**Tutorial TCP/IP y**

**Descripción técnica.**

(Cap. 1, 3, 4, 5, 12, 17, 22 y 24)

* [**https://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf**](https://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf)

### 1.1 Modelo arquitectónico TCP/IP

El conjunto de protocolos TCP/IP se llama así por dos de sus protocolos más importantes:

Protocolo de control de transmisión (TCP) y Protocolo de Internet (IP). Un menos utilizado

Su nombre es el Conjunto de Protocolos de Internet, que es la frase utilizada en los documentos oficiales de estándares de Internet. En este libro, usamos el término más común y más corto, TCP/IP, para referirnos a todo el conjunto de protocolos.

#### 1.1.1 Interconexión

El objetivo principal del diseño de TCP/IP era construir una interconexión de redes, denominada *internetwork*, o *Internet*, que proporcionara servicios de comunicación universales a través de redes físicas heterogéneas. La clara ventaja de este tipo de interconexión es la posibilidad de comunicación entre hosts en diferentes redes, tal vez separadas por una gran área geográfica.

Las palabras internetwork e internet son simplemente una contracción de la frase interconnected network. Sin embargo, cuando se escribe con "I" mayúscula, Internet se refiere al conjunto mundial de redes interconectadas. Por lo tanto, Internet es una Internet, pero no se aplica lo contrario. A veces se denomina *Internet conectada*.

Internet se compone de los siguientes grupos de redes:

Redes troncales: Grandes redes que existen principalmente para interconectar otras redes. También conocidos como puntos de acceso a la red (NAP) o puntos de intercambio de Internet (IXP). En la actualidad, las columnas vertebrales están formadas por entidades comerciales.

Redes regionales que conectan, por ejemplo, universidades y colegios.

Redes comerciales que proporcionan acceso a las redes troncales a los suscriptores, y redes propiedad de organizaciones comerciales para uso interno que también tienen conexiones a Internet.

Redes locales, como las redes universitarias de todo el campus.

En la mayoría de los casos, el tamaño de las redes está limitado por el número de usuarios que pueden pertenecer a la red, por la distancia geográfica máxima que puede abarcar la red o por la aplicabilidad de la red a determinados entornos. Por ejemplo, una red Ethernet está inherentemente limitada en términos de tamaño geográfico. Por lo tanto, la capacidad de interconectar un gran número de redes en algunos

La forma jerárquica y organizada permite la comunicación de dos hosts cualesquiera que pertenezcan a esta interred.

La Figura 1-1 muestra dos ejemplos de Internet. Cada uno consta de dos o más redes físicas.

rconectado por un

Dos redes inte

rou

Ter

Iguales

Internet A

Enrutador

R

Uno

Virtual

Red

Netw

Orko 1

Red 2

Enrutador

R

Red

Trabajo 3

Netw

Orko 1

Red 2

Enrutador

R

Múltiples redes

s interconectados por routers

(

también se ve como 1 red virtual, una red de Internet

)

*Figura 1-1 Ejemplos de Internet: Dos conjuntos de redes interconectadas, cada una vista como una red lógica*

Otro aspecto importante de la interconexión TCP/IP es la creación de una abstracción estandarizada de los mecanismos de comunicación proporcionados por cada tipo de red. Cada red física tiene su propia interfaz de comunicación dependiente de la tecnología, en forma de interfaz de programación que proporciona funciones básicas de comunicación (primitivas). TCP/IP proporciona servicios de comunicación que se ejecutan entre la interfaz de programación de una red física y las aplicaciones de usuario. Permite una interfaz común para estas aplicaciones, independiente de la red física subyacente. Por lo tanto, la arquitectura de la red física está oculta para el usuario y para el desarrollador de la aplicación. La aplicación solo necesita codificar para la abstracción de comunicación estandarizada para poder funcionar bajo cualquier tipo de red física y plataforma operativa.

Como es evidente en la Figura 1-1, para poder interconectar dos redes, necesitamos una computadora que esté conectada a ambas redes y pueda reenviar paquetes de datos de una red a otra; Una máquina de este tipo se llama enrutador. El término enrutador IP también se utiliza porque la función de enrutamiento forma parte de la parte del protocolo de Internet del conjunto de protocolos TCP/IP (consulte 1.1.2, "Las capas del protocolo TCP/IP" en la página 6).

Para poder identificar un host dentro de la interred, a cada host se le asigna una dirección, llamada dirección IP. Cuando un host tiene varios adaptadores de red (interfaces), como en el caso de un enrutador, cada interfaz tiene una dirección IP única. La dirección IP consta de dos partes:

Dirección IP = <número de red><número de host>

El número de red que forma parte de la dirección IP identifica la red dentro de Internet y es asignada por una autoridad central y es única en Internet. La autoridad para asignar la parte del número de host de la dirección IP reside en la organización que controla la red identificada por el número de red. Describimos el esquema de direccionamiento en detalle en 3.1.1, "Direccionamiento IP" en la página 68.

#### 1.1.2 Las capas de protocolo TCP/IP

Como la mayoría del software de red, TCP/IP se modela en capas. Esta representación en capas conduce al término pila de protocolos, que se refiere a la pila de capas en el conjunto de protocolos. Se puede utilizar para posicionar (pero no para comparar funcionalmente) el conjunto de protocolos TCP/IP con otros, como la arquitectura de red de sistemas (SNA) y el modelo de interconexión de sistemas abiertos (OSI). Las comparaciones funcionales no se pueden extraer fácilmente de esto, porque existen diferencias básicas en los modelos en capas utilizados por los diferentes conjuntos de protocolos.

Al dividir el software de comunicación en capas, la pila de protocolos permite la división del trabajo, la facilidad de implementación y las pruebas de código, y la capacidad de desarrollar implementaciones de capas alternativas. Las capas se comunican con las de arriba y de abajo a través de interfaces concisas. En este sentido, una capa proporciona un servicio para la capa directamente encima de ella y hace uso de los servicios proporcionados por la capa directamente debajo de ella. Por ejemplo, la capa IP proporciona la capacidad de transferir datos de un host a otro sin ninguna garantía de entrega confiable o supresión de duplicados. Los protocolos de transporte como TCP hacen uso de este servicio para proporcionar a las aplicaciones una entrega de flujo de datos confiable y en orden.

La Figura 1-2 muestra cómo se modelan los protocolos TCP/IP en cuatro capas.

Aplicaciones

Transporte

Redes

Interfaz de red

y

Hardware

Aplicaciones

TCP/UDP

ICMP

IP

ARP/RARP

Interfaz de red

y Hardware

.......

.......

.......

.......

*Figura 1-2 La pila de protocolos TCP/IP: Cada capa representa un paquete de funciones* Estas capas incluyen:

|  |  |
| --- | --- |
| **Capa de aplicación** | La capa de aplicación es proporcionada por el programa que utiliza TCP/IP para la comunicación. Una aplicación es un proceso de usuario que coopera con otro proceso, generalmente en un host diferente (también hay un beneficio en la comunicación de la aplicación dentro de un solo host). Algunos ejemplos de aplicaciones son Telnet y el protocolo de transferencia de archivos (FTP). La interfaz entre las capas de aplicación y transporte está definida por los números de puerto y los sockets, que describimos con más detalle en 4.1, "Puertos y sockets" en la página 144. |
| **Capa de transporte** | La capa de transporte proporciona la transferencia de datos de extremo a extremo mediante la entrega de datos desde una aplicación a su par remoto. Se pueden admitir varias aplicaciones simultáneamente. El protocolo de capa de transporte más utilizado es el Protocolo de control de transmisión (TCP), que proporciona una entrega de datos fiable orientada a la conexión, supresión de datos duplicados, control de congestión y control de flujo. Discutimos esto con más detalle en 4.3, "Protocolo de control de transmisión (TCP)" en    Página 149. |

Otro protocolo de capa de transporte es el Protocolo de datagramas de usuario (consulte 4.2, "Protocolo de datagramas de usuario (UDP)" en la página 146). Proporciona un servicio sin conexión, poco confiable y de mejor esfuerzo. Como resultado, las aplicaciones que utilizan UDP como protocolo de transporte tienen que proporcionar su propia integridad, control de flujo y control de congestión de extremo a extremo, si lo desean. Por lo general, UDP es utilizado por aplicaciones que necesitan un mecanismo de transporte rápido y pueden tolerar la pérdida de algunos datos.

|  |  |
| --- | --- |
| **Capa de trabajo en red** | La capa entre redes, también llamada *capa de Internet* o *capa de red*, proporciona la imagen de "red virtual" de Internet (esta capa protege los niveles superiores de la arquitectura de red física que se encuentra debajo). El protocolo de Internet (IP) es el protocolo más importante de esta capa. Es un protocolo sin conexión que no asume la confiabilidad de las capas inferiores. IP *no* proporciona confiabilidad, control de flujo ni recuperación de errores. Estas funciones deben proporcionarse a un nivel superior.  IP proporciona una función de enrutamiento que intenta entregar los mensajes transmitidos a su destino. Analizamos la P.I. en detalle en el Capítulo 3, "Protocolos de interconexión", en la página 67. Una unidad de mensajes en una red IP se denomina *datagrama IP*. Esta es la unidad básica de información transmitida a través de redes TCP/IP. Otros protocolos de capa de red son IP, ICMP, IGMP, ARP y RARP. |
| **Capa de interfaz de red** La capa de interfaz de red, también llamada *capa de enlace* o *capa de enlace de datos*, es la interfaz con el hardware de red real. Esta interfaz puede o no proporcionar una entrega confiable, y puede estar orientada a paquetes o flujos. De hecho, TCP/IP no especifica ningún protocolo aquí, pero puede usar casi cualquier interfaz de red disponible, lo que ilustra la flexibilidad de la capa IP. Algunos ejemplos son IEEE 802.2, X.25 (que es fiable en sí mismo), ATM, FDDI e incluso SNA. Discutimos algunas redes e interfaces físicas en el Capítulo 2, "Interfaces de red" en la página 29. | |

Las especificaciones TCP/IP no describen ni estandarizan ningún protocolo de capa de red per se; solo estandarizan las formas de acceder a esos protocolos desde la capa entre redes.

