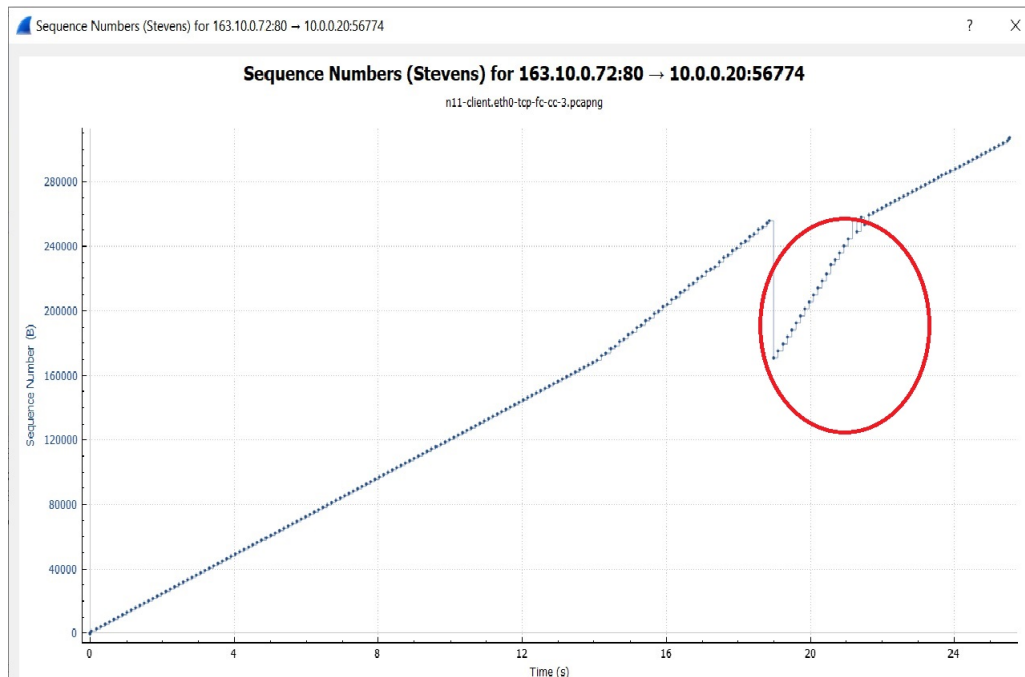


Figura 4.10: Gráfico time sequence (Stevens) 1

781	52.402111992	163.10.0.72	10.0.0.20	TCP	66	5001 → 49012 [ACK] Seq=1 Ack=611057 Win=4224 Len=0 TSval=1942813641 TSecr=22097115
782	52.435809549	163.10.0.72	10.0.0.20	TCP	66	5001 → 49012 [ACK] Seq=1 Ack=613953 Win=1408 Len=0 TSval=1942813675 TSecr=22097195
783	54.015743155	10.0.0.20	163.10.0.72	TCP	1474	[TCP Window Full] 49012 → 5001 [PSH, ACK] Seq=613953 Ack=1 Win=64256 Len=1408 TSval=22100785 TSecr=1942813675
784	54.016132342	163.10.0.72	10.0.0.20	TCP	66	[TCP ZeroWindow] 5001 → 49012 [ACK] Seq=1 Ack=615361 Win=0 Len=0 TSval=1942815255 TSecr=22100785
785	55.551703545	10.0.0.20	163.10.0.72	TCP	66	[TCP Keep-Alive] 49012 → 5001 [ACK] Seq=615360 Ack=1 Win=64256 Len=0 TSval=22100785 TSecr=1942815255
786	55.551897560	163.10.0.72	10.0.0.20	TCP	66	[TCP ZeroWindow] 5001 → 49012 [ACK] Seq=1 Ack=615361 Win=0 Len=0 TSval=1942816791 TSecr=22100785
787	57.343968554	fe80::200:ff:fe::ff02::2		ICMPv6	70	Router Solicitation from 00:00:00:aa:00:1b
788	58.623741227	10.0.0.20	163.10.0.72	TCP	66	[TCP Keep-Alive] 49012 → 5001 [ACK] Seq=615360 Ack=1 Win=64256 Len=0 TSval=22103857 TSecr=1942819863
789	58.624092674	163.10.0.72	10.0.0.20	TCP	66	[TCP ZeroWindow] 5001 → 49012 [ACK] Seq=1 Ack=615361 Win=0 Len=0 TSval=1942819863 TSecr=22103857
790	63.966366667	163.10.0.72	10.0.0.20	TCP	66	[TCP Window Update] 5001 → 49012 [ACK] Seq=1 Ack=615361 Win=90112 Len=0 TSval=1942825205 TSecr=22109199
791	63.966399753	10.0.0.20	163.10.0.72	TCP	1514	49012 → 5001 [ACK] Seq=615361 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
792	63.966401305	10.0.0.20	163.10.0.72	TCP	1514	49012 → 5001 [ACK] Seq=616809 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205

Figura 4.11: Muestra 6

caso, el flujo de datos se dirige desde el cliente al servidor ya que estamos usando iPerf, una herramienta que permite medir el máximo ancho de banda disponible en redes IP.

El servidor (163.10.0.72) desde el comienzo de la captura va variando su ventana de recepción (generalmente aumentándola) en forma dinámica mientras que el cliente (10.0.0.20) la mantiene, en toda la muestra, relativamente estable en su valor de inicio especificado en el triple handshake.

En determinado momento el servidor comienza a bajar el valor de su ventana de recepción quizás porque no puede procesar los segmentos en relación con la velocidad de llegada de los mismos, esto podría ser un problema si no logra incrementarla rápidamente.

En la Figura 4.9 puede verse que en el evento 782 el servidor le informa al cliente que su ventana de recepción es de $W=1408$ o sea que va a poder recibir solamente esa cantidad de bytes ya que su buffer está prácticamente lleno.

El cliente manda su próximo segmento, evento 783, con longitud exactamente igual a la cantidad de bytes de la ventana de recepción del servidor y con le avisa que le completó su ventana de recepción (window full), lo que significa que momentáneamente el servidor no podrá recibir más bytes, por lo que en sus segmentos TCP de ahora en más y hasta que pueda solucionar el inconveniente, especificará su ventana de recepción con $W=0$ (zero window) (eventos 784,786 y 789 de la Figura 4.9.), esto cancela el envío de datos desde el cliente que se limita a enviar segmentos "keep alive" para mantener la comunicación activa.

En el evento 790, el servidor informa que ya a liberado su ventana de recepción (buffer) mediante un anuncio de actualización de la misma con $W=90112$, saliendo de esta manera de su estado anterior o sea que ya está listo nuevamente para recibir datos.

Seguidamente, se puede ver en la captura que el servidor incrementa constantemente su ventana de recepción y su comportamiento se puede visualizar en el diagrama window scaling, ver Figura 4.12.

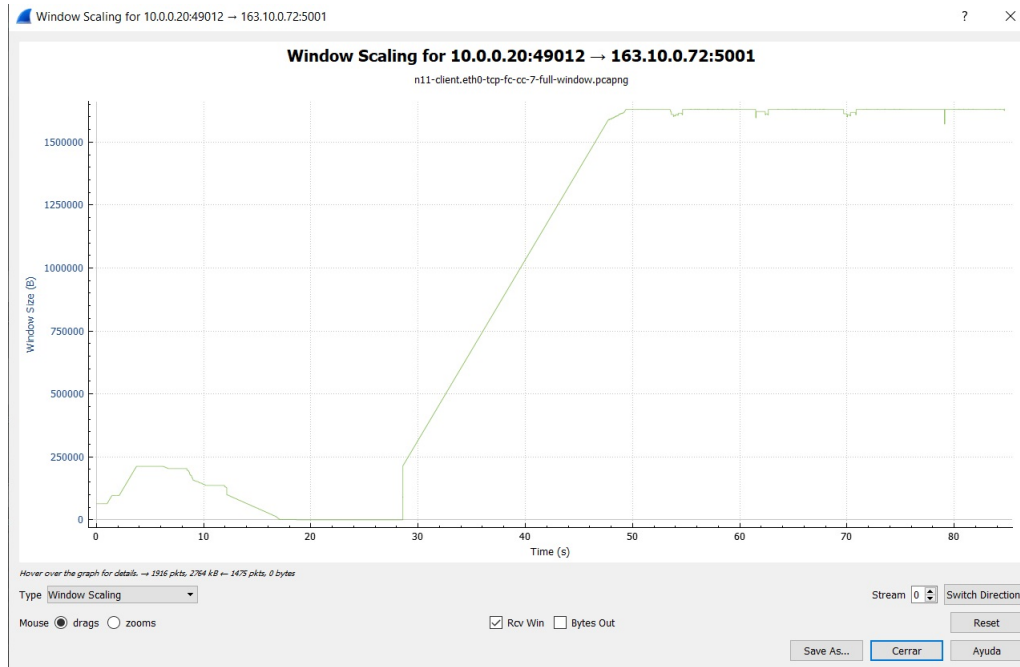


Figura 4.12: Gráfico variación ventana de recepción

En el gráfico time sequence de Stevens, Figura 4.13, podemos ver el comportamiento del cliente que en determinado momento tuvo que dejar de transmitir ya que el servidor le comunicó que su buffer de recepción estaba lleno, se indica con rojo el período de tiempo en que sucedió dicho acontecimiento. Además, se pueden ver algunos puntos que se desprenden de la continuidad de la curva que corresponden a segmentos retransmitidos por el cliente ya sea por que se venció su RTO o bien debido a los SACK enviados por el servidor.

Siguiendo con el análisis de la captura citada anteriormente y luego del mecanismo de control de flujo visto, desde el evento 791 y hasta el 800, ver Figura 4.14, el cliente envía diez segmentos de 1448 bytes cada uno y en forma continua, ésto podría deberse a una versión particular del mecanismo de control de congestión “slow start” que sirve para detectar el máximo volumen de datos que se podría enviar a la red sin correr el peligro de experimentar pérdidas; ese volumen de datos permitido, como es lógico, es variable ya que depende de las condiciones de la red en cada momento.

Ahora, si bien el cliente envía esa secuencia de diez segmentos, se puede ver que el servidor los confirma de a uno y a su tiempo, ésto se puede apreciar en el evento 801 dónde el servidor le comunica al cliente que le confirma solamente el primero de los diez enviados y le solicita que le envíe el segundo de esa secuencia y así sucesivamente (por ej. en el evento 826 tenemos el ACK del segmento del evento 799, en el 916 el ACK de segmento del 849, en el 1031 el ACK del segmento del 924 ...etc).

Éste corrimiento a la larga puede traer problemas ya que el servidor retransmitirá el segmento cuando se venza su RTO (time out de retransmisión), o sea que en ese tiempo no le haya llegado el

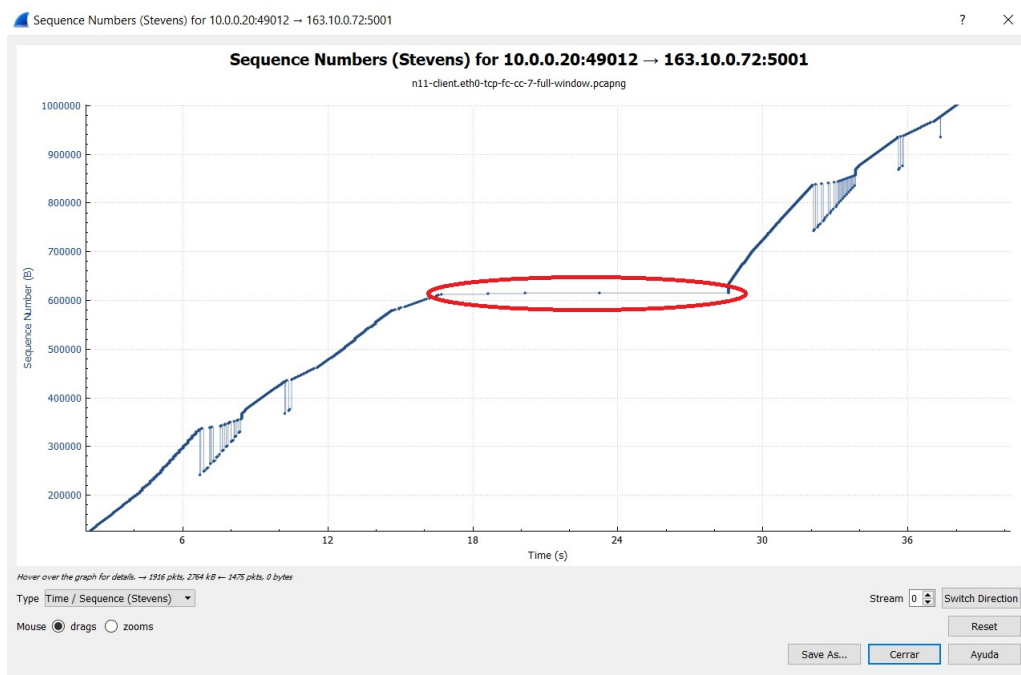


Figura 4.13: Gráfico time sequence (Stevens) 2

Time	Seq	Len	Win	Flags	Details
789	58.624002674	163.10.0.72	10.0.0.20	TCP	66 [TCP ZeroWindow] 5001 → 49012 [ACK] Seq=1 Ack=615361 Win=0 Len=0 TSval=1942819863 TSec=
790	63.966366667	163.10.0.72	10.0.0.20	TCP	66 [TCP Window Update] 5001 → 49012 [ACK] Seq=1 Ack=615361 Win=90112 Len=0 TSval=1942825205
791	63.966399753	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=615361 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
792	63.966401305	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=616809 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
793	63.966412735	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=618257 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
794	63.966413817	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=619705 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
795	63.966421606	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=621153 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
796	63.966422620	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=622601 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
797	63.966430128	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=624049 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
798	63.966431130	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=625497 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
799	63.966438015	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=626945 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
800	63.966438981	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=628393 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
801	63.966566647	163.10.0.72	10.0.0.20	TCP	66 5001 → 49012 [ACK] Seq=1 Ack=616809 Win=88704 Len=0 TSval=1942825205 TSecr=22109199
802	63.966586739	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=629841 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205
803	63.966587793	163.10.0.72	10.0.0.20	TCP	1514 49012 → 5001 [ACK] Seq=631289 Ack=1 Win=64256 Len=1448 TSval=22109199 TSecr=1942825205

Figura 4.14: Muestra 7

ACK correspondiente a un segmento enviado porque se perdió en el camino o porque el mismo haya llegado a destiempo. Cabe recordar que el RTO no es fijo, es un tiempo que se está continuamente recalculando y tiene en cuenta, entre otras cosas, el RTT (round trip time) o sea el tiempo que transcurre desde que un segmento sale del emisor hasta que éste recibe la confirmación de que ha sido recibido por el receptor (en realidad se tomará un promedio de varios RTT). Ejemplos de retransmisiones por RTO se pueden ver en los eventos 1037 y 1039 de la captura y que toman como RTO base el calculado con el segmento del evento 1035.

Cabe acotar que el mecanismo de envío de varios segmentos consecutivos se repite varias veces en la captura, siendo comunes los envíos de 2, 4 y 6 de ellos.

Para terminar el análisis, presentamos los Round Trip Time (RTT) de todos los segmentos de la captura, ver Figura 4.15, donde se puede observar un incremento al comienzo, un posterior decremento cuando el servidor va agotando su ventana de recepción hasta que se llena ($W=0$) y un posterior incremento cuando ya pudo desagotar su buffer de recepción.

Del gráfico se desprende que si no tuviéramos en cuenta lo ocurrido con la ventana de recepción del servidor, podríamos considerar un valor promedio de RTT del orden de los 1.500 milisegundos.

