Tema 3. Internet. Estructura y protocolos

1. Interconexión de redes. Internet.

La interconexión de redes de diferentes tecnologías requiere de una capa nueva de abstracción llamada capa de red. Desde la década de 1980 empezaron a surgir diferentes modelos que permitían la interconexión de diferentes redes en una sola, aunque previamente ya se podían interconectar algunas redes o algunos computadores mediante líneas alquiladas básicamente telefónicas. El modelo TCP/IP fue desarrollado en 1973 para la red ARPANET y se convirtió en la base de la existencia de Internet, ya que todas las redes que se iban conectando a la primera ARPANET debían utilizar esa familia de protocolos.

No obstante, la interconexión de redes requiere varias cuestiones a tener en cuenta dada la diversidad de parámetros existentes como:

- Esquemas de direccionamiento
- Tamaños máximos de paquetes
- Valores de temporizadores
- Mecanismos de acceso
- Controles de flujo

Para automatizar y unificar la resolución de los problemas que la interconexión de redes conlleva, se comenzó la fabricación de diferentes dispositivos para cada tipo de red, que resuelven parte de los problemas de interconexión, además de diferentes implementaciones de software desde las capas de firmware a las de aplicación. Estos dispositivos llamados genéricamente gateways o pasarelas tenían (y tienen) que adaptar las tecnologías de las redes para las que fueron diseñados a las características del protocolo IP. Un ejemplo de la funcionalidad de estos dispositivos se puede ver en la figura 1, donde se interconectan a través de una red IP dos segmentos de tecnología Ethernet 802.3.

Otra de las cuestiones importantes que debía resolver la interconexión de redes era la identificación de los tipos de transporte de datos que podían implementar, dado que las diferentes redes y servicios que empezaban a interconectarse no tenían un objetivo común. Se definieron por tanto dos tipos de servicios básicos, que se implementan no sólo en la interconexión de redes sino también en las propias tecnologías de acceso de cada una de las redes interconectadas y en los enlaces entre los dispositivos intermedios que permiten la interconexión:

- Servicios orientados a conexión: este tipo de servicios prioriza la integridad y seguridad de los datos sobre la velocidad. Realizan una conexión identificada sobre la cual establecen sistemas de control del flujo, control de errores y de pérdidas de paquetes. En las capas de red se establecen los llamados *Circuitos Virtuales*, donde los paquetes van siempre a través del mismo camino y en el mismo orden. En redes de conmutación de circuitos, las capas de enlace y de red ya proveen por definición de este tipo de conmutación estos servicios. La capa de red en la que se basa Internet con el protocolo IP no provee este tipo de servicios, y se cede a la capa de transporte con el protocolo TCP esa función.
- Servicios sin conexión: este tipo de servicios prioriza la velocidad sobre la integridad y seguridad del transporte. Básicamente no establecen ningún tipo de circuito entre los extremos y dejan a las capas superiores las labores (si es que las consideran necesarias) de control y de integridad de las transmisiones. Suelen utilizarse para transmisiones con altos requerimientos de tiempo real. El protocolo IP provee este tipo de servicio, y es apoyado por el protocolo de transporte UDP para la identificación de las aplicaciones que lo utilizan.

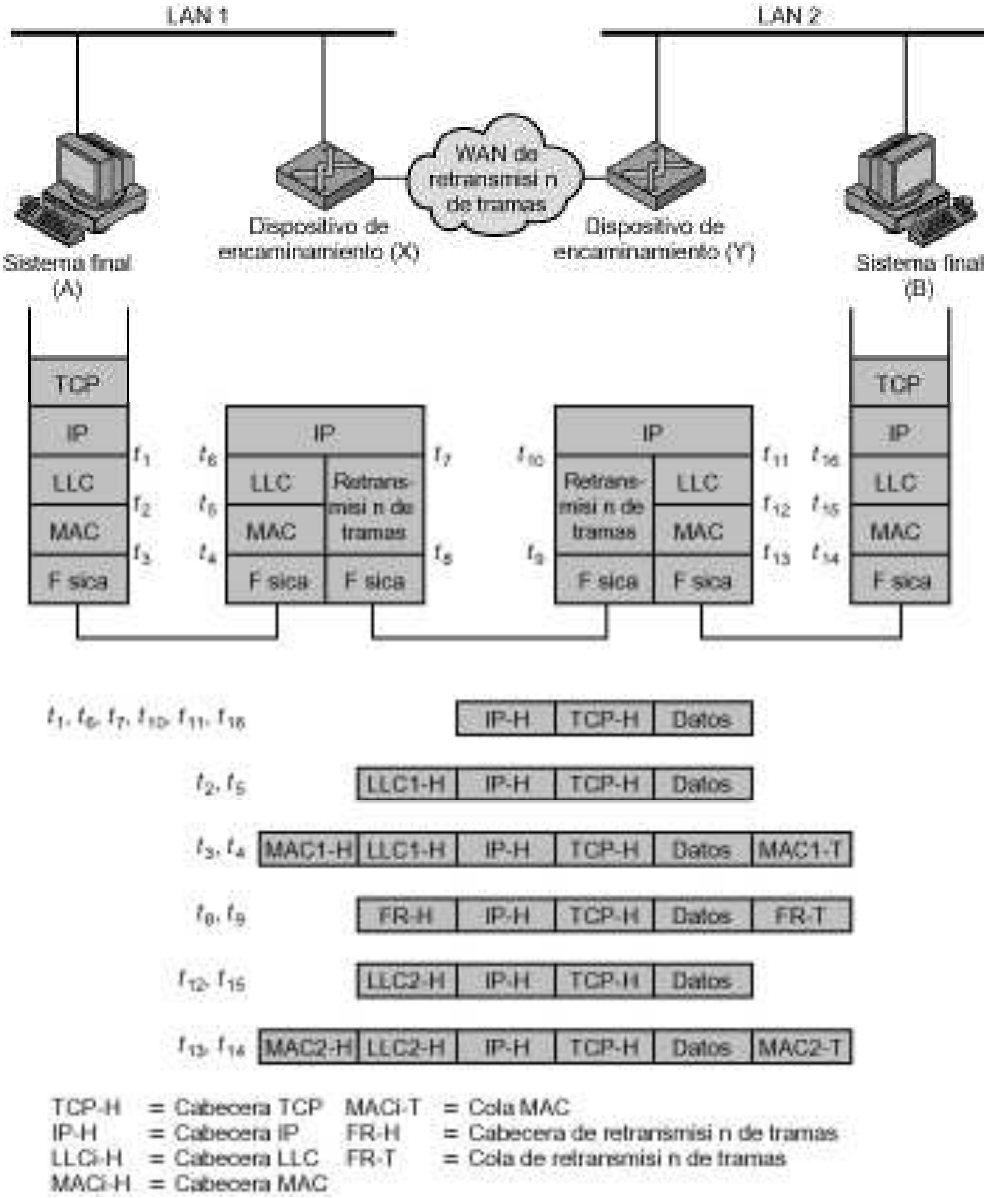

Figura 1: Dispositivos de interconexión

2. Estructura de Internet

Internet es una red de redes de la cual forman parte millones de dispositivos conectados que ejecutan aplicaciones de red utilizando protocolos. La interconexión de las redes que forman parte de Internet se realiza mediante un protocolo común llamado protocolo de Internet o Internet Protocol (IP), que permite crear un sistema de direcciones único y una estructura de PDU de red unificada y reconocida por todos los miembros de la red.

De forma simplificada, además del protocolo IP, puede considerarse que Internet lo componen tres tipos de elementos físicos:

- Los dispositivos terminales llamados **hosts**.
- Los **enlaces** de comunicaciones, que en algunos casos son también redes de diferentes tecnologías como ATM o Ethernet.
- Los dispositivos intermedios, llamados **routers** que permiten enrutar los paquetes hacia su destino final. Los routers interconectan redes entre sí. Existen los dispositivos terminales de red, llamados pasarelas o gateways, que adaptan las direcciones y los formatos de los paquetes entre el formato de Internet y las redes terminales donde se ubican los hosts. A estos gateways se les conoce popularmente también como routers.

De forma concreta, puede decirse que: "... un dispositivo está en Internet si dispone de la pila de protocolos TCP/IP (y en particular el protocolo IP) y puede ser visible mediante una dirección IP pública y única."

La organización general de Internet es muy heterogénea, aunque para la mayoría de las situaciones pueden definirse otros tres componentes fundamentales:

- Los dispositivos terminales llamados **hosts**, que ejecutan aplicaciones de red que permiten comunicarse extremo a extremo con otros hosts de Internet.
- Las **redes de acceso** cuyas tecnologías pueden ser muy diversas, destacándose XDSL, CaTV o Wifi/WiMax/LMDS. Estas redes son las también llamadas de *última milla*, y permiten la conexión de los hosts o redes terminales al núcleo de Internet.
- El **núcleo** de Internet. El núcleo es un conjunto de routers interconectados que realizan el transporte de datos entre las diferentes redes que componen Internet. El núcleo es en sí mismo una red heterogénea con enlaces entre los routers de diferentes tecnologías como ATM, o enlaces por satélite, cuyo punto en común es la capacidad de enrutamiento a nivel 3 de la arquitectura TCP/IP.

2.1. Hosts finales

Son los dispositivos en los que generalmente comienza y finaliza el transporte datos. Pueden ser dispositivos de usuario (como teléfonos móviles, teléfonos IP o computadoras personales), dispositivos esclavos (como elementos domóticos, robots o autómatas) o dispositivos servidores (como servidores de correo o de web). En todos ellos deben correr aplicaciones para gestionar los datos recibidos y/o enviados. Los datos son el contenido final de los paquetes que circulan por Internet.

La transmisión de los datos extremo a extremo responde a dos modelos básicos: el modelo **cliente/-servidor** y el modelo **peer to peer**.

En el modelo **cliente/servidor** un host cliente hace una solicitud y recibe un servicio proporcionado por un host servidor. Existen numerosos servicios de Internet que manejan este modelo de transmisión, algunos imprescindibles para el funcionamiento de la red, como el servicio de nombres de dominio o DNS; otros servicios son el servicio web (en general utilizando HTTP) o el servicio de transmisión de ficheros o FTP. Cuando el host servidor tiene que dar servicio a multitud de clientes, se suelen utilizar balanceadores de carga o *load-balancers*. Estos *load balancers* son dispositivos hardware o software que reparten la carga de los servicios en varios servidores, a pesar de que el cliente tenga la sensación de que siempre es uno (figura 3). Se pueden destacar varias técnicas de load balancing:

