1 PUENTES Y CONMUTADORES LAN

Autor: Rogelio Montañana

1.1 INTRODUCCIÓN

Existen muchas circunstancias en las que no se quiere o no se puede tener una sola red; en estos casos es posible tener múltiples redes locales interconectadas mediante el uso de unos dispositivos llamados **puentes**. Los puentes se encargan de capturar las tramas de una red local, y reenviarlas selectivamente a otra red local. Para esto analizan la dirección de origen y destino de la trama a nivel MAC. Algunas de las situaciones en las que puede ser conveniente utilizar puentes son las siguientes:

- **Interoperabilidad**: Se dispone de redes basadas en medios físicos diferentes. Por ejemplo en una empresa puede haberse empezado a tender una red Token Ring en unos edificios y Ethernet en otros.

- **Distancia**: Se necesita cubrir una distancia mayor que la que puede cubrirse con una red local (por ejemplo más de 4 Km en Ethernet a 10 Mb/s).

- **Número de ordenadores**: Se quiere conectar más equipos que los que se permiten en una red local (más de 1024 en Ethernet, o mas de 72-250 en Token Ring).

- **Tráfico**: Si existe una elevada cantidad de tráfico, principalmente de carácter local, se puede reducir éste dividiendo la red en varias mediante puentes. Por ejemplo si en una empresa cada departamento tiene su propio servidor mucho de su tráfico será local.

- **Fiabilidad**: Si se quiere evitar que un problema en un ordenador pueda colapsar toda la red (por ejemplo en Ethernet una tarjeta o transceiver averiado puede inutilizar toda la red). Si se divide la red por zonas el problema afectará a menos equipos.

- **Seguridad**: En una red local cualquier ordenador funcionando en modo *promiscuo* puede ver todas las tramas. La división en varias redes evita en cierta medida que los paquetes puedan ser vistos fuera de la red.

1.2 PUENTES

De acuerdo con la funcionalidad que desempeñan podemos clasificar los puentes en los siguientes tipos:

- Puentes transparentes

- Puentes remotos, también llamados encapsuladores

- Puentes traductores

- Puentes con encaminamiento desde el origen

Ahora veremos cada uno de ellos en detalle.

1.2.1 Puentes transparentes

Como su nombre indica, lo que se pretende con estos puentes es que puedan utilizarse sin alterar para nada el protocolo o la configuración de los ordenadores. Normalmente estos equipos no necesitan ningún tipo de configuración previa, actuando como dispositivos 'plug and play'.

Veamos como funciona un puente transparente. Supongamos un puente que une dos redes, LAN1 y LAN2. El puente tendrá dos interfaces físicas, cada una conectándole con cada una de las dos LANs. Al encender el puente éste empieza reenviando todas las tramas que recibe por LAN1 a LAN2, y viceversa. En todo momento el puente actúa en modo promiscuo, es decir, capturando todas las tramas que se envían

por cada una de las redes a las que está conectado, independientemente de cual sea la dirección de destino.

Además de reenviar las tramas de forma indiscriminada, el puente va silenciosamente extrayendo de cada una la dirección de origen y la dirección de destino; la de origen la anota en una tabla hash correspondiente a la LAN por la que ha llegado la trama, y la de destino la busca en la misma tabla. Supongamos que el puente recibe por la interfaz LAN1 una trama que lleva la dirección de origen A y la dirección de destino B. Primeramente el puente buscará en su tabla de direcciones si en la columna LAN1 aparece B; si es así sencillamente descartará la trama, ya que sabe que A y B están ambas en LAN1 y no hay ninguna necesidad de reenviar esa trama. Si B no aparece en la tabla de LAN1 el puente reenviará la trama a LAN2. En segundo lugar el puente actualizará su tabla de direcciones de LAN1 añadiendo A, salvo que ya exista tal entrada en cuyo caso simplemente actualizará el contador de tiempos asociado con dicha entrada. Es posible que B esté en LAN1 y el puente no le tenga aún 'fichado' (porque B no haya enviado aún ninguna trama), pero ante la duda el puente se 'cura en salud' y reenvía la trama por la otra interfaz. Esta estrategia de tirar por elevación enviando la información en caso de duda se denomina inundación (*flooding*)

El mecanismo utilizado por los puentes para averiguar que ordenadores tienen conectados en cada una de sus redes tiene algunas consecuencias que merece la pena destacar:

- Un ordenador 'tímido', es decir, que no emita ninguna trama, no puede ser localizado, por lo que los puentes enviarán por todas sus interfaces (excepto aquella por la que les ha llegado) cualquier trama dirigida a dicho ordenador. Sin embargo no es probable que un ordenador que recibe tráfico permanezca callado durante mucho tiempo (o de lo contrario pronto dejará de recibirlo, ya que la mayoría de los protocolos requieren alguna contestación, al menos de vez en cuando).

- Las tramas enviadas a direcciones multicast o broadcast (las que tienen a 1 el primer bit) siempre son retransmitidas por los puentes por todas sus interfaces, ya que en principio puede haber destinatarios en cualquier parte (los puentes no almacenan direcciones multicast en sus tablas).

A fin de adaptarse a cambios en la red (por ejemplo un ordenador es desenchufado físicamente de LAN1 y enchufado en LAN2), las entradas en las tablas de direcciones son eliminadas cuando han pasado varios minutos sin que la dirección correspondiente haya enviado ninguna trama.

Existen puentes multipuerta, es decir, con múltiples interfaces, que permiten interconectar varias LANs en una misma caja. El algoritmo es similar, manteniéndose en este caso una tabla de direcciones para cada interfaz. Las tablas se van llenando con las direcciones 'escuchadas' en cada interfaz; cuando se recibe una trama en cualquiera de las interfaces se busca la dirección de destino en la columna de dicha interfaz; si el destinatario se encuentra allí la trama simplemente se descarta, si no se busca en las columnas correspondientes a las demás interfaces; si se encuentra en alguna columna se manda a la interfaz correspondiente. Por último, si no se encuentra en ninguna de las tablas se envía a todas las interfaces excepto aquella por la que llegó (inundación).

Los puentes han de mantener una tabla de direcciones para cada una de sus puertas; la cantidad de memoria destinada a dichas tablas es limitada, y en redes grandes puede llegar a agotarse. Los fabricantes suelen especificar el número máximo de direcciones MAC que sus puentes son capaces de soportar. Algunos equipos se bloquean sin más explicaciones cuando se les llena la tabla de direcciones MAC.

1.2.1.1 Spanning Tree

En algunas situaciones es interesante unir dos LANs con más de un puente, normalmente por razones de fiabilidad o redundancia. Analicemos a la luz del algoritmo antes descrito que ocurriría si dos puentes unen dos LANs. Imaginemos LAN1 y LAN2, unidas por los puentes B1 y B2. Una estación en LAN1 emite una trama que lleva una dirección de destino F aun no registrada; al no aparecer en sus tablas B1 reenvía la trama a LAN2; lo mismo hace B2. A continuación B1 detecta en LAN2 la trama generada por B2 y, al no encontrar en sus tablas la dirección de destino (ya que F aún no esta 'fichado') la reenvía a LAN1. Análogamente B2 detecta en LAN2 la trama de B1 y al no estar F en su tabla de direcciones, la reenvía a LAN1. LAN1 recibe pues dos copias de una trama que ya tenía, y se repite el proceso: B1

recoge la trama de B2 y la retransmite; B2 hace lo mismo con la de B1, y así sucesivamente. El ciclo no tiene fin, por lo que una sola trama es suficiente para saturar ambas redes.

Para evitar este problema existe un mecanismo que permite a los puentes comunicarse entre sí, pasándose información sobre la topología de las conexiones existentes; una vez averiguada dicha topología los puentes desactivarán las conexiones redundantes para garantizar que haya un único camino (directo o indirecto) uniendo todas las redes, de forma que se evite la creación de bucles. Las conexiones que lógicamente se pongan fuera de servicio quedarán listas para entrar en funcionamiento si las conexiones activas fallan por algún motivo. El algoritmo se repite cada cierto tiempo, por lo que si alguno de los enlaces queda fuera de funcionamiento por algún motivo (por ejemplo una avería) en la siguiente ronda se habilitará algún camino alternativo que lo sustituya. El protocolo que permite esto se conoce como Spanning Tree Protocol (STP) y también como Spanning Tree Learning Bridge Protocol, y forma parte de la especificación IEEE 802.1D.

La topología de una interconexión de LANs con puentes podemos representarla con un grafo en el que los nodos son las LANs y los arcos los puentes que las unen. El algoritmo spanning tree consiste en dejar únicamente un camino para llegar a cada una de las redes, para lo cual se suprime del grafo toda unión que ocurra en sentido descendente entre ramas distintas del árbol. Este tipo de estructura es lo que se conoce como spanning tree (que podemos traducir como árbol de expansión), de ahí el nombre del protocolo.

Con el Spanning Tree es posible tener varios puentes conectando dos redes sin que se produzcan conflictos. En la práctica lo que ocurre sencillamente es que se inhabilitan todos los caminos posibles, menos uno. Los otros quedan como rutas alternativas dispuestas a entra en funcionamiento en caso de avería en la principal. No es posible con Spanning Tree tener varias conexiones activas al mismo tiempo, lo cual permitiría repartir el tráfico entre varios puentes, mejorando así el rendimiento de la conexión (como se hace por ejemplo en etherchannel).

Aunque forma parte del estándar 802.1D y debería formar parte de cualquier puente transparente, no todos los equipos implementan el protocolo spanning tree. Al construir una topología basta con que uno de los puentes que forman el bucle incorpore el spanning tree para que el puente ‘rompa’ el bucle y la red funcione correctamente.

1.2.2 Puentes remotos o ‘encapsuladores’

En ocasiones se tiene necesidad de conectar entre sí dos redes locales remotas como si fueran la misma LAN. Para esto se usa un tipo de puentes denominados puentes remotos. Los mecanismos básicos de funcionamiento son los mismos que para los puentes locales (transparentes y con encaminamiento desde el origen), salvo que el puente está constituido por dos 'medios puentes' interconectados por una línea dedicada cuya velocidad típicamente suele estar entre 64 Kb/s y 2,048 Mb/s. También se pueden unir los puentes remotos por redes X.25, Frame Relay o incluso radioenlaces.

El protocolo spanning tree también se utiliza en puentes remotos. Para representar topológicamente un puente remoto el enlace punto a punto se debe considerar como una LAN con un puente en cada extremo.

Dado que generalmente los puentes remotos se conectan mediante líneas de menor velocidad que las redes a las que enlazan, es frecuente que dicha conexión sea el factor limitante de las prestaciones de la red (aun cuando el algoritmo propio de los puentes evita que el tráfico local cruce al otro lado). Esto es especialmente crítico cuando se utilizan líneas de baja velocidad (por ejemplo 64 Kb/s) y mas aun cuando se trata de LANs grandes en las que el tráfico broadcast/multicast es importante (recordemos que este tipo de tráfico *siempre* atraviesa los puentes transparentes).

1.3 REDES LOCALES CONMUTADAS

En la discusión anterior sobre diversos tipos de puentes hemos supuesto casi todo el tiempo que el puente tenía únicamente dos interfaces. Hoy en día son muy populares los puentes transparentes con elevado número de puertas, habiendo modelos que pueden llegar a más de 500. Estos equipos suelen ir equipados

con chips VLSI diseñados específicamente para este tipo de tareas denominados ASIC (Application Specific Integrated Circuit), gracias a los cuales desarrollan su tarea con gran rapidez. A estos puentes multipuerta de alta velocidad se les suele llamar conmutadores LAN o switches LAN, ya que gracias al encaminamiento inteligente que realizan mediante sus tablas de direcciones actúan de hecho conmutando tramas entre sus múltiples puertas. También se les denomina a veces conmutadores de nivel 2 (nivel de enlace) por contraste con los llamados conmutadores de nivel 3, que realizan la conmutación al nivel de red, también llamados routers (normalmente el término conmutador de nivel 3 se utiliza para indicar un router que dispone de hardware específicamente diseñado para labores de routing y que por tanto realiza esta función de manera notablemente mas rápida que un router convencional). A las redes locales basadas en conmutadores se las suele llamar redes locales conmutadas (switched LANs).

En conmutadores Ethernet se suele decir que cada puerto constituye un *dominio de colisiones* independiente, ya que las colisiones que se producen en un puerto no afectan a los demás.

Los conmutadores Token Ring pueden funcionar según el principio de puente por encaminamiento desde el origen o puente transparente.

Con conmutadores LAN de elevado número de puertas es posible incrementar de forma notable la capacidad de una red local con una modificación mínima de la misma. Por ejemplo, si en una red se sustituye un hub o concentrador Ethernet de 24 puertas por un conmutador LAN de 24 puertas, el rendimiento máximo teórico se ha multiplicado por 24 sin haber tenido que modificar el cableado o las tarjetas de red de los ordenadores.

En el caso de redes Ethernet el uso de conmutadores tiene un efecto adicional en la mejora del rendimiento; recordemos que en redes CSMA/CD la eficiencia disminuye a medida que aumenta el número de ordenadores, por lo que si se divide una red en varias mediante un conmutador se consigue un mejor rendimiento en cada una de ellas al soportar un número más reducido de equipos.

Un problema que se presenta con los conmutadores LAN es que se pueden producir situaciones de congestión, para las que no disponen de muchos mecanismos de control pues funcionan únicamente a nivel de enlace. Por ejemplo, si en un conmutador de puertos a 10 Mb/s diferentes puertos quieren enviar tráfico a la vez a un mismo puerto a 10 Mb/s cada uno y esta situación se mantiene durante bastante tiempo el conmutador puede agotar el espacio en buffers disponible, perdiéndose tramas a partir de ese momento. En el caso de Ethernet este problema se resuelve mediante el control de flujo.

Al igual que los puentes, los conmutadores LAN han de mantener en memoria las tablas de direcciones MAC en cada una de sus puertas. Las especificaciones de un conmutador LAN indican normalmente el número máximo de direcciones que puede soportar. Si el número de equipos activos en la red es superior a este número máximo el rendimiento de la red se ve afectado ya que las entradas en la tabla de direcciones caducan con demasiada rapidez, provocando tráfico adicional en la red debido al mecanismo de inundación

1.3.1 Almacenamiento y Reenvío vs ‘Cut-Through’

Dado que los conmutadores LAN son esencialmente puentes con muchas puertas su funcionamiento normal requiere que antes de retransmitir una trama la reciban en su totalidad para que puedan comprobar el CRC y descartarla en caso de que éste sea erróneo. Esto obliga a que los conmutadores funcionen como dispositivos de almacenamiento y reenvío. En el caso de tramas grandes el requerimiento de comprobación del CRC introduce un retardo notable en la propagación de la trama, a menudo superior al que introduce el propio proceso de conmutación; además si la trama ha de atravesar varios conmutadores el almacenamiento y reenvío se ha de realizar en cada uno de ellos. Si no se hiciera la comprobación del CRC la velocidad de conmutación podría aumentarse notablemente y el retardo reducirse, ya que el conmutador podría empezar a enviar los bits tan pronto hubiera recibido la dirección MAC a la que va dirigida la trama. Esto es lo que se denomina funcionamiento en modo ‘Cut-Through’ (literalmente abrirse camino) implementado en algunos conmutadores. El problema del funcionamiento Cut-Through es que se pueden estar produciendo errores de CRC que pasen inadvertidos. En este caso las tramas erróneas serán descartadas por el host de destino, por lo que no hay riesgo de que se interpreten como correctos datos erróneos, pero aun así la situación es perjudicial para la red puesto que se esta ocupando ancho de banda con tráfico inútil.

Para evitar este problema cuando los conmutadores funcionan en modo Cut-Through siguen comprobando el CRC; cuando se produce un error no pueden descartar la trama puesto que esta ya ha sido transmitida, pero sí que pueden poner a la estación emisora bajo sospecha y a partir de ese momento pasar a funcionar en modo almacenamiento y reenvío de forma selectiva, únicamente para las tramas que tengan como dirección de origen la de la estación sospechosa; si se comprueba más tarde que el error detectado fue algo esporádico se volverá al modo Cut-Through para esa estación, de lo contrario se la mantendrá en estado vigilado, es decir en modo almacenamiento y reenvío; esta forma de proceder se basa en la hipótesis de que una estación que genera tramas correctas tiene una alta probabilidad de seguir generando tramas correctas, mientras que una que genera tramas erróneas, normalmente por algún problema de tipo físico, tendrá una probabilidad mayor de seguir generando tramas erróneas.

1.4 CONMUTACIÓN EN REDES ETHERNET

La introducción de los conmutadores LAN permite una serie de mejoras en el funcionamiento de las redes locales que comentaremos a continuación Aunque nos centraremos en el caso concreto de Ethernet, por ser la más común de las tecnologías actuales, la mayoría de estos principios son aplicables a otros protocolos MAC, aunque su desarrollo está en general más retrasado que en Ethernet.

1.4.1 Transmisión full dúplex

Una red Ethernet puede funcionar en modo full dúplex si se dan simultáneamente las tres condiciones siguientes:

- Que el medio físico permita transmisión full-dúplex; esto se cumple en todos los casos habituales excepto 10BASE5, 10BASE2 y 100BASE-T4.
- Que sólo haya dos estaciones conectadas entre sí (por ejemplo conmutador-conmutador, conmutador-host o host-host).
- Que los adaptadores de red y transceivers de ambos equipos soporten el funcionamiento en modo full-dúplex.

Con sólo dos estaciones en la red y un canal de comunicación independiente para cada sentido el medio de transmisión no es compartido, por lo que no hay ninguna necesidad de protocolo MAC; se puede por tanto inhabilitar el CSMA/CD y manejar el medio físico como si se tratara de un enlace punto a punto full dúplex de la velocidad de la red (10, 100 ó 1000 Mb/s). Al no haber colisiones en full dúplex no rige la limitación de distancia impuesta por el 2τ de la Ethernet 'tradicional' o half dúplex. La única restricción es la que viene impuesta por la atenuación de la señal según el medio físico utilizado. Por ejemplo 100BASE-FX, que tiene una distancia máxima en half dúplex de 412 m, puede llegar en full dúplex a 2 Km.

Cuando una estación se configura en modo full dúplex sin que se den las tres condiciones antes mencionadas el rendimiento decae de forma espectacular, ya que se producen colisiones que no son detectadas.

Aprovechando la supresión de la restricción en distancia debida al CSMA/CD algunos fabricantes suministran transceivers láser que utilizando fibra monomodo en tercera ventana permiten llegar en Ethernet a distancias de más de 100 Km, a cualquiera de las velocidades habituales (10, 100 ó 1000 Mb/s). Estos equipos no están estandarizados por lo que si se utilizan es conveniente poner en ambos extremos sistemas del mismo fabricante, o asegurarse previamente de su compatibilidad e interoperabilidad. Mediante dispositivos regeneradores de la señal de bajo costo es posible extender este alcance en principio indefinidamente, habiéndose hecho pruebas a distancias de hasta 800 Km. De esta forma Ethernet se convierte en una alternativa interesante en redes de área extensa. Evidentemente cuando se transmite la señal a grandes distancias se introduce un retardo, a veces importante, debido a la velocidad de propagación en el medio físico.

Además de aumentar el rendimiento y permitir distancias mayores el uso de full dúplex simplifica el funcionamiento, puesto que se suprime el protocolo MAC. El aumento en el rendimiento obtenido por la transmisión full dúplex normalmente sólo es significativo en conexiones conmutador-conmutador o conmutador-servidor. En un equipo monousuario el full dúplex supone una mejora marginal ya que las aplicaciones casi siempre están diseñadas para dialogar de forma half-dúplex[1].

En Gigabit Ethernet full dúplex se suprimen la extensión de protadora y las ráfagas de tramas, puesto que son innecesarias. Por tanto las ventajas en Gigabit Ethernet full dúplex son aun mayores que las obtenidas en Ethernet o Fast Ethernet, hasta el punto que algunos autores dudan de que lleguen a extenderse en el mercado productos Gigabit Ethernet half dúplex [13].

Para permitir el funcionamiento full dúplex en Gigabit Ethernet sin tener que recurrir a la conmutación por puerta, que podría resultar excesivamente cara en algunas situaciones, se han ideado recientemente unos dispositivos que son algo intermedio entre los concentradores y los conmutadores, denominados 'buffered repeaters', 'buffered distributor', 'full duplex repeater' o 'full duplex distributor'. Un 'buffered repeater' es un conmutador que carece de tabla de direcciones MAC, por lo que cualquier trama que recibe la replica en todas sus interfaces por inundación, actuando de la misma forma que un conmutador cuando recibe una trama dirigida a una dirección que no aparece en sus tablas. Por tanto desde este punto de vista un buffered repeater actúa como un concentrador. Sin embargo a diferencia del concentrador, que reproduce la trama bit a bit, el buffered repeater la almacena en su totalidad antes de reenviarla, actuando como un puente; esto le permite funcionar en modo full dúplex, con lo que no sufre las limitaciones de distancia del half dúplex; tampoco tiene que detectar colisiones o generar extensiones de portadora, lo cual simplifica la electrónica asociada. Se espera que el buffered repeater sea bastante más barato de fabricar que un conmutador de Gigabit Ethernet, ya que debido a su funcionamiento el tráfico total agregado de un buffered repeater está limitado a 1 Gb/s, lo cual simplifica el diseño respecto a un conmutador normal, que en principio debe poder soportar un tráfico total agregado igual a la suma del de todas sus interfaces. Estrictamente hablando los buffered repeaters no son parte del estándar Gigabit Ethernet; y podrían aplicarse igualmente a Ethernet, Fast Ethernet u otras redes 802 ya que su principio de funcionamiento es el mismo que el de los conmutadores (que corresponde al estándar 802.1D. de puentes transparentes).

A la vista de estos desarrollos, que muy probablemente dejarán en desuso la Gigabit Ethernet half dúplex, cabría preguntarse por que razón el subcomité 802.3z emprendió la ardua tarea de estandarizar Gigabit Ethernet half dúplex, con toda la complejidad que esto supuso (definición del concepto de extensión de portadora y ráfagas de tramas, por ejemplo). La explicación es de tipo político: para que el grupo que definía Gigabit Ethernet pudiera constituirse como un subcomité de 802.3 era necesario que contemplara el uso de CSMA/CD (y por ende el funcionamiento half dúplex), ya que ésta es la característica esencial que identifica al subcomité 802.3 dentro del comité 802. En caso de no haber contemplado el CSMA/CD el grupo de Gigabit Ethernet habría tenido que solicitar al IEEE la creación de un nuevo subcomité 802; esto habría retrasado considerablemente la estandarización, cosa no deseada por ninguno de los participantes en el grupo.

1.4.2 Control de flujo

El funcionamiento full dúplex se introdujo inicialmente como una extensión no estándar por parte de varios fabricantes. Cuando en 1997 el subcomité 802.3x lo estandarizó incluyó además una nueva funcionalidad, el control de flujo, que en Ethernet se implementa mediante el comando PAUSE. El receptor puede en cualquier momento enviar al emisor un comando PAUSE indicándole por cuanto tiempo debe dejar de enviarle datos. Durante ese tiempo el receptor puede enviar nuevos comandos PAUSE prolongando, reduciendo o suprimiendo la pausa inicialmente anunciada. Con esto se pretende evitar el desbordamiento de los buffers del receptor con el consiguiente descarte de tramas, lo cual causaría males mayores. El control de flujo está especialmente indicado en el caso de conmutadores, sobre todo si forman parte del backbone de una red. Puede establecerse de forma asimétrica, por ejemplo en una conexión conmutador-host puede que de flujo sobre el host, pero no a la inversa.

[1] Con la notable excepción de la videoconferencia punto a punto, donde la información fluye de forma full-dúplex bastante simétrica.

Desde el punto de vista de Ethernet el control de flujo se implementó como un nuevo tipo de protocolo de red. Para que funcione correctamente es fundamental que las tramas de control de flujo sean rápidamente identificadas por los conmutadores, por lo que esta función se implementa normalmente en hardware. Esto es mucho más fácil de hacer si los comandos de control de flujo pueden identificarse en la cabecera MAC de la trama y no en la LLC, por lo que se vio que era más eficiente utilizar el formato DIX que el 802.2, ya que permitía poner el campo *tipo* en la cabecera MAC. Se propuso entonces al comité 802.3 un nuevo formato de trama, que coincidía precisamente con el formato DIX. El comité aceptó la propuesta, pero ya puestos decidió estandarizar el nuevo formato para todos los posibles protocolos de Ethernet, no sólo para el control de flujo. Como consecuencia de esto desde 1997 los dos formatos de trama: el 802.2 y el DIX son 'legales' según el estándar 802.3; la distinción entre ambos se hace según el valor del campo tipo/longitud, como ya era habitual en todas las implementaciones.

Cuando en una conexión full dúplex se implementa control de flujo cada equipo debe disponer de espacio suficiente en buffers para aceptar todo el tráfico proveniente del otro extremo en caso de que este envíe un comando PAUSE. Dicho espacio ha de ser como mínimo igual a la cantidad de datos que el otro equipo pueda transmitir durante el tiempo de ida y vuelta (ya que mientras el comando PAUSE viaja se considere conveniente que el conmutador ejerza control hacia el emisor éste continúa enviando datos). Dicho de otro modo, hay que reservar un espacio en buffers igual al doble de lo que 'cabe' en el cable. Por ejemplo en una conexión 1000BASE-LX full dúplex de 5 Km (tiempo de ida y vuelta 50 µs) se deberá disponer de 50.000 bits (6,1 KBytes) para buffers. En el caso de conexiones Ethernet de larga distancia (superiores a las permitidas por el estándar) puede ocurrir que el espacio en buffers sea insuficiente para albergar todos los datos recibidos cuando se emite un comando PAUSE.

1.4.3 Autonegociación

Los medios físicos 1000BASE-T y 100 BASE-TX utiliza el mismo conector que 10BASE-T. Esto permite aprovechar la misma instalación de cableado y latiguillos para las tres redes, lo cual da gran flexibilidad. Sin embargo desde el punto de vista del usuario supone también la posibilidad de cometer errores, ya que la compatibilidad de conectores no garantiza la compatibilidad de medios físicos. El funcionamiento full dúplex y el control de flujo plantean un problema parecido, ya que al ser partes opcionales del estándar no tienen por qué estar disponibles en todos los equipos. Esta diversidad de posibilidades en cuanto a velocidad y funcionalidades disponibles en el mismo conector requiere una laboriosa tarea de documentación para asegurar el correcto funcionamiento de una red. Para simplificar esta tarea se añadió al estándar 802.3 una característica denominada autonegociación, consistente en que cuando dos equipos se conectan intercambian unas señales anunciando sus posibilidades, de acuerdo con un protocolo especial. Esto les permite 'negociar' y funcionar de la forma compatible más eficiente posible[2].

En el caso de un conector RJ-45 se negocia en primer lugar el medio físico en el siguiente orden:

1.- 1000BASE-T
2.- 100BASE-T2
3.- 100BASE-TX
4.- 100BASE-T4
5.- 10BASE-T

Una vez acordada la velocidad se negocia el funcionamiento half/full-dúplex, y por último el control de flujo y si éste se establece con carácter simétrico o asimétrico.

En el caso de la fibra óptica el medio físico (es decir la velocidad) no es negociable, ya que la longitud de onda cambia (excepto entre 10BASE-FL y 1000BASE-SX, que como utilizan la misma ventana pueden negociar entre sí). En este caso sólo se negocia el funcionamiento full dúplex y el control de flujo.

La autonegociación es opcional, por lo que conviene comprobar que esté soportada por los equipos antes de dejarlo todo en manos del funcionamiento automático. Aun en el caso de que se soporte autonegociación es conveniente comprobar que se ha pactado el funcionamiento más eficiente posible,

[2] La idea es parecida a la utilizada en los módems de red telefónica conmutada, que mediante un proceso de negociación acuerdan funcionar según la velocidad más alta soportada por ambos.

sobre todo en conexiones entre conmutadores, ya que en este caso la ausencia de control de flujo o el funcionamiento half dúplex puede suponer una diferencia importante de rendimiento.

En algunos casos la autonegociación puede causar problemas. Por ejemplo si conectamos mediante cableado categoría 3 dos equipos que soportan 100BASE-TX y 100BASE-T4 las señales de autonegociación, que tienen unos requerimientos ínfimos en cuanto a ancho de banda, se transmiten perfectamente en el cable, pero no verifican su categoría (ya que incluir esta verificación en el transceiver sería muy costoso). Por tanto la negociación dará como resultado el funcionamiento en 100BASE-TX. Una conexión 100BASE-TX sobre cableado categoría 3 no funcionará o lo hará con muchos errores, por lo que en este caso será necesario configurar manualmente los equipos y forzar el uso de 100BASE-T4 para que la red funcione correctamente. Afortunadamente esta situación se da raramente ya que muy pocos equipos implementan 100BASE-T4.

La autonegociación sólo es posible en conmutadores y hosts, no en concentradores, ya que estos requieren funcionar a la misma velocidad en todos sus puertos, y siempre en modo half dúplex. En el mercado existen equipos denominados 'concentradores' con autonegociación 100/10 por puerto; estos equipos en realidad son internamente un conmutador con dos puertos, uno de 10 y uno de 100 Mb/s, que tiene un concentrador de 10 y uno de 100 Mb/s conectados a cada puerto del conmutador; los puertos físicos se adscriben internamente a uno u otro concentrador en función de la velocidad del equipo que se conecta.

1.4.4 Agregación de enlaces

La agregación de enlaces, también llamada 'trunking' o multiplexado inverso, es una técnica desarrollada inicialmente por la empresa Kalpana para utilizar varios enlaces Ethernet full-dúplex en la comunicación entre dos equipos, realizando reparto del tráfico entre ellos. Hoy en día esta funcionalidad es ofrecida por multitud de fabricantes para todas las velocidades de Ethernet.

En principio según el estándar si dos conmutadores se unen por dos enlaces el protocolo Spanning Tree desactivará uno de ellos, dejándolo preparado para entrar en funcionamiento en caso de fallo del otro[3]. La agregación de enlaces requiere deshabilitar el Spanning Tree entre los enlaces que se agregan, para así poder repartir el tráfico entre ellos. Los enlaces pueden ser de 10, 100 ó 1000 Mb/s, pero han de ser todos de la misma velocidad.

Además de permitir acceder a capacidades superiores cuando no es posible cambiar de velocidad por algún motivo, la agregación de enlaces permite un crecimiento gradual a medida que se requiere, sin necesidad de cambios traumáticos en las interfaces de red o en la infraestructura. Aunque existen en el mercado productos que permiten agregar hasta 32 enlaces full dúplex, parece que cuatro es un límite razonable, ya que al aumentar el número de enlaces la eficiencia disminuye, y por otro lado el costo de las interfaces aconseja entonces pasar a la velocidad superior en vez de agregar enlaces.

La estandarización de la técnica de agregación de enlaces ha sido llevada a cabo por el grupo de trabajo 802.3ad, que terminó la elaboración del documento en fase de borrador en 1998. Su ratificación es ya inminente. Actualmente existen multitud de productos en el mercado conformes con dicho borrador que pueden interoperar entre sí.

La agregación de enlaces requiere evidentemente el uso de múltiples cables, cosa que no siempre es posible, sobre todo si se trata de enlaces a gran distancia. En el caso de fibra óptica el problema puede resolverse mediante la técnica conocida como WDM (Wavelength Division Multiplexing) que consiste en multiplexar varias señales en una misma fibra utilizando longitudes de onda ligeramente distintas dentro de la misma ventana (equivalente a usar luz de diferentes 'colores'). La WDM se utiliza desde hace algún tiempo en enlaces de área extensa donde el mejor aprovechamiento de las fibras compensa el elevado costo de los equipos. Recientemente han aparecido dispositivos WDM a precios asequibles (unos 2 millones de pesetas) que multiplexan cuatro señales Gigabit Ethernet en una misma fibra, pudiendo así transmitir 4 Gb/s full dúplex por un par de fibras. Como detalle curioso comentaremos que con WDM también es posible multiplexar el canal de ida y el de vuelta en una misma fibra, con lo que es posible tener comunicación full dúplex por una sola fibra.

[3] En el caso de que ninguno de los dos conmutadores implementara el protocolo Spanning Tree se produciría un bucle que colapsaría e inutilizaría ambos enlaces.

1.5 DISEÑO DE REDES LOCALES

1.5.1 Planificación de capacidad. Dimensionamiento

El responsable de una red local tiene a menudo que plantearse la mejora de partes de la misma para evitar que los niveles de saturación produzcan una merma en la calidad de servicio percibida por los usuarios, cosa que no debería ocurrir en una red local donde en principio el ancho de banda está disponible a bajo costo.

Para esta toma de decisiones es necesario disponer de parámetros de medida objetivos, que nos permitan comparar los niveles de calidad de servicio de acuerdo con criterios homogéneos para toda la red. Como ya hemos comentado las colisiones de una red Ethernet no son en sí mismas una medida adecuada del grado de saturación de una red, ya que su interpretación está íntimamente ligada al tamaño de trama medio de una red.

Un parámetro mucho más apropiado es el nivel de ocupación medio de la red. Este valor puede obtenerse a partir de los contadores de tráfico de las interfaces en los diversos dispositivos de la red (conmutadores, routers, etc.), por ejemplo cada cinco minutos vía SNMP. De esta forma es posible calcular y representar gráficamente el tráfico medio en una red a intervalos de cinco minutos. Una vez hemos recopilado esta información para todas las interfaces que constituyen nuestra red local podemos comparar los valores y aplicar criterios objetivos para decidir donde es pertienente adoptar medidas para ampliar la capacidad.

Según Seifert [13], [16] en el caso de una red local utilizada para aplicaciones típicas de ofimática con decenas de estaciones por red se puede considerar que se da una carga excesiva en la red si se da alguna de las siguientes circunstancias:

- Se supera el 50% de ocupación durante 15 minutos, o bien
- Se supera el 20-30% durante una hora, o bien
- Se supera el 10-20% durante ocho horas.

En principio una red podría estar al 100% de ocupación durante cinco minutos, y eso no sería motivo para plantearse un aumento de capacidad. La razón es la siguiente: esa ocupación podría estar provocada por un usuario que transfiere un fichero grande (por ejemplo 400 Mbytes en una red de 10 Mb/s). Si ese tipo de utilización es esporádico normalmente el tiempo de respuesta será aceptable, y si es frecuente provocará que se supere alguno de los umbrales antes mencionados.

Una posible excepción a la regla anterior serían las redes en las que se utilicen aplicaciones en tiempo real (vídeoconferencia o vídeo bajo demanda, por ejemplo). En este caso el criterio debería ser mas exigente (por ejemplo 50% de ocupación durante 5 minutos) y aun así se pueden producir colapsos momentaneos. En realidad si se quiere utilizar este tipo de aplicaciones con garantías en una red Ethernet es preciso utilizar conmutadores hasta el puesto del usuario final.

1.5.2 Diseño de redes Ethernet

Gracias a los desarrollos habidos en los últimos años Ethernet presenta una enorme gama de posibilidades: diversos medios físicos (cobre y fibra), velocidades (10/100/1000 Mb/s) y modos de funcionamiento (puertos compartidos/conmutados, transmisión half/full dúplex y agregación de enlaces). Esto da una gran versatilidad que permite diseñar una red local completa cualesquiera que sean las necesidades que se presenten. Vamos a dar en este apartado algunas indicaciones generales sobre aspectos a tener en cuenta al diseñar una red local basada en Ethernet.

En primer lugar se plantea la disyuntiva del medio compartido o conmutado, es decir, si se debe conectar los equipos de usuario final a concentradores (hubs) o directamente a conmutadores.

Analizando la evolución a lo largo de varios años del costo por puerto de los concentradores frente a los conmutadores observamos que la diferencia es cada vez menor, como se aprecia en la tabla 1.1.

Tipo de red	1991	1993	1996	1998
10 Mb/s compartidos	15-30	7-15	3-10	1,5-6
10 Mb/s conmutados	150-200	40-90	15-30	4-10
Ratio 10 Mb/s	8:1	6:1	3:1	2:1
100 Mb/s compartidos			15-30	5-10
100 Mb/s conmutados			70-150	10-20
Ratio 100 Mb/s			5:1	2:1
1 Gb/s compartidos				(70-150)
1 Gb/s conmutados				(100-300)
Ratio 1 Gb/s				2:1

Tabla 1.1.- Precio por puerto (en miles de pesetas) de Ethernet compartida y conmutada. Datos obtenidos de [13].

En 1991 la relación para Ethernet 10 Mb/s conmutado:compartido era de 8:1, mientras que en 1998 se había reducido a 2:1. Una evolución análoga ha ocurrido con Fast Ethernet, que también se encuentra actualmente en 2:1. Es previsible que para Gigabit Ethernet se mantenga una relación similar (en el caso de que se lleguen a desarrollar concentradores de Gigabit Ethernet).

Esta evolución se explica porque el costo de la electrónica de conmutación disminuye a mayor velocidad que el del resto de componentes. Extrapolando los datos de la tabla anterior podemos esperar que el ratio descienda aún más en el futuro. Si además tenemos en cuenta que el costo de la instalación física (cableado) y de las interfaces de red son los mismos en ambos casos, el ratio es aún menor. Por todos estos motivos hoy en día se considera que el diseño de una red Ethernet debe basarse normalmente en el uso de conmutación a nivel del usuario final, es decir se recomienda la completa supresión de los concentradores.

El uso de conmutadores permite suprimir totalmente el protocolo CSMA/CD, y eventualmente extender el funcionamiento en modo full dúplex a toda la red. El uso de la transmisión full dúplex es especialmente importante en el caso de conexiones conmutador-conmutador, conmutador-router y conmutador-servidor.

Respecto a los medios físicos disponemos básicamente de tres alternativas cuyas distancias y costo relativo aparecen en la tabla 1.2.

Medio físico	10 Mb/s	100 Mb/s	1000 Mb/s	Costo relativo
Cobre UTP-5	150 m	100 m	100 m	1
F.O. 1ª ventana	2 Km	500 m	275-550m	2
F.O. 2ª ventana	-	2 Km	550m-5Km	6

Tabla 1.2.- Medios físicos más comunes en Ethernet

A pesar del desarrollo de optoelectrónica VCSEL de bajo costo en primera ventana, las interfaces en fibra óptica seguirán siendo más caras que las de UTP-5. Por tanto es preferible utilizar cable de cobre siempre que las distancias lo permitan, salvo que las condiciones ambientales aconsejen otra cosa.

En cuanto a la elección de velocidad de la red, esto dependerá evidentemente del tipo de aplicaciones y de las necesidades. En la tabla 1.3 damos unas recomendaciones orientativas en función del tipo de equipo a conectar.

Equipo a conectar	Tipo de conexión aconsejada
Puesto de trabajo	o 10BASE-T conmutada full duplex o 100BASE-TX conmutada full duplex
Servidor	o 100BASE-TX conmutada full duplex o 2, 3 ó 4 100BASE-X conmutada full duplex (etherchannel, 802.3ad) o 1000BASE-T conmutada full duplex con buffered repeater
Backbone (conmutador-conmutador)	o 100BASE-X conmutada full duplex o 2, 3 ó 4 100BASE-X conmutada full duplex (etherchannel, 802.3ad) o 1000BASE-X conmutada full duplex o 2, 3 ó 4 1000BASE-X conmutada full duplex (etherchannel, 802.3ad)

Tabla 1.3.- Recomendaciones de diseño de una red local Ethernet

1.5.3 Redes locales virtuales (VLANs)

Una de las grandes virtudes de los puentes y los conmutadores es su sencillez de manejo. Debido a su funcionamiento transparente es posible realizar una compleja red, incluso con enlaces WAN si se utilizan puentes remotos, sin tener que configurar ningún router. A finales de los ochenta se puso de moda la idea de desarrollar grandes redes, incluso a nivel de redes nacionales, basadas únicamente en el uso de puentes transparentes.

Sin embargo pronto se vio que esta estrategia tenía dos inconvenientes serios:

- Los puentes propagan el tráfico broadcast y multicast; generalmente los protocolos orientados a redes locales hacen un uso exhaustivo de este tipo de tramas, especialmente las broadcast, para anunciar todo tipo de servicios (a veces se les llama por esto 'protocolos charlatanes'); incluso IP, que no es especialmente pródigo en mensajes broadcast los emplea para la resolución de direcciones (protocolos ARP, RARP, BOOTP y DHCP). La proliferación de tráfico broadcast en una red es especialmente grave más que por el ancho de banda desperdiciado por el consumo de ciclos de CPU que se produce en todos los ordenadores de la red; este no es el caso con las tramas multicast, ya que cuando una trama multicast no incumbe a una estación (es decir, dicha estación no pertenece a dicho grupo multicast) es descartada por la interfaz de red local.

- La transparencia de los puentes hace difícil establecer mecanismos de control, protección y filtrado de tráfico, por lo que las redes muy grandes basadas en puentes se hacen inmanejables. Además en los casos en que quieren controles o mecanismos de gestión se han de utilizar direcciones MAC que no son agregables, es decir no existe ningún prefijo común en la dirección MAC que permita referirse o identificar una parte de la red, ya que la asignación no ha seguido ningún criterio geográfico ni se corresponde con la topología de la red.

Como consecuencia de esto la creación de grandes redes locales está desaconsejada y es práctica habitual en estos casos separar mediante routers las diversas partes de la red; este es el caso en un campus o gran edificio, por ejemplo. Los routers, al actuar a nivel de red, aíslan las tramas broadcast y multicast y facilitan la gestión al realizar una agregación de las direcciones de nivel de red

Es difícil fijar un número concreto para el máximo de nodos que debe haber en una red local, ya que este depende de las características de la red y los protocolos utilizados, pero a título orientativo diremos que en redes que utilizan un protocolo relativamente discreto en cuanto al tráfico broadcast (como es el caso de IP) en torno a 800 ordenadores como máximo por red local es un número razonable. Si la misma red local maneja varios protocolos de nivel de red simultáneamente el número máximo no debería estar por encima de los 200-300. Estos últimos valores de 200-300 ordenadores también se aplican cuando se utiliza alguno de los protocolos calificados de 'charlatanes', como AppleTalk o NetBios (NetBios es el protocolo de red utilizado comúnmente por los sistemas operativos Windows de Microsoft, aunque estos servicios también se pueden soportar sobre TCP/IP). Muchos programas analizadores permiten medir la cantidad de tráfico broadcast de una red (en paquetes por segundo) pudiendo incluso disparar una alarma cuando esta cantidad supera un determinado valor umbral fijado como excesivo por el usuario. El uso de estas herramientas en una red nos permitirá medir el tráfico broadcast de cada zona y tomar así las

decisiones apropiadas sobre una posible división o reestructuración de la red en base a datos objetivos y cuantificables.

Dividir una red local con criterios geográficos resulta relativamente sencillo, ya que normalmente la topología del cableado nos permite realizar esa división de manera directa. Por ejemplo supongamos que queremos dividir en varias una red local que abarca el campus de una universidad. Si decidimos crear una red local por edificio conectaremos entre sí los conmutadores de cada edificio y a su vez conectaremos cada edificio a una interfaz diferente del router que interconecte todo el campus. Sin embargo a menudo se requiere realizar una división lógica de acuerdo a criterios funcionales, que no siempre coinciden con la ubicación física. Por ejemplo en el caso de una universidad se podría pensar por razones de eficiencia y seguridad en crear una red para investigación, otra para docencia y otra para tareas administrativas y de gestión. Normalmente habrá varios edificios en los que habrá que dotar una serie de puestos de cada una de las tres redes mencionadas, en cuyo caso habría que instalar en los correspondientes armarios de cableado conmutadores independientes e interconectarlos entre sí por separado. Esto provoca una red compleja y muy cara, ya que en muchos casos habrá equipos infrautilizados.

La solución a este problema es la creación de redes locales virtuales, o VLANs. Podemos pensar en las VLANs como una forma de realizar una partición lógica de un conmutador en otros más pequeños, de forma que aunque se trata de un solo equipo dividimos los puertos en grupos que son completamente independientes entre sí. Esta funcionalidad está disponible hoy en día en la mayoría de los conmutadores del mercado, excepto en los de gama más baja.

Supongamos, siguiendo con nuestro supuesto anterior, que hemos decidido en efecto dividir la red de campus en tres VLANs que denominamos I (de investigación), D (de docencia) y A (de administración); en un momento dado nos encontramos en un armario de cableado configurando un conmutador de 16 puertos y se nos plantea la necesidad de suministrar servicio a 4 equipos de la VLAN I, 4 de la D y 4 de la A. Podríamos asignar por ejemplo los puertos 1 a 4 a la VLAN I, 5 a 8 a la VLAN D y 9 a 12 a la VLAN A, dejando los puertos 13 a 16 libres para futuras ampliaciones (los puertos de una misma VLAN no tienen por que estar contiguos, aunque a menudo se hace así por sencillez). A partir de ese momento el conmutador se comportará como cuatro conmutadores ‘virtuales’ de 4 puertos cada uno (los correspondientes a las tres VLANs y un cuatro correspondiente a los puertos no asignados). De esta forma podemos ir asignando puertos a una u otra VLAN de forma flexible en función de las necesidades.

Nos queda por resolver la conexión de nuestras tres VLANs con el resto de la red. Una posibilidad sería asignar los puertos 13, 14 y 15 a cada una de las tres VLANs y conectarlos con tres cables a tres puertos del conmutador principal del edificio, asignados a las tres VLANs. Siguiendo este sistema llegaríamos al router del campus donde un conmutador con puertos en las tres VLANs se conectaría a tres interfaces físicas diferentes del router. Aunque físicamente las tres VLANs comparten los conmutadores sigue habiendo tres redes separadas en el cableado, ya que nunca viajan por un mismo cable tramas de VLANs diferentes.

Cabría pensar en un nivel adicional de optimización en el que se compartiera un mismo cable para diferentes VLANs. Esto permitiría un ahorro considerable en el número de puertos consumidos en los enlaces troncales o de agregación, especialmente cuando se manejan muchas VLANs; por ejemplo en nuestro caso podríamos emplear solo un puerto (digamos el 13) para conectar las tres VLANs, liberando así los puertos 14 y 15 para otros usos. Esto se denomina configurar un enlace ‘trunk’ o troncal. Como es lógico los enlaces Trunk suelen ser de mayor capacidad que los puertos normales del conmutador ya que soportan un tráfico más elevado; por ejemplo en un conmutador de puertos a 10 Mb/s el enlace trunk típicamente será de 100 Mb/s y en uno con puertos de 100 Mb/s será de Gigabit Ethernet.

Los enlaces Trunk suponen un cambio importante en el funcionamiento de los conmutadores, ya que al mezclar tramas de diferentes VLANs por el mismo cable es preciso marcarlas o etiquetarlas de alguna manera a fin de poder entregarlas a la VLAN adecuada en el otro extremo. El marcado se hace añadiendo un campo nuevo en la cabecera de la trama MAC, lo cual hace que el tamaño de la trama Ethernet supere ligeramente la longitud máxima de 1500 bytes en algunos casos, ya que un conmutador puede recibir una trama de 1500 bytes y si la ha de enviar por un enlace trunk tendrá que incorporarle la etiqueta correspondiente (en ningún caso está permitido fragmentar la trama original). Los primeros sistemas de etiquetado de tramas eran propietarios, por lo que los enlaces trunk solo podían interoperar entre equipos del mismo fabricante. Hoy en día existe un formato estándar para colocar las etiquetas de VLAN que es el

conocido como IEEE 802.1q que es el que utilizan prácticamente la totoalidad de los equipos actuales. De esta forma es posible diseñar complejas redes con VLANs utilizando equipos de diferentes fabricantes

Es interesante comentar las consecuencias que la creación de VLANs tiene sobre el funcionamiento del protocolo Spanning Tree. En principio en un conmutador todos los puertos se encuentran asignados a la VLAN por defecto y cualquier bucle que se realice a través de cualquiera de ellos será detectado y desactivado por el conmutador. Sin embargo cuando creamos VLANs nuevas el conmutador ejecuta una instancia diferente de Spanning Tree para cada VLAN con los puertos que tiene asignados, y detectará de forma completamente independiente la aparición de bucles en cualquiera de sus VLANs. Por ejemplo, si en el conmutador de nuestro ejemplo anterior conectamos el puerto 4 con el 5 antes de crear las VLANs esa conexión se detecta como un bucle con lo que el conmutador desactiva una de las dos puertas y por tanto el enlace. En cambio después de haber creado las VLANs esa conexión ya no es un bulce, sino una conexión entre la VLAN I y la D.

En algunos casos una buena combinación de VLANs Spanning Tree permite conseguir mejoras en el rendimiento. Veámoslo con un ejemplo. Supongamos que tenemos una red formada por dos conmutadores, cada uno con 48 puertos 10BASE-T y 2 puertos 100BASE-F; en los puertos 10BASE-T hay configuradas varias VLANs y los dos conmutadores están unidos entre sí por uno de los puertos 100BASE-F configurado como truk (el otro puerto 100BASE-F está libre). Evidentemente por dicho enlace trunk circula mezclado el tráfico correspondiente a todas las VLANs definidas. Supongamos que hemos medido el tráfico en el enlace trunk y hemos observado que se encuentra saturado, por lo que necesitamos aumentar la capacidad de la conexión entre ambos conmutadores y no podemos comprar nuevo equipamiento. Para ello conectamos los dos equipos a través de la segunda interfaz 100BASE-F, pero como los conmutadores no soportan agregación de enlaces el Spanning Tree desactiva la segunda conexión, con lo que no hemos conseguido ningún aumento de la capacidad. Ahora bien, el Spanning Tree elige como vía preferente de comunicación uno de los dos enlaces 100BASE-F en base a un algoritmo determinista, en el que intervienen entre otros parámetros configurables por el usuario. Por defecto todas las VLANs tienen los mismos valores de estos parámetros y todas eligen el mismo enlace como vía preferente, pero es posible alterar el valor de esos parámetros en algunas VLANs forzando así la elección inversa, de forma que podremos repartir el tráfico entre los dos enlaces y en la práctica aumentar el flujo de tráfico entre ambos conmutadores. Jugando con la asignación preferente para cada VLAN de uno u otro enlace trunk podremos conseguir un reparto lo más equilibrado posible entre ambos.

Una propiedad interesante de las VLANs es la posibilidad de configurar interfaces virtuales en los hosts. Supongamos por ejemplo que en nuestro supuesto de campus con tres VLANs, I, D y A, tenemos un servidor importante y potente que deseamos esté accesible de forma directa en las tres, de forma que cualquier host de cualquiera de las VLANs pueda acceder a él sin necesidad de pasar por un router. Una posibilidad sería dotar a dicho servidor de tres interfaces de red y conectar cada una de ellas a un puerto del conmutador asignado a cada una de las VLANs. Cada interfaz recibiría una dirección de red correspondiente a la VLAN en la que se encontrara. Sin embargo esta solución se hace inmanejable si aumenta el número de VLANs. Otra posibilidad, más interesante, sería configurar una interfaz de red del servidor como tres interfaces virtuales y conectarla a un puerto trunk del conmutador. Para esto necesitaremos disponer de drivers con soporte de IEEE 802.1q para la interfaz de red y el sistema operativo que estemos utilizando. Dada la gama de posibilidades existente a veces hay la tendencia a ubicar los servidores principales de una red en todas las VLANs de ésta, a fin de evitar el paso por el router y conseguir una comunicación más directa con los hosts. Conviene destacar sin embargo que en caso de ubicar un host en varias VLANs (tanto si es con una interfaz física como virtual en cada una de ellas) el host tendrá que procesar los paquetes broadcast que reciba en todas las VLANs en las que esté presente.

1.6 EJERCICIOS

1. Indique si es verdadera o falsa cada una de las siguientes afirmaciones:

a) Cuando una red crece de forma considerable es conveniente utilizar puentes para reducir el tráfico broadcast generado a nivel MAC.

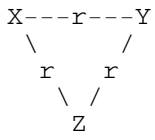
b) El Spanning Tree es un protocolo que permite tener enlaces redundantes en una red, pero no repartir tráfico entre ellos.

c) La comunicación full dúplex en redes locales solo es posible cuando la red esta formada por dos estaciones únicamente.

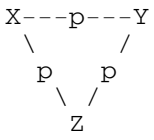
2. Suponga que en un punto de un edificio tiene que unir entre si tres redes Ethernet (segmentos 10Base5 por ejemplo). Para ello puede utilizar bien un puente con tres interfaces, o un repetidor también con tres interfaces. En principio por lo que se refiere a la topología de la red y el número de estaciones puede utilizar uno u otro. Cual sería la diferencia entre emplear el puente o el repetidor? En que caso serían ambos equivalentes, desde el punto de vista del tráfico que circularía por cada segmento?.

3. Siguiendo con el supuesto anterior de los tres segmentos Ethernet (que llamaremos X, Y y Z), suponga ahora que dispone para unirlos de tres puentes y tres repetidores, cada uno con dos interfaces, y puede utilizar cualquier combinación de estos equipos para unir los segmentos entre sí. Diga cuales de las siguientes topologías serían válidas y cuales no, y comente que ocurriria en cada una de ellas. Suponga que los puentes implementan el protocolo spanning-tree.

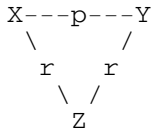
A:



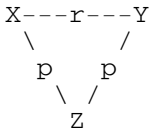
B:



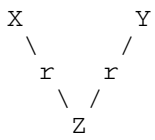
C:



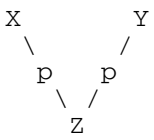
D:



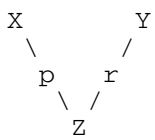
E:



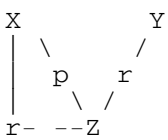
F:



G:



H:



4. Las prestaciones de un puente se suelen medir en pps (paquetes por segundo) filtrados por interfaz y pps transmitidos; el primer parámetro indica el número de paquetes que el puente es capaz de analizar en cada interfaz (entendiendo por análisis el proceso de interpretar la dirección MAC en la trama y buscarla en sus tablas); el segundo indica que cantidad de paquetes puede pasar el puente de una red a otra (evidentemente un número igual o menor que el primero). Calcule el número de pps que debe poder filtrar y transmitir un puente con dos interfaces 10Base2 para asegurar que el puente no supondrá en ningún caso un factor limitante del rendimiento de la red.

5. Los puentes disponen de una zona de memoria limitada, destinada a almacenar las direcciones MAC de los equipos que están presentes en cada una de las LANs a las que están conectados. Una empresa tiene un puente con capacidad máxima de 1024 direcciones que está conectado a dos redes Ethernet, A y B, con 400 y 800 ordenadores respectivamente. Cuando todos los ordenadores están conectados el puente se bloquea, ya que su tabla de direcciones se desborda. Para resolver el problema el administrador de la red ha decidido partir la Ethernet B por la mitad, instalando otro puente que conecte 400 equipos a cada lado; de esta forma cada puente solo está conectado directamente a 800 equipos (400 en cada red). Ha resuelto el administrador el problema?

6. Una empresa posee una LAN Ethernet formada por dos segmentos de cable coaxial, A y B, unidos entre sí mediante un repetidor. La empresa dispone de dos servidores principales, X e Y, que son accedidos por los demás ordenadores de la empresa. El administrador de la red encarga una auditoría de la red a una empresa especializada, la cual emite el siguiente informe:

> *El tráfico se comporta de forma regular con un volumen medio de 4 Mb/s, el 50% del cual es generado por el servidor X, el 40% por el servidor Y y el 10% restante por otros ordenadores. El 80% del tráfico generado por el servidor X va dirigido a ordenadores situados en el segmento A, y el 75% del tráfico generado por el servidor Y está destinado a ordenadores ubicados en el segmento B.*

Los servidores están ubicados físicamente en la misma habitación que el repetidor, y pueden estar conectados indistintamente a uno u otro de ambos segmentos. Inicialmente ambos servidores se encontraban en el segmento A, pero a la vista del informe anterior el administrador decide cambiar el servidor Y al segmento B. Diga si dicha decisión es correcta, incorrecta o irrelevante en cuanto al rendimiento que podrá obtenerse de la red.

Más adelante la empresa decide sustituir el repetidor por un puente transparente de dos interfaces, para aumentar así la capacidad de su red. Cual sería en este caso la forma óptima de conectar los servidores? y la forma pésima, es decir, la menos eficiente?

7. Un puente transparente con dos interfaces tiene un defecto en la memoria RAM que utiliza para almacenar las direcciones MAC, como consecuencia de la cual ocasionalmente las direcciones contenidas en sus tablas son incorrectas (la tasa de error es de uno en un millón). Que consecuencias tiene esto para el funcionamiento de la red?

a) No funciona en absoluto
b) Funciona normalmente
c) Funciona normalmente, salvo una pequeña merma de la eficiencia
d) Funciona normalmente, salvo una pequeña merma de la eficiencia y una remota posibilidad de que alguna estación quede inaccesible durante algunos minutos.

8. Una empresa tiene dos oficinas, en cada una de las cuales hay una red local constituida por una red 802.3 (10BASE2) a la cual están conectados los ordenadores. Las oficinas están conectadas mediante dos líneas E1, y en cada oficina hay dos puentes transparentes, cada uno conectado a una de las líneas y a la LAN. Los cuatro puentes son del mismo modelo y del mismo fabricante e implementan el protocolo spanning tree.

La única aplicación de red que utilizan los empleados de la empresa es un sistema de vídeoconferencia punto a punto (es decir, uno a uno), y solo la utilizan para comunicar con empleados de la otra oficina, nunca dentro de la misma oficina. La aplicación utilizada genera un paquete cada 40 milisegundos y para funcionar correctamente necesita 160 Kb/s (medidos a nivel de red, es decir contando el paquete de nivel 3 incluida la cabecera).

Calcule:

a) El número máximo de vídeoconferencias simultáneas que podrán mantenerse entre las dos oficinas
b) Si en un momento en que se están celebrando ese número de vídeoconferencias se mide con un analizador el tráfico a nivel físico en la red local de una de las oficinas, que valor se obtendría?
c) El número de paquetes por segundo que estaría procesando cada uno de los puentes en ese momento.

Suposiciones:

- No será factor limitante la capacidad de proceso o la memoria de los ordenadores, ni la de los puentes.
- Cada ordenador solo puede mantener como máximo una vídeoconferencia
- Aunque los puentes encapsulan las tramas 802.3 en tramas HDLC, considere despreciable el overhead introducido por las cabeceras HDLC y el debido al relleno de bits.

9. A continuación se adjunta la salida (parcial) obtenida por consola de dos comandos ejecutados en el conmutador que desde el Servicio de Informática conectaba el campus de Burjassot en 1999. En este caso se refleja únicamente la información relativa a los edificios A, B, C y D del campus. La conexión de estos edificios es como sigue:

- Todos los ordenadores del edificio A se conectan al conmutador por 10BASE-FL. Todo el edificio forma una única red Ethernet de 10 Mb/s compartida.
- Los edificios B y D se conectan cada uno mediante dos enlaces 10BASE-FL (aproximadamente la mitad de los ordenadores de cada edificio a través de cada uno de los dos enlaces). Cada una de estas dos mitades forma una red Ethernet (un dominio de colisiones).
- En el edificio C hay un conmutador con una puerta 100BASE-FX y múltiples puertas de 10 Mb/s (10BASE-T); el conmutador se conecta por la puerta 100BASE-FX al conmutador del Servicio de Informática y las puertas 10BASE-T se usan para repartir lo mejor posible los ordenadores situados en el edificio en dominios de colisión diferentes.

Los comandos muestran una serie de contadores de tráfico de cada uno de los puertos; los contadores fueron borrados a las 14:28 aproximadamente y la ejecución que se muestra corresponde a las 15:32, aproximadamente. Haciendo uso de la información disponible calcule:

- El tráfico medio de cada puerto, en tramas/s y bits/s.
- El tráfico broadcast y multicast (en tramas/s y bits/s).
- El tamaño medio de las tramas recibidas por cada puerto, y el tamaño medio de trama para el tráfico en los seis puertos (ojo, este último no necesariamente será la media de los seis anteriores).
- Realice los comentarios y extraiga las conclusiones que considere pertinentes a la vista de los valores mostrados en estos comandos.

```
cataciuv> show port
Port  Name               Status     Vlan       Level  Duplex Speed Type
----- ------------------ ---------- ---------- ------ ------ ----- ------------
 4/1  Edif. A (1/1)      connected  2          normal   half    10 10BaseFL MM
 4/2  Edif. B (1/2)      connected  2          normal   half    10 10BaseFL MM
 4/3  Edif. B (2/2)      connected  2          normal   half    10 10BaseFL MM
 4/5  Edif. D (1/2)      connected  2          normal   half    10 10BaseFL MM
 4/6  Edif. D (2/2)      connected  2          normal   half    10 10BaseFL MM
 7/5  Edificio C         connected  2          normal   half   100 100BaseFX MM

Port  Align-Err  FCS-Err    Xmit-Err   Rcv-Err    UnderSize
----- ---------- ---------- ---------- ---------- ---------
 4/1           0          0          0          0         0
 4/2           4          4          0          0         6
 4/3           0          3          0          0         0
 4/5         907        130          0          0         0
 4/6           5          1          0          0        25
 7/5           0          0          0          0         0
```

```
Port  Single-Col Multi-Coll Late-Coll  Excess-Col Carri-Sen Runts     Giants
----- ---------- ---------- ---------- ---------- --------- --------- ---------
 4/1         495        230          0          0         0         0         0
 4/2        1669        602          0          0         0         4         0
 4/3         172         40          0          0         0         0         0
 4/5       10601       8565          0          0         0        48         0
 4/6       12642       5040          0          6         0         6         0
 7/5          94          0          0          0         0         0         0

Last-Time-Cleared
--------------------------
Mon Mar 29 1999, 13:27:49
cataciuv> show mac
MAC      Rcv-Frms   Xmit-Frms  Rcv-Multi  Xmit-Multi Rcv-Broad  Xmit-Broad
-------- ---------- ---------- ---------- ---------- ---------- ----------
 4/1          38364     228619        628     104850       1318      88342
 4/2          77597     260150       1039     104440       3910      85750
 4/3          11875     222617        137     105342       1390      88272
 4/5         725014     584790       2315     103164       5030      84632
 4/6        1049886     595129       1299     104179       7850      81813
 7/5         148030     325113       9600      96204       2473      87192

Port     Rcv-Unicast          Rcv-Multicast        Rcv-Broadcast
-------- -------------------- -------------------- --------------------
 4/1                   36427                  628                 1320
 4/2                   72772                 1039                 3914
 4/3                   10350                  137                 1391
 4/5                  717826                 2317                 5032
 4/6                 1042406                 1299                 7870
 7/5                  136001                 9610                 2477

Port     Xmit-Unicast         Xmit-Multicast       Xmit-Broadcast
-------- -------------------- -------------------- --------------------
 4/1                   35484               104921                88472
 4/2                   70088               104510                85879
 4/3                   29052               105412                88402
 4/5                  397102               103232                84765
 4/6                  409260               104250                81926
 7/5                  113668                96260                87327

Port     Rcv-Octet            Xmit-Octet
-------- -------------------- --------------------
 4/1                10996295             47524699
 4/2                13737961             55135179
 4/3                 1160993             41191428
 4/5               533770839            131307431
 4/6               506340919            428444894
 7/5                26532573             91511781

Last-Time-Cleared
--------------------------
Mon Mar 29 1999, 14:27:49
cataciuv> show time
Mon Mar 29 1999, 15:32:22 MET
cataciuv> exit
```

Información suplementaria:

Descripción de los campos del comando 'show port' (obtenida directamente del manual del fabricante)

Field	Description
Port	Module and port number.
Name	Name (if configured) of the port.
Status	Status of the port (connected, notconnect, connecting, standby, faulty, inactive, shutdown, disabled, or monitor).

Vlan	VLANs to which the port belongs.
Level	Level setting for the port (normal or high).
Duplex	Duplex setting for the port (auto, full, fdx, half, hdx, a-half, a-hdx, a-full, a-fdx).
Speed	Speed setting for the port (auto, 10, 100,155, a-10, a-100, 4, 16, a-14, a-16).
Type1	Port type, for example, 10BaseT, 10BaseFL MM, 100BaseTX, 100BaseT4, 100BaseFX MM, 100BaseFX SM, 10/100BaseTX, TokenRing, FDDI, CDDI, and RSM.
Align-Err	Number of frames with alignment errors (frames that do not end with an even number of octets and have a bad CRC) received on the port.
FCS-Err	Number of frame check sequence errors that occurred on the port.
Xmit-Err	Number of transmit errors that occurred on the port (indicating that the internal transmit buffer is full).
Rcv-Err	Number of receive errors that occurred on the port (indicating that the internal receive buffer is full).
UnderSize	Number of received frames less than 64 octets long (but are otherwise well-formed).
Single-Coll	Number of times one collision occurred before the port successfully transmitted a frame to the media.
Multi-Coll	Number of times multiple collisions occurred before the port successfully transmitted a frame to the media.
Late-Coll	Number of late collisions (collisions outside the collision domain).
Excess-Col	Number of excessive collisions that occurred on the port (indicating that a frame encountered 16 collisions and was discarded).
Carri-Sen	Number of times the port sensed a carrier (to determine whether the cable is currently being used).
Runts	Number of received runt frames (frames that are smaller than the minimum IEEE 802.3 frame size) on the port.
Giants	Number of received giant frames (frames that exceed the maximum IEEE 802.3 frame size) on the port.
Last-Time-Cleared	Last time the port counters were cleared.

Descripción de los campos del comando 'show mac' (obtenida directamente del manual del fabricante)

Field	**Description**
MAC	Module and port.
Rcv-Frms	Frames received on the port.
Xmit-Frms	Frames transmitted on the port.
Rcv-Multi	Multicast frames received on the port.
Xmit-Multi	Multicast frames transmitted on the port.
Rcv-Broad	Broadcast frames received on the port.
Xmit-Broad	Broadcast frames transmitted on the port.
Rcv-Unicast	Number of unicast frames received on the port.
Rcv-Multicast	Number of multicast frames received on the port.
Rcv-Broadcast	Number of broadcast frames received on the port.
Xmit-Unicast	Number of unicast frames transmitted on the port.
Xmit-Multicast	Number of multicast frames transmitted on the port.
Xmit-Broadcast	Number of broadcast frames transmitted on the port.
Rcv-Octet	Number of octet frames received on the port.
Xmit-Octet	Number of octet frames transmitted on the port.
Last-Time-Cleared	Date and time of the last clear counters command.

10. Suponga que tiene una red local con dos VLANs que denominaremos ‘Roja’ y ‘Azul’. Un experto ha monitorizado la red durante períodos de intensa actividad y ha llegado a la conclusión de que una de las dos

debe partirse porque tiene un tráfico broadcast excesivo, pero ha perdido la nota que indicaba a que red se refería. Como aun dispone del analizador usted decide repetir las medidas para averiguar a cual de ellas se refería el experto, y obtiene los siguientes resultados:

	Tráfico broadcast	
VLAN	**Paquetes/s**	**Kbits/s**
Roja	100	60
Azul	50	100

A cual de las dos redes se refería el experto?

1.7 SOLUCIONES

S1.-

a) **Falsa**. El uso de puentes no aísla el tráfico broadcast.

b) **Verdadera**.Entre dos redes solo un enlace puede estar activo en un momento dado, pues de lo contrario se producirían bucles.

c) **Verdadera**.La comunicación full dúplex inhibe el protocolo MAC y utiliza el medio de transmisión como si se tratara de un enlace punto a punto, por lo que solo puede funcionar entre dos estaciones.

S2.-

Con el puente el tráfico local de cada segmento no pasa a los otros dos, mientras que con el repetidor si. En un caso ideal en que todo el tráfico de cada segmento fuera local el rendimiento de la red en su conjunto podría llegar a ser de 30 Mb/s utilizando un puente, mientras que con el repetidor estaría limitado a 10 Mb/s.

Las dos soluciones serían equivalentes en el caso de que todo el tráfico fuera broadcast o multicast.

S3.-

Son válidas todas excepto la A (suponiendo que todos los puentes soportan el protocolo spanning tree):

A. Al existir un bucle entre repetidores cualquier señal eléctrica emitida será repetida indefinidamente, lo cual bloquearía toda la red con una colisión permanente en cuanto se intentara transmitir la primera trama.

B. En este caso los puentes detectarían el bucle gracias al spanning tree, con lo que uno de ellos se desactivaría automáticamente, quedando por tanto una topología en V similar a la mostrada en F. Incluye un cierto grado de redundancia que permitiría seguir funcionando en caos de avería de uno de los puentes.

C. El puente detectaría el bucle por spanning tree y se desactivaría, quedando una topología igual que la mostrada en E. Posee redundancia ya que en caso de avería de uno de los repetidores la comunicación se reanudaría a través del puente.

D. Uno de los dos puentes se desactivaría, quedando una topología similar a la mostrada en G. La red podría funcionar en caso de avería de uno cualquiera de los tres elementos.

E. Caso normal de una red formada por tres segmentos unidos entre sí por repetidores. No hay redundancia.

F. Caso normal de tres redes unidas entre sí por puentes. No hay redundancia.

G. Dos redes (ZY y X) unidas entre sí por un puente. Sin redundancia.

H. Red formada por tres segmentos unidos mediante repetidores. El puente detecta el bucle y se desactiva, quedando una topología equivalente a E. En caso de avería del repetidor X-Z el puente entraría en funcionamiento, dando una topología equivalente a G.

S4.-

Para calcular la cantidad máxima de paquetes que puede circular por una red 802.3 utilizaremos el tamaño mínimo de paquete, o sea 64 bytes. En la práctica dichas tramas van acompañadas de 8 bytes adicionales (por el preámbulo y el delimitador de inicio) y de 12 bytes 'virtuales' correspondientes al interframe gap, dando un total de 84 bytes que tardan en enviarse 67,2 μs. Por tanto el número máximo de paquetes que pueden viajar por una red 802.3 es de $1/(67{,}2 * 10^{-6})$, o sea 14881 paquetes por segundo.

Para que un puente con dos interfaces no suponga factor limitante debe poder filtrar en total **29762 pps**, y **transmitir 14881 pps** entre ambas interfaces. Obsérvese que en el primer caso suponemos que ningún paquete atraviesa el puente, mientras que en el segundo caso suponemos que todo el tráfico pasa de un lado a otro; por tanto en el primer caso cada red tiene 10 Mb/s de tráfico propio, mientras que en el segundo el tráfico propio de una y otra red suma 10 Mb/s (podrían ser por ejemplo 7 Mb/s en una red y 3 en la otra).

S5.-

Con la nueva topología se dispone de tres redes en vez de dos; supongamos que las llamamos A, B y C. Las redes A y B estan unidas por el puente 'viejo', causante de los problemas, que denominamos P1, mientras que B y C están unidas por el puente nuevo, que llamamos P2.

Dividiendo por la mitad la red B no se ha resuelto el problema, ya que aun cuando ahora no hay mas de 800 ordenadores directamente conectados a cada puente éstos siguen teniendo que almacenar en sus tablas las direcciones de todas las estaciones, puesto que cuando un puente propaga una trama de una red a otra lo hace repitiéndola con la misma dirección origen.

Aun en el caso de que no hubiera ningún intercambio de tramas entre ordenadores de la red A y la red C, en cuanto un ordenador de C enviara su primera trama ésta sería propagada por inundación a la red B, momento en el que el puente que conecta con A tendría que anotar en sus tablas la correspondiente dirección origen.

En ambos casos la red solo funcionaría correctamente en el caso de que hubiera menos de 1024 ordenadores enviando tramas.

S6.-

Dado que los servidores X e Y son los que originan la mayor parte del tráfico en la red (90%) es previsible que la mayoría de las colisiones en la red se produzcan por tráfico entre estos dos equipos. Para reducir el riesgo de colisión es conveniente reducir el tiempo de propagación, por lo que la decisión de poner cada servidor en un lado diferente del repetidor es incorrecta, pues aumenta el tiempo de propagación entre X e Y y con ello el riesgo de colisión.

En el caso de poner un puente se separa el tráfico, con lo que la forma óptima de conectar los servidores sí que es la que había adoptado el administrador de la red, es decir el servidor X en el segmento A y el Y en el B; de esta forma el tráfico de X, que va dirigido principalmente a ordenadores ubicados en la red A, no carga a la red Y, y otro tanto ocurre con el tráfico del servidor Y y la red B. A partir de los datos del enunciado podemos calcular como se distribuye el tráfico en este caso:

Tráfico generado por X: 4 * 0,5 = 2 Mb/s

Tráfico generado por Y: 4 * 0,4 = 1,6 Mb/s

Tráfico de X dirigido a A: 2 * 0,8 = 1,6 Mb/s

Tráfico de X dirigido a B: 2-1,6 = 0,4 Mb/s

Tráfico de Y dirigido a B: 1,6 * 0,75 = 1,2 Mb/s

Tráfico de Y dirigido a A: 1,6 - 1,2 = 0,4 Mb/s

Tráfico total en la red A: 2 + 0,4 = 2,4 Mb/s

Tráfico total en la red B: 1,6 + 0,4 = 2 Mb/s

Tráfico total en el puente: 0,8 Mb/s (0,4 en cada sentido)

La forma pésima o menos eficiente sería justo al revés, poner el servidor X en el segmento B y el Y en el A, ya que así se obliga a que una mayoría del tráfico de A pase por B y viceversa. Cuantitativamente sería:

Tráfico en la red A: 1,6 + 1,6 = 3,2 Mb/s

Tráfico en la red B: 2 + 1,2 = 3,2 Mb/s

Tráfico total en el puente: 2,8 Mb/s (1,2 Mb/s de A a B y 1,6 Mb/s de B a A).

S7.-

Los puentes capturan todas las tramas que llegan a sus interfaces y analizan las direcciones de origen, a partir de las cuales construyen sus tablas de direcciones. Dichas tablas les permiten optimizar el enrutamiento de las tramas, enviándolas únicamente a la interfaz donde se encuentra la estación de destino si ésta está identificada; en caso contrario aplican la técnica de inundación, esto es envían la trama por todas sus interfaces excepto por la que ha llegado.

Las tablas de los puentes almacenan parejas dirección MAC-número de interfaz;la dirección MAC tiene 6 bytes, mientras que el número de interfaz dependerá del tipo de puente y del número de éstas, pero generalmente será bastante menor (un byte sería suficiente para 256 interfaces).

Si como consecuencia de un error en la memoria RAM se altera una entrada en la tabla de direcciones pueden ocurrir dos cosas: que afecte a una dirección MAC o a un número de interfaz. En el primer caso (mas probable ya que las direcciones MAC ocupan mas memoria) una dirección desaparece y otra aparece en la tabla; la dirección que desaparece de las tablas seguiría funcionando normalmente, aunque con una pequeña merma en la eficiencia, ya que la siguiente trama después del error llegaría a su destino al aplicar el puente la técnica de inundación; la dirección que aparece como consecuencia del error no corresponderá normalmente a ninguna estación de la red, por lo que pasará desapercibida hasta que caduque por antigüedad. En el remoto caso de que la nueva dirección errónea coincidiera con alguna estación de la red, la estación real quedaría inaccesible hasta que la nueva entrada caducara por antigüedad.

Si el error en la RAM afecta a una entrada en la tabla de interfaces la estación correspondiente quedaría inaccesible hasta que la nueva entrada caducara por antigüedad. Sin embargo como estas entradas tienen un tamaño mucho menor la probabilidad de que esto ocurra es menor.

En resumen, el efecto percibido sería el d):

Funciona normalmente, salvo una pequeña merma de la eficiencia y una remota posibilidad de que alguna estación quede inaccesible durante algunos minutos.

S8.-

a) Aun cuando hay dos líneas E1 solo se podrá utilizar una, ya que el protocolo spanning tree se ocupará de desactivar la otra para evitar que se produzcan bucles entre ambas redes.

Calculamos en primer lugar el tamaño de cada trama transmitida:

160000 * 0,04 = 6400 bits -> 800 bytes

Estos 800 bytes son el paquete a nivel de red. Dicho paquete se incluirá normalmente en una trama LLC (802.2). Sabemos que la cabecera LLC son normalmente 8 bytes (ver apuntes) lo cual nos da un tamaño de trama LLC de 808 bytes.

Para su envío por la red local esta trama LLC se incluirá como parte de datos de una trama 802.3. Esto significa que la trama 802.3 tendrá una longitud de **826 bytes**, los 808 de la trama LLC datos mas los campos de dirección origen (6 bytes), dirección destino (6), longitud (2) y checksum (4). Dicha longitud está dentro de los márgenes que establece la norma 802.3 (entre 64 y 1518) por lo que no necesitaremos utilizar relleno ni fragmentar la trama generada por la aplicación. Obsérvese que el preámbulo (7 bytes), el delimitador de inicio de trama (1 byte) y el interframe gap (equivalente a 12 bytes) , al no formar parte del nivel MAC de la trama 802.3 no son transmitidos por los puentes.

Así pues transmitimos 826 bytes, equivalentes a 6608 bits, cada 40 milisegundos. Esto supone un caudal medio de 165,2 Kb/s (6608/0,04). Cada videoconferencia genera 165,2 Kb/s en cada sentido, pero como la línea E1 tiene una capacidad de 2048 Kb/s en cada sentido (pues es full duplex) bastará con calcular un solo sentido para saber el número máximo de vídeoconferencias que pueden celebrarse simultáneamente:

2048/165,2 = **12,4 -> 12** videoconferencias simultáneas

Esto en realidad supone 24 ordenadores participando, 12 en cada lado.

b) Para calcular el tráfico en la red local sí que deberemos tomar en cuenta el preámbulo y el delimitador de inicio, pero no el interframe gap ya que éste no representa la transmisión de información alguna. Tendremos pues que considerar la transmisión de una trama de 834 bytes, o sea 6672 bits, cada 40 milisegundos, lo que nos da un caudal de 166,8 Kb/s por vídeoconferencia (6672/0,04). Pero esta vez si que debemos considerar ambos sentidos, ya que el medio de transmisión de la LAN no es full duplex. Será por tanto:

166,8 * 12 * 2 = **4003,2 Kb/s**

Es decir, tenemos una ocupación de la red del **40,03 %**

c) Cada vídeoconferencia genera 25 paquetes por segundo (pps) en cada sentido. Los puentes activos están en todo momento recibiendo 25*12 = 300 pps de su interfaz LAN y enviándolos por la línea E1, y recibiendo 300 pps por la interfaz WAN y enviándolos por la LAN. Así pues **cada uno procesa 600 paquetes por segundo**.

En cuanto a los puentes que están inactivos por el Spanning Tree, no reenvían paquete alguno y por tanto cabría pensar que no procesan ningún paquete; pero estrictamente hablando están recibiendo los 600 pps de la LAN; dichos paquetes han de procesarlos y descartarlos, ya que esta es la única manera como pueden mantenerse alerta para el caso en que reciban algún paquete del protocolo propio de los puentes que les indicará que deben entrar en funcionamiento (por ejemplo por avería de la otra línea E1). Ciertamente el protocolo de control de los puentes introduciría un tráfico residual que no hemos considerado, pero que podemos considerar despreciable.

S9.-

El período de tiempo al que corresponde la muestra es de las 14:27:49 a las 15:32:22, o sea 3873 segundos.

Tráfico en bits/s

Para calcular el tráfico por puerto en bits/s sumaremos los valores de las columnas Rcv-Octet y Xmit-Octet. Por ejemplo para el puerto 4/1 sería:

10996295 + 47524699 = 58520994 bytes = 468167950 bits

Esto nos da un tráfico medio de:

468167950 / 3873 = 120880 bits/s = 120,9 Kb/s

De la misma forma calculamos el tráfico medio para los demás puertos:

Puerto	Kbits Tx + Rx	Tráfico medio Tx + Rx (Kb/s)
4/1	468168	120,9
4/2	550985	142,3
4/3	338819	87,5
4/5	5320626	1373,8
4/6	7478286	1930,9
7/5	944355	243,8

Podemos ver que los puertos 4/5 y 4/6 (correspondientes al bloque D) son con diferencia los que soportan mas tráfico; en el caso del puerto 4/6 la ocupación se enuentra muy próxima al 20%, valor considerado suficiente para plantear una mejora de la red. Es muy probable que en caso de repetir la medida en otra hora mas punta del día (por ejemplo de 12 a 13 horas) el resultado hubiera sido apreciablemente mayor.

Tráfico en tramas/s

Aquí procedemos como antes, sumando esta vez las columnas Rcv-Frms y Xmit-Frms:

Puerto	Tramas Tx + Rx	Tráfico medio (tramas/s)
4/1	266983	68,9
4/2	337747	87,2
4/3	234492	60,5
4/5	1309804	338,2
4/6	1645015	424,7
7/5	473143	122,2

Tráfico broadcast/multicast

Para el tráfico broadcast/multicast calcularemos el número de tramas por segundo a partir de los datos Rcv-Multi, Xmit-multi, Rcv-Broad y Xmit-Broad; alternatiamente podríamos haber utilizado los datos Rcv-Multicast, Rcv-Broadcast, Xmit-Multicast y Xmit-Broadcast:

Puerto	Tramas multicast Tx + Rx	Tráfico medio multicast (tramas/s)
4/1	105478	27,2
4/2	105479	27,2
4/3	105479	27,2
4/5	105479	27,2
4/6	105478	27,2
7/5	105804	27,3

Puerto	Tramas broadcast Tx + Rx	Tráfico medio broadcast (tramas/s)
4/1	89660	23,2
4/2	89660	23,2
4/3	89662	23,2
4/5	89662	23,2
4/6	89663	23,2
7/5	89665	23,2

Puede apreciarse que el tráfico broadcast y multicast es prácticamente el mismo en todos los puertos, salvo por una pequeña diferencia en el caso del tráfico multicast del puerto 7/5 que seguramente se debe a la diferencia de tiempo que se produce en la obtención de los contadores del sistema. También se observa alguna pequeña diferencia, seguramente por el mismo motivo, entre los valores de las columnas Rcv-Multi, Xmit-multi, Rcv-Broad y Xmit-Broad y los de las columnas Rcv-Multicast, Rcv-Broadcast, Xmit-Multicast y Xmit-Broadcast.

Sabiendo que el tráfico broadcast/multicast asciende a 23,2 + 27,2 = 50,4 tramas/s podemos calcular fácilmente el número de tramas unicast por segundo en cada puerto:

Puerto	Tramas unicast Tx + Rx	Tráfico medio unicast (tramas/s)
4/1	71845	18,5
4/2	142608	36,8
4/3	39351	10,1
4/5	1114663	287,8
4/6	1449874	374,3
7/5	277674	71,8

Tamaño medio de tramas

A partir del número de bits calculado en 1 y del número de tramas calculado en 2 obtenemos el tamaño medio de trama para cada puerto:

Puerto	Tamaño medio de trama (en Bytes)
4/1	219,2
4/2	203,9
4/3	180,6
4/5	507,8
4/6	568,3
7/5	249,5

El tamaño medio de trama para los seis puertos lo obtendremos dividiendo el número total de bits por el número total de tramas:

15101239000/(8*4267184) = **442,4 Bytes**

Podemos observar una relación entre el tamaño medio de trama en cada puerto y la cantidad de tramas unicast en dicho puerto. A medida que crece la proporción relativa de estas tramas crece el tamaño medio. Esto nos hace pensar que las tramas broadcast/multicast tienen un tamaño de trama pequeño en comparación con el de las tramas unicast.

Nos falta calcular la cantidad de tráfico en bits/s que se produce como consecuencia de las emisiones multi/broadcast. Este no lo podemos calcular de manera directa ya que no disponemos de la cantidad de octetos multi/broadcast transmitidos, solo del número de tramas.

Dado que las tramas multi/broadcast son las mismas en todos los puertos su tamaño medio será también el mismo en todos. Sin embargo el tráfico unicast es distinto en cada puerto, por lo que su tamaño medio será también diferente. Supongamos que denominamos T_m el tamaño de trama multi/broadcast en bits, y $T_{u4/1}$, $T_{u4/2}$, etc. el tamaño de trama unicast de cada puerto (en bits). Se cumple entonces el siguiente sistema de ecuaciones:

$$120900 = 18{,}5\ T_{u4/1} + 50{,}4\ T_m$$

$$142300 = 36{,}8\ T_{u4/2} + 50{,}4\ T_m$$

$$87500 = 10{,}1\ T_{u4/3} + 50{,}4\ T_m$$

$$1373800 = 287{,}8\ T_{u4/5} + 50{,}4\ T_m$$

$$1930900 = 374{,}3\ T_{u4/6} + 50{,}4\ T_m$$

$$243800 = 71{,}8\ T_{u7/5} + 50{,}4\ T_m$$

Como tenemos un sistema de 6 ecuaciones con 7 incógnitas no es posible resolverlo sin hacer alguna simplificación. Podemos por ejemplo suponer que el tamaño de las tramas unicast será el mismo en dos puertos. Con esto obtendremos un valor aproximado del tamaño de las tramas multicast que nos permitirá conocer el tamaño de las tramas unicast en el resto de los puertos. Esta aproximación será tanto más válida cuanto mayor sea el tráfico multicast, ya que menor será el error introducido por la aproximación anterior. Tomaremos por tanto para nuestra aproximación los datos correspondientes a los puertos 4/1 y 4/3:

$$120900 = 18{,}5\ T_u + 50{,}4\ T_m$$

$$87500 = 10{,}1\ T_u + 50{,}4\ T_m$$

De aquí obtenemos:

$T_u = 3976$ bits = 497 bytes

$T_m = 939$ bits = 117 bytes

De lo cual obtenemos que el tráfico multi/broadcast de cada puerto es de:

50,4 * 939 = 47325 bits/s = **47,3 Kb/s**

Esto representa el 54% del tráfico total en el puerto 4/3 y el 39% en el 4/1

Una vez sabido el tamaño de trama multi/broadcast podemos calcular el tamaño medio de trama unicast en cada puerto:

Puerto	Tamaño medio trama unicast (en Bytes)
4/1	497
4/2	323
4/3	497
4/5	576
4/6	629
7/5	342

Colisiones

Vamos a intentar calcular la tasa de colisiones en cada puerto. Dado que cuando ocurren múltiples colisiones no se nos da el número de reintentos vamos a adoptar la simplificación mas optimista de suponer que las múltiples colisiones se resuelven siempre en el segundo intento, por lo que las contaremos como dos colisiones sencillas:

Puerto	Colisiones	Tramas transmitidas	Tasa de colisiones (%)
4/1	955	228619	0,4
4/2	2873	260150	1,1
4/3	252	222617	0,1
4/5	27731	584790	4,5
4/6	22722	595129	3,7
7/5	94	325113	0,03

Como era de esperar los puertos con más tráfico (4/5 y 4/6) son los que tienen una tasa de colisiones mayor. El puerto 7/5, al ser una conexión directa entre dos conmutadores, presenta una tasa de colisiones mucho menor que el resto de puertos; de hecho este puerto podría estar funcionando en modo full-dúplex, con lo que las colisiones se habrían suprimido totalmente.

También podemos comparar la relación entre colisiones sencillas y colisiones múltiples:

Puerto	Relación colisiones múltiples/sencillas
4/1	0,46
4/2	0,36
4/3	0,23
4/5	0,81
4/6	0,40
7/5	0,00

Esta relación, que es del mismo orden en todos los puertos salvo el 7/5, nos indica que el nivel de saturación de la red cuando ha estado en uso ha sido relativamente parecido en todos los casos, es decir, que el bajo tráfico registrado en los puertos 4/1, 4/2 y 4/3 se debe a que han sido usados durante poco tiempo dentro del intervalo estudiado, pero durante ese tiempo han tenido un tráfico de una intensidad comparable al del puerto 4/6; destaca el puerto 4/5 con un valor próximo a la unidad, lo cual hace pensar que ha tenido un uso mas intenso que los demás puertos, aunque haya registrado valores de tráfico medio inferiores al 4/6.

Se observan 6 'Excess-Col' en el puerto 4/6,.lo cual hace pensar que se haya producido algún momento de mucha saturación en la red, o en el mal funcionamiento de algún equipo.

Fiabilidad

En cuanto a la tasa de errores podemos para calcularla agrupar los que aparecen como 'Align-Err' y los 'FCS-Err'. Si suponemos que cada trama errónea contiene un solo bit erróneo podemos calcular fácilmente la tasa de error o BER. Debemos tomar en cuenta al hacer este cálculo que la detección de errores solo se realiza sobre las tramas recibidas, no sobre las transmitidas:

Puerto	Errores	Mbits recibidos	Tasa de error (BER)
4/1	0	88,0	$< 10^{-8}$
4/2	8	110	$7 * 10^{-8}$
4/3	3	9,3	$3 * 10^{-7}$
4/5	1037	4270	$2 * 10^{-7}$
4/6	6	4051	$1,5 * 10^{-9}$
7/5	0	21,2	$< 5 * 10^{-9}$

Se observa que la tasa de error es anormalmente elevada en los puertos 4/3 y 4/5, por lo que estas redes deberían revisarse.

S10.-

El mayor problema del tráfico broadcast no es el caudal que ocupa éste en la red local, sino la cantidad de ciclos de CPU que se pierden en todas las máquinas de la VLAN al tener que procesar estas tramas en el nivel de red del host. Este efecto es directamente proporcional al número de tramas broadcast por segundo, y prácticamente independiente del tamaño de éstas, por lo que la VLAN que habría que dividir primero es la roja, aun cuando la VLAN azul tiene un mayor caudal de tráfico broadcast.

En la práctica sería difícil que se diera el supuesto de este ejercicio, ya que generalmente el tamaño medio de las tramas broadcast en diferentes redes es similar, por lo que suele ser equivalente medirlo en paquetes por segundo o en bits por segundo.

2 EL NIVEL DE RED: GENERALIDADES

Autor: Rogelio Montañana

2.1 INTRODUCCIÓN

El principal objetivo de la capa de red es encaminar los paquetes del origen al destino. Esta es la única capa que 've' y conoce la topología de la red, que está formada por dos tipos de **nodos:**

- **Nodos terminales**: generan o reciben paquetes de otros nodos, nunca encaminan paquetes dirigidos a terceros.

- **Nodos intermedios o de encaminamiento**: se utilizan para encaminar paquetes entre los nodos terminales. Suelen ser ordenadores especializados dedicados y diseñados específicamente para esa función, con sistemas operativos en tiempo real, aunque en ocasiones también se utilizan para desempeñar esta función ordenadores normales.

La terminología de estos dos tipos de nodos es muy diversa y varía según el tipo de red y la 'cultura' de que se trate. La tabla 2.1 refleja las denominaciones más habituales, aunque la terminología no se sigue a rajatabla.

Tipo de nodo	Internet (IP)	X.25	ATM	ISO
Nodo terminal	Host	DTE (Data Terminating Equipment)	Host	End System (ES)
Nodo intermedio o de encaminamiento	Router o Gateway (obsoleta)	DCE (Data Communications Equipment)	Conmutadores o Switches	Intermediate System (IS)

Tabla 2.1.- Denominación de los dos tipos de nodos posibles en diferentes redes

A veces un mismo equipo desempeña ambas funciones a la vez, en diferentes redes. Por ejemplo si se utiliza una red ATM para interconectar routers IP éstos actuarán como nodos terminales (hosts) en la red ATM mientras que para la red IP serán routers.

Por otro lado, en ocasiones los nodos intermedios son origen o destino de los paquetes; por ejemplo un router en una red IP o un conmutador en una red ATM que participan en un protocolo de routing tienen que intercambiar información con sus homólogos; lo mismo ocurre cuando han de emitir o recibir mensajes de control, tales como avisos de congestión, reportar situaciones anómalas o errores que se producen por diversos motivos; en estos casos esos routers o conmutadores actúan como nodos terminales ya que son la fuente o destino de la información.

La denominación conmutador resulta particularmente confusa, pues se utiliza comúnmente para referirse al menos a tres tipos diferentes de dispositivos de comunicaciones, a saber:

- Conmutadores LAN: los dispositivos ya descritos que conmutan tramas en base a la información contenida en la cabecera MAC. Actúan por tanto a nivel de enlace.

- Conmutadores ATM: son los dispositivos que conmutan celdas ATM, que actúan a nivel de red.

- Conmutadores de nivel 3: como su nombre indica estos dispositivos actúan a nivel de red, pero a diferencia de los conmutadores ATM lo hacen con protocolos no orientados a conexión, como IP e IPX. Se trata por tanto de routers, siendo su única diferencia respecto de estos la implementación en hardware de los algoritmos de enrutamiento más habituales, con lo que resultan especialmente rápidos[1]. Esta denominación suele emplearse sobre todo cuando la función de router está integrada en un dispositivo que también actúa como puente LAN. En este

[1] Llegados a este punto es justo indicar que en la actualidad muchos routers altos de gama también implementan en hardware los algoritmos de enrutamiento para los casos más normales.

contexto también se hace alusión a veces a los conmutadores de nivel 2, para referirse a los conmutadores LAN.

Dado que su función es interconectar redes los nodos intermedios tienen normalmente varias interfaces físicas, y los nodos terminales normalmente una. Aunque poco frecuente (y poco útil) un nodo intermedio puede tener una sola interfaz; más frecuente es el caso en que un nodo terminal tiene varias interfaces por razones de rendimiento o fiabilidad. Cuando un nodo terminal tiene varias interfaces decimos que está 'multihomed'.

En cada interfaz física de un nodo funciona una instancia independiente del nivel físico y del nivel de enlace; por el contrario el nivel de red es normalmente global para todo el nodo. En IP cada interfaz física tiene una dirección de red diferente, al menos (puede tener varias).

Los nodos intermedios junto con las líneas que los unen constituyen la subred (recordemos que la subred es la frontera entre el usuario y el proveedor del servicio).

En una LAN broadcast (por ejemplo Ethernet) todos los nodos terminales comunican directamente entre sí, sin necesidad de nodos intermedios, por lo que la capa de red es prácticamente inexistente (esto incluye el caso en que la LAN incluya puentes o conmutadores de cualquier tipo). A cambio el nivel de enlace tiene una complejidad mayor (subcapa MAC y puentes).

Los puentes MAC, en especial los puentes por encaminamiento desde el origen, desempeñan una función hasta cierto punto equivalente a la de un router.

Los servicios que ofrece el nivel de red deberán en lo posible aislar al nivel de transporte de detalles tales como tipo de tecnología física utilizada (LAN, WAN, broadcast, etc.), número y topología de las subredes, etc. Las direcciones de red deberán tener un formato homogéneo, cualquiera que sea el medio físico o subred utilizados.

Los servicios de red pueden ser orientados a conexión (CONS) o no orientados a conexión (CLNS). Ejemplos de servicios CLNS son el protocolo IP, el ISO CLNS elaborado a imagen y semejanza de IP, y el IPX desarrollado por Novell para su sistema operativo en red Netware. Ejemplos de servicios CONS son X.25, Frame Relay y ATM. En los capítulos siguientes hablaremos en detalle de IP, Frame Relay y ATM.

La características principales de un servicio CLNS son:

- Se implementan las primitivas SEND PACKET y RECEIVE PACKET, y poco más.
- No se garantiza el orden de llegada de los **datagramas**.
- Cada datagrama ha de llevar la dirección de destino. Cada uno ha de averiguar la ruta por sí mismo.
- Cada **router** ha de mantener una tabla que indique por que interfaz debe encaminar los paquetes en función de su destino, pero no conserva información de las conexiones (pues no las hay); se dice que el router no tiene estados o que es 'stateless'.
- Si un router de la red queda fuera de servicio la única consecuencia será la pérdida de los datagramas que estuvieran en proceso allí en ese momento, no se pierde ninguna información sobre la conexión, ya que no había conexión; el resto de routers de la red intentará buscar una ruta alternativa para comunicarse.
- Es difícil llevar a cabo un control de congestión, u ofrecer una QoS (Quality of Service, calidad de servicio).

Las principales características de un servicio CONS son:

- Las dos entidades finales (hosts) establecen primero la conexión de forma explícita (un **circuito virtual** o VC) y le asignan un identificador. Esta conexión se utiliza para enviar todos los datos y luego se libera, también de forma explícita.

- El orden de entrega de los paquetes está garantizado.

- Los paquetes no necesitan llevar la dirección de destino (pero sí el número del VC por el que van a viajar). La ruta está marcada por el VC mientras éste está establecido. Cada nodo intermedio (conmutador) ha de tomar nota de los VCs existentes en cada momento (decimos que tiene estados o que es 'statefull'). Cada **conmutador** ha de mantener una tabla que le indique la interfaz de entrada y de salida para cada VC que pasa por él.

- Si un conmutador queda fuera de servicio desaparecerán todos los VCs que pasen por él en ese momento, y los nodos afectados dejarán de comunicar (aunque podrán establecer un nuevo VC si hay un camino alternativo); la información de los VCs no estará presente cuando el conmutador vuelva a entrar en servicio (salvo que se trate de PVCs).

- Es fácil efectuar un control de congestión, y también asegurar una QoS, ya que se tiene una información exacta de que conexiones discurren por cada línea física en cada momento.

En una red CLNS el nivel de transporte es normalmente más complejo pues ha de desempeñar mas funciones que en una red CONS.

Una red CONS puede ser fiable (X.25) o no fiable (Frame Relay o ATM), mientras que una red CLNS normalmente es no fiable[2]. Generalmente cuando se quiere un servicio fiable en una red CLNS se implementa un servicio CONS a nivel de transporte; este es el caso por ejemplo cuando se utiliza el transporte TCP (CONS) sobre una red IP (CLNS).

En este capítulo veremos en primer lugar las funciones principales del nivel de red y la forma como normalmente se realizan. En el siguiente veremos en detalle el funcionamiento del nivel de red de IP, tanto en lo que respecta al protocolo IP mismo como a los protocolos de control y de routing. Continuaremos con el nivel de red en Frame Relay y ATM que tienen algunos puntos en común.

2.2 ALGORITMOS DE ENCAMINAMIENTO

La función fundamental de la capa de red es el enrutamiento o encaminamiento, es decir averiguar por que interfaz se han de mandar los paquetes recibidos para que lleguen a su destino. Con redes basadas en datagramas esta decisión se toma para cada paquete y el nodo que la realiza se denomina router o encaminador. Con redes orientadas a conexión (basadas en circuitos virtuales) la decisión se toma en el momento de establecer el circuito virtual, y a partir de entonces solo se ‘conmutan’ paquetes, de ahí la denominación de conmutador. Se supone que la conmutación es una tarea sencilla y que puede hacerse con extrema rapidez, pues no requiere tomar complejas decisiones.

Para poder encaminar los paquetes es preciso primero conocer cual es la ruta a seguir hacia el destino especificado. El mecanismo que nos permite elegir la ruta a utilizar para posible destino es lo que denominamos un algoritmo de encaminamiento o de routing. Además de otras importantes virtudes un algoritmo de routing debe ser óptimo y justo. Estos conceptos a veces se contraponen, ya que el algoritmo que permite un aprovechamiento óptimo de los recursos no siempre es el que ofrece el reparto más equitativo.

Podemos dividir los algoritmos de routing en dos grandes grupos, estáticos y dinámicos. Los algoritmos de **routing estático** toman las decisiones utilizando información previamente recopilada sobre el estado de la red; en cambio los algoritmos de **routing dinámico** utilizan información recopilada en tiempo real sobre el estado de la red; dicha información se actualiza constantemente mediante paquetes de control que intercambian los routers a través de la misma red.

[2] En este contexto entendemos por fiable que garantiza la entrega.

En el encaminamiento o routing estático las rutas se fijan en función de la capacidad de la línea, el tráfico medio u otros criterios similares. Las tablas de rutas se cargan de forma estática en cada router, por lo que no se necesita intercambiar información y por tanto no se requiere un protocolo de routing. Con el encaminamiento estático no es posible responder a situaciones cambiantes (p. ej. saturación, exceso de tráfico o fallo de una línea). En el encaminamiento estático los cálculos de las rutas óptimas se llevan a cabo en diferido, por lo que es posible aplicar algoritmos tan sofisticados como se quiera aun cuando requieran gran cantidad de recursos de cálculo o de memoria.

En cambio en el encaminamiento o routing dinámico las rutas óptimas se recalculan continuamente en función de la información que los routers reciben en tiempo real sobre el estado de la red. Es preciso utilizar un **protocolo de routing** que permita a los routers intercambiar continuamente esa información, pero a cambio se consigue un mecanismo **autoadaptativo** que puede responder a situaciones cambiantes intentando resolver los problemas que se produzcan. Los algoritmos no pueden ser demasiado complejos, pues han de implementarse en los routers y ejecutarse en tiempo real con los recursos de CPU y memoria de que el router dispone.

Haciendo un símil con un viaje en coche podríamos decir que el encaminamiento estático equivale a planificar la ruta para un viaje en coche antes de salir de casa utilizando los mapas y nuestro conocimiento a priori sobre el estado de las carreteras y la densidad de tráfico habitual en éstas; en cambio con encaminamiento dinámico iríamos fijando la ruta sobre la marcha en base a la información que obtuviéramos por radio, o de los puestos de información, sobre el estado de las carreteras, pudiendo así reaccionar a situaciones cambiantes tales como atascos, accidentes, puertos cerrados, fenómenos atmosféricos, etc. En el primer caso no intercambiamos ninguna información durante el viaje, mientras que en el segundo sí.

2.2.1 El principio de optimalidad

Aunque parezca una perogrullada es un principio fundamental para los algoritmos de encamiamiento que podemos enunciar así:

> *Si B está en la ruta óptima de A a C, entonces el camino óptimo de B a C está incluido en dicha ruta.*

Una consecuencia importante de este principio es que todas las rutas óptimas para llegar a un punto determinado en una red forman un árbol con raíz en el punto de destino. El árbol no contiene bucles, decimos que es un *spanning tree* y siempre es posible llegar al punto de destino en un número finito de saltos ('hops').

2.2.2 Encaminamiento por el camino más corto y métricas

Existen diversos algoritmos que permiten calcular el camino más corto entre dos nodos de un grafo. Uno de los mas conocidos es debido a Dijkstra.

Esta técnica se utiliza tanto en routing estático como dinámico.

Para saber elegir el camino más corto primero debemos definir que entendemos por distancia. Es evidente que en redes telemáticas no tiene mucho sentido emplear la distancia física. En los casos más simples la distancia se mide como el número de saltos o ‘hops’ (hop = salto en inglés); a mayor número de saltos mayor distancia. Evidentemente esto es satisfactorio únicamente en casos muy simples en que todos los enlaces tiene la misma capacidad. Normalmente la ‘distancia’ se calcula a partir de uno o varios de los siguientes parámetros:

- La capacidad del enlace (información estática)
- El tráfico medio (puede ser información estática o dinámica)
- El retardo (información dinámica medida a partir de los paquetes enviados)

- La fiabilidad (información dinámica medida a partir de los paquetes enviados)

Combinando de determinada forma estos parámetros se calcula una cantidad que será la 'distancia' utilizada en el cálculo de las rutas óptimas. A menudo a esa distancia se la denomina métrica. La forma de calcular la métrica se elige al configurar el router, pero es importante que sea consistente en todos los routers que participan en el protocolo de routing ya que de lo contrario se pueden dar situaciones asimétricas generalmente absurdas.

Cuando el parámetro utilizado para el cálculo de la distancia es invariable con el tiempo (por ejemplo capacidad del enlace) puede aplicarse a un algoritmo de encaminamiento estático. Se calculan todas las rutas y se elige la óptima para cada caso; una vez realizado este proceso se cargan dichas rutas en la configuración de los routers.

Si se emplean parámetros dinámicos (por ejemplo el tráfico medio o la fiabilidad) debe utilizarse un algoritmo dinámico. En este caso la información se propaga en toda la red y los cálculos se hacen de manera descentralizada entre todos los routers.

2.2.3 Encaminamiento basado en el flujo

Este algoritmo toma en cuenta la cantidad de tráfico medio que soportan las líneas, y en base a esta información intenta optimizar el conjunto de las rutas para utilizar el camino menos congestionado en cada caso.

Para aplicarlo se ha de conocer la matriz de tráfico entre los nodos y es conveniente que tráfico sea regular. Esto se puede obtener a partir de medidas de tráfico real o, si esto no es posible, en base a estimaciones lo más aproximadas posible. Se pueden aplicar algoritmos relativamente sofisticados ya que el cálculo de rutas se hace offline y se carga en el router después como rutas estáticas. Puede ser útil para diseñar la topología de una red, por ejemplo si se conectan una serie de oficinas con enlaces punto a punto se pueden plantear diversas topologías y estudiar cual es la más adecuada; también es útil cuando se trata de PVCs Frame Relay o ATM. Este tipo de estudios y optimizaciones forman parte de lo que se conoce como ingeniería de tráfico.

2.2.4 Encaminamiento por inundación ('flooding')

La inundación consiste en enviar cada paquete por todas las interfaces, excepto por la que se ha recibido. Esta es la técnica que se aplicaba en las tramas de descubrimiento en los puentes por encaminamiento desde el origen y también en los puentes transparentes cuando la dirección de destino era desconocida.

La inundación puede multiplicar el tráfico si existen bucles en la topología, ya que en ese caso se envían paquetes duplicados. En el caso extremo la red puede llegar a bloquearse, como ocurría en los puentes transparentes para lo cual se desarrolló el protocolo spanning tree. Aparte del spanning tree, que es una solución drástica a este problema (puesto que sencillamente bloquea los caminso redundantes) existen dos posibles soluciones, a saber:

Incorporar en el paquete un campo contador decremental de saltos, de forma que su valor se reduzca en uno con cada salto y que cuando dicho campo sea cero el paquete se descarte. El valor inicial deberá ser adecuado para garantizar que el paquete llegará a todos los rincones de la red.

- Identificar todos los paquetes de manera no ambigua (por ejemplo numerándolos) para que cada router mantenga una lista de los paquetes enviados; así puede evitar reenviarlos de nuevo; la lista no necesita ser exhaustiva, basta con indicar por ejemplo 'recibidos del router X todos los paquetes hasta el 1900, y además el 1912 y 1915'.

- Para limitar este problema los paquetes se fija un número máximo de saltos, que suele ser igual al número de saltos que hay entre los dos puntos más alejados de la red (decimos que ese número de saltos constituye el tamaño o *diámetro* de la red).

También puede usarse *inundación selectiva*. En este caso el paquete se envía sólo por las líneas que aproximadamente apuntan en la dirección correcta; por ejemplo si estamos en Madrid y el paquete va hacia Córdoba se enviará por las líneas que van al sur solamente.

La inundación es importante porque se utiliza como mecanismo para distribuir la información de routing en algunos algoritmos de routing que veremos y también en algunos algoritmos de routing multicast.

2.2.5 Encaminamiento por vector distancia

Este algoritmo se aplica en diversos protocolos de routing. También se conoce como algoritmo de Bellman-Ford o Ford-Fulkerson, que fueron los autores de la idea.

En el encaminamiento por vector distancia cada router mantiene una tabla o vector que le indica la distancia mínima conocida hacia cada posible destino y que línea o interfaz debe utilizar para llegar a él. La tabla se actualiza regularmente con información obtenida de los routers vecinos. Cada router manda la tabla completa de distancias a todos sus vecinos, y solo a ellos. A partir de la información que tiene y la recibida de sus vecinos cada router recalcula continuamente su tabla de distancias.

La métrica (el valor utilizado para elegir la ruta óptima) puede ser número de saltos, retardo, paquetes encolados, etc., o una combinación de estos u otros parámetros. Para medir el retardo el router puede enviar paquetes de prueba que deben ser respondidos por el router remoto, aunque también es frecuente que los retardos se asignen de acuerdo con un convenio preestablecido en función del tipo de interfaz y de la capacidad del enlace, sin hacer ninguna medida real sobre el terreno. Cada router sólo conoce el valor de los parámetros para los enlaces con sus vecinos, los valores correspondientes a enlaces más lejanos los conoce de manera indirecta en base a la información sumarizada que sus vecinos le facilitan.

En el routing por vector distancia las noticias buenas se propagan rápidamente, pero se reacciona lentamente a las malas. Esto se conoce como el problema de la **cuenta a infinito**. Se han ideado multitud de trucos para resolver este problema, pero para cada nueva propuesta se ha encontrado una situación patológica en la que falla. No existe al parecer una solución definitiva a este problema, si bien la combinación de varios de esos ‘trucos’ parece dar un resultado más que aceptable en la práctica. A pesar de sus inconvenientes el algoritmo del vector distancia se utiliza aun bastante en la actualidad, y tiene fervientes partidarios sobre todo debido a su sencillez y consiguiente economía de recursos.

El algoritmo del vector distancia fue utilizado en la ARPANET original. También se utilizó en DECNET e IPX, y se usa en Appletalk. Se usa actualmente en el protocolo RIP (Routing Information Protocol), que hasta 1988 era el único protocolo de routing utilizado en Internet. También se utiliza en los protocolos propietarios IGRP y EIGRP de Cisco, ampliamente extendidos.

2.2.6 Encaminamiento por el estado del enlace

El algoritmo de encaminamiento basado en el estado del enlace se conoce también como algoritmo de Dijkstra o algoritmo SPF (Shortest Path First).

Este algoritmo apareció como un intento de resolver los problemas que planteaba el encaminamiento por vector distancia, fundamentalmente el de la cuenta a infinito. Se trata de un algoritmo mas sofisticado y robusto, pero también más complejo. Su funcionamiento se describe mejor dividiéndolo en cuatro fases:

1. Descubrir los routers vecinos y averiguar sus direcciones.
2. Medir el retardo o costo de llegar a cada vecino
3. Construir un paquete que resuma toda esta información, y enviarlo a **todos** los routers de la red
4. Calcular el camino mas corto a cada router

Veamos con más detalle cada una de ellas:

1. Para darse a conocer cada router cuando arranca envía paquetes de presentación (HELLO) por todas sus interfaces; los paquetes HELLO son respondidos con mensajes identificativos por los routers que los reciben.

2. Para conocer el retardo de sus enlaces el router envía paquetes de prueba (ECHO) que son respondidos por los routers remotos; de esta forma el router puede medir el tiempo de ida y vuelta, del que puede deducirse el retardo.

3. Con la información obtenida el router construye un paquete de información (denominado LSP, Link State Packet) y lo envía a todos los routers de la red. Para ello utiliza inundación. Los paquetes se numeran para detectar y descartar duplicados, e ignorar paquetes obsoletos (por ejemplo si llega el paquete 26 después de haber recibido el 28 se ignora y se descarta). Además cada paquete tiene una vida limitada, al cabo de la cual es también descartado.

4. Con toda la información obtenida el router construye el árbol de expansión de las rutas óptimas a cada destino de la red aplicando el algoritmo de Dijkstra (inventado por Edsgar Dijkstra). De esta forma conoce la topología de la red.

En el routing por el vector distancia cada router envía información sólo a sus vecinos, pero esta información incluye a todos los nodos de la red. En cambio en el routing por el estado del enlace cada router envía su LSP a toda la red, pero éste solo contiene información relativa a sus vecinos más próximos. En el algoritmo basado en el estado del enlace cada router puede, a partir de la información obtenida, conocer su árbol de expansión o spanning tree completo, mientras que esto no es posible con routing por el vector distancia.

La distribución de los LSPs merece un comentario especial. Como hemos dicho, una vez distribuida la información de los LSPs en toda la red y convergida la topología cada router conoce perfectamente el árbol de expansión que le corresponde. Cabría pensar entonces en optimizar la distribución de los LSPs, haciéndola de acuerdo con dicho árbol de expansión en lugar de utilizar inundación. Sin embargo si en ese caso se produjera un fallo de alguno de los enlaces del árbol los LSPs no se distribuirían en toda la red, quedando esta dividida en dos. La red quedaría parcialmente fuera de servicio y se habría perdido una de las ventajas principales de utilizar un protocolo de routing, que es la posibilidad de reaccionar a los cambios en la topología debidos a fallos de la red. Por otro lado el uso de la inundación para distribuir los LSPs es menos problemático de lo que a primera vista podría parecer, ya que gracias a la detección de LSPs duplicados que realizan los routers el máximo número de veces que un mismo LSP puede pasar por un mismo enlace es de dos, y en muchos casos solo pasa una vez por cada enlace.

El algoritmo del estado del enlace no tiene el problema de la cuenta a infinito, es más robusto y fiable, pero también es más complejo, requiere mayor cantidad de cálculos y por tanto una CPU más potente y mayor cantidad de memoria RAM en el router.

Entre los protocolos de routing que utilizan algoritmos basados en el estado del enlace se encuentra OSPF (Open Shortest Path First) que es el protocolo de routing estándar de Internet (aunque también se utilizan otros). Otro protocolo basado en el algoritmo del estado del enlace también utilizado en Internet y que proviene del mundo OSI es IS-IS (Intermediate System-Intermediate System). IS-IS es multiprotocolo, es decir, soporta múltiples protocolos de red por encima. OSPF esta basado en IS-IS, pero no es multiprotocolo.

2.2.7 Encaminamiento jerárquico

Aplicamos encaminamiento jerárquico en la decisión de la ruta óptima cuando planificamos un viaje largo por carretera. Por ejemplo para ir en coche de Valencia a Bruselas utilizamos un mapa detallado de España y Bélgica, pero no empleamos normalmente un mapa detallado de Francia, nos basta con uno de las carreteras principales de ese país.

A medida que una red telemática crece la cantidad información de routing aumenta de forma no lineal, ya que cada router ha de calcular sus rutas óptimas a todos los demás. Generalmente el aumento en el número de routers de una red viene acompañado de un aumento aún mayor en el número de enlaces que

los unen, lo cual hace crecer el número de vectores distancia o el tamaño de los LSP. El tráfico de routing en la red aumenta y lo mismo ocurre con la CPU y memoria consumida en los routers para realizar los cálculos necesarios para obtener las rutas óptimas. En resumen, los algoritmos de routing no son *escalables*[3].

Para resolver este problema muchos protocolos de routing contemplan la posibilidad de establecer dos o más niveles jerárquicos; la red se divide en regiones, y sólo un número reducido de routers de cada región (los routers interregionales) puede comunicar con el exterior. Las rutas pueden no ser tan óptimas en algún caso, pero se simplifica la gestión y el mantenimiento de las tablas de routing y se reduce el tráfico en la red debido al protocolo de routing.

2.2.8 Encaminamiento broadcast

En algunos casos se necesita enviar un paquete a todos los destinos posibles en una red, es decir se quiere hacer un envío broadcast.

La forma más sencilla de conseguirlo es **inundación**, técnica especialmente apropiada en este caso que ya hemos descrito en detalle anteriormente.

Otro método es el **routing multidestino**, que consiste en mandar un único paquete con todas las direcciones de destino; el paquete es replicado en cada router por las interfaces por donde debe enviarse, es decir, las que son parte de la mejor ruta para alguno de los destinos indicados.

Otro algoritmo es construir el árbol de expansión o **spanning tree** con raíz en el origen y seguirlo, replicando el paquete allí donde haya una bifurcación. El spanning tree no tiene bucles. Este sistema es óptimo, ya que se asegura que la distribución se hará generando el número mínimo de paquetes y sin envíos duplicados. Pero esto requiere que cada router conozca cuales de sus interfaces forman parte del spanning tree para el router origen y cuales no, es decir los routers han de conocer en detalle la topología de la red. Con routing del estado del enlace los routers poseen esta información.

Por último el algoritmo de **encaminamiento por el camino inverso** o RPF (Reverse Path Forwarding) es en cierto modo una aproximación o una versión simplificada del spanning tree que se puede aplicar cuando se tiene información parcial de la topología, como en el caso de los algoritmos basados en el vector distancia. El mecanismo es el siguiente: el router examina la dirección origen del paquete recibido, y la interfaz por la que le ha llegado; si esa interfaz es la que el router tiene identificada como el camino más corto para llegar a la dirección de origen se considera bastante probable que el paquete no sea un duplicado, por lo que lo reenviará por todas las interfaces, excepto por aquella por la que vino; si la interfaz por la que llegó el paquete no es la indicada para enviar paquetes a dicha dirección de origen se considera que se trata de un duplicado y se descarta. Esta técnica realiza dos envíos extra por cada posible bucle en la red, es decir consigue una eficiencia bastante buena aunque no es óptima ya que se generan algunos envíos duplicados. Además parte de la suposición de que las rutas óptimas son siempre simétricas; si esta condición no se cumple el algoritmo no funciona.

2.2.9 Encaminamiento multicast

El problema del encaminamiento multicast tiene dos partes: por un lado la gestión de los grupos multicast y por otro el encaminamiento multicast propiamente dicho.

---

[3] Decimos que algo (un servicio, algoritmo, etc.) no es escalable cuando un incremento en las condiciones de partida (número de usuarios, cantidad de variables, etc.) requiere incrementar en mayor proporción los recursos asociados (para proveer el servicio, resolver el algoritmo, etc.). Por ejemplo en nuestro caso duplicar el número de routers hace aumentar en un factor superior a dos el tráfico de routing y la complejidad de los cálculos. En el caso de servicios a veces se considera que un aumento lineal no es suficiente, es decir, para que el servicio se pueda calificar de escalable es preciso que los recursos puedan crecer en una proporción menor, o mucho menor, que el número de usuarios.

Para el envío de paquetes multicast primero hay que crear un grupo multicast. Una vez creado el grupo los usuarios pueden unirse a él (*join*) o abandonarlo (*leave*). Las tareas de gestión del grupo son el objeto de un protocolo de gestión de grupos que es independiente del routing multicast.

Cuando un usuario (o más exactamente un proceso que depende de un host conectado a un router), se une a un grupo multicast debe informar a su router y éste informará de ello a sus vecinos.

Dentro de una red local sin routers que la dividan (una Ethernet conmutada, por ejemplo) el envío de paquetes multicast resulta especialmente sencillo, ya que al tener una topología plana desde el punto de vista del routing los paquetes simplemente se envían al medio físico y serán captados por todas aquellas estaciones que pertenezcan a ese grupo multicast. Como el tráfico multicast atraviesa los conmutadores los usuarios podrán unirse al grupo desde cualquier parte de la red local.

En una red con routers (una red de área extensa por ejemplo) la situación es mucho más compleja: los routers que constituyen el grupo multicast forman un árbol de distribución (spanning tree) con raíz en el router donde se encuentra el emisor de dicho tráfico multicast. Cuando un host quiere unirse a un grupo multicast notifica a su router, el cual solicita entonces unirse al grupo multicast correspondiente, es decir solicita ser añadido al árbol de distribución multicast (salvo que el router ya estuviera en el árbol, en cuyo caso solo toma nota del nuevo host interesado en el grupo y no manda ningún mensaje al exterior). Cuando más tarde el usuario quiere abandonar el grupo enviará un nuevo mensaje a su router indicándoselo; en este momento el router pedirá a su vez al siguiente en el árbol que le corte el flujo multicast que había pedido (suponiendo que no haya entonces ningún otro usuario dependiente de dicho router que quiera seguir participando en el grupo multicast). La labor de cortar la distribución multicast se conoce como 'pruning' o podado del árbol[4]. El objetivo del podado es dejar en el árbol sólo las ramas necesarias para hacer llegar el tráfico multicast a aquellos routers que tienen usuarios que forman parte del grupo multicast.

Un mismo grupo multicast puede tener varias fuentes, es decir varios emisores, en cuyo caso existirá un árbol de distribución diferente para cada fuente. Además una misma fuente puede emitir simultáneamente en varios grupos multicast. En principio un emisor en un grupo multicast no tiene por que ser miembro de dicho grupo; sin embargo por la forma como se desarrollan las aplicaciones multicast y como funcionan sus protocolos de control asociados en la práctica el emisor de un grupo multicast es también miembro de dicho grupo y viceversa, es decir los miembros en principio solo receptores de un grupo son también emisores, si bien ocasionales, del mismo.

Como puede intuirse por los aspectos comentados en este apartado el routing multicast es algo bastante más complejo que el routing unicast, más incluso que el routing broadcast, debido en buena medida al control que los routers han de mantener de los hosts activos en cada grupo en su red local. Esto obliga a los routers a mantener una cierta información de estado que si no responde rigurosamente a la realidad acaba por degradar completamente el servicio. Esta ha sido la principal causa de que el routing multicast siga siendo aun hoy en día un servicio experimental poco extendido en Internet.

2.3 ALGORITMOS DE CONTROL DE CONGESTIÓN

Denominamos congestión a la circunstancia en la que el rendimiento de la red (o una parte de ella) se degrada.

Un ejemplo típico de congestión sería la situación en la que un router con varias líneas de 2 Mb/s recibe tráfico entrante por todas ellas dirigido a una sola. Inicialmente el router intentará salvar la situación utilizando sus buffers, pero si la situación dura lo suficiente los buffers se llenarán y el router empezará a descartar paquetes Normalmente la congestión se produce por tráfico excesivo, pero también puede producirse por un router sobrecargado o de capacidad insuficiente para el tráfico que soporta (CPU lenta).

[4] Podar = ‘to prune’ en inglés.

Conviene no confundir la congestión con el control de flujo, aunque ambas estén relacionadas. A diferencia de la congestión el control de flujo es una circunstancia que solo se da en conexiones punto a punto a nivel de enlace o a nivel de transporte. Una de las posibles consecuencias del control de congestión es ejercer control de flujo sobre el nodo o nodos que están produciendo la congestión.

La congestión es generalmente un problema más complejo que el control de flujo, ya que generalmente el emisor del tráfico es un router, es decir un intermediario que no tiene el control de la situación. Lo más que puede hacer un router es enviar un mensaje de control de congestión; a menudo para cuando el mensaje llega al verdadero causante de la congestión el tráfico ya ha cesado, con lo que resulta inútil tomar medidas. Este problema se acentúa especialmente en redes de alta velocidad y elevado retardo (gran distancia).

2.3.1 Principios generales del control de congestión

Para el control de la congestión caben dos planteamientos:

- Diseñar las cosas desde el principio para que la congestión no pueda llegar a ocurrir (medicina preventiva).
- Tomar medidas que permitan detectar la congestión y adoptar medidas correctoras en su caso (medicina curativa).

La primera técnica es mas segura, pero puede provocar ineficiencias si se aplican las limitaciones con demasiada severidad. La segunda permite aprovechar mejor la red, pero en caso de congestión puede ser difícil controlar la situación.

Entre los parámetros que permiten detectar la presencia de congestión se encuentran los siguientes:

A nivel de red:

- Porcentaje de paquetes descartados
- Longitud media de las colas en las interfaces de los routers

A nivel de transporte:

- Retardo medio por TPDU (Transport Protocol Data Unit)
- Desviación media del retardo por TPDU (jitter)
- Número de TPDUs que se pierden o dan timeout y se retransmiten (se supone que esto no se debe a errores)

Para informar sobre situaciones de congestión el receptor pueden utilizar paquetes de alerta generados a propósito o incluir la información en un campo especial de los paquetes de datos (aviso 'piggybacked'). También puede el emisor enviar paquetes de sondeo para averiguar el estado de la red.

Para resolver la congestión solo hay dos posibles medidas:

- Reducir el tráfico solicitando al emisor que pare de enviar.
- Aumentar la capacidad; a corto plazo esto puede hacerse por ejemplo activando canales RDSI o buscando rutas alternativas; a más largo plazo será preciso contratar enlaces de más capacidad o enlaces adicionales.

2.3.2 Factores que pueden influir en la congestión

Podemos encontrar factores que influyen en la congestión de una red en el nivel de enlace, el nivel de red y el nivel de transporte.

Entre los factores a nivel de enlace que pueden influir en la congestión se encuentran:

- El intervalo de timeout; si es pequeño originará retransmisiones innecesarias
- El tamaño de ventana; si es grande es más fácil que se produzca congestión
- El uso de retroceso n o repetición selectiva; el retroceso n genera más tráfico
- El uso o no de ACK piggybacked; si no se usa se genera más tráfico

Todos estos factores suponen la existencia a nivel de enlace de un protocolo orientado a conexión, es decir que solicita confirmación mediante ACK de los envíos realizados. Dado que hoy en día esto no es normal en la práctica el nivel de enlace tiene poco que ver con la congestión.

En el nivel de red los factores que influyen en la congestión son los siguientes:

- Servicio CONS vs CLNS, o uso de circuitos virtuales vs datagramas :hay mejor control de la congestión cuando se trata de circuitos virtuales.
- Mecanismos de encolamiento y criterios de selección y priorización de paquetes (Calidad de Servicio).
- Mecanismos de descarte de paquetes: a veces hay paquetes marcados como candidatos a la hora de descartar.
- Algoritmo de routing: si contempla la posibilidad de utilizar caminos alternativos quizá sea posible evitar una zona congestionada.
- Vida media de los paquetes, o timeout: si el valor es muy pequeño los paquetes tendrán que ser retransmitidos, si es excesivo serán descartados cuando lleguen y se habrán transmitido inútilmente.

En el nivel de transporte se dan básicamente los mismos factores que en el nivel de enlace; la principal diferencia está en la estimación del timeout adecuado, que es mucho más difícil al no tratarse de una comunicación entre entidades vecinas.

2.3.3 Perfiles de tráfico y vigilancia de tráfico ('traffic shaping' y 'traffic policing')

El tráfico a ráfagas es la principal causa de congestión. Si todos los ordenadores transmitieran siempre un flujo constante sería muy fácil evitar la congestión.

Los *perfiles de tráfico* (*traffic shaping*) establecen unos márgenes máximos al tráfico a ráfagas. Suelen utilizarse para fijar una Calidad de Servicio (Quality of Service, QoS) entre el operador y el usuario; entretanto el usuario respete lo establecido el operador se compromete a no descartar paquetes; el perfil de tráfico actúa pues como un contrato que vincula a ambas partes.

Se denomina *vigilancia de tráfico* (*traffic policing*) a la labor de monitorización o seguimiento del tráfico introducido por el usuario en la red para verificar que no excede el perfil pactado.