En la Figura 1-3 se incluye un modelo de capas más detallado.

Aplicaciones

Transporte

Redes

Interfaz de red

y Hardware

SMTP, Telnet, FTP, Gopher...

UDP

TCP

IP

ICMP

ARPRARP

Ethernet, Token-Ring, FDDI, X.25, Inalámbrico, asíncrono, ATM,

SNA...

*Figura 1-3 Modelo arquitectónico detallado*

#### 1.1.3 Aplicaciones TCP/IP

Los protocolos de más alto nivel dentro de la pila de protocolos TCP/IP son los protocolos de aplicación. Se comunican con aplicaciones en otros hosts de Internet y son la interfaz visible para el usuario del conjunto de protocolos TCP/IP.

Todos los protocolos de aplicación tienen algunas características en común:

Pueden ser aplicaciones escritas por el usuario o aplicaciones estandarizadas y enviadas con el producto TCP/IP. De hecho, el conjunto de protocolos TCP/IP incluye protocolos de aplicación como:

* Telnet para el acceso de terminal interactivo a hosts de Internet remotos
* Protocolo de transferencia de archivos (FTP) para transferencias de archivos de disco a disco de alta velocidad
* Protocolo simple de transferencia de correo (SMTP) como sistema de correo por Internet

Estos son algunos de los protocolos de aplicación más implementados, pero existen muchos otros. Cada implementación particular de TCP/IP incluirá un conjunto menor o mayor de protocolos de aplicación.

Utilizan UDP o TCP como mecanismo de transporte. Recuerde que UDP no es confiable y no ofrece control de flujo, por lo que en este caso, la aplicación debe proporcionar su propia funcionalidad de recuperación de errores, control de flujo y control de congestión. A menudo es más fácil crear aplicaciones sobre TCP porque es un protocolo confiable, orientado a la conexión, amigable con la congestión y habilitado para el control de flujo. Como resultado, la mayoría de los protocolos de aplicación usarán TCP, pero hay aplicaciones basadas en UDP para lograr un mejor rendimiento a través de una mayor eficiencia del protocolo.

La mayoría de las aplicaciones utilizan el modelo de interacción cliente/servidor.

##### El modelo cliente/servidor

TCP es un protocolo peer-to-peer orientado a la conexión. No hay relaciones amo/subordinado. Sin embargo, las aplicaciones suelen utilizar un modelo cliente/servidor para las comunicaciones, como se muestra en la figura 1-4.

Un *servidor* es una aplicación que ofrece un servicio a los usuarios de Internet. Un *cliente* es un solicitante de un servicio. Una aplicación consta de una parte de servidor y otra de cliente, que pueden ejecutarse en el mismo sistema o en sistemas diferentes. Por lo general, los usuarios invocan la parte cliente de la aplicación, que crea una *solicitud* para un servicio determinado y la envía a la parte servidor de la aplicación mediante TCP/IP como vehículo de transporte.

El servidor es un programa que recibe una solicitud, realiza el servicio requerido y devuelve los resultados en una *respuesta*. Por lo general, un servidor puede tratar con múltiples solicitudes y múltiples clientes solicitantes al mismo tiempo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | |  | | |  | |  | | |  | |
|  | Cli  Un | ente |  |  |  | Cliente  B | |  | ..... |  | Servidor | |  |  |
|  | TC | P/IP |  | TCP/IP | |  | TCP/IP | |
|  |  |  |  |  |  |  |  |  |
|  |  | |  |  | |  |  | |
|  | | Red de Internet | | | | | | | |  | | | |

*Figura 1-4 El modelo cliente/servidor de las aplicaciones*

La mayoría de los servidores esperan las solicitudes en un  *puerto conocido* para que sus clientes sepan a qué puerto (y a su vez, a qué aplicación) deben dirigir sus solicitudes. Normalmente, el cliente utiliza un puerto arbitrario denominado *puerto efímero* para su comunicación. Los clientes que desean comunicarse con un servidor que no utiliza un puerto conocido deben tener otro mecanismo para saber a qué puerto deben dirigir sus solicitudes. Este mecanismo puede emplear un servicio de registro como portmap, que utiliza un puerto conocido. Para obtener información detallada sobre los protocolos de aplicación TCP/IP, consulte la Parte 2,

"Protocolos de aplicación TCP/IP" en la página 405.

##### Puentes, enrutadores y puertas de enlace

Hay muchas formas de proporcionar acceso a otras redes. En una interred, esto se hace con *enrutadores*. En esta sección, distinguimos entre un enrutador, un puente y una puerta de enlace para permitir el acceso remoto a la red:

|  |  |
| --- | --- |
| **Puente** | Interconecta segmentos LAN en el nivel de capa de interfaz de red y reenvía tramas entre ellos. Un puente realiza la función de un relé MAC y es independiente de cualquier protocolo de capa superior (incluido el protocolo de enlace lógico). Proporciona conversión del protocolo de capa MAC, si es necesario.  Se dice que un puente es *transparente* para la propiedad intelectual. Es decir, cuando un host IP envía un datagrama IP a otro host en una red conectada por un puente, envía el datagrama directamente al host y el datagrama "cruza" el puente sin que el host IP remitente sea consciente de ello. |
| **Enrutador** | Interconecta redes a nivel de capa de interconexión y  enruta paquetes entre ellos. El router debe comprender la estructura de direccionamiento asociada con los protocolos de red que admite y tomar decisiones sobre si reenviar paquetes o cómo. Los routers son capaces de seleccionar las mejores rutas de transmisión y los tamaños óptimos de los paquetes. La función básica de enrutamiento se implementa en la capa IP de la pila de protocolos TCP/IP, por lo que cualquier host o estación de trabajo que ejecute TCP/IP a través de más de una interfaz podría, en teoría y también con la mayoría de las implementaciones TCP/IP actuales, reenviar datagramas IP. Sin embargo, los routers dedicados proporcionan un enrutamiento mucho más sofisticado que las funciones mínimas implementadas por IP. |

Debido a que IP proporciona esta función básica de enrutamiento, el término

"Enrutador IP" se usa a menudo. Otros términos más antiguos para enrutador son "puerta de enlace IP", "puerta de enlace de Internet" y "puerta de enlace". El término *pasarela* se utiliza ahora normalmente para las conexiones en una capa superior a la capa entre redes.

Se dice que un router es *visible* para IP. Es decir, cuando un host envía un datagrama IP a otro host en una red conectada por un enrutador, envía el datagrama al enrutador para que pueda reenviarlo al host de destino.

**Entrada** Interconecta redes en capas más altas que puentes y enrutadores. Por lo general, una puerta de enlace admite la asignación de direcciones de una red a otra y también puede proporcionar la transformación de los datos entre los entornos para admitir la conectividad de aplicaciones de extremo a extremo. Las puertas de enlace suelen limitar la interconectividad de dos redes a un subconjunto de los protocolos de aplicación admitidos en cualquiera de ellas. Por ejemplo, un host de máquina virtual que ejecuta TCP/IP se puede usar como puerta de enlace de correo SMTP/RSCS.

**Nota:** El término "puerta de enlace", cuando se usa en este sentido, *no es*  sinónimo de "puerta de enlace IP".

Se dice que una puerta de enlace es *opaca* a IP. Es decir, un host no puede enviar un datagrama IP a través de una puerta de enlace; solo puede enviarlo *a* una puerta de enlace. La información de protocolo de nivel superior transportada por los datagramas es transmitida por la puerta de enlace utilizando cualquier arquitectura de red que se utilice en el otro lado de la puerta de enlace.

Estrechamente relacionado con los enrutadores y las puertas de enlace está el concepto de *firewall*, o *puerta de enlace de firewall*, que se utiliza para restringir el acceso desde Internet o alguna red que no sea de confianza a una red o grupo de redes controladas por una organización por razones de seguridad. Consulte 22.3, "Cortafuegos" en la página 794 para obtener más información sobre los cortafuegos.

### 1.2 Las raíces de Internet

Las redes se han convertido en una parte fundamental, si no la más importante, de los sistemas de información actuales. Forman la columna vertebral para el intercambio de información en empresas, grupos gubernamentales y grupos científicos. Esa información puede tomar varias formas. Pueden ser notas y documentos, datos para ser procesados por otra computadora, archivos enviados a colegas y flujos de datos multimedia.

A finales de la década de 1960 y en la década de 1970 se instalaron varias redes, cuando el diseño de redes era el tema "más avanzado" de la investigación informática y de los implementadores sofisticados. Dio lugar a múltiples modelos de red, como la conmutación de paquetes

redes de área local de detección de colisiones, redes jerárquicas y muchas otras tecnologías de comunicaciones excelentes.