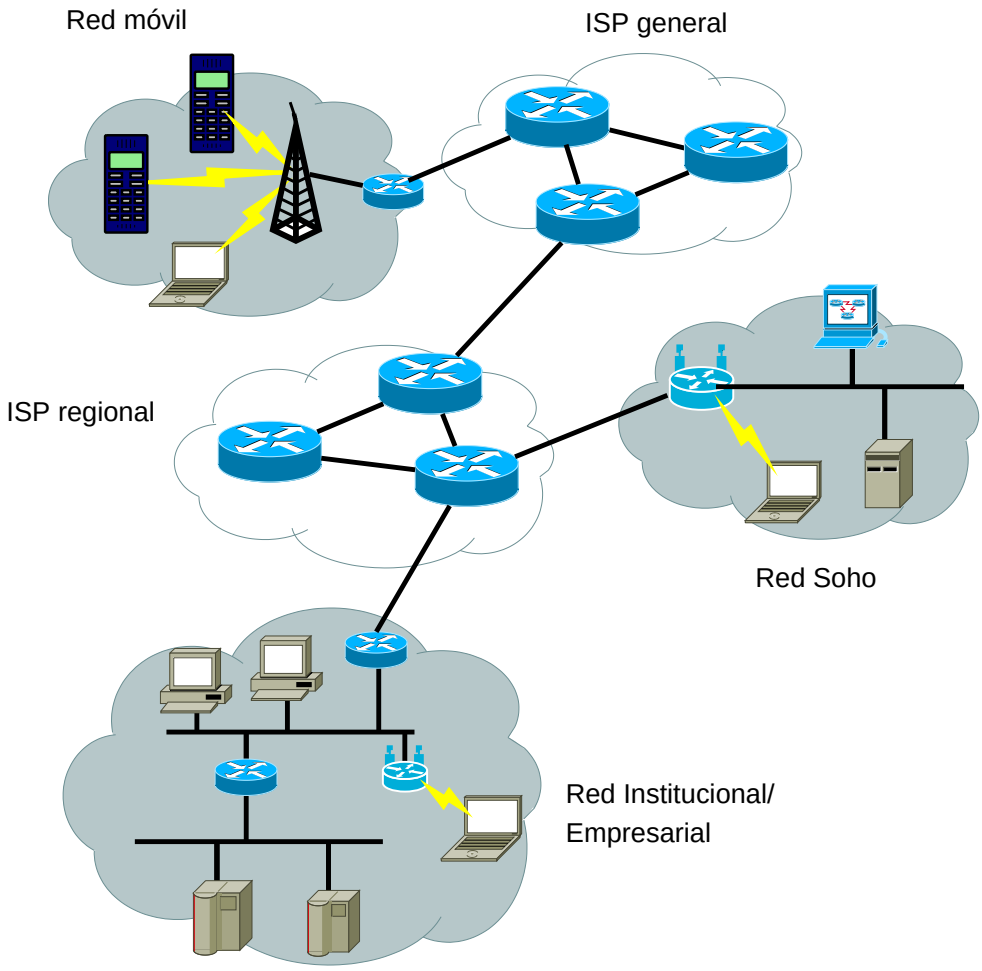

Figura 2: Estructura general

Round-robin DNS es una técnica por la cual un nombre DNS tiene asignadas varias IP's, y resuelve con una u otra en función de diferentes parámetros, como la IP origen.

Algoritmias de repartición el dispositivo load balancer reparte la carga entre diferentes servidores en una granja de servidores en función de algoritmias de ocupación, tiempo, aleatoriedad, etc.

Criterios de arquitectura el dispositivo load balancer reparte diferentes tareas entre diferentes equipos servidores. Un ejemplo puede ser el módulo "mod_proxy" de Apache, que divide consultas HTTP y HTTPS, puede utilizar hosting virtual, y demás funcionalidades.

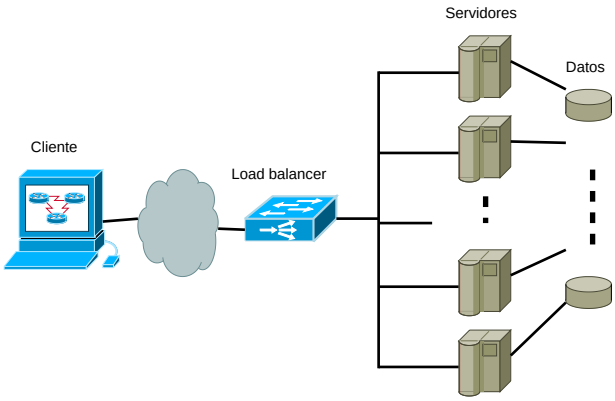

Figura 3: Modelo cliente/servidor con *load-balancer*

En el modelo **peer to peer** un host final transfiere datos directa o indirectamente a otro host final. En este modelo, el uso de servidores es residual sirviendo fundamentalmente para la localización de ambos extremos; como ejemplos sirven Skype o BitTorrent. Cuando se forman redes de transmisión entre los hosts finales se habla de redes superpuestas u Overlay Networks, como TOR. En la figura 4 se muestran los diferentes modelos.

Existen modelo mixtos, donde un servidor permite la interconexión de hosts finales y el servicio proporciona la propia plataforma de interconexión. Un ejemplo típico es el de las redes sociales sobre plataforma única, como Facebook o Tuenti. En general, se utiliza el protocolo HTTP como plataforma.

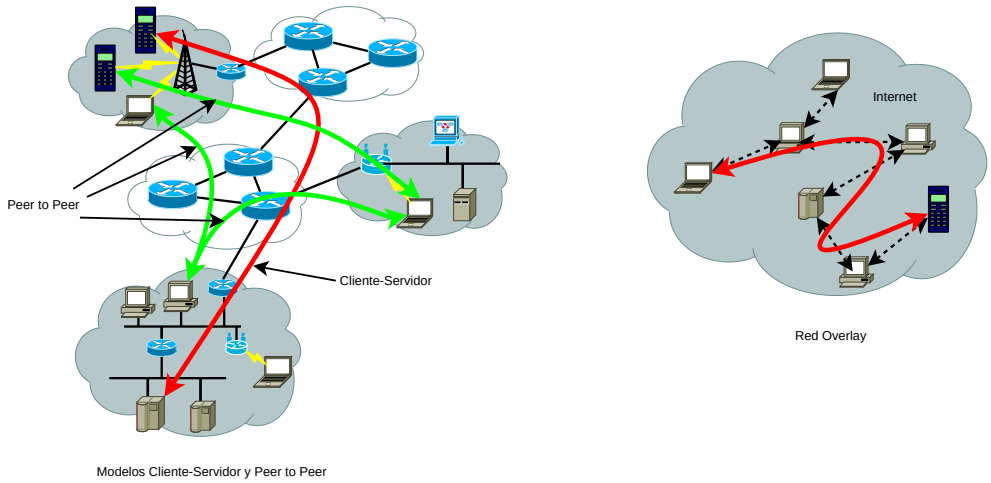

Figura 4: Modelos de transporte entre hosts finales. Cliente/servidor, Peer to peer, Red Overlay.

2.2. Redes de acceso

También llamadas de última milla, son las que proporcionan el enlace entre el host terminal y el primer primer router de Internet integrado en el núcleo. En ellas hay que prestar atención a las prestaciones de calidad de servicio, la capacidad en ambos sentidos de la comunicación y si son dedicadas o nó.

Las redes de acceso pueden ser diferentes en función de la tipología del host terminal:

- Redes de acceso residenciales
- Redes de acceso institucionales
- Redes de acceso móviles

2.3. Núcleo

El núcleo de la red es una malla de routers que permiten la interconexión entre las diferentes redes que componen Internet. Los routers funcionan en el nivel 3 IP de la arquitectura TCP/IP, y sus tecnologías de interconexión pueden ser muy variadas, desde Ethernet, ATM hasta conexiones vía satélite. Estas tecnologías pueden dividirse en tecnologías de conmutación de circuitos, tecnologías de conmutación de paquetes o tecnologías mixtas llamadas de circuito virtual.

Tecnologías de conmutación de circuitos : el transporte de datos se realiza a través de circuitos dedicados donde la capacidad de los canales está completamente reservada para enlaces entre dispositivos. En estas tecnologías se reservan los recursos de la transmisión extremo a extremo, y la capacidad total del canal de comunicaciones es la del dispositivo o enlace con menor capacidad. En la conmutación de circuitos, el rendimiento puede garantizarse, pues los recursos del canal no se comparten con otros tráficos de datos y se reservan con técnicas de multiplexación. En el núcleo de Internet, algunos enlaces entre routers usan estas tecnologías.

Tecnologías de conmutación de paquetes : el transporte de datos se realiza dividiendo los datos en paquetes donde cada uno de ellos puede utilizar canales diferentes de comunicación. En la conmutación de paquetes, cada paquete usa toda la capacidad del enlace, aunque los recursos se utilizan bajo necesidad, por lo que se puede producir congestión si es mayor la demanda que la capacidad. Para la gestión del uso de los recursos, los routers utilizan colas de transmisión y necesitan procesar cada paquete de datos por separado. Los paquetes se mueven por saltos de dispositivo a dispositivo (router a router) y cada nodo (o router) recibe el paquete antes de reenviarlo. En la conmutación de paquetes se habla de multiplexación estadística, pues la secuencia de transmisión de paquetes entre hosts no tiene una organización temporal.

Tecnologías de conmutación de circuito virtual : el transporte de datos se realiza mediante conmutación de paquetes, pero con reserva de recursos mediante paquetes previos de establecimiento de conexión. Una vez que la conexión entre origen y destino está establecida, los encaminadores origen de la conexión asignan un identificador llamado "Identificador de Circuito Virtual" (o VCI por sus siglas en inglés) a cada paquete de esa conexión y los encaminadores intermedios reconocen ese VCI (que puede variar en cada enlace). Cuando un paquete con un VCI llega a un encaminador intermedio, este ya lo reenvía por la salida correspondiente asignada en el proceso de establecimiento de la conexión, por lo que no necesitan todo el campo de direcciones para dirigir el paquete a su siguiente salto. Este proceso combina las ventajas de la conmutación de paquetes y la conmutación de circuitos y es utilizado por tecnologías muy eficientes como ATM o MPLS.

La estructura del núcleo de Internet es jerárquica, con proveedores de tráfico llamados ISP (Internet Service Providers) que son a su vez redes de conmutadores, enlaces y routers y disponen en general de pocos dispositivos enrutadores pero con enlaces a grandes distancias. Existen ISP de 3 niveles, en función del ámbito que abarcan, aunque no existe una organización centralizada que asigne esos niveles. La figura 5 muestra esquemáticamente los ISP de diferentes niveles.

Los diferentes ISP intercambian tráfico entre sí en lugares físicos llamados **IXP** (Internet eXchange Point), **PoP** (Point of Presence) o **puntos neutros**, donde instalan sus routers para intercambiar rutas con los routers de otros ISP. En general, el intercambio de rutas se realiza mediante el protocolo BGP (Border Gateway Protocol). Cuando dos ISP se conectan entre sí se dice que son igualitarios.

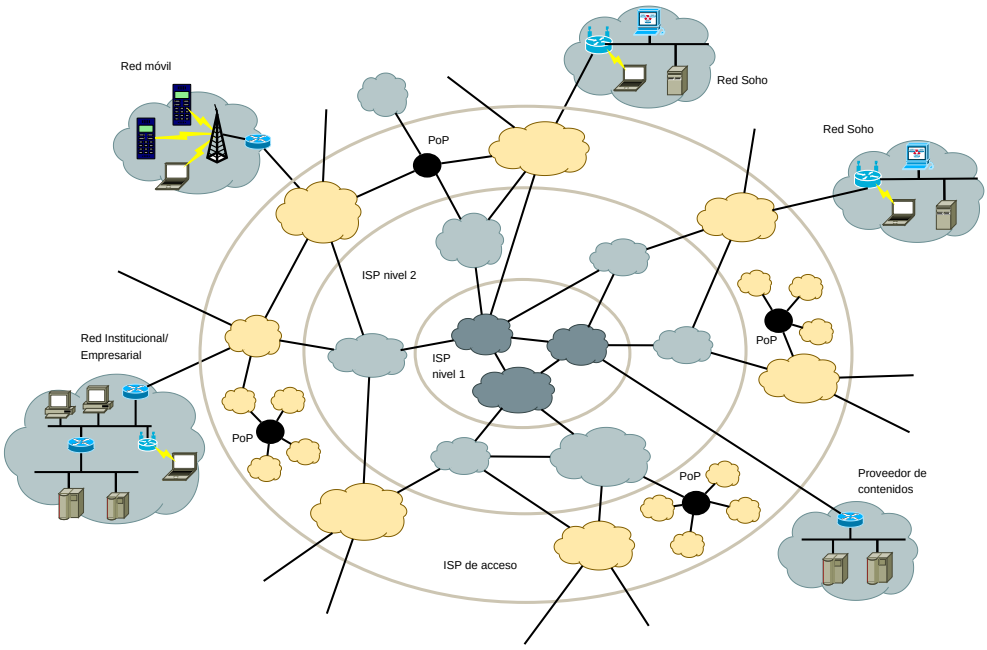

Figura 5: Estructura jerárquica de Internet

2.3.1. ISP de nivel 1 (Tier 1 Network)

Estos proveedores tienen coberturas nacional o continental. Las redes que los forman se llaman usualmente **redes troncales de Internet**. Los ISP de nivel 1 se conectan todos entre sí con uno o varios enlaces, y las velocidades de cada enlace están entre los 622 Mbps hasta los 10 Gbps, por lo que sus enrutadores son de muy altas prestaciones. En algunas ocasiones, también dan servicio a organizaciones finales o a proveedores de contenidos, pero en general, dan servicio de transporte de datos a los ISP de nivel 2. Estos ISP tienen routers conectados a numerosos PoP dispersos entre las localizaciones geográficas que abarcan. Algunos proveedores de nivel 1 son: Level 3 Communications, Deutsche Telecom, Telefónica, Orange, Verizon, AT-T.