Existen varias implementaciones de TCP y se podría decir que fue evolucionando con el tiempo, todas estas versiones se enfocaron casi exclusivamente a resolver el problema del control de congestión

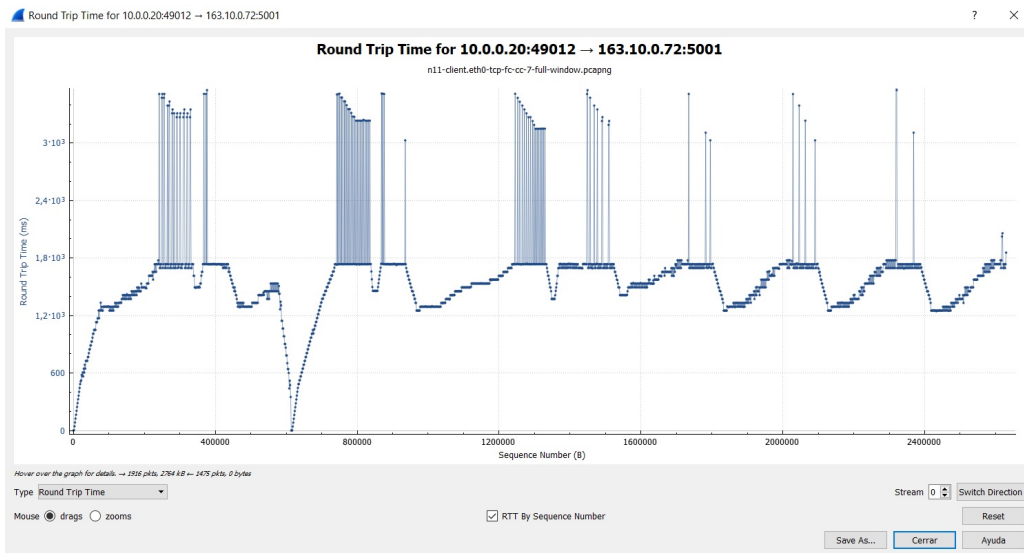


Figura 4.15: Gráfico de Round Trip Time

que se puede presentar en Internet debido a distintas causas (caídas de enlaces, routers bajo de recursos, problemas de ruteo ...etc).

El desafío es y será tratar de detectar la congestión lo antes posible y después actuar en consecuencia, siendo la mejor solución el modelo end-to-end o el basado en la red que menos impacte en ella y el que permita aprovechar los recursos para que la transferencia de segmentos prácticamente se degrade lo menos posible. Los algoritmos de control de congestión tratan de determinar dinámicamente el ancho de banda disponible y la latencia de la red, luego en función de su análisis, se modifica la tasa de envío del emisor para evitar el colapso de la red.

Si bien en principio podría confundirse los conceptos de control de flujo y el de control de la congestión, hay que tener en cuenta que el primero tiene que ver con un mecanismo propio de los extremos (cliente-servidor), mientras que la congestión es un problema que experimenta la red en un determinado momento. Ambos controles son importantes a la hora de analizar las estrategias de implementación, pero está claro que la congestión tendrá un impacto mayor ya que podría hacer colapsar una parte o toda la red.

Algunas aplicaciones que usan TCP como protocolo de transporte son: HTTP, telnet, FTP ...etc.

Para más información sobre el protocolo de transporte TCP ver [Pos81].

REFERENCIAS

[Pos80] Jon Postel. «Rfc-768: User datagram protocol (udp)», 1980.

[Pos81] Jon Postel. «Rfc-793: Transmission control protocol (tcp)», 1981.

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 el 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 inicialmente 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

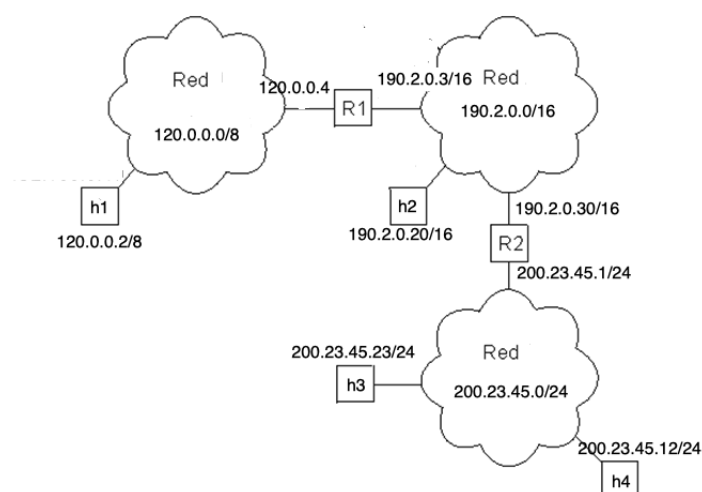


Figura 5.2: Red IP

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

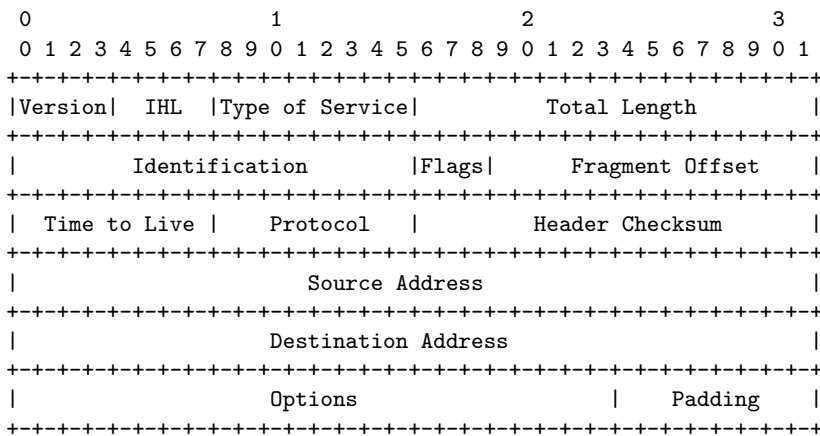
esa interconexión que es el Router, R1 y R2 en la figura, que en [Pos81a] lo denomina Gateway.

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 identificar 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 2021¹. 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 “subnetada”, 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:

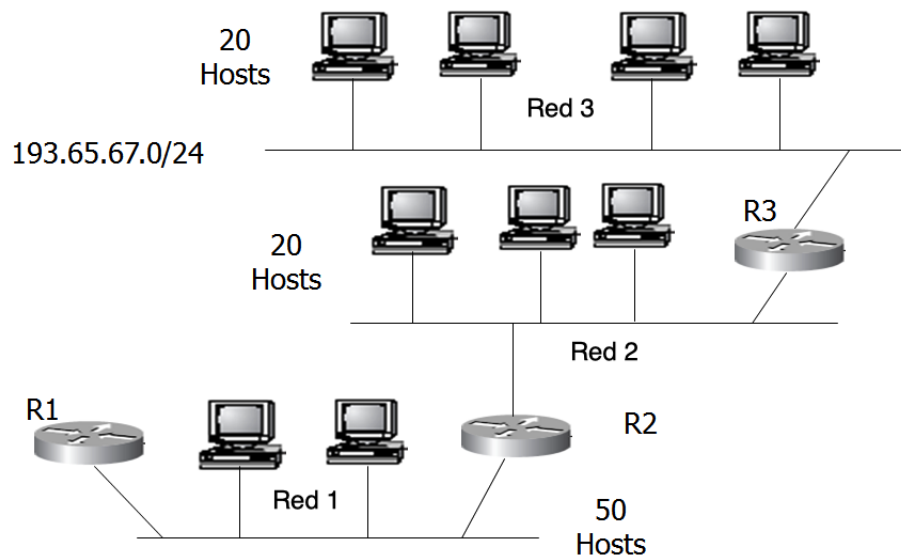
Notación Moderna	
Prefijo:	/27
Red:	200.23.45.64/27
Broadcast:	200.23.45.95/27

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 de la 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, “Classless Inter-Domain Routing”. 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:

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|           Identification           |Flags|      Fragment Offset      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

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 fragmento 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

<pre> ▶ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) ▶ Ethernet II, Src: 00:01:02:03:04:05, Dst: 00:05:04:03:02:01 ▼ Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.2 0100 = Version: 4 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 84 Identification: 0x0f4a (3914) ▼ Flags: 0x20, More fragments 0... = Reserved bit: Not set .0.. = Don't fragment: Not set ..1. = More fragments: Set Fragment Offset: 0 Time to Live: 64 Protocol: Trunk-1 (23) Header Checksum: 0xc7f5 [correct] [Header checksum status: Good] [Calculated Checksum: 0xc7f5] Source Address: 192.168.1.1 Destination Address: 192.168.1.2 ▶ Data (64 bytes) </pre>	<pre> ▶ Frame 2: 57 bytes on wire (456 bits), 57 bytes captured (456 bits) ▶ Ethernet II, Src: 00:01:02:03:04:05, Dst: 00:05:04:03:02:01 ▼ Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.2 0100 = Version: 4 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 43 Identification: 0x0f4a (3914) ▼ Flags: 0x00 0... = Reserved bit: Not set .0.. = Don't fragment: Not set ..0. = More fragments: Not set Fragment Offset: 64 Time to Live: 64 Protocol: Trunk-1 (23) Header Checksum: 0xe816 [correct] [Header checksum status: Good] [Calculated Checksum: 0xe816] Source Address: 192.168.1.1 Destination Address: 192.168.1.2 ▶ Data (23 bytes) </pre>
(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 unidad 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.

No.	Time	Source	Destination	Length	Info
1	12:04:32	10.0.0.20	163.10.0.72	98	Echo (ping) request id=0x0096
2	12:04:32	163.10.0.72	10.0.0.20	98	Echo (ping) reply id=0x0096
3	12:04:33	10.0.0.20	163.10.0.72	98	Echo (ping) request id=0x0096
4	12:04:33	163.10.0.72	10.0.0.20	98	Echo (ping) reply id=0x0096
5	12:04:34	10.0.0.20	163.10.0.72	98	Echo (ping) request id=0x0096


```

▶ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface
▶ Ethernet II, Src: 00:00:00:aa:00:1b, Dst: 00:00:00:aa:00:1a
▼ Internet Protocol Version 4, Src: 10.0.0.20, Dst: 163.10.0.72
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x8201 (33281)
  ▶ Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x0b42 [correct]
    [Header checksum status: Good]
    [Calculated Checksum: 0x0b42]
    Source Address: 10.0.0.20
    Destination Address: 163.10.0.72
▼ Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xbcf9 [correct]
  [Checksum Status: Good]
  Identifier (BE): 150 (0x0096)
  Identifier (LE): 38400 (0x9600)
  Sequence Number (BE): 1 (0x0001)
  Sequence Number (LE): 256 (0x0100)
  [Response frame: 2]
  Timestamp from icmp data: Apr 14, 2021 09:04:32.000000000 -03
  [Timestamp from icmp data (relative): 0.090578575 seconds]
▶ Data (48 bytes)

```

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 protocolo 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 mensajes 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:


```

▼ Internet Control Message Protocol
  Type: 0 (Echo (ping) reply)
  Code: 0
  Checksum: 0xc4f9 [correct]
  [Checksum Status: Good]
  Identifier (BE): 150 (0x0096)
  Identifier (LE): 38400 (0x9600)
  Sequence Number (BE): 1 (0x0001)
  Sequence Number (LE): 256 (0x0100)
  [Request frame: 1]
  [Response time: 0.387 ms]
  Timestamp from icmp data: Apr 14, 2021 09:04:32.000000000 -03
  [Timestamp from icmp data (relative): 0.090965746 seconds]
  ▶ Data (48 bytes)

```

Figura 5.6: Ping-2

No.	Time	Source	Destination	Length	Info
15	12:04:53	10.0.0.20	163.10.0.72	74	Payload Unk: 4
16	12:04:53	10.0.0.1	10.0.0.20	102	Time-to-live exceeded (Time to
17	12:04:53	10.0.0.20	163.10.0.72	74	Payload Unk: 4
18	12:04:53	10.0.0.1	10.0.0.20	102	Time-to-live exceeded (Time to
19	12:04:53	10.0.0.20	163.10.0.72	74	Payload Unk: 4

```

▶ Frame 15: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface
▶ Ethernet II, Src: 00:00:00:aa:00:1b, Dst: 00:00:00:aa:00:1a
▼ Internet Protocol Version 4, Src: 10.0.0.20, Dst: 163.10.0.72
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x23b5 (9141)
  ▶ Flags: 0x00
    Fragment Offset: 0
  ▶ Time to Live: 1
    Protocol: UDP (17)
    Header Checksum: 0xe896 [correct]
    [Header checksum status: Good]
    [Calculated Checksum: 0xe896]
    Source Address: 10.0.0.20
    Destination Address: 163.10.0.72
  ▶ User Datagram Protocol, Src Port: 60378, Dst Port: 33434

```

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 mismo. 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 tráfico cuyas dificultades queremos diagnosticar. También recordemos que es necesario que ICMP esté

```

▶ Frame 16: 102 bytes on wire (816 bits), 102 bytes captured (816 bits)
▶ Ethernet II, Src: 00:00:00:aa:00:1a, Dst: 00:00:00:aa:00:1b
▶ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.20
▼ Internet Control Message Protocol
  Type: 11 (Time-to-live exceeded)
  Code: 0 (Time to live exceeded in transit)
  Checksum: 0xa29f [correct]
  [Checksum Status: Good]
  Unused: 00000000
▼ Internet Protocol Version 4, Src: 10.0.0.20, Dst: 163.10.0.72
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 60
  Identification: 0x23b5 (9141)
  ▶ Flags: 0x00
  Fragment Offset: 0
  ▶ Time to Live: 1
  Protocol: UDP (17)
  Header Checksum: 0xe896 [correct]
  [Header checksum status: Good]
  [Calculated Checksum: 0xe896]
  Source Address: 10.0.0.20
  Destination Address: 163.10.0.72
▶ User Datagram Protocol, Src Port: 60378, Dst Port: 33434