Uno de los sistemas más utilizados para establecer perfiles de tráfico es el conocido como *algoritmo del pozal agujereado* (*leaky bucket*): el host puede enviar ráfagas que son almacenadas en un buffer en la

interfaz o conmutador de acceso a la red, de forma que la red recibe siempre un caudal constante; si la ráfaga es de tal intensidad o duración que el buffer (pozal) se llena, los paquetes excedentes son descartados, o bien son enviados a la red con una marca especial que les identifica como de 'segunda clase' (hasta cierto punto podríamos decir que son paquetes 'ilegales'); dichos paquetes de segunda clase serán los primeros candidatos a descartar en caso de congestión, puesto que se supone que el usuario que los envió era consciente de que estaba 'infringiendo las reglas' y que por tanto viajaban sin garantías. Esta técnica se utiliza en ATM y en Frame Relay y se está experimentando su utilización en IP.

Para definir un pozal agujereado se utilizan dos parámetros: ρ especifica el caudal con que sale el flujo a la red, y C indica la capacidad del buffer. Por ejemplo supongamos que con un pozal agujereado de ρ=20 Mb/s y C= 10 Mbits un ordenador envía una ráfaga de 10 Mbits en 50 mseg (equivalente a 200 Mb/s), con lo cual casi 'llena' el pozal; la ráfaga tardará en enviarse a la red 500 mseg; momento en el que el pozal se vaciará. Si el ordenador envía otra ráfaga de 10 Mbits antes de que el pozal se haya vaciado por completo el buffer se llenará y se perderán los paquetes excedentes, o bien se enviarán marcados como descartables.

El pozal agujereado resuelve el problema de las ráfagas en la red, pero no estimula el ahorro. Supongamos dos hosts, A y B, ambos con pozales agujereados de ρ=20 Mb/s y C= 10 Mbits; el host A lleva una hora transmitiendo a la red con un caudal constante de 20 Mb/s, mientras que el segundo lleva una hora sin transmitir. De repente los dos hosts tienen necesidad de emitir una ráfaga y como ambos hosts tienen su pozal vacío los dos se encuentran en igualdad de condiciones, es decir al host B no le ha beneficiado en nada el menor consumo de recursos que ha hecho durante la hora anterior. El *algoritmo del pozal con crédito* (*token bucket*) es una versión mejorada del pozal agujereado que intenta resolver este problema, premiando al host inactivo (B en nuestro ejemplo) frente al que esta siempre transmitiendo (A). El mecanismo que sigue para ello es el siguiente: cuando el host no envía datos el pozal va sumando créditos (tokens) hasta un máximo igual a C, la capacidad del buffer; el crédito se acumula a la misma velocidad con que se vacía el pozal, es decir con un caudal igual al parámetro ρ; cuando hay tráfico el crédito acumulado puede utilizarse para enviar ráfagas con un caudal *M* mayor de lo normal; cuando se agota el crédito el caudal vuelve a su valor normal ρ y el algoritmo funciona como un pozal agujereado. Un detalle sutil a tener en cuenta es que mientras se consumen los créditos (es decir, mientras sale tráfico con caudal M) también se acumula crédito con un caudal ρ. Una vez agotado todo el crédito, cuando el funcionamiento revierte al de un pozal agujereado normal, deja de acumularse crédito entretanto esté saliendo tráfico a la red con el caudal ρ. Podemos imaginar el pozal con crédito como un pozal dotado de dos agujeros, uno pequeño (ρ) y uno grande (M), con un dispositivo mecánico que permite abrir uno u otro, pero no ambos a la vez; el agujero grande se abre cuando el usuario tiene crédito; cuando lo agota se cierra y se abre el agujero pequeño; el usuario acumula crédito siempre que el pozal no esté enviando tráfico por el agujero pequeño, bien porque tenga abierto el grande (y tenga por tanto cerrado el pequeño) o porque no haya tráfico que enviar. Los parámetros que definen un pozal con crédito son el caudal del agujero pequeño ρ, la capacidad del buffer *C* y el caudal del agujero grande *M* (normalmente igual a la velocidad de la interfaz física).

Supongamos que ρ=20 Mb/s, C= 10 Mbits y M = 200 Mb/s y que como en el ejemplo anterior el host envía una ráfaga de 10 Mbits en 50 mseg. Supongamos también tres posibles situaciones del pozal: a) lleno de créditos (10 Mbits), b) medio lleno de créditos (5 Mbits) y c) sin créditos. En el primer caso, si el buffer está lleno de créditos cuando llega la ráfaga ésta será enviada a la red a 200 Mb/s, o sea a la misma velocidad con que la envía el host. Si el buffer está solo medio lleno (5 Mbits de créditos) se producirá una ráfaga de 200 Mb/s durante 27,78 mseg y seguirá un caudal de 20 Mb/s durante 222,2 mseg (hay que tomar en cuenta que *durante* los primeros 27,78 mseg siguen llegando créditos pues esta cerrado el agujero pequeño, pero no después ya que se van consumiendo a medida que llegan). Por último, si no hubiera créditos disponibles cuando llegara la ráfaga el comportamiento sería idéntico al del pozal agujereado.

Quizá merece la pena comentar como se calcula el valor de 27,78 mseg del ejemplo anterior para el caso del pozal medio lleno de créditos. Sabemos que mientras esta abierto el agujero grande el pozal suma créditos a una velocidad de 20 Mb/s; por tanto el total de créditos obtenidos en un tiempo *t* será:

créditos totales = créditos iniciales + créditos acumulados = 5.000.000 + 20.000.000 * t

Por otro lado, sabemos que mientras está abierto el agujero grande los créditos se consumen a razón de 200 Mb/s, o sea:

créditos consumidos = 200.000.000 * t

Para calcular cuanto tiempo estará abierto el agujero grande igualamos las dos expresiones anteriores:

5.000.000 + 20.000.000 * t = 200.000.000 * t

De donde despejando obtenemos para t el valor de 0,02778 seg

El pozal con crédito supone una mejora respecto al pozal agujereado por cuanto que distingue entre un usuario que ha estado transmitiendo y otro que no ha transmitido nada, pero tiene el inconveniente de que mientras hay crédito inyecta tráfico en la red con un caudal M que normalmente coincide con la capacidad de la interfaz física, lo cual puede resultar excesivo en ocasiones. Para dosificar esta avalancha a veces se combina un pozal con crédito seguido de un pozal agujereado de ρ mayor, con lo que se permiten esas ráfagas pero de forma más controlada. Por ejemplo si el pozal con crédito del ejemplo anterior (ρ=20 Mb/s, C= 10 Mbits y M = 200 Mb/s) se combina con un pozal agujereado de ρ=50 Mb/s y C= 10 Mbits el resultado para cada uno de los tres supuestos anteriores será como sigue:

a) La ráfaga es enviada a la red a 50 Mb/s.

b) Se envía una ráfaga de 50 Mb/s durante un tiempo *t* donde *t* se calcula a partir de la siguiente fórmula:

   Bits enviados a 50 Mb/s = Bits recibidos a 200 Mb/s + Bits recibidos a 20 Mb/s

   que aplicada a este caso resulta:

   50.000.000 * t = 0,02778 * 200.000.000 + (t – 0,02778) * 20.000.000

   Despejando obtenemos un valor de t de 0,16668, por lo que la ráfaga de 50 Mb/s será de 166,7 ms, que irá seguida de un flujo de 20 Mb/s durante 83,3 ms, hasta completar el envío de los 10 Mb.

c) Al no haber crédito en el pozal en este caso la ráfaga se envía toda a 20 Mb/s por lo que el segundo pozal no tiene efecto en este caso.

2.3.4 Control de admisión

El control de admisión se aplica únicamente a las redes orientadas a conexión y consiste en evitar el establecimiento de nuevos circuitos virtuales que pasen por una zona de la red que se considera congestionada. Si se conoce la capacidad máxima que necesita cada circuito virtual en principio es posible controlar el acceso de forma que nunca se produzca congestión. Generalmente un control de admisión estricto no usa los recursos de manera eficiente ya que al reservar para cada circuito la capacidad máxima la red se encuentra infrautilizada la mayor parte del tiempo, por lo que a veces se preve un cierto grado de sobresuscripción, u 'overbooking' (de forma parecida a las compañías aéreas cuando venden mas billetes que el número de asientos disponibles en el avión).

La red telefónica es un ejemplo extremo de control de admisión; en esta red se asigna y reserva un circuito de 64 Kb/s en el momento de establecer el circuito, para cada comunicación telefónica; la posibilidad o no de comunicación depende exclusivamente de la disponibilidad de recursos en el momento de establecer la comunicación. Gracias a este control de admisión estricto la red telefónica no tiene nunca problemas de congestión, aunque esta ventaja se consigue a costa de un derroche de recursos y un sobredimensionamiento de la red.

2.3.5 Paquetes de alerta

Los paquetes de asfixia se puede aplicar tanto en redes de circuitos virtuales como de datagramas.

En esta técnica el router o conmutador comprueba regularmente cada una de sus líneas, monitorizando por ejemplo la carga relativa (nivel de ocupación) o la longitud de las colas en las interfases. Cuando el parámetro inspeccionado supera un determinado valor considerado umbral de peligro se envía un paquete de alerta al host considerado 'culpable' para que reduzca el ritmo.

Los paquetes de alerta tienen el riesgo de producir comportamientos oscilantes, ya que cuando se percibe una situación peligrosa se envían avisos que pueden bloquear a todos los emisores; más tarde al detectar que el problema está resuelto se pide reanudar los envíos, con lo que hay riesgo de caer nuevamente en la situación de alerta, y así sucesivamente. Este problema se resuelve en parte haciendo que el router reaccione con cierta inercia a los cambios que se producen en el parámetro monitorizado; para ello se suelen utiliza fórmulas del tipo:

$$u_n = au_{n-1} + (1-a)\, f$$

donde f es el valor más reciente del parámetro medido (grado de utilización del enlace, por ejemplo), u_n el valor medio en la n-ésima iteración, y a una constante que permite regular el peso que se le quiere dar a los valores anteriores, o dicho de otro modo regular la rapidez con que se reaccionará a situaciones cambiantes. Cuanto mayor sea el valor de a mas lenta será la respuesta a los cambios. De este modo se evita responder a cambios momentáneos que harían mas mal que bien en el rendimiento de la red. Por ejemplo con $a = 0{,}5$ el valor actual sería la media aritmética entre el último valor instantáneo y el valor promedio anterior (que a su vez era la media aritmética entre el penúltimo valor instantáneo y el valor promedio anterior, y así sucesivamente).

Normalmente los paquetes de alerta se envían a los hosts que generan el tráfico, ya que son éstos y no los routers los verdaderos causantes de la congestión. Los hosts cuando reciben estos paquetes suelen reducir (por ejemplo a la mitad) la velocidad con la que envían datos a la red. Esto lo pueden hacer de varias maneras, por ejemplo reduciendo el tamaño de ventana del protocolo a nivel de transporte o cambiando los parámetros del pozal agujereado o del pozal con crédito, si utilizan alguno de estos algoritmos para controlar el flujo. En una situación de congestión normalmente muchos hosts recibirán este tipo de paquetes.

En ocasiones interesa que los paquetes de alerta tengan un efecto inmediato en cada uno de los routers por los que pasan en su camino hacia el host que provoca la congestión. De esta forma la congestión se reduce de forma inmediata, distribuyendo el tráfico en ruta entre los buffers de los routers que hay en el camino entretanto el mensaje de alerta llega al host que genera el tráfico. Esto es especialmente importante cuando se trata de una conexión de alta velocidad y elevado retardo.

Por desgracia la obediencia a los paquetes de alerta es completamente voluntaria. Si un host obedece las indicaciones y reduce su ritmo mientras otro no lo hace el primero saldrá perjudicado pues obtendrá una parte aún menor de la ya escasa capacidad disponible. No es posible obligar a los hosts a obedecer las indicaciones de los paquetes de alerta.

En situaciones de saturación se plantea el problema de como repartir la capacidad disponible de forma justa entre los usuarios. Existen varios algoritmos que intentan resolver este problema, por ejemplo:

- *Encolamiento justo (fair queuing):* el router mantiene una cola independiente por cada host y envía los paquetes en turno rotatorio ('round robin'). Aquí se da un problema parecido al de Ethernet, ya que el reparto es equitativo en paquetes por host por unidad de tiempo; en este caso los usuarios o aplicaciones que manejan paquetes grandes obtienen más recursos que los que manejan paquetes pequeños. Una versión mejorada de este algoritmo intenta hacer un reparto equitativo en número de bytes (o bits) transmitidos por host por unidad de tiempo.

- *Encolamiento justo ponderado (weighted fair queuing):* es similar al anterior, pero permite además establecer prioridades, ya que en ocasiones interesa dar más prioridad a algunos hosts (por ejemplo servidores) o a algún tipo de tráfico (aplicaciones en tiempo real). La prioridad se

puede establecer por direcciones, por tipo de aplicación o por una combinación de estos u otros factores. Además de prioridades también se pueden reservar capacidades para ciertas direcciones o aplicaciones, por ejemplo 'reservar el 30% de la capacidad para el servidor ftp' o 'reservar el 20% de la capacidad para paquetes provinientes de la dirección IP 147.156.1.1'.

2.3.6 Descarte de paquetes

El último recurso para resolver un problema de congestión es descartar paquetes. Esto evidentemente ayuda a reducir la congestión puesto que se reduce el tráfico en la red, pero además del alivio inmediato que supone el descarte de paquetes hay un efecto a más largo plazo debido a que en muchos casos los protocolos de transporte incorporan mecanismos de autorregulación que entran en funcionamiento cuando se detecta la pérdida de algún paquete. El ejemplo más claro en este sentido lo constituye TCP, que se basa en este mecanismo implícito de notificación para detectar y resolver las situaciones de congestión. Actualmente este es prácticamente el único mecanismo de control de congestión que funciona con carácter general en la Internet.

El descarte de paquetes puede dar lugar en ciertas circunstancias a situaciones oscilantes por razones similares a las comentadas al hablar de los paquetes de alerta. Supongamos que como consecuencia de la congestión el router descarta paquetes de muchos hosts a la vez; todos los hosts a los que se les haya descartado siquiera sea un paquete detectarán esta circunstancia y todos ellos bajarán drásticamente su ritmo de emisión de paquetes. Al ver que ya no se pierden más paquetes los hosts irán recuperando paulatinamente la confianza hasta que llegará un momento en que se repita la situación de congestión y se produzca de nuevo el descarte masivo, con lo que los hosts disminuirán de nuevo a ritmos mucho más conservadores. Como consecuencia de este comportamiento se producen situaciones cíclicas en las que se alternan unos períodos de congestión con otros de baja actividad en los que la red está infrautilizada. Como cabría esperar el resultado es nefasto desde el punto de vista del rendimiento de la red. La solución a este problema, que se viene utilizando desde hace varios años en los routers del 'core' de Internet, consiste en descartar paquetes mucho antes de que los indicadores de congestión lleguen a tener valores peligrosos, pero no descartar muchos paquetes sino solo algunos correspondientes a hosts elegidos al azar; de esta forma el aviso de congestión llega solo a una parte de los hosts que reducen su caudal de tráfico, pero la mayoría sigue normalmente. Esta técnica, conocida como RED[5], permite un aprovechamiento mucho más eficiente de los enlaces.

En ocasiones los paquetes llevan alguna indicación de su grado de importancia, en cuyo caso los routers intentan descartar los menos importantes primero. Por ejemplo, sería bastante grave si un router para resolver una situación de congestión descartara paquetes de alerta.

A veces el nivel de aplicación puede dar información sobre la prioridad de descarte de los paquetes. Por ejemplo en aplicaciones isócronas (audio y vídeo en tiempo real) suele ser preferible descartar el paquete viejo al nuevo ya que el viejo seguramente es inútil, mientras que en transferencia de ficheros ocurre lo contrario pues el receptor necesita recibirlos todos y el más antiguo causará antes retransmisión por agotamiento del timeout. En los ficheros MPEG[6], debido a la técnica de compresión utilizada algunos fotogramas son completos (los denominados fotogramas *intra*) y otros son diferencias respecto a los anteriores y/o posteriores (llamados fotogramas *predictivos* y *bidireccionales*); descartar un paquete perteneciente a un fotograma intra es más perjudicial para la calidad de la imagen que descartar uno de un fotograma predictivo o bidireccional, ya que el defecto repercutirá en todos los fotogramas que dependan de él . En ATM y Frame Relay existen situaciones en las que el usuario puede inyectar en la red un

[5] El mecanismo RED fue inventado por Cisco. Inicialmente los ingenieros se referían a este acrónimo como Random Early Discard, ya que en esencia lo que hacía era descartar paquetes; sin embargo los especialistas de marketing consideraron que reconocer abiertamente que sus routers descartaban paquetes podía perjudicar la imagen de la marca. Por este motivo las siglas se mantuvieron, pero cambiando el significado oficial de la D a 'Detect' (Random Early Detect), lo cual bien pensado resulta aún peor, ya que parece indicar que la pronta detección de las situaciones de congestión se producirá de forma aleatoria, lo cual resultaría preocupante en caso de ser cierto. Aunque no guste a los especialistas de marketing incluso los mejores routers de Cisco (como los de cualquier otra marca) descartan paquetes cuando sufren congestión y sus buffers se llenan.

[6] MPEG es el formato estándar ISO de vídeo digital comprimido.

caudal superior al contratado, pero dicho tráfico excedente puede ser descartado en caso de congestión; para esto los conmutadores ATM y Frame Relay 'marcan' los paquetes o celdas que entran por encima del caudal pactado. Si la aplicación no marca ningún paquete como descartable la red marcará automáticamente aquellos que superen el caudal pactado, que podrían ser precisamente los más importantes para la aplicación. En vez de dejar en manos de la red tan importante decisión la aplicación puede especificar que paquetes deben marcarse como descartables porque sabe que son menos importantes. En cualquier caso la red se asegurará de que el tráfico no descartable inyectado nunca supere el caudal contratado.

En algunos casos el paquete transmitido por la red es parte de una secuencia correspondiente a otro paquete de mayor tamaño que viaja fragmentado. En estos casos si se descarta un paquete cualquiera de la secuencia se tendrá que reenviar todo el grupo, por lo que en caso de descartar uno es conveniente descartar todos los demás ya que son tráfico inútil. Un buen ejemplo de esto se da cuando se utiliza una red ATM para transportar datagramas IP; generalmente cada datagrama ocupa varias celdas ATM (los datagramas IP sobre ATM pueden llegar a tener una longitud de 9180 bytes, por lo que un paquete IP se puede llegar a ocupar hasta 192 celdas ATM). Si se descarta una sola celda de un datagrama ya no tiene sentido enviar el resto. La mayoría de los conmutadores ATM actuales implementan las técnicas conocidas como Early Packet Discard y Partial Packet Discard consistentes en que el conmutador realiza un descarte inteligente de celdas cuando transporta datagramas IP, es decir si se detecta que una celda perteneciente a un datagrama IP ha sido descartada automáticamente se descartan todas las demás celdas pertenecientes al mismo datagrama. Los conmutadores que no implementan estas técnicas sufren una mayor merma de rendimiento cuando se produce algún descarte de celdas. En una experiencia con un conmutador ATM de primera generación (que no disponía de descarte inteligente de celdas) se probó a inyectar por una línea E3 (34,368 Mb/s) un caudal IP ligeramente superior a la capacidad de la línea, con lo que se producía un cierto descarte de celdas; el resultado fue el que se muestra en la tabla 2.2.

Caudal inyectado (Mb/s)	Celdas perdidas (%)	Rendimiento UDP (Mb/s)
34	0,0	29,9
35	0,8	2,8
36	4,0	0,005

Tabla 2.2.- Rendimiento obtenido para tráfico UDP/IP en un conmutador ATM sin descarte inteligente de celdas.

Como puede verse el efecto sobre el rendimiento es dramático. Esta experiencia se realizó con el protocolo de transporte UDP. De haber utilizado TCP el propio mecanismo de control de congestión que incorpora este protocolo habría forzado una autorregulación del tráfico con lo que la merma de rendimiento habría sido mucho menor.

2.4 EJERCICIOS

1. Indique si es verdadera o falsa cada una de las siguientes afirmaciones:

a) Los protocolos de routing basados en el estado del enlace emplean algoritmos mas complejos que los basados en el vector distancia, pero a cambio permiten obtener un detalle completo de la topología de la red.

b) La principal ventaja de utilizar múltiples niveles jerárquicos en un protocolo de routing estriba en la posibilidad de elegir la ruta óptima en cada caso.

c) La transmisión multicast optimiza el uso de las líneas de transmisión, al permitir que múltiples usuarios reciban un mismo flujo de datos enviados desde una fuente.

d) La noción de router 'stateless' (sin estados) está asociada a protocolos de red no orientados a conexión (p. ej. IP o CLNP).

e) En general los protocolos de red orientados a conexión (Frame Relay o ATM por ejemplo) permiten controlar mejor las situaciones de congestión que los no orientados a conexión, (IP o CLNP por ejemplo).

2. Cuando se estudian mecanismos de control de congestión en una red de ordenadores existen muchos paralelismos con los problemas de tráfico en las grandes ciudades; incluso algunas técnicas son comunes a ambas situaciones. Indique un mecanismo aplicado en el control de congestión de redes telemáticas que nunca será implementado (afortunadamente) en las vías públicas.

3. Una empresa dispone de una aplicación de audioconferencia (telefonía sobre Internet) y desea utilizarla para mantener reuniones regulares entre dos oficinas, las cuales disponen de sendas redes locales unidas entre sí por dos routers y una línea de 128 Kb/s; la línea tiene unos niveles de ocupación elevados a ciertas horas del día y su velocidad no puede aumentarse por problemas presupuestarios; afortunadamente el nivel de ocupación en dicha línea sigue un perfil muy regular en función de la hora del día, y se dispone de una gráfica que muestra el tráfico medio para cada hora del día.

La aplicación de audioconferencia codifica la voz con un sistema de compresión que genera un paquete de 164 bytes (incluidas las cabeceras a nivel de red) cada 40 milisegundos; para que la voz llegue de forma comprensible y sea posible la interactividad es preciso que el retardo medio no supere los 80 milisegundos.

Indique cual es el grado de ocupación (en Kb/s y en %) por encima del cual no merece la pena intentar establecer la audioconferencia, ya que esta no podrá celebrarse en condiciones aceptables.

Considere que todo el trafico por la línea esta formado por paquetes de 164 bytes.

Suponga ahora que la empresa decide sustituir la línea de 128 Kb/s por una línea de 2048 Kb/s. ¿Cual sería en ese caso el grado de ocupación máximo tolerable (en Kb/s y en %) para poder celebrar la audioconferencia?

2.5 SOLUCIONES

S1.-

a) **Verdadera.** El routing basado en el vector distancia solo permite saber cual es el siguiente nodo en el camino óptimo para cada destino, pero se desconoce la ruta, solo se sabe la distancia. En routing basado en el estado del enlace cada nodo sabe la mejor ruta a cada destino, con lo que dispone de un 'mapa' completo de la red; esto requiere algoritmos mas complejos de cálculo de las rutas, pero permite un routing mas robusto.

b) **Falsa**. La razón de establecer niveles jerárquicos es reducir la cantidad de información que maneja el protocolo de routing, pero esto puede hacer que las rutas no sean óptimas ya que cada nodo no 've' toda la red.

c) **Verdadera**.

d) **Verdadera**. Un router sin estados no ha de recordar nada relativo a las conexiones que pasan por él.

e) **Verdadera**. Al ser orientados a conexión pueden ejercer *control de admisión*, cosa que no es factible en los protocolos de red no orientados a conexión. Además pueden aplicar todas las técnicas habituales en éstos.

S2.-

El descarte de paquetes.

S3.-

Tamaño de los paquetes: 164 bytes = 1312 bits

Por teoría de colas sabemos que el retardo viene dado por:

$$\text{retardo (en segundos)} = 1 / (pps_{max} - pps_{real})$$

Donde pps_{max} representa el número máximo de paquetes por segundo que pueden circular por la línea, y pps_{real} es el número de paquetes que están circulando.

En nuestro caso la línea es de 128 Kb/s, por lo que

$$pps_{max} = 128000 / 1312 = 97{,}56 \text{ pps}$$

La audioconferencia emite un paquete cada 40 ms, o sea 25 paquetes por segundo (1/0,04). Si la audioconferencia ha de compartir la línea con un tráfico de x paquetes por segundo generado por otras aplicaciones el tráfico real (pps_{real})será de x+25, por lo que para que el retardo no supere los 80 ms deberá cumplirse:

$$0{,}08 = 1 / (97{,}56 - x - 25)$$

Despejando obtenemos que x = 60,06 pps; con paquetes de 1312 bits esto equivale **a 78,8 Kb/s**, o una ocupación del **61,6%.** Por encima de esta ocupación no debería celebrarse la audioconferencia.

En el caso de una línea de 2048 Kb/s los cálculos serían como sigue:

$$pps_{max} = 2048000 / 1312 = 1560,98 \text{ pps}$$

$$0,08 = 1 / (1560,98 - x - 25)$$

En este caso el valor umbral es de 1523 pps, equivalente a **1998,8 Kb/s** o una ocupación del **97,6%**. Por tanto con la línea de 2 Mb/s se puede tener un nivel de ocupación mucho mayor manteniendo el mismo retardo que en la línea de baja velocidad.

3 EL NIVEL DE RED EN INTERNET

Autor: Rogelio Montañana

3.1 INTRODUCCIÓN

La Internet es un compendio de redes diferentes que comparten una pila de protocolos comunes. Cada una de estas redes es administrada por una entidad diferente: universidades, redes académicas nacionales, proveedores comerciales también llamados ISPs (Internet Service Providers), operadores, empresas multinacionales, etc. Como consecuencia de esto las políticas de uso son muy variadas.

Técnicamente a nivel de red la Internet puede definirse como un conjunto de redes o *sistemas autónomos*[1] conectados entre sí que utilizan el protocolo de red IP.

IP es una red de datagramas, no orientada a conexión, con servicio 'best effort', es decir no ofrece calidad de servicio o QoS (Quality of Service)[2]. La entrega de los paquetes no está garantizada ya que en momentos de congestión éstos pueden ser descartados sin previo aviso por los routers que se encuentren en el trayecto.

3.2 EL DATAGRAMA IP

En una red IP absolutamente toda la información viaja en datagramas IP. Esto incluye tanto la intercambiada a nivel de transporte por TCP y UDP como cualquier información de control que tenga que intercambiarse, por ejemplo para routing dinámico, mensajes de error, etc.

El datagrama IP tiene dos partes: cabecera y texto; la cabecera tiene una parte fija de 20 bytes y una opcional de entre 0 y 40 bytes. La longitud total de la cabecera siempre es múltiplo de 4; esto garantiza un proceso eficiente por parte de equipos (hosts o routers) cuya arquitectura optimiza el acceso a direcciones de memoria que estén en frontera de 32 bits. La estructura de la cabecera IP es la que se muestra en la tabla 3.1.

Campo	Longitud (bits)
Versión	4
IHL (Internet Header Length)	4
DS (Differentiated Services)	8
Longitud total	16
Identificación	16
Reservado	1
DF (Don't Fragment)	1
MF (More Fragments)	1
Fragment offset	13
TTL (Time To Live)	8
Protocolo (de transporte)	8
Checksum (de cabecera)	16
Dirección de origen	32
Dirección de destino	32
Opciones	0-320

Tabla 3.1.- Estructura de la cabecera de un datagrama IP

El campo **versión** permite que coexistan en la misma red sin ambigüedad paquetes de distintas versiones de IP; la versión actualmente utilizada (que corrresponde a la estructura de datagrama que estamos viendo) es la 4. Como veremos mas tarde, se empieza a extender el uso de una nueva versión (la 6) con una estructura de datagrama diferente.

[1] Más tarde definiremos el significado del término *sistema autónomo*.

[2] Esta afirmación es incorrecta estrictamente hablando, ya que recientemente se han incorporado una serie de mejoras y modificaciones que permiten disponer de Calidad de Servicio en una red IP.

El campo **IHL** especifica la longitud de la cabecera, ya que ésta puede variar debido a la presencia de campos opcionales. Se especifica en palabras de 32 bits. La longitud mínima es 5 y la máxima 15, que equivale a 40 bytes de información opcional. La longitud de la cabecera siempre ha de ser un número entero de palabras de 32 bits, por lo que si la longitud de los campos opcionales no es un múltiplo exacto de 32 bits se añade un relleno al final de la cabecera.

El campo **Differentiated Services** ha sustituido recientemente al antes denominado tipo de servicio. Su finalidad es implementar Calidad de Servicio en redes IP mediante la arquitectura denominada Servicios Diferenciados o Diffserv. La descripción de este campo y su funcionamiento la haremos más adelante al hablar de Calidad de Servicio.

El campo **longitud total** especifica la longitud del datagrama completo (cabecera incluida) en bytes. El valor máximo es 65535 bytes. Este campo sirve para saber donde termina el datagrama información que es realmente necesaria sólo en muy pocos casos, ya que en la mayoría de situaciones puede deducirse a partir de la información que posee el nivel de enlace. Por ejemplo en PPP, ATM o LANs no Ethernet el nivel de enlace sabe o puede deducir fácilmente la longitud del datagrama; el único caso en Ethernet donde es preciso indicar la longitud del datagrama es cuando se utiliza encapsulado DIX (donde el campo longitud se ha reemplazado por el campo Ethertype) y se envía una trama que no llega a 64 bytes; en este caso la trama contendrá relleno y la única forma de averiguar donde empieza éste es a partir de la longitud total indicada en la cabecera IP.

Los cuatro campos siguientes (Identificación, DF, MF y Fragment Offset) están relacionados con la fragmentación de datagramas que explicaremos más adelante.

El campo **TTL** (Time To Live) sirve para descartar un datagrama cuando ha dado un número excesivo de saltos o ha pasado un tiempo excesivo viajando por la red y es presumiblemente inútil. Se trata de un contador regresivo que indica el tiempo de vida restante del datagrama medido en segundos, de forma que si llega a valer cero el datagrama debe ser descartado. Además cada router por el que pasa dicho datagrama está obligado a restar uno del TTL, independientemente del tiempo que tarde en reenviarlo. Esto evita que por algún problema en las rutas se produzcan bucles y un datagrama permanezca 'flotando' indefinidamente en la red. Para medir el tiempo que un datagrama emplea en pasar de un router a otro sería preciso que los relojes de ambos estuvieran sincronizados; como esto no siempre ocurre en la práctica el tiempo en tránsito se ignora y el TTL se convierte simplemente en un contador de saltos; en muchas implementaciones ni siquiera cuando el datagrama esta mas de un segundo esperando en un router se disminuye por ello el valor del TTL, aunque esto ocurre raramente. En el caso particular de producirse fragmentación el host receptor sí puede retener datagramas durante varios segundos, mientras espera a recibir todos los fragmentos; en este caso el host sí disminuirá el valor del TTL en uno por cada segundo de espera, pudiendo llegar a descartar datagramas por este motivo, como veremos más adelante. Es habitual utilizar para el TTL los valores 64 o 255. El comando de prueba ping permite fijar el valor inicial del TTL en los datagramas de prueba enviados; por ejemplo en los sistemas operativos windows se utiliza para este fin la opción *–i* del comando ping.

El campo **protocolo** especifica a que protocolo del nivel de transporte corresponde el datagrama. La tabla de protocolos válidos y sus correspondientes números son controlados por el IANA (Internet Assigned Number Authority) y pueden consultarse en su web, www.iana.org/numbers.html. La tabla 3.2 muestra a título de ejemplo algunos de los posibles valores de este campo.

Valor	Protocolo	Descripción
0		Reservado
1	ICMP	Internet Control Message Protocol
2	IGMP	Internet Group Management Protocol
3	GGP	Gateway-to-Gateway Protocol
4	IP	IP en IP (encapsulado)
5	ST	Stream
6	TCP	Transmission Control Protocol
8	EGP	Exterior Gateway Protocol
17	UDP	User Datagram Protocol
29	ISO-TP4	ISO Transport Protocol Clase 4
38	IDRP-CMTP	IDRP Control Message Transport Protocol
80	ISO-IP	ISO Internet Protocol (CLNP)
88	IGRP	Interior Gateway Routing Protocol (Cisco)
89	OSPF	Open Shortest Path First
255		Reservado

Tabla 3.2.- Ejemplo de valores y significados del campo protocolo en un datagrama

Merece la pena llamar la atención sobre el valor 4 del campo protocolo, que está reservado para el uso de IP para transportar IP, es decir al encapsulado de un datagrama IP dentro de otro. Esta técnica permite realizar un encaminamiento desde el origen de los paquetes encapsulando el datagrama en otro dirigido al nodo intermedio por el que queremos pasar. Esto es como si para enviar una carta a Sevilla vía Madrid metiéramos el sobre dirigido a Sevilla en otro que lleve una dirección de Madrid; la carta llegaría a Madrid y de allí se enviaría a Sevilla.

El campo **checksum** sirve para detectar errores producidos en la cabecera del datagrama; no es un CRC sino el complemento a uno en 16 bits de la suma complemento a uno de toda la cabecera tomada en campos de 16 bits (incluidos los campos opcionales si los hay). Para el cálculo el campo checksum se pone a sí mismo a ceros. El checksum permite salvaguardar al datagrama de una alteración en alguno de los campos de la cabecera que pudiera producirse por ejemplo por un problema hardware en un router. Obsérvese que el checksum solo cubre la cabecera del datagrama, no los datos. El campo checksum se ha de recalcular en cada salto, ya que al menos el TTL cambia. En routers con mucho tráfico el recálculo del checksum supone un inconveniente desde el punto de vista de rendimiento.

Los campos **dirección de origen** y **dirección de destino** corresponden a direcciones IP de cuatro bytes según el formato que veremos luego.

Los campos opcionales de la cabecera no siempre están soportados en los routers y se utilizan muy raramente. Cada campo opcional está compuesto por una etiqueta, seguida opcionalmente de información adicional. Los más importantes son los siguientes:

- *Record route (registrar la ruta)*: Esta opción solicita a los routers por los que pasa que anoten su dirección en la cabecera del datagrama, con lo que al final del trayecto se dispone de una traza de la ruta seguida para fines de prueba o diagnóstico de problemas. Esto es como si cada router estampara su sello en el datagrama antes de reenviarlo. El máximo de direcciones que puede registrarse es de 9, ya que si se registraran 10 se ocuparían los 40 bytes de opciones y no quedaría sitio para la propia opción record route. El comando de prueba ping permite solicitar el uso de este campo para registrar la ruta; por ejemplo en los sistemas operativos windows se utiliza para este fin la opción *–r* del comando ping.

- *Timestamp (marca de tiempos)*: esta opción actúa de manera similar a record route, pero el router en lugar de anotar su dirección anota el instante (tiempo desde la medianoche en milisegundos) en el que el datagrama pasa por él; es un campo entero de 32 bits. También puede acompañarse el timestamp de la dirección IP del router; en este caso cada anotación ocupa ocho bytes y solo pueden producirse cuatro ya que la quinta superaría el tamaño máximo del campo opciones (la propia indicación de que se trata de un campo timestamp ya ocupa cuatro bytes). El

comando de prueba ping permite solicitar el uso de este campo para registrar la ruta y hora; por ejemplo en los sistemas operativos windows se utiliza para este fin la opción *-s* del comando ping.

- *Source routing (encaminamiento desde el origen)*: permite al emisor especificar la ruta que debe seguir el datagrama hasta llegar a su destino. Existen dos variantes: *strict source routing (encaminamiento estricto desde el origen)* especifica la ruta exacta salto a salto, de modo que si en algún caso la ruta marcada no es factible se producirá un error (una ruta no es factible cuando el router siguiente en la secuencia no es accesible directamente, es decir a nivel de enlace, desde el router actual). El funcionamiento de esta opción es equivalente al de los puentes con encaminamiento desde el origen. La opción *loose source routing (encaminamiento aproximado desde el origen)* indica una serie de routers por los que debe pasar un datagrama, pero deja la posibilidad de pasar por otros no especificados para llegar al siguiente router marcado en la secuencia. La limitación en la longitud de las opciones impone también aquí un límite máximo en el número de saltos que pueden especificarse en estas opciones, por lo que a menudo se utiliza en su lugar la técnica antes descrita de encapsulado IP en IP. Ambas variantes actúan también como record route, pues la secuencia no se va borrando a medida que se utiliza con lo que al final del trayecto el campo opciones contiene la ruta que se ha utilizado. El comando de prueba ping permite solicitar el uso de estos campo para especificar la ruta; por ejemplo en los sistemas operativos windows se utilizan las opciones *–j* y *–k* para indicar loose source routing y strict source routing, respectivamente.

El uso de los campos opcionales de la cabecera IP tiene generalmente problemas de rendimiento, ya que las implementaciones de los routers optimizan el código para las situaciones normales, es decir para datagramas sin campos opcionales. Aun en el caso de que las opciones estén implementadas lo harán generalmente de forma poco eficiente, ya que en el diseño del equipo no se ha hecho énfasis en su optimización. Por tanto solo deben utilizarse en situaciones de prueba o diagnóstico de errores, nunca en entornos en producción.

3.2.1 Fragmentación

El tamaño de un datagrama IP se especifica en un campo de dos bytes en la cabecera, por lo que su valor máximo es de 65535 bytes, pero muy pocos protocolos o tecnologías a nivel de enlace admiten enviar tramas de semejante tamaño. Normalmente el nivel de enlace no fragmenta[3], por lo que tendrá que ser IP el que adapte el tamaño de los datagramas para que quepan en las tramas del nivel de enlace; por tanto en la práctica el tamaño máximo del datagrama viene determinado por el tamaño máximo de trama característico de la red utilizada. Este tamaño máximo de datagrama se conoce como MTU (Maximum Transfer Unit). La tabla 3.3 muestra algunos ejemplos de valores de MTU característicos de las redes más habituales.

[3]Existen algunas excepciones a esto, siendo las más conocidas X.25 y ATM. X.25 utiliza normalmente paquetes de 128 bytes y ATM emplea celdas de 48 bytes. En ambos casos el nivel de 'enlace' (X.25 o ATM) dispone de mecanismos de fragmentación propios de forma que el tamaño que IP percibe es mayor (hasta 1600 bytes en el caso de X.25 y hasta 9180 en el de ATM).

Protocolo a nivel de enlace	MTU (bytes)
PPP (valor por defecto)	1500
PPP (bajo retardo)	296
SLIP	1006 (límite original)
X.25	1600 (RFC 1356)
Frame relay	1600 normalmente (depende de la red)
SMDS	9235
Ethernet DIX	1500
Ethernet LLC-SNAP	1492
IEEE 802.4/802.2	8166
Token Ring 16 Mb/s	17940 (token holding time 8 ms)
Token Ring 4 Mb/s	4440 (token holding time 8 ms)
FDDI	4352
Hyperchannel	65535
Classical IP over ATM	9180

Tabla 3.3.- Valor de MTU para los protocolos más comunes a nivel de enlace.

Podemos distinguir dos situaciones en las que se produce fragmentación. La primera, que podemos denominar *fragmentación en ruta*, se produce cuando un datagrama es creado por un host en una red con un valor determinado de MTU y en su camino hacia el host de destino ha de pasar por otra red con una MTU menor. Por ejemplo un datagrama creado en una Token Ring con MTU de 4440 bytes dirigido a una Ethernet con MTU de 1500 bytes. En estos casos el router que hace la transición a la red de MTU menor ha de fragmentar los datagramas para que no excedan el tamaño de la nueva red.

La segunda, que podemos llamar *fragmentación en origen*, se produce como consecuencia del diseño de la aplicación. Por ejemplo muchas implementaciones de NFS generan datagramas de 8 Kbytes de datos (8212 bytes con la cabecera IP). Un host en una red Ethernet que utilice NFS tendrá que fragmentar cada datagrama en seis fragmentos antes de enviarlo, aun cuando el host de origen y destino se encuentren ambos en el mismo concentrador de Ethernet. Como podemos ver en la tabla 3.3 solo una minoría de las redes existente soporta datagramas NFS sin fragmentar.

Existe una fuerte polémica sobre la conveniencia de que realice la fragmentación en ruta, ya que esto perjudica seriamente la eficiencia de los routers por la carga de CPU que supone y la de la red ya que la probabilidad de perder algún framento es mayor y en ese caso es preciso reenviar todos los fragmentos. En general se considera preferible realizar la fragmentación en origen siempre que esto sea posible. Una forma fácil de conseguirlo es utilizar un MTU de 1500, valor que es soportado por la mayoría de las redes existentes, con lo que se minimizan los casos en que los routers han de recurrir a la fragmentación. Otra forma de evitar la fragmentación es emplear la técnica de descubrimiento de la MTU del trayecto que describiremos más tarde.

La fragmentación se hace cortando la parte de datos del datagrama en trozos tales que cada fragmento con su cabecera quepa en la MTU de la nueva red (redondeado por abajo para que la cantidad de datos sea múltiplo de ocho bytes). Todos los campos de la cabecera del datagrama original se replican en los fragmentos excepto aquellos que se emplean para distinguir los fragmentos y que describiremos a continuación. Una vez fragmentado un datagrama no se reensambla hasta que llegue al host de destino, aun cuando en el trayecto pase a redes que admitan una MTU mayor. Los estándares Internet recomiendan que todas las redes que soporten TCP/IP tengan una MTU de al menos 576 bytes y esta condición la cumplen la mayoría de las redes, aunque no todas. La MTU mínima imprescindible para funcionar en TCP/IP es de 68 bytes, valor que corresponde a 60 bytes de cabecera (el máximo con todos los campos opcionales) y 8 bytes de datos, que es el fragmento mínimo de datos que puede hacerse.

El campo **Identificación** de la cabecera IP lo usa el emisor para marcar en origen cada datagrama emitido; de esta forma en caso de que se produzca fragmentación el receptor podrá reconocer las partes que corresponden al mismo datagrama original, ya que todas irán acompañadas de la misma identificación.

El bit **DF** (Don't Fragment) cuando está a 1 indica a los routers que este datagrama no debe fragmentarse. Normalmente esto se hace por uno de los dos motivos siguientes:

a) El receptor no está capacitado para reensamblar los fragmentos. Por ejemplo cuando un ordenador arranca su sistema operativo a través de la red normalmente el software que tiene para iniciar el proceso es muy sencillo y no soporta el reensamblado. En este caso el ordenador solicitará que el ejecutable correspondiente se le envíe como un único datagrama. Evidentemente para que esto sea posible será necesario que el trayecto completo por el que ha de transitar ese datagrama soporte el tamaño de MTU correspondiente.

b) Cuando se aplica la técnica de descubrimiento de la MTU del trayecto o 'path MTU discovery' para averiguar la MTU de una ruta. Esta técnica consiste en que el host de origen envía un datagrama del tamaño máximo al host de destino con el bit DF puesto; si el datagrama no puede pasar en algún punto del trayecto el router correspondiente genera un mensaje de error que es devuelto al host emisor; entonces este envía otro datagrama mas pequeño, también con el bit DF a 1 y así sucesivamente, hasta que consigue que algún datagrama pase sin fragmentar al host de destino; tanteando de esta forma consigue averiguar cual es la máxima MTU de la ruta y a partir de ahí utiliza este como valor máximo para todos los datagramas[4]. Para acelerar el proceso de tanteo algunos routers incorporan en los mensajes de error información sobre la MTU máxima que puede admitir la red que ha provocado el rechazo.

El comando de prueba ping permite especificar la no fragmentación. Por ejemplo en los sistemas operativos windows se utiliza para este fin la opción *–f*. Esto combinado con la especificación de un dterminado tamaño de paquete (opción *–l* en windows) permite hacer sondeos manuales de la MTU de un trayecto, lo cual resulta muy útil en la resolución de problemas o en pruebas de rendimiento.

El bit **MF** (More Fragments) puesto a 1 especifica que este datagrama es realmente un fragmento de un datagrama mayor, y que no es el último. Si está a 0 indica que este es el último fragmento o bien que el datagrama no ha sido fragmentado.

El campo **Fragment offset** sirve para indicar, en el caso de que el datagrama sea un fragmento, en que posición del datagrama original se sitúan los datos que contiene. Los cortes siempre se realizan en múltiplo de 8 bytes, que es la unidad elemental de fragmentación, por lo que el Fragment offset cuenta los bytes en grupos de 8. Gracias a eso este contador requiere únicamente 13 bits en vez de los 16 que harían falta si contara bytes (2^{13}=8192, 8192 x 8 = 65536). De los tres bits que se ganan dos se utilizan en los flags DF y MF y el tercero no se utiliza.

Los fragmentos de un datagrama pueden llegar desordenados a su destino; el receptor podrá identificarlos gracias al campo Identificación. La longitud total del datagrama puede calcularla cuando recibe el último fragmento (identificado por el bit MF a 0) a partir de los campos Longitud y Fragment offset; la longitud será *fragment_offset*8 + longitud*.

Cuando se fragmenta un datagrama el host receptor retiene en su buffer los fragmentos y los reensambla cuando los ha recibido todos. Mientras mantiene retenido un fragmento el host va restando cada segundo una unidad al campo TTL. Cuando el valor de TTL es igual a cero descarta el fragmento. Si alguno de los fragmentos de un datagrama se pierde el resto terminarán desapareciendo a medida que agoten su TTL. No existe ningún mecanismo en IP que contemple el reenvío de datagramas o de fragmentos. Si el protocolo utilizado a nivel superior contempla reenvío de paquetes perdidos (por ejemplo TCP a nivel de transporte) se provocará el reenvío del datagrama correspondiente. Normalmente el segundo envío se verá sometido a la misma fragmentación que el primero, pero el segundo no podrá en ningún caso aprovechar fragmentos residuales que pudiera haber en el host receptor correspondientes al primer envío, ya que desde el punto de vista del nivel IP se trata de dos datagramas distintos e independientes que reciben identificaciones diferentes.

Veamos un ejemplo de cómo se produciría la fragmentación en ruta de un datagrama. Supongamos que generamos en una red Token Ring un datagrama de 4000 bytes de datos (es decir 4000 bytes más la

[4] Esta técnica supone que la ruta no se modificará durante la sesión, cosa que no siempre es cierta. Algunas implementaciones revisan la MTU del trayecto periódicamente durante la sesión para asegurarse de que se está utilizando el valor óptimo en cada momento (ver el ejercicio número 14).

cabecera IP) que ha de pasar por una red Ethernet DIX (MTU de 1500 bytes); el resultado de la fragmentación será el siguiente:

	Campos de cabecera en datagrama IP					**Datos**
Datagrama Original	Id = X	L = 4020	DF = 0	MF = 0	Offset = 0	ABCDEF GHIJKL MNOP
Fragmento 1	Id = X	L = 1500	DF = 0	MF = 1	Offset = 0	ABCDEF
Fragmento 2	Id = X	L = 1500	DF = 0	MF = 1	Offset = 185	GHIJKL
Fragmento 3	Id = X	L = 1060	DF = 0	MF = 0	Offset = 370	MNOP

El primer y segundo fragmentos contendrán 1480 bytes de datos y 20 de cabecera; el tercero tendrá 1040 de datos y 20 de cabecera.

Puede suceder que un datagrama que ha sido fragmentado en un punto determinado de la ruta tenga que ser fragmentado de nuevo más tarde porque pase a otra red de MTU aun menor. Supongamos que los fragmentos 2 y 3 anteriores pasan después por una red PPP con bajo retardo (MTU de 296 bytes); el resultado será el siguiente:

	Campos de cabecera en datagrama IP					**Datos**
Fragmento 2	Id = X	L = 1500	DF = 0	MF = 1	Offset = 185	GHIJKL
Fragmento 2a	Id = X	L = 292	DF = 0	MF = 1	Offset = 185	G
Fragmento 2b	Id = X	L = 292	DF = 0	MF = 1	Offset = 219	H
Fragmento 2c	Id = X	L = 292	DF = 0	MF = 1	Offset = 253	I
Fragmento 2d	Id = X	L = 292	DF = 0	MF = 1	Offset = 287	J
Fragmento 2e	Id = X	L = 292	DF = 0	MF = 1	Offset = 321	K
Fragmento 2f	Id = X	L = 140	DF = 0	MF = 1	Offset = 355	L

	Campos de cabecera en datagrama IP					**Datos**
Fragmento 3	Id = X	L = 1060	DF = 0	MF = 0	Offset = 370	MNOP
Fragmento 3a	Id = X	L = 292	DF = 0	MF = 1	Offset = 370	M
Fragmento 3b	Id = X	L = 292	DF = 0	MF = 1	Offset = 404	N
Fragmento 3c	Id = X	L = 292	DF = 0	MF = 1	Offset = 438	O
Fragmento 3d	Id = X	L = 244	DF = 0	MF = 0	Offset = 472	P

Aquí cada fragmento (excepto el último de cada grupo) lleva 272 bytes de datos y 20 de cabecera. Aunque la MTU posible es de 296 bytes los datagramas generados son de 292 bytes porque la parte de datos de los fragmentos siempre debe ser múltiplo de 8 bytes. Obsérvese que después de una fragmentación múltiple solo el último fragmento del último grupo (el 3d en nuestro ejemplo) tiene puesto a 0 el bit MF.

Es interesante analizar el comportamiento de los campos opcionales cuando se realiza fragmentación. Por ejemplo el campo opcional Strict Source Route se replica en todos los fragmentos ya que de lo contrario no puede asegurarse que todos vayan por la ruta indicada. Lo mismo ocurre con el campo Loose Source Routing. En el caso de los campos Record Route y Timestamp los campos opcionales solo se copian en el primer fragmento, lo cual supone que los demás pueden ir por otras rutas; pero se supone que en estas opciones no se quiere obligar a utilizar un trayecto determinado, sino simplemente se pretende obtener información sobre la ruta utilizada; en estas condiciones se ha considerado razonable suministrar únicamente la información relativa al primer fragmento de la secuencia, evitando así el costo que supondría ejecutar la opción para cada fragmento, lo cual en la mayoría de los casos daría solo información redundante.

3.3 DIRECCIONES IP

Cada interfaz de red de cada nodo (host o router) en una red IP se identifica mediante al menos una dirección única de 32 bits. Las direcciones IP se suelen representar por cuatro números decimales separados por puntos, que equivalen al valor de cada uno de los cuatro bytes que componen la dirección. Por ejemplo una dirección IP válida sería 147.156.23.208.

Si un nodo dispone de varias interfaces físicas (cosa habitual en los routers) cada una de ellas deberá tener necesariamente una dirección IP distinta si se desea que sea accesible de forma diferenciada para este protocolo. Es posible también y en algunas situaciones resulta útil, definir varias direcciones IP asociadas a una misma interfaz física.

Como ocurre en la mayoría de las redes las direcciones IP tienen una estructura jerárquica. Una parte de la dirección corresponde a la red, y la otra al host dentro de la red. Cuando un router recibe un datagrama por una de sus interfaces compara la parte de red de la dirección con las entradas contenidas en sus tablas (que normalmente sólo contienen direcciones de red, no de host) y envía el datagrama por la interfaz correspondiente.

En el diseño inicial de la Internet se reservaron los ocho primeros bits para la red, dejando los 24 restantes para el host; se creía que con 254 redes habría suficiente para la red experimental de un proyecto de investigación del Departamento de Defensa americano. Pronto se vio que esto resultaba insuficiente, por lo que se reorganizó el espacio de direcciones reservando unos rangos para definir redes más pequeñas. El resultado de esa reorganización es lo que hoy conocemos como las redes clase A, B y C:

- Una red de clase A se caracteriza por tener a 0 el primer bit de dirección; el campo red ocupa los 7 bits siguientes y el campo host los últimos 24. Puede haber hasta 128 redes de clase A con 16777216 direcciones cada una.

- Una red de clase B tiene el primer bit a 1 y el segundo a 0; el campo red ocupa los 14 bits siguientes, y el campo host los 16 últimos. Puede haber 16384 redes clase B con 65536 direcciones cada una.

- Una red clase C tiene los primeros tres bits a 110; el campo red ocupa los siguientes 21 bits, y el campo host los 8 últimos. Puede haber hasta 2097152 redes clase C con 256 direcciones cada una.

Para indicar qué parte de la dirección corrresponde a la red y qué parte al host se suele utilizar una notación denominada **máscara**, consistente en poner a 1 los bits que corresponden a la parte de red y a 0 los que corresponden a la parte host. Así por ejemplo diremos que una red clase A tiene una máscara 255.0.0.0, lo cual equivale a decir que los ocho primeros bits especifican la red y los 24 restantes el host. Análogamente decimos que una red clase B tiene una máscara 255.255.0.0 y una clase C una máscara 255.255.255.0.

Las máscaras permiten extraer de forma sencilla la parte de red o de host de una dirección. Por ejemplo un router que ha de enviar un datagrama puede realizar un AND entre la dirección de destino y la máscara correspondiente, con lo que extraerá la parte de red de la dirección.

Existen además direcciones (no redes) clase D cuyos primeros cuatro bits valen 1110. Las direcciones clase D se utilizan para definir grupos multicast. El grupo queda definido por los 28 bits siguientes. Puede haber hasta 268435456 direcciones multicast en Internet. Las direcciones clase D nunca puede aparecer como direcciones de origen de un datagrama.

Por último, la clase E, que corresponde al valor 1111 en los primeros cuatro bits, no se utiliza de momento y está reservada para usos futuros.

De los valores de los primeros bits de cada una de las clases antes mencionadas se puede deducir el rango de direcciones que corresponde a cada una de ellas. Así pues, en la práctica es fácil saber a que clase pertenece una dirección determinada sin más que observar el primer byte de su dirección. La tabla 3.4 resume la información esencial sobre las clases de redes de Internet.

Clase	Primeros bits	Bits red/host	Núm. Redes	Núm. Direcc.	Máscara	Rango de direcciones
A	0---	7/24	128	16777216	255.0.0.0	0.0.0.0 - 127.255.255.255
B	10--	14/16	16384	65536	255.255.0.0	128.0.0.0 -191.255.255.255
C	110-	21/8	2097152	256	255.255.255.0	192.0.0.0 -223.255.255.255
D	1110			268435456		224.0.0.0 -239.255.255.255
E	1111					240.0.0.0 -255.255.255.255

Tabla 3.4 - Clases de direcciones Internet y sus principales características

La asignación de direcciones válidas de Internet la realizan los NICs (NIC = Network Information Center). Al principio había un NIC para toda la Internet pero luego se crearon NICs regionales. Actualmente existen los tres siguientes:

- América: www.internic net
- Europa: www.ripe.net
- Asia y Pacífico: www.apnic net

Estos NICs asignan direcciones IP a los proveedores de Internet y a las grandes organizaciones. Los proveedores a su vez asignan direcciones a las organizaciones de menor tamaño y a sus usuarios.

Existen unas reglas y convenios que asignan significados especiales a determinadas direcciones IP que es importante conocer:

1. La dirección **255.255.255.255** se utiliza para indicar broadcast en la propia red. La utilizaría por ejemplo como dirección de destino un host que está arrancando en una red local y que para averiguar la red en la que se encuentra y su propia dirección IP necesita localizar un servidor que le de los parámetros de configuración básicos. Solo se puede utilizar como dirección de destino, nunca como dirección de origen.

2. La dirección **0.0.0.0** identifica al host actual. La utilizaría el host del supuesto anterior como dirección de origen de sus datagramas. Solo se puede utilizar como dirección de origen, no de destino.

3. Las direcciones con el **campo host todo a ceros** identifican redes y por tanto no se utilizan para ningún host. Se emplean para especificar rutas y nunca deberían aparecer somo direcciones de origen o destino de un datagrama. Por ejemplo la dirección 147.156.0.0 identifica la red clase B que pertenece a la Universidad de Valencia.

4. La dirección con el **campo host todo a unos** se utiliza como dirección broadcast dentro de la red y por tanto no se utiliza para ningún host. Solo puede ser una dirección de destino. Por ejemplo para enviar un mensaje broadcast a la red de la Universidad de Valencia utilizaríamos la dirección 147.156.255.255.

5. La dirección con el **campo red todo a ceros** identifica a un host en la propia red, cualquiera que esta sea; por ejemplo si queremos enviar un datagrama al primer host (0.1) de una red clase B podemos utilizar la dirección 0.0.0.1. Esto permite enviar un datagrama a un host en una red sin saber el número de ésta, aunque es preciso conocer si es clase A, B o C para saber que tamaño tiene la parte red de la dirección.

6. La dirección **127.0.0.1** se utiliza para pruebas loopback; todas las implementaciones de IP devuelven a la dirección de origen los datagramas enviados a esta dirección sin intentar enviarlos a ninguna parte[5].

7. Las redes **127.0.0.0, 128.0.0.0, 191.255.0.0, 192.0.0.0** y el rango de **240.0.0.0** en adelante (clase E) están reservados y no deben utilizarse.

8. Las redes **10.0.0.0** (clase A), **172.16.0.0 a 172.31.0.0** (clase B) y **192.168.0.0 a 192.168.255.0** (clase C) están reservadas para redes privadas ('intranets') por el RFC 1918; estos números no se asignan a ninguna dirección válida en Internet y por tanto pueden utilizarse para construir redes, por ejemplo detrás de un cortafuego, sin riesgo de entrar en conflicto de acceso con redes válidas de la Internet.

Como consecuencia de las reglas 3 y 4 antes mencionadas siempre hay dos direcciones inútiles en una red, la primera y la última. Por ejemplo, si tenemos la red 200.200.200.0 (clase C) tendremos que reservar la dirección 200.200.200.0 para denotar la red misma, y la dirección 200.200.200.255 para envíos broadcast a toda la red; dispondremos pues de 254 direcciones para hosts, no de 256.

3.3.1 Subredes

Supongamos que una empresa dispone de varias oficinas, cada una con una red local, todas ellas interconectadas entre sí, y que desea unirlas mediante el protocolo TCP/IP; una de dichas oficinas (la principal) dispone además de una conexión a Internet. Supongamos también cada oficina tiene suficiente con 254 direcciones de hosts. En principio sería posible asignar una red clase C diferente para cada oficina, pero esto supone solicitar al NIC una red para cada oficina que se conecte, y al ser cada una independiente de las demás la gestión se complica; por ejemplo sería preciso anunciar en Internet la ruta para cada nueva red para que la oficina correspondiente fuera accesible. Dado que cada red sería en principio independiente de las demás no habría una forma sencilla de agrupar las redes de la organización.

Hay un mecanismo que permite dividir una red IP en trozos o subredes, de forma que la empresa de nuestro ejemplo podría solicitar una clase B y asignar fragmentos de dicha red a cada oficina a medida que se fueran incorporando a la red. Esto equivale a crear un nivel jerárquico intermedio entre la red y el host, permitiendo así grados variables de agrupación según el nivel en el que nos encontramos. Supongamos que a la empresa de nuestro ejemplo se le asigna una red clase B, la 156.134.0.0; de los 16 bits que en principio corresponden al host podría reservar los primeros 8 para la subred y dejar los 8 siguientes para el host, con lo que dispondrá de 256 subredes de 256 direcciones cada una. Desde fuera la red de la empresa seguirá siendo 156.134.0.0, ya que la estructura de subred no será visible.

[5] Según el estándar todas las direcciones de la red 127.0.0.0 deberían ser loopback, pero en muchos casos esto sólo funciona para la primera dirección (127.0.0.1).

Las subredes se añadieron a la Internet en 1982; con ello se consiguió una mayor flexibilidad en el reparto de direcciones dentro de una red.

Para dividir la red en subredes se define una nueva *máscara*. Como siempre los bits a 1 de la máscara identifican la parte de red (en este caso la parte de red-subred) y los bits a cero corresponden al host. Por ejemplo, la máscara 255.255.255.0 aplicada sobre una red clase B la divide en 256 subredes de 256 direcciones cada una, pues tiene puestos a 1 los primeros 24 bits; en cierto modo podríamos decir que esta máscara convierte una red clase B en 256 subredes clase C. Se pueden hacer divisiones que no correspondan a bytes enteros, por ejemplo la máscara 255.255.252.0 hace subredes mas grandes, reserva los primeros 6 bits para la subred y deja 10 para el host, con lo que podría haber hasta 64 subredes con 1024 direcciones cada una.

Cuando se crean subredes hay dos direcciones en cada subred que quedan automáticamente reservadas: las que corresponden al campo host todo a ceros y todo a unos; estas se emplean para designar la subred y para broadcast dentro de la subred, respectivamente. Así si la red 156.134.0.0 se subdivide con la máscara 255.255.255.0 se crean 256 subredes del tipo *156.134.subred.host*, cada una con 256 direcciones. En cada subred hay 254 direcciones aprovechables para hosts, ya que la primera dirección (*156.134.subred.0*) identifica a la subred y la última (*156.134.subred.255*) es la dirección broadcast de esa subred. Por tanto el número de hosts de una subred es siempre dos menos que el rango de direcciones que abarca. Una consecuencia de lo anterior es que resulta absurdo crear subredes con la máscara 255.255.255.254, ya que el campo host tendría un bit solamente y no quedaría ninguna dirección aprovechable para hosts.

Del mismo modo que los valores todo ceros o todo unos del campo host están reservados con un significado especial, los valores todo ceros y todo unos del campo subred (es decir la primera y la última subredes) también son especiales. El valor todo ceros se utiliza para representar la subred misma; por ejemplo si a la red 156.134.0.0 le aplicamos la máscara 255.255.255.0 la primera subred (campo subred todo a ceros) no debería utilizarse, pues resultaría ambiguo el significado de la dirección 156.134.0.0, que representaría tanto a dicha subred como a la red entera. Análogamente la última subred (campo subred todo a unos) tampoco debería utilizarse porque entonces la dirección 156.134.255.255 significaría tanto broadcast en dicha subred como en la red entera. Por consiguiente en el campo subred también se pierden siempre dos direcciones, y tampoco tendría sentido crear máscaras con el campo subred de un bit, como sería el caso de una máscara 255.255.128.0 en el caso de una red clase B.

Mientras que la restricción de las direcciones todo ceros o todo unos en el campo host se ha de respetar siempre, existen muchas situaciones en las que interesa aprovechar la subred toda a ceros o toda a unos, violando la segunda norma antes mencionada. Esta violación, permitida por muchas implementaciones, se conoce como **subnet-zero** y se adopta para aprovechar mejor el espacio de direcciones disponible; con subnet-zero es posible por ejemplo dividir una red clase B por la mitad en dos subredes mediante la máscara 255.255.128.0, cosa que no sería posible si no se permitiera esta pequeña 'infracción'.

En todos nuestros ejemplos la parte de subred de la máscara es contigua a la parte de red; en un principio se permitía crear subredes con máscaras no contiguas, por ejemplo dividir una clase B con la máscara 255.255.0.255; en este caso el host vendría especificado por el tercer byte y la subred por el cuarto. Dado que esta práctica solo resta claridad y hace menos eficiente el proceso de direcciones en los routers, actualmente la norma exige que la máscara sea contigua.

Para terminar de clarificar la creación de subredes y el uso de máscaras resumimos en la tabla 3.5 todas las posibles subredes y máscaras que se pueden utilizar con una red clase B. En el caso de una red clase C las posibles subredes y máscaras son las que se recogen en la tabla 3.6.

Bits de subred	Número de subredes	Nº subredes (subred cero)	Bits de host	Número de hosts	Máscara
0	0	0	16	65534	255.255.0.0
1	0	2	15	32766	255.255.128.0
2	2	4	14	16382	255.255.192.0
3	6	8	13	8190	255.255.224.0
4	14	16	12	4094	255.255.240.0
5	30	32	11	2046	255.255.248.0
6	62	64	10	1022	255.255.252.0
7	126	128	9	510	255.255.254.0
8	254	256	8	254	255.255.255.0
9	510	512	7	126	255.255.255.128
10	1022	1024	6	62	255.255.255.192
11	2046	2048	5	30	255.255.255.224
12	4094	4096	4	14	255.255.255.240
13	8190	8192	3	6	255.255.255.248
14	16382	16384	2	2	255.255.255.252
15	32766	32768	1	0	255.255.255.254
16	65534	65536	0	0	255.255.255.255

Tabla 3.5.- Subredes y máscaras que pueden definirse en una red clase B

Bits de subred	Número de subredes	Nº subredes (subred cero)	Bits de host	Número de hosts	Máscara
0	0	0	8	254	255.255.255.0
1	0	2	7	126	255.255.255.128
2	2	4	6	62	255.255.255.192
3	6	8	5	30	255.255.255.224
4	14	16	4	14	255.255.255.240
5	30	32	3	6	255.255.255.248
6	62	64	2	2	255.255.255.252
7	126	128	1	0	255.255.255.254
8	254	256	0	0	255.255.255.255

Tabla 3.6.- Subredes y máscaras que pueden definirse en una red clase C

La división en subredes no ha de hacerse necesariamente de forma homogénea en todo el espacio de direcciones, como hemos hecho hasta ahora. Supongamos que la empresa de nuestro ejemplo anterior tiene una serie de oficinas pequeñas que tienen bastante con subredes de 256 direcciones, pero otras mas grandes requieren un número mayor; podríamos partir la red en 256 subredes de 256 hosts y asignar varias de esas subredes a las oficinas mayores, pero esto complicaría la gestión y las tablas de rutas. La solución más adecuada en este caso sería dividir la red en subredes de diferentes tamaños y asignar a cada oficina una subred adecuada a sus necesidades. Asi en nuestro ejemplo se podría dividir la red 156.134.0.0 de la siguiente manera:

16 subredes de 256 direcciones:

Subred	Máscara	Subred/bits de máscara
156.134.0.0	255.255.255.0	156.134.0.0/24
156.134.1.0	255.255.255.0	156.134.1.0/24
156.134.2.0	255.255.255.0	156.134.2.0/24
156.134.3.0	255.255.255.0	156.134.3.0/24
156.134.4.0	255.255.255.0	156.134.4.0/24
156.134.5.0	255.255.255.0	156.134.5.0/24
156.134.6.0	255.255.255.0	156.134.6.0/24
156.134.7.0	255.255.255.0	156.134.7.0/24
156.134.8.0	255.255.255.0	156.134.8.0/24
156.134.9.0	255.255.255.0	156.134.9.0/24
156.134.10.0	255.255.255.0	156.134.10.0/24
156.134.11.0	255.255.255.0	156.134.11.0/24
156.134.12.0	255.255.255.0	156.134.12.0/24
156.134.13.0	255.255.255.0	156.134.13.0/24
156.134.14.0	255.255.255.0	156.134.14.0/24
156.134.15.0	255.255.255.0	156.134.15.0/24

16 subredes de 1024 direcciones

Subred	Máscara	Subred/bits de máscara
156.134.16.0	255.255.252.0	156.134.16.0/22
156.134.20.0	255.255.252.0	156.134.20.0/22
156.134.24.0	255.255.252.0	156.134.24.0/22
156.134.28.0	255.255.252.0	156.134.28.0/22
156.134.32.0	255.255.252.0	156.134.32.0/22
156.134.36.0	255.255.252.0	156.134.36.0/22
156.134.40.0	255.255.252.0	156.134.40.0/22
156.134.44.0	255.255.252.0	156.134.44.0/22
156.134.48.0	255.255.252.0	156.134.48.0/22
156.134.52.0	255.255.252.0	156.134.52.0/22
156.134.56.0	255.255.252.0	156.134.56.0/22
156.134.60.0	255.255.252.0	156.134.60.0/22
156.134.64.0	255.255.252.0	156.134.64.0/22
156.134.68.0	255.255.252.0	156.134.68.0/22
156.134.72.0	255.255.252.0	156.134.72.0/22
156.134.76.0	255.255.252.0	156.134.76.0/22

3 subredes de 4096 direcciones:

Subred	Máscara	Subred/bits de máscara
156.134.80.0	255.255.240.0	156.134.80.0/20
156.134.96.0	255.255.240.0	156.134.96.0/20
156.134.112.0	255.255.240.0	156.134.112.0/20

Y por último una subred de 32768 direcciones:

Subred	Máscara	Subred/bits de máscara
156.134.128.0	255.255.128.0	156.134.128.0/17

La técnica que hemos aplicado en el ejemplo anterior de dividir una red en subredes de diferentes tamaños se conoce comúnmente como **máscaras de tamaño variable**. Obsérvese que en este ejemplo hemos seguido la práctica de subnet-zero, es decir, se han considerado válidas subredes con el campo subred todo a ceros y todo a unos, de lo contrario no habría sido posible crear una subred con 32768 direcciones. Obsérvese en la columna de la derecha el uso de la notación *subred/bits de máscara* para representar las subredes; esta notación abreviada está sustituyendo en algunos casos a la tradicional y se emplea en la configuración de algunos equipos, en RFCs y otros documentos.

Cuando el campo subred ocupa bytes enteros es trivial la identificación de subredes. Las cosas se complican un poco más cuando no es así, por ejemplo en el caso anterior la primera subred de 1024 direcciones abarca un bloque de cuatro valores en el tercer byte, del 16 al 19; los bloques han de empezar en valores múltiplo de cuatro, si en vez de empezar en el 16 hubiéramos empezado en el 15 (15, 16, 17 y 18) no habría sido posible crear una subred pues la dirección 156.134.15.0 no tiene una máscara de 22 bits común con las otras tres; por consiguiente las subredes formadas con máscara de 22 bits (grupos de cuatro números) necesariamente han de empezar en valores múltiplos de cuatro (0, 4, 8, etc.). Análogamente puede deducirse que las subredes de máscara de 20 bits (4096 direcciones) han de empezar en valores múltiplos de 16, y así sucesivamente.

3.3.2 Superredes: Routing classless (CIDR)

El rápido crecimiento de la Internet está creando varios problemas, el más importante de los cuales es el agotamiento del espacio de direcciones IP. La causa de esto ha sido en parte la excesiva disparidad de tamaños entre las diferentes clases de redes. Hace ya mucho tiempo que han dejado de asignarse redes clase A, debido a su escaso número y tamaño excesivo. Las organizaciones tenían pues que elegir entre solicitar una clase B o una clase C. En muchos casos una clase B era excesiva, pero una C resultaba insuficiente, por lo que la mayoría de las organizaciones optaban por solicitar una clase B, aunque a menudo no necesitaban tantas direcciones. A la vista del rápido agotamiento de redes clase B debido a este motivo se pensó en crear grupos de clases C, de forma que las organizaciones pudieran optar por niveles intermedios entre las redes B y C, más adecuados a sus necesidades; por ejemplo una organización que necesite 2048 direcciones puede hoy en día solicitar un grupo de ocho redes clase C . De esta forma se reduce el problema de escasez de direcciones, pero se crea un problema nuevo: el crecimiento de las tablas de rutas. Antes, cuando a una organización que se conectaba a Internet se le asignaba una red esto suponía una nueva entrada en las tablas de rutas de Internet, pero al dar grupos de clases C se requiere una entrada diferente para cada red asignada. Esto habría provocado un crecimiento exponencial en las tablas de rutas de los routers que forman el ‘backbone’, cuyas capacidades se encuentran ya bastante cerca del límite de la tecnología.

El gran tamaño de las tablas de rutas de Internet se debe también al mecanismo de asignación de direcciones que se ha seguido, que durante mucho tiempo ha sido estrictamente cronológico. Al no haber una correspondencia entre la ubicación geográfica de una organización o el proveedor que la conecta a Internet y su rango de direcciones no es posible resumir o ‘sumarizar’ las tablas de rutas; la información se ha de incluir enumerando una a una todas las redes existentes. Esto se podría resolver con una organización jerárquica de direcciones de acuerdo con criterios geográficos, como ocurre por ejemplo en el direccionamiento de la red telefónica que identifica a cada país con prefijo, cada región dentro de un país con un subprefijo, y así sucesivamente.

Actualmente hay mas de 100.000 redes registradas en la Internet. Además del costo en memoria RAM que supone el mantener tablas extremadamente grandes en los routers, los algoritmos de búsqueda se complican y no funcionan adecuadamente con esas tablas, ya que fueron diseñados pensando en tablas con muchas menos entradas. A principios de los 90 el crecimiento de la Internet se estaba produciendo a un ritmo tal que el número de redes conectadas se duplicaba cada 9 meses, mientras que la tecnología sólo permite duplicar la capacidad y potencia de los routers cada 18 meses. En esta situación la explosión de las tablas de routing se estaba convirtiendo en un problema aún más grave que la escasez de direcciones. Según cálculos hechos por la IETF en 1993 de seguir produciéndose el crecimiento al mismo ritmo en el número de redes y rutas la Internet se habría colapsado hacia 1998.

Los dos problemas antes descritos, desperdicio del espacio de direcciones debido a la rigidez en la asignación de rangos (redes clase B o C) y crecimiento de las tablas de rutas, se resolvieron conjuntamente en 1993 con la adopción de un sistema denominado CIDR (Classless InterDomain Routing) descrito en los RFCs 1466, 1518 y 1519 que consiste en dos medidas complementarias.

La primera medida consiste en establecer una jerarquía en la asignación de direcciones. En vez de utilizar un criterio puramente cronológico, que desde le punto de vista geográfico o de topología de la red equivale a una asignación aleatoria, los rangos se asignan por continentes. Inicialmente se realizó la asignación de una parte del espacio de clase C de la siguiente manera:

Multi regional:	192.0.0.0 - 193.255.255.255
Europa:	194.0.0.0 - 195.255.255.255
Otros:	196.0.0.0 - 197.255.255.255
Noteamérica:	198.0.0.0 - 199.255.255.255
Centro y Sudamérica:	200.0.0.0 - 201.255.255.255
Anillo Pacífico:	202.0.0.0 - 203.255.255.255
Otros:	204.0.0.0 - 205.255.255.255
Otros:	206.0.0.0 - 207.255.255.255

Algunos de estos grupos se han ampliado posteriormente con nuevos rangos. A su vez cada proveedor Internet solicita rangos propios al NIC que le corresponde según el continente donde se encuentra. Con esta distribución regional de los números es en principio posible agrupar las entradas en las tablas de rutas; por ejemplo un router en Japón podría tener una sola entrada en sus tablas indicando que todos los paquetes dirigidos a las redes 194.0.0.0 hasta 195.255.255.0 se envíen a la interfaz por la cual accede a Europa, evitando así las 131.072 entradas que normalmente harían falta para este rango de direcciones.

Una consecuencia curiosa de la asignación de rangos de direcciones por proveedor es que si una empresa cambia de proveedor normalmente tendrá que 'devolver' al primero sus direcciones, y solicitar direcciones al nuevo proveedor; por supuesto tendrá que modificar la configuración de todas las máquinas que tuviera utilizando direcciones del primero.

Para que la sumarización de rutas (o agrupamiento de redes clase C) sea posible es preciso introducir una ligera modificación en el software de los routers, ya que en principio el software no considera el rango 194.0.0.0–195.255.255.255 como una sola red sino como 131.072 redes clase C. Para resolver este problema se ha extendido el concepto de subred en sentido contrario, es decir la máscara no solo puede crecer hacia la derecha para dividir una red en subredes sino que puede menguar hacia la izquierda para agrupar varias redes en una mayor (de ahí la denominación de ‘superredes’). Dicho de otra forma, la parte red de la dirección vendrá especificada por la longitud de la máscara únicamente, no teniendo ya ningún significado la clasificación tradicional en clases A, B y C de acuerdo con el valor de los primeros bits; solo se repeta dicho significado en el caso de las clases D (multicast) y E (reservado). Dicha supresión de las clases tradicionales es lo que da nombre a esta técnica conocida como enrutamiento entre dominios **sin clases** o CIDR (Classless InterDomain Routing).

La segunda medida adoptada por CIDR es en realidad un complemento de la anterior. Consiste sencillamente en dar a cada organización (bien directamente o a través de su proveedor correspondiente) la posibilidad de solicitar rangos de direcciones ajustado a sus necesidades previstas, dándole siempre un rango contiguo y que tenga una máscara de red común; por ejemplo un rango de 2048 direcciones se daría asignando los primeros 21 bits de la dirección y podría estar formado por ejemplo por el rango que va del 194.0.8.0 al 194.0.15.255.

Supongamos que la Universidad de Málaga solicita a su proveedor direcciones de red y justifica la previsión de tener 1200 hosts en un plazo razonable; como 1024 direcciones no serían suficientes su proveedor le asigna 2048 direcciones, equivalentes a ocho redes clase C. Supongamos que el proveedor tiene disponible el rango 195.100.0.0 a 195.100.255.255 y que ha ido asignando direcciones a sus clientes por orden cronológico, quedándole libre a partir de la 195.100.12.0; no es posible asignar a la Universidad de Málaga de la 195.100.12.0 a la 195.100.19.255 ya que aunque el rango es contiguo las direcciones no tienen una máscara común (los primeros 21 bits no son iguales). El primer rango de 2048 direcciones con máscara común es el 195.100.16.0-195.100.23.255, que es el que se deberá asignar. Las direcciones 195.100.12.0 a 195.100.15.255 quedarían libres para otros clientes con menores necesidades.

Con esta asignación de direcciones y sin utilizar CIDR el aspecto de la tabla de rutas para la Universidad de Málaga sería por ejemplo:

```
A 195.100.16.0/24 por 192.168.1.2
A 195.100.17.0/24 por 192.168.1.2
A 195.100.18.0/24 por 192.168.1.2
A 195.100.19.0/24 por 192.168.1.2
A 195.100.20.0/24 por 192.168.1.2
A 195.100.21.0/24 por 192.168.1.2
A 195.100.22.0/24 por 192.168.1.2
A 195.100.23.0/24 por 192.168.1.2
```

donde se supone que 192.168.1.2 es la interfaz por la que se accedería a la Universidad de Málaga.

El uso de CIDR permite sintetizar estas ocho entradas en una sola:

```
A 195.100.16.0/21 por 192.168.1.2
```

El software de hosts normalmente no soporta CIDR, por lo que éstos siguen 'pensando' en términos de redes clase A, B y C. Por tanto cuando dos hosts de la misma organización ubicados en redes clase C diferentes desean intercambiar tráfico han de hacer uso del router para iniciar el diálogo.

El espacio de redes clase A, que suponen la mitad del espacio total, está asignado actualmente sólo en un 50% aproximadamente. Está prevista la posibilidad de dividir cuando sea necesario la parte no utilizada de este rango de direcciones en redes de menor tamaño para su asignación mediante CIDR, igual que se hace actualmente con el rango 192.0.0.0-207.255.255.255.

La aplicación de CIDR ha permitido extender considerablemente la vida prevista inicialmente del espacio de direcciones de 32 bits de IPv4.

3.4 PROTOCOLOS DE CONTROL DE INTERNET

Antes hemos dicho que todo el tráfico de Internet está formado por datagramas IP. Normalmente los datagramas transportan TPDUs (Transport Protocol Data Unit) de TCP o UDP, que son los dos protocolos de transporte utilizados en Internet. Todas las aplicaciones de interés para el usuario de Internet (correo electrónico, transferencia de ficheros, videoconferencia, etc.) generan tráfico TCP o UDP. Sin embargo cuando vimos en la estructura de la cabecera de un datagrama el valor del campo protocolo observamos que existen muchos posibles significados del contenido de un datagrama IP, aparte de los 'normales' que serían TCP y UDP. Algunos de los datos que pueden transportarse en datagramas IP son mensajes de protocolos de control de Internet como los que veremos en este apartado. Los protocolos de control son una parte necesaria para el correcto funcionamiento de la red. Aquí describiremos los más utilizados que son ICMP, ARP, RARP, BOOTP y DHCP.

3.4.1 ICMP (Internet Control Message Protocol)

En el funcionamiento normal de una red se dan a veces situaciones extraordinarias que requieren enviar avisos especiales. Por ejemplo, si un datagrama con el bit DF (Don't Fragment) puesto a 1 no puede pasar por una determinada red el router donde se produce el problema debe devolver un mensaje al host emisor indicándole lo sucedido. El mecanismo para reportar todos estos incidentes en Internet es el protocolo conocido como ICMP, que veremos a continuación. Aquí solo describiremos someramente los más importantes, para una descripción detallada de todos ellos se puede consultar el RFC 792.

Conviene recordar que los mensajes ICMP viajan por la red como datagramas IP (con el valor 1 en el campo 'protocolo'), sujetos en los routers a las mismas reglas que cualquier otro datagrama. Los mensajes ICMP son generados por el host o router que detecta el problema o situación extraordinaria y dirigidos al host o router que aparece en el campo dirección origen del datagrama que causó el problema.

Para facilitar la identificación del datagrama por parte del host emisor la mayoría de los mensajes ICMP incluyen, además del código de error correspondiente, la cabecera y los primeros ocho bytes de datos del datagrama original.

A continuación describiremos los mensajes ICMP más importantes:

- **DESTINATION UNREACHABLE**. Este mensaje se produce cuando no se puede entregar el datagrama en su destino por diversas situaciones, entre las que destacaremos a modo de ejemplo las dos siguientes: a) cuando un router se encuentra con un datagrama que tiene puesto a 1 el bit DF (Don't Fragment) y que no cabe en la MTU de la red por la que ha de enviarlo, y b) cuando un router no encuentra en sus tablas ninguna ruta por la que pueda llegar a la dirección para la que va dirigido un datagrama. Obsérvese que cuando un router tiene configurada ruta por defecto nunca enviará mensajes ICMP Destination Unreachable por este segundo motivo.

- **SOURCE QUENCH**. Este mensaje se creó para permitir a los routers solicitar una reducción en el tráfico generado por los hosts en caso de congestión, por lo que actuaban como paquetes de asfixia. En la práctica se ha visto que el uso de este tipo de mensajes agrava los problemas en caso de congestión, por lo que la recomendación actual es que los routers no deben generar paquetes SOURCE QUENCH en ningún caso.

- **ECHO REQUEST y ECHO REPLY:** se usan para detectar si un destino determinado está operativo. Al recibir el mensaje el destino debe responder con el comando ICMP **ECHO REPLY**. El comando *ping*, muy utilizado en Internet para pruebas de accesibilidad, utiliza este comando. En la mayoría de las implementaciones de ping es posible especificar el tamaño del datagrama a enviar, y también la frecuencia de los envíos, con lo que se puede utilizar para generar un tráfico en la red de forma controlada. El comando ping suministra información del tiempo de ida y vuelta y porcentaje de datagramas perdidos, por lo que permite tener una idea bastante aproximada del estado de la red.

- **TIME EXCEEDED** se envía al emisor cuando un paquete es descartado porque su TTL ha llegado a cero. Esto puede se síntoma de que se ha producido algún bucle en la red, o que el valor del TTL utilizado es demasiado bajo para el diámetro de la red. Hay un programa muy utilizado para diagnóstico de redes, denominado *traceroute*[6], que averigua la ruta a un destino determinado enviando paquetes con TTL de 1, 2, 3, y así sucesivamente. A partir de los mensajes ICMP **TIME EXCEEDED** recibidos el programa puede deducir la ruta completa hasta el destino especificado. Traceroute mide además el tiempo de ida y vuelta en cada caso, y para dar una información estadísticamente más significativa normalmente envía tres paquetes para cada valor de TTL y muestra los tiempos de ida y vuelta de cada uno, dando así información no sólo de la ruta seguida sino de los retardos en cada parte del trayecto.

- **REDIRECT** se utiliza para alertar al host emisor cuando se sospecha que un paquete se está encaminando incorrectamente. Por ejemplo este mensaje lo utilizan los routers cuando reciben de un host datagramas que van dirigidos a otro host que se encuentra en la misma LAN.

3.4.2 Resolución de direcciones: ARP

Cuando se utilizan líneas punto a punto en una red la interfaz física fija el siguiente nodo al que se dirige el datagrama, ya que para cada enlace existe un único destinatario posible. En cambio cuando se utiliza una red multiacceso (por ejemplo RDSI, ATM o una LAN cualquiera) la tecnología utilizada para enviar los datagramas permite llegar por la misma interfaz física a más de un destinatario. En este caso es necesario algún mecanismo que permita saber a cual de todos los destinos posibles se dirigen los paquetes. Todas las redes multiacceso disponen de un sistema de direccionamiento propio, por ejemplo en una RDSI las direcciones serían los números de teléfono RDSI con los que queremos conectar, en una LAN serían las direcciones MAC de las estaciones, etc. En todos estos casos es el nivel de red el encargado de realizar la correspondencia entre la dirección en la tecnología multiacceso correspondiente y la dirección de red, correspondencia que se conoce como **resolución de direcciones**.

[6] Este programa está disponible en UNIX con su propio nombre y en Windows con el nombre *tracert*.

Existen múltiples mecanismos posibles para realizar la resolución de direcciones. De ellos comentaremos aquí algunos de los más utilizados.

a) **Tabla estática mantenida manualmente en cada nodo**. En este caso se tiene en cada nodo de la red multiacceso la tabla completa de equivalencia de direcciones. Esta técnica se utilizaba en las primeras redes locales y se utiliza actualmente en RDSI, X.25, Frame Relay y también en ATM en algunos casos. El principal problema que tiene es la necesidad de actualizar las tablas en todos los nodos de la red cada vez que se produce alguna modificación en la tabla de direcciones.

b) **Tabla dinámica mantenida de forma automática en un servidor de la red conocido por todos**. Cuando un nodo quiere enviar un mensaje a un destino determinado indica al servidor la dirección de red que busca y éste le devuelve la dirección de la red multiacceso correspondiente. Cualquier equipo que desee adherirse a la red deberá en primer lugar registrarse en el servidor para que éste incluya la entrada correspondiente en sus tablas. Este mecanismo se utiliza como veremos más adelante en IP sobre redes ATM como parte de la técnica denominada 'Classical IP over ATM'; en ese caso el servidor facilita la dirección ATM que corresponde a una dirección IP determinada. Los principales problemas de esta solución son la necesidad del registro previo y que el servidor se convierte en una pieza clave de la red que puede limitar la disponibilidad y el rendimiento en algunos casos.

c) **Establecer un mecanismo trivial por el que se pueda deducir la dirección de la red multiacceso a partir de la dirección de red**. De esta forma cualquier nodo puede derivar la dirección de la red multiacceso a partir de la dirección de red. Este mecanismo se emplea por ejemplo en DECNET que construye la dirección MAC añadiendo a la dirección de red un prefijo determinado y conocido por todos los nodos. Para que esto sea posible no se utiliza la dirección global grabada en la tarjeta de red y se recurre al uso de direcciones MAC locales (las fijadas por software, que tienen a 1 el segundo bit) para poder imponer la dirección MAC a partir de la dirección de red. Este mecanismo no puede ser utilizado simultáneamente por mas de un protocolo de red ya que se produciría conflicto entre las direcciones MAC requeridas por los diversos protocolos. Como veremos más adelante un mecanismo de este tipo se utiliza en IP para la resolución de direcciones multicast en Ethernet.

d) **Utilizar un mensaje broadcast para lanzar a la red una pregunta solicitando respuesta del nodo en la red multiacceso que posee la dirección de red buscada**. Esta técnica da máxima flexibilidad ya que los equipos no necesitan registrarse y no hay un servidor centralizado del que dependa el funcionamiento de la red. Sin embargo solo es factible en redes de naturaleza broadcast, como las redes locales. El principal inconveniente de esta solución es el uso de paquetes broadcast que produce una degradación del rendimiento de toda la red[7]. Esta técnica es la utilizada en IP sobre redes locales de todo tipo.

En este apartado veremos en detalle el funcionamiento del protocolo ARP (Address Resolution Protocol) que utiliza el mecanismo descrito en último lugar. Veamos como funciona con un ejemplo:

Supongamos que Luis tiene su PC conectado a una red local Ethernet mediante una tarjeta configurada con la dirección IP 147.156.1.2 y la máscara 255.255.0.0; Luis quiere iniciar una sesión de terminal remoto telnet con un ordenador cuya dirección IP es 147.156.1.3. En el momento en que Luis teclea *telnet 147.156.1.3* el software del PC compara la dirección de destino con la suya propia y con su máscara y deduce que el ordenador de destino se encuentra en la misma red local[8]. El PC genera entonces un mensaje ARP con la pregunta ¿quién tiene la dirección 147.156.1.3? y lo envía en una trama Ethernet que tiene como dirección MAC de destino la dirección broadcast (todo a unos); la trama es recibida y procesada por todas las máquinas de la red que en ese momento estén activas, siendo retransmitida a través de los conmutadores o puentes transparentes locales o remotos que haya. Eventualmente una

[7] El tráfico broadcast es altamente perjudicial para el rendimiento por dos razones: no es aislado por los conmutadores y además tiene que ser recibido y procesado por todas las estaciones de la red, ya que en principio les incumbe. En una red de gran tamaño con mucho tráfico broadcast el consumo en ciclos de CPU debido a este motivo puede llegar a ser considerable.

[8] A los efectos del funcionamiento de ARP es irrelevante el hecho de que los dos ordenadores se encuentren en el mismo dominio de colisiones o en dos diferentes unidos entre sí por puentes o conmutadores.

máquina (y solo una) se reconoce propietaria de la dirección IP solicitada y responde al mensaje; la respuesta puede ser y normalmente será una trama unicast puesto que el ordenador de destino ya conoce la dirección MAC del PC que lanzó la pregunta; la respuesta incluye la dirección MAC solicitada, por lo que a partir de ese momento ambos ordenadores pueden comunicarse mediante tramas unicast, de forma que la generación de tráfico broadcast queda estrictamente limitada al primer mensaje enviado por el PC de Luis.

Cada ordenador de la red local mantiene en memoria una tabla denominada *ARP cache* con las parejas de direcciones MAC-IP utilizadas recientemente. Por tanto si Luis termina su sesión telnet pero olvidó algún detalle podrá durante varios minutos establecer otra conexión telnet con el mismo ordenador sin necesidad de enviar otro paquete ARP broadcast. Generalmente cuando un ordenador envía un mensaje ARP buscando a otro *todas* las máquinas de la red, no sólo la destinataria del mensaje, aprovechan para 'fichar' al emisor, anotándolo en su ARP cache. De esta forma si más tarde necesitan contactar con dicha máquina podrán hacerlo directamente, sin necesidad de enviar un mensaje ARP broadcast.

Las entradas en la ARP cache expiran pasados unos minutos sin que haya tráfico con la máquina correspondiente, para permitir que se produzcan cambios en la tabla por ejemplo por avería de una tarjeta de red o cambio de la dirección IP de un host. El comando *arp –a*, disponible en UNIX y en Windows, nos permite conocer en cualquier momento la tabla ARP cache disponible en un host.

En caso de que en una misma red local haya por error más de un ordenador con la misma dirección IP se pueden producir resultados inesperados[9]; en el mejor de los casos el primero en responder será el único en ser incluido en la ARP cache de los ordenadores que busquen esa dirección, dejando a los demás inaccesibles. Un caso típico en que se da este problema es cuando dos o mas usuarios comparten un disquete con software de red en el cual están incluidos además del software los ficheros de configuración que contienen la dirección IP (afortunadamente este problema hoy en día es menos frecuente ya que normalmente los sistemas operativos incorporan el software de red). En una red de grandes dimensiones es fundamental disponer de mecanismos que permitan detectar y minimizar el riesgo de tener direcciones duplicadas.

Aunque lo hemos visto aplicado al caso concreto de resolver una dirección IP en una red local, el protocolo ARP tiene un ámbito bastante más amplio. A nivel Ethernet ARP es un protocolo diferente de IP, con un valor de Ethertype de 806 (hexadecimal). Esto permite que los routers no confundan los paquetes ARP con los paquetes IP y no los propaguen; de este modo los paquetes ARP quedan siempre confinados en red local donde se producen, evitando así que el tráfico broadcast que generan se propague a otras redes.

Para poder adaptarse a cualquier tipo de red broadcast y a cualquier protocolo a nivel de red el paquete ARP prevé el uso de longitudes arbitrarias tanto de la dirección de red como de la dirección de enlace. A título de ejemplo la tabla 3.7 muestra el formato del paquete ARP para el caso más habitual, que corresponde a una red con direcciones MAC de 6 bytes y direcciones IP de 4 bytes. Los campos se describen a continuación:

Campo	**Longitud (Bytes)**
Tipo de hardware	2
Tipo de protocolo	2
Long. Dirección hardware	1
Long. Dirección red	1
Código operación	2
Dirección MAC emisor	6
Dirección IP emisor	4
Dirección MAC destino	6
Dirección IP destino	4

Tabla 3.7.- Estructura de un paquete ARP IP para redes 802

[9] Esta es una forma educada de decir que se pueden producir caídas de sistema operativo.

Tipo de hardware: especifica el tipo de red local, por ejemplo el código 1 identifica Ethernet.

Tipo de protocolo: especifica el protocolo de red utilizado. Se emplean los mismos códigos que en el Ethertype (por ejemplo x0800 en el caso de IP).

Long. Dirección hardware: se especifica en bytes. Por ejemplo en el caso de direcciones MAC la longitud es de seis bytes.

Long. Dirección red: también en bytes, por ejemplo cuatro en el caso de IP.

Código operación: vale 1 en el caso de una pregunta ARP (¿quién tiene la dirección IP tal?) y 2 en el de una respuesta.

3.4.3 Resolución inversa de direcciones

A veces se plantea el problema inverso a ARP, es decir hallar la dirección IP a partir de una determinada dirección LAN. Por ejemplo cuando se arranca una estación de trabajo 'diskless', es decir sin disco, ésta ha de cargar su sistema operativo desde otro ordenador normalmente situado en la misma red local, pero desconoce todo lo relativo a su configuración de red, incluida la dirección IP; lo único que la estación conoce en principio es su dirección MAC, que se encuentra escrita en su tarjeta de red local.

3.4.3.1 RARP

Para estas situaciones se inventó RARP (Reverse Address Resolution Protocol), que funciona de la siguiente manera: la estación diskless envía un mensaje broadcast en el que indica su dirección MAC y solicita que alguien le informe de cual es la dirección IP que le corresponde. En la red habrá un servidor RARP encargado de atender este tipo de peticiones que consultará sus tablas y devolverá la dirección IP correspondiente.

RARP utiliza el mismo formato de mensaje que ARP. La única diferencia es que utiliza los códigos de operación 3 y 4 para la pregunta y respuesta RARP, respectivamente. Además en vez de utilizar el Ethertype x0806 emplea el x8035; esto permite a los hosts que no soportan el protocolo RARP descartar estos mensajes rápidamente, sin tener que analizar su contenido.

3.4.3.2 BOOTP

RARP aporta funcionalidades interesantes pero tiene algunas limitaciones importantes. Como la consulta se realiza mediante un mensaje broadcast el servidor RARP ha de estar en la misma LAN que el cliente; por tanto es necesario prever un servidor RARP en cada LAN, ya que los mensajes broadcast a nivel MAC no atraviesan los routers. Otra limitación es el hecho de que el protocolo RARP solo preve el envío de la dirección IP, cuando sería interesante aprovechar el mensaje para informar al cliente de todo el conjunto de parámetros relacionados con la configuración de la red (la máscara, el router por defecto, etc.). Para superar estas limitaciones de RARP se inventó el protocolo BOOTP (BOOTstrap Protocol).

Cuando un host lanza una pregunta BOOTP lo hace en un datagrama IP con la dirección de destino 255.255.255.255 y dirección de origen 0.0.0.0. De esta forma el datagrama es recibido por todos los hosts de la LAN. Si alguno de ellos es el servidor BOOTP consultará sus tablas y devolverá la información requerida al solicitante, en otro datagrama IP. Resulta interesante analizar como se envía dicha contestación. En principio, al tratarse de un datagrama IP el proceso servidor BOOTP debería consultar su tabla ARP cache para averiguar la dirección MAC del destinatario; es evidente que la ARP cache no contendrá dicha dirección puesto que el servidor BOOTP no ha realizado todavía ningún envío al cliente. Por tanto el paso siguiente sería que el servidor enviara un mensaje ARP request preguntado quien tiene la dirección IP solicitada, pero en este caso el cliente no respondería puesto que aún desconoce cual es su dirección IP. Esto es lo que en el RFC 951 (que especifica el protocolo BOOTP) se describe como el problema ‘del huevo y la gallina’. Existen dos posibles soluciones a este problema:

a) Enviar la respuesta en broadcast, es decir a la dirección 255.255.255.255. Esta es la solución más habitual.

b) Actualizar la tabla ARP cache para incluir en ella la entrada correspondiente al cliente y poder así enviar el datagrama a su destino. Este es un procedimiento irregular, ya que en principio el único autorizado a modificar la ARP cache de un host es el proceso ARP. Solo en algunos casos el kernel o los drivers disponen de mecanismos que permitan al proceso servidor BOOTP realizar dicha modificación.

Los mensajes BOOTP se envían en datagramas UDP/IP. Esto tiene la ventaja de permitir que el servidor BOOTP no esté ubicado en la misma LAN que el cliente, pero para que esto funcione es necesario disponer en la LAN del cliente de un router que actúe como agente de reenvío de mensajes BOOTP. Dicho router se encargará de recoger los BOOTP request de la LAN y redirigirlos hacia el servidor. El intercambio de mensajes BOOTP entre el agente de reenvío y el servidor BOOTP se realiza de forma normal, pero el problema que se plantea es la entrega del BOOTP reply al cliente, que como ya hemos visto requiere seguir un procedimiento especial (normalmente un envío broadcast). Para que quede claro quien es el responsable de realizar dicho procedimiento 'singular' en el sentido inverso, cuando el router reenvía el BOOTP request anota en él la dirección IP de la interfaz por la que lo recibió; dicha información acompañará el BOOTP reply correspondiente y permitirá al router realizar la entrega según el procedimiento singular. Para que este procedimiento funcione es preciso que las rutas sean simétricas, es decir que el BOOTP request y el BOOTP reply utilicen el mismo camino.

Además de la dirección el protocolo BOOTP permite indicar el nombre del host, la máscara de subred, el router por defecto, servidor de nombres, etc. Aunque no se utiliza a menudo también puede indicarse el nombre y ubicación del fichero desde el cual debe hacer boot la máquina cliente, si se desea que arranque de forma remota desde otra máquina. BOOTP fue el primer protocolo estándar para arranque automático de ordenadores en TCP/IP.

BOOTP permite centralizar en uno o unos pocos servidores la configuración de todas las máquinas de una red y debido a su mayor flexibilidad ha desplazado prácticamente por completo a RARP. Esta facilidad de configuración centralizada resulta especialmente útil en redes grandes, ya que permite hacer de manera cómoda cambios en la configuración de una red sin tener que hacer los cambios localmente máquina a máquina. La configuración centralizada ayuda a evitar la duplicidad de direcciones IP, que puede dar muchos problemas como comentábamos al hablar de ARP.

3.4.3.3 DHCP

Tanto RARP como BOOTP requieren una asignación estática biunívoca entre direcciones MAC y direcciones IP; hacen falta tantas direcciones IP como ordenadores vayan a utilizar el protocolo TCP/IP. Existen situaciones en las que esto no es conveniente, por ejemplo:

- Una empresa con 500 ordenadores quiere conectarse a la Internet, de forma que cualquiera de ellos pueda salir al exterior; para esto dispone de una clase C; se calcula que nunca habrá más de 200 ordenadores simultáneamente conectados a la Internet, por lo que en principio una red clase C sería suficiente, pero la necesidad de asignar una dirección IP a cada ordenador requiere disponer de 500 direcciones IP si se quiere ofrecer conectividad a todos ellos.

- En una universidad se dispone de una sala para la conexión a Internet de los ordenadores portátiles de los alumnos. No se conocen las direcciones MAC de los ordenadores que se utilizarán ni cuantos serán, lo único que se sabe es que no habrá en ningún momento más de 50 conectados simultáneamente ya que esta es la capacidad de la sala.

Evidentemente en estas situaciones es necesario un mecanismo más flexible de asignación de direcciones IP que el ofrecido por BOOTP.

Para resolver estos problemas el IETF diseñó en 1993 el protocolo DHCP (Dynamic Host Configuration Protocol), que es similar a BOOTP pero es más versátil en los mecanismos de asignación de direcciones IP. En DHCP los clientes pueden recibir sus direcciones por uno de los tres mecanismos siguientes:

- Asignación indefinida y estática. En este caso la dirección es fijada por el administrador. Este equivale a BOOTP.

- Asignación automática. La asignación es también estática pero la elección de la dirección IP correspondiente es tomada por el servidor DHCP la primera vez que el equipo le solicita su dirección.

- Asignación dinámica. En este caso el cliente recibe la dirección IP del servidor durante un tiempo limitado. Pasado ese tiempo el cliente debe renovar su solicitud o de lo contrario la concesión expirará. De esta forma una misma dirección puede ser reutilizada por diferentes máquinas en momentos diferentes. Esta modalidad es también conocida como 'alquiler de direcciones'.

La mayor flexibilidad de DHCP le ha convertido en el protocolo preferido para la configuración remota de ordenadores en redes locales. Con DHCP se mejora notablemente la seguridad y fiabilidad de una red; también se simplifican las labores de administración de la misma.

Con la asignación dinámica de direcciones DHCP desempeña en redes locales un papel similar a PPP en las conexiones de RTC.

Un inconveniente de la asignación dinámica de direcciones es que si se desea rastrear un problema y, como es lo normal, sólo se dispone de la dirección IP resulta más difícil (a veces imposible) averiguar que ordenador o usuario ha sido el causante del problema. Otro problema es la asociación de direcciones y nombres en el DNS; con la asignación dinámica diferentes máquinas pueden recibir el mismo nombre en diferentes momentos.

DHCP es lo más parecido a la autoconfiguración en IPv4.

RARP se describe en el RFC 903. BOOTP se describe en los RFC 951, 1497 y 1542. DHCP se describe en el RFC 1541.

3.5 PROTOCOLOS DE ROUTING

La Internet está formada por multitud de redes interconectadas, pertenecientes a diversas empresas y organizaciones. Todas estas redes interconectadas comparten a nivel de red el protocolo IP. Al margen de esta interoperabilidad existen dos aspectos fundamentales en los que las redes pueden diferir entre si:

- **El protocolo de routing utilizado**: existen como veremos multitud de protocolos de routing diferentes, unos basados en el algoritmo del vector distancia y otros en el del estado del enlace; incluso utilizando el mismo algoritmo se pueden emplean protocolos diferentes. Aun utilizando el mismo algoritmo y protocolo de routing dos proveedores diferentes normalmente no querrán que sus routers intercambien entre sí la misma información de routing que intercambian internamente.

- La **política de intercambio de tráfico**: un proveedor puede tener acuerdos bilaterales o multilaterales para intercambiar tráfico con otros, pero normalmente no estará dispuesto a ser utilizado como vía de tránsito para el tráfico entre dos proveedores si esto no está expresamente recogido en los acuerdos, aun cuando desde el punto de vista de topología de la red pueda ser ese el camino más corto entre ambas.

3.5.1 Sistema Autónomo

Entendemos por Sistema Autónomo (AS, Autonomous System) la subred que es administrada o gestionada por una autoridad común, que tiene un protocolo de routing homogéneo mediante el cual intercambia información en toda la subred y que posee una política común para el intercambio de tráfico con otras redes o sistemas autónomos.

Normalmente cada ISP (Internet Service Provider) constituye su propio sistema autónomo; por ejemplo en España (como en otros países) la red académica RedIRIS (que a estos efectos es un ISP mas) tiene un sistema autónomo propio. Los sistemas autónomos reciben números de dos bytes que se registran en el IANA de forma análoga a las direcciones IP; por ejemplo el sistema autónomo de RedIRIS tiene el número 766.

De la misma forma que existen unos rangos de direcciones IP reservados para redes privadas existe un rango de números de sistemas autónomos reservados para sistemas autónomos privados, que son los que van del 64512 al 65535. Por ejemplo los routers de las Universidades de Valencia y Politécnica de Valencia forman un sistema autónomo privado, ya que intercambian información de routing entre ellos pero no con otros , por tanto tienen asignado el sistema autónomo 65432.

Así pues, como mínimo en la Internet se dan dos niveles jerárquicos de routing, el que se realiza dentro de un sistema autónomo (AS) y el que se efectúa *entre* sistemas autónomos. Al primero lo denominamos routing interno, o routing en el interior de la pasarela (pasarela es una antigua denominación de router). Al routing entre sistemas autónomos lo denominamos routing externo, o también routing exterior a la pasarela. Dado que los requerimientos en uno y otro caso son muy diferentes, se utilizan protocolos de routing distintos. Los protocolos de routing dentro del sistema autónomo se denominan IGP (Interior Gateway Protocol), mientras que los utilizados entre sistemas autónomos se llaman EGP (Exterior Gateway Protocol).

3.5.2 Protocolos de routing interno (IGP)

En la Internet se usan actualmente diversos protocolos de routing interno. Estos pueden agruparse en protocolos de vector distancia entre los que destacamos RIP, RIPv2, IGRP y EIGRP, y protocolos del estado del enlace de los que los más importantes son IS-IS y OSPF. Los describiremos someramente, centrándonos en el caso de OSPF que es el más importante por su mayor uso y sofisticación.

3.5.2.1 RIP y RIPv2

RIP (Routing Information Protocol) es uno de los protocolos de routing más antiguos y deriva del protocolo de routing de XNS (Xerox Network Systems); RIP sufre los problemas típicos de los algoritmos basados en el vector distancia, tales como la cuenta a infinito, etc. Además RIP arrastra otros problemas que son consecuencia de ser un protocolo de routing muy antiguo, como son:

- Métricas basadas exclusivamente en número de saltos
- No soporta subredes ni máscaras de tamaño variable (si en RIPv2).
- No permite usar múltiples rutas simultáneamente.
- Se genera una gran cantidad de información de routing que se intercambia cada 30 segundos. Con el paso del tiempo los routers tienden a sincronizarse de forma que todos acaban enviando los paquetes a la vez; esto provoca congestión y parones en la red durante el momento en que se intercambia la información de routing.

Algunos de estos problemas aumentan a medida que crece el tamaño de los sistemas autónomos, por lo que en la práctica no es aconsejable usar RIP en ninguna red que tenga mas de 5 a 10 routers. A pesar de todos sus inconvenientes RIP aún se utiliza en algunas partes de Internet. Existen dos versiones de RIP: la versión 1, que se definió en el RFC 1058 y se publicó en 1983 (aunque se empezó a utilizar mucho antes) se ha declarado histórica, es decir su uso está desaconsejado. En vista de la popularidad de RIP y

de los muchos problemas que presentaba en 1993 se publicó RIP versión 2, que intentaba resolver al menos algunos de ellos (RFC 1388).

RIP está implementado en muchas versiones de UNIX, aunque desgraciadamente se trata generalmente de RIPv1.

3.5.2.2 IGRP y EIGRP

A pesar de sus inconvenientes, el routing por vector distancia tiene algunos serios partidarios. Quizá el más importante sea la empresa Cisco, actualmente el principal fabricante de routers en el mundo. En 1988, cuando el único protocolo de routing estandarizado y ampliamente utilizado era RIP, Cisco optó por crear un protocolo de routing propio denominado IGRP (Interior Gateway Routing Protocol) para resolver algunos de los problemas que presentaba RIP. IGRP está basado también en el vector distancia. Cisco siguió (y sigue) apostando por los protocolos de routing basados en el vector distancia ya que en 1993 produjo un nuevo protocolo denominado EIGRP (Enhanced IGRP) que introducía mejoras importantes respecto a IGRP, pero basado también en el vector distancia. Hay que resaltar que tanto IGRP como EIGRP son protocolos propietarios, y no hay implementaciones de ellos para equipos de otros fabricantes, por lo que el uso de estos protocolos requiere que todos los routers del sistema autónomo correspondiente sean de Cisco. Los routers Cisco también pueden funcionar con protocolos estándar, tales como RIP OSPF e IS-IS.

EIGRP es el protocolo de routing utilizado por ejemplo en el sistema autónomo de la Universidad de Valencia.

3.5.2.3 OSPF

La respuesta del IETF a los problemas de RIP fue OSPF (Open Shortest Path First), protocolo de routing basado en el estado del enlace. OSPF fue desarrollado entre 1988 y 1990, y en 1991 ya se había producido OSPF versión 2. OSPF esta basado en IS-IS y muchos de los conceptos que maneja son comunes a ambos protocolos. Es un estándar Internet y es el protocolo actualmente recomendado por el IAB para sustituir a RIP. Su complejidad es notablemente superior, mientras que la descripción de RIP ocupa menos de 20 páginas la especificación de OSPF emplea más de 200. La especificación vigente de OSPF está en el RFC 2328.

Entre las características más notables de OSPF podemos destacar las siguientes:

- Es un algoritmo dinámico autoadaptativo, que reacciona a los cambios de manera automática y rápida.
- Soporta una diversidad de parámetros para el cálculo de la métrica, tales como capacidad (ancho de banda), retardo, etc.
- Realiza balance de carga si existe más de una ruta hacia un destino dado.
- Establece mecanismos de validación de los mensajes de routing, para evitar que un usuario malintencionado envíe mensajes engañosos a un router.
- Soporta rutas de red, de subred y de host.
- Se contempla la circunstancia en la que dos routers se comuniquen directamente entre sí sin que haya una línea directa entre ellos, por ejemplo cuando están conectados a través de un túnel.

OSPF permite dos niveles de jerarquía creando lo que se denominan áreas dentro de un sistema autónomo. De esta forma un router sólo necesita conocer la topología e información de routing correspondiente a su área, con lo que la cantidad de información de routing se reduce. En redes complejas esta es una característica muy valiosa.

Los algoritmos de routing por el estado del enlace se aplican dentro de cada área. En todo Sistema Autónomo (AS) hay al menos un área, el área 0 denominada backbone. Un router puede pertenecer simultáneamente a dos o mas áreas, en cuyo caso debe disponer de la información de routing y ejecutar los cálculos correspondientes a todas ellas. Al menos un router de cada área debe estar además en el backbone, para conectar dicha área con el resto del Sistema Autónomo. Dos áreas sólo pueden hablar entre sí a través del backbone.

En OSPF se contemplan cuatro clases de routers:

- Routers *backbone*; son los que se encuentran en el área 0 ó backbone.
- Routers *internos*; los que pertenecen únicamente a un área.
- Routers *frontera de área*; son los que están en mas de un área, y por tanto las interconectan (una de las áreas interconectadas siempre es necesariamente el backbone).
- Routers *frontera de Sistema Autónomo*; son los que intercambian tráfico con routers de otros Sistemas Autónomos. Estos routers pueden estar en el backbone o en cualquier otra área.

Existen tres tipos de rutas: intra-área, inter-área e inter-AS. Las rutas intra-área son determinadas directamente por cualquier router, pues dispone de toda la información; las rutas inter-área se resuelven en tres fases: primero ruta hacia el backbone, después ruta hacia el área de destino dentro del backbone, y por último ruta hacia el router deseado en el área de destino.

Para el intercambio de información cada router envía cuando arranca unos mensajes de salutación, denominados HELLO, por todas sus interfaces. Este y otros mensajes los reenvía periódicamente para asegurarse de que las líneas permanecen operativas. Con la información que posee y la recabada en respuesta a sus mensajes el router calcula las rutas óptimas para cada destino en cada momento.

Cuando en una red local hay varios routers resulta poco eficiente que cada uno intercambie toda la información de routing que posee con todos los demás, ya que mucha de la información sería redundante. En una red local con *n* routers esto produciría $(n^2-n)/2$ intercambios de información diferentes. En estos casos OSPF prevé que uno de los routers se convierta en el *router designado*, siendo éste el único que intercambiará información con todos los demás. De esta forma el número de intercambios de información se reduce a $n-1$.

3.5.2.4 IS-IS

El protocolo de routing IS-IS (Intermediate System-Intermediate System) está basado en el algoritmo del estado del enlace; además IS-IS permite hacer routing integrado, es decir calcular las rutas una vez y aplicarlas para todos los protocolos utilizados, permitiendo así auténtico routing multiprotocolo (de este tema hablaremos en más detalle en el capítulo de Internetworking). Soporta hasta ocho niveles jerárquicos, para reducir así la cantidad de información de routing intercambiada. IS-IS fue diseñado para el protocolo DECNET (de Digital) y adoptado después por ISO como protocolo de routing para el protocolo de red CLNP[10]. Una variante de IS-IS se utiliza en Netware de Novell.

IS-IS también se utiliza en algunas zonas de la Internet. El protocolo IS-IS se especifica en el RFC 1142.

IS-IS no es un estándar Internet, aunque se especifica en el RFC 1142. Actualmente es el protocolo utilizado en las redes grandes, por ejemplo la gran mayoría de las redes de los ISPs utilizan IS-IS en lugar de OSPF. Actualmente es el protocolo utilizado por RedIRIS, que migró del EIGRP que utilizaba anteriormente.

[10] CLNP (ConnectionLess Network Protocol) es un protocolo desarrollado por ISO a imagen y semejanza de IP. Su mayor diferencia estriba en el uso de direcciones OSI de 20 bytes en vez de lasde 4 bytes de IP.

3.5.3 Protocolos de routing externo

Todos los protocolos de routing hasta ahora descritos se emplean dentro de sistemas autónomos. Normalmente un sistema autónomo corresponde a una subred que tiene una entidad común desde el punto de vista administrativo y de gestión, puede ser por ejemplo la red de una gran empresa, de un proveedor de servicios Internet o la red académca de un país como RedIRIS. En estos casos se supone que el protocolo de routing ha de buscar la ruta óptima atendiendo únicamente al criterio de minimizar la 'distancia' medida en términos de la métrica elegida para la red.

La selección de rutas para el tráfico entre sistemas autónomos plantea un problema diferente, ya que la cuestión no se reduce a la selección de la ruta óptima sino que se debe atender a criterios externos que obedezcan a razones de tipo político, económico, administrativo, etc. (recordemos que se trata de decidir el routing entre redes que pertenecen a organizaciones diferentes). Un ejemplo típico de este tipo de restricciones es el caso en que la ruta óptima entre dos sistemas autónomos, X e Y, pasa por un tercero Z que no desea que su red sea utilizada como vía de tránsito. Para dar cabida a la utilización de criterios externos en el cálculo de las rutas entre sistemas autónomos se utilizan en este caso otro tipo de protocolos de routing, denominados protocolos de routing externo.

Hasta 1990 se utilizaba como protocolo de routing externo en la Internet el denominado EGP (Exterior Gateway Protocol), diseñado entre 1982 y 1984. Como era de esperar EGP no fue capaz de soportar la enorme evolución que sufrió Internet y como ya era habitual el IETF desarrolló un nuevo protocolo de routing externo, denominado BGP (Border Gateway Protocol). La primera especificación de BGP apareció en 1989; desde entonces el IETF ha producido cuatro versiones de BGP; las especificaciones actualmente vigentes de BGP-4 se encuentran en el RFC 1771.

Los routers que utilizan BGP (pertenecientes a diferentes ASes) forman entre ellos una red e intercambian información de routing para calcular las rutas óptimas; se utiliza el vector distancia, pero para evitar el problema de la cuenta a infinito la información intercambiada incluye, además de los routers accesibles y el costo, la ruta completa utilizada para llegar a cada posible destino; de esta forma el router que recibe la información puede descartar las rutas que pasan por él mismo que son las que podrían dar lugar al problema de la cuenta a infinito. La especificación de la ruta completa permite también a los routers revisar si dicha ruta es conforme con las políticas que se hayan especificado en cuanto a tránsito por otros sistemas autónomos.

BGP permite introducir manualmente restricciones o reglas de tipo 'político'; éstas se traducen en que cualquier ruta que viola la regla recibe automáticamente una distancia de infinito.

Para simplificar la gestión de los Sistemas Autónomos se crean Confederaciones de Sistemas Autónomos; una confederación es vista como un único Sistema Autónomo desde el exterior. Esto equivale a introducir en el protocolo de routing externo dos niveles jerárquicos, con lo que se reduce la información de routing de forma análoga a lo que ocurría con las áreas de OSPF dentro de un Sistema Autónomo.

3.5.4 Puntos neutros de interconexión

Cuando dos ISPs están conectados a la Internet en principio siempre es posible el intercambio de información entre ellos. Sin embargo este intercambio no siempre ocurre por el camino óptimo. Por ejemplo, si en España dos ISPs contratan conectividad Internet, uno a Telefónica y el otro a British Telecom (BT), su intercambio de tráfico se realizará normalmente a través de algún punto de interconexión fuera de España, lo cual no es óptimo ya que consume costosos recursos de líneas internacionales. La solución a este problema es la realización de un acuerdo bilateral entre los dos proveedores (Telefónica y BT), de forma que se establezca un enlace directo entre sus sistemas autónomos en España para que puedan intercambiar tráfico sin necesidad de salir fuera. Cuando el número de proveedores crece la cantidad de enlaces que hay que establecer aumenta de forma proporcional a $(n^2-n)/2$ donde n es el número de proveedores que intercambian tráfico [11]. Para reducir

[11] El problema que se plantea es en cierto modo parecido al que se da cuando varios routers de una misma red local participan en un protocolo de routing, problema que OSPF resolvía mediante el mecanismo del router designado.

el número de enlaces se suelen crear los denominados puntos neutros de interconexión, consistentes en un nodo normalmente gestionado por una entidad independiente (para garantizar su 'neutralidad') al cual se conectan los proveedores que desean participar. En principio en el ámbito del punto neutro el intercambio de tráfico debería ocurrir sin restricciones entre todos los proveedores; sin embargo en la práctica los proveedores han de establecer acuerdos bilaterales, por lo que no siempre dos proveedores conectados a un mismo punto neutro intercambian tráfico.

La implementación de un punto neutro es muy sencilla. Se trata básicamente de un conmutador LAN Ethernet/Fast Ethernet en el que a cada proveedor se le asigna una LAN a la que puede conectar sus routers. Los routers de diferentes proveedores intercambian información de routing mediante BGP. Cada proveedor ha de conseguir los medios necesarios para conectar su red al punto neutro (líneas dedicadas, etc.). Físicamente las instalaciones de un punto neutro han de cumplir unos requisitos de fiabilidad y seguridad muy altos, ya que se trata de un elemento crítico en el funcionamiento de la red.

Existen gran cantidad de puntos neutros de interconexión en Internet. En España el primer punto neutro de interconexión que se creó en febero de 1997 fue el ESPANIX (http://www.espanix net/) ubicado en el Centro de Proceso de Datos de Banesto, en Madrid. Dado que se trata de un punto neutro de interconexión a nivel nacional solo se conectan a él los proveedores que tienen conectividad internacional propia. Los proveedores que no tienen enlaces internacionales se conectan normalmente a través de alguno de esos proveedores con conectividad internacional, por lo que acceden de forma indirecta al punto neutro. Actualmente (octubre de 2003) existen 29 proveedores conectados a ESPANIX.

Posteriormente han ido apareciendo otros puntos neutros en España. En junio de 1999 se creó el CATNIX (www.catnix.net) en el CESCA (Centre de Supercomutació de Catalunya) en Barcelona, que conecta actualmente a 10 proveedores. En julio de 2002 el GALNIX (www.galnix net) en el CESGA (Centro de Supercomputación de Galicia) que conecta 6 proveedores. También se está poniendo en marcha el EUSKONIX en la Universidad del País Vasco y en Madrid han aparecido recientemente otros dos puntos neutros, el MAD-IX (www.mad-ix.net) y el NAP (www napmadrid.com).

3.5.5 Routing Multicast

Las direcciones IP clase D (entre 224.0.0.0 y 239.255.255.255) están reservadas para tráfico multicast. Se pueden crear dos tipos de grupos multicast, temporales y permanentes. Algunos grupos permanentes que están ya predefinidos son los siguientes:

224.0.0.1	Todos los hosts en una LAN
224.0.0.2	Todos los routers en una LAN
224.0.0.5	Todos los routers OSPF en una LAN
224.0.0.6	Todos los routers OSPF designados en una LAN

Los grupos temporales se han de crear cada vez que se van a utilizar. Un host (más exactamente un proceso en un host) puede solicitar unirse a un grupo multicast (join), o puede decidir abandonarlo (leave). Cuando el último proceso de un host abandona un grupo multicast el host mismo abandona el grupo. En Internet el protocolo que gestiona todas las operaciones relacionadas con los grupos multicast es el IGMP (Internet Group Management Protocol). Para captar una emisión multicast es preciso fromar parte del grupo correspondiente, pero no es necesario estar en dicho grupo para realizar una emisión multicast hacia él.

En una LAN, al ser el medio físico intrínsecamente broadcast, no es necesaria ninguna acción especial para permitir el tráfico multicast; los paquetes viajan por la red y los hosts capturan los que corresponden a los grupos multicast a los que pertenecen. El router multicast de la LAN pregunta aproximadamente una vez por minuto a los hosts de su LAN (dirección 224.0.0.1) en que grupos multicast están interesados. Los hosts devuelven la relación de direcciones clase D en las que están interesados. De acuerdo con las respuestas que recibe el router van actualizando el árbol de distribución, añadiendo o suprimiendo (podando) ramas de éste. Si el detecta que alguna dirección multicast ha dejado de interesar (es decir, ya no hay miembros de ese grupo en la LAN) envía un mensaje al router siguiente en el árbol solicitándole le ‘pode’, es decir le suprima de la distribución. Inversamente puede también solicitar su adhesión a un grupo en el que antes no estuviera.

En enlaces punto a punto los routers han de revisar regularmente el árbol de distribución multicast, a fin de ‘podar’ de éste las ramas innecesarias e incluir las que presenten nuevos miembros del grupo multicast; de esta forma se evita cargar las líneas con tráfico innecesario.

La resolución de direcciones multicast en redes locales no se realiza mediante el protocolo ARP, ya que no tendría sentido que el emisor multicast tuviera que preguntar a los miembros del grupo cual es la dirección MAC multicast correspondiente a una dirección IP determinada. En este caso se aplica la técnica de emplear un algoritmo que permita deducir la dirección MAC a partir de la dirección IP (RFC 1112): la dirección IP multicast está representada por 28 bits (ya que los cuatro primeros siempre son 1110) por lo que se podría trasladar por ejemplo a los 28 bits menos significativos de la dirección MAC, fijando los 20 primeros. Dado que

el OUI (Organizationally Unique Identifier) ocupa los 24 primeros bits de la dirección MAC para poder utilizar cuatro de estos bits en las direcciones multicast habría sido necesario reservar para este fin 16 valores consecutivos de OUI, cosa que llegó a ser propuesta por el IETF al IEEE, pero no fue considerada aceptable por éste. En su lugar se decidió utilizar la mitad del OUI que había reservado para el IETF (01.00.5E), dejando así 23 bits libres en los que se mete la parte menos significativa de la dirección IP multicast de 28 bits. La correspondencia por tanto no es biunívoca, ya que existen 32 direcciones IP multicast diferentes por cada dirección MAC multicast; por ejemplo las direcciones IP multicast 224.0.0.1 y 224.128.0.1 que tienen iguales los 23 últimos bits se mapean ambas en la dirección MAC multicast 01.00.5E.00.00.80[12]. Podrá por tanto ocurrir que en una misma LAN dos grupos IP multicast diferentes utilicen la misma dirección MAC multicast; en este caso algunos hosts capturarán tramas multicast que luego, al examinar la cabecera IP, descubrirán que no iban dirigidas a ellos; esto produce una pequeña pérdida de eficiencia por el tiempo que el nivel de red emplea en analizar una trama que no iba dirigida a él, pero la probabilidad de que esto ocurra en la práctica es tan pequeña que la pérdida de eficiencia es despreciable[13]. Aun cuando se produzca coincidencia de direcciones MAC entre dos grupos diferentes nunca debe producirse por este motivo la entrega al nivel de transporte de datagramas que no vayan dirigidos al grupo multicast al que pertenece el host.

El soporte de tráfico multicast en routers es relativamente reciente, y existen aún pocas experiencias de redes con tráfico multicast en redes de área extensa en producción, casi todas ellas limitadas a redes académicas.

La mayor experiencia de routing multicast que ha habido en la Internet es la conocida como MBone (Multicast Backbone). La red MBone existe desde 1992 y se emplea principalmente con un conjunto de aplicaciones de videoconferencia. Las reuniones del IETF, congresos y todo tipo de eventos se transmiten regularmente por MBone. Hay retransmisiones de ámbito mundial, continental, nacional y regional.

A veces se quiere interconectar dos redes con tráfico multicast que están conectadas entre sí mediante una red que solo soporta tráfico unicast. En estos casos es posible construir un túnel entre dos routers de las redes multicast y enviar los datagramas multicast encapsulados en paquetes unicast; dado que estos paquetes son vistos como tráfico normal por la red unicast los routers multicast pueden manejarlos sin ningún problema. En estos casos se habla de 'islas' multicast unidas a través de túneles unicast. En parte la red MBone está construida mediante túneles de este tipo usando hosts UNÍX como routers multicast (existe software de routing multicast de libre distribución disponible para los principales sistemas operativos UNIX, como Linux, Solaris, Iris, etc.).

Un problema que se plantea con el tráfico multicast en MBone es la forma de limitar el ámbito o alcance de los paquetes. En principio un usuario puede estar interesado en realizar una emisión multicast en un ámbito restringido (por ejemplo a España únicamente). En MBone esto se puede resolver de dos formas: la antigua, aun bastante utilizada el campo TTL de la cabecera IP para limitar el alcance del datagrama. Los routers multicast que se encuentran en la frontera del ámbito correspondiente (en nuestro ejemplo los routers internacionales) restarán varias unidades al TTL de forma que esos paquetes no puedan salir al exterior. La solución más reciente a este problema consiste en asignar el ámbito de acuerdo con el rango de direcciones multicast de que se trate, y elegir la dirección de acuerdo con este criterio. Para este fin se ha reservado el rango de 239.0.0.0 a 239.255.255.255 según se especifica en el RFC 2365.

3.6 IPV6

Aunque la adopción de medidas paliativas como CIDR ha dado un respiro momentáneo en el problema del agotamiento del espacio de direcciones y tablas de routing, es evidente que hay que ampliar el campo dirección en la cabecera de los datagramas IP.