2.3.2. ISP de nivel 2 (Tier 2 Network)

Los ISP de nivel 2 tienen cobertura regional o nacional y su conexión al resto de Internet se realiza a través de los ISP de nivel 1, aunque en algunos casos también se intercambian tráfico directamente entre sí. Se dice que un ISP de nivel 2 es un cliente de un ISP de nivel 1, y que un ISP de nivel 1 es un proveedor de un ISP de nivel 2. Los ISP de nivel 1 cobran unas tasas a sus ISP cliente, o a los proveedores de contenido que están directamente conectados a él, y las tasas dependen de las velocidades y servicios de conexión que les ofrecen. Algunos ISP de nivel 2 son: Vodafone, Easynet, British Telecom, Virgin Media, Tele2.

2.3.3. ISP de nivel 3 (Tier 3 Network)

Por debajo de los ISP de nivel 2 se encuentran los ISP de nivel 3, que también suelen ser los proveedores a los usuarios u organizaciones finales y son llamados **ISP de acceso**. Los ISP de nivel 3 son clientes de los ISP de nivel 2, aunque pueden ser también clientes directamente de los ISP de nivel 1. También existe la posibilidad de que algunos ISP de nivel 3 se encuentren conectados entre sí, sin haber pasado por los ISP de nivel 2. Suelen tener ámbito local y disponen de tecnologías de acceso a los usuarios finales, a los que conectan mediante sus routers al resto de Internet. Estos ISP cobran de los usuarios finales por tiempo y por capacidad de transmisión contratada. La empresa gallega del cable R es un ejemplo de ISP de nivel 3.

3. Retardos y pérdidas

Cuando un paquete viaja a través de la red entre diferentes dispositivos, tarda un tiempo en llegar a su destino. A este tiempo se le llama retardo y se suele representar de diferentes formas, aunque aquí se representará por la letra griega tau (τ) o eventualmente por la letra d (del inglés *delay*). El retardo es uno de los parámetros mas importantes de una red o un enlace, y suele determinar la operatividad de la misma para determinados tipos de tráfico.

Las causas de los retardos pueden ser muy variadas, aunque se pueden identificar varios orígenes que definirán unos tipos de retardos determinados. Los mas importantes son el retardo de procesamiento de nodo (τ_{proc}), el retardo de cola (τ_q), el retardo de propagación (τ_p) y el retardo de transmisión (τ_t). La figura 6 intenta reproducir los orígenes de cada uno de ellos.

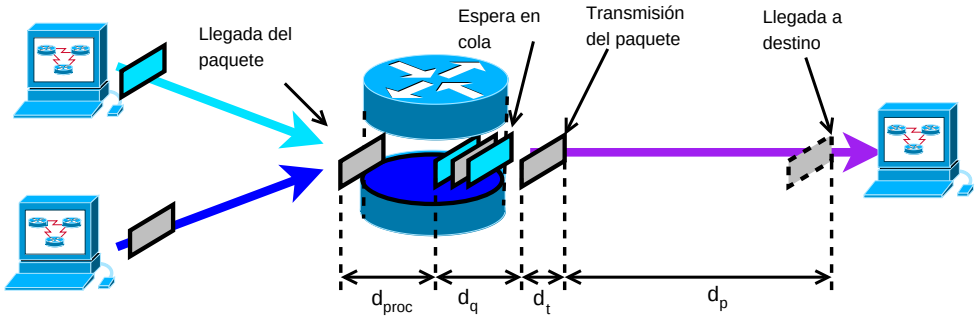

Figura 6: Retardos de un enlace a través de un router

3.1. Retardo de procesamiento

Cuando un paquete llega a un router de Internet, este tiene que identificar y procesar los campos de su cabecera IP, para entre otras cosas determinar su destino, si tiene errores o si el paquete lleva demasiados saltos entre routers. Este retardo es muy dificil de determinar mediante formulación matemática y depende de las capacidades de procesamiento del dispositivo intermedio así como de la complejidad del análisis y las acciones a tomar por parte del router. No obstante, en general suele ser el menor de todos los retardos involucrados y está en el orden de algunos microsegundos.

3.2. Retardo de cola

Este es uno de los retardos mas importantes en la transmisión de paquetes a través de routers. En las redes de conmutación de paquetes, el medio es compartido por muchos tipos de tráfico. En general, la forma de priorizar paquetes para un enlace de salida en un router suele llevar el principio FIFO (First In First Out), es decir, el primer paquete que llega es el primero que sale. Para los demás paquetes, existe una cola de salida donde van situándose los mismos esperando su turno de salida. El tiempo que un paquete está en la cola es el retardo de cola, y depende de la cantidad de tráfico que quiere salir por ese enlace, del tipo de cola y de la longitud de la misma. Cuando un paquete llega a una cola de salida que ya está llena, el router lo descarta produciéndose la **pérdida** del paquete.

Dado que el tamaño de una cola se mide en paquetes, el retardo máximo que se produce en un paquete en la cola de salida de un enlace de capacidad C tiene relación directa con el tamaño de dicha cola (representado como q_s en octetos) y con el tamaño de los paquetes precedentes, que se representa por p_s suponiendo que todos los paquetes precedentes tienen el mismo tamaño.

$$\tau_{q_{max}} = \frac{8p_s \cdot q_s}{C} \tag{1}$$

Esta fórmula mide básicamente el tiempo que necesitan todos los paquetes precedentes en ser transmitidos a través del enlace de salida de la cola para dejarla libre. El tiempo de transmisión de un paquete es el llamado retardo de transmisión, que se verá a continuación.

Las colas en un router tienen diferentes algoritmias de gestión, destacándose las colas de tipo Drop Tail mayoritarias en Internet, con una gestión tipo FIFO, y las de tipo RED (Random Early Detection), donde la gestión de la cola depende de un algoritmo de eliminación de paquetes aleatorio una vez que el llenado de la cola sobrepasa un límite determinado.

El retardo de cola suele estar entre varios microsegundos hasta varios milisegundos.

3.3. Retardo de transmisión

Este retardo mide el tiempo que tarda un paquete en ser depositado íntegramente en el enlace, y tiene una relación directa con la capacidad C del mismo. Ya que un paquete no ha sido transmitido hasta que se han depositado todos los bits que lo componen, el retardo de transmisión tiene también relación con el tamaño del paquete de datos, de la forma

$$\tau_t = \frac{8p_s}{C} \tag{2}$$

3.4. Retardo de propagación

Este retardo mide el tiempo que tarda cualquier bit del paquete de datos en recorrer la distancia del enlace. Al ser la transmisión física por ondas electromagnéticas, la velocidad de propagación coincide con la velocidad de la luz en el medio de transmisión.

Si la velocidad de la luz en el medio de transmisión se representa por c metros por segundo, y la longitud del enlace es de e metros, el retardo de propagación viene dado por

$$\tau_p = \frac{e}{c} \tag{3}$$

Este retardo suele ser insignificante con pocos enlaces de cortas distancias, pero se convierte en el mayor de todos los retardos en enlaces vía satélite, al ser las distancias del orden de decenas de miles de kilometros.

3.5. Retardo total de un paquete de datos

El retardo total de un paquete de datos a través de Internet es la suma de todos los retardos vistos anteriormente para cada uno de los enlaces que atraviesa el paquete. Si el número de enlaces es L, y l se utiliza como subíndice de un enlace en particular, el retardo total de un paquete sería:

$$\tau_{max} = \sum_{l=1}^{L} \left(\tau_{proc_l} + \tau_{p_l} + \tau_{q_l} + \tau_{t_l}\right) \tag{4}$$

$$\tau_{max} = \sum_{l=1}^{L} \left(\tau_{proc_l} + \frac{e_l}{c_l} + \frac{8p_s(q_{s_l}+1)}{C_l}\right) \tag{5}$$

Normalmente es casi imposible medir el retardo de un paquete a través de Internet con precisión, dado el conocido problema de la sincronización exacta de los extremos de la comunicación. En su lugar se mide el *Round Trip Time* (RTT), que es el tiempo que le lleva a un paquete ir al destino y volver al origen. Se suele estimar el retardo como la mitad del RTT, aunque evidentemente, este valor no tendría porqué ser exacto, por varias razones:

- Los caminos de ida y de vuelta de un paquete no tienen porqué atravesar los mismos routers ni enlaces.
- La existencia de numerosos enlaces asimétricos (diferente capacidad de un enlace en ambos sentidos de la comunicación) sobre todo en las redes de acceso.
- Diferente ocupación de las colas en los enlaces de ida y de vuelta.

4. Protocolo IP

El protocolo IP es un protocolo de conmutación de paquetes de datos que trabaja entre las capas de enlace y de transporte, por lo que no tiene en cuenta parámetros como el control de acceso al medio o el control del flujo de datos. Básicamente se encarga de disponer de un sistema de direccionamiento unificado y centralizado para toda la red y de disponer de mecanismos de enrutamiento y control de bucles. El protocolo IP recibe los datos de las capas inmediatamente superior o inferior y encapsula o desencapsula el paquete de datos con la cabecera IP mostrada en la figura 7.

El protocolo IP ha trascendido la propia naturaleza de Internet como red de redes y se utiliza de forma masiva también para la interconexión de dispositivos en redes privadas, aprovechándo los servicios y aplicaciones que a lo largo de los años se han ido desarrollando sobre él. Cuando una red privada utiliza el protocolo IP y los basados en él para la interconexión de sus dispositivos, se dice que trabaja como una Intranet.

Es necesario destacar que el protocolo IP no es el único protocolo de interconexión de redes, existiendo el protocolo IPX, aunque claramente en fase de desuso.