El resultado de todo este gran know-how fue que cualquier grupo de usuarios pudo encontrar una red física y un modelo arquitectónico adecuado a sus necesidades específicas. Esto abarca desde líneas asíncronas de bajo costo sin otra recuperación de errores que una función de paridad bit por bit, pasando por redes de área amplia con todas las funciones (públicas o privadas) con protocolos confiables como redes públicas de conmutación de paquetes o redes SNA privadas, hasta redes de área local de alta velocidad pero distancia limitada.

La desventaja del desarrollo de conjuntos de protocolos tan heterogéneos es la situación bastante dolorosa en la que un grupo de usuarios quiere extender su sistema de información a otro grupo de usuarios que han implementado una tecnología de red diferente y protocolos de red diferentes. Como resultado, incluso si pudieran ponerse de acuerdo sobre alguna tecnología de red para interconectar físicamente los dos entornos, sus aplicaciones (como los sistemas de correo) aún no podrían comunicarse entre sí debido a los diferentes protocolos e interfaces de aplicación.

Esta situación fue reconocida a principios de la década de 1970 por un grupo de investigadores estadounidenses financiados por la Agencia de Proyectos de *Investigación Avanzada de Defensa* (DARPA). Su trabajo abordó el *trabajo en red,* o la interconexión de redes. Otras organizaciones oficiales se involucraron en esta área, como el UIT-T (anteriormente CCITT) y la ISO. El objetivo principal era definir un conjunto de protocolos, detallados en una suite bien definida, para que las aplicaciones pudieran comunicarse con otras aplicaciones, independientemente de la tecnología de red subyacente o de los sistemas operativos donde se ejecuten esas aplicaciones.

La organización oficial de estos investigadores fue el Grupo de Trabajo de la Red ARPANET, que tuvo su última reunión general en octubre de 1971. DARPA continuó su investigación para un conjunto de protocolos de interconexión, desde el primer protocolo de host a host del Programa de Control de Red (NCP) hasta el conjunto de protocolos TCP/IP, que tomó su forma actual alrededor de 1978. En ese momento, DARPA era bien conocida por ser pionera en la conmutación de paquetes a través de redes de radio y canales satelitales. Las primeras implementaciones reales de *Internet* se encontraron alrededor de 1980, cuando DARPA comenzó a convertir las máquinas de su red de investigación (ARPANET) para utilizar los nuevos protocolos TCP/IP. En 1983, la transición se completó y DARPA exigió que *todas las* computadoras dispuestas a conectarse a su ARPANET usaran TCP/IP.

DARPA también contrató a Bolt, Beranek y Newman (BBN) para desarrollar una implementación de los protocolos TCP/IP para Berkeley UNIX® en el VAX y financió a la Universidad de California en Berkeley para distribuir el código de forma gratuita con su sistema operativo UNIX. La primera versión de la clase *Distribución de software de Berkeley* (BSD) para incluir el conjunto de protocolos TCP/IP se puso a disposición en 1983 (4.2BSD). A partir de ese momento, TCP/IP se extendió rápidamente entre universidades y centros de investigación y se ha convertido en el subsistema de comunicaciones estándar para toda la conectividad UNIX. La segunda versión (4.3BSD) fue distribuido en 1986, con actualizaciones en 1988 (4.3BSD Tahoe) y 1990 (4.3BSD Reno). 4.4BSD fue lanzado en 1993. Debido a las limitaciones de financiación, 4.4BSD fue la última versión del BSD por el Grupo de Investigación de Sistemas Computacionales de la Universidad de California en Berkeley.

A medida que el trabajo en red TCP/IP se extendió rápidamente, se crearon nuevas redes de área amplia en los EE. UU. y se conectaron a ARPANET. A su vez, se agregaron al conjunto de redes interconectadas otras redes en el resto del mundo, no necesariamente basadas en los protocolos TCP/IP. El resultado es lo que se describe como *Internet*. En las siguientes secciones describimos algunos ejemplos de las diferentes redes que han desempeñado un papel clave en este desarrollo.

#### 1.2.1 ARPANET

A veces conocido como el "abuelo de las redes de paquetes", el ARPANET fue construido por DARPA (que se llamaba ARPA en ese momento) a fines de la década de 1960 para acomodar equipos de investigación sobre tecnología de conmutación de paquetes y para permitir el intercambio de recursos para los contratistas del Departamento de Defensa. La red interconectaba centros de investigación, algunas bases militares y ubicaciones gubernamentales. Pronto se hizo popular entre los investigadores por su colaboración a través del correo electrónico y otros servicios. A finales de 1975 se convirtió en una empresa de servicios de investigación dirigida por la Agencia de Comunicaciones de Defensa (DCA) y en 1983 se dividió en MILNET para la interconexión de emplazamientos militares y ARPANET para la interconexión de emplazamientos de investigación. Esto constituyó el comienzo de la Internet con "I mayúscula".

En 1974, ARPANET se basaba en líneas arrendadas de 56 Kbps que interconectaban *nodos de conmutación de paquetes* (PSN) dispersos por los Estados Unidos continentales y Europa occidental. Se trataba de miniordenadores que ejecutaban un protocolo conocido como *1822* (por el número de un informe que lo describía) y dedicadas a la tarea de conmutación de paquetes. Cada PSN tenía al menos dos conexiones a otras PSN (para permitir un enrutamiento alternativo en caso de falla del circuito) y hasta 22 puertos para conexiones de computadora de usuario (*host*). Estos sistemas 1822 ofrecían una entrega fiable y controlada por flujo de un paquete a un nodo de destino. Esta es la razón por la que el protocolo NCP original era un protocolo bastante simple. Fue reemplazado por los protocolos TCP/IP, que no asumen la confiabilidad del hardware de red subyacente y se pueden usar en redes distintas de 1822. Este protocolo 1822 no se convirtió en un estándar de la industria, por lo que DARPA decidió más tarde reemplazar la tecnología de conmutación de paquetes 1822 con el  *estándar CCITT X.25*.

El tráfico de datos superó rápidamente la capacidad de las líneas de 56 Kbps que componían la red, que ya no podían soportar el rendimiento necesario. Hoy en día, ARPANET ha sido reemplazado por las nuevas tecnologías en su papel de columna vertebral en el lado de la investigación de la Internet conectada (ver NSFNET más adelante en este capítulo), mientras que MILNET continúa formando la columna vertebral del lado militar.

#### 1.2.2 NSFNET

NSFNET, la Red de la Fundación Nacional de Ciencias (NSF, por sus siglas en inglés), es una red de tres niveles internetwork en los Estados Unidos que consiste en:

La red troncal: Una red que conecta redes de nivel medio administradas y operadas por separado y centros de supercomputación financiados por la NSF. La red troncal también tiene enlaces transcontinentales con otras redes como EBONE, la red troncal IP europea.

Redes de nivel medio: tres tipos de redes (redes regionales, basadas en disciplinas y redes de consorcios de supercomputadoras).

Redes de campus: Ya sean académicas o comerciales, conectadas a las redes de nivel medio.

A lo largo de los años, la NSF actualizó su red troncal para satisfacer las crecientes demandas de sus clientes:

Primera red troncal: Originalmente establecida por la NSF como una red de comunicaciones para que los investigadores y científicos accedieran a las supercomputadoras de la NSF, la primera red troncal de la NSFNET utilizó seis microcomputadoras DEC LSI/11 como conmutadores de paquetes, interconectadas por líneas arrendadas de 56 Kbps. En Carnegie Mellon existía una interconexión primaria entre la red troncal NSFNET y ARPANET, que permitía el enrutamiento de datagramas entre los usuarios conectados a cada una de esas redes.

Segunda red troncal: La necesidad de una nueva red troncal apareció en 1987, cuando la primera se sobrecargó a los pocos meses (el crecimiento estimado en ese momento era del 100% anual). La NSF y MERIT, Inc., un consorcio de redes informáticas de ocho universidades apoyadas por el estado en Michigan, acordaron desarrollar y administrar una nueva red troncal de mayor velocidad con mayores capacidades de transmisión y conmutación. Para gestionarlo, definieron los *Servicios de Información* (SI), que está compuesto por un Centro de Información y un Grupo de Soporte Técnico. El Centro de Información es responsable de la difusión de información, la gestión de recursos de información y la comunicación electrónica. El Grupo de Apoyo Técnico presta apoyo directamente sobre el terreno. El objetivo es proporcionar un sistema de información integrado con interfaces fáciles de usar y gestionar, accesibles desde cualquier punto de la red, con el apoyo de un conjunto completo de servicios de formación.

Merit y NSF llevaron a cabo este proyecto en asociación con IBM y MCI. IBM proporcionó el software, la conmutación de paquetes y el equipo de gestión de redes, mientras que MCI proporcionó las instalaciones de transporte de larga distancia. Instalada en 1988, la nueva red utilizaba inicialmente circuitos arrendados de 448 Kbps para interconectar 13 *sistemas de conmutación nodal* (NSS), suministrados por IBM. Cada NSS estaba compuesto por nueve sistemas IBM RISC (que ejecutaban una versión de IBM de 4.3BSD

UNIX) acoplado libremente por dos redes de anillo de tokens de IBM (para redundancia). Uno

El Intercambio de Red Digital Integrado (IDNX) suministrado por IBM se instaló en cada una de las 13 ubicaciones, para proporcionar:

* Enrutamiento alternativo dinámico
* Asignación dinámica de ancho de banda

Tercera red troncal: En 1989, la topología de los circuitos troncales NSFNET se reconfiguró después de las mediciones de tráfico y la velocidad de las líneas arrendadas aumentó a T1 (1.544 Mbps) utilizando principalmente fibra óptica.