En un intento por resolver este problema el IETF empezó a trabajar ya en 1990 en una nueva versión de IP. Aunque fuera el más importante, la escasez de direcciones no era el único problema que presentaba el protocolo IP, por lo que ya puesto a diseñar un nuevo protocolo el IETF decidió abordar también otras deficiencias detectadas. Los objetivos planteados fueron los siguientes:

- **Direcciones**: Establecer un espacio de direcciones que no se agote en el futuro previsible.
- **Eficiencia**: Reducir el tamaño de las tablas de routing. Simplificar el protocolo (la cabecera IP) para permitir a los routers procesar los paquetes más rápidamente.
- **Seguridad**: Ofrecer mecanismos que permitan incorporar fácilmente en el protocolo medidas de seguridad (privacidad y validación) mediante técnicas criptográficas.
- **Tipo de servicio**: Manejar mejor los diferentes tipos de servicio, en especial para poder ofrecer garantías de Calidad de Servicio y para permitir el uso de aplicaciones en tiempo real.

[12] Es importante recordar en este punto que las direcciones MAC en Ethernet se representan empezando por el bit menos significativo de cada byte. En cambio en Token Ring y FDDI la representación es al contrario.
[13] Suponiendo un reparto aleatorio la probabilidad de coincidencia entre dos grupos multicast sería de una en diez millones ($2^5 / 2^{28} = 0{,}0000001$)

- **Multicasting**: Facilitar el uso de aplicaciones multicast.
- **Movilidad**: Permitir la movilidad de un host sin necesidad de cambiar su dirección.
- **Evolución**: Contemplar un mecanismo que permita extender el protocolo en el futuro.
- **Compatibilidad**: Permitir la coexistencia del protocolo nuevo con el viejo.

El IETF hizo una convocatoria pública (RFC 1550) para que se presentaran propuestas de como podría ser el nuevo protocolo. De las propuestas iniciales se fueron descartando las consideradas menos interesantes, y finalmente la propuesta finalmente adoptada fue un híbrido de dos de las presentadas.

Durante el período en que se celebraban en el seno del IETF las discusiones sobre las diferentes propuestas el nuevo protocolo se denominó IPng o IP next generation (de la serie de televisión Star Trek cuya segunda parte recibió esa denominación[14]. Finalmente el nombre oficial elegido fue IP Versión 6 o abreviadamente IPv6 (el IP actualmente en uso es Versión 4, y la Versión 5 se había utilizado ya para un protocolo experimental denominado ST, STream protocol). El protocolo se especificó en el RFC 1883 publicado en diciembre de 1995; este RFC ha sido sustituido posteriormente por el RFC 2460 que introdujo pequeños cambios en la cabecera relativos al campo Differentiated Services, publicado en diciembre de 1998.

Uno de los aspectos más polémicos que rodearon la estandarización de IPv6 fue la decisión sobre el tamaño de las direcciones a utilizar. Se barajaron alternativas que iban desde ocho hasta 20 bytes. Probablemente la opción de 20 bytes era la más razonable, no porque hicieran falta direcciones tan grandes sino porque este era el formato utilizado en las direcciones OSI y existía ya un protocolo muy similar a IP que utilizaba este formato de direcciones, que era CLNP. De esta forma IPv6 podría haberse reducido simplemente a la adopción de CLNP, y de hecho esta opción llegó a plantearse seriamente en el seno del IAB. Pero cuando la noticia llegó a oídos del IETF se produjo un clamor en contra de CLNP, fundamentalmente por razones políticas: después de los años de enfrentamiento vividos entre los partidarios de los protocolos OSI y los de TCP/IP se consideraba inaceptable y políticamente incorrecto que el IAB adoptara CLNP como el IP del futuro, lo cual habría significado que después de todo había algo aprovechable en los protocolos OSI. Finalmente el IAB aceptó el requerimiento del IETF, desestimó la propuesta de adoptar CLNP y se creó un protocolo con direcciones de 16 bytes.

En el mercado existen ya varias implementaciones de IPv6 para hosts y para routers, aunque su uso hasta ahora se ha limitado a experiencias piloto.

IPv6 coincide con IPv4 en ofrecer un servicio de entrega de datagramas sin garantías, es decir 'best effort'. Se contemplan sin embargo algunas opciones que permiten ofrecer calidad de servicio.

IPv6 no es realmente compatible con IPv4 ya que utiliza un formato de cabecera diferente, pero lo es (con pequeñas modificaciones) con los demás protocolos de Internet. La implantación del nuevo protocolo se está realizando de forma gradual mediante la creación de 'islas' IPv6 en las que se utiliza el nuevo protocolo; para la interconexión de esas islas a través del backbone IPv4 se utilizan túneles, de forma similar a lo que se hace con la red MBone. La red experimental así formada se conoce como 6Bone (IPv6 Backbone) y empezó a funcionar en 1996. En España ya hay una red 6Bone funcionando entre varias universidades españolas desde 1997. La Universidad de Valencia se incorporó a esta red en 1998.

Los protocolos de routing se han tenido que modificar para tener en cuenta las características propias y el nuevo formato de direcciones que utiliza IPv6; así se ha creado por ejemplo RIPv6 y OSPFv6.

Muy brevemente algunas de las principales virtudes de IPv6 son las siguientes:

- Las direcciones son de 16 bytes, lo cual da un espacio de direcciones increíblemente grande, suficiente con creces para todo futuro uso previsible.

[14] Por alguna razón esta serie, y la ciencia ficción en general, siempre ha gozado de gran popularidad entre los ‘gurús’ de Internet

- La cabecera se simplifica, pasando de 13 a 8 campos[15]. Se acelera así el proceso en los routers.
- Se ha mejorado el soporte de los campos opcionales en la cabecera.
- La seguridad (validación y privacidad) es ahora una parte fundamental del protocolo.
- Se dan más facilidades que antes para especificar el tipo de servicio.

3.6.1 Direcciones en IPv6

Una dirección IPv6 está compuesta por 16 bytes. Los primeros bits identifican el tipo de dirección, de manera análoga a IPv4. Existen muchas clases de direcciones, pero no todas tienen asignado el mismo rango, y la mayoría están reservadas para usos futuros.

Se ha previsto un rango específico para direcciones IPv4. De esta forma cualquier dirección IPv4 puede incluirse en un datagrama IPv6.

Una parte del espacio de direcciones se reserva para distribución geográfica, de manera similar a como se hace actualmente en la Internet con CIDR.

Otra parte se ha reservado para repartir direcciones por proveedor. Se ha contemplado la posibilidad de que la Internet (o una parte de ella) evolucione hacia una red que interconecte las redes de los grandes proveedores a nivel mundial, siendo secundaria en este caso la ubicación geográfica. Se contempla para este caso una estructura de direcciones jerárquica con varios niveles.

No se ha previsto ninguna dirección específica para broadcast, ya que esto se considera un caso particular de multicast. Para las direcciones multicast se ha previsto también un rango específico, y en la propia dirección multicast se ha reservado un campo de 4 bits que permite especificar el ámbito o alcance que se pretende tenga la emisión. El ámbito puede valer entre 0 y 15; el valor 14 se utiliza para indicar todo el planeta, y el valor 15 podría eventualmente utilizarse para transmisiones multicast que abarquen el espacio interestelar. Esto es similar a lo que se hace en IPv4 cuando se limita el ámbito en base a la dirección (RFC 2365).

Además de envíos unicast y multicast pueden hacerse envíos *anycast*, en los que el paquete se envía a uno cualquiera de los miembros de un grupo, sin importar a cual. Esto permite por ejemplo acceder a un servidor multihomed haciendo balance de carga entre varias interfaces, o por aquella que esté mas cerca del solicitante. También facilita configuraciones redundantes donde un determinado servicio puede ser facilitado por mas de un servidor.

Se ha previsto un rango de direcciones de significado puramente local, equivalentes a las actuales redes privadas (intranets), para casos en que por razones de seguridad se quiera estar completamente aislado del exterior.

La escritura de direcciones de 16 bytes usando el sistema tradicional resulta muy farragosa, por ejemplo:

128.0.0.0.0.0.0.0.1.35.69.103.137.171.205.239

Para evitarlo se ha diseñado una notación en la que las direcciones se escriben como 8 grupos de 4 dígitos hexadecimales separados por dos puntos; por ejemplo la dirección anterior se escribiría así:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Dado que muchas direcciones contendrán gran cantidad de ceros se ofrece la posibilidad de utilizar una notación abreviada en la que los ceros a la izquierda pueden omitirse, y además si uno o más grupos tienen todos los dígitos a cero se pueden omitir poniendo en su lugar dobles dos puntos (::). Así por ejemplo la dirección anterior se escribiría:

[15] En IPv4 no se consideran campos los bits reservados ni las opciones.

8000::123:4567:89AB:CDEF

Para evitar ambigüedad la notación abreviada :: sólo puede utilizarse una vez en una dirección.

Por último, para facilitar la escritura de direcciones IPv4 se prevé también el uso de la notación decimal si se desea utilizando puntos, por ejemplo :

::147.156.11.11

Gracias a la mucho mayor longitud del campo dirección en IPv6 se pueden reservar los últimos bytes de la dirección para incluir una parte local globalmente única en la dirección, que normalmente es una dirección MAC IEEE[16]. Este 'truco' (utilizado ya en redes OSI y ATM) permite la autoconfiguración de los sistemas: el equipo fija la parte host de su dirección a partir de la dirección contenida en su tarjeta de red y escucha por el cable para que el router le informe de la parte de red, con lo que la conexión de nuevos equipos a una red puede hacerse verdaderamente 'Plug-and-Play'. Una empresa que estuviera conectada a Internet a través de un proveedor podría cambiar de proveedor y cambiar todas sus direcciones sin más que cambiar en el router la parte de red de la dirección. Los ordenadores obtendrían el nuevo prefijo del router y le añadirían cada uno la parte host correspondiente. El uso de la dirección MAC como sufijo garantiza que la dirección sea única. La autoconfiguración también facilita grandemente la movilidad de equipos.

Con 16 bytes es posible crear 2^{128} direcciones, equivalente a aproximadamente 3×10^{38}; aunque el reparto produce mucho despilfarro es evidente que IPv6 no andará escaso de direcciones en mucho tiempo.

3.6.2 La cabecera de IPv6

Para tener una idea más exacta de las características de IPv6 vamos a repasar brevemente los campos de la cabecera, cuya estructura aparece en la tabla 3.8.

Campo	Longitud (bits)
Versión	4
DS (Differentiated Services)	8
Etiqueta de flujo	20
Longitud de carga útil	16
Siguiente cabecera	8
Límite de saltos	8
Dirección origen	128
Dirección destino	128

Tabla 3.8.- Formato de la cabecera de un datagrama IP Versión 6

El campo **versión** vale siempre 6. En principio este campo debería servir para distinguir las versiones de IP, de forma que todas pudieran identificarse como un mismo protocolo a nivel de enlace, por ejemplo utilizando el mismo valor de Ethertype. En la práctica muchas implementaciones de IPv4 no comprueban este campo sino que suponen que el datagrama es IPv4 cuando la cabecera del nivel de enlace especifica

[16] A partir del RFC 2373 se ha previsto utilizar para IPv6 el nuevo formato de direcciones MAC de ocho bytes aprobado por el IEEE, denominadas direcciones EUI-64 (EUI = Extended Unique Identifier). Las direcciones EUI-64 utilizan el mismo prefijo de tres bytes que las direcciones MAC tradicionales para identificar al fabricante pero amplían a cinco bytes la parte que identifica al equipo. Se ha definido una forma estándar de convertir las direcciones de 48 bits en direcciones de 64 bits.

protocolo IP, por lo que a pesar de existir el campo versión es necesario asignar a IPv6 un valor propio en el nivel de enlace, como si se tratara de un protocolo diferente de IPv4.

El campo **DS (Differentiated Services)** se utiliza para especificar parámetros de Calidad de Servicio de acuerdo con la arquitectura Diffserv que veremos más adelante.

El campo **etiqueta de flujo** permite identificar los paquetes que pertenecen a una sesión concreta entre dos hosts, normalmente para solicitar un trato preferente o una determinada Calidad de Servicio. A este campo y al concepto de flujo nos referiremos más adelante cuando hablemos de Calidad de Servicio.

El campo **longitud de carga útil** indica el tamaño del paquete en bytes, excluidos los 40 bytes de la cabecera. El valor máximo es 65535. El paquete máximo es pues 65575.

El campo **siguiente cabecera** indica si esta cabecera esta seguida por alguna de las cabeceras opcionales. Si no hay cabeceras opcionales este campo indica el protocolo del nivel de transporte al que pertenece este paquete, para lo cual utiliza los mismos códigos numéricos que en IPv4 (ver tabla 3.2).

El campo **límite de saltos** equivale al antiguo campo TTL, pero se ha decidido ponerle un nombre que refleje su uso real. Como en IPv4 el campo tiene 8 bits, por lo que el máximo número de saltos que puede especificarse es también 255. Algunos opinaban que este valor era demasiado bajo y que en algunos casos podría plantear problemas. Actualmente ya hay situaciones en la Internet donde se esta próximo a los 64 saltos.

Por último, los campos **dirección de origen** y **dirección de destino** corresponden a las direcciones de 16 bytes que hemos descrito.

Los campos de la cabecera IPv4 que han desaparecido de la IPv6 son los siguientes:

IHL (Internet Header Length): este campo no existe pues la cabecera tiene ahora una longitud fija.

El campo *protocolo* no aparece pues su función la desempeña el campo *siguiente cabecera*.

El campo *checksum* se ha suprimido ya que no se consideró justificada la pérdida de rendimiento que supone el cálculo del checksum en cada salto ante la rara eventualidad de que se produzca un error en el nivel de red, tomando en cuenta que normalmente tanto el nivel de enlace como el de transporte realizan su propia comprobación de errores, por lo que realmente el checksum del datagrama IP no protege de errores de transmisión sino solamente de errores internos que se puedan producir en el router (por fallos en la memoria RAM por ejemplo). La supresión de este campo fue algo muy debatido durante la elaboración del nuevo estándar.

Todos los campos relativos a fragmentación han desaparecido de la cabecera porque en IPv6 la fragmentación se controla mediante cabeceras adicionales o extendidas; además en IPv6 todos los nodos han de aceptar paquetes de 1280 bytes como mínimo y sólo se permite la fragmentación en origen, es decir el emisor debe generar datagramas suficientemente pequeños para que puedan llegar a su destino sin fragmentaciones adicionales. Normalmente el emisor realizará el *Path MTU Discovery*, como ya era habitual en muchas implementaciones de IPv4.

3.6.3 Cabeceras extendidas

En IPv4 la cabecera puede contener algunos campos opcionales, principalmente para diagnóstico de problemas de routing, pero debido a su escasa longitud esos campos son poco utilizados. El sistema es demasiado rígido y poco eficiente, ya que los campos opcionales tienen que ser procesados en todos los routers del trayecto. En IPv6 se ha habilitado un mecanismo más flexible y eficiente; las cabeceras opcionales aparecen como cabeceras adicionales al final de la cabecera estándar; su presencia queda indicada por el campo *siguiente cabecera*, que en caso de que no haya opciones inidcará el protocolo de transporte (normalmente TCP o UDP). De esta forma los campos opcionales en IPv6 pueden extenderse cuando se considere necesario. Además se prevé un mecanismo por el cual se puede indicar si las opciones deben ser procesadas por todos los routers del trayecto o solo por el ultimo, lo cual permite una

reducción significativa del trabajo a desarrollar por los routers intermedios cunado se trata de una opción que solo debe ser procesada por el nivel de red en el host de destino.

La cabecera *salto-a-salto* indica información que debe ser examinada por todos los routers por los que transite este datagrama. Hasta ahora solo se le ha definido a esta cabecera una opción, que permite especificar datagramas de longitud superior a 64 KB; estos datagramas (que pueden llegar a tener hasta 4 GB) se conocen como *jumbogramas*.

La cabecera *routing* realiza las funciones combinadas del strict source routing y loose source routing de IPv4; el máximo número de direcciones que pueden especificarse es de 24.

La cabecera *fragment* se utiliza cuando hay que fragmentar un datagrama; el mecanismo utilizado es muy similar al de IPv4 (campos Identificación, Desplazamiento del fragmento y MF), con la diferencia de que en IPv6 solo se permite la fragmentación en origen. De esta forma se simplifica notablemente la complejidad de proceso en los routers.

La cabecera *authentication* permite el uso de técnicas criptográficas para incorporar un mecanismo de firma digital por el cual el receptor del paquete puede estar seguro de la autenticidad del emisor.

La cabecera *encrypted security payload* permite el envío de información encriptada para que solo pueda ser leída por el destinatario. Evidentemente la encriptación afecta únicamente a los datos, no incluye la cabecera del datagrama puesto que ésta ha de ser leída e interpretada por cada router por el que pasa.

3.7 EJERCICIOS

1. Indique si es verdadera o falsa cada una de las siguientes afirmaciones:

a) Cuando un router IP detecta un error en el checksum de un datagrama envía un mensaje ICMP al emisor notificándole para que lo reenvíe.

b) Una de las ventajas de IP es su capacidad de reencaminar el tráfico por rutas alternativas en caso de problemas en la red; sin embargo esta característica solo esta disponible cuando se utiliza un protocolo de routing, no cuando se emplean rutas estáticas.

c) El protocolo OSPF permite crear niveles jerárquicos dentro de un sistema autónomo.

d) Los protocolos de routing empleados dentro de un sistema autónomo, tales como RIP u OSPF, no son apropiados para su utilización entre sistemas autónomos debido a los mayores retardos de las comunicaciones en este caso, al tratarse normalmente de enlaces que cubren grandes distancias.

e) En IP los datagramas pueden llegar desordenados y son en todo caso los niveles superiores (TCP por ejemplo) los que se ocupan de reordenar los datos; sin embargo cuando hay fragmentación en ruta los fragmentos deben llegar necesariamente en orden puesto que el datagrama original se ha de recomponer a nivel IP al no tener el nivel de transporte conocimiento de la fragmentación, por lo que carece en este caso de mecanismos para detectar los cambios de orden.

f) Cuando se crea una subred con direcciones IP el numero de hosts es siempre dos menos que el espacio de direcciones correspondiente, es decir, si se utilizan *n* bits para la parte host el numero de hosts posibles es de 2^n-2.

g) El protocolo BOOTP permite averiguar la dirección MAC que corresponde a una dirección IP dada.

h) El protocolo ARP solo es necesario en redes broadcast.

2. Suponga que se tiene una red formada por tres routers, X, Y y Z, unidos entre sí mediante tres líneas de 64 Kb/s en topología triangular. ¿Que diferencia supondría el uso de routing dinámico o estático desde el punto de vista de la eficiencia y fiabilidad de la comunicación entre ellos?

3. Un datagrama IP que utiliza la opción *source routing* llega a un punto en la red en el que tiene que ser fragmentado. Tendrá este campo opcional que copiarse en cada uno de los fragmentos, o bastará con que se copie en el primero de ellos únicamente?

4. En IPv6 se modifica sustancialmente la cabecera de los datagramas IP, debido especialmente al cambio en la longitud de las direcciones de 32 a 128 bits. Como afectará esto al funcionamiento de los puentes transparentes? Y al de los puentes con encaminamiento desde el origen?

5. Diga cual(es) de los siguientes protocolos permite(n) realizar asignación dinámica de direcciones IP:

A: _ BOOTP B: _ DHCP C: _ RARP
D: _ ARP E: _ PPP F: _ SLIP

6. Un sistema autónomo es:

- A: _ Una red que utiliza protocolo TCP/IP
- B: _ Un conjunto de routers que intercambian información de routing mediante un protocolo común
- C: _ Un conjunto de routers y hosts con una adminstración centralizada
- D: _ La parte de la Internet que corresponde a un Proveedor

7. La fragmentación en ruta es una causa de ineficiencia en redes IP. Algunos hosts intentan evitarla aplicando la técnica conocida como descubrimiento de la MTU del trayecto ('path MTU discovery') consistente en enviar datagramas de prueba con el bit DF (Don't Fragment) puesto; por los mensajes de error recibidos es posible averiguar el tamaño máximo de datagrama que se puede utilizar.

Algunos hosts cuando utilizan el 'path MTU discovery', una vez han averiguado la MTU del trayecto realizan dos acciones no habituales:

- Siguen poniendo a 1 el bit DF en todos los datagramas que envían.
- De vez en cuando envían un datagrama de un tamaño mayor que el máximo permitido por el trayecto, con el bit DF puesto.

¿Podría dar alguna razón que justifique estos comportamientos?

8. A continuación aparece el fichero de configuración de un router. Dibuje el croquis correspondiente indicando la dirección IP de cada interfaz, que rangos de direcciones está accesible por cada una de ellas, y todo lo que pueda inferir sobre la topología de la red a partir de la información facilitada. Realice los comentarios que considere oportunos sobre los comandos y su significado.

Notas:

- La configuracion presentada es ficticia, no corresponde a ninguna configuración real.
- Se utilizan únicamente rutas estáticas.
- En la configuración se utilizan máscaras de tamaño variable.

```
----------------------  FICHERO DE CONFIGURACION  ----------------
-

version 11.1
hostname router
!
ip subnet-zero
!
interface Ethernet0
 description conexion red local UV
 ip address 147.156.1.11 255.255.128.0
!
interface Ethernet1
 description conexion aula SIUV
 ip address 147.156.147.129 255.255.255.224
!
interface serial0
 description primera conexion Magisterio
 ip address 192.168.1.1 255.255.255.252
 bandwidth 64
!
interface serial1
 description segunda conexion Magisterio
 ip address 192.168.1.5 255.255.255.252
 bandwidth 64

interface serial2
 description conexion UJI
 ip address 130.206.211.5 255.255.255.252
 bandwidth 2048
!
interface serial3
 description salida al resto del mundo
 ip address 130.206.211.2 255.255.255.252
 bandwidth 2048
!
! Ruta host
ip route 130.206.211.174 255.255.255.255 147.156.147.130
! Subred Magisterio (balance de trafico entre ambas lineas)
ip route 147.156.198.0 255.255.254.0 192.168.1.2
ip route 147.156.198.0 255.255.254.0 192.168.1.6
! Clase C del IATA.CSIC (Paterna)
ip route 193.145.246.0 255.255.255.0 147.156.15.9
! Red de la UJI
ip route 150.208.0.0 255.255.0.0 130.206.211.6
! Ruta por defecto
ip route 0.0.0.0 0.0.0.0 130.206.211.1
! Ruta loopback
ip route 127.0.0.1 255.255.255.255 Null0
```

9. Suponga que en la configuración del ejercicio anterior se suprime la línea:

```
ip route 0.0.0.0 0.0.0.0 130.206.211.1
```

Indique que ocurriría en este caso con un datagrama que llegara a este router por la interfaz Ethernet1 que tuviera como dirección de origen la 147.156.147.132 y como dirección de destino la 138.247.12.32.

¿Y que pasaría si llegara un datagrama por la interfaz serial3 con dirección de origen la 138.247.12.32 y de destino la 147.156.147.132?

10. Una empresa tiene una red local conectada a Internet mediante un router que solo posee dos interfaces, una ethernet y una puerta serie conectada a una línea punto a punto de 2048 Kb/s. La configuración del router es la siguiente:

```
version 11.1
hostname router
!
interface Ethernet0
 description conexion red local
 ip address 190.190.190.1 255.255.255.0
!
interface serial0
 description salida al resto del mundo
 ip address 192.168.200.6 255.255.255.252
 bandwidth 2048
!
! Ruta por defecto
ip route 0.0.0.0 0.0.0.0 192.168.200.5
! Ruta loopback
ip route 127.0.0.1 255.255.255.255 Null0
```

Se desea sustituir el router por otro, también con una interfaz serie y una ethernet.

Para hacer el cambio de la manera mas sencilla posible el administrador de la red ha preparado el nuevo router con una configuración idéntica al antiguo; de esta forma los usuarios con sus ordenadores configurados con la dirección 190.190.190.1 como salida (gateway) al exterior no tendrán que modificar nada. Calcula el administrador que con la ayuda de otra persona puede sustituir físicamente un router por el otro en cinco segundos, tiempo que espera pase desapercibido por sus usuarios, acostumbrados al lento funcionamiento de la Internet (el funcionamiento no orientado a conexión de IP permite hacer estas maniobras sin perturbar la red de forma apreciable). El administrador decide pues hacer el cambio en horas normales de funcionamiento.

El cambio se efectúa según lo previsto, y después de hacerlo el administrador va a un PC próximo, lo enciende y comprueba que se puede conectar a Internet a través del nuevo router sin problemas, por lo que considera terminada con éxito la tarea. Justo en ese momento empieza a recibir multitud de llamadas de los usuarios que se quejan de no poder salir a Internet. El administardor revisa y comprueba la configuración y todas las conexiones, pero no encuentra ningún error; pasados unos minutos, y sin que el administrador haya hecho nada, los usuarios empiezan a poder conectar de nuevo con el exterior y poco a poco el teléfono deja de sonar.

Podría decir por que ha fallado el plan? Como debería haber procedido el administrador para hacer el cambio en horas normales de funcionamiento sin afectar el servicio de forma apreciable?

11. La siguiente es una configuración hipotética de un router:

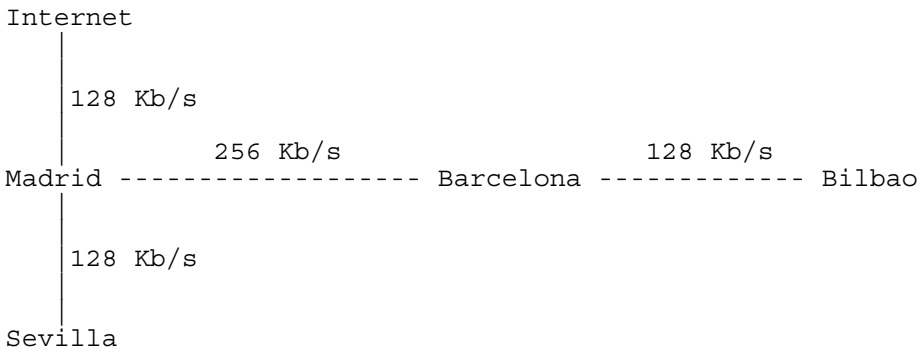
```
Version 11.1
hostname centro-remoto
ip subnet-zero
ip classless
!
interface Ethernet0
 description conexion LAN
 ip address 194.125.1.63 255.255.255.192
!
interface Ethernet1
 description conexion LAN
 ip address 195.0.0.195 255.255.255.128
!
interface serial0
 description conexion WAN
 ip address 195.100.1.2 255.255.255.252
!
interface serial1
 description conexion WAN
 ip address 197.160.1.1 255.255.255.252
!
ip route 157.34.33.0  255.255.255.255 195.0.0.199
ip route 160.87.34.0  255.255.248.0   195.100.1.1
ip route 198.0.0.0    255.254.0.0     197.160.1.2
ip route 0.0.0.0      0.0.0.0         195.100.1.1
```

Comente el significado de los comandos que aparecen.

Dibuje el esquema de la red correspondiente.

En los parámetros de algunos comandos existen valores inadecuados. Localícelos y coméntelos.

12. Una empresa de consultoría informática tiene su oficina principal en Madrid, con sucursales en Barcelona, Bilbao y Sevilla. Cada una de las cuatro oficinas tiene una red local basada en los protocolos TCP/IP. Se desea unirlas todas entre sí, para lo cual se realiza un estudio necesidades y se evalúa el costo de diversas alternativas. Además se quiere dar acceso a Internet a todas las oficinas. Como resultado de todo ello se decide montar una red con la siguiente topología:



Se prevé un máximo de 100 ordenadores en la oficina de Madrid, 50 en la de Barcelona, 25 en la de Bilbao y 20 en la de Sevilla. Se necesita que todos los ordenadores tengan acceso directo a Internet, es decir, tengan números IP públicos o 'legales'. Para ello la empresa ha obtenido del NIC la red 194.100.100.0.

Se le pide que:

a) Diseñe un esquema de reparto de direcciones IP entre las diferentes oficinas que satisfaga los requerimientos planteados.

b) Rellene los campos que faltan en la siguiente configuración, correspondiente al router Cisco de Madrid:

```
Version 11.1
hostname Router-Madrid
ip subnet-zero
!
interface Ethernet0
description conexion LAN Madrid

ip address    ............   ............
!
interface serial0
description conexion WAN Internet
bandwidth 128

ip address    ............   ............
!
interface serial1
description conexion WAN Barcelona
bandwidth 256

ip address    ............   ............
!
interface serial2
description conexion WAN Sevilla
bandwidth 128

ip address    ............   ............
!
!
ip route ............  ............   ............

ip route ............  ............   ............

ip route ............  ............   ............

ip route      0.0.0.0  ............   ............

ip route    127.0.0.1  ............   ............
```

13. Una empresa farmacéutica tiene una red local Ethernet con varios ordenadores, y tiene contratada una conexión a Internet con el proveedor X por la cual todos los ordenadores pueden enviar y recibir del exterior datagramas IP, sin restricciones. Los ordenadores tienen números de la red 199.199.199.0 (máscara 255.255.255.0) asignada por el proveedor a la empresa, dentro del rango que le corresponde. La conexión física se realiza mediante una línea punto a punto, para lo cual en la red local hay un router con una interfaz ethernet y una serie. El router tiene una segunda interfaz serie actualmente no utilizada.

El proveedor X ha ido ampliando el número de usuarios sin aumentar de forma proporcional la infraestructura, con lo cual ha ido bajando la calidad de servicio que ofrece. Como consecuencia de ello la empresa decide contratar una segunda conexión a Internet con el proveedor Y, mas caro pero que ofrece un servicio de mayor calidad. El proveedor Y asigna a la empresa la red 200.200.200.0 (máscara 255.255.255.0), y la conexión se realiza también mediante una línea punto a punto, como en el caso del proveedor X.

A pesar de tener una nueva conexión con el proveedor Y se quiere mantener la conexión al proveedor X dado su bajo costo, para utilizarla en servicios poco críticos, como por ejemplo las news. Algunas máquinas de la red local mantendrán direcciones de la red 199.199.199.0, y deberán utilizar el proveedor X, mientras que a otras se les asignarán direcciones de la red 200.200.200.0 y deberán utilizar el proveedor Y.

Diga si para efectuar la conexión al proveedor Y es posible utilizar la interfaz serie que queda libre en el router existente, o si por el contrario es necesario adquirir un segundo router para dicha conexión. Explique su respuesta.

Escriba un ejemplo de la configuración que habría(n) de tener el (los) router(s) necesario(s) para realizar las conexiones indicadas. En todo momento se utiliza routing estático. Recuerde que una interfaz puede tener mas de una dirección IP.

3.8 SOLUCIONES

S1.-

a) **Falsa**. El protocolo IP no implementa ningún mecanismo de reenvío de la infromación. En caso de error en la comprobación del checksum el datagrama es simplemente descartado.

b) **Verdadera**. En caso de rutas estáticas sería necesaria una reconfiguración manual de los equipos para modificar la ruta utilizada.

c) **Verdadera**. Los niveles jerárquicos se establecen mediante las denominadas *áreas*.

d) **Falsa**. Entre sistemas autónomos (ASes) se requieren otros protocolos de routing debido a la necesidad de establecer reglas de tipo 'político', por ejemplo un proveedor que no quiere ser utilizado como vía de tránsito para la comunicación entre otros dos proveedores con los cuales está conectado.

e) **Falsa**. Los datagramas en IP siempre pueden llegar desordenados, independientemente de que se trate de fragmentos o de datagramas enteros. En caso de fragmentación el campo offset permite al nivel IP del receptor reordenar adecuadamente los fragmentos.

f) **Verdadera**. Se reservan dos direcciones para usos especiales: la que tiene el campo host todo a ceros para indicar la subred, y la que lo tiene todo a unos para transmisiones broadcast dentro de la subred. Esta restricción se aplica siempre, incluso cuando se especifica subnet-zero; subnet-zero permite eludir una restricción similar que normalmente se aplica al campo subred.

g) **Falsa**. Es lo contrario: BOOTP permite averiguar la dirección IP que corresponde a una dirección MAC (normalmente la de la máquina que está haciendo la consulta).

h) **Verdadera**. En redes punto a punto cada ordenador sabe por sus tablas de routing por donde ha de acceder a una dirección de red determinada.

S2.-

Con routing dinámico el tráfico podría reencaminarse por la ruta alternativa en caso de caída de uno de los tres enlaces. Además el tráfico se podría repartir de forma mas o menos homogénea entre las dos rutas posibles, con lo que en el caso límite se podría llegar a tener una capacidad de 128 Kb/s entre dos nodos en el supuesto de que el otro no generara ni recibiera ningún tráfico. Ninguna de estas cosas sería posible con routing estático. Por otro lado el uso de routing dinámico provocaría una pequeña proporción de tráfico en la red debido a la información intercambiada por los routers, aunque esto se vería sobradamente compensado por las ventajas antes mencionadas. Así pues el routing dinámico permitiría tener una red mas fiable y eficiente.

S3.-

Tendrá que copiarse en todos, ya que cada fragmento puede seguir en principio una ruta diferente y solo así podremos estar seguros de que se respeta la ruta marcada.

S4.-

No afecta en absoluto a ninguno de los dos puesto que tanto los puentes transparentes como los de encaminamiento desde el origen utilizan direcciones MAC, que son direcciones a nivel de enlace. Las direcciones IPv4 o IPv6 son direcciones a nivel de red, que van en la cabecera del datagrama, y no son visibles para los puentes ya que forman parte de lo que se consideran 'datos'.

S5.-

DHCP y PPP

S6.-

B: Un conjunto de routers que intercambian información de routing mediante un protocolo común

S7.-

En IP la ruta seguida por los datagramas puede variar, ya que las tablas de encaminamiento de los routers pueden verse modificadas, bien porque se esté utilizando un protocolo de routing dinámico que reacciona a los cambios que se producen en la red, o porque utilizando routing estático se modifiquen 'en caliente' las tablas de encaminamiento de algún router del trayecto. Por tanto es posible que no todos los datagramas de una sesión entre dos hosts discurran por la misma ruta.

Poniendo a 1 el bit DF en todos los datagramas el host emisor comprueba si algún datagrama es fragmentado debido a alteraciones en la ruta que causen una reducción del MTU, cosa que de otro modo podría pasar desapercibida; al estar puesto el bit DF el intento de fragmentación provocaría que el datagrama fuera rechazado por el router, devolviendo un mensaje ICMP al host emisor que así se daría cuenta de la situación.

Análogamente la emisión cada cierto tiempo de un datagrama con el bit DF puesto de tamaño mayor que la MTU negociada, permite al host emisor detectar si la MTU del trayecto ha aumentado debido a las modificaciones que se puedan haber producido en la ruta durante la sesión.

S8.-

Se intercalan los comentarios en el fichero de configuración.

```
-----------------------  FICHERO DE CONFIGURACION  -------------
----

version 11.1
hostname router
```

Especifica la versión de software utilizada y el nombre del router

```
!
ip subnet-zero
```

Indica que se permite el uso no estándar de poner completamente a ceros o a unos el campo subred

```
!
```

```
interface Ethernet0
 description conexion red local UV
 ip address 147.156.1.11 255.255.128.0
```

Indica que la interfaz Ethernet0 tiene la dirección IP 147.156.1.11 y que esta interfaz está integrada en (y por tanto da acceso a) una subred formada por las direcciones 147.156.0.0 hasta 147.156.127.255

```
!
interface Ethernet1
 description conexion aula SIUV
 ip address 147.156.147.129 255.255.255.224
```

La interfaz Ethernet1 tiene la dirección IP 147.156.147.129 y está integrada en una subred formada por las direcciones 147.156.147.128 hasta 147.156.147.159

```
!
interface serial0
 description primera conexion Magisterio
 ip address 192.168.1.1 255.255.255.252
 bandwidth 64
```

La interfaz serial0 tiene la dirección IP 192.168.1.1 y está integrada en una subred formada por las direcciones 192.168.1.0 hasta 192.168.1.3; corresponde a una línea de 64 Kb/s que en el otro lado tiene la dirección 192.168.1.2 (las direcciones 0 y 3 están reservadas pues corresponden al campo host todo a ceros y todo a unos, respectivamente).

```
!
interface serial1
 description segunda conexion Magisterio
 ip address 192.168.1.5 255.255.255.252
 bandwidth 64
```

La interfaz serial1 tiene la dirección IP 192.168.1.5 y está integrada en una subred formada por las direcciones 192.168.1.4 hasta 192.168.1.7. Corresponde a una línea de 64 Kb/s que en el otro lado tiene la dirección 192.168.1.6.

```
interface serial2
 description conexion UJI
 ip address 130.206.211.5 255.255.255.252
 bandwidth 2048
!
```

La interfaz serial2 tiene la dirección IP 130.206.211.5 y está integrada en una subred formada por las direcciones 130.206.211.4 hasta 130.206.211.7. Corresponde a una línea de 2048 Kb/s que en el otro lado tiene la dirección 130.206.211.6.

```
interface serial3
 description salida al resto del mundo
 ip address 130.206.211.2 255.255.255.252
 bandwidth 2048
```

La interfaz serial3 tiene la dirección IP 130.206.211.2 y está integrada en una subred formada por las direcciones 130.206.211.0 hasta 130.206.211.3. Corresponde a una línea de 2048 Kb/s que en el otro lado tiene la dirección 130.206.211.1.

```
!
! Ruta host
ip route 130.206.211.174 255.255.255.255 147.156.147.130
```

Esta es una ruta host, que indica la ruta a seguir para el destino 130.206.211.174; las rutas host tienen máxima prioridad, por lo que esta se seguirá independientemente de lo que indiquen otras rutas declaradas para la red 130.206 o para sus posibles subredes.

```
! Subred Magisterio (balance de trafico entre ambas lineas)
ip route 147.156.198.0 255.255.254.0 192.168.1.2
ip route 147.156.198.0 255.255.254.0 192.168.1.6
```

Estas dos rutas indican que todos los datagramas destinados a direcciones en el rango 147.156.198.0-147.156.199.255 deberán enviarse por las interfaces serial0 y serial1, realizando balance de tráfico entre ambas líneas.

```
! Clase C del IATA.CSIC (Paterna)
ip route 193.145.246.0 255.255.255.0 147.156.15.9
```

Indica que la red clase C 193.145.246.0 está accesible a través de la dirección 147.156.15.9. Por información dada anteriormente el router sabe que dicha dirección está accesible a través de la interfaz Ethernet0.

```
! Red de la UJI
ip route 150.208.0.0 255.255.0.0 130.206.211.6
```

La red 150.208.0.0 (una clase B completa) está accesible a través de la dirección 130.206.211.6; por la información anterior el router ya sabe que dicha dirección corresponde al otro extremo de la interfaz serial2.

```
! Ruta por defecto
ip route 0.0.0.0 0.0.0.0 130.206.211.1
```

Esta instrucción indica que todo datagrama que no haya podido ser enrutado aplicando alguna de las rutas explícitas (de red o de host) debe enviarse a la dirección 130.206.211.1; por la información dada en las instrucciones anteriores el router ya sabe que dicha dirección corresponde al otro extremo de la línea a la que está conectada la interfaz serial3.

```
! Ruta loopback
ip route 127.0.0.1 255.255.255.255 Null0
```

Esta instrucción, que corresponde a una ruta host, indica que cualquier datagrama dirigido a la dirección 127.0.0.1 deberá descartarse, de acuerdo con lo que especifican los estándares Internet.

Obsérvese que para su aplicación las instrucciones `ip route` se clasifican previamente en tres categorías:

- Rutas *host*: se caracterizan por tener la máscara toda a unos (255.255.255.255)
- Rutas *red*: se caracterizan por tener máscaras con parte a ceros y parte a unos
- Ruta *por defecto*: se caracteriza por tener la máscara toda a ceros (0.0.0.0)

Independientemente de su ubicación en el fichero de configuración siempre se aplican por este orden: primero las reglas correspondientes a rutas host, en segundo lugar las rutas de red, y por último la ruta por defecto.

Obsérvese que las máscaras que acompañan a la dirección IP de cada interfaz indican de manera implícita la red o subred en la que dicha interfaz está integrada; por tanto no es necesario incluir una instrucción `ip route` para indicar por donde se debe encaminar el tráfico dirigido a dichas direcciones.

S9.-

En el caso de un datagrama de 147.156.147.132 a 138.247.12.32 entraría en el router por la interfaz Ethernet1 y el router buscaría alguna entrada 'ip route' que le permitiera saber por donde enviarlo. Al no haber ninguna entrada que incluya dicha dirección (pues se ha suprimido la ruta por defecto) el router descartará el datagrama y enviará un comando ICMP DESTINATION

UNREACHABLE al host origen, esto es a la dirección 147.156.147.132. Obsérvese que todas las decisiones de encaminamiento se toman en base a la dirección de destino; normalmente el router solo utilizará la dirección de origen contenida en el datagrama cuando tenga que devolver al emisor algún mensaje ICMP.

En el caso del datagrama de 138.247.12.32 dirigido a 147.156.147.132 sería encaminado correctamente a la interfaz Ethernet1, ya que el router sabe que dicha dirección debe estar accesible a través de esa interfaz.

Tenemos en este caso un ejemplo de rutas asimétricas: mientras que en un sentido no es posible entregar los datagramas en el otro la red funciona sin problemas. Sin embargo la mayoría de las aplicaciones requieren comunicación full dúplex para su correcto funcionamiento, por lo que en la práctica un error como este casi siempre habría supuesto la imposibilidad de comunicación entre ambas direcciones.

S10.-

El problema ocurrido se debe a que en el momento de reemplazar el router viejo por el nuevo los ordenadores que estaban trabajando con la Internet tenían en su memoria la tabla ARP cache, esto es la equivalencia entre direcciones IP y direcciones MAC; al cambiar el router viejo por el nuevo se mantenía la dirección IP, pero se modificaba la dirección MAC puesto que cada interfaz Ethernet tiene una dirección diferente, pero los usuarios que utilizaban la tabla ARP existente en su cache intentaban salir al exterior buscando el router viejo, ahora desconectado de la red.

Para solucionarlo se podía haber hecho la sustitución en los siguientes pasos:

I. Configurar el nuevo router con idéntica configuración al viejo, pero sin conectarlo a la red.

II. Cargar en el router viejo la siguiente configuración (hemos marcado con negrita las partes modificadas o añadidas respecto a la configuración original):

```
version 11.1
hostname router
!
interface Ethernet0
 description conexion red local
 ip address 190.190.190.100 255.255.255.0
!
interface serial0
shutdown
 description salida al resto del mundo
 ip address 192.168.200.6 255.255.255.252
 bandwidth 2048
!
! Ruta por defecto
ip route 0.0.0.0 0.0.0.0 190.190.190.1
! Ruta loopback
ip route 127.0.0.1 255.255.255.255 Null0
```

III. Inmediatamente después enchufar a la red Ethernet el nuevo router, desenchufar la línea serie del router viejo y enchufarla al router nuevo. Mantener enchufado a la Ethernet el router viejo hasta pasados quince minutos.

De esta forma el router nuevo aceptaría todas las conexiones que se produjeran a partir de ese momento, mientras que los ordenadores que ya estuvieran trabajando anteriormente y que tuvieran en su tabla ARP la dirección MAC del router viejo seguirían accediendo a éste, el cual

reencaminaria los datagramas a su nuevo destino. Pasados unos minutos las entradas de la tabla ARP se habrán renovado, por lo que a partir de ese momento ya sería posible desconectar completamente el router viejo.

S11.-

```
Version 11.1
```

Indica la versión del software utilizado

```
hostname centro-remoto
```

Indica el nombre del router

```
ip subnet-zero
```

Indica que se puede infringir la regla que prohibe utilizar los valores extremos (todo ceros o todo unos) del campo subred.

```
ip classless
```

Indica que el software del router soporta CIDR (Classless InterDomain Routing)

```
!
interface Ethernet0
 description conexion LAN
 ip address 194.125.1.63 255.255.255.192
```

Define la interfaz de conexión a red ethernet. A la vista de la máscara utilizada esta ethernet comprende un rango de 64 direcciones que por la dirección de esta interfaz se deduce que debe ser del 194.125.1.0 al 194.125.1.63; de estas solo 62 son aprovechables, ya que la primera y la última estan reservadas (subred y broadcast). La dirección IP de esta interfaz es incorrecta ya que corresponder a la dirección broadcast de la subred (campo host todo a unos). La interfaz debería tener una dirección en el rango 194.125.1.1 a 194.125.1.62.

```
!
interface Ethernet1
 description conexion LAN
 ip address 195.0.0.195 255.255.255.128
!
```

Interfaz de conexión a red ethernet, con un rango de 128 direcciones: 195.0.0.128 a 195.0.0.255.

```
interface serial0
 description conexion WAN
 ip address 195.100.1.2 255.255.255.252
!
```

Interfaz de conexión WAN. Subred formada por cuatro direcciones, de la 195.100.1.0 a la 195.100.1.3. Como solo dos direcciones son aprovechables la dirección del otro lado de la línea (del otro router) ha de ser la 195.100.1.1.

```
interface serial1
 description conexion WAN
 ip address 197.160.1.1 255.255.255.252
!
```

Interfaz WAN. Rango 197.160.1.0 a 197.160.1.3. La interfaz del otro lado tiene que ser 197.160.1.2.

```
ip route 157.34.33.0  255.255.255.255 195.0.0.199
```

Ruta host (máscara toda a unos). Los datagramas dirigidos a 157.34.33.0 se enviarán a 195.0.0.199, que es un host conectado a la ethernet1.

```
ip route 160.87.34.0  255.255.248.0   195.100.1.1
```

Esta parece ser una subred de una clase B que esta accesible a través de la interfaz serial0. Por la máscara utilizada se emplean 5 bits para la parte subred y 11 para la parte host, sin embargo la dirección 160.87.34.0 no tiene a cero los últimos 11 bits, por lo que no es una dirección de red válida para dicha máscara. Un valor válido habría sido por ejemplo 160.87.32.0.

```
ip route 198.0.0.0     255.254.0.0      197.160.1.2
```

Esta ruta dirige a la interfaz serial1 todos los datagramas que lleven una dirección de destino comprendida en el rango de 198.0.0.0 a 198.1.255.255. Se está haciendo agregación de direcciones (mediante CIDR) puesto que se trata de dos redes clase C diferentes.

```
ip route 0.0.0.0       0.0.0.0          195.100.1.1
```

Esta es la ruta por defecto, que indica que cualquier datagrama cuya dirección no sea resuelta por la rutas anteriores deberá ser enviado por la interfaz serial0.

S12.-

a)

Dado que el número de direcciones de una subred IP ha de ser necesariamente potencia entera de 2 (para que pueda identificarse mediante una máscara común), las subredes mínimas que podemos asignar en este caso son las siguientes:

Oficina	Rango	Rango útil	Máscara
Madrid	128	126	255.255.255.128
Barcelona	64	62	255.255.255.192
Bilbao	32	30	255.255.255.224
Sevilla	32	30	255.255.255.224

Para hacer esto con la red clase C de que disponemos es preciso crear máscaras de tamaño variable y utilizar todas las subredes, incluídas las que tienen el campo subred todo a ceros y todo a unos, es decir suprimir la restricción subred cero. El reparto podría ser por ejemplo el siguiente:

Oficina	Subred	Máscara	Rango	Direcciones útiles
Madrid	194.100.100.0	255.255.255.128	194.100.100.0-127	126
Barcelona	194.100.100.128	255.255.255.192	194.100.100.128-191	62
Bilbao	194.100.100.192	255.255.255.224	194.100.100.192-223	30
Sevilla	194.100.100.224	255.255.255.224	194.100.100.224-255	30

b)

Una posible configuración para el router de Madrid sería la siguiente (la parte añadida aquí aparece en negrita):

```
Version 11.1
hostname Router-Madrid
ip subnet-zero
!
interface Ethernet0
description conexion LAN Madrid

ip address 194.100.100.1 255.255.255.128
```

```
!
interface serial0
description conexion WAN Internet
bandwidth 128

ip address 192.168.1.2 255.255.255.252
!
interface serial1
description conexion WAN Barcelona
bandwidth 256

ip address 192.168.2.1 255.255.255.252
!
interface serial2
description conexion WAN Sevilla
bandwidth 128

ip address 192.168.3.1 255.255.255.252
!
! Ruta a Barcelona
ip route 194.100.100.128 255.255.255.192 192.168.2.2
! Ruta a Bilbao
ip route 194.100.100.192 255.255.255.224 192.168.2.2
! Ruta a Sevilla
ip route 194.100.100.224 255.255.255.224 192.168.3.2
! Ruta a Internet
ip route 0.0.0.0  0.0.0.0 192.168.1.1
! Ruta loopback
ip route 127.0.0.1  255.255.255.255 Null0
```

Obsérvese que para las interfaces WAN se han utilizado direcciones privadas según el RFC1918 y máscaras 255.255.255.252, que dan solamente 2 direcciones útiles.

S13.-

Suponemos que para la red 199.199.199.0 el router es la dirección 199.199.199.1, y para la 200.200.200.0 es la 200.200.200.1.

Solución utilizando el router existente:

```
hostname router
interface Ethernet0
description conexion LAN
ip address 199.199.199.1 255.255.255.0
ip address 200.200.200.1 255.255.255.0 secondary
interface serial0
description linea con el proveedor X
ip address 192.168.1.5 255.255.255.252
interface serial1
description linea con el proveedor Y
ip address 192.168.2.5 255.255.255.252
ip route 0.0.0.0 0.0.0.0 192.168.1.6
ip route 0.0.0.0 0.0.0.0 192.168.2.6
```

Esta configuración produciría que los datagramas de salida se repartieran entre las dos interfases por igual, balanceando tráfico, pero no conseguiría el efecto buscado de utilizar el proveedor X para la red 199.199.199.0 y el Y para la 200.200.200.0. Además, datagramas provinientes de la red 199.199.199.0 se enviarían por el proveedor Y y viceversa. Podría ser que los routers de los proveedores X e Y tengan filtros de entrada para impedir la entrada por estas interfases de datagramas provinientes de redes que no sean las suyas.

La comunicación entre las dos redes (199.199.199.0 y 200.200.200.0) se realizaría a través de la interfaz Ethernet del router. En realidad dicha interfaz solo actuaría en el primer paquete enviado de cada lado, ya que enviaría a los hosts correspondientes un mensaje ICMP REDIRECT

indicando que pueden comunicar por la LAN. Sin embargo si el router careciera de una de las dos direcciones IP la comunicación sería imposible. (suponemos que los hosts de las dos redes solo conocen su router por defecto).

Solución utilizando dos routers:

```
hostname router1
interface ethernet0
description conexion LAN
ip address 199.199.199.1 255.255.255.0
ip address 200.200.200.2 255.255.255.0 secondary
interface serial0
description linea con el proveedor X
ip address 192.168.1.5 255.255.255.252
ip route 0.0.0.0 0.0.0.0 192.168.1.6

hostname router2
interface ethernet0
description conexion LAN
ip address 200.200.200.1 255.255.255.0
ip address 199.199.199.2 255.255.255.0 secondary
interface serial0
description linea con el proveedor Y
ip address 192.168.2.5 255.255.255.252
ip route 0.0.0.0 0.0.0.0 192.168.2.6
```

En este caso los host de la red 199.199.199.0 usarán el router 1, que accede a la Internet por el proveedor X, y los de la red 200.200.200.0 usarán el router 2, que accede a través del proveedor Y.

La comunicación entre las dos redes se hará usando el router1 para el sentido 199 -> 200 y el router2 para el sentido inverso. Análogamente al caso anterior se enviaría un ICMP REDIRECT para informar a los hosts que pueden comunicar directamente, sin necesidad de utilizar el router.

En caso de que alguno de los dos routers no tuviera la dirección secundaria la comunicación en el sentido correspondiente se haría a través de la Internet. Por ejemplo supongamos que se suprime la dirección secundaria en router1, en tal caso los datagramas enviados por hosts de la red 199 hacia la 200 viajarían a través de la Internet (con el consiguiente retardo); tendríamos pues una ruta asimétrica, ya que el viaje de vuelta (de la 200 a la 199) si que se llevaría a cabo por la vía local, es decir mediante ICMP REDIRECT seguido de conexión directa por la LAN del host 200 al 199.

4 EJEMPLOS DE REDES

Autor: Rogelio Montañana

4.1 INTRODUCCIÓN

Una vez hemos visto en detalle el funcionamiento del nivel de red pasaremos a describir algunas de las redes más representativas que existen en el mundo o en nuestro entorno. De esta forma veremos como se aplican en la práctica algunos de los conceptos que hemos descrito

4.2 ARPANET, NSFNET Y LA INTERNET

ARPANET tuvo su origen en un proyecto del Departamento de Defensa de los Estados Unidos que se desarrolló en la segunda mitad de la década de los sesenta, y que pretendía crear una red de ordenadores resistente a ataques militares[1]. Para esto se optó por crear una red de conmutación de paquetes formada por ordenadores especializados denominados IMPs (Interface Message Processors) que se interconectaban por líneas telefónicas punto a punto de 56 Kb/s. Para aumentar la fiabilidad de la red se diseñó una topología mallada con algún enlace redundante. Los IMPs (que en realidad eran primitivos routers) eran miniordenadores Honeywell con 24 KBytes de memoria RAM, especialmente modificados para desarrollar esta tarea; un router pequeño actual tiene típicamente de 4 a 8 MBytes de memoria RAM.

Dado que en aquellas fechas aun no existían redes locales, cada IMP se conectaba directamente a un solo ordenador multiusuario del tipo mainframe; este ordenador o host daba servicio a un gran número de usuarios y les permitía utilizar la red a través de sus aplicaciones. En ARPANET la 'subred' estaba constituida por los routers (los IMPs) y las líneas punto a punto que los interconectaban.

ARPANET empezó a funcionar en diciembre de 1969 con cuatro nodos. En septiembre de 1972 ya tenía 34 repartidos por todo Estados Unidos.

Paulatinamente se fueron ampliando las posibilidades de los IMPs, permitiendo conexiones de varios hosts en cada IMP, de hosts remotos, o incluso de simples terminales. También se hicieron pruebas con medios de transmisión distintos de las líneas telefónicas, tales como transmisiones por radio o vía satélite. En estas pruebas se puso de manifiesto que los protocolos desarrollados inicialmente no eran aptos para interconectar redes con diferentes tecnologías, por lo que se diseñaron otros nuevos. El embrión de los que hoy conocemos como TCP/IP fue desarrollado por Cerf y Kahn en 1974 en la Universidad de Stanford muy cerca de Palo Alto, en lo que hoy se conoce como el Silicon Valley. A partir de 1983 el TCP/IP fue el conjunto oficial de protocolos de ARPANET. Para fomentar el uso de estos protocolos ARPA adjudicó diversos contratos a la Universidad de California en Berkeley y a BBN (empresa encargada de gestionar ARPANET) para que integraran los nuevos protocolos en el UNIX de Berkeley, con lo que su distribución no tenía costo para las universidades. La decisión fue muy acertada pues era un momento propicio y muchas universidades de Estados Unidos se conectaron a la nueva red. El principal problema que tenía ARPANET era que sólo las universidades o centros de investigación que tenían contratos con el Departamento de Defensa podían conectarse a esa red, y muchas universidades importantes no tenían tales contratos.

Para soslayar este problema y en vista del éxito que adquirió ARPANET la NSF (National Science Foundation) desarrolló en 1984 una red propia, técnicamente muy similar a ARPANET, pero que permitía a las universidades conectarse sin las restricciones que imponía ARPANET. La nueva red, denominada NSFNET, empezó de una forma muy similar a ARPANET interconectando seis superordenadores que la NSF tenía repartidos por Estados Unidos utilizando como routers microordenadores adaptados; el superordenador iba conectado al router y éste a dos o tres líneas, consiguiendo así una red mallada para aumentar la redundancia. En la NSFNET también se utilizaban los protocolos TCP/IP. La Universidad Carnegie-Mellon, que estaba conectada a NSFNET y ARPANET, hacía de pasarela o 'puente' entre ambas redes permitiendo así el intercambio de datagramas, y por tanto de correo electrónico, etc.

[1] Existe una leyenda según la cual ARPANET tenía como objetivo desarrollar una red de comunicaciones que pudiera resistir un ataque nuclear. Aunque cabe pensar que el funcionamiento no orientado a conexión de esta red haría factible tal posibilidad, lo cierto es que la eventualidad de un ataque nuclear no se contempló en el proyecto ARPANET.

El crecimiento de la NSFNET fue espectacular. Multitud de universidades, centros de investigación, museos y bibliotecas se conectaron a ella. En poco tiempo las líneas iniciales de 56 Kb/s que constituían la espina dorsal o *backbone* de la red tuvieron que ser aumentadas a 448 Kb/s, y después a 1.5 Mb/s.

Después de que en 1984 se interconectaron ARPANET y NSFNET también lo hicieron muchas redes regionales y de otros países, aprovechando la ventaja que suponía tener un protocolo común. Así se constituyó poco a poco una superred o *interred* que desde el punto de vista del usuario era una única red puesto que se empleaban en toda ella los mismos protocolos. De manera coloquial los usuarios empezaron a llamar a esta superred la *internet*, hasta el punto que este llegó a ser su nombre propio; no podemos fijar una fecha concreta como el origen de *La Internet*, aunque sin duda ARPANET y NSFNET han sido sus dos principales impulsoras.

A medida que NSFNET y otras redes iban adquiriendo importancia el papel de ARPANET disminuía, hasta desaparecer en 1990. La popularidad atrajo también a empresas privadas que querían entrar en la red, pero esto no era posible en una red financiada por fondos públicos. Por esto en 1990 IBM, MCI y MERIT, impulsados por la NSF, crearon una empresa sin ánimo de lucro denominada ANS (Advanced Networks and Services) que se ocupó de gestionar la NSFNET, que entonces pasó a llamarse ANSNET; al mismo tiempo se aumentó la capacidad del backbone a 45 Mb/s. ANSNET fue vendido a America Online en 1995; para entonces ya existían numerosas empresas que operaban redes IP comerciales en los Estados Unidos.

En Europa (y en España) el desarrollo de La Internet estuvo frenado durante la década de los ochenta por razones políticas, ya que los gobiernos de la mayoría de los países y la Comisión Europea en particular, que financiaban las redes de investigación (equivalentes en Europa a la NSFNET) seguían la directriz de fomentar el uso de protocolos OSI, impidiendo de esta forma el desarrollo de TCP/IP[2]. La comunicación de los investigadores europeos con sus colegas americanos sólo podía efectuarse a través de *pasarelas*, generalmente limitadas al correo electrónico en modo texto. Pero el éxito del TCP/IP en las redes de investigación americanas, unido al lento y escaso desarrollo que estaban teniendo los productos basados en protocolos OSI provocó hacia 1991 un cambio en la actitud de los gobiernos y las redes europeas de investigación, permitiendo la conexión a la Internet americana y la coexistencia de ambos protocolos. De forma inmediata se produjo una explosión en el número de usuarios de Internet en Europa, con una tasa de crecimiento que superó incluso a la ocurrida en Estados Unidos unos años antes. Actualmente los protocolos OSI han caído prácticamente en desuso y el predominio de TCP/IP es absoluto. La red europea de I+D, denominada TEN-155, es gestionada por Dante, empresa con sede en el Reino Unido (www.dante.org).

Hoy en día la Internet crece en el mundo a un ritmo que aproximadamente duplica su tamaño cada año. Es de esperar que en el futuro ese ritmo disminuya. El crecimiento tan elevado, unido a la privatización de la infraestructura básica ha traído como consecuencia indeseable la saturación de la red en muchas áreas. En octubre de 1996 se puso en marcha una iniciativa, liderada por 34 universidades de Estados Unidos, denominada Internet 2 (http://www.internet2.edu/). El objetivo de Internet 2 es rescatar el espíritu original de la NSFNET, es decir crear un entorno de red específicamente orientado al ámbito académico e investigador que ofrezca al usuario unos recursos y una calidad de servicio que le permitan experimentar con nuevas aplicaciones. Las actividades de Internet 2 se centran fundamentalmente en los siguientes ámbitos:

- Nuevos Protocolos: IPv6
- Routing y transmisión Multicast
- Calidad de Servicio y aplicaciones en tiempo real

Aunque nació como iniciativa de una serie de universidades de los Estados Unidos Internet 2 ha establecido vínculos con redes de investigación de altas prestaciones de otros países, entre ellos España.

Internet 2 no posee una infraestructura propia, sino que aprovecha las redes de alta velocidad de los países participantes para desarrollar sus experiencias. Concretamente en Estados Unidos las infraestructuras de alta velocidad utilizadas por Internet 2 son básicamente las redes vBNS (Very High-Speed Backbone Network Service) y Abilene. Podemos hacer cierto paralelismo entre los orígenes de Internet y los de

[2] La excepción a esta regla eran los países nórdicos y el Reino Unido, que estuvieron integrados en Internet bastante antes que el resto de Europa.

Internet 2 diciendo que vBNS y Abilene son respectivamente la 'ARPANET' y la 'NSFNET' de Internet 2. De la misma forma que en su momento el ámbito de Internet superó con creces el de sus dos redes originarias hoy en día el ámbito de Internet 2 supera ampliamente el de estas dos redes, ya que en Internet 2 se encuentran incluidas redes de muchos otros países.

4.2.1 La Internet Society

Cuando se puso en marcha la ARPANET el Departamento de Defensa creó un comité que supervisaba su evolución. En 1983 dicho comité recibió el nombre de Internet Activities Board (IAB), que luego se cambió a Internet Architecture Board. Este comité estaba constituido por diez miembros. Dada la naturaleza de las instituciones que constituían la ARPANET (y después la NSFNET) los miembros del IAB representaban básicamente a universidades y centros de investigación..

El IAB informaba al Departamento de Defensa y a la NSF de la evolución de la red y las posibles mejoras a realizar, ya que eran estas organizaciones las que financiaban por aquellos años el backbone de la Internet. El IAB también se ocupaba de detectar -después de intensas discusiones- donde era necesario o conveniente especificar un nuevo protocolo. Cuando esto ocurría se anunciaba dicha necesidad en la red y normalmente siempre surgían voluntarios que realizaban los desarrollos necesarios. La información circulaba en forma de documentos técnicos denominados RFCs (Request For Comments), cuyo nombre ya da una idea del talante abierto y democrático con el que se tomaban las decisiones técnicas. Los RFCs siguen siendo en la actualidad el mecanismo de publicación de todos los estándares Internet. Todos los RFCs se mantienen en múltiples servidores en la red y cualquiera que lo desee puede consultarlos, redistribuirlos, reproducirlos, etc., sin coste alguno; a título comparativo diremos que los documentos de la ITU y la ISO solo pueden obtenerse comprándolos a la oficina correspondiente y no está permitida su reproducción o redistribución. Actualmente (abril de 2001) hay 3070 RFCs y su número crece continuamente.

En 1989 el IAB fue reorganizado para acomodarse a la evolución sufrida por la red. Su composición fue modificada para que representara a un rango más amplio de intereses ya que la anterior resultaba excesivamente restringida al ámbito académico y tenía un procedimiento de nombramiento no democrático (los miembros salientes nombraban a los entrantes). Además se crearon dos subcomités dependientes del IAB, el IRTF (Internet Research Task Force) y el IETF (Internet Engineering Task Force); el IRTF se concentraría en los problemas a largo plazo, mientras que el IETF debía resolver las cuestiones de ingeniería más inmediatas. En la actualidad la mayoría de los RFCs son elaborados por el IETF y aprobados por el IAB.

En 1991 se creó la Internet Society (ISOC), una asociación internacional para la promoción de la tecnología y servicios Internet en todos los ámbitos de la sociedad. Cualquier persona física u organización que lo desee puede ser miembro de la ISOC sin más que pagar la cuota anual correspondiente. La ISOC está gobernada por un consejo de administración (Board of Trustees) cuyos miembros son elegidos por todos los miembros de la ISOC por votación de entre una serie de candidatos propuestos. La ISOC absorbió en su seno el IAB con sus dos subcomités, el IRTF y el IETF, pero cambió el mecanismo de elección; los miembros del IAB, que son considerados un 'comité de sabios', son ahora nombrados por el consejo de administración de la ISOC.

Como ya hemos dicho la mayoría del trabajo técnico de Internet se desarrolla en el seno del IETF, por lo que esta organización tiene para nosotros un particular interés. Inicialmente el IETF se dividió en grupos de trabajo, cada uno con un problema concreto a resolver. Los presidentes de dichos grupos de trabajo se reunían regularmente constituyendo lo que se llamaba el Comité Director. A medida que iban apareciendo problemas nuevos se iban creando grupos de trabajo, llegando a haber más de 70; para hacer una organización más manejable los grupos se agruparon entonces en ocho áreas; actualmente el Comité Director está formado por los presidentes de cada área.

Paralelamente a la modificación de las estructuras organizativas se formalizaron los procedimientos de estandarización. Actualmente una propuesta de nuevo estándar debe explicarse con todo detalle en un RFC y tener el interés suficiente en la comunidad Internet para que sea tomada en cuenta; en ese momento se convierte en un *Estándar Propuesto (Proposed Standard).* Para avanzar a la etapa de *Borrador de Estándar (Draft Standard)* debe haber una implementación operativa que haya sido probada de forma exhaustiva por dos instalaciones independientes al menos durante cuatro meses. Si supera esta

fase, el software funciona y el IAB considera que la idea es buena se declarará el RFC como un *Estándar Internet (Internet Standard)*. Además se prevé también un mecanismo de 'envejecimiento', es decir que un protocolo pueda quedar anticuado debido a la aparición de otros nuevos y más avanzados que le sustituyan, por lo que un Estándar Internet puede pasar al estado de Obsoleto o *Histórico*. El hecho de exigir implementaciones operativas probadas antes de declarar un estándar oficial pone de manifiesto la filosofía pragmática que siempre ha caracterizado a Internet, radicalmente opuesta a ISO e ITU-T donde un estándar podía aprobarse sin que apareciera jamás una implementación en la vida real. No todos los RFCs siguen el proceso evolutivo descrito más arriba; un RFC puede describir un *Protocolo Experimental* que nunca llegue a aceptarse como Estándar y pasar después de un tiempo al estado de Histórico. Por último, algunos RFCs tienen un carácter *Informativo*, es decir no describen ningún nuevo protocolo sino que simplemente explican detalles con carácter aclaratorio o divulgativo. Un RFC *Informativo* también puede declararse *Histórico* si se considera que ya no aporta información de interés para la situación actual de la red.

Merece la pena destacar que debido al mecanismo evolutivo por el cual se han ido consolidando las diversas entidades que establecen los estándares Internet (la ISOC, el IAB o el IETF) no se ha producido ningún acuerdo entre gobiernos, como ocurrió con la ISO o la ITU-T. Tampoco existe en la ISOC, el Board of Trustees, el IAB o el IETF ninguna representación institucional de los gobiernos de los países, ni hay ninguna regla que fije un número mínimo o máximo de representantes por países en dichos foros. El enfoque diametralmente opuesto entre Internet y las organizaciones de estándares más 'tradicionales' -ISO e ITU-T- provocó, sobre todo durante los años ochenta, una fuerte rivalidad entre Internet y estas organizaciones, que veían con recelo el crecimiento de los protocolos TCP/IP (que para ISO e ITU-T eran estándares *de facto*). Determinadas decisiones técnicas adoptadas en el seno del IETF y el IAB, incluso en los años noventa (por ejemplo las relativas a IPv6) , pueden comprenderse mejor tomado en cuenta dicha rivalidad.

4.3 LA RED NACIONAL DE I+D, REDIRIS

Como en muchos otros países, en España el desarrollo de las redes de ordenadores se produjo inicialmente por la necesidad de comunicación de las universidades y centros de investigación. Las primeras conexiones se llevaron a cabo en 1984 cuando tres universidades, dos en Madrid y una en Barcelona, se conectaron a la red EARN (European Academic and Research Network) que, patrocinada en sus inicios por IBM, utilizaba unos protocolos de este fabricante llamados NJE (Network Job Entry), anteriores incluso a SNA. Un año más tarde una serie de grupos de investigación en física de altas energías establecieron otra red independiente denominada FAENET (Física de Altas Energías NETwork) que utilizaba protocolos DECNET desarrollados por el fabricante DEC (Digital Equipment Corporation), ya que estos grupos utilizaban equipos VAX/VMS de dicho fabricante. Ambas redes (EARN y FAENET) disponían de enlaces internacionales, pero no estaban interconectadas entre sí dentro de España, por lo que la comunicación entre ellas sólo podía establecerse a través de una pasarela en otro país, y sólo para aplicaciones muy concretas (correo electrónico y poco más). Las velocidades de interconexión eran de 2.4 a 9.6 Kb/s, según los casos.

En 1988 el PNID (Plan Nacional de Investigación y Desarrollo), órgano dependiente de la CICYT (Comisión Interministerial de Ciencia y Tecnología), en un intento por armonizar estas iniciativas y para extender el uso de las redes a toda la comunidad universitaria española, inició el Programa IRIS (llamado más tarde RedIRIS), delegando su gestión en Fundesco. Siguiendo las directrices entonces normales en todas las redes de investigación europeas, el Programa IRIS fomentaba casi exclusivamente el uso de protocolos OSI, lo cual supuso que en la práctica el avance de la red se viera limitado por la falta de software adecuado para muchas plataformas hardware, quedando el correo electrónico (que en OSI utilizaba la norma X.400) prácticamente como el único servicio universalmente disponible. Para la transmisión de los datos se utilizaba el servicio Iberpac (red pública X.25) de Telefónica, con velocidades de acceso de 2.4 a 9.6 Kb/s. La red, basada inicialmente en los accesos Iberpac, fue evolucionando gradualmente a una red X.25 privada; para ello se contrataba con Telefónica líneas dedicadas de 64 Kb/s que se utilizaban para interconectar conmutadores X.25 propiedad de RedIRIS. Los primeros de estos conmutadores se instalaron en Madrid, Barcelona y Sevilla, ampliándose después a Valencia y otras comunidades autónomas; esta red privada X.25 se denominó ARTIX (ARTeria de Interconexión X.25).

En 1991, coincidiendo con el cambio a la nueva denominación, RedIRIS empezó a ofrecer un nuevo servicio de interconexión de redes de área local mediante protocolos IP denominado SIDERAL, con lo que el uso de los protocolos TCP/IP, ya presentes por aquel entonces en las redes locales de muchas universidades, se extendió rápidamente a la WAN. Simultáneamente siguió extendiéndose la red ARTIX, es decir se fueron instalando conmutadores X.25 propiedad de RedIRIS en todas las Comunidades Autónomas y reemplazando los accesos X.25 por líneas dedicadas de 64 Kb/s, con una topología de estrella arborescente mallada gracias a la existencia de algunos enlaces redundantes. La red ARTIX actuaba como una red multiprotocolo, ya que viajaban sobre ella paquetes de diversos protocolos: OSI (los 'propios' de RedIRIS), TCP/IP (el nuevo servicio SIDERAL), DECNET (la red FAENET) y NJE/SNA (la red EARN). El uso de X.25 provocaba una pérdida de rendimiento notable y hacía inviable la utilización de velocidades superiores a los 64 Kb/s ya que los conmutadores no eran capaces de inyectar en la red caudales mayores aunque se les dotara de líneas de mayor capacidad.

En 1994 la CICYT decidió traspasar la gestión de la red de Fundesco al CSIC. En ese mismo año se decidió, a la vista de que el tráfico IP era mayoritario, convertir ARTIX en una red IP nativa para evitar la pérdida de rendimiento de X.25 y permitir la evolución a velocidades superiores; para esto se sustituyeron los conmutadores X.25 por routers IP, creando en España una red similar a lo que había sido la NSFNET diez años antes. Al utilizar IP como medio de 'transporte' el tráfico remanente de otros protocolos (DECNET y NJE/SNA principalmente) se encapsulaba ahora en datagramas IP para su envío por la red.

En 1995 la CICYT firmó un acuerdo de cooperación tecnológica con Telefónica, gracias al cual RedIRIS podía utilizar los servicios avanzados experimentales de transmisión de datos, que básicamente eran el servicio Frame Relay (de nombre comercial *Red Uno*) y sobre todo el servicio ATM (nombre comercial *Gigacom*). Con esto se interconectaron a alta velocidad (2 Mb/s o más) todas las Comunidades Autónomas con RedIRIS en Madrid, siguiendo una topología de estrella simple, no mallada. Las conexiones de 2 Mb/s se realizaban por Frame Relay, mientras que las de velocidades superiores se efectuaban por ATM. Desde Madrid se establecía un PVC (CBR en el caso de ATM) con cada una de las comunidades autónomas. El caudal del PVC dependía de las necesidades de cada Comunidad Autónoma. Los caudales eran asimétricos, mayores en sentido 'descendente' (Madrid→comunidad) que en sentido ascendente (comunidad→Madrid) pues el tráfico de los usuarios tenía estas características. Como mecanismo de seguridad se optó, en vez de mallar la red, por utilizar RDSI para establecer conexiones de emergencia en caso de fallo de algún enlace.

A medida que las necesidades de cada Comunidad Autónoma han aumentado se ha ido ampliando el caudal contratado en cada circuito virtual. Hoy en día todas las Comunidades Autónomas utilizan el servicio ATM, pues todas superan el caudal de 2 Mb/s, que es el máximo disponible en el servicio Frame Relay.

En enero de 2001 empezó el desarrollo de una nueva fase en la evolución de la infraestructura de RedIRIS, sustituyendo los circuitos virtuales ATM por enlaces SONET/SDH STM-1 (155 Mb/s). Este cambio no se ha llevado a cabo en todas las comunidades autónomas sino únicamente en aquellas cuyo volumen de tráfico justifica dicha migración (Cataluña, Valencia, Andalucía y Castilla-León). Aunque el servicio contratado en estos casos era el enlace de 155 Mb/s la línea se conectaba al conmutador ATM de RedIRIS, el cual se conectaba además al router de RedIRIS en dicha Comunidad Autónoma y a los centros de dicha Comunidad Autónoma que disponían de conexiones ATM (con enlaces de 34 o 155 Mb/s). Se creaba de esta forma una red ATM privada que en cierto modo representaba una vuelta a los tiempos de ARTIX, pero utilizando ahora ATM y enlaces de 155 Mb/s en vez de X.25 y enlaces de 64 Kb/s. Merece la pena mencionar que tanto el servicio ATM Gigacom de Telefónica como la red ATM privada de RedIRIS utilizan exclusivamente circuitos virtuales permanentes y no emplean por tanto ningún protocolo de routing (ya que al configurarse los circuitos de forma manual la ruta la elige el operador en el momento de definirlos). En cualquier caso la determinación de la ruta en la red privada ATM de RedIRIS era trivial al tratarse de una topología de estrella simple centrada en Madrid existe un solo camino posible entre dos nodos cualesquiera de la red.

La evolución de circuitos ATM a enlaces SONET/SDH fue provocada en parte por la baja fiabilidad del servicio Gigacom de Telefónica. Los enlaces SONET/SDH eran un servicio más básico por lo que se suponía que debían tener un funcionamiento más fiable. Además los enlaces se constituyeron en una topología de anillo con fibras redundantes, con lo que la red física debería proporcionar una alta fiabilidad ante averías. La realidad ha sido que la fiabilidad de estos circuitos (suministrados también por Telefónica) ha sido bastante menor de lo que se había previsto.

La escasa fiabilidad de la red provocó que en el plan para la nueva infraestructura de red, denominada RedIRIS2, se diseñara la red de forma que la redundancia se ofreciera a nivel IP, es decir que el operador suministrara una red mallada de enlaces interconectando los routers sobre la cual RedIRIS ofrecería un servicio de alta fiabilidad.

En junio de 2002 se publicó el pliego de especificaciones técnicas para que en base a él las empresas interesadas realizaran sus propuestas para la nueva RedIRIS2. Se presentaron siete empresas resultando adjudicado el concurso a Red Eléctrica Telecomunicaciones (Albura) empresa filial de Red Eléctrica de España. Este tipo de servicios se ofrece gracias a los tendidos de fibra óptica que Red Eléctrica en la red de distribución de alta tensión. La topología resultante es una red fuertemente mallada (todos los nodos tienen dos o más enlaces de interconexión con el resto de la red) y de muy elevada capacidad, pues siete nodos disfrutan de dos o más enlaces de 2,5 Gb/s (STM-16) y otros siete de dos o más enlaces de 622 Mb/s (STM-4). Los tres nodos restantes disponen de enlaces de 155 Mb/s (STM-1). Se utilizan enlaces SONET/SDH conectados a interfaces POS (Packet Over SONET) en los routers, lo cual mejora el rendimiento respecto de ATM en un 16% aproximadamente. Los enlaces de 622 y 155 Mb/s son un servicio de alquiler de capacidad, es decir que RedIRIS alquila a Albura un circuito de esa capacidad entre los puntos indicados; en cambio los enlaces de 2,5 Gb/s son un servicio de lambda oscura, es decir que RedIRIS alquila a Albura una determinada longitud de onda en la fibra que une los dos puntos entre los que se constituye el circuito. Esta red entró en funcionamiento en 2003.

Aparte de las conexiones con las diversas Comunidades Autónomas RedIRIS soporta una serie de conexiones externas a través de las cuales intercambia tráfico con los demás ISPs. Como es lógico estas conexiones también se encuentran en permanente evolución.

Además de los servicios básicos de conectividad a nivel IP RedIRIS ofrece una serie de servicios adicionales entre los que cabe mencionar los siguientes:

- Enrutamiento multicast con conectividad a la red MBone a nivel mundial.
- Acceso a la red experimental de IPv6, 6Bone
- Diversos servicios del nivel de aplicación, tales como:
    - Noticias (netnews)
    - Listas de distribución (ListServ)
    - Proxy/Cache (red jerarquizada de servidores repartida por toda España)

5 EL NIVEL DE TRANSPORTE EN INTERNET

Autor: Rogelio Montañana

5.1 INTRODUCCIÓN

El nivel de transporte se encarga de suministrar el servicio de transporte de bits a las aplicaciones. Éstas funcionan generalmente según el paradigma cliente-servidor, por el cual una aplicación (cliente) toma la iniciativa y solicita los servicios a la otra (servidor).

Como ya sabemos la comunicación 'peer to peer' entre dos entidades del nivel de transporte ocurre en realidad gracias a los servicios ofrecidos por el nivel de red. Mientras que el nivel de red se ocupa de resolver los problemas propios de la topología de ésta (rutas y congestión fundamentalmente) el nivel de transporte sólo existe en las dos entidades extremas de la comunicación, por lo que también se le llama nivel host-host o extremo a extremo. El nivel de transporte no es consciente, ni debe serlo, de la manera como físicamente están interconectados los dos hosts, que puede ser por una LAN, una WAN o una combinación de múltiples redes de ambos tipos.

La unidad básica de intercambio de información a nivel de enlace se denomina *trama* (porque los datos van 'rodeados' de información de control por delante y por detrás). En el nivel de red esta unidad básica se conoce como *paquete*. No existe un término equivalente para la unidad de transferencia de información en el nivel de transporte; a falta de mejor alternativa utilizaremos para este fin el término OSI *TPDU* (Transport Protocol Data Unit); en la Internet se suele utilizar el término *mensaje* en el caso de UDP (servicio no orientado a conexión), y *segmento* en el de TCP (servicio orientado a conexión), pero esta nomenclatura no es compartida por otros protocolos de transporte.

Generalmente las aplicaciones requieren que el nivel de transporte les garantice la entrega de los datos al destinatario, sin errores, pérdidas ni datos duplicados; para que esto sea posible el nivel de transporte ofrecerá normalmente un servicio orientado a conexión, con retransmisiones en caso necesario. Este es el caso por ejemplo del protocolo TCP de Internet, utilizado en muchas aplicaciones como FTP (File Transfer Protocol, transferencia de ficheros), SMTP (Simple Mail Transfer Protocol, correo electrónico) , HTTP (HyperText Transfer Protocol, usado en tráfico Web), etc.

En ocasiones las aplicaciones se conforman -o incluso prefieren- un servicio menos fiable en el que los mensajes se envían sin pedir confirmación, de forma independiente unos de otros. Este tipo de servicio se suministra normalmente con un protocolo no orientado a conexión. El protocolo UDP de Internet es un ejemplo de este tipo de servicio. Entre los casos en que se quiere un servicio de este tipo se encuentran por ejemplo las aplicaciones en tiempo real ya que en ese caso no se quiere incurrir en el retardo propio de un protocolo orientado a conexión.

Al igual que en Internet en OSI también hay dos protocolos de transporte, uno orientado a conexión y uno no orientado a conexión. La tabla 6.1 muestra los más importantes protocolos de nivel de red y de transporte de Internet y sus corresponidentes protocolos OSI.

Tipo de protocolo	Internet	OSI
Nivel de red	IP (Internet Protocol)	CLNP (ConnectionLess Network Protocol)
Routing interno	OSPF (Open Shortest Path First)	IS-IS (Intermediate System to Intermediate System)
Routing externo	BGP (Border Gateway Protocol)	IDRP (InterDomain Routing Protocol)
Nivel de transporte, orientado a conexión	TCP (Transmission Control Protocol)	TP4 (Transport Protocol clase 4)
Nivel de transporte, no orientado a conexión	UDP (User Datagram Protocol)	TP0 (Transport Protocol clase 0)

Tabla 6.1.- Correspondencia de protocolos Internet y OSI a nivel de red y de transporte

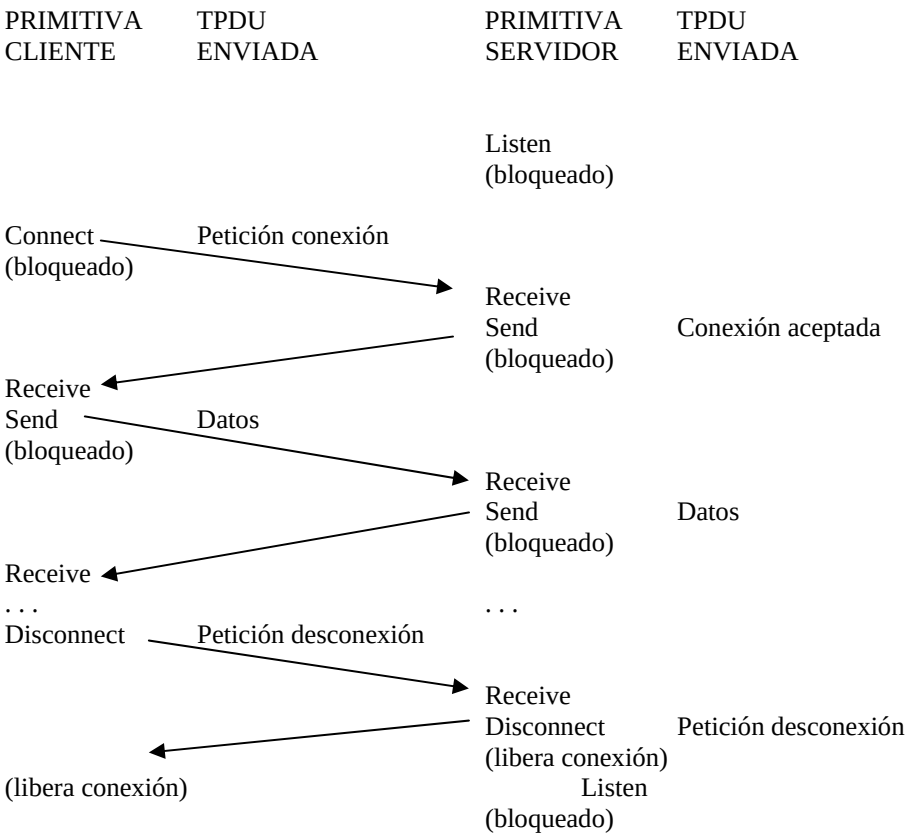
Normalmente las entidades del nivel de transporte se implementan como procesos en el sistema operativo del host, o bien procesos de usuario. El sistema operativo puede ser monousuario o multiusuario. En

muchos casos el host tiene una sola instancia del nivel de red, y una sola del de transporte, pero muchas de los niveles superiores (aplicación o sesión); el nivel de transporte se encarga de mutiplexar el tráfico recibido de las diversas entidades de nivel superior en una única conexión a través del nivel de red.

Los protocolos de transporte no orientados a conexión, como UDP, son protocolos muy sencillos, que existen únicamente para permitir la correcta conexión de la capa de aplicación y la de red; actúan como una capa de adaptación bastante primitiva. Por esto la mayoría de nuestra discusión se centrará en los protocolos orientados a conexión, a los que nos referiremos implícitamente la mayor parte del tiempo.

5.1.1 Primitivas del servicio de transporte

En un servicio de transporte básico orientado a conexión (CONS) la secuencia de primitivas podría ser algo como lo siguiente:



5.1.2 La interfaz sockets

Como ya vimos en el tema 1 la interfaz utilizada entre los diferentes niveles de un mismo sistema no forma parte de un protocolo de comunicaciones. Dos sistemas necesitan acordar las reglas que seguirá la comunicación entre ellos en cada uno de los niveles, pero la forma como se realiza la comunicación vertical, es decir , la que ocurre dentro de un sistema es asunto interno que no incumbe a los protocolos. Esa comunicación vertical se realiza normalmente mediante las denominadas APIs (Application Programming Interfaces).

Aunque no requerida por las comunicaciones, la estandarización de las APIs comporta unos beneficios evidentes por la posibilidad de aprovechar software entre sistemas diferentes. Esto es especialmente cierto en el nivel de transporte, ya que es aquí donde interaccionarán más programas diferentes, correspondientes a las diversas aplicaciones que se desarrollen. Muchas implementaciones de TCP/IP disponen de una API para programación aplicaciones denominada sockets (literalmente enchufes, aunque nunca se utiliza esta denominación). Los sockets se introdujeron con el sistema operativo UNIX BSD

(Berkeley Software Distribution) en 1982. La interfaz sockets es multiprotocolo, soporta TCP, UDP y otros protocolos.

Aun cuando no forman parte de ningún estándar oficial ni están recogidos en ningún RFC, los sockets son la API más extendida en programación de aplicaciones TCP/IP y forman un estándar 'de facto'. Existen implementaciones para muchos sistemas operativos no UNIX, e incluso en casos en que el sistema operativo no los incorpora suele haber una librería de rutinas sockets que permite adaptar programas con relativa facilidad. Por ejemplo, la interfaz WinSock permite adaptar aplicaciones MS Windows sobre diversas implementaciones de TCP/IP. Al no ser un estándar puede haber pequeñas diferencias entre implementaciones, por lo que es conveniente disponer siempre de la documentación correspondiente al software que se utiliza.

La filosofía básica de los sockets deriva directamente del sistema de entrada/salida de UNIX, con ampliaciones que permiten por ejemplo a un proceso servidor ponerse 'a la escucha'. Algunas de las rutinas generan y envían las TPDUs a partir de sus argumentos; éstas (las TPDUs) sí forman parte del protocolo, por lo que deben de ser conformes con el estándar correspondiente.

5.2 ELEMENTOS DE PROTOCOLOS DE TRANSPORTE

Al ocuparse de la comunicación extremo a extremo o punto a punto, el nivel de transporte se parece en algunos aspectos al nivel de enlace. Así por ejemplo, entre los asuntos de los que normalmente habrá de ocuparse se encuentran el control de errores (incluyendo mensajes perdidos o duplicados) y el control de flujo. Aunque las técnicas que se aplican son parecidas, existen importantes diferencias entre ambos motivadas por el hecho de que en el nivel de enlace hay sólo un hilo físico (o su equivalente) entre las dos entidades comunicantes, mientras que en el nivel de transporte hay toda una red. Las mayores diferencias entre el nivel de transporte y el de enlace son las siguientes:

- El retardo que se observa en el nivel de transporte es normalmente mucho mayor y sobre todo más variable (mayor jitter) que en el de enlace.

- En el nivel de enlace el medio físico entre las dos entidades tiene una capacidad de almacenamiento de información normalmente muy reducida y siempre la misma; en el de transporte los routers intermedios pueden tener una capacidad considerable y esta puede variar con el estado de la red.

- En el nivel de enlace se asegura que las tramas llegarán al receptor en el mismo orden que han salido del emisor (salvo que se pierdan, en cuyo caso no llegarán); en el nivel de transporte esto es cierto solo cuando se utiliza un servicio orientado a conexión en el nivel de red; si se utiliza un servicio no orientado a conexión el receptor podría recibir los datos en orden distinto al de emisión.

- En el nivel de enlace las dos entidades se 'ven' directamente (suponiendo una comunicación dúplex); a veces incluso de forma permanente, por ejemplo en una comunicación síncrona tipo HDLC están continuamente emitiendo la secuencia 01111110; esto permite que el emisor sepa en todo momento si el receptor está operativo, y el receptor sabe que los datos recibidos corresponden todos a una misma sesión del emisor. En el nivel de transporte la comunicación es indirecta; el emisor podría enviar datos, quedar fuera de servicio y más tarde entrar en funcionamiento otra vez; si no se adoptan las medidas oportunas el receptor podría recibir todos esos datos sin siquiera percatarse de que corresponden a dos sesiones distintas del emisor (incluso podrían pertenecer a dos usuarios distintos).

Recordemos que en el modelo OSI cada nivel presta sus servicios al nivel superior a través del SAP (Service Access Point), cada uno de los cuales es identificado por una dirección. Por ejemplo en Internet la dirección SAP por la que el nivel de red accede al servicio es la dirección IP del host, que hemos visto en el tema anterior. La dirección SAP por la que el nivel de transporte accede al servicio de red está formada por el campo *protocolo* del datagrama IP (6 para TCP y 17 para UDP, por ejemplo). A su vez el nivel de transporte ofrece sus servicios al nivel de aplicación a través de unos SAPs específicos, que en el

caso de Internet son los denominados *ports* o *puertos*. Estos puertos se denominan también TSAPs (Transport Service Acces Point).

Para que un proceso cliente de un host pueda comunicar con un proceso servidor en otro host haciendo uso de los servicios de su nivel de transporte es preciso que conozca el TSAP correspondiente en el host de destino. Normalmente esta información forma parte del estándar del protocolo, por lo que es universalmente conocido y cualquier cliente que lo desee sabe que TSAP debe utilizar para acceder a dicho servidor. En cambio el TSAP del cliente no necesita ser conocido por otros usuarios y puede ser diferente para cada conexión.

5.2.1 Establecimiento de una conexión

En principio para establecer una conexión el cliente emite una TPDU de petición de conexión, y el servidor responde con una TPDU de aceptación; a partir de ese momento puede empezar el intercambio de datos. Sin embargo cuando analizamos el proceso de conexión con mayor detalle encontramos diversos problemas que pueden presentarse y que hay que prever.

Recordemos que en el nivel de transporte puede haber una gran fluctuación (a veces del orden de segundos) en el tiempo que tardan en llegar las TPDUs a su destino; las TPDUs pueden perderse o llegar duplicadas, ya que si el emisor no recibe confirmación reenviará la misma TPDU pasado el timeout. Imaginemos que el cliente intercambia una serie de TPDUs con el servidor, y cuando ya ha terminado la transacción cierra la sesión; segundos mas tarde de algún rincón de la red aparecen la misma secuencia de TPDUs del cliente duplicadas que llegan al servidor de nuevo; éste realizaría la misma transacción otra vez, con efectos posiblemente desastrosos[1].

Para evitar este tipo de problemas se utiliza para establecer la conexión el mecanismo conocido como *saludo a tres vías* (three-way handshake). La idea es que el servidor sólo aceptará la conexión después de haber pedido al cliente confirmación de que desea realizarla. En principio esto por sí solo no resuelve nuestro problema, ya que cabría pensar que después de la TPDU de petición inicial duplicada la red le entregue al servidor la TPDU de confirmación, también retrasada.

La solución a este problema es la siguiente: tanto el cliente como el servidor utilizan un protocolo de ventana deslizante para el envío de las TPDUs, para lo cual emplean un número de secuencia; a diferencia del número de secuencia que vimos en el nivel de enlace, el del nivel de transporte emplea rangos muy amplios (por ejemplo en TCP el número de secuencia se almacena en un campo de 32 bits, con lo que es un número módulo 2^{32}). Tanto el cliente como el servidor eligen de forma aleatoria o pseudoaleatoria el valor inicial del número de secuencia que van a utilizar, cada uno por separado para cada sentido de la comunicación. El cliente informa al servidor en su primera TPDU del número de secuencia elegido; por su parte el servidor le responde en otra TPDU con el número de secuencia elegido por él, incluyendo en ésta un ACK piggybacked de la TPDU recibida. De esta forma si el servidor recibe una TPDU de petición de conexión vieja responderá con una TPDU al cliente en la que pondrá en el campo ACK el número de secuencia recibido; cuando la respuesta llegue al cliente éste verá que ese número no corresponde con ninguna conexión que él tuviera pendiente de confirmación, por lo que rechazará la conexión; el servidor por su parte esperará recibir en el campo ACK de la siguiente TPDU un valor que corresponda con el que él ha enviado en la anterior.

La técnica de los números de secuencia aleatorios evita también el riesgo de que un proceso cliente que cae por algún motivo (por ejemplo por un fallo de corriente) utilice la misma conexión cuando reaparece más tarde, ya que normalmente el nuevo proceso intentará utilizar un número de secuencia diferente. Esta es una medida de seguridad ya que el nuevo proceso cliente podría pertenecer a otro usuario; supongamos por ejemplo que al inicio de la conexión se realiza una identificación con clave usuario/password ante el servidor, en tal caso el nuevo cliente podría acceder a todos los recursos del usuario anterior sin identificarse.

Generalmente se establece una vida máxima para las TPDUs en la red; de esta forma se reduce el riesgo de recibir duplicados retrasados. Cuando un sistema cae y vuelve a arrancar se recomienda esperar al

[1] Si por ejemplo la transacción consiste en la transferencia de dinero entre cuentas bancarias se realizarían dos trasferencias en vez de una.

menos el tiempo de vida de las TPDUs antes de activar el nivel de transporte; de esta manera es imposible que una TPDU de la sesión anterior pueda aparecer por alguna parte cuando se inicia la sesión nueva. En Internet por ejemplo el tiempo de vida máximo recomendado de las TPDUs es de 2 minutos, y se controla mediante el campo TTL en el datagrama IP.

Una vez establecidos los números de secuencia es posible utilizar para el intercambio de TPDUs cualquier protocolo de ventana deslizante. A diferencia del nivel de enlace, donde el protocolo se basa en numerar tramas, en el nivel de transporte se suelen numerar bytes, ya que el tamaño de las TPDUs puede ser muy variable. Para las retransmisiones se puede utilizar tanto *retroceso n* como *repetición selectiva*.

5.2.2 Terminación de una conexión

Una conexión puede terminarse de forma simétrica o asimétrica. La terminación asimétrica es unilateral, es decir uno de los dos hosts decide terminar y termina la conexión en ambos sentidos. En la terminación simétrica cada host corta la conexión únicamente en el sentido en el que emite datos; podemos considerar la terminación simétrica como dos circuitos simplex donde cada uno es controlado por el emisor.

La terminación asimétrica se considera anormal y puede provocar la pérdida de información, ya que cuando un host ha enviado la TPDU de desconexión ya no acepta más datos; entretanto el otro host podría haber enviado una TPDU de datos que no será aceptada.

En la terminación simétrica -la más normal- el host 1 'invita' al host 2 a desconectar mediante una TPDU DISCONNECT REQUEST; el host 2 responde con otra DISCONNECT REQUEST, a la cual el host 1 responde con una TPDU ACK y cierra la conexión; el host 2 cerrará la conexión al recibir el ACK. Por este mecanismo se asegura que no se pierden TPDUs 'en ruta' ya que ambos hosts tienen aviso previo de la desconexión y dan su conformidad explícitamente. Este mecanismo supone el intercambio de tres mensajes de forma análoga al proceso de conexión, por lo que también se denomina saludo a tres vías (aunque quizá debería llamarse ‘despedida a tres vías’); no existe forma fiable de terminar la conexión en menos mensajes sin correr el riesgo de perder datos.

Si se pierde alguna de las TPDUs de desconexión el mecanismo del saludo atres vías falla pues los hosts se quedan esperando eternamente la respuesta. Para evitar esto se utiliza un mecanismo de timeouts que resuelve el problema reenviando la TPDU perdida si se trata de un DISCONNECT REQUEST, o cerrando la conexión por timeout cuando lo que se ha perdido es el ACK. En el Tanenbaum Fig. 6-14 aparece una relación de los casos 'patológicos' que pueden ocurrir y como se resuelven.

Existen muchas circunstancias que pueden provocar que una conexión se quede medio abierta, es decir abierta sólo por un lado. Por ejemplo, un host puede quedar fuera de servicio sin previo aviso y el otro, que tenía una conexión abierta con él, quedar a la espera sin saber que ha ocurrido. Para resolver estas situaciones se prevé normalmente un tiempo máximo durante el cual una conexión puede estar abierta sin tráfico; pasado ese tiempo los hosts se envian mensajes de prueba (denominados keep-alive en TCP) para comprobar que el otro lado aún responde. Los valores de timeout para el envío de mensajes keep-alive son increíblemente grandes (la documentación de TCP sugiere 2 horas como valor por defecto). Un valor muy pequeño podría provocar que un fallo momentáneo en la red cerrara conexiones a nivel de transporte, perdiendo así la principal ventaja de las redes de datagramas.

Analicemos ahora que ocurre si dos hosts tienen establecida una conexión entre ellos y falla la red que los une. En el caso de utilizar un servicio orientado a conexión (X.25, ATM) generalmente la sesión termina de manera abrupta, pues la vía de comunicación (el circuito virtual) ha desaparecido y para restaurar la comunicación habrá que restablecer el circuito, presumiblemente por un camino físico diferente. En el caso de utilizar un servicio de red no orientado a conexión (IP, OSI CLNP) la red reencaminará los datagramas por una ruta alternativa (suponiendo que exista) por lo que lo único que el nivel de transporte detectará es la pérdida de unas pocas TPDUs, pero la conexión no se cortará.

5.2.3 Control de flujo y de buffers

El control de flujo en el nivel de transporte es fundamental, ya que la velocidad con que los datos llegan al receptor puede ser muy variable al intervenir multitud de factores.

Como ya hemos dicho se suelen utilizar protocolos de ventana desilzante. Mientras que en el nivel de enlace se asignaba de manera estática un espacio de buffers a cada conexión, en el nivel de transporte esta estrategia no es adecuada, pues el número de conexiones simultáneas puede variar muchísimo al no haber una interfaz física asociada a cada conexión.

Por este motivo la asignación de espacio para buffers en el nivel de transporte tiene dos características singulares que le diferencian del nivel de enlace. En primer lugar el espacio de buffers es común y compartido por todas las conexiones, entrantes y salientes. En segundo lugar el reparto del espacio entre las conexiones activas se hace de forma dinámica de acuerdo con las necesidades; una conexión con poco tráfico recibirá menos asignación que una con mucho tráfico. En todo momento cada conexión tiene asignado un espacio para emisión y uno para recepción; el espacio de emisión está ocupado con TPDUs pendientes de ser enviadas o de confirmación; el espacio de recepción tiene una parte ocupada con TPDUs recibidas pendientes de ser aceptadas por el nivel de aplicación, y otra libre reservada para TPDUs que puedan llegar del otro host.

Otra diferencia respecto al nivel de enlace estriba en que, mientras que el tamaño de las tramas suele ser constante para una conexión física dada, el tamaño de las TPDUs puede ser muy variable. Para optimizar la utilización del espacio se asignan segmentos de buffer de longitud variable. Para una máxima flexibilidad en este sentido tanto los números de secuencia como los tamaños de ventana cuentan generalmente bytes, no TPDUs.

La parte de buffer que el receptor tiene reservada para TPDUs que puedan llegarle es anunciada al emisor regularmente, para que éste sepa que cantidad de datos esta dispuesto a aceptar el receptor. Este espacio puede fluctuar mucho con el tiempo en función de la actividad que tenga esa y el resto de conexiones que mantenga el host.

Con este modo de funcionamiento el receptor realmente controla la situación, ya que si indica una ventana cero el emisor tendrá que esperar y no enviarle datos mientras el receptor no le anuncie una ventana mayor.

Veamos un ejemplo sencillo de como funcionaría una sesión TCP:

HOST 1	HOST 2
Seq=1000,Win=4000 →	
	← Seq=1500,Ack=1001,Win=4000
Seq=1001,Ack=1501,datos(1000) →	
	← Seq=1501,Ack=2001,datos(1000)
Seq=2001,Ack=2501,datos(1000) →	
Seq=3001,Ack=2501,datos(1000) →	
Seq=4001,Ack=2501,datos(1000) →	
bloqueado	
	← Seq=2501,Ack=5001,Win=0
…esperando…	
	← Seq=2501,Ack=5001,Win=2000
Seq=5001,Ack=2501,datos(1000) →	
	← Seq=2501,Ack=6001,Win=3000
...	

Supongamos ahora que en el ejemplo anterior se hubiera perdido la cuarta TPDU enviada de host1 a host2 (la que aparece en cursiva); en ese caso el host-2 no habría enviado el ACK 5001, y el host-1, al agotar el timeout correspondiente a esa TPDU la habría reenviado; funcionando con repetición selectiva la secuencia sería la siguiente:

HOST 1 HOST 2

Seq=1000,Win=4000 →

←Seq=1500,Ack=1001,Win=4000

Seq=1001,Ack=1501,datos(1000) →

←Seq=1501,Ack=2001,datos(1000)

Seq=2001,Ack=2501,datos(1000) →

Seq=3001,Ack=2501,datos(1000) → (perdida)

Seq=4001,Ack=2501,datos(1000) →

←Seq=2501,Ack=3001

…

Seq=3001,Ack=2501,datos(1000) → (reenviada por timeout)

bloqueado

←Seq=2501,Ack=5001,Win=0

…esperando…

←Seq=2501,Ack=5001,Win=2000

Seq=5001,Ack=2501,datos(1000) →

←Seq=2501,Ack=6001,Win=3000

…

En caso de funcionar con retroceso n las cosas habrían sido ligeramente diferentes:

HOST 1 HOST 2

Seq=1000, Win=4000 →

← Seq=1500,Ack=1001, Win=4000

Seq=1001, Ack=1501,datos(1000) →

← Seq=1501,Ack=2001,datos(1000)

Seq=2001,Ack=2501,datos(1000) →

Seq=3001,Ack=2501,datos(1000) → (perdida)

Seq=4001,Ack=2501,datos(1000) → (ignorada)

← Seq=2501,Ack=3001

…

Seq=3001,Ack=2501,datos(1000) → (reenviada por timeout)

← Seq=2501,Ack=4001

…

Seq=4001,Ack=2501,datos(1000) → (reenviada por timeout)

bloqueado

← Seq=2501,Ack=5001,Win=0

...esperando...

← Seq=2501,Ack=5001,Win=2000

Seq=5001,Ack=2501,datos(1000) →

← Seq=2501,Ack=6001,Win=3000

…

En redes no orientadas a conexión los datagramas (y por tanto las TPDUs) pueden llegar desordenados, por lo que el nivel de transporte debe estar preparado para recibir números de secuencia desordenados (siempre y cuando se encuentren dentro del rango correspondiente a la ventana vigente en ese momento).

5.2.4 Multiplexación

En las redes públicas de conmutación de paquetes (X.25, frame relay y ATM), que son orientadas a conexión, el usuario paga por cada circuito virtual, lo cual estimula a utilizar el mínimo número de circuitos posible. Generalmente es el nivel de transporte el encargado en estos casos de multiplexar la diferentes conexiones solicitadas por el nivel de aplicación en una única conexión a nivel de red; dicho en

terminología OSI el nivel de transporte presenta diferentes TSAPs sobre un único NSAP. Esto se conoce como multiplexación hacia arriba, ya que visto en el modelo de capas supone que varias direcciones del nivel de transporte confluyan en una única dirección del nivel de red.

También en redes no orientadas a conexión (IP o ISO CLNP) el nivel de transporte suele ocuparse de multiplexar el tráfico de las diferentes aplicaciones y usuarios (cada aplicación puede estar siendo utilizada por varios usuarios) en una única dirección a nivel de red.

Existen otras situaciones en las que interesa hacer multiplexación en sentido opuesto. Supongamos por ejemplo el caso de un servidor al que para mejorar su rendimiento se le han instalado dos interfaces de red, por ejemplo dos controladores Ethernet; supongamos que el servidor utiliza TCP/IP, y posee por tanto una dirección IP para cada interfaz. El servidor esta dedicado a una única aplicación, por lo que utiliza un único puerto. En este caso necesitamos hacer multiplexación hacia abajo. En ocasiones, debido a la forma como funcionan los protocolos o como se reparte la capacidad disponible, la multiplexación hacia abajo es interesante incluso cuando hay una sola interfaz física con la red; por ejemplo, si el protocolo a nivel de transporte no utiliza ventana deslizante sino parada y espera el crear varias conexiones a nivel de transporte para un mismo usuario del nivel de aplicación permite aprovechar los tiempos de espera que produce el protocolo. También sería útil la multiplexación si el algoritmo de reparto de recursos se basa en los usuarios del nivel de transporte; presentando varias conexiones obtendremos mejor rendimiento (seguramente con detrimento de los demás usuarios).

5.2.5 Recuperación de caídas

Ya hemos visto el tipo de medidas preventivas que se adoptan para evitar que cuando una instancia del nivel de transporte en un host cae y se levanta mas tarde no sea posible recuperar la conexión previamente establecida, y haya que crear una nueva. Esto es una medida de seguridad fundamental para evitar inconsistencias en la información y accesos no autorizados.

Otro problema importante es que ocurre cuando cae todo un host, lo cual provoca la caída simultánea del nivel de transporte y el nivel de aplicación. Supongamos por ejemplo que un cliente está realizando una serie de actualizaciones en una base de datos, cada una de ellas contenida en una TPDU; a cada transacción el servidor responde con un ACK indicando que ha efectuado la operación correspondiente. En un determinado momento el servidor cae, rearrancando a continuación; podemos concluir que si el cliente ha recibido el ACK es que la actualización se ha efectuado, y si no no, pero como la actualización y el envío del ACK son sucesos consecutivos y no simultáneos siempre ocurrirá uno primero y el otro después; cualquiera que sea el orden elegido siempre podrá ocurrir que el host caiga entre ambos eventos, con lo que tendremos o bien una actualización efectuada y no notificada, o una notificación enviada al cliente de una actualización no realizada. Este tipo de problemas solo puede resolverse a nivel de aplicación mediante una posterior verificación de lo que realmente ha ocurrido.

5.3 LOS PROTOCOLOS DE TRANSPORTE DE LA INTERNET: TCP Y UDP

Como ya hemos comentado existen dos protocolos de transporte en la Internet: TCP es fiable, orientado a conexión con control de flujo, y UDP es no fiable (sin confirmación) no orientado a conexión y sin control de flujo. La TPDU de TCP se denomina *segmento*, y la de UDP *mensaje* o también *datagrama UDP*.

TCP prevé una comunicación full dúplex punto a punto entre dos hosts, no hay soporte para tráfico multicast. En UDP la comunicación es simplex (aunque obviamente un datagrama UDP puede ser respondido por el receptor con otro); en UDP es posible el tráfico multicast o broadcast.

El protocolo TCP es mucho más complejo que UDP.

5.3.1 TCP (Transport Control Protocol)

Recordemos que el nivel de transporte debe ofrecer algún mecanismo que permita distinguir a que aplicación van dirigidos los datos, lo que hemos denominado los TSAPs. En TCP los TSAPs se denominan *ports* o *puertos*.

En sentido estricto una conexión entre dos entidades usuarias del nivel de transporte queda identificada por los TSAPs en los que conectan cada una (podemos pensar en el TSAP como el conector telefónico, diríamos entonces que una conversación telefónica queda perfectamente especificada por los números de teléfono de las dos rosetas donde están enchufados los aparatos con los que se está hablando). Recordaremos que un TSAP en Internet está especificado por:

- Dirección donde 'conecta' el nivel de red: dirección IP de los dos hosts
- Dirección donde conecta el nivel de transporte (campo protocolo del datagrama IP): normalmente TCP ya que UDP al ser no orientado a conexión no puede establecer conexiones.
- Dirección donde conecta el nivel de aplicación: esto es el puerto.

Dado que la conexión en el nivel de transporte siempre se suele realizar con el protocolo TCP este dato es innecesario y se suele omitir. Sin embargo en un mismo host un número de port puede ser utilizado simultáneamente por una aplicación para UDP y por otra para TCP; esto no plantea ningún conflicto ya que son TSAPs diferentes.

Así pues, una conexión de dos entidades usuarias del nivel de transporte se especifica por la combinación:

Dirección IP host 1 + port host 1 + dirección IP host 2 + port host 2

El port es un número entero entre 0 y 65535. Por convenio los números 0 a 1023 están reservados para el uso de servicios estándar, por lo que se les denomina *puertos bien conocidos* o *well-known ports*. Cualquier número por encima de 1023 está disponible para ser utilizado libremente por los usuarios. Los valores vigentes de los puertos bien conocidos se pueden consultar por ejemplo en el web de la IANA (Internet Assigned Number Authority) www.iana.org/numbers html. En la tabla 6.2 se recogen algunos de los más habituales.

Puerto	Aplicación	Descripción
9	Discard	Descarta todos los datos recibidos (para pruebas)
19	Chargen	Intercambia cadenas de caracteres (para pruebas)
20	FTP-Data	Transferencia de datos FTP
21	FTP	Diálogo en transferencia de ficheros
23	TELNET	Logon remoto
25	SMTP	Correo electrónico
110	POP3	Servidor de correo
119	NNTP	News

Tabla 6.2.- Algunos ejemplos de puertos 'bien conocidos' de TCP

El Tanenbaum, como otros libros de texto, dice (pag. 523 y 526) que los puertos bien conocidos son los que están por debajo del 256; sin embargo el estándar establece que están reservados para este fin todos los valores por debajo de 1024.

Para comprender la relación entre los puertos y las conexiones en TCP veamos un ejemplo concreto: supongamos que cinco usuarios desde un host de dirección IP 134.123.1.2 inician una sesión de logon remoto (Telnet) hacia el host de dirección 221.198.34.21; cada uno de ellos ejecutará en su host un programa telnet cliente que abrirá una conexión con el servidor Telnet (puerto 23) en el otro; las conexiones establecidas podrían ser por ejemplo:

Usuario 1: 134.123.1.2.1024 con 221.198.34.21.23
Usuario 2: 134.123.1.2.1025 con 221.198.34.21.23
Usuario 3: 134.123.1.2.1026 con 221.198.34.21.23
Usuario 4: 134.123.1.2.1030 con 221.198.34.21.23
Usuario 5: 134.123.1.2.1031 con 221.198.34.21.23

Aquí hemos empleado la notación 'dirección IP. Puerto' para identificar el socket[2]; cada conexión queda identificada de forma no ambigua por los dos sockets que conecta. Obsérvese que la asignación de puertos para los clientes se hace por simple orden de llegada a partir del primer número de puerto no reservado. En el servidor todas las conexiones utilizan el puerto 23 (pues todas acceden al mismo proceso, el servidor telnet); en cambio en el cliente cada usuario es un proceso diferente y utiliza un puerto distinto.

Complicando un poco más el ejemplo anterior podríamos imaginar que el host cliente estuviera 'multihomed', es decir que tuviera dos interfaces físicas (por ejemplo dos tarjetas LAN) y por tanto tuviera dos direcciones IP; supongamos que los usuarios utilizan ambas interfaces alternativamente, en ese caso las conexiones podrían ser:

Usuario 1: 134.123.1.2.1024 con 221.198.34.21.23
Usuario 2: 134.123.1.3.1024 con 221.198.34.21.23
Usuario 3: 134.123.1.2.1025 con 221.198.34.21.23
Usuario 4: 134.123.1.3.1025 con 221.198.34.21.23
Usuario 5: 134.123.1.2.1030 con 221.198.34.21.23

Por otro lado el host cliente podría simultáneamente a las sesiones telnet enviar datagramas UDP al servidor. Aunque en este caso no se establece una conexión (pues se trata de un servicio CLNS) hay un puerto de origen y uno de destino; podría haber datagramas que tuvieran como puerto de origen el 1024 en el host 134.123.1.2 y como destino el 23 en 221.198.34.21; esto no causaría ninguna ambigüedad ya que el campo protocolo de la cabecera IP permitiría distinguir ambos tipos de paquetes entregando cada uno al servicio correspondiente del nivel de transporte en el host de destino.

5.3.2 La cabecera de segmento TCP

La cabecera de un segmento TCP tiene la estructura que se muestra en la tabla 6.3.

[2] La denominación socket empleada para la combinación IP.puerto es la misma que la de las APIs utilizadas en UNIX para acceder a los servicio TCP.

Campo	Longitud (bits)
Puerto origen	16
Puerto destino	16
Número de secuencia	32
Número de ACK	32
Longitud de cabecera TCP	4
Reservado	4
CWR (Congestion Window Reduced)	1
ECE (ECN Echo)	1
URG (Urgent)	1
ACK (Acknowledgement)	1
PSH (Push)	1
RST (Reset)	1
SYN (Synchronize)	1
FIN (Finish)	1
Tamaño de ventana	16
Checksum	16
Puntero de datos urgentes	16
Opciones	0, 32, 64, …
Datos	0-523960 (65495 bytes)

Tabla 6.3.- Estructura de la cabecera de un segmento TCP

Puerto origen y *puerto destino:* identifican los puertos que se van a utilizar en cada host para comunicar con las aplicaciones que intercambian datos.

Número de secuencia: indica el número de secuencia que corresponde en la conexión al primer byte que se envía en el campo datos de ese segmento.

Número de ACK: indica el número de secuencia del primer byte del próximo segmento que se espera recibir del otro lado.

Longitud de cabecera TCP: especifica la longitud en palabras de 32 bits, excluido el campo datos (el campo opciones hace que dicha longitud pueda variar).

A continuación hay 4 bits no utilizados, seguidos por ocho flags indicadores de un bit de longitud cada uno:

- CWR: Congestion Window Reduced. Tiene que ver con el control de congestión de IP que no describiremos aquí
- ECE: ECN Echo (ECN=Explicit Congestion Notification). Tiene que ver con el control de congestión de IP que no describiremos aquí.
- URG (urgent): sirve para indicar que el segmento contiene datos urgentes; en ese caso el campo puntero de datos urgentes contiene la dirección donde terminan éstos.
- ACK (acknowledgement): indica que en este segmento el campo *Número de ACK* tiene el significado habitual (número del próximo byte que se espera recibir), de lo contrario carece de significado. En la práctica el bit ACK esta a 1 siempre, excepto en el primer segmento enviado por el host que inicia la conexión.
- PSH (push): indica que el segmento contiene datos PUSHed. Esto significa que deben ser enviados rápidamente a la aplicación correspondiente, sin esperar a acumular varios segmentos.
- RST (reset): se usa para indicar que se debe abortar una conexión porque se ha detectado un error de cualquier tipo; por ejemplo una terminación unilateral de una conexión o que se ha

recibido un segmento con un valor inadecuado del *número de secuencia* o *número de ACK*, posiblemente producido por un duplicado retrasado de un intento de conexión.

- SYN (synchronize): este bit indica que se está estableciendo la conexión y está puesto sólo en el primer segmento enviado por cada uno de los dos hosts en el inicio de la conexión.

- FIN (finish): indica que no se tienen más datos que enviar y que se quiere cerrar la conexión. Para que una conexión se cierre de manera normal cada host ha de enviar un segmento con el bit FIN puesto.

Tamaño de ventana: indica la cantidad de bytes que se está dispuesto a aceptar del otro lado en cada momento. Se supone que se garantiza una cantidad suficiente de espacio en buffers. Mediante este parámetro el receptor establece un control de flujo sobre el caudal de datos que puede enviar el emisor.

Checksum: sirve para detectar errores en el segmento recibido; estos podrían ser debidos a errores de transmisión no detectados, a fallos en los equipos (por ejemplo en los routers) o a problemas en el software (por ejemplo reensamblado incorrecto de fragmentos). Recordemos que el datagrama IP contenía un checksum, pero aquel solo comprendía la información de cabecera y además ese checksum desaparece en IPv6. El algoritmo utilizado en TCP es el mismo que el de IP: sumar todos los campos de 16 bits usando aritmética de complemento a 1 y calcular el complemento a 1 del resultado, pero en este caso el checksum se hace sobre todo el segmento, incluidos los datos, no sólo sobre la información de cabecera. Para el cálculo del checksum se antepone al segmento una pseudocabecera que incluye la dirección IP de origen y destino, el protocolo de transporte utilizado (TCP en este caso) y la longitud del segmento. La pseudocabecera (así denominada porque sólo se utiliza en el cálculo, pero no forma parte del segmento) permite a TCP comprobar que no ha habido errores en la transmisión a nivel IP (detectar por ejemplo cuando un segmento ha sido entregado al host equivocado).

Puntero de datos urgentes: indica el final de éstos, ya que el segmento podría contener datos no urgentes. TCP no marca el principio de los datos urgentes, es responsabilidad de la aplicación averiguarlo.

El campo opciones habilita un mecanismo por el cual es posible incluir extensiones al protocolo. Entre las más interesantes se encuentran las siguientes:

- Tamaño máximo de segmento

- Uso de repetición selectiva (en vez de retroceso n)

- Uso de NAK (acuse de recibo negativo en caso de no recepción de un segmento)

- Uso de ventana mayor de 64 Kbytes mediante el empleo de un factor de escala

De las opciones hablaremos más adelante.

5.3.3 Tamaño de segmento y fragmentación

TCP divide (o agrupa) los mensajes recibidos del nivel de aplicación en TPDUs denominadas segmentos; en el momento de establecer la conexión cada host informa al otro del máximo tamaño de segmento que está dispuesto a aceptar; este tamaño deberá ser como mínimo de 536 bytes, correspondiente normalmente a un datagrama IP de 576 bytes menos 20 bytes de cabecera IP y 20 de cabecera TCP (la longitud de segmento se refiere a la parte de datos de TCP).

Un segmento TCP siempre se transporta en un datagrama IP. Cuando la red ha de fragmentar en algún punto un datagrama IP el datagrama *sigue fragmentado el resto del viaje hasta el host de destino*; una vez allí *el nivel de red* se ocupa de juntar todos los fragmentos en su buffer y reconstruir el datagrama original, del cual extrae entonces el segmento original y lo pasa a TCP; si uno de los fragmentos se pierde el nivel de red del receptor será incapaz de reconstruir el datagrama original, y por tanto descartará, una vez expirado el TTL, todos los fragmentos recibidos y no pasará ninguna parte de ese segmento al nivel de transporte; TCP agotará el timer, detectará el segmento faltante y pedirá retransmisión al emisor. Por tanto *cuando un fragmento de un datagrama se pierde todos los fragmentos se retransmiten nuevamente*;

el proceso se repite hasta que todos los fragmentos consiguen llegar correctamente al receptor. También puede suceder que la fragmentación se produzca ya en el host emisor del segmento, en cuyo caso realizará fragmentado todo el trayecto.

El Tanenbaum tercera edición (pag. 525) dice:

> *Un segmento que es demasiado grande para una red por la que debe transitar puede ser dividido en múltiples segmentos por un router. Cada nuevo segmento obtiene su propia cabecera TCP e IP, con lo que la fragmentación por routers aumenta el overhead total (porque cada segmento adicional añade 40 bytes de información de cabecera extra).*

El párrafo anterior contiene varios errores que es preciso aclarar. En primer lugar los routers no analizan ni procesan *segmentos*, sino *paquetes* o *datagramas*; por tanto tampoco fragmentan segmentos, sino paquetes; cada paquete-fragmento recibe una cabecera IP propia, pero no una nueva cabecera TCP; mas aun, el nivel de red ni siquiera analiza al fragmentar un paquete si lo que contiene es un segmento TCP, un datagrama UDP o cualquier otra de las posibilidades (podría ser un mensaje ICMP, por ejemplo). Por tanto el overhead añadido por la fragmentación no es de 40 bytes por fragmento (20 de la cabecera IP y 20 de la TCP), sino de 20 (mas el correspondiente a la información de control de las tramas adicionales que se produzcan a nivel de enlace). El nivel de transporte no se ve en absoluto involucrado en el proceso de fragmentación y reensamblado de los paquetes, que sólo afecta al nivel de red en el router (o host) que realiza la fragmentación y en el host al que va destinado el datagrama, que es el que efectúa el reensamblado. Afortunadamente el párrafo erróneo ha sido suprimido en la cuarta edición del libro (pág. 536).

En ocasiones el host emisor emplea la técnica de *descubrimiento de la MTU del trayecto* o '*Path MTU discovery*' (MTU=Maximum Transfer Unit) para averiguar el tamaño de datagrama máximo que puede utilizar, con lo que puede generar en origen los datagramas de tamaño apropiado evitando así la fragmentación en ruta. Para ello el host emisor envía un datagrama del tamaño máximo que pretende utilizar con el bit DF (Don't Fragment) puesto; si el datagrama llega a su destino deducirá que todo el trayecto soporta esa MTU; en caso contrario el host emisor recibirá un mensaje ICMP *Destination Unreachable* generado por el router donde se produzca el rechazo. En muchas implementaciones el mensaje ICMP de rechazo vendrá acompañado de una indicación del tamaño de MTU máximo que se habría tolerado en esa red, para facilitar así al host emisor la labor de tanteo. Cuando esto no ocurra el host emisor tendrá que deducir ese tamaño máximo a partir de una búsqueda binaria, hasta dar con el tamaño adecuado y reintentará con un datagrama más pequeño, hasta conseguir que llegue. En IPv6 la técnica de 'Path MTU discovery' es obligatoria, ya que la fragmentación en ruta no está permitida.

5.3.4 Flujo de datos en TCP

Los segmentos, al viajar en datagramas IP, pueden perderse, llegar desordenados o duplicados. Es responsabilidad de TCP resolver todos estos problemas y generar un flujo de bits fiable para el nivel de aplicación. Cada segmento es acomodado el solo en un datagrama, si bien luego puede tener que ser fragmentado para su envío, como hemos visto en el apartado anterior. Nunca se combinan en un segmento datos pertenecientes a dos conexiones TCP diferentes.

Las aplicaciones que comunican haciendo uso de los servicios que ofrece TCP no tienen conciencia de la existencia de segmentos o datagramas. Para ellas la comunicación se realiza como un flujo continuo de bits (stream), como si hubiera un hilo físico que las comunicara. Si desean que la información sea transmitida a la aplicación receptora en mensajes discretos deberán incluir los delimitadores adecuados, ya que no hay ninguna garantía de que TCP genere un segmento por cada mensaje recibido de la aplicación. TCP puede tomarse la libertad de agrupar o separar los datos recibidos de la aplicación según le convenga; por ejemplo, podría decidir retener los mensajes recibidos de la aplicación para agruparlos y crear segmentos de mayor tamaño usando así la red de manera más eficiente.

Para poder enviar datos prioritarios y responder ante situaciones urgentes existen dos mecanismos extraordinarios por los que las aplicaciones pueden indicar a TCP su deseo de enviar rápidamente datos al otro extremo.

Uno de ellos es el denominado indicador PUSH. Cuando una aplicación desea que los datos entregados a TCP salgan enseguida, sin esperar más datos de la aplicación, lo indica poniendo a 1 el indicador PUSH; este indicador se utiliza por ejemplo cuando en una transferencia FTP se envía el último segmento, o cuando en una sesión telnet el usuario pulsa la tecla return[3]; en estos casos si no se utilizara PUSH se correría el riesgo de que TCP se quedara esperando mas datos de la aplicación para construir un segmento mayor.

El otro mecanismo de envío rápido de datos se denomina *datos urgentes*, y como su nombre indica es aún más expeditivo que el anterior. Por ejemplo en una sesión telnet se utiliza este procedimiento para enviar los caracteres DEL, CTRL-C, o cuando se pulsa la tecla BREAK o ATTN. En estos casos no solo se desea enviar cuanto antes los caracteres recién tecleados, sino que se quiere que estos pasen por delante de cualesquiera otros que hubiera pendientes de enviar a la aplicación y se le entreguen a ésta sin esperar a que los solicite (por ejemplo para abortar una ejecución en marcha cuando ésta no espera leer datos).

La existencia de datos 'Pushed' y de datos urgentes se indica mediante los correspondientes bits indicadores o 'flags' en la cabecera TCP. En el caso de datos urgentes se indica además el punto en el segmento donde terminan éstos; es responsabilidad de la aplicación averiguar en que punto del segmento empiezan los datos urgentes.

5.3.5 Intercambio de información en TCP

El intercambio de segmentos en TCP se desarrolla de acuerdo con un protocolo de ventana deslizante con un número de secuencia de 32 bits. El número de secuencia cuenta bytes transmitidos, por lo que la secuencia se reinicia cada 4 GB (equivalentes a 4,3 minutos en una línea SDH de 155 Mb/s, suponiendo que toda la capacidad útil de la línea se utilizara para esa conexión TCP).

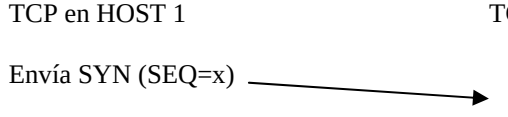
TCP utiliza ACK piggybacked, por lo que en la cabecera de cada segmento hay previstos dos campos de 32 bits, uno para el número de secuencia y otro para el número de ACK. El campo número de secuencia indica el número del primer byte transmitido dentro de ese segmento. El campo ACK indica el número del primer byte que se espera recibir en el siguiente segmento (o sea se sigue el estilo del campo 'next' en HDLC). En la práctica el ACK piggybacked raramente se aprovecha ya que la mayoría de las aplicaciones generan tráfico muy asimétrico; normalmente uno de los dos TCP se ve oligado a enviar muchos segmentos vacíos con el único fin de informar al emisor que los datos han sido recibidos.

El mecanismo normal de funcionamiento de TCP es retroceso n, aunque también puede utilizarse repetición selectiva si las dos entidades participantes lo soportan y lo negocian al iniciar la conexión.