4.1. Cabecera

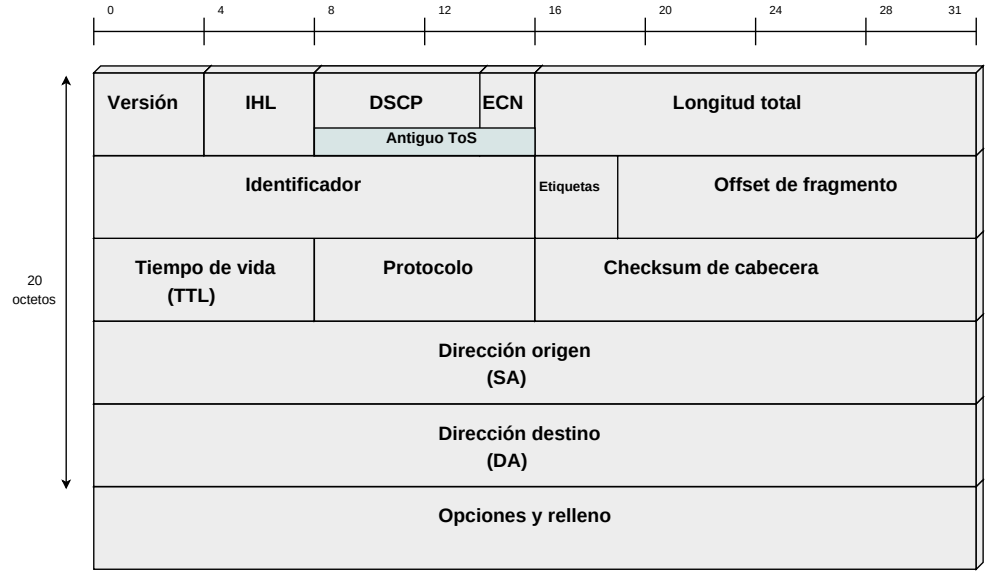

Figura 7: Cabecera del protocolo IP

Descripción de los campos:

- Versión (4 bits): Nº de versión del protocolo IP. Actualmente en la 4, y definiendo la 6.
- IHL, Longitud de cabecera de Internet (4 bits) expresada en bloques de 32 bits.
- Differentiated Services Code Point (DSCP) (6 bits). Nuevo campo (antes formaba parte junto con el nuevo campo ECN, del campo TOS - Type of Service, de 8 bits). Codifica diferentes tipos de servicios diferenciados. Definido en la RFC 2474.
- ECN, Explicit Congestion Notification (2 bits), también antes pertenecientes al campo TOS, al igual que el DSCP. Indica sin necesidad de perder paquetes la existencia de congestión en la red. Ambos extremos y los routers intermedios deben reconocerlo, pues es opcional. Se define en el RFC 3168.
- Longitud total (16 bits): longitud total del datagrama en octetos.

- Identificador (16 bits): nº de secuencia que identifica un datagrama antes de ser fragmentado. Junto con los campos dirección origen, destino y protocolo, se utiliza este campo para identificar los fragmentos de un datagrama, y junto con el campo "Fragment offset" podría re-ensamblar todos los fragmentos en un datagrama.
- Etiquetas (3 bits): 3 bits, bit 0 reservado (puesto siempre a 0), bit 1 DF fragmentación (1 no fragmentar), bit 2 - MF mas fragmentos (1, mas fragmentos, 0 último fragmento).
- Desplazamiento del fragmento (13 bits): posición de los datos dentro del datagrama completo antes de ser fragmentado. Se mide en bloques de 64 bits (8 octetos). . El primer fragmento de un datagrama tiene siempre valor 0.
- TTL, tiempo de vida (8 bits): nº de saltos que puede permanecer un paquete en una red. Cada router hace disminuir el contador en 1.
- Protocolo: protocolo de un usuario de IP (ICMP, TCP, UDP, ...)
- Suma de comprobación de la cabecera (16 bits): control de error mediante análisis de la cabecera. Es el complemento a 1 de las palabras de 16 bits de la cabecera. Algunos campos cambian por lo que se comprueba en cada salto.
- Dirección origen (SA): código de direccionamiento de 32 bits, dando un número de host (interfaz lógica IP) y/o un número de red.
- Dirección destino (DA). Misma estructura que la dirección SA.
- Opciones (variable): contiene opciones del usuario a la red. Normalmente no se usa, salvo para definir prioridades y calidad de servicio.
- Relleno: obliga a la cabecera del datagrama IP a tener un tamaño múltiplo de 32 bits (4 octetos).
- Datos: lleva información del protocolo y/o los datos del datagrama de la capa superior.

4.2. Direccionamiento

Direcciones IP: dirección de 32 bits única en la interred de trabajo . Consta de:

- Número o identificador de red
- Número o identificador de host

Se representa en notación decimal con los octetos separados por puntos, por ejemplo 172.16.122.204 es 10101100 00010000 01111010 11001100. La asignación de estas direcciones es gestionada por la IANA (Internet Assigned Numbers Authority).

4.2.1. Clases de direcciones

Para facilitar la administración, las direcciones IP están divididas en clases, como puede verse en la figura 8:
lstlisting
Hay varias observaciones que resaltar:

- El número de redes y hosts disponibles cambia en función de cada clase; algunos de estos datos queda reflejado en la tabla 1.
- La falta de 2 hosts en cada clase, se debe a lo siguiente:
  - Un número de host todo ceros está reservado para identificar la red.
  - Un número de host todo unos está reservado para identificar la dirección del broadcast de red.

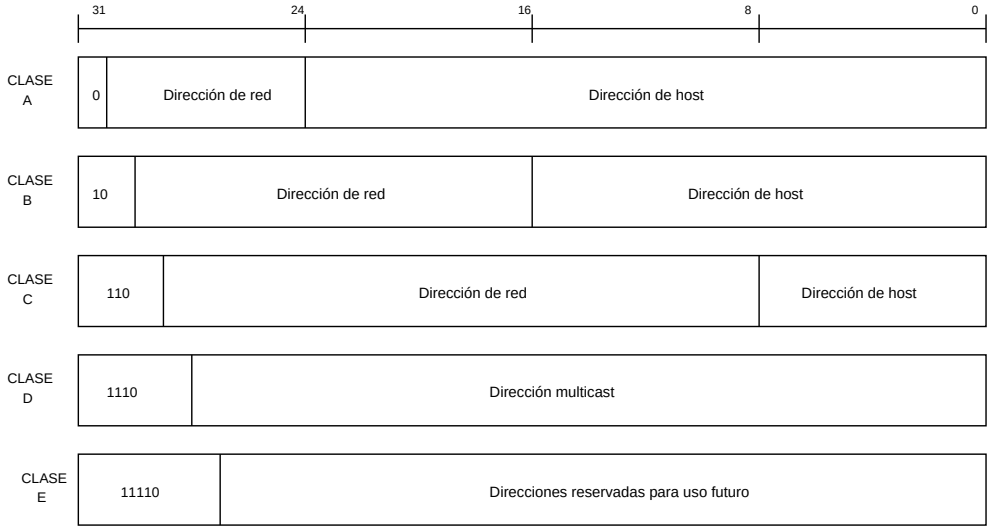

Figura 8: Cabecera del protocolo IP

Cuadro 1: Clases de direcciones IP

clase	Rango	Nº redes	Nº hosts/red
A	1.0.0.0 - 127.255.255.255	128	16.777.214
B	128.0.0.0 - 191.255.255.255	16304	65534
C	192.0.0.0 - 223.255.255.255	2.097.152	254
D	224.0.0.0 - 239.255.255.255		
E	240.0.0.0 - 255.255.255.255		

- Se reserva la dirección 0.0.0.0 por parte del IANA para un host desconocido y para cuando se arranca el protocolo. En general esta dirección se utiliza comlstlisting o dirección por defecto, debiendo el entorno donde se use identificar su significado. Si se usa como destino de una ruta, este destino será la red por defecto, es decir, cualquier red o host del que no exista un destino predefinido. Si se usa como el siguiente salto de una ruta, esta dirección indicará la única dirección IP que puede acceder a la red de destino.

- La clase B (169.254.0.0, 16 bits red, 16 bits hosts) está definida en el RFC 3927. Es una red con direcciones para autoconfiguración. Se suelen llamar direcciones autoconfiguradas, link-local addresses o APIPA. Se suelen autoconfigurar si no hay IP estática y si la consulta a un servidor DHCP ha sido fallida. Los routers no la reenvían, al igual que los rangos privados (RFC 1918).

- Se reserva la clase A 127.xxx.yyy.zzz para la loopback address (representada habitualmente por "lo"), usada por un dispositivo para direccionarse a sí mismo (los paquetes enviados a estas direcciones salen a la red y se tratan como paquetes de entrada). Es útil para comprobar el funcionamiento de la capa IP y las que están por encima.

La utilización de estas clases por defecto se llama *classfull*. El problema es que el diseño clasfull no es escalable, y no permite subdividir las clases en subredes.

4.2.2. Direcciones sin clases

Actualmente, se mantiene esta división por clases, aunque se ha ampliado a un sistema donde las direcciones de host pueden a su vez subdividirse en direcciones de subred y direcciones de host. A este nuevo sistema se le llama CIDR (Classless InterDomain Routing). Además, se pueden agrupar redes de la misma clase en redes de mayor amplitud. Para realizar estas modificaciones en el sistema classfull, se ha incluido el concepto de máscara de red. A todo este conjunto se le llama VLSM (Variable Lenght Subnet Masking).

El número de host queda dividido en dos campos, el primero indica una subred dentro de la red principal (clase A,B,C) y el segundo indica un número de host dentro de la subred. La máscara de red indica qué parte de la dirección IP pertenece al número de subred, y qué parte de la dirección IP pertenece al número de host, es decir, la máscara separa los bits de host de los bits de red y subred. Cada LAN tiene un número de subred. De esa forma, una subred queda unívocamente determinada por su número de subred y su máscara. La máscara tendrá por tanto el siguiente valor en binario:

- Unos para los bits de red y de clase de red

- Unos para los bits de subred

- Ceros para los bits de host

El número de subred se puede calcular con el número de un host cualquiera de la subred y la máscara de la subred haciendo un AND lógico binario entre ambas direcciones (host y máscara). El uso de máscaras permite a un computador determinar si un paquete de salida va destinado a otro computador en la misma LAN o a otra LAN, mientras el mundo exterior a la LAN (otras redes) sigue viendo la red como antes.

La representación de la máscara de red se puede hacer de dos formas:

- mediante la representación decimal del conjunto de 4 octetos que componen la dirección IP (p.ej. 255.255.255.0), o

- mediante el número de unos consecutivos (p.ej. /24). A esta notación se la llama NSC.

4.3. Enrutamiento

El transporte de datos a través de Internet necesita de un sistema de enrutamiento para que los paquetes lleguen a su destino. El sistema de enrutamiento está basado en las tablas de rutas que cada router intermedio tiene que mantener. Las rutas son los registros de las tablas de rutas y tienen que disponer como mínimo de dos campos, el destino, y el siguiente salto. El destino puede ser una red o un equipo concreto y el siguiente salto debe estar accesible y directamente conectado a nivel de red, y debe

ser un host. Habitualmente, las tablas de rutas implementan otros campos, entre los que cabe destacar la métrica (que mide el coste de esa ruta) y la interfaz de salida del router hacia el siguiente salto.

El router y su sistema de enrutamiento (demonio de enrutamiento) se encargan de gestionar las diferentes tablas de rutas que puede almacenar, activando unas sobre otras, dando prioridades a unas sobre otras o seleccionando en función de otros parámetros unas sobre otras (como se puede ver en el enrutamiento basado en políticas en la subsección 4.3.2).

Cuando un paquete IP llega a un router, este analiza su cabecera IP. Seguidamente consulta los registros de la tabla de rutas seleccionada por el demonio de enrutamiento en orden descendente y cuando la dirección de destino del paquete IP forma parte de la que aparezca en el campo "destino" de la tabla de rutas seleccionada, reenvía el paquete IP hacia la dirección marcada como "siguiente salto" en el registro correspondiente de la tabla de rutas.

Puede haber diversas tablas de rutas, y es el demonio de enrutamiento el que se encarga de gestionar cual de ellas aplicar o como gestionarlas, sin existir un estándar para ello. No obstante pueden encontrarse dos tipos de tablas:

- Las tablas de kernel y estáticas, gestionadas por el sistema operativo y por el administrador:
  - Las rutas de kernel suelen ser en función de las direcciones IP de las interfaces y son gestionadas por el sistema operativo
  - Las rutas estáticas, donde el administrador da de alta el registro de ruta al completo.

  En Linux, estas tablas son la tabla "local" y la tabla "main".
- Las tablas de enrutamiento dinámico, que dependen del protocolo utilizado.
  - Rutas RIP: son rutas que vienen asignadas a través del protocolo de enrutamiento RIP.
  - Rutas OSPF: son rutas que vienen asignadas a través del protocolo de enrutamiento OSPF.
  - Rutas BGP: son rutas que vienen asignadas a través del protocolo de enrutamiento BGP.

  En Linux, estas rutas son creadas por los demonios de enrutamiento de cada protocolo englobados en la suite *quagga*, y son incluidas directamente en la tabla "main".