```

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 protocolo 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 un 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


```

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:feeb:9792,
      Dst: fe80::5054:ff:fe12:3461
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... = Traffic Class: 0x00
    ... .. 0000 0000 0000 0000 = Flow Label: 0x000000
Payload Length: 64
Next Header: ICMPv6 (58)
Hop Limit: 64
Source Address: fe80::5054:21ff:feeb: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 por 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.

CAPÍTULO 6

CONSTRUCTORES DE CAMINOS (RUTEO)

ANDRES BARBIERI Y MATÍAS ROBLES

En este capítulo se profundizará en el concepto de ruteo IP y se verán algunos detalles sobre la arquitectura de la red de redes: Internet. Se intentará dar un panorama de la infraestructura y topología de la red que hará posible que el requerimiento HTTP, generado en los primeros capítulos, pueda llegar a destino. Se verá cómo trabajan las componentes intermedias llamadas routers, haciendo posible que encuentren el camino sobre la red, indicando donde intervienen los protocolos de ruteo. No se verán los detalles del funcionamiento específico de cada uno ni se explicarán cuestiones como el formato específico de los mensajes que utilizan. Para profundizar sobre el tema se puede consultar la bibliografía [Hal00] y [KR12].

Introducción

De acuerdo a lo visto en el capítulo sobre IP (Internet Protocol), este es el protocolo de red (L3) que hace posible que los mensajes encapsulados en paquetes, o más específicamente datagramas, lleguen a su destino. La red esta formada por un gran número de equipos intermedios que trabajan a diferentes niveles o capas¹ de acuerdo al modelo OSI. A nivel de red (L3) son los routers quienes saben hablar IP y lo implementan como servicio de entrega de datos, dando forma así a lo que se conoce como la **red IP**. La red IP recibirá la información a transmitir y hará su mejor esfuerzo por entregarla correctamente al destino. La clave de esta transmisión está en el despacho o encaminado, en inglés *switching* o *forwarding*, de los datagramas. Este encaminado es realizado por los routers, quienes están interconectados para poder pasar los datagramas de unos a otros, y así hacer llegar los datos hasta quien tenga la dirección IP destino. La forma de conexión suele hacerse por diferentes enlaces y ser de variadas tecnologías, a menudo cambia dinámicamente, por esto se puede decir que un datagrama suele tener caminos alternativos hasta el destino, además dos datagramas del mismo flujo podrían no seguir el mismo. Los routers están encargados de la selección del camino, la cual se hace en cada nodo (salto a salto) de la red que corra IP, en base a su tabla de ruteo, donde se indica los destinos y sus próximos saltos (next-hop) y/o interfaces de salida (output interface). Cada router elige el siguiente paso.

Como protocolo común en todos los nodos (hosts y routers) esta el protocolo **ENRUTADO (Routed) IP** (sea IPv4, IPv6 o ambos), el cual, a menudo, requiere los servicios del protocolo de **ENRUTAMIENTO/RUTEO (Routing)** para construir las tablas de ruteo. Se pueden distinguir dos tareas (ver figura 6.1):

¹Mayormente: L1 (físico), L2 (enlace) y L3 (de red), donde el/los protocolo/s L3 son independientes de L1 y L2. "L" proviene de Layer (capa en inglés)

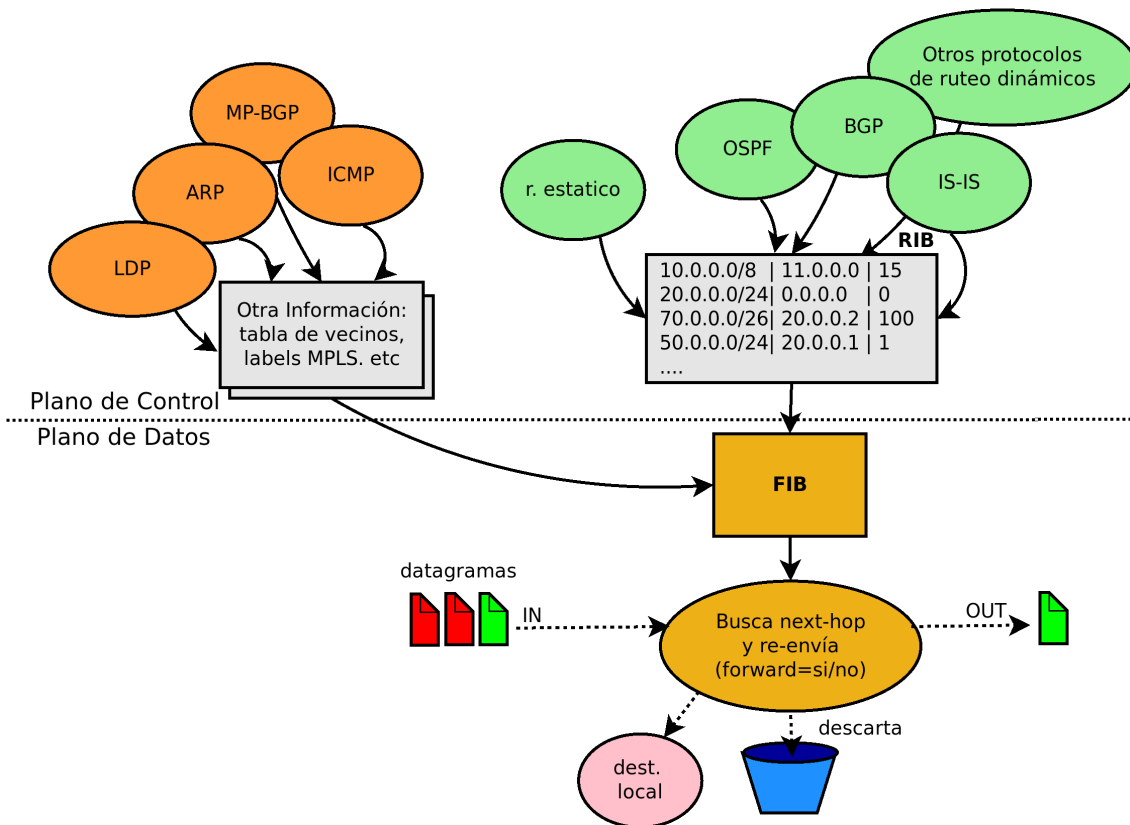


Figura 6.1: Tareas realizadas por los routers en sus respectivos planos.

El despacho/encaminamiento de paquetes (forwarding/switching): que selecciona un port de salida en función de la dirección IP destino del datagrama y la tabla de ruteo. Es usado por el protocolo enrutado y corresponde al **plano de datos**. Todos los nodos, hosts y routers, que corren IP pueden hacerlo. Para el caso de los hosts, o nodos finales, estos no hacen *forwarding*, es decir, no re-envían un mensaje que les llega y no es para ellos. Solo despachan mensajes que tienen como origen a ellos mismos. El datagrama IP tiene como dirección IP origen alguna de las direcciones que tienen asignadas.

Armado de tabla de Ruteo (Routing): proceso a través del cual se construye la tabla de ruteo, conocida como **RIB (Routing Information Base)**, mediante los protocolos de enrutamiento, de forma manual o usando las herramientas que se dispongan. Esta tarea corresponde al **plano de control**. Solo algunos routers lo hacen, no todos los nodos de la red. Los hosts podrían participar en el proceso de forma pasiva.

Los routers, como equipos intermedios, a menudo construyen una tabla para optimizar el proceso de despacho conocida como **FIB (Forwarding Information Base)** que se hace a partir de la RIB y se optimiza generando una estructura de datos más eficiente, por ejemplo, implementada mediante una TCAM en *multilayer-switching*. La generación de la tabla de ruteo RIB es la clave en este capítulo. Existen dos formas o criterios de genera la tabla de ruteo:

- De forma estática.
- De forma dinámica.

Cada una tiene sus ventajas y desventajas y se aplicarán de acuerdo al contexto. La primera, la forma estática, indica que el encargado de administrar el router configurará de manera manual a través de una interfaz, comúnmente de línea de comando, esta tabla. Por ejemplo en la plataforma GNU/Linux mediante los comandos `route(8)` o `ip(8)`. A continuación se muestran comandos que generan el mismo resultado para agregar una ruta por omisión o default (default gateway).