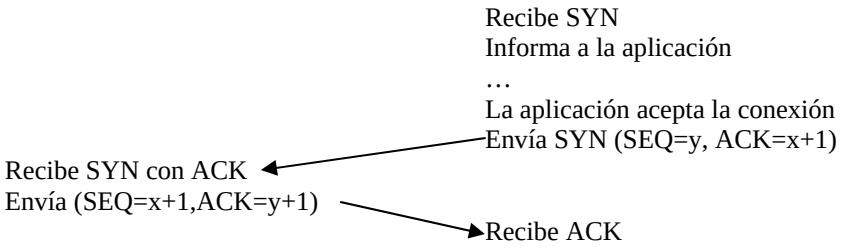
Cada segmento enviado incluye un tamaño de ventana en el que el receptor anuncia la cantidad de datos que está dispuesto a recibir. De esta forma el receptor puede ejercer control de flujo sobre el emisor. El tamaño de ventana es un campo de 16 bits, por lo que el valor máximo es de 64 Kbytes.

5.3.6 Gestión de conexión TCP

El mecanismo utilizado en TCP para establecer una conexión es el de saludo a tres vías. Un proceso normal sería el siguiente:



[3] En realidad esta afirmación solo es correcta cuando se utiliza telnet con eco local; en la mayoría de los casos el telnet utiliza eco remoto, es decir el telnet remoto (servidor) ha de procesar lo que se teclea carácter a carácter, realizando la representación en pantalla si procede. Cuando se funciona con eco remoto el telnet cliente pone el bit PUSH para cada carácter que se teclea. Aunque es mucho menos eficiente que el eco local el eco remoto es hoy en dia la forma de funcionamiento más habitual, ya que muchos editores de pantalla completa (el vi por ejemplo) necesitan eco remoto para funcionar correctamente.



En el primer segmento el host 1 indica al host 2 que desea establecer una conexión (bit SYN puesto) y le informa del número de secuencia que ha elegido (x); el host 2 le responde con otro segmento en el que acepta la conexión (bit SYN puesto) y le indica el número de secuencia que él ha elegido para las transmisiones en el sentido contrario (y); además le acusa recibo de su número de secuencia al indicarle en el ACK que el próximo byte que espera recibir de él es x + 1. Host 1 responde con un tercer mensaje en el que acusa recibo del número de secuencia de host 2.

En este ejemplo hemos supuesto el caso más simple de establecimiento de una conexión. El Host 1 podría incluír ya en esos primeros segmentos datos dirigidos a la aplicación; esto está permitido siempre y cuando los datos sean retenidos por el TCP receptor hasta que la aplicación correspondiente decida si acepta la conexión.

Una vez establecida la conexión puede empezar el intercambio de información; cada lado puede enviar datos al otro de forma independiente, sin necesidad de que el otro le corresponda con más datos. Normalmente si fluyen datos en ambos sentidos los ACK irán incluidos (‘piggybacked’) en los segmentos de vuelta, pero si el tráfico discurre predominantemente en un sentido (como es lo normal) se producirán segmentos vacíos con el único fin de acusar recibo de los envíos. Los valores de SEQ y ACK se van incrementando a medida que progresa la transmisión. Los valores de ventana anunciados por cada host permiten al otro conocer la disponibilidad que tiene de recibir datos.

El número de secuencia inicial es elegido por cada host de forma pseudoaleatoria. El RFC 793, que describe el estándar TCP, recomiendan utilizar para esto un contador que se incremente en una unidad cada 4μs, aproximadamente; esto se puede conseguir fácilmente basándose en algún reloj del sistema. Con este algoritmo el número de secuencia se da la vuelta cada 4 horas 46 minutos, aproximadamente. Esta forma de elegir el número de secuencia inicial reduce la probabilidad de que si uno de los dos hosts cae y rearranca pueda coincidir el número de secuencia nuevo con el viejo, lo cual podría producir que se tomaran como válidos segmentos duplicados retrasados, o que el TCP del otro lado continuara dialogando con el nuevo proceso como si fuera el viejo. Para aumentar aún más la seguridad el estándar recomienda que se garantice una separación mínima en el timepo de 2 minutos desde que cae un TCP hasta que se levanta el nuevo, para asegurar de esa forma que no pueden aparecer en el TCP de destino duplicados retrasados que puedan mezclarse en la nueva conexión (2 minutos es un valor máximo bastante normal del parámetro TTL o tiempo de vida, que fija el tiempo máximo que un datagrama puede estar pululando por la red antes de desaparecer).

Para comprender hasta que punto es importante la no coincidencia de números de secuencia entre sesiones imaginemos la siguiente situación: dos usuarios X e Y mantienen ambos una conexión telnet desde la máquina 167.172.23.43 a la máquina 144.38.76.3; en un primer momento sus conexiones son:

Usuario X:	167.172.23.43.1024 con 144.38.76.3.23
Usuario Y	167.172.23.43.1025 con 144.38.76.3.23

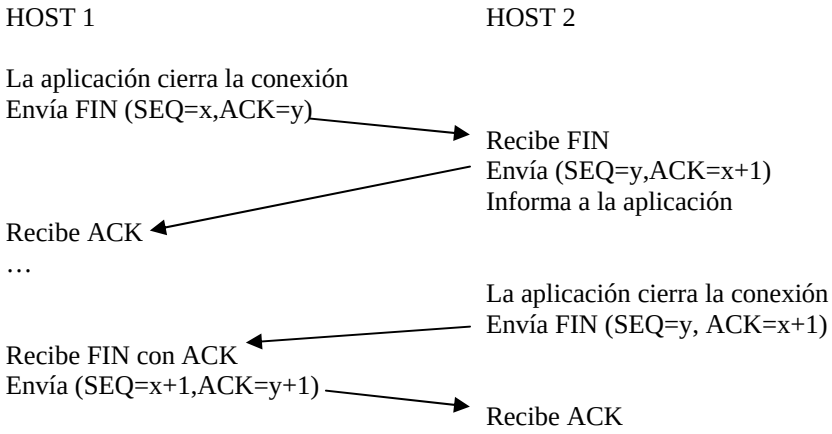
1024 y 1025 son los puertos utilizados por los procesos clientes telnet en 167.172.23.43 y 23 es el puerto utilizado por el proceso servidor telnet en 144.38.76.3.

Supongamos ahora que el host 167.172.23.43 (cliente) cae y se levanta a continuación, y que los dos usuarios reanudan sus respectivas sesiones telnet (que han quedado medio abiertas en el servidor) pero ahora se conecta Y antes que X. Las conexiones antiguas han desaparecido, y dado que los puertos se asignan por orden de llegada ahora se asignarían en orden inverso:

Usuario X:	167.172.23.43.1025 con 144.38.76.3.23
Usuario Y	167.172.23.43.1024 con 144.38.76.3.23

En condiciones normales los clientes telnet intentarán abrir nuevas conexiones, con lo que el servidor cerrará las antiguas. Pero si de alguna forma los clientes telnet entraran en las conexiones viejas del servidor sin cerrarlas cada usuario entraría directamente en la sesión del otro sin necesidad de identificarse con la contraseña correspondiente, lo cual evidentemente no es aceptable. La utilización de números de secuencia grandes y aleatorios suministra un cierto nivel de seguridad ante estas situaciones.

Para terminar una conexión TCP se utiliza normalmente el procedimiento simétrico del saludo a tres vías, en el cual cada lado cierra su parte de forma independiente como si se tratara de una conexión simplex. El intercambio de mensajes típico es el siguiente:



5.3.7 Estados de TCP

El software (o proceso) TCP de un host puede mantener en un momento dado múltiples conexiones simultáneas con homólogos suyos en otros hosts. Para cada una de esas conexiones TCP conserva en todo momento información del estado en que se encuentra (por ejemplo conexión establecida, pendiente de establecer o no conexión).

La secuencia de estados que se suceden en el establecimiento de una conexión TCP en un servidor aparece en la tabla 10.4, mientras que la tabla 10.5 nos muestra la secuencia equivalente en el caso de un cliente.

Estado del servidor	Evento	Descripción
CLOSED		Estado (ficticio) en que se encuentra una conexión antes de existir
	La aplicación en el servidor hace una apertura pasiva	
LISTEN		El servidor espera una conexión del cliente
	El TCP del servidor recibe un SYN, devuelve un SYN/ACK	
SYN-RECEIVED		El servidor espera un ACK
	El TCP del servidor recibe un ACK	
ESTABLISHED		El ACK ha sido recibido y la conexión se ha establecido

Tabla 10.4.- Secuencia de estados que ocurren en el establecimiento de una conexión TCP en un servidor

Estado del cliente	Evento	Descripción
CLOSED		Estado (ficticio) en que se encuentra una conexión antes de existir
	La aplicación cliente solicita una conexión. El TCP del cliente envía un SYN	
SYN-SENT		El TCP cliente ha enviado un SYN al servidor
	El TCP cliente recibe SYN/ACK y envía ACK	
ESTABLISHED		La transferencia de datos puede comenzar

Tabla 10.5.- Secuencia de estados que ocurren en el establecimiento de una conexión TCP en un cliente

Una vez en el estado ESTABLISHED ambos TCP permanecerán así hasta que se inicie el procedimiento de cierre de la conexión. En condiciones normales cualquiera de los dos procesos puede iniciar la desconexión enviando un segmento con el bit FIN puesto; la secuencia de estados en el TCP que inicia la desconexión es la que se muestra en la tabla 10.6, mientras que la tabla 10.7 muestra la secuencia de estados en el TCP que es 'invitado' a cerrar la conexión.

Estado del TCP que cierra la conexión	Evento	Descripción
ESTABLISHED	La aplicación local solicita cerrar	
	TCP envía FIN/ACK	
FIN-WAIT-1		El TCP local está esperando respuesta del otro lado. En este punto aún pueden llegar datos válidos.
	TCP recibe un ACK	
FIN-WAIT-2		El TCP local ha recibido un ACK del otro lado, pero no un FIN. Se siguen aceptando los datos que pudieran llegar del otro lado
	TCP recibe FIN/ACK. Envía ACK	
TIME-WAIT		La conexión se mantiene en el limbo ante la posibilidad de recibir aun datos duplicados o un FIN duplicado. El tiempo de espera es igual al doble del tiempo de vida de un segmento
CLOSED		Se borra toda la información relativa a esta conexión.

Tabla 10.6.- Secuencia de estados que ocurren en el cierre de una conexión TCP en el host que inicia la desconexión

Estado del TCP 'invitado' a cerrar	Evento	Descripción
ESTABLISHED	TCP recibe FIN/ACK	
CLOSE-WAIT		Ha llegado un FIN
	TCP envía un ACK	
		TCP espera a que su aplicación cierre la conexión. La aplicación podría aún enviar más datos
	La aplicación local cierra la conexión	
	TCP envía FIN/ACK	
LAST-ACK		TCP Está esperando el ACK final
	TCP recibe un ACK	
CLOSED		Se borra toda la información sobre la conexión

Tabla 10.7.- Secuencia de estados que ocurren en el cierre de una conexión TCP en el host que es invitado a terminar la conexión

Los nombres utilizados en estas tablas corresponden con los empleados en el RFC 793, y en muchas implementaciones de TCP. El comando *netstat -an*, que permite examinar el estado de las conexiones existentes en un host UNIX, utiliza esta misma nomenclatura.

5.3.8 Conexiones medio abiertas y timer de keepalive

En principio el estándar TCP no establece el requerimiento de que una conexión TCP tenga un tráfico mínimo para permanecer establecida. Cabría pensar en la posibilidad de que una conexión TCP estuviera abierta durante días sin transmitir ni un solo segmento, y en principio no habría razón para terminarla. En la práctica esto supone que si en una conexión desaparece uno de los dos TCP el otro podría quedar eternamente esperando dándose lo que se denomina una conexión medio abierta. Si todo funcionara como es debido las conexiones medio abiertas nunca deberían ocurrir, puesto que cada TCP debería cerrar ordenadamente sus conexiones. Pero a veces los procesos TCP terminan de forma abrupta, no dando tiempo al cierre ordenado de sus conexiones; esto es especialmente común en tiempos recientes a partir de la proliferación de los ordenadores personales conectados directamente a Internet en los que el usuario con frecuencia no termina los procesos de la forma correcta, o desconecta de la red su ordenador dejando las conexiones medio abiertas en el otro lado.

Esas conexiones medio abiertas consumen recursos inútilmente, por lo que es conveniente establecer algún mecanismo por el cual un TCP pueda ‘sondear’ periódicamente sus conexiones para comprobar que el otro lado aún está operativo; si el TCP de una conexión deja de responder durante un número determinado de mensajes de sondeo se considera que esa conexión está medio abierta y se procede a cerrarla de la forma mas civilizada posible, liberando así los recursos que está utilizando.

Los mensajes de sondeo son lo que se conoce como mensajes ‘keep-alive’ y la periodicidad con la que se producen viene fijada por el parámetro denominado timer de keepalive. Los mensajes de keepalive fueron añadidos a posteriori al TCP, y se implementan de una forma muy sencilla: el TCP envía un segmento que repite el último byte enviado; el TCP receptor no pasará este byte a la aplicación, pero debe generar un segmento de ACK; con esto el emisor ya sabe que su interlocutor está operativo.

Los keepalive se suelen implementar en servidores, que son los que mas sufren el problema de las conexiones medio abiertas. Por la forma como se implementan los mensajes de keepalive funcionan aun cuando el TCP remoto no implemente keepalive, ya que se emplea una característica que forma parte del funcionamiento estándar de TCP. El mecansimo de keepalive no debe ser tan estricto que una pérdida momentánea de coenctividad produzca el cierre de una conexión TCP, ya que en una red como Internet es fundamental permitir que las cosas funcionen aun cuando haya fallos momentáneos en el nivel de red, que sabemos que no es fiable. El timer de keepalive puede tener valores en torno a los dos minutos; los tiempos recomendados para cortar una conexión TCP inactiva pueden llegar a ser de hasta dos horas.

5.3.9 Política de transmisión de TCP

Como ya hemos comentado, el receptor en una transmisión TCP anuncia regularmente el tamaño de ventana que tiene disponible para que el emisor sepa cuantos datos más puede enviarle. Cuando un receptor tiene lleno su buffer anuncia una ventana de 0 bytes, con lo que el emisor queda bloqueado hasta nueva orden.

Existen dos circunstancias en las que con ventana cero (ventana cerrada) el emisor puede enviar datos. Una es cuando se presentan datos urgentes; estos siempre deben ser aceptados por el receptor, ya que se supone que no pueden esperar. La otra excepción es que el emisor puede siempre enviar un segmento de un byte de datos, para forzar al receptor a devolver un ACK y así comprobar cual es la situación; esto evita situaciones de bloqueo que de otro modo podrían producirse por la pérdida de segmentos ACK en la red. La periodicidad con la que el emisor 'tantea' al receptor con segmentos de un byte para comprobar que su ventana sigue cerrada se fija con el parámetro conocido como timer de persistencia.

Salvo por lo requerido en el bit PUSH o en datos urgentes, los emisores TCP pueden organizarse los envíos como mejor les convenga. TCP podría por ejemplo decidir agrupar la información que recibe de la aplicación para enviar segmentos mas grandes y así reducir el overhead de proceso y de cabeceras que supone el envío de segmentos pequeños. Incluso el uso del bit PUSH no garantiza la inmediata expedición de un segmento, en situaciones de veradera congestión el TCP puede decidir ignorar el bit Push y agrupar varios mensajes de la aplicación en el mismo segmento.

5.3.10 Problemas de paquetes pequeños

5.3.10.1 Algoritmo de Nagle

Un caso extremo de baja eficiencia se produce cuando la aplicación en el lado del emisor genera un byte de datos cada vez; imaginemos por ejemplo lo que ocurre cuando se efectúa una conexión telnet mediante una emulación de terminal ANSI (VT100 o similar); muchos programas, como por ejemplo el editor vi de UNIX, necesitan para funcionar correctamente que el host remoto procese cada carácter que se teclea, por lo que es preciso utilizar el modo de eco remoto mediante el cual el host remoto se encarga de representar en pantalla cada carácter que se teclea. En estas condiciones por cada tecla pulsada la aplicación envía a TCP los caracteres uno a uno; en principio TCP debería enviar cada carácter en un segmento conteniendo 20 bytes de cabecera al cual el nivel de red añadiría 20 de la cabecera IP; ese segmento TCP recibe a continuación su correspondiente segmento vacío (es decir sin datos) con el ACK del anterior. Como la representación en pantalla en el host local se realiza a través del host remoto instantes más tarde la aplicación del sistema remoto (el servidor telnet) responde con el eco del carácter pulsado, que es de nuevo un datagrama IP de 41 bytes, a lo cual el host cliente responde con otro datagrama IP de 40 bytes que contiene otro segmento ACK vacío; en total se transfieren 162 bytes para dos caracteres transmitidos, lo cual da una eficiencia del 1,2% (2/162) (y aquí no hemos considerado la información de control del nivel de enlace, tramas Ethernet por ejemplo, que añadiría aún más overhead).

Para evitar estas situaciones la mayoría de las implementaciones de TCP fijan un timeout de 500 mseg antes de enviar un ACK vacío; si en ese tiempo se genera algún tráfico de vuelta el ACK puede viajar 'piggybacked' en él; por ejemplo en nuestro caso salvo que el host estuviera muy cargado la aplicación telnet respondería antes de 500 mseg y el ACK podría viajar en el mismo segmento que lleva el carácter de vuelta; de esta forma la eficiencia mejora ya que se suprime un ACK; la eficiencia sería entonces del 1,6 % (2/122).

Para mejorar aún más la eficiencia en estas situaciones se utiliza lo que se conoce como el *algoritmo de Nagle*. La idea es muy simple: cuando el tráfico de la aplicación llega al TCP en bytes uno por uno se envía el primero en un segmento y se retienen los demás hasta recibir el ACK correspondiente al byte enviado; una vez recibido el ACK se envía un segmento con todos los bytes que hubiera pendientes y se empieza a acumular de nuevo hasta recibir el siguiente ACK. También se envía un segmento si el número de caracteres acumulados en el buffer es igual a la mitad de la ventana, o al máximo tamaño del segmento.

En cierto modo el algoritmo de Nagle sustituye el protocolo de ventana deslizante por un mecanismo de parada y espera.

El algoritmo de Nagle es autoadaptativo, ya que en una red muy cargada los ACK tardarán más en llegar, con lo que automáticamente los segmentos que se envíen serán mayores y el overhead disminuirá; el usuario observará una latencia mayor en la red, pero esto es un mal menor cuando se trata de reducir la congestión en la red. Cuando la red esté descargada y responde muy rápido cada carácter tecleado viaja en un segmento independiente , y será respondido con otro que contendrá el carácter de eco, sin más retardo que el tiempo que tarde el host en responder.

El algoritmo de Nagle se puede aplicar a datos 'Pushed' pero no a datos urgentes, y se debe inhibir cuando se desea transferir información en tiempo real; por ejemplo la posición del ratón en una sesión X-Windows debe ser transmitida inmediatamente ya que de lo contrario el movimiento en pantalla resulta errático (como es bien sabido el uso de terminales X es poco eficiente y requiere gran cantidad de recursos).

5.3.10.2 Síndrome de la ventana tonta y solución de Clark

Imaginemos ahora la situación inversa; en vez de enviar los datos byte a byte es la aplicación en el TCP receptor la que recoge los bytes de uno en uno. La situación que se daría sería la siguiente:

1. El TCP emisor envía datos sin parar al TCP receptor
2. El buffer del TCP receptor se llena
3. El TCP receptor notifica al emisor que su ventana está cerrada (ventana 0)
4. El TCP emisor deja de enviar datos
5. La aplicación receptora lee 1 byte del buffer de TCP
6. El TCP receptor envía un ACK al emisor para anunciarle que dispone de 1 byte de espacio
7. El TCP emisor envía un segmento con 1 byte de información útil
8. Volvemos al punto 2.

El bucle se repite hasta terminar la sesión. Se están generando como antes segmentos con un byte de información útil, pero esta vez el causante es el receptor que los recoge en pequeñas dosis. Este comportamiento se conoce como *síndrome de la ventana tonta*.

La solución a esto, propuesta por Clark en el RFC 813, consiste en que el TCP del receptor no debe notificar el cambio de ventana al emisor entretanto no tenga una cantidad razonable de espacio libre en su buffer; por razonable se entiende cualquiera de las dos condciones siguientes: el espacio suficiente para aceptar un segmento de la longitud máxima admitida en esa conexión, o la mitad del espacio total del buffer.

El algoritmo de Nagle y la solución de Clark son dos mecanismos complementarios que intentan resolver un mismo problema: el causado por la transmisión innecesaria de segmentos pequeños.

5.3.11 Control de congestión en TCP

Como ya hemos explicado el TCP receptor puede controlar el flujo de datos que recibe del emisor anunciando un valor adecuado del tamaño de ventana. Si el receptor anuncia un tamaño de ventana 0 (lo que se conoce como 'cerrar la ventana') el emisor dejará de transmitir hasta nueva orden (salvo por las dos excepciones ya mencionadas, datos urgentes y segmentos con un byte).

Sin embargo la falta de espacio de buffer en el receptor es solo una de las causas por las que el emisor puede verse obligado a bajar el ritmo de emisión; la otra es la presencia de congestión en la red. Para aclarar la diferencia entre estos dos importantes conceptos imaginemos el siguiente experimento:

- Un supercomputador establece una conexión TCP con un ordenador personal poco potente a través de una conexión directa ATM OC3 (155,52 Mb/s) 'back to back'. El circuito se establece mediante la categoría de servicio UBR. Al transferir datos se mide un rendimiento máximo de 50

Mb/s; analizando la situación se observa que el ordenador personal tiene su buffer lleno prácticamente todo el tiempo, por lo que su TCP está continuamente cerrando su ventana; evidentemente el subsistema de entrada/salida del ordenador personal no es capaz de absorber los datos al ritmo que los envía el supercomputador.

- En una segunda prueba se repite la misma transferencia entre ambos ordenadores, pero en vez de un enlace directo se conectan a través de una red ATM pública (utilizando también un servicio UBR). En este caso el rendimiento de la transferencia en horas punta es de solo 10 Mb/s, aun cuando se observa que el TCP del ordenador personal tiene espacio de sobra en sus buffers de entrada.

La diferencia estriba en que en el primer caso está actuando como factor limitante el control de flujo en el receptor y en el segundo la congestión en la red. Presumiblemente la red pública no es capaz en horas punta de dedicar a esa conexión los recursos necesarios para transmitir los 50 Mb/s que puede absorber el ordenador personal. Basta que uno solo de los enlaces del trayecto se vea afectado de congestión para limitar el tráfico en todo el trayecto, y por tanto el rendimiento de la comunicación entre los hosts.

Imaginemos por un momento que pasaría si TCP no incluyera ningún mecanismo de control (mejor dicho de autocontrol) en situaciones de congestión; dado que el receptor anuncia buffers disponibles el emisor inyectará paquetes en la red al ritmo que se lo permita su conexión física; cuando los paquetes lleguen a la parte congestionada de la red, no pudiendo aceptar todo el tráfico entrante, los routers empezarán a acumularlos en sus buffers y cuando estos se llenen los descartarán; en el lado del TCP receptor se recibirán solo una parte de los segmentos, por lo que o bien se devolverán segmentos NAK al emisor, o bien sencillamente no se enviarán los ACKs y se esperará que el emisor reenvíe por timeout; en cualquier caso el emisor tiene que reenviar datos. Los segmentos descartados en ruta son inútiles y cargan innecesariamente las líneas por las que pasan; este tráfico inútil podría propagar la congestión a zonas de la red que en principio no la sufrían, aumentando así la magnitud del problema y disminuyendo aún mas el rendimiento en toda la red. Este efecto de 'bola de nieve' se denomina colapso de congestión ('congestion collapse') y puede llegar a dejar toda una red completamente fuera de servicio.

Evidentemente TCP no puede ser ajeno a las situaciones de congestión en la red, pero como notificamos al emisor que hay saturación en algún punto del trayecto y que debe bajar el ritmo con que envía datos? TCP no emplea mecanismos explícitos para notificar la congestión. El mecanismo que emplea es indirecto y se basa en la pérdida de datagramas por la red. Esto se basa en una premisa fundamental: el medio físico se considera *altamente fiable* por lo que siempre que el TCP emisor detecte que los segmentos no han sido recibidos en su destino (al no recibir los correspondientes ACKs) deducirá que la red está descartando paquetes por congestión y bajará el ritmo de sus envíos. Dado que el control de congestión de TCP se basa en la fiabilidad del medio físico cuando esta condición no se cumple (radioenlaces móviles, por ejemplo) es preciso realizar modificaciones a los algoritmos normales de TCP o incorporar en el nivel de enlace mecanismos de comprobación o corrección de errores que suministren esa fiabilidad, ya que de lo contrario TCP interpreta como congestión los problemas debidos al medio físico y si en estas condiciones se baja el ritmo el rendimiento decrece aún más.

Para autoregularse el TCP emisor maneja una *ventana de congestión* que le indica que cantidad de datos puede inyectar en la red en un momento dado. Dicha ventana actúa simultáneamente y en paralelo a la ventana que anuncia el receptor indicando los buffers disponibles, que podríamos denominar *ventana de control de flujo*. En cada momento el emisor tomará en consideración la mas pequeña de las dos ventanas, para asegurarse de que no satura al receptor y que tampoco provoca, o contribuye a agravar, una situación de congestión en la red.

Mientras que la ventana de control de flujo es notificada al emisor por el receptor, la ventana de congestión es calculada por el emisor a partir de la cantidad de retransmisiones que tiene que realizar; cuando ve que no se produce ninguna retransmisión va aumentando paulatinamente la ventana, hasta llegar al punto donde falla algún segmento (es decir, agota el timer y se retransmite), momento en el cual la reduce (suponiendo que la ventana de control de flujo no imponga ninguna limitación). Generalmente la ventana crece de froma lenta y gradual, mientras que la reducción se lleva a cabo de manera drástica. Los algoritmos de crecimiento y disminución de la ventana de congestión en TCP son siempre autoadaptativos y forman una parte fundamental del rendimiento del protocolo; estos algoritmos han sido y son objeto de detallados estudios y experimentaciones, por lo que son altamente sofisticados y funcionan bien en situaciones muy diversas.

Inicialmente la ventana de congestión se fija a un valor igual al del máximo tamaño de segmento negociado en el momento de establecer la conexión (que depende a su vez del MTU); supongamos por ejemplo que dicho tamaño es de 1 Kbyte y que todos lols segmentos que se van a generar tendrán este tamaño. Inicialmente TCP envía un segmento de 1 KByte e inicia un timer; si se recibe el ACK antes de expirar el timer significa que el segmento ha llegado a su destino correctamente, por lo que la ventana de congestión se amplía a 2 KBytes; a continuación se envían 2 segmentos, y se inicia el timer (en realidad 2 timers, uno por segmento); por cada segmento confirmado dentro del intervalo previsto se amplía la ventana en un segmento (1 KByte), por lo que, suponiendo que no se pierda ninguno, en el ciclo siguiente la ventana pasará de 2 a 4 KBytes; en condiciones normales esto supone que la ventana crece exponencialmente, ya que se duplica en cada envío; esta técnica se denomina *slow-start*, aunque no es precisamente lenta, sino todo lo contrario; por ejemplo empezando en 1 KByte en sólo 7 iteraciones llegaría a 64 Kbytes, valor máximo permitido por el tamaño de ventana de TCP. En condiciones normales (sin congestión) el slow-start provoca que la ventana de congestión crezca rápidamente, con lo que pronto supera a la ventana de control de flujo, momento a partir del cual prevalece ésta y la ventana de congestión deja de crecer.

Pero que ocurre cuando se detecta la pérdida de un segmento, es decir, hay congestión en la red? Supongamos en nuestro ejemplo que hemos ido duplicando el tamaño de la ventana de congestión hasta llegar a 32 KBytes, momento en el cual la red pierde un datagrama IP, es decir expira un timeout. Cuando ocurre esto el slow-start entra en una nueva fase. De entrada la ventana de congestión se reduce, y lo hace de una manera drástica, al valor inicial de un segmento (1 KByte en nuestro ejemplo). Además se establece un 'umbral de peligro' en un valor igual a la mitad del tamaño que tenía la ventana cuando se produjo la retransmisión (en nuestro ejemplo el umbral de peligro sería 16 KBytes) Ahora la ventana de congestión empieza a crecer como antes, pero sólo hasta el umbral de peligro; a partir de ahí la ventana se incrementa de forma mucho más lenta en sólo un segmento cada vez (un proceso que podríamos llamar *'very-slow-start'*); cada vez que se produce una retransmisión se recalcula el umbral de peligro como la mitad de la ventana vigente en ese momento, y se empieza una nueva fase. El proceso se repite indefinidamente mientras dure la transmisión.

Supongamos que la ventana de control de flujo del receptor es siempre mayor que la de congestión (y por tanto no se ejerce control de flujo) y que la pérdida de paquetes se presenta siempre justo cuando la ventana de congestión supera los 20 KBytes; supongamos también que todos los segmentos que se transmiten son de 1 Kbyte; la evolución de la ventana de congestión sería entonces la siguiente:

Fase	**Umbral de peligro (en Kbytes)**	**Tamaños sucesivos de la ventana (en Kbytes)**
Primera	64 (valor por defecto)	1,2,4,8,16,32
Segunda	16	1,2,4,8,16,17,18,19,20,21
Tercera	10,5	1,2,4,8,10,11,12,13,14,15,16,17,18,19,20,21

A partir de la tercera fase el proceso se repite indefinidamente. Cabría pensar que el mecanismo no se estabiliza nunca, y en efecto así es; aun en el caso de que fijáramos la ventana de congestión en el valor de 20 KBytes tampoco se estabilizaría, ya que en la práctica el tamaño de la ventana de congestión cambia continuamente. En última instancia la única forma que tiene el emisor de ajustar la ventana de congestión al límite de sus posibilidades es tanteando y fallando de vez en cuando. Se podría argumentar que no es preciso retroceder en tan gran medida en caso de fallo, pero recordemos que cuando se produce congestión en una red es mejor pasarse de precavido, pues de lo contrario el problema se puede hacer inmanejable.

5.3.12 Gestión de timers en TCP

Hasta ahora hemos supuesto que TCP era capaz de detectar los segmentos perdidos fijando un valor adecuado para el timer de retransmisión; en realidad para estar completamente seguros de que no va a llegar el segmento ACK habría que esperar dos veces el valor del TTL, lo cual en prácticamente todos los casos es excesivo. Para el timer se suele elegir un tiempo por encima del cual la probabilidad de recibir el ACK sea muy pequeña. La elección de un valor adecuado para este timer tiene una consecuencia directa en el funcionamiento eficiente de TCP; si el timer es demasiado alto el emisor esperará innecesariamente

en muchos casos por ACKs que nunca llegarán, y si es demasiado bajo se producirán reenvíos innecesarios de segmentos que habían sido correctamente recibidos.

La elección de valores de timer adecuados es mucho más compleja en el nivel de transporte que en el nivel de enlace. Además de las fluctuaciones naturales debidas a las diferencias en capacidad y retardo de unas conexiones a otras, el nivel de transporte ha de hacer frente a oscilaciones debidas a la presencia de elementos intermedios (routers y enlaces) y de situaciones de congestión que están fuera de su control; aun en ausencia de congestión los routers pueden tener largas colas de paquetes que atender, los enlaces pueden ser de diversas velocidades, y la ruta puede variar durante la conexión.

Por todo esto los valores del timer de retransmisión en el nivel de transporte se establecen mediante algoritmos autoadaptativos que dinámicamente ajustan los valores al estado de la red, según es percibido éste por el nivel de transporte en el host emisor.

El algoritmo utilizado en TCP para el cálculo de los timers fue diseñado por Van Jacobson, que es también el autor de la técnica slow-start que hemos visto antes. En realidad la gestión de los timers es una parte del slow-start necesaria para un efectivo control de la congestión en TCP.

Para estimar el timer de retransmisión TCP mide lo que tardan en llegar los ACK de los segmentos enviados; se supone que estos tiempos son una buena estimación del tiempo de ida y vuelta o RTT (Round Trip Time) de los segmentos, en base al cual ha de calcularse el valor del timer de retransmisión. El valor medio del RTT (que denominaremos MRTT) se estima mediante la fórmula iterativa siguiente:

$$MRTT_n = \alpha\, MRTT_{n-1} + (1-\alpha)\, RTT_n \qquad (10.1)$$

donde RTT_n es el tiempo de ida y vuelta medido para el último (n-ésimo) ACK recibido. El parámetro α permite ajustar el peso o la importancia que se quiere dar al último valor frente a los anteriores; con un α pequeño se consigue que los valores anteriores tengan poca relevancia, adaptándose así a situaciones cambiantes con rapidez. Con α grande se reacciona con más inercia a los cambios. En TCP α vale normalmente 7/8. Lo que calcula esta fórmula es pues una media aritmética ponderada de los valores de RTT, dándoles un peso inversamente proporcional a su antigüedad (cuanto más viejo menos importante).

Obtener una buena estimación del valor medio de RTT resuelve sólo una parte del problema; sabemos que los valores de RTT se distribuirán alrededor del valor medio, pero cual es el valor adecuado del timer de retransmisión, o sea, ¿cual es el valor umbral a partir del cual podemos considerar que el ACK no llegará?. Las primeras implementaciones de TCP utilizaban 2*MRTT como valor del timer, pero eso tenía el inconveniente de que cuando los valores de RTT fluctuaban mucho el umbral de 2*MRTT resultaba demasiado bajo y se producían excesivas retransmisiones innecesarias; en cambio cuando la dispersión era pequeña 2*MRTT daba un valor excesivo ya que en la mayoría de los casos no había que esperar tanto para dar por perdido un segmento. Para tener una estimación más precisa del timeout necesitamos saber además del valor medio el grado de dispersión, o dicho de otro modo conocer la anchura de la campana de distribución de los valores, lo que en estadística se conoce como la desviación estándar. Para estimar esta magnitud de manera sencilla se utiliza la siguiente fórmula:

$$D_n = \beta\, D_{n-1} + (1-\beta)\, \left| MRTT_{n-1} - RTT_n \right| \qquad (10.2)$$

Como antes β es un factor que permite regular la inercia a los cambios (a mayor β mayor inercia); normalmente suele valer 3/4[4]. De forma parecida al cálculo de MRTT el valor actual es una media ponderada del valor instantáneo y de los valores anteriores, con un peso decreciente en función de la antigüedad.

Una vez obtenidos MRTT y D podemos calcular el timeout de retransmisión. Para esto se utiliza generalmente la fórmula siguiente:

$$\text{Timeout de retransmisión} = MRTT + 4 * D \qquad (10.3)$$

[4] Se suelen preferir en estas fórmulas los factores que son potencias enteras de 2 en el denominador ya que esto hace más sencillos, y por tanto más rápidos, los cálculos.

El cálculo del RTT de los segmentos reenviados plantea un problema. No es posible saber con seguridad si el ACK se debe al primer o al segundo envío y una interpretación errónea podría alterar de manera importante el valor de MRTT. La solución a este problema, conocida como *algoritmo de Karn*, consiste sencillamente en ignorar a efectos del cálculo del MRTT (y de D) los ACK de los segmentos que son retransmitidos. Sin embargo esto plantea otro problema: podría ocurrir que el RTT aumentara de forma repentina (por ejemplo por un cambio en la ruta de los datagramas) hasta el punto que superara el timeout; a partir de ese momento todos los segmentos serían retransmitidos, con lo que los valores de MRTT y D (y por tanto el timeout de retransmisión) se mantendrían constantes, puesto que los ACK recibidos serían ignorados; estas múltiples retransmisiones provocarían una notable pérdida de eficiencia. Para evitarlo el algoritmo de Karn prevé que cada vez que se produzca una retransmisión el timeout de retransmisión se duplique; de esta forma si por alguna razón el RTT aumenta de forma repentina el timer de retransmisión crece rápidamente evitando así que se produzcan muchas retransmisiones; una vez el TCP emisor vuelve a recibir ACKs de segmentos no retransmitidos vuelve a calcular el tiemout de retransmisión a partir de los nuevos valores de MRTT y D de acuerdo con la fórmula 10.3. Esta técnica, denominada *retroceso exponencial* del timer, tiene el interesante efecto colateral de reducir el tráfico en situaciones de congestión.

5.3.13 Opciones del protocolo TCP

El protocolo TCP está en evolución permanente; las mejoras se experimentan en prototipos y luego se documentan en RFCs, convirtiéndose en extensiones opcionales al protocolo básico. Las extensiones son generalmente compatibles entre sí y se van incorporando paulatinamente en muchas implementaciones de TCP. Cuando dos TCPs conectan negocian entre ellos la relación de extensiones que cada uno quiere utilizar, y emplean solo aquellas que están soportadas por ambos. Veamos algunas de esas extensiones.

El máximo tamaño de ventana estándar de TCP es de 64 KBytes - 1. El tamaño de ventana establece la máxima cantidad de datos que pueden estar pendientes de confirmación en una comunicación. Cuando la comunicación utiliza un canal de elevada capacidad o gran latencia (es decir elevado valor de RTT) es posible que el emisor tenga que esperar a recibir confirmación antes de seguir enviando; por ejemplo en una comunicación vía satélite es normal tener valores de RTT de 500 mseg; si un host transmite datos a otro mediante TCP por un enlace vía satélite de 2 Mb/s no podrá enviar más de 64 Kbytes cada 500 mseg, ya que una vez ha llenado la ventana tiene que esperar a recibir el ACK; pero 64 Kbytes cada 500 mseg equivale a 64 * 1024 * 8 / 0,5 = 524.288 / 0,5 = 1,049 Mb/s, por lo que el enlace solo podrá aprovecharse al 50% aproximadamente. En general el rendimiento de una conexión siempre vendrá limitado por la fórmula:

Capacidad máxima = Tamaño de ventana / RTT

Es decir, siempre que el producto capacidad*RTT de una conexión sea superior al tamaño de ventana el rendimiento vendrá limitado por ésta ya que la conexión no es capaz de 'llenar la tubería' de datos y se producirán tiempos de espera en la comunicación. Cuando se diseñó TCP era impensable tener este tipo de problemas, que aparecieron inicialmente con la disponibilidad de enlaces vía satélite de alta velocidad (2 Mb/s). Hoy en día la posibilidad de tener enlaces de alta velocidad en redes de área extensa plantea otras situaciones en las que también se da este problema, por ejemplo una comunicación TCP sobre una línea ATM OC3 de 155,52 Mb/s con un RTT de 8 ms puede obtener con la ventana estándar un ancho de banda máximo de 524.288 / 0,008 = 65,5 Mb/s; para aprovechar completamente la línea sería precisa una ventana mínima de 155.520.000 * 0,008 = 1.244.160 bits = 152 Kbytes[5]. El RFC 1323, incorporado en muchas implementaciones de TCP, resuelve este problema ya que permite utilizar un factor de escala para ampliar el tamaño de ventana hasta 2^{30} (1 GByte).

Otro campo en el que TCP ha sido mejorado es el tipo de actuación en caso de errores. Las primeras implementaciones utilizaban retroceso n, con lo que cuando se perdían segmentos el rendimiento caía de forma apreciable. El RFC 1106, incorporado en muchas implementaciones, prevé el funcionamiento con repetición selectiva, de manera que el emisor sólo tiene que reenviar el segmento o segmentos que han llegado erróneos.

[5] En realidad la capacidad efectiva en IP de un enlace OC-3 después de restar el overhead de SDH y ATM es de unos 135 Mb/s, por lo que sería suficiente con una ventana de 132 Kbytes.

RFC 1106 también propone un mecanismo de envío de acuses de recibo negativos (NAK) cada vez que se recibe un segmento sin haber recibido el anterior. Sin embargo el RFC 1106 también toma en cuenta que al ser transportados en datagramas los segmentos pueden no llegar en orden, por lo que la no recepción de un segmento en el momento esperado no quiere decir necesariamente que éste se haya perdido; lo que hace el emisor en estos casos es esperar a que el receptor insista, por ejemplo si le comunica que ha recibido los tres segmentos siguientes a uno dado y sigue faltando éste es bastante probable que se haya perdido, con lo que lo reenvía sin esperar a agotar el timeout. Evidentemente este mecanismo solo se aplicará si la retransmisión no ha sido provocada antes por agotamiento del timer de retransmisión.

5.3.14 UDP (User Datagram Protocol)

TCP tiene la robustez y funcionalidades propias de un protocolo de transporte orientado a conexión; sin embargo esa robustez y funcionalidad conllevan una cierta complejidad; por ejemplo cualquier transmisión de información TCP requiere como mínimo el intercambio de seis mensajes para establecer la comunicación y terminarla; además mientras una conexión existe ocupa una serie de recursos en el host. A veces no se requiere toda esa funcionalidad; en esos casos se prefiere que el nivel de transporte preste un servicio más sencillo, no orientado a conexión y no fiable (por no fiable queremos decir que el receptor no acusa recibo de las TPDUs enviadas). Algunos ejemplos de situaciones en las que es más conveniente un servicio no orientado a conexión son las siguientes:

- El tipo de aplicación no requiere una fiabilidad total y no puede tolerar el retardo producido por los ACKs y las retransmisiones de TCP; este es el caso por ejemplo en la transmisión de vídeo o audio en tiempo real.

- La aplicación por su propia naturaleza requiere el envío de uno o dos mensajes únicamente; ejemplos de estas aplicacioens son las siguientes: sincronización de relojes (NTP), consultas al servidor de nombres (DNS), mensajes de gestión de la red (SNMP), etc.; en el DNS no se considera necesario un transporte fiable porque si se pierde el datagrama la aplicación lo reenviará; en el caso de NTP o SNMP la pérdida de un datagrama no es importante porque la información se está actualizando a intervalos regulares (por ejemplo cada 5 minutos).

- Se desea hacer envíos multidestino (multicast o broadcast); esto sólo es posible con un protocolo no orientado a conexión, ya que por su propia naturaleza los protocolos orientados a conexión son punto a punto (en TCP no es posible establecer conexiones multipunto).

El protocolo no orientado a conexión de Internet se conoce como UDP (User Datagrama Protocol); su nombre ya da una idea de la naturaleza no fiable del servicio de transporte ofrecido. Entre las aplicaciones que utilizan UDP se encuentran TFTP (Trivial File Transfer Protocol), DNS (Domain Name Server), SNMP (Simple Network Management Protocol), NTP (Network Time Protocol) y NFS (Network File System)[6], etc.

Las TPDUs intercambiadas por UDP se denominan *mensajes* o *datagramas UDP*. Recordemos que los mensajes UDP se identifican por el valor 17 en el campo protocolo del datagrama IP.

Una característica interesante de UDP es que puede ser utilizado por aplicaciones que necesitan soporte de tráfico multicast o broadcast. Con TCP esto no es posible debido a la naturaleza punto a punto, orientada a conexión del protocolo.

UDP no suministra ningún mecanismo de control de flujo o control de congestión. Cuando lo que se envía es únicamente un mensaje (por ejemplo una consulta al DNS) esto es innecesario, ya que

[6] La arquitectura de NFS presupone la no existencia de una conexión entre el servidor y el cliente, razón por la cual se adapta mejor a funcionar sobre UDP. En principio NFS fue una aplicación diseñada para su uso en redes locales, donde el retardo suele ser bajo y constante y la pérdida de datagramas muy rara. En este entorno UDP da normalmente un rendimiento aceptable. Sin embargo cuando se utiliza NFS en redes de área extensa el rendimiento puede no ser satisfactorio debido a la mayor fluctuación del retardo y la pérdida de datagramas debido a congestión. Para resolver estos problemas algunos fabricantes han desarrollado implementaciones de NFS que utilizan TCP como protocolo de transporte. Aunque supone una mejora interesante esto no está ampliamente disponible.

presumiblemente un mensaje aislado no creará problemas de congestión y será siempre aceptado en destino (y de no ser así el mismo problema habría surgido con TCP para el inicio de la conexión). Si se va a enviar un flujo de mensajes, por ejemplo vídeo o audio en tiempo real, se deberán tomar las medidas adecuadas para asegurar la capacidad suficiente en la red (mediante mecanismos de reserva de capacidad, por ejemplo) y evitar la congestión no excediendo lo solicitado en el momento de hacer la reserva; afortunadamente en estos casos se suele conocer a priori con bastante aproximación el tráfico que se va a introducir en la red.

En caso de congestión en la red parte de los datagramas serán descartados por la red sin informar por ningún mecanismo al emisor, ni al receptor. En caso de saturación del receptor este sencillamente ignorará los datagramas que no pueda aceptar. En algunos se contemplan a nivel de aplicación mecanismos de control que permiten al receptor detectar si se producen pérdidas (por ejemplo numerando los datagramas) informando al emisor para que baje el ritmo de emisión si se rebasa un umbral determinado.

De forma similar a los segmentos TCP, los mensajes UDP se dirigen a la aplicación adecuada mediante el puerto de destino, especificado en la cabecera. Análogamente a TCP los puertos UDP se identifican mediante un campo de 16 bits (números entre 0 y 65535). Aun en el caso de coincidir en número con un puerto TCP son TSAPs diferentes, como ya hemos comentado antes. Al igual que en TCP los valores por debajo de 1024 están reservados para los puertos denominados *'bien conocidos' (well-known ports)*, aunque su significado es diferente en la mayoría de los casos, pues también lo son los servicios. La tabla 6.8 muestra algunos de los puertos UDP más utilizados.

Servicio	**Puerto**	**Descripción**
Echo	7	Devuelve el datagrama al emisor
Discard	9	Descarta el datagrama
Daytime	13	Devuelve la hora del día
Quote	17	Devuelve una 'frase del día'
Chargen	19	Generador de caracteres
Nameserver	53	Servidor de nombres de dominios
Bootps	67	Puerto servidor utilizado para cargar información de configuración Bootp
Bootpc	68	Puerto cliente utilizado para recibir información de configuración
TFTP	69	Trivial File Transfer Protocol
SunRPC	111	Sun Remote Procedure Call
NTP	123	Network Time Protocol
SNMP	161	Usado para recibir consultas de gestión de la red
SNMP-trap	162	Usado para recibir avisos de problemas en la red

Tabla 6.8.- Algunos de los puertos UDP más utilizados

La estructura de un mensaje UDP se muestra en la tabla 6.9.

Campo	**Longitud (bits)**
Puerto origen	16
Puerto destino	16
Longitud	16
Checksum	16
Datos	0-524056 (65507 bytes)

Tabla 6.9.- Estructura de un mensaje UDP

A contnuación describimos el significado de cada uno de los campos de la cabecera UDP:

- *Puerto origen* especifica el puerto de la aplicación que genera el mensaje. Este valdrá normalmente cero, salvo que la aplicación solicite una respuesta.

- *Puerto destino* especifica el puerto de la aplicación a la que va dirigido el mensaje.

- *Longitud* indica la longitud del mensaje, incluyendo los campos de cabecera.

- *Checksum*: el uso de este campo es opcional en IPv4, obligatorio en IPv6 (ya que en ese caso se ha suprimido el chechsum a nivel de red). Cuando se envía información en tiempo real (audio o vídeo digitalizado) su uso puede omitirse. Para el cálculo se aplica el mismo algoritmo que en TCP (suma complemento a 1 de todo el mensaje dividido en campos de 16 bits, y complemento a 1 del resultado). En el cálculo se utiliza todo el mensaje, incluida la cabecera y se antepone una pseudocabecera similar a la utilizada en TCP (con la dirección IP de origen, de destino, el tipo de protocolo de transporte y la longitud del mensaje) de forma que se verifica que sean correctos no solo los datos del mensaje UDP sino también los datos fundamentales de la cabecera IP. Si la verificación del checksum en el receptor da error el mensaje es simplemente descartado sin notificarlo al nivel de aplicación ni al emisor.

- *Datos* contiene los datos a transmitir. Un mensaje UDP ha de estar contenido necesariamente en un datagrama IP, lo cual fija la longitud máxima de este campo.

De la misma forma que un host o un router pueden tener que fragmentar un datagrama que contenga un segmento TCP, es posible que el host emisor o algún router intermedio tengan que fragmentar un mensaje UDP porque sea mayor que la MTU permitida en la red por la que ha de enviarse. Análogamente a los segmentos TCP la fragmentación ocurre de froma transparente a UDP y la cabecera del mensaje solo aparecerá en el primer fragmento; en cambio cada framento deberá incluir una nueva cabecera IP.

El funcionamiento del protocolo UDP está descrito en el RFC 768.

5.4 EJERCICIOS

1. Indique si es verdadera o falsa cada una de las siguientes afirmaciones:

a) La fragmentación en Internet es siempre competencia del nivel de red, es decir del protocolo IP.

b) Siempre que se establece una comunicación TCP entre dos hosts en una red local se ejecuta previamente el protocolo ARP.

c) El nivel de transporte siempre ha de asegurar un envío fiable de los datos, ya que en algunos protocolos (p. ej. IP) el nivel de red no lo garantiza.

d) En TCP no es posible enviar un segmento sin haber recibido el ACK del anterior.

e) El bit PSH (datos Pushed) puede utilizarse en TCP para forzar el envío de datos en segmentos independientes.

f) En TCP el checksum se comprueba siempre; en caso de discrepancia entre el valor calculado y el recibido el segmento es descartado.

g) En UDP el uso del checksum es opcional; esto permite un rápido proceso de la información en transmisiones en tiempo real.

h) El paso de IPv4 a IPv6 no requiere ningún cambio en TCP o UDP, ya que en la información de control (cabecera) de estos protocolos no aparecen direcciones IP.

i) En un mismo host un número de puerto no puede ser utilizado simultáneamente por TCP y UDP

2. Describa brevemente en que consiste la técnica conocida como 'slow-start'

3. Explique la diferencia entre control de flujo y control de congestión.

4. El tamaño máximo de un segmento TCP es 65.515 bytes. Podría explicar de donde viene este valor?

5. Tanto el datagrama IP como el segmento TCP tienen un campo denominado *checksum*, que permite detectar errores de transmisión. Dado que TCP realiza la comprobación del checksum, ¿no resulta redundante que el nivel de red haga otra comprobación? Además, existe una diferencia importante en la manera como TCP e IP calculan el checksum, ¿sabe cual es?

6. La fragmentación y reensamblado de datagramas es competencia de IP y se realiza de forma transparente a TCP. ¿Significa esto que TCP no tiene que preocuparse de que los datos le puedan llegar con el orden alterado?

7. Una empresa dispone de un router con una interfaz ethernet y una línea serie, para conexión a la Internet, y desea configurar en él un filtro para que actúe de cortafuego. Se ha decidido que los empleados de la empresa podrán establecer conexiones TCP con el exterior, pero no se aceptarán conexiones TCP que se intenten establecer desde fuera (por ejemplo se podrá hacer telnet a máquinas de fuera de la empresa pero no se aceptará conexiones telnet desde fuera). De que forma se podría efectuar este filtrado?. El router tiene acceso a toda la información de cabecera del nivel de red y nivel de transporte, pero no a la parte de datos del nivel de transporte. Cuando recibe un datagrama el router sabe por que interfaz le ha llegado.

8. Una aplicación genera un mensaje de 1540 bytes que entrega a TCP para su envío, el cual lo incluye en un segmento y lo pasa a IP para que lo envíe a su destino; no se utilizan campos opcionales ni en la cabecera IP ni en la cabecera TCP. La red inicialmente tiene un MTU de 4000 bytes, pero en algún punto del camino el datagrama ha de atravesar una red cuyo MTU es de 800 bytes. Indique cuantos datagramas y cuantos bytes recibe el nivel de red en el host de destino.

9. Suponga que abre una sesión TCP entre dos hosts para transferir datos, y se reserva para dicha conexión una capacidad de 100 Mb/s (por ejemplo mediante RSVP). El tiempo de ida y vuelta es de 20 ms. ¿Sería posible ocupar en su totalidad el ancho de banda reservado? En caso negativo calcule cual sería el ancho de banda máximo que podría aprovecharse. Se supone que no se utilizan los campos opcionales de TCP y por tanto el tamaño máximo de ventana es el habitual.

10. Suponga que utiliza slow-start en una línea con un tiempo de ida y vuelta de 10 milisegundos. La ventana receptora es de 24 KBytes y el tamaño máximo de segmento es de 2 KBytes. ¿Cuanto tiempo pasará antes de poder enviar la primera ventana completa? Suponga que no hay congestión.

11. En una conexión TCP el valor promedio del tiempo de ida y vuelta MRTT es en un instante dado es de 30 mseg y los siguientes ACK llegan después de 26, 32 y 24 mseg respectivamente. ¿Cual es la nueva estimación de MRTT? Utilice $\alpha = 0{,}9$.

12. Suponga que realiza una conexión TCP a través de un medio físico poco fiable, es decir con un BER elevado. Como utiliza PPP a nivel de enlace el receptor comprueba el CRC y si la trama es erónea la descarta sin pedir reenvío al emisor. Como consecuencia de esto una de cada 10 tramas se pierde, con lo que el TCP emisor tiene que retransmitirla. No se produce fragmentación, por lo que cada segmento TCP se envía en una y solo una trama PPP.

a) Indique como evolucionará la ventana de congestión en el TCP emisor si en la conexión se envían 50 segmentos de 1024 bytes cada uno. Suponga que TCP utiliza retransmisión selectiva y que el receptor no impone limitación por control de flujo al emisor.

b) Indique cualitativamente que merma de rendimiento cabrá esperar en la conexión como consecuencia de la elevada tasa de error:

    A: Menor del 10%
    B: Alrededor del 10%
    C: Mayor del 10%.

c) Diga como influiría el valor del RTT (Round Trip Time) en su respuesta a la pregunta anterior.

Suponga que las tramas perdidas son justamente las número 10, 20, etc.

5.5 SOLUCIONES

S1.-

a) **Verdadero**.

b) **Falso**. Podría ocurrir que la dirección requerida se encuentre ya en la ARP cache del host (por ejemplo porque ya haya establecidas otras conexiones TCP entre esos mismos hosts, o porque haga poco tiempo que terminaron la anterior conexión), en cuyo caso no es necesario resolverla de nuevo.

c) **Falso**. En ocasiones es suficiente un servicio menos 'fiable', como UDP.

d) **Falso**. TCP utiliza un protocolo de ventana deslizante que permite enviar varios segmentos sin esperar confirmación.

e) **Falso**. El bit PSH no garantiza el envío en segmentos independientes; por ejemplo si se utiliza el algoritmo de Nagle puede suceder que varios grupos de datos 'pushed' de la aplicación se envíen en un mismo segmento.

f) **Verdadero**.

g) **Verdadero**. Su uso es opcional en el emisor, pero no en el receptor. Si el emisor decide utilizar checksum el receptor deberá comprobar siempre su valor y descartar en caso necesario.

h) **Falsa**. El cálculo del checksum en TCP y UDP incluye la pseudocabecera que esta formada entre otras cosas por las direcciones IP de origen y destino. El cálculo del checksum tiene por tanto que modificarse al pasar de IPv4 a IPv6.

i) **Falsa**. Puede utilizarse sin ambigüedad, ya que ambos se diferenciarían por el campo protocolo del datagrama IP.

S2.-

Es una técnica de ventana deslizante de tamaño variable que permite a algunos protocolos de transporte (TCP por ejemplo) adaptarse por tanteos sucesivos al estado de la red, respondiendo a las situaciones de congestión - detectadas por la pérdida de un segmento - con una reducción drástica en el tamaño de ventana. En principio la ventana tiene de tamaño un segmento y se va duplicando en cada envío hasta llegar a un valor límite (normalmente 64 KBytes) o hasta que se produce la primera pérdida, momento en el que se establece un 'umbral de peligro' igual a la mitad del tamaño de la ventana en ese instante. A partir de ahí la ventana vuelve a crecer de forma exponencial como antes hasta llegar al umbral de peligro, a partir del cual solo crece de forma lineal, es decir en un segmento cada vez. Este proceso se repite indefinidamente mientras dure la conexión TCP.

S3.-

El control de flujo se establece en una conexión extremo a extremo entre dos entidades (a nivel de enlace o a nivel de transporte) para evitar que un emisor activo sature a un receptor lento o sobrecargado. El control de congestión se establece en una red para evitar que se produzca saturación en alguna línea, lo cual provocaría la pérdida de paquetes y por consiguiente una merma en el rendimiento de toda la red.

S4.-

La cabecera de un datagrama IP tiene un campo de 16 bits que especifica la longitud total del datagrama, por lo que ésta puede ser como máximo de 65535 bytes. De estos al menos 20 corresponden a la cabecera IP, por lo que la longitud máxima de un segmento TCP puede ser como máximo 65515 bytes. La máxima cantidad de datos que puede transportar un segmento TCP es de 65495 bytes (65515 menos 20 de la cabecera TCP sin opciones).

Advertencia: En el Tanenbaum (pag. 526) hay un cálculo equivocado de la cantidad de datos que como máximo puede contener un segmento TCP. Dicha cantidad no es 65515 sino 65495 bytes por lo que se ha explicado.

S5.-

El nivel IP realiza la comprobación en cada salto, mientras que TCP la realiza únicamente en el host de destino. El checksum de IP sólo comprueba la información de la cabecera IP, no la cabecera TCP ni la parte de datos. El checksum de TCP comprueba tanto su cabecera como los datos, y añade una pseudocabecera que le permite verificar los datos fundamentales de la cabecera IP (dirección origen, dirección destino, protocolo y longitud del segmento).

Debido a que se ha de calcular múltiples veces el checksum de IP se intenta que sea sencillo y rápido de calcular, mientras que el de TCP puede ser exhaustivo ya que solo se comprueba una vez. La comprobación de IP permite detectar errores en la información de control que podrían encaminar un datagrama al destino equivocado, pero nada mas. Por otro lado la inclusión de la pseudocabecera en el checksum de TCP le permite detectar si el datagrama ha sido entregado al destino incorrecto, aun en el caso de que falle la comprobación de IP.

S6.-

En el caso de que un segmento se fragmente TCP no tendrá que preocuparse del orden de los fragmentos, ya que el nivel IP del receptor le presentará el segmento tal como estaba antes de la fragmentación. Sin embargo si datagramas que corresponden a segmentos diferentes llegan desordenados IP no los ordena, ya que no hay campo en la cabecera IP que especifique el orden de emisión. En tal caso será el TCP del receptor el que, ayudado de los números de secuencia, se ocupe de reordenar los diferentes segmentos en su buffer antes de pasar la información a la aplicación correspondiente.

S7.-

El router debería filtrar, es decir descartar, todos los datagramas que le lleguen por la interfaz serie y que cumplan simultáneamente las dos condiciones siguientes:

- Tener el valor 6 en el campo protocolo de la cabecera IP (esto indica que el datagrama contiene un segmento TCP).
- Tener a 1 el bit SYN y a 0 el bit ACK de los flags de la cabecera TCP. Estas dos características solo se dan en un segmento que intente iniciar una conexión TCP desde fuera.

Normalmente al descartar el datagrama el router debería enviar un mensaje ICMP DESTINATION UNREACHABLE al emisor, explicando en el código del mensaje el motivo por el cual ha sido rechazado.

S8.-

El segmento creado por TCP tendrá una longitud de 1560 bytes; ésta será la información que se fragmente a nivel IP. Como el MTU es de 800 bytes en cada datagrama se ocuparán 20 bytes con la cabecera IP y quedarán 780 para transportar el segmento; pero como los fragmentos han de tener una longitud múltiplo de 8 en realidad solo se pueden aprovechar como mucho 776 bytes de cada datagrama; necesitamos por tanto enviar tres datagramas que tienen la siguiente estructura:

Cab. IP (20 Bytes)	Cab. TCP (20 Bytes)	Datos (756 Bytes)

Cab. IP (20 Bytes)	Datos (776 Bytes)

Cab. IP (20 Bytes)	Datos (8 Bytes)

En total se envían tres datagramas y 1620 bytes. Obsérvese que el proceso de fragmentación no altera ni el número de segmentos enviados ni la cantidad de bytes transmitidos a nivel TCP.

S9.-

La máxima cantidad de datos que una sesión TCP puede enviar sin esperar acuse de recibo es igual al tamaño máximo de ventana, 64 KBytes – 1 o sea 65535 bytes, que equivalen a 524.280 bits. Vamos a suponer en primer lugar que los TCPs hubieran negociado un tamaño máximo de segmento igual al tamaño máximo de ventana. Si el envío se hace en segmentos de 65535 bytes el TCP emisor funcionando a 100 Mb/s necesitará 5,24 ms para introducir el segmento en la línea de transmisión y luego tendrá que esperar 20 ms mas hasta recibir el ACK correspondiente (suponiendo que el tiempo empleado en generar el ACK por parte del TCP receptor sea despreciable). Una vez recibido el ACK se enviará otro segmento durante 5,24 ms y se esperará 20 ms para recibir la respuesta. Se irán pues alternando períodos de 5,24 ms de actividad y 20 ms de espera, con lo que el funcionamiento será similar a un protocolo de parada y espera; y la tasa de ocupación de la línea será de 5,24/(5,24+20) = 0,208 = 20,8 % = 20,8 Mb/s.

Si en vez de utilizar segmentos de 64 KBytes empleamos segmentos por ejemplo de 1 KByte los ACKs empezaran a producirse tan pronto llegue la primera trama, sin necesidad de esperar a que el receptor reciba los 64 Kbytes de información. Con esto solapamos el tiempo de generación (que es ahora de 0,08 ms) y el tiempo de ida y vuelta (20 ms), dando una eficiencia mas alta; veamos en detalle lo que ocurre:

0 ms	El TCP emisor empieza a enviar el primer segmento.
0,08 ms	El TCP emisor empieza a enviar el segundo segmento.
0,16 ms	El TCP emisor empieza a enviar el tercer segmento.
…	
5,16 ms	El TCP emisor empieza a enviar el segmento número 64
5,24 ms	El TCP emisor termina de emitir el segmento número 64 y queda a la espera.
10 ms	El TCP receptor empieza a recibir el primer bit del primer segmento.
10,08 ms	Llega al receptor el último bit del primer segmento; se devuelve un segmento vacío con el ACK correspondiente.
20,08 ms	Llega al TCP emisor el primer ACK; empieza a enviarse el primer bit del segmento número 65.

A partir de aquí el ciclo se repite. Conseguimos pues tener la línea ocupada 5,24 ms de cada 20,08 ms, con lo que el rendimiento será de 5,24/20,08 = 0,261 = 26,1 % = 26,1 Mb/s. La mejora en la eficiencia se debe en este caso al uso de ventana deslizante.

En el caso límite, con segmentos muy pequeños, se conseguiría un total solapamiento entre los tiempos de transmisión (5,24 ms) y de propagación (20 ms): 5,24/20 = 0,262 = 26,2 % = 26,2 Mb/s.

Por tanto **no podría utilizarse todo el ancho de banda reservado, sino solamente 26,2 Mb/s como máximo.**

Una forma mas rápida de calcular la respuesta anterior sería haber utilizado la fórmula:

Tamaño de ventana = capacidad * RTT

Sabiendo que RTT = 20 ms y que el tamaño de ventana es 64 Kbytes, despejamos la capacidad:

Capacidad = Tamaño de ventana / RTT = 64 * 1024 * 8 / 0.02 = 26,2 Mb/s

Para un total aprovechamiento de la línea habría que utilizar la opción de TCP que permite ampliar el tamaño de la ventana, ya que así podríamos 'llenar' de datos la línea de transmisión y suprimir los tiempos de espera; la ventana mínima que nos permitiría hacer esto sería:

Tamaño ventana = $100 * 10^6 * 20 * 10^{-3} = 2 * 10^6$ bits

$2 * 10^6$ bits / 8 = 250000 bytes

250000 bytes / 1024 = 244,1 KBytes

S10.-

El crecimiento de la ventana se producirá de la siguiente forma:

Ciclo	Número de segmentos	Cantidad de KBytes	Tiempo transcurrido (ms)
Primero	1	2	0
Segundo	2	4	10
Tercero	4	8	20
Cuarto	8	16	30
Quinto	12	24	40

Por tanto la respuesta es: 40 milisegundos.

S11.-

Los valores de MRTT calculados son sucesivamente **29,6, 29,84 y 29,256 milisegundos**.

S12.-

a) Como funcionamos con repetición selectiva cuando se pierde una trama solo se reenvía el segmento correspondiente a dicha trama.

Aquí es importante distinguir las tramas de los segmentos. Para ello usaremos la notación *Tm(Sn)* donde *m* indica el número de trama y *n* el número de segmento. A efectos de los errores producidos consideraremos que las tramas nunca se repiten, lo que se reenvía en caso necesario son los segmentos.

La siguiente tabla muestra la evolución de la ventana de congestión; en negritas aparecen los envíos perdidos. Como era previsible se envían 55 tramas y 50 segmentos.

Ventana de congestión (KBytes)	Tramas(Segment.) enviados	Umbral de peligro (KBytes)
1	T1(S1)	64
2	T2(S2) T3(S3)	"
4	T4(S4) T5(S5) T6(S6) T7(S7)	"
8	T8(S8) T9(S9) **T10(S10)** T11(S11) T12(S12) T13(S13) T14(S14) T15(S15)	"
1	T16(S10)	4
2	T17(S16) T18(S17)	"
4	T19(S18) **T20(S19)** T21(S20) T22(S21)	"
1	T23(S19)	2
2	T24(S22) T25(S23)	"
3	T26(S24) T27(S25) T28(S26)	"
4	T29(S27) **T30(S28)** T31(S29) T32(S30)	"
1	T33(S28)	2
2	T34(S31) T35(S32)	"
3	T36(S33) T37(S34) T38(S35)	"
4	T39(S36) **T40(S37)** T41(S38) T42(S39)	"
1	T43(S37)	2
2	T44(S40) T45(S41)	"
3	T46(S42) T47(S43) T48(S44)	"
4	T49(S45) **T50(S46)** T51(S47) T52(S48)	"
1	T53(S46)	2
2	T54(S49) T55(S50)	"

b) En principio atendiendo solo a la cantidad de datos retransmitidos cabría esperar una merma del rendimiento de un 10% aproximadamente (se envían 55 tramas para 50 tramas útiles). Sin embargo, el hecho de que cada trama perdida provoque la reducción a 1 de la ventana de congestión y el reinicio del mecanismo de 'slow-start' produce una merma que será con toda seguridad superior al 10%.

c) El valor de RTT influye de forma decisiva en lo dicho anteriormente. En el caso límite en que el RTT fuera nulo la reducción sería exactamente del 10%, ya que no habría penalización por el hecho de tener que reinciar el 'slow-start' cada vez (con RTT nulo da lo mismo tener una ventana grande que pequeña, ya que no hemos de esperar para recibir el ACK).

En el caso real en que el RTT no sea cero la merma de rendimiento será tanto mayor cuanto mayor sea el RTT de la conexión TCP.

6 REDES FRAME RELAY Y ATM

Autor: Rogelio Montañana

6.1 FRAME RELAY

6.1.1 Formato del campo dirección en Frame Relay

Como ya sabemos Frame Relay es una red de conmutación de paquetes orientada a conexión; por tanto para que la comunicación sea posible es necesario que antes se establezca un circuito virtual entre dos hosts de la red. Los circuitos virtuales pueden ser permanentes o conmutados, y se identifican mediante los denominados DLCI (Data Link Connection Identifier) que desempeñan en Frame Relay un papel similar a los VPI/VCI (Virtual Path Identifier/Virtual Channel Identifier) en ATM o a los VCN (Virtual Circuit Number) en X.25. Los DLCI tienen por defecto 10 bits de longitud, aunque se han definido extensiones que permiten utilizar DLCIs de 16, 17 o 23 bits.

Recordemos cual es la estructura de una trama Frame Relay:

Campo	Tamaño (bytes)	Valor
Delimitador	1	01111110
Dirección	2	Variable
Datos	0-8188	Variable
Checksum	2	Variable
Delimitador	1	01111110

Tabla 4.1.- Estructura de una trama Frame Relay

La esencia del funcionamiento de Frame Relay se encuentra en el campo Dirección, que tiene la siguiente estructura:

DLCI superior						C/R	0	DLCI inferior				FECN	BECN	DE	1

Tabla 4.2.- Estructura del campo dirección en una trama LAPF normal

El significado de los diversos campos es el siguiente:

- **DLCI (Data Link Connection Identifier)**: este campo (que se encuentra dividido en dos partes) tiene una longitud total de 10 bits. Especifica por que circuito virtual debe circular la trama correspondiente.

- **C/R (Command/Response)**: el significado de este bit es específico de la aplicación y no se utiliza en el protocolo Frame Relay estándar.

- **FECN (Forward Explicit Congestion Notification):** como su nombre indica este campo de un bit se emplea en el control de congestión del que hablaremos mas tarde.

- **BECN (Backward Explicit Congestion Notification):** lo veremos en el control de congestión.

- **DE (Discard Elegibility):** este bit tiene una finalidad similar al CLP (Cell Loss Priority) de ATM, es decir, sirve para marcar las tramas de 'segunda clase', que son aquellas que el usuario ha metido en la red superando el caudal que tenía contratado.

El estándar prevé también el uso de campos dirección de 3 y 4 bytes con tamaños del DLCI de 16, 17 ó 23 bits; estos formatos no los describiremos ya que el DLCI de 10 bits y la dirección de 2 bytes son los

valores por defecto y los formatos extendidos no añaden nada nuevo sobre el modo de funcionamiento de Frame Relay.

Para establecer un SVC en Frame Relay se utiliza el protocolo de señalización Q.933 de la ITU-T, que es un subconjunto del Q.931 utilizado en RDSI. De forma análoga a RDSI o ATM, Frame Relay utiliza señalización *fuera de banda* para establecer la conexión, es decir, se reserva un DLCI especial (el DLCI 0) para las funciones relativas al establecimiento y terminación del SVC. La siguiente tabla compara las funciones de señalización principales de Frame Relay, ATM y RDSI:

	Frame Relay	**ATM**	**RDSI**
Protocolo ITU-T de señalización	Q.933	Q.2931	Q.931
Canal de señalización	DLCI 0	VP 0 VC 5	Canal D

Tabla 4.3.- Protocolo y canal de señalización utilizado en RDSI, Frame Relay y ATM

Para conectar un host a una red Frame Relay se establece un enlace mediante una línea punto a punto entre el host y el conmutador Frame Relay al que se conecta. Una vez hecha la conexión física el host establece un circuito virtual por el DLCI 0 con el conmutador Frame Relay y si la red soporta el protocolo de señalización Q.933 se establecerán circuitos virtuales conmutados (SVCs) de manera dinámica. Si la red no soporta Q.933 (que es el caso de la mayoría de las redes Frame Relay públicas) solo se podrán establecer circuitos virtuales permanentes (PVCs). La elección de la ruta se realiza en el momento de establecer el circuito; si se trata de un SVC la elección la realiza el protocolo de señalización Q.933, si es un PVC la realiza el operador al configurar el PVC en los equipos.

Para que en una red Frame Relay pueda funcionar el protocolo de señalización y los SVCs es imprescindible utilizar un sistema de direccionamiento. Esto es menos importante en el caso de una red que no soporte SVCs, ya que entonces solo el operador tiene que identificar los nodos de la red. Dado que las redes Frame Relay no soportan normalmente SVC no veremos en este apartado el formato de direcciones Frame Relay ni el protocolo de señalización Q.933.

Un PVC o SVC se constituye por una entrada en la tabla de circuitos de cada conmutador Frame Relay por el que pasa. En cada conmutador el circuito es una correspondencia entre el puerto de entrada y su DLCI con el puerto de salida y su DLCI (que normalmente será diferente). Las tramas Frame Relay que viajan por ese circuito están continuamente cambiando el valor de su campo DLCI.

Frame Relay no soporta emisiones multicast. Evidentemente siempre es posible realizar una emisión de este tipo enviando paquetes de forma exhaustiva uno a uno a todos los destinos deseados, pero esta emisión ‘desde fuera’ no optimiza la distribución de los paquetes de acuerdo con la topología de la red.

6.1.2 Control de tráfico en Frame Relay

El control de tráfico en Frame Relay se basa en la especificación de varios parámetros, el más importante de los cuales es el denominado CIR (Committed Information Rate). En el caso de circuitos permanentes el CIR se especifica en el momento de configurar los equipos; en el de circuitos conmutados es solicitado por el usuario en el momento de efectuar la llamada; en este último caso la red puede tener que rechazar la llamada si no dispone de la capacidad solicitada.

El control de tráfico en Frame Relay se realiza de la siguiente forma. El conmutador Frame Relay al que esta conectado el equipo del usuario realiza una monitorización permanente del tráfico que el usuario inyecta en la red por el circuito virtual. Si el usuario no supera en ningún momento el CIR sus tramas viajarán todas con el bit DE (Discard Elegibility) a cero; sin embargo, si el usuario excede dicha capacidad el conmutador Frame Relay pondrá a 1 el bit DE en aquellas tramas que se encuentren (en todo o en parte) por encima de la capacidad especificada en el CIR. Un segundo parámetro, conocido como EIR (Excess Information Rate), especifica un caudal adicional que el usuario no deberá superar nunca, ya que las tramas recibidas por encima de este valor serán directamente descartadas por el conmutador.

La implementación práctica del algoritmo que acabamos de describir utiliza en realidad otros dos parámetros:

- B_c: Tamaño de ráfaga comprometida (Committed burst size). Indica la cantidad máxima de bits que la red se compromete a enviar, en condiciones normales, durante un intervalo de tiempo T. Estos datos pueden estar o no contiguos, es decir pueden formar parte de una o de varias tramas.

- B_e: Tamaño de ráfaga excedente (Excess burst size). Indica la máxima cantidad de bits que, además de B_c, podrá el usuario intentar enviar por la red, durante un intervalo de tiempo T. No hay compromiso en la transferencia de estos datos, o dicho con más precisión, hay una menor probabilidad de que estos datos lleguen a su destino que los que son enviados dentro de B_c.

Entre los parámetros B_c y CIR se cumple la relación:

$$B_c = CIR * T$$

Análogamente entre B_e y el EIR se cumple la relación:

$$B_e = EIR * T$$

Los parámetros CIR y B_c configuran un pozal agujereado donde ρ = CIR y C = B_c; mientras que EIR y B_e configuran un segundo pozal agujereado con ρ = EIR y C = B_e. El tráfico enviado a la red por el primer pozal tiene el bit DE a 0. El segundo pozal actúa como desbordamiento del primero y marca el tráfico que envía a la red con el bit DE a 1. El tráfico excedente de este segundo pozal es descartado.

Para comprender como funciona el control de tráfico en Frame Relay supongamos que un usuario contrata con Telefónica un acceso Frame Relay con una línea física E1, es decir con una capacidad máxima entre su ordenador y el conmutador Frame Relay de 2.048 Kb/s. El usuario contrata además un PVC con un CIR de 1.024 Kb/s; Telefónica configura el enlace con un EIR de 384 Kb/s y establece el valor de T en 1 segundo (con lo que automáticamente han quedado fijados los valores de B_c y B_e en 1.024.000 y 384.000 bits). Obsérvese que aunque se han definido varios parámetros el único especificado en el contrato del usuario con Telefónica (y el único de cuyo valor el usuario tiene conocimiento oficial) es el CIR.

En esta situación nuestro usuario desea enviar un flujo de vídeo en tiempo real a un destino remoto, sin ningún tipo de control de flujo por parte del receptor y sin atender a ninguna notificación de congestión que pueda venir de la red. Supongamos que el usuario dispone de un parámetro en su ordenador mediante el cual puede fijar el caudal de tráfico que inyecta en la red. Supongamos también que el envío se hace utilizando siempre tramas de 50.000 bits (6.250 bytes). Si el usuario fija el flujo de datos a transmitir en 2.000 Kb/s estará inyectando en el conmutador Frame Relay 40 tramas por segundo; en estas condiciones las primeras veinte tramas serán aceptadas sin más, las ocho siguientes serán aceptadas pero se les pondrá a uno el bit DE ya que superan el valor de B_c, y las doce restantes serán simplemente descartadas puesto que superan el valor de B_e.

Si el usuario reduce ahora el caudal a 1.400 Kb/s enviará 28 tramas por segundo, 20 de las cuales tendrán el bit DE a cero y las ocho siguientes a uno; de esta forma el usuario está aprovechando casi al máximo la capacidad de la red, pero no tiene la seguridad de que todas las tramas lleguen a su destino.

Por último, si el usuario quiere tener máximas garantías de que todas las tramas llegarán a su destino deberá reducir el flujo a un valor no superior al CIR, por ejemplo a 1.000 Kb/s, en cuyo caso emitirá 20 tramas por segundo y todas serán enviadas con el bit DE a cero.

Conviene destacar el hecho de que, independientemente del flujo que el usuario especifique en su aplicación, el enlace físico es en todos los casos de 2.048 Kb/s con lo que una trama de 50.000 bits siempre se transmitirá en 24,4 ms; así en el caso de transmitir un flujo de 2.000 Kb/s el emisor está 24,4 ms enviado y 0,6 ms esperando; en caso de transmitir a 1.000 Kb/s el emisor está 24,4 ms enviando y 25,6 ms esperando.

El bit DE también puede ser puesto de forma voluntaria por el usuario. Esto es interesante si el usuario (o la aplicación) puede identificar algunas tramas como más importantes que otras. Por ejemplo en vídeo comprimido MPEG existen tres tipos de fotogramas, los intra, los predictivos y los bidireccionales. Si se pierde un fotograma intra la calidad se ve mucho más afectada que si se trata de un predictivo o bidireccional. En nuestro ejemplo, en el caso de transmitir a 1.400 Kb/s (28 tramas por segundo) la aplicación podría elegir en cada grupo de 28 tramas ocho que correspondan a fotogramas predictivos o bidireccionales y marcarlas con DE a 1, con lo que evitará que el conmutador Frame Relay asigne indiscriminadamente el bit DE a las últimas ocho tramas de cada intervalo, que podrían contener algún fotograma intra.

Analicemos ahora con más detalle el ejemplo anterior aplicando el algoritmo del pozal agujereado. Supongamos el caso concreto en que nuestro usuario deseara enviar el flujo de vídeo a 2.000 Kb/s, pero únicamente desea hacerlo durante un segundo; antes habíamos llegado a la conclusión de que de las 40 tramas transmitidas cada segundo 20 eran enviadas normalmente a la red, 8 llevaban el bit DE puesto y 12 eran descartadas. En realidad esa conclusión solo es correcta en el caso de que se envíe el flujo de 2.000 Kb/s de manera sostenida. Analicemos que ocurriría si se envía ese caudal durante únicamente un segundo; pasados 500 milisegundos el usuario ha introducido en el pozal C las 20 tramas que caben en él, por lo que la siguiente trama ya debería pasar al pozal E y consecuentemente llevar puesto el bit DE. Sin embargo debemos tomar en cuenta que en 500 milisegundos el pozal C ya ha enviado a la red una cantidad de tráfico igual a CIR * 0,5 = 512.000 bits, o sea 10 tramas; cabrán pues 10 tramas más en el pozal C (en realidad serán 9 ya que la primera trama no puede ser enviada hasta que ha sido recibida en su totalidad para poder comprobar el CRC, y esto introduce un retraso de 24,4 milisegundos en el envío de tramas); pasados 750 milisegundos han entrado en el pozal 30 tramas y han salido 14 (la primera sale pasados 24,4 ms), con lo que el pozal aún no está lleno. Para este caso podemos calcular el número de tramas enviadas por el pozal C en un tiempo t dado con la fórmula:

$$\text{Tramas enviadas} = (t-0{,}0244) * 1.024.000 / 50.000 = (t - 0{,}0244) * 20{,}48$$

Por otro lado, las tramas recibidas en el pozal en un tiempo t serán:

$$\text{Tramas recibidas} = t * 2.000.000 / 50.000 = t * 40$$

Suponiendo que antes estuviera vacío el pozal C se llenará cuando:

$$\text{Tramas recibidas} = \text{Tramas enviadas} + B_c/50.000$$

O sea:

$$t * 40 = (t-0{,}0244) * 20{,}48 + 20{,}48$$

De aquí podemos calcular que t = 1,02 segundos, es decir, que con los parámetros que hemos puesto en este ejemplo se podría transmitir durante 1,02 segundos sin desbordar el pozal.

En realidad el cálculo de la fórmula anterior no es correcto y solo sirve como una primera aproximación, ya que hemos realizado las operaciones aritméticas con decimales sin efectuar los truncamientos apropiados. Veamos que ocurre con las fórmulas anteriores para t = 1s:

Tramas recibidas = 40

Tramas enviadas = 19,98 = **19 tramas**

Capacidad pozal = 20,48 = **20 tramas**

Tomando en cuenta los truncamientos resulta que al cabo de un segundo se ha descartado una trama.

6.1.3 Control de congestión en Frame Relay

El propio control de tráfico de Frame Relay ya es un primer mecanismo de control de la congestión, ya que suaviza las ráfagas que podrían introducir los usuarios. A pesar de eso la congestión es aún posible por lo que Frame Relay incorpora diversos mecanismos para el control de la congestión.

Cuando se produce congestión en una red Frame Relay se aplica la técnica de paquetes de asfixia que vimos en el apartado 2.3.5; de esta forma se advierte a los hosts emisores de la situación para que adopten medidas correctoras. En caso de que dichos avisos no sean atendidos, o no lleguen con la suficiente rapidez, la red empezará a descartar tramas, primero las que tengan a 1 el bit DE, y si esto no es suficiente también las demás.

Para detectar cuando hay peligro de congestión los conmutadores Frame Relay monitorizan constantemente el tamaño de cada una de sus colas; cuando algún valor es superior al valor considerado el umbral de peligro el conmutador correspondiente deberá identificar la conexión o conexiones causantes del problema, y enviar avisos a los hosts respectivos. Cualquier conmutador de la red puede tomar la iniciativa de enviar avisos de congestión a los hosts que considere responsables de la situación.

La detección y eventual resolución de las situaciones de congestión en Frame Relay afecta a circuitos virtuales, no a interfaces. Un mismo host que tenga establecidos diferentes circuitos virtuales por una misma línea de acceso Frame Relay podría percibir una congestión severa en uno de los circuitos y tener poco cargados los demás.

El aviso de congestión normalmente viaja 'piggybacked' en una trama de datos, en los bits denominados FECN y BECN del campo dirección (de forma parecida a lo que se hace en ATM con el bit central del campo PTI). El significado de los bits FECN y BECN es el siguiente:

- BECN (Backward Explicit Congestion Notification): este bit se pone a uno en las tramas que van dirigidas al host causante de la congestión (por tanto se ha de poner en las tramas ‘de regreso’). El host que recibe el mensaje debe entender que la congestión se está produciendo por el tráfico que él está introduciendo por el circuito virtual por el cual recibe el aviso y que por tanto debe iniciar los procedimientos previstos para reducir el caudal en ese circuito virtual. La denominación 'backward' indica que la congestión se está produciendo en sentido contrario al sentido en que viaja el aviso.

- FECN (Forward Explicit Congestion Notification): se pone a uno para indicar a un host que existe congestión en la red, y que el problema está producido por el tráfico que él está recibiendo por el circuito virtual por el que ha recibido el aviso; por tanto el host deberá emplear los mecanismos a su alcance para conseguir que su interlocutor introduzca un caudal de tráfico menor en ese circuito virtual. La denominación 'forward' indica que la congestión en este caso se produce en el mismo sentido en que viaja el aviso.

Supongamos que dos hosts, A y B, establecen un circuito a través de una red Frame Relay, y que se produce una situación de congestión en la red, de forma que afecta a la comunicación en el sentido A→B, pero no en el sentido contrario. En este caso las tramas que A recibirá de la red llevarán puesto a 1 el bit BECN, y las que reciba B llevarán a 1 el bit FECN.

Cabría pensar que la utilización del bit FECN es innecesaria, ya que el bit BECN permite informar con mayor rapidez al host emisor. Sin embargo en algunos casos el emisor ignora las indicaciones que recibe de la red y la única posibilidad de influir en su comportamiento es a través del receptor.

En Frame Relay se prevé también la posibilidad de que un conmutador envíe una trama de control especial a un host en situaciones de congestión, cuando no haya tráfico de retorno en ese DLCI que permita enviar el aviso en el bit BECN.

Aunque las posibilidades de control de congestión en Frame Relay son amplias gracias a los mecanismos descritos, la realidad es que muchas implementaciones no los utilizan. Frame Relay no define las acciones a desarrollar en caso de congestión. Se supone que los protocolos de nivel superior adoptarán

las medidas que consideren mas oportunas. La realidad es que en muchos casos estos avisos son ignorados.

6.2 LA CAPA DE RED EN ATM

> *En sus orígenes, Internet y la red telefónica fueron guiadas por un número reducido de individuos competentes que decidieron estándares esenciales por consenso y en base al mérito técnico. A medida que estas redes crecieron, y conforme mas gente se implicaba en ellas, el proceso de estandarización necesariamente se ralentizó, pero las decisiones (técnicamente acertadas) hechas en los primeros días aun dirigen la evolución de la red. Por el contrario, los estándares ATM están siendo definidos en el fórum ATM por un grupo grande de empresas con intereses mutuamente incompatibles, y no todas con experiencia en construir y operar redes ATM. Al intentar estandarizar antes de que los investigadores hayan llegado al consenso, los estándares del fórum ATM a veces reflejan estimaciones obsoletas de capacidades técnicas, tales como cuanto buffering y proceso es posible en los conmutadores, y están abiertas a compromisos puramente políticos. La experiencia con OSI, que intentó estandarizar la conmutación de paquetes antes de que la tecnología estuviera madura, sugiere que este proceso dará lugar a redes innecesariamente caras. Esto bien puede retrasar la eventual adopción de tecnología ATM. El reto para las redes ATM es dejar suficiente libertad en los estándares para permitir que la tecnología innovadora se consolide.*
>
> *S. Keshav, 'An Engineering Approach to Computer Networking'*

ATM es una red orientada a conexión (CONS), y como tal crea circuitos virtuales (denominados canales virtuales o *Virtual Channel* en el estándar) entre los sistemas que desean intercambiar información. En ATM los nodos terminales se denominan *hosts* y los de encaminamiento *conmutadores* (análogamente a X.25 o frame relay); los conmutadores ATM son siempre equipos de comunicaciones altamente especializados y de elevadas prestaciones, nunca ordenadores de propósito general. Una red (o subred) ATM está formada por una serie de conmutadores unidos entre sí por líneas punto a punto de alta velocidad, normalmente SONET/SDH de 155,52 Mb/s en adelante (OC-3c), aunque también existen interfaces de velocidades inferiores; la interfaz que conecta los hosts con la subred (es decir con los conmutadores) se denomina UNI (User-Network Interface), y la que comunica los conmutadores entre sí es la NNI (Network-Network Interface).

A diferencia de X.25 ATM no envía acuses de recibo de las celdas recibidas; ni siquiera se verifica el contenido de la celda, salvo la cabecera, ya que se diseñó pensando en medios de transmisión altamente fiables, como las fibras ópticas. Además, uno de los objetivos de ATM es enviar tráfico isócrono, para el cual las retransmisiones serían peores que dejar pasar algunos errores.

La única cosa que ATM garantiza totalmente es que las celdas que se envían por un circuito virtual serán entregadas en su destino en el mismo orden en que han salido. Se podrá perder alguna (pocas) pero bajo ningún concepto se permite alterar el orden de llegada, ni duplicar celdas.

En ATM se pueden agrupar los canales virtuales entre dos nodos en lo que se conoce como trayectos virtuales o *Virtual Paths* (VPs). Podemos pensar en los VCs como pares de hilos de cobre que enlazan dos nodos y que están numerados en ambos extremos para su fácil identificación; en tal caso los VPs serían como mangueras que agrupan gran cantidad de estos pares y que también se numeran para su fácil identificación. Para poder establecer una comunicación entre dos nodos es preciso localizar un par libre en ambos extremos, para lo cual es necesario especificar el número de VP y de VC. El uso de VPs tiene algunas ventajas cuando se trata de reencaminar todos los circuitos virtuales que hay entre dos nodos determinados, ya que si se quiere reencaminar todos los VCs de un VP se puede rencaminar el bloque sin necesidad de renumerar los VCs uno a uno.

6.2.1 Formato de la cabecera de las celdas

La celda ATM está formada por 5 bytes de cabecera y 48 de carga útil (payload). Existen dos formatos de cabecera según se trate del interfaz UNI o NNI, como se muestra en la tabla 4.4. Veamos el significado de estos campos en detalle.

Campo	Longitud (bits)	
	Formato UNI	**Formato NNI**
GFC (General Flow Control)	4	0
VPI (Virtual Path Identifier)	8	12
VCI (Virtual Channel Identification)	16	16
PTI (Payload Type)	3	3
CLP (Cell Loss Priority)	1	1
HEC (Header Error Control)	8	8

Tabla 4.4.- Formato de la cabecera de una celda ATM

El campo **GFC** sólo aparece en el formato UNI y no se utiliza.

VPI identifica el Virtual Path por el que debe circular la celda. Una interfaz UNI puede tener hasta 256 Virtual Paths, mientras que una NNI puede tener hasta 4096.

VCI identifica el Virtual Channel por el que debe circular el paquete dentro del Virtual Path especificado. Puede haber hasta 65536 VCs en cada VP.

El campo **PTI** contiene tres bits; la tabla 4.5 detalla el significado de cada uno de los ocho posibles valores del campo PTI. Como puede verse, el primer bit indica si se trata de una celda de usuario (0) o de mantenimiento de la red (1), el segundo se utiliza para informar de situaciones de congestión mediante un aviso 'piggybacked' y el tercero indica si la celda es de tipo 0 o de tipo 1; algunos protocolos de transporte de ATM (AAL5 por ejemplo) utilizan la distinción entre tipo 0 y tipo 1 para marcar la última celda de una secuencia, como veremos más adelante.

Valor PTI	Significado
000	Celda de usuario tipo 0; no se detecta congestión
001	Celda de usuario tipo 1; no se detecta congestión
010	Celda de usuario tipo 0; se detecta congestión
011	Celda de usuario tipo 1; se detecta congestión
100	Información de mantenimiento entre conmutadores vecinos
101	Información de mantenimiento entre conmutadores de origen y destino
110	Celda de gestión de recursos (utilizada para control de congestión ABR)
111	Reservado

Tabla 4.5.- Significado del campo PTI

El campo **CLP** sirve para distinguir entre celdas importantes y no importantes, de cara a un posible descarte por congestión. Las celdas con CLP 1 serán descartadas primero. Si por ejemplo la red permite al usuario enviar un caudal superior al acordado normalmente marcará el tráfico excedente con el bit CLP para indicar que se trata de tráfico de 'segunda' clase, que es el primer candidato a ser descartado por la red en caso de apuro. Obviamente las celdas que se envían para notificar una situación de congestión tiene este bit a 0 para evitar en lo posible que sean descartadas.

El campo **HEC** es, como ya vimos en el capítulo 3, un CRC de los primeros 32 bits que detecta todos los errores simples y la mayoría de los errores múltiples.

6.2.2 Puesta en marcha de una conexión ATM

En ATM existen tanto PVCs como SVCs. Los PVCs se configuran de manera estática en los conmutadores. Un PVC está establecido siempre que estén operativos los conmutadores por los que pasa y los enlaces que los unen, es decir siempre que la red está operativa. En cambio los SVCs se crean y destruyen dinámicamente, según se necesita. El protocolo utilizado para establecer SVCs en ATM se denomina Q.2931, y esta basado en el Q.931 utilizado en la señalización de RDSI. Q.2931 es bastante complejo, por lo que solo veremos algunos aspectos básicos de su funcionamiento.

Cuando un nodo desea establecer un VC con otro ha de enviar un mensaje solicitando la conexión por el VC reservado para señalización, que por convenio es el VP 0 VC 5; en cierto modo podemos considerar este VP/VC equivalente al canal D de RDSI.

El establecimiento y finalización de una conexión se lleva a cabo mediante un conjunto de mensajes que pasan del nodo de origen al nodo de destino, recibiendo confirmación en cada nodo intermedio. Para establecer el VC el nodo de origen enviará un mensaje SETUP con la dirección del nodo de destino; cada nodo intermedio reenviará el mensaje al siguiente nodo en la dirección adecuada y responderá con un mensaje CALL PROCEEDING al anterior; cuando el mensaje llegue finalmente al destinatario éste responderá con un mensaje CONNECT que será enviado al originador; como respuesta al CONNECT cada nodo responderá con un CONNECT ACK.

En las redes ATM está soportado el tráfico multicast. En una emisión multicast no se envían celdas duplicadas por un enlace físico, sino que se replican justo en el conmutador donde se produce la bifurcación de caminos físicos, con lo que se consigue una optimización del tráfico. Para esto se establecen circuitos multipunto. Una vez se ha establecido un VC entre dos nodos se puede 'invitar' a otros a participar mediante el comando ADD PARTY

6.2.3 Direccionamiento en ATM

Para que el protocolo de señalización Q.2931 y el establecimiento de circuitos virtuales conmutados en ATM funcionen correctamente es necesario disponer de un esquema de direcciones adecuado. Inicialmente la ITU-T acordó utilizar direcciones con el formato E.164, que son ni mas ni menos que las utilizadas por la red telefónica pública internacional; esto permite integrar las direcciones ATM con las de la RDSI de banda estrecha. Las direcciones E.164 estan formadas por hasta 15 dígitos decimales (aunque la ITU-T está considerando extenderlo a 16 ó 17) que se representan en código BCD (es decir dos dígitos por byte); la dirección se ajusta por la derecha y se rellena con ceros por la izquierda en caso necesario. Una dirección E.164 tiene los siguientes campos:

Relleno	Código de país	Código de área/proveedor	Número de suscriptor

- El **código de país** tiene de uno a tres dígitos y especifica el código correspondiente al país según el estándar E.163; por ejemplo 34 representa España, 1 Estados Unidos, 507 Panamá, etc.

- El **código de área/proveedor** sirve para identificar al operador dentro de un país y/o el código de área dentro del país; en el caso de España identifica a la provincia (96 para Valencia, 945 para Álava, etc.). Tiene una longitud variable.

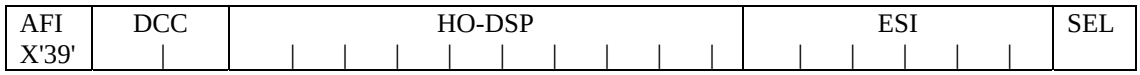
- El **número de suscriptor**: identifica al usuario del servicio. Este podría ser un host o un conmutador ATM del usuario (pero siempre conectado a la red mediante un interfaz UNI). Un ejemplo de número de suscriptor es cualquier número de teléfono analógico o RDSI.

Las direcciones E.164 tienen una estructura jerárquica en la que la longitud de cada campo no esta establecida de antemano, sino que depende del valor de la propia dirección. Así el código de país puede tener de uno a tres dígitos, el código de área en España puede tener dos o tres dígitos, etc.

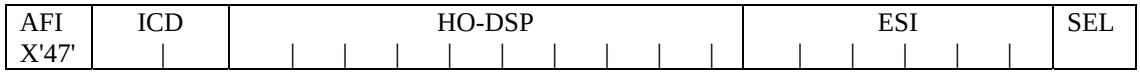
Prácticamente todas las redes públicas ATM (o sea las operadas por compañías telefónicas) utilizan direccionamiento E.164, ya que esto permite la integración total con RDSI y con la red telefónica

convencional. De cualquier forma el direccionamiento solo es importante cuando la red soporta SVCs, cosa que no ocurre por ejemplo con el servicio ATM de Telefónica que solo soporta PVCs.

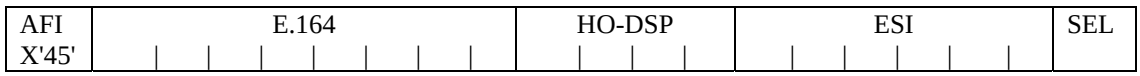
Dado que el espacio de direcciones E.164 es relativamente pequeño no es factible su utilización en redes ATM privadas, por lo que en este caso se suele utilizar el direccionamiento especificado por el ATM forum, que se basa en el OSI de 20 bytes denominado direccionamiento NSAP (Network Service Access Point); las direcciones NSAP tienen una longitud de 20 bytes. Las direcciones NSAP de ATM pueden tener cualquiera de los tres siguientes formatos:



Formato DCC (Data Country Code)



Formato ICD (International Code Designator)



Formato NSAP E.164

El primer byte contiene el AFI (Authority and Format Identifier) y su valor indica cual de los tres tipos de dirección se está utilizando.

El formato DCC (Data Country Code) se utiliza para hacer una distribución geográfica por países; en este caso los bytes 2 y 3 especifican el código del país, pero de una forma completamente distinta a como se hace en las direcciones E.164; dado que se trata de direcciones ISO aquí se emplean los códigos de país según el estándar ISO 3166, que son siempre de tres dígitos (por ejemplo a España le corresponde el código 724); el código se almacena en formato BCD como en E.164, pero en vez de ajustarlo por la derecha se ajusta por la izquierda y se rellena a unos por la derecha; así por ejemplo en el caso de España una dirección ATM tendría en el campo DCC el valor X'724F'.

Una vez asignado el código del país el miembro ISO de dicho país (AENOR en el caso de España) asigna los bytes 4 y 5, que son los dos primeros de la parte que genéricamente se denomina HO-DSP (High Order-Domain Specific Part); estos dos bytes especifican redes u organizaciones registradas dentro del país; por ejemplo en España AENOR ha asignado a RedIRIS el HO-DSP X'1001'. Los ocho bytes restantes del HO-DSP los distribuye la autoridad registrada según su propio criterio (RedIRIS en nuestro ejemplo). Normalmente esta parte de la dirección se estructura en varios campos con una estructura jerárquica descendente, para facilitar el routing jerárquico.

El campo ESI (End System Identifier) sirve, como su nombre indica, para identificar al equipo final con el que se quiere conectar; normalmente este campo contiene una dirección MAC IEEE de seis bytes que identifica al equipo, con lo que se asegura que dicha dirección es única a nivel global (mundial).

El campo SEL (Selector) permite distinguir diferentes entidades (por ejemplo protocolos) en el nivel de red en el mismo host. Cumple una función similar al campo protocolo en una trama PPP, o al DSAP y SSAP en un paquete LLC.

Veamos a modo de ejemplo como se estructuran las direcciones ATM de la red académica española, RedIRIS:

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|39|72 4F|10 01|rr|pp ii|ii|dd dd nn cc|ee ee ee ee ee ee|ss|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

El significado de cada uno de los campos es el siguiente:

- **X'39'**: indica que se utiliza formato DCC
- **X'724F'**: Código ISO de España (rellenado a unos por la derecha)
- **X'1001'**: Código asignado a RedIRIS por el miembro español de ISO, AENOR.
- **rr**: Indica la Comunidad Autónoma. X'30' representa la Comunidad Valenciana.
- **pp**: Indica la región (provincia, comarca, etc.) dentro de una Comunidad Autónoma. Si no se han asignado valores a este campo se utiliza el valor X'10'.
- **iiii**: Representa la Universidad u Organización. X'0001' representa la Universidad de Valencia.
- **dddd**: Area dentro de una organización. X'0012' es el Campus de Burjassot.
- **nn**: Red dentro de un área. X'11' red del Servicio de Informática.
- **cc**: Conmutador dentro de una red
- **eeee.eeee.eeee**: Equipo final.
- **ss**: selector dentro del equipo final.

Por ejemplo la dirección del conmutador ATM de la Universidad de Valencia en el Campus de Burjassot es X'39.724F.1001.3010.0001.0012.1101.0090.b1f4.2c01.00'

En realidad la estructura de la dirección a partir del campo iiii es solo orientativa ya que cada institución puede decidir organizarlo de manera diferente.

El formato ICD (International Code Designator) está pensado para organizaciones o redes transnacionales que no puedan utilizar el formato DCC (por ejemplo la ONU); en este caso la organización o red correspondiente ha de obtener un identificador (un ICD) de la ISO, para utilizarlo en sus direcciones; a partir de ahí la organización o red decide como estructura la parte HO-DSP de la dirección. Los campos ESI y SEL tienen el mismo significado que en el formato DCC. Por ejemplo la red académica NORDUnet, que se extiende por Suecia, Noruega, Finlandia y Dinamarca, utiliza direcciones ICD con el valor X'0023' en el campo ICD, y a partir de ahí define una estructura con múltiples niveles, similar a la de RedIRIS, para los 10 bytes siguientes. Los fabricantes de equipos ATM también suelen utilizar direcciones ICD; por ejemplo Cisco tiene registrado el ICD X'0091' y lo utiliza en las direcciones que asigna por defecto a todos sus equipos.

El tercer formato, NSAP E.164, contiene en los ocho bytes siguientes al AFI una dirección E.164 empaquetada en 8 bytes y ajustada por la derecha. Este formato permite transportar una dirección E.164 por una red con direccionamiento NSAP.

Los tres formatos de direccionamiento NSAP tienen en común dos aspectos importantes:

- Los 13 primeros bytes se utilizan para encaminamiento dentro de la red y los siete últimos para encaminamiento hacia el equipo final, también denominados a veces encaminamiento de nivel 2 y de nivel 1, respectivamente. Podemos considerar que los 13 primeros bytes constituyen la parte de red y los siete últimos la parte de host.

- La parte del ESI contiene una dirección MAC de seis bytes global y única que puede ser utilizada para la autoconfiguración de los equipos; en este caso el equipo pondría por su cuenta la parte de host (los siete últimos bytes) y obtendría del conmutador ATM al que se conecta la parte de red (los 13 primeros).

En ATM un conmutador posee una dirección única para todas sus interfaces; en esto es radicalmente diferente de un router IP que tenía una dirección IP diferente para cada interfaz. En cambio un host con dos interfaces ATM tendrá normalmente dos direcciones diferentes si estas interfaces se conectan a conmutadores diferentes.

6.2.4 Encaminamiento y conmutación de celdas

El uso de Virtual Paths (VP) para agrupar Virtual Channels (VC) permite facilitar la gestión de las conexiones ATM, por ejemplo:

- Una vez fijado un VP entre dos hosts la creación de nuevos VCs entre ellos no requiere decisiones de encaminamiento, y puede hacerse sin intervención de los conmutadores intermedios; los nuevos VCs irán sencillamente por dentro del mismo VP.

- Si todos los VCs entre dos destinos discurren por el mismo VP el encaminamiento de celdas en los conmutadores será mas sencillo pues solo será preciso consultar la tabla de VPs, no la de VPs+VCs que tiene muchas mas entradas.

- En caso de tener que modificar la ruta, por ejemplo por avería de una línea o un conmutador, en vez de tener que redirigir los VCs uno a uno se puede hacer por grupos de VCs.

En el libro de texto (Tanenbaum pag. 456-458, figuras 5-66 a 5-68) hay una clara explicación, que no repetiremos aquí, de como se lleva a cabo el establecimiento de VPs y el encaminamiento de celdas entre conmutadores ATM. Solo cabe destacar que los valores de VPI tienen significado local dentro de cada conmutador en una red ATM; por tanto en cada salto que realiza una celda dentro de la red el campo VPI es modificado y por consiguiente el campo HEC es recalculado cada vez.

Si en vez de realizar el encaminamiento por VPs se hiciera por VCs el mecanismo sería el mismo salvo por el hecho de que el tratamiento se haría a nivel del campo VC en cada celda. En realidad sería posible modificar en cada salto tanto el VP como el VC.

6.2.5 Routing en ATM

El funcionamiento orientado a conexión de ATM establece una diferencia importante en cuanto al routing dinámico respecto a IP. Mientras que en IP el protocolo de routing dinámico elige la ruta óptima en cada momento, en ATM la elección se realiza solo en el momento de establecer el circuito; si más tarde el trayecto elegido sufre congestión el circuito no se reconducirá por otros caminos que pueda haber mas descargados de tráfico, a no ser que el circuito se termine y se realice una nueva conexión.

En una red ATM mallada se dispone en general de varias rutas alternativas para el establecimiento de un VC. Para la elección de la ruta se pueden emplear tablas estáticas contenidas en los conmutadores (routing estático), o utilizar un protocolo de routing dinámico que permita a los conmutadores elegir el camino óptimo cada vez que se establece un circuito. Como en el caso de IP si queremos realmente aprovechar las posibilidades de resistencia a fallos que nos brinda una red mallada necesitamos implementar routing dinámico en la red; de lo contrario será necesaria una intervención manual para cambiar las rutas y eludir así la parte no operativa de la red. Además conviene recordar que esto solo tiene sentido si la red implementa un protocolo de señalización que haga posible el establecimiento de SVCs, pues si tenemos PVCs la intervención manual será necesaria para configurar el circuito por la nueva ruta.

El routing dinámico en ATM es más complejo que en IP, ya que además de los factores relativos a carga y estado de los enlaces se ha de tomar en consideración los requerimientos de calidad de servicio especificados en la solicitud de conexión, y la cantidad de recursos libres en cada uno de los enlaces de cada una de las rutas posibles. El protocolo de routing utilizado en ATM se denomina P-NNI (Private Network-to-Network Interface) y se basa en el algoritmo del estado del enlace, igual que OSPF e IS-IS. De la misma forma que éstos, P-NNI permite definir niveles jerárquicos (denominados ‘peer-groups’), cada uno de los cuales va asociado a una parte de la dirección ATM. En el caso de utilizar direcciones NSAP de 20 bytes el máximo de la dirección que puede asignarse a la parte red es de 13 bytes, que son 104 bits; por esta razón el número máximo de niveles jerárquicos que permite P-NNI es de 105, asociados a los 104 bits mas un nivel superior englobando todos los inferiores.

A diferencia de lo que ocurre en IP en ATM el conmutador que solicita establecer el circuito fija la ruta completa, no solo el siguiente salto. De esta forma si en algún tramo del trayecto no es posible cumplir los requisitos de calidad de servicio planteados se produce un mensaje de vuelta atrás denominado ‘CrankBack’ y el conmutador que inició la solicitud se replantea de nuevo la ruta óptima desde el principio, pudiendo entonces elegir una ruta completamente distinta a la inicialmente prevista. El proceso descrito se realiza independientemente para cada peer-group por parte del conmutador de entrada a dicho

peer-group. Es evidente que para poder realizar con máximas de probabilidades de éxito este proceso el conmutador ATM necesita disponer de una información lo más completa posible sobre la topología de la red, lo que explica porqué no hay en ATM protocolos de routing basados en el algoritmo del vector distancia.

6.2.6 Categorías de servicio

Para poder satisfacer un amplio espectro de necesidades se definen en ATM las denominadas *Categorías de Servicio*, que difieren en el nivel de garantía que dan al usuario respecto de la disponibilidad de los recursos de red solicitados. En un servicio ATM ofrecido por un operador el usuario contrataría una determinada categoría de servicio y en función de ello pactaría con el operador una serie de parámetros que especificarían la capacidad y la calidad de servicio que obtuviera. Normalmente una misma capacidad tendrá un precio diferente según la calidad de servicio solicitada. De esta forma se pretende que el usuario contrate en cada caso la categoría de servicio que mejor se acomode a sus necesidades, y a su presupuesto.

El servicio **CBR (Constant Bit Rate)** garantiza una capacidad determinada y constante, independientemente de la utilización que hagan de la red otros usuarios. Este servicio es el mas sencillo de implementar y el mas seguro de todos, ya que la red reserva la capacidad solicitada en todo el trayecto de forma estática. No se realiza ningún tipo de control de congestión, ya que se supone que ésta no puede ocurrir. El servicio CBR es equivalente a una línea dedicada punto a punto. Las principales diferencias (y ventajas) de CBR respecto a las líneas punto a punto son las siguientes:

- Las líneas punto a punto son siempre simétricas; un circuito CBR puede contratarse asimétrico y adaptarse así a las necesidades del usuario.

- Las líneas punto a punto solo están disponibles con determinadas capacidades preestablecidas, a veces con enormes separaciones entre sí (512 Kb/s, 2 Mb/s, 34 Mb/s); cuando se requieren capacidades intermedias es preciso recurrir a agregar varios enlaces de la capacidad inferior, lo cual es costoso y poco eficiente; un CBR se puede contratar con el caudal que se desee (normalmente redondeado a un valor entero en Kb/s).

- Una línea punto a punto tiene un proceso de instalación complejo; si se desea aumentar la capacidad es preciso instalar una nueva línea, con el consiguiente costo y tiempo de instalación. El caudal de un CBR se puede modificar en cuestión de minutos.

- Una línea punto a punto tiene reservada la capacidad de forma permanente; un CBR puede contratarse con un perfil horario determinado, de forma que tenga una mayor capacidad precisamente en las horas del día en que más se necesita, reduciéndolo en las que no se utiliza.

Debido a su sencillez CBR es el servicio mas extendido actualmente en las redes ATM públicas. El servicio ATM de Telefónica (Gigacom) ofrece actualmente VCs CBR con velocidades de entre 64 Kb/s y 155 Mb/s. Es posible contratar un perfil horario con hasta cuatro cambios de caudal cada 24 horas.

El servicio **VBR (Variable Bit Rate)** está pensado para cuando se prevé un tráfico a ráfagas. Tiene dos modalidades: RT-VBR, con requerimientos de bajo retardo y jitter para aplicaciones en tiempo real (vídeoconferencia, vídeo bajo demanda, etc.), y NRT-VBR para aplicaciones en las que el control del retardo no es tan importante, por ejemplo transferencia de ficheros. En VBR el usuario especifica un caudal medio pero, en función de sus necesidades y del estado de la red, podrá utilizar ocasionalmente caudales superiores, lo cual da mayor flexibilidad y permite ajustar el caudal a las necesidades medias. Para el control de las ráfagas se utiliza el algoritmo del pozal agujereado. En algunos servicios VBR el tráfico excedente sale marcado con el bit CLP. Desde el punto de vista de la red VBR tiene una complejidad superior a CBR.

El servicio **ABR (Available Bit Rate)** es de todas las categorías de servicio que ofrece ATM la que mas se parece a Frame Relay. ABR está pensado para tráfico a ráfagas, se supone que habrá instantes de gran demanda de capacidad seguidos de otros de total inactividad. ABR permite establecer un ancho de banda mínimo garantizado y fijar un valor máximo orientativo. ABR es la única categoría de servicio ATM en

la que se prevé que la red suministre control de flujo al emisor para que reduzca el ritmo en caso de congestión; esta circunstancia hace de ABR la categoría de servicio mas apropiada para tráfico de datos, por ejemplo para enviar datagramas IP, pero lo hace poco apropiado para aplicaciones isócronas. Debido a su funcionalidad ABR es la categoría de servicio mas compleja de implementar, por lo que no esta aún muy desarrollada en la práctica.

El servicio **UBR (Unspecified Bit Rate)** puede considerarse el de mas baja calidad. Es en cierto modo equivalente al servicio que ofrece Frame Relay cuando se utiliza un CIR 0. No existe ningún tipo de garantía en cuanto al retardo o ancho de banda, y tampoco se informa al emisor en caso de congestión. UBR utiliza la capacidad sobrante en la red de las demás categorías de servicio; por este motivo UBR será presumiblemente el servicio de menor coste cuando esté disponible comercialmente. Puede utilizarse para enviar tráfico IP cuando el costo sea el factor principal y la calidad de servicio no sea importante (por ejemplo para enviar tráfico de news).

Por último diremos que el ATM fórum ha definido una variante del servicio UBR denominada UBR+, que consiste en añadir al servicio UBR la posibilidad de especificar una capacidad mínima requerida. UBR+ sería similar al servicio ABR pero sin el control de congestión.

6.2.7 Calidad de servicio y descriptores de tráfico

La posibilidad de establecer Calidad de Servicio es una de las grandes virtudes de ATM. No es sorprendente pues que en las redes ATM se puedan fijar una larga serie de parámetros que definen los niveles mínimos de calidad que el operador debe ofrecer al usuario para cada una de las categorías de servicio antes definidas. Estos parámetros podemos clasificarlos en dos grupos, los parámetros de tráfico y los parámetros de Calidad de Servicio. No todos los parámetros tienen sentido en todas las categorías de tráfico.

Los parámetros de tráfico son los siguientes:

- **PCR (Peak Cell Rate) y CDVT (Cell Delay Variation Tolerance):** Máximo caudal que permite el VC y tolerancia (pequeña) respecto a este caudal

- **SCR (Sustainable cell rate) y BT (Burst Tolerance):** Caudal medio máximo permitido y tolerancia a ráfagas (grande) respecto a este caudal

- **MCR (Minimum Cell Rate, tasa de celdas mínima):** velocidad mínima que se considera aceptable para establecer el circuito ATM.

Los parámetros de Calidad de Servicio son los siguientes:

- **Max. CTD (Maximum Cell Transfer Delay, máximo retardo de transferencia de una celda):** es el retardo o latencia máximo permitido, o sea el tiempo máximo que puede tardar la red en transmitir una celda de un extremo a otro del circuito.

- **Peak-to-Peak CDV (Peak to Peak Cell Delay Variation, variación del retardo pico a pico):** es el 'jitter' o máxima fluctuación que se podrá producir en el retardo de las celdas.

- **CLR (Cell Loss Ratio, tasa de pérdida de celdas):** porcentaje máximo aceptable de celdas que la red puede descartar debido a congestión. Cuando una celda es entregada en el destino con un retardo superior a MCTD se considera celda perdida.

No todos los parámetros son aplicables a todas las categorías de servicio. Por ejemplo en un servicio CBR se especifica PCR, pero no SCR ni MCR puesto que todos tendrían el mismo valor. En un servicio UBR no se especifica ningún parámetro, ya que UBR ofrece un servicio ‘best effort’.

La siguiente tabla resume que parámetros se especifican normalmente en cada categoría de servicio:

	CBR	VBR-RT	VBR-NRT	ABR	UBR+	UBR
PCR/CDVT	SI	SI	SI	SI	NO	NO
SCR/BT	NO	SI	SI	NO	NO	NO
MCR	NO	NO	NO	SI	SI	NO
Max. CTD	SI	SI	NO	SI	NO	NO
Pk-t-**P**k CDV	SI	SI	NO	NO	NO	NO
CLR	SI	SI	SI	SI	NO	NO

Tabla 4.6.- Parámetros QoS aplicables a cada categoría de servicio ATM

6.2.8 Conformación y vigilancia del tráfico

En una red que pretende ofrecer calidad de servicio a sus usuarios es fundamental que el usuario no exceda los límites de caudal pactados en su contrato de servicio. Esta tarea tiene dos aspectos íntimamente relacionados: por una parte el autocontrol que el usuario ejerce sobre el tráfico que inyecta en la red y que denominamos conformado de tráfico o 'traffic shaping' y por otra la vigilancia que la red ha de ejercer sobre el tráfico de dicho usuario que denominamos vigilancia de tráfico o 'traffic policing' y que normalmente realiza el conmutador ATM que conecta a dicho usuario a la red.

Los mecanismos de control de tráfico dependen mucho de la categoría de tráfico de que estemos hablando. En el caso de tráfico CBR las labores de conformado y vigilancia son muy simples, ya que el usuario dispone de una capacidad constante asignada y reservada de forma estática. Supongamos por ejemplo que un usuario contrata un circuito CBR con un PCR de 10.000 celdas/s (equivalente a 4,24 Mb/s); en este caso el usuario estará autorizado a introducir una celda en la red cada 100 μs; si su conexión física con la red ATM es un enlace OC3 (155,52 Mb/s) el usuario tardará 2,7 μs en enviar una celda, por lo que después de cada envío deberá estar como mínimo 97,3 μs sin enviar otra, ya que de lo contrario excedería su PCR y el conmutador descartaría todas las celdas que llegaran antes del plazo previsto.

En el extremo opuesto se encuentra el tráfico UBR. Aquí no se pacta ningún parámetro por lo que el usuario recibe un servicio 'best effort'; normalmente esto se traduce en que los circuitos UBR utilizan la capacidad sobrante al resto de circuitos de la red, con lo que un usuario UBR podría llegar a ocupar totalmente la capacidad de un enlace en momentos en que no hubiera tráfico de otro tipo (a excepción del caudal que estuviera asignado a circuitos CBR), pero este tráfico de mínima prioridad será desplazado en cuanto otros usuarios tuvieran alguna necesidad. El servicio UBR es el que se utiliza normalmente en las redes locales ATM.

En tráfico ABR se fija una capacidad mínima requerida (MCR) y una máxima prevista (PCR); en el momento de establecer el circuito la red se asegura de que la capacidad correspondiente al MCR esté disponible, pero no promete nada respecto a PCR, que estará sujeto a disponibilidad. Dado que el usuario recibe realimentación de la red en caso de congestión, se supone que si inyecta un tráfico superior a MCR y la red no puede soportarlo recibirá mensajes de la red que le harán reducir el caudal. No se establece la duración de las ráfagas (parámetro MBS) ya que éstas pueden tener una duración considerable en ratos de baja carga en la red. De la misma forma que en UBR un usuario de servicio ABR podría emplear cantidades importantes de la red en momentos de baja utilización de otros usuarios, pero sin superar en ningún momento el caudal especificado en el parámetro PCR.

El control de tráfico VBR es el mas complejo. Los parámetros especificados, SCR, PCR y MBS, configuran un pozal agujereado En el caso mas sencillo utilizaríamos SCR como caudal ρ, MBS sería el tamaño del buffer C, y PCR el caudal con que el host envía los datos al pozal. El forum ATM prevé tres posibles algoritmos diferentes que pueden utilizarse para especificar el comportamiento del tráfico VBR frente a las ráfagas, basados en el uso de uno o dos pozales agujereados. Se prevé también la posibilidad de que el usuario o la aplicación controle el marcado del bit CLP en las celdas para evitar el descarte de tráfico importante. Aun cuando se permite la emisión de ráfagas la aplicación del algoritmo del pozal agujereado implica que en un servicio VBR el caudal medio nunca supera el valor de SCR pactado.

6.2.9 Control de congestión

6.2.9.1 Control de admisión y reserva de recursos

Dado que ATM ofrece un servicio CONS es posible comprobar en el momento de establecer el circuito si existen en la red recursos suficientes para satisfacer la solicitud, es decir ejercer control de admisión. En los servicios CBR, VBR, ABR y UBR+ la solicitud de establecer el VC viene acompañada de unos requerimientos medios y máximos de ancho de banda, con los que es posible realizar una comprobación e incluso reserva de recursos en todo el trayecto.

6.2.9.2 Control de congestión basado en el ritmo

Como ya hemos visto la única categoría de servicio que contempla mecanismos de control de congestión es ABR. El tráfico CBR nunca debería encontrar congestión pues funciona como una línea dedicada; el VBR aplica conformado y vigilancia de tráfico con lo que se supone que las ráfagas podrán ser amortiguadas por los bufers de los conmutadores, y el UBR se supone que se dará cuenta de la congestión por el descarte de celdas.

El mecanismo que se sigue para notificar de la presencia de congestión en tráfico ABR es el siguiente: el emisor de celdas genera cada *k* celdas de datos una celda especial llamada de gestión de recursos (celda RM, Resource Management); la celda RM llega al receptor el cual la examina, la modifica, y la devuelve al emisor. En caso de congestión el receptor puede informar al emisor en la celda RM devuelta. Si la congestión es tan severa que las celdas RM se pierden el emisor detectará que no está recibiendo las celdas RM devueltas al ritmo previsto y sospechará que hay congestión.

El control de congestión en tráfico ABR se lleva a cabo ajustando un parámetro denominado ACR (Actual Cell rate, flujo de celdas real) en un valor comprendido entre los parámetros MCR (Minimum Cell Rate) y PCR (Peak Cell Rate) pactados entre el usuario y la red en el momento de establecer el circuito. Las celdas RM tienen un campo denominado ER (Explicit Rate) donde se anotan el valor de Cell Rate que puede utilizarse. Una celda RM empieza su viaje de ida y vuelta con un valor de ER por ejemplo igual a PCR; cada conmutador por el que pasa la celda RM revisa el valor de ER que ésta lleva anotado y si le parece aceptable lo respeta, pero si el conmutador tiene congestión y le parece excesivo lo reduce (nunca lo puede aumentar); la revisión del ER se lleva a cabo tanto en la ida como en la vuelta. Al final el emisor cuando recibe la celda RM devuelta puede saber por el valor de ER que ACR debe utilizar.

Además en situaciones de congestión severa un conmutador puede generar celdas RM propias y enviarlas al emisor. Como tercera vía para notificar la congestión los conmutadores ATM pueden poner a 1 el bit medio del campo PTI (Payload Type) de la cabecera en las celdas de datos que envían al causante de los problemas, que actúa de forma parecida al bit BECN de Frame Relay..

6.2.10 Los Protocolos de Transporte de ATM

Dentro del modelo ATM la capa que se ocupa de la comunicación host-host, y que por tanto podemos considerar de transporte, es la denominada capa de adaptación ATM, o capa AAL (ATM Adaptation Layer).

Existen importantes diferencias entre los protocolos de transporte habituales, como TCP, y los utilizados en ATM. Esto se debe a que las redes ATM no fueron diseñadas en principio para el transporte de datos informáticos, sino para la transmisión de datos en tiempo real, principalmente voz y vídeo digitalizado.

La ITU-T define la capa AAL en la recomendación I.363. Esta recomendación ha sido fruto de diversos compromisos y reajustes sobre la marcha, y para comprender su estado actual tenemos que referirnos aunque sea brevemente a su evolución histórica.

Dado que el objetivo de la capa AAL es adaptar diversos tipos de tráfico para su transporte sobre redes ATM, la ITU-T empezó estudiando y clasificando las clases de tráfico que podían tener cierto interés. Desde el punto de vista de la ITU-T (principalmente formada por compañías telefónicas) los parámetros relevantes para esa clasificación eran tres:

- Tiempo real o no tiempo real (tráfico isócrono o asíncrono)
- Caudal de tráfico constante o variable
- Servicio orientado a conexión o no orientado a conexión

Con tres parámetros y dos posibles valores para cada uno se obtienen en total ocho combinaciones. De éstas la ITU-T decidió que sólo cuatro tenían sentido; por ejemplo se consideró que un tráfico en tiempo real con caudal constante no orientado a conexión no tenía sentido o era algo inútil, por lo que se descartó esa combinación. Las cuatro clases que se consideraron interesantes se denominaron A, B, C y D; en la tabla 8.7 se recogen las características relevantes de cada una de ellas.

Clase	Tiempo real	Caudal de tráfico	CONS-CLNS
A	Si	Constante	CONS
B	Si	Variable	CONS
C	No	Variable	CONS
D	No	Variable	CLNS

Tabla 8.7.- Clases de tráfico ATM definidas por la ITU-T

Aunque esta clasificación hoy en día está obsoleta tiene cierto interés porque fue la base para definir los protocolos de adaptación de ATM (AAL). Inicialmente se definieron cuatro protocolos AAL, denominados AAL1 a AAL4, que correspondían a las cuatro clases descritas. A medida que se iban especificando los detalles de cada AAL se observó que los requerimientos de las clases C y D eran tan similares que no se justificaba la existencia de dos protocolos diferentes, por lo que AAL3 y AAL4 fueron agrupados en un protocolo conjunto, denominado por ello AAL3/4. La gama de protocolos AAL quedaba así reducida a tres posibilidades: AAL1 para servicios CONS en tiempo real con tasa constante, AAL2 para servicios CONS en tiempo real con tasa variable y AAL3/4 para tráfico no en tiempo real con tasa variable; este último sería el protocolo normalmente elegido cuando se tratara de transmitir datos sin el requerimiento de tiempo real, por ejemplo para la interconexión de redes locales.

El grupo de trabajo que se ocupaba en la ITU-T de especificar los protocolos AAL estaba formado pricipalmente por representantes de las operadoras, que eran (y son) los representantes 'genuinos' de la ITU-T. Por aquel entonces recibía una cierta atención una tecnología de redes MAN denominada DQDB (Distributed Queue Dual Bus) estandarizada por el IEEE bajo la denominación 802.6; este estándar se perfilaba como una via por la cual las operadoras podrían ofrecer servicios de transporte de datos a los usuarios. Cabe pensar que redes DQDB distantes se unieran entre sí utilizando redes ATM, y en particular el protocolo AAL3/4. No resulta pues extraño que AAL3/4 se diseñara pensando en disfrutar una buena compatibilidad con DQDB, de forma que fuera fácil el intercambio de datos entre ambas tecnologías. Sin embargo a cambio de esta virtud, de cuestionable utilidad por cuanto que DQDB es una tecnología hoy olvidada, el protocolo AAL3/4 es poco eficiente e introduce un overhead excesivo tanto en la información de control que requiere como en la cantidad de proceso necesario para generarla e interpretarla. Las empresas fabricantes de equipos informáticos (conmutadores y adaptadores ATM), que se incorporaron tarde al proceso de estandarización de los protoclos AAL, se dieron cuenta de esto cuando ya no era posible modificar el AAL3/4, por lo que decidieron crear un nuevo protocolo que denominaron AAL5, para transportar la misma clase de datos que AAL3/4 pero de forma más eficiente.

Dos hosts acuerdan el protocolo AAL a utilizar cuando establecen una conexión o VC (Virtual Channel); a partir de ese momento el protocolo permanece inalterado durante toda la conexión. Un VC no puede transportar simultáneamente tráfico utilizando diferentes protocolos AAL; sin embargo un mismo enlace físico si puede ser utilizado por diferentes AALs si cada uno utiliza un VC diferente.

Conviene diferenciar las *clases de tráfico* a las que nos estamos refiriendo de las *categorías de servicio* que hemos visto en el nivel de red. La categoría de servicio es algo que se pacta con el prestador del

servicio (por ejemplo con el operador a la hora de solicitar un VC en una red ATM pública), mientras que el protocolo AAL es algo que solo afecta a los dos hosts comunicantes de forma transparente a la red y a todos los conmutadores ATM por los que pase ese VC. Mientras que la categoría de servicio se pacta especificando una serie de parámetros (de tráfico y de Calidad de Servicio) no existe ningún parámetro que intercambiar entre dos hosts cuando eligen utilizar uno u otro protocolo AAL.

No obstante lo anterior, es evidente que los protocolos AAL que transmiten tráfico en tiempo real necesitarán de la red una categoría de servicio que garantice estos parámetros. Por tanto normalmente una comunicación entre dos hosts mediante el protocolo AAL1 requerirá de la red un servicio CBR para funcionar correctamente, y una mediante AAL2 requerirá un servicio VBR-rt. Los protocolos AAL3/4 y AAL5 pueden utilizar cualquiera de las categorías de servicio, incluídas las CBR y VBR-rt cuando las características del tráfico a transmitir lo justifiquen.