Para la asignación de rutas estáticas se utilizan juegos de comandos dependientes del sistema operativo. En Linux, por ejemplo sería:

```
# ip route add 192.168.10.0/24 via 192.168.1.1
```

donde 192.168.10.0/24 es la red de destino y 192.168.1.1 es el siguiente salto.

Los routers, cuando les llega un paquete con una dirección IP de destino, consultan la tabla de rutas principal en orden descendente, y el primer registro que coincida con la dirección destino es el elegido para reenviar el paquete. Esto implica que cuando el demonio de enrutamiento da de alta rutas en la tabla, esta se reordena en orden de máscara de red de destino mas concreta a máscara mas genérica, de forma que en caso de que un paquete tenga varias opciones de salida, el demonio de enrutamiento seleccione la que delimite mejor el destino solicitado. La ruta de conexión a Internet se llama ruta por defecto, y la red IP de destino en el registro correspondiente de la tabla es la red 0.0.0.0/0

Por ejemplo, si a un router con sistema operativo Linux se le dan de forma consecutiva las siguientes rutas:

```
# ip route add 192.168.25.0/24 via 192.168.1.2
# ip route add 192.168.25.0/29 via 192.168.1.3
# ip route add 0.0.0.0/0 via 192.168.1.1
# ip route add 192.168.25.0/30 via 192.168.1.4
```

y le llega un paquete con destino a la dirección 192.168.25.2, el router dirigirá el paquete hacia 192.168.1.4.

4.3.1. Enrutamiento tradicional o basado en destino

El enrutamiento tradicional analiza la cabecera IP de un paquete entrante, y en particular la dirección IP de destino sin tener en cuenta otros campos del paquete. Seguidamente realiza las consultas en su tabla principal de rutas (tabla *main*) en orden descendente y cuando la dirección de destino del paquete IP está incluida en el campo "destino" de la tabla de rutas, reenvía el paquete IP hacia la dirección marcada como "siguiente salto" en el registro correspondiente de la tabla de rutas.

4.3.2. Enrutamiento basado en políticas (o Policy Based Routing PBR)

Al sistema de enrutamiento tradicional se le llama "enrutamiento basado en el destino", básicamente porque el único campo que tiene en cuenta un router para redirigir el paquete es el campo IP destino.

No obstante, en determinadas circunstancias, un router debería poder direccionar paquetes en función de otros campos, como la IP origen, protocolo IP, puertos de transporte o cualquier otro campo. A este sistema de enrutamiento se le llama "enrutamiento basado en políticas" (o policy based routing PBR).

Esta técnica permite dirigir paquetes a un mismo destino a través de diferentes rutas. Un ejemplo práctico de enrutamiento basado en políticas es la posibilidad de disponer de varias salidas a Internet en una red corporativa y seleccionar una salida u otra en función del origen (la red segura a través de la linea dedicada y la red de usuarios a través del XDSL) o incluso del tipo de tráfico (el correo electrónico y el tráfico web a través de la conexión XDSL y el resto de tráfico a través de la linea dedicada).

Para el PBR se utiliza en general una base de datos de reglas de enrutamiento (RPDB), que selecciona una ruta (o tabla de rutas) cuando se aplica alguna de las reglas almacenadas. En Linux, la gestión del PBR se hace mediante el comando "ip rule".

4.3.3. Proceso completo

Cuando un router recibe un paquete por una interfaz, el router realiza una serie de acciones:

1. Borra la cabecera de enlace de la interfaz de entrada.
2. Consulta la cabecera IP:
   - Resta 1 al campo TTL
   - Si el resultado del TTL es cero elimina el paquete y envia un paquete ICMP tipo 11 (ver 5.3) a la dirección IP origen. Si el resultado es distinto de cero continúa el proceso.
   - Consulta la dirección IP destino
3. Busca según la algoritmia del demonio de enrutamiento la ruta hacia esa dirección IP destino. Utilizando la algoritmia de Linux *iproute2* como ejemplo:
   - Consulta de la caché de rutas (*ip route show cache*). Si no encuentra una ruta adecuada al destino del paquete, pasa al siguiente punto.
   - Consulta por orden de prioridad las reglas (*ip rule*) existentes en la routing policy database (RPDB). Cuando encuentra una regla adecuada, consulta la tabla asignada a esa regla.
   - En la tabla consulta en orden descendente (de máscara mas concreta a mas genérica) el destino de cada ruta hasta que coincida en host o en red con la dirección destino del paquete. Si no hay ningún registro en el que el destino pueda aplicarse al paquete que se está procesando, no hay ruta hacia ese destino y se realizan dos acciones: se descarta el paquete y se envia un paquete ICMP tipo 3 al origen.
4. Si encuentra ruta hacia el destino, deposita el paquete en la interfaz de salida, y le añade la cabecera de enlace asociada a esa interfaz.

5. Protocolos de Internet

Sobre el protocolo IP o basados en él, hay disponibles multitud de protocolos, tanto de transporte como directamente de usuario. En esta sección se analizarán algunos de ellos y se recomienda una visualización detenida de la figura 9 donde se ubican algunos de estos protocolos muy conocidos en la red en la arquitectura de protocolos.

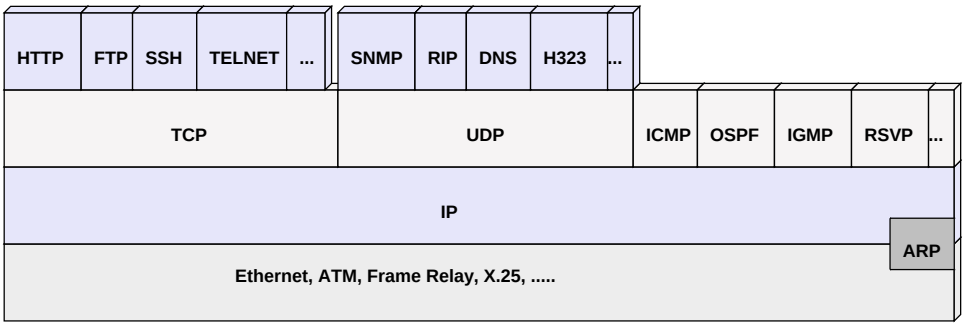

Figura 9: Protocolos importantes de la arquitectura IP

5.1. Protocolo DHCP

DHCP es un protocolo de configuración de host dinámico, descrito en RFC2131, que funciona en modo cliente/servidor. El programa cliente se encuentra en la mayoría de los sistemas operativos estando basado en el protocolo BOOTP con quien comparte puertos de transporte y el uso del protocolo UDP.

5.1.1. BOOTP

En algunas redes de área local o incluso extensa, se pueden utilizar dispositivos terminales sin memoria permanente que guarde la configuración de red, y que por tanto esta tenga que configurarse de forma manual cada vez que se inician. Algunos de estos hosts se conocen como estaciones de trabajo o terminales tontos (*thin clients*); otros pueden ser routers, conentradores de terminal u otros dispositivos de red o periféricos. Inicialmente, el protocolo RARP automatizaba parte de la configuración de estos clientes, pero mantenía la necesidad de la intervención humana para finalizar la configuración.

BOOTP es un protocolo que transfiere la configuración de red hacia este tipo de terminales. Lo hace en modo cliente/servidor, donde un servidor BOOTP dispone de la lógica y la información necesarias para la configuración de varios dispositivos terminales. BOOTP entrega el código y la configuración necesarias para que estos dispositivos arranquen. Los hosts terminales que requieren de estos servicios, disponen de una codificación mínima, sin información de configuración, donde implementan la lógica BOOTP de cliente típicamente en una ROM, y así comenzar el proceso de descarga del código e información necesarias para arrancar. BOOTP permite de esa forma el arranque remoto en redes IP.

El proceso de descarga no viene definido por BOOTP, utilizándose el protocolo TFTP . El proceso BOOTP involucra los siguientes pasos:

1. El cliente determina su propia dirección hardware; esta dirección está normalmente en una ROM.
2. Un cliente BOOTP envía su propia dirección hardware en un datagrama UDP al puerto 67 del servidor. El contenido completo de este datagrama puede verse en la figura 10. Si el cliente conoce su propia dirección IP y/o la dirección del servidor, rellenará los campos correspondientes del datagrama pero en general los clientes BOOTP no tienen datos de configuration IP. Si el cliente no conoce su propia dirección IP, usa la dirección genérica 0.0.0.0. Si el cliente no conoce la dirección IP del servidor BOOTP, usa la dirección broadcast (255.255.255.255).
3. El servidor recibe el datagrama y busca la dirección hardware del cliente en su fichero de configuración, donde tiene asociada a esta la dirección IP del cliente. El servidor rellena los campos restantes en el datagrama UDP y lo devuelve al puerto UDP 68 del cliente. Para ello utiliza dos métodos:

- Si el cliente conoce su propia dirección IP (se incluyó en la petición BOOTP), entonces el servidor devuelve el datagrama directamente a esta dirección, es decir la dirección IP destino del datagrama BOOTP es la dirección IP del cliente. Como es probable que en la caché ARP del servidor no esté la dirección hardware que está asociada a la dirección IP del cliente, se utilizará ARP para determinarla.
- Si el cliente no conoce su propia dirección IP (0.0.0.0 en la petición BOOTP), entonces el servidor no puede usar ARP para encontrar la dirección hardware del cliente porque el cliente no conoce su propia dirección IP y por tanto no puede responder a una petición ARP. Por tanto, el servidor envía una respuesta a la dirección IP de broadcast, y se utiliza el campo "ID de transacción" para que cada cliente identifique una respuesta dirigida a él.

4. Cuando recibe la respuesta, el cliente BOOTP grabará su propia dirección IP (permitiendo que responda a las peticiones ARP) y comenzará el proceso de arranque o bootstrap.

Figura 10: Datagrama BOOTP

Por tanto, un terminal que utilice BOOTP para iniciarse, debe incluir en su ROM o en algún otro medio de almacenamiento permanente una lógica básica del protocolo IP, del protocolo ARP, del protocolo BOOTP y del protocolo TFTP.

5.1.2. DHCP

DHCP es la evolución del protocolo BOOTP, funcionando al igual que este en los puertos UDP 67 (servidor) y UDP 68 (cliente). DHCP ha permitido que el uso inicial para el que estaba destinado de proveer configuración de red, haya derivado en ser uno de los ejes principales de la centralización y mantenimiento de las configuraciones de red para los equipos cliente terminales. Ello implica unos requerimientos de seguridad importantes desde el administrador, puesto que este protocolo no incluye practicamente ninguna medida adicional.

DHCP ofrece tres tipos de asignación de direcciones, a diferencia de BOOTP, donde sólo se produce asignación de dirección IP cuando la dirección MAC del host está registrada y asociada.

- Asignación automática: DHCP asigna automaticamente una dirección IP a un cliente.
- Asignación manual: El administrador asigna una dirección IP al cliente. DHCP comunica la dirección al cliente
- Asignación dinámica: DHCP asigna, o alquila una dirección IP al cliente por un periodo limitado

El proceso de configuración de un cliente DHCP consta de varios pasos que se enumeran a continuación.

1. El cliente envía petición de IP al servidor para obtener una IP (a veces el cliente sugiere la IP, como cuando pide una extensión de un alquiler DHCP). El cliente solicita un DHCPDISCOVER (broadcast)
2. El servidor al recibir broadcast, si puede servir esa petición ofrece una IP al cliente como DCPOFFER unicast.
3. Si el cliente acepta la propuesta envia otro broadcast, un DHCPREQUEST (se envia en broadcast para que otros posibles servidores DHCP sepan que se aceptó una oferta), y pide otros parametros IP.
4. El servidor que recibe el DHCPREQUEST envia un recibo unicast DHCPACK (es posible aunque poco probable que no lo envie por haber alquilado esa información a otro cliente). Recibir el DHCPACK permite al cliente comenzar a usar la dirección asignada
5. Si el cliente detecta que la dirección ya está en uso en el segmento local envia un DHCPDECLINE y el proceso recomienza. Si el cliente recibe un DHCPNACK del servidor tras haber enviado un DHCPREQUEST recomienza el proceso.