Debido a la necesidad cada vez mayor de mejorar las capacidades de conmutación y transmisión de paquetes, se agregaron tres NSS a la red troncal y se actualizó la velocidad del enlace. La migración de la red troncal NSFNET de T1 a T3 (45 Mbps) se completó a finales de 1992. La migración posterior a niveles gigabit ya ha comenzado y continúa en la actualidad.

En abril de 1995, el gobierno de los Estados Unidos suspendió su financiación de NSFNET. Esto fue, en parte, una reacción al creciente uso comercial de la red. Casi al mismo tiempo, NSFNET migró gradualmente el tráfico principal de la red troncal en los EE. UU. a proveedores de servicios de red comerciales, y NSFNET volvió a ser una red para la comunidad de investigación. La red troncal principal ahora se ejecuta en cooperación con MCI y se conoce como vBNS (servicio de red troncal de muy alta velocidad).

NSFNET ha desempeñado un papel clave en el desarrollo de Internet. Sin embargo, muchas otras redes también han desempeñado su papel y también forman parte de Internet en la actualidad.

#### 1.2.3 Uso comercial de Internet

En los últimos años, Internet ha crecido en tamaño y alcance a un ritmo mayor de lo que nadie podría haber predicho. Una serie de factores clave han influido en este crecimiento. Algunos de los hitos más significativos han sido la distribución gratuita de Gopher en 1991, la primera publicación, también en 1991, de la especificación para hipertexto y, en 1993, el lanzamiento de Mosaic, el primer navegador basado en gráficos. Hoy en día, la gran mayoría de los hosts que ahora están conectados a Internet son de carácter comercial. Se trata de un área de conflicto potencial y real con los objetivos iniciales de Internet, que eran fomentar las comunicaciones abiertas entre las instituciones académicas y de investigación. Sin embargo, el crecimiento continuo en el uso comercial de Internet es inevitable, por lo que será útil explicar cómo se está produciendo esta evolución.

Una iniciativa importante a tener en cuenta es la de la Política de *Uso Aceptable* (PUA). La primera de estas políticas se introdujo en 1992 y se aplica al uso de NSFNET. En el corazón de esta AUP se encuentra el compromiso de "apoyar la investigación y la educación abiertas". Bajo "Usos inaceptables" hay una prohibición de "uso para actividades con fines de lucro", a menos que esté cubierto por el Principio General o como un uso específicamente aceptable. Sin embargo, a pesar de esta postura aparentemente restrictiva, la NSFNET se utilizó cada vez más para una amplia gama de actividades, incluidas muchas de carácter comercial, antes de volver a sus objetivos originales en 1995.

La provisión de una PUA es ahora común entre los proveedores de servicios de Internet, aunque la PUA ha evolucionado generalmente para ser más adecuada para uso comercial. Algunas redes todavía ofrecen servicios libres de cualquier PUA.

Centrémonos ahora en los proveedores de servicios de Internet que han sido más activos en la introducción de usos comerciales en Internet. Dos que vale la pena mencionar son PSINet y UUNET, que comenzaron a fines de la década de 1980 para ofrecer acceso a Internet tanto a empresas como a individuos. El CERFnet, con sede en California, proporcionó servicios libres de cualquier PUA. Poco después se formó una organización para interconectar PSINet, UUNET y CERFnet, llamada Intercambio Comercial de Internet (CIX), basada en el entendimiento de que el tráfico de cualquier miembro de una red puede fluir sin restricciones sobre las redes de los otros miembros. A partir de julio de 1997, CIX había crecido a más de 146 miembros de todo el mundo, conectando las redes de los miembros. Casi al mismo tiempo que se formó CIX, IBM, MCI y Merit, Inc. formaron una empresa sin fines de lucro, Advance Network and Services (ANS), para operar conexiones troncales T1 (posteriormente T3) para NSFNET. Este grupo se dedicó activamente a aumentar la presencia comercial en Internet.

ANS formó una subsidiaria orientada comercialmente llamada ANS CO+RE para proporcionar vínculos entre los clientes comerciales y los dominios de investigación y educación. ANS CO+RE proporciona acceso a NSFNET, además de estar vinculado a CIX. En 1995 ANS fue adquirida por America Online.

En 1995, cuando la NSFNET estaba volviendo a su función académica anterior, la arquitectura de Internet cambió de tener una sola red troncal dominante en los EE.UU. a tener una serie de redes troncales operadas comercialmente. Para que las diferentes redes troncales puedan intercambiar datos, la NSF estableció cuatro puntos de acceso a la red (NAP) para que sirvieran como puntos de intercambio de datos entre los proveedores de servicios de la red troncal.

Otro tipo de intercambio es el Ethernet del Área Metropolitana (MAE). Varios MAE han sido establecidos por Metropolitan Fiber Systems (MFS), que también tiene su propia red troncal. Los NAP y MAE también se denominan puntos públicos de intercambio (IXP). Los proveedores de servicios de Internet (ISP) suelen tener conexiones a varios IXP para el rendimiento y la copia de seguridad. Para obtener una lista actualizada de IXP, consulte el Punto de Intercambio en:

[http://www.ep.net](http://www.ep.net/)

Al igual que CIX en los Estados Unidos, los proveedores europeos de Internet formaron el RIPE

(Réseaux IP Européens) para garantizar la coordinación técnica y administrativa. RIPE se creó en 1989 para proporcionar un servicio IP uniforme a los usuarios de toda Europa. Hoy en día, las redes troncales de Internet más grandes funcionan a OC48 (2,4 Gbps) u OC192 (9,6 Gbps).

#### 1.2.4 Internet2

El éxito de Internet y la consiguiente congestión frecuente de la NSFNET y su sustitución comercial provocaron cierta frustración entre la comunidad investigadora que anteriormente había disfrutado del uso exclusivo de Internet. Por lo tanto, la comunidad universitaria, junto con socios gubernamentales y de la industria, y alentada por el componente de financiación de la iniciativa Next Generation Internet (NGI), han formado el  *proyecto Internet2*.

La iniciativa NGI es un programa federal de investigación que está desarrollando tecnologías de redes avanzadas, introduciendo aplicaciones revolucionarias que requieren tecnologías de redes avanzadas y demostrando estas capacidades tecnológicas en bancos de pruebas de alta velocidad.

##### Misión

La misión de Internet2 es facilitar y coordinar el desarrollo, la operación y la transferencia de tecnología de aplicaciones avanzadas basadas en redes y servicios de red para promover el liderazgo de los Estados Unidos en investigación y educación superior y acelerar la disponibilidad de nuevos servicios y aplicaciones en Internet.

Internet2 tiene los siguientes objetivos:

Demostrar nuevas aplicaciones que pueden mejorar drásticamente la capacidad de los investigadores para colaborar y realizar experimentos.

Demostrar una mejor prestación de servicios educativos y de otro tipo (por ejemplo, atención sanitaria, vigilancia del medio ambiente, etc.) aprovechando la *proximidad virtual* creada por una infraestructura de comunicaciones avanzada.

Apoye el desarrollo y la adopción de aplicaciones avanzadas proporcionando middleware y herramientas de desarrollo.

Facilitar el desarrollo, la implementación y la operación de una infraestructura de comunicaciones asequible, capaz de soportar una calidad de servicio (QoS) diferenciada basada en los requisitos de aplicación de la comunidad investigadora y educativa.

Promover la experimentación con la próxima generación de tecnologías de la comunicación.

Coordinar la adopción de normas de trabajo acordadas y prácticas comunes entre las instituciones participantes para garantizar la calidad del servicio y la interoperabilidad de extremo a extremo.

Catalizar las alianzas con organizaciones gubernamentales y del sector privado.

Fomentar la transferencia de tecnología de Internet2 al resto de Internet.

Estudiar el impacto de las nuevas infraestructuras, servicios y aplicaciones en la educación superior y en la comunidad de Internet en general.

##### Participantes de Internet2

Internet2 cuenta con 180 universidades participantes en todo Estados Unidos. Las organizaciones afiliadas proporcionan al proyecto valiosos aportes. Todos los participantes en el proyecto Internet2 son miembros de la Corporación Universitaria para el Desarrollo Avanzado de Internet (UCAID).

En la mayoría de los aspectos, los acuerdos de asociación y financiación para Internet2 serán paralelos a los de los anteriores esfuerzos conjuntos de creación de redes entre el mundo académico y el gobierno, de los cuales el proyecto NSFnet es un ejemplo muy exitoso. El gobierno de los Estados Unidos participará en Internet2 a través de la iniciativa NGI y programas relacionados.

Internet2 también se une a los líderes corporativos para crear los servicios de red avanzados necesarios para cumplir con los requisitos de las aplicaciones de banda ancha en red. Los socios de la industria trabajan principalmente con equipos universitarios regionales y de campus para proporcionar los servicios y productos necesarios para implementar las aplicaciones desarrolladas por el proyecto. Entre las principales empresas que participan actualmente en Internet2 se encuentran Alcatel, Cisco Systems, IBM, Nortel Networks, Sprint y Sun Microsystems™. El apoyo adicional para Internet2 proviene de la colaboración con organizaciones sin fines de lucro que trabajan en investigación y redes educativas. Entre las organizaciones afiliadas comprometidas con el proyecto se encuentran MCNC, Merit, los Institutos Nacionales de Salud (NIH) y el Sistema de Universidades Estatales de Florida.