Al margen de lo que sea mas adecuado, la elección de categoría de servicio estará supeditada a lo que esté soportado por los hosts y la red por la que se comunican; la elección del protocolo AAL solo estará supeditada a las posibilidades de los hosts ya que para la red no participa en el protocolo AAL. Al ser este un campo muy cambiante en cuanto a la especificación de los estándares existen muchas diferencias de funcionalidad entre los productos disponibles en el mercado y la interoperabilidad entre diversos fabricantes no está ni mucho menos asegurada, aun cuando la documentación indique una estricta adherencia a estándares.

6.2.10.1 Estructura de la Capa de Adaptación ATM

La capa AAL se compone de dos subcapas. La inferior denominada subcapa de *segmentación y reensamblado* o **SAR** (Segmentation And Reassembly) se ocupa, como su nombre indica, de crear en el emisor las celdas a partir de los datos recibidos de la subcapa superior, y de reconstruir en el receptor los datos originales a partir de las celdas recibidas. La superior se denomina subcapa de *convergencia* o **CS** (Convergence Sublayer) y actúa de interfaz entre la capa AAL y la aplicación, permitiendo adaptar diversos tipos de tráfico para su transporte sobre redes ATM.

La subcapa SAR construye las celdas con el formato propio del protocolo AAL utilizado, que puede incluir determinados campos de control. En recepción se encarga de la verificación e interpretación de esos campos de control, y de la reconstrucción de los datos para su envío a la subcapa CS. Los cinco primeros bytes de cabecera de la celda ATM contienen la información del nivel ATM, por lo que para la subcapa SAR la celda ATM sólo tiene 48 bytes, que corresponden a la parte denominada carga útil o 'payload' de la celda. En esos 48 bytes debe estar incluida cualquier información de control que requiera el protocolo AAL utilizado.

La subcapa superior, llamada de convergencia o CS (Convergence Sublayer) se subdivide a su vez en dos partes, una inferior y otra superior. La inferior depende del protocolo AAL utilizado pero no de la aplicación, por lo que se denomina *parte común* de la subcapa de convergencia o **CPCS** (Common Part Convergence Sublayer), y una superior que es específica de la aplicación por lo que se la llama subcapa de convergencia *específica del servicio* o **SSCS** (Service Specific Convergence Sublayer); en algunos casos esta última puede no existir. Por su naturaleza dependiente de la aplicación no veremos aquí la subcapa específica (SSCS), y siempre que nos refiramos a la subcapa CS estaremos haciéndolo implícitamente a la parte común (CPCS).

La subcapa CS se ocupa de recibir los mensajes llegados de la aplicación (o de la parte específica si existe) y pasarlos a la subcapa SAR para su proceso. En algunos protocolos AAL la subcapa CS añade una información de control a los datos recibidos, creando así una estructura que denominaremos mensaje; el mensaje es enviado (haciendo uso de los servicios de la subcapa SAR) a la subcapa CS en el otro extremo, la cual realiza el proceso inverso interpretando y verificando la información de control recibida para pasar el mensaje a la aplicación, o a la parte específica si la hay. En algunos protocolos AAL (concretamente AAL 1 y AAL 2) las funciones de la subcapa CS son mínimas, ya que no se construye mensaje alguno; el flujo de datos es formateado en celdas directamente por la subcapa SAR.

Todos los protocolos AAL calculan CRCs. En el caso de AAL 1 el CRC sólo afecta a la cabecera de la parte de carga útil de la celda. En AAL 2, AAL 3/4 y AAL 5 se calcula un CRC de los datos enviados,

que se comprueba en recepción, pero en caso de error el receptor se limita a descartar los datos erróneos, o a pasarlos e informar de ello a la aplicación. No existe en los protocolos AAL ningún mecanismo de notificación al emisor para el reenvío de datos erróneos; si se desea éste debe ser implementado por los niveles superiores.

El AAL 3/4 ha caído en desuso al ser reemplazado por el AAL 5. El AAL 2 se especificó en su momento de manera incompleta y ahora se está redefiniendo. Por tanto nosotros describiremos solamente el AAL 1 y el AAL 5, que son los únicos que se utilizan en la práctica.

6.2.10.2 AAL 1

Este protocolo está pensado para transmitir tráfico en tiempo real con caudal constante. Por tanto se corresponde normalmente con la categoría de servicio CBR. AAL 1 sirve por ejemplo para transportar circuitos de voz digitalizada (PCM) cuando se conectan centralitas mediante una red ATM mediante lo que se conoce como servicio de emulación de circuitos o CES (Circuit Emulation Services). Es bastante habitual emular circuitos E1 (2,048 Mb/s) que permiten interconectar centrales telefónicas soportando hasta 30 conversaciones simultáneas (audio no comprimido). Se puede utilizar este protocolo con equipos de compresión de audio siempre y cuando el algoritmo de compresión utilizado genere una tasa de bits constante. También hay algunos sistemas de videoconferencia especialmente diseñados para funcionar con AAL1 sobre circuitos CBR; estos sistemas se caracterizan también por emplear algoritmos de compresión que funcionan con una tasa de bits constante, como el M-JPEG[1].

El protocolo AAL1 garantiza un mínimo retardo, un jitter pequeño y un reducido overhead de proceso y de información de control.

En AAL 1 la subcapa CS se ocupa de compensar las irregularidades que se puedan producir en el tráfico entrante para ajustarlo lo más posible a un caudal constante. Para esto se construye un buffer con unas pocas celdas antes de empezar a entregar los datos a la aplicación correspondiente. El número de celdas utilizado como buffer depende de la velocidad del circuito.

Los datos son recibidos de la aplicación normalmente como un flujo continuo de bits sin ninguna separación que permita identificar en él mensajes discretos. La subcapa CS no tiene un protocolo propio, es decir, no incorpora información de control. La subcapa SAR si lo tiene; utiliza un byte de la parte de carga útil de cada celda para incluir una información de control constituida por un número de secuencia y un CRC del número de secuencia; de esta forma el receptor puede verificar con seguridad que las celdas le están llegando todas y en orden, y corregir pequeñas pérdidas que se puedan producir, o notificar a la aplicación en caso de que las pérdidas sean superiores a lo tolerable.

6.2.10.3 AAL 5

AAL5, que es lo más parecido a un servicio de transporte de datos en ATM, se asemeja en cierto modo a UDP. Se supone que si el usuario desea un transporte fiable incorporará su propio protocolo encima de AAL5. En la práctica, para no reinventar la rueda, se suelen utilizar protocolos ya existentes, por ejemplo TCP. En realidad lo que se suele hacer es encapsular paquetes del nivel de red, por ejemplo datagramas IP, o incluso tramas MAC como veremos luego. Aunque esta solución no es ideal, pues aumenta la cantidad de información de control, es la más extendida actualmente.

Como ya hemos explicado, AAL5 fue propuesto después de los demás AALs por las empresas informáticas como alternativa a AAL3/4, y fue rápidamente adoptado por el ATM Forum y también por la ITU-T. Al principio AAL5 fue denominado SEAL (Simple Efficient Adaptation Layer) lo cual da una idea de los principios que dirigieron su diseño.

[1] La compresión MJPEG consiste en enviar el flujo de vídeo como una secuencia de fotogramas independientes aplicando a cada uno de ellos el algoritmo de compresión JPEG como si se tratara de una fotografía independiente. Al no aprovechar la redundancia temporal de la información de vídeo la eficiencia es menor que con MPEG, pero la tasa de bits es más constante.

En AAL5 la subcapa CS recoge de la aplicación un mensaje discreto que puede tener una longitud de entre 0 y 65.535 bytes. A este mensaje AAL5 le añade una cola de 8 bytes de información y construye un mensaje con la siguiente estructura:

Carga útil (0-65535)	Relleno (0-47)	UU	CPI	Longitud	CRC

- El *relleno* se utiliza para asegurar que la longitud total del mensaje (incluida la cola) sea un múltiplo de 48 bytes.

- El *campo UU* (User to User), de un byte de longitud, queda a disposición de la aplicación para la transmisión de información usuario-usuario de forma transparente (la aplicación puede ser la parte de la subcapa CS específica de la aplicación). Este campo puede utilizarse, por ejemplo, para multiplexar varias conexiones, o para números de secuencia.

- El *campo CPI* (Common Part Indicator), también de un byte, indica el significado del resto de los campos de control. De momento solo se ha definido un significado, que es el que describimos a continuación.

- El *campo Longitud* indica la longitud de la parte de carga útil, sin contar el relleno si lo hubiera. Como tiene dos bytes la longitud máxima es de 65.535 bytes.

- El ultimo campo es un *CRC* de 32 bits, el mismo que se utiliza habitualmente en las redes locales.

En AAL 5 la subcapa SAR se limita a cortar el mensaje que recibe de la subcapa CS en trozos de 48 bytes que acomoda en la parte de carga útil de celdas consecutivas, sin incluir ninguna información de control adicional. Para que el receptor pueda detectar el final de los mensajes se marca como tipo 1 la última celda de cada mensaje, poniendo a 1 el último bit del campo PTI (Payload Type Identifier) en la cabecera ATM de dicha celda.

Como puede verse la labor de la capa de adaptación en AAL5 es relativamente sencilla. Solo se calcula un CRC por mensaje, por lo que si se consigue que los mensajes sean relativamente grandes el ahorro respecto a AAL3/4 es considerable. Esto también hace más rápido el proceso. Además, el hecho de no tener información de control por celda reduce el overhead, ya que los 8 bytes de información de control del mensaje suponen un costo despreciables si éste es bastante grande.

AAL5 es con diferencia el protocolo más utilizado para transmisión de datos a través de redes ATM. Como veremos a continuación las dos aproximaciones más utilizadas para la transmisión de datos en redes ATM, LAN Emulation y Classical IP over ATM, utilizan AAL5. Todo hace pensar que esta tendencia seguirá en el futuro.

6.3 EJERCICIOS

1. Indique si es verdadera o falsa cada una de las siguientes afirmaciones:

a) El bit DE en Frame Relay puede ser puesto tanto por el host emisor como por el conmutador a través del cual se accede a la red.

b) En Frame Relay cuando una trama es descartada por congestión el emisor es notificado del hecho para que realice la retransmisión.

c) Los campos VPI/VCI de una celda ATM pueden tener un valor diferente en cada salto que realizan en la red hasta llegar a su destino.

d) Los parámetros de Calidad de Servicio (PCR, SCR, MCR, etc.) se aplican por igual a todas las Categorías de servicio definidas en ATM.

e) Cuando se establece un circuito CBR en ATM la capacidad reservada y no utilizada por ese circuito puede ser aprovechada por otros circuitos, por ejemplo UBR, que compartan la conexión física.

f) ABR es la única categoría de servicio de ATM que implementa mecanismos de realimentación para el control de la congestión.

g) Cuando se aplican mecanismos de vigilancia del tráfico en una red (traffic policing) siempre que el usuario inyecta un tráfico mayor de lo pactado se descarta el tráfico sobrante.

h) Aunque IP permite que los datagramas lleguen desordenados a su destino, cuando se utiliza AAL5 las celdas que forman parte de un mismo datagrama deben llegar en el mismo orden en que han sido emitidas, pues de lo contrario los datos recibidos serán inútiles.

2. ¿Que categoría de servicio ATM utilizaría si tuviera que simular una línea dedicada?

3. ¿Que categoría de servicio ATM se parece mas a Frame Relay?

4. De acuerdo con el algoritmo del pozal agujereado, si el host envía datos con un caudal superior al del agujero durante un tiempo suficientemente largo el pozal siempre se desborda. Explique como es posible entonces que en algunos casos se pueda en redes Frame Relay enviar datos durante largo tiempo (horas) con un caudal superior al CIR.

5. En IPv4 cada datagrama lleva la dirección de destino, que ocupa 32 bits. Por consiguiente el número máximo de nodos en una red IPv4 es de 2^{32} (unos 4.300 millones). En ATM cada celda generada por un host (celda UNI) lleva un campo VPI-VCI que indica el host de destino, con una longitud total de 24 bits, por lo que el número máximo de nodos en una red ATM es de 2^{24} (unos 16 millones). Es correcto este razonamiento?

6. Diga que campo (o campos) de un datagrama IP cambia cuando éste pasa de un router al siguiente; suponga que no se utilizan campos opcionales y que no se produce fragmentación ni congestión. Diga que campo (o campos) de una celda ATM cambia cuando ésta pasa de un conmutador al siguiente. Suponga que no se produce congestión.

7. La fábrica de juguetes Patinete dispone de oficinas en Ibi (Alicante), Valencia, Sevilla, Bilbao y Madrid. En cada una de estas oficinas hay una LAN a la que se conectan diversos ordenadores con múltiples protocolos, y un router multiprotocolo. Para la interconexión de las oficinas el responsable informático de Patinete contrata cinco accesos físicos de 1,984 Mb/s a Red Uno (servicio frame relay de Telefónica). Cuando descubre que para poder conectar realmente las oficinas ha de contratar además PVCs (Permanent Virtual Circuits) decide contratar el asesoramiento de la empresa Baudio, especializada en comunicaciones. Los CIR (Committed Information Rate) que Telefónica le ofrece son 0, 64 y 256 Kb/s; el EIR (Excess Information Rate) es en todos los casos 384 Kb/s (en la práctica Telefonica ofrece una gama de valores mucho mas amplia y el EIR es un dato que no se facilita al cliente y que no forma parte de las condiciones contractuales).

Baudio se ofrece a realizar un estudio de *todas* las topologías de red posibles, cobrando una cuota de solo 100 pesetas por topología analizada. Considerando que la tarifa es muy razonable, el responsable informático de Patinete encarga el estudio sin pensarlo mas.

Se le pide que:

a) Deduzca una expresión que permita a Baudio calcular el costo de un estudio similar para el caso de *n* accesos físicos frame relay y *m* posibles valores de CIR.
b) Calcule a cuanto ascenderá la factura que Baudio presentará a Patinete.
c) Calcule cual es el caudal de tráfico máximo que puede fluir en total entre las cinco oficinas a las once de la mañana, hora en que la Red Uno está completamente saturada. Repita el mismo cálculo para las tres de la madrugada, cuando la Red Uno está completamente descargada
d) Explique la diferencia entre un PVC con CIR 0 Kb/s y la ausencia de PVC.

Aclaraciones y simplificaciones:

- Considere que los PVCs siempre son full-duplex, es decir, si se contrata un PVC para la comunicación A->B es preciso contratar también un PVC para la comunicación B->A; sin embargo los CIR pueden ser diferentes para cada sentido.
- No es posible establecer un PVC de una oficina consigo misma.
- No considere el caso de mas de un PVC entre dos oficinas.

8. Una empresa dispone de oficinas en Madrid, Barcelona y Sevilla. En cada una de estas oficinas tiene una red local que conecta varios ordenadores, y cada oficina tiene un router con dos interfaces; una lo conecta a la red local, mientras que la otra lo conecta a la red Frame Relay de Telefónica.

En base a una estimación del tráfico previsto se ha calculado la matriz de tráfico, donde se expresa en MBytes la cantidad de información que previsiblemente intercambiarán diariamente las oficinas:

Desde → **Hacia**	Barcelona	Madrid	Sevilla
Barcelona		155	60
Madrid	185		90
Sevilla	45	125	

Las oficinas trabajan de 8 a 20 horas, y todo el tráfico se produce durante la jornada laboral. Para conseguir un tiempo de respuesta aceptable se quiere que el nivel medio de ocupación de las líneas no supere el 30%.

Calcule cual será la configuración óptima de la red, es decir la que satisfaga los requerimientos planteados con un costo mensual mínimo.

Información suplementaria:

Para conectarse a la red Frame Relay cada oficina ha de contratar:

Un acceso físico. El costo del acceso físico depende de la velocidad según la siguiente tarifa:

Velocidad (en Kb/s)	**Costo mensual (en Ptas)**
64	60.241
128	89.630
256	155.422
512	221.687
1984	303.614

Un circuito virtual permanente (PVC) con cada oficina con la que quiera comunicar. El circuito se ha de contratar con un caudal determinado (denominado CIR, Committed Information Rate)

independientemente para cada sentido de la comunicación, y puede ser asimétrico. El costo de un PVC nacional (es decir entre diferentes provincias) depende del CIR según la tarifa siguiente:

CIR (en Kb/s)	Costo mensual (en Ptas)
0	602
8	2.164
16	4.283
32	8.482
48	12.595
64	16.625
96	24.689
128	32.589
192	48.395
256	63.882
384	94.864
512	141.253
1024	170.964
1536	187.200
1984	203.598

9. El servicio de conexión a Internet que ofrece Telefónica, denominado Ibernet, utiliza una red Frame Relay como medio de transporte, de forma que los datagramas IP viajan como datos en tramas Frame Relay. Por tanto para que una empresa se pueda conectar a Internet mediante este servicio ha de contratar:

- Un acceso físico a la red Frame Relay
- Un PVC con Ibernet, especificando el CIR que desea en cada sentido

Una empresa desea conectar a Internet un servidor de vídeo bajo demanda, empleando para ello el servicio Ibernet. El vídeo se suministra como un flujo MPEG-1; se sabe que se emiten 24 fotogramas por segundo, y que uno de cada 12 es un fotograma de referencia (fotograma I en terminología MPEG), siendo el resto fotogramas de diferencias respecto a los anteriores (fotogramas P y B). Se sabe que cada fotograma I ocupa aproximadamente 15000 bytes. Las tramas Frame Relay generadas son todas de 1500 bytes.

Diga cual o cuales de las combinaciones siguientes no provocarán la pérdida de información por desbordamiento del buffer en el conmutador Frame Relay (que regula el tráfico mediante el algoritmo del pozal agujereado):

- Acceso físico de 512 Kb/s, CIR saliente de 384 Kb/s
- Acceso físico de 512 Kb/s, CIR saliente de 512 Kb/s
- Acceso físico de 2048 Kb/s, CIR saliente de 384 Kb/s
- Acceso físico de 2048 Kb/s, CIR saliente de 512 Kb/s

El valor de T (del cual se deduce B_c) es 180 ms.

El CIR entrante es de 64 Kb/s en todos los casos; se supone que en este sentido el flujo de información es mínimo (se utiliza únicamente para las funciones de control de vídeo).

Aproximaciones:

- Ignore el tráfico producido por los fotogramas P y B, y por el audio.

- Considere despreciable el tráfico producido por las cabeceras de los datagramas IP, tramas Frame Relay, etc.

- Considere que el EIR es 0 en todos los casos (se supone que al ser fotogramas I el usuario no quiere correr el riesgo de que parte de las tramas se envíen con el bit DE puesto).

10. En una conexión ATM entre dos hosts se le pide que mida el tiempo de ida y vuelta de las celdas. Para esto dispone únicamente del reloj del sistema en el host emisor, que tiene una resolución máxima de un milisegundo (el reloj da el tiempo en un dato de tipo entero que mide el tiempo contando el número de milisegundos transcurridos desde un instante concreto). Aun cuando la resolución es de un milisegundo se sabe que la precisión del reloj del sistema es de un microsegundo. En una experiencia envía 100.000 celdas y mide el tiempo de ida y vuelta para cada una, resultando 0 milisegundos en 35.000 ocasiones y 1 milisegundo en 65.000 ocasiones. Podría decir cual es el tiempo de ida y vuelta?

6.4 SOLUCIONES

S1.-

a) **Verdadera**. El conmutador pondrá el bit DE en las tramas que no puedan ser enviadas por el primer pozal (cuyo 'agujero' corresponde al CIR). A su vez el host también puede poner a 1 el bit DE de forma voluntaria en tramas poco importantes si desea reservar la capacidad de su primer pozal (que no lleva el bit DE) para las tramas importantes.

b) **Falsa**. No es posible notificar al emisor ya que las tramas no van numeradas, por lo que no hay un mecanismo que permita referirse a ellas de forma individualizada.

c) **Verdadera**. Los valores de VPI/VCI se negocian de forma independiente entre cada dos nodos del camino, por lo que tienen un significado puramente local.

d) **Falsa**. Cada categoría de servicio tiene su conjunto propio de parámetros aplicables.

e) **Falsa**. Un circuito CBR emula una línea dedicada, por lo que la capacidad se le reserva de forma exclusiva; si el circuito no se utiliza en su totalidad la capacidad sobrante es desperdiciada.

f) **Verdadera**. Existen varios mecanismos por los cuales un circuito ABR puede notificar al emisor la existencia de congestión en la red.

g) **Falsa**. En muchos casos el tráfico excedente sale marcado de alguna forma, para poder descartarlo en caso necesario, por ejemplo el bit DE en Frame Relay o el bit CLP en ATM.

h) **Verdadero**. Esto no requiere ninguna acción especial ya que ATM siempre garantiza que se mantiene el orden de los envíos.

S2.-

CBR (Constant Bit Rate).

S3.-

ABR (Available Bit Rate). Esta es la única que implementa control de flujo, cosa que también ocurre con Frame Relay.

S4.-

La razón estriba en que Frame Relay utiliza dos pozales, cada uno con su propio juego de parámetros. El primero tiene un agujero igual al CIR, mientras que el segundo tiene un agujero igual al EIR. El tráfico enviado por el segundo pozal sale con el bit DE puesto. Si la red está saturada las tramas marcadas con DE se descartan en primer lugar, pero si está poco utilizada todas llegaran a su destino. Por tanto en horas de baja utilización un host puede enviar de forma sostenida un caudal igual a CIR+EIR. Bajo ninguna circunstancia puede enviarse por un circuito Frame Relay un caudal superior a este.

S5.-

No es correcto. El número de VPI-VCI no identifica un nodo en la red ATM, sino una conexión virtual desde un nodo con su vecino inmediato. Por tanto lo que limita el campo VPI-VCI es el número máximo de conexiones virtuales simultáneas que puede tener un host de la red ATM. El formato de direcciones ATM NSAP, el mas habitual en redes ATM privadas, emplea direcciones de 20 bytes.

S6.-

En un datagrama IP (IPv4) se modifica el campo TTL (Time to Live) ya que se resta 1 en cada salto. Como consecuencia de esto también se modifica el campo checksum.

En una celda ATM se modifica (en general) el valor de VPI-VCI, y como consecuencia de esto el campo HEC (Header Error Check).

S7.-

a)

Sea m el número posible de valores del CIR.

Para una red entre dos puntos (un único VC posible) hay m^2+1 posibles topologías. Por ejemplo para m=3 se pueden hacer nueve combinaciones diferentes de valores de CIR, y la décima sería la ausencia de Circuito Virtual:

	CIR de B hacia A		
	0 Kb/s	64 Kb/s	256 Kb/s
CIR de A hacia B			
0 Kb/s	0/0	0/64	0/256
64 Kb/s	64/0	64/64	64/256
256 Kb/s	256/0	256/64	256/256

Si en vez de 2 puntos hay n el número de VCs posibles es de $(n^2-n)/2$, pues sería la mitad de la matriz de n x n puntos menos la diagonal. Por ejemplo para 5 puntos:

	A	B	C	D	E
A		X	X	X	X
B			X	X	X
C				X	X
D					X
E					

Las casillas marcadas con X indican los VCs posibles.

Cada uno de estos $(n^2-n)/2$ VCs puede tener m^2+1 posibles valores de CIR. Como cada VC es independiente y distinto del resto el número total de topologías para n puntos y m valores de CIR es:

$$(m^2+1)^{(n^2-n)/2}$$

Por tanto para m = 3 y n = 5 el número de topologías posibles es de $\mathbf{10^{10}}$.

b)

A razónde 100 pesetas por topología el costo total del estudio sería de **10^{12}** pesetas.

c)

El caudal máximo en horas punta será el que corresponde a 10 VCs, cada uno transmitiendo 256 Kb/s en cada sentido:

10 VCs * 2 sentidos * 256 Kb/s = **5,12 Mb/s**

Otra forma válida de razonarlo sería decir que cada nodo tiene 4 VCs contra los demás, y que mete por cada VC 256 Kb/s: 5 nodos * 4 VCs * 256 Kb/s = 5,12 Mb/s

El caudal máximo en horas valle vendrá limitado por el EIR o por la velocidad del acceso físico. El EIR es de 384 Kb/s, por lo que por cada VC podrían pasar 640 Kb/s (EIR + CIR). Razonando como antes el resultado sería:

10 * 2 * 640 Kb/s = 12,8 Mb/s

Sin embargo, con 640 Kb/s por VC cada nodo tendría que meter en la red 640 * 4 = 2,56 Mb/s, lo cual supera la capacidad del acceso físico. Por tanto el caudal máximo en este caso vendrá limitado por la capacidad de los acceso físicos, y será:

5 * 1,984 Mb/s = **9,92 Mb/s**

d)

La diferencia entre un PVC con CIR 0 Kb/s y la ausencia de PVC es la siguiente:

Por un PVC con CIR cero puede circular tráfico siempre y cuando la red no esté saturada y el EIR no sea también cero. En cambio si no existe PVC no puede circular tráfico alguno.

S8.-

Cálculo de los caudales mínimos a partir de la matriz de tráfico:

nMB = número de MegaBytes transmitidos durante el día

nb = número de bits transmitidos durante el día

c_min = caudal mínimo (en Kb/s) para transferir la información en 12 horas

caudal = caudal necesario (en Kb/s) para que la ocupación media no supere el 30%

Se cumple que:

nb = nMB * 1024 * 1024 * 8

c_min = nb / (3600 * 12 * 1000)

caudal = c_min / 0.3

Sustituyendo obtenemos:

caudal = nMB * (1024 * 1024 * 8)/(3600 * 12 * 1000 * 0.3) = 0.647 * nMB

Aplicando esta fórmula calculamos la matriz de caudales reales (valores en Kb/s):

Desde → **Hacia**	Barcelona	Madrid	Sevilla	**Totales hacia**
Barcelona	-	100	39	139
Madrid	120	-	58	178
Sevilla	29	81	-	110
Totales desde	149	181	97	-

A partir de aquí calcularemos el costo para cada una de las cuatro topologías posibles:

1ª topología: tres circuitos:

	Caudal Ida	**CIR Ida**	**Costo**	**Caudal Vuelta**	**CIR Vuelta**	**Costo**
Barcelona-Madrid	120	128	32589	100	128	32589
Barcelona-Sevilla	29	32	8482	39	48	12595
Madrid-Sevilla	81	96	24689	58	64	16625
TOTAL			65760			61809

El funcionamiento de estos circuitos requiere los siguientes accesos físicos:

	Total saliente	**Total entrante**	**Acceso físico**	**Costo**
Barcelona	149	139	256	155422
Madrid	181	178	256	155422
Sevilla	97	110	128	89630
TOTAL				400474

Costo total al mes: 528043 pesetas

Obsérvese que en el cálculo de los accesos físicos se suman los caudales calculados para cada sentido en cada caso, no los que corresponden al CIR contratado, ya que estos últimos superarín lo que requiere el enunciado en sentido estricto.

2ª topología: suprimimos el circuito Madrid-Sevilla:

	Caudal Ida	**CIR Ida**	**Costo**	**Caudal Vuelta**	**CIR Vuelta**	**Costo**
Barcelona-Madrid	178	192	48395	181	192	48395
Barcelona-Sevilla	110	128	32589	97	128	32589
TOTAL			80984			80984

El funcionamiento de estos circuitos requiere los siguientes accesos físicos:

	Total saliente	**Total entrante**	**Acceso físico**	**Costo**
Barcelona	288	278	512	221687
Madrid	181	178	256	155422
Sevilla	97	110	128	89630
TOTAL				466739

Costo total al mes: 628707 pesetas

3ª topología: suprimimos el circuito Barcelona-Sevilla:

	Caudal Ida	**CIR Ida**	**Costo**	**Caudal Vuelta**	**CIR Vuelta**	**Costo**
Barcelona-Madrid	149	192	48395	139	192	48395
Madrid-Sevilla	110	128	32589	97	128	32589
TOTAL			80984			80984

El funcionamiento de estos circuitos requiere los siguientes accesos físicos:

	Total saliente	Total entrante	Acceso físico	Costo
Barcelona	149	139	256	155422
Madrid	249	246	256	155422
Sevilla	97	110	128	89630
TOTAL				400474

Costo total al mes: 562442 pesetas

4ª topología: suprimimos el circuito Barcelona-Madrid:

	Caudal Ida	CIR Ida	Costo	Caudal Vuelta	CIR Vuelta	Costo
Barcelona-Sevilla	149	192	48395	139	192	48395
Madrid-Sevilla	181	192	48395	178	192	48395
TOTAL			96790			96790

El funcionamiento de estos circuitos requiere los siguientes accesos físicos:

	Total saliente	Total entrante	Acceso físico	Costo
Barcelona	149	139	256	155422
Madrid	181	178	256	155422
Sevilla	317	330	512	221687
TOTAL				532531

Costo total al mes: 726111 pesetas

Por tanto la topología más barata es la planteada en primer lugar, según la cual se establecerían tres PVCs entre las tres ciudades.

S9.-

La velocidad del acceso físico representa la velocidad de la conexión host-red, es decir la de la conexión del host con el conmutador Frame Relay al que se encuentra directamente conectado. El CIR fija la velocidad con la que dicho conmutador introducirá tráfico en la red.

Como se generan tramas de 1500 bytes (12000 bits) cada fotograma I de 15000 bytes (120000 bits) generará una ráfaga de diez tramas de 12000 bits cada una. Como hay dos fotogramas I por segundo se produce una de estas ráfagas cada 500 milisegundos.

Caso 1: Acceso físico a 512 Kb/s y CIR de 384 Kb/s

Para aplicar el pozal agujereado calculamos primero el tamaño del buffer:

B_c = CIR * T = 384000 * 0.18 = 69120 bits

Esto equivale a 69120/12000 = 5,76 tramas, por lo que en el pozal caben **5 tramas**.

El tiempo que tarda el host en emitir una trama es: 12000/512000 = 0,0234 s = 23,4 ms Para emitir diez tramas tardará 234 ms.

Por tanto el host estará durante 234 ms emitiendo tramas y en silencio durante 266 ms (500-234). Para saber si se pierde información nos fijaremos precisamente en el instante en que el host acaba de terminar la ráfaga (milisegundo 234), ya que es este el momento en que el pozal estará

mas lleno; si en ese momento no se ha desbordado podemos asegurar que no se desbordará nunca.

En principio con un CIR de 384 Kb/s se habrán enviado a la red 0,234 * 384000 = 89856 bits, equivalentes a 7,488 tramas; sin embargo aquí no hemos tomado en cuenta que para poder enviar el primer bit a la red el conmutador ha de recibir antes la primera trama completa, ya que solo después de haberla recibido del host y verificado el CRC se puede empezar la transmisión. Esto supone que durante los primeros 23,4 no se envía nada a la red, por lo que solo se ha estado enviando durante 234 - 23,4 = 210,6; la cantidad de bits enviados es pues 0,2106 * 384000 = 80870 bits equivalentes a 6,74 tramas; por tanto a los 234 ms de iniciada la ráfaga se habrán enviado a la red **6 tramas**.

Por tanto en este caso **no se pierde información ya que se han enviado 6 y tenemos 4 en el pozal que tiene capacidad para 5. Se habría podido soportar una ráfaga de hasta 16 tramas sin perder datos**.

Caso 2: Acceso físico a 512 Kb/s y CIR de 512 Kb/s

Aquí no hay retención de tráfico en el conmutador pues el caudal de entrada y de salida son idénticos**. No existe pérdida de información cualquiera que sea la longitud de la ráfaga recibida del host**.

En este caso es la propia línea de acceso frame relay la que gracias a su bajo caudal amortigua el efecto de la ráfaga del host. La limitación vendría impuesta por el acceso físico, que podría como máximo aceptar de la aplicación 21 tramas cada 500 milisegundos (500/23,4 = 21,4)

Caso 3: Acceso físico a 2048 Kb/s y CIR de 384 Kb/s

Repetimos los cálculos de forma análoga al caso 1 anterior:

El tamaño del pozal (B_c) depende únicamente del CIR y T, que en este caso tienen el mismo valor que en el caso 1, por lo que en el pozal caben **5 tramas**

Tiempo para emitir una trama: 12000/2048000 = 0,00586 s = 5,86 ms

10 tramas: 58,6 ms

El host emite diez tramas en 58,6 ms y está en silencio durante 441,4 ms (500-58,6).

Veamos ahora que ocurre en el instante crítico, que es cuando el host ha terminado de enviar la ráfaga al conmutador, cosa que ocurre a los 58,6 ms de haber empezado a emitir la ráfaga.

Procediendo como antes restamos el tiempo de llegada de la primera trama al conmutador, ya que hasta que no tiene una trama entera no empieza a transmitir: 58,6 - 5,86 = 52,74 ms. En 52,74 ms con un CIR de 384 Kb/s se envían a la red 0,05274 * 384000 = 20252 bits, equivalentes a 1,69 tramas;, por lo que se envía **una trama**. Como en el pozal caben 5 tramas, **cuando el host acaba de emitir las diez tramas se han perdido cuatro**.

La máxima ráfaga en este caso sin perder datos sería de cinco tramas.

Caso 4: Acceso físico a 2048 Kb/s y CIR de 512 Kb/s

B_c = 512000 * 0.18 = 92160 bits 92160/12000 = 7,68 = **7 tramas**

Tiempo para emitir una trama: 12000/2048000 = 0,00586 s = 5,86 ms

10 tramas: 58,6 ms

Como en el caso anterior el host emite diez tramas en 58,6 ms y está en silencio durante 441,4 ms.

Las tramas enviadas a la red a los 58,6 ms son: 0,05274 * 512000 = 27003 bits, equivalente a 2,25 tramas, por lo que se han enviado **dos tramas**. Como en el pozal caben 7 tramas **cuando el host acaba de emitir las diez tramas se ha perdido una**.

La máxima ráfaga en este caso sin perder datos sería de nueve tramas.

S10.-

Se calcula la media ponderada de todos los valores:

(35.000*0 + 65.000*1)/100.000 = **0,65 ms**

Solución Alternativa:

Es evidente que el valor pedido es inferior a 1 ms. En este caso la probabilidad p de que el reloj cambie mientras la celda está de viaje es proporcional a la relación entre el tiempo de ida y vuelta (RTT) y la resolución del reloj, 1 ms:

p = RTT / 1 ms, por tanto RTT = p * 1 ms

Como p = 65.000/100.000 = 0,65, entonces RTT = **0,65 ms**

7 INTERNETWORKING

Autor: Rogelio Montañana

7.1 INTRODUCCIÓN

El término internetworking se utiliza para designar la unión de redes diferentes a cualquier nivel (físico, de enlace, etc.) de forma que desde los niveles superiores se aprecie como una única red homogénea. Las redes pueden diferir en el medio físico (por ejemplo Ethernet-Token Ring o LAN-WAN) o en la pila de protocolos utilizados (TCP/IP, DECNET o SNA, por ejemplo).

7.1.1 Dispositivos básicos de Internetworking

Algunos de los dispositivos utilizados para realizar internetworking son los siguientes:

- **Repetidores y amplificadores**: estos dos tipos de dispositivos funcionan a nivel físico. Los repetidores retransmiten la señal digital bit a bit, mientras que los amplificadores actúan sobre la señal analógica. Por ejemplo en Ethernet 10-BASE5 se utilizan repetidores cuando la distancia supera los 500 metros Los repetidores mantienen intacta la señal mientras que los amplificadores la distorsionan; por este motivo normalmente se puede atravesar un número ilimitado de repetidores (al margen de otras consideraciones) mientras que el número de amplificadores por los que puede pasar la señal sin regenerarse está limitado.

- **Puentes (bridges) y conmutadores LAN**: son dispositivos que analizan la trama a nivel de enlace. Se utilizan sobre todo en LANs 802.x, donde funcionan en la subcapa MAC (por lo que también se denominan puentes MAC). Permiten una cierta optimización del tráfico y la interconexión de LANs de diferentes tipos (con algunas restricciones). Los conmutadores LAN utilizados actualmente, son en realidad puentes multipuerta con conmutación por hardware. Es posible montar una red LAN-WAN con puentes remotos, aunque las posibilidades de routing que da el Spanning Tree son tan escasas que esto solo es viable en redes muy simples. Además los puentes hacen un uso tremendamente ineficientes de los enlaces WAN pues hacen pasar por ellos todo el tráfico broadcast/multicast. Diremos por último que todo lo anterior está referido a los puentes transparentes; los puentes por encaminamiento permiten un mayor control de las rutas y presentan algunas ventajas, sobre todo en el caso de las conexiones WAN, ya que en realidad desempeñan funciones de routing; sin embargo plantean otro tipo de inconvenientes, fundamentalmente la complejidad que requiere su implementación en cada host de la red y el hecho de que sólo se puedan utilizar en redes Token Ring.

- **Routers y conmutadores de nivel de red (ATM, F.R:, X.25)**: permiten la interconexión a nivel de red, analizan el contenido del paquete (dirección de destino o número de circuito virtual) y toman decisiones de encaminamiento en base a sus tablas de rutas o de circuitos establecidos. Se suelen utilizar para interconectar LANs con WANs, y también para conexiones LAN-LAN o WAN-WAN. A diferencia de los puentes los routers no mantienen una tabla exhaustiva de todas las direcciones de la red sino que realizan una sumarización gracias a la estructura jerárquica de las direcciones de red. Por su parte los conmutadores, al identificar de forma explícita las conexiones establecidas, consiguen reducir el tamaño de la tabla utilizada para enrutar los paquetes. A diferencia de los puentes los routers (y los conmutadores) no propagan el tráfico broadcast/multicast a nivel de enlace, por lo que restringen el ámbito de funcionamiento de protocolos como ARP.

  Muchos routers pueden también actuar como puentes MAC, e incluso como routers para unos protocolos y como puentes para otros. Se suele utilizar la denominación *brouter* para designar estos equipos, indicando así su doble funcionalidad.

  La mayoría de los routers actuales soportan múltiples protocolos; así es posible con un solo router unir dos LANs en las que coexistan TCP/IP y DECNET, por ejemplo, sin necesidad de utilizar múltiples líneas o routers; es importante destacar que un router multiprotocolo no suministra una comunicación *entre* los diferentes protocolos que soporta, simplemente permite que los diversos protocolos de red compartan la infraestructura de comunicaciones, es decir si dos LANs tienen máquinas unas con TCP/IP y otras con DECNET un router multiprotocolo

permitirá la comunicación entre las máquinas TCP/IP de una y otra LAN, o de las máquinas DECNET de una y otra LAN, pero no la comunicación de una máquina DECNET con una TCP/IP.

Respecto al protocolo de routing un router multiprotocolo puede funcionar de dos maneras:

- *Routing integrado*: todos los protocolos enrutados comparten un protocolo de routing común; en este caso cada router mantiene únicamente una tabla de rutas, y efectúa el cálculo de rutas únicamente una vez; todos los protocolos enrutados utilizan las mismas rutas, por lo que todos los routers de la red deben soportar todos los protocolos. Ejemplos de protocolos de routing integrado son el IS-IS y EIGRP, que están preparados para entornos multiprotocolo.

- *Buques en la noche ('Ships in the night', SIN):* esta técnica consiste en utilizar un protocolo de routing diferente para cada protocolo enrutado; en este caso los algoritmos utilizados y la topología de la red puede variar para cada protocolo y los cálculos han de hacerse independientemente para cada uno; evidentemente las rutas elegidas pueden no ser las mismas para todos los protocolos. Cada router encamina los paquetes de cada protocolo de forma independiente y éstos viajan por las líneas sin verse unos a otros, de ahí el nombre de Ships in the Night que recibe esta técnica. Con esta técnica cada router sólo ha de enrutar los paquetes correspondientes a aquellos protocolos en los que participa.

Por ejemplo la red de la Universidad de Valencia soporta dos protocolos, TCP/IP y Appletalk. Como protocolo de routing se utiliza EIGRP que permite routing integrado, con lo que se simplifica la cantidad de proceso en los routers, que de esta forma no han de realizar dos veces el cálculo de las rutas óptimas.

- **Pasarelas**: cualquier dispositivo que permite la intercomunicación a niveles superiores al de red se denomina genéricamente pasarela (gateway en inglés). Así nos referimos a pasarelas del nivel de transporte o del de aplicación. Una pasarela del nivel de aplicación es un host que implementa dos (o mas) pilas completas de protocolos y que puede 'trasvasar' datos de una aplicación a su equivalente en la otra pila de protocolos, realizando las conversiones adecuadas. Un ejemplo de esto es el intercambio de mensajes de correo electrónico, por ejemplo entre SMTP (protocolo de correo en Internet) y X.400 (protocolo de correo OSI). Otro ejemplo es el servicio que permite enviar desde un formulario Web un mensaje corto a un teléfono GSM.

  Los problemas que se plantean en la interconexión de redes diferentes a nivel de pasarelas de aplicación son en cierto modo parecidos a los que se dan a niveles inferiores, como ocurre al interconectar redes Ethernet y Token Ring. Por ejemplo, si se dispone una pasarela de correo electrónico entre dos redes diferentes, una de las cuales implementa acuse de recibo y la otra no, se plantea el problema de qué hacer en la pasarela cuando un mensaje es enviado de la primera a la segunda; a lo sumo se podría acusar recibo cuando el mensaje se recibe en la pasarela, pero eso no significa que el mensaje haya llegado a su destino; la otra alternativa sería simplemente no enviar ningún acuse de recibo en este caso.

  Además de permitir la interconexión de protocolos distintos las pasarelas de aplicación también se emplean para conectar redes con el mismo protocolo. Existen diversas razones por las que esto puede ser conveniente, como por ejemplo las siguientes:

  - **Seguridad (cortafuegos):** si se quiere tener un control riguroso del tráfico entre una determinada red y el exterior se dispone un cortafuegos que aísle dicha red; a menudo los cortafuegos (o firewalls) actúan como pasarelas a nivel de transporte o de aplicación.

  - **Eficiencia (servidores proxy/cache):** los servidores cache permiten capturar una copia de la información que un usuario se trae del exterior para disponer de ella a<nte la eventualidad de que más tarde otro usuario requiera la misma información. Esto mejora el tiempo de respuesta al usuario ya que la información solicitada se le sirve localmente y reduce el nivel de ocupación de la línea WAN o la conexión a Internet, normalmente de

elevado costo. Para realizar su función los servidores caché actúan como pasarelas del nivel de aplicación, normalmente para el protocolo http.

- **NAT (traducción de direcciones):** Como veremos más adelante existen situaciones en las que es necesario manejar un direccionamiento diferente en la red interna y la externa; en estos casos es preciso utilizar un dispositivo NAT (Network Address Translation) que nos permita el intercambio de paquetes entre ambas redes. Algunas aplicaciones requieren una complejidad mayor en el proceso de traducción de direcciones hasta el punto de que esto solo es posible mediante un programa que interprete y modifique la información contenida en el paquete a nivel de aplicación.

7.2 CORTAFUEGOS

Especialmente con el crecimiento comercial de la Internet muchas empresas se enfrentan ante la necesidad de conectar sus redes al exterior; sin embargo esto plantea varios problemas de seguridad por diversos motivos, por ejemplo:

- Los ordenadores de la red local contienen información de carácter confidencial cuya salvaguardia resulta vital para la empresa.

- Del exterior pueden llegar virus u otro tipo de programas que perjudicarían seriamente los ordenadores de la empresa.

- Los empleados pueden utilizar la conexión a Internet para salir a lugares no autorizados (por ejemplo servidores de información no relacionados con su actividad en la empresa).

Para resolver los problemas de acceso seguro hacia/desde el exterior se ha creado el concepto de *cortafuego* o *firewall*, que consiste en un dispositivo formado por uno o varios equipos que se sitúan entre la red de la empresa y la red exterior (normalmente la Internet); el cortafuego analiza todos los paquetes que transitan entre ambas redes y filtra los que no deben ser reenviados, de acuerdo con un criterio establecido de antemano.

En su versión más simple el cortafuego consiste únicamente en un router en el que se han configurado diversos filtros, por ejemplo impidiendo o limitando el acceso a determinadas direcciones de red, o el tráfico de ciertas aplicaciones o una combinación de ambos criterios. Dado que los usuarios siguen teniendo conexión directa a nivel de red con el exterior esta solución no es muy fiable; además las posibilidades de definir filtros en los routers son limitadas y el rendimiento se resiente de forma considerable si al router se le carga con una tarea de filtrado compleja.

El siguiente nivel de cortafuego está formado por un host que conecta por una parte a la Internet y por otra a la red corporativa, actuando él como router. El host implementa un servidor Web proxy que actúa como pasarela de aplicación para los servicios que se quieren permitir, limitado por las restricciones, filtros o reglas que se han especificado. El ordenador que actúa como barrera entre la red interna y la Internet se denomina host bastión (‘bastion host’) en terminología de cortafuegos. Esta solución ofrece una seguridad mayor que la anterior ya que el servidor proxy , al ser un host, puede procesar filtros mas complejos que un router; pero si un usuario malintencionado (hacker o cracker) consiguiera instalar un programa 'espía' (sniffer) en el host bastión podría capturar tráfico de la red interna de la empresa, ya que está directamente conectado a la LAN.

En un nivel mayor de seguridad el cortafuego se implementa mediante un host y dos routers, conectados entre sí por una pequeña red local; uno de los routers conecta a su vez con la red local de la empresa y el otro con la Internet; el host implementa una pasarela multiaplicación (servidor proxy) como antes, pero al no estar el directamente conectado a la red local de la empresa incluso en el caso de que un hacker consiguiera instalar en él un sniffer no podría capturar tráfico confidencial, pues solo podrá ver los paquetes dirigidos a él. En este modelo la red que une los routers con el host se denomina *zona desmilitarizada* o zona DMZ (Demilitarized Zone, algo así como una zona neutra de seguridad). En ocasiones se utilizan otras variantes de arquitectura de cortafuego aún mas complejas.

En seguridad de redes, además de una buena arquitectura de cortafuego es importante comprobar que no hay 'agujeros' que permitan el acceso no controlado por otras vías. Por ejemplo, en una empresa con una red altamente protegida podría una oficina regional haber establecido una conexión local con otra institución sin conocimiento de los responsables de seguridad, o disponer de un sistema no seguro de autentificación de usuarios en el acceso por red conmutada para teletrabajo. En suma, que además de un buen cortafuego hay que tener una política de seguridad rigurosa si se quiere que las medidas sean efectivas, de lo contrario habremos puesto una puerta blindada, pero nos pueden entrar ladrones en casa por el 'trastero' con gran facilidad.

7.3 TÚNELES

En ocasiones se quiere intercambiar paquetes entre dos redes que utilizan el mismo protocolo, pero que están unidas por una red que utiliza un protocolo diferente. Por ejemplo supongamos que un banco dispone de una red de área extensa con protocolo SNA que une todas sus oficinas, y dos de éstas disponen además de redes locales TCP/IP. Si se desea que estas dos oficinas intercambien tráfico TCP/IP sin establecer para ello una nueva línea ni instalar routers multiprotocolo en todo el trayecto se puede establecer un túnel entre ambas oficinas. Los dos nodos SNA ubicados en los extremos del túnel serán los encargados de añadir a los paquetes IP la cabecera SNA adecuada para que lleguen al otro extremo. Los paquetes IP viajarán ‘encapsulados’ a través del túnel en paquetes SNA, de forma que los paquetes IP no sean vistos por la red SNA. Los conceptos de túnel y de encapsulado de paquetes van siempre asociados entre sí.

En Internet hay situaciones en las que los túneles se utilizan de forma habitual; por ejemplo la interconexión de routers multicast a través de routers unicast para constituir la red MBone, o la interconexión de ‘islas’ IPv6 para constituir el 6bone. En estos casos los túneles se utilizan para enviar paquetes del mismo protocolo (IP) pero de distinto tipo (multicast en unicast) o versión (IPv6 en IPv4). En algunos casos resulta útil encapsular incluso un paquete en otro del mismo tipo y versión; por ejemplo encapsulando un paquete IPv4 en otro podemos forzar la ruta que ha de seguir, cumpliendo así una función similar a la opción ‘loose source routing’ de la cabecera IP, pero sin las limitaciones que tiene ésta en el número de direcciones. Otro uso del encapsulado podría ser para superar el valor máximo del campo TTL; al utilizar encapsulado la cabecera interior empezaría a descontar su TTL sólo a partir del punto de salida del túnel.

Uno de los usos más habituales de los túneles actualmente es para la creación de redes privadas virtuales o VPNs, como veremos a continuación.

Los túneles no son una solución deseable en sí mismos, ya que a lo largo del túnel los paquetes han de llevar dos cabeceras, lo cual supone un mayor overhead; además los extremos del túnel se convierten en puntos simples de fallo y potenciales cuellos de botella en el rendimiento de la red.

7.3.1 Redes Privadas Virtuales (VPNs)

Entendemos por red privada virtual o VPN (Virtual Private Network) la interconexión de un conjunto de ordenadores haciendo uso de una infraestructura pública, normalmente compartida, para simular una infraestructura dedicada o privada. Entre las características propias de una red privada virtual se encuentra la posibilidad de utilizar direccionamiento no integrado en la red del proveedor del servicio.

Existen muchas maneras de crear y utilizar VPNs, tanto en IP como en otros protocolos. Aquí solo comentaremos algunas de sus posibilidades más habituales para que sirvan como ejemplo de su aplicación.

Supongamos que un viajante quiere conectarse remotamente a la red de su empresa para consultar una base de datos y para ello emplea los servicios de un proveedor comercial cualquiera, ya que esto le permite conectarse a su red pagando tarifa metropolitana. Supongamos también que por razones de seguridad o de licencias de software la empresa se ha visto obligada a limitar el acceso a dicha base de datos únicamente a los ordenadores de la red de la empresa, es decir aquellos que tienen direcciones IP de

su red (supongamos que se trata de la red 199.1.1.0/24). Como nuestro viajante al conectarse recibe una dirección IP de la red de su proveedor (supongamos que se trata de la red 200.1.1.0/24) no le es posible acceder a dichos servicios.

Este problema puede resolverse creando un túnel VPN (Virtual Private Network) entre el ordenador del viajante y un ordenador de la red de su empresa. El túnel le va a permitir a nuestro usuario remoto acceder a la red de la empresa como si estuviera conectado en la red local (aunque con una velocidad que dependerá obviamente de la conexión física que utilice). El túnel se creará entre su ordenador y un equipo que denominaremos 'servidor de túneles' ubicado en la red local de la empresa, el cual se encargará de encapuslar y desencapsular los paquetes de este usuario. Veamos en detalle como funcionaría el túnel VPN en este caso concreto.

En primer lugar, la empresa deberá tener el servidor de túneles conectado a su red local; dicho servidor puede ser un equipo específicamente diseñado para este fin, un router o un servidor de propósito general Linux o Windows 2000 Server por ejemplo (el software necesario para actuar como servidor de túneles VPN está incluido en Windows 2000 server). Supongamos que el servidor de túneles está conectado a la red local mediante una interfaz Fast Ethernet y que tiene la dirección IP 199.1.1.10. Además de disponer de una dirección propia el servidor de túneles debe tener asignado un rango de direcciones que utilizará para establecer los túneles. Supongamos que no esperamos más de 10 usuarios simultáneos de este servicio, por lo que reservamos para este fin las últimas 10 direcciones útiles de la red de la empresa, es decir el rango que va de la 199.1.1.245 a la 199.1.1.254 ambas inclusive.

Supongamos ahora que nuestro viajante se conecta por red telefónica a su proveedor habitual y que mediante la asignación dinámica de direcciones de PPP recibe de éste la dirección 200.1.1.20. A continuación se conecta con el servidor de túneles y, previa autentificación mediante usuario/contraseña, el servidor crea un túnel para él y le asigna la dirección 199.1.1.245. A partir de ese momento los datagramas que envíe el viajante llevarán dos cabeceras, una exterior y otra interior. Si el viajante hace desde su PC un ping a una dirección cualquiera de la empresa (por ejemplo la 199.1.1.69) enviará un datagrama cuya cabecera exterior llevará como dirección de origen 200.1.1.20 y de destino 199.1.1.10 (el servidor de túneles) y en la cabecera interior llevará como dirección de origen 199.1.1.245 y como destino 199.1.1.69; este datagrama hará su viaje en dos etapas; en la primera llegará al servidor de túneles (199.1.1.10), que lo despojará de su cabecera exterior y procesará a continuación la cabecera interior, enviando el datagrama al destino deseado en la red de la empresa. El datagrama de respuesta al ping seguirá un proceso simétrico inverso, es decir hará la primera parte de su viaje con una sola cabecera que contendrá 199.1.1.69 como dirección de origen y 199.1.1.245 como destino; ese datagrama será entregado al servidor de túneles, que se encargará entonces de incorporarle la cabecera exterior con dirección de origen 199.1.1.10 y de destino 200.1.1.20; cuando llegue al ordenador del viajante el software cliente VPN le despojará de la cabecera exterior y lo procesará como un datagrama dirigido a él proveniente de 199.1.1.69. Así pues el uso del túnel VPN permite a nuestro viajante acceder a la red de su empresa como si estuviera conectado en la red local detrás del servidor de túneles. El software necesario para crear túneles VPN como el descrito en este ejemplo está integrado en sistemas tales como Windows 98 o Windows 2000 profesional.

Obsérvese que mientras está operativo el túnel todo el tráfico del usuario remoto con cualquier destino de Internet (pertenezca o no a la empresa) se realiza a través del servidor de túneles. Esto significa que el usuario sufrirá las limitaciones asociadas a la capacidad del servidor de túneles y, si accede a destinos ubicados fuera de la red local de su empresa sufrirá también las limitaciones que le imponga la conexión a Internet que tenga su empresa; por tanto lo lógico sería utilizar la conexión a través del túnel únicamente cuando se requiera acceder a servicios de acceso restringido dentro de la red de la empresa.

Además de conectar un ordenador aislado como en el ejemplo anterior es posible establecer un túnel VPN entre routers. Esto nos permite conectar toda una red local a través de un solo equipo, y además no requiere soporte de túneles en el host final. En este caso los routers serán los encargados de realizar la labor de encapsulado/desencapsulado de datagramas. Las VPNs permiten conectar por ejemplo una oficina remota a una oficina principal utilizando Internet, con lo que los costes se pueden reducir considerablemente, especialmente si se trata de conexiones de larga distancia. Además, la reciente aparición de conexiones a Internet de alta velocidad y bajo costo (fundamentalmente ADSL y cable módems) hace especialmente atractivo el uso de túneles VPN sobre este tipo de servicios para constituir

enlaces internos en una red sin incurrir en los elevados costos de constituir una infraestructura dedicada mediante enlaces punto a punto o circuitos frame relay o ATM.

7.4 IPSEC

Con bastante frecuencia cuando se establecen túneles VPN como hemos visto en los ejemplos anteriores, además de la integración de las direcciones de red se plantea un requerimiento desde el punto de vista de la seguridad, dado que los datagramas viajan por una infraestructura pública en la que la información puede ser capturada y/o modificada por usuarios externos. Por este motivo la constitución de túneles VPN viene acompañada a menudo de un requerimiento de seguridad, constituyendo lo que podríamos denominar redes privadas virtuales seguras.

El problema de una comunicación segura a través de una red se resuelve normalmente a nivel de enlace, a nivel de red o a nivel de aplicación. Cada una de estas tres alternativas tiene sus ventajas e inconvenientes:

- **Nivel de enlace:** La seguridad a nivel de enlace se implementa en los dispositivos que se conectan al medio de transmisión, por ejemplo dos routers que se comunican mediante una línea punto a punto. En este caso los mecanismos de seguridad se pueden aplicar de forma transparente al protocolo utilizado a nivel de red. Sin embargo en una red grande requiere encriptar y desencriptar la información en cada salto que da el paquete; aun en el caso de utilizar dispositivos que realicen esta tarea por hardware el retardo que esto puede introducir cuando el número de saltos es elevado puede hacer inviable el uso de aplicaciones en tiempo real que requieren cierto nivel de Calidad de Servicio. Además implementar seguridad a nivel de enlace requiere controlar la infraestructura de la red, cosa que no es factible cuando se utilizan los servicios de un operador o un ISP. Un ejemplo de seguridad a nivel de enlace se da en las redes de televisión por cable (CATV) en que la información viaja encriptada (DES) entre el Cable Módem y el CMTS (Cable Modem Termination System); al ser las redes CATV un medio broadcast es necesario el uso de encriptación para evitar la captación de información por parte de usuarios que no son los destinatarios de la misma.

- **Nivel de red:** Esta es la aproximación adoptada por los estándares IPSec. En este caso la seguridad se limita al protocolo IP y otros protocolos sólo podrán aprovecharla si se encapsulan previamente en paquetes IP. La seguridad a nivel de red puede aplicarla el usuario de forma transparente al proveedor del servicio y encaja de forma muy adecuada con el concepto de VPNs pudiendo crear lo que denominamos redes privadas virtuales seguras.

- **Nivel de aplicación:** esta es la aproximación adoptada por ejemplo en correo electrónico por PEM (Privacy Enhanced Mail) o PGP (Pretty Good Privacy), en HTTP por Secure HTTP o SSL (Secure Sockets Layer), o por SNMP versión 3. El principal inconveniente de abordar la seguridad a nivel de aplicación estriba precisamente en la necesidad de incorporar funcionalidades similares en cada uno de los protocolos del nivel de aplicación que deban utilizarlas, replicando así gran cantidad de tareas en diferentes partes de código. La ventaja es que la seguridad se puede desarrollar de forma selectiva, aplicándola únicamente en el intercambio de información confidencial o importante. Además, al aplicarse los mecanismos de encriptado o validación de la información en el nivel más alto posible el nivel de seguridad obtenido es máximo ya que se reduce el riesgo de que la información pueda ser interceptada o modificada por otros usuario o procesos distintos del destinatario de ésta.

Denominamos IPSec al conjunto de estándares que trata todo lo relacionado con el intercambio seguro de información a través de Internet. En realidad IPSec es una arquitectura (descrita en el RFC 2401, de 66 páginas), y un conjunto de protocolos y mecanismos de autenticación y encriptado.

Las principales funcionalidades que incorpora IPSec son las siguientes:

- AH (Autentication Header): en este caso no se pretende garantizar la confidencialidad de la información sino únicamente su veracidad. La cabecera de autentificación asegura al receptor

que el datagrama no ha sido alterado durante su viaje por la red, ni en su contenido ni en su cabecera (salvo por la modificación del campo TTL y por consiguiente del checksum, que se han de realizar en cada salto); por consiguiente además de garantizarse el contenido se garantiza la autenticidad del remitente. AH se describe en el RFC 2402, de 22 páginas.

- ESP (Encapsulating Security Payload): garantiza la confidencialidad de la información. La parte de datos del datagrama (incluida la cabecera de transporte) viaja encriptada de forma que sólo el destinatatrio pueda descifrar su contenido. Opcionalmente ESP puede incluir la funcionalidad de AH, es decir garantizar que el contenido no ha podido ser alterado por terceros. ESP se describe n el RFC 2406, de 22 páginas.

- ISAKMP (Internet Security Association and Key Management Protocol): consiste en una serie de mecanismos seguros para realizar, de forma manual o automática, el intercambio de claves necesarias para las labores de encriptado y autentificación realizadas por AH y ESP. ISAKMP se describe en el RFC 2408, de 86 páginas.

Podemos distinguir dos modos de funcionamiento de IPSec:

a) Modo transporte: la encriptación se realiza extremo a extremo, es decir del host de origen al host de destino. Para extender en una empresa el uso de IPSec en modo transporte es necesario que todos los hosts tengan una implementación de IPSec.

b) Modo túnel: el encriptado se efectúa únicamente entre los routers de acceso a los hosts implicados. En este caso la información viaja no encriptada en la parte de la red local. El funcionamiento de IPSec en modo túnel permite integrar de forma elegante IPSec en una VPN, ya que el mismo dispositivo que realiza el túnel VPN se encarga de realizar las labores correspondientes al túnel IPSec.

Evidentemente el modo transporte es más fiable puesto que ofrece comunicación segura host a host. Sin embargo el modo túnel tiene la ventaja de que permite incorporar la seguridad sin necesidad de incorporar IPSec en los hosts; aunque la seguridad que se obtiene en este caso no es tan alta la sencillez de implantación es mucho mayor y se consiguen la mayor parte de los beneficios del modo transporte, ya que se protege la parte más expuesta del trayecto, que corresponde precisamente a la infraestructura pública o del operador. En función de las circunstancias que rodeen cada caso se deberá optar por una u otra, pudiendo haber incluso situaciones híbridas en las que en una misma empresa determinado tipo de información baste protegerla con el modo túnel mientras que para algún host concreto, que maneje información de mayor importancia, se deba utilizar el modo transporte.

Aunque en IPSec se prevé la posibilidad de utilizar una amplia diversidad de algoritmos de autentificación y encriptado el único exigido para todas las implementaciones es el DES (Data Encription Standard) que utiliza claves de 56 bits. Desde hace unos años se sabe que DES es relativamente poco seguro, ya que el código puede ser descifrado utilizando fuerza bruta en un tiempo no demasiado grande con un ordenador potente actual, por lo que también se suele utilizar bastante Triple DES, que es mucho más seguro aunque también algo más costoso de calcular. En un futuro próximo se prevé utilizar el algoritmo AES (Advanced Encryption Standard) del cual aún no existen implementaciones comerciales.

Uno de los problemas que plantea la encriptación es el consumo intensivo de CPU. Esto suele ser un problema especialmente en los routers y servidores de túneles que atienden túneles IPSec con la función ESP activada (la función AH no requiere encriptación). En estos casos se pueden concentrar cientos de usuarios y flujos de varios Megabits por segundo en un mismo equipo, que ha de realizar las labores de encriptado/desencriptado para todos ellos, pudiendo verse limitado el rendimiento de las comunicaciones por la capacidad de la CPU del equipo utilizado. Para evitar este problema muchos servidores de túneles y routers permiten incorporar módulos que realizan los algoritmos de encriptación por hardware para hacerlos con mucha mayor rapidez.

7.5 TRADUCCIÓN DE DIRECCIONES (NAT)

Hace unos años cuando una organización deseaba conectar su red de forma permanente a la Internet lo normal era que solicitara al NIC una red adecuada a sus necesidades (clase B o C normalmente) y asignara números IP de dicha red a todas sus máquinas. De esta forma todos los ordenadores tenían acceso directo a Internet con todas las funcionalidades. Pero debido al crecimiento exponencial de Internet cada vez resulta más difícil obtener números IP del NIC correspondiente; por otro lado, en muchas ocasiones no se necesita, o incluso no se desea, disponer de un acceso directo a Internet con completa funcionalidad, por razones de seguridad fundamentalmente. Estos dos motivos, la seguridad y la dificultad para conseguir direcciones públicas, han impulsado a las organizaciones a hacer un mayor uso de las redes privadas según los rangos especificados en el RFC 1918 (10.0.0.0, 172.16.0.0 a 172.31.0.0, y 192.168.0.0 a 192.168.255.0). Estas redes privadas no pueden en principio intercambiar datagramas directamente con el exterior, por lo que han de utilizar un equipo intermedio que se ocupe de realizar la traducción de direcciones o NAT (Network Address Translation).

El NAT puede realizarse en un host (por ejemplo en Linux la función NAT se denomina 'IP Masquerade') aunque también se implementa en muchos routers. Dado que generalmente la función de NAT se realiza en la frontera entre una red local y el exterior la opción del router es bastante popular.

Normalmente NAT solo traduce paquetes IP correspondientesa ICMP o a los protocolos de transporte TCP y UDP. El resto de protocolos (fundamentalmente protocolos de routing) no se traducen pues no tendría sentido intercambiar información de routing a través de un NAT. Otra restricción que impone el NAT es que solo puede haber un solo punto de comunicación entre la red privada y la red pública, es decir la conexión al exterior solo puede hacerse en un router, aunque dicho router puede mantener conexiones con varios ISPs simultáneamente.

7.5.1 Tipos de NAT

Inicialmente los NAT solo permitían iniciar conexiones desde 'dentro', es decir desde la red privada hacia el exterior. Este modo de funcionamiento, que se considera una medida de seguridad, se denomina **NAT tradicional**. Mas recientemente se ha extendido el uso de NATs que también permiten el inicio de la sesión desde fuera, por lo que a este tipo de NAT se le denomina **NAT bidireccional**.

La traducción se puede hacer de dos maneras: modificando únicamente la dirección IP, que es lo que denominamos **NAT básico**, o cambiando también el número de puerto TCP o UDP, que llamaremos **NAPT (Network Address Port Translation)**.

Por último, si la traducción entre la dirección pública y la privada se realiza de acuerdo con una tabla de equivalencia que se carga en la configuración del dispositivo NAT y si dicha tabla no se modifica dinámicamente decimos que se trata de **NAT Estático**. En cambio si la tabla de equivalencia es gestionada dinámicamente por el dispositivo NAT de forma que las direcciones y/o números de puerto se puedan reutilizar decimos que tenemos un **NAT dinámico**. Generalmente el NAT estático es bidireccional mientras que el NAT dinámico es unidireccional (también llamado tradicional).

Combinando el NAT Básico o el NAPT con las modalidades estática y dinámica obtenemos cuatro combinaciones de NAT que pasamos a describir a continuación:

- NAT básico estático. La tabla de equivalencias IP privada – IP pública está incluida en la configuración del dispositivo NAT y solo se modifica por decisión del administrador de la red. La correspondencia es biunívoca, es decir a cada dirección privada le corresponde una pública y viceversa. Los números de puerto no se modifican. Con este tipo de NAT es preciso disponer de un número de direcciones públicas igual al de direcciones privadas. Este NAT tiene interés fundamentalmente en situaciones en las que se desea máxima sencillez y no se necesita reducir el número de direcciones públicas, por ejemplo se quiere conectar a Internet una red clase C privada y se dispone para ello de una red clase C pública.

- NAT básico dinámico. La tabla de equivalencias de IP privada a IP pública se construye de forma dinámica, a medida que lo requieren los hosts. Las entradas caducan al terminar la conexión (TCP) o pasado un tiempo de inactividad (UDP), lo cual permite la reutilización de las direcciones. Los números de puerto no se modifican. El número de direcciones públicas puede

ser inferior al de direcciones privadas, pero ha de ser suficiente para el número de ordenadores que se quieren conectar simultáneamente al exterior. Permite un ahorro de direcciones frente al NAT estático, pero al no modificar el número de puerto es más fácil de implementar y tiene menos restricciones que el NAPT. En cierto modo es equivalente a utilizar DHCP con aisgnación dinámica de direcciones.

- NAPT (Network Address Port Translation) estático. En este caso la tabla de equivalencia se carga de forma estática en la configuración del equipo, pero las entradas incluyen no sólo la dirección IP sino también el número de puerto (TCP o UDP). El NAPT estático permite virtualizar servidores: por ejemplo es posible asignar el puerto 21 (servidor FTP) de una dirección pública del NAT al puerto 21 de un host en la red privada, y el puerto 80 (servidor Web) de la misma dirección a otro host de la red privada.

- NAPT (Network Address Port Translation) dinámico. En este caso la tabla de equivalencias se construye dinámicamente a medida que los hosts lo requieren, como en el NAT básico dinámico. Sin embargo, a diferencia de aquel las entradas de la tabla incluyen no solo la dirección IP, sino también el número de puerto. En el caso de TCP las entradas caducan al terminar la conexión. EN UDP, donde no hay conexión como tal, las entradas caducan pasado un tiempo de inactividad. En ambos casos es posible aprovechar una misma dirección IP pública para conectar al exterior diversos ordenadores simultáneamente, ya que se aprovecha el número de puerto UDP o TCP para multiplexar conexiones de hosts diferentes. Este NAT permite un aprovechamiento máximo de las direcciones públicas, pero tiene mayor complejidad de implementación.

Las cuatro modalidades antes descritas pueden coexistir en una misma red, por ejemplo utilizando NAT o NAPT estático para los servidores que deban ser accesibles desde el exterior y NAT o NAPT dinámico para el resto de los hosts.

Las modificaciones que NAT introduce en el paquete IP son las siguientes:

- Cabecera IP: Además de modificar las direcciones de origen y/o destino el valor del campo checksum en la cabecera del datagrama cambia y por tanto ha de recalcularse.

- Cabecera de transporte (TCP/UDP): El campo checksum en la cabecera de transporte (TCP o UDP) ha de recalcularse, ya que la pseudocabecera incluye las direcciones IP de origen y destino. Además en el caso de NAPT se ha de modificar el valor del puerto de origen o destino.

- Mensajes ICMP: Los mensajes ICMP siempre incluyen en la parte de datos la cabecera a nivel de red y de transporte del paquete IP que originó el mensaje ICMP. El dispositivo NAT ha de localizar allí la dirección IP y modificarla. En el caso de hacer NAPT se ha de modificar también el número de puerto TCP/UDP que aparece en la cabecera embebida.

- Mensajes SNMP. Los mensajes de gestión, que notifican cambios en la situación de los diferentes dispositivos, suelen llevar en su parte de datos direcciones IP que el NAT debe localizar y modificar.

7.5.2 Limitaciones de NAT

A medida que aumenta el nivel de sofisticación aumenta como es lógico la complejidad del NAT. En el caso del NAT o NAPT dinámico el router ha de conservar una información de estado, ya que ha de mantener control de las conexiones existentes. Esta información de estado hace más difícil establecer un router de backup ya que esta información de estado ha de trasladarse a dicho router en caso de caída del principal. Además el router ha de tomar decisiones respecto a cuando termina una conexión para liberar la correspondiente entrada en su tabla (la de dirección IP en el caso de NAT o la de dirección IP y el puerto TCP/UDP en el caso de NAPT). En TCP es fácil detectar el cierre de la conexión a través de los segmentos que tienen el bit FIN puesto, pero en UDP no existe un procedimiento explícito de desconexión, por lo que en estos casos normalmente se fija un tiempo de inactividad a partir del cual una

conexión se considera inexistente, o bien se espera a tener que liberar recursos y entonces se cierra la conexión que lleva más tiempo inactiva.

Incluso en el caso más sencillo del NAT básico estático se plantean problemas de difícil solución que hacen que determinadas aplicaciones no funcionen a través de estos dispositivos. Algunos ejemplos de situaciones en las que el NAT tiene dificultades o no funciona son las siguientes:

- Protocolo FTP. En el momento de establecer una conexión FTP los hosts intercambian una serie de mensajes que contienen entre otra información sus direcciones IP. Lógicamente estas direcciones han de modificarse. El problema en este caso es que las direcciones no están codificadas en el habitual formato binario de 32 bits sino como caracteres ASCII. Esto plantea problemas cuando el número de caracteres de la dirección vieja y la nueva no coinicide. Cuando la nueva dirección es menor, por ejemplo convertir de “200.200.200.1” (13 caracteres) a “192.168.1.1” (11 caracteres) el proceso de traducción puede optar por rellenar el campo con ceros (“192.168.001.1”) para mantener constante el número de octetos. Pero cuando la traducción se realiza en sentido inverso es preciso aumentar en dos octetos la longitud del segmento TCP correspondiente. Como consecuencia de ello el NAT a partir de ese momento ha de incrementar en dos octetos los contadores de bytes que aparecen en los campos número de secuencia y número de ACK para esa conexión TCP; esto supone que el NAT ha de mantener *información de estado* para esa conexión, mientras exista. Incluso podría darse el caso de que el aumento en dos octetos del segmento provocara la fragmentación del datagrama correspondiente.

- En general cualquier protocolo del nivel de aplicación que incluya en la parte de datos información sobre direcciones IP o números de puerto TCP/UDP supone un reto para un dispositivo que hace NAT, ya que la detección y modificación de dichas direcciones requiere que el NAT analice y modifique información que se encuentra en la parte de datos del paquete IP, muchas veces perteneciente al nivel de aplicación. Algunos ejemplos de esto son el protocolo de la ITU-T H.323 utilizado en videoconferencia y en telefonía por Internet (voz sobre IP); también se encuentran en este caso los juegos interactivos de Internet, las aplicaciones tipo Napster, etc. Normalmente el funcionamiento de estas aplicaciones a través de un NAT sólo se consigue cuando el NAT está especialmente preparado para ello.

- La comunicación de aplicaciones NetBIOS sobre TCP/IP tiene problemas parecidos a las aplicaciones del apartado anterior: la información sobre direcciones IP aparece de forma no consistente y con desplazamientos variables en la información intercambiada por los hosts, lo cual hace difícil que los NAT las modifiquen; como consecuencia de ello no es posible utilizar TCP para transportar NetBIOS cuando se atraviesa un NAT.

- IPSec solo puede utilizarse de forma limitada a través de un NAT. Esto se debe a que IPSec incorpora una cabecera de autentificación (AH, Autentification Header) que permite al receptor detectar si el paquete IP ha sido modificado en ruta. Evidentemente la cabecera AH no incluye el campo TTL ni el checksum, ya que se sabe que estos campos cambian de valor durante el camino de un datagrama, pero si incluyen las direcciones IP de origen y destino. Como estas direcciones se modifican en el NAT el receptor cuando comprueba la cabecera AH detecta que el datagrama ha sido alterado y lo descarta. Cuando se hace NAT sólo es posible utilizar IPSec en modo túnel y sólo si el túnel se realiza en el mismo dispositivo que hace el NAT, o después. En estos casos la cabecera AH se calcula después de haber realizado el cambio de direcciones.

A pesar de todas las limitaciones reseñadas NAT resulta un mecanismo muy útil en un amplio abanico de situaciones, por lo que es seguro que se seguirá utilizando NAT en Internet durante bastantes años, al menos hasta que se generalice el uso de IPv6. Con ese nuevo protocolo y su abundante espacio de direcciones parece difícil concebir un escenario en el que tenga sentido utilizar NAT.

Los aspectos más relevantes de NAT se discuten en los RFC 1631 y 2663.

7.6 EJERCICIOS

1. Indique si es verdadera o falsa cada una de las siguientes afirmaciones:

a) Para transmitir simultáneamente paquetes de mas de un protocolo a nivel de red (p. ej. IP e IPX) por un circuito ATM se emplea normalmente encapsulado IEEE 802.2.

b) La principal finalidad de un cortafuego o firewall es conectar una red local con el exterior de forma que múltiples usuarios puedan compartir una misma dirección IP.

c) La técnica denominada 'buques en la noche' (ships in the night, SIN) requiere el uso de un protocolo de routing distinto para cada protocolo soportado.

d) El encapsulado es una técnica que permite aumentar la proporción de información útil (datos) que se envía por una línea determinada.

2. Una empresa posee dos oficinas, en cada una de las cuales hay instalada una red local. En cada red local funcionan diversos protocolos. Se quiere unir ambas oficinas, estimándose en 128 Kb/s el ancho de banda necesario. La empresa puede disponer de enlaces a 64 Kb/s a un precio especialmente bajo, por lo que utilizará dos de estas líneas. En cuanto a los equipos a utilizar se plantean cuatro alternativas:

a) Cuatro puentes transparentes sin spanning tree
b) Cuatro puentes transparentes con spanning tree
c) Cuatro routers multiprotocolo con routing estático
d) Cuatro routers multiprotocolo con routing dinámico

En todos los casos los equipos tienen una interfaz serie (para conectar a la línea de 64 Kb/s) y una interfaz ethernet.

Serían válidas las cuatro? Discuta las ventajas e inconvenientes de cada una.

7.7 SOLUCIONES

S1.-

a) **Verdadera**.

b) **Falsa**. La principal finalidad es proteger la red de ataques externos.

c) **Verdadera**. El nombre pretende reflejar precisamente el hecho de que los paquetes originados por cada protocolo de routing viajan por la misma red física sin verse mutuamente, y normalmente sin interferirse unos con otros.

d) **Falsa**. El encapsulado, al añadir mas información de control, reduce en la práctica la proporción de información útil que se envía por la línea.

S2.-

a) Puentes transparentes sin spanning tree: solo se podrá utilizar una línea, teniendo que dejar la otra desconectada ya que de lo contrario se producirán bucles que bloquearían la red. La pareja de puentes no utilizada se podría tener preparada para conectar manualmente la segunda línea en caso de avería en la primera conexión.

b) Puentes transparentes con spanning tree: se podrán conectar las dos líneas y los cuatro equipos, pero de forma automática se desactivará uno de los caminos. El spanning tree permitirá que la segunda línea entre en funcionamiento de forma automática en caso de problemas.

c) Routers multiprotocolo con routing estático: al disponer de dos routers en cada lado se podría repartir el rango de direcciones en dos subredes de forma que cada subred se encamine por una línea distinta. Esto permitiría repartir el tráfico entre las dos líneas de forma estática, cosa que nunca es óptima pero al menos aprovecha las dos líneas en alguna medida. En caso de avería en una de las líneas las subredes correspondientes quedarían aisladas entretanto no se hiciera una reconfiguración manual de los equipos.

d) Routers multiprotocolo con routing dinámico: en este caso el protocolo de routing efectuará un reparto óptimo del tráfico entre las dos líneas; además en caso de avería de una de las líneas el tráfico será reencaminado automáticamente por la línea disponible.

Dado que se requiere disponer de 128 Kb/s entre las oficinas las únicas soluciones válidas serían la c) y la d), siendo esta última la mas robusta y la que presenta mejor aprovechamiento de la infraestructura.

8 EL NIVEL DE APLICACIÓN EN INTERNET

8.1 Servidor de Nombres (DNS domain name server)

8.1.1 Introducción

Para poder comunicarse con un ordenador necesitamos saber su dirección IP. Dado que las direcciones IP son difíciles de memorizar o recordar necesitamos asignar a las direcciones nombres significativos, por ejemplo, para poder ver la página web de la universidad de Valencia, www.uv.es, necesitaríamos saber la dirección IP del servidor web de esta, que en este caso es *147.156.1.46*, lo mismo para cualquier otra página web que se nos ocurra, www.yahoo.com, www.sun.com, robotica.uv.es, etc. Este método es inviable y engorroso.

En los comienzos de Internet se utilizaba una única tabla centralizada de traducción de nombres a direcciones. En los 70 el ARPANET estaba formada por unos cientos de máquinas y un sólo archivo, *HOSTS.TXT*[1],(que se conserva como reminiscencia en la mayoría de sistemas operativos actuales, permitiendo trabajar a una máquina en red sin un servidor DNS) que contenía toda la información que se necesitaba sobre esas máquinas. El centro de información de red del Departamento de defensa disponía de la versión maestra de la tabla y otros sistemas realizaban una copia regularmente.

Con el paso del tiempo y cuando Internet, con los protocolos TCP/IP "explotó", este método, como podemos imaginar, presento serios inconvenientes, entre los que podemos destacar los siguientes:

- El tráfico y la carga de red para la máquina que contenía las tablas que hacía posible el mapeo era desbordante.
- La consistencia del archivo era muy difícil de mantener, cuando el *HOST.TXT* llegaba a una maquina muy lejana era ya obsoleto.
- No se podía garantizar la no duplicidad de nombres, dado que mantener una administración central en una red Internacional era algo muy complicado.
- El método era claramente no Escalable.

Conclusión, el método se convirtió en obsoleto y muy ineficiente a medida que aumentaron el número de máquinas. Es entonces cuando surgió un nuevo sistema de resolución de nombres, *DNS (Domain Name System)*, que solucionó los problemas anteriores.

8.1.2 DNS, el Sistema de Nombres de Dominio

El Sistema de Nombres de Dominio básicamente es un esquema que permite asignar nombres significativos de alto nivel a grandes conjuntos de máquinas y direcciones IP. El esquema es jerárquico y basado en dominio utilizando una Base de Datos Distribuida para implementarlo. Se utiliza un mecanismo Cliente/Servidor, donde unos programas llamados servidores de nombres contienen información acerca de un segmento de la base de datos y la ponen a disposición de los clientes, llamados *resolvers*.

Los llamados *resolvers* son rutinas de biblioteca que crean preguntas y las envían en forma de paquete UDP a un servidor DNS local. Este devuelve al resolver la dirección IP correspondiente al nombre solicitado en la pregunta. El motivo de utilizar UDP como capa de transporte, se justifica porque estas aplicaciones se basan en consulta y contestación, de forma atomatizada, sin crear ningún tipo de flujo ni conexión. Además, como es realizada a nivel local, la probabilidad de error es muy baja.

[1] Actualmente en los sistemas operativos UNIX, mantienen un fichero con características similares en /etc/hosts y en Windows2k /winnt/system32/drivers/etc/hosts

El servicio DNS los podemos situar en la arquitectura de TCP/IP como protocolo de aplicación, tanto sobre UDP como TCP en el puerto 53 de la capa de transporte[2]. Véase la siguiente figura:

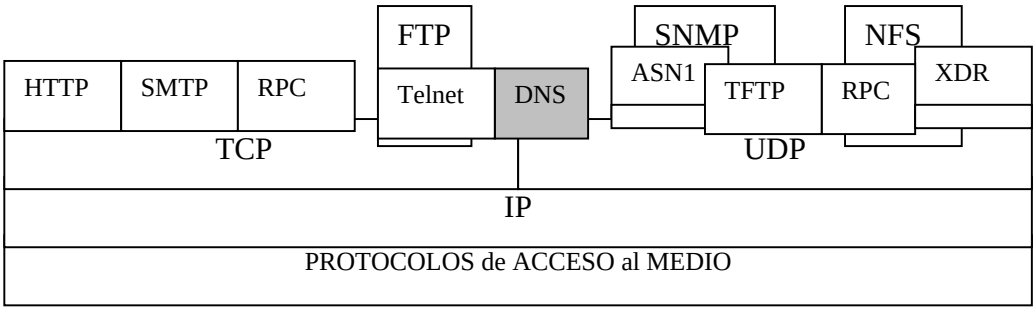


Figura 8.1: Modelo TCP/IP y la aplicación DNS

Gracias a la estructura y funcionamiento del DNS se eliminaron los problemas inherentes al sistema de archivo plano. A continuación describimos brevemente el porqué:

- *Carga de la red y de los hosts:* este problema ya no existe debido a que la información esta distribuida por toda la red, al tratarse de una base de datos distribuida.
- *Duplicidad de Nombres:* el problema se elimina debido a la existencia de dominios controlados por un único administrador. Puede haber nombres iguales pero en dominios diferentes.
- *Consistencia de la Información:* ahora la información que esta distribuida es actualizada automáticamente sin intervención de ningún administrador.

8.1.3 Nombres de dominio, delegación de autoridad y registro de recursos

El Sistema de Nombres de Dominio lo podemos desglosar en dos conceptos independientes:

- el primero, especifica la sintaxis del nombre y las reglas para delegar la autoridad respecto al nombre
- el segundo especifica la implementación en ordenadores de la base de datos distribuida que transforma de una manera eficiente los nombres a direcciones IP.

Como veremos, se puede considerar el DNS como una forma de inventariar todos los recursos de una red, donde los recursos puede ser desde máquinas, servidores de correo, direcciones postales, etc información propia de la administración de una red.

Nombres de Dominio

Como dijimos anteriormente el DNS se basa en un esquema jerárquico de nombres denominado *nombre de dominio.* Un *nombre de dominio* consiste en una secuencia de etiquetas separadas por un punto. Por ejemplo, *robotica.uv.es*, contiene tres etiquetas, el dominio de nivel inferior es *robotica.uv.es,* el segundo nivel de dominio es *uv.es*, y el nivel superior es *es.* Como jerarquía que es podemos apreciar que el dominio de nivel local corresponde con la primera etiqueta, y el de nivel superior la última etiqueta.

[2] . UDP es utilizado para resolver preguntas locales de clientes y TCP es utilizado como veremos para transferencias de zonas entre servidores DNS secundarios

Como veremos posteriormente, en la practica un dominio es un índice en la base de datos de DNS. Un dominio puede ser una máquina o puede ser un nodo del cual pueden partir otros dominios (o ambas cosas a la vez).

En Internet se ha decidido particionar el nivel superior en dos tipos de dominios, los geográficos y los genéricos. Con los primeros se pretende dividir los recursos a direccionar por país (por ej, *uv.es*), también son conocidos como ISO3166 . Con los segundos se pretende un división en función del tipo de organización. A continuación podemos ver una tabla con algunos de los dominios genéricos más utilizados, que corresponde a la tabla TLD (Top Domain Level):

Nombre de Dominio	Significado
COM	Organizaciones comercianles, Microsoft.com, ibm.com
EDU	Universidades, Instituciones academicas,...
GOV	Instituciones Gubernamentales
MIL	Organizaciones militares
ORG	Organizaciones no comerciales
NET	Grupos relacionados con la Red
INT	Organizaciones Internacionales

Tabla 8.1: Dominio genéricos. Tabla Top Domain Level (TLD)

Todos los dominios en Internet pueden representarse mediante un árbol. Las hojas del árbol serían los dominios que ya no contiene más dominios (subdominios, aunque este término no esta definido es bastante utilizado). Entonces, cada dominio en Internet estaría definido por la trayectoria hacia arriba desde él a la raíz (esta está vacía). El árbol tendría la siguiente forma:

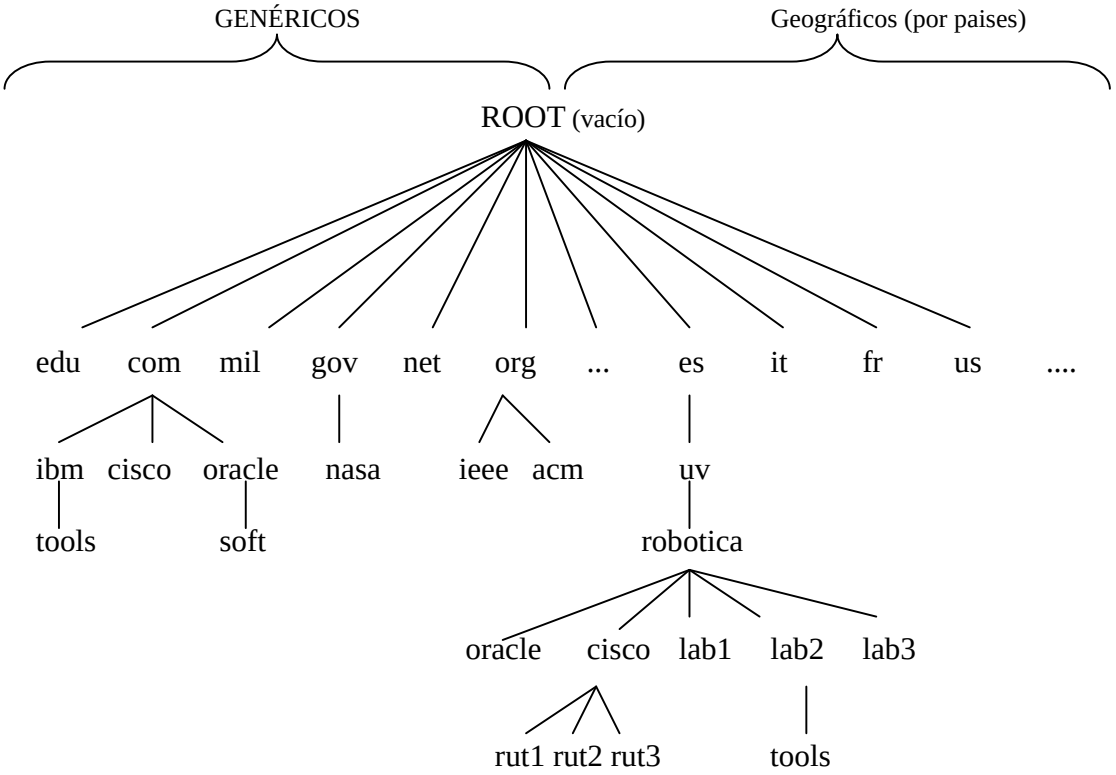


Figura 8.2: árbol de la jerarquía de nombres de Internet

Como puede verse en el árbol al ser jerárquico no hay conflicto cuando dos nombres son iguales, mientras pertenezcan a dominios distintos. En la figura 8.2, por ejemplo, esto se da con *oracle.com* y *oracle.robotica.uv.es* .

Los nombres pueden ser texto ASCII de hasta 63 caracteres de longitud, mientras que los nombres de trayectoria completa (desde la hoja hasta la raíz) no debe sobrepasar los 255 caracteres.

Delegación de la Autoridad

La organización que posee un Nombre de Dominio, como por e.j. *uv.es*, es responsable del funcionamiento y mantenimiento de los servidores de nombres que traducen sus propios nombres a direcciones. Se crea entonces un administrador local quien bajo su responsabilidad introduce altas, bajas y cambios de forma rápida y precisa. E incluso puede delegar parte de los dominios que caen bajo su responsabilidad en otro administrador.

Para ver más claramente la relación entre la jerarquía de nombres y la delegación de la autoridad para los nombres veamos el siguiente ejemplo (véase la figura 8.2): una máquina del Instituto de Robótica de la Universidad de Valencia tiene el nombre de dominio oficial *oracle.robotica.uv.es* . El nombre de la máquina fue registrado por el administrador local de la red del mencionado Instituto. El Administrador de esta red local había obtenido previamente autorización para la administración del subdominio *robotica.uv.es* de una Autoridad de la red Universitaria, es decir, del Administrador de la red *uv.es.* Este a su vez había obtenido permiso de la Autoridad que Administra el dominio geográfico *.es* . Quién a su vez había obtenido el permiso para administrar el dominio superior geográfico de la Autoridad de Internet.

Para aclarar más si cabe el concepto de delegación de Autoridad, cuando alguien (Administrador ,Entidad, Grupo,..) recibe la responsabilidad de Administrar un dominio, a partir de ese dominio es él, el único responsable de administrar como mejor crea conveniente todos los subdominios que caen bajo el nodo que él administra. Consecuencia de esto lo podemos ver en el dominio superior *us* .La autoridad que lo administra a creído conveniente dividirlo en un segundo nivel por estado. De esta forma quién quiera conectarse bajo el dominio de EE.UU, *us*, *deberá hacerlo obligatoriamente bajo un dominio de segundo nivel.* Por ejemplo, imaginemos la empresa *ComputerUncleTom* situada en *Virgina* en EE.UU. La empresa desea conectarse a internet bajo el dominio superior *us*. Ésta no podría hacerlo directamente, debería conectarse como dominio de segundo nivel, es decir, *computeruncletom.va.us*

Nombres absolutos y relativos

En la configuración de un servidor DNS como veremos a continuación, el cliente que realiza las consultas al servidor, recibe contestaciones de carácter absoluto. Esto quiere decir que se detalla el nombre completo a partir de la dirección raíz, por ejemplo glup.irobot.uv.es.

Sin embargo, para configurar un servidor, introducir todo el nombre completo en los registros que veremos a continuación, puede ser un poco tedioso y por eso la utilización de nombres relativos. Los nombres relativos son nombres que completan su nombre en función del dominio del cual están registrados. Por ejemplo, en el dominio uv.es, la máquina con nombre relativo “glup.irobot”, tomará como nombre absoluto “glup.irobot.uv.es”.

Por tanto, el nombre absoluto no requiere de ninguna referencia a un dominio, dado que es un nombre completo. Para indicar que un nombre es absoluto, terminará su nombre con “.”, en caso contrario, al nombre relativo que termina sin “.”se le añadirá la coletilla del dominio.

Esta distinción es importante y hay que tenerla en cuenta al configurar los registros del DNS, dado que si algún registro por descuido es dejado sin “.”, el DNS añadirá su dominio. Por ejemplo, en el caso de tener un registro con valor “glup.uv.es” sin “.”en el valor de un registro, el DNS cuando se le consulte dicho registro devolverá “glup.uv.es.uv.es”.

Registro de Recursos

Cuando un *resolvedor* da un nombre al DNS para que este le responda, el nombre dado puede transformarse en más de un aspecto en el sistema de dominio. Es decir, la respuesta no tiene porque se únicamente una dirección IP, sino que pueden ser muchos tipos de recursos, como por ejemplo, una dirección que sólo acepta correo electrónico. Por tanto, cuando el cliente envía al DNS el nombre del objeto o recurso que busca indica el tipo de objeto que busca (también existe la posibilidad de pedir todos los recursos asociados con el nombre).

Los datos del DNS se almacenan como una serie de entradas de texto, llamadas *Registros de Recursos* (RR). Un RR esta formado por cinco tuplas y tiene el siguiente formato:

[Nombre_dominio] [TTL] [Clase] Tipo Dato_Registro(Valor)

Adicionalmente se pueden añadir comentarios en la misma línea incluyendo al principio del comentario *;* .

- El Primer campo, *Nombre_dominio*, indica el nombre de dominio. Puede haber más de un registro por dominio. Este campo a veces puede omitirse, en ese caso siempre se toma por defecto el último nombre de domino indicado con anterioridad.

- El campo *TTL*, se refiere al tiempo de vida, e indica la estabilidad del registro, es decir, cuanto tiempo debe guardarse en caché después de almacenarse.

- El campo *Clase* se refiere al tipo de información. Actualmente sólo se utiliza *IN*, para información de Internet. Este campo si se omite, al igual que el campo *Nombre* se toma el último valor indicado con anterioridad. A veces, el orden de los campos *TTL* y *Clase* pueden intercambiarse, no dando a confusión dado que TTL es numérico.

El campo *Tipo* indica el tipo de registro, puede contener algunos de los siguientes valores (los más importantes):

Tipo de Registro	Descripción
SOA	Inicio de autoridad, Start Of Authority), identifica el dominio o la zona y se fijan una serie de parámetros para esta zona, información utilizada por los DNS secundarios que trabajen bajo esta SOA.
NS name server	El nombre de dominio se hace corresponder con el nombre de una computadora de confianza para el dominio, es decir, un servidor de nombres con nombre absoluto.
A address	Dirección IP de un host. Si este tiene varias direcciones IP, multihomed, habrá un registro diferente por cada una de ellas.
CNAME	Es un alias que se corresponde con el nombre canónico verdadero, con nombre absoluto.
MX	Se trata de un intercambiador de correo (Mail eXchanger), es decir, un dominio dispuesto a *aceptar* solo correo electrónico con nombre absoluto.
TXT	Texto, es una forma añadir comentarios a la Base de Datos. Por Ej., para dar la dirección posta del dominio miempresa.com .

PTR	Apuntador, hace corresponder una dirección IP con el nombre de un sistema. Usado en archivos dirección-nombre.
HINFO	Información del Host, tipo y modelo de computadora.
MINFO	Información de Buzón o lista de correo.
WKS	Servicios públicos (Well-Known Services). Puede listar los servicios de las aplicaciones disponibles en el ordenador.

Tabla 8.2: Descripción del campo TIPO en los registros de recursos del DNS

Volviendo a la estructura general del RR el campo valor será un número o texto ascii dependiendo del tipo de registro. También decir, que el orden en la tabla de los RR no es significativo.

Ahora presentamos un ejemplo (ficticio) del tipo de información que podríamos encontrar en la base de datos del dominio de una empresa conectada a Internet, en este caso *computeruncletom.va.us* :

```
;Fichero Configuración
;Datos autorizados para computeruncletom.va.us
;
computeruncletom.va.us. 86400 IN    SOA   (1; Serie
                                    28800 ;Refrescar cada 8 horas
                                    3600  ;Reintentar cada hora
                                    604800 ;Expirar al cabo 1 semana
                                    86400 ) ;El defecto es un dia
;información de la empresa
computeruncletom.va.us. 86400 IN    TXT   "Empresa de Informatica"
computeruncletom.va.us. 86400 IN    TXT   "Virginia, EE.UU"

;Servidores de correo
computeruncletom.va.us. 86400 IN    MX    1 correo.computeruncletom.va.us.
computeruncletom.va.us. 86400 IN    MX    2 postal.computeruncletom.va.us.

;Nuestros servidores de nombres
computeruncletom.va.us. 86400 IN    NS    mesa.computeruncletom.va.us.
computeruncletom.va.us. 86400 IN    NS    silla.computeruncletom.va.us.

;Direcciones de hosts
correo.computeruncletom.va.us.      86400 IN    A     192.253.253.2
postal.computeruncletom.va.us.      86400 IN    A     192.253.253.3
start.computeruncletom.va.us.       86400 IN    A     192.253.253.4
start.computeruncletom.va.us.       86400 IN    A     192.253.253.5
start.computeruncletom.va.us.       86400 IN   HINFO  Linuix SuSE 7.2
mesa.computeruncletom.va.us.        86400 IN    A     192.253.253.6
silla.computeruncletom.va.us.       86400 IN    A     192.253.253.7

;Alias
www.computeruncletom.va.us.   86400 IN CNAME start.computeruncletom.va.us.
ftp.computeruncletom.va.us.   86400 IN CNAME start.computeruncletom.va.us.
…                             …        …    …           …
…                             …        …    …           …
```

A continuación describimos el significado de cada línea (omitimos la líneas que son comentarios, ;…) :

- En la primera línea damos información del dominio que estamos administrando.
- En la línea siguiente damos información textual de nuestro dominio, en este caso a que se dedica la empresa y donde se localiza.

- En la cuarta y quinta línea estamos facilitando nuestros servidores de correo, es decir, máquinas que recibirán todo el correo dirigido a *quiensea@computeruncletom.va.us* Primero se intentará con la máquina de nombre *correo* y en caso de que esta fallará se intentaría con *postal.*[3]
- Las líneas seis y siete contienen información de nuestros servidores de nombres
- A continuación indicamos las direcciones IP de nuestras máquinas. En esta sección véase que damos información del sistema operativo de una nuestra máquinas, *start*, que además es mutihomed.
- Por último, indicamos los alias, en este caso www.computeruncletom.va.us y ftp.computeruncletom.va.us. Nótese que de esta forma nuestra máquina *start* será un servidor web y un servidor ftp.

En este ejemplo no hemos puesto las direcciones IP a usar para buscar los dominios de nivel superior, están son facilitadas por los servidores raíz.

Otro ejemplo, en este caso del servidor DNS de la Universitat de València, lo podemos ver en las siguientes configuraciones a través de diferentes registros de recusos:

```
uv.es.    SOA    gong.ci.uv.es. root.gong.ci.uv.es.
      (2002041601 -->número de serie
      86400 --> DNS secundario deben conectar cada 24 horas
      7200 -> si no lo consiguen deben reintentar cada 2 horas
      2592000 ->los datos en DNS secundario caducan a los 30 días
      172800)->tiempo de vida por defecto de los registros

uv.es.                          MX      10   sello.ci.uv.es.
uv.es.                          MX      20   postin.uv.es.

sello.ci                        MX      10   sello.ci.uv.es.
sello.ci                        MX      20   postin.uv.es.

sello.ci                        A       147.156.1.82
postin                          A       147.156.1.90

post                            CNAME   sello.ci.uv.es.

uv.es.      NS     sun.rediris.es. ;DNS secundario
            NS     gong.ci.uv.es.   ;DNS primario
            NS     chico.rediris.es.    ;DNS secundario
            NS     qfgate.quifis.uv.es. ;DNS secundario

localhost                       A       127.0.0.1

gong.ci                         A       147.156.1.1
gong.ci                         MX    10   gong.ci.uv.es.; Mail 1ª op
gong.ci                         MX    20   postin.uv.es.; Mail 2ª op
gong                            CNAME  gong.ci.uv.es.

sweb.informat                   A       147.156.16.46
```

[3] . Este procedimiento de intercambiadores de correo funciona siempre y cuando los agentes de transferencia de correo SMTP acepten intercambiadores. En el caso de tener nosotros configurado un intercambiador de correo, un intercambiador es la posibilidad de indicar de forma explícita a otra máquina que acepte de forma temporal el correo a nuestro agente SMTP en el caso que esté fuera de servicio. El agente externo hará la entrega la intercambiador configurado, independientemente de los usuarios y de sus cuentas. En el momento que nuestra máquina vuelva a estar operativa, nuestro agente SMTP solicitará al intercambiador que entregue todo el buzón que ha recibido para nosotros. Este proceso en los agentes SMTP se conoce como "relaying"

```
sweb.informat                   MX      10   sweb.informat.uv.es.
sweb.informat                   MX      20   postin.uv.es.
informat.informat               CNAME   sweb.informat.uv.es.
informatica                     CNAME   sweb.informat.uv.es.
sweb                            CNAME   sweb.informat.uv.es.

proxy                           A       147.156.1.18
proxy                           A       147.156.1.162

dkw.ci                          A       147.156.1.46
dkw.ci                          MX      10   dkw.ci.uv.es.
dkw.ci                          MX      20   postin.uv.es.
www                             CNAME   dkw.ci.uv.es.
infoserver                      CNAME   dkw.ci.uv.es.
```

8.1.4 Zonas de autoridad

Toda la información acerca de un espacio de dominio se guarda en una computadora que llamamos Servidor de Nombres. Esto es la implementación física del llamado programa servidor que antes hemos mencionado. Generalmente, los servidores tienen información completa acerca de una parte del espacio de dominio de nombres, que llamamos *zona de autoridad*. Más exactamente una *zona* es la porción del espacio de nombres de dominio de la que es responsable un determinado servidor DNS. La zona de autoridad de estos servidores abarca al menos un dominio y también pueden incluir subdominios, aunque a veces los servidores de un dominio puede delegar sus dominios en otros servidores.

La diferencia entre una *zona* y un *dominio* es que la primera contiene los nombres de dominio y datos que representan a un dominio y un *dominio* es un nombre a que agrupa a otras máquinas o dominios inferiores.

Veamos en la figura 8.3 a continuación, una forma de dividir el espacio de nombres de la figura 8.2:

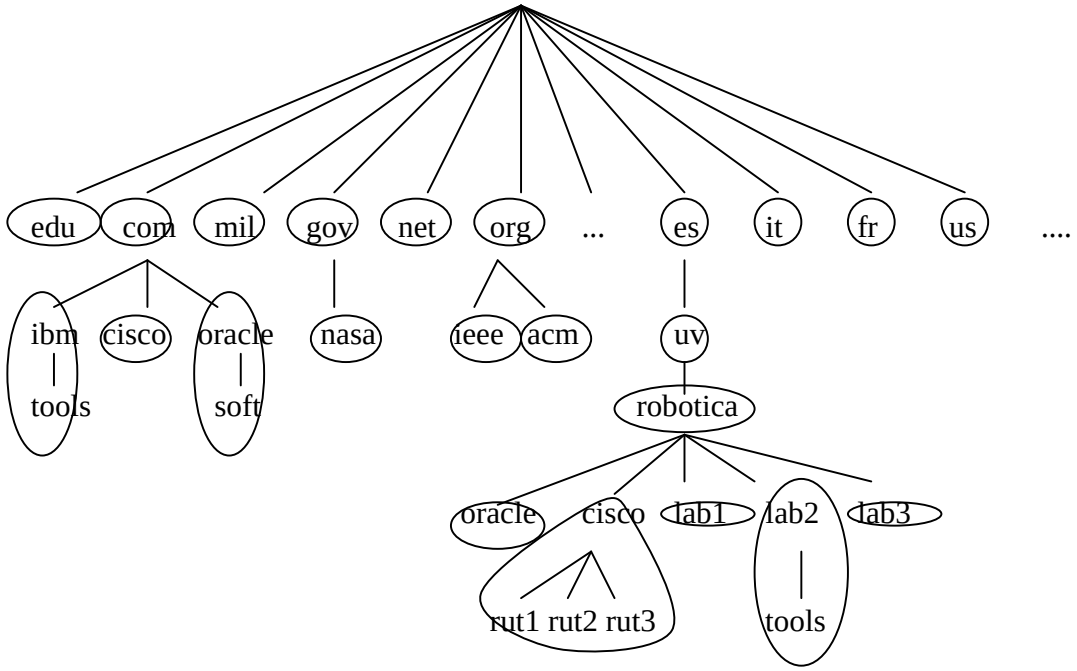


Fig. 8.3: División en zonas de parte del Espacio de Dominio de Nombres.

Cada zona tendrá asignada un servidor de nombres primario que obtiene su información de su base de datos local y uno o más servidores secundarios que obtienen su información del servidor de nombres primario. Recordemos que (delegación de autoridad) el lugar donde se colocan los limites de una zona dentro de una zona es responsabilidad del administrador de esa zona.

Hasta ahora hemos nombrado sólo dos tipos de servidores los Primarios y los Secundarios. Según la configuración del servidor podemos encontrarnos con cuatro tipos de servidores, desempañando cada uno una función distinta. Veamos en detalle cada uno de ellos en la siguiente tabla:

Tipo de Servidor	Descripción
Primarios (*Primary Name Servers)*	Almacenan la información de su zona en una base de datos local. Son responsables de mantener la información actualizada y cualquier cambio debe ser notificado a este servidor
Secundarios (*Secundary Name Servers*)	Son aquellos que obtienen los datos de su zona desde otro servidor que tenga autoridad para esa zona. El proceso de copia de la información se denomina *transferencia de zona.*
Maestros (*Master Name Servers)*	Los servidores maestros son los que transfieren las zonas a los servidores secundarios. Cuando un servidor secundario arranca busca un servidor maestro y realiza la transferencia de zona. Un servidor maestro para una zona puede ser a la vez un servidor primario o secundario de esa zona. Estos servidores extraen la información desde el servidor primario de la zona. Así se evita que los servidores secundarios sobrecargen al servidor primario con transferencias de zonas.
Locales (*Caching-only servers)*	Los servidores locales no tienen autoridad sobre ningún dominio: se limitan a contactar con otros servidores para resolver las peticiones de los clientes DNS. Estos servidores mantienen una *memoria caché* con las últimas preguntas contestadas. Cada vez que un cliente DNS le formula una pregunta, primero consulta en su memoria caché. Si encuentra la dirección IP solicitada, se la devuelve al cliente; si no, consulta a otros servidores, apunta la respuesta en su memoria caché y le comunica la respuesta al cliente.

Tabla. 8.3: Clasificación de servidores DNS.

8.1.5 El proceso de resolución de nombres de dominio

La resolución de un nombre de dominio es la traducción del nombre a su correspondiente dirección IP. En este proceso como dijimos anteriormente hay dos partes el cliente, *Resolver*, y el servidor, *el* DNS.

Un *Resolver* tiene las siguiente tareas:

- Interrogar al Servidor de nombres, DNS
- Interpretar las respuestas, que pueden ser RR o errores
- Y devolver información al programa que ha solicitado la resolución

Para este proceso de traducción los *Resolvers* pueden formular dos tipos de preguntas: recursivas e iterativas:

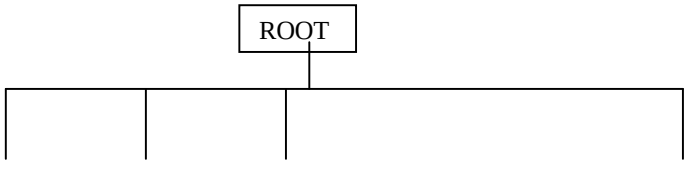
- Cuando un cliente formula una pregunta *recursiva* a un servidor DNS, éste debe intentar por todos los medios resolverla aunque para ello tenga que preguntar a otros servidores. Esta es la forma de interrogación más frecuente.
- Si el cliente formula una pregunta *iterativa* a un servidor DNS, este servidor devolverá o bien la dirección IP si la conoce o si no, la dirección de otro servidor que sea capaz de resolver el nombre. Esta forma de interrogación es poco utilizada y requiere que el cliente realice la consulta a otros DNS directamente.

Veamos un ejemplo: queremos visualizar en el browser la página web de la dirección www.uv.es. En primer lugar, el navegador tiene que resolver el nombre de dominio a una dirección IP. Después podrá comunicarse con la correspondiente dirección IP, abrir una conexión TCP con el servidor y mostrar en pantalla la página principal de la Universidad de Valencia. Los pasos que se van produciendo a lo largo del proceso de resolución los podemos resumir en los siguientes:

1. Nuestro ordenador (cliente DNS) formula una *pregunta recursiva* a nuestro servidor DNS local (generalmente el proveedor de Internet).
2. El servidor local es el responsable de resolver la pregunta, aunque para ello tenga que reenviar la pregunta a otros servidores. Si el usuario a solicitado información local, el servidor extrae la respuesta de su propia base de datos. Si la información solicitada es sobre un ordenador externo, el servidor comprueba primero su caché de consultas recientes, si la respuesta esta aquí entonces la devuelve (no es de confianza). Suponemos que no conoce la dirección IP asociada a www.uv.es; entonces formulará una *pregunta iterativa* al servidor del dominio raíz.
3. El servidor del dominio raíz no conoce la dirección IP solicitada, pero devuelve la dirección del servidor del dominio *es*.
4. El servidor local reenvía la pregunta iterativa al servidor del dominio *es*.
5. El servidor del dominio *es* tampoco conoce la dirección IP preguntada, aunque sí conoce la dirección del servidor del dominio *uv.es*, por lo que devuelve esta dirección.
6. El servidor local vuelve a reenvíar la pregunta iterativa al servidor del dominio *uv.es*.
7. El servidor del dominio *uv.es* conoce la dirección IP de www.uv.es y devuelve esta dirección al servidor local.
8. El servidor local por fin ha encontrado la respuesta y se la reenvía a nuestro ordenador. Al mismo tiempo la respuesta es almacenada en la caché para futuras referencias. El tiempo máximo que permanece la información almacenada en la caché se configura en los servidores de confianza.

8.1.6 Preguntas inversas

Los clientes DNS también pueden formular *preguntas inversas*, es decir, conocer el nombre de dominio dada una dirección IP. Para evitar una búsqueda exhaustiva por todo el espacio de nombres de dominio, se ha creado un dominio especial llamado *in-addr.arpa*. Cuando un cliente DNS desea conocer el nombre de dominio asociado a la dirección IP *w.x.y.z* formula una pregunta inversa a *z.y.x.w.in-addr.arpa*. La inversión de los bytes es necesaria debido a que los nombres de dominio son más genéricos por la derecha, al contrario que ocurre con las direcciones. Véase la figura siguiente:



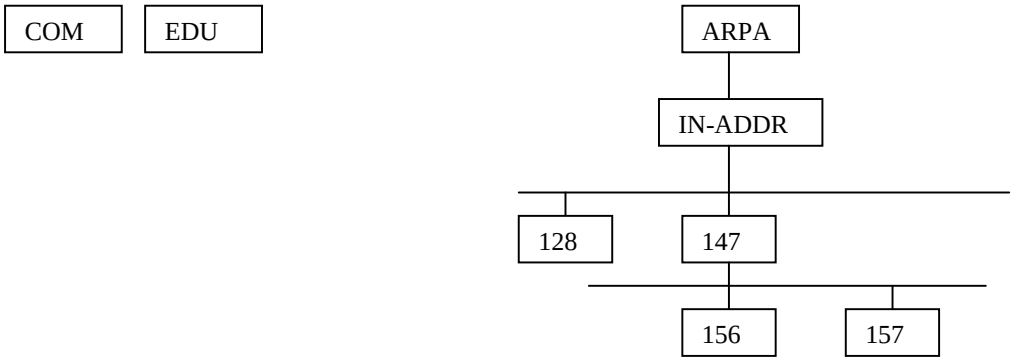


Figura.8.4 . Árbol para la resolución Inversa

La organización que posee una dirección de red es responsable de registrar todas sus traducciones de dirección a nombre en la base de datos del DNS. Esto se hace en una tabla que es independiente de las correspondencias entre nombre y direcciones. Como puede verse en el árbol de la fig. 8.4, el subárbol especial de dominio in-addr.arpa se creó para apuntar hacia todas esas tablas de red. Cuando se colocan las direcciones en el árbol, tiene sentido poner el primer número en la parte superior y bajar poco a poco.

8.1.7 Formato de los mensajes del DNS.

Los mensajes de consulta y respuesta intercambiados entre los clientes y los servidores del DNS tienen un formato muy sencillo. Un servidor añade la información requerida a la consulta original y la envía de vuelta. El formato general del mensaje se muestra a continuación:

Cabecera.
Consulta (o consultas).
(En la respuesta) RR de respuesta.
(En la respuesta) RR que identifican servidores con autorización.
(En la respuesta) RR con información adicional.

Tabla. 8.4: Formato general de un mensaje DNS.

La cabecera contiene los campos que se muestran en la tabla siguiente:

Campo	**Descripción**
ID	Identificador para hacer corresponder una respuesta con su petición
Parámetros	Consulta o respuesta. Consulta normal o inversa. En respuestas, si es de confianza. En respuestas, si está truncada. Recursivo o no. En respuestas, si la recursión está disponible.

	En respuestas, código de error.
Número de consultas.	Proporcionado en una consulta y en una respuesta.
Número de respuestas.	Proporcionado en una respuesta.
Número de registros de autoridad.	Proporcionado en una respuesta. La información de los registros de autoridad incluye los nombres de los servidores que contienen los datos de confianza.
Número de registros adicionales.	Proporcionado en una respuesta. La información incluye las direcciones de los servidores de confianza.

Tabla. 8.5: Descripción de los campos de cabecera de un mensaje DNS.

La sección de consulta contiene los campos que se muestran en la tabla siguiente. Normalmente, un mensaje contiene una única consulta, pero se permite concatenar varias peticiones en la sección de consulta.

Campo	Descripción
Nombre	Nombre de dominio o dirección de IP en el subárbol IN-ADDR.ARPA.
Tipo	Tipo de consulta, por ejemplo A o NS.
Clase	IN para Internet, se representa como 1.

Tabla. 8.6: Campos de la sección de consulta de un mensaje DNS.

Las secciones de respuesta, información de autoridad e información adicional se estructuran todas ellas de la misma forma. Consisten en una secuencia de registros de recurso que contienen los campos que se muestran en la tabla siguiente:

Campo	Descripción
Nombre	Nombre del nodo para este registro.
Tipo	Tipo de registro, como SOA o A, indicado por un código numérico.
Clase	IN, se representa como 1.
TTL	Tiempo de vida, un entero con signo de 32 bits que indica cuánto tiempo el registro puede permanecer en la caché.
RDLENGTH	Tamaño del campo de datos de recursos.
RDATA	La información, por ejemplo, para un registro de direcciones, es la dirección IP. Para un registro SOA, incluye información más extensa.

Tabla. 8.7: Campos de la respuesta de un mensaje DNS.

La sección de información de autoridad identifica los servidores de nombres de confianza para el dominio. La sección de información adicional proporciona información, como las direcciones IP de los servidores de nombres de confianza.

8.1.8 Ejemplo de consulta de direcciones: comandos nslookup (host o dig)

Un programa cliente capaz de consultar información sobre el sistema de nombres de dominio es parte estándar de los productos TCP/IP y se denomina *resolutor*. Normalmente, un resolutor trabaja discretamente en segundo plano y los usuarios no conocen su presencia, es decir que, toda consulta de un cliente DNS a su

servidor suele realizarla el programa que invocamos (telnet, ftp, mail, navegador web, etc.). Por ejemplo, un usuario solicita una conexión telnet a *glup.irobot.uv.es*, la aplicación telnet del usuario llama a un programa resolutor local que busca la dirección IP de ese ordenador (*147.156.160.55*) sin que el usuario tenga conciencia de ello.

Además de este trabajo en segundo plano, el usuario puede conectarse directamente al programa resolutor enviando consultas y resolviendo respuestas. Esto puede hacerse en WindowsNT/2000 y UNIX con el comando *nslookup*[4]. El comando "*nslookup*" en algunas distribuciones de Linux no está soportado y han creado el comando "*host*" para consulta de direcciones IP y "*dig*" para consulta de servidores DNS activos. Veamos el diálogo establecido con "*nslookup*", con ejecución equivalente a "*host*" y "*dig*":

1. Inmediatamente después de que el usuario teclee *nslookup*, el servidor local por defecto se identifica mostrando en pantalla su nombre y dirección. En este caso el nombre es *glup.irobot.uv.es* y la dirección *147.156.160.55*.

2. El usuario teclea el nombre del ordenador cuya dirección desea conocer.

3. La petición se envía al servidor.

4. Después de cada consulta, el servidor (*glup.irobot.uv.es*) se identifica y proporciona la respuesta.

5. Si el usuario ha solicitado información local, el servidor extrae la respuesta de su propia base de datos.

6. Si el usuario ha solicitado información sobre un ordenador externo, el servidor comprueba primero su caché de consultas recientes para ver si la información está disponible y, si no es así, interacciona con un servidor remoto de confianza para obtener la respuesta.

7. Cuando se recibe una respuesta del servidor remoto de confianza, la respuesta se almacena en la caché de disco del servidor local para futuras referencias y, a continuación, se envía al usuario solicitante. El tiempo de validez de la respuesta en la caché se configura en los servidores remotos de confianza y se envía como parte de la respuesta.

   Veamos un ejemplo de funcionamiento del comando nslookup.

>nslookup
Default Name Server: glup.irobot.uv.es Se muestra el nombre y dirección del servidor local.
Address: 147.156.160.55

>mozart.econom.uv.es. El usuario realiza una consulta local.
Name Server: glup.irobot.uv.es De nuevo nombre y dirección del servidor local.
Address: 147.156.160.55

Name: mozart.econom.uv.es El nombre de la consulta.
Address: 147.156.209.32 La respuesta.

>ftp.funet.fi. El usuario realiza una consulta remota.
Name Server: glup.irobot.uv.es De nuevo nombre y dirección del servidor local.
Address: 147.156.160.55

[4] "*Nslookup*" viene de *network search look up*, echa un vistazo y busca en la red

Name: ftp.funet.fi El nombre de la consulta.
Address: 128.214.248.6 La respuesta.

>ftp.funet.fi. El usuario repite la consulta remota.
Name Server: glup.irobot.uv.es De nuevo nombre y dirección del servidor local.
Address: 147.156.160.55

Non-authoritative answer: Está respuesta se obtuvo de la cache.
Name: ftp.funet.fi El nombre de la consulta.
Address: 128.214.248.6 La respuesta.

Una pregunta que puede hacerse es: ¿Porque el servidor continúa identificándose a sí mismo?. Esto es debido a que en una organización mantiene en funcionamiento dos o más servidores, ya que uno de ellos podría estar muy ocupado o incluso, fuera de servicio, por ejemplo, para mantenimiento. Si el resolutor no puede obtener una respuesta del primer sistema de la lista, lo intenta con el siguiente. Un administrador que utilice nslookup puede ver de inmediato que servidor está atendiendo la consulta**.**

Traducción de dirección a nombre.

El sistema de nombres de dominio es versátil y permite realizar traducciones de dirección a nombres. La forma de hacerlo utilizando nslookup puede parecer un poco extraña:

- Se determina que la consulta sea de tipo ptr.
- Se escribe la dirección en sentido inverso, seguida de .in-addr.arpa.

Por ejemplo:

>nslookup
>set type=ptr
>1.1.156.147.in-addr.arpa.

Name Server: glup.irobot.uv.es Nombre y dirección del servidor local.
Address: 147.156.160.55

1.1.156.147.in-addr.arpa name= gong.ci.uv.es

Esta singularidad tiene sentido cuando se conoce cómo están construidas las consultas globales inversas. La organización que posee una dirección de red es responsable de registrar todas sus traducciones de dirección a nombre en la base de datos del DNS. Esto se hace en una tabla que es independiente de las correspondencias entre nombre y direcciones.

El subárbol especial de dominio *in-addr.arpa* como hemos dicho anteriormente se creó para apuntar hacia todas esas tablas de red. Cuando se colocan las direcciones en el árbol, tiene sentido poner el primer número en la parte superior y bajar poco a poco. De está forma, todas las direcciones del tipo *147.x.x.x* están debajo del nodo *147.*

Si leemos las estiquetas del árbol utilizando el mismo convenio de abajo hacia arriba que utilizábamos para los nombres, la dirección aparece en sentido inverso, es decir, *1.1.156.147.in-addr.arpa*. Este tecnicismo puede ocultarse en la mayoría de servidores nslookup, pues introduciendo la pregunta:

>147.156.1.1
Name Server: glup.irobot.uv.es Nombre y dirección del servidor local.
Address: 147.156.160.55

Name: gong.ci.uv.es
Address: 147.156.1.1

Se obtiene el nombre del ordenador como si la pregunta hubiera sido la de la dirección conocido el nombre del ordenador.

Con el comando "host" resulta de forma similar. Dicho programa es */usr/bin/host*[5].

Así, si deseamos consultar la dirección IP del ordenador "glup" del Instituto de Robótica de la Universitat de València, basta con ejecutar:

host glup.irobot.uv.es

Y obtener como respuesta la siguiente línea.

glup.irobot.uv.es has address 147.156.222.65

De igual forma, podemos realizar una consulta inversa, esto es, introducir la dirección IP de un ordenador y obtener su nombre. Por ejemplo:

host 147.156.222.65

Devuelve como respuesta la línea:

65.222.156.147.in-addr.arpa domain name pointer glup.irobot.uv.es

Servidores de nombres de dominios locales y globales.

Si se posee una red TCP/IP independiente, se puede utilizar el software del DNS para crear una base de datos primaria de traducción de nombres y duplicarla en los lugares de la red que se estime conveniente. Todas las consultas de usuarios serán atendidas por los servidores propios de nombres. Pero si la red se conecta a Internet, los servidores de nombres necesitarán obtener información global. Esto se consigue inscribiendo la organización que quiere conectarse a Internet en una autoridad de registro, en este caso InterNIC, e identificando los nombres y direcciones de, al menos, dos servidores de nombres de dominio que vaya a utilizar. La autoridad de inscripción InterNIC añade esta información a su listado principal de servidores de nombres de dominio.

Este listado principal se duplica en muchos servidores raíz, que tienen un papel fundamental a la hora de procesar consultas remotas. Por ejemplo, supongamos que se envía una petición de traducción de nombre a dirección para *ftp.funet.fi* al servidor local de nombres de dominio:

- El servidor comprueba si *ftp.funet.fi* pertenece al dominio local.
- Si no es así, el servidor comprueba si el nombre se encuentra en su caché.

[5] En versiones anteriores de Linux y en otros sistemas UNIX, el programa que permite realizar las consultas recibe el nombre de nslookup. Este programa todavía se encuentra en la versión actual de Linux, pero se indica que esta obsoleto (deprecated).

- Si el nombre no está en la caché, el servidor envía una consulta a un servidor raíz.

- El servidor raíz devuelve el nombre y dirección de los servidores de nombres de dominio que contienen información sobre *funet.fi*.

Para ver la lista de servidores raíz, ejecutamos *nslookup* indicando que la consulta sea de tipo *ns*. Si introducimos <<.>> (que indica la raíz), nos devuelve los nombres y direcciones de varios servidores base.

>nslookup
>set type=ns
>.

Non-authoritative answer:
(root) nameserver = B.ROOT-SERVERS.NET
(root) nameserver = C.ROOT-SERVERS.NET
(root) nameserver = D.ROOT-SERVERS.NET
(root) nameserver = E.ROOT-SERVERS.NET
(root) nameserver = I.ROOT-SERVERS.NET
(root) nameserver = F.ROOT-SERVERS.NET
(root) nameserver = G.ROOT-SERVERS.NET
(root) nameserver = J.ROOT-SERVERS.NET
(root) nameserver = K.ROOT-SERVERS.NET
(root) nameserver = L.ROOT-SERVERS.NET
(root) nameserver = M.ROOT-SERVERS.NET
(root) nameserver = A.ROOT-SERVERS.NET
(root) nameserver = H.ROOT-SERVERS.NET

Authoritative answers can be found from:
(root) nameserver = B.ROOT-SERVERS.NET
(root) nameserver = C.ROOT-SERVERS.NET
(root) nameserver = D.ROOT-SERVERS.NET
(root) nameserver = E.ROOT-SERVERS.NET
(root) nameserver = I.ROOT-SERVERS.NET
(root) nameserver = F.ROOT-SERVERS.NET
(root) nameserver = G.ROOT-SERVERS.NET
(root) nameserver = J.ROOT-SERVERS.NET
(root) nameserver = K.ROOT-SERVERS.NET
(root) nameserver = L.ROOT-SERVERS.NET
(root) nameserver = M.ROOT-SERVERS.NET
(root) nameserver = A.ROOT-SERVERS.NET
(root) nameserver = H.ROOT-SERVERS.NET
B.ROOT-SERVERS.NET internet address = 128.9.0.107
C.ROOT-SERVERS.NET internet address = 192.33.4.12
D.ROOT-SERVERS.NET internet address = 128.8.10.90
E.ROOT-SERVERS.NET internet address = 192.203.230.10
I.ROOT-SERVERS.NET internet address = 192.36.148.17
F.ROOT-SERVERS.NET internet address = 192.5.5.241
G.ROOT-SERVERS.NET internet address = 192.112.36.4

Los servidores raíz proporcionan referencias directas a servidores de los dominios de segundo nivel, como COM, EDU, GOV, etc.

Podemos obtener la información sobre los servidores de nombres de una organización por medio de *nslookup* especificando *set type=ns* y el dominio de la organización:

>nslookup
>set type=ns
>uv.es

Name Server: glup.irobot.uv.es
Address: 147.156.160.55

uv.es nameserver = gong.ci.uv.es
uv.es nameserver = sun.rediris.es
uv.es nameserver = chico.rediris.es
uv.es nameserver = qfgate.quifis.uv.es
gong.ci.uv.es internet address = 147.156.1.1
sun.rediris.es internet address = 130.206.1.2
chico.rediris.es internet address = 130.206.1.3
qfgate.quifis.uv.es internet address = 147.156.122.64

Sin embargo, en lugar de que InterNIC mantenga actualizadas las listas de servidores pertenecientes a organizaciones de todo el mundo, cada país mantiene su servicio propio de inscripción y publica una lista de servidores de dominio en sus propios servidores principales.

Cuando se buscan servidores de nombres para un código de país, la base de datos raíz de InterNIC devuelve una lista de nombres y direcciones de servidores raíz de ese país. Así, si en el dialogo anterior podemos preguntar por la lista de servidores españoles.

>es.