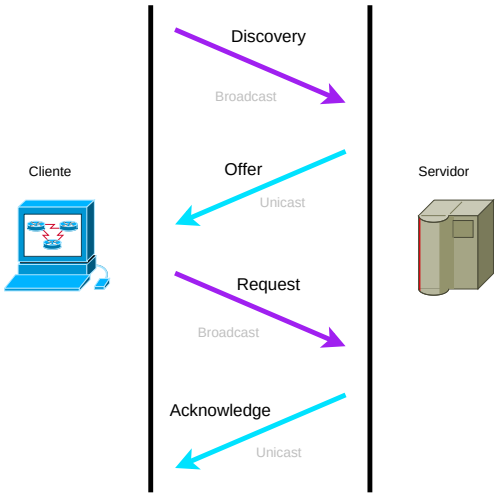

Figura 11: Proceso DHCP

Diferencias entre BOOTP y DHCP

Cuadro 2: BOOTP vs DHCP

BOOTP	DHCP
Mapeo estático	Mapeo dinámico
Asignación permanente	Alquiler
4 parámetros	mas de 30

5.2. TCP/UDP

El nivel o capa de transporte proporciona servicios extremo a extremo, esto es, a las aplicaciones de usuario. Está situada encima de la capa de red y debajo de la capa de aplicación. Los protocolos de transporte de un host se comunican mediante TPDU (Transport Protocol Data Units) con los protocolos de transporte de los hosts remotos. Funcionan con primitivas y parámetros. Los servicios proporcionados pueden clasificarse en:

- Confiables, u orientado a conexión. Estos servicios permiten abrir un canal de comunicaciones extremo a extremo que puede ofrecer servicios como el control del flujo, recuperación ante pérdidas y entrega en orden. Estos protocolos priman la confiabilidad sobre la velocidad. Son adecuados para servicios como el correo electrónico (SMTP, POP), el envío de comandos o configuraciones (TELNET, SSH) o la transferencia de ficheros (FTP). El protocolo de transporte confiable por excelencia en la pila de protocolos TCP/IP es TCP.
- No confiables, o no orientado a conexión. Los protocolos no confiables no suelen ofrecer servicios como entrega en orden o recuperación ante pérdidas. Algunos protocolos no confiables sí permiten en cambio un cierto control del flujo. Estos protocolos priman la velocidad sobre la confiabilidad y son adecuados para transmisiones multimedia, videoconferencia (H323), voz sobre IP (VoIP), monitorización (SNMP) y telecontrol o teleoperación, aunque también hay protocolos para configuración que lo usan (DNS, DHCP). El protocolo de transporte no confiable sobre redes IP por excelencia es UDP.

Los serviciops que puede ofrecer la capa de transporte son:

- Direccionamiento/multiplexación: es un servicio de direccionamiento e identificación de aplicaciones de la capa superior. Se implementa mediante direcciones de usuario (habitualmente llamadas puertos) tanto origen como destino.
- Control de flujo. La capa de enlace ya lo contempla, pero la capa de transporte también pues el flujo es mas variable (depende de la interred). Se utilizan mecanismos parecidos a los de enlace (asignación de créditos – ventana deslizante).
- Transporte en orden. Al igual que en enlace de datos, se asignan números de secuencia. Este servicio está implícito en TCP.
- Retransmisión ante fallos. Fallo puede ser corrupción de datos o pérdida de paquete. Estrategia de TPDU de asentimiento. Existe también la confirmación de paquetes en bloque.
- Detección de duplicados: existe la posibilidad de recibir duplicados cuando un paquete se retransmite al haberse perdido el asentimiento.
- Establecimiento/cierre de la conexión: Para saber que el destino existe y poder negociar determinados parámetros.
- Recuperación ante caídas.

5.2.1. Protocolo TCP

Está referenciado en el RFC 793 y el protocolo IP identifica que el contenido de datos es del protocolo TCP con el campo "protocolo" de la cabecera IP puesto a 6. Diseñado para comunicación segura entre procesos de hosts diferentes. Proporciona servicio orientado a conexión a la capa de aplicación. Básicamente TCP tiene 3 objetivos:

- Adaptar la velocidad de transmisión a la capacidad del enlace extremo a extremo.
- Evitar congestión en la redes
- Crear una conexión fiable mediante la retransmisión de paquetes perdidos.

Su cabecera se muestra en la figura 12

- **Puerto origen** (16 bits): identifica la aplicación o servicio origen de la transmisión.
- **Puerto destino** (16 bits): identifica la aplicación o servicio destino de la transmisión.
- **Número de secuencia** (32 bits): número de secuencia del primer octeto de datos (excepto si SYN=1). En ese caso se produce un sincronismo y el primer octeto de datos es un número inicial +1.

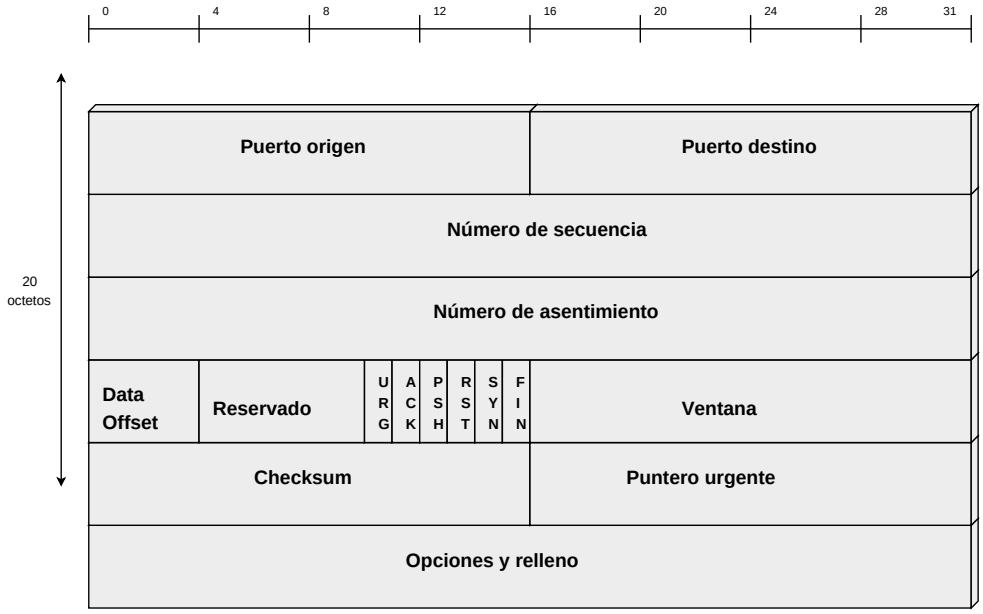

Figura 12: cabecera TCP

- **Número de confirmación o asentimiento** (32 bits): número de secuencia del siguiente octeto esperado por la entidad de transporte.
- **Longitud de cabecera** (4 bits): número de palabras de 32 bits en la cabecera
- **Reservados** (6 bits): para uso futuro
- Indicadores (6 bits)
  - **URG**: urgente
  - **ACK**: confirmación
  - **PSH**: función de carga.
  - **RST**: Puesta a cero
  - **SYN**: sincronización de números de secuencia
  - **FIN**: fin de datos
- **Ventana** (16 bits): crédito de octetos de datos que el que envía está dispuesto a aceptar desde el que se indica en el campo de confirmación. Control de flujo.
- **Suma de verificación** (16 bits): complemento a 1 de una parte del paquete.
- **Puntero Urgente** (16 bits): señala al octeto que sigue a los datos urgentes.
- **Opciones** (variable): información de parámetros al comenzar la conexión.

Su funcionamiento básico consiste en tres pasos fundamentales:

1. Establecimiento de la conexión:

   Diálogo en tres sentidos. Si SYN = 1 la PDU es de establecimiento de la conexión, con el campo SN=X como número de secuencia inicial. El receptor responde con SYN = 1, SN=Y y AN=X+1. Ahora el receptor espera recibir una PDU con SN=X+1 y el emisor responde con AN= Y+1.

   SYN=1, SN=X →
   ← SYN=1, SN=Y, AN=X+1
   SYN=0, SN=X+1, AN=Y+1 →

2. Transferencia de datos:

   En este paso existen varias posibilidades:

   - Datos urgentes. El indicados URG se pone a uno (URG=1)
   - Transferencia de datos en bloque. En general se establece mediante control de flujo de números secuencia. El indicador PSH se pone a uno (PSH=1) en la última TPDU (fin de bloque).

3. Cierre de la conexión:

   Existen dos cierres de conexión:

   - Cierre ordenado: última PDU a enviar FIN=1
   - Cierre brusco: se envía por parte del último que debe transmitir un RST y se borran las "cachés" o memorias temporales.

A nivel de programación, se utilizan sockets de tipo STREAM.
Algunos aspectos de funcionamiento:

- Sistema de asentimientos ACK. Los paquetes ACK informan a la fuente cual es el número de secuencia del siguiente paquete esperado. Este número va en el campo AN, y puede enviarse en un paquete de datos estándar (etiqueta ACK=0). Los ACKs tienen como objetivo regular la velocidad de transmisión de TCP (asegurando que un paquete es transmitido cuando los anteriores ya han abandonado la red), y mantener una comunicación confiable.

  Un ACK informa al origen de cual es el siguiente octeto de datos esperado (por tanto paquete). Si un paquete se ha perdido en medio de una secuencia, y el receptor recibe un paquete con un SN no esperado y superior al anterior, entonces envía un ACK con el último AN recibido de forma consecutiva; a esto se le llama ACK implicito, es decir, que asintiendo el AN X es como haber asentido todos los anteriores, y el origen sabe que los paquetes que contienen datos previos al X han llegado correctamente. Este método es robusto frente a la pérdida de ACKs, pues con el último ACK recibido se confirma la información de todos los anteriores.

  Un paquete se considera perdido si:

  - Llegan 3 ACKs para el mismo paquetes, o
  - Cuando un paquete es transmitido se activa un temporizador. Si el ACK de ese paquete no llega antes de que expire el temporizador el paquete se considera perdido. El valor del temporizador depende en función del RTT y de la varianza D del RTT.

  El protocolo calcula ese valor en cada iteración y suele tomar el valor:

  $$T_o = \overline{RTT} + 4D \tag{6}$$

  Tras recibir un nuevo ACK y por tanto tener un valor RTT actualizado, el valor de $\overline{RTT}$ y de D se actualiza de la forma:

  $$\begin{aligned} \overline{RTT} &\leftarrow a \cdot \overline{RTT} + (1-a)RTT \\ D &\leftarrow a \cdot D + (1-a)|\overline{RTT} - RTT| \end{aligned} \tag{7}$$

  El valor de a es un indicador de "memoria" donde un valor alto significa que el valor actual es menos importante que los anteriores, y un valor bajo implica una variación mas hacia los nuevos valores.

- Ventana deslizante

  La idea básica es sencilla; TCP dispone de un campo "ventana" W que indica el número de octetos que el origen está dispuesto a aceptar sin generar asentimientos y por tanto sin que el receptor necesite conocer que han llegado hasta el final de la transmisión. Si ese campo "ventana" es pequeño, se le permite crecer bastante rápidamente, y cuando llega a valores altos ya sólo crecería lentamente. Como ejemplo, imaginemos un valor $W = W_{th}$ a partir del cual se considera que W es grande; W empieza con el valor 1; cuando recibe su ACK correspondiente W pasa a valer 2, después pasa a valer 4, y así hasta que valga W_{th}. En ese momento pasa a una segunda fase, donde el valor de W se incrementa en $1/W$ cada vez que llega un ACK.