Para obtener más información sobre Internet2, consulte su página web en: [http://www.internet2.edu](http://www.internet2.edu/)

#### 1.2.5 El modelo de referencia de interconexión de sistemas abiertos (OSI)

El Modelo de Referencia OSI (Open Systems Interconnect) (ISO 7498) define un modelo de siete capas de comunicación de datos con transporte físico en la parte inferior capa y protocolos de aplicación en las capas superiores. Este modelo, que se muestra en la Figura 1-5, es ampliamente aceptado como base para la comprensión de cómo debe funcionar una pila de protocolos de red y como una herramienta de referencia para comparar la implementación de la pila de red.

Aplicación

Presentación

Sesión

Transporte

Red

Enlace de datos

Físico

Aplicación

Presentación

Sesión

Transporte

Red

Enlace de datos

Físico

*Figura 1-5 El modelo de referencia OSI*

Cada capa proporciona un conjunto de funciones a la capa superior y, a su vez, se basa en las funciones proporcionadas por la capa inferior. Aunque los mensajes solo pueden pasar verticalmente a través de la pila de una capa a otra, desde un punto de vista lógico, cada capa se comunica directamente con su capa del mismo nivel en otros nodos.

Las siete capas son:

|  |  |
| --- | --- |
| **Aplicación** | Aplicaciones de red, como emulación de terminales y transferencia de archivos |
| **Presentación** | Formateo de datos y encriptación |
| **Sesión** | Establecimiento y mantenimiento de sesiones |
| **Transporte** | Suministro de servicios de extremo a extremo fiables y no fiables |
| **Red** | Entrega de paquetes, incluido el enrutamiento |
| **Enlace de datos**  **Físico** | Encuadre de unidades de información y comprobación de errores  Transmisión de bits en el hardware físico |

A diferencia del PCT/IP, el enfoque de la OSI partió de un borrón y cuenta nueva y definió las normas, adhiriéndose estrictamente a su propio modelo, utilizando un proceso formal de comité sin necesidad de implementaciones. Los protocolos de Internet utilizan un enfoque de ingeniería menos formal, en el que cualquiera puede proponer y comentar sobre la solicitud de comentarios, conocida como RFC, y se requieren implementaciones para verificar la viabilidad. Los protocolos OSI se desarrollaron lentamente, y debido a que la ejecución de la pila de protocolos completa requiere muchos recursos, no se han implementado ampliamente, especialmente en el mercado de computadoras de escritorio y pequeñas. Mientras tanto, TCP/IP e Internet se estaban desarrollando rápidamente, y el despliegue se producía a un ritmo muy elevado.

### 1.3 Estándares TCP/IP

TCP/IP ha sido popular entre los desarrolladores y los usuarios por igual debido a su apertura inherente y renovación perpetua. Lo mismo ocurre con Internet como red de comunicaciones abierta. Sin embargo, esta apertura podría convertirse fácilmente en algo que puede ayudarte y perjudicarte si no se controla de alguna manera. A pesar de que no existe un órgano de gobierno general que emita directivas y regulaciones para Internet (el control se basa principalmente en la cooperación mutua), la Internet Society (ISOC) sirve como organismo de normalización para la comunidad de Internet. Está organizado y gestionado por el Consejo de Arquitectura de Internet (IAB).

El propio IAB confía en el Grupo de Trabajo de Ingeniería de Internet (IETF) para emitir nuevos estándares, y en la Autoridad de Números Asignados de Internet (IANA) para coordinar los valores compartidos entre múltiples protocolos. El editor de RFC es responsable de revisar y publicar nuevos documentos de estándares.

El propio IETF está gobernado por el Grupo Directivo de Ingeniería de Internet (IESG) y se organiza a su vez en forma de Áreas y Grupos de Trabajo donde se discuten nuevas especificaciones y se proponen nuevos estándares.

El Proceso de Estándares de Internet, descrito en RFC 2026, El Proceso de Estándares de Internet, Revisión 3, se refiere a todos los protocolos, procedimientos y convenciones que se utilizan en o por Internet, ya sea que formen parte o no del conjunto de protocolos TCP/IP.

Los objetivos generales del Proceso de Normalización de Internet son:

Excelencia técnica

Implementación y pruebas previas Documentación clara, concisa y fácil de entender

Apertura y equidad

Puntualidad

El proceso de normalización se resume de la siguiente manera:

Con el fin de que una nueva especificación sea aprobada como norma, los solicitantes tienen que presentar esa especificación al IESG, donde se discutirá y revisará su mérito técnico y su viabilidad, y también se publicará como un borrador de documento en Internet. Esto no debería tomar menos de dos semanas ni más de seis meses.

Una vez que el IESG llega a una conclusión positiva, emite una notificación de última llamada para permitir que la especificación sea revisada por toda la comunidad de Internet.

Después de la aprobación final por parte del IESG, se recomienda un borrador de Internet al Grupo de Trabajo de Ingeniería de Internet (IETF), otra subsidiaria del IAB, para su inclusión en el Standards Track y para su publicación como una Solicitud de Comentarios (ver 1.3.1, "Solicitud de Comentarios (RFC)" en la página 22).

Una vez publicada como RFC, una contribución puede avanzar en el estado descrito en 1.3.2, "Estándares de Internet" en la página 24. También puede revisarse con el tiempo o eliminarse gradualmente cuando se encuentren mejores soluciones.

Si el IESG no aprueba una nueva especificación después de seis meses de su presentación, o si un documento ha permanecido sin cambios dentro de los seis meses posteriores a su presentación, se eliminará del directorio de borradores de Internet.

#### 1.3.1 Solicitud de comentarios (RFC)

El conjunto de protocolos de Internet sigue evolucionando a través del mecanismo de solicitud de *comentarios* (RFC). Los investigadores están diseñando e implementando nuevos protocolos (en su mayoría protocolos de aplicación), y se presentan a la atención de la comunidad de Internet en forma de un borrador de Internet (ID).[[1]](#footnote-1) La mayor fuente de identificadores es el Grupo de Trabajo de Ingeniería de Internet (IETF), que es una subsidiaria de la IAB. Sin embargo, cualquiera puede enviar un memorándum propuesto como identificación al Editor de RFC. Hay un conjunto de reglas que los autores de RFC/ID deben seguir para que se acepte una RFC. Estas reglas se describen en un RFC (RFC 2223), que también indica cómo presentar una propuesta para un RFC.

Una vez que se ha publicado una RFC, todas las revisiones y reemplazos se publican como nuevas RFC. Se dice que una nueva RFC que revisa o reemplaza una RFC existente

"actualizar" o "dejar obsoleto" ese RFC. Se dice que el RFC existente está "actualizado por" o

"obsoleto por" el nuevo. Por ejemplo, RFC 1542, que describe el protocolo BOOTP, es una "segunda edición", que es una revisión de RFC 1532 y una enmienda a RFC 951. Por lo tanto, RFC 1542 se etiqueta así: "RFC 1532 obsoleto; Actualiza RFC 951". En consecuencia, nunca hay confusión sobre si dos personas se refieren a versiones diferentes de un RFC, porque nunca hay más de una versión actual.

Algunos RFC se describen como *documentos de información*, mientras que otros describen

Protocolos de Internet. El Consejo de Arquitectura de Internet (IAB, por sus siglas en inglés) mantiene una lista de las RFC que describen el conjunto de protocolos. A cada uno de ellos se le asigna un *estado* y un *estado*.

Un protocolo de Internet puede tener uno de los siguientes estados:

|  |  |
| --- | --- |
| **Estándar** | La IAB lo ha establecido como un protocolo oficial para Internet. Estos se dividen en dos grupos:  Protocolo IP y superior, protocolos que se aplican a todo Internet  Protocolos específicos de la red, generalmente especificaciones de cómo hacer IP en tipos particulares de redes |
| **Proyecto de norma** | La IAB está considerando activamente este protocolo como un posible protocolo estándar. Se desea que se realicen pruebas y comentarios sustanciales y generalizados. Envíe comentarios y resultados de pruebas al IAB. Existe la posibilidad de que se introduzcan cambios en un proyecto de protocolo antes de que se convierta en norma. |
| **Norma propuesta** | Estas son propuestas de protocolo que podrían ser consideradas por el IAB para su estandarización en el futuro. Es deseable que varios grupos implementen y prueben. Es probable que se revise el protocolo. |
| **Experimental** | Un sistema no debe implementar un protocolo experimental a menos que esté participando en el experimento y haya coordinado su uso del protocolo con el desarrollador del protocolo. |
| **Informativo** | Los protocolos desarrollados por otras organizaciones de estándares, o proveedores, o que por otras razones están fuera del ámbito de la IAB pueden publicarse como RFC para la conveniencia de la comunidad de Internet como protocolos informativos. En algunos casos, la IAB también podría recomendar el uso de estos protocolos en Internet. |
| **Histórico** | Estos son protocolos que es poco probable que alguna vez se conviertan en estándares en Internet, ya sea porque lo han sido.  reemplazado por desarrollos posteriores o por falta de interés. |