```
# ip route add default via 10.0.0.1
# route add default gw 10.0.0.1
# ip route add 0.0.0.0/0 via 10.0.0.1
```

Las características de lo que se conoce como ruteo estático son:

- Las rutas son establecidas manualmente por el administrador.
- Propenso a errores debido a que se cargan a mano.
- Si se cambia la topología (agregan o quitan enlaces) se requieren cambios manuales en los routers.
- Sirve cuando se tiene una red sencilla.
- No tiene problemas de seguridad ni de incompatibilidad.
- No implica costo de procesamiento extra (ni cómputo, ni memoria, ni uso de red).
- Mayor control
- Esquema NO escalable y NO tolerante a fallos.

Por otro lado, está la manera dinámica de generar la tabla de ruteo, ruteo dinámico. En este caso, los routers que conforman la red intercambiarán información y calcularán la tabla de ruteo en base a la información que hayan obtenido. Es un proceso distribuido donde no todos participan de forma activa, solo algunos routers. Las características del ruteo dinámico son:

- Requiere una configuración inicial por parte del administrador.
- Una vez dada la configuración inicial, los routers intercambiarán datos.
- Si se cambia la topología se adapta de forma automática.
- Facilita el mantenimiento cuando se tiene una red compleja.
- Implica costos extras de procesamiento (se requiere más cómputo, memoria y uso de red).
- Esquema escalable y tolerante a fallos.
- Resolución de problemas/debugging suele ser más complejo.
- Se generan caminos “óptimos” de acuerdo a la información manejada por el protocolo (métrica, costo).

Como un nuevo enfoque al ruteo dinámico surge SDN (Software Defined Networks), técnica, o de forma más genérica concepto, que permite controlar de forma centralizada los diferentes routers y componentes de la red pudiendo adaptar de manera más adecuada, o ad-hoc, la red al tráfico. SDN esta siendo desplegado masivamente en grandes Centros de Datos, pero en redes de core (núcleo) y distribución aún su uso es muy incipiente. En este texto solo se hará esta referencia al tema. Para más información consultar [NG13].

Historia

La Internet en su evolución, luego de migrar a TCP/IP siguiendo la idea de red de conmutación de paquetes (a principios de los 80s), seguía siendo una red simple, sin divisiones organizativas, en la cual los routers, en esa época llamados gateways, compartían todos con todos la información de enrutamiento utilizando el protocolo GGP (Gateway to Gateway Protocol)[RHS82]. GGP era un protocolo sencillo de enrutamiento que calculaba en forma distribuida la mejor ruta basado en la mínima cantidad de saltos. Debido al crecimiento de la red, la red GGP comenzó a convertirse en un sistema muy difícil de manejar y poco escalable. El primer paso hacia el cambio fue dividir la red común en diferentes sistemas interconectados, los que se llamaron (y aún se llaman) Sistemas Autónomos, notados AS (Autonomous Systems). GGP siguió ejecutando un tiempo en el núcleo (backbone) de Internet luego de la división en AS, pero al poco tiempo (principios de los 80s) fue reemplazado por un protocolo nuevo: EGP (Exterior Gateway Protocol)[Ros82][Mil84]. Al mismo tiempo surgían protocolos de enrutamiento dinámico para ejecutar dentro de cada sistema autónomo como fue en sus principios RIPv1[Hed88] para 1988 y OSPFv1[Moy89] para 1989. EGP, que corría directamente sobre IP, pretendía cubrir las carencias de GGP y fue pensado para ejecutar en una red dividida en AS. EGP comenzó a tener problemas cuando la topología de Internet empezó a cambiar al agregarse nuevos enlace pasando de forma de árbol a una de tipo mesh. A raíz de esto, el ruteo comenzó a no tomar las rutas óptimas y a generar routing loops o blackholes. Otro problema fue que la cantidad de rutas que se debían intercambiar crecieron y, en consecuencia, desbordaron las MTU (Maximum Transfer Unit) utilizadas. Esto último llevó a pensar en un protocolo que soportase de forma adecuada la fragmentación sin pérdidas. Debido a sus carencias y problemas, EGP fue reemplazado por BGP (Border Gateway Protocol). BGP, como sucesor de EGP, es un protocolo para intercambiar rutas entre diferentes sistemas autónomos. Fue definida la versión 1 de BGP en el estándar RFC-1105[LR89] en 1989. La versión de BGP 4 especificada en RFC-4271[RLH06] es la actual.

Protocolos de ruteo dinámico

La idea de dividir la red en sistemas autónomos parte en poder simplificar la operación de la misma y desacoplar políticas y decisiones de ruteo de un dominio a otro y, de esta forma, es posible utilizar protocolos de ruteo independientes en cada AS.

Sistema Autónomo (Autonomous System, AS): Es un conjunto de redes bajo la misma administración (podría ser gestionada por más de un operador de red) que utiliza un protocolo de ruteo o combinaciones de estos para rutear internamente (IGPs), independientemente de la red de su proveedor y de sus pares. Define una política de ruteo común (métricas, QoS, IGPs, etc). Cada AS

en Internet para poder conectarse a otras redes va a tener la necesidad de intercambiar tráfico con otros AS. Para esto, previamente deberán intercambiar rutas (redes). Formalmente debe tener un número identificador: ASN (AS Number) relacionado con el protocolo BGP, otorgado por los RIRs: LACNIC, RIPE, ARIN, AFRINIC o ARIN y asignados por el IANA a estos. Inicialmente, el ASN era un valor de 16 bits y, como sucede con los bloques IP, existe un rango privado. Con el crecimiento de Internet y el surgimiento de nuevas redes debió expandirse a 32 bits. Los sistemas autónomos pertenecen a diferentes organizaciones como ISPs (Internet Service Providers), proveedores de contenido, educación, gobierno, ONGs, etc.

En la actualidad, a la Internet se la puede describir como un gran conjunto de sistemas autónomos que intercambian rutas con otros AS mediante BGP (plano de control) y corren, como protocolo común, IP (plano de datos). Cada sistema autónomo es conformado por una o más redes y estos se interconectan para formar este gran conjunto. La red, o las redes del sistema autónomo, estarán formadas por una serie de routers que ejecutarán e implementarán la política de ruteo mediante la combinación de ruteo estático y/o ruteo dinámico.

Los protocolos de ruteo dinámicos se suelen clasificar en diferentes tipos de acuerdo a donde se utilizan (ver figura 6.2):

IGP (Interior Gateway Protocols): protocolos interiores de ruteo que trabajan dentro del mismo AS.

Ejemplos de este tipo son: RIP, OSPF, IS-IS, EIGRP.

EGP (Exterior Gateway Protocols): protocolos exteriores de ruteo que trabajan entre diferentes AS.

Ejemplos de este son: EGP (ya en desuso) y BGP.

Otra clasificación que suele hacerse es acorde a cómo trabajan:

Protocolos de Vector de Distancia (DV Distance Vector): corren un algoritmo distribuido conocido como Bellman-Ford. En general, no son óptimos y son poco escalables. Propensos a bucles (loops). Ejemplos: RIPv1 y v2, IGRP, GGP.

Protocolos de Vector de Camino (PV Path Vector): corren un algoritmo distribuido con atributos que los hacen que no sean propensos a bucles. Ejemplos BGP, EGP.

Protocolos de Estado de Enlace (LS Link State): corren un algoritmo mediante el cual distribuyen la información y luego, usando la técnica de Dijkstra o similar, calculan el camino a cada destino. Ejemplos de este grupo son OSPF e IS-IS.

Vector de Distancia Avanzado (Advanced VD): Un ejemplo de este tipo es EIGRP, considerado a veces como híbrido.

Estructura de Internet

Hoy en día no se podría indicar que Internet tiene una estructura y arquitectura bien definida. Su forma es consecuencia de la evolución misma del mundo y el uso de la tecnología. La verdad es que su crecimiento en recursos, usos y usuarios ha sido exponencial. A grandes rasgos se puede tener una idea de la infraestructura que hoy soporta la gran cantidad de servicios que la usan. La red estará

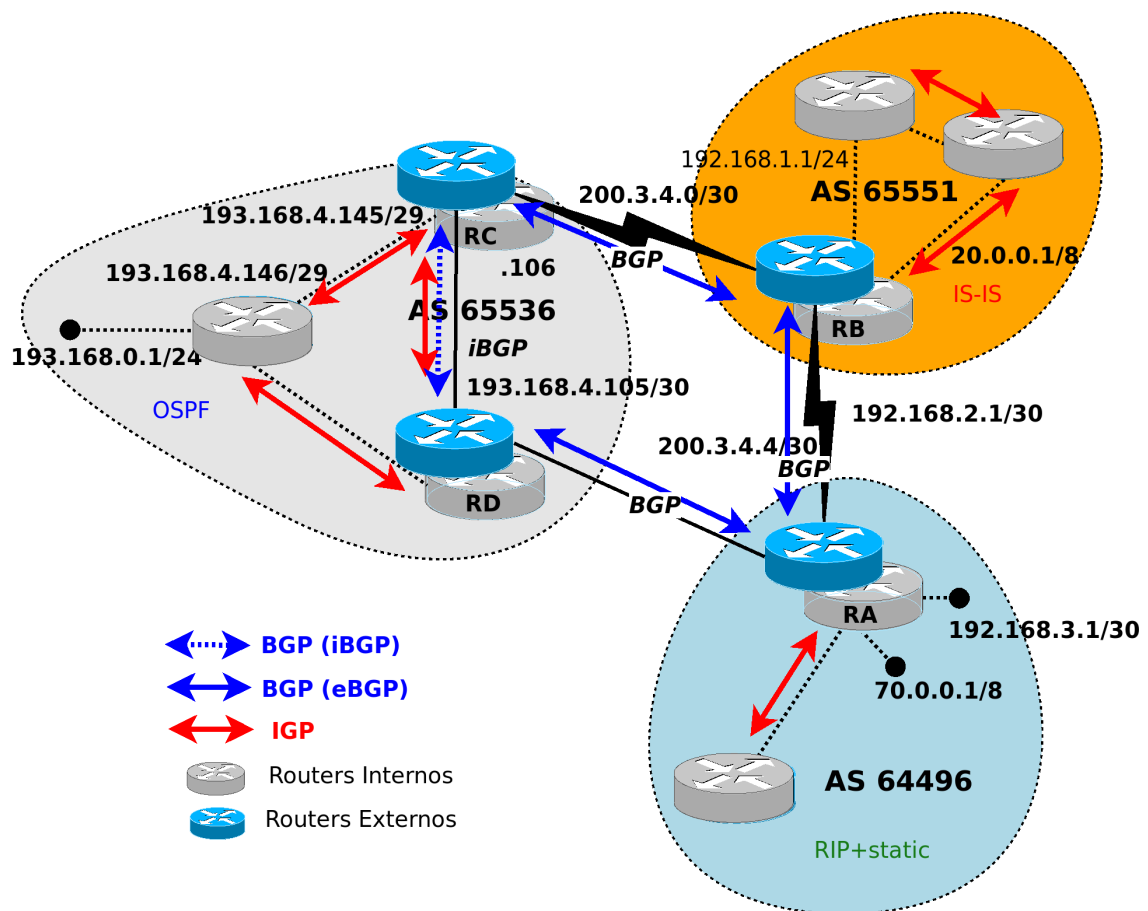


Figura 6.2: Ruteo IGP vs EGP.

formada por nodos intermedios, routers y switches, de diferentes tecnologías en L2, pero con la misma en L3, IP. El equipamiento estará agrupado en AS y como unidad de estudio se usará esta medida de granularidad más gruesa. Como parte de red también se pueden incluir los nodos llamados finales, en la terminología de IP: hosts. Los hosts serán los responsables de dar una interfaz de conexión, por ejemplo con los usuarios, y proveer los contenidos. Los hosts pertenecerán o estarán conectados a un AS. Esta separación entre nodos finales, los hosts, e intermedios, los routers, se basa en el principio de end-to-end que aparece como base arquitectónica de la red en gran parte de su desarrollo[Car96][BM02][KE04], aunque a veces es cuestionado. Este principio indica que el mantenimiento de estados y tareas más complejas se deleguen a los extremos dejando tareas más sencillas e intensas en el medio. En general, se puede clasificar a la unidad de estudio de este capítulo, los AS, de acuerdo a su conectividad y el tipo de servicio que brindan. Una clasificación posible de AS es (ver figura de ejemplo 6.3):

De Tránsito: llevan tráfico de redes de un AS a redes de otros AS y pueden aplicar políticas de ruteo al tráfico. Los AS/redes de tránsito “forwardean” en forma selectiva. Proveen tránsito a Internet a redes de otras organizaciones. Son los proveedores de proveedores (ISPs de ISPs), mayoristas de Internet, y también pueden ofrecer servicio a usuarios finales, como otras organizaciones.

No Tránsito: no llevan, transportan tráfico de otros AS. Se pueden sub-clasificar en:

Stub / single-homed AS: son AS que solo tiene un camino o salida de conexión con el resto de las redes.

Dual-homed AS: tiene más de una conexión hacia “afuera”, más de una salida, pero solo contra un mismo AS.

Multi-homed Non-Transit AS: conectados a múltiples AS. Tienen varias salidas al resto de las redes. No rutean tráfico de terceros.

Otra clasificación que se puede encontrar es:

AS de acceso: (Access AS) son AS que brindan/venden el acceso a otras redes (AS), e.g. acceso a Internet. Pueden ser Stub o Multi-homed. ISPs más chicos. Servicio a usuarios finales (end-users/home-users). Proveen la red de acceso, llamada habitualmente última milla y el acceso a Internet.

AS de acceso/transito: (Transit AS) son AS que brindan/venden servicio a otras redes y además son Transito. La mayoría de los AS de Transito también dan acceso. Venden a usuarios finales y a intermediarios (transit, peering) como otros ISPs. Llamados Transit & Access Providers.

AS de contenidos: (Content AS) proveen hosting de contenido y distribución. El ejemplo más común son las CDNs (Content Delivery Networks). Muchos AS finales también se pueden clasificar en esta categoría.

AS finales: (End AS) pueden ser Empresas, Organizaciones Educativas/Investigación como Universidades, o ONGs. Acceden a la red, no brindan/venden acceso fuera de la organización.

Los AS pueden tener diferentes formas de interconectarse:

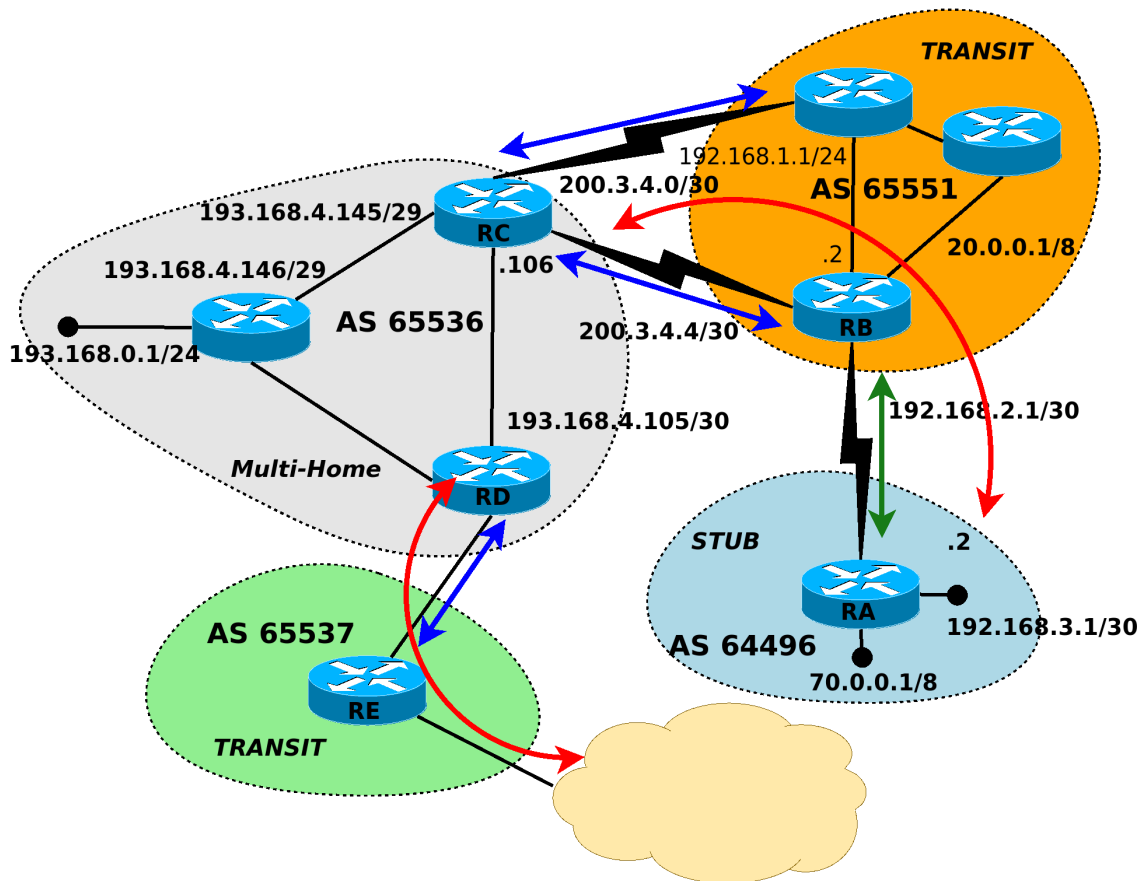


Figura 6.3: Ejemplo de AS de tránsito, stub o multihome.

Par-a-par: Peering, relación entre pares. Interconexión voluntaria entre dos redes de “común y libre acuerdo”. No hay costo por tráfico intercambiado. En general, no hay tránsito.

Cliente/proveedor: Customer/Provider. Relación comercial. Interconexión donde el cliente paga al proveedor. Proveedor provee tránsito, llamado comúnmente tránsito IP, *IP Transit*.

El despliegue de este tipo de conexiones se puede hacer de diferentes maneras. Las relaciones de peering se hacen a través de puntos de Intercambio o **IXP** (antiguamente de acuerdo a la terminología de NSF llamados NAP -Network Access Point-). Los IXP (Internet Exchange Point) son puntos de intercambio multilaterales en zonas neutras, en general, sin tránsito. Conocidos como lugares de **peering público**. Se implementan mediante una red de tecnología L2 de alta velocidad, conformada por uno o más switches, con el propósito de que los AS conecten sus enlaces a sus routers para intercambiar tráfico e información de ruteo. Otra alternativa a los IXP son las interconexiones directas, **Direct-Interconnections**. Estas se implementan mediante enlaces directos L1/L2 entre las redes de los AS interesados. Se realizan mediante acuerdos/contratos de negocios bilaterales o para aumentar la capacidad. Se conocen como **peering privados**.

Las relaciones entre clientes y proveedores se realizan en los **PoP (Point of Presence)**, punto de presencia de las redes de los proveedores. Los PoP son lugares físicos donde los proveedores tienen enlaces y equipamiento: servidores, routers, switches. Son extensiones de la red de su/s AS y es allí donde los clientes se conectan. Habitualmente, los proveedores materializan la conexión, enlaces, hasta el cliente desde sus PoP. En estos casos, las conexiones son directas entre AS, aunque puede verse en

algunos IXP donde se vende tránsito IP. El cliente puede ser un AS final o puede ser un AS de tránsito de otro ISP.

Otro tipo de servicio que puede encontrarse a diferencia del tránsito IP, o el de hacer un peering, es el de transporte del tráfico desde un punto hasta otro, por ejemplo, de una sucursal de una empresa a la casa central o hacia un IXP o PoP. Este servicio usa la infraestructura de la red del proveedor, su/s AS/s, como medio para llegar de un lugar a otro. Al proveedor del servicio de transporte se lo llama **carrier**. Este servicio tiene dos modalidades con objetivos bien distintos. En el primer caso es cuando se contrata el servicio para montar la WAN (Wide Area Network) de una organización sobre otra red que ofrece la cobertura requerida. Se contratan enlaces virtuales sobre los servicios del carrier para formar una red privada del contratista. Si bien la infraestructura física del carrier es compartida los enlaces virtuales son privados. Para eso suelen usarse diferentes tecnologías que van variando con el tiempo, como es MPLS [RVC01]. El segundo caso está asociado a llevar tráfico IP de una red a otra de forma pública, como es el caso hacia un IXP. El carrier, en este caso, llevará los datos del cliente hasta un punto de intercambio público. El enlace de transporte puede estar compartido por varios clientes. Todas estas conexiones y las formas en que se realizan genera una “jerarquía” de AS. Dentro de los cuales tenemos diferentes roles:

- ISPs de distintos tamaños:

ISP locales o Tier 3: en general AS stub. Ofrece servicios a usuarios finales y a organizaciones.

Contratan servicios de tránsito IP a otros ISP. Se considera que en general no hacen peering, aunque podrían conectarse a IXP locales.

ISP Regionales o Tier 2: AS stub o tránsito. Ofrecen servicios a usuarios finales, a organizaciones y a otros ISPs. Tienen acuerdos de peering con algunos ISPs pero requieren contratar servicios de tránsito IP a otros para tener conectividad a todas las redes públicas en Internet. Comercializan tránsito IP y/o transporte. Tienen cobertura nacional o regional.

ISP Tier 1: AS de tránsito. Conforman el núcleo, backbone de Internet. No requieren comprar tránsito IP para llegar a toda la Internet. Están conectados directamente a todos los demás Tier 1 y a una gran cantidad de Tier 2, de los cuales a gran parte le pueden vender servicio. Ofrecen servicios a otros ISP y a grandes organizaciones. Dentro de la misma organización tienen subsidiarias/empresas para ofrecer servicios a usuarios finales. Comercializan tránsito IP y/o transporte (carriers). Tienen cobertura internacional, son poseedores de enlaces que conectan diferentes continentes del mundo. No son muchos.

- Proveedores de contenido.
- Organizaciones finales.

Los AS con sus conexiones hoy dan forma a una topología de mesh (malla) jerárquica donde el Tier 1 forma el núcleo, luego a estos se conectan los Tier 2 y luego los Tier 3. Los AS finales pueden conectarse a uno o varios AS en los diferentes niveles o anillos, pero sin ofrecer tránsito. En la figura 6.4 se muestra un diagrama ilustrativo de esta relación entre AS y los caminos del tráfico.

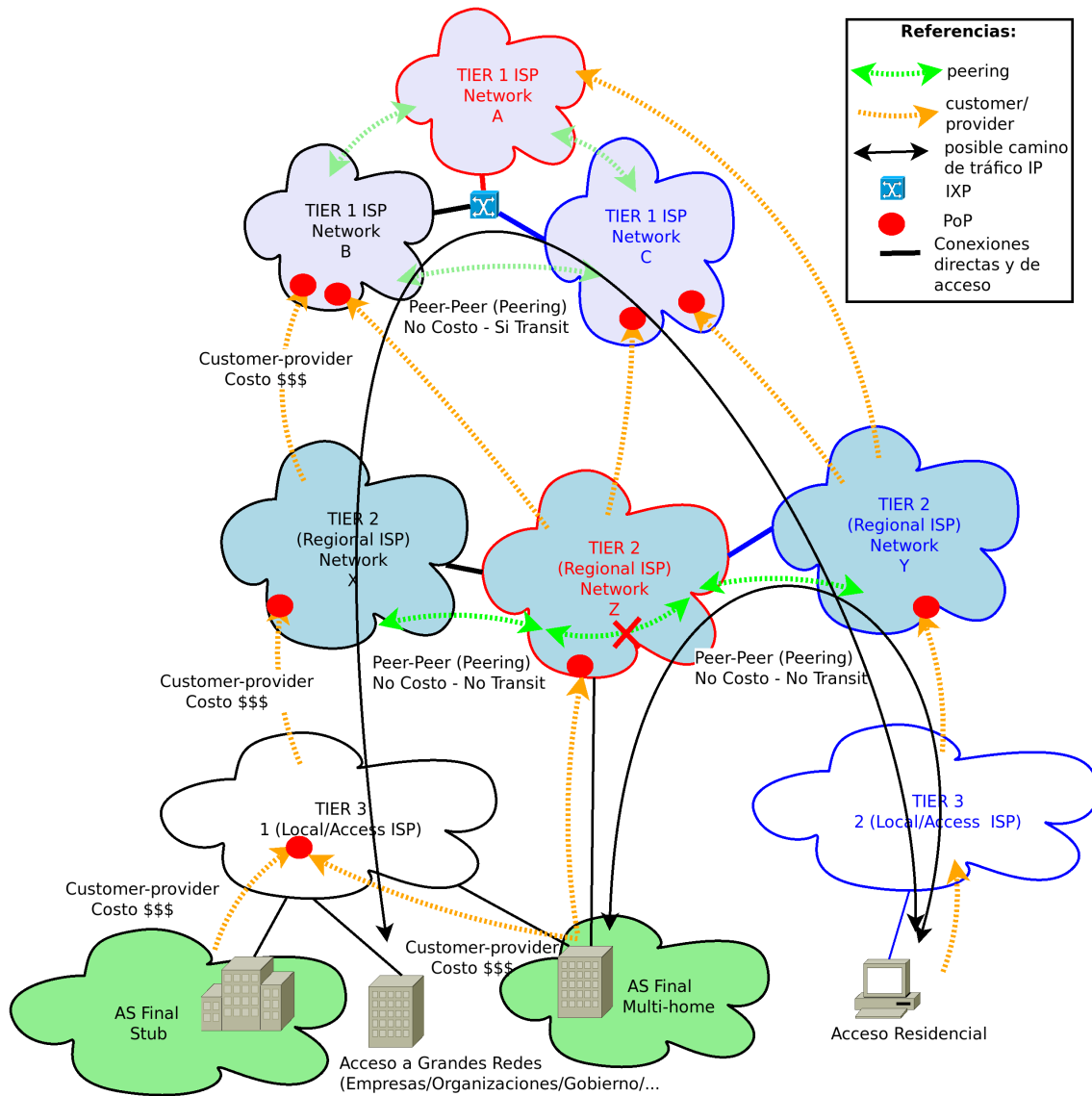


Figura 6.4: Diagrama de la relación entre los sistemas autónomos.

Ejemplo de ruteo en la Red

Para ilustrar lo mencionado anteriormente se mostrará en la topología de estudio el funcionamiento del ruteo y los conceptos vistos. Ver diagrama en capítulos anteriores o sobre la figura 6.8.

Dentro de la red de acceso

El requerimiento HTTP ha iniciado una conexión TCP entre el host *n11-client* y el servidor que hostea la página `http://www.unlp.edu.ar`, *n13-www*. Esta conexión TCP encapsula el tráfico HTTP que se transporta hacia el destino sobre el protocolo IP. Previo a esto los mensajes de DNS también debieron transportarse en IP a otros destinos. Si se analiza la conexión TCP usada por HTTP se puede ver que el tráfico IP debe atravesar varios routers pertenecientes a diferentes AS hasta llegar al servidor web: *IP: 163.10.0.72*.

La tabla del ruteo del cliente se puede inspeccionar y ver que la ruta default, esta es vía el router 10.0.0.1, dentro de una red privada[RMKdG96].