- Pérdidas

  No sólo el valor de W es dinámico, sino también el valor de W_{th} que es fijado a la mitad de W cuando se detecta la pérdida de un paquete. Existen dos variantes fundamentales de TCP en función de con qué valor se inicializa W tras detectar una pérdida. En la primera, llamada "Tahoe", el valor de W se pone a 1 y comienza una fase de arranque lento. En la segunda llamada "Reno" o "New Reno" W coje el valor 1 sólo si la pérdida de paquete ha sido detectada por expiración del temporizador, y se pone a $W/2$ si la pérdida ha sido detectada por la llegada de varios ACKs.

- Inicialización de la conexión

  El host origen envía un paquete SYNC de 40 octetos hacia el destino. El destino envía un ACK (también de 40 octetos) llamado "sync ACK". Una vez recibido este ACK el origen comienza el envío de datos. Si alguno de estos paquetes se pierde, tras un periodo de entre 3 y 6 segundos, se retransmite. Si un paquete retransmitido también se pierde se reenvía otro tras un tiempo el doble del anterior.

  Este mecanismo permite ataques del tipo "Syn Flood", que consisten en el envío de paquetes SYNC, activar los temporizadores y no responder al ACK del destino.

5.2.2. Protocolo UDP

Está basado en el RFC 768 y el protocolo IP identifica que el contenido de datos es del protocolo UDP con el campo "protocolo" de la cabecera IP puesto a 11. Proporciona servicio no orientado a conexión, es decir, es un servicio no seguro por lo que no está garantizada la entrega ni la protección contra duplicados. Como contraprestación se reduce mucha información suplementaria. Como ejemplos de protocolos que utilizan modos de transporte no confiables sobre UDP están el DNS, SNMP o H.323.

Este tipo de servicios se justifica en los casos de recolección de datos, donde la monitorización en tiempo real o semi-real no se ve gravemente afectada por la pérdida de un dato; la difusión de datos como valores de reloj; cuando se regulan servicios orientados a conexión en la capa de aplicación (teleoperación o telecontrol); aplicaciones en tiempo real, voz, vídeo etc.

La cabecera tiene la forma de la figura 13.

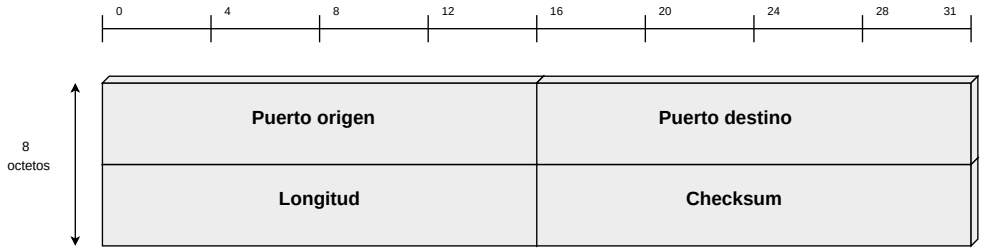

Figura 13: cabecera UDP

UDP no lleva ningún campo de opciones. Básicamente complementa a IP con dirección de puerto. El campo "longitud" incluye los datos, y el campo "suma de comprobación" es opcional (en caso en que no se use se pone todo a ceros).

A nivel de programación, se utilizan sockets de tipo DATAGRAM..

5.3. ICMP

Cuando un "router" o un host de destino debe informar al host fuente acerca del procesamiento de datagramas, utiliza ICMP (Ïnternet Control Message Protocol"Protocolo de Mensajes de Control de Internet). ICMP está documentado en el RFC 792, y el protocolo IP identifica que el contenido de datos es del protocolo ICMP con el campo "protocolo" de la cabecera IP puesto a 1. ICMP puede caracterizarse del modo siguiente:

- ICMP usa IP como si ICMP fuera un protocolo del nivel superior (es decir, los mensajes ICMP se encapsulan en datagramas IP). ICMP es parte integral de IP y debe ser implementado por todo módulo IP. Pero ICMP no está encapsulado como paquetes TCP/UDP, ya que no necesita la capa de transporte. El filtrado de paquetes ICMP en firewalls se suele hacer por el tipo de mensaje ICMP, como por ejemplo la deshabilitación de las respuestas al "ping" mediante el filtrado del tipo 8.
- ICMP se usa para informar de algunos errores. Aún puede ocurrir que los datagramas no se entreguen y que no se informe de su pérdida. La fiabilidad debe ser implementada por los protocolos de nivel superior que usan IP.
- ICMP puede informar de errores en cualquier datagrama IP con la excepción de errores en los propios mensajes ICMP, para evitar repeticiones infinitas.
- Para datagramas IP fragmentados, los mensajes ICMP sólo se envían para errores ocurridos en el fragmento cero. Es decir, los mensajes ICMP nunca se refieren a un datagrama IP con un campo de desplazamiento de fragmento.
- Los mensajes ICMP nunca se envían en respuesta a datagramas con una dirección IP de destino que sea de broadcast o de multicast.
- Los mensajes ICMP nunca se envían en respuesta a un datagrama que no tenga una dirección IP de origen que represente a un único host. Es decir, la dirección de origen no puede ser cero, una dirección de loopback, de broadcast o de multicast.
- Los mensajes ICMP nunca se envían en respuesta a mensajes ICMP de error. Pueden enviarse en respuesta a mensajes ICMP de consulta (los tipos de mensaje ICMP 0, 8, 9, 10 y 13 al 18).

El formato básico del paquete ICMP puede verse en la figura 14.

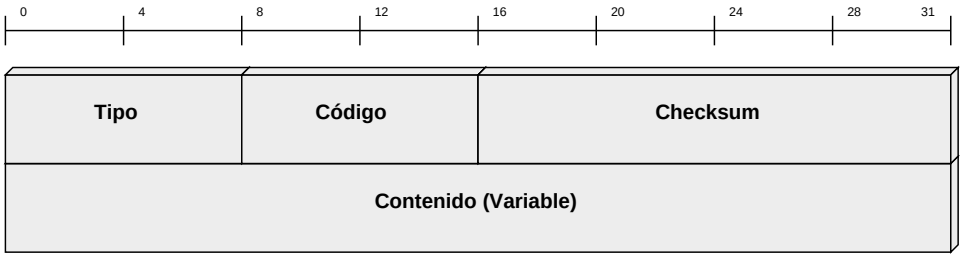

Figura 14: cabecera ICMP

El campo "tipo" indica el tipo de paquete ICMP. El campo código indica fundamentalmente tipos de respuesta para diferentes tipos de mensaje. Cada tipo de mensaje rellena el campo código con contenidos diferentes en función de la respuesta.

No se utilizan los 256 tipos disponibles en el campo "tipo" de la cabecera ICMP, estando algunos reservados. Algunos de los mas importantes pueden verse en la tabla 3.

Funcionamiento básico:

- Se utiliza como herramienta de resolución de problemas.
- Algunos tipos de mensaje se implementan en algunos programas (nivel aplicación):
- Ping. Programa de solicitud de confirmación de eco. Este comando está disponible en prácticamente todas las plataformas que existen. Puede tener diferentes opciones, a destacar:
  - Tamaño de paquete
  - No fragmentación
  - Registro de ruta

Cuadro 3: Principales tipos de mensajes ICMP

Identificador (Tipo)	Tipo de mensaje
0	Echo Reply (respuesta de eco)
3	Destination Unreacheable (destino inaccesible)
4	Source Quench (disminución del tráfico desde el origen)
5	Redirect (redireccionar - cambio de ruta)
8	Echo (solicitud de eco)
9	Router-advertisement (Publicación de rutas)
10	Router-solicitation (Petición de ruta).
11	Time Exceeded (tiempo excedido para un datagrama)
12	Parameter Problem (problema de parámetros)
13	Timestamp (solicitud de marca de tiempo)
14	Timestamp Reply (respuesta de marca de tiempo)
15	Information Request (solicitud de información)
16	Information Reply (respuesta de información).
17	Addressmask (solicitud de máscara de dirección)
18	Addressmask Reply (respuesta de máscara de dirección)

- Trace. Programa de traceo de rutas. Este comando está disponible en prácticamente todas las plataformas. Tiene diferentes nombres en función de la plataforma:
  - Tracert: Windows 9X, 2000, NT ...
  - Traceroute: UNIX, Linux
  - Trace: cisco IOS.
- Se puede enviar como respuesta por un error detectado (detección de un problema en la red o en la cabecera IP de un paquete) :
- Puede sugerir mejoras en las rutas.
- Se puede solicitar una comprobación, a voluntad del administrador.
- Se puede consultar la ruta hacia un destino.
- Funcionamiento de algunos códigos ICMP:
  - ICMP 3, Destination Unreachable

    Este paquete ICMP se genera cuando en alguno de los routers intermedios por los que pasa el transporte de datos IP no conoce ninguna ruta para llegar al destino marcado por la dirección IP destino. La figura 15 muestra un esquema de funcionamiento.

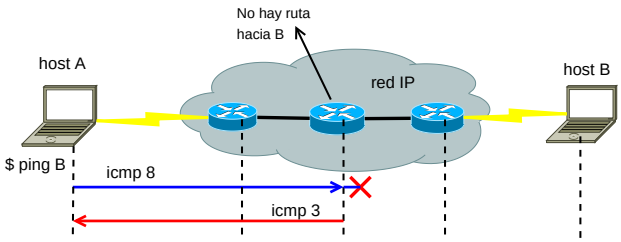

Figura 15: ICMP tipo 3 (Destination unreachable)

- ICMP 8,0. Echo Request, Echo Reply

  El paquete ICMP tipo 8 se genera a demanda del usuario para la comprobación de conectividad IP hacia un destino. El paquete ICMP tipo 0 es la respuesta del destino hacia el origen y se genera tras la llegada de un paquete ICMP tipo 8. Los paquetes ICMP tipo 8 y 0 llevan en su campo de contenido un número de secuencia. La aplicación "ping" calcula con los paquetes ICMP tipo 0 recibidos el valor del round trip time (RTT) y la pérdida de paquetes mediante el control de números de secuencia recibidos, que debe ser igual al de los enviados por los ICMP tipo 8. La figura 16 muestra un esquema de funcionamiento.

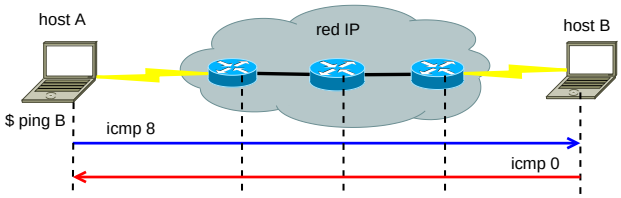

Figura 16: ICMP tipo 8,0 (Echo request, Echo response)

- ICMP 10,9. Router solicitation, Router advertisement

  El paquete ICMP tipo 9 se genera por parte de los routers a través de interfaces con soporte multicast para informar de su existencia. Los hosts con soporte multicast lo reciben y toman las acciones pertinentes. Un host puede generar un ICMP tipo 10 para no esperar por el ICMP tipo 9 que le llega de forma periodica.

- ICMP 11. Time Exceeded

  Este paquete ICMP se genera automáticamente cuando en alguno de los routers intermedios por los que pasa el transporte de datos IP el campo TTL llega con valor igual a 1, de forma que lo descarta por haber agotado su ruta. La figura 17 muestra un esquema de funcionamiento.