El estado del protocolo puede ser cualquiera de los siguientes:

|  |  |
| --- | --- |
| **Obligatorio** | Un sistema debe implementar los protocolos requeridos. |
| **Recomendado** | Un sistema debe implementar el protocolo recomendado. |
| **Electivo** | Un sistema puede o no implementar un protocolo electivo. La noción general es que si vas a hacer algo como esto, debes hacer exactamente esto. |
| **Uso limitado** | Estos protocolos son para uso en circunstancias limitadas. Esto puede deberse a su estado experimental, naturaleza especializada, funcionalidad limitada o estado histórico. |
| **No se recomienda** | Estos protocolos no se recomiendan para uso general. Esto puede deberse a su funcionalidad limitada, naturaleza especializada o estado experimental o histórico. |

#### 1.3.2 Estándares de Internet

La norma propuesta, el proyecto de norma y los protocolos normalizados se describen como parte de la vía de *normalización de Internet*. Cuando un protocolo alcanza el estado estándar, se le asigna un número estándar (STD). El propósito de los números STD es indicar claramente qué RFC describen los estándares de Internet. Los números STD hacen referencia a varias RFC cuando la especificación de una norma se extiende a través de varios documentos. A diferencia de las RFC, en las que el número se refiere a un documento específico, los números STD no cambian cuando se actualiza una norma. Sin embargo, los números STD no tienen números de versión porque todas las actualizaciones se realizan a través de RFC y los números RFC son únicos. Por lo tanto, para especificar claramente a qué versión de un estándar se refiere, se debe indicar el número estándar y todos los RFC que incluye. Por ejemplo, el Sistema de Nombres de Dominio (DNS) es STD 13 y se describe en las RFC 1034 y 1035. Para hacer referencia a la norma, se debe utilizar un formulario como "STD-13/RFC1034/RFC1035".

En el caso de algunas RFC de seguimiento de estándares, la categoría de estado no siempre contiene suficiente información para ser útil. Por lo tanto, se complementa, especialmente en el caso de los protocolos de encaminamiento, con una *declaración de aplicabilidad*, que se da en la STD 1 o en una RFC separada.

A lo largo de este libro se harán referencias a los RFC y a los números STD, ya que forman la base de todas las implementaciones de protocolos TCP/IP.

Los siguientes estándares de Internet son de particular importancia:

STD 1 – Estándares de Protocolo Oficial de Internet

Esta norma proporciona el estado y el estado de cada protocolo o estándar de Internet y define los significados atribuidos a cada estado o estado. Es emitido por el IAB aproximadamente trimestralmente. En el momento de escribir este artículo, este estándar se encuentra en RFC 3700.

STD 2 – Números de Internet asignados

Este estándar enumera los números asignados actualmente y otros parámetros de protocolo en el conjunto de protocolos de Internet. Es emitido por la Autoridad de Números Asignados de Internet (IANA). La edición actual en el momento de escribir este artículo es RFC 3232.

STD 3 – Requisitos del anfitrión

Esta norma define los requisitos para el software de alojamiento de Internet (a menudo por referencia a las RFC relevantes). La norma consta de tres partes:

* RFC 1122 – Requisitos para hosts de Internet – capa de comunicaciones
* RFC 1123 - Requisitos para hosts de Internet - aplicación y soporte
* RFC 2181 – Aclaraciones a la especificación DNS

STD 4 – Requisitos del router

Esta norma define los requisitos para el software de puerta de enlace a Internet IPv4 (router). Se define en RFC 1812 - Requisitos para enrutadores IPv4.

##### Para su información (FYI)

Una serie de RFC que pretenden ser de amplio interés para los usuarios de Internet se clasifican como  *documentos For Your Information* (FYI). Con frecuencia contienen información introductoria u otra información útil. Al igual que los números STD, un número FYI no se cambia cuando se emite un RFC revisado. A diferencia de los STD, los FYI corresponden a un único documento RFC. Por ejemplo, FYI 4 - FYI on Questions and Answers - Answers to Frequently Asked "New Internet User Questions" (Preguntas y respuestas sobre preguntas frecuentes sobre los "nuevos usuarios de Internet"), se encuentra actualmente en su quinta edición. Los números RFC son 1177, 1206, 1325 y 1594, y 2664.

##### Obtención de RFC

Los documentos de RFC e identificación están disponibles públicamente y en línea, y se pueden obtener mejor en el sitio web del IETF: [http://www.ietf.org](http://www.ietf.org/)

Puede encontrar una lista completa de los estándares de Internet actuales en RFC 3700 - Estándares de protocolo oficial de Internet.

### 1.4 El futuro de Internet

Tratar de predecir el futuro de Internet no es una tarea fácil. Pocos habrían imaginado, incluso hace cinco años, hasta qué punto Internet se ha convertido ahora en una parte de la vida cotidiana en los negocios, los hogares y las escuelas. Sin embargo, hay una serie de cosas de las que podemos estar bastante seguros.

#### 1.4.1 Aplicaciones multimedia

Los requisitos de ancho de banda seguirán aumentando a tasas masivas; no solo el número de usuarios de Internet está creciendo rápidamente, sino que las aplicaciones que se utilizan son cada vez más avanzadas y, por lo tanto, consumen más ancho de banda. Están surgiendo nuevas tecnologías, como la multiplexación por división de onda densa (DWDM), para satisfacer estas demandas de gran ancho de banda que se imponen en Internet.

Gran parte de esta creciente demanda puede atribuirse al mayor uso de aplicaciones multimedia. Un ejemplo es el de la tecnología de voz sobre IP. A medida que esta tecnología madure, es casi seguro que veremos un intercambio de ancho de banda entre voz y datos a través de Internet. Esto plantea algunas preguntas interesantes para las compañías telefónicas. El costo para un usuario de una conexión a Internet entre Raleigh, NC y Santiago, Chile es el mismo que el de una conexión dentro de Raleigh, no así para una conexión telefónica tradicional. Inevitablemente, las conversaciones de voz se convertirán en conversaciones de video a medida que las llamadas telefónicas se conviertan en videoconferencias.

Hoy en día, es posible escuchar emisoras de radio de casi cualquier parte del mundo a través de Internet con calidad FM. Podemos ver canales de televisión de todo el mundo, lo que lleva al claro potencial de utilizar Internet como vehículo para entregar películas y todo tipo de señales de video a los consumidores de todo el mundo. Sin embargo, todo tiene un precio, ya que la infraestructura de Internet debe adaptarse a demandas de ancho de banda tan altas.

#### 1.4.2 Uso comercial

Internet ha experimentado una explosión en términos de uso comercial. Hoy en día, casi todas las grandes empresas dependen de Internet, ya sea para el marketing, las ventas, el servicio al cliente o el acceso de los empleados. Se espera que estas tendencias continúen. Las tiendas electrónicas continuarán floreciendo al brindar comodidad a los clientes que no tienen tiempo para dirigirse a las tiendas tradicionales.

Las empresas dependerán cada vez más de Internet como medio para comunicarse con las sucursales de todo el mundo. Con la popularidad de las redes privadas virtuales (VPN), las empresas pueden realizar de forma segura sus negocios internos en un área amplia utilizando Internet; Los empleados pueden trabajar desde casa, lo que genera un entorno de *oficina virtual* . Es probable que las reuniones virtuales sean algo habitual.

#### 1.4.3 Internet inalámbrico

Sin embargo, quizás el crecimiento más extendido en el uso de Internet es el de las aplicaciones inalámbricas. Recientemente, ha habido un enfoque increíble en la habilitación de la computación inalámbrica y omnipresente. Este enfoque ha sido motivado en gran medida por la conveniencia de la conectividad inalámbrica. Por ejemplo, no es práctico conectar físicamente una estación de trabajo móvil, que, por definición, es libre de itinerancia. Restringir una estación de trabajo de este tipo a una geografía física simplemente frustra el propósito. En otros casos, la conectividad por cable simplemente no es factible. Algunos ejemplos son las ruinas de Macchu Picchu o las oficinas de la Capilla Sixtina. En estas circunstancias, las estaciones de trabajo fijas también se benefician de un acceso a la red que de otro modo no estaría disponible.

Protocolos como Bluetooth, IEEE 802.11 y Wireless Application Protocol (WAP) están allanando el camino hacia una Internet inalámbrica. Si bien los beneficios personales de dicho acceso son bastante ventajosos, aún más atractivas son las aplicaciones comerciales que se ven facilitadas por dicha tecnología. Todas las empresas, desde las fábricas hasta los hospitales, podrían mejorar sus respectivos servicios. Los dispositivos inalámbricos se convertirán en equipo estándar en los vehículos, no solo para el disfrute personal del conductor, sino también para el flujo de información de mantenimiento a su mecánico de automóviles favorito. Las aplicaciones son ilimitadas.

### 1.5 RFC relevantes para este capítulo

Las siguientes RFC proporcionan información detallada sobre los protocolos y arquitecturas de conexión presentados a lo largo de este capítulo:

[RFC 2026 – El Proceso de Estándares de Internet -- Revisión 3 (octubre de 1996)](ftp://ftp.rfc-editor.org/in-notes/rfc2026.txt)

[RFC 2223 – Instrucciones para RFC (octubre de 1997)](ftp://ftp.rfc-editor.org/in-notes/rfc2223.txt)

[RFC 2900 – Estándares de Protocolo Oficial de Internet (agosto de 2001)](ftp://ftp.rfc-editor.org/in-notes/rfc2900.txt)

[RFC 3232 - Números asignados: RFC 1700 se reemplaza por un en línea](ftp://ftp.rfc-editor.org/in-notes/rfc3232.txt)

[Base de datos (enero de 2002)](ftp://ftp.rfc-editor.org/in-notes/rfc3232.txt)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **3** | |

## Capítulo 3. Protocolos de interconexión

Este capítulo proporciona una visión general de los protocolos más importantes y comunes asociados con la capa de interconexión entre redes TCP/IP. Entre ellas se encuentran:

Protocolo de Internet (IP)

Protocolo de mensajes de control de Internet (ICMP)

Protocolo de resolución de direcciones (ARP)

Protocolo de configuración dinámica de host (DHCP)

Estos protocolos realizan el direccionamiento, el enrutamiento y la entrega de datagramas, la configuración dinámica de direcciones y la resolución entre las direcciones de la capa entre redes y las direcciones de la capa de interfaz de red.