Name Server: glup.irobot.uv.es
Address: 147.156.160.55

Non-authoritative answer:
es nameserver = SUN.REDIRIS.ES
es nameserver = CHICO.REDIRIS.ES
es nameserver = PRADES.CESCA.ES
es nameserver = NS.EUNET.ES
es nameserver = SUNIC.SUNET.SE
es nameserver = NS.EU.NET
es nameserver = RS0.INTERNIC.NET
es nameserver = NS.UU.NET

Authoritative answers can be found from:
es nameserver = SUN.REDIRIS.ES
es nameserver = CHICO.REDIRIS.ES

es nameserver = PRADES.CESCA.ES
es nameserver = NS.EUNET.ES
es nameserver = SUNIC.SUNET.SE
es nameserver = NS.EU.NET
es nameserver = RS0.INTERNIC.NET
es nameserver = NS.UU.NET
SUN.REDIRIS.ES internet address = 130.206.1.2
CHICO.REDIRIS.ES internet address = 130.206.1.3
PRADES.CESCA.ES internet address = 192.94.163.152
NS.EUNET.ES internet address = 193.127.1.11
SUNIC.SUNET.SE internet address = 192.36.125.2
NS.EU.NET internet address = 192.16.202.11
RS0.INTERNIC.NET internet address = 198.41.0.5
NS.UU.NET internet address = 137.39.1.3

De hecho, el sistema de nombres de dominio es muy flexible y permite una cadena larga de referencias. Un país puede dividirse en regiones de nombres y la raíz nacional podría apuntar a servidores raíz de cada región.

De forma similar, cualquier organización puede establecer un árbol raíz propio que apunte a servidores de nombres de dominio de confianza para partes de su dominio de nombres. En la práctica, se realizan relativamente pocas subdivisiones y, por tanto, los nombres se encuentran en muy pocos pasos.

Otro tipo de información, se podría extraer a través de la siguiente consulta:

>nslookup
>set q=any
>uv.es
>ls –d uv.es

lo que permite listar y enumerar la información completa del dominio uv.es.

8.1.9 Conexión a Internet y diseño de la base de datos de un servidor de nombres.

La conexión de un servidor de nombres de dominio particular a la base de datos mundial de Internet necesita:

1. Registrar uno o más bloques de direcciones IP y, opcionalmente, un número de sistema autónomo.
2. Asignar nombres y direcciones a los ordenadores propios.
3. Obtener la lista de servidores raíz que, en conjunto, cubran el servicio mundial.
4. Construir un servidor de nombres de dominio primario y, al menos, una copia secundaria.
5. Comprobar los servidores.
6. Pasar a la condición de operativo.
7. Registrar los nombres de dominio y servidores de la organización en los servicios de inscripción de la región.

El diseño de la base de datos puede realizarse estructurando toda la información como una base de datos única, si la organización es pequeña, o bien, en organizaciones grandes, distribuidas geográficamente, puede tener más sentido delegar la gestión del árbol de nombres de la organización en los administradores de esos lugares, e incluso, tener en funcionamiento servidores de nombre independientes en esas ubicaciones.

El árbol de nombres de una organización se compone de una o más zonas. Una zona es una parte contigua del árbol de nombres que se administra como una unidad. Por ejemplo, si observamos la figura 8.5 vemos que corresponde a una empresa con una central y dos sucursales. La base de datos raíz de Internet apuntará a los servidores de nombres de la oficina central. Estos servidores responderán directamente a peticiones de nombres que pertenezcan a su zona. Si se solicita un nombre de otra de las zonas, el servidor de la oficina central devolverá los nombres y direcciones de los servidores adecuados. Entonces, el servidor de nombres de dominio que originó la consulta enviará la petición al servidor correcto para esa zona.

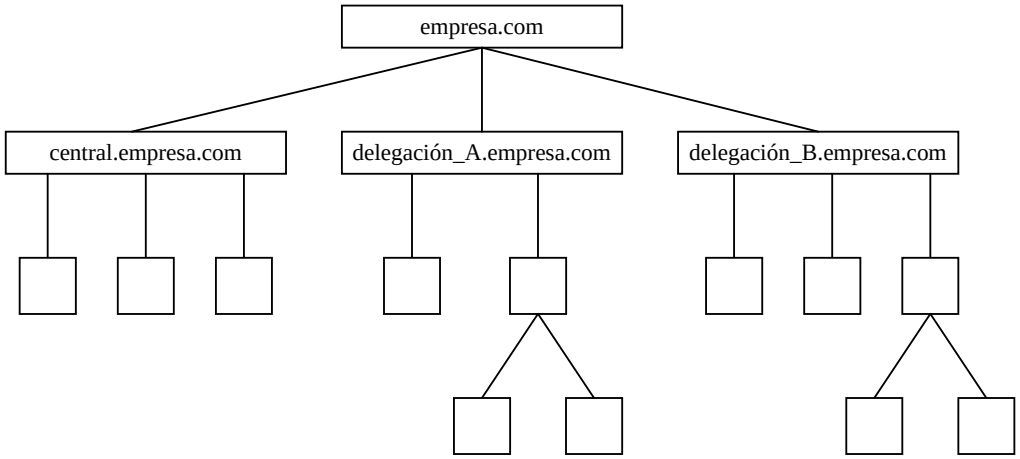


Figura 8.5: Ejemplo de árbol de nombres de una organización.

Para muchas organizaciones resulta más sencillo tener en funcionamiento un único grupo de servidores primarios y secundarios en su red interna, incluso aunque los datos se repartan para varias zonas. Resulta completamente aceptable la utilización de un servidor para varias zonas, e incluso, para varios dominios. Los datos de cada zona se almacenarán en un archivo diferente. Cada archivo puede actualizarlo un administrador diferente si es necesario.

Para poder acceder a una copia de la información sobre una o más zonas se instala un servidor secundario. Este obtiene su información del servidor primario de la zona mediante una transferencia de zona.

Se puede configurar un servidor secundario para que consiga información de varias zonas, de servidores primarios diferentes. De esta forma, un servidor puede actuar como secundario para varios primarios diferentes. Un servidor puede actuar, incluso, como primario para algunas zonas y secundario para otras. Un servidor de nombres necesita, para poder resolver los nombres:

- Una lista de los servidores raíz mundiales para conocer a dónde debe enviar las consultas externas. Se puede copiar un archivo de InterNIC que contiene esta lista del registro.

- Una lista de nombres con sus direcciones correspondientes.

- Una lista de direcciones con sus nombres correspondientes.

Como hemos visto, los datos del DNS se almacena como una serie de entradas de texto o registros de recursos con el formato:

[nombre] [TTL] [clase] Tipo-Registro Dato-Registro [; comentario]

8.1.10 Configuración de un cliente y un servidor de DNS estático.

Veremos a continuación como configurar, en UNIX, un ordenador como cliente de DNS así como la forma de realizar la configuración de un ordenador como servidor de DNS, tanto como servidor primario como servidor secundario de la red.

Aunque en este apartado sólo vamos a estudiar DNS estáticos, hay otras configuraciones que se llaman DNS dinámicos, que lo que hacen en vincular las direcciones IP asignadas por DHCP (*Dynamic Host Configuration Protocol*) con los nombres de forma dinámica.

Configuración de un cliente de DNS.

La existencia de un cliente DNS es necesaria en todo ordenador conectado a Internet, excepto, que deseemos realizar cualquier operación en Internet conociendo la dirección IP de los ordenadores y no su nombre, y además deseemos no utilizar servicios como el de los servidores virtuales de páginas Web.

El funcionamiento del cliente DNS se basa en tres ficheros, *hosts*, *host.conf* y *resolv.conf*, que pasamos a explicar de forma detallada a continuación.

El primero de estos ficheros, */etc/hosts*, puede considerarse como una herencia del antiguo fichero host.txt de ARPANET que hemos comentado con anterioridad. El fichero contiene la relación entre los nombres y las direcciones IP de todos los ordenadores necesarios en el arranque del ordenador, cuando este todavía no tiene activados sus interfaces de acceso a la red y por tanto no puede consultar los servidores de DNS. Aunque dicho fichero es de forma general una sola línea, pueden añadirse en el todas las líneas que se deseen conteniendo cada una la relación entre el nombre de un ordenador y su dirección IP. Un ejemplo del fichero es el siguiente:

127.0.0.1 glup localhost localhost.localdomain
147.156.222.65 glup glup.uv.es

La primera línea, 127.0.0.1 es siempre necesaria, pues hace referencia a *localhost*, que es el nombre por defecto del ordenador en el arranque, antes de activar sus dispositivos de acceso a la red.

El segundo de estos ficheros, */etc/host.conf*, indica el orden en que deben utilizarse las posibilidades disponibles para la resolución de los nombres, esto es, si primero debe mirarse el fichero */etc/hosts* y en caso de no encontrar allí la resolución del nombre debe consultarse al servidor de nombres o bien debe hacerse al contrario. La entrada típica de este fichero es:

order hosts,bind

Que indica que el orden es primero mirar el fichero local (*hosts*) y, en caso de no obtener respuesta, recurrir al servidor de nombres (*bind*).

El tercer fichero, */etc/resolv.conf*, indica el dominio al que pertenece este ordenador y las direcciones IP de los servidores de nombres a los que pueden efectuarse las consultas. Los parámetros de configuración más utilizados en este fichero son *domain*, *search* y *nameserver*.

El parámetro *domain* indica el dominio al que pertenece el ordenador, de forma que se puedan realizar consultas de nombres de ordenadores pertenecientes al dominio sin que sea necesario escribir el nombre

completo del ordenador, esto es, su nombre más el nombre del dominio. Así, la especificación con el parámetro *domain* del dominio *uv.es*, permite realizar una consulta de nombre de ordenador "post" y obtener la respuesta 147.156.1.112, sin necesidad de preguntar por su nombre completo, esto es "post.uv.es".

El segundo parámetro, *search*, es una generalización del anterior, de forma que es posible especificar hasta seis dominios donde la suma de la longitud de los nombres de todos ellos no exceda de 256 caracteres. Este parámetro, a pesar de representar una ventaja sobre el anterior, no suele usarse, o bien se usa con un solo dominio, el del ordenador servidor de DNS, pues en caso contrario obliga a los servidores a generar una gran cantidad de tráfico, realizando consultas a los servidores de todos los dominios que se especifiquen hasta encontrar la respuesta, si esta existe. Como consecuencia añadida del efecto anterior, la resolución de nombres puede volverse muy lenta, llegando los clientes a producir fallos por suceder un "timeout" en la espera de la respuesta.

El último parámetros, *nameserver*, indica las direcciones IP de los servidores de red que podemos consultar. Pueden existir tantas entradas *nameserver* como se deseen, consultándose los servidores en el orden en que se encuentran sus entradas. Por ello, deben ponerse en primer lugar aquellos servidores que se consideren más rápidos y disponibles, pues ello disminuye el tiempo de respuesta, tanto por la rapidez del servidor, como por el hecho de que si un servidor esta disponible de forma intermitente en el tiempo, toda pregunta que se le realice cuando no esta disponible deberá esperar el que suceda un "*timeout*" para enviar dicha pregunta al siguiente servidor de nombres.

Un ejemplo de fichero */etc/resolv.conf* es el siguiente:

```
domain          uv.es
nameserver              147.156.222.65
nameserver              147.156.1.1
nameserver              147.156.1.3
```

Configuración de un servidor primario de DNS.

Para configurar un servidor primario de DNS es necesario configurar cinco ficheros. Estos ficheros son:

/etc/named.boot
/var/named/db.127.0.0.local
/var/named/db.irobot
/var/named/db.147.156.160
/var/named/db.cache

El archivo */etc/named.boot* especifica el directorio y el nombre del dominio en el que se localizan los archivos del DNS. Este archivo es leído por el programa servidor de DNS, llamado *named* y usado para su configuración. Un ejemplo típico del fichero */etc/named.boot* para un servidor primario es:

```
;
; Archivo /etc/named.boot
;
domain          irobot.uv.es    ; Dominio del cual es servidor primario.
directory       /var/named      ; Directorio de trabajo.
;
primary         0.0.127.IN-ADDR.ARPA            db.127.0.0.local
primary         irobot.uv.es                    db.irobot
primary         160.156.147.IN-ADDR.ARPA        db.147.156.160
cache           .                               db.cache
```

El archivo *db.127.0.0.local* ,cuya localización siempre se encuentra referida al directorio especificado en el archivo archivo */etc/named.boot*, se utiliza para especificar la interfaz de loopback para el servidor primario de nombres de dominio. Un ejemplo típico es:

```
;
; Fichero primario inverso para la red local 127
;
@       IN      SOA     glup.irobot.uv.es.      root.glup.irobot.uv.es.
                        (
                        1       ; Serie
                        28800   ; Refrescar cada 8 horas
                        3600    ; Reintentar cada hora
                        604800  ; Expirar al cabo de una semana
                        86400   ; El defecto es un dia
                        )
        IN      NS      glup.irobot.uv.es.
1       IN      PTR     localhost.
```

El archivo *db.cache* proporciona información sobre los servidores raíz de DNS. Un ejemplo de este archivo es:

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;             submenu           InterNIC Registration Archives
;           file                named.root
;
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
;
;
; formerly NS.INTERNIC.NET
;
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
```

```
;
.                    3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.  3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                    3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.  3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                    3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.  3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                    3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.  3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                    3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.  3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                    3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.  3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
.                    3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.  3600000      A     198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                    3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.  3600000      A     193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                    3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.  3600000      A     198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                    3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.  3600000      A     202.12.27.33
; End of File
```

Los archivos *db.irobot* y *db.147.156.160* proporcionan información sobre los ordenadores y direcciones de IP, en formato directo en el archivo *db.irobot* y en formato inverso en el archivo *db.147.156.160*, asociadas al dominio *irobot.uv.es*. Un ejemplo típico de estos archivos es:

```
;
; Archivo /var/named/db.irobot
```

```
;

@       IN      SOA     glup.irobot.uv.es.      root.glup.irobot.uv.es.
                        (
                        1       ; Serie
                        28800   ; Refrescar cada 8 horas
                        3600    ; Reintentar cada hora
                        604800  ; Expirar al cabo de una semana
                        86400   ; El defecto es un dia
                        )
        IN      NS      glup.irobot.uv.es.
amparo  IN      A       147.156.160.54
glup    IN      A       147.156.160.55
carpanta IN     A       147.156.160.56


;
; Archivo /var/named/db.147.156.160
;
@       IN      SOA     glup.irobot.uv.es.      root.glup.irobot.uv.es.
                        (
                        1       ; Serie
                        28800   ; Refrescar cada 8 horas
                        3600    ; Reintentar cada hora
                        604800  ; Expirar al cabo de una semana
                        86400   ; El defecto es un dia
                        )
        IN      NS      glup.irobot.uv.es.
54      IN      PTR     amparo.irobot.uv.es.
55      IN      PTR     glup.irobot.uv.es.
56      IN      PTR     carpanta.irobot.uv.es.
```

Configuración de un servidor secundario de DNS.

La configuración de un servidor secundario de DNS es mucho más sencilla. Basta con configurar solo tres ficheros, estos fichero son:

/etc/named.boot
/var/named/db.127.0.0.local
/var/named/db.cache

El archivo */etc/named.boot* especifica el directorio y el nombre del dominio en el que se localizan los archivos del DNS. Este archivo es leído por el programa servidor de DNS, llamado *named* y usado para su configuración. Un ejemplo típico del fichero */etc/named.boot* para un servidor secundario es:

```
;
; type          domain          source file
;

domain          uv.es                   ; Dominio del cual es servidor secundario
directory       /var/named      ; Directorio de trabajo

primary         0.0.127.IN-ADDR.ARPA                    db.127.0.0.local
secondary       uv.es                   147.156.1.1     db.uv.bak
secondary       156.147.IN-ADDR.ARPA    147.156.1.1     db.147.156.bak
cache           .                                       db.cache
```

Si se observa, en las líneas *secondary* de la definición del servidor secundario de DNS, aparece la dirección IP (*147.156.1.1*) del servidor primario de DNS del cual se deben leer las tablas con las direcciones y nombres de los ordenadores del dominio del que se es servidor secundario, y guardarlas en los ficheros indicados (*db.uv.bak* y *db.147.156.bak*).

Los ficheros, *db.uv.bak* y *db.147.156.bak*, son copiados del servidor primario indicado, por lo cual no es necesario crearlos ni realizar ninguna acción sobre los mismos, pues son los administradores del servidor primario los que se encargan de introducir las altas, bajas o modificaciones que hayan podido producirse.

Los ficheros *db.127.0.0.local* y *db.cache* son iguales a los mostrados en el caso del servidor primario de DNS y no requieren por tanto una mayor explicación.

8.1.11 RFCs

RFC's principalesRFC 920: Domain Requierments
RFC 1101: DNS Enconding of Network Names and Other TypesRFC 1033 : Domain Adminstrators Operations Guide RFC 1034: Domain Names – Concepts and FacilitiesRFC 1035: Domain Names – Implementation and SpecificationRFC 1591: Domain Name System Structure and DelegationRFC 1183: New RR Types
También se está trabajando **en DNS y seguridad** para evitar el ataque conocido como DNS Spoofing o suplantación. RFC 2535.

8.2 Correo electrónico (Simple Mail Transfer Protocol: SMTP)

8.2.1 Introducción

El correo electrónico o email, ha existido desde hace más de dos décadas. Los primeros sistemas de correo electrónico simplemente consistían en protocolos de transferencia de archivos, con la convención de que la primera línea de cada mensaje (es decir del archivo) contenía la dirección del destinatario. A medida que pasó el tiempo, las limitaciones de este enfoque se hicieron obvias. Algunas eran:

- El envío de un mensaje a un grupo de gente era laborioso. Los administradores con frecuencia necesitaban esta facilidad para enviar un memorando a todos sus subordinados.

- Los mensajes no tenían estructura interna, dificultando el proceso del mismo por parte de la computadora. Por ejemplo, si un mensaje reenviado se incluía en el cuerpo de otro mensaje, la extracción de la parte reenviada del mensaje recibido era difícil.

- El remitente nunca sabía si el mensaje había llegado o no.

- Si alguien planeaba ausentarse por un viaje durante varias semanas y quería que el correo entrante fuera manejado por otra persona, esto no era fácil de arreglar.

- La interfaz de usuario estaba mal integrada al sistema de transmisión, requiriendo que el usuario editara un archivo, luego saliera del editor e invocara el programa de transferencia.

- No era posible crear y enviar mensajes que contuvieran una mezcla de texto, dibujos, facsímiles y voz.

A medida que se acumuló experiencia, se propusieron sistemas de correo electrónico más elaborados. Así, en 1982 se publicaron las propuestas de correo electrónico del ARPANET como RFC 821 (protocolo de transmisión) y RFC 822 (formato de mensaje). Dos años después, el CCITT elaboró su recomendación X.400, que luego se tomó como base para el MOTIS de OSI. En 1988, el CCITT modificó el X.400 para alinearlo con MOTIS. MOTIS debía ser la aplicación insignia del OSI, un sistema que era todo para todos.

Tras una década de competencia, los sistemas de correo basados en RFC 822 se usan ampliamente y se han convertido en los estándares de facto de Internet y los basados en el X.400 han desaparecido. La razón del éxito del RFC 822 no es que sea mejor que X.400, sino que X.400 está mal diseñado y es tan complejo que nadie pudo implementarlo bien. Dada la opción entre un sistema de correo electrónico sencillo pero funcional basado en el RFC 822 y un sistema de correo electrónico X.400 supuestamente maravilloso pero no funcional, muchas organizaciones escogieron el primero.

8.2.2 Arquitectura de los sistemas de correo electrónico.

La idea clave de todos los sistemas modernos de correo electrónico es la distinción entre la envoltura y su contenido. La envoltura encapsula el mensaje o envolvente primitiva, contiene toda la información necesaria para transportar el mensaje, como la dirección de destino, prioridad y nivel de seguridad, etc., información que es independiente del mensaje mismo y está especificada en RFC821. El mensaje se encuentra en el interior de la envoltura y contiene dos partes: la cabecera y el cuerpo, ambos especificados en RFC822[6]. La

[6] En realidad se ha creado cierta confusión entre la envolvente primitiva y la propia cabecera del cuerpo del mensaje, con lo cual, son los propios agentes de transferencia MTA los que deciden cómo componer la envolvente y cabecera en base a la información que disponen entre ambas.

cabecera contiene información de control para la computadora. El cuerpo es por completo para el destinatario humano. Los actuales sistemas de correo electrónico desempeñan cinco funciones básicas:

1. La composición, que es el proceso de crear mensajes y respuestas. Aunque puede crearse un mensaje usando cualquier editor de texto para el cuerpo del mensaje, el sistema mismo puede proporcionar asistencia para el direccionamiento y los numerosos campos de cabecera que forman parte de cada mensaje. Por ejemplo, al contestar un mensaje, el sistema de correo electrónico puede extraer la dirección del emisor e insertarla automáticamente en el lugar adecuado del mensaje de respuesta.

2. La transferencia, que se refiere a mover mensajes del emisor al destinatario. En gran medida, esto requiere establecer una conexión con el destino o alguna máquina intermedia, enviar el mensaje y liberar la conexión. El sistema de correo electrónico debe hacer esto automáticamente, sin molestar al usuario.

3. La generación del informe, tiene que ver con indicar al remitente lo que ocurrió con el mensaje. Existen muchas aplicaciones en las que la confirmación de la entrega es importante y puede incluso tener implicación legal.

4. La visualización de los mensajes de entrada es necesaria para que la gente pueda leer su correo electrónico. A veces se requiere cierta conversión o debe invocarse un visor especial, por ejemplo si el mensaje es un archivo PostScript o voz digitalizada. También se intentan a veces conversiones y formatos de presentaciones sencillos.

5. La disposición es el paso final y se refiere a lo que el destinatario hace con el mensaje tras recibirlo. Las posibilidades incluyen tirarlo antes de leerlo, desecharlo tras leerlo, guardarlo, etc. También debe ser posible recuperar y releer mensajes guardados, reenviarlos o procesarlos de otras maneras.

Además de estos servicios básicos, la mayoría de los sistemas de correo electrónico proporcionan una gran variedad de características avanzadas como puede ser el reenvío automático de correo, la creación de buzones de correo independientes para almacenar el correo entrante, etc.

Un sistema de correo electrónico consiste generalmente en dos subsistemas distintos:

- Los agentes de transferencia de mensajes son los encargados de mover los mensajes del origen al destino. Por lo general son "daemons" del sistema operativo que trabajan en segundo plano y se encargan de recoger el correo electrónico, transferir el correo electrónico de una computadora a otra hasta su destino y almacenarlo una vez llegado al destino. Los agentes de transferencia del mensaje usan la envoltura para calcular la ruta que debe seguir el correo, de igual forma a como sucede en una oficina postal.

- Los agentes de usuario permiten a la gente leer y enviar el correo electrónico. Los agentes de usuario son programas locales que proporcionan un método basado en comandos, basado en menús o basado en la interacción gráfica, para comunicarse con el sistema de correo electrónico.

Los agentes de transferencia de mensajes: SMTP y ESMTP

El sistema de transferencia de mensajes se encarga de transmitir los mensajes del remitente al destinatario. La manera más sencilla de hacer esto es establecer una conexión de transporte de la máquina de origen a la de destino y sencillamente transferir el mensaje.

En Internet, el correo electrónico se transfiere de la computadora origen a la computadora destino, mediante el establecimiento de una conexión TCP desde la computadora origen al puerto 25 de la computadora destino. Escuchando este puerto está un "daemon" de correo electrónico capaz de usar el protocolo SMTP (Simple Mail Transfer Protocol). Este "daemon" acepta conexiones de entrada y copia mensajes de ellas a los buzones

adecuados. Si no puede entregarse un mensaje, se devuelve al transmisor un informe de error que contiene la primera parte del mensaje que no pudo entregarse[7].

SMTP: Protocolo sencillo de transferencia de correo.

El SMTP es un sencillo protocolo cliente/servidor en formato ASCII. Establecida una comunicación TCP entre la computadora transmisora del correo, que opera como cliente, y el puerto 25 de la computadora receptora del correo, que opera como servidor, la computadora cliente permanece a la espera de recibir un mensaje de la computadora servidor. El servidor comienza por enviar una línea de texto que proporciona su identidad e indica si está preparado o no para recibir correo. Si no lo está, el cliente libera la conexión y lo intenta después.

Si el servidor está dispuesto a aceptar correo electrónico, el cliente anuncia de quién viene el mensaje, y a quién está dirigido. Si existe tal destinatario en el destino, el servidor da al cliente permiso para enviar el mensaje. Entonces el cliente envía el mensaje y el servidor acusa su recibo. Si existe más correo electrónico se envía ahora. Una vez que todo el correo ha sido intercambiado en ambas direcciones, se libera la conexión.

Los comandos y respuestas del protocolo SMTP así como su significado pueden verse en las tablas siguientes. Son intercambiados tal cual entre los agentes en formato ASCII, por lo cual podemos reliazar una conexión al servidor post.uv.es con un telnet post.uv.es 25 e introducir dichos comandos, siguiendo el protocolo.

Comando[8]	**Descripción**
HELO	Identifica el remitente al destinatario.
MAIL FROM	Identifica una transacción de correo e identifica al emisor.
RCPT TO	Se utiliza para identificar un destinatario individual. Si se necesita identificar múltiples destinatarios es necesario repetir el comando.
DATA	Permite enviar una serie de líneas de texto. El tamaño máximo de una línea es de 1.000 caracteres. Cada línea va seguida de un retorno de carro y avance de línea <CR><LF>. La última línea debe llevar únicamente el carácter punto "." seguido de <CR><LF>.
RSET	Aborta la transacción de correo actual.
NOOP	No operación. Indica al extremo que envíe una respuesta positiva. Utilizados para keepalives.
QUIT	Pide al otro extremo que envíe una respuesta positiva y cierre la conexión.
VRFY	Pide al receptor que confirme que un nombre identifica a un destinatario valido.
EXPN	Pide al receptor la confirmación de una lista de correo y que devuelva los nombres de los usuarios de dicha lista.
HELP	Pide al otro extremo información sobre los comandos disponibles.
TURN	El emisor pide que se inviertan los papeles, para poder actuar como receptor. El receptor puede negarse a dicha petición.
SOML	Si el destinatario está conectado, entrega el mensaje directamente al terminal, en caso contrario lo entrega como correo convencional.
SAML	Entrega del mensaje en el buzón del destinatario. En caso de estar conectado también lo hace al terminal.
SEND	Si el destinatario está conectado, entrega el mensaje directamente al terminal.

[7] Hay configuraciones en los agentes SMTP para aceptar intercambiadores de correo, tal como hemos visto en el DNS con los registros tipo MX.

[8] Los comandos en negrillas son obligatorios en todos los agentes, los demás son opcionales y un agente SMTP puede interpretarlos o no. En ocasiones son limitados y no se aceptan comandos opcionales por temas de seguridad.

Tabla 8.8: Comandos del protocolo SMTP.

Código	Descripción
211	Estado del sistema.
214	Mensaje de ayuda.
220	Servicio preparado.
221	Servicio cerrando el canal de transmisión.
250	Solicitud completada con éxito.
251	Usuario no local, se enviará a <dirección de reenvío>
354	Introduzca el texto, finalice con <CR><LF>.<CR><LF>.
421	Servicio no disponible.
450	Solicitud de correo no ejecutada, servicio no disponible (buzón ocupado).
451	Acción no ejecutada, error local de procesamiento.
452	Acción no ejecutada, insuficiente espacio de almacenamiento en el sistema.
500	Error de sintaxis, comando no reconocido**. Si agente es SMTP y se conecta ESMTP**
501	Error de sintaxis en parámetros o argumentos.
502	Comando no implementado.
503	Secuencia de comandos errónea.
504	Parámetro no implementado.
550	Solicitud no ejecutada, buzón no disponible.
551	Usuario no local, pruebe <dirección de reenvío>. **Si no tiene buzón**
552	Acción de correo solicitada abortada.
553	Solicitud no realizada (error de sintaxis).
554	Fallo en la transacción.

Tabla 8.9: Códigos de respuesta del protocolo SMTP.

Un ejemplo de dialogo donde las líneas del cliente se indican por "C:" y las del servidor por "S:", puede verse a continuación:

S: 220 servicio SMTP bugs.uv.es listo
C: HELO glup.uv.es
S: 250 bugs.uv.es dice hola a glup.uv.es
C: MAIL FROM <profesor@glup.uv.es>
S: 250 transmisor OK
C: RCPT TO: <alumno@bugs.uv.es>
S: 250 receptor OK
C: DATA
S: 354 envía correo; termina con una línea únicamente con"."
C: From: profesor@glup.uv.es
C: To: alumno@bugs.uv.es
C: MIME-Version: 1.0
C: Message-Id: <0123456789.AA00123@glup.uv.es>
C: Content-Type: multipart/alternative; boundary=abcdef
C: Subject: Mensaje de correo
C:
C: Éste es el preambulo, el agente de usuario lo ignora.
C:
C: --abcdef
C: Content-Type: text/richtext
C:

C: Aquí va el mensaje de correo en texto.
C:
C: --abcdef
C: Content-Type: message/external-body;
C: access-type="anon-ftp",
C: site="glup.uv.es",
C: directory="pub",
C: name="mensaje.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --abcdef
C: .
S: 250 mensaje aceptado
C: QUIT
S: 221 bugs.uv.es cerrando conexión

La sintaxis de los comandos del cliente se especifica con rigidez. La sintaxis de las respuestas es menos rígida, sólo cuenta el código numérico, pudiendo cada implementación del protocolo SMTP poner la cadena de texto que desee después del código numérico.

Aunque el protocolo SMTP está bien definido por el RFC 821, pueden surgir algunos problemas. Así, algunas implementaciones más viejas no pueden manejar mensajes mayores de 64 Kbytes. Otro problema se relaciona con las terminaciones de temporización. Si el cliente y el servidor tienen temporizaciones distintas, uno de ellos puede terminar mientras que el otro continúa trabajando, terminando inesperadamente la conexión. Por último, en ocasiones pueden dispararse tormentas de correo infinitas. Por ejemplo, si el ordenador 1 contiene la lista de correo A y el ordenador 2 contiene la lista de correo B y cada lista contiene una entrada para la otra lista, entonces un mensaje enviado a cualquiera de las listas generará una cantidad sin fin de correo electrónico.

Para superar algunos de estos problemas, se ha definido el SMTP extendido (ESMTP) en el RFC 1425. Los clientes que deseen usarlo deben enviar un mensaje EHLO, en lugar de HELO. Si el saludo se rechaza, esto indica que el servidor es un servidor SMTP normal (basado en el RFC 821) y el cliente debe proceder de la manera normal. El tipo de protocolo que implementa un agente SMTP o ESMTP lo podemos ver en el código de respuesta a comandos ESMTP, ya que si la contestación a EHLO es 500, nos da a entender que el agente es SMTP y no interpreta ESMTP.

Relaying o intercambiadores de correo.

Dentro de los comandos utilizados para intercambiarse información los servidores de correo, cabe destacar la opción de "relaying" o confiar en otro servidor para atender las peticiones a un servidor de correo, en el momento que este servidor quede fuera de servicio.

En los comandos anteriores, si el servidor de correo no acepta "relaying", entonces cuando en la entrega, tras "RCPT TO" se especifica una máquina que no es la propia (por ejemplo el servidor es glup.uv.es y RCTP es para felici@uv.es, otra máquina que no es glup.uv.es), se descarta el mensaje. Por el contrario, si aceptamos intercambiadores, aceptaremos el mensaje de otro servidor (en este caso de uv.es, cuando esté fuera de servicio), al cual cuando esté operativa se le reenviará el correo. Para ello, no requiere habilitarse cuentas de usuario de los otros servidores que se está haciendo intercambio, simplemente se recoge el correo de otra máquina que no recibe el correo y a la cual, posteriormente se le hará entrega. Es decir, se actúa como un intermediario o recogedor, para el caso que falle el servidor.

Pasarelas de correo electrónico.

El correo electrónico usando SMTP funciona mejor cuando el transmisor y el receptor están en Internet y pueden manejar conexiones TCP entre ellos. Sin embargo, muchas máquinas que no están en Internet también quieren transmitir y recibir correo electrónico de instalaciones de Internet. Por ejemplo, muchas compañías intencionadamente no quieren estar en Internet por razones de seguridad. Algunas de ellas incluso se aíslan de Internet levantando muros de seguridad entre ellas mismas e Internet.

Otro problema ocurre cuando el transmisor sólo habla RFC 822 y el receptor sólo habla un protocolo de correo X.400 o alguno patentado, específico del proveedor. Dado que estos dos mundos difieren en cuanto al formato de mensaje y los protocolos, es imposible la comunicación directa.

Ambos problemas se resuelven usando pasarelas de correo electrónico. En la figura siguiente, el ordenador 1 sólo habla TCP/IP y RFC 822, mientras que el ordenador 2 sólo habla TP4 y X.400 de OSI. No obstante, pueden intercambiar correo usando una pasarela de correo electrónico.

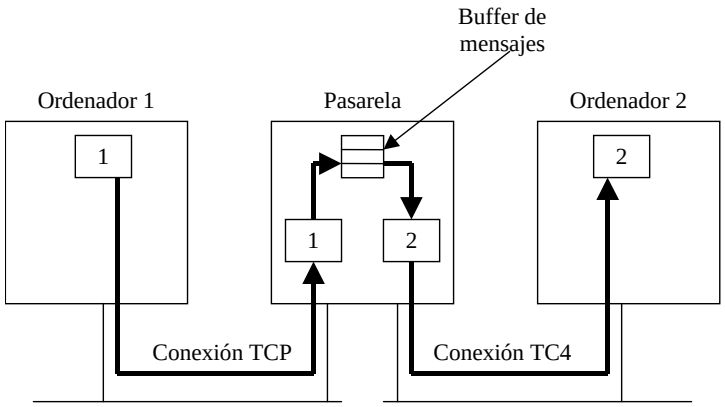


Figura 8.6: Esquema de una pasarela de correo electrónico.

El procedimiento usado consiste en que el ordenador 1 establezca una conexión TCP con la pasarela y luego use SMTP para transferir el mensaje a la pasarela. El "daemon" de la pasarela pone el mensaje en el buffer de mensajes destinados al ordenador 2. Después, se establece una conexión TP4 (el equivalente OSI del TCP) con el ordenador 2 y el mensaje se transfiere usando el equivalente OSI del SMTP.

Parece sencillo, pero no lo es. El primer problema es que las direcciones Internet y las direcciones X.400 son diferentes por completo. Se requiere un complejo mecanismo de transformación entre ellas. El segundo problema es que los campos de envoltura y cabecera presentes en un sistema pueden no estar presentes en el otro. Por ejemplo, si un sistema requiere clases de prioridad y el otro no tiene este concepto, en un sentido de envío debe desecharse información valiosa y en el otro sentido debe generarse de la nada.

Un problema más grave es qué debe hacerse si son incompatibles partes del cuerpo. ¿Qué debe hacer una pasarela con un mensaje Internet cuyo cuerpo contiene una referencia a un archivo de sonido que debe ser obtenido mediante FTP si el sistema de destino no maneja este concepto?. ¿Qué debe hacer la pasarela cuando el sistema X.400 le dice que entregue un mensaje a cierta dirección pero, si falla, envíe el contenido por fax?, pues el uso del fax no es parte del estándar RFC 822. Por tanto, para los mensajes de texto ASCII sencillos y sin estructura, las pasarelas son una solución razonable, pero para cualquier cosa más complicada la idea tiende a desmoronarse.

Protocolos de entrega final del correo electrónico.

Hasta ahora hemos visto como envían y reciben las computadoras el correo electrónico a través de SMTP. Sin embargo, en muchas compañías los usuarios trabajan en un PC de escritorio que no está en Internet y que es incapaz de enviar o recibir correo electrónico. Sin embargo, la compañía puede tener uno o más servidores que pueden enviar y recibir correo electrónico.

Para poder recibir el correo electrónico, un PC debe comunicarse con un servidor de correo electrónico usando alguna clase de protocolo de entrega. Generalmente para poder enviar se realizar a través del propio protocolo SMTP.[9] Los protocolos más comunes son:

- El protocolo POP3 (Post Office Protocol) se define en el RFC 1225. El POP3 tiene comandos para que un usuario establezca una sesión (USER y PASS), la termine (QUIT), obtenga mensajes (RETR) y los borre (DELE). El protocolo mismo consiste en texto ASCII y se asemeja un poco al SMTP. El objetivo del POP3 es obtener correo electrónico del buzón remoto y almacenarlo en la máquina local del usuario para su lectura posterior. Hay versiones actualmente que permiten configurar con POP3 la no descarga del buzón de correo.

- El protocolo IMAP (Interactive Mail Access Protocol) se define en el RFC 1064. IMAP se diseñó para ayudar al usuario que tiene varias computadoras (una estación de trabajo, un PC, un computador portátil). La idea en que se basa IMAP es que el servidor de correo electrónico mantenga un depósito central al que puede accederse desde cualquier máquina. Por tanto, a diferencia del POP3, el IMAP no copia el correo electrónico en la máquina personal del usuario dado que el usuario puede tener varias.

- Un tercer protocolo de entrega es DMSP (Distributed Mail System Protocol) que es parte del sistema PCMAIL y se describe en el RFC 1056. Éste no supone que todo el correo está en un servidor, como el POP3 y el IMAP. En cambio, permite a los usuarios descargar correo del servidor a una estación de trabajo, PC o portátil y luego desconectarse. El correo electrónico puede leerse y contestarse estando desconectado. Al ocurrir una reconexión después, el correo electrónico se transferirá y el sistema se sincronizará.

Además de la posibilidad de entregar el correo a un PC, los agentes de transferencia de mensajes poseen herramientas adicionales que permiten establecer filtros. Los filtros son reglas que se consultan cuando llega un correo electrónico o cuando se inicia el agente de usuario. Cada regla especifica una condición y una acción. Por ejemplo, una regla podría indicar que todos los mensajes recibidos de un cliente se resalten en negrita, etc.

Otra característica de los agentes de transferencia es la capacidad de reenviar el correo electrónico entrante a una dirección diferente. Esta dirección puede ser incluso una computadora operada por un sistema localizador comercial, que entonces llama al usuario por radio o satélite, presentando la línea *Subject:* en su localizador.

Por último, otra característica común es la capacidad de instalar un "daemon" de vacaciones, programa que examina cada mensaje de entrada y envía al transmisor una respuesta como:

Estoy de vacaciones. Regresaré el 15 de Agosto. Que tenga feliz día.

[9] Como correo saliente, se puede especificar o SMTP o SendMail en el caso de una máquina Unix. Si es SMTP, el cliente conectará con su servidor local para pasar los correos por SMTP. Si es Sendmail el correo saliente, tratará de conectar directamente con el agente SMTP remoto. En ocasiones, por temas de seguridad esta conexión no se puede realizar, porque sólo se permite la salida y conexión de SMTP a través del servidor de correo de la corporación. Con SMTP como correo saliente en el usuario, es la configuración habitual en el caso de tener un cortafuegos con la restricción anterior de salida.

La mayoría de los "daemons" de vacaciones mantienen un registro de a quiénes se enviaron respuestas prefabricadas y se abstienen de enviar a la misma persona una segunda respuesta. Los mejores verifican si el mensaje de entrada se envió a una lista de correo, y de ser así, no envían ninguna respuesta prefabricada.

Listas de distribución

Una lista de correo o distribución es un conjunto de direcciones de correo electrónico junto con un software de control. Recibe un mensaje electrónico y lo retransmite automáticamente a todos aquellos que se hayan suscrito previamente, sin que el remitente tenga que enviar el mensaje a cada uno de ellos por separado, o mantener un fichero con las direcciones de todos los destinatarios. Solo necesita el nombre de la lista, enviar un único mensaje a la lista, y el servidor ya se encarga de distribuirlo.

La finalidad básica de una lista de correo es permitir una comunicación rápida y facil **entre un grupo de personas**; de **cada uno** de ellos a **todos** los demás. En general no se pretende que sea una via para que **un** usuario distribuya mensajes a un grupo de direcciones, que se limitan a recibirlos (aunque puede haber listas así configuradas para temas específicos), sino que **todos puedan enviar mensajes a los demás**, así como recibir los de ellos.

La lista tiene un **nombre** (dirección electrónica, en realidad) y en ella se trata una determinada temática (el nombre de la lista suele escogerse para que de una idea de la temática de que trata). Un usuario puede suscribirse a la lista y, a partir de ese momento, los mensajes que envía a la lista son recibidos por un servidor, el cual los distribuye a todos los demás miembros de la lista. La distribución de los mensajes, el alta y baja de usuarios, etc. son gestionados por el *software de la lista* (probablemente con ayuda ocasional de la persona responsable de la lista). Adicionalmente dicho software suele proporcionar una serie de utilidades adicionales, como pueden ser:

- proporcionar información sobre la temática, difusión de la lista, y sobre los suscritos a ella.
- control (más o menos amplio) de acceso a la lista: quien puede o no suscribirse a la lista, quien puede enviar mensajes a ella, etc.
- archivar automáticamente los mensajes enviados a la lista, y proporcionar algún medio de acceso a ellos para referencia futura. Algunos de los productos proporcionan un interfaz web para acceder a los archivos, incluso con capacidades de realizar búsquedas selectivas en ellos.

Existen diversos paquetes de software de gestión de listas de correo. En la *Universitat de València* se instaló el Majordomo (Julio 1997, vers. 1.94.3). Desde principios del 2000 se utiliza el software LISTSERV, el cual posee características superiores al anterior (en cuanto a rendimiento y facilidad de uso).

Los agentes de usuario.

Hemos visto hasta ahora los agentes de transferencia de correo. Veremos a continuación los agentes de usuario.

Un agente de usuario es normalmente un programa que acepta una variedad de comandos para componer, recibir y contestar los mensajes, así como para manipular los buzones de correo. Algunos agentes de usuario tienen una interfaz elegante operada por menús o por iconos que requieren el uso de un ratón, mientras que otros esperan comandos de un carácter desde el teclado. Funcionalmente, todos son iguales.

La labor de un agente de usuario puede dividirse, para su estudio, en su capacidad de procesar y enviar correo electrónico y su capacidad de recibir el mismo.

Envío de correo electrónico.

Para enviar un mensaje de correo electrónico, el usuario debe proporcionar el mensaje, la dirección de destino y, posiblemente, algunos otros parámetros (por ejemplo, prioridad o nivel de seguridad). El mensaje puede producirse con un editor de textos independiente, un programa de procesamiento de textos o, posiblemente, un editor de textos incorporado en el agente de usuario. La dirección de destino debe estar en un formato que

el agente de usuario pueda manejar. La mayoría de agentes de usuario esperan direcciones DNS de la forma *buzón_de_correo@ubicación*. Sin embargo, las direcciones X.400 tienen un aspecto radicalmente diferente del de las direcciones DNS. Las direcciones X.400 se componen de pares atributo=valor, por ejemplo:

/C=US/SP=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/

Esta dirección especifica un país, estado, localidad, dirección personal y nombre común. Son posibles muchos otros atributos, de modo que puede enviarse correo electrónico a alguien cuyo nombre no se conoce, siempre y cuando se conozca un número suficiente de los otros atributos (por ejemplo, compañía y puesto). Esta forma de nombrar es menos funcional que los nombres DNS, sin embargo, los diseñadores de X.400 supusieron que la gente usaría alias (cadenas cortas asignadas por el usuario) para identificar a los destinatarios, por lo que nunca verían direcciones completas. Sin embargo, el software necesario nunca estuvo ampliamente disponible, por lo que la gente que enviaba correo a los usuarios con direcciones X.400 tenía que escribir cadenas como la anterior.

La mayoría de sistemas de correo electrónico manejan listas de correo, por lo que un usuario puede enviar el mismo mensaje a un grupo de personas con un solo comando. Si la lista de correo se mantiene localmente, el agente de usuario puede enviar un mensaje por separado a cada destinatario. Sin embargo, si la lista se mantiene remotamente, entonces los mensajes se multiplicaran en la maquina remota.

Lectura de correo electrónico.

Por norma general, al arrancar un agente de usuario, éste buscará en el buzón del usuario el correo electrónico recibido antes de presentar cualquier otra cosa en la pantalla. Entonces puede anunciar la cantidad de mensajes en el buzón o presentar un resumen de una línea de cada uno y esperar un comando.

Existen incontables programas de correo electrónico, cada uno con sus interfaces propias, etc. Como norma general, todos ellos constan de una serie de comandos o iconos de interfaz que permiten leer el correo, responder al mismo, etc. No parece conveniente, dada la sencillez de los mismos dedicar líneas a describirlos.

Formato de los buzones: mailbox V7

Mientras el correo se guarda en el buzón del usuario a la espera de la entrega final, se utiliza un formato determinado en la propia máquina servidora. El más frecuente es separar los **diferentes correos** con una línea empezando por "**From** " y si se repite en el interior del texto, entonces se introduce el prefijo ">". Este formato es conocido por V7 Mailbox, por ser una convención en Unix Version 7. Existen otros formatos como BABYL, MMDF, MH y QMAIL MAILDIR

Seguridad en el correo

Inicialmente, la seguridad no estaba incluida en Internet, pues el objetivo era extenderse en un ámbito de investigadores y universidades. •Actualmente los temas de seguridad son importantes, ya que muchos **impresentables** hacen uso incorrecto de los servicios de Internet. Por ello, el intercambio de contraseñas entre clientes y servidores que se realiza en modo texto, así como el correo,...deben protegerse.

Estos temas (PGP, PEM, SSMTP, POP3s etc) los retomaremos en el capítulo de seguridad. Son en definitiva, aspectos que integran la seguridad en estos protocolos. También es aconsejable instalar antivirus y que inspeccionen los buzones de correo, para evitar la propagación de virus entre usuarios.

8.2.3 Formato de los mensajes.

Pasemos ahora al formato de los mensajes de correo electrónico. Primero veremos el correo electrónico ASCII básico usando el RFC 822, viendo posteriormente las extensiones multimedia del mismo.

RFC 822.

Los mensajes con formato RFC 822 están formados por una envoltura primitiva (descrita en el RFC 821), algunos campos de cabecera, una línea en blanco, y el cuerpo del mensaje. Cada campo de cabecera consiste en una sola línea de texto ASCII que contiene el nombre del campo, dos puntos (:) y, para la mayoría de los campos un valor. Los principales campos de cabecera de cabecera son:

Cabecera	**Descripción**
To:	Direcciones de email de los destinatarios primarios.
Cc:	Direcciones de email de los destinatarios secundarios. En términos de entrega no existe diferencia con los destinatarios primarios.
Bcc:	Direcciones de email de las copias al carbón ciegas. Es como el campo anterior excepto que esta línea se borra de todas las copias enviadas a los destinatarios primarios y secundarios.
From:	Persona o personas que crearon el mensaje.
Sender:	Dirección de correo del remitente. Puede omitirse si es igual al campo anterior.
Received:	Línea agregada por cada agente de transferencia en la ruta. La línea contiene la identidad del agente, la fecha y hora de recepción del mensaje y otra información que puede servir para detectar fallos en el sistema de enrutamiento. Esta cabecera se va apilando en los emails a medida que el mensaje va saltando entre los diferentes servidores de correo por los que transcurre el correo.
Return-Path:	Puede usarse para identificar una trayectoria de regreso al remitente.

Tabla 8.10: Campos de cabecera de RFC822.

Además de los campos anteriores, los mensajes RFC 822 pueden contener una variedad de campos de cabecera usados por los agentes de usuario o los destinatarios. Los más comunes son:

Cabecera	**Descripción**
Date:	Fecha y hora de envío del mensaje.
Reply-To:	Se usa cuando la persona que escribió el mensaje y la que lo envió no desean ver la respuesta.
Message-Id:	Número único para referencia posterior a este mensaje. Este campo tiene un formato que incluye un número y la dirección de correo completa de quien manda el mensaje.
In-Reply-To:	Identificador del mensaje al que éste corresponde.
References:	Otros identificadores de mensaje.
Keywords:	Claves seleccionadas por el usuario.
Subject:	Resumen corto del mensaje para exhibir en una línea.

Tabla 8.11: Campos de cabecera auxiliares de RFC822.

El RFC 822 explícitamente indica que los usuarios pueden inventar cabeceras nuevas para uso privado siempre y cuando comiencen con la cadena *X*-. Se asegura que no habrá cabeceras futuras que usen nombres que comiencen con *X*-.

MIME- Extensiones multipropósito de correo Internet.

Al comienzo de Internet, el correo consistía exclusivamente en mensajes de texto escritos en inglés y expresados en ASCII. En ese entorno, el RFC 822 hacía todo el trabajo. Sin embargo, hoy día, la red Internet presenta problemas a esa restricción, problemas que incluyen:

- Mensajes en idiomas con acentos (español, francés y alemán).
- Mensajes en alfabetos no latinos (hebreo y ruso).
- Mensajes en idiomas sin alfabetos (chino y japonés).
- Mensajes que no contienen texto (audio y vídeo).

Se propuso una solución en el RFC 1341 y se actualizó en el RFC 1521. Esta solución, llamada MIME (Multipurpose Internet Mail Extensions) se usa ampliamente.

Cabe destacar que MIME no afecta al protocolo SMTP si no a los clientes de correo.

La idea básica de MIME es continuar usando el RFC 822, pero agregar una estructura al cuerpo del mensaje y definir reglas de codificación para los mensajes no ASCII. Al no desviarse del RFC 822, los mensajes MIME pueden enviarse usando los programas y protocolos de correo electrónico existentes. Todo lo que tiene que cambiarse son los programas transmisores y receptores. MIME define cinco nuevas cabeceras de mensaje, como puede verse en la tabla siguiente:

Cabecera	Descripción
MIME-Version:	Identifica la version de MIME. Si no existe se considera que el mensaje es texto normal en inglés.
Content-Description:	Cadena de texto que describe el contenido. Esta cadena es necesaria para que el destinatario sepa si desea descodificar y leer el mensaje o no.
Content-Id:	Identificador único, usa el mismo formato que la cabecera estándar Message-Id.
Content-Transfer-Encoding:	Indica la manera en que está envuelto el cuerpo del mensaje.
Content-Type:	Especifica la naturaleza del cuerpo del mensaje.

Tabla 8.12: Formatos de cabecera MIME.

Veamos más detenidamente las dos últimas cabeceras enumeradas en la tabla anterior.

El *Content-Transfer-Encoding* indica la manera en que está envuelto el cuerpo para su transmisión a través de una red donde se podría tener problemas con la mayoría de los caracteres distintos de letras, números y signos de puntuación. Existen cinco tipos de codificación de los mensajes conocidos con el nombre de esquemas:

1. El esquema más sencillo es simplemente texto ASCII. Los caracteres ASCII usan 7 bits y pueden transmitirse directamente mediante el protocolo de correo electrónico siempre y cuando ninguna línea exceda de 1000 caracteres.

2. El siguiente esquema más sencillo es lo mismo, pero usando caracteres de 8 bits, es decir, todos los valores de 0 a 255. Este esquema de codificación viola el protocolo original del correo electrónico de Internet, pero es usado por algunas partes de Internet que implementan ciertas extensiones al protocolo original. Mientras que la declaración de la codificación no la hace legal, hacerla explícita puede cuando menos explicar las cosas cuando algo sucede mal. Los mensajes que usan codificación de 8 bits deben adherirse a la longitud máxima de línea estándar.

3. Los mensajes que usan codificación binaria. Éstos son archivos binarios arbitrarios que no sólo utilizan los 8 bits, sino que ni siquiera respetan el límite de 1000 caracteres por línea. Los programas ejecutables caen en esta categoría. No se da ninguna garantía de que los mensajes en binario llegarán correctamente.

4. La manera correcta de codificar mensajes binarios es usar codificación base64, a veces llamada armadura ASCII. En este esquema, se dividen grupos de 24 bits en unidades de 6 bits, enviándose cada unidad como carácter ASCII legal. La codificación es 'A' para 0, 'B' para 1, etc., seguidas por las 26 letras minúsculas, los 10 dígitos y, por último, + y / para los valores 62 y 63 respectivamente. Las secuencias == y = se usan para indicar que el último grupo contenía solo 8 o 16 bits, respectivamente. Los retornos de carro y avances de línea se ignoran, por lo que pueden introducirse a voluntad para mantener la línea lo suficientemente corta.

5. En el caso de mensajes que son casi completamente ASCII, pero con algunos caracteres no ASCII, la codificación base64 es algo ineficiente. En cambio, se puede usar una codificación conocida como codificación entrecomillada-imprimible. Ésta codificación es ASCII de 7 bits, con todos los caracteres por encima de 127 codificados como un signo de igual seguido del valor del carácter en dos dígitos hexadecimales.

Veamos a continuación un ejemplo de codificación base64. Supongamos que deseamos enviar el siguiente fragmento de código binario de un programa:

	214	04	23	97	08	200
Binario	11010110	00000100	00010111	01100001	00001000	11001000

Dividimos cada grupo de 24 bits en grupos de 6 bits, con lo que resultan lo grupos de seis bits siguientes:

110101	100000	010000	010111	011000	010000	100011	001000

Calculando ahora su valor decimal y teniendo en cuenta la codificación en caracteres descrita con anterioridad:

53	32	16	23	24	16	35	8
1	g	Q	X	Y	Q	j	I

Por lo cual el mensaje de correo que enviaríamos seria: 1gQXYQjI

El *Content-Type* especifica la forma del cuerpo del mensaje. Existen siete tipos definidos en el RFC 1521, cada uno de los cuales tiene uno o más subtipos. El tipo y el subtipo se separan mediante un carácter diagonal (/), como:

Content-Type: video/mpeg

El subtipo debe indicarse explícitamente en la cabecera; no se proporcionan valores predeterminados. La lista inicial de tipos y subtipos especificada por el RFC 1521 puede verse en la tabla siguiente. Se han agregado muchos otros desde entonces y se agregan nuevas entradas a medida que surge la necesidad.

Tipo	**Subtipo**	**Descripción**
Text	Plain	Texto sin formato.
	Richtext	Texto con comandos de formato sencillos.
Image	Gif	Imagen fija en formato GIF.
	Jpeg	Imagen fija en formato JPEG.
Audio	Basic	Sonido.

Video	Mpeg	Película en formato MPEG.
Application	Octet-stream	Secuencia de bytes no interpretada.
	Postscript	Documento imprimible PostScript.
Message	Rfc822	Mensaje MIME RFC 822.
	Partial	Mensaje dividido para su transmisión.
	External-body	El mensaje mismo debe obtenerse de la red.
Multipart	Mixed	Partes independientes en el orden especificado.
	Alternative	Mismo mensaje en diferentes formatos.
	Parallel	Las partes deben verse simultáneamente.
	Digest	Cada parte es un mensaje RFC 822 completo.

Tabla 8.13: Tipos y subtipos del formato MIME.

Repasemos la lista de tipos. El tipo *text* es para texto normal. La combinación *text/plain* es para mensajes ordinarios que pueden visualizarse como se reciben, sin codificación ni ningún procesamiento posterior. Esta opción permite el transporte de mensajes ordinarios en MIME con sólo unas pocas cabeceras extra.

El subtipo *text/richtext* permite la inclusión de un lenguaje de marcación sencillo en el texto. Este lenguaje proporciona una manera independiente del sistema para indicar negritas, cursivas, tamaños en puntos menores y mayores, sangrías, etc. El lenguaje de marcación se basa en el SGML (Standard Generalized Markup Language), también usado como base del HTML usado en el Web.

El siguiente tipo MIME es *image*, que se usa para transmitir imágenes fijas. Hoy día se usan muchos formatos para almacenar y transmitir imágenes, tanto con compresión como sin ella. Dos de estos, *gif* y *jpeg* son subtipos oficiales.

Los tipos *audio* y *video* son para sonido e imágenes en movimiento, respectivamente. Nótese que *video* sólo incluye la información visual, no la pista de sonido. Si se debe transmitir una película con sonido, tal vez sea necesario transmitir las partes de vídeo y de audio por separado, dependiendo del sistema de codificación usado.

El tipo *application* es un tipo general para los formatos que requieren procesamiento externo no cubierto por ninguno de los otros tipos. Un *octet-stream* simplemente es una secuencia de bytes no interpretados. A la recepción de una de tales secuencias, un agente de usuario debería presentarla en la pantalla sugiriendo al usuario que se copie en un archivo y solicitando un nombre de archivo. El otro subtipo definido es *postscript*, que se refiere al lenguaje PostScript producido por Adobe Systems y usado por muchas páginas impresas. Aunque un agente de usuario puede llamar a un intérprete PostScript externo para visualizar los archivos PostScript entrantes, hacerlo no está exento de riesgos al ser PostScript un lenguaje de programación completo, lo cual permite que alguien, además de exhibir texto, pueda leer, modificar o borrar los archivos del usuario y tener otros efectos secundarios desagradables.

El tipo *message* permite que un mensaje esté encapsulado por completo dentro de otro. Este esquema es útil para reenviar, por ejemplo, correo electrónico. Cuando se encapsula un mensaje RFC 822 completo en un mensaje exterior, debe usarse el subtipo *rfc822*. El subtipo *partial* hace posible dividir un mensaje encapsulado en pedazos y enviarlos por separado. Los parámetros hacen posible ensamblar correctamente todas las partes en el destino. Por último el subtipo *external-body* puede usarse para mensajes muy grandes, por ejemplo películas de vídeo. En lugar de incluir el archivo mpeg en el mensaje, se da una dirección de FTP y el agente de usuario del receptor puede obtenerlo a través de la red cuando se requiera.

El último tipo es *multipart*, que permite que un mensaje contenga más de una parte, con el comienzo y el fin de cada parte claramente delimitados. El subtipo *mixed* permite que cada parte sea diferente. En contraste, el subtipo *alternative* indica que cada parte contiene el mismo mensaje, pero expresado en un medio o codificación diferente. El subtipo *parallel* se usa cuando todas las partes deben "verse" simultáneamente. Por

ejemplo, las películas con frecuencia tienen un canal de audio y un canal de vídeo. Por último, el subtipo *digest* se usa cuando se juntan muchos mensajes en un mensaje compuesto.

Veamos un ejemplo de mensaje MIME:

From: profesor@glup.uv.es
To: alumno@bugs.uv.es
MIME-Version: 1.0
Message-Id: <0123456789.AA00123@glup.uv.es>
Content-Type: multipart/alternative; boundary=abcdef
Subject: Mensaje de correo

Éste es el preambulo, el agente de usuario lo ignora.

--abcdef
Content-Type: text/richtext

Aquí va el mensaje de correo en texto.

--abcdef
Content-Type: message/external-body;
access-type="anon-ftp",
site="glup.uv.es",
directory="pub",
name="mensaje.snd"

content-type: audio/basic
content-transfer-encoding: base64
--abcdef

8.3 SNMP: Simple Network Management Protocol

8.3.1 Introducción

Es una aplicación para analizar el estado de los dispositivos de la red, permitiendo su manejo (configuración) y monitorización, con lo cual permite diagnosticar problemas existentes. Antiguamente en ocasiones de un mal funcionamiento de la red se utilizaba ***ping*** y ***traceroute***, pero la información suministrada por este comando es muy limitada para detección de problemas.

La administración de redes gira entorno al concepto de MIB. Un MIB es una base de datos estructurada de información que esta físicamente localizada en un dispositivo de red. El acceso a esta información lo proporciona el protocolo denominado SNMP (Simple Network Management Protocol). Un subconjunto de esta base de datos, es el MIB-2. Esto es una base de datos con información común que está soportada por todos los dispositivos de red.

8.3.2 Estándares de la estructura de la información de administración

El SMI (Structure of Management Information) presenta una estructura en forma de árbol global para la información de administración, convenciones, sintaxis y las reglas para la construcción de MIBs.

Un MIB es una base de datos que contiene información que caracteriza los aspectos funcionales y operacionales de un dispositivo de red. Los MIBs contienen objetos que necesitan ser gestionados para controlar un dispositivo. Estos objetos se pueden considerar como variables a controlar en cada dispositivo y cada variable refleja un estado o información del dispositivo, como por ejemplo si está o no conectado, el número de errores, tamaño de las colas, etc.

Actualmente existen gran variedad de MIBs. Algunos son específicos de un fabricante y otros muchos son comunes. Los MIBs comunes (por ejemplo: MIB-2, RMON y RMON2) son muy importantes en la administración de redes. Proporcionan una serie de objetos que son soportados por la mayoría de dispositivos, dando al administrador una importante herramienta para el control de dispositivos independientemente del fabricante.

Todos lo objetos de un MIB poseen un identificador único (OID). Este identificador está formado por, enteros separados por puntos o bien etiquetas o bien de forma mixta, como se identifica en el árbol SMI, como se ve en la figura. Se necesita conocer el OID de un objeto si se pretende acceder a este objeto dentro de un MIB. El árbol global de objetos es completamente extensible a nuevos MIBs, ya que existen infinidad de ramas que pueden ser añadidas por debajo de ciertas ramas (privadas o experimentales) designadas por el IETF. Esta estructura proporciona un código de identificación que es único para cada rama. Ejemplo: *iso.org.dod.internet.mgmt.mib_2.interfaces*, ó 1.3.6.1.2.1.2.

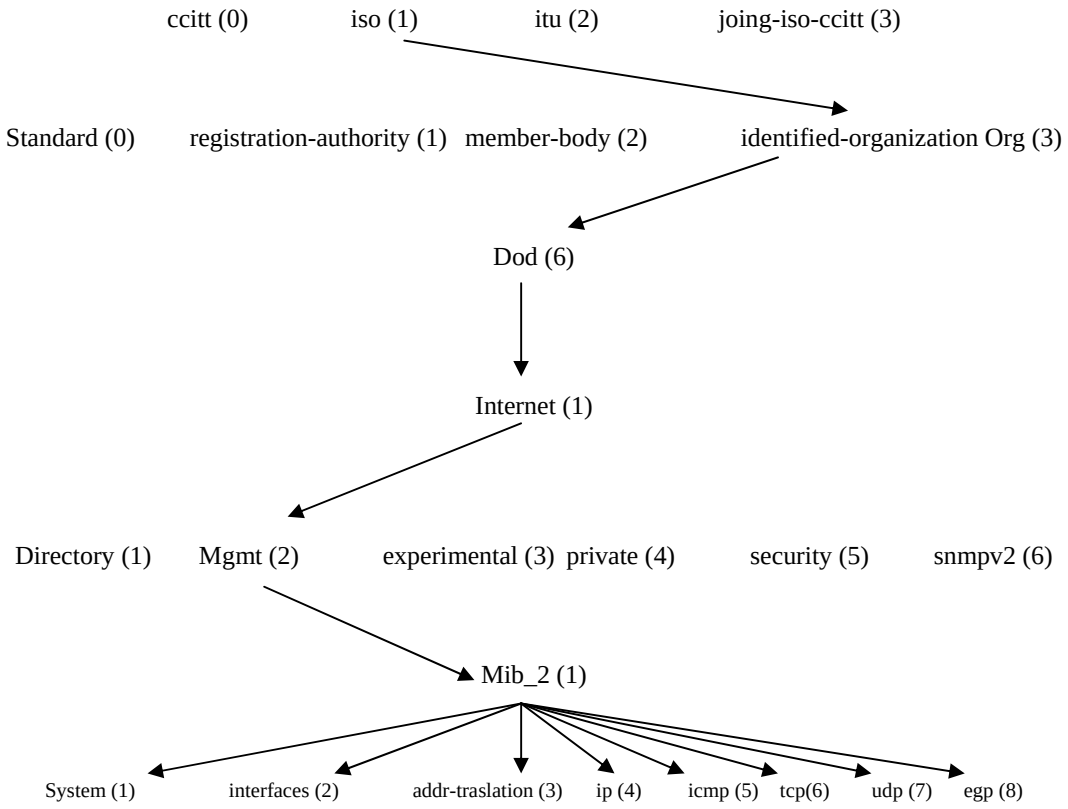


Figura 8.7: Árbol SMI global estructurado para sintaxis y convenciones

Como podemos observa, la raíz del árbol SMI (iso(1).org(3).dot(6).internet(1).mgmt(2) o 1.3.6.1.2) es usado repetidamente en los OIDs dentro de SNMP, por lo que a menudo se define como el prefijo principal de todos los objetos del MIB. Esto es así, porque las variables o objetos manejados por SMNP en la administración de equipos en Internet, caen debajo de esta rama del árbol.

8.3.3 El estándar SNMP

SNMP es el protocolo de administración que facilita la comunicación entre la estación administradora y el agente de un dispositivo de red. SNMP se puede implementar usando comunicaciones UDP o TCP, pero por norma general, se suelen usar comunicaciones UDP en la mayoría de los casos. Con UDP, el protocolo SNMP se implementa utilizando los puertos 161 y 162. El puerto 161 se utiliza para las transmisiones normales de comando SNMP, mientras que el puerto número 162 se utiliza para los mensajes de tipo “trap”.

El modelo subyacente del SNMP es una estación administradora que envía solicitudes a los agentes de nodos administrados, haciendo consultas sobre cualquiera de las variables o objetos del MIB. Veremos ahora el protocolo mismo al que le hablan la estación administradora y los agentes.

La forma normal de uso del SNMP es que la estación administradora envíe una solicitud a un agente pidiéndole información o mandándole actualizar su estado de cierta manera. Idealmente, la respuesta del agente simplemente es la información solicitada o la confirmación de que ha actualizado su estado según se solicitó. Los datos se envían empleando la sintaxis de transferencia del ASN.1. Es un formato para representación de los datos para ínter actuación de diferentes fabricantes. Sin embargo, también pueden informarse varios errores, como “no existe tal variable”.

El SNMP proporciona un acceso remoto a todos los objetos soportados por un dispositivo de red. Este protocolo facilita el muestreo del dispositivo con el objetivo de chequear los posibles estados en los que se encuentra este, recoger características operacionales, y recibir notificaciones sobre eventos producidos por algún cambio importante en el dispositivo. El SNMP fue definido para ser un protocolo simple, con el objetivo de obtener una amplia y rápida aceptación. En un principio fue así, pero posteriores versiones han complicado el protocolo a cambio de conseguir mejores prestaciones en diversos aspectos.

Todos los datos SNMP siguen la estructura de los MIBs definidos mediante el SMI. Actualmente, los datos que se envían a través del cable están codificados en un subconjunto de la ASN.1 (Abstract Syntax Notation One).

8.3.4 El estándar MIB-2

Este es el MIB más importante para SNMP. El MIB-2 está ampliamente expendido dentro de Internet, ya que proporciona un importante conjunto de posibilidades para una gran variedad de dispositivos diferentes.

MIB-2

Cuando un dispositivo de red soporta el MIB-2, permite tener acceso a una colección de datos de administración comunes y de gran utilidad. Cuando se tiene un conjunto discreto de diversos dispositivos que soportan el MIB-2, se tiene acceso al mismo conjunto de datos de administración en todos los dispositivos. Esto es importante, ya que permite tratar idénticamente los diferentes dispositivos, sin importar de qué dispositivo se trate.

Por ejemplo, MIB-2 proporciona los contadores necesarios para calcular la utilización de una red Ethernet o Token Ring. La utilización proporciona una medida del porcentaje de tiempo durante el que un recurso es utilizado dentro de un cierto intervalo de tiempo dado. Esto permite que una aplicación de administración obtenga la utilización de un determinado dispositivo X, seguidamente calcule la utilización de un dispositivo Y, consiguiendo así obtener una serie de estadísticas que van a facilitar la comparación entre estos dos dispositivos. Debido a que las redes suelen estar compuestas por dispositivos de diferentes fabricantes, la obtención de características de este tipo, va a permitir tomar una serie de decisiones importantes en algunos casos.

Leyendo un MIB

El SMI proporciona las reglas básicas y convenciones usadas para definir el MIB y la estructura de los identificadores de los objetos utilizables. Estos objetos están formados por los siguientes campos:

- Nombre (usando la convención del SMI)
- Tipo de dato (puede ser un entero, texto, cadena, etc.)
- Descripción (explicación del contenido)
- Acceso al objeto (si es posible acceso de lectura o escritura)
- Identificador del objeto (identificador único dentro del árbol de objetos)

Determinando el identificador de un objeto

El identificador de un objeto es necesario para acceder y cambiar el valor de un objeto en el caso que se pueda escribir sobre él. El siguiente código extraído del MIB-2 muestra la sintaxis basada en el SMI usada para representar el objeto **sysDescr** (objeto que contiene una descripción del sistema en el que se encuentra) y la posición de **sysDescr** dentro del árbol global del MIB. El identificador de **sysDescr** es 1.3.6.1.2.1.1.1. como podemos comprobar en el árbol SMI contiene referencias a otros MIBs que proporcionan información externa a este fichero. De esta simple forma es posible identificar cualquier objeto del árbol conociendo las ramas de las que procede.

Por tanto, a partir del árbol de objetos, deducimos que el identificador de **sysDescr** es el siguiente:

Iso=1
Org=3
Dod=6
Internet=1
Mgmt=2
Mib-2=1
System=1
***SysDescr**=1*

Con lo que obtenemos el identificador que estábamos buscando.

Adicionalmente, cada objeto dentro de un dispositivo, incluye atributos. Por tanto, el resto de la definición de **sysDescr**, se completa y vendría dado a través de otros atributos de objetos.

Atributos de objetos

Estos atributos son:

SYNTAX: indica el tipo de dato del objeto. En este caso, el objeto **sysDescr** es un DisplayString, definido en el SMI como una cadena de texto de 0 a 255 caracteres. Existen muchos otros tipos de datos definidos en SMI como veremos.

ACCESS: para **sysDescr** es *solo lectura*. Esto significa que el objeto solamente puede ser leído, pero no escrito por la estación administradora. Los diferentes tipos de acceso son: *read-only*, *read-write*, *write-only*, o *not-accessible* para objetos que existen dentro del agente pero que no son accesibles externamente.

STATUS: describe la versión de SNMP para administrar dicho objeto.

DESCRIPTION: proporciona una descripción textual del objeto. En el caso de sysDescr, la descripción nos dice que el objeto contiene la descripción del sistema que contiene al dispositivo

Tipo de datos comunes

El SMI proporciona un gran número de tipos de datos, aunque en la mayoría de las ocasiones se utilizan un número reducido de estos. Estos tipos de datos más comunes son:

- Integer: número entero, positivo o negativo, limitado a una precisión de 32 bits.
- Octec String: secuencia ordenada de 0 o más octetos.
- Display String: un octet string que contiene cualquier carácter ASCII
- Object Identifier: un identificador único en el árbol MIB, definido oficialmente
- Object Descriptor: forma textual imprimible de un OID. Mientras un OID es un array de enteros, el object descriptor es un string del array de enteros separados por puntos decimales.
- Sequence: es un grupo de 1 o más objetos ordenados, almacenados como una fila en una tabla.
- Ipaddress: Es una dirección IP de 32 bits representada por 4 bytes
- Counter: entero positivo que se incrementa hasta alcanzar el valor máximo. Una vez alcanzado, si se vuelve a incrementar, toma el valor 0.
- Gauge: es un entero positivo que puede incrementarse o decrementarse. Cuando se alcanza el valor máximo, este se mantiene hasta que no se decremente. Es decir que no vuelve al valor 0.
- TimeTicks: entero positivo que cuenta el tiempo en centésimas de segundo.

Areas funcionales del MIB-2

Por conveniencia, los objetos administrados por SNMP se agrupan en categorías, que se localizan bajo la rama de Mib_2 en el árbol SMI. Los objetos que están dentro del MIB-2 están agrupados en una serie de

ramas, en función de un conjunto de características comunes. Los objetos que pertenecen a cada uno de estos grupos se caracterizan por un prefijo y un identificador que indican a qué grupo pertenecen. Por ejemplo, todos los objetos dentro del grupo system tienen el prefijo "sys".

Una breve descripción de cada uno de estos grupos y de algunas de sus componentes más destacadas:

1. System: (prefijo "sys")Contiene información general sobre el dispositivo. Muchos de los objetos son configurables por el administrador. Los objetos que lo forman son:

    - SysDescr: descripción textual del dispositivo.
    - SysObjectID: indica una identificación especifica del vendedor
    - SysUpTime: es el tiempo en centésimas de segundo, desde que el dispositivo fue arrancado.
    - SysContact: es un campo que puede contener el nombre de la persona responsable del dispositivo.
    - SysName: define un nombre para el dispositivo dentro de la red.
    - SysLocation: define donde esta localizado el dispositivo
    - SysServices: es un entero de indica que niveles de red soporta el dispositivo.

2. Interfaces: (prefijo "if") Contiene una fila de información sobre los enlaces del dispositivo y una cuenta de estas filas. Algunos de los objetos más importantes que contiene este grupo son la velocidad del enlace (ifSpeed), el tipo de enlace (ifType), el estado operacional del enlace (ifOperStatus), y una serie de contadores de errores. Este grupo es crítico cuando se maneja un dispositivo que posea diferentes enlaces.
3. IP: (prefijo "ip") Contiene información útil para manejar el dispositivo en el nivel 3 (nivel de red). El *ipRouteTable* contiene información para determinar como enrutar paquetes, (como por ejemplo, destino del enrutado, varias medidas de enrutado, y el protocolo de enrutado).
4. ICMP: (prefijo "icmp") Contiene información para monitorizar el protocolo ICMP (contadores de paquetes, contadores de errores, etc.)
5. TCP: (prefijo "tcp") Contiene información importante para monitorizar el protocolo TCP. Una tabla importante contenida dentro de este grupo es la tabla de conexiones (tcpConnTable). Esta tabla contiene mapeado de direcciones y de puertos y el estado de la conexión (por ejemplo: *cerrada* o *establecida*).
6. UDP: (prefijo "udp") contiene información sobre el protocolo UDP, como contadores de paquetes y de errores.
7. EGP: (prefijo "egp") Contiene información sobre este protocolo usado para el intercambio de información de enrutado entre routers.
8. Transmission: (prefijo "transmission") Realmente esto no es un grupo, ya que es un nodo de posición en el árbol MIB-2. Para cada transmisión estándar, existe un subárbol de administración MIB. Por ejemplo existen subárboles para FDDI, Token Ring, y Ethernet.
9. SNMP: (prefijo "snmp") Contiene información sobre SNMP como paquetes entrantes, salientes y un número de errores.

Ejemplo: sysName, como hemos visto nos informa del nombre del sistema y se puede ser alcanzado por 1.3.6.1.2.1.1.5.

VARIABLES MÁS IMPORTANTES del MIB-2 DENTRO DEL AREA FUNCIONAL INTERFACE

Son **l**as variables que tienen una mayor relevancia dentro del MIB-2. Como hemos comentado en anteriores ocasiones, el MIB-2 posee cientos de variables, pero en la práctica, únicamente nos van a importar un grupo limitado de estas. Una breve descripción del objetivo de cada una de estas variables las vemos en la siguiente table:

IfAdminStatus:	Se utiliza para cambiar el estado operacional de un enlace. Unicamente es accesible por el administrador de la red.
IfOperStatus:	Nos indica el estado operacional de un determinado enlace.
IfInErrrors:	Nos indica el número de paquetes entrantes en los que el agente ha encontrado algún tipo de error
IfInDiscards:	El número de paquetes entrantes que el agente ha seleccionado para ser eliminados. No necesariamente son solo los que contienen errores. Una razón para eliminar paquetes suele ser liberar espacio en los buffers de entrada.
IfInOctets:	Indica el número total de octetos recibidos en el enlace.
IfOutErrors:	Indica el número total de paquetes que no han podido ser enviados por causa de un error.
IfOutDiscards:	El número de paquetes salientes que el agente ha seleccionado para ser eliminados. Además de aquellos que contienen algún error, también se eliminan para liberar espacio en los buffers de salida.
IfOutOctets:	El número total de octetos transmitidos fuera del enlace.
IfSpeed:	Nos da una estimación del ancho de banda actual que dispone el enlace en bits por segundo. En aquellos enlaces en los que el ancho de banda no varíe o en los que no sea posible hacer una aproximación exacta, este objeto suele contener el ancho de banda nominal.

Tabla 8.14: Objetos del área funcional Interfaz

8.3.5 Mensajes SNMP

El SNMP define siete mensajes que pueden enviarse. Los seis mensajes de un iniciador se listan en la tabla siguiente (el séptimo mensaje es el de respuesta).

Get_request	Solicita el valor de una o más variables
Get_next_request	Solicita el valor de la variable que sigue a esta.
Get_bulk_request	Obtiene una tabla grande
Set_request	Actualiza una o más variables
Inform_request	Mensaje de administrador a administrador describiendo la MIB local
Snmp_v2_trap	Informe de interrupciones de agente a administrador.

Tabla 8.15: Mensajes SNMP

Los tres primeros mensajes solicitan la devolución de valores de variables. El primer formato nombra explícitamente las variables que quiere. En el caso de que se pida más de una variable, se deberá construir una lista, que contenga las variables solicitadas. La construcción de esta lista debe seguir unas normas comunes al agente y a la estación, para que sea posible la comunicación y el entendimiento entre ambos. El segundo solicita la siguiente variable, permitiendo al administrador ir paso por paso a través de la MIB en orden alfabético (la predeterminada es la primera variable). El tercero es para transferencias grandes con tablas.

Los siguientes mensajes permiten al administrador actualizar las variables de un agente, hasta el límite permitido por la especificación del objeto, claro está. Sigue una solicitud informal que permite a un administrador indicarle a otro las variables que está manejando. Por último, viene el mensaje enviado de un agente a un administrador cuando ha habido una interrupción.

Actualmente existen 2 versiones del protocolo SNMP (aunque ya se está empezando a extender la tercera). Seguidamente voy a exponer brevemente algunas de las particularidades más destacadas de cada versión.

EJEMPLO de CONSULTA SNMP: snmpget(): cálculo de utilización de una interface de un enlace serie

En primer lugar, la consulta debe ir precedida por el comando que se quiere ejecutar. En nuestro caso, vamos a utilizar el comando *snmpget*, ya que es el idóneo para preguntar por una serie limitada de variables conocidas.

El primer parámetro que posee este comando es el identificador de la máquina que se desea consultar. Como identificador se pude usar tanto el nombre como la dirección IP. Seguidamente debemos colocar la comunidad a la que pertenecen las variables a consultar. Este campo se utiliza para restringir el acceso a determinada información. Por norma general, la comunidad más utilizada es “public”.

Y finalmente se colocan los identificadores de los objetos que se desea consultar. Estos identificadores siguen la notación SMI vista anteriormente.

Para calcular la utilización de un enlace, debemos seguir los siguientes pasos. La utilización de un enlace viene determinada tanto para la *utilización del interface entrante y la utilización del interface saliente* en un router. Por tanto la utilización de un interface se define como:

*(numero de bits procesados)/((tiempo que ha tardado)*velocidad)*

es decir los bits mandados, frente a los bits que hubiese podido mandar si la utilización hubiese sido de 1. En este caso, las variables a procesar serían *ifInOctets*, *ifOutOctets* y *ifSpeed* como se ha visto anteriormente.

Los valores de estas variables son incrementales, es decir, que su valor se va incrementando hasta que la variable toma su máximo valor, cuando se vuelva a incrementar, la variable tomará el valor 0. Como a nosotros nos interesa el incremento de una muestra a la siguiente, debemos calcular la diferencia entre muestras para saber en cuanto se ha incrementado cada variable. El cálculo de este incremento no va a ser necesario en todas las variables, ya que el valor de la velocidad de la línea IfSpeed, debe permanecer

constante. (A no ser que se produzca un cambio en la configuración del dispositivo, pero en este caso no se presentará el incremento, sino el valor real).

Un ejemplo de su manejo es:

"snmpget(dominio,@numerosASN1)" ó "*snmpget dominio palabras clave separadas por espacios*"

Un dominio de un router puede ser

Name: institutro.ci.uv.es Address: 147.156.160.1

Por ejemplo *snmpget post.uv.es public system.sysDescr.0* que pregunta por la variable system.sysDescr.0, siendo dicha notación, correspondiente a "system.variable.puerto", es decir consulta la variable **sysDescr** dentro del objeto system, en la interfaz 0 del dispositivo a monitorizar. En el caso de puerto=0, se refiere a la propia máquina local.

Por tanto con SNMP se ha de implementar una consulta de las siguientes variables:
ifInOctets, ifOutOctets y *ifSpeed*. Un ejemplo de dicha consulta es

snmpget ronmbre_router public ifInOctets.1 ifOutOctects.1

para hacer una consulta para obtener en decimal las variables *ifInOctets* y *ifOutOctects* de la máquina *nombre_router*, en su interfaz o puerto 1.

En Linux, las consultas se pueden hacer desde programa con la petición al sistema a través de *system*("llamada_al_sistema") definido en #include <stdlib h> con *int system (const char * string).*

8.3.6 SNMPv2

El SNMPv2 es un intento de solucionar las deficiencias que con el paso del tiempo han ido surgiendo en la primera versión. Estas deficiencias están causadas básicamente, por el intento de mantener la v1 lo más simple posible, además de por el incremento en la complejidad en las redes de ordenadores, y la necesidad de un protocolo más sofisticado. La aparición de esta segunda versión no estuvo exenta de polémica, ya que al intentar mejorar deficiencias como, la falta de seguridad en las transmisiones, se llego a un protocolo demasiado complicado, por lo que ha sido revisado en varias ocasiones. Las características más destacables que se añaden en esta segunda versión son:

1. *Seguridad:* La primera versión no era segura, ya que se enviaba el nombre de la comunidad en texto llano, por lo que era posible hacer cambios en la configuración de un dispositivo por cualquier persona que tuviese acceso a la red. En la segunda versión se incorporan técnicas de encriptación para intentar solucionar este problema.
2. *Operaciones con grandes volúmenes:* La segunda versión incorpora el PDU GetBulkRequest. Este PDU proporciona a las estaciones administradoras la posibilidad de trabajar con grandes cantidades de datos, de una sola vez. Esto es especialmente importante a la hora de recuperar grandes tablas, ya que con el comando GETNEXT se recogía una fila en cada llamada. Este comando llena el buffer de respuesta hasta su capacidad máxima, con lo que el tráfico por la red disminuye al disminuir el número de peticiones necesarias para obtener una tabla
3. *Nuevo formato para los eventos:* Los traps en la primera versión tenían un formato diferente al de cualquiera de los otros comandos (GET, SET o GETNEXT). En la segunda versión se unifica este formato, siendo el mismo para un trap que para un comando de petición.
4. *Comunicación administrador-administrador:* Un nuevo PDU que se añade es el Inform_Request PDU, con este se facilita la comunicación entre diferentes estaciones administradoras, permitiendo la existencia de jerarquías que son necesarias en sistemas muy complejos. Este estilo de administración reduce el trafico de administración y sobre todo la cantidad de información a transmitir.

5. *Tipos de datos de 64 bits:* Los avances a nivel físico, exigían la aparición de nuevos tipos de datos más eficaces.
6. *Mejoras en las adquisiciones:* En la primera versión, cuando una de las variables en la VBL de un comando GET fallaba, hacia que todo el comando fallara, por lo que debía eliminarse esta variable de la lista, y volverse a consultar. En SNMPv2 este procesamiento es automático, y el error se indica en el valor de retorno de la variable.

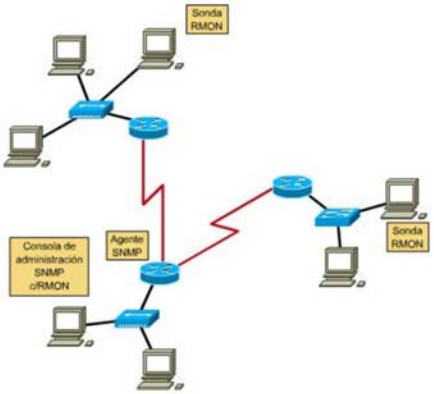


Figura 8.8: Esquema de red con RMON monitorizando otros segmentos.

8.3.7 RMON y monitorización remota

SNMP gestiona dispositivos individuales, pero no permite diagnosticar fallos en un red remota u otro segmento de red. Para ello, el software de monitorización debe trasladarse a cada segmento de red.

Esto se puede resolver mediante el uso de agentes en los segmentos remotos de red, utilizando equipos especiales o bien ordenadores de propósito general, llamados sondas RMON (Remote MONitor). Una de las mejoras principales de SNMP se denomina Monitoreo Remoto (RMON) como puede verse en la figura 8.8.

Las extensiones de RMON a SNMP brindan la capacidad para observar la red como un todo, aunque esté distribuida, en contraste con el análisis de dispositivos individuales, declarándose para ello una MIB especial para guardar información de monitorización de un segmento de red diferente. La MIB asociada es 1.3.6.1.2.1.16

Las sondas RMON recopilan información y tiene la misma función que un agente SNMP, transmitiendo la información periódicamente. Además, pueden procesar la información a enviar a la estación de administrador.

La RMON está localizada en cada segmento de red y pueden introducirse en un host, en un switch, en un router o en un dispositivo específico para ello. Además, permite añadir redundancia a la administración de la red, ya que RMON permite volcar los datos a varias consolas de administración. La RMON1 trabaja en información de capa 1 y 2. La RMON2 trabaja en información de capa 3 y superiores.

9 SEGURIDAD

Al principio de su existencia, las redes de ordenadores fueron usadas generalmente para el envío de correo electrónico y para compartir recursos, generalmente impresoras, en empresas de mediano/gran tamaño. En estas condiciones la seguridad carecía prácticamente de importancia y no fue objeto de atención. Sin embargo, en la actualidad millones de ciudadanos usan redes para transacciones bancarias, compras, etc., la seguridad aparece como un problema potencial de grandes proporciones. Los problemas de seguridad de las redes pueden dividirse de forma general en cuatro áreas interrelacionadas:

- El secreto, encargado de mantener la información fuera de las manos de usuarios no autorizados.
- La validación de identificación, encargada de determinar la identidad de la persona/computadora con la que se esta hablando.
- El no repudio, encargado de asegurar la “firma” de los mensajes, de igual forma que se firma en papel una petición de compra/venta entre empresas.
- El control de integridad, encargado de asegurar que el mensaje recibido fue el enviado por la otra parte y no un mensaje manipulado por un tercero.

Aunque muchos de estos problemas tratan de resolverse en capas de la red que se encuentran por debajo de la capa de aplicación, por ejemplo en la capa de red pueden instalarse muros de seguridad para mantener adentro (o afuera) los paquetes, en la capa de transporte pueden cifrarse conexiones enteras terminal a terminal, ninguna de ellas resuelve completamente los problemas de seguridad antes enumerados.

La resolución de estos problemas de seguridad se realiza como una parte previa o de apoyo de la capa de aplicación. A continuación se expondrán distintos trabajos que tratan de resolver cada uno de los cuatro problemas de seguridad planteados con anterioridad, esto es, el secreto, la validación de identificación, el no repudio y el control de integridad.

Antes de comenzar este capítulo, pasemos a realizar una serie de definiciones:

1. La Organización Internacional de Estándares (ISO), como parte de su norma 7498 en la que se establece el modelo de referencia para la interconexión de sistemas abiertos, define la ***seguridad informática como una serie de mecanismos que minimizan la vulnerabilidad de bienes y recursos***, donde un bien se define como algo de valor y la vulnerabilidad se define como la debilidad que se puede explotar para violar un sistema o la información que contiene. Para ello, se han desarrollado protocolos y mecanismos adecuados, para preservar la seguridad.
2. El **criptoanálisis**, es la ciencia que se encarga de descifrar los mensajes (los intrusos utilizan estas técnicas), mientras que la **criptografía** busca métodos más seguros de cifrado, y se puede clasificar en:
    - Criptografía **clásica**: cifrados rudimentarios basados en sustitución y trasposición
    - Criptografía **moderna**: cifrados basados en algoritmos parametrizados en base a claves
3. "**seguridad de una red**" implica la seguridad de cada uno de las computadoras de la red
4. "**hacker**": cualquier barrera es susceptible de ser superada y tiene como finalidad la de salir de un sistema informático (tras un ataque) sin ser detectado. Es un programador
5. "**cracker**": no es un programado y utiliza sus ataques para sacar beneficio económico
6. **"Amenaza o ataque":** intento de sabotear una operación o la propia preparación para sabotearla (poner en compromiso), que a su vez, estas amenazas se pueden realizar por:
    - **Compromiso**: la entidad atacante obtiene el control de algún elemento interno de la red, por *ejemplo utilizando cuentas con contraseña trivial o errores del sistema*
    - **Modificación**: la entidad atacante modifica el contenido de algún mensaje o texto
    - **Suplantación**: la entidad atacante se hace pasar por otra persona**Reenvío**: la entidad atacante obtiene un mensaje o texto en tránsito y más tarde lo reenvía para duplicar su efecto**Denegación de servicio**: la entidad atacante impide que un elemento cumpla su función

También es importante resaltar, que los temas que vinculan a seguridad son muy peliagudos y están íntimamente relacionados con **temas legales**, los cuales no debemos de dejar de lado. Muestra de ello, es que en muchos gobiernos el uso de información cifrada está prohibido. Los Gobiernos tratan de implantar reglas (o estándares de cifrado) que ellos mismos puedan descifrar fácilmente. Por ejemplo en Francia y EEUU no están permitidas transacciones cifradas, que el gobierno no sea capaz de descifrar, pues pueden utilizarse para comercio de armas, delincuencia, ...

Este capítulo lo vamos a estructurar en cuatro partes:

1.- Secretos: criptografía
2.- Protocolos de seguridad: autenticación y validación
3.- Aplicaciones y seguridad: integridad y no repudio, firmas digitales y certificados
4.- Redes y seguridad

9.1 SECRETOS

Resolución del problema de seguridad del secreto.

La resolución del problema del secreto en la red (y del secreto de los mensajes en cualquier sistema de comunicación), ha estado siempre unido al cifrado (codificación) de los mensajes. Hasta la llegada de las computadoras, la principal restricción del cifrado consistía en la capacidad del empleado encargado de la codificación para realizar las transformaciones necesarias. Otra restricción adicional consistía en la dificultad de cambiar rápidamente el método de cifrado, pues esto implicaba entrenar a una gran cantidad de empleados. Sin embargo, el peligro de que un empleado fuera capturado por el enemigo, etc., ha hecho indispensable la capacidad de cambiar el método de cifrado al instante. De estos requisitos se deriva el modelo de la figura siguiente:

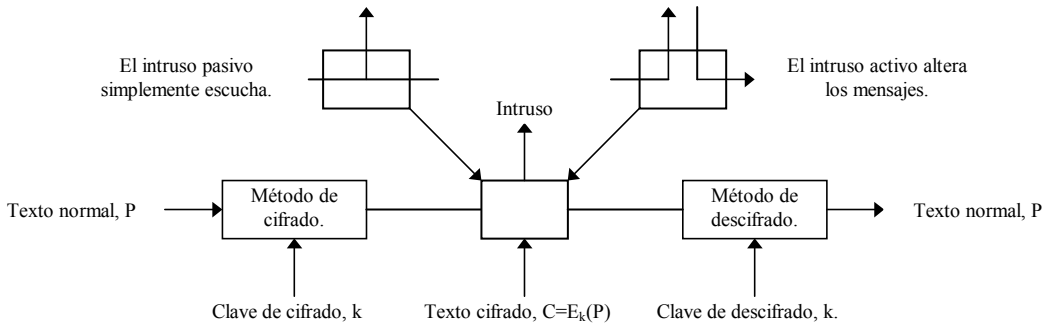


Figura 1 : Modelo de cifrado de datos

Los mensajes a cifrar, conocidos como texto normal, se transforman mediante una función parametrizada por una clave. La salida del cifrado, conocida como texto cifrado, es transmitida después. Si un intruso escucha y copia el texto cifrado, a diferencia del destinatario original, no conoce la clave de cifrado y no puede descifrar fácilmente el texto cifrado.

A partir de aquí usaremos $C=E_k(P)$ para indicar que el cifrado del texto normal P usando la clave K da el texto cifrado C. Del mismo modo $P=D_k(C)$ representa el descifrado de C para obtener el texto normal nuevamente. Por tanto, $D_k(E_k(P))=P$. Esta notación sugiere que E y D son sólo funciones matemáticas de dos parámetros, de los cuales hemos escrito uno (la clave) como subíndice, en lugar de como argumento, para distinguirlo del mensaje.

Todo esto que estamos hablando queda vinculado a la criptografía: KRYPTOS oculto + GRAPHE escrito en grigo.

Una regla fundamental de la criptografía es que se debe suponer que el criptoanalista conoce el método general de cifrado usado, esto es, el criptoanalista conoce E, pues la cantidad de esfuerzo necesario para inventar, probar e instalar un método nuevo cada vez que el viejo es conocido siempre hace impracticable mantenerlo en secreto. Aquí es donde entra la clave. La clave consiste en una cadena relativamente corta que selecciona uno de los muchos cifrados potenciales. En contraste con el método general, que tal vez se cambie cada cierto número de años, la clave puede cambiarse con la frecuencia que se requiera, por lo cual nuestro modelo es un método general estable y conocido públicamente pero parametrizado por una clave secreta y fácilmente cambiable. Un ejemplo de esto es una cerradura de combinación. Todo el mundo conoce como funciona, pero la clave es secreta. Una longitud de clave de tres dígitos significa que existen 1000 posibilidades, una longitud de clave de seis dígitos implica un millón de posibilidades.

Desde el punto de vista del criptoanalista, el problema del criptoanálisis presenta tres variaciones principales:

1.- Cuando el criptoanalista tiene una cierta cantidad de texto cifrado pero no tiene texto común se enfrenta al problema de sólo texto cifrado

2.- Cuando el criptoanalista tiene texto cifrado y el texto normal correspondiente se enfrenta al problema del texto cifrado de texto normal conocido

3.- cuando el criptoanalista tiene la capacidad de cifrar textos normales que puede escoger se enfrenta al problema de texto cifrado de texto normal seleccionado.

Principios criptográficos fundamentales.

Aunque la criptografía es muy variada como veremos a continuación, existen dos principios fundamentales que sostienen la criptografía y que es importante entender.

El primer principio es que todos los mensajes cifrados deben contener redundancia, es decir, información no necesaria para entender el mensaje. Un ejemplo puede dejar claro la necesidad de esto. Considere una compañía de compras por red que tiene 60000 productos. Pensando en la eficiencia, los programadores han decidido que los mensajes de pedidos deben consistir en el nombre del cliente de 16 bytes seguido de un campo de datos de 4 bytes (2 para la cantidad y 2 para el número del producto). Los últimos 4 bytes deben cifrarse usando una clave muy grande conocida solo por el cliente y por la compañía.

Inicialmente esto puede parecer seguro, y lo es, pues los intrusos pasivos no pueden descifrar los mensajes. Sin embargo, también tiene un fallo que lo vuelve inútil. Supóngase que alguien consigue una lista de compradores de productos (los 16 bytes del número de cliente), entonces es fácil trabajar en un programa para generar pedidos ficticios usando nombres reales de los clientes. Dado que no tiene una lista de claves, pone números aleatorios en los últimos 4 bytes y envía cientos de pedidos. Al llegar estos mensajes, la computadora usa el nombre del cliente para localizar la clave y descifrar el mensaje. Para mala suerte, casi todos los mensajes de 4 bytes descifrados son válidos, pues excepto que el pedido sea de cantidad 0 al descifrarlo o corresponda a un producto cuyo código no existe (cuya probabilidad solo es de un 8.5%), el pedido será atendido.

Este problema puede resolverse agregando redundancia a todos los mensajes. Por ejemplo, si se extienden los mensajes de pedido a 12 bytes, de los cuales los 8 primeros deben ser ceros, entonces este ataque ya no funciona, pues la probabilidad de generar un mensaje valido es prácticamente cero. Sin embargo la adición de redundancia simplifica a los criptoanalistas el descifrado de los mensajes, pues ahora un criptoanalista puede saber que ha descifrado correctamente un mensaje al comprobar que los 8 primeros bytes son ceros. Por ello, una cadena aleatoria de palabras sería mejor para incluir en la redundancia.

Otro ejemplo de este primer principio, puede ser el concepto de un CRC (Cyclic Redundant Check), es decir una información redundante puesta al final de un mensaje, evitando al máximo que otros puedan generar información que pueda ser interpretada e introduzca vulnerabilidad al proceso de comunicaciones.

El segundo principio criptográfico es que deben tomarse algunas medidas para evitar que los intrusos activos reproduzcan mensajes viejos. Si no se toman tales medidas, alguien puede conectarse a la línea telefónica de la empresa de pedidos por red y simplemente continuar repitiendo mensajes válidos enviados previamente. Una de tales medidas es la inclusión en cada mensaje de una marca de tiempo válida durante, digamos, 5 minutos. El receptor puede entonces guardar los mensajes unos 5 minutos, para compararlos con los mensajes nuevos que lleguen y filtrar los duplicados. Los mensajes con mayor antigüedad que 5 minutos pueden descartarse, dado que todas las repeticiones enviadas más de 5 minutos después también se rechazarán como demasiado viejas.

A partir de ahora realizaremos un estudio de las técnicas de cifrado, empezando por las técnicas clásicas para pasar a estudiar las técnicas criptográficas modernas.

Criptografía clásica.

Pasemos a analizar los métodos de la criptografía clásica. La criptografía clásica se basa en algoritmos sencillos y claves muy largas para la seguridad. Las técnicas criptográficas clásicas son básicamente dos,

el cifrado por sustitución y el cifrado por transposición. Antes de comenzar su exposición, es necesario exponer el criterio de notación que tomaremos. Tomaremos como texto normal aquel que se encuentra representado por letras minúsculas y como texto cifrado el que se encuentre representado por letras mayúsculas.

Cifrado por sustitución.

El cifrado por sustitución se basa en la sustitución de cada letra o grupo de letras por otra letra o grupo de letras para disfrazarla. Uno de los cifrados por sustitución más antiguos conocidos es el cifrado de Cesar, atribuido al emperador romano Julio Cesar. En este método la letra *a* se convierte en *D*, la *b* en *E*, la *c* en *F*, ... , y *z* se vuelve *C*. Así, el mensaje *ataque* se convierte en *DWDTXH*. Una generalización del cifrado de Cesar permite que el alfabeto de texto cifrado se desplaza k letras en lugar de siempre 3, con lo cual k se convierte en la clave de cifrado.

La siguiente mejora es hacer que cada uno de los símbolos del texto normal, por ejemplo las 26 letras del alfabeto inglés, tengan una correspondencia biunívoca con alguna otra letra. Por ejemplo:

Texto normal: a b c d e f g h i j k l m n o p q r s t u v w x y z
Texto cifrado: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

Este sistema general se llama sustitución monoalfabética, siendo la clave la cadena de 26 letras correspondiente al alfabeto completo. A primera vista, esto podría parecer un sistema seguro, porque aunque el criptoanalista conoce el sistema general (sustitución letra por letra), no sabe cuál de las $26!=4x10^{26}$ claves posibles se está usando. Sin embargo, si se cuenta con una cantidad pequeña de texto cifrado, el cifrado puede descifrarse fácilmente. El ataque básico aprovecha las propiedades estadísticas de los lenguajes naturales. En inglés, la letra *e* es la más común, seguida de *t*, *o*, *a*, *n*, *i*, etc. Las combinaciones de letras más comunes o digramas son *th*, *in*, *er*, *re* y *an*. Las combinaciones de tres letras más comunes o trigramas son *the*, *ing*, *and* e *ion*.

Un criptoanalista que intenta descifrar una codificación monoalfabética comenzaría por contar la frecuencia relativa de todas las letras del texto cifrado. Entonces podría asignar tentativamente la más común a la letra *e* y la siguiente más común a la letra *t*. Vería entonces los trigramas para encontrar uno común de la forma *tXe*, lo que sugerirá que *X* es *h*. De la misma manera si el patrón *thYt* ocurre con frecuencia, *Y* probablemente representa a la letra *a*. Con esta información puede buscar trigramas frecuentes de la forma *aZW*, que con probabilidad es *and*, y acabar de forma similar descifrando el texto cifrado.

Por ejemplo, intentemos descifrar el siguiente fragmento de un poema de Carrol Lewis:

```
kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm
ur mfudm zhx mftnm zhx mdzythc pzq ur ezsszcdm zhx gthcm
zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthcm
sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk
rui mubd ur om zid uok ur sidzkf zhx zyy ur om zid rzk
hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk
```

El porcentaje de ocurrencias de letras, digramas, trigramas y palabras en el idioma ingles es:

Letras		Digramas		Trigramas		Palabras	
E	13.05	TH	3.16	THE	4.72	THE	6.42
T	9.02	IN	1.54	ING	1.42	OF	4.02
O	8.21	ER	1.33	AND	1.13	AND	3.15
A	7.81	RE	1.30	ION	1.00	TO	2.36
N	7.28	AN	1.08	ENT	0.98	A	2.09
I	6.77	HE	1.08	FOR	0.76	IN	1.77
R	6.64	AR	1.02	TIO	0.75	THAT	1.25

S	6.46	EN	1.02	ERE	0.69	IS	1.03
H	5.85	TI	1.02	HER	0.68	I	0.94
D	4.11	TE	0.98	ATE	0.66	IT	0.93
L	3.60	AT	0.88	VER	0.63	FOR	0.77
C	2.93	ON	0.84	TER	0.62	AS	0.76
F	2.88	HA	0.84	THA	0.62	WITH	0.76
U	2.77	OU	0.72	ATI	0.59	WAS	0.72
M	2.62	IT	0.71	HAT	0.55	HIS	0.71
P	2.15	ES	0.69	ERS	0.54	HE	0.71
Y	1.51	ST	0.68	HIS	0.52	BE	0.63
W	1.49	OR	0.68	RES	0.50	NOT	0.61
G	1.39	NT	0.67	ILL	0.47	BY	0.57
B	1.28	HI	0.66	ARE	0.46	BUT	0.56
V	1.00	EA	0.64	CON	0.45	HAVE	0.55
K	0.42	VE	0.64	NCE	0.45	YOU	0.55
X	0.30	CO	0.59	ALL	0.44	WHICH	0.53
J	0.23	DE	0.55	EVE	0.44	ARE	0.50
Q	0.14	RA	0.55	ITH	0.44	ON	0.47
Z	0.09	RO	0.55	TED	0.44	OR	0.45

En nuestro texto, el porcentaje de aparición de letras es:

Z	11.03	T	5.88	S	2.57	L	0.74
D	10.29	H	5.51	B	2.21	Q	0.37
K	8.82	X	3.68	E	2.21	V	0.00
F	8.46	R	3.68	Y	2.21	J	0.00
M	7.72	O	2.94	A	1.84	W	0.00
U	6.62	P	2.57	G	1.10		
I	5.88	C	2.57	N	1.10		

Por lo tanto, comparando tenemos que lo más probable es que E se corresponda con la z o con la d y la T con la otra letra o la k o la f.

Observemos la aparición de palabras. La palabra kfd aparece 5 veces y la palabra zhx aparece 6 veces, entonces, lo más probable es que una corresponda a la palabra THE y la otra corresponda a la palabra AND.

Veamos el análisis letra a letra:

Supongamos que kfd corresponde con AND y zhx corresponde con THE, entonces la frecuencia de aparición de las letras en el texto sería:

Letra	**Porcentaje**	**Sustitución**	**Porcentaje**	**Diferencia**
k	8.82	A	7.81	1.01
f	8.46	N	7.28	1.18
d	10.29	D	4.11	6.18
z	11.03	T	9.02	2.01
h	5.51	H	5.85	0.34
x	3.68	E	13.05	9.37

Si suponemos ahora que kfd corresponde con THE y zhx corresponde con AND, entonces la frecuencia de aparición de las letras en el texto sería:

Letra	**Porcentaje**	**Sustitución**	**Porcentaje**	**Diferencia**
k	8.82	T	9.02	0.20
f	8.46	H	5.85	2.61
d	10.29	E	13.05	2.76
z	11.03	A	7.81	3.22
h	5.51	N	7.28	1.77
x	3.68	D	4.11	0.43

Se observa que en la segunda sustitución, la diferencia de porcentajes es mucho menor que en la primera, por lo cual suponemos como correcta la segunda sustitución, quedando el texto como:

THE TtbE HAm eubE THE pAyiom mAtD Tu TAyg ur bANa THtNcm
ur mHuEm AND mHtnm AND mEAytNc pAq ur eAssAcEm AND gtNcm
AND pHa THE mEA tm sutytNc HuT AND pHETHEi ntcm HAlE ptNcm
soT pAtT A stT THE uamTEim eitED sEruiE pE HAlE uoi eHAT
rui mubE ur om AiE uoT ur siEATH AND Ayy ur om AiE rAT
Nu Hoiia mAtD THE eAinENTEi THEa THANgED Htb boeH rui THAT

Si observamos ahora los trigramas, el trigrama ING debe aparecer con frecuencia, por lo cual, observando el texto encontramos 5 ocurrencias de tNc, por lo cual podemos suponer que la t corresponde a la I y la c a la G. Además, en la tabla siguiente puede observarse que la diferencia en porcentaje de aparición es pequeña.

Letra	Porcentaje	Sustitución	Porcentaje	Diferencia
T	5.88	I	6.77	0.89
C	2.57	G	1.39	1.18

Realizando la sustitución:

THE TIbE HAm eubE THE pAyiom mAID Tu TAyg ur bANa THINGm
ur mHuEm AND mHInm AND mEAyING pAq ur eAssAGEm AND gINGm
AND pHa THE mEA Im suIyING HuT AND pHETHEi nIGm HAlE pINGm
soT pAIT A sIT THE uamTEim eiIED sEruiE pE HAlE uoi eHAT
rui mubE ur om AiE uoT ur siEATH AND Ayy ur om AiE rAT
Nu Hoiia mAID THE eAinENTEi THEa THANgED HIb boeH rui THAT

Continuando, la palabra ur aparece 6 veces, si suponemos corresponde a OF, y con la tabla de apariciones de letras, tenemos:

Letra	Porcentaje	Sustitución	Porcentaje	Diferencia
u	6.62	O	8.21	1.59
r	3.58	F	2.88	0.70

Por lo cual, realizando la sustitución:

THE TIbE HAm eObE THE pAyiom mAID TO TAyg OF bANa THINGm
OF mHOEm AND mHInm AND mEAyING pAq OF eAssAGEm AND gINGm
AND pHa THE mEA Im sOIyING HOT AND pHETHEi nIGm HAlE pINGm
soT pAIT A sIT THE OamTEim eiIED sEFOiE pE HAlE Ooi eHAT
FOi mObE OF om AiE OoT OF siEATH AND Ayy OF om AiE FAT
NO Hoiia mAID THE eAinENTEi THEa THANgED HIb boeH FOi THAT

Si realizamos ahora la comparación entre las letras que aparecen con más frecuencia en el texto y que aún no han sido sustituidas (m el 7.72 e i el 5.88) con las de mayor frecuencia no encontradas todavía (R el 6.64 y S el 6.46), observamos que si sustituimos m por S e i por R, obtenemos palabras como HAS, THINGS, SHOES, ARE, con lo cual, realizando la sustitución:

THE TIbE HAS eObE THE pAyRoS SAID TO TAyg OF bANa THINGS
OF SHOES AND SHInS AND SEAyING pAq OF eAssAGES AND gINGS
AND pHa THE SEA IS sOIyING HOT AND pHETHER nIGS HAlE pINGS
soT pAIT A sIT THE OaSTERS eRIED sEFORE pE HAlE OoR eHAT
FOR SObE OF oS ARE OoT OF sREATH AND Ayy OF oS ARE FAT
NO HoRRa SAID THE eARnENTER THEa THANgED HIb boeH FOR THAT

Realizando un análisis palabra a palabra, podemos reconocer algunas (por ejemplo Ayy debe ser ALL, etc.) y procediendo de esta forma, obtenemos el texto descifrado:

THE TIME HAS COME THE WALRUS SAID TO TALK OF MANY THINGS

```
OF SHOES AND SHIPS AND SEALING WAX OF CABBAGES AND KINGS
AND WHY THE SEA IS BOILING HOT AND WHETHER PIGS HAVE WINGS
BUT WAIT A BIT THE OYSTERS CRIED BEFORE WE HAVE OUR CHAT
FOR SOME OF US ARE OUT OF BREATH AND ALL OF US ARE FAT
NO HURRY SAID THE CARPENTER THEY THANKED HIM MUCH FOR THAT
```

Otro enfoque posible para descifrar el cifrado por sustitución es adivinar una palabra o frase probable. Por ejemplo, dado el siguiente texto cifrado de una compañía contable (mostrado en bloques de cinco caracteres):

```
CTBMN  BYCTC  BTJDS  QXBNS  GSTJC  BTSWX  CTQTZ  CQVUJ
QJSGS  TJQZZ  MNQJS  VLNSX  VSZJU  JDSTS  JQUUS  JUBXJ
DSKSU  JSNTK  BGAQJ  ZBGYQ  TLCTZ  BNYBN  QJSW
```

Una palabra muy probable en un mensaje de una compañía contable en inglés es *financial*. Usando nuestro conocimiento de que *financial* tiene la letra i repetida, con cuatro letras intermedias entre su aparición, buscamos letras repetidas en el texto cifrado con este espaciado. Encontramos 12 casos, en las posiciones 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 y 82. Sin embargo, sólo dos de éstos, el 31 y el 42, tiene la siguiente letra (correspondiente a *n* en el texto normal) repetida en el lugar adecuado. De estos dos, sólo el 31 tiene también la *a* en la posición correcta, por lo que sabemos que *financial* comienza en la posición 30. A partir de aquí, la deducción de la clave es fácil usando las estadísticas de frecuencia del texto en inglés.

Otros métodos de cifrado por sustitución son:

- Cifrado de **Polybius**, se introduce el alfabeto dentro de una tabla por filas, donde la table tiene un número de columnas determinado, este número de columnas es la clave conociendo el algoritmo. El texto normal se codifica en base a las coordenadas de las letras del texto normal dentro de dicha tabla. La clave de este cifrado está en la disposición del alfabeto en la tabla, es decir el número de columnas de la tabla.

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	A	B	C	D	E	F	g
2	H	I	J	K	L	M	N
3	Ñ	O	P	Q	R	S	t
4	U	V	W	X	Y	Z	+

  Si P=HOLA, entonces el cifrado consiste en codificar *(fila, columna)* por tanto el cifrado es (2,1), (3,2), (2,5), (1,1)

- Cifrado de **Trithemius,** es un método de sustitución progresivo, basado en el cifrado de Julio Cesar, donde el valor de k varía incrementalmente de forma conocida en los extremos. Ejemplo: clave k=+2 y texto normal P="Hola" *=>H(+2)=J, o(+3)=r, l(+4)=o, a(+5)=f:: p*or tanto el texto cifrado sería *C="Jrof"*

Cifrado por transposición.

Los cifrados por sustitución conservan el orden de los símbolos de texto normal, pero los disfrazan. Los cifrados por transposición en contraste, reordenan las letras pero no las disfrazan. Un ejemplo de cifrado por transposición es la transposición columnar. La clave del cifrado es una palabra o frase que no contiene letras repetidas. En este ejemplo, la clave es MEGABUCK. El propósito de la clave es numerar las columnas, estando la columna 1 bajo la letra clave más cercana al inicio del alfabeto, y así sucesivamente. El texto normal se escribe horizontalmente en filas, el texto cifrado se lee por columnas (o también por filas según el método empleado), comenzando por la columna cuya letra clave es la más baja.

```
M E G A B U C K
7 4 5 1 2 8 3 6
```

Texto normal:

```
p l e a s e t r
a n s f e r o n
e m i l l i o n
d o l l a r s t
o m y s w i s s
b a n k a c c o
u n t s i x t w
o t w o a b c d
```

```
pleasetransferonemilliondollarstom
yswissbankaccountsixtwotwo
```

Texto cifrado:

```
AFLLSKSOSELAWAIATOOSSCTCLNMOMANTES
ILYNTWRNNTSOWDPAEDOBUOERIRICXB
```

Para descifrar un cifrado por transposición, el criptoanalista debe primero ser consciente de que está tratando con un cifrado por transposición. Observando la frecuencia de *E*, *T*, *A*, *O*, *I*, *N*, etc., es fácil ver si se ajustan al patrón usual del texto normal. De ser así, es evidente que se trata de un cifrado por transposición, pues en tal cifrado cada letra se representa a sí misma.

El siguiente paso es adivinar la cantidad de columnas. En muchos casos, puede adivinarse una palabra o frase probable por el contexto del mensaje. Por ejemplo, supóngase que nuestro criptoanalista sospecha que la frase de texto normal *milliondolars* aparece en algún lugar del mensaje. Observe que los diagramas *MO*, *IL*, *LL*, *LA*, *IR* y *OS* ocurren en el texto cifrado como resultado de que esta frase da la vuelta. Si se hubiera usado una clave de longitud siete, habrían ocurrido los diagramas *MD*, *IO*, *LL*, *LL*, *IA*, *OR* y *NS*. De hecho, para cada longitud de clave, se produce un grupo diferente de diagramas de texto cifrado. Buscando las diferentes posibilidades, el criptoanalista con frecuencia puede determinar fácilmente la longitud de la clave.

El paso restante es ordenar las columnas. Cuando la cantidad de columnas, k, es pequeña, puede examinarse cada uno de los pares de columnas k(k-1) para ver si la frecuencia de sus diagramas es igual a la del texto normal. El par con mejor concordancia se supone correctamente ubicado. Ahora cada columna restante se prueba tentativamente como el sucesor de este par. La columna cuyas frecuencias de digramas y trigramas produce la mejor concordancia se toma tentativamente como correcta. La columna antecesora se encuentra de la misma manera. El proceso completo se repite hasta encontrar un orden potencial. Es probable que el texto normal sea reconocible en algún punto (por ejemplo, sí aparece *milloin*, quedará claro dónde está el error).

Por ejemplo, intentemos descifrar el siguiente fragmento de un libro de texto sobre computación, por lo que "computer" es una palabra de probable aparición. El texto normal consiste por entero en letras (sin espacios). El texto cifrado se dividió en bloques de cinco caracteres para hacerlo más legible.