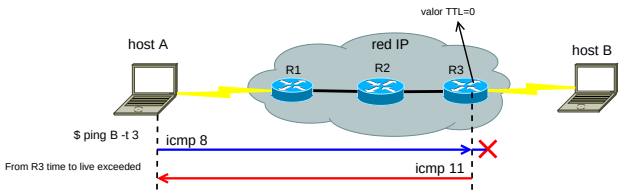

Figura 17: ICMP tipo 11 (Time Exceeded)

  Este tipo de mensajes ICMP se usa también por el programa "traceroute", poniendo el campo TTL de forma consecutiva a valores que empiezan por 1 y finalizan cuando llega un ICMP tipo 0 del destino. De esa forma, todos los routers intermedios generan un mensaje de Time Exceeded (ICMP tipo 11) cuando el valor del TTL que reciben es 1 (puesto que cada router siempre resta 1 al campo TTL de la cabecera IP), y el host, con ese mensaje ICMP 11 presenta la dirección IP y resultados de retardo de cada router intermedio antes de llegar al destino indicado por el programa "traceroute".

  La figura 18 muestra un esquema de funcionamiento.

- ICMP 5, Redirect

  Siguiendo la figura 19, este paquete ICMP se genera automáticamente cuando un router R1 detecta que se puede llegar a la dirección IP destino a través de otro router R2 que está en la misma red IP que el equipo que le ha enviado el paquete IP original. El paquete ICMP 5 se envía al equipo que le ha enviado el paquete IP original, y este actualiza su tabla de rutas temporal para ese destino.

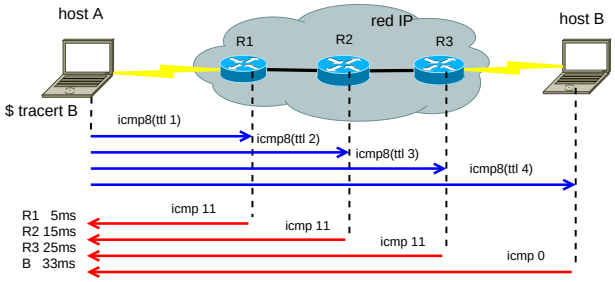

Figura 18: ICMP tipo 8,11 para traceroute (Echo request, Time Exceeded)

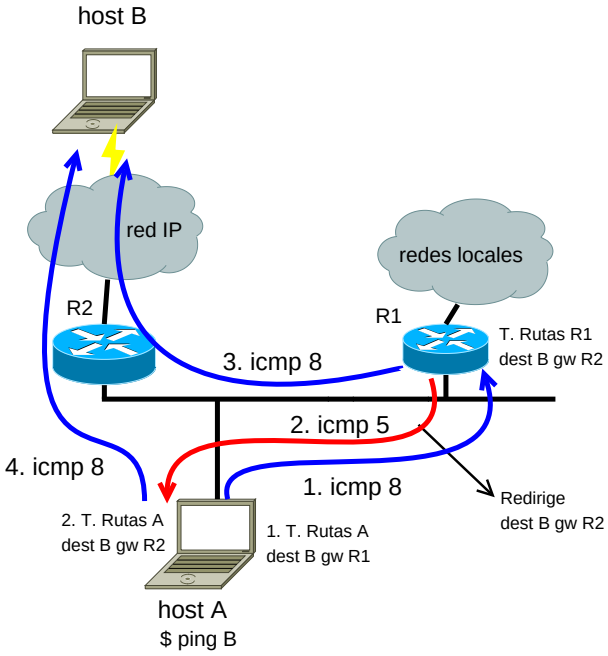

Figura 19: ICMP tipo 5 (Redirect

6. Sockets

La programación de protocolos IP se realiza utilizando *Sockets*. Un Socket (del inglés socket, enchufe) es un objeto de programación puesto a disposición por parte del sistema operativo que sirve como conector para las comunicaciones. Un programa usa Sockets para intercambiar datos con otros programas, tanto si estos programas están en la misma máquina como en otras, en cuyo caso la comunicación se hace a través de redes. Prácticamente todos los lenguajes de programación ponen a disposición funciones para la implementación de sockets, en particular sockets para el protocolo IP; de hecho, los sockets son la API estándar para la programación de conexiones TCP/IP, es decir, para permitir la comunicación entre procesos usando el protocolo IP.

En general se distingue entre **Sockets Stream** y **Sockets Datagram**. La distinción básica entre ellos es que los sockets Stream permiten la transmisión de datos en forma de flujos, y los sockets Datagram en forma de mensajes; por ello, los sockets Stream se usan de forma prácticamente exclusiva para la transmisión de datos mediante el protocolo TCP y los sockets Datagram para la transmisión de datos mediante UDP.

Las funciones que implementan los sockets Stream y Datagram normalmente rellenan las cabeceras TCP y UDP respectivamente de forma automática en función de los parámetros que se pasan en las funciones. Para el caso en el que se desarrollen protocolos (o simples intercambios de información o datos) que no utilicen TCP o UDP como transporte, existen los **Sockets Raw** que permiten la creación de cabeceras propias así como la manipulación manual de las cabeceras TCP o UDP. Los sockets Raw son los sockets originarios de la programación de sockets, y son imprescindibles para la programación en routers (u otros dispositivos de red) o la programación de Packet-Sniffers (capturadores de paquetes) o Packet-Injectors (inyectores de paquetes).

Un socket es el punto de conexión local a un programa remoto, representado este normalmente por la dirección de host y el SAP (dirección IP, puerto). El socket debe disponer de su propia dirección local (dirección IP, puerto). La asignación de la dirección IP local se suele hacer de forma automática (dirección IP 0.0.0.0 o como valor simbólico "INADDR_ANY"); en cambio la asignación del puerto local puede hacerse de forma específica (con la función "bind" usada para la creación de servidores) o también de forma automática (la asignación depende de cada sistema operativo, pero suele hacerse utilizando puertos libres a partir del 4096) mediante el envío previo de un paquete (función "sendto" en Socket DATAGRAM) o de una solicitud de conexión (función "connect" en Sockets STREAM). La comunicación entre dos programas en hosts distintos debe disponer de al menos un socket en cada uno de los hosts, pues son los sockets los que se comunican entre sí; para el envío de datos es necesario conocer la dirección del socket destino (dirección IP + puerto). Básicamente, un socket STREAM o DATAGRAM queda unívocamente determinado por una dirección IP y puerto locales y un socket RAW es una dirección IP y un *Protocol* (campo protocolo de la cabecera IP) locales. De forma resumida: un socket sólo puede ser referenciado desde el exterior si su dirección es conocida, pero se puede utilizar un socket local (para la transmisión de datos) aunque no se conozca la dirección que tiene (pero evidentemente sí será necesario conocer la dirección remota).

6.1. Sockets DATAGRAM

La arquitectura Cliente/Servidor en sockets Datagram no está claramente definida, salvo por quién envía primero los mensajes (Cliente) y quién los recibe primero (Servidor). Muchos autores utilizan el término "emisor/receptor" en su lugar. De esa forma, puede definirse un proceso de programación con sockets DATAGRAM como:

Lado Cliente :

1. Creación del socket.
2. Envío de datos a un destino conocido (utilizando la función "sendto") o a varios (así se da a conocer el puerto local asignado a ese socket o conjunto de sockets).
3. Recepción de datos de cualquier origen (que conozca el puerto local, habitualmente el destino anterior) al socket. Para ello se utiliza la función "recvfrom".

En pseudocódigo:

```
Variable definition;

Main {
  set_destination_address_port DstAddrPort;
  SocketUDP=socket(AF_INET, SOCK_DGRAM, 0);
  while true {
    set_data_to_send DataSend;
    sendto(SocketUDP, DataSend, DstAddrPort);
    recvfrom(SocketUDP, DataRecv, SrcAddrPort);
    make_something_with_recv_data DataRecv;
  }
  close(SocketUDP);
}
```

Lado Servidor :

1. Creación del socket.
2. Asociación del socket con un puerto local (función "bind")
3. Recepción de paquetes de cualquier origen. De esa forma se conoce también dónde hay que enviar confirmaciones u otros datos.
4. Envío de datos hacia cualquier destino.

En pseudocódigo:

```
Variable definition;

Main {
  set_local_port LocalPort;
  SocketUDP=socket(AF_INET, SOCK_DGRAM, 0);
  bind(SocketUDP, LocalPort);
  while true {
    recvfrom(SocketUDP, DataRecv, SrcAddrPort);
    make_something_with_recv_data DataRecv;
    set_data_to_send DataSend;
    sendto(SocketUDP, DataSend, DstAddrPort);
  }
  close(SocketUDP);
}
```

Una alternativa sería crear dos servidores (con puertos específicos asignados con la función "bind") y que estos se comuniquen entre sí pudiendo enviar paquetes sin necesidad de recibir primero.

6.2. Socket STREAM

En este caso, al ser las conexiones TCP orientadas a conexión, cada socket identifica una conexión entre dos hosts específicos. En todo proceso TCP el cliente inicia una solicitud de conexión hacia un servidor. El servidor habilita un socket servidor inicial(llamado socket de escucha) para la recepción de solicitudes de conexión de clientes; este socket está habilitado mientras esté habilitado el servidor. Cuando el servidor acepta una solicitud de conexión de un cliente que le llega al socket servidor, habilita un segundo socket (llamado socket de diálogo) para la transmisión/recepción de datos con ese cliente específico. El socket de diálogo escucha en la misma dirección y puerto que el socket de escucha, por lo que el socket de diálogo en el servidor queda identificado unívocamente por la tupla *IP/Puerto local/IP/Puerto remoto*.

De esa forma, un cliente TCP dispone de sólo un socket, y en cambio un servidor TCP dispone de tantos sockets como clientes conectados, mas un socket servidor.

Lado Cliente :

1. Creación del socket
2. Establecimiento de la conexión con el servidor (la dirección IP y el puerto son conocidos). Se usa la función "connect".
3. Envío y/o recepción de datos hacia/desde el servidor desde/hacia el socket. Se usan las funciones "send" (o "write") y "recv" (o "read").

4. Cierre del socket y cierre de la conexión.

En pseudocódigo:

```
Variable definition;

Main {
  set_destination_address DstAddr;
  SocketTCP=socket(AF_INET, SOCK_STREAM, 0);
  connect(SocketTCP, DstAddr);
  while true {
    set_data_to_send DataSend;
    send(SocketTCP, DataSend, 0);
    recv(SocketTCP, DataRecv, 0);
    make_something_with_recv_data DataRecv;
  }
  close(SocketTCP);
}
```

Lado Servidor :

1. Creación del socket
2. Asociación del socket con un puerto local (función "bind")
3. Espera por solicitudes de conexión. Se usa la función "listen".
4. Si llega una solicitud, la acepta y asigna un nuevo socket a ese cliente. Se utiliza la función "accept". Ese nuevo socket tiene asignada tanto la dirección y puerto local, como la dirección y puerto del cliente.
5. Envío y recepción de paquetes de ese cliente a través del nuevo socket. Se usan las funciones "send" (o "write") y "recv" (o "read").
6. Cierre del socket y de la conexión

En pseudocódigo:

```
Variable definition;

Main {
  set_local_port LocalPort;
  SocketTCP=socket(AF_INET, SOCK_STREAM, 0);
  bind(SocketTCP, LocalPort);
  listen(SocketTCP, MaxAllowedClients);
  while true {
    SocketClient=accept(SocketTCP, ClientAddr);
    recv(SocketClient, DataRecv, 0);
    make_something_with_recv_data DataRecv;
    set_data_to_send DataSend;
    send(SocketClient, DataSend, 0);
    close(SocketClient);
  }
}
```

6.3. Socket RAW

Los paquetes que se han creado con sockets RAW se pasan a los drivers de las interfaces sin modificar, o son recibidos desde esos drivers igualmente sin modificar. Los sockets RAW requieren por tanto una programación mas específica de lo que se pretende hacer, es decir, del tipo de protocolo que se quiere implentar. Básicamente su programación consiste en rellenar (o leer) parte de la cabecera MAC mediante el acceso a las estructuras "ethhdr" (capa 2) y el campo de datos MAC, es decir, si se pretende utilizar IP, habría que utilizar estructuras que definen la cabecera IP y rellenarlas.