© Derechos de autor IBM Corp. 1989-2006. Todos los derechos reservados.

### 3.1 Protocolo de Internet (IP )

IP es un protocolo estándar con número STD 5. La norma también incluye ICMP

(consulte 3.2, "Protocolo de mensajes de control de Internet (ICMP)" en la página 109) e IGMP (consulte 3.3, "Protocolo de administración de grupos de Internet (IGMP)" en la página 119). IP tiene un estado de obligatorio.

La especificación IP actual se encuentra en RFC 950, RFC 919, RFC 922, RFC 3260 y RFC 3168, que actualiza RFC 2474, y RFC 1349, que actualiza RFC 791. Consulte 3.8, "RFC relevantes para este capítulo" en la página 140 para obtener más detalles sobre los RFC.

IP es el protocolo que oculta la red física subyacente mediante la creación de una vista de *red virtual* . Es un protocolo de entrega de paquetes poco confiable, de mejor esfuerzo y sin conexión. Tenga en cuenta que el mejor esfuerzo significa que los paquetes enviados por IP pueden perderse, llegar desordenados o incluso duplicarse. IP asume que los protocolos de capa superior abordarán estas anomalías.

Una de las razones para utilizar un protocolo de red sin conexión era minimizar la dependencia de centros informáticos específicos que utilizaban redes jerárquicas orientadas a la conexión. El Departamento de Defensa de los Estados Unidos tenía la intención de desplegar una red que seguiría funcionando si se destruyeban partes del país. Se ha demostrado que esto es cierto para Internet.

#### 3.1.1 Direccionamiento IP

Las direcciones IP se representan mediante un valor binario sin signo de 32 bits. Por lo general, se expresa en un formato decimal con puntos. Por ejemplo, 9.167.5.8 es una dirección IP válida. La forma numérica es utilizada por el software IP. La asignación entre la dirección IP y un nombre simbólico más fácil de leer, por ejemplo, myhost.ibm.com, se realiza mediante el *Sistema de nombres de dominio (DNS),* que se describe en 12.1, "Sistema de nombres de dominio (DNS)" en la página 426.

##### La dirección IP

Los estándares de direccionamiento IP se describen en RFC 1166. Para identificar un host en el

Internet, a cada host se le asigna una dirección, la *dirección IP* o, en algunos casos, la dirección de *Internet*. Cuando el host está conectado a más de una red, se denomina *multihost* y tiene una dirección IP para cada interfaz de red. La dirección IP consta de un par de números:

Dirección IP = <número de red><número de host>

La parte del número de red de la dirección IP es administrada por uno de los tres Registros Regionales de Internet (RIR):

Registro Americano de Números de Internet (ARIN): Este registro es responsable de la administración y registro de los números de Protocolo de Internet (IP) para América del Norte, América del Sur, el Caribe y África subsahariana.

Reseaux IP Europeans (RIPE): Este registro es responsable de la administración y el registro de los números de Protocolo de Internet (IP) para Europa, Oriente Medio y partes de África.

Centro de Información de Redes de Asia Pacífico (APNIC): Este registro es responsable de la administración y el registro de los números de Protocolo de Internet (IP) dentro de la región de Asia Pacífico.

Las direcciones IP son números de 32 bits representados en forma decimal *con puntos* (como la representación decimal de cuatro valores de 8 bits concatenados con puntos). Por ejemplo, 128.2.7.9 es una dirección IP, siendo 128.2 el número de red y 7.9 el número de host. A continuación, explicamos las reglas utilizadas para dividir una dirección IP en sus partes de red y host.

El formato binario de la dirección IP 128.2.7.9 es:

10000000 00000010 00000111 00001001

Las direcciones IP son utilizadas por el protocolo IP para identificar de forma única un host en Internet (o, más generalmente, en cualquier Internet). En sentido estricto, una dirección IP identifica una interfaz que es capaz de enviar y recibir datagramas IP. Un sistema puede tener varias interfaces de este tipo. Sin embargo, tanto los hosts como los enrutadores deben tener al menos una dirección IP, por lo que esta definición simplificada es aceptable. Los datagramas IP (los paquetes de datos básicos intercambiados entre hosts) son transmitidos por una red física conectada al host. Cada datagrama IP contiene una *dirección IP de origen* y una *dirección IP de destino*. Para enviar un datagrama a un destino IP determinado, la dirección IP de destino debe traducirse o asignarse a una dirección física. Esto puede requerir transmisiones en la red para obtener la dirección de red física del destino. (Por ejemplo, en las LAN, el Protocolo de Resolución de Direcciones, descrito en 3.4, "Protocolo de Resolución de Direcciones (ARP)" en la página 119, se utiliza para traducir direcciones IP a direcciones MAC físicas).

##### Direcciones IP basadas en clases

Los primeros bits de la dirección IP especifican cómo se debe separar el resto de la dirección en su parte de red y host. Los términos *Dirección de red* y *netID* a veces se utilizan en lugar del número de red, pero el término formal, utilizado en RFC 1166, es el número de red. Del mismo modo, los términos *Dirección de host* y *ID de host* a veces se utilizan en lugar del número de host.

Hay cinco clases de direcciones IP. Se muestran en la Figura 3-1.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Clase A  Clase B  Clase C  Clase D  Clase E | 1 2 3  01 8 6 4 1 | | | | | | | | | | |
|  | |  | | | | |  | | |  |
| 0 |  | netID | | | | | ID de host | | |  |
|  |  | | | | | | | | |
|  | | | | | | | | | |
|  | | |  | | | | |  | |  |
|  | 10 | | netID | | | | | ID de host | |  |
|  |  | | | | | | | | |
|  | | | | | | | | | |
|  | | | |  | | | | |  |  |
|  | 110 | | | netID | | | | | ID de host |  |
|  |  | | | | | | | | |
|  | | | | | | | | | |
|  | | | | |  | | | | |  |
|  | 1110 | | | | multidifusión | | | | |  |
|  |  | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | |  | | | |  |
|  | 11110 | | | | | Uso futuro/experimental | | | |  |
|  |  | | | | | | | | |

*Figura 3-1 IP: Clases asignadas de direcciones IP*

Dónde:

|  |  |
| --- | --- |
| **Direcciones de clase A** | Estas direcciones utilizan 7 bits para la <red> y 24 bits para la parte <host> de la dirección IP. Esto permite 27-2 (126) redes, cada una con 224-2 (16777214) hosts, un total de más de 2.000 millones de direcciones. |
| **Direcciones de clase B** | Estas direcciones utilizan 14 bits para la <red> y 16 bits para la parte <host> de la dirección IP. Esto permite 214-2 (16382) redes, cada una con 216-2 (65534) hosts, un total de más de 1.000 millones de direcciones. |
| **Direcciones de clase C** | Estas direcciones utilizan 21 bits para la <red> y 8 bits para la parte <host> de la dirección IP. Eso permite 221-2 (2097150) redes, cada una con 28-2 (254) hosts, un total de más de 500 millones de direcciones. |
| **Direcciones de clase D** | Estas direcciones están reservadas para la multidifusión (una especie de |
| **Direcciones de clase E** | radiodifusión, pero en un área limitada, y solo a los presentadores utilizando la misma dirección de Clase D).  Estas direcciones están reservadas para uso futuro o experimental. |

Una dirección de clase A es adecuada para redes con un número extremadamente grande de hosts. Las direcciones de clase C son adecuadas para redes con un pequeño número de hosts. Esto significa que las redes de tamaño medio (aquellas con más de 254 hosts o donde hay una expectativa de más de 254 hosts) deben usar direcciones de Clase B. Sin embargo, el número de redes pequeñas y medianas ha crecido muy rápidamente. Se temía que si se hubiera permitido que este crecimiento continuara sin disminuir, todas las direcciones de red de Clase B disponibles se habrían utilizado a mediados de la década de 1990. Esto se denominó el problema de agotamiento de la dirección IP (consulte 3.1.5, "El problema de agotamiento de la dirección IP" en la página 86).

La división de una dirección IP en dos partes también separa la responsabilidad de seleccionar la dirección IP completa. La parte del número de red de la dirección es asignada por los RIR. La parte del número de host es asignada por la autoridad que controla la red. Como se muestra en la siguiente sección, el número de host se puede subdividir aún más: Esta división está controlada por la autoridad que administra la red. No está controlado por los RIR.

##### Direcciones IP reservadas

Un componente de una dirección IP con un valor *de todos los bits 0* o *todos los bits 1* tiene un significado especial:

Todos los bits 0: Una dirección con todos los bits cero en la parte del número de host se interpreta como *este* host (dirección IP con <dirección de host>=0). Todos los bits cero en la parte del número de red son *esta* red (dirección IP con <dirección de red>=0). Cuando un host desea comunicarse a través de una red, pero aún no conoce la dirección IP de la red, puede enviar paquetes con <dirección de red>=0. Otros hosts de la red interpretan la dirección como si se refiriera a *esta* red. Sus respuestas contienen la dirección de red completa, que el remitente registra para su uso futuro.