```
# ip route show
default via 10.0.0.1 dev eth0
10.0.0.0/24 dev eth0 proto kernel scope link src 10.0.0.20
# netstat -nr
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.0.1 0.0.0.0 UG 0 0 0 eth0
10.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
```

En este caso el router *n10* se encargará de traducir mediante el proceso de NAT[SH99] la *IP: 10.0.0.20* a una dirección pública, única y ruteable asignada dentro del rango IP al proveedor, en el ejemplo: *40.0.0.2*. Luego, buscará en su tabla de ruteo el próximo salto para el destino final.

Ruteo dentro de la red del ISP

Dentro de la red del proveedor el tráfico IP se encaminará al router de borde del AS. Este ruteo se hará salto a salto y las tablas de ruteo se conformarán de acuerdo a la configuración del ISP. Primero, pasa por red de acceso del ISP que está conformada por el bloque *IP 40.0.0.0/24*. Luego, el router de borde de la red de acceso, el cual no corre BGP, se conecta a la red de distribución/núcleo, bloque *138.0.0.0/24*. En la topología se utilizaron bloques públicos para mostrar el funcionamiento de los protocolos. Es importante aclarar que estos, como el resto de los usados en el ejemplo, están en uso y pertenecen a organizaciones que hoy se conectan a Internet y rutean con los mismos. En cambio los números de sistema autónomo utilizados son privados y no concuerdan con los bloques IP públicos. En el ejemplo la red es muy sencilla y no se requiere usar un IGP. Se resuelve, en la mayoría de los casos, la conformación de las tablas de ruteo de forma estática. Por simplicidad se integra la red de acceso con el tier 3. Esto, en la realidad, a veces no sucede. Para mostrar el funcionamiento de OSPFv2[Moy98] se puede activar el mismo en los routers del ISP. A partir de esto se pueden ver algunos de los mensajes intercambiados e inspeccionar como queda conformada la tabla de adyacencias de OSPF y la de ruteo (RIB) en el router *n6*.

```
n6# show ip ospf neighbor
OSPF Instance: 1
Neighbor ID Pri State Dead Time Address Interface RXmtL RqstL DBsmL
138.0.0.1 1 Full/Backup 32.319s 138.0.0.1 eth1:138.0.0.2 1 0 0

n6# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
```



```

F - PBR, f - OpenFabric,
> - selected route, * - FIB route, q - queued route, r - rejected route

C * 16.0.0.0/16 is directly connected, eth3, 00:00:51
C>* 16.0.0.0/16 is directly connected, eth3, 00:00:51
O[1]>* 40.0.0.0/24 [110/20] via 138.0.0.1, eth1, 00:00:03
S 40.0.0.0/24 [200/0] via 138.0.0.1, eth1, 00:00:53
B>* 50.0.0.0/28 [20/0] via 90.0.0.1, eth2, 00:00:50
B>* 80.0.0.0/24 [20/0] via 90.0.0.1, eth2, 00:00:50
B 90.0.0.0/24 [20/0] via 90.0.0.1 inactive, 00:00:50
C>* 90.0.0.0/24 is directly connected, eth2, 00:00:52
O[1] 138.0.0.0/24 [110/10] is directly connected, eth1, 00:00:53
C>* 138.0.0.0/24 is directly connected, eth1, 00:00:53
B>* 163.10.0.0/24 [20/0] via 90.0.0.1, eth2, 00:00:50
B>* 198.41.0.0/24 [20/0] via 90.0.0.1, eth2, 00:00:50
B>* 199.7.83.0/24 [20/0] via 90.0.0.1, eth2, 00:00:50
C>* 201.0.0.0/24 is directly connected, eth0, 00:00:54

```

En este caso las rutas involucradas en OSPFv2 como IGP están marcadas con la letra "O". Se ve que la ruta a la red de acceso, 40.0.0.0/24, se aprendió por este IGP, además de tenerla estática con una distancia administrativa más baja, 200.

```

...
O[1]>* 40.0.0.0/24 [110/20] via 138.0.0.1, eth1, 00:00:03
S 40.0.0.0/24 [200/0] via 138.0.0.1, eth1, 00:00:53
...

```

Esta métrica, llamada distancia administrativa, hace de valor unificador para poder comparar rutas aprendidas por distintos protocolos de ruteo. En la salida del comando, mostrado sobre la plataforma usada, se ven los valores *OSPF* = 110 y *STATIC* = 200. Estos se pueden modificar a mano, como en este caso que se cambió el valor de las rutas estáticas de 1 por 200. Luego, cada protocolo de ruteo utilizará sus propias métricas para comparar. Por ejemplo, RIP cuenta saltos y OSPF compara ancho de banda digital. Para OSPF, el valor obtenido en la ruta hacia 40.0.0.0/24 es 20. El router *n6* va a ejecutar el IGP: OSPF y el EGP: BGP ya que es un router borde. El router *n7* solo ejecuta el IGP: OSPF o trabajará con ruteo estático. Los comandos de activación e inspección de los estados de los IGP varían de una plataforma a otra.

En la figura 6.5 se puede inspeccionar un mensaje del protocolo OSPFv2 de intercambio de rutas entre *n6* y *n7*.

Ruteo entre los ISP y el núcleo de Internet

Si nos situamos un nivel más arriba de la jerarquía, en *n6*, se tiene que el router de borde de la red del proveedor va a establecer dos sesiones del protocolo de ruteo dinámico BGP, una con *n3* mediante una conexión de IXP y otra con *n4* con una conexión directa.

En el gráfico 6.6 se muestran las divisiones y relaciones entre los diferentes AS. Se marcan, además, las sesiones BGP y las adyacencias OSPF.

En la topología, por simplicidad, solo se muestran tres niveles: la red de acceso y dos de tiers: los tier 2 y 3 juntos y el tier 1. Los sistemas autónomos donde se encuentran los DNS externos no se configuraron con routers de borde ni ASN propio, y se los conectó directamente a routers de otros AS. Esto último también se hace para dejar el ruteo en el ejemplo más sencillo. En el núcleo de la red, conformado por los routers *n2*, *n3*, *n4*, todos van a intercambiar rutas mediante BGP para que el tráfico IP pueda llegar a los diferentes puntos de la red. Estos routers en la jerga se les llama hablantes BGP (*BGP speakers*) y establecen sesiones de vecindad usando este protocolo que se monta sobre el transporte de TCP, port 179. En la figura 6.7 se puede ver un update BGP, una vez ya establecida la vecindad, entre *n2* y *n3* mediante el cual le publica las redes que serían alcanzables con sus correspondientes atributos.

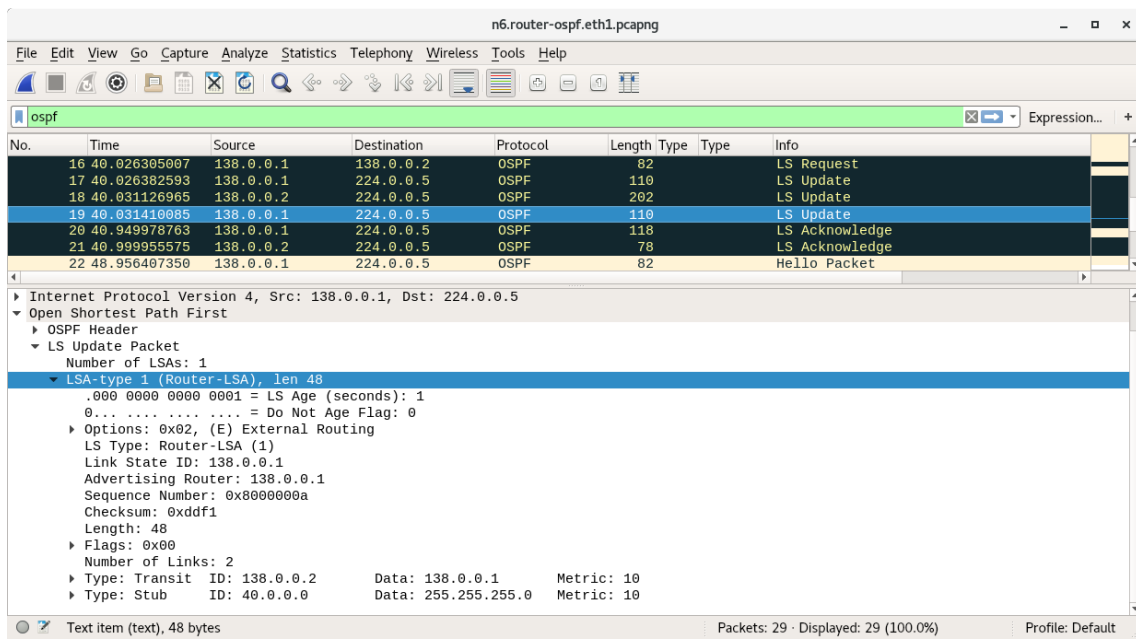


Figura 6.5: Captura de mensaje de OSPFv2 entre routers del mismo AS.

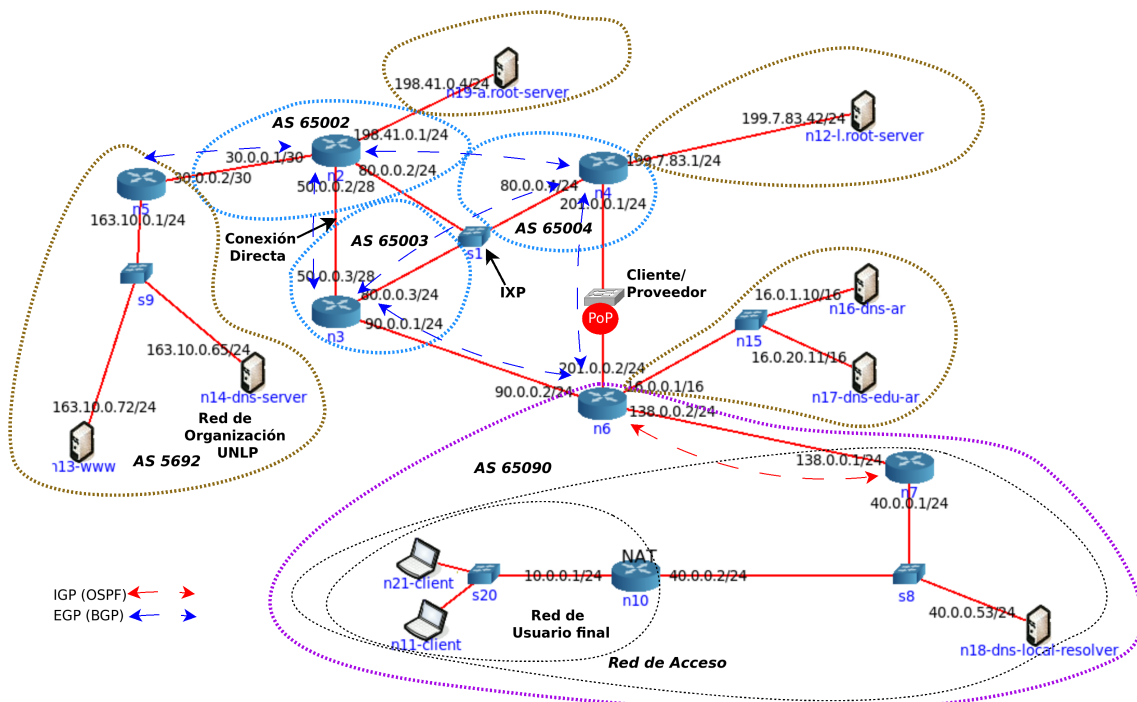


Figura 6.6: Diagrama de las distintas redes y los sistemas autónomos (AS).

The screenshot shows a Wireshark capture of BGPv4 traffic. The packet list pane displays several BGP messages:

No.	Time	Source	Destination	Protocol	Length	Type	Info
74	1.041687953	80.0.0.4	80.0.0.2	BGP	141		OPEN Message
76	1.041746591	80.0.0.2	80.0.0.4	BGP	85		KEEPALIVE Message
77	1.041884319	80.0.0.4	80.0.0.2	BGP	85		KEEPALIVE Message
88	1.993755048	80.0.0.2	80.0.0.3	BGP	153		UPDATE Message, UPDATE Message

The selected packet (88) details are as follows:

- Frame 88: 153 bytes on wire (1224 bits), 153 bytes captured (1224 bits) on interface 0
- Ethernet II, Src: 00:00:00:aa:00:00 (00:00:00:aa:00:00), Dst: 00:00:00:aa:00:01 (00:00:00:aa:00:01)
- Internet Protocol Version 4, Src: 80.0.0.2, Dst: 80.0.0.3
- Transmission Control Protocol, Src Port: 179, Dst Port: 57422, Seq: 95, Ack: 95, Len: 87
- Border Gateway Protocol - UPDATE Message
 - Marker: ffffffffffffffffffffffffffffffff
 - Length: 64
 - Type: UPDATE Message (2)
 - Withdrawn Routes Length: 0
 - Total Path Attribute Length: 28
 - Path attributes
 - Path Attribute - ORIGIN: IGP
 - Path Attribute - AS_PATH: 65002
 - Path Attribute - NEXT_HOP: 80.0.0.2
 - Path Attribute - MULTI_EXIT_DISC: 0
 - Network Layer Reachability Information (NLRI)
 - 50.0.0.0/28
 - 198.41.0.0/24
 - 80.0.0.0/24

Figura 6.7: Captura de mensaje de BGPv4 entre routers de diferentes AS, tier 1.

Volviendo a bajar un nivel, al tier 2 ó 3, se puede observar que la red, donde se encuentra el servidor web que aloja la página solicitada por el cliente, se conecta mediante el vínculo entre los routers de borde $n5$ con $n2$. Estos también se comunicarán por BGP para intercambiar rutas. Si se observa la salida de los comandos de BGP en el router $n3$ se puede ver con que routers va a hablar el protocolo en la tabla de vecinos y que AS tiene cada uno.