```
aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo
gorch srsoc nntii imiha oofpa gsivt tpsit lbolr otoex
```

Si la palabra “computer” aparece en el texto y la transposición columnar es de clave menor que 8, entonces, según el caso deberían aparecer juntas las siguientes letras de la palabra “computer” según el tamaño de la clave

Tamaño	**Letras**	**Tamaño**	**Letras**	**Tamaño**	**Letras**
2	CM	3	CP	4	CU
5	CT	6	CE	7	CR

Observando el texto vemos que aparece CE, por lo cual el tamaño de la clave es de tamaño 6. Ordenando el texto en seis columnas, obtenemos:

```
ADIGIT
ALCOMP
UTERIS
AMACHI
NETHAT
CANSOL
VEPROB
LEMSFO
```

```
RPEOPL
EBYCAR
RYINGO
UTINST
RUCTIO
NSGIVE
NTOITX
```

Que directamente puede leerse, sin necesidad de realizar ninguna transposición columnar como:

```
A DIGITAL COMPUTER IS A MACHINE THAT CAN SOLVE PROBLEMS FOR
PEOPLE BY CARRYING OUT INSTRUCTIONS GIVE TO IT
```

Rellenos de una sola vez.

La construcción de un cifrado inviolable en realidad es bastante sencilla. La técnica se conoce desde hace décadas y consiste en escoger una cadena de bits al azar como clave. Luego se convierte el texto normal en una cadena de bits, por ejemplo usando su representación ASCII. Por último, se calcula el or exclusivo, conocido como XOR y cuya tabla de valores lógicos puede verse a continuación.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

El texto cifrado resultante no puede descifrarse porque cada texto normal posible es un candidato igualmente probable. El texto cifrado no proporciona al criptoanalista ninguna información en absoluto. En una muestra suficientemente grande de texto cifrado, cada letra ocurrirá con la misma frecuencia, al igual que cada digrama (combinación de dos letras) y cada trigrama (combinación de tres letras). Como ejemplo de cifrado basado en relleno de una sóla vez, cifremos el mensaje "texto cifrado" con la cadena "En un lugar de la Mancha de cuyo nombre…"

Texto original	t	e	x	t	o		c	i	f	r	a	d	o
Codificación ASCII (hex)	74	65	78	74	6F	20	63	69	66	72	61	64	6F
Texto de cifrado	E	n		u	n		l	u	g	a	r		d
Codificación ASCII (hex)	45	6E	20	75	6E	20	6C	75	67	61	72	20	64
Codificación cifrada (hex)	*31*	*0B*	*58*	*01*	*01*	*00*	*0F*	*1C*	*01*	*13*	*13*	*44*	*08*

Si procedemos ahora a descifrarlo con la clave de codificación, obtenemos el mensaje original ya que aplicamos la función XOR 2 veces:

Codificación cifrada (hex)	*31*	*0B*	*58*	*01*	*01*	*00*	*0F*	*1C*	*01*	*13*	*13*	*44*	*08*
Texto de cifrado	E	n		u	n		l	u	g	a	r		d
Codificación ASCII (hex)	45	6E	20	75	6E	20	6C	75	67	61	72	20	64
Codificación ASCII (hex)	74	65	78	74	6F	20	63	69	66	72	61	64	6F
Texto original	t	e	x	t	o		c	i	f	r	a	d	o

Sin embargo, este método tiene varias desventajas prácticas. En primer lugar, la clave no puede memorizarse, por lo que tanto el transmisor como el receptor deben llevar una copia por escrito consigo. Además, la cantidad total de datos que pueden transmitirse está limitada a la cantidad de clave disponible. Otro problema es la sensibilidad del método a la pérdida o inserción de caracteres. Si el transmisor y el receptor pierden la sincronía, todos los datos a partir de ahí aparecerán alterados.

Criptografía moderna.

La criptografía moderna usa las mismas ideas básicas que la criptografía tradicional, la transposición y la sustitución, pero su orientación es distinta. Mientras la criptografía tradicional usaba algoritmos sencillos y claves muy largas para la seguridad, hoy en día es cierto la afirmación contraria: el objetivo es hacer algoritmos de cifrado tan complicados y rebuscados que incluso si el criptoanalista obtiene cantidades enormes de texto cifrado a su gusto, no será capaz de entender nada y por tanto de descifrarlo.

Las transposiciones y sustituciones pueden implantarse mediante circuitos sencillos. En la figura siguiente se muestran dos dispositivos conocidos como caja P, que se usa para efectuar una transposición de una entrada de 12 bits; y otro dispositivo conocido como caja S, en el cual ingresa un texto normal de 3 bits y sale un texto cifrado de 3 bits. La potencia real de estos elementos básicos sólo se hace aparente cuando ponemos en cascada una serie completa de estas cajas para formar un cifrado de producto como podemos ver en la figura siguiente.

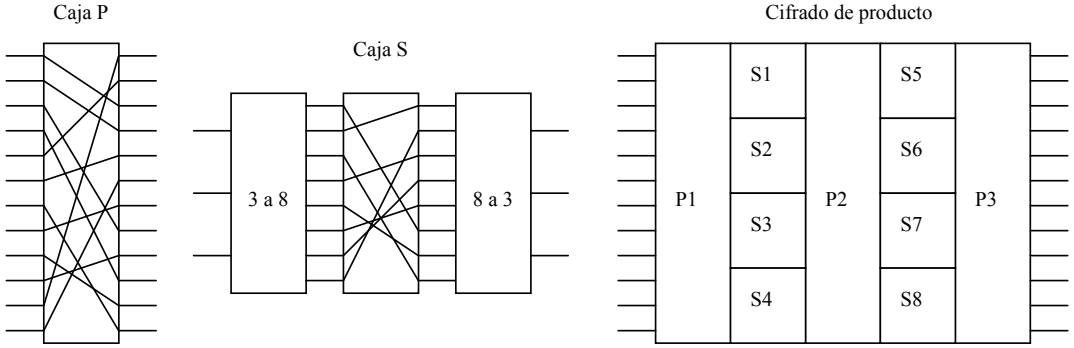


Figura 2: Ejemplo de cifrado de producto mediante cajas P y cajas S.

El cifrado moderno se divide actualmente en cifrado de clave privada y cifrado de clave pública:

- en el cifrado de clave privada las claves de cifrado y descifrado son la misma (o bien se deriva de forma directa una de la otra), debiendo mantenerse en secreto dicha clave. **Ejemplo**: DES (Data Encryption Standar), DES triple e IDEA (International Data Encryption Algorithm). El cifrado de clave privada, es más rápido que el de clave pública (de 100 a 1000 veces), y por tanto se utiliza generalmente en el *intercambio de información dentro de una sesión*. Estas claves también son conocidas como claves de sesión o de cifrado simétricas, ya que en ambos extremos se posee la misma clave.

- en el cifrado de clave pública, las claves de cifrado y descifrado son independientes, no derivándose una de la otra, por lo cual puede hacerse pública la clave de cifrado siempre que se mantenga en secreto la clave de descifrado. **Ejemplo**: Cifrado RSA (Rivest, Shamir, Adleman).El cifrado de clave pública es más lento y por tanto *se utiliza para intercambiar las claves de sesión*. Como este algoritmo utiliza dos claves diferentes, una privada y otra pública el cifrado se conoce como cifrado asimétrico

Veremos a continuación algunos algoritmos de cifrado modernos.

Cifrado con clave privada (simétrica)

Los métodos de cifrado de clave privada, también son conocidos como de clave simétrica, porque ambos extremos conocen la clave para poder descifrar.

Cifrado DES (Data Encryption Standar).

El algoritmo DES es un algoritmo de cifrado de clave privada desarrollado por IBM a principios de la década de los 70 a partir de otro algoritmo conocido como Lucifer que utilizaba claves de 112 bits y que

fueron reducidas a 56 bits en el algoritmo DES. La reducción del tamaño de las claves, propuesta por la NSA (Agencia Nacional de Seguridad) originó controversia debido a que sé penso que la NSA había debilitado intencionadamente el algoritmo del DES para poder desencriptarlo.

Dicha controversia llegó a su fin cuando en 1994 IBM publicó un artículo describiendo los criterios del desarrollo del algoritmo DES. El artículo indica que el diseño se realizó de forma que el algoritmo DES fuera resistente a criptoanálisis, pero lo suficientemente sencillo como para poder ser implementado en un circuito electrónico con la tecnología de principios de los 70. Por tanto, el algoritmo DES se diseño de forma que no pudiera ser descifrado por criptoanálisis, pero si que puede ser descifrado probando todas las claves posibles, asumiendo que se cuenta con el hardware adecuado.

En la siguiente figura se muestra un esbozo del algoritmo DES. El texto normal se cifra en bloques de 64 bits, produciendo 64 bits de texto cifrado. El algoritmo, que se parametriza con la clave de 56 bits, tiene 19 etapas diferentes.

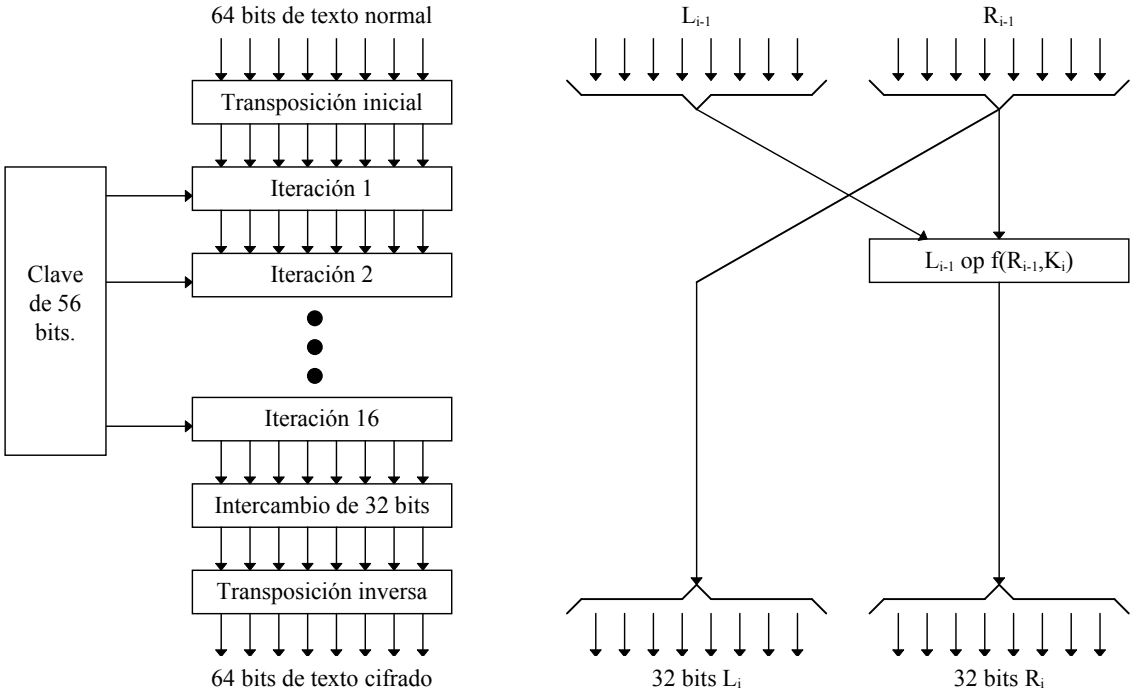


Figura 3: Algoritmo del cifrado DES.

La primera etapa es una transposición, independiente de la clave, del texto normal de 64 bits. La última etapa es el inverso exacto de esta transposición. La etapa previa a la última intercambia los 32 bits de la izquierda y los 32 bits de la derecha. Las 16 etapas restantes son funcionalmente idénticas, pero se parametrizan mediante diferentes funciones de la clave. El algoritmo se ha diseñado para permitir que el descifrado se haga con la misma clave que el cifrado, simplemente ejecutando los pasos en orden inverso. Cada una de las 16 etapas intermedias toma dos entradas de 32 bits y produce dos salidas de 32 bits. La salida de la izquierda es simplemente una copia de la entrada de la derecha. La salida de la derecha es el or exclusivo a nivel de bit de la entrada izquierda y una función de la entrada derecha y la clave de esta etapa K_i. Toda la complejidad reside en esta función.

La función consiste en cuatro pasos, ejecutados en secuencia. Primero se construye un número de 48 bits, E, expandiendo el R_{i-1} de 32 bits según una regla fija de transposición y duplicación. Después, se aplica un or exclusivo a E y K_i. Esta salida entonces se divide en ocho grupos de 6 bits, cada uno de los cuales se alimenta a una caja S distinta. Cada una de las 64 entradas posibles a la caja S se transforma en una salida de 4 bits. Por último estos 8x4 bits se pasan a través de una caja P.

En cada una de las 16 iteraciones, se usa una clave diferente. Antes de iniciarse el algoritmo, se aplica una transposición de 56 bits a la clave. Justo antes de cada iteración, la clave se divide en dos unidades de

28 bits, cada una de las cuales se gira hacia la izquierda una cantidad de bits dependiente del número de iteración. K_i se deriva de esta clave girada aplicándole otra transposición de 56 bits. En cada vuelta (o iteración) se extrae y permuta de los 56 bits un subgrupo de 48 bits diferente.

Descifrado del DES.

Aunque en un principio el DES parecía inviolable, dos investigadores de criptografía de Standford diseñaron en 1977 una máquina para violar el DES y estimaron que podría construirse por unos 20 millones de dólares. Dado un trozo pequeño de texto normal y el texto cifrado correspondiente, esta máquina podría encontrar la clave mediante una búsqueda exhaustiva del espacio de claves de 2^{56} en menos de un día. Otros autores han diseñado máquinas capaces de descifrar el DES mediante búsqueda exhaustiva en unas cuatro horas.

Otra estrategia similar es descifrar el DES por software. Aunque el cifrado es 1.000 veces más lento por software que por hardware, una computadora casera de alto nivel aún puede efectuar unos 250.000 cifrados/segundo en software, y es probable que este ociosa unos 2 millones de segundos al mes. Este tiempo de inactividad podría usarse para descifrar el DES. Si alguien pusiera un mensaje en uno de los grupos de noticias de Internet, no debería ser difícil conseguir las 140.000 personas necesarias para revisar las $7x10^{16}$ claves en un mes.

Probablemente la idea más innovadora para descifrar el DES es la lotería china. En este diseño, cada radio y televisión tiene que equiparse con un chip DES barato capaz de realizar 1 millón de cifrados/segundo en hardware. Suponiendo que cada una de los 1200 millones de personas de China tiene una radio o televisor, cada vez que el gobierno quiera descifrar un mensaje codificado DES, simplemente difunde el par texto normal/texto cifrado y cada uno de los 1200 millones de chips comienzan a buscar en su sección preasignada del espacio de claves. En 60 segundos se encontrarán una o más correspondencias. Para asegurar que se informe, los chips podrían programarse para anunciar un mensaje de que se ha ganado un premio informado a la autoridad competente.

La conclusión que se puede obtener de estos argumentos es que el DES no es seguro. Sin embargo, surge la idea de ejecutar el DES dos veces, con dos claves de 56 bits distintas. Esto proporciona un espacio de 2^{112}, esto es, $5x10^{33}$, lo cual provocaría que el ataque de la lotería china requiriese 100 millones de años. Sin embargo dos investigadores han desarrollado un método que hace sospechoso al doble cifrado. El método de ataque se llama encuentro a la mitad y funciona como sigue. Supóngase que alguien ha cifrado doblemente una serie de bloques de texto normal usando el modo de libro de código electrónico. Para unos pocos valores de i, el criptoanalista tiene pares igualados (P_i,C_i) donde:

$$C_i=E_{k2}(E_{k1}(P_i))$$

Si ahora aplicamos la función de descifrado D_{k2} a cada lado de la ecuación, obtenemos:

$$D_{k2}(C_i)=E_{k1}(P_i)$$

Porque el cifrado de x y su descifrado posterior con la misma clave produce x.

El ataque de encuentro a la mitad usa esta ecuación para encontrar las claves DES, K_1 y K_2, como sigue:

1. Calcular $R_i=E(P_1)$ para los 2^{56} valores de i, donde E es la función de cifrado DES. Ordenar esta tabla en orden ascendente según R_i.

2. Calcular $S_j=Dj(C_1)$ para todos los 2^{56} valores de j, donde D es la función de descifrado DES. Ordenar esta tabla en orden ascendente según S_j.

3. Barrer la primera tabla en busca de un R_i igual a algún S_j de la segunda tabla. Al encontrar un par, tenemos un par de claves (i,j) tal que $D_j(C_1)=E_i(P_1)$. Potencialmente i es K1 y j es K2.

4. Comprobar si $E_j(E_i(P_2))$ es igual a C_2. Si lo es, intentar todos los demás pares (texto normal, texto cifrado). De no serlo, continuar buscando pares en las dos tablas.

Ciertamente ocurrirán muchas falsas alarmas antes de encontrar las claves reales, pero tarde o temprano se encontrarán. Este ataque requiere sólo 2^{57} operaciones de cifrado o descifrado (para construir las dos tablas), solamente el doble que 2^{56}. Sin embargo requiere un total de 2^{60} bytes de almacenamiento para las dos tablas, por lo que actualmente no es factible en su forma básica, aunque se han desarrollado varias optimizaciones y concesiones que permiten menos almacenamiento a expensas de más computo. En conclusión, el cifrado doble usando el DES tampoco es mucho más seguro que el cifrado sencillo.

Cifrado DES triple.

Como hemos visto, el cifrado realizado con el algoritmo DES es descifrable mediante diversos tipo de ataques de fuerza bruta, hecho que corroboro IBM en su artículo de 1994, indicando de forma explícita que la NSA decidió que así fuera con el fin de poder desencriptar los mensajes que deseara.

Parece, por tanto, que el cifrado con el algoritmo DES no es seguro, sin embargo, el cifrado triple con el algoritmo DES es otro asunto. El método seleccionado, que se ha incorporado al estándar internacional 8732, se ilustra en la siguiente figura.

Figura 4: Esquema del cifrado DES triple.

Aquí se usan dos claves y tres etapas. En la primera etapa el texto normal se cifra con K_1. En la segunda etapa el algoritmo DES se ejecuta en modo de descifrado, usando K_2 como clave. Por último, se hace otro cifrado usando K_1. El hecho de que se usen dos claves y en modo EDE (cifrado-descifrado-cifrado) en lugar de EEE (cifrado-cifrado-cifrado) es debido a dos motivos:

1. En primer lugar, los criptógrafos admiten que 112 bits son suficientes para las aplicaciones comerciales por ahora. Subir a 168 bits (3 claves) simplemente agregaría carga extra innecesaria de administrar y transportar otra clave.

2. En segundo lugar, la razón de cifrar, descifrar y luego cifrar de nuevo es la compatibilidad con los sistemas DES de una sola clave. Tanto las funciones de cifrado como de descifrado son correspondencias entre números de 64 bits. Desde el punto de vista criptográfico, las dos correspondencias son igualmente robustas. Sin embargo, usando EDE en lugar de EEE, una computadora que usa cifrado triple puede hablar con otra que usa cifrado sencillo simplemente estableciendo $K_1=K_2$. Esta propiedad permite la introducción gradual del cifrado triple.

Cifrado IDEA (International Data Encryption Algorithm).

Después de comprobar la debilidad del algoritmo DES en su forma simple, diversos trabajos propusieron nuevos métodos de cifrados de bloques (BLOWFISH, Crab, FEAL, KHAFRE, LOKI91, NEWDES, REDOCII, SAFER K64). Sin embargo el más interesante e importante de los cifrados posteriores al algoritmo DES es el algoritmo IDEA, algoritmo internacional de cifrado de datos.

El algoritmo IDEA es un algoritmo de clave privada que fue diseñado por dos investigadores en Suiza, usa una clave de 128 bits, lo que lo hará inmune durante décadas a los ataques de la fuerza bruta, la lotería china y a los ataques de encuentro a la mitad. No hay ninguna técnica o máquina conocida actualmente que se crea que puede descifrar el algoritmo IDEA.

La estructura básica del algoritmo se asemeja al algoritmo DES en cuanto a que se alteran bloques de entrada de texto normal de 64 bits en una secuencia de iteraciones parametrizadas para producir bloques de salida de texto cifrado de 64 bits, como se puede ver en la figura siguiente.

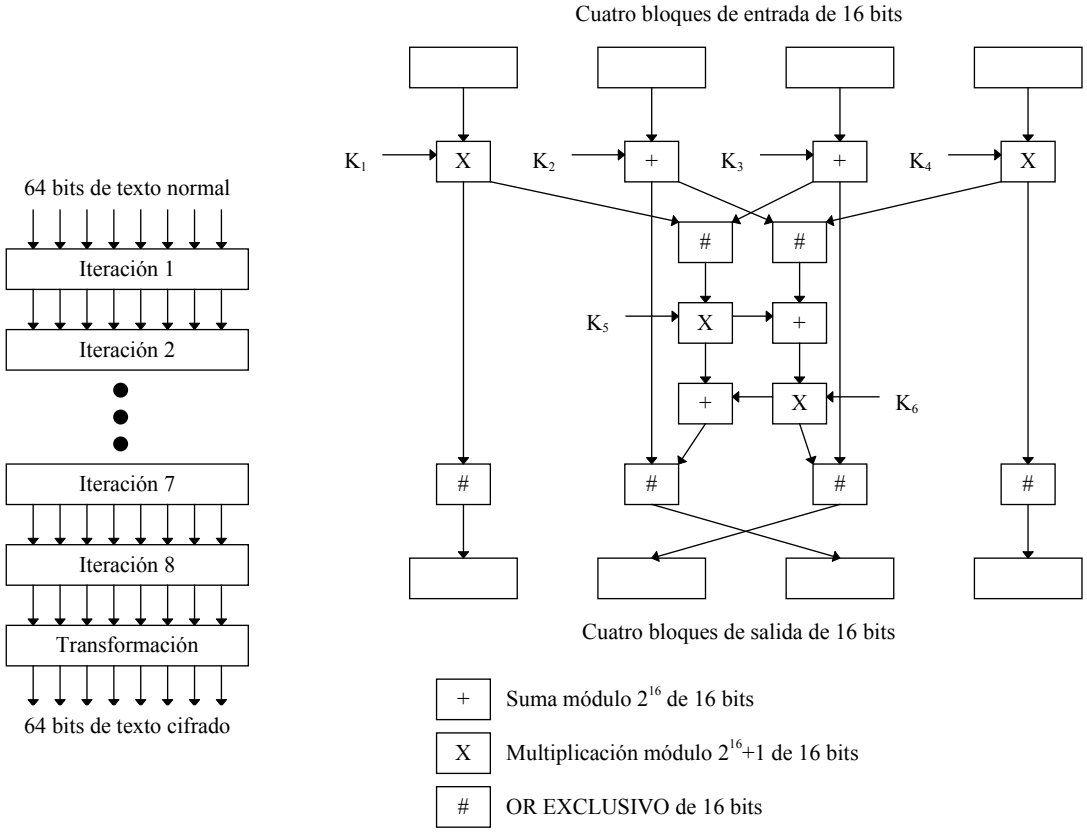


Figura 5: Algoritmo del cifrado IDEA.

Dada la extensa alteración de bits (por cada iteración, cada uno de los bits de salida depende de cada uno de los bits de entrada), basta con ocho iteraciones. Como con todos los cifrados de bloque, el algoritmo IDEA también puede usarse en el modo de realimentación de cifrado y en los demás modos del algoritmo DES. El algoritmo IDEA usa tres operaciones, todas sobre números sin signo de 16 bits. Estas operaciones son un or exclusivo, suma módulo 2^{16} y multiplicación módulo $2^{16}+1$. Las tres operaciones se pueden efectuar fácilmente en una microcomputadora de 16 bits ignorando las partes de orden mayor de los resultados. Las operaciones tienen la propiedad de que ningunos dos pares obedecen la ley asociativa ni la ley distributiva, dificultando el criptoanálisis. La clave de 128 bits se usa para generar 52 subclaves de 16 bits cada una, 6 por cada una de las ocho iteraciones y 4 para la transformación final. El descifrado usa el mismo algoritmo que el cifrado, sólo que con subclaves diferentes.

Cifrado AES (Advanced Encryption Estándar) o Rijndael.

Es considerado el sucesor de DES. Este algoritmo se adoptó oficialmente en octubre del 2000 como nuevo Estándar Avanzado de Cifrado (AES) por el NIST (National Institute for Standards and Technology) para su empleo en aplicaciones criptográficas.

Sus autores son dos, los belgas Joan Daemen y Vincent Rijmen, de ahí su nombre Rijndael. Tiene como peculiaridad que todo el proceso de selección, revisión y estudio tanto de este algoritmo como de los restantes candidatos, se efectuó de forma pública y abierta, por lo que, toda la comunidad criptográfica mundial ha participado en su análisis, lo cual convierte a Rijndael en un algoritmo perfectamente digno de la confianza de todos.

AES es un sistema de cifrado por bloques, diseñado para manejar longitudes de clave variables, comprendidas entre los 128 y los 256 bits.

AES es un algoritmo que se basa en aplicar un número determinado de rondas a un valor intermedio que se denomina estado. Dicho estado puede representarse mediante una matriz rectangular de bytes, que posee cuatro filas, y Nb columnas. Así, por ejemplo, si nuestro bloque tiene 160 bits (tabla 5), Nb será igual a 5.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$

Tabla 5: Ejemplo de matriz de estado con N_b=5 (160 bits).

La clave tiene una estructura análoga a la del estado, y se representará mediante una tabla con cuatro filas y N_k columnas. Si nuestra clave tiene, por ejemplo, 128 bits, N_k será igual a 4 (tabla 6).

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Tabla 6: Ejemplo de clave con N_k=4 (128 bits).

El bloque que se pretende cifrar o descifrar se traslada directamente byte a byte sobre la matriz de estado, siguiendo la secuencia $a_{0,0}$, $a_{1,0}$, $a_{2,0}$, $a_{3,0}$, $a_{0,1}$…, y análogamente, los bytes de la clave se copian sobre la matriz de clave en el mismo orden, a saber, $k_{0,0}$, $k_{1,0}$, $k_{2,0}$, $k_{3,0}$, $k_{0,1}$….

Siendo *B* el bloque que queremos cifrar, y *S* la matriz de estado, el algoritmo AES con *n* rondas queda como sigue:

1. Calcular K_0,K_1, …K_n subclaves a partir de la clave K.

2. $S \leftarrow B \oplus K_0$

3. Para i = 1 hasta n hacer

4. Aplicar ronda i-ésima del algoritmo con la subclave Ki.

El *algoritmo de descifrado* consistirá en aplicar las inversas de cada una de las funciones en el orden contrario, y utilizar los mismos Ki que en el cifrado, sólo que comenzando por el último.

Es un algoritmo resistente al criptoanálisis tanto lineal como diferencial y uno de los más seguros en la actualidad ya que para recuperar la clave a partir de un par texto cifrado-texto claro hay que realizar una búsqueda exhaustiva.

Las Rondas de AES

Puesto que AES permite emplear diferentes longitudes tanto de bloque como de clave, el número de rondas requerido en cada caso es variable. En la tabla 7 se especifica cuantas rondas son necesarias en función de Nb y Nk.

	N_b=4 (128 bits)	N_b=6 (192 bits)	N_b=8 (256 bits)
N_k=4 (128 bits)	10	12	14
N_k=6 (192	12	12	14

bits)			
N_k=8 (256 bits)	14	14	14

Tabla 7: Número de rondas para AES en función de los tamaños de clave y bloque.

Siendo S la matriz de estado, y K_i la subclave correspondiente a la ronda i-ésima, cada una de las rondas posee la siguiente estructura:

1. S ← ByteSub(S)

2. S ← DesplazarFila(S)

3. S ← MezclarColumnas(S)

4. S ← Ki ⊕ S

Estas cuatro funciones están diseñadas para proporcionar resistencia frente a criptoanálisis lineal y diferencial. Cada una de las funciones tiene un propósito:

- Funciones DesplazarFila y MezclarColumnas permiten obtener un alto nivel de difusión a lo largo de varias rondas.
- Función ByteSub consiste en la aplicación paralela de s-cajas.
- La capa de adición de clave es un simple or-exclusivo entre el estado intermedio y la subclave correspondiente a cada ronda.

La última ronda es igual a las anteriores, pero eliminando el paso 3.

Función ByteSub

La transformación ByteSub es una sustitución no lineal que se aplica a cada byte de la matriz de estado, mediante una s-caja 8*8 invertible, que se obtiene componiendo dos transformaciones:

1. Cada byte (2^8) genera el polinomio irreducible $m(x) = x^8 + x^4 + x^3 + x + 1$, y sustituido por su inversa multiplicativa. El valor cero queda inalterado.

2. Después se aplica la siguiente transformación afín en GF(2), siendo $x_0, x_1, \ldots, x_7$ los bits del byte correspondiente, e $y_0, y_1, \ldots, y_7$ los del resultado:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

La función inversa de ByteSub sería la aplicación de la inversa de la s-caja correspondiente a cada byte de la matriz de estado.

Función DesplazarFila

Esta transformación consiste en desplazar a la izquierda cíclicamente las filas de la matriz de estado. Cada fila f_i se desplaza un número de posiciones c_i diferente. Mientras que c_0 siempre es igual a cero (esta fila siempre permanece inalterada), el resto de valores viene en función de N_b y se refleja en la tabla 8.

N_b	c_1	c_2	c_3
4	1	2	3
6	1	2	3
8	1	3	4

Tabla 8: Valores de c_i según el tamaño de bloque N_b

La función inversa de DesplazarFila será, obviamente, un desplazamiento de las filas de la matriz de estado el mismo número de posiciones que en la tabla 2.5, pero a la derecha.

Función MezclarColumnas

Para esta función, cada columna del vector de estado se considera un polinomio cuyos coeficientes pertenecen a GF(28) y se multiplica módulo $x^4 + 1$ por:

$$c(x) = 03x^4 + 01x^2 + 01x + 02$$

donde 03 es el valor hexadecimal que se obtiene concatenando los coeficientes binarios del polinomio correspondiente en GF(28), en este caso 00000011, o sea, x+1, y así sucesivamente.

La inversa de MezclarColumnas se obtiene multiplicando cada columna de la matriz de estado por el polinomio:

$$d(x) = 0Bx^4 + 0Dx^2 + 09x + 0E$$

Cálculo de las Subclaves

Las diferentes subclaves Ki se derivan de la clave principal K mediante el uso de dos funciones: una de expansión y otra de selección. Siendo n el número de rondas que se van a aplicar, la función de expansión permite obtener, a partir del valor de K, una secuencia de $4*(n+1)*N_b$ bytes. La selección simplemente toma consecutivamente de la secuencia obtenida bloques del mismo tamaño que la matriz de estado, y los va asignando a cada Ki.

Sea K(i) un vector de bytes de tamaño $4*N_k$, conteniendo la clave, y sea W(i) un vector de $N_b*(n+1)$ registros de 4 bytes, siendo n el número de rondas. La función de expansión tiene dos versiones, según el valor de N_k:

a) Si Nk ≤ 6:

1. Para i desde 0 hasta N_k - 1 hacer
2. W(i) ← (K(4 * i), K(4 * i + 1), K(4 * i + 2), K(4 * i + 3))
3. Para i desde Nk hasta Nb * (n + 1) hacer
4. tmp ← W(i - 1)
5. Si i mod Nk = 0
6. tmp ← Sub(Rot(tmp)) ⊕ R(i/Nk)
7. W(i) ← W(i - Nk) ⊕ tmp

b) Si Nk > 6:

1. Para i desde 0 hasta Nk - 1 hacer
2. W(i) ← (K(4 * i), K(4 * i + 1), K(4 * i + 2), K(4 * i + 3))
3. Para i desde Nk hasta Nb * (n + 1) hacer
4. tmp ← W(i - 1)
5. Si i mod Nk = 0
6. tmp ←Sub(Rot(tmp)) ⊕ $R_c(i/N_k)$
7. Si i mod Nk = 4
8. tmp ← Sub(tmp)
9. W(i) ← W(i - N_k) ⊕ tmp

En los algoritmos anteriores, la función Sub devuelve el resultado de aplicar la s-caja de AES a cada uno de los bytes del registro de cuatro que se le pasa como parámetro. La función Rot desplaza a la izquierda una posición los bytes del registro, de tal forma que si le pasamos como parámetro el valor (a, b, c, d) nos devuelve (b, c, d, a). Finalmente, $R_c(j)$ es una constante definida de la siguiente forma:

- $R_c(j) = (R(j), 0, 0, 0)$
- Cada R(i) es el elemento de $GF(2^8)$ correspondiente al valor $x^{(i-1)}$.

Otros cifrados simétricos: métodos de bloque y flujo

En general, estos métodos simétricos tienen el inconveniente de la distribución de claves. La ventaja es que son rápidos.

Otro problema que aparece, es el procesamiento de la información por bloques de un tamaño conocido. Por ejemplo, DES procesa a bloques de 64 bits (8 bytes), lo cual con un intruso avispado, podría hacer movimiento de bloques dentro del mismo cifrado, sin levantar alarma. Otros cifrados de bloque que también sufren este problema son RC5, Blowfish

El problema de trabajar con bloques de 64 bytes se resuelve por **diferentes métodos**:
•*basados en **flujo**, que operan por bloques, pero convolucionando (por ejemplo con una XOR) la salida actual con salidas anteriores. Ejemplos: RC2, RC4 y CAST.*
•*con **rellenos variables** por bloque, por ejemplo insertando 0s e indicando la cantidad de rellenos*
•*con **algoritmos de bloque y clave variable** como el AES (Advanced Encryption Standard)*

Circuitos y hardware asociado.

En la actualidad existe hardware utilizado para cifrar comunicaciones a nivel de red y/o enlace. Ejemplo de ello son las Tarjetas Advancde Integration Module (AIM) para diferente gama de routers Cisco.

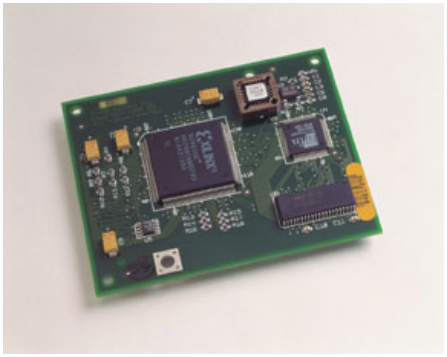


Figura 6: Tarjeta 2600 AIM 2 Mbps

Cifrado con clave pública (asimétrica)

Los métodos de cifrado de clave pública, también son conocidos como de clave asimétrica, porque son algoritmos que se basan en un par de claves, una pública que dispone todo el mundo y una clave asociada a dicha clave pública, que es privada y que guarda el usuario encarecidamente.

Cifrado con clave pública RSA (Rivest, Shamir, Adleman)

Históricamente, el problema de distribución de claves siempre ha sido la parte débil de la mayoría de criptosistemas. Sin importar lo robusto que sea el criptosistema, si un intruso puede robar la clave, el sistema no vale nada.

En 1976, dos investigadores de la Universidad de Stanford propusieron una clase nueva de criptosistema, en el que las claves de cifrado y descrifrado eran diferentes y la clave de descifrado no podía derivarse de la clave de cifrado. En su propuesta, el algoritmo de cifrado (con clave), E, y el algoritmo de descifrado (con clave), D, tenían que cumplir los tres requisitos siguientes:

1. D(E(P))=P.

2. Es excesivamente difícil deducir D de E.

3. E no puede descifrarse mediante un ataque de texto normal seleccionado.

El método funciona como sigue. Una persona A que quiera recibir mensajes secretos, primero diseña dos algoritmos E y D, que cumplan los requisitos anteriores. El algoritmo de cifrado y la clave, E_A de cifrado, se hacen públicos, de ahí el nombre de criptografía de clave pública. Esto podría hacerse poniéndolos en un archivo accesible a cualquiera que quiera leerlo. A publica también el algoritmo de descifrado, pero mantiene secreta la clave de descifrado D_A. Por tanto E_A es pública, pero D_A es secreta.

Ahora veamos si podemos resolver el problema de establecer un canal seguro entre A y B, que nunca han tenido contacto previo. Se supone que tanto la clave de cifrado de A, E_A, como la clave de cifrado de B, E_B, están en un archivo de lectura pública. Ahora, A toma su primer mensaje P, calcula $E_B(P)$ y lo envía a B. B entonces lo descifra aplicando su clave secreta D_B (es decir, calcula $D_B(E_B(P))=P$). Nadie más puede leer el mensaje cifrado, $E_B(P)$, porque se supone que el sistema de cifrado es robusto y porque es demasiado difícil derivar D_B de la E_B públicamente conocida. A y B ahora pueden comunicarse con seguridad.

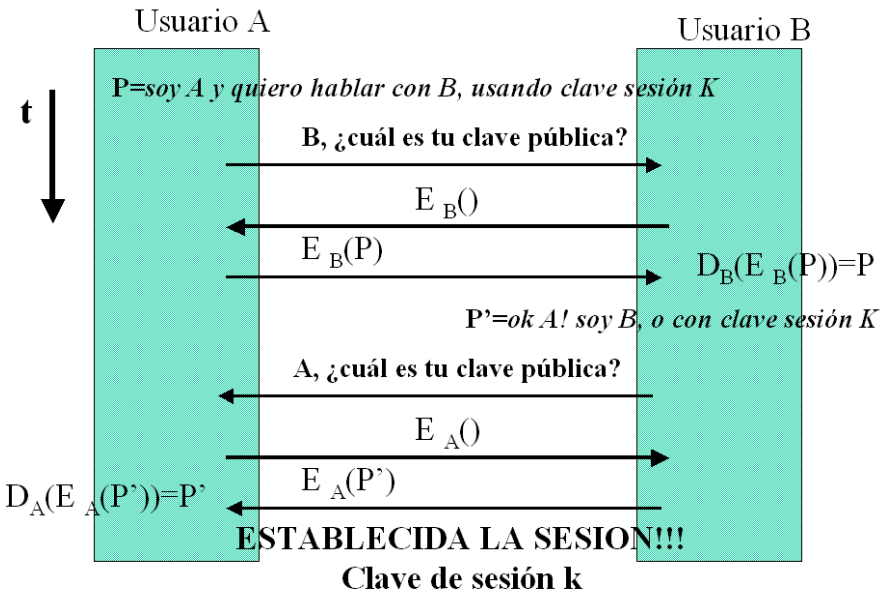


Figura 7: Escenario de establecimiento de sesión con clave asimétrica o pública

La única dificultad del método anterior estriba en que necesitamos encontrar algoritmos que realmente satisfagan los tres requisitos. Debido a las ventajas potenciales de la criptografía de clave pública, muchos investigadores están trabajando en este tema, y ya se han publicado algunos algoritmos. Un buen método fue descubierto por un grupo del M.I.T y es conocido como RSA por las iniciales de sus descubridores (Rivest, Shamir, Adleman). Su método se basa en:

1. Seleccionar dos números primos grandes, *p* y *q* (generalmente mayores que 10^{100}).

2. Calcular $n=pxq$ y $z=(p-1)x(q-1)$.

3. Seleccionar un número primo con respecto a *z* (es decir, un número sin ningún factor común con *z*), llamándolo *d*.

4. Encontrar e tal que $exd=1 \bmod z$

Con estos parámetros calculados por adelantado, estamos listos para comenzar el cifrado. Dividimos el texto normal (considerado como una cadena de bits) en bloques, para que cada mensaje de texto normal, P, caiga en el intervalo $0<P<n$, Esto puede hacerse agrupando el texto normal en bloques de k bits, donde k es el entero más grande para el que $2^k<n$ es verdad.

Para cifrar un mensaje, P, calculamos $C=P^e \pmod n$. Para descifrar C, calculamos $P=C^d \pmod n$. Puede demostrarse que, para todos los P del intervalo especificado, las funciones de cifrado y descifrado son inversas. Para ejecutar el cifrado, se necesitan e y n. Para llevar a cabo el descifrado se requieren d y n. **Por tanto, la clave pública consiste en el par (e,n) y la clave privada consiste en (d,n).**

La seguridad del método se basa en la dificultad de factorizar números grandes. Si el criptoanalista pudiera factorizar n (conocido públicamente), podría encontrar p y q, y a partir de éstos, z. Equipado con el conocimiento de z y de e, puede encontrar d usando el algoritmo de Euclides. Afortunadamente, los matemáticos han estado tratando de factorizar números grandes durante los últimos 300 años y las pruebas acumuladas sugieren que se trata de un problema excesivamente difícil. De acuerdo con los descubridores del RSA, la factorización de un número de 200 dígitos requiere 4 mil millones de años de tiempo de cómputo; la factorización de un número de 500 dígitos requiere 10^{25} años. En ambos casos se supone el mejor algoritmo conocido y una computadora con un tiempo de instrucción de 1 microsegundo. Aun si las computadoras continúan aumentando su velocidad, pasarán siglos antes de que sea factible la factorización de un número de 500 dígitos, y entonces simplemente se puede escoger un p y un q todavía más grandes.

Un ejemplo pedagógico trivial del algoritmo RSA se puede ver en la figura siguiente, en la cual se cifra un texto en el que la letra "a" se representa por el valor 1, la "b" por el valor 2, etc.

Texto normal (P)		Texto cifrado (C)			Después del descifrado	
Simbólico	*Numérico*	P^3	$P^3 \pmod{33}$	C^7	$C^7 \pmod{33}$	*Simbólico*
s	19	6859	28	13492928512	19	S
u	21	9261	21	1801088541	21	U
z	26	17576	20	1280000000	26	Z
a	01	1	01	1	01	A
n	14	2744	05	78125	14	N
n	14	2744	05	78125	14	N
e	05	125	26	8031810176	05	E
Cálculo del transmisor				Cálculo del receptor		

Para este ejemplo hemos seleccionado p=3 y q=11, dando n=33 y z=20. Un valor adecuado de d es d=7, puesto que 7 y 20 no tienen factores comunes. Con estas selecciones, e puede encontrarse resolviendo la ecuación $7e=1 \pmod{20}$, que produce e=3. El texto cifrado C, de un mensaje de texto normal, P, se da por la regla $C=P^3 \pmod{33}$. El texto cifrado lo descifra el receptor de acuerdo con la regla $P=C^7 \pmod{33}$. En la figura se muestra como ejemplo el cifrado del texto normal "*suzanne*".

Dado que los números primos escogidos para el ejemplo son tan pequeños, P debe ser menor que 33, por lo que cada bloque de texto normal puede contener sólo un carácter. El resultado es un cifrado por sustitución monoalfabética, no muy impresionante. En cambio, si hubiéramos seleccionado p y q del orden de 10^{100} podríamos tener n del orden de 10^{200}, por lo que cada bloque podría ser de hasta 664 bits (83 caracteres de 8 bits), contra 64 bits (8 caracteres de 8 bits) para el algoritmo DES.

Sin embargo, el algoritmo RSA es demasiado lento para poder cifrar grandes volúmenes de datos, por lo cual suele usarse para distribuir claves de sesión de una sola vez para su uso con los algoritmos DES, IDEA u otros semejantes.

El tamaño de las claves del RSA puede ser variable, mínimo 500 bits, pero lo habitual suele ser utilizar claves de 512 bits (32 dígitos hexadecimales), 768 o 1024 bits.

Se puede demostrar y comprobar que si invertimos el orden de las claves, el proceso es exactamente igual de seguro, es decir, es lo mismo $D_B(E_B(P))=P$ que $E_B(D_B(P))=P$, y según el algoritmo $C=P^d(\text{mod } n)$ y $P=C^e(\text{mod } n)$. Con ello este algoritmo puede tener utilidad para 2 tipos de situaciones. En el primer caso $D_B(E_B(P))=P$ para cifrar algo cuando se manda a B, porque utilizo su clave pública y sólo B puede descifrar con su clave privada. En el segundo caso, $E_B(D_B(P))=P$, B puede hacer uso de la clave privada para mandar información, y que TODO el mundo pueda leer si pide la clave pública de B y asegura que sólo B puede haber cifrado P, porque sólo B dispone de la clave privada correspondiente.

Otros algoritmos de cifrado con clave pública

Cabe comentar finalmente que existen otros algoritmos de clave pública (asimétrica), basados siempre en resolución de problemas matemáticos de cálculo costoso, que tienen un procedimiento directo fácil, pero difícil en inverso. Estos métodos son de resolución computacional imposible, para poder resolver la función matemática inversa.

Algoritmo de ElGamal

Fue diseñado en un principio para producir firmas digitales, pero posteriormente se extendió también para codificar mensajes. Se basa en el problema de los logaritmos discretos, que está íntimamente relacionado con el de la factorización, y en el de Diffie-Hellman.

Para generar un par de claves, se escoge un número primo n y dos números aleatorios p y x menores que n. Se calcula entonces

$$y = p^x \text{ (mod n)}$$

La clave pública es (p, y, n), mientras que la clave privada es x.

Escogiendo n primo, garantizamos que sea cual sea el valor de p, el conjunto $\{p, p^2, p^3 \ldots\}$ es una permutación del conjunto $\{1, 2,\ldots n-1\}$ lo suficientemente grande.

Firmas Digitales de ElGamal

Para firmar un mensaje m basta con escoger un número k aleatorio, tal que mcd(k, n-1) =1, y calcular

$a = p^k \text{ (mod n)}$

$b = (m - xa)k^{-1} \text{ (mod (n - 1))}$

La firma la constituye el par (a, b). En cuanto al valor k, debe mantenerse en secreto y ser diferente cada vez. La firma se verifica comprobando que

$y^a a^b = p^m \text{ (mod n)}$

Codificación de ElGamal

Para codificar el mensaje m se escoge primero un número aleatorio k primo relativo con (n-1), que también será mantenido en secreto. Calculamos entonces las siguientes expresiones

$a = p^k \text{ (mod n)}$

$b = y^k m \text{ (mod n)}$

El par (a, b) es el texto cifrado, de doble longitud que el texto original. Para decodificar se calcula

$m = b * a^{-x} \text{ (mod n)}$

Algoritmo de Rabin

El sistema de clave asimétrica de Rabin se basa en el problema de calcular raíces cuadradas modulo un número compuesto. Este problema se ha demostrado que es equivalente al de la factorización de dicho número.

En primer lugar escogemos dos números primos, p y q, ambos congruentes con 3 módulo 4 (los dos últimos bits a 1). Estos primos son la clave privada. La clave pública es su producto, $n = pq$.

Para codificar un mensaje m, simplemente se calcula

$c = m^2 \pmod n$

La decodificación del mensaje se hace calculando lo siguiente:

$m_1 = c^{(p+1)/4} \pmod p$
$m_2 = (p - c^{(p+1)/4}) \pmod p$
$m_3 = c^{(q+1)/4} \pmod q$
$m_4 = (q - c^{(q+1)/4}) \pmod q$

Luego se escogen a y b tales que $a = q(q^{-1} \pmod p)$ y $b = p(p^{-1} \pmod q)$. Los cuatro posibles mensajes originales son

$m_a = (am_1 + bm_3) \pmod n$
$m_b = (am_1 + bm_4) \pmod n$
$m_c = (am_2 + bm_3) \pmod n$
$m_d = (am_2 + bm_4) \pmod n$

Desgraciadamente, no existe ningún mecanismo para decidir cuál de los cuatro es el auténtico, por lo que el mensaje deberá incluir algún tipo de información para que el receptor pueda distinguirlo de los otros.

Algoritmo DSA

El algoritmo DSA (Digital Signature Algorithm) es una parte el estándar de firma digital DSS (Digital Signature Standard). Este algoritmo, propuesto por el NIST, data de 1991, es una variante del método asimétrico de ElGamal.

Creación del par clave pública-clave privada

El algoritmo de generación de claves es el siguiente:

1. Seleccionar un número primo *q* tal que $2^{159} < q < 2^{160}$.
2. Escoger *t* tal que $0 \le t \le 8$, y seleccionar un número primo *p* tal que $2^{511+64t} < p < 2^{512+64t}$, y que además q sea divisor de (p - 1).
3. Seleccionar un elemento $g \in Z^*_p$ y calcular $\alpha = g^{(p-1)/q} \bmod p$.
4. Si $\alpha = 1$ volver al paso 3.
5. Seleccionar un número entero aleatorio a, tal que $1 \le a \le q - 1$
6. Calcular $y = \alpha^a \bmod p$.
7. La clave pública es (p, q, α, y). La clave privada es *a*.

Generación y verificación de la firma

Siendo h la salida de una función resumen sobre el mensaje m, la generación de una firma se hace mediante el siguiente algoritmo:

1. Seleccionar un número aleatorio k tal que $0 < k < q$.
2. Calcular $r = (\alpha^k \bmod p) \bmod q$.
3. Calcular $k^{-1} \bmod q$.
4. Calcular $s = k^{-1}(h + ar) \bmod q$.
5. La firma del mensaje m es el par (r, s).

El destinatario efectuará las siguientes operaciones, suponiendo que conoce la clave pública (p, q, α , y), para verificar la autenticidad de la firma:

1. Verificar que $0 < r < q$ y $0 < s < q$. En caso contrario, rechazar la firma.
2. Calcular el valor de h a partir de m.
3. Calcular $\omega = s^{-1} \bmod q$.
4. Calcular $u_1 = \omega * h \bmod q$ y $u_2 = \omega * r \bmod q$.
5. Calcular $v = (\alpha^{u1} y^{u2} \bmod p) \bmod q$.
6. 6. Aceptar la firma si y sólo si v = r**Localización de los dispositivos de cifrado**

Existen dos alternativas para la ubicación del cifrado, y vamos a analizarla desde el punto de vista de una sesión TCP/IP:

-cifrado de enlace a enlace, donde cada nodo intermedio (router) debe descifrar los paquetes
-cifrado extremo a extremo, donde sólo los extremos pueden descifrar la información, pero las cabeceras de los paquetes han de viajar descifradas para que los routers puedan encaminar

y una combinación de los anteriores, **cifrado mixto enlace-enlace, extremo-extremo**, donde las cabeceras van cifradas enlace a enlace y los datos extremo a extremo

Con esta estrategia de localización de los cifradores, podemos ocultar la información, pero realmente a un intruso sólo se le oculta las direcciones y los contenidos, pero no el **volumen de información** intercambiado. Para ocultar dicha información, se integra **tráfico de relleno**, para ocultar el volumen de información real.

Por ejemplo, si un usuario trata de acceder a un servidor de banca electrónica y la información intercambiada es mínima, frente a otro usuario que el volumen de información intercambiado es grande, da a entender que el segundo tiene más trato con el banco y el volumen de dinero debería ser mayor.

Comentario clave pública vs clave privada

Dada la lentitud y coste computacional de los métodos de las claves asimétricas, del orden de 1000 veces mayor, aunque es sí más seguros, en la realidad toda conexión remota constar de 2 partes. Un primer establecimiento para validar y autenticar, además de negociar una clave secreta para la sesión, utilizando para ello la clave pública y una segunda parte, que consiste en el cifrado de forma simétrica (DES, 3DES, IDEA, ...) con la clave llave secreta negociado.

PROTOCOLOS DE SEGURIDAD: AUTENTICACIÓN Y VALIDACIÓN

La validación de identificación es la técnica mediante la cual un proceso comprueba que su compañero de comunicación es quien se supone que es y no un impostor. La verificación de la identidad de un proceso remoto ante un intruso activo malicioso es sorprendentemente difícil y requiere protocolos complejos basados en criptografía.

Los métodos de autenticación o validación pueden estar basados en una clasificación más amplia como podemos ver a continuación:

- biomédicas, por huellas dactilares, retina del ojo, ...son muy seguros pero muy costosos
- tarjetas inteligentes que guardan información de los certificados de un usuario
- métodos clásicos basados en contraseña, son sistemas basados en esta tecnología están:
  - –Método tradicional o comprobación local
  - –Método distribuido o en red

Los métodos basados en protocolo, también son conocidos como métodos distribuidos o basados en red.

El modelo general que usan todos los protocolos de validación de identificación basado en red es éste. Un usuario (en realidad, un proceso) iniciador, digamos A, quiere establecer una conexión segura con un segundo usuario, B. A y B se llaman principales, los personajes centrales de nuestra historia. B es un banquero con el que A quiere hacer negocios. A comienza por enviar un mensaje a B o a un centro de distribución de claves (KDC) confiable, que siempre es honesto. Siguen varios otros intercambios de mensajes en diferentes direcciones. A medida que se envían estos mensajes, un intruso malicioso, C, puede interceptarlos, modificarlos o reproducirlos para engañar a A y a B, o simplemente para estropear sus actividades.

No obstante, una vez que se ha completado el **protocolo**, A está segura de que está hablando con B, y B está seguro de que está hablando con A. Es más, en la mayoría de los protocolos, los dos también habrán establecido una clave de sesión secreta para usarla durante toda la conversación subsiguiente. En la práctica, como hemos dicho en la sección anterior, por razones de prestaciones, todo el tráfico de datos se cifra usando criptografía de clave secreta, aunque la criptografía de clave pública se usa ampliamente con los protocolos de validación de identificación mismos y para establecer la clave de sesión. Por esto, la clave pública es utilizada únicamente para identificarse inicialmente y conseguir una clave de sesión. Obviamente esta clave debe ser mucho más confidencial que la clave de sesión, dado que si un intruso la obtuviera, podría descifrar y obtener desde ese momento TODAS las claves de TODAS las sesiones que realizáramos.

El objetivo es usar una clave de sesión nueva, seleccionada aleatoriamente, para cada nueva conexión con tal de reducir al mínimo la cantidad de tráfico que es enviado y así la cantidad de texto cifrado que puede obtener un intruso está limitado sólo al de la sesión. En el caso de que el intruso descifrara y obtuviera la clave, con suerte, la única clave presente en ese momento sería la clave de la sesión. Las claves públicas o secretas compartidas siempre se guardan bajo fuerte custodia, ya que son la clave para obtener todas las demás..

Los métodos distribuidos o en red además se clasifican en:

clave secreta (*privada*)
- basados en clave compartida, Diffie Hellman
- basado en servidores con registro de todos los usuarios. Ejemplo TACACS+ y Radius
- basados en NIS, NIS+ (*Network Information Service*)
- basados en centros de distribución de clavesbasados en criptografía, Kerberos

clave pública (que veremos en mayor detalle en el siguiente apartado)
- basados en servidores de directorios X.500, LDAP (*Lighweight Directory Access Protocol)*
- basados en Certificados

Pasemos a ver con detalle cada uno de ellos.

Validación de identificación de clave secreta.

Veremos a continuación distintos protocolos de validación basados en el uso de una clave secreta compartida por los usuarios A y B o por los usuarios y por un centro de distribución de claves (KDC).

Validación de identificación basada en clave secreta compartida.

Para nuestro primer protocolo de validación de identificación, supondremos que A y B ya comparten una clave secreta K_{AB}. Esta clave compartida podría haberse acordado telefónicamente o en persona pero, en cualquier caso, no a través de la (insegura) red.

Este protocolo se basa en un principio encontrado en muchos protocolos de validación de identificación: una parte envía un número aleatorio a la otra, que entonces lo transforma de una manera especial y devuelve el resultado. Tales protocolos se llaman protocolos de reto-respuesta. En éste, y en los subsiguientes protocolos de validación de identificación se usara la siguiente notación.

- A y B son las identidades de las procesos que desean verificar su identidad y C es la identidad del intruso.
- R_i son los retos, donde el subíndice identifica el retador.
- K_i son las claves, donde i indica el dueño; K_s es la clave de la sesión.

La secuencia de mensajes de nuestro primer protocolo de validación de identificación de clave compartida se muestra en la figura siguiente.

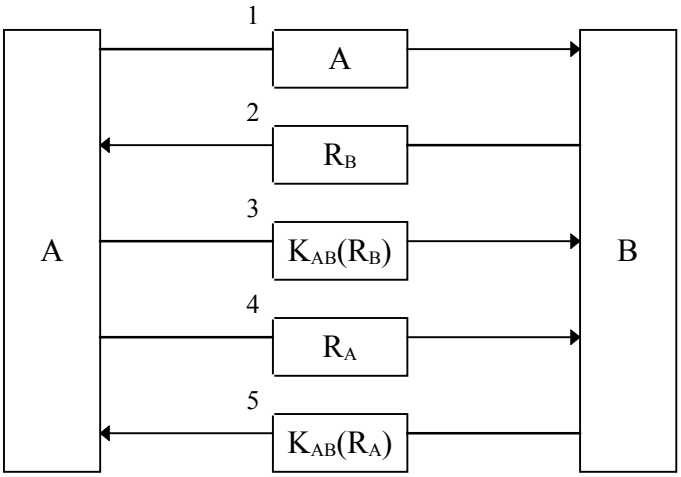


Figura 8: Validación de identificación de clave compartida.

En el mensaje 1, A envía su identidad, A, a B de manera que la entienda B. B, por supuesto, no tiene manera de saber si este mensaje vino de A o de C, por lo que escoge un reto, un número aleatorio grande, R_B, y lo envía a A como mensaje 2, en texto normal. A entonces cifra el mensaje con la clave que comparte con B y envía el texto cifrado $K_{AB}(R_B)$, en el mensaje 3. Cuando B ve este mensaje, de inmediato sabe que vino de A porque C no conoce K_{AB} y, por tanto, no pudo haberlo generado. Es más, dado que se selecciono R_B al azar de un espacio grande (digamos, números aleatorios de 128 bits), es muy poco probable que C haya visto R_B y su respuesta en una sesión previa.

En este punto, B está seguro que está hablando con A, pero A no está segura de nada; hasta donde sabe, C podría haber interceptado el mensaje 1 y enviado R_B como respuesta. Para saber a quién le está hablando, A selecciona un número al azar, R_A, y lo envía a B como texto normal, en el mensaje 4. Cuando B responde con $R_{AB}(K_A)$, A sabe que está hablando con B. Si desean establecer ahora una clave de sesión, A puede seleccionar una K_S, y enviársela a B cifrada con K_{AB}.

Aunque el protocolo anterior funciona, contiene mensajes de más. Estos pueden eliminarse combinando información y reduciendo el protocolo a tres mensajes, como puede verse en la siguiente figura:

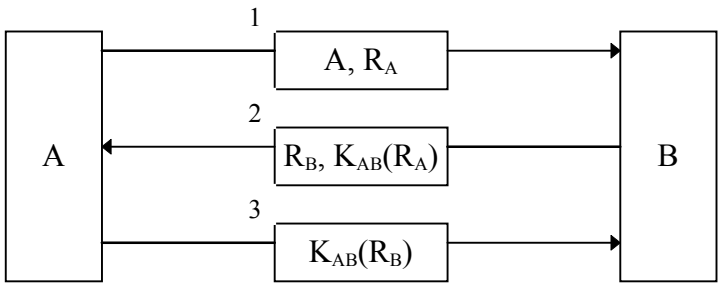


Figura 9: Mejora del algoritmo de validación de identificación de clave compartida.

Sin embargo, estos protocolos poseen un fallo. Un intruso C puede burlar este protocolo usando lo que se conoce como ataque por reflexión. En particular, C puede inutilizarlo si es posible abrir al mismo tiempo varias sesiones con B. Esta situación sería cierta si, por ejemplo, B es un banco y está preparado para aceptar muchas conexiones simultáneas de los cajeros bancarios.

El ataque por reflexión de C se muestra en la figura siguiente; comienza cuando C indica que es A y envía R_C. B responde, como de costumbre, con su propio reto, R_B. Ahora C está atorado, pues no conoce $K_{AB}(R_B)$.

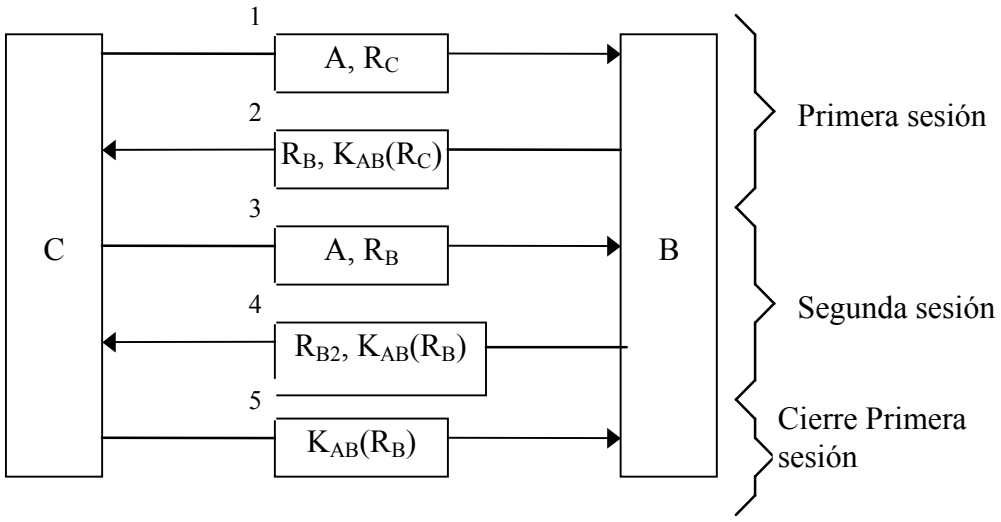


Figura 10: Ataque por reflexión al algoritmo de validación de identificación de clave compartida.

C puede abrir una segunda sesión con el mensaje 3, proporcionando como reto el R_B tomado del mensaje 2. B lo cifra y envía de regreso $K_{AB}(R_B)$ en el mensaje 4. Ahora C tiene toda la información faltante, por lo que puede completar la primera sesión y abortar la segunda. B ahora está convencido de que C es A, por lo que le autoriza cualquier operación que desee hacer como si fuera A.

Tres reglas generales que frecuentemente son de ayuda son las siguientes:

1. Hacer que el iniciador demuestre quién es antes de que lo tenga que hacer el respondedor. En este caso, B regala información valiosa antes de que C dé cualquier prueba de su identidad.

2. Hacer que el iniciador y el respondedor usen diferentes claves para comprobación, incluso si esto significa tener dos claves compartidas, K_{AB} y K'_{AB}.

3. Hacer que el iniciador y el respondedor tomen sus retos de diferentes conjuntos. Por ejemplo, el iniciador debe usar números pares y el respondedor números nones.

Las tres reglas se violaron aquí, con resultados desastrosos. Nótese que nuestro primer protocolo de validación de identificación (de cinco mensajes) requiere que A compruebe primero su identidad, por lo que ese protocolo no está a merced del ataque por reflexión.

Establecimiento de una clave compartida: intercambio de claves Diffie-Hellman.

Hasta ahora hemos supuesto que A y B comparten una clave secreta. Pero puede no ser así, sin embargo, y afortunadamente, existe una manera de que completos desconocidos establezcan una clave secreta a plena luz del día, aún con C registrando cuidadosamente cada mensaje.

El protocolo que permite que dos extraños establezcan una clave secreta compartida se llama intercambio de claves Diffie-Hellman, y funciona como sigue. A y B tiene que acordar dos números primos grandes, n y g, donde (n-1)/2 también es primo y g debe cumplir ciertas condiciones. Esto números pueden ser públicos, por lo que cualquiera de ellos puede escoger n y g y decírselo al otro abiertamente. Ahora, A escoge un número grande (digamos de 512 bits), x, y lo mantiene en secreto. De la misma manera, B escoge un número secreto grande y.

A inicia el protocolo de intercambio de claves enviando a B un mensaje que contiene (n, g, g^x mod n), como se muestra en la figura siguiente. B responde enviando a A un mensaje que contiene g^y mod n. Ahora A toma el número que B le envió y lo eleva a la potencia x para obtener $(g^y \text{ mod } n)^x$. B lleva a cabo una tarea parecida para obtener $(g^x \text{ mod } n)^y$. Por las leyes de la aritmética modular, ambos cálculos arrojan g^{xy} mod n. Ahora A y B comparten una clave secreta g^{xy} mod n.

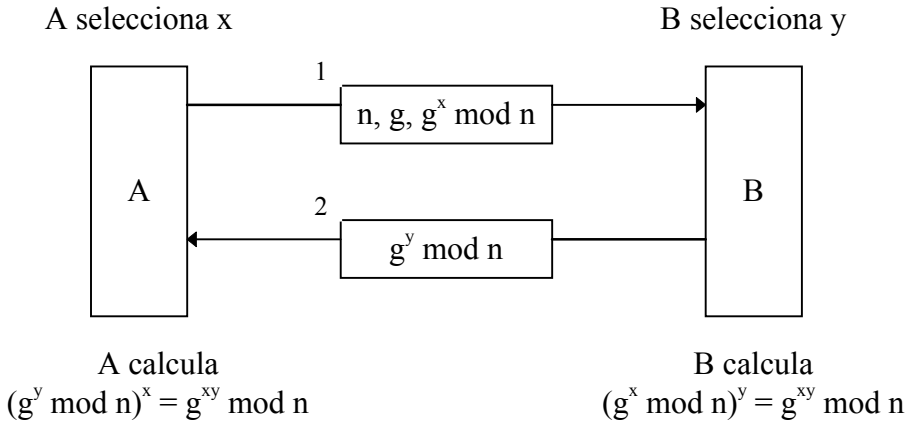


Figura 11:Intercambio de claves Diffie-Hellman.

Por supuesto, C ha visto ambos mensajes, así que conoce g y n del mensaje 1. Si C pudiera calcular x y y, podría averiguar la clave secreta. El problema es que, dado sólo g^x mod n, no es posible encontrar x. No se conoce un algoritmo práctico para calcular logaritmos discretos módulo un número primo muy grande.

A pesar de la elegancia del algoritmo Diffie-Hellman, hay un problema: cuando B recibe la tripleta (n, g, g^x mod n), no sabe si es A o C quien se la ha enviado. No hay manera de saberlo. Desgraciadamente, C puede explotar este hecho para engañar tanto a A como a B, como se ilustra en la siguiente figura.

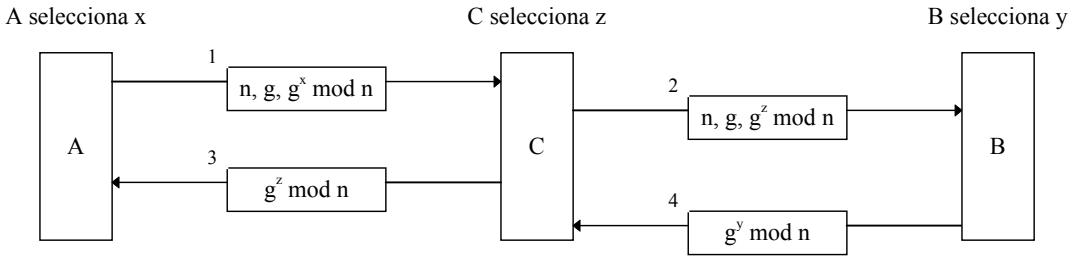


Figura 12: Ataque de brigada de cubetas al intercambio de claves Diffie-Hellman.

Ahora todos hacen la aritmética modular. A calcula la clave secreta como g^{xz} mod n, y también lo hace C (para los mensajes de A). B calcula g^{yz} mod n, al igual que C (para los mensajes de B). A piensa que esta hablando con B, por lo que establece una clave de sesión con C. Lo mismo hace B. Cada mensaje que A envía durante la sesión cifrada es capturado por C, almacenado, modificado y pasado (opcionalmente) a B. Lo mismo ocurre en la otra dirección. C ve todo y puede modificar los mensajes si lo desea, mientras que tanto A como B están pensando equivocadamente que tienen un canal seguro entre ambos. Este ataque se conoce como ataque de brigada de cubetas o ataque del hombre (mujer) en medio.

Validación de identificación usando un centro de distribución de claves.

El establecimiento de un secreto compartido con un extraño casi funcionó, pero no por completo. Por otra parte no valía la pena, pues para hablarle a n personas de esta manera se requerían n claves. Para la gente muy popular, la administración de claves se volvería una verdadera carga, especialmente si cada clave tuviera que almacenarse en un chip de una tarjeta de plástico individual.

Un enfoque diferente es introducir un centro de distribución de claves confiables (KDC). En este modelo, cada usuario tiene una sola clave compartida con el KDC. La validación de identificación y la administración de claves de sesión ahora pasan a través del KDC. El protocolo de validación e identificación más simple conocido es la rana de boca amplia, que puede verse en la siguiente figura:

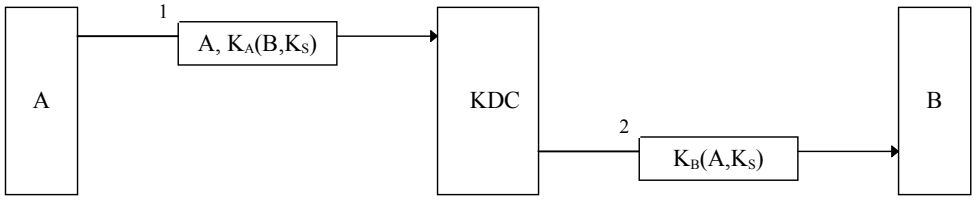


Figura 13: Protocolo de validación de rana de boca amplia.

El concepto en que se basa el protocolo de boca amplia es sencillo: A escoge una clave de sesión, K_S e indica al KDC que desea hablar con B usando K_S. Este mensaje se cifra con la clave secreta que comparte A (solamente) con el KDC, K_A. El KDC descifra este mensaje, extrayendo la identidad de B y la clave de sesión; luego construye un mensaje nuevo que contiene la identidad de A y la clave de sesión y envía este mensaje a B. Este cifrado se hace con K_B, la clave secreta que B comparte con el KDC. Cuando B descifra el mensaje, sabe que A quiere hablar con él, y también conoce la clave que desea usar.

La validación de identificación aquí es gratuita. El KDC sabe que el mensaje 1 debe haber venido de A, puesto que nadie más habría sido capaz de cifrarlo con la clave secreta de A. De la misma manera, B sabe que el mensaje 2 debe haber venido del KDC, en quien confía, puesto que nadie más sabe su clave secreta.

Sin embargo, este protocolo tiene un fallo, un intruso C puede copiar el mensaje 2 de la figura y todos los mensajes que le siguen y reproducirlos de nuevo para B, con lo cual B creerá estar hablando con A. Este problema se llama ataque por repetición, con lo cual, aunque no pueda modificar C los mensajes, puede hacer interferir a B y pueda generar a la larga en un ataque que se llama denegación de servicio.

Son posibles varias soluciones al ataque por repetición. La primera es incluir una marca de tiempo en cada mensaje. Entonces, si alguien recibe un mensaje obsoleto, puede descartarlo. El problema de este enfoque es que los relojes nunca están perfectamente sincronizados en toda una red, por lo que tiene que haber un intervalo durante el cual la marca de tiempo sea válida. C puede repetir el mensaje durante ese tiempo y salirse con la suya.

La segunda solución es poner un número de mensaje único, de una sola ocasión, a veces llamado núnico, en cada mensaje. Cada parte entonces tiene que recordar todos los núnicos y rechazar cualquier mensaje que contenga un núnico previamente usado. Pero los núnicos tienen que recordarse indefinidamente, no vaya a ser que C esté tratando de reproducir un mensaje de años de antigüedad. También, si una máquina se cae y pierde la lista de núnicos, nuevamente es vulnerable a un ataque por repetición. Las marcas de tiempo y los núnicos pueden combinarse para limitar el tiempo que pueden recordarse los núnicos, pero ciertamente el protocolo se volverá mucho más complicado.

Un enfoque más refinado para la validación de identificación es usar un protocolo multisentido de reto-respuesta. Un ejemplo bien conocido de tal protocolo es el de validación de identificación Needham-Schroeder, que puede verse en la siguiente figura:

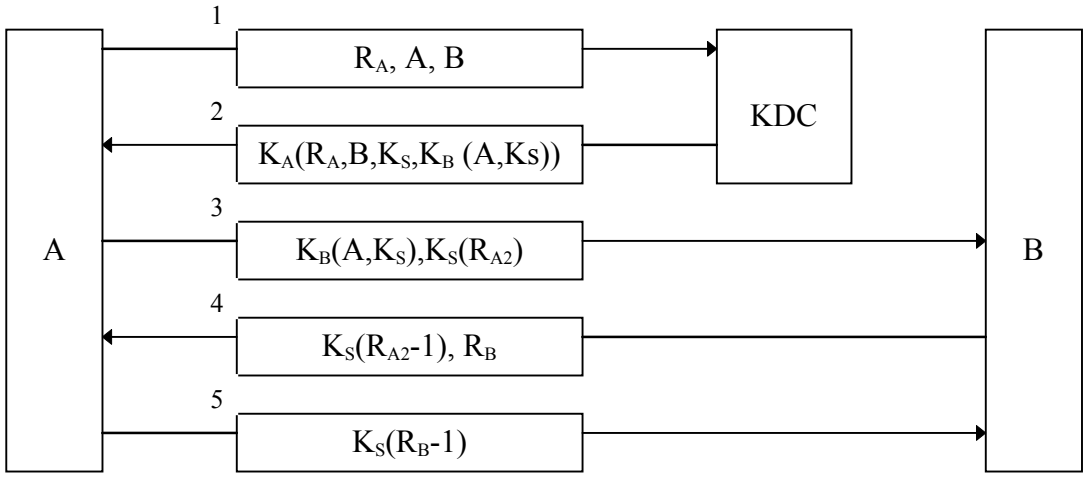


Figura 14: Protocolo de validación de identificación Needham-Schoeder.

El protocolo comienza cuando A indica al KDC que quiere hablar con B. Este mensaje contiene un número aleatorio grande, R_A, como núnico. El KDC devuelve el mensaje 2 que contiene el número aleatorio de A, una clave de sesión, y un billete que puede enviar a B. El objeto del número aleatorio, R_A, es asegurar a A que el mensaje 2 es reciente, y no una repetición. La identidad de B también se incluye por si a C se le ocurre reemplazar B del mensaje 1 por su propia identidad, para que el KDC cifre el billete al final del mensaje 2 con K_C en lugar de K_B. El billete cifrado con K_B se incluye dentro del mensaje cifrado para evitar que C lo reemplace por otra cosa en su camino hacia A.

A ahora envía el billete a B, junto con un nuevo número aleatorio, R_{A2}, cifrado con la clave de la sesión, K_S. En el mensaje 4, B devuelve $K_S(R_{A2}-1)$ para demostrar a A que está hablando con el verdadero B. El envío de regreso de $K_S(R_{A2})$ no habría funcionado, puesto que C lo podría haber robado del mensaje 3.

Tras recibir el mensaje 4, A está convencida de que está hablando con B, y de que no se pudieron haber usado repeticiones hasta el momento. A fin de cuentas, A acaba de generar R_{A2}, hace unos cuantos milisegundos. El propósito del mensaje 5 es convencer a B de que realmente está hablando con A, y de que tampoco se han usado repeticiones aquí. Al hacer que cada parte genere un reto y responda a otro, se elimina la posibilidad de un ataque por repetición.

Aunque este protocolo parece bastante sólido, tiene una ligera debilidad, si C llega a obtener una clave de sesión vieja en texto normal, puede iniciar una nueva sesión con B repitiendo el mensaje 3 correspondiente a la clave obtenida y convencerlo de que es A. Otway y Rees publicaron un protocolo que resuelve el problema. La figura siguiente muestra el protocolo de Otway-Rees:

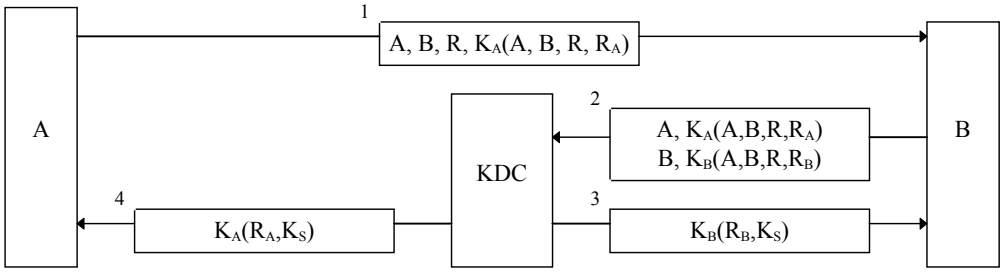


Figura 15: Protocolo de validación de identificación Otway-Ress.

En el protocolo Otway-Rees, A comienza por generar un par de números aleatorios: R, que se usará como identificador común, y R_A, que A usará para retar a B. Cuando B recibe este mensaje, construye un mensaje nuevo a partir de la parte cifrada del mensaje de A, y uno análogo propio. Ambas partes cifradas con K_A y K_B identifican a A y a B, contienen el identificador común y contienen un reto.

El KDC comprueba que el R de ambas partes es igual. Los R podrían no serlo porque C alteró el R del mensaje 1 y reemplazó parte del mensaje 2. Si los dos R son iguales, el KDC se convence de que el mensaje de solicitud de B es válido, y genera una clave de sesión y la cifra dos veces, una para A y otra para B. Cada mensaje contiene un número aleatorio del receptor como prueba de que el KDC, y no C, generó el mensaje. En este punto, tanto A como B poseen la misma clave de sesión y pueden comenzar a comunicarse. La primera vez que intercambien mensajes de datos, ambos podrán ver que el otro tiene una copia idéntica de K_S, por lo que la validación de identificación está completa.

Protocolo de autenticación Kerberos.

El protocolo de autenticación Kerberos fue diseñado por el MIT (Massachusetts Institute of Technology) para autenticar la identidad de los usuarios de una red digital de comunicaciones insegura, así como para distribuir las claves secretas de sesión que permitan a los usuarios de la red establecer comunicaciones seguras. Es conveniente resaltar que como usuario se engloba tanto a personas físicas que se comunican a través de ordenadores como a posibles servicios ofrecidos por la red.

El protocolo está basado en criptografía de clave secreta y básicamente actúa como un árbitro en quien los usuarios confían. Para desarrollar sus funciones, Kerberos comparte con cada usuario una clave secreta diferente intercambiada con éste a través de un canal seguro. El conocimiento de dicha clave se utiliza como prueba de identidad del usuario. En estas condiciones como Kerberos conoce las claves secretas de todos los usuarios, puede demostrar a cualquiera de ellos la autenticidad de la identidad de otro. Además Kerberos genera claves secretas transitorias, denominadas claves de sesión, las cuales son transmitidas exclusivamente a cada pareja de usuarios que desean entrar en comunicación.

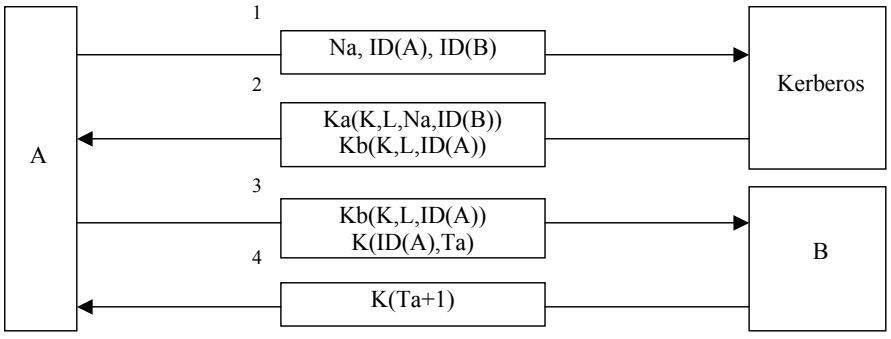


Figura 16: Protocolo de validación de identificación Kerberos.

Veamos el funcionamiento básico del protocolo. Para ello consideremos únicamente a los tres participante básicos del mismo: la autoridad de autenticación y servidor de claves Kerberos y dos usuarios de la red A y B. Al comienzo del protocolo A y B no comparten ninguna clave secreta. Sin embargo, cada uno de ellos comparte con Kerberos una clave secreta de un criptosistema simétrico. Sea

$E_d(\ldots)$ dicho criptosistema (clave simétrica *d*) y sean *Ka* y *Kb* las respectivas claves secretas compartidas con Kerberos por los usuarios A y B. Asimismo, sean *ID(A)* e *ID(B)* los respectivos identificadores de A y B en la red de comunicación. El funcionamiento del protocolo se describe a continuación:

1- El usuario A solicita a Kerberos una credencial para identificarse ante B, así como una clave de sesión que le permita establecer dicha comunicación. Para ello A envía a Kerberos un valor aleatorio *Na* y las identidades *ID(A)* e *ID(B)*.

2- Kerberos genera una clave de sesión aleatoria *K* y define el período de validez *L* de la credencial, cifrando a continuación los valores *K*, *L*, *Na* y la identidad *ID(B)* del usuario B con la clave secreta *Ka* del usuario A, obteniendo el valor *m* dado por $m = E_{Ka}(K, L, Na, ID(B))$.

   Seguidamente Kerberos calcula la credencial *c* cifrando *K*, *L* y la identidad *ID(A)* del usuario A con la clave secreta *Kb* del usuario B, es decir $c = E_{Kb}(K, L, ID(A))$.

   En estas condiciones Kerberos envía la pareja de valores *(m, c)* al usuario A.

3- El usuario A descifra m con su clave secreta *Ka* y recupera *K*, *L*, *Na* y la identifidad *ID(B)* del usuario B para el que fue emitida la credencial *c*. Entonces A verifica que el valor aleatorio *Na* corresponde con el que él previamente envió y guarda *L* como referencia. A continuación, A calcula el autenticador *a* para la credencial *c* cifrando su identidad *ID(A)* y su sello temporal *Ta* con la clave de sesión *K*, es decir $a = E_K(ID(A), Ta)$.

   Con todo ello el usuario A envía a B la pareja de valores *(c, a)*.

4- El usuario B recibe los valores *(c, a)* y descifra la credencial c con su clave secreta *Kb*, recuperando de esta forma *K*, *L* y la identidad *ID(A)*. Entonces utiliza la clave de sesión recuperada *K* para descifrar el autenticador y recuperar los valores *ID(A)* y *Ta* con el que comprueba que:

   - Las identidades de la credencial y el autenticador coinciden.
   - El sello *Ta* es válido y se encuentra en los límites de *L*.

5- Si las comprobaciones son satisfactorias, B se convence de la autenticidad de la identidad de A, así como de que la clave de sesión cifrada en la credencial *c* es la misma con la que se ha cifrado el autenticador *a*. En tal caso, B envía a A la conformidad *h* dada por $h = E_K(Ta + 1)$.

6- Por su parte el usuario A descifra *h* con la clave de sesión *K* y verifica que el valor recuperado es *Ta+1*, lo cual asegura a A que la clave de sesión *K* ha sido correctamente recibida por el usuario B.

Es conveniente hacer notar las diferentes funciones de los mensajes transmitidos en el protocolo anterior. Así por ejemplo, los valores cifrados *c* y *m* tienen la función de garantizar la confidencialidad en la transmisión de la clave secreta de sesión *K*, mientras que los valores *a* y *h* se utilizan para la confirmación de la clave, es decir, para convencer a los usuarios A y B de que ambos poseen la misma clave secreta de sesión *K*.

El propósito del sello temporal *Ta* y del período de validez *L* es doble. Por una parte, tienen como objetivo que el usuario A pueda utilizar la credencial c para realizar sucesivas autenticaciones ante B durante el período de validez de la clave sin necesidad de que intervenga Kerberos. El otro objetivo es el de prevenir un posible ataque activo al protocolo consistente en el almacenamiento de las claves para su posterior reutilización. Esto se consigue puesto que una clave no se acepta como válida si su sello temporal no se encuentra dentro de los límites del período de validez de la misma.

A continuación comentamos algunos puntos sobre Kerberos en plan resumen:

1. Kerberos viene en la mayoría de distribuciones UNIX y/o Linux, utilizado en procesos como login, rlogin o NFS (Network File System)
2. es un sistema centralizado y no distribuido en una red, lo cual si falla .o si se ve comprometido, se compromete toda la red, además las claves almacenadas son privadas y no públicas. En versión 4 y/o 5 incorpora servidores secundarios requiere la kerberización (modificación) de todas las aplicaciones, así como de la sincronización de todas las máquinas
4. las versiones más actualizadas son v4 y v5, la diferencia estriba en que v5 solicita tanto login como password del usuario para conectar al servidor Kerberos, de forma que un intruso conociendo un usuario no pueda por fuerza bruta de contraseñas descifrar la contestación del servidor. Dicha información de contraseña junto con información de tiempo, va en el reto inicial R_A
5. cuando un usuario está más de 8 horas (por defecto) delante de una estación de trabajo kerberizada, debe identificarse otra vez
6. algunos inconvenientes citados se han resuelto con versiones mejoradas o con la utilización de otras aplicaciones, como SSH (Secure Shell)
7. en ningún momento los passwords viajan por la red ni guardados en memoria, sólo las credenciales
8. cabe destacar del protocolo visto, que realmente se distinguen dos partes, por una lado la autenticación y por otro la gestión de tickets para los diferentes servicios

Veamos un pequeño ejemplo de funcionamiento con Kerberos

Escenario: usuario A a través de login requiere de las credenciales necesarias para acceder a otros servicios

Pasos:

1.- al teclear el nombre, el login kerberizado envía el nombre al servidor Kerberos solicitando un ticket

2.- si el usuario es conocido, le manda un mensaje cifrado, donde utilizará su contraseña (password) para descifrarlo. De esta forma la contraseña nunca viaja por la red

3.- de dicho mensaje extrae el ticket para realizar la petición de servicio

Validación de identificación usando criptografía de clave pública.

La validación de identificación mutua también puede hacerse usando criptografía de clave pública. Para comenzar, supongamos que A y B ya conocen las claves públicas del otro (un asunto nada trivial). Ellos quieren establecer una sesión y luego usar criptografía de clave secreta en esa sesión, ya que típicamente es de 100 a 1000 veces más rápida que la criptografía de clave pública. El propósito de este intercambio inicial entonces es validar la identificación de ambos y acordar una clave de sesión secreta compartida.

Este procedimiento puede hacerse de varias maneras. Una típica se muestra en la figura siguiente:

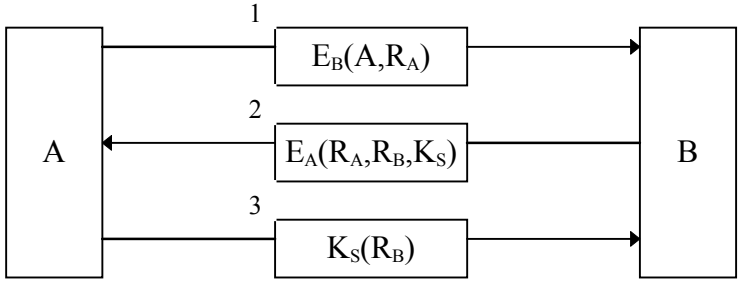


Figura 17 Protocolo de validación de identificación usando criptografía de clave pública.

Aquí, A comienza por cifrar su identidad y un número al azar, R_A, usando la clave pública (o de cifrado) de B, E_B. Cuando B recibe este mensaje, no sabe si vino de A o de C, pero entra en el juego y devuelve a A un mensaje que contiene el R_A de A, su propio número aleatorio, R_B, y la clave de sesión propuesta K_S.

Cuando A recibe el mensaje 2, lo descifra usando su clave privada y encuentra R_A en él, lo que le hace sentirse cómodo. El mensaje debió haber venido de B, puesto que C no tiene manera de determinar R_A.

Es más, el mensaje debe ser reciente y no una repetición, puesto que acaba de enviar R_A a B. A acepta la sesión devolviendo el mensaje 3. Cuando B ve R_B cifrado con la clave de sesión que él acaba de generar, sabe que A recibió el mensaje 2 y comprobó R_A.

El protocolo sin embargo tiene una debilidad, ya que supone que A y B conocen la clave pública del otro. Supongamos que no es así. A podría simplemente enviar a B su clave pública en el primer mensaje y pedir a B que devuelva la suya en el siguiente. El problema de este enfoque es que está sujeto a un ataque de brigada de cubetas. C puede capturar el mensaje de A a B y devolver su propia clave a A. A pensará que tiene una clave para hablar con B cuando, de hecho, tiene una clave para hablar con C. Ahora C puede leer todos los mensajes cifrados con lo que A piensa es la clave pública de B. El intercambio inicial de claves públicas puede evitarse almacenando todas las claves públicas en una base de datos pública. Así, A y B pueden obtener la clave pública del otro de la base de datos. Sin embargo, C aún puede poner en práctica el ataque de brigada de cubetas interceptando las solicitudes a la base de datos y enviando respuestas simuladas que contengan su propia clave.

Rivest y Shamir han diseñado un protocolo que fustra el ataque de brigada de cubetas de C. En su protocolo de interbloqueo, tras el intercambio de claves públicas, A envía solo la mitad de su mensaje a B al mismo tiempo que B espera la mitad de A, digamos por ejemplo que sólo los bits pares (después del cifrado). B entonces responde con sus bits pares y A también responde con sus bits pares. Tras recibir los bits pares de A y B, A envía sus bits nones, y luego lo hace B. El truco aquí es que, cuando C recibe los bits pares de A o de B, no puede descifrar el mensaje, aunque tiene la clave privada. En consecuencia, C es incapaz de volver a cifrar los bits pares usando la clave pública de A o de B. Si C envía basura a B o a A, el protocolo continuará, pero B o A pronto se dará cuenta de que el mensaje reensamblado no tiene sentido y de que ha sido engañado.

TACACS, XTACACS, TACACS+, RADIUS y otros

Todos son protocolos e implementaciones de control de acceso por validación y autenticación, que permiten que un servidor del acceso de red (**NAS** *Network Access Servers o* **RAS** *Remote Access Server*, por ejemplo un router Cisco 2511 o 5300) obtenga datos de administación del usuario a un servidor central y por tanto se descarge de la tarea administrativa de autenticación y comprobación de dicha información.

También, cabe considerar otro tipo de servidor para autenticación muy extendido en las redes Microsoft llamado Active Directory, que permite gestionar los usuarios y sus permisos (exportando incluso el escritorio) independientemente de la máquina conectada dentro de un dominio Microsoft. Los dominios Microsoft son gestionados por el Controlador de Dominio (Domain Controller, DC). Además, es importante resaltar que Active Directory incluye Kerberos y servidores de directorio como veremos a continuación, concretamente LDAP.

Todos ellos se pueden considerar el soporte a los sistemas de autenticación basado en contraseñas, no son considerados sistemas de clave simétrica compartida, porque en un principio no están pensados para intercambiar claves para cifrar, simplemente para poder autenticar.

Por tanto, como parte de estos protocolos de autenticación, hemos incluido estos protocolos muy utilizados en los sistemas de acceso en red, desde conexiones remotas.

Hay tres versiones del protocolo de autenticación "TACACS" (Terminal Access Controller Access Control System, sistema de control de acceso del Controlador de Acceso de Terminales). El primer es TACACS ordinario, fue el protocolo utilizado por Cisco en sus NAS, y ha estado en el uso por muchos años. El segundo es una extensión al primero, comúnmente llamado Extended Tacacs o XTACACS, introducido en 1990. Ambos se documentan en RFC1492. El tercero, TACACS+ que, a pesar del nombre, es totalmente un nuevo protocolo y no es compatible con TACACS o XTACACS.

TACACS y XTACACS carecen de muchas características de TACACS+ y RADIUS, y actualmente están fuera de mantenimiento. Debido a esto TACACS y XTACACS no serán discutidos.

RADIUS (Remote Authentication Dial In User Service) es un protocolo de control de accesos desarrollado por Livingston Enterprises y que la IETF ha recogido en los RFCs 2865 y 2866. Fue diseñado para autenticar usuarios y utiliza una arquitectura cliente/servidor. El servidor contiene información de los usuarios, almacenando sus contraseñas y sus perfiles, y el cliente es el encargado de pasar las peticiones de conexión de los usuarios al servidor para que éste las autentique y responda al cliente diciéndole si ese usuario está o no registrado.

Un ejemplo de uso de RADIUS se puede dar en un ISP, donde el NAS (Network Access Server) hace de cliente RADIUS y un host del ISP podría hacer de servidor RADIUS. El NAS recibe vía módem una petición de acceso y envía los datos que el usuario ha proporcionado al servidor RADIUS. Es éste el que consulta su base de datos y comprueba si el usuario que ha realizado la llamada es en realidad quien dice ser. En ese caso, responde al NAS con una respuesta de aceptación de la llamada y, opcionalmente, con datos sobre el perfil del usuario (tipo de servicio, protocolo de conexión, IP asignada, ...). En caso contrario notifica al NAS que debe rechazar la petición de conexión. Opcionalmente, el servidor RADIUS guardará **logs (*informes o históricos del sistema para auditoría***) de las conexiones en las que estemos interesados.

Además de estos logs, RADIUS también puede hacer de servidor para registrar y almacenar todos los logs que acontecen en el NAT o RAS. Este paso está descrito e implementado en los routers de Cisco Systems con la AAA Authentication, Authoring y Accounting, quién es, qué puede hacer y auditoría respectivamente.

APLICACIONES DE SEGURIDAD: FIRMA DIGITAL Y CERTIFICADOS

Firmar un documento en realidad de forma tradicional, con bolígrafo, consiste en afirmar que lo escrito en el documento es cierto y con la estampación a mano de la firma, estando yo físicamente como persona, corroboro que soy yo quien afirma dicho documento. En este proceso se genera una copia adicional que sirve como resguardo para posterior comprobación.

En el procedimiento electrónico hacemos igual, de forma que la integridad del documento se obtiene a través de unos compendios y acuses de recibo, involucrando a las personas interesadas, y la validación e identificación del firmante se realiza a través de claves, que corroboran y permiten afirmar que yo soy quien digo ser. En este proceso participan o terceros que certifican nuestra identidad o terceros en los cuales tenemos confianza y a su vez ellos confían en otros. Esto lo veremos más detalladamente.

Pasemos a ver a continuación dichos elementos.

Resolución del control de integridad.

En muchas situaciones se requiere conocer la validez de un mensaje, esto es, conocer que no ha sido modificado por el camino y que nos llega tal y como fue escrito originalmente. A continuación describiremos un esquema de validación de integridad. Este esquema se basa en la idea de una función de dispersión unidireccional que toma una parte arbitrariamente grande de texto común y a partir de ella calcula una cadena de bits de longitud fija. En algunos contextos también se llama "**huella digital**" a este proceso de resumen. La idea de integridad, viene a ser la misma idea de un CRC añadido al final de una trama, de forma que comprueba la integridad de las tramas recibidas.

En el caso de control de integridad, la función para generar el resumen del documento con el cual comprobar la integridad también se llama función de dispersión o compendio de mensaje (message digest), que tiene tres propiedades importantes:

1. Dado un texto P, es fácil calcular su compendio de mensaje MD(P).

2. Dado un compendio de mensaje MD(P), es imposible encontrar P.

3. Nadie puede generar dos mensajes que tengan el mismo compendio de mensaje.

Para cumplir con el tercer criterio, la dispersión debe ser de cuando menos de 128 bits de longitud, y preferentemente mayor.

Los compendios de mensaje funcionan tanto en clave privada como en clave pública, siendo los de mayor uso el MD5 y el SHA.

Compendio de mensaje MD5.

El MD5 es la quinta de una serie de funciones de dispersión diseñadas por Ron Rivest. Opera alterando los bits de una manera tan complicada que cada bit de salida es afectado por cada bit de entrada. Su algoritmo es el siguiente:

- Se coge el mensaje original y se rellena hasta alcanzar una longitud de 448 módulo 512 bits, esto es, el mensaje debe tener una longitud en bits tal que al dividirla por 512 proporcione como resto de la operación el valor 448.

- A continuación se añade al mensaje la longitud original del mismo como un entero de 64 bits, por lo cual el mensaje total a codificar es un múltiplo de 512 bits.

- Se inicializa un buffer de 128 bits con un valor fijo.

- Ahora empieza el cálculo del compendio. Cada ronda toma un bloque de 512 bits de entrada y lo mezcla por completo con el buffer de 128 bits y los valores de una tabla construida a partir de la función matemática seno. Este proceso continúa hasta que todos los bloques de entrada se han consumido.

Una vez terminado el cálculo, el buffer de 128 bits contiene el valor del compendio de mensaje.

Compendio de mensaje SHA.

El SHA (Secure Hash Algoritm), fue desarrollado por la NSA y procesa los datos de entrada en bloques de 512 bits, pero a diferencia del MD5 genera un compendio de mensaje de 160 bits. Su algoritmo es el siguiente:

- Se coge el mensaje original y se rellena hasta alcanzar una longitud de 448 módulo 512 bits, esto es, el mensaje debe tener una longitud en bits tal que al dividirla por 512 proporcione como resto de la operación el valor 448.

- A continuación se añade al mensaje la longitud original del mismo como un entero de 64 bits, por lo cual el mensaje total a codificar es un múltiplo de 512 bits.

- Se inicializa un buffer de 160 bits con un valor fijo.

- Ahora empieza el cálculo del compendio. Cada ronda toma un bloque de 512 bits de entrada y lo mezcla con el buffer de 160 bits, utilizando 80 rondas para cada bloque de entrada y modificando cada 20 rondas las funciones de mezcla del bloque y el buffer. Este proceso continúa hasta que todos los bloques de entrada se han consumido.

Una vez terminado el cálculo, el buffer de 160 bits contiene el valor del compendio de mensaje. El SHA es 2^{32} veces más seguro que el MD5, pero es más lento su cálculo que el cálculo del MD5.

Hasta la fecha, tanto el MD5 como el SHA se han mostrado inviolables, pues aún con ataques sofisticados ideados (como el ataque de cumpleaños), se tardarían más de 500 años en calcular los compendios de dos cartas con 64 variantes cada una, e incluso entonces no se garantiza una equivalencia.

Resolución del repudio: Firmas digitales.

La validación de identificación y autenticidad de muchos documentos legales, financieros y de otros tipos se determina por la presencia o ausencia de una firma manuscrita autorizada. Para que los sistemas computerizados de mensajes reemplacen el transporte físico de papel y tinta, debe encontrarse una solución a estos problemas.

El problema de inventar un reemplazo para las firmas manuscritas es difícil. Básicamente, lo que se requiere es un sistema mediante el cual una parte pueda enviar un mensaje "firmado" a otra parte de modo que:

1. El receptor pueda verificar la identidad proclamada del transmisor.

2. El transmisor no pueda repudiar después el contenido del mensaje.

3. El receptor no haya podido confeccionar el mensaje él mismo.

El primer requisito es necesario, por ejemplo, en los sistemas financieros. Cuando la computadora de un cliente ordena a la computadora de un banco que compre una tonelada de oro, la computadora del banco necesita asegurarse de que la computadora que da la orden realmente pertenece a la compañía a la que se le aplicará el débito.

El segundo requisito es necesario para proteger al banco contra fraudes. Supongamos que el banco compra una tonelada de oro, e inmediatamente después cae el precio del oro. Un cliente deshonesto podría demandar al banco, alegando que nunca emitió una orden para comprar el oro. Cuando el banco presenta el mensaje ante el juez, el cliente niega haberlo enviado.

El tercer requisito es necesario para proteger al cliente en el caso de que el precio del oro suba y que el banco trate de falsificar un mensaje firmado en el que el cliente solicitó un lingote de oro en lugar de una tonelada.

Al igual que la criptografía, las firmas digitales se dividen en dos grandes grupos, firmas de clave secreta y firmas de clave pública.

Firmas de clave secreta.

Un enfoque de las firmas digitales sería tener una autoridad central que sepa todo y en quien todos confíen, digamos X. Cada usuario escoge entonces una clave secreta y la lleva personalmente a las oficinas de X. Por tanto, sólo A y X conocen la clave secreta de A, K_A, etc.

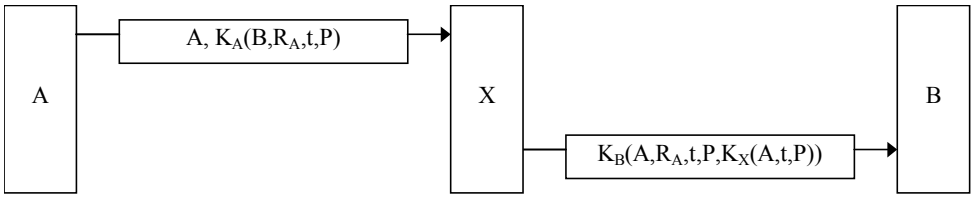


Figura 18: firma con clave secreta compartida con centro de distribución de claves

Cuando A quiere enviar un mensaje de texto normal firmado, P, a su banquero, B, genera $K_A(B,R_A,t,P)$ y lo envía como se muestra en la figura anterior. X ve que el mensaje es de A, lo descifra y envía un mensaje a B como se muestra. El mensaje a B contiene el texto normal del mensaje de A y también el mensaje firmado $K_X(A,t,P)$, donde t es una marca de tiempo. Ahora B atiende la solicitud de A.

Si ahora A niega el envío del mensaje, cuando el caso llega al juez y A niegue haber enviado a B el mensaje, B indica que el mensaje vino de A y no de un tercero C pues X no hubiera aceptado un mensaje de A a menos que estuviese cifrado con K_A, por lo que no hay posibilidad de que C enviará a X un mensaje falso en nombre de A. B además presenta la prueba $K_X(A,t,P)$. Entonces el juez pide a X (en quien todo el mundo confía) que descifre la prueba. Cuando X testifica que B dice la verdad el caso queda resuelto.

Un problema potencial del protocolo de firma anterior es que C repita cualesquiera de los dos mensajes. Para minimizar este problema, se usan en todos los intercambios marcas de tiempo. Es más, B puede revisar todos los mensajes recientes para ver si se usó R_A en cualquiera de ellos. De ser así, el mensaje se descarta como repetición. Nótese que B rechazará los mensajes muy viejos con base en la marca de tiempo. Para protegerse contra ataques de repetición instantánea, B simplemente examina el R_A de cada mensaje de entrada para ver si un mensaje igual se recibió de A durante el tiempo de validez de la marca temporal. Si no, B puede suponer con seguridad que ésta es una solicitud nueva.

Firmas de clave pública.

Un problema estructural del uso de la criptografía de clave secreta para las firmas digitales es que todos tienen que confiar en X. Es más, X lee todos los mensajes firmados. Los candidatos más lógicos para operar el servidor X son el gobierno, los bancos y los abogados. Estas organizaciones no tiene porqué inspirar confianza completa a todos los ciudadanos. Por tanto, sería bueno si la firma de documentos no requiriese una autoridad confiable.

Afortunadamente, la criptografía de clave pública puede hacer una contribución importante aquí. Supongamos que los algoritmos públicos de cifrado y descifrado tienen la propiedad de que E(D(P))=P

además de la propiedad normal de D(E(P))=P (el RSA tiene esta propiedad por lo que el supuesto es razonable). Suponiendo que éste es el caso, A puede enviar un mensaje de texto normal firmado y cifrado, P, a B transmitiendo $E_B(D_A(P))$. Firmado por $D_A(P)$ y cifrado por $E_B()$, de forma que sólo B podrá leerlos. Nótese que A conoce su propia clave de descifrado (privada), D_A, así como la clave pública de B, E_B, por lo cual la construcción de este mensaje es algo que A puede hacer.

Cuando B recibe el mensaje, lo transforma usando su clave privada, como es normal, produciendo $D_A(P)$, como se muestra en la figura siguiente:

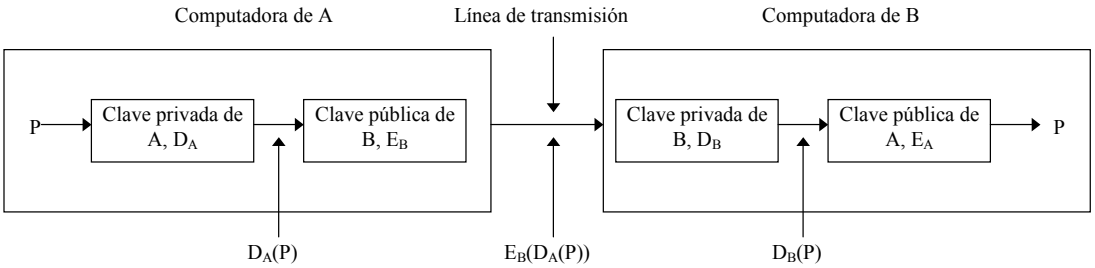


Figura 19: firma digital con clave pública

B almacena este texto en un lugar seguro y lo descifra usando E_A para obtener el texto normal original.

En todo el proceso, en resumen lo que se ha realizado es $E_A(D_B(E_B(D_A(P))))$.

Para ver cómo funciona la propiedad de firma, supongamos que A niega haber enviado el mensaje P a B. Cuando el caso llega al juez, B puede presentar tanto P como $D_A(P)$. El juez puede comprobar fácilmente que B tiene un mensaje válido cifrado por D_A con solo aplicarle E_A. Puesto que B no conoce la clave privada de A, la única forma en que B pudo haber adquirido el mensaje cifrado con ella sería que A en efecto lo hubiera enviado.

Sin embargo existen dos problemas, por un lado B puede demostrar que un mensaje fue enviado por A siempre y cuando D_A permanezca en secreto. Si A divulga su clave secreta, el argumento ya no se mantiene. Por otro lado, si A decide cambiar su clave, algo legal y probablemente buena idea de forma periódica, cuando se aplique la actual E_A a $D_A(P)$ no se obtiene P. En consecuencia, parece que sí que se requiere alguna autoridad para registrar todos los cambios de clave y sus fechas.

En principio cualquier algoritmo de clave pública puede usarse para firmas digitales. El estándar de facto de la industria es el algoritmo RSA y muchos productos de seguridad lo usan.

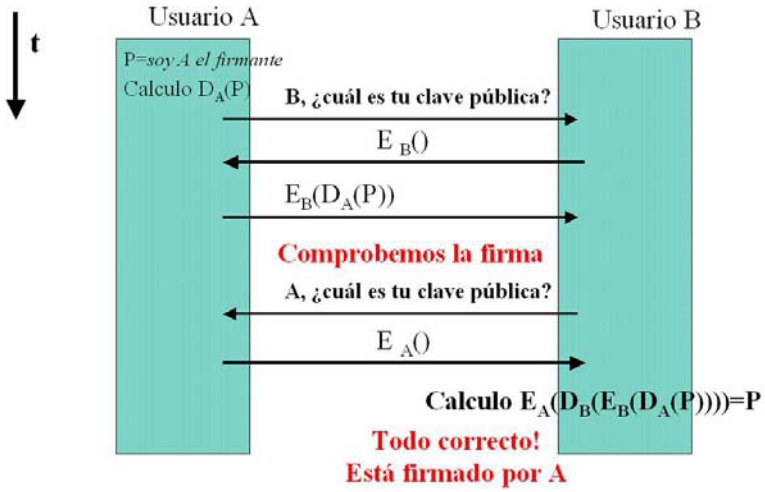


Figura 20: protocolo de firma digital con clave pública

Con todo ello, el protocolo quedaría tal como se ve en la figura 20.

Un ejemplo de comprobación de utilización de firma digital lo podemos ver en la comprobación de paquetes software "rpm" *en algunas distribuciones de Linux. Para comprobar la firma digital la aplicación* ***rpm*** *posee la opción "checksig": "rpm –checksig X.i386.rpm*" siendo *X.i386.rpm* el paquete a instalar. Esta opción en *rpm* permite comprobar la integridad del paquete y el origen. La **integridad** se comprueba con la firma basada con **md5.** La **comprobación del origen** (validación) se realiza con la clave pública de GNU (si se dispone). Si no disponemos de dicha firma, la comprobación de origen no puede realizarse y con ello, la GPG (*Gnu Privacy Guard*) no podrá efectuarse:

$$\mathbf{E_{GNU}(D_{GNU}(}\textit{compendio-rpm}\mathbf{))}=\textit{compendio-rpm}$$

Aplicaciones seguras

Visto todo el conjunto de protocolos, tanto para intercambiar información cifrada, autenticar y validar, generar soluciones al no repudio y el proceso de firma digital, pasemos a ver aplicaciones que catalogamos como seguras que hacen uso de estos elementos o están vinculadas con ellas.

Aplicaciones seguras y uso de certificados

Uno de los problemas que parecen en la distribución de claves públicas, es la propia distribución de dicha clave, que como hemos visto puede sufrir del ataque de alguien en medio. Una solución es la aplicación de métodos de interbloqueo, pero la solución más adoptada y estandarizada en la utilización de certificados.

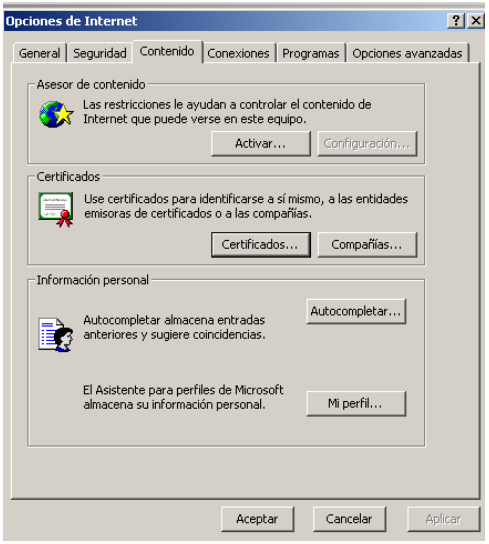


Figura 20: Configuración de Certificados en Herramientas: Opciones de Internet: Contenidos: Certificados en Internet Explorer 5.0

El **certificado digital** es un vínculo entre una clave pública y una identidad, que se consigue mediante una firma digital por una tercera parte o autoridad de certificación (***CA****: Certificaciont Authority o Autoridad de Certificación*) comprobando la integridad. Se considera como un objeto firmado con: *identidad del sujeto, clave, periodo de validez, identidad emisor*, ... La Fábrica Nacional de Moneda y Timbre (FNMT), Verisign, Deutsche Telecom., Microsoft, Xcert,... son diferentes autoridades de certificación.

A modo de ejemplo, en el caso de pedir un certificado para un ciudadano español, su certificado sería un archivo, firmado con la clave privada de CA con la identidad, la clave pública del dicha identidad, atributos varios y compendio de dicha información:

$$\mathbf{D_{CA}(identidad}, \mathbf{clave}, \mathbf{atributos}, \textit{compendio}\{\textbf{\textit{identidad, clave, atributos}}\})$$

La mayoría de navegadores disponen ya en la distribución, en los propios CDs de instalación, un directorio lleno de **certificados raiz**, es decir, certificados de las propias autoridades de certificación firmados por ellas mismas. Vemos una lista de los certificados raiz más habituales en los navegadores:

Enviado a	***Emitido por***
ABA.ECOM Root CA	ABA.ECOM Root CA
AC del Colegio Nacional de Correduria Publica Mexicana, A.C.	AC del Colegio Nacional de Correduria Publica Mexicana, A.C.
Baltimore EZ by DST	Baltimore EZ by DST
Belgacom E-Trust Primary CA	Belgacom E-Trust Primary CA
C&W HKT SecureNet CA Class A	C&W HKT SecureNet CA Class A
CA 1	CA 1
Certiposte Classe A Personne	Certiposte Classe A Personne
Certiposte Serveur	Certiposte Serveur
Certisign - Autoridade Certificadora - AC2	Certisign - Autoridade Certificadora - AC2
Class 1 Primary CA	Class 1 Primary CA
Copyright (c) 1997 Microsoft Corp.	Copyright (c) 1997 Microsoft Corp.
Deutsche Telekom Root CA 1	Deutsche Telekom Root CA 1
DST (ANX Network) CA	DST (ANX Network) CA
DST (NRF) RootCA	DST (NRF) RootCA
DST-Entrust GTI CA	DST-Entrust GTI CA
Entrust.net Secure Server Certification Authority	Entrust.net Secure Server Certification Authority
Equifax Secure Certificate Authority	Equifax Secure Certificate Authority
FNMT Clase 2 CA	**FNMT Clase 2 CA**
GlobalSign Root CA	GlobalSign Root CA
http://www.valicert.com/	http://www.valicert.com/
IPS SERVIDORES	IPS SERVIDORES
Microsoft Authenticode(tm) Root Authority	Microsoft Authenticode(tm) Root Authority
Microsoft Root Authority	Microsoft Root Authority
NetLock Uzleti (Class B) Tanusitvanykiado	NetLock Uzleti (Class B) Tanusitvanykiado
PTT Post Root CA	PTT Post Root CA
Saunalahden Serveri CA	Saunalahden Serveri CA
Secure Server Certification Authority	Secure Server Certification Authority
SecureNet CA Class A	SecureNet CA Class A
SecureSign RootCA	SecureSign RootCA
SERVICIOS DE CERTIFICACION - A.N.C.	SERVICIOS DE CERTIFICACION - A.N.C.
SIA Secure Client CA	SIA Secure Client CA
SIA Secure Server CA	SIA Secure Server CA
Swisskey Root CA	Swisskey Root CA
TC TrustCenter Class CA	TC TrustCenter Class CA
TC TrustCenter Time Stamping CA	TC TrustCenter Time Stamping CA
Thawte Personal CA	Thawte Personal CA
UTN - DATACorp SGC	UTN - DATACorp SGC
UTN-USERFirst-Client Authentication and Email	UTN-USERFirst-Client Authentication and Email
VeriSign Commercial Software Publishers CA	VeriSign Commercial Software Publishers CA
VeriSign Individual Software Publishers CA	VeriSign Individual Software Publishers CA
VeriSign Trust Network	VeriSign Trust Network
Xcert EZ by DST	Xcert EZ by DST

La **autoridad de certificación** es reconocida y aceptada por todos, e imposible de suplantar. Por regla general, por seguridad no se trabaja directamente con la autoridad de certificación, si no con un intermediario o autoridad de registro.

La **autoridad de registro** atiende las peticiones de certificado y proporciona diferentes métodos de petición de certificados.

Para implementar el sistema de autenticación en base a certificados a través de Autoridad de Certificación (CA) han de considerarse los siguientes elementos:

- Una política de certificación, incluyendo el ámbito de actuación (identificando los elementos a certificar, tanto personas como procesos, como servidores y/o clientes) así como la relación de la CA con otras CA, por regla general de forma jerárquica
- Un certificado de la CA, de otra CA superior u homóloga que asegure quien dice ser y valida su clave pública
- Los certificados de los usuarios (p.ej X.509), así como el procedimiento de certificación y de revocación. La revocación permite consultar directamente qué certificados no son válidos
- Los protocolos de autenticación, gestión y obtención de certificados, así como de distribución de ellos son como veremos a continuación:

–o por bases de datos (p.ej directorio X.500) o comúnmente llamados servidores de directorios, que veremos a continuación

–o bien directamente del usuario en tiempo de conexión (WWW con SSL, Secure Socket Layer)

Por tanto para la puesta en marcha de una CA se requiere generar un par de claves (tanto privada como pública), proteger la clave privada y generar el certificado de la propia CA a través de otra CA.

El **certificado raíz** es un certificado emitido de la CA para sí misma con su clave pública, para comprobar certificados emitidos por ella. Se suele instalar previamente dicho certificado en el navegador para poder utilizar los certificados de dicha CA. Los navegadores llevan por defecto muchos de ellos.

Lista de certificados revocados (o CRL *Certificate Revocation List***)** es una lista donde se recogen todos los certificados de la CA *dados de baja por caducidad o por problemas varios* como que se haya hecho pública la clave privada de un usuario, y por tanto cualquier firma emitida con posterioridad a la revocación no tiene validez. Este documento también es firmado por la propia CA.

Pero, esta infraestructura de autoridades en muchas ocasiones no se dispone, ni de Autoridades de Certificación ni de Registro. Con este planteamiento inicial una solución tomada estriba en la confianza de los propios usuarios entre ellos. Este tipo de redes se llaman **redes de confianza**. Por ejemplo, si Juan confía plenamente en Luis y Luis ha aceptado la identificación de Pedro, Juan podría inicialmente aceptar a Pedro, porque Luis es de su confianza. Una implementación de este tipo de certificados lo ofrece PGP como veremos en el apartado de correo seguro.

Pasemos a ver ahora la estructura de los certificados X.509, ampliamente utilizados actualmente**Certificado X.509**

X.509 es el protocolo que se utiliza para certificar las claves públicas, con lo que los usuarios pueden intercambiar datos de manera segura. X.509 está basado en criptografía asimétrica y firma digital y se emplea para autentificar la información en redes externas, en redes internas, en el correo electrónico y en general en aplicaciones que requieran autenticación.

Para ello, el certificado se cifra con la clave privada de la CA y todos los usuarios poseen la clave pública del CA.

El protocolo X.509 fue creado por la UIT para servir al X.400 (correo electrónico de OSI) y su origen se encuentra en el servicio de directorio X.500. En X.509 se define un framework (una capa de abstracción) para suministrar servicios de autenticación a los usuarios del directorio X.500.

El objetivo de este protocolo es asegurar la integridad, la privacidad y el no repudio de los mensajes.

X.509 se utiliza en SSL en los navegadores (https), PEM en el correo electrónico seguro, S/MIME, SET (Secure Electronic Transaction) que define una arquitectura para E-CommerceLos campos del X.509 escritos en ASN1 son:Versión: La del protocolo X.509 (actualmente versión 3)Número de serie: Es un número asignado por el CA y que identifica de manera única el certificado.Algoritmo de la firma del certificado: Identifica el algoritmo utilizado para firmar el certificado.Autoridad de certificación (CA): Es el nombre de la CA.Fecha de inicio y final: tiempo de validezUsuario: Es el nombre del usuario.Clave pública: Es la clave del usuario.Identificador único del CA: Es el número que identifica al CA. Es único en el mundo.Identificador único del usuario: Es el número que identifica al usuario para todos sus certificados.Extensiones: Si hay extensiones de la información

- Firma de la CA: Firma todos los campos anteriores empleando, para ello, su clave privada.

Servidor de directorios.

La idea de un directorio es el de un listín de teléfonos. Pero si queremos utilizarlo en el ámbito para organizar la información de una institutción etc, requiere que sea electónico, para permitir modificaciones en tiempo real, búsquedas aproximadas, etc.

Los servidores de directorio están basados en el protocolo X.500, primer protocolo de servidor de directorios, conocido como padre, y está pensado para guardar todo tipo de información referente a una institución, tanto de personas (datos personales, correo, teléfono), como de objetos (salones de actos, sala

de reuniones,...), pero X.500 debido a que es complicado, se pensó en una simplificación: LDAP (Lighweight Directory Access Protocol)(RFC 1478 y 2251).

La relación de los servidores de directorios con los certificados, es que tras largos debates, este tipo de servidores cumplen las características idóneas para albergar los certificados de los usuarios en una corporación.

LDAP es una simplificación de X.500 y se considera como una base de datos jerárquica orientada a objetos, a la cual se realizan consultas (búsquedas e inserciones) de registros. Estos registros pueden contener cualquier tipo de información, pero generalmente en una red incluyen información con usuarios y perfiles, para autenticar de forma muy rápida. Los mecanismos de consulta de esta base de datos es propietario, es decir no tiene similitud con SQL (Structured Queueing Language).

LDAP y en general los servidores de directorios han aprovechado la rapidez en las lecturas, frente a las bases de datos. Las bases de datos de propósito general están optimizadas para lecturas/escrituras, mientras que los servidores de directorios fundamentalmente las peticiones que van a atender son de lectura. Otra opción a las bases de datos para poder gestionar esta información, el servicio de directorios es el servicio DNS. Sin embargo hubo detractores a utilizar DNS como servidor de directorios y por tanto se optó por crear un servidor específico e independiente.

LDAP está basado en arquitectura de cliente/servidor y puede utilizar Kerberos en el proceso de autenticación. Los servicios ofrecidos por los servidores se sirven en el puerto 389 normalmente, aunque se puede configurar, además los servidores permiten varias peticiones en curso.

LDAP se utiliza principalmente para realizar consultas que permitan averiguar qué permisos puede tener un usuario dentro de la red controlada por servidores LDAP para ejecutar ciertos servicios. Veamos un ejemplo de validación de permisos. En este caso vamos a considerar el acceso de los profesores a las actas por red de la Universitat de Valencia, acceso al control de actas. El protocolo implementado es como sigue:

1. el cliente (usuario o profesor) quiere conectarse al gestor de actas
2. tras la identificación del cliente (usuario), el gestor de actas consulta al servidor LDAP para averiguar los permisos del cliente (usuario) en el acceso a la gestión de actas.
3. según los permisos, atiende al cliente

En esta base de datos, las clases no tienen métodos, sólo atributos, y los atributos son propiedades de una clase y cada atributo es un nombre (tipo) y una lista de valores. El nombre del atributo es una cadena de caracteres o un OID (Object Identifier) y se pueden estructurar jerárquicamente permitiendo herencias. Ejemplo de estos atributos son:

CN	Nombre habitual o canónic name
O	Organización
OU	Departamento
C	País
MAIL	Dirección de correo electrónico
PHONE	Número de teléfono
DC	Componente de dominio
DN	Nombre Distinguido y cada entrada tiene uno, único y propio

Las aplicaciones más frecuentes de LDAP son muy variadas, desde directorio de correo (email) para encontrar la dirección de correo de alguien o encontrar el certificado digital de alguien, pasando por una libreta de direcciones y teléfonos, o haciendo una sustitución de NIS (difusión de ficheros de contraseñas).

Otra aplicación de LDAP puede ser la integración de todos los recursos de la organización: personal, utilización y disponibilidad de salas de Reuniones, cuentas de usuario. Por ejemplo una forma de identificar de forma unívoca a una asignatura, es a través de la cadena observada en la figura.

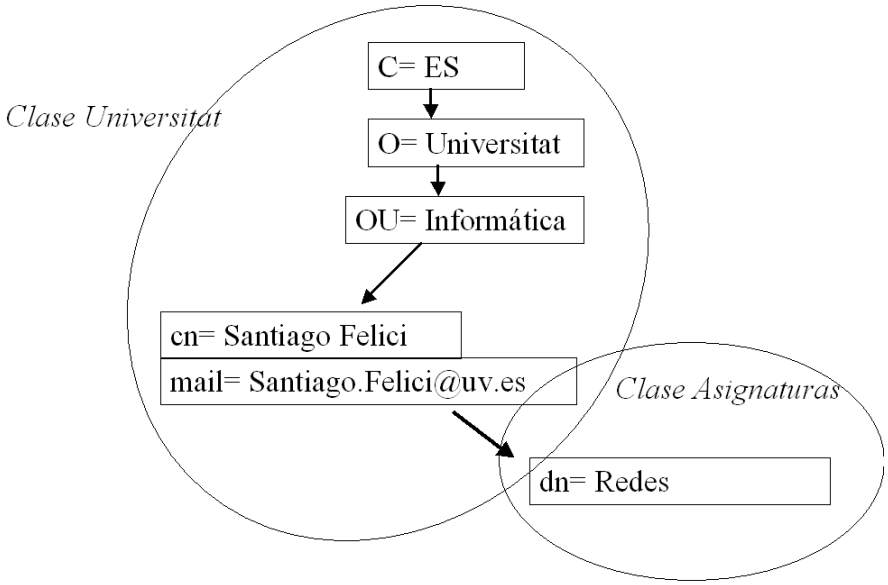


Figura 21: Ejemplo de jerarquía de clases y sus atributos en LDAP

El acceso de LDAP mediante JAVA, se realiza por JNDI (Java Naming Directory Interface) y como hemos dicho es una sintaxis propia, diferente a SQL. Existe un formato para exportar (a X.500) información entre servidores de directorios, conocido como LDIF.

Las distribuciones de LDAP más populares son:
-OpenLDAP (versión gratuita http://www.openldap.org)
-Iplanet Directory Server (versión comercial http://www.iplanet.com)

Aplicaciones seguras para confidencialidad del correo electrónico.

Cuando un mensaje de correo electrónico se envía entre dos sitios distantes, generalmente transitará por docenas de máquinas en el camino. Cualquiera de éstas puede leer y registrar el mensaje para su uso posterior, por lo que la confidencialidad es inexistente. No obstante, mucha gente quiere poder enviar correo electrónico que el destinatario pueda leer, pero nadie más. Este deseo ha estimulado a varias personas y grupos a aplicar principios criptográficos al correo electrónico para producir correo seguro. Veremos a continuación los compendios de mensajes, que permiten validar un mensaje, pasando a continuación a explicar dos de sistemas de correo electrónico seguro de amplio uso en Internet: PGP y PEM.

PGP: Bastante buena confidencialidad.

El primer sistema de correo electrónico seguro que veremos es PGP. PGP (Pretty Good Privacy, bastante buena confidencialidad) es el producto de una sola persona, Phil Zimmermann. PGP es un paquete completo de seguridad de correo electrónico que proporciona confidencialidad, validación de identificación, firmas digitales y compresión basado en el **envoltorio digital**, es decir un procesamiento completo que se le realiza al correo a mandar para que cumpla los requisitos fijados por PGP. El paquete completo, incluido código fuente, se distribuye por Internet de forma gratuita, por lo que es el de más amplio uso hoy en día.

El PGP intencionadamente usa algoritmos criptográficos existentes en lugar de inventar nuevos; PGP se basa en los algoritmos de encriptación RSA e IDEA y el compendio de mensaje MD5, algoritmos que han resistido análisis extensos de otras personas y no fueron diseñados ni influidos por ninguna agencia gubernamental que estuviera tratando de debilitarlos. Para ver como funciona PGP, consideremos la siguiente figura:

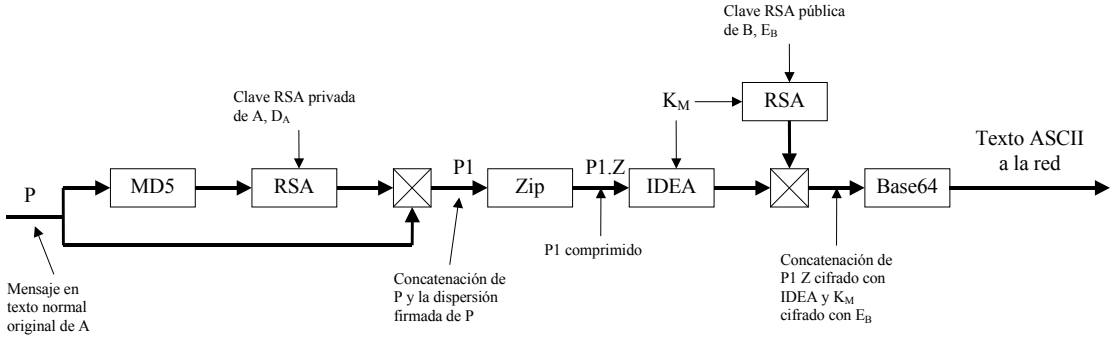


Figura 22: diagrama de bloques de cifrado en PGP

Aquí, A quiere enviar un mensaje de texto normal firmado, P, a B de una manera segura. Tanto A como B tienen claves RSA privadas (D_X) y públicas (E_X). Supongamos que cada uno conoce la clave pública del otro.

A comienza por invocar el programa PGP en su computadora. El PGP primero dispersa su mensaje, P, usando MD5, y luego cifra la dispersión resultante con su clave RSA privada, D_A. Cuando B recibe el mensaje, puede descifrar la dispersión con la clave pública de A y comprobar que es correcta.

La dispersión cifrada y el mensaje original ahora están concatenados en un solo mensaje P1, y comprimidos mediante el programa ZIP. Llamemos a la salida de este paso P1.Z.

A continuación, el PGP solicita a A una entrada al azar. Tanto el contenido como la velocidad de tecleo se usan para generar una clave de mensaje IDEA de 128 bits, K_M (llamada clave de sesión). K_M se usa ahora para cifrar P1.Z con IDEA en modo de realimentación de cifrado. Además, K_M se cifra con la clave pública de B, E_B. Estos dos componentes se concatenan y convierten a base64. El mensaje resultante contiene entonces sólo letras, dígitos y los símbolos +, / e =, lo que significa que puede ponerse en un cuerpo RFC 822 y esperar que llegue sin modificación.

Llamaremos todo este proceso realizado con el correo como envoltorio digital.

Cuando B recibe el mensaje deshace el envoltorio y para ello, invierte la codificación base64 y descifra la clave IDEA usando su clave RSA privada. Con la clave así obtenida, B descifra el mensaje para obtener P1.Z. Tras descomprimir esto, B separa el texto normal de la dispersión cifrada y descifra la dispersión usando la clave pública de A. Si la dispersión del texto normal concuerda con su propio cálculo MD5, sabe que P es el mensaje correcto y que vino de A.

La administración de claves ha recibido mucha atención en el PGP, puesto que es el punto débil de todos los sistemas de seguridad. Cada usuario mantiene localmente dos estructuras de datos: un anillo con claves privadas y un anillo de claves públicas. Estos anillos se llaman **repositorios de llaves o llaveros**. El anillo de claves privadas contiene uno o más pares de clave privada-clave pública personales. La razón para reconocer varios pares por usuario es permitir a los usuarios cambiar sus claves públicas periódicamente o cuando se piensa que una está comprometida, sin invalidar los mensajes que actualmente están en preparación o en tránsito. Cada par asociado tiene un identificador, para que el remitente de un mensaje pueda indicar al destinatario la clave pública usada para cifrarlo. El anillo de claves públicas contiene claves públicas de los corresponsales del usuario; se requieren para cifrar las claves de mensaje asociadas a cada mensaje. Cada entrada del anillo de claves públicas contiene no sólo la clave pública, sino también su identificador y una indicación del nivel de confianza del usuario en la clave.

PEM: Correo con confidencialidad mejorada.

En contraste con el PGP, el PEM (Privaty Enhanced Mail, correo con confidencialidad mejorada), es un estándar oficial de Internet y se describe en cuatro RFC, del RFC 1421 al RFC 1424. De manera muy

general, el PEM cubre el mismo territorio que el PGP: confidencialidad y validación de identificación, para sistemas de correo electrónico basados en el RFC 822.

Los mensajes enviados usando PEM primero se convierten a una forma canónica para que puedan tener las mismas convenciones de espacios blancos (por ejemplo, tabuladores, espacios al final) y el uso de retornos de carro y avances de línea. Esta transformación se lleva a cabo para eliminar los efectos de los agentes de transferencia de mensajes que modifican los mensajes que no son de su gusto. Sin canonización, tales modificaciones pueden afectar las dispersiones de los mensajes en sus destinos.

A continuación, se calcula a dispersión del mensaje usando MD5. Después se cifra la concatenación con la dispersión y el mensaje usando DES. El mensaje cifrado puede entonces codificarse con codificación base64 y transmitirse al destinatario.

Como en el PGP, cada mensaje se cifra con una clave de una sola vez que se incorpora en el mensaje. La clave puede protegerse mediante RSA o con DES triple. En la práctica, se usa RSA pues el PEM no indica cómo se debe hacer la administración de claves con DES.

La administración de claves es más estructurada que en el PGP. Las claves se certifican mediante autoridades certificadoras que producen certificados indicando el nombre del usuario, su clave pública y la fecha de expiración de la clave. Cada certificado tiene un número de serie único que lo identifica. Los certificados incluyen una dispersión MD5 firmada por la clave privada de la autoridad certificadora.

Aplicaciones seguras basadas en tarjetas inteligentes y PKI (Public Key Infraestructure)

Utilización y tipo de tarjetas.

Las tarjetas son un elemento fundamental en el control de acceso a los edificios y permiten la identificación a través de ella, bien con códigos PIN (Personal Identification Number) o/y bien a través de fotografías escaneadas en su superficie. La tarjeta (genérica) es un dispositivo de plástico de dimensiones determinadas, capaz de almacenar y, en algunos casos, procesar información de manera segura. Fue inventado por *Roland Moreno* (periodista francés) a finales de los 70 y actualmente su complejidad es tal que utilizan *tecnología VLSI* con microprocesador y sistemas de ficheros cifrado.

Las utilidades que puede tener son almacenamiento y procesamiento de datos confidenciales, estado de las cuentas de crédito, historiales médicos (identificación de pacientes), números de identificación personal, claves privadas (control de acceso), dinero electrónico (micropagos)
Las propiedades que tienen las tarjetas son respecto a:
•seguridad, en el almacenamiento de la información (sistemas de fichero) y en las operaciones que realizan (clave simétrica y clave asimétrica)
•portabilidad de información, como historiales, claves, ...
•coste asequible
•utilizan protocolos propietarios para su programación, como para la comunicación lector-tarjeta, incluso la comunicación entre lectores entre sí, aunque en todos los protocolos existe una función básica para el procesado de revocación que consiste en la gestión de dos listas, *lista blanca* o de autorizados y *lista negra* o de personas excluidas (caducados, extraviados, revocados...)

El tipo de tarjetas existentes en el mercado hoy en día son:

- Magnéticas (ISO 7811): son muy extendidas, pero son de fácil fraude, con poco espacio de memoria 180 bytes. Son utilizada en accesos e identificación

- Con memoria (ISO 7816): con chip integrado que almacena información de forma “segura”. Son utilizadas como monederos, ejemplo tarjetas para teléfonos públicos. Se las conoce como *Chipcards*

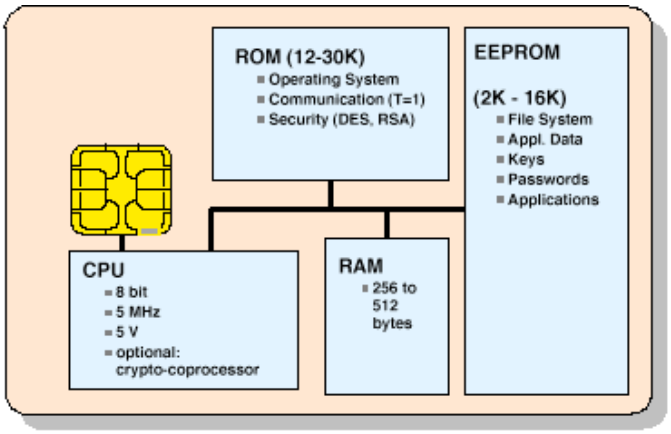


Figura 23: ejemplo de tarjeta inteligente con microprocesador, dónde la CPU está en la parte de acceso al bus interno de la tarjeta donde conecta la ROM, RAM y EEPROM

- **Con microprocesador** (ISO 7816/ISO 14443): son tarjetas seguras y pueden ser programadas, con memoria de entre 2-4kbytes. Se consideran multiaplicación, pero tienen el inconvenientes que sólo aceptan aplicaciones desarrolladas por el propio fabricante. Son utilizadas en módulos SIM, comercio electrónico, donde el usuario de identifica con PIN (personal identification number). Son conocidas como *SmartCards*

- **JAVA cards**: formada por capas, donde las aplicaciones (applets) y el S.O. Son independientes, utilizan un subconjunto de lenguaje JAVA, Java Card Api 2.1, permiten aplicaciones independiente del hardware escritas en lenguaje de alto nivel, son 10 a 200 veces mas lentas que las anteriores. Este tipo de tarjetas incorpora una máquina virtual JAVA y te permite la programación de Applets.

La utilización de las tarjetas y su aplicación dentro de una infraestructura de red, permite gestionar de forma segura muchas transacciones y el control de acceso a todos los servicios en una red. Este mecanismos o infraestructura se conoce como Infraestructura de Clave Pública (PKI).

PKI: Public Key Infraestructure

Una PKI se define como la infraestructura de red formada por servidores y servicios que en base a claves públicas gestionan de forma segura todas las transacciones realizadas a través de la red. Actualmente está en fase de estandarización con un RFC. Las operaciones básicas realizadas en una PKI son la certificación (medio por el cual las claves se publican) y la validación, a través de revocación y autenticación. Las claves públicas de los servicios, usuarios, aplicaciones, clientes, ... pueden ubicarse en servicios de directorios, en autoridades de certificación, en tarjetas inteligentes, ... Los fabricantes de PKI más representativos son RSA Labs, Verisign GTE Cyber Trust, Xcert, Netscape,....

La PKI como su nombre indica hace uso de los algoritmos de cifrado con clave pública y las aplicaciones (u operaciones) que se realizan con ellas son:

1. para cifrar, por ejemplo en el caso que A quiere mandar P a B, para ello utiliza la clave pública de B, enviando $E_B(P)$ y B utilizando su clave privada realiza el paso inverso D_B ($E_B(P)$)
2. para firmar y autenticar, por ejemplo en el caso que A quiere mandar P a B y firmarlo. Entonces para ello A manda {P, $D_A(P)$} utiliza su clave privada y B utilizando la clave pública de A, realiza la comprobación E_A ($D_A(P)$)

Una de las funciones importantes de la PKI es la validación de la información de los certificados. Para ello el usuario dispone de dos formas principales de validación, o preguntar por la validez a la CA (validación conocida como on line) y/o examinar el periodo de validez incluido en el certificado (off-line). El problema en esta última forma de validación, estriba en la complejidad de gestionar la revocación de certificados. La información del certificado es inválida si, la clave privada de la entidad se compromete, o si los datos de la entidad cambian.

Si la validación se realiza on-line, la revocación resulta sencilla. Si la validación se realiza off-line, la revocación se realiza mediante métodos como las listas de certificados revocados o CRLs (***Certificate Revocated List***), que es una lista de los certificados revocados, publicada y firmada periódicamente por la CA. El usuario chequea esta lista para comprobar la validez de los certificados.Las distribución y gestión

de las claves dentro de las PKI es lo que se llama repositorios (llaveros) de Certificados y son los elementos necesarios para almacenar los certificados de los usuarios. Para poder almacenarlos se suele utilizar servidores de directorios basados en X.500 y LDAP. Cabe destacar que los principales navegadores llevan soporte para servidores de directorio y también pueden gestionar las claves directamente con SSL (Secure Socket Layer) con total seguridad.
Las PKI pueden estar basadas en certificados X.509 y/o el soporte de repositorio de llaves (llaveros) de PGP, ya que éste incluye opción para clave pública, bien por confianza directa, es decir reside en los propios usuarios y confianza jerárquica, a través de una CA.

Aplicaciones seguras: Secure Socket Layer

Secure Socket Layer (SSL) es una pila de protocolos diseñada por Netscape y permite configurar al navegador en conexión segura. En el caso de una conexión web, aparece el protocolo ***https***. La pila de protocolos SSL está localizada sobre el nivel de transporte y es independiente del protocolo superior (interfaz similar a BSD Sockets). SSL protege del ataque de alguien al medio a través de certificados digitales X.509v3.

Obviamente el establecer sesiones seguras con SSL en la práctica reduce las prestaciones de un sistema normal, debido al cifrado en el establecimiento de la conexión

SSL es un conjunto de protocolos que se establecen entre la capa de transporte y aplicación en el modelo TCP/IP. Estos protocolos son Alert Protocol (AP), Handshake Protocol (HP), Change Cipher Spec (CCSP) y Application Data Protocol (ADP), que son protocolos que se ubican al mismo nivel, por debajo de la aplicación en sí. Por debajo de todos ellos, encargado de cifrar y fragmentar los paquetes se encuentra el Record Protocol (RP). Con esta pila de protocolos, cada sesión tiene sus parámetros criptográficos y no varían durante la sesión, salvo mensajes Change Cipher Spec (CCSP).

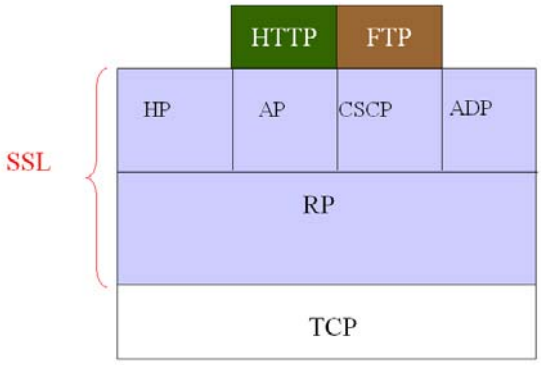


Figura 24. Pila de protocolos en SSL

Cuando el navegador ha negociado una conexión segura por SSL, es decir opera en modo de conexión segura, aparece el dibujo de un candado o llave en la barra inferior.

Los puertos utilizados para las aplicaciones que utilizan SSL son 443 para HTTPS, 465 para SSMTP, 563 para SNEWS, 636 para SS-LDAP, 995 para SPOP3.

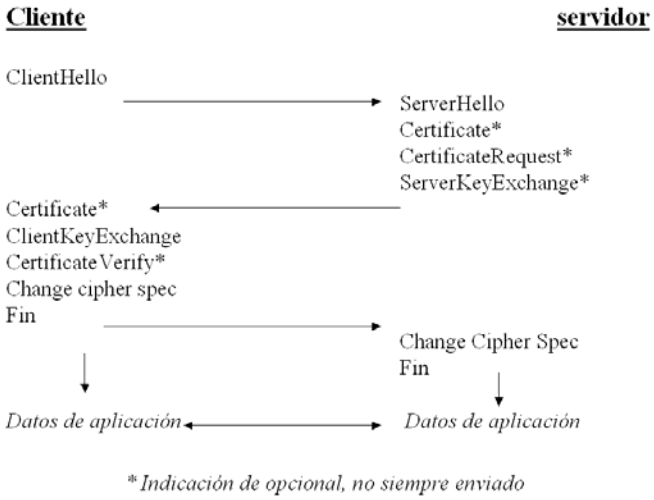


Figura 25. Figura de establecimiento de conexión con SSL utilizando HP y CSCP

El establecimiento de una sesión se realiza con *Handshake Protocol (HP) como podemos ver en la figura anterior y sigue los siguientes pasos:*

1. El **cliente inicia** la conexión y le **responde el servidor**
2. Se negocian los parámetros de conexión segura
3. El **servidor** *envía su certificado firmado* (opcional)
4. Certificado expedido por una CA, o validado *ad hoc*
5. Puede requerir un certificado del cliente
6. El **cliente** *envía una clave simétrica*
7. La genera y cifra (con la clave pública del servidor)
8. Se intercambian mensajes *Change Cipher Spec (CCSP)*
9. El cliente informa de los cifradores disponibles
10. El servidor elige el más fuerte común a ambos
11. Ya puede comenzar la comunicación cifrada

El protocolo AP *Alert Protocol es utilizado para el* intercambio de mensajes especiales, como Warning: para notificar avisos o Error (fatal) para indicar errores irrecuperables.

Una mejora del protocolo SSL es el Transport Layer Security (TLS), que es una generalización de SSL para ser un estándar no dependiente de un solo fabricante (Netscape). La v1.0 es el RFC-2246 del IETF (1.999), es una ligera modificación de SSL v3.0. Adicionalmente a *SSL*, tiene una extensibilidad del protocolo, ya que se pueden agregar nuevos protocolos por encima del *Record Protocol. Los protocolos que c*omprende son similares a los de SSL: Record Protocol (RP), Alert Protocol (AP), Handshake Protocol (HP), Change Cipher Spec (CCSP) y Application Data Protocol (ADP).

9.4 REDES Y SEGURIDAD

La seguridad es un concepto relativo, pues los mecanismos que minimizan la vulnerabilidad de un bien o recurso en un determinado caso, pueden ser insuficientes en otro caso. Esta suficiencia o insuficiencia vendrá determinada por la importancia de los bienes y recursos que posea, de forma que un ordenador que solo contenga contabilidad doméstica puede considerarse seguro sin la presencia de ningún mecanismo, mientras que un ordenador que contenga la contabilidad de una empresa debe poseer mecanismos para asegurar la imposibilidad de manipulación de la misma.

Las amenazas a la seguridad pueden clasificarse, atendiendo a la intencionalidad de las mismas en accidentales o intencionadas. Las amenazas accidentales son las que se producen sin necesidad de un intento premeditado, como por ejemplo una avería en el sistema. Las amenazas intencionadas pueden

variar desde el examen casual de la información de un ordenador hasta ataques sofisticados utilizando conocimientos especiales sobre el sistema.

Si en lugar de atender a la intencionalidad atendemos al daño ocasionado, las amenazas a la seguridad se clasifican en pasivas y activas. Las amenazas pasivas son las que no conllevan ninguna modificación en la información que posee el sistema y por tanto no se modifica ni su operación ni su estado. Las amenazas activas suponen una alteración del sistema y un cambio en su estado de operación. Aunque obviamente las amenazas activas son mucho más perjudiciales que las pasivas, estas deben ser tenidas en cuenta pues en muchos casos pueden convertirse en activas con la intervención de un agente distinto.

La seguridad adquiere cada vez más importancia a medida que la red informática se encuentra presente en más aspectos de la economía mundial, aspectos como el comercio electrónico, la transacción de información confidencial a través de la misma, etc.

Obviamente, la seguridad es un tema que debe inquietar a cualquier organización que hoy día decida conectar su red a otras sobre Internet. Basta echar un vistazo a las estadísticas para tomar conciencia del riesgo que se corre: el número de incidentes contra sistemas conectados casi se duplica cada año,según el Computer Emergency Response Team Coordination Center (CERT-CC). Y no debeextrañarnos, si tenemos en cuenta el vertiginoso crecimiento de Internet en los últimos años, queimplica, por una parte, nuevas redes susceptibles de ser atacadas, y por otra, nuevos atacantes en potencia.

Lo cierto es que tal y como están las cosas, atacar una red conectada a Internet que no haya sido protegida de un modo "especial" (es tan frecuente como erróneo creer que una filosofía de seguridad tradicional, basada en contraseñas y protección de ficheros, es suficiente para protegerse en Internet), es relativamente fácil si se sabe cómo, y mucho más aún si se utilizan sistemas operativos antiguos que no.han sido actualizados ni debidamente "parcheados". En la red es posible encontrar, sin mucho esfuerzo, listas de debilidades tanto de protocolos como de sistemas operativos, así como guías que señalan los pasos a seguir para explotar dichas debilidades. Incluso existen servidores de ftp anónimo con todo tipo de herramientas orientadas a tomar el control de cualquier máquina

Todas las líneas actuales de investigación en seguridad de redes comparten una idea: la concentración de la seguridad en un punto. Se obliga a que todo el tráfico entre la red que se pretende proteger y las redes externas pase por un mismo punto. Este punto se conoce con el nombre de cortafuego, y físicamente puede ser desde un simple host hasta un complejo conjunto de redes separadas por routers. El empleo de un cortafuego presenta enormes ventajas sobre los enfoques de seguridad en redes tradicionales (que requieren la seguridad individual de cada host conectado, y por tanto sólo pueden justificarse en entornos con un reducido número de máquinas), permitiendo concentrar todos los esfuerzos en el control de tráfico a su paso por el cortafuego.

Peligros y modos de ataque

El proceso de diseñar un sistema de seguridad podría decirse que es el encaminado a cerrar las posibles vías de ataque. Se hace imprescindible, por tanto, adquirir un profundo conocimiento acerca de las debilidades que los atacantes aprovechan, y del modo en que lo hacen. La variedad de ataques posibles contra un sistema es excesivamente amplia y variada a primera vista. Sin embargo, analizándolos con más detenimiento, observamos que la mayoría de ellos no aprovechan una única debilidad, sino una combinación de éstas, y que en el fondo, el tipo de debilidades es, afortunadamente, más reducido. Sin embargo, eso tampoco es tranquilizador si, como veremos, hay problemas de difícil solución.

Últimamente vemos aparecer en la red diversas taxonomías de vulnerabilidades y tipos de ataques. Pasemos a ver una lista con los tipos de ataques que actualmente se pueden realizar sobre Internet, explicando brevemente en qué consiste cada uno y qué debilidades aprovecha.

- *Sniffing* : este ataque consiste en escuchar los datos que atraviesan la red, sin interferir con la conexión a la que corresponden. Se utiliza principalmente para obtener contraseñas, y en algunos casos para obtener información confidencial. Para proteger los contraseñas contra el sniffing basta con emplear mecanismos de autenticación y encriptación.

- *Spoofing* **:** es el nombre que se le da a los intentos del atacante por ganar el acceso a un sistema haciéndose pasar por otro que dispone de los privilegios suficientes para realizar la conexión. El ataque que más se suele utilizar sobre conexiones TCP es el conocido como *adivinación del número de secuencia*. Se basa en la idea de que si un atacante puede predecir el número inicial de secuencia de la conexión TCP generado por la máquina destino, entonces el atacante puede adoptar la identidad de máquina "confiada". . VER ANEXO 1.

- *Hijacking* **:** consiste en robar una conexión después de que el usuario ha superado con éxito el proceso de identificación ante el sistema. El ordenador desde el que se lanza el ataque ha de estar en alguna de las dos redes extremo de la conexión, o al menos en la ruta entre ambas. El único método seguro para protegerse contra este tipo de ataques es el uso de encriptación. . VER ANEXO 1.

- *Ingeniería social:* son ataques que aprovechan la buena voluntad de los usuarios de los sistemas atacados. Un ejemplo de ataque de este tipo es el siguiente: se envía un correo con el remite "root" a un usuario, en una gran red académica (donde frecuentemente los usuarios no conocen a los administradores), con el mensaje "por favor, cambie su contraseña a murcia1". El atacante entonces espera un poco, y entra con ese contraseña. A partir de ahí puede emplear otras técnicas de ataque (bugs del sistema para obtener un control total de la máquina, confianza transitiva para entrar en otras máquinas de la red, etc). Ante este tipo de ataques la mejor defensa es educar a los usuarios acerca de qué tareas no deben realizar jamás, y qué información no deben suministrar a nadie, salvo al administrador en persona.

- *Explotar bugs del software:* aprovechan errores del software. A la mayor parte del software se le ha añadido la seguridad demasiado tarde, cuando ya no era posible rediseñarlo todo. Además, muchos programas corren con demasiados privilegios, lo que les convierte en objetivo de los hackers, que únicamente han de hacerse con una copia del software a explotar y someterlo a una batería de pruebas para detectar alguna debilidad que puedan aprovechar. Entre los posibles ataques que se pueden efectuar está el desbordamiento de pila, consistente en introducir datos en la pila a través de funciones de entrada salida, de forma que permitan modificar las posiciones de retorno de las funciones y con ello ejecutar código que permite al atacante tomar control del sistema. Esta secuencia de ataques, se conoce como ***exploits***. La solución a este problema de desbordamiento se realiza a través de funciones de entrada salida limitada, Las funciones strcpy(), strcat(), gets(), son potencialmente vulnerables.

- *Confianza transitiva* **:** en sistemas Unix existen los conceptos de *confianza entre hosts y entre usuarios*. Se dice que un sistema es *confiado* para otro cuando desde el primero, cualquier usuario puede establecer un conexión al segundo sin necesidad de dar un contraseña. Se dice que un usuario sobre un sistema es *confiado* para otro sistema cuando ese usuario, desde el primer sistema, puede establecer un conexión al segundo sin necesidad de dar un contraseña. Así, cualquier atacante que tome el control de una máquina, probablemente podrá conectarse a otras gracias a la confianza entre hosts y/o entre usuarios

- *Ataques dirigidos por datos* **:** son ataques que tienen lugar en modo diferido, sin la participación activa por parte del atacante en el momento en el que se producen. El atacante se limita a hacer llegar a la víctima una serie de datos que al ser interpretados ejecutarán el ataque propiamente dicho.

- *Caballo de Troya* **:** un programa que se enmascara como algo que no es, normalmente con el propósito de conseguir acceso a una cuenta o ejecutar comandos con los privilegios de otro usuario.

- *Denegación de servicios* **:** estos ataques no buscan ninguna información contenida en las máquinas atacadas ni conseguir acceso a ellas. Únicamente van encaminados a impedir que sus usuarios legítimos puedan usarlas. El caso más típico es el *mail bombing:* envío de cantidades ingentes de correo a la máquina atacada hasta saturarla. Puesto que es casi imposible evitar todos los ataques de denegación de servicio, lo más importante es configurar los servicios para que si uno de ellos es inundado, el resto permanezca funcionando mientras se encuentra y soluciona el problema. VER ANEXO 2.

- *Enrutamiento fuente:* los paquetes IP admiten opcionalmente el enrutamiento fuente, con el que la persona que inicia la conexión TCP puede especificar una ruta explícita hacia él. La máquina destino debe usar la inversa de esa ruta como ruta de retorno, tenga o no sentido, lo que significa que un atacante puede hacerse pasar por cualquier máquina en la que el destino confíe (obligando a que la ruta hacia la máquina real pase por la del atacante). Dado que el enrutamiento fuente es raramente usado, la forma más fácil de defenderse contra ésto es deshabilitarlo en el router.

- *Adivinación de contraseñas:* un elevado porcentaje de penetraciones en sistemas se deben al fallo del sistema de contraseñas. El fallo más común es la mala elección de contraseñas por parte de los usuarios. Este se suele llevar a cabo en dos formas básicas. La primera consiste en intentar entrar usando pares cuenta-contraseña conocidos o asumidos (muchos sistemas operativos disponen de cuentas administrativas con contraseñas por defecto, que pese a no ser comentadas en los manuales del sistema, son conocidas por los atacantes). El segundo modo en que los hackers obtienen los contraseñas es mediante el uso de crackers (programas que comparan un diccionario de términos contra ficheros de contraseñas robados). Para protegerse contra estos ataques es vital tanto la educación al usuario sobre cómo elegir su contraseña, como mantener asegurado el fichero de contraseñas, de modo que no pueda ser robado.

- *Icmp redirect y destination unreachable* **:** muchos mensajes ICMP recibidos en un host son específicos a una conexión particular o son disparados por un paquete enviado por ese host. La intención es limitar el alcance de los cambios dictados por ICMP. Desafortunadamente las viejas implementaciones de ICMP no usan esta información extra, y cuando llega uno de esos mensajes, todas las conexiones entre el par de hosts que intervienen en la conexión que propició el mensaje se ven afectadas. Además, con la opción redirect, alguien puede alterar la ruta a un destino para que las conexiones en las que esté interesado pasen por su máquina, de forma que pueda intervenirlas. Los mensajes redirect deben obedecerlos sólo los hosts, no los routers, y sólo cuando estos provengan de un router de una red directamente conectada.

Elementos de seguridad

Una vez conocidos los peligros a los que nos enfrentamos, necesitamos medios para pretegernos contra ellos. En principio, limitando el tráfico entre nuestra red y las externas, a aquel que se considere seguro, o al menos que esté justificado, limitaremos el número de ataques posibles. A continuación veremos que el *filtro de paquetes* y los *servidores proxy* nos permiten ésto. Además, si decidimos permitir que se pueda acceder a nuestras máquinas desde el exterior, habremos de asegurarnos de que los intentos de conexión provienen de quienes dicen provenir. Para ello no podemos fiarnos de los contraseñas convencionales, puesto que un ataque por sniffing daría el contraseña al atacante. Veremos a continuación métodos de *autenticación* que nos solucionan este problema. Por último, si creemos que nuestra red puede ser objeto de un ataque *hijackin*g, necesitamos alguna técnica para impedirlos. En este caso necesitaremos *encriptar* la conexión.

Los métodos a aplicar en una red para ofrecer barreras de seguridad son:

1. Métodos criptográficos: redes privadas virtuales, IPSec, SSH (Secure Shell)
2. Seguridad perimetral: cortafuegos, NAT (Network Address Traslation) e IDS (Intrusión Detection System)
3. Seguridad en el sistema centralizado (envolventos y/o proxies)

Descritos los métodos, los elementos utilizados por dichos métodos son:

Criptografía: mediante el uso de la criptografía se intenta proteger la información a base de codificarla de una forma desconocida a los elementos que no forman parte de la comunicación: *algoritmos de clave privada o simétricos* (DES, TDES, IDEA, RC4 y Skipjack), *algoritmos de clave pública o asimétricos* (RSA). Las aplicaciones básicas de los algoritmos criptográficos son: el *cifrado* es la encriptación de un mensaje con una clave; la *firma digital* para protegerlos contra la falsificación, permitiendo al receptor probar la fuente y la integridad de los mismos, una *función hash segura.*

Autenticación: la autenticación es el proceso seguido por una entidad para probar su identidad ante otra. Distinguimos dos tipos de autenticación: la de un usuario a una máquina durante la secuencia de login inicial, y la de máquina a máquina durante una operación. Los contraseñas tradicionales son demasiado débiles para usarlos sobre una red, y por tanto se usan *contraseñas no reusables.* Estos cambian cada vez que se usan, y por tanto no son sensibles al sniffing. El método de autenticación por dirección IP del host (o bien su nombre DNS) es susceptible de ser atacado mediante spoofing con relativa facilidad, y por tanto se usan técnicas de criptografía, contando con un Centro de Distribución de Claves (KDC) para la distribución y verificación de las mismas. El KDC más conocido es *Kerbero*s.

Filtro de paquetes: los routers permiten realizar un filtrado de paquetes en base a la información contenida en sus cabeceras. Básicamente, la información que se suele examinar es: la dirección IP origen, la dirección IP destino, el tipo de protocolo (TCP, UDP o ICMP), el campo de opciones IP, el puerto origen TCP o UDP, el puerto destino TCP o UDP, el campo de banderas TCP y el tipo de mensaje ICMP. Además de la información contenida en el paquete, se puede tener en cuenta la interfaz de red por la que llega el paquete. El hecho de que los servidores de servicios Internet residan en ciertos números de puertos concretos, permite al router bloquear o permitir la conexión a esos servicios simplemente especificando el número de puerto apropiado en el conjunto de reglas especificado para el filtro de paquetes. El filtro de paquetes es transparente a los usuarios, es decir, no requiere conocimientos ni cooperación por su parte. Sin embargo, las reglas de filtro son difíciles de definir, y una vez definidas, duras de testear. Una relación de puertos y su asignación lo podemos ver en el ANEXO 3.

Servidores proxy o pasarelas: son aplicaciones que nos permiten redirigir el tráfico del nivel de aplicación a través de un cortafuego en el acceso a una red (ejemplo con Socks, Sock-et-s) y/o puentear con otra aplicación, dentro de un sistema centralizado. En este último caso también se llama envolvente.. Al cliente le presentan la ilusión de que está tratando directamente con el servidor real. El servidor real cree que está tratando directamente con un usuario en el host donde está corriendo el proxy. Este sistema no siempre es transparente al usuario, puesto que algunos proxies requieren software cliente especial, o bien el software estándar utilizándolo con procedimientos especiales. Los servicios proxy sólo son efectivos usados en conjunción con un mecanismo que restrinja las comunicaciones directas entre los hosts internos y externos (bien con un dual-homed host, bien con filtro de paquetes).

Estos tres últimos medios son analizados o estudiados desde el punto de vista de pasarelas, como una clasificación más general. Cualquier dispositivo que permite la intercomunicación a niveles superiores al de red se denomina genéricamente pasarela (gateway en inglés). Así nos referimos a pasarelas del nivel de transporte o del de aplicación. Una pasarela del nivel de aplicación es un host que implementa dos (o mas) pilas completas de protocolos y que puede 'trasvasar' datos de una aplicación a su equivalente en la otra pila de protocolos, realizando las conversiones adecuadas. Un ejemplo de esto es el intercambio de mensajes de correo electrónico, por ejemplo entre SMTP (protocolo de correo en Internet) y X.400 (protocolo de correo OSI). Otro ejemplo es el servicio que permite enviar desde un formulario Web un mensaje corto a un teléfono GSM.

Los problemas que se plantean en la interconexión de redes diferentes a nivel de pasarelas de aplicación son en cierto modo parecidos a los que se dan a niveles inferiores, como ocurre al interconectar redes Ethernet y Token Ring. Por ejemplo, si se dispone una pasarela de correo electrónico entre dos redes diferentes, una de las cuales implementa acuse de recibo y la otra no, se plantea el problema de qué hacer en la pasarela cuando un mensaje es enviado de la primera a la segunda; a lo sumo se podría acusar recibo cuando el mensaje se recibe en la pasarela, pero eso no significa que el mensaje haya llegado a su destino; la otra alternativa sería simplemente no enviar ningún acuse de recibo en este caso.

Además de permitir la interconexión de protocolos distintos las pasarelas de aplicación también se emplean para conectar redes con el mismo protocolo o puenteando a servidores controladn su acceso. Existen diversas razones por las que esto puede ser conveniente, como por ejemplo las siguientes:

- **Seguridad (cortafuegos):** si se quiere tener un control riguroso del tráfico entre una determinada red y el exterior se dispone un cortafuegos que aísle dicha red; a menudo los cortafuegos (o cortafuegos) actúan como pasarelas a nivel de transporte o de aplicación.

- **Eficiencia (servidores proxy/cache):** los servidores cache permiten capturar una copia de la información que un usuario se trae del exterior para disponer de ella a<nte la eventualidad de que más tarde otro usuario requiera la misma información. Esto mejora el tiempo de respuesta al usuario ya que la información solicitada se le sirve localmente y reduce el nivel de ocupación de la línea WAN o la conexión a Internet, normalmente de elevado costo. Para realizar su función los servidores caché actúan como pasarelas del nivel de aplicación, normalmente para el protocolo http.

- **NAT (traducción de direcciones):** Como veremos más adelante existen situaciones en las que es necesario manejar un direccionamiento diferente en la red interna y la externa; en estos casos es preciso utilizar un dispositivo NAT (Network Address Translation) que nos permita el intercambio de paquetes entre ambas redes. Algunas aplicaciones requieren una complejidad mayor en el proceso de traducción de direcciones hasta el punto de que esto solo es posible mediante un programa que interprete y modifique la información contenida en el paquete a nivel de aplicación.

- **Envolventes o proxies de aplicación**: son programas sencillos, son pasarelas a nivel de aplicación, que gestionan todo el control de acceso a los servidores dentro de un sistema centralizado. Ej *tcpwrappers*.

Seguridad en red basada en criptografía

Túneles

En ocasiones se quiere intercambiar paquetes entre dos redes que utilizan el mismo protocolo, pero que están unidas por una red que utiliza un protocolo diferente. Por ejemplo supongamos que un banco dispone de una red de área extensa con protocolo SNA que une todas sus oficinas, y dos de éstas disponen además de redes locales TCP/IP. Si se desea que estas dos oficinas intercambien tráfico TCP/IP sin establecer para ello una nueva línea ni instalar routers multiprotocolo en todo el trayecto se puede establecer un túnel entre ambas oficinas. Los dos nodos SNA ubicados en los extremos del túnel serán los encargados de añadir a los paquetes IP la cabecera SNA adecuada para que lleguen al otro extremo. Los paquetes IP viajarán 'encapsulados' a través del túnel en paquetes SNA, de forma que los paquetes IP no sean vistos por la red SNA. Los conceptos de túnel y de encapsulado de paquetes van siempre asociados entre sí.

En Internet hay situaciones en las que los túneles se utilizan de forma habitual; por ejemplo la interconexión de routers multicast a través de routers unicast para constituir la red MBone, o la interconexión de 'islas' IPv6 para constituir el 6bone. En estos casos los túneles se utilizan para enviar paquetes del mismo protocolo (IP) pero de distinto tipo (multicast en unicast) o versión (IPv6 en IPv4). En algunos casos resulta útil encapsular incluso un paquete en otro del mismo tipo y versión; por ejemplo encapsulando un paquete IPv4 en otro podemos forzar la ruta que ha de seguir, cumpliendo así una función similar a la opción 'loose source routing' de la cabecera IP, pero sin las limitaciones que tiene ésta en el número de direcciones. Otro uso del encapsulado podría ser para superar el valor máximo del campo TTL; al utilizar encapsulado la cabecera interior empezaría a descontar su TTL sólo a partir del punto de salida del túnel.

Uno de los usos más habituales de los túneles actualmente es para la creación de redes privadas virtuales o VPNs, como veremos a continuación.

Los túneles no son una solución deseable en sí mismos, ya que a lo largo del túnel los paquetes han de llevar dos cabeceras, lo cual supone un mayor overhead; además los extremos del túnel se convierten en puntos simples de fallo y potenciales cuellos de botella en el rendimiento de la red.

Redes Privadas Virtuales (VPNs)

Entendemos por red privada virtual o VPN (Virtual Private Network) la interconexión de un conjunto de ordenadores haciendo uso de una infraestructura pública, normalmente compartida, para simular una

infraestructura dedicada o privada. Entre las características propias de una red privada virtual se encuentra la posibilidad de utilizar direccionamiento no integrado en la red del proveedor del servicio.

Existen muchas maneras de crear y utilizar VPNs, tanto en IP como en otros protocolos. Aquí solo comentaremos algunas de sus posibilidades más habituales para que sirvan como ejemplo de su aplicación.

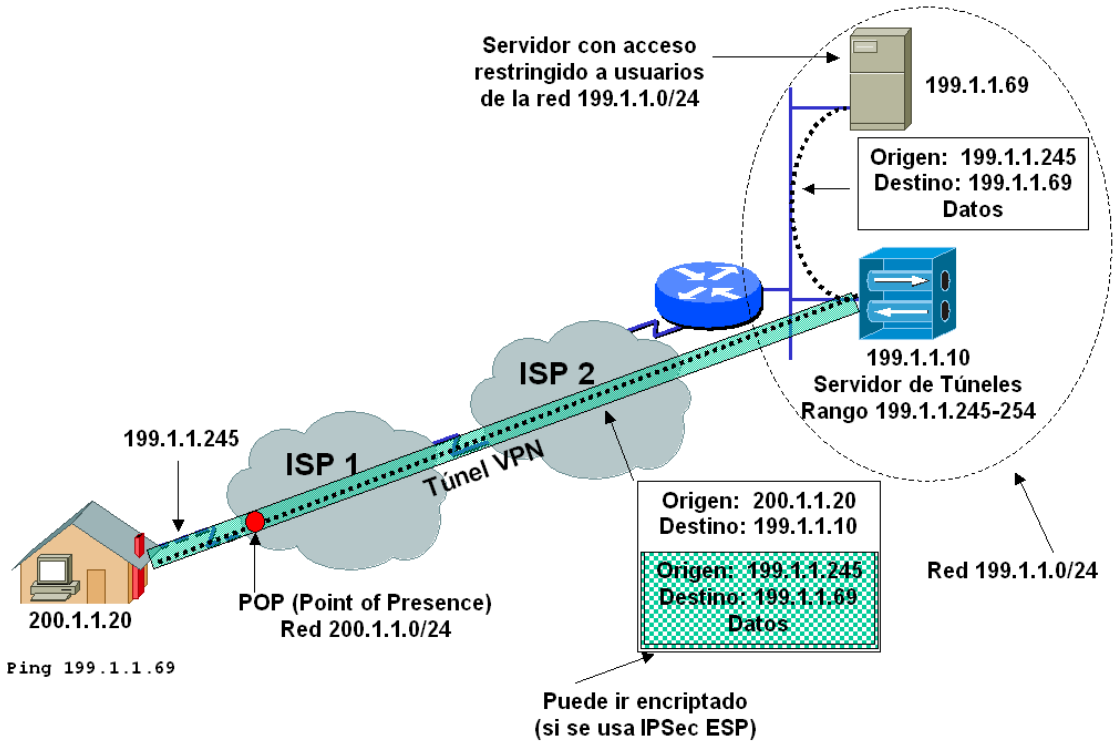


Figura 26: ejemplo de acceso a la Intranet 199.1.1.0/24 a través de VPN

Supongamos que un viajante quiere conectarse remotamente a la red de su empresa para consultar una base de datos y para ello emplea los servicios de un proveedor comercial cualquiera, ya que esto le permite conectarse a su red pagando tarifa metropolitana. Supongamos también que por razones de seguridad o de licencias de software la empresa se ha visto obligada a limitar el acceso a dicha base de datos únicamente a los ordenadores de la red de la empresa, es decir aquellos que tienen direcciones IP de su red (supongamos que se trata de la red 199.1.1.0/24). Como nuestro viajante al conectarse recibe una dirección IP de la red de su proveedor (supongamos que se trata de la red 200.1.1.0/24) no le es posible acceder a dichos servicios.

Este problema puede resolverse creando un túnel VPN (Virtual Private Network) entre el ordenador del viajante y un ordenador de la red de su empresa. El túnel le va a permitir a nuestro usuario remoto acceder a la red de la empresa como si estuviera conectado en la red local (aunque con una velocidad que dependerá obviamente de la conexión física que utilice). El túnel se creará entre su ordenador y un equipo que denominaremos 'servidor de túneles' ubicado en la red local de la empresa, el cual se encargará de encapuslar y desencapsular los paquetes de este usuario. Veamos en detalle como funcionaría el túnel VPN en este caso concreto.

En primer lugar, la empresa deberá tener el servidor de túneles conectado a su red local; dicho servidor puede ser un equipo específicamente diseñado para este fin, un router o un servidor de propósito general Linux o Windows 2000 Server por ejemplo (el software necesario para actuar como servidor de túneles VPN está incluido en Windows 2000 server). Supongamos que el servidor de túneles está conectado a la red local mediante una interfaz Fast Ethernet y que tiene la dirección IP 199.1.1.10. Además de disponer

de una dirección propia el servidor de túneles debe tener asignado un rango de direcciones que utilizará para establecer los túneles. Supongamos que no esperamos más de 10 usuarios simultáneos de este servicio, por lo que reservamos para este fin las últimas 10 direcciones útiles de la red de la empresa, es decir el rango que va de la 199.1.1.245 a la 199.1.1.254 ambas inclusive.

Supongamos ahora que nuestro viajante se conecta por red telefónica a su proveedor habitual y que mediante la asignación dinámica de direcciones de PPP recibe de éste la dirección 200.1.1.20. A continuación se conecta con el servidor de túneles y, previa autenticación mediante usuario/contraseña, el servidor crea un túnel para él y le asigna la dirección 199.1.1.245. A partir de ese momento los datagramas que envíe el viajante llevarán dos cabeceras, una exterior y otra interior. Si el viajante hace desde su PC un ping a una dirección cualquiera de la empresa (por ejemplo la 199.1.1.69) enviará un datagrama cuya cabecera exterior llevará como dirección de origen 200.1.1.20 y de destino 199.1.1.10 (el servidor de túneles) y en la cabecera interior llevará como dirección de origen 199.1.1.245 y como destino 199.1.1.69; este datagrama hará su viaje en dos etapas; en la primera llegará al servidor de túneles (199.1.1.10), que lo despojará de su cabecera exterior y procesará a continuación la cabecera interior, enviando el datagrama al destino deseado en la red de la empresa. El datagrama de respuesta al ping seguirá un proceso simétrico inverso, es decir hará la primera parte de su viaje con una sola cabecera que contendrá 199.1.1.69 como dirección de origen y 199.1.1.245 como destino; ese datagrama será entregado al servidor de túneles, que se encargará entonces de incorporarle la cabecera exterior con dirección de origen 199.1.1.10 y de destino 200.1.1.20; cuando llegue al ordenador del viajante el software cliente VPN le despojará de la cabecera exterior y lo procesará como un datagrama dirigido a él proveniente de 199.1.1.69. Así pues el uso del túnel VPN permite a nuestro viajante acceder a la red de su empresa como si estuviera conectado en la red local detrás del servidor de túneles. El software necesario para crear túneles VPN como el descrito en este ejemplo está integrado en sistemas tales como Windows 98 o Windows 2000 profesional.

Obsérvese que mientras está operativo el túnel todo el tráfico del usuario remoto con cualquier destino de Internet (pertenezca o no a la empresa) se realiza a través del servidor de túneles. Esto significa que el usuario sufrirá las limitaciones asociadas a la capacidad del servidor de túneles y, si accede a destinos ubicados fuera de la red local de su empresa sufrirá también las limitaciones que le imponga la conexión a Internet que tenga su empresa; por tanto lo lógico sería utilizar la conexión a través del túnel únicamente cuando se requiera acceder a servicios de acceso restringido dentro de la red de la empresa.

Además de conectar un ordenador aislado como en el ejemplo anterior es posible establecer un túnel VPN entre routers. Esto nos permite conectar toda una red local a través de un solo equipo, y además no requiere soporte de túneles en el host final. En este caso los routers serán los encargados de realizar la labor de encapsulado/desencapsulado de datagramas. Las VPNs permiten conectar por ejemplo una oficina remota a una oficina principal utilizando Internet, con lo que los costes se pueden reducir considerablemente, especialmente si se trata de conexiones de larga distancia. Además, la reciente aparición de conexiones a Internet de alta velocidad y bajo costo (fundamentalmente ADSL y cable módems) hace especialmente atractivo el uso de túneles VPN sobre este tipo de servicios para constituir enlaces internos en una red sin incurrir en los elevados costos de constituir una infraestructura dedicada mediante enlaces punto a punto o circuitos frame relay o ATM.

IPSEC

Con bastante frecuencia cuando se establecen túneles VPN como hemos visto en los ejemplos anteriores, además de la integración de las direcciones de red se plantea un requerimiento desde el punto de vista de la seguridad, dado que los datagramas viajan por una infraestructura pública en la que la información puede ser capturada y/o modificada por usuarios externos. Por este motivo la constitución de túneles VPN viene acompañada a menudo de un requerimiento de seguridad, constituyendo lo que podríamos denominar redes privadas virtuales seguras.

El problema de una comunicación segura a través de una red se resuelve normalmente a nivel de enlace, a nivel de red o a nivel de aplicación. Cada una de estas tres alternativas tiene sus ventajas e inconvenientes:

- **Nivel de enlace:** La seguridad a nivel de enlace se implementa en los dispositivos que se conectan al medio de transmisión, por ejemplo dos routers que se comunican mediante una línea

punto a punto. En este caso los mecanismos de seguridad se pueden aplicar de forma transparente al protocolo utilizado a nivel de red. Sin embargo en una red grande requiere encriptar y desencriptar la información en cada salto que da el paquete; aun en el caso de utilizar dispositivos que realicen esta tarea por hardware el retardo que esto puede introducir cuando el número de saltos es elevado puede hacer inviable el uso de aplicaciones en tiempo real que requieren cierto nivel de Calidad de Servicio. Además implementar seguridad a nivel de enlace requiere controlar la infraestructura de la red, cosa que no es factible cuando se utilizan los servicios de un operador o un ISP. Un ejemplo de seguridad a nivel de enlace se da en las redes de televisión por cable (CATV) en que la información viaja encriptada (DES) entre el Cable Módem y el CMTS (Cable Modem Termination System); al ser las redes CATV un medio broadcast es necesario el uso de encriptación para evitar la captación de información por parte de usuarios que no son los destinatarios de la misma.

- **Nivel de red:** Esta es la aproximación adoptada por los estándares IPSec. En este caso la seguridad se limita al protocolo IP y otros protocolos sólo podrán aprovecharla si se encapsulan previamente en paquetes IP. La seguridad a nivel de red puede aplicarla el usuario de forma transparente al proveedor del servicio y encaja de forma muy adecuada con el concepto de VPNs pudiendo crear lo que denominamos redes privadas virtuales seguras.

- **Nivel de aplicación:** esta es la aproximación adoptada por ejemplo en correo electrónico por PEM (Privacy Enhanced Mail) o PGP (Pretty Good Privacy), en HTTP por Secure HTTP o SSL (Secure Sockets Layer), o por SNMP versión 3. El principal inconveniente de abordar la seguridad a nivel de aplicación estriba precisamente en la necesidad de incorporar funcionalidades similares en cada uno de los protocolos del nivel de aplicación que deban utilizarlas, replicando así gran cantidad de tareas en diferentes partes de código. La ventaja es que la seguridad se puede desarrollar de forma selectiva, aplicándola únicamente en el intercambio de información confidencial o importante. Además, al aplicarse los mecanismos de encriptado o validación de la información en el nivel más alto posible el nivel de seguridad obtenido es máximo ya que se reduce el riesgo de que la información pueda ser interceptada o modificada por otros usuario o procesos distintos del destinatario de ésta.

Denominamos IPSec al conjunto de estándares que trata todo lo relacionado con el intercambio seguro de información a través de Internet. En realidad IPSec es una arquitectura (descrita en el RFC 2401, de 66 páginas), y un conjunto de protocolos y mecanismos de autenticación y encriptado.

Las principales funcionalidades que incorpora IPSec son las siguientes:

- AH (Autentication Header): en este caso no se pretende garantizar la confidencialidad de la información sino únicamente su veracidad. La cabecera de autenticación asegura al receptor que el datagrama no ha sido alterado durante su viaje por la red, ni en su contenido ni en su cabecera (salvo por la modificación del campo TTL y por consiguiente del checksum, que se han de realizar en cada salto); por consiguiente además de garantizarse el contenido se garantiza la autenticidad del remitente. AH se describe en el RFC 2402, de 22 páginas.

- ESP (Encapsulating Security Payload): garantiza la confidencialidad de la información. La parte de datos del datagrama (incluida la cabecera de transporte) viaja encriptada de forma que sólo el destinatatrio pueda descifrar su contenido. Opcionalmente ESP puede incluir la funcionalidad de AH, es decir garantizar que el contenido no ha podido ser alterado por terceros. ESP se describe n el RFC 2406, de 22 páginas.

- ISAKMP (Internet Security Association and Key Management Protocol): consiste en una serie de mecanismos seguros para realizar, de forma manual o automática, el intercambio de claves necesarias para las labores de encriptado y autenticación realizadas por AH y ESP. ISAKMP se describe en el RFC 2408, de 86 páginas.

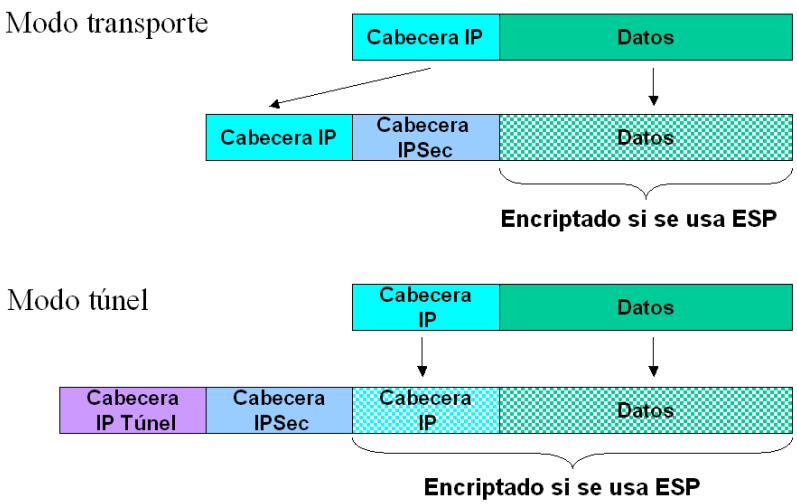


Figura 27: modos de funcionamiento de IPSec: modo transporte y túnel

Podemos distinguir dos modos de funcionamiento de IPSec:

a) Modo transporte: la encriptación se realiza extremo a extremo, es decir del host de origen al host de destino. Para extender en una empresa el uso de IPSec en modo transporte es necesario que todos los hosts tengan una implementación de IPSec.

b) Modo túnel: el encriptado se efectúa únicamente entre los routers de acceso a los hosts implicados. En este caso la información viaja no encriptada en la parte de la red local. El funcionamiento de IPSec en modo túnel permite integrar de forma elegante IPSec en una VPN, ya que el mismo dispositivo que realiza el túnel VPN se encarga de realizar las labores correspondientes al túnel IPSec.

Evidentemente el modo transporte es más fiable puesto que ofrece comunicación segura host a host. Sin embargo el modo túnel tiene la ventaja de que permite incorporar la seguridad sin necesidad de incorporar IPSec en los hosts; aunque la seguridad que se obtiene en este caso no es tan alta la sencillez de implantación es mucho mayor y se consiguen la mayor parte de los beneficios del modo transporte, ya que se protege la parte más expuesta del trayecto, que corresponde precisamente a la infraestructura pública o del operador. En función de las circunstancias que rodeen cada caso se deberá optar por una u otra, pudiendo haber incluso situaciones híbridas en las que en una misma empresa determinado tipo de información baste protegerla con el modo túnel mientras que para algún host concreto, que maneje información de mayor importancia, se deba utilizar el modo transporte.

Aunque en IPSec se prevé la posibilidad de utilizar una amplia diversidad de algoritmos de autenticación y encriptado el único exigido para todas las implementaciones es el DES (Data Encription Standard) que utiliza claves de 56 bits. Desde hace unos años se sabe que DES es relativamente poco seguro, ya que el código puede ser descifrado utilizando fuerza bruta en un tiempo no demasiado grande con un ordenador potente actual, por lo que también se suele utilizar bastante Triple DES, que es mucho más seguro aunque también algo más costoso de calcular. En un futuro próximo se prevé utilizar el algoritmo AES (Advanced Encryption Standard) del cual aún no existen implementaciones comerciales.

Uno de los problemas que plantea la encriptación es el consumo intensivo de CPU. Esto suele ser un problema especialmente en los routers y servidores de túneles que atienden túneles IPSec con la función ESP activada (la función AH no requiere encriptación). En estos casos se pueden concentrar cientos de usuarios y flujos de varios Megabits por segundo en un mismo equipo, que ha de realizar las labores de encriptado/desencriptado para todos ellos, pudiendo verse limitado el rendimiento de las comunicaciones

por la capacidad de la CPU del equipo utilizado. Para evitar este problema muchos servidores de túneles y routers permiten incorporar módulos que realizan los algoritmos de encriptación por hardware para hacerlos con mucha mayor rapidez.

Conexión segura, SSH: Secure Shell

Otra forma de proteger la información transmitida por la red desde la capa de aplicación a través de túneles extremo a extremo a nivel de aplicación, son las aplicaciones SSH, Secure Shell (SSH), por Tatu Ylonen (1.995). La interfaz de usuario es una línea de comandos segura de la capa de aplicación, inicialmente pensado para evitar el paso de contraseñas en las conexiones de telnet. Se considera el sustituto de los protocolos "r" (rsh, rcp, rlogin, ...) SSH es una aplicación especificada en drafts del IETF e implementa la parte de control a través del puerto 22.

Las funciones de SSH son, autenticación de equipos y usuarios (utilizando diferentes métodos de autenticación: RSA, Kerberos, ...), envío de contraseñas cifrados, permite generar canales X11 seguros para exportar pantalla por el uso de conexión cifrada, permite incorporar servidores externos de autenticación como Radius y Tacacs+, realiza el intercambio de claves públicas RSA, además que puede utilizar servidores de gestión de claves tanto con enfoque distribuido (claves en los equipos) como con enfoque centralizado (claves en una CA): certificados X.509

Actualmente, a modo de ejemplo SSH es utilizado como protocolo de conexión desde el exterior del Dpto de Informática de la Universitat de Valencia. Se puede consultar más información y descargar aplicaciones SSH en http://informatica.uv.es/~carlos/adm/comun/conexion_remota.htmlSSH comprende un conjunto de protocolos diferentes:

Transport Layer Protocol (SSH-TRANS) que implementa la parte autenticación del servidor y cliente, confidencialidad, integridad, compresión. Este protocolo funciona de la siguiente manera:

1. El cliente inicia una conexión contra el servidor y se verifica la versión del protocolo SSH (1.0 ó 2.0, incompatibles)
2. Negociación de los servicios requeridos, utilizando Diffie-Hellman para intercambio de la clave aleatoria. Si no se pueden soportar todos los servicios, se aborta
3. Intercambio de las claves de sesión que permiten cifrar y autenticar la sesión. Este intercambio de claves pueden ser renegociadas posteriormente
4. Finalmente intercambio de datos cifrados utilizando algoritmo simétrico negociado para cifrado, algoritmo hash negociado para la integridad y algoritmo de compresión negociado.

User Authentication Protocol (SSH-USERAUTH) es un protocolo por encima de SSH-TRANS y funciona de la siguiente manera

1. El servidor informa de los tipos de autenticación y crea una lista de métodos propuestos
2. El cliente escoge el método de autenticación en cualquier orden
3. El servidor acepta el método de autenticación y tiene la libertad de no aceptarlo

Los métodos de autenticación utilizados normalmente son contraseña tradicional, clave pública con RSA, y confianza basada en equipos.

Connection Protocol (SSH-CONN) es un protocolo por encima de SSH-TRANS y se encarga de establecer canales (multiplexados en una sesión) y a cada canal se les asigna un identificador único. En estos canales se aplica control de flujo por ventana y funciona de la siguiente manera

1. El iniciador abre el canal y especifica los parámetros de conexión (id, ventana, etc)
2. Intercambian datos en el canal
3. Un participante cierra el canal

Los tipos de canales son interactive Session como por ejemplo en consola remota o ejecución remota, para exportar consolas con X11 Redirect de forma que se redirigen todas las conexiones X11 por el canal y el canal tipo TCP/IP Redirect, que permite capturar un par (puerto TCP, IP) y se envía en modo túnel.

Seguridad en red perimetral

Cortafuegos (firewalls)

Especialmente con el crecimiento comercial de la Internet muchas empresas se enfrentan ante la necesidad de conectar sus redes al exterior; sin embargo esto plantea varios problemas de seguridad por diversos motivos, por ejemplo:

- Los ordenadores de la red local contienen información de carácter confidencial cuya salvaguardia resulta vital para la empresa.
- Del exterior pueden llegar virus u otro tipo de programas que perjudicarían seriamente los ordenadores de la empresa.
- Los empleados pueden utilizar la conexión a Internet para salir a lugares no autorizados (por ejemplo servidores de información no relacionados con su actividad en la empresa).

Para resolver los problemas de acceso seguro hacia/desde el exterior se ha creado el concepto de *cortafuego* o *firewall*, que consiste en un dispositivo formado por uno o varios equipos que se sitúan entre la red de la empresa y la red exterior (normalmente la Internet); el cortafuego analiza todos los paquetes que transitan entre ambas redes y filtra los que no deben ser reenviados, de acuerdo con un criterio establecido de antemano.

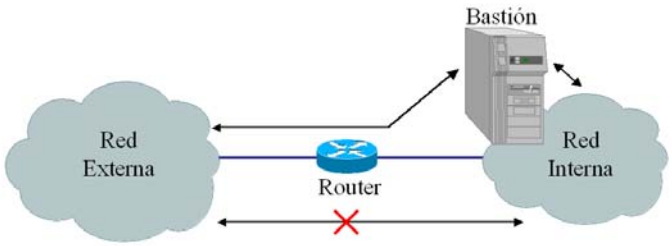


Figura 28: cortafuegos basado en router

En su versión más simple el cortafuego consiste únicamente en un router en el que se han configurado diversos filtros, por ejemplo impidiendo o limitando el acceso a determinadas direcciones de red, o el tráfico de ciertas aplicaciones o una combinación de ambos criterios, como vemos en la figura anterior. Dado que los usuarios siguen teniendo conexión directa a nivel de red con el exterior esta solución no es muy fiable; además las posibilidades de definir filtros en los routers son limitadas y el rendimiento se resiente de forma considerable si al router se le carga con una tarea de filtrado compleja.

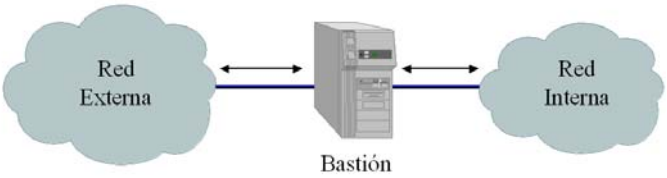


Figura 29: cortafuegos basadon en host bastión (*dual homed gateway*)

El siguiente nivel de cortafuego está formado por un host (*dual homed gateway*) que conecta por una parte a la Internet y por otra a la red corporativa, actuando él como router, como vemos en la figura anterior. El host implementa un servidor Web proxy que actúa como pasarela de aplicación para los servicios que se quieren permitir, limitado por las restricciones, filtros o reglas que se han especificado. El ordenador que actúa como barrera entre la red interna y la Internet se denomina host bastión ('bastion host') en terminología de cortafuegos. Esta solución ofrece una seguridad mayor que la anterior ya que el servidor proxy , al ser un host, puede procesar filtros mas complejos que un router; pero si un usuario malintencionado (hacker o cracker) consiguiera instalar un programa 'espía' (sniffer) en el host bastión podría capturar tráfico de la red interna de la empresa, ya que está directamente conectado a la LAN.

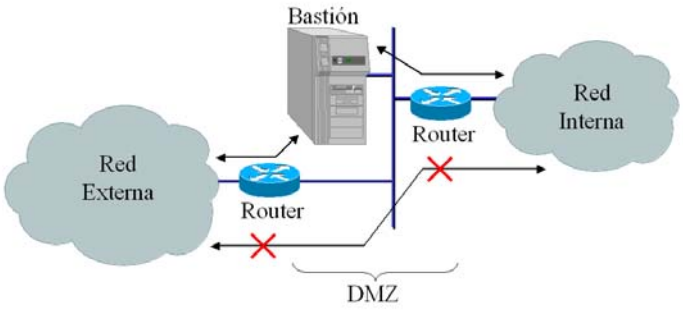


Figura 30: cortafuegos con zona desmiliatarizada, subred apantallada (*screened subnet*)

En un nivel mayor de seguridad el cortafuego se implementa mediante un host y dos routers, conectados entre sí por una pequeña red local; uno de los routers conecta a su vez con la red local de la empresa y el otro con la Internet; el host implementa una pasarela multiaplicación (servidor proxy) como antes, pero al no estar el directamente conectado a la red local de la empresa incluso en el caso de que un hacker consiguiera instalar en él un sniffer no podría capturar tráfico confidencial, pues solo podrá ver los paquetes dirigidos a él. Esta configuración se llama subred apantallada o *screened subnet*. En este modelo la red que une los routers con el host se denomina *zona desmilitarizada* o zona DMZ (Demilitarized

Zone, algo así como una zona neutra de seguridad). En ocasiones se utilizan otras variantes de arquitectura de cortafuego aún mas complejas.

En seguridad de redes, además de una buena arquitectura de cortafuego es importante comprobar que no hay 'agujeros' que permitan el acceso no controlado por otras vías. Por ejemplo, en una empresa con una red altamente protegida podría una oficina regional haber establecido una conexión local con otra institución sin conocimiento de los responsables de seguridad, o disponer de un sistema no seguro de autenticación de usuarios en el acceso por red conmutada para teletrabajo. En suma, que además de un buen cortafuego hay que tener una política de seguridad rigurosa si se quiere que las medidas sean efectivas, de lo contrario habremos puesto una puerta blindada, pero nos pueden entrar ladrones en casa por el 'trastero' con gran facilidad.

Traducción de direcciones (NAT)

Hace unos años cuando una organización deseaba conectar su red de forma permanente a la Internet lo normal era que solicitara al NIC una red adecuada a sus necesidades (clase B o C normalmente) y asignara números IP de dicha red a todas sus máquinas. De esta forma todos los ordenadores tenían acceso directo a Internet con todas las funcionalidades. Pero debido al crecimiento exponencial de Internet cada vez resulta más difícil obtener números IP del NIC correspondiente; por otro lado, en muchas ocasiones no se necesita, o incluso no se desea, disponer de un acceso directo a Internet con completa funcionalidad, por razones de seguridad fundamentalmente. Estos dos motivos, la seguridad y la dificultad para conseguir direcciones públicas, han impulsado a las organizaciones a hacer un mayor uso de las redes privadas según los rangos especificados en el RFC 1918 (10.0.0.0, 172.16.0.0 a 172.31.0.0, y 192.168.0.0 a 192.168.255.0). Estas redes privadas no pueden en principio intercambiar datagramas directamente con el exterior, por lo que han de utilizar un equipo intermedio que se ocupe de realizar la traducción de direcciones o NAT (Network Address Translation).

El NAT puede realizarse en un host (por ejemplo en Linux la función NAT se denomina 'IP Masquerade') aunque también se implementa en muchos routers. Dado que generalmente la función de NAT se realiza en la frontera entre una red local y el exterior la opción del router es bastante popular.

Normalmente NAT solo traduce paquetes IP que en el campo protocolo tienen el valor TCP, UDP o ICMP. El resto de protocolos (fundamentalmente protocolos de routing) no se traducen pues no tendría sentido intercambiar información de routing a través de un NAT.

Los tipos de NAT, inicialmente los NAT solo permitían iniciar conexiones desde 'dentro', es decir desde la red privada hacia el exterior. Este modo de funcionamiento, que se consideraba una medida de seguridad, se denomina **NAT tradicional**. Mas recientemente se ha extendido el uso de NATs que también permiten el inicio de la sesión desde fuera, por lo que a este tipo de NAT se le denomina **NAT bidireccional**.

La traducción se puede hacer de dos maneras: modificando únicamente la dirección IP, a lo que denominamos **NAT básico**, o cambiando también el número de puerto TCP o UDP, que llamaremos **NAPT (Network Address Port Translation)**.

Por último, si la traducción de la dirección pública y la privada se realiza de acuerdo con una tabla de equivalencia que se carga en la configuración del dispositivo NAT y que no se modifica dinámicamente decimos que se trata de **NAT Estático**. En cambio si dicha tabla de equivalencia es gestionada dinámicamente por el dispositivo NAT de forma que las direcciones y/o números de puerto se puedan reutilizar decimos que tenemos un **NAT dinámico**.

Combinando el NAT Básico o el NAPT con las modalidades estática y dinámica obtenemos cuatro combinaciones de NAT que pasamos a describir a continuación:

- NAT básico estático. La tabla de equivalencia IP privada – IP pública está incluida en la configuración del dispositivo NAT y solo se modifica por decisión del administrador de la red. La correspondencia es biunívoca, es decir a cada dirección privada le corresponde una pública y

viceversa. Los números de puerto no se modifican. Con este tipo de NAT es preciso disponer de un número de direcciones públicas igual al de direcciones privadas. Este NAT tiene interés fundamentalmente en situaciones en las que se desea máxima sencillez y no se necesita reducir el número de direcciones públicas, por ejemplo se quiere conectar a Internet una red clase C privada y se dispone para ello de una clase C pública.

- NAT básico dinámico. La tabla de equivalencias de IP privada a IP pública se construye de forma dinámica, a medida que lo requieren los hosts. Las entradas caducan al terminar la conexión (TCP) o pasado un tiempo de inactividad (UDP), lo cual permite su reutilización. Los números de puerto no se modifican. El número de direcciones públicas puede ser inferior al de direcciones privadas, pero ha de ser suficiente para el número de ordenadores que se quieren conectar simultáneamente al exterior. Permite un ahorro de direcciones frente al NAT estático, pero al no modificar el número de puerto es mas fácil de implementar y tiene menos restricciones que el NAPT.

- NAPT (Network Address Port Translation) estático. En este caso la tabla de equivalencia se carga de forma estática en la configuración del equipo, pero las entradas incluyen no solo la dirección IP sino también el número de puerto TCP o UDP. El NAPT estático permite virtualizar servidores: por ejemplo es posible asignar el puerto 21 (servidor FTP) de una dirección pública del NAT al puerto 21 de un host en la red privada, y el puerto 80 (servidor Web) de la misma dirección a otro host de la red privada.

- NAPT (Network Address Port Translation) dinámico. En este caso la tabla de equivalencias se construye dinámicamente a medida que los hosts lo requieren, como en el NAT básico dinámico. Sin embargo, a diferencia de aquel las entradas de la tabla incluyen no solo la dirección IP, sino también el número de puerto. Las entradas caducan al terminar la conexión (TCP) o pasado un tiempo de inactividad (UDP), lo cual permite su reutilización. En este caso es posible aprovechar una misma dirección IP pública para conectar al exterior diversos ordenadores simultáneamente, ya que se aprovecha el número de puerto UDP o TCP para multiplexar conexiones de hosts diferentes. Este NAT permite un aprovechamiento máximo de las direcciones públicas, pero es el que tiene mayor complejidad de implementación.

Las cuatro modalidades antes descritas pueden coexistir en una misma red, por ejemplo utilizando NAT o NAPT estático para los servidores que deban ser accesibles desde el exterior y NAT o NAPT dinámico para el resto de los hosts.

Las modificaciones que NAT introduce en el paquete IP son las siguientes:

- Cabecera IP: Al modificar las direcciones de origen y/o destino el valor del campo checksum en la cabecera del datagrama cambia y por tanto ha de recalcularse.

- Cabecera de transporte (TCP/UDP): El campo checksum en la cabecera de transporte (TCP o UDP) ha de recalcularse, ya que la pseudocabecera incluye las direcciones IP de origen y destino. Además en el caso de NAPT se ha de modificar el valor del puerto de origen o destino.

- Mensajes ICMP: Los mensajes ICMP suelen incluir embebida la cabecera del datagrama IP y el principio de la cabecera TCP/UDP que originó el mensaje ICMP. El dispositivo NAT ha de localizar en el mensaje ICMP la dirección IP y modificarla; además ha de modificar consecuentemente el checksum de la cabecera IP embebida. En el caso de hacer NAPT se ha de modificar también el número de puerto TCP/UDP que aparece en la cabecera embebida.

- Mensajes SNMP. Los mensajes de gestión, que notifican cambios en la situación de los diferentes dispositivos, suelen llevar en su parte de datos direcciones IP que el NAT debe localizar y modificar.

Limitaciones de los NAT, a medida que aumenta el nivel de sofisticación aumenta como es lógico la complejidad del NAT. En el caso del NAT o NAPT dinámico el router ha de conservar una información de estado, ya que ha de mantener control de las conexiones existentes. Esta información de estado hace mas difícil establecer un router de backup ya que esta información de estado ha de trasladarse a dicho

router en caso de caída del principal. Además el router ha de tomar decisiones respecto a cuando termina una conexión para liberar la correspondiente entrada en su tabla (la de dirección IP en el caso de NAT o la de dirección IP + puerto TCP/UDP en el caso de NAPT). En TCP es bastante fácil detectar el cierre de la conexión a través de los segmentos con el bit FIN, pero en UDP no existe un procedimiento explícito de desconexión, por lo que en estos casos normalmente se fija un tiempo de inactividad a partir del cual una conexión se considera inexistente, o bien se espera a tener que liberar recursos y entonces se cierra la conexión que lleva más tiempo inactiva.

Incluso en el caso más sencillo del NAT básico estático se plantean problemas de difícil solución que hacen que determinadas aplicaciones no funcionen a través de estos dispositivos. Algunos ejemplos de situaciones en las que el NAT tiene dificultades o no funciona son las siguientes:

- Protocolo FTP. En el momento de establecer una conexión FTP los hosts intercambian una serie de mensajes que contienen entre otra información sus direcciones IP. Lógicamente estas direcciones han de modificarse. El problema es que esas direcciones se encuentran codificadas como caracteres ASCII, no en binario como ocurre normalmente. Así, si la dirección de origen es por ejemplo "200.200.200.1" (13 caracteres) y la de destino es "192.168.1.1" (11 caracteres) el proceso de traducción puede optar por rellenar el campo con ceros ("192.168.001.1") para mantener constante el número de octetos. Pero cuando la traducción se realiza en sentido inverso es inevitable aumentar en dos octetos la longitud del segmento TCP correspondiente; como consecuencia de ello el NAT a partir de ese momento ha de incrementar en dos octetos los contadores de bytes que aparecen en los cmapos número de secuencia y número de ACK para esa conexión TCP; esto supone que el NAT ha de mantener una cierta *información de estado* para esa conexión mientras exista. Incluso puede darse el caso de que el aumento en dos octetos del segmento provoque la fragmentación del datagrama correspondiente.

- En general cualquier protocolo del nivel de aplicación que incluya en la parte de datos información sobre direcciones IP o números de puerto TCP/UDP supone un reto para un NAT, ya que la detección y modificación de dichas direcciones es difícil, compleja y requiere que el NAT analice información perteneciente al nivel de aplicación. Algunos ejemplos de esto son el protocolo de la ITU-T H.323 utilizado en videoconferencia y en telefonía por Internet (voz sobre IP); también se encuentran en este caso los juegos interactivos de Internet, las aplicaciones tipo Napster, etc. Normalmente el funcionamiento de estas aplicaciones a través de un NAT sólo se consigue cuando se dispone de una pasarela a nivel de aplicación, programa que realiza las modificaciones necesarias en los datos para que la aplicación pueda utilizarse a través de un NAT. Las pasarelas a nivel de aplicación sólo están disponibles para algunas aplicaciones estándar.

- La comunicación de aplicaciones NetBIOS sobre TCP/IP tiene problemas parecidos a las aplicaciones del apartado anterior: la información sobre direcciones IP aparece de forma no consistente y con desplazamientos variables en la información intercambiada por los hosts, lo cual impide que los NAT las modifiquen; como consecuencia no es posible utilizar TCP para transportar NetBIOS cuando se atraviesa un NAT.

- IPSec solo puede utilizarse de forma limitada a través de un NAT. Esto se debe a que IPSec incorpora una cabecera de autenticación (AH, Authentification Header) que permite al receptor detectar si el paquete IP ha sido modificado en ruta. Evidentemente la cabecera AH no incluye el campo TTL ni el checksum, ya que se sabe que estos campos cambian de valor durante el camino de un datagrama, pero si incluyen las direcciones IP de origen y destino. Como estas direcciones se modifican en el NAT el receptor cuando comprueba la cabecera AH detecta que el datagrama ha sido alterado y lo descarta. En el caso de utilizar IPSec en modo túnel y realizar el túnel en el mismo dispositivo que realiza el NAT las limitaciones se reducen.

A pesar de todas las limitaciones reseñadas NAT resulta un mecanismo muy útil en un amplio abanico de situaciones, por lo que es seguro que se seguirá utilizando NAT en Internet durante bastantes años, al menos hasta que se generalice el uso de IPv6. Con ese nuevo protocolo y su abundante espacio de direcciones parece difícil concebir un escenario en el que sea aconsejable utilizar NAT.

Los aspectos más relevantes de NAT se discuten en los RFC 1631 y 2663.

Detección de intrusos o IDS (Intrusión Detecction System)

Las vulnerabilidades de los diferentes sistemas dentro de una red son los caminos para realizar los ataques. En muchas ocasiones, el atacante enmascara el ataque en tráfico permitido por el cortafuegos y por tanto para delatarlo se necesita un IDS, y ambos son complementarios.

Las características deseables para un IDS son que esté continuamente en ejecución y debe poderse analizar él mismo y detectar si ha sido modificado por un atacante, utilizar los mínimos recursos posibles y adaptarse fácilmente a los cambios de sistemas y usuarios, por lo que en ocasiones poseen inteligencia para adaptarse (aprender por su experiencia) y configurarse.

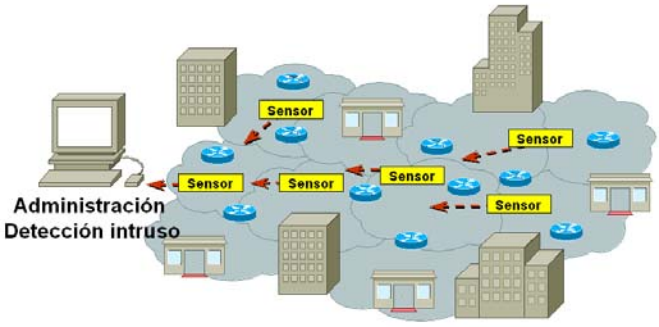


Figura 31: ejemplo de detección de intrusos en red

Una clasificación según su localización es:

- NIDS (Network Intrusion Detection System): detecta los paquetes armados maliciosamente y diseñados para no ser detectados por los cortafuegos. Consta de un sensor situado en un segmento de la red y una consola. La ventaja que tiene este tipo de cortafuegos es que no se requiere instalar software adicional en ningún servidor. Inconveniente: es local al segmento, si la información cifrada no puede procesarla
- HIDS (Host Intrusion Detection System): analiza el tráfico sobre un servidor. Las ventajas que tiene es que registra comandos utilizados, es más fiable, mayor probabilidad de acierto que NIDS.

Clasificación según modelos de detección:

- Detección de mal uso: verifica sobre tipos ilegales de tráfico, secuencias que previamente se sabe se utilizan para realizar ataques (conocidas como exploits)

- Detección de uso anómalo: verifica diferencias estadísticas del comportamiento normal de una red, según franjas horarias, según la utilización de puertos (evitaría el rastreo de puertos)

Clasificación según naturaleza

- Pasivos: registran violación y genearan una alerta

- Reactivos: responden ante la situación, anulando sesión, rechazando conexión por el cortafuegos, etc

Honey Pot (jarrones de miel)

En ocasiones es interesante aprender de los propios atacantes. Para ello, en las redes se ubican servidores puestos adrede para que los intrusos los saboteen y son monitorizados por sistemas que actúan como puentes a los servidores, registrando de forma transparente los paquetes que acceden a dichos servidores.

De esta forma, detectado un ataque (por modificación de la estructura de archivos), se recompone la traza del atacante (secuencia de paquetes registrados en el monitor puente) y se pasa a un análisis forense. Este análisis forense concluye, en caso de detectar un nuevo ataque, en una nueva regla de detección.

Seguridad en red basada en sistema centralizado

Encapsuladores (proxies) y pasarelas

Los envolventes son un invento reciente de la seguridad en UNIX. Aunque nacieron por la necesidad de modificar el funcionamiento de los sistemas operativos sin acceso a su código fuente, su uso como herramienta de seguridad se ha extendido por numerosas razones:

- La lógica de seguridad se encapsula dentro de un programa sencillo y fácil de verificar.
- El programa envuelto permanece separado del envolvente, lo cual permite actualizar el envolvente o el programa envuelto de forma individual.
- Como los envolventes llaman al programa envuelto mediante la llamada al sistema *exec()*, un mismo envolvente puede utilizarse para controlar el acceso a un enorme grupo de programas envueltos.

Envolvente de correo.

El correo electrónico es una fuente de problemas de seguridad. Para solucionarlo, se ha diseñado un envolvente de sendmail que puede obtenerse en la dirección de FTP ftp.tis.com como usuario anonymous. El envolvente de sendmail consiste en dos programas, smap y smapd.

El programa smap acepta mensajes de la red y los escribe en un directorio del disco. Aunque el programa se ejecuta con los permisos de *root*, lo realiza en un sistema de archivos restringido con *chroot*, desde el cual no se puede acceder al resto del disco. El programa es llamado por el gestor de demonios *inetd* y termina al recoger el correo.

El programa smapd revisa periódicamente el directorio donde escribe smap y envía los mensajes recibidos mediante el *sendmail* o algún otro programa.

Envolvente de acceso.

Un superservidor es un demonio o programa en ejecución en segundo plano en UNIX, que está pendiente de las peticiones externas a servicios determinados escuchando en los puertos de servicio, de forma que una vez establecida la conexión, lanza el proceso o demonio que atiende dicha petición. Así de esta forma se reduce el número de procesos en ejecución en el sistema.

Los envolventes participan en la parte lógica de control de acceso, que forma parte de la decisión al lanzar o no el servicio solicitado.

En *Unix*, el superservidor más utilizado es "*inetd*" cuya configuración se encuentra en el fichero "*inetd.conf*". Pero "*inetd*" no realiza ningún control de seguridad en el momento de lanzar un cierto servicio y por tanto, es necesario encapsular los servicios lanzados a través un programa, que controle el acceso a los servicios. De esta forma, el programa encapsulador puede englobar de forma sencilla toda la lógica de seguridad.

Uno de los encapsuladores más utilizados y de propósito general es *tcpwrappers (*y su demonio es conocido como *tcpd)*. El programa es de libre distribución, desarrollado por Wietse Venema y forma parte de la distribución de la mayoría de sistema operativos *Linux*. El programa *tcpwrappers* permite un alto nivel de control sobre las conexiones TCP y permite enviar un mensaje al cliente que se conecta, pudiendo hacerle advertencias legales y avisos (*banners*), ejecutar consulta doble de la dirección IP, comparar el nombre del cliente y el servicio solicitado con una lista de control de acceso, ejecutar órdenes o comandos, registrar la información mediante "*syslog*", emplear "*ident*" (RFC 1413) para averiguar el nombre del usuario, ...

El envolvente *tcpwrapper* posee dos formas de configuración, o bien modificando los ficheros de configuración, */etc/hosts.allow* y */etc/hosts.deny,* o bien a través de sólo */etc/hosts.allow* permitiendo un mayor número de opciones en su configuración, siendo este último el modo utilizado finalmente.

Al utilizarse conjuntamente "*inetd*" y *tcpwrappers* se ha desarrollado un nuevo programa superservidor extendido llamado "x*inetd*" (*Extended Internet Service Daemon*). Este programa, integra el encapsulador *tcpwrappers* a través de las librerías "*libwrap*". Adicionalmente a las funcionalidades de *tcpwrappers* incluye además accesos por franja horaria, evita el ataque DOS (*Deny of Service*), ... Estas librerías hacen uso de la configuración del fichero */etc/hosts.allow,* cuya estructura son entradas con la siguiente sintaxis está es:

lista_de_Demonios: lista_de_clientes : option : option : option: ...

donde:

-lista_de_Demonios: contiene servicios especificados en el fichero */etc/services* y cuyos programas generalmente se localizan en */usr/sbin*

-lista_de_clientes: patrones por nombres ".uv.es"(para todo el dominio de la *Universitat*), por direcciones "147.156." (para especificar 147.156.x.x), utiliza los comodines ALL (siempre cumple), LOCAL (perteneciente al dominio del servidor), KNOWN/UNKNOWN (usuarios cuyo nombre se conoce/desconoce), PARANOID (clientes cuyo nombre no encaja con su dirección IP). Estos comodines pueden modificarse con el operador *EXCEPT*.

-options : puede ser {*allow*|*deny*}, ejecución de órdenes ({*spawn*|*twist*}[1] *comando_de_shell*), emitir *banner*[2]*s*, ...

Dichos ficheros son evaluados y en el momento de la primera coincidencia de servicio y cliente, se ejecutan las opciones anexas. Si entre dichas opciones no aparece la opción {*allow*|*deny*}, se sobreentiende que se está permitido para la ejecución del servicio solicitado. Para ver con detalle las diferentes opciones, consultar el manual "*hosts.allow*" y "*hosts_options*".

El envolvente de acceso más conocido es tcpwrapper. El programa puede obtenerse usando FTP anónimo desde la dirección ftp.porcupine.org/pub/security/tcp_wrappers_XXX.tar.gz, donde XXX es la versión. El envolvente forma parte de la distribución de la mayoría de sistema operativos Linux.

El programa tcpwrapper permite un alto nivel de control sobre las conexiones TCP. El programa se inicia por el demonio *inetd* y entonces lleva a cabo una o más de las siguientes acciones:

- Envía un mensaje al cliente que se conecta, pudiendo hacerle advertencias legales y avisos.
- Ejecuta una consulta doble de la dirección IP. Con ello se asegura de que la dirección y el nombre del ordenador remoto coinciden. En caso de no ser así se almacena la información en un fichero de registro y puede opcionalmente cortar la conexión.

---

[1] Con "*spawn"* se lanza un proceso hijo que atiende la línea de comandos pasado, donde los *stdin, stdout, stderr* están conectados a /dev/null, para evitar conflicto con la comunicación con el cliente. Previo a la ejecución del hijo, se evalúan los parámetros y expansiones que contiene. Con "*twist"* los *stdin, stdout, stderr* están conectados al cliente remoto, con lo cual toma control del proceso que llevaba la evaluación de control de acceso. Este comando por tanto ha de ser el último en ejecutarse.

[2] El comando *banners* tiene como argumento el directorio donde se encuentran los ficheros que permite visualizar en el cliente un mensaje. El nombre de los ficheros coincide con el nombre del servicio solicitado

- Compara el nombre del cliente y el servicio solicitado con una lista de control de acceso, comprobando si el cliente o el cliente y el servicio particular están autorizados o denegados.

El envolvente tcpwrapper posee dos ficheros de configuración, */etc/hosts.allow* y */etc/hosts.deny*. Cuando sucede una conexión se realiza lo siguiente:

1. Se examina el archivo /etc/hosts.allow para comprobar si el par (ordenador, servicio) están permitidos.

2. Si no se encontró, se examina el archivo /etc/hosts.deny para comprobar si el par (ordenador, servicio) debe ser rechazado.

3. Si no se encontró el par (ordenador, servicio) en los ficheros anteriores, la conexión se permite.

Tcpwrapper posee un lenguaje de configuración muy simple y flexible, pudiendo configurar el envolvente como se desee.

SOCKS: Sock-et-S

Los Socks son proxies que actúan como representante local (apoderado) de un servidor remoto para atravesar el cortafuegos. Permite controlar de forma rigurosa el acceso a Internet de las diferentes conexiones realizadas entre elementos en ambas partes de un cortafuegos.

Para configurar SOCKS, hay que compilar las aplicaciones para modificar las llamadas a los sockets. Los ficheros de configuración en el servidor /etc/sockd.conf y en el cliente /etc/socks.conf.

Más información en http://www.socks nec.com
uncionamiento de Socks es el siguiente, cuando un cliente SOCKS conecta a un servidor de Internet, el servidor SOCKS intercepta la conexión y la filtra en base a número de puerto, origen, destino, clave de usuario, ... En el caso, que sea admisible, puentea la conexión directamente con el cliente.

ANEXO 1: SPOOFING Y HIJACKING, SUPLANTACIÓN DE IPS Y ROBO DE SESIONES[3]

En redes de computadoras podemos distinguir dos tipos de ataques : pasivos y activos. En el primero, el atacante se dedica únicamente a escuchar de la red. Ésta es quizá la forma más habitual de penetración: un intruso que haya logrado acceso a un host de nuestra red, podría lanzar un programa sniffer y dedicarse a escuchar los 100 primeros bytes de todas las conexiones dirigidas a los puertos telnet, ftp, pop3, login y demás protocolos que pasen por la red nuestros contraseñas sin encriptar.

Para evitar esto han aparecido mejoras, como por ejemplo los contraseñas de un solo uso (one-time contraseñas), Kerberos, ... en donde el usuario tiene una lista de claves de acceso y usa una cada vez para registrarse en el sistema. De esta forma, si alguien logra ver nuestra clave de acceso circulando por la red no podrá utilizarla, ya que habrá caducado.

Con esta solución hacemos más robusta la autenticación por medio de contraseñas. Pero ¿qué ocurre si por debajo de ésta se realiza una autenticación por dirección IP ?, es decir, ¿ qué ocurre si dependiendo de la dirección IP de nuestro interlocutor actuamos de una forma u otra, pidiendo contraseña o no?. Esto puede llegar a ser un problema grave y lo veremos en el siguiente texto.

El spoofing es una técnica que consiste en engañar a un computador haciéndole creer que esta dialogando con un host cualquiera, cuando en realidad lo está haciendo con nosotros. Esto puede que no parezca una amenaza, ya que normalmente la autenticación se hace mediante contraseña, por lo que aunque en nuestra conexión con un servidor suplantemos la identidad de otro host, tendremos que conocer la clave de acceso para hacer uso de un servicio dado.

El problema aparece cuando algunos de nuestros hosts establecen una relación de confianza : entonces, si alguien es capaz de engañarles diciendo que es uno del grupo, esto podría causar estragos en la seguridad de nuestra red. Pero, ¿cómo se plasman esas relaciones de confianza en la realidad ?.

En UNIX existen un par de ficheros que permiten, por medio del comando rlogin, hacer conexiones sin necesidad de entrar ningún contraseña. Estos ficheros son los siguientes:

- $HOME/.rhosts
- /etc/host.equiv

El fichero rhosts se encuentra en el directorio raíz de cada usuario y permite añadir pares 'máquina-nombre de usuario' que indican máquinas y usuarios en los que se puede confiar. Por ejemplo, supongamos que tenemos dos cuentas llamadas satomi en dos computadoras distintas, vega.domain.cxm y antares.domain.cxm.

Nos gustaría conectarnos de una a otra sin necesidad de pasar contraseñas por la red, por razones de rapidez y de seguridad, frustrando así a los sniffers. La forma de hacerlo sería añadir en /home/satomi/.rhosts de vega.domain.cxm la siguiente línea:

antares.domain.cxm satomi

y en antares.domain.cxm :

vega.domain.cxm satomi

Si además quisiésemos que el usuario adela de altair.algo.cxm pudiese acceder también a nuestra cuenta de antares.algo.cxm sin entrar contraseña, deberíamos añadir en el rhosts de antares lo siguiente:

altair.domain.cxm adela

[3] Esta parte ha sido elaborado como trabajo de la asignatura de Redes por el alumno Emilio Mira.

El fichero /etc/host.equiv tiene un uso más general y tan solo puede ser con½gurado por el administrador del equipo. En él aparecen los nombres de las máquinas de las que aceptaremos conexiones sin pedir contraseñas, pero tan solo cuando el nombre de la cuenta del host remoto coincida con el nombre de la cuenta en el nuestro.

En nuestro ejemplo de antes, si el fichero /etc/host.equiv de antares contuviese :

altair.domain.cxm
vega.domain.cxm

cualquier usuario de altair y vega podría hacer rlogin a antares sobre una cuenta del mismo nombre que la remota sin necesidad de contraseña.

Cuando el comando rlogin envía una solicitud de conexión, el destino primero mira si el host que laoriginó se encuentra en /etc/host.equiv. De ser así iniciaría una sesión sin pedir contraseña sobre una cuenta con su mismo nombre de usuario. En caso contrario, buscaría en el .rhosts del usuario donde queremos iniciar una sesión el nombre de la máquina origen y el usuario que originó la conexión. Si existe, no pediría contraseña. Solo en caso contrario necesitaríamos entrar la clave de acceso.

Es ahora cuando nos damos cuenta de lo siguiente : ¾ Dónde recae la autenticación de este servicio ?. En la dirección IP del origen. En general, los llamados 'servicios R' de U nix , el sistema X Window y los RPC, basan su autenticación en la dirección IP del host origen, y es ahí donde recae el problema. Veamos ahora las posibles variantes del spoofing.

IP spoofing

El IP spoofing es sin duda el más habitual. En 1985, Robert Morris, que por entonces trabajaba en AT&T, ya dio cuenta de él en su artículo _A weakness in the 4.2BSD UNIX TCP/IP software_. Diez años más tarde, Kevin Mitnick utilizó esta técnica para asaltar sistemas informáticos del Gobierno de EEUU e importantes empresas y universidades, convirtiéndose en el cracker más buscado de la historia. La base del IP spoo½ng es la siguiente : un host puede enviar paquetes con una dirección IP distinta a la suya, haciéndole creer al receptor que está dialogando con otro cualquiera, y en el peor de los casos, con un host de confianza.

'Hijacking' o robo de sesión

Supongamos que estamos en una red Ethernet, donde varias estaciones de trabajo realizan conexiones TCP a un servidor que se encuentra en nuestro mismo segmento, y pongámonos en la piel de un intruso. .Si tenemos suficientes privilegios en nuestra máquina, podríamos utilizar un programa sniffer y escuchar todo lo que circula por este tramo, por ejemplo, una sesión telnet de una estación de trabajo con el servidor. En nuestra escucha pasiva a nivel de transporte podremos conocer el número de puerto y de secuencia en ambas partes con tan solo fijarnos en un segmento TCP. Esto es así porque en un solo segmento viajan los puertos origen y destino, el número de secuencia del emisor y el siguiente número de secuencia que se esperaba de la otra parte. Veamos el siguiente ejemplo :

Supongamos que somos el atacante y nuestro sniffer es el programa UNIX tcpdump. Estamos escuchando una conexión telnet entre una estación de trabajo con dirección IP 192.168.23.30, y un servidor con 192.168.23.1. Aparecen entonces dos paquetes :

192.168.23.30.1140 > 192.168.23.1.telnet: P 1320825536:1320825537(1) ack 1036160984 win 5840 <nop,nop,timestamp 4356776 390625> (DF)

192.168.23.1.telnet > 192.168.23.30.1140: P 1036160984:1036160985(1) ack 1320825537 win 5840 <nop,nop,timestamp 390646 4356776> (DF)

Esta sintaxis es la que utiliza el programa tcpdump. Los campos que nos interesan son los siguientes :

dirección_origen.puerto > dirección_destino.puerto : flags_TCP num_de_secuencia_inicial : num_de_secuencia_final (tamaño_del_los_datos_TCP) num_de_ack opciones

Vemos como el primer segmento va del cliente al servidor y es, seguramente, una tecla pulsada. El segundo es la respuesta del servidor reconociendo al cliente su último envío y enviándole la misma tecla para que pueda hacer el eco, tal y como lo establece el protocolo telnet.

Es entonces cuando el atacante podría entrar en juego: podría enviar un paquete IP con la dirección origen de la estación de trabajo, dirección destino del servidor, puerto origen y destino el que corresponda, y número de secuencia y ACK el calculado a partir de lo que escuchó. De esta forma, el servidor reconocería este paquete como válido y lo aceptaría aumentando el número de secuencia que espera de la estación de trabajo:

192.168.23.30.1140 > 192.168.23.1.telnet: P 1320825537:1320825560(23) ack 1036160985 win 5840 <nop,nop,timestamp 286213 6365981> (DF)

192.168.23.1.telnet > 192.168.23.30.1140: P 1036160985:1036161018(23) ack 1320825560 win 5840 <nop,nop,timestamp 6412341 286245> (DF)

¿ Qué ocurrirá con la estación de trabajo ?. Simplemente, que cuando envíe paquetes al servidor, éstos serán rechazados, ya que sus números de secuencia no corresponderán con el esperado; el servidor pensará que son paquetes duplicados. El propietario legítimo de esa conexión verá como, sin causa aparente, su sesión se ha colgado, y normalmente iniciará una nueva sin darle mayor importancia. Al fin y al cabo esas cosas ocurren.

Mientras tanto, el intruso podría haber enviado un paquete que contuviese un comando a su elección, con el consiguiente riesgo para el usuario legítimo.

Spoofing simple

Veamos ahora otra forma por la cual un atacante podría penetrar en un sistema. Supongamos el mismo escenario que en el caso del hijacking : cliente, servidor y atacante en el mismo segmento Ethernet. Supongamos que el usuario satomi de la máquina cliente tiene otra cuenta con el mismo nombre en el servidor, y para facilitar su conexión desde su lugar de trabajo habitual ha añadido en su rhosts del servidor la siguiente linea :

cliente.algo.cxm satomi

El atacante conoce esto e intenta lo siguiente: hace un programa que emule el funcionamiento del comando rlogin, pero que en la dirección origen de los paquetes que envía ponga la dirección IP de cliente.algo.cxm y en los campos 'usuario local' y 'usuario remoto' que el protocolo rlogin utiliza escriba satomi. Además, este programa deberá poner la interfaz en modo promiscuo para ver las respuestas que el servidor le da al verdadero cliente. Una vez está listo, el atacante empieza la prueba.

Su programa comienza con una conexión TCP al puerto 513 del servidor. Para ello, envía el siguiente segmento :

192.168.23.30.1022 > 192.168.23.1.login: S 915208618:915208618(0) win 32120 <mss 1460,sackOK,timestamp 963912 0,nop,wscale 0> (DF)

Vemos como el segmento enviado usa el flag SYN y el número de secuencia 915208618 para establecer la conexión.

Cuando el servidor lo recibe, envía su respuesta a la solicitud de conexión con un número de secuencia elegido por él y aceptando el número de secuencia del cliente incrementándolo en una unidad :

192.168.23.1.login > 192.168.23.30.1022: S 184994651:184994651(0) ack 915208619 win 32120 <mss 1460,sackOK,timestamp 53591392 963912,nop,wscale 0> (DF)

Para finalizar el saludo a tres vías de TCP, el intruso envía el último paquete al servidor :

> 192.168.23.30.1022 > 192.168.23.1.login: . ack 184994652 win 32120 <nop,nop,timestamp 963931 53591392> (DF)

Debemos darnos cuenta que, para construirlo, ha necesitado ver el número de secuencia que el servidor ha elegido, el 184994651, y lo ha incrementado en uno en señal de aprobación. Esto no habría sido posible si la interfaz no hubiese estado en modo promiscuo, ya que ese paquete no iba dirigido a él, sino al verdadero cliente. Es más, para que esta técnica funcione, el atacante debe estar situado en el mismo segmento Ethernet del servidor o del cliente, en el caso de que sean distintos.

Pero antes de enviar este último paquete, el atacante se da cuenta que un paquete viajó desde el verdadero cliente hasta el servidor.

> 192.168.23.30.1022 > 192.168.23.1.login: R 0:0(0) ack 184994652 win 0

Éste es un paquete que lleva el flag RESET . ¿Qué ocurrió?. Simplemente, que la computadora cliente estaba encendida, por lo que respondió al servidor con lo siguiente : _No quiero hacer ninguna conexión al puerto 513_. Cuando esto llega al servidor se aborta el intento de conexión.

Obviamente el spoofing simple no acaba aquí. ¾ Qué habría ocurrido si la máquina cliente hubiese estado apagada ?. Jamás habría respondido al servidor por lo que nuestro intruso podría haber seguido con el ataque. Pero ahora aparece un nuevo problema : si el cliente esta apagado no podrá responder a la consulta ARP que el servidor realiza para conocer su dirección MAC (es bastante probable que se tenga que hacer esta consulta si el cliente estaba apagado), por lo que jamás responderá a una solicitud de conexión por parte del falso cliente. Para evitarlo, es el intruso el que responde a la consulta ARP complicando un poco más su software.

Una vez se ha establecido la conexión, al intruso tan solo le resta seguir el protocolo del rlogin enviando el nombre de usuario local y el remoto, y acabará teniendo una sesión sin el consentimiento de su propietaria.

Por si esto fuese poco, para el caso en el que el verdadero cliente esté encendido, en algunos sistemas operativos es posible bloquear momentáneamente la escucha sobre un puerto. A esta técnica se le conoce como SYN flooding. Consiste en enviar en un breve periodo de tiempo muchas solicitudes de conexión (paquetes SYN) sin enviar el tercer paquete del saludo a tres vías. De esta forma se consigue llenar la cola de conexiones entrantes con conexiones medio abiertas, provocando que, hasta que no caduquen, ninguna solicitud de conexión será tramitada. Después del SYN flooding sobre el cliente, el intruso tendría tiempo para lanzar la conexión sobre el servidor sin que el cliente interfiriese en este proceso.

Para finalizar, comentar que hemos supuesto que el atacante conocía la existencia y el contenido del .rhosts de satomi en el servidor. Esto no es tan fácil, puesto que este fichero se halla dentro de $HOME. Pero es posible probar esta técnica de forma indiscriminada sobre todos los usuarios del servidor usando como IP origen la de sus respectivos puestos de trabajo, o cualquier computadora desde donde hallan hecho conexiones al servidor.

Spoofing a ciegas

Este tipo de spoofing es más complicado y difícil que el anterior, aunque igual de peligroso. El escenario ahora es distinto : supongamos que el intruso intenta llevar a cabo el ataque que vimos antes pero desde fuera del segmento del cliente y del servidor y fuera también de la ruta que los une. ¾ Cual es el problema ahora?. Esta vez, cuando su programa envíe el primer paquete para la conexión TCP, se quedará esperando la respuesta del servidor para conocer qué número de secuencia eligió, pero esta respuesta nunca llegará, ya que ahora este paquete de respuesta no pasará por él. Si no conoce el número de secuencia que el servidor eligió no será posible establecer una conexión TCP.

Pero el protocolo IP puede hacerle la vida más fácil a nuestro intruso. Existen un par de opciones denominadas 'enrutamiento estricto desde el origen' y 'enrutamiento libre desde el origen', que básicamente sirven para lo mismo : indicarle a los routers por donde encaminar nuestros paquetes y los

paquetes de vuelta. De esta forma el atacante incluirá en sus paquetes esta opción con los routers que debe atravesar para que los paquetes devueltos por el servidor no vayan al verdadero cliente, sino que lo hagan a su máquina. Afortunadamente, es raro que los routes actuales hagan caso a esta opción de un paquete IP, por lo que no tendremos en cuenta esta posibilidad.

Otra forma de conseguir un rerouting sería modificar la información de los routers. Esto puede ser complicado en caso de un routing estático, pero cuando en nuestra red usamos routing dinámico, el atacante podría enviar información de routing falsa al router, dependiendo claro está del protocolo que utilice (RIP, EGP, ...). De todas formas no trataremos este caso aquí.

Volvamos a cual era la traba que le aparecía al intruso : le era imposible ver el número de secuencia que el servidor elegía para iniciar la conexión. Pero, ¿de qué forma genera un host ese número de secuencia?.La especificación advertía originalmente que el número inicial de secuencia debería generarse a partir de un reloj interno de 32 bits que se suponía se incrementaba aproximadamente cada 4 microsegundos, aunque la realidad es muy distinta. En la mayoría de sistemas actuales podemos encontrar tres formas de generar estos números :

- La regla 64K : Esta regla es sorprendentemente simple. Se elige un número inicial cuando arranca el host, y se incrementa cada segundo por una constante, normalmente 128,000. Cuando una conexión es iniciada este contador aumenta en 64,000.

- Incremento por unidades de tiempo : esta es la regla más parecida a la especificación, en donde cada x unidades de tiempo se incrementa el contador, pero cada fabricante la implementa de una forma distinta, eligiendo unidades de tiempo distintas. Además, dependiendo de la carga del sistema este incremento no será ni mucho menos perfecto. Como ejemplo, en los antiguos núcleos de Linux el generador de números de secuencia se incrementaba en 1 cada microsegundo.

- Aleatorio : un ejemplo lo tenemos en los núcleos de Linux actuales.

Visto esto, el intruso podría optar por intentar adivinar el número de secuencia que va a elegir el servidor. Esto no parece muy difícil si el objetivo usa la regla 64K., pero es casi imposible si su generador de números de secuencia es aleatorio.

Supongamos ahora que el servidor utiliza la regla 64K. Esto se puede saber haciendo varias conexiones consecutivas : si los números de secuencia que recibimos difieren en 64,000 o en 192,000 sin duda estamos en este caso. Es entonces cuando el intruso empieza el ataque. Ahora su software se complica, debe averiguar, mediante la forma que hemos comentado antes, cual va a ser el número de secuencia que el servidor elegirá.

Recibe los siguientes números de secuencia como respuesta a sus conexiones :

1. 1217261284 2. 1217325284 3. 1217389284 4. 1217581284

Vemos como la diferencia entre 1-2 , 2-3 es de 64,000 y entre 3-4 es de 192,000 por lo que es probable que el siguiente número de secuencia elegido por el servidor sea 1217581284 + 64000, siempre y cuando el tiempo que tarda el paquete SYN en llegar al servidor sea de menos de un segundo. Así que el intruso prueba este número y envía estos paquetes separados por un pequeño instante de tiempo :

> 192.168.23.30.1177 > 192.168.23.1.login: S 712432833:712432833(0) win 32120 <mss1460, sackOK, timestamp 395512 0,nop,wscale 0> (DF)
>
> 192.168.23.30.1177 > 192.168.23.1.login: . ack 1217645284 win 32120 <nop,nop,timestamp 963931 53591392> (DF)

El primero inicia una conexión y el segundo confirma el número de secuencia elegido por el servidor. Ahora bien, él cómo sabe el atacante si su intento de conexión ha tenido éxito ?. Simplemente no lo sabe, ya que nunca va a recibir ningún paquete procedente del servidor. El único que los recibirá será el legítimo cliente, el cual habrá sido adormilado mediante un SYN flooding o algo similar que consiga el

mismo efecto. Después de esto, al cracker solo le resta enviar la misma información que usaba para emular el protocolo rlogin y esperar a que funcione.

Pero a primera vista puede parecer que algo se nos ha pasado por alto : ¿ qué ocurre con la información que envía el servidor ?. Si el atacante no puede ver ningún mensaje del servidor, acabará perdiendo la pista del número de ACK que debe enviarle para reconocerle que han llegado sus segmentos TCP. Pero esto no tiene demasiada importancia, ya que cuando el servidor vea que no ha recibido ACK tras un tiempo dado, comenzará a retransmitir. Aunque el tipo de retransmisión depende del algoritmo utilizado, esto es algo que no interfiere con los segmentos del intruso, que serán todos aceptados.

Detectar y evitar el IP spoofing

La detección de estos ataques no es fácil, pero se pueden hacer algunas cosas. Por ejemplo, en la shell sh de UNIX (bash en Linux) existe un fichero de historia que almacena los últimos comandos ejecutados. Con la ayuda de este fichero podemos descubrir si hemos sido víctimas de un ataque hijacking : si en algún momento se cuelga nuestra sesión telnet con otro computador, podemos comprobar qué comandos se ejecutaron por última vez viendo este fichero de historia. Si vemos comandos que no ejecutamos nosotros del tipo :

echo atacante.domain.com evil >> $HOME/.rhosts

sin duda hemos sido víctimas de un robo de sesión.

Para detectar ataques del tipo spoofing 'a ciegas' podemos incorporar al software servidor un tcpwrapper o envolvente que se encargue de registrar las conexiones anómalas, como por ejemplo las que no trasmiten datos (consisten en el establecimiento de la conexión mediante el saludo a tres vías y la inmediata desconexión), las que se quedan medio abiertas (tan solo envían el primer segmento SYN), etc. De esta forma estaremos registrando parte de los intentos de adivinar el siguiente número de secuencia, como ya se explicó antes.

Pero esto no es infalible, ya que si nuestro host fuese un servidor ftp público o un servidor web, cualquiera podría hacer conexiones consecutivas aparentemente normales a esos servicios pero con el fín de conocer el algoritmo de generación de números de secuencia.

Para evitar estos ataques podemos adoptar algunas medidas. El uso de conmutadores en nuestra red soluciona completamente el problema del robo de sesión y el spoo½ng simple, ya que evitan las escuchas de conversaciones ajenas. El spoofing 'a ciegas' requiere otras medidas como pueden ser :

- Configuración de los routers que dan acceso al exterior para que no permitan la entrada de paquetes con una dirección origen perteneciente a nuestra red.
- Uso de sistemas con generadores de números de secuencia aleatorios para evitar que puedan ser adivinados.
- Uso de sistemas protegidos al ataque de SYN flooding.
- Evitar en lo posible el uso de los ½cheros rhosts y /etc/host.equiv .

Pese a que este ataque es tan conocido, algunos sistemas operativos como el IRIX 6.2 ó 6.5 usan generadores de números de secuencia fáciles de adivinar, y otros como el HP-UX 10.20 implementan la regla 64K.

Existen escaneadores de puertos como el nmap que además nos dice qué sistema operativo se ejecuta en el host que estamos escaneando y la dificultad de adivinar sus números de secuencia, por lo que un racker que encuentre una máquina que utilice la reglas 64K. no dudará en hacer un seguimiento de ésta. Si, además, el host en cuestión tiene abierto el puerto finger, el intruso podrá ver gratuitamente los nombres de usuario y las direcciones de las máquinas desde las que hicieron su conexión, y a partir de ahí construirse una base de datos para luego hacer una prueba masiva.

ARP spoofing

El Protocolo de Resolución de Direcciones (ARP) también puede ser utilizado para un ataque de spoofing. Veamos un ejemplo de como podría llevarse a cabo por un intruso :

Supongamos que en nuestra LAN tenemos dos computadoras : sirio, un servidor, y algol,un PC Linux usado por satomi. El usuario satomi, también tiene una cuenta con el mismo nombre en sirio, y ha añadido en el fichero .rhosts de éste último la siguiente entrada :

algol satomi

Esto facilitará sus conexiónes por rlogin. Además, como los administradores son un poco paranoicos, para evitar el IP spoofing toda la red está formada por conmutadores y el servidor tiene un generador de números de secuencia aleatorios.

En nuestra LAN también existen otros hosts, y uno de ellos está siendo utilizado por un trabajador descontento para lanzar un ataque sobre sirio. El intruso conoce la existencia en sirio del ½chero rhosts de satomi, y también conoce las medidas de seguridad que los administradores han adoptado. Es entonces cuando se le ocurre lanzar un ataque de spoo½ng ARP contra sirio suplantando a algol.

El ataque consiste en enviar a sirio información ARP falsa diciendo que la dirección MAC del host del atacante corresponde con la dirección IP de algol. ¾ Cómo se puede hacer esto ?. Muchas implementaciones de ARP, cuando escuchan una consulta ARP (recordemos que la consulta es broadcast), fichan esa dirección MAC aunque ellos no sean los buscados. Lo que podría hacer el intruso sería enviar una consulta ARP buscando a un host inexistente y, dentro del paquete ARP, cambiar su dirección MAC por la dirección MAC dealgol. Tan solo tiene que hacerlo dentro del paquete y no en la cabecera MAC porque lo que recibe el software ARP de sirio es la consulta libre de la cabecera MAC, que fue eliminada por la capa de nivel de enlace.

Una vez hecha la consulta, sirio incorporará en su cache ARP esta información, y cuando se quiera comunicar con algol sus paquetes irán dirigidos a la dirección IP de algol pero a la dirección MAC del intruso, y éste se podría aprovechar la relación de con½anza que existe entre el sirio y algol.

Una forma de frustrar este ataque es establecer asignaciones estáticas de direcciones hardware, pero si se invento ARP fue precisamente para evitar esto. En redes pequeñas quizá pueda llevarse a cabo, pero cuando una LAN es demasiado grande esto puede requerir mucho tiempo y esfuerzo.

Otra solución menos incomoda sería el ARPWATCH, un programa que vigila los cambios en la cache ARP y nos informa a través del correo electrónico y ½cheros de log de posibles variaciones.

DNS spoofing

Este es quizá el menos habitual de los ataques de spoofing, pero conviene conocerlo. El ½n del intruso es engañar a un host con una correspondencia 'nombre del host-dirección IP' falsa. Esto se puede conseguir de tres formas :

1. Falsificando una respuesta del DNS al host que se quiere atacar, de tal forma que la información alterada quede en su cache.
2. Modificando una transferencia de zona de un servidor de nombres primario a uno secundario.
3. Logrando acceso al DNS primario y modificando su base de datos para que una consulta posterior por parte del host a atacar le devuelva información falsa.

Siguiendo con el ejemplo anterior, supongamos ahora que los administradores de sirio han detectado ataques ARP spoofing con la ayuda del ARPWATCH y deciden hacer permanente la cache ARP solamente en este servidor. El intruso se da cuenta de esto y, conociendo una vulnerabilidad del DNS, logra acceso a él. Entonces modifica la entrada de algol cambiando la dirección IP que había por la suya.

Cuando sirio reciba una conexión del intruso como usuario satomi, mirará el fichero .rhosts de satomi y vera que si la conexión se hace desde algol no requerirá passsword. Hace una consulta al DNS para conocer la dirección IP de algol y el servidor de nombres le responde con la IP del intruso. sirio

comprueba el origen de la conexión rlogin y ve que coincide con la IP devuelta por el DNS, por lo que acepta su conexión y no le pide ningún tipo de contraseña.

Este ataque también se podría haber llevado a cabo sin haber conseguido acceso al servidor de nombres, simplemente falsificando la respuesta del DNS ante la consulta que hace sirio o modificando una transferencia de zona de un primario a un secundario en el caso de que la consulta de sirio fuese al secundario.

Si el DNS tiene contacto directo con la red externa, para evitar que un intruso pueda explotar un fallo en su implementación o en su configuración podríamos aislarlo de la red. Podemos conseguir este aislamiento utilizando dos servidores de nombres y un cortafuego. El cortafuego se situaría entre nuestra red y la red exterior, y colocaríamos un DNS al frente del cortafuego y otro detrás. El de detrás del cortafuegos almacenaría todos los nombres y direcciones IP de nuestra red, y sería el encargado de responder consultas a hosts internos. El que se encuentra al frente estaría vacío de información y se dedicaría a responder consultas de hosts externos preguntando al DNS interno. Así, nuestro cortafuego filtraría las conexiones externas al servidor de nombres interno y tan solo permitiría consultas de fuera que viniesen del DNS al frente, evitando cualquier intento de irrumpir en el servidor de nombres que contiene toda la información de nuestra red.

Si además configuramos nuestro router de entrada para que no deje pasar paquetes de fuera con una dirección origen perteneciente a nuestra red, evitaremos cualquier intento de spoo½ng sobre el DNS interno suplantando al externo.

ANEXO 2: ATAQUES DE DENEGACIÓN DE SERVICIO[4]

El DoS (Denial of Service), también conocido como Negación del Servicio, no es mas que una forma voluntaria o involuntaria (normalmente con propósito) de dejar sin servicio a un nodo de la red. Suele ser voluntaria (utilizada por los típicamente llamados hackers) y orientada hacia nodos servidores.

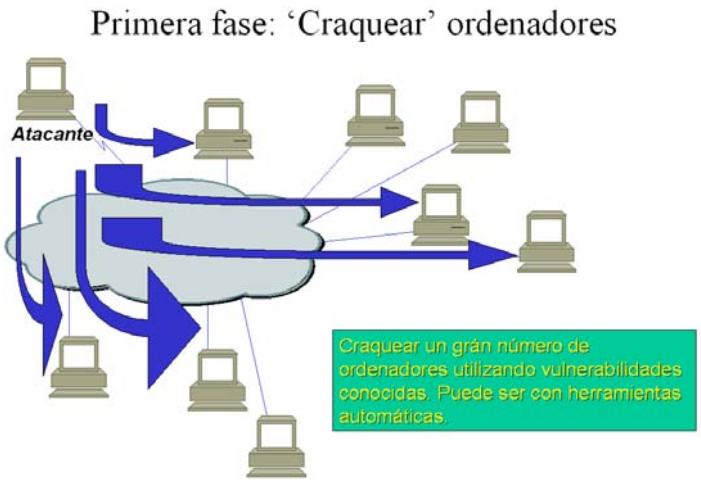
Esto se consigue mediante el uso abusivo de recursos de la máquina, mediante "inundación" de peticiones por red, etc... aunque hay muchas formas mas. El unico propósito de esto cuando es voluntario, puede ser la intención de aislar una red dejando sin servicio el nodo que conecta con el exterior, reconducir el tráfico actuando falsamente como dicho nodo, etc... aunque tambien es muy utilizado, como es lógico, por administradores de sistemas y demas operadores para comprovar la vulnerabilidad de su red.

El problema del DoS es que cualquier red es vulnerable a ello independientemente del Sistema Operativo utilizado (aunque muchas veces suele ser el centro del error). Esto sucede, ya que se basa en la ineficiencia del sistema frente a inundaciones (flooding) de paquetes, también debido a los desbordamientos de buffers, pilas y demas estructuras. Todo esto unido a fallos en los protocolos, como el IPv4 y aprovechamiento de las capacidades de los UDP, los ICMP, etc...

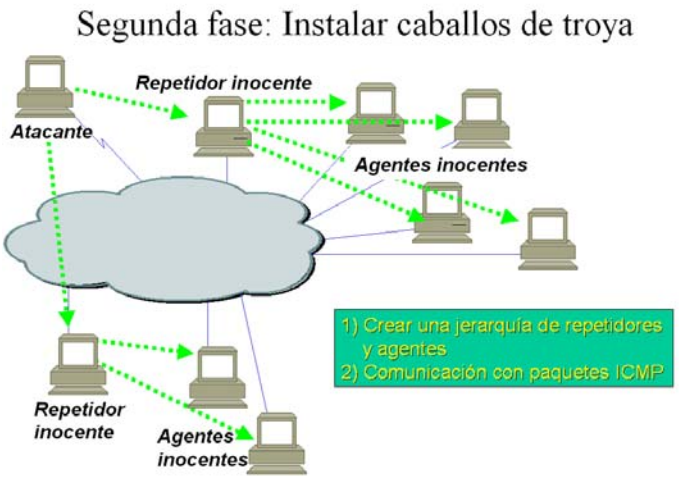
Las formas de crear un DoS son muy diversas. Las mas conocidas son: Syn Flooding, Echo/Chargen, Smurf, ataques DNS, TearDrop, Spool, Proquota, ICMP Flooding, UDP spoofing, Ping of Death, Snork, trinoo, TFN2, stacheldraht, etc... otros mas simples y normalmente accidentales son el nonblocking I/O, bucle fork(), Stacks & Buffers, etc... (estos se sulen basar en errores de programación).

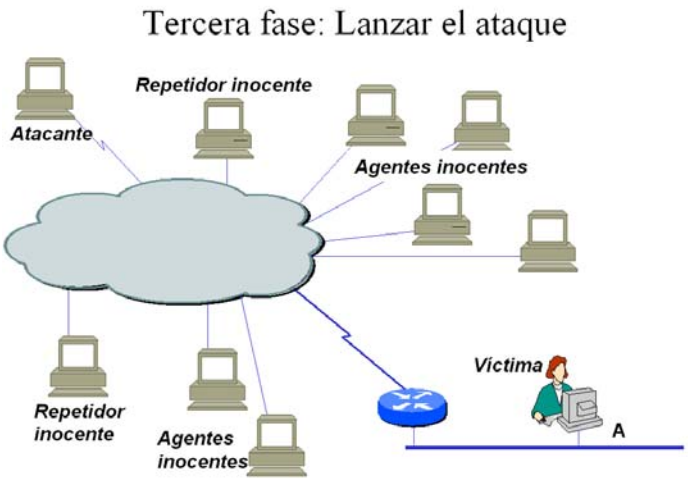
Los ataques DoS se podrian clasificar muy basicamente en: remotos y locales. Remotos serian aquellos realizados desde un host distinto a la red de la maquina, mientras que locales serian aquellos relizados en la propia máquina objetivo. Dentro de los remotos podemos incluir los DDoS (Distribuited Denial of Service)

Veamos un ejemplo en las siguientes figuras:



[4] Este anexo corresponde en parte a un trabajo de la asignatura de Redes, entregado por el alumno Alberto Chovares





Un DDoS es un DoS coordinado realizado desde multiples hosts hacia un objetivo. Utilizando la arquitectura Cliente/Servidor se puede iniciar un DoS a un objetivo desde cada host utilizando uno como "master" que a traves de dicha arquitectura de programacion controla a los demas. De este modo se puede conseguir inundar al objetivo con miles de peticiones provocándole una "negacion de servicio".

Vamos a ver los tipos básicos de DoS....

Errores de programacion

Estos suelen ser causados involuntariamente y su error provoca un DoS. En el primer ejemplo se produce un uso abusivo de los recursos y en el segundo un bloqueo. La gran ventaja de este tipo de DoS, es que son fácilmente detectables y corregibles.

EJEMPLO 1