```
n3# show ip bgp summary
```

```
IPv4 Unicast Summary:
BGP router identifier 80.0.0.3, local AS number 65003 vrf-id 0
BGP table version 15
RIB entries 17, using 3128 bytes of memory
Peers 4, using 82 KiB of memory

Neighbor      V      AS MsgRcvd MsgSent  TblVer  InQ OutQ  Up/Down State/PfxRcd
50.0.0.2      4      65002    28     28       0    0    0 00:12:23      5
80.0.0.2      4      65002    20     22       0    0    0 00:12:24      5
80.0.0.4      4      65004    22     25       0    0    0 00:12:24      5
90.0.0.2      4      65090    30     30       0    0    0 00:12:22      3
```

También se puede inspeccionar la tabla BGP, donde se encuentran todas las rutas aprendidas mediante este protocolo.

```
Total number of neighbors 4
n3# show ip bgp
```

```
BGP table version is 15, local router ID is 80.0.0.3, vrf id 0
Default local pref 100, local AS 65003
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete

Network      Next Hop      Metric LocPrf Weight Path
*> 16.0.0.0/16 90.0.0.2      0         0 65090 i
*> 40.0.0.0/24 90.0.0.2      0         0 65090 i
* 50.0.0.0/28  50.0.0.2      0         50 0 65002 i
*              80.0.0.2      0         0 65004 65002 i
*>              80.0.0.2      0         0 65002 i
* 80.0.0.0/24  50.0.0.2      0         50 0 65002 i
*              80.0.0.2      0         0 65002 i
*              80.0.0.4      0         0 65004 i
*>              0.0.0.0      0        32768 i
*> 90.0.0.0/24  0.0.0.0      0        32768 i
*> 138.0.0.0/24 90.0.0.2      0         0 65090 i
* 163.10.0.0/24 50.0.0.2     50         0 65002 5692 i
```

```

*           80.0.0.2           0 65004 65002 5692 i
*>          80.0.0.2           0 65002 5692 i
* 198.41.0.0/24 50.0.0.2       0 50      0 65002 i
*           80.0.0.2           0 65004 65002 i
*>          80.0.0.2           0 65002 i
* 199.7.83.0/24 50.0.0.2       0 50      0 65002 65004 i
*           80.0.0.4           0 65002 65004 i
*>          80.0.0.4           0 65004 i
Displayed 9 routes and 20 total paths

```

En esta tabla se pueden ver varios de los atributos asociadas a las rutas, por ejemplo: el próximo salto (next-hop), Metric (MED -Multi-exit discriminator-), la preferencia local (local-pref), el peso (weight) y la lista de AS a atravesar en la columna *Path*. Para las mismas rutas se pueden tener diferentes caminos. A partir de la tabla BGP y de información de otros protocolos se conformará la tabla de ruteo o RIB. Se muestra a continuación.

```

n3# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       0 - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

B>* 16.0.0.0/16 [20/0] via 90.0.0.2, eth2, 00:12:42
B>* 40.0.0.0/24 [20/0] via 90.0.0.2, eth2, 00:12:42
B  50.0.0.0/28 [20/0] via 80.0.0.2, eth0, 00:12:44
C>* 50.0.0.0/28 is directly connected, eth1, 00:12:44
C>* 80.0.0.0/24 is directly connected, eth0, 00:12:45
C>* 90.0.0.0/24 is directly connected, eth2, 00:12:43
B>* 138.0.0.0/24 [20/0] via 90.0.0.2, eth2, 00:12:42
B>* 163.10.0.0/24 [20/0] via 80.0.0.2, eth0, 00:12:43
B>* 198.41.0.0/24 [20/0] via 80.0.0.2, eth0, 00:12:44
B>* 199.7.83.0/24 [20/0] via 80.0.0.4, eth0, 00:12:44

```

En este caso las rutas aprendidas por el protocolo BGP e insertadas en la RIB se marcan con la letra “B” y tienen una distancia administrativa de 20². Por ejemplo:

```

...
B>* 163.10.0.0/24 [20/0] via 80.0.0.2, eth0, 00:12:43
...

```

En el caso de los routers que tienen más de un enlace podrán a través de las métricas del protocolo y/o los atributos que se le agregan a las rutas tratar de definir sus preferencias por algún camino. En el caso de BGP los atributos son varios, pudiéndose con estos manejar el tráfico de acuerdo a políticas o cuestiones que no necesariamente tienen que ver con la tecnología de los enlaces. Para los IGP, este manejo, en la mayoría de los casos, se deja que se haga de forma automática por los routers sin definir muchas reglas por encima de lo que decide el protocolo.

Si se analizan las entradas en los routers del camino se puede ver que cada uno tendrá un entrada que indica como llegar al próximo salto. La vuelta, probablemente, sea por el mismo camino aunque se podría llegar a tomar otro. Una herramienta para descubrir el camino que atraviesan los mensajes desde un punto a otro de la red es `traceroute(8)`. Por ejemplo, se puede ejecutar desde el cliente y verla en acción.

```

root@n11-client.conf# traceroute -n 163.10.0.72
traceroute to 163.10.0.72 (163.10.0.72), 30 hops max, 60 byte packets
 1 10.0.0.1 0.098 ms 0.030 ms 0.025 ms
 2 40.0.0.1 0.189 ms 0.078 ms 0.036 ms
 3 138.0.0.2 0.064 ms 0.042 ms 0.040 ms
 4 90.0.0.1 0.072 ms 0.049 ms 0.046 ms
 5 80.0.0.2 0.070 ms 0.055 ms 0.054 ms
 6 30.0.0.2 0.073 ms 0.102 ms 0.062 ms
 7 163.10.0.72 0.146 ms 0.100 ms 0.160 ms

```

²En varias de las plataformas de ruteo, la distancia administrativa para el caso de eBGP es 20 y para iBGP su valor es de 200, aunque pueden variar o, en ocasiones, ser para ambos protocolos iguales sus valores.

Las entradas en los routers del camino, uno por uno, serían las que se muestran en las salidas a continuación. En el router *n10* que cumple el rol de router de la red del usuario final.

```
n10# ip route show
default via 40.0.0.1 dev eth0
10.0.0.0/24 dev eth1 proto kernel scope link src 10.0.0.1
40.0.0.0/24 dev eth0 proto kernel scope link src 40.0.0.2
```

En el router *n7* de la red de acceso del proveedor.

```
n7# show ip route 163.10.0.0
Routing entry for 0.0.0.0/0
  Known via "ospf[1]", distance 110, metric 1
  Last update 00:15:49 ago
    138.0.0.2, via eth0, weight 1

Routing entry for 0.0.0.0/0
  Known via "static", distance 1, metric 0, best
  Last update 00:16:41 ago
    * 138.0.0.2, via eth0, weight 1
```

En el router de borde *n6* de la red del proveedor.

```
n6# show ip route 163.10.0.0
Routing entry for 163.10.0.0/24
  Known via "bgp", distance 20, metric 0, best
  Last update 00:17:03 ago
    * 90.0.0.1, via eth2, weight 1
```

En los routers de núcleo/backbone *n3* y *n2*.

```
n3# show ip route 163.10.0.0
Routing entry for 163.10.0.0/24
  Known via "bgp", distance 20, metric 0, best
  Last update 00:17:32 ago
    * 80.0.0.2, via eth0, weight 1
```

```
n2# show ip route 163.10.0.0
Routing entry for 163.10.0.0/24
  Known via "bgp", distance 20, metric 0, best
  Last update 00:18:09 ago
    * 30.0.0.2, via eth2, weight 1
```

En el router de borde del proveedor de contenidos, *n5*, en este caso la red de la UNLP que provee el sitio *www.unlp.edu.ar*.

```
n5# show ip route 163.10.0.0
Routing entry for 163.10.0.0/24
  Known via "connected", distance 0, metric 0, best
  Last update 00:19:00 ago
    * directly connected, eth1
```

En la figura 6.8 se marca el camino indicado por la herramienta en la topología de test.

De esta forma se han dado algunos detalles de los protocolos y la red IP que hará posible establecer un camino para que el tráfico desde el cliente llegue al destino, como así también las respuestas.

REFERENCIAS

- [BM02] R. Bush y D. Mayer. «Rfc 3439: Some internet architectural guidelines and philosophy», 2002.
- [Car96] B. Carpenter(Ed.). «Rfc 1958: Architectural principles of the internet», 1996.
- [Hal00] Bassam Halabi. «Internet routing architectures». Cisco Press, 2000. ISBN: 157870233X.
- [Hed88] C. Hedrick. «Rfc 1058: Routing information protocol (rip)», 1988.
- [KE04] Kempf y Austein (Editors). «Rfc 3724: The rise of the middle and the future of end-to-end: Reflections on the evolution of the internet architecture», 2004.
- [KR12] J. Kurose y K. Ross. «Computer networking: A top-down approach, 6th. ed.». Addison-Wesley, 2012. ISBN: 0132856204.
- [LR89] K. Lougheed y Y. Rekhter. «Rfc 1105: A border gateway protocol (bgp)», 1989.
- [Mil84] D. L. Mills. «Rfc 904: Exterior gateway protocol formal specification», 1984.
- [Moy89] J. Moy. «Rfc 1131: The ospf specification», 1989.
- [Moy98] J. Moy. «Rfc 2328: Ospf version 2», 1998.
- [NG13] Thomas D. Nadeau y Ken Gray. «Sdn: Software defined networks». O'Reilly Media, Inc., 2013. ISBN: 9781449342302.
- [RHS82] Robert Robert Hinden y Alan Sheltzer. «Rfc 823: The darpa internet gateway (ggp)», 1982.
- [RLH06] Y. Rekhter(Ed.), T. Li(Ed.) y S. Hares(Ed.). «Rfc 4271: A border gateway protocol 4 (bgp-4)», 2006.
- [RMKdG96] Y. Rekhter, B. Moskowitz, D. Karrenberg y deGroot G.J. «Rfc 1918: Address allocation for private internets», 1996.
- [Ros82] Eric Rosen. «Rfc 827: Exterior gateway protocol (egp)», 1982.
- [RVC01] E. Rosen, A. Viswanathan y R. Callon. «Rfc 3031: Multiprotocol label switching architecture», 2001.
- [SH99] P. Srisuresh y M. Holdrege. «Rfc 2663: Ip network address translator (nat) terminology and considerations», 1999.

CAPÍTULO 7

ENLACE: ETHERNET, ARP Y SWITCHING

MATÍAS ROBLES

Introducción

Para finalizar con el conjunto de actores introducidos en el Capítulo 1 vamos a analizar los dos últimos, Ethernet[SZ14] y ARP[Plu82], correspondientes a la capa de enlace. Esta capa se diferencia del resto de las capas vistas hasta el momento en que puede ir cambiando a medida que un mensaje se mueve entre distintas redes. Por ejemplo, el host origen de un paquete IP podría estar en una red wireless, del tipo IEEE 802.11, que define un formato específico de trama (en inglés frame) en la capa de enlace, pero, si en su camino hacia el host destino debe ser enviado por una red con una tecnología diferente, posiblemente Ethernet, el paquete IP será encapsulado en otro tipo de trama.

En ese camino entre un origen y un destino el direccionamiento lógico, direcciones IP, es común a todas las redes, pero no sucede lo mismo con el direccionamiento físico, que es propio de cada tecnología de interfaz de red (network interface). En el caso de Ethernet, ese direccionamiento físico está representado por direcciones MAC (Medium Access Control, también llamada Media Access Control). Para poder enviar un mensaje son necesarias direcciones de esos dos niveles, pero son dos identificadores totalmente diferentes. Si se necesita enviar un mensaje a un host y se conoce su dirección IP se requiere algún mecanismo que sea capaz de averiguar la dirección MAC asociada a esa dirección IP. Cómo funciona este mecanismo es uno de los temas que se verán en este capítulo.

Antes de introducirnos en esos temas es necesario que se comprendan dos conceptos fundamentales para entender el contenido de este capítulo: dominio de colisión y dominio de broadcast. El primero, define cual es la parte de la red en la que si un nodo transmite tiene posibilidades de colisionar con la transmisión de otro nodo y, por su parte, el segundo define cual es la parte de una red en la que un mensaje de tipo broadcast viajará sin ser detenido. El alcance de cada uno de esos dominios depende de los dispositivos de red que se utilicen. Los switches dividen los dominios de colisión y los routers hacen lo mismo tanto con los dominios de colisión como con los de broadcast (los hubs no dividen ninguno de los dos dominios). Usando el gráfico en la figura 7.1, que es una parte de la red usada en el desarrollo del libro, podemos ver cuales son cada uno de estos dominios.

La posibilidad de colisión se debe a que Ethernet originalmente se definió como un mecanismo de acceso a un medio de forma compartida, CSMA/CD, donde solo se podía tener una transmisión en

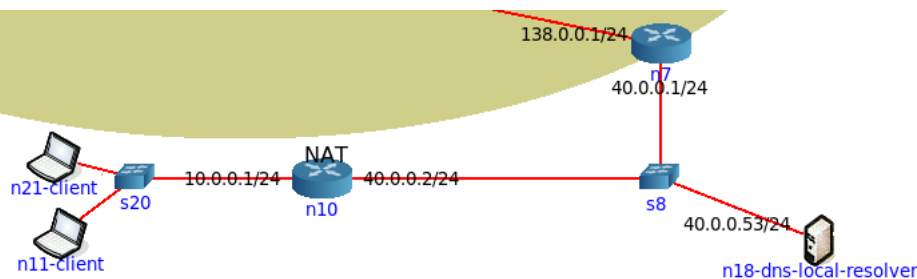


Figura 7.1: Dominios de Colisión y de Broadcast

curso. En la parte de la red donde está el switch *s20* hay 3 dominios de colisión. Cada uno de los nodos están conectados punto a punto a ese switch, inclusive la interfaz del router en esa red, lo que se conoce como micro-segmentación, y cada una de esas conexiones es un dominio de colisión diferente. Por su parte, se tiene un solo dominio de broadcast. Dentro de esa misma red, cualquier mensaje de tipo broadcast llegará a todos los nodos, pero el router no lo reenviará hacia la otra red: el router detiene los mensajes broadcast. Siguiendo con el gráfico en la red donde está el switch *s8* también se tienen 3 dominios de colisión y uno solo de broadcast.

Ethernet

Ethernet es un estándar que define las dos últimas capas del modelo OSI, la capa de enlace y la física, y consta de dos versiones: Ethernet, definida por el consorcio DIX (Digital Equipment Corporation (DEC), Xerox e Intel) y actualmente en la versión 2, Ethernet II, y el estándar IEEE 802.3. Ambas versiones son similares y coexisten en la realidad (si se realiza una captura de tráfico muy probablemente se verá tráfico de ambas versiones). Solo tiene alcance dentro del dominio de broadcast en el cual es generada, lo que significa que una trama, ese es el nombre que recibe una PDU en la capa de enlace aunque también se lo suele llamar marco o directamente en su término nativo frame, no será reenviada por un dispositivo de capa 3. Esto implica que la trama viajará por la propia LAN sin alteraciones ni modificaciones y será recibida y procesada por uno o más equipos según el direccionamiento y los equipos de red involucrados.

Con el fin de comprender el papel que desempeña en todo el proceso que se viene desarrollando a lo largo del libro, es que se comenzará por analizar una trama Ethernet, la cual se puede ver en la figura 7.2



Figura 7.2: Trama Ethernet II/IEEE 802.3

Dirección MAC Destino: 6 bytes. Puede ser de tipo unicast, multicast o broadcast. Indica cual o cuales son los posibles receptores de la trama.

Dirección MAC Origen: 6 bytes. Solo puede ser de tipo unicast e indica cual es el dispositivo que generó la trama. Un dispositivo utilizará su dirección MAC como dirección origen en cualquier trama

que transmita. Si tuviese más de una interfaz de red seleccionará la dirección MAC de la interfaz por donde enviará la trama.

Tipo/Longitud: 2 bytes. Este campo es diferente en Ethernet II y IEEE 802.3. En Ethernet II se usa como Tipo e indica que protocolo de capa superior se está transportando en el campo Datos, en cambio, en IEEE 802.3 se usa como Longitud e indica la cantidad de bytes transportados en el campo Datos. Si el valor del campo es menor o igual a 1500 (decimal) entonces el campo está siendo usado como Tipo y si es mayor o igual a 1536 decimal (0x600 hexadecimal) entonces se lo está usando como Longitud.

Datos: mínimo de 46 bytes, máximo de 1500. Si es menor a 46 bytes se deben agregar bytes de padding hasta llegar a ese tamaño. En Ethernet II, el tamaño mínimo de trama esta garantizado por la capa superior y en IEEE 802.3 el campo Longitud indica la cantidad de datos sin relleno (padding). Por ejemplo, en este campo viajarían los paquetes IP o ARP.

FCS (Frame Check Sequence): 4 bytes. Permite chequear la integridad de la trama. Su valor es calculado usando un CRC, Cyclic Redundancy Check, por el emisor. Si su chequeo falla en el receptor de la trama, éste la descartará sin avisarle al emisor.

Una trama Ethernet, sin modificaciones, tiene un tamaño máximo de 1518 bytes, pero este valor puede sufrir alteraciones. Por ejemplo, si definimos VLANs usando el estándar IEEE 802.1Q se le deben agregar 4 bytes a cada trama para transportar información de la VLAN, lo que lleva el tamaño de la misma a 1522 bytes. Evoluciones del estándar Ethernet permiten sobrepasar este límite dando lugar a lo que se conoce como jumbo-frames, comúnmente puede llevar una carga útil (datos) de hasta 9000 bytes.

En cuanto al direccionamiento en la trama, tanto las direcciones origen como destino son direcciones MAC. Estas son direcciones de 48 bits que se las divide en 2 partes de 24 bits cada una. La primera mitad es conocida como OUI, Organizationally Unique Identifier, sirve para identificar a los distintos fabricante (Cisco, Intel, Broadcom, etc.) y es administrada por la IEEE. La segunda mitad de la dirección es asignada por cada uno de los fabricantes, quienes deben garantizar la unicidad de cada dirección MAC. Cada dispositivo en el mundo que forme parte de una red de tipo Ethernet tiene una dirección MAC diferente. Esta dirección pertenece a la placa de red, no al dispositivo (si se le cambia la placa de red a un dispositivo también se le cambia su dirección MAC).

Ethernet ha estado evolucionando constantemente desde sus inicios en la década del 70, pasando de los 10 Mbits/s iniciales a los 100 Gbit/s en la actualidad. También, ha dejado de ser un medio half-duplex (todos los dispositivos pueden transmitir pero no simultáneamente) para transformarse en un medio completamente full-duplex (todos los dispositivos pueden transmitir simultáneamente), lo que sucede desde la versión de los 10 Gbit/s (IEEE 802.3ae del año 2002). Aunque originalmente no fue la única tecnología disponible para las redes LANs, en sus inicios compitió entre otras con Token Bus (IEEE 802.4) y Token Ring (IEEE 802.5), con el tiempo se convirtió en el estándar de-facto para ese tipo de redes. En los últimos años se puede ver que las redes WI-FI, IEEE 802.11, han gozado de gran

popularidad entre las redes LAN.

Entre las características en las que se ha basado su éxito se puede mencionar su simplicidad. Comparada con el resto de sus competidores, Ethernet es más simple de entender, implementar y configurar. Simplemente, se puede decir que en Ethernet cuando un nodo quiere enviar una trama, lo hace. No tiene que esperar un turno o, como en Token Ring, un token que lo habilite a transmitir. Sin embargo, es preciso indicar que tiene un procesamiento distinto entre las versiones half-duplex y full-duplex.

Al operar en modo half-duplex, cuando un nodo quiere transmitir hace uso del método de acceso CSMA/CD (Carrier Sense Multiple Access/Collision Detection). Sin profundizar en los detalles, antes de transmitir una trama el nodo debe "sensar el medio", es decir, escuchar el medio para determinar si hay otra transmisión en curso. Si la hay, debe esperar hasta que la actual transmisión finalice para luego intentar transmitir. En caso contrario, el nodo envía su trama. Si dos o más dispositivos transmiten simultáneamente las tramas colisionarán y, en consecuencia, se perderán. Obviamente, estas tramas se deben volver a enviar, pero si todos los nodos involucrados en la colisión deciden enviar esas tramas inmediatamente después de la colisión, volverán a colisionar. Para evitar esto, es que cada nodo, de manera totalmente independiente, ejecuta un algoritmo de back-off que indica un tiempo aleatorio que cada nodo deberá esperar antes de volver a transmitir. Esto no evita por completo que al transmitir nuevamente esas tramas vuelvan a colisionar. En este caso, los nodos involucrados deberán ejecutar nuevamente el algoritmo de back-off. Esto se repetirá en cada nodo hasta que pueda transmitir la trama de manera exitosa o hasta que de error después de una cantidad de intentos fallidos. Esta aleatoriedad, no poder garantizar quien será el próximo en transmitir, es lo que convierte a Ethernet en un protocolo no determinístico.

Si la trama es transmitida exitosamente, sin colisiones, arribará a un nodo o varios nodos destino. Al recibir la trama, cada nodo primero debe verificar la integridad de la misma, lo que realiza mediante el cálculo del CRC. Si éste es correcto, la trama es aceptada; en caso contrario, es descartada y lo hace en modo silencioso, sin avisar nada al emisor. Algún protocolo de la capa superior en el nodo emisor deberá encargarse de detectar la pérdida de los datos e iniciar una retransmisión. A continuación, el receptor controla que la MAC destino de la trama coincida con la MAC asignada a la interface de red por donde se la recibió. Si coincide, la aceptará para su procesamiento, caso contrario, la descartará. También aceptará tramas con dirección MAC destino de tipo broadcast o multicast, en este último caso solo si el nodo está escuchando en la correspondiente dirección multicast.

En el modo full-duplex, las estaciones se comunican usando un medio de transmisión dedicado, punto a punto. Cuando un nodo quiere transmitir no debe esperar para hacerlo ni tiene que quedarse "sensando" el medio por una posible colisión una vez que lo hizo. No hay posibilidad de colisiones en un medio de este tipo. Consecuentemente, tampoco es necesario ejecutar un algoritmo de acceso al medio, como CSMA/CD, debido a que no hay contención por la utilización de un medio compartido. Si se quiere tener una red LAN de tipo full-duplex es necesario contar con un switch y conectar un único

dispositivo a cada uno de sus puertos. No es posible, en estos casos, utilizar un hub o repetidor. De hacerlo, el funcionamiento de la misma pasaría a ser half-duplex.

ARP

Si bien Ethernet es la tecnología de LAN más utilizada en la actualidad, no es la única. Como se explicó en la introducción del capítulo, un paquete IP puede pasar por distintos tipos de redes físicas con sus propios formatos de trama y direccionamiento en su capa de enlace. Este funcionamiento podría requerir, entre otras cosas, realizar una conversión entre las direcciones utilizadas en la capa de red (direccionamiento lógico) y las de la correspondiente capa de enlace (direccionamiento físico). En nuestro caso nos referimos específicamente al procedimiento de mapear direcciones IP de 32 bits en direcciones MAC de 48 bits. Este método es común para todos los estándares que utilicen direcciones MAC, como sucede con las redes Ethernet o WI-FI. El protocolo de capa de enlace encargado de realizar esta tarea es ARP (Address Resolution Protocol) definido en la RFC-826. Si bien este protocolo puede ser usado para otras finalidades, como la detección de direcciones duplicadas, en este capítulo nos vamos a enfocar en la funcionalidad de mapear direcciones IP en direcciones MAC.

Algunos autores suelen ubicar a ARP en el medio de las capas de red y enlace indicando que pertenece a la capa 2.5 (dos y medio).

Aunque es un protocolo presente en prácticamente todas las redes LANs su uso es transparente tanto para los usuarios de la red, inclusive el administrador de la misma, como para las aplicaciones. ARP realiza todo su trabajo de forma dinámica, no necesita ninguna configuración ni administración. El mapeo de una dirección IP a la correspondiente dirección MAC se va haciendo a medida que se necesita, lo que posibilita que la red se adapte a los cambios que vayan sucediendo. Esto implica que si un nodo cambia su placa de red, y en consecuencia su dirección MAC, el administrador de la red no deberá hacer nada para que los demás nodos de la LAN se den cuenta de este cambio.

ARP funciona únicamente con IPv4. En IPv6, en su reemplazo, se utiliza una de las funciones provistas en el protocolo Neighbor Discovery.

ARP no está definido para ningún tipo de direccionamiento en particular, es un protocolo genérico desarrollado para mapear direcciones de diferentes protocolos de capa de red en direcciones físicas de distintos tamaños, pero en la RFC-826 se habla específicamente de direcciones físicas de 48 bits de una red Ethernet. Y en cuanto a la capa de red, raramente se la utiliza con direcciones distintas a IPv4. Es un protocolo con un funcionamiento muy simple que solo define dos tipos de mensajes, ARP Request y ARP Reply, que al enviarse son encapsulados en el campo Datos de una trama Ethernet. En el análisis de esos paquetes ARP solo nos vamos a enfocar en los campos "Hardware Address", que se corresponden con las direcciones MAC, y "Protocol Address", que lo hacen con las direcciones IPv4.

Cuando un host tiene un mensaje para enviar va a necesitar, entre otra información, 4 direcciones: IP Origen e IP Destino, como parte de su capa de red, y MAC Origen y MAC Destino, como parte de su capa de enlace. De estas 4 direcciones el host conoce 3, sus propias direcciones, IP Origen y MAC

Origen; y la IP Destino, que es la dirección del host al que se quiere llegar (la sabe porque se indica en el propio comando, por ej. ping 163.10.5.66, o porque la deduce usando el servicio de DNS). Lo que el emisor no conoce y necesita determinar es la dirección MAC Destino, la cual le permitirá completar la trama Ethernet. Esta dirección es la que indica a qué host va dirigida la trama. Pero que dirección aprenderá va a depender de si el host destino se encuentra en la misma red que el host origen o no.

Lo primero que hará el host origen será chequear su tabla de ruteo para determinar si el host destino pertenece a su propia red, es decir, si son “vecinos”. Si es así, enviará un mensaje de tipo ARP Request solicitando la dirección MAC asociada a la IP Destino. La dirección MAC corresponderá al nodo final al que se está queriendo acceder. Por el contrario, si no son vecinos, el ARP Request preguntará por la dirección MAC del default gateway del host origen. Como se indicó más arriba, ARP se encapsula en Ethernet y los mensaje ARP Request se envían en tramas Ethernet de tipo broadcast. El límite de estas tramas son los routers, que, como se sabe, dividen los dominios de broadcast y, en consecuencia, no reenvían los broadcasts. De esto último se puede inferir que un host no puede utilizar ARP para averiguar la dirección MAC de un host que no se encuentre en su misma red.

Aunque no sea un router, un host tiene su propia tabla de ruteo, generalmente con dos entradas: una que indica la red a la que pertenece el host y otra con la ruta default con su default-gateway. Esta información puede ser configurada estáticamente o aprendida por el protocolo DHCP

Indistintamente de a quien se envía el ARP Request, la respuesta vendrá en un mensaje ARP Reply siempre que esté disponible el nodo por el cual se consulta. Estas respuestas serán de tipo unicast y también viajarán en tramas Ethernet.

Caso práctico

Resumiendo lo visto hasta ahora, una vez que se ingresa la URL: `http://www.unlp.edu.ar` en el navegador del host *n11-client* lo primero que se dispara es la resolución de nombres, descubrir la dirección IP asociada al servidor de esa página. El resolver local de este host genera una consulta DNS que es enviada, de forma recursiva, al servidor de nombres que tiene configurado. Para esto utiliza un protocolo de capa de transporte, UDP, que al bajar en el stack TCP/IP se transforma en el payload de un paquete IP. Este paquete IP sigue bajando por el stack y es pasado a la capa de enlace, que en nuestro caso es Ethernet. Acá se deberá formar la correspondiente trama que el host *n11-client* enviará por su interfaz llamada `eth0`, cuya configuración podemos ver en la figura 7.3. Para esto el host *n11-client* necesitará conocer la dirección MAC del dispositivo al cual le tiene que enviar esa trama. Es en este momento donde se recurre al protocolo ARP para averiguar esa dirección. Obviamente, esto será necesario si es que el host no tiene esa información de algún intercambio anterior en su tabla ARP. Asumiendo que esto último es cierto, es que se analizará cómo se realiza este procedimiento. Si el host ya tiene esa información no sería necesario utilizar ARP.

De acuerdo a la configuración que el host *n11-client* aprendió mediante DHCP, su servidor de DNS es el host llamado *n18-dns-local-resolver*, que tiene la dirección IP 40.0.0.53, lo que ubica a ambos hosts en redes diferentes. Esto nos lleva a la siguiente pregunta, ¿podemos averiguar la dirección MAC

```

root@n11-client:/tmp/pycore.35177/n11-client.conf# ip addr show eth0
47: eth0@if48: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:00:00:aa:00:11 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.20/24 brd 10.0.0.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:feaa:11/64 scope link
        valid_lft forever preferred_lft forever

```

Figura 7.3: Configuración interface eth0 de *n11client*

```

root@n11-client:/tmp/pycore.35177/n11-client.conf# ip route show
default via 10.0.0.1 dev eth0
10.0.0.0/24 dev eth0 proto kernel scope link src 10.0.0.20
root@n11-client:/tmp/pycore.35177/n11-client.conf# █

```

Figura 7.4: Tabla ruteo del host *n11client*

asociada a esa dirección IP?. Como se explicó anteriormente, esto no es posible debido a que ambos hosts se encuentran en redes diferentes. Entonces, ¿qué dirección MAC debe aprender el host *n11-client* para terminar de armar la trama? La respuesta es simple, la de su default gateway. Entre la información aprendida mediante DHCP, el host también recibirá cual es su default gateway, el router al que le deberá enviar toda la información que no vaya dirigida a hosts dentro de su misma red. Esto lo puede saber consultando su propia tabla de ruteo, que se puede observar en la imagen 7.4, y que tiene solo dos entradas (podría tener más de dos). La línea que empieza con default es la que nos indica la ruta default y contiene la dirección IP, 10.0.0.1, correspondiente al default gateway. En el gráfico de la red esa dirección se corresponde con la interface eth1 del router *n10*, cuya configuración se puede ver en la figura 7.5. Es por la dirección MAC asociada a esta dirección IP que se debe realizar la consulta.

Una vez que se sabe que dirección MAC se debe consultar, comienza el proceso. Obviamente, no se puede consultar directamente a un host con una dirección IP asignada cuál es su dirección MAC, esta información es la que se está intentando determinar. Como el host no sabe a quién consultar para obtener lo que necesita, consulta a todos los hosts en su red mediante el envío de un mensaje de tipo broadcast. Todos los hosts en su misma red recibirán ese mensaje, inclusive el router en la interface conectada a esa red, quien no lo retransmitirá hacia otras redes. Ese mensaje es un ARP Request que se puede ver en figura 7.6.

Como se explicó anteriormente, estos mensajes viajan en tramas Ethernet de tipo broadcast, que se ven en el campo *Destination* de la trama. La MAC origen, campo *Source*, corresponde al host *n11-client* que es quien hizo el envío (comparar este dato con el del ether/link de la figura 7.3). También, es posible deducir que se está utilizando el estándar Ethernet II y no el IEEE 802.3 porque tiene el campo *Type* que, en este caso, indica que se está transportando un paquete ARP. A continuación, se puede ver el contenido de este paquete, particularmente los campos *Sender MAC* y *Sender IP*, que contienen