```
main( )
{
        while (1)
                fork ( );
}
```

Aunque este simple programa esta escrito con proposito, muchas veces suele ocurrir la aparicion de un fork( ) reiterativo, en este caso por culpa de un bucle... Esto hace que la máquina consuma todos los recursos disponibles para estos procesos a medida que van aumentando exponencialmente. Normalmente esto se puede evitar limitando el numero de procesos permitidos a un usuario.

EJEMPLO 2

```
.
.
.
for (i=0; i<= maxi; i++)
{
        if ( (sockfd=client[i])<0)
                continue;
        if (FD_ISSET(sockfd,&rset))
        {
                if ( (n=Readline(sockfd,line,MAXLINE) )==0)
                {
.
.
.
```

Este es un fragmento de código extraido de un servidor TCP para mostrar otro de los problemas. Aquí el error se encuentra, en que cada vez que el servidor acepta un cliente, lee un linea (un numero determinado de bytes) del socket y sigue con los demás. El problema aparece cuando un cliente envia simplemente un byte y por las razones que sean ya no envia mas. Esto produce bloqueo DoS en el servidor, este problema de programación se conoce tambien como *nonblocking I/O*.

Consumo de ancho de banda

Este es el tipo de DoS mas utilizado, se suele utilizar en una red local o particular, pero es mas comun consumir recursos remotamente. Existen dos formas básicas de que se produzca:

Escenario 1
El host remoto que provoca el DoS en el servidor lo consigue porque tiene un ancho de banda mayor. Por ejemplo, una conexión T1(1'544Mbps) inunda un enlace de 128Kbps. Este tipo de negación de servicio se suele producir en redes de baja velocidad.

Escenario 2
Suponiendo un ancho de banda menor se puede "amplificar" uniendo varios sitios para inundar la red objetivo. Esto seria utilizando un DDoS. Por ejemplo inundar una línea T3(45Mbps) mediante la unión de muchas líneas de 56Kbps hasta superar el ancho de la T3.

Inanicion de recursos

Este difiere del consumo de ancho de banda, en que esta mas enfocado al consumo de recursos del sistema que de recursos de red. Generalmente esto se produce por la saturacion de la CPU, memoria, sistema cuotas, de archivos... Esto produce normalmente un fallo general del sistema.

Enrutamiento

Un ataque DoS basado en enrutamiento consiste en que los atacntes manipulan las tablas de distribucion o enrutamiento, produciendo así la negación del servicio a los sistemas legítimos. La mayoria de los

protocolos de enrutamiento, tales como RIPv1 (Routing Information Protocol) y BGPv4 (Border Gateway Protocol), carecen o tienen una autenticación muy sencilla.
De esta manera se puede crear un DoS y/o en su defecto enrutar o redirigir el tráfico de forma diferente, incluso a un "agujero negro" (una red que no existe).

DNS

Los DoS sobre servidores de nombres de dominio (DNS), son tan problemáticos como los anteriores, almacenando información falsa en la caché del servidor. Esto hace posible al igual que anteriormente redirigir al host que consulta el DNS a otras direcciones o incluso a ninguna. Hoy en dia esto es muy utilizado, haciendo que grandes sitios web queden inaccesibles durante cierto tiempo.

Explicando ya, formas concretas y para no comentar todas las condiciones DoS imaginables, comentaremos las mas relevantes, conocidas y utilizadas....

Smurf y Fraggle

Este tipo de ataque realiza una petición de ping difundida y dirigida una red de sistemas que responderá. Se puede enviar a un direccion de la red o una dirección de difusión. Este requiere un dispositivo que ejecute la función de difusión de la capa 3 (IP) a la capa 2 (red).

En una red de clase C (24 bits) tendriamos que la dirección de red sería .0 y la de difusion sería .255 (este tipo de difusión se utiliza para diagnósticos sobre el funcionamiento de las propias máquinas).
Este ataque envia paquetes ICMP ECHO falsificados a la red de difusión que actuará de red amplificadora. La dirección de origen del paquete ICMP ECHO debe ser la dirección del host/server objetivo, mediante *spoofing* (flasificando el paquete). Esto provocará la respuesta de toda la red hacia el objetivo, terminando en un DoS.

Esta es una variante del Smurf que en vez de utilizar paquetes ICMP utiliza UDP asignandoles a la direccion origen la direccion del objetivo igual que antes. La diferencia es que ha de recurrir a enviar a la red "amplificadora" y concretamente al puerto 7 (echo) de sus hosts. De esta manera devuelven un "echo" de lo recibido y se obtiene el mismo resultado anterior, DoS.

Para evitar esto, es posible desactivar la función de difusiones dirigidas de los routers, también se pueden configurar los sistemas operativos para rechazar paquetes echo de difusión. Ademas se puede limitar el trafico de ICMP, UDP en los routers.

SYN flooding

Hasta que llegaron tipos nuevos de DoS (smurf, fraggle...) este era la condición de DoS más utilizada. Este es conocido como inundación SYN.

Cuando se inicia una comunicación TCP, comienza un proceso de tres vias que consiste en: un paquete SYN del cliente al servidor (estado *listen*), un paquete SYN/ACK de respuesta del servidor (estado *syn_recv*) y por ultimo un ACK desde el cliente (estdo *established*). Cuando la conexión no se ha establecido totalmente, se dice que es una conexión potencial. Normalmente el protocolo designa pocos recursos a una conexión potencial (que aun no está establecida) y por tanto es facil provocar un DoS.

Supongamos un sistema A que envia un paquete SYN al sistema B (objetivo) pero mediante IP Spoofing (se conoce como spoofing a falsear la direccion origen, en este caso un host inexistente). En este momento B enviará un SYN/ACK a la dirección inexistente (si existiera, B recibiria un paquete RST de respuesta indicando que no ha pedido comunicación). Como no recibe ACK inmediatamente se crea una cola que según el protocolo dura cierto tiempo (estdo *syn_recv*). En este momento si B recibe mas peticiones del mismo modo (antes de terminar el tiempo para la cola) agotará sus recursos porque no podra eliminar las colas y caera en DoS.

Aumentar el tamaño de la cola es una de las soluciones posibles aunque tiene como desventaja un consumo de recursos que puede afectar al sistema. Otra solución podria ser disminuir el tiempo de establecimiento de conexión aunque no es realmente buena. Hoy en dia los sistemas operativos ya tienen soluciones diversas. Linux por ejemplo utiliza SYN "cookie", que contraresta el flood de SYN mediante criptografia y permite a los usuarios legítimos seguir conectandose. WinNT utiliza la asignación de recursos adicionales cuando la cola cae por debajo de un umbral.

TearDrop / NewTear / SynDrop / Bonk / Boink / SSPing / Targa / ...

Este tipo de ataque explota las vulnerabilidades del código de ensamblaje de paquetes de las implantaciones específicas de las pilas IP. Como los paquetes atraviesan diferentes redes, puede ser necesario fragmentarlos basandose en la maxima unidad de transmision en redes, MTU (Maximum Transmission Unit).

Algunos kernels Linux, Win95 y WinNT tenian el problema de comprobar si los paquetes eran demasiado grandes (para fragmentarlos), pero no si eran demasiado pequeños... Con esto, enviando paquetes de un tamaño específicamente pequeño podia caer el sistema en DoS debido a un error de ensamblaje.

Hoy en dia este problema, de momento, esta corregido para los nuevos kernels Linux y existen service packs para WinNT i patch para Windows 9x.

Ping of Death: ping de la muerte

Los paquetes IP (segun la RFC-791) pueden ser mayores de 65535 bytes de longitud incluida la cabecera (20 bytes si no se especifican opciones). Para los paquetes mayores la subcapa no puede reconstruirlos. Esto consiste en enviar paquetes ICMP tan grandes que al reconstruirlos, se desborden los buffers provocando DoS. (Actualmente este problema esta corregido y habian patch para Win95 y WinNT)

EJEMPLOS

ping -l 65510 direccionIP
ping -165527 -s 1 direccionIP

Snork

Este metodo consiste en enviar, mediante spoofing, paquetes UDP al puerto 135 de un servidor. (En este caso el spoofing consiste en incluir como direccion origen la de otro servidor). Cuando el servidor B recibe el paquete UDP parece que se ha equivocado y le envia un REJECT al servidor A , el cual responde con otro REJECT. Esto solo dura unos minutos hasta que "rompan" el paquete, pero si se utilizan muchos mas se creara un loop que consumira los recursos e incluso el ancho de banda.

KillWin / WinNuke

Enviando al puerto 139 (Netbios) datos especificando "Out of Band" se puede crear un DoS. Se consigue activando en la cabecera TCP el URGENT bit flag. De este modo el sistema operativo utiliza el URGENT POINTER para determinar donde terminan los datos urgentes. Cuando el puntero apunta al final de la trama y encuentra datos no esperados o no normales provoca el error del sistema.

Chargen y Chargen-Echo

Parecido al Fraggle, este se basa en enviar datagramas UDP mediante spoofing (con la direccion origen = direccion objetivo) al puerto 19 (chargen) de todos los hosts de una subred (con direccion broadcast) de forma que responderan inundando el objetivo con el generador de caracteres.

Combinando el puerto 19 (chargen) y el puerto 7 (echo) se puede conseguir un DoS si creamos un socket UDP al puerto 7 y enviamos un datagrama UDP al puerto 19. Esto provoca que el chargen genere caracteres que seran respondidos al puerto echo y que este a su vez devolvera al puerto 19 y asi sucesivamente.

Trinoo / Tribal Flood Network / Stacheldraht / ...

Es un DDoS que necesita de otros hosts. A partir de una lista de IPs uno actua de "master" iniciando un UDP flooding al objetivo con la ayuda de los demas hosts a través de un puerto. Se puede determinar viendo si existen conexiones TCP de tipo 6 en el puerto 27655 (normalmente utilizado).

El TFN y sus variantes (existen varias), es otra variante de Trinoo que necesita un master y hosts para realizar un flooding coordinado, pero a diferencia de Trinoo este puede realizarlo mediante UDP flood, TCP SYN flood, ICMP echo y ICMP broadcast. Además utiliza encriptacion "blowfish" para enviar su lista de IPs.

ANEXO 3: PUERTOS ASIGNADOS EN PROTOCOLOS TCP Y UDP. FICHERO /ETC/SERVICES

```
# Revision: 1.32.214.7 $ $Date: 97/09/10 14:50:42 $
#
# This file associates official service names and aliases with
# the port number and protocol the services use.
#
# Some of the services represented below are not supported on HP-UX.
# They are provided solely as a reference.
#
# The form for each entry is:
# <official service name>  <port number/protocol name>  <aliases>
#
# See the services(4) manual page for more information.
# Note: The entries cannot be preceded by a blank space.
#
tcpmux         1/tcp                # TCP port multiplexer (RFC 1078)
echo           7/tcp                # Echo
echo           7/udp                #
discard        9/tcp  sink null     # Discard
discard        9/udp  sink null     #
systat        11/tcp  users         # Active Users
daytime       13/tcp                # Daytime
daytime       13/udp                #
qotd          17/tcp  quote         # Quote of the Day
chargen       19/tcp  ttytst source  # Character Generator
chargen       19/udp  ttytst source  #
ftp-data      20/tcp                # File Transfer Protocol (Data)
ftp           21/tcp                # File Transfer Protocol (Control)
telnet        23/tcp                # Virtual Terminal Protocol
smtp          25/tcp                # Simple Mail Transfer Protocol
time          37/tcp  timeserver    # Time
time          37/udp  timeserver    #
rlp           39/udp  resource      # Resource Location Protocol
whois         43/tcp  nicname       # Who Is
domain        53/tcp  nameserver    # Domain Name Service
domain        53/udp  nameserver    #
bootps        67/udp                # Bootstrap Protocol Server
bootpc        68/udp                # Bootstrap Protocol Client
tftp          69/udp                # Trivial File Transfer Protocol
rje           77/tcp  netrjs        # private RJE Service
finger        79/tcp                # Finger
http          80/tcp  www           # World Wide Web HTTP
http          80/udp  www           # World Wide Web HTTP
link          87/tcp  ttylink       # private terminal link
supdup        95/tcp                #
hostnames    101/tcp  hostname      # NIC Host Name Server
tsap         102/tcp iso_tsap iso-tsap # ISO TSAP (part of ISODE)
pop          109/tcp postoffice pop2 # Post Office Protocol - Version 2
pop3         110/tcp  pop-3         # Post Office Protocol - Version 3
portmap      111/tcp  sunrpc        # SUN Remote Procedure Call
portmap      111/udp  sunrpc        #
ident        113/tcp  authentication # RFC1413
sftp         115/tcp                # Simple File Transfer Protocol
uucp-path    117/tcp                # UUCP Path Service
nntp         119/tcp  readnews untp  # Network News Transfer Protocol
ntp          123/udp                # Network Time Protocol
netbios_ns   137/tcp                # NetBIOS Name Service
netbios_ns   137/udp                #
netbios_dgm  138/tcp                # NetBIOS Datagram Service
netbios_dgm  138/udp                #
netbios_ssn  139/tcp                # NetBIOS Session Service
netbios_ssn  139/udp                #
bftp         152/tcp                # Background File Transfer Protocol
snmp         161/udp  snmpd         # Simple Network Management Protocol Agent
snmp-trap    162/udp  trapd         # Simple Network Management Protocol Traps
bgp          179/tcp                # Border Gateway Protocol
# PV performance tool services entries
pvserver     382/tcp    # PV server
pvalarm      383/tcp    # PV alarm management
#
# UNIX services
#
biff         512/udp  comsat         # mail notification
```

```
exec         512/tcp                 # remote execution, passwd required
login        513/tcp                 # remote login
who          513/udp  whod           # remote who and uptime
shell        514/tcp  cmd            # remote command, no passwd used
syslog       514/udp                 # remote system logging
printer      515/tcp  spooler        # remote print spooling
talk         517/udp                 # conversation
ntalk        518/udp                 # new talk, conversation
route        520/udp  router routed  # routing information protocol
efs          520/tcp                 # Extended file name server
timed        525/udp  timeserver     # remote clock synchronization
tempo        526/tcp  newdate        #
courier      530/tcp  rpc            #
conference   531/tcp  chat           #
netnews      532/tcp  readnews       #
netwall      533/udp                 # Emergency broadcasting
uucp         540/tcp  uucpd          # uucp daemon
remotefs     556/tcp  rfs_server rfs # Brunhoff remote filesystem
ingreslock  1524/tcp                 #
#
# Other HP-UX services
#
lansrm       570/udp                 # SRM/UX Server
DAServer     987/tcp                 # SQL distributed access
instl_boots 1067/udp                 # installation bootstrap protocol server
instl_bootc 1068/udp                 # installation bootstrap protocol client
nfsd-keepalive 1110/udp              # Client status info
nfsd-status    1110/tcp              # Cluster status info
msql        1111/tcp                 # Mini SQL database server
rlb         1260/tcp                 # remote loopback diagnostic
clvm-cfg    1476/tcp                 # HA LVM configuration
diagmond    1508/tcp                 # Diagnostic System Manager
nft         1536/tcp                 # NS network file transfer
sna-cs      1553/tcp                 # SNAplus client/server
sna-cs      1553/udp                 # SNAplus client/server
ncpm-pm     1591/udp                 # NCPM Policy Manager
ncpm-hip    1683/udp                 # NCPM Host Information Provider
cvmon       1686/udp                 # Clusterview cvmon-cvmap communication
registrar   1712/tcp                 # resource monitoring service
registrar   1712/udp                 # resource monitoring service
ncpm-ft     1744/udp                 # NCPM File Transfer
psmond      1788/tcp                 # Predictive Monitor
psmond      1788/udp                 # Hardware Predictive Monitor
pmlockd     1889/tcp                 # SynerVision locking daemon
pmlockd     1889/udp                 #
nfsd        2049/udp                 # NFS remote file system
netdist     2106/tcp                 # update(1m) network distribution service
rfa         4672/tcp                 # NS remote file access
veesm       4789/tcp                 # HP VEE service manager
hacl-hb     5300/tcp                 # High Availability (HA) Cluster heartbeat
hacl-gs     5301/tcp                 # HA Cluster General Services
hacl-cfg    5302/tcp                 # HA Cluster TCP configuration
hacl-cfg    5302/udp                 # HA Cluster UDP configuration
hacl-probe  5303/tcp                 # HA Cluster TCP probe
hacl-probe  5303/udp                 # HA Cluster UDP probe
hacl-local  5304/tcp                 # HA Cluster Commands
hacl-test   5305/tcp                 # HA Cluster Test
hacl-dlm    5408/tcp                 # HA Cluster distributed lock manager
lanmgrx.osB 5696/tcp                 # LAN Manager/X for B.00.00 OfficeShare
r4-sna-cs   5707/tcp                 # SNA client/server (up to Release 4.1)
SNAplus     5708/udp                 # SNA logical network A (up to Release 4.1)
r4-sna-ft   5709/tcp                 # SNA file transfer (up to Release 4.1)
hcserver    5710/tcp                 # HP Cooperative Services
grmd        5999/tcp                 # graphics resource manager
spc         6111/tcp                 # sub-process control
desmevt     6868/tcp                 # DE/ Services Monitor, Event Service
pdclientd   6874/tcp                 # Palladium print client daemon
pdeventd    6875/tcp                 # Palladium print event daemon
iasqlsvr    7489/tcp                 # Information Access
recserv     7815/tcp                 # SharedX Receiver Service
ftp-ftam    8868/tcp                 # FTP->FTAM Gateway
mcsemon     9999/tcp                 # MC/System Environment monitor
console     10000/tcp                # MC/System Environment console multiplexor
actcp       31766/tcp                # ACT Call Processing Server
#
# Kerberos (Project Athena/MIT) services
#
kerberos5    88/udp   kdc                 # Kerberos 5 kdc
klogin       543/tcp                  # Kerberos rlogin -kfall
kshell       544/tcp  krcmd            # Kerberos remote shell -kfall
ekshell      545/tcp  krcmd            # Kerberos encrypted remote shell -kfall
```

```
kerberos     750/udp  kdc            # Kerberos (server) udp -kfall
kerberos     750/tcp  kdc            # Kerberos (server) tcp -kfall
kerberos_master 751/tcp kadmin       # Kerberos kadmin
krbupdate    760/tcp  kreg           # Kerberos registration -kfall
kpasswd      761/tcp  kpwd           # Kerberos "passwd" -kfall
eklogin     2105/tcp                 # Kerberos encrypted rlogin -kfall
# The X10_LI server for each display listens on ports 5800 + display number.
# The X10_MI server for each display listens on ports 5900 + display number.
# The X11 server for each display listens on ports 6000 + display number.
# The X11 font server listens on port 7000.
# Do NOT associate other services with these ports.
# Refer to the X documentation for details.

hpoms-ci-lstn   5403/tcp      #SAP spooler support
hpoms-dps-lstn  5404/tcp      #SAP spooler support
samd            3275/tcp        # sam daemon

dtspc   6112/tcp       #subprocess control
```