```

root@n10:/tmp/pycore.35177/n10.conf# ip addr show dev eth1
45: eth1@if46: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:00:00:aa:00:10 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.1/24 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:feaa:10/64 scope link
        valid_lft forever preferred_lft forever

```

Figura 7.5: Configuración interface eth1 del router *n10*

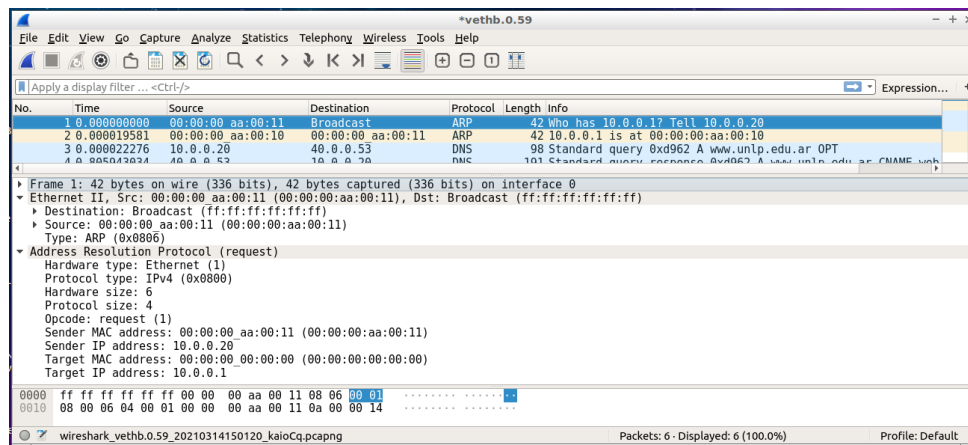


Figura 7.6: ARP Request

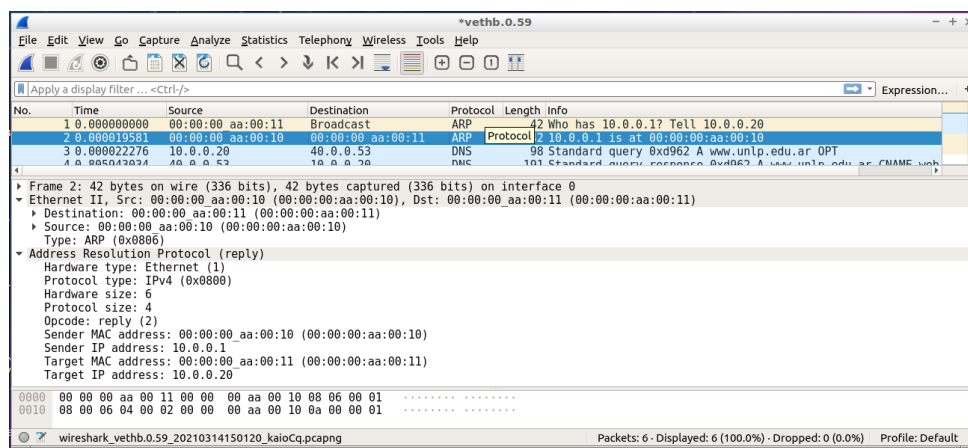


Figura 7.7: ARP Reply

las direcciones MAC e IP respectivamente del host que envió el mensaje, y *Target MAC* y *Target IP*. *Target MAC* contiene todos 0 (ceros) debido a que es la información que se está averiguando y *Target IP* la dirección IP del nodo del cual intentamos conocer su dirección MAC.

Después que la trama broadcast fue enviada será recibida por el switch *s20* que, al ver que es una trama de tipo broadcast, la reenviará por todos sus puertos activos menos por el que la recibió. Todos los hosts dentro de la red recibirán esta trama y la procesarán, pero solo responderá aquel que tenga configurada la dirección IP igual a la que está en el campo *Target IP*. En nuestro ejemplo, ese nodo será el router *n10* que tiene configurada esa IP en su interface *eth1* (ver la figura 7.5). Este responderá con un mensaje de tipo ARP Reply, que podemos ver en la figura 7.7, y que, a diferencia del ARP Request, es de tipo unicast. Esto implica que la respuesta solo será recibida por el nodo cuya dirección MAC se encuentra el campo *Destination* en la trama Ethernet.

Al observar el paquete ARP Reply, y compararlo con el correspondiente ARP Request, se puede ver que los valores de los campos *Sender* y *Target* están invertidos y todos completos. En el campo *Sender MAC* viaja la dirección MAC que hace falta para completar la trama Ethernet. A partir de este momento, el host *n11-client* tiene toda la información que necesita para enviar la trama correspondiente, que en este caso llevará un mensaje DNS Query como datos de la capa de aplicación.

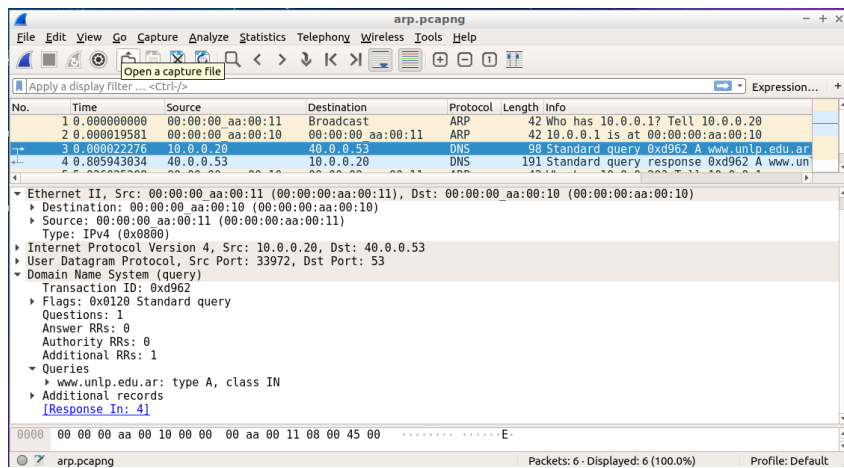


Figura 7.8: Mensaje DNS Query

Para hacer más eficiente el proceso, el router *n10* aprende la dirección MAC del host *n11-client* del ARP Request que recibió de éste. Esto evita que si *n10* le tiene que responder al host *n11-client* deba ejecutar el proceso ARP nuevamente. Además, la información aprendida se mantiene en una tabla ARP por un tiempo determinado, que es configurable. Cada vez que se referencia una entrada en esa tabla su tiempo de vida asociado se vuelve a regenerar. Si una entrada no es referenciada por un determinado tiempo, se elimina de la tabla. Si esto sucede, al volver a necesitar esa dirección MAC se deberá ejecutar nuevamente el proceso ARP. Es posible definir una entrada en la tabla de manera permanente.

Un dispositivo con más de una interfaz, como el caso de un router, tendrá una tabla ARP por cada interfaz que tenga IP configurado)

Es importante comprender que, en este ejemplo, la dirección MAC Destino de la trama Ethernet que transporta el DNS Query no indica el mismo destino que la dirección IP Destino del paquete IP. Esto se debe a que, si bien el paquete IP va destinado al host *n18-dns-local-resolver*, la trama Ethernet va dirigido a la interfaz *eth1* del router *n10*. Si se observa la figura 7.8 se puede ver que la dirección MAC destino de la trama Ethernet indica la interfaz *eth1* del router *n10*, pero la dirección IP destino del paquete IP referencia al host *n18-dns-local-resolver*.

Una vez que la trama llega al router *n10* y éste la acepta, es destruida y el contenido de su campo Datos, que es un paquete IP, se pasa a la capa correspondiente. El router hará lo que sabe hacer, rutear el paquete comparando la dirección IP Destino del paquete contra su tabla de ruteo. En nuestro ejemplo, reenviará el paquete por la interfaz *eth0*, por lo tanto, pasará ese paquete a la capa de enlace que, al ser también una red de tipo Ethernet, deberá ejecutar nuevamente el proceso ARP si es que no tiene la información necesaria en la tabla ARP. En resumen, el proceso ARP se ejecutará por cada red Ethernet que atraviese el mensaje entre un origen y un destino, siempre que no tenga la información necesaria para terminar de armar la correspondiente trama Ethernet.

De manera similar, si desde el host *n11-client*, con dirección IP 10.0.0.20, se quiere acceder a un host vecino, digamos que se envía un mensaje ICMP con el comando `ping(8)` a la dirección IP

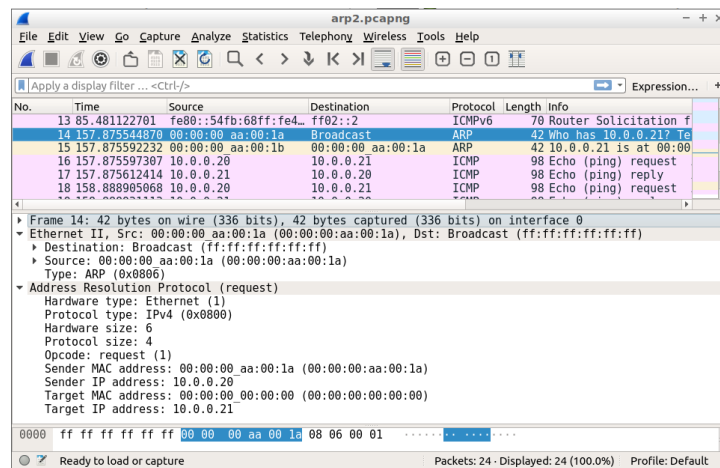


Figura 7.9: ARP Request - Host origen y destino en la misma LAN *n11client*

10.0.0.21 del host *n21-client*, también sería necesario ejecutar el proceso ARP. En este caso, el ARP Request no consultará por la dirección MAC de su default-gateway sino que lo hará por la dirección MAC correspondiente a la dirección IP que tiene *n21-client* asignada, que es la dirección que se indicó en el comando ping, lo que se puede ver en la figura 7.9. Y, a diferencia del ejemplo anterior, cuando se forme la trama Ethernet para enviar el mensaje del ping, tanto la dirección MAC Destino de la trama como la dirección IP Destino dentro del paquete IP harán referencia al mismo host destino.

Switching

Por último, para terminar de entender todo el proceso detallado hasta el momento, es necesario conocer el trabajo que realiza un switch y, en este caso, el que realiza en particular el switch *s20*. A medida que las tramas pasan por él, un switch, a diferencia de un hub que trabaja en la capa física, va aprendiendo donde se encuentran cada uno de los hosts de su red local, lo que le permitirá decidir que puerto debe utilizar para reenviar una trama dirigida a un host determinado, pero no aprenderá nada de los dispositivos fuera de la red en la que se encuentra. Esto hace que tenga una mínima inteligencia para decidir por donde reenviar una trama en forma eficiente. La información necesaria para tomar esas decisiones la almacenará en una tabla CAM, Content Addressable Memory, que se irá actualizando dinámicamente.

Usando el intercambio de mensajes ARP del ejemplo visto y asumiendo que tiene su tabla CAM vacía cuando reciba el ARP Request enviado por el host *n11-client*, el switch *s20* hará dos cosas: por un lado, agregará en su tabla CAM una nueva entrada en la que indicará que la MAC Origen de la trama, la perteneciente al host *n11-client*, se encuentra en el puerto e0 del switch; por el otro, reenviará la trama, que al ser de tipo broadcast la reenviará por todos los puertos menos por el que la recibió. Luego, cuando reciba el ARP Reply proveniente del router *n10* procederá de la misma manera: agregará una entrada en la tabla CAM que indique que la MAC Origen de la trama, la perteneciente a la interfaz eth1 del router *n10*, se encuentra en el puerto e2. Y, para reenviarla, como la trama es de tipo unicast lo que hará será comparar la dirección MAC Destino de la trama contra la tabla CAM. En este caso esa dirección coincidirá con una de las entrada de la tabla, la que le indicará que la trama debe ser

reenviada únicamente por el puerto e0 del switch (un hub no tiene esta tabla por lo que todas las tramas que recibe las retransmite por todos los puertos menos por el que la recibió). Si no se encuentra una coincidencia en la tabla CAM, aunque la dirección destino de la trama sea de tipo unicast, la reenviará por todos los puertos menos por el que la recibió, como un hub. En la tabla 7.1 se puede ver como quedaría la tabla CAM del switch *s20* luego del intercambio descrito.

Cuadro 7.1: Tabla CAM del switch *s20*

Puerto	MAC
e0	00:00:00:aa:00:11
e2	00:00:00:aa:00:10

REFERENCIAS

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

[SZ14] Charles E. Spurgeon y Joann Zimmerman. «Ethernet: The definitive guide». OReilly, 2014.
ISBN: 9780321336316.

LOS AUTORES

Luis Marrone

Ingeniero Electromecánico orientación electrónica. UBA. Diploma de Honor.

Profesor titular en Redes de Datos II. Profesor y coordinador de la Maestría en Redes de Datos de la Facultad de Informática de la UNLP. Profesor del Doctorado en Informática de la UNLP.

Investigador full time en el LINTI, Facultad de Informática de la UNLP en temas afines con protocolos , análisis de performance, modelos de tráfico y redes de sensores.

Publicaciones en congresos y revistas de alcance nacional e internacional. Autor de libros de Redes y Arquitectura de computadores.

Matías Robles

Licenciado en Informática, Facultad de Informática, UNLP. Magister en Redes de Datos, Facultad de Informática, UNLP. Jefe de Trabajos Práctica en Redes y Comunicaciones de la carrera Licenciatura en Sistemas, Facultad de Informática, UNLP. Integrante del staff de docentes de la Maestría en Redes de Datos y de la Especialidad de Redes de Datos y Seguridad de la Facultad de Informática, UNLP.

Néstor Castro

Ingeniero Electricista. Universidad Tecnológica Nacional (U.T.N.) Profesor Adjunto en Arquitectura de Computadoras, Organización de Computadoras y Tecnología, Ambiente y Sociedad de la Licenciatura en Informática y Licenciatura en Sistemas de la Facultad de Informática de la Universidad Nacional de La Plata (U.N.L.P.). Integrante del staff de docentes de la Maestría en Redes de Datos y de la Especialidad de Redes de Datos y Seguridad de la Facultad de Informática, U.N.L.P. Investigador del Laboratorio de Investigación en Nuevas Tecnologías Informáticas (LINTI) de la Facultad de Informática de la U.N.L.P. en las temáticas de Internet de las Cosas (IoT) y Smart Cities. Actividad extensionista en temáticas de energías renovables y residuos de aparatos eléctricos y electrónicos (RAEE) en proyectos de la Facultad de Informática de la U.N.L.P. Publicación: - “Las inundaciones en La Plata, Berisso y Ensenada: análisis de riesgo, estrategias de intervención. Hacia la construcción de un observatorio ambiental”. Capítulo 9. 2021. (EDULP). ISBN: 978-987-8475-08-0.

Andrés Barbieri

Magister en Redes de Datos de la Facultad de Informática de la UNLP. Docente en las cátedras de Redes y Comunicaciones y Redes II de la Facultad de Informática en la Universidad Nacional de La Plata.

Paradigma TCP-IP / Luis Marrone ... [et al.] ; coordinación general de Luis Marrone. -
1a ed. - La Plata : Universidad Nacional de La Plata ; EDULP, 2023.
Libro digital, PDF - (Libros de cátedra)

Archivo Digital: descarga
ISBN 978-950-34-2249-6

1. Tecnología Informática. I. Marrone, Luis, coord.
CDD 004.02

Diseño de tapa: Dirección de Comunicación Visual de la UNLP

Universidad Nacional de La Plata – Editorial de la Universidad de La Plata
48 N.º 551-599 / La Plata B1900AMX / Buenos Aires, Argentina
+54 221 644 7150
edulp.editorial@gmail.com
www.editorial.unlp.edu.ar

Edulp integra la Red de Editoriales Universitarias Nacionales (REUN)

Primera edición, 2023
ISBN 978-950-34-2249-6
© 2023 - Edulp

e
exactas

The logo for Edulp, featuring a stylized oak leaf above the word "Edulp" in a serif font, with "EDITORIAL DE LA UNLP" in a smaller sans-serif font below it.



UNIVERSIDAD
NACIONAL
DE LA PLATA