Todos los bits 1: Una dirección con todos los bits uno se interpreta como *todas las* redes o *todos los* hosts. Por ejemplo, lo siguiente significa todos los hosts de la red 128.2 (dirección de clase B):

128.2.255.255

Esto se denomina dirección de difusión dirigida porque contiene una <dirección de red> válida y una <dirección de host> de difusión.

Loopback: La red de Clase A 127.0.0.0 se define como la red de loopback. Las direcciones de esa red se asignan a interfaces que procesan datos dentro del sistema local. Estas interfaces de bucle invertido no acceden a una red física.

##### Direcciones IP de uso especial

RFC 3330 analiza las direcciones IP de uso especial. Proporcionamos una breve descripción de estas direcciones IP en la Tabla 3-1.

*Tabla 3-1 Direcciones IP de uso especial*

|  |  |
| --- | --- |
| **Bloque de direcciones** | **Uso actual** |
| 0.0.0.0/8 | "Esta" red |
| 14.0.0.0/8 | Redes de datos públicos |
| 24.0.0.0/8 | Redes de televisión por cable |
| 39.0.0.0/8 | Reservado pero sujeto a asignación |
| 128.0.0.0/16 | Reservado pero sujeto a asignación |
| 169.254.0.0/16 | Enlace local |
| 191.255.0.0/16 | Reservado pero sujeto a asignación |
| 192.0.0.0/24 | Reservado pero sujeto a asignación |
| 192.0.2.0/24 | Test-Net 192.88.99.0/24 Retransmisión 6to4 anycast |
| 198.18.0.0/15 | Pruebas comparativas de dispositivos de interconexión de red |
| 223.255.255.0/24 | Reservado pero sujeto a asignación |
| 224.0.0.0/4 | Multidifusión |
| 240.0.0.0/4 | Reservado para uso futuro |

#### 3.1.2 Subredes IP

Debido al crecimiento explosivo de Internet, el principio de direcciones IP asignadas se volvió demasiado inflexible para permitir cambios fáciles en las configuraciones de la red local. Esos cambios pueden producirse cuando:

Se instala un nuevo tipo de red física en una ubicación.

El crecimiento del número de hosts requiere dividir la red local en dos o más redes separadas.

Las distancias crecientes requieren dividir una red en redes más pequeñas, con puertas de enlace entre ellas.

Para evitar tener que solicitar direcciones de red IP adicionales, se introdujo el concepto de subredes IP. La asignación de subredes se realiza localmente. Toda la red sigue apareciendo como una red IP para el mundo exterior.

La parte del número de host de la dirección IP se subdivide en un segundo número de red y un número de host. Esta segunda red se denomina *subred* o *subred*. La red principal ahora consta de una serie de subredes. La dirección IP se interpreta como:

<número de red><número de subred><número de host>

La combinación de número de subred y número de host a menudo se denomina *dirección local* o parte *local* de la dirección IP. *Las subredes* se implementan de una manera transparente para las redes remotas. Un host dentro de una red que tiene subredes es consciente de la estructura de subredes. Un host en una red diferente no lo es. Este host remoto sigue considerando la parte local de la dirección IP como un número de host.

El administrador local elige la división de la parte local de la dirección IP en un número de subred y un número de host. Cualquier bit de la parte local se puede utilizar para formar la subred. La división se realiza mediante una máscara de subred de 32 bits. Los bits con un valor de cero bits en la máscara de subred indican las posiciones atribuidas al número de host. Los bits con un valor de uno indican las posiciones atribuidas al número de subred. Las posiciones de bits en la máscara de subred que pertenecen al número de red original se establecen en unos, pero no se utilizan (en algunas configuraciones de plataforma, este valor se especificaba con ceros en lugar de unos, pero de cualquier manera no se utiliza). Al igual que las direcciones IP, las máscaras de subred suelen escribirse en forma decimal con puntos.

El tratamiento especial de todos los bits cero y todos los bits uno se aplica a cada una de las tres partes de una dirección IP en subred, al igual que a las dos partes de una dirección IP que no ha sido dividida en subredes (consulte "Direcciones IP reservadas" en la página 71). Por ejemplo, la creación de subredes en una red de clase B puede utilizar uno de los siguientes esquemas:

El primer octeto es el número de subred; El segundo octeto es el número de host. Esto proporciona 28-2 (254) subredes posibles, cada una con hasta 28-2 (254) hosts. Recordemos que restamos dos de las posibilidades de dar cuenta de los casos de todos unos y todos ceros. La máscara de subred es 255.255.255.0.

Los primeros 12 bits se utilizan para el número de subred y los últimos cuatro para el número de host. Esto proporciona 212-2 (4094) subredes posibles, pero solo 24-2 (14) hosts por subred. La máscara de subred es 255.255.255.240.

En este ejemplo, hay varias otras posibilidades para asignar las partes de subred y host de la dirección. Antes de definir esta estructura, se debe tener en cuenta el número de subredes y hosts, así como los requisitos futuros. En el último ejemplo, la red de clase B con subredes tiene 16 bits que se dividen entre los campos de número de subred y número de host. El administrador de red define un número mayor de subredes, cada una con un número pequeño de hosts, o un número menor de subredes, cada una con muchos hosts.

Al asignar la parte de subred de la dirección local, el objetivo es asignar un *número* de bits al número de subred y el resto a la dirección local.

Por lo tanto, es normal utilizar un bloque contiguo de bits al principio de la parte de dirección local para el número de subred. Esto hace que las direcciones sean más legibles. (Esto es especialmente cierto cuando la subred ocupa 8 o 16 bits). Con este enfoque, cualquiera de las máscaras de subred anteriores son máscaras "aceptables". Las mascarillas como 255.255.252.252 y 255.255.255.15 son "inaceptables". De hecho, la mayoría de las implementaciones TCP/IP no admiten máscaras de subred no contiguas. Su uso está universalmente desaconsejado.

##### Tipos de subredes

Hay dos tipos de subredes: estáticas y de longitud variable. Las subredes de longitud variable son más flexibles que las estáticas. El enrutamiento IP nativo y la versión 1 de RIP solo admiten subredes estáticas. Sin embargo, la versión 2 de RIP admite subredes de longitud variable (consulte el Capítulo 5, "Protocolos de enrutamiento" en la página 171).

###### Subredes estáticas

La creación de subredes estáticas implica que todas las subredes obtenidas de la misma red utilizan la misma máscara de subred. Aunque esto es simple de implementar y fácil de mantener, puede desperdiciar espacio de direcciones en redes pequeñas. Considere una red de cuatro hosts con una máscara de subred de 255.255.255.0. Esta asignación desperdicia 250 direcciones IP. Todos los hosts y enrutadores son necesarios para admitir subredes estáticas.

###### Subredes de longitud variable

Cuando se utilizan subredes de longitud variable o máscaras de subred de longitud variable (VLSM), las subredes asignadas dentro de la misma red pueden usar diferentes máscaras de subred. Una subred pequeña con solo unos pocos hosts puede usar una máscara que se adapte a esta necesidad. Una subred con muchos hosts requiere una máscara de subred diferente. La capacidad de asignar máscaras de subred de acuerdo con las necesidades de las subredes individuales ayuda a conservar las direcciones de red. Las subredes de longitud variable dividen la red de modo que cada subred contenga suficientes direcciones para admitir el número necesario de hosts.

Una subred existente se puede dividir en dos partes agregando otro bit a la parte de subred de la máscara de subred. Otras subredes de la red no se ven afectadas por el cambio.

###### Mezcla de subredes estáticas y de longitud variable

No todos los dispositivos IP incluyen soporte para subredes de longitud variable. Inicialmente, parece que la presencia de un host que solo admite subredes estáticas impide el uso de subredes de longitud variable. Este no es el caso. Los enrutadores que interconectan las subredes se utilizan para ocultar las diferentes máscaras de los hosts. Los hosts siguen utilizando el enrutamiento IP básico. Esto descarga las complejidades de las subredes a enrutadores dedicados.

##### Ejemplo de subredes estáticas

Considere la red de Clase A que se muestra en la Figura 3-2.

0

Clase A

netID

ID de host

1 2 3

01

8 6 4

1

*Figura 3-2 IP: Dirección de clase A sin subredes* Utilice la dirección IP que se muestra en la Figura 3-3.

|  |
| --- |
| 00001001 01000011 00100110 00000001 una dirección de 32 bits  9 67 38 1 notación decimal (9.67.38.1) |

*Figura 3-3 Dirección IP*

La dirección IP es 9.67.38.1 (Clase A) con 9 como <dirección de red> y 67.38.1 como <dirección de host>.

Es posible que el administrador de red desee elegir los bits del 8 al 25 para indicar la dirección de la subred. En ese caso, los bits del 26 al 31 indican las direcciones del host. La Figura 3-4 muestra la dirección de subred derivada de la dirección de Clase A original.

1 2 3

01

8 6 4

1

Clase A

Subred

Número de subred

0

anfitrión

IDENTIFICACIÓN

netID

*Figura 3-4 IP: Dirección de clase A con máscara de subred y dirección de subred*

Una máscara de bits, conocida como máscara de subred, se utiliza para identificar qué bits del campo de dirección de host original indican el número de subred. En el ejemplo anterior, la máscara de subred es 255.255.255.192 (o 11111111 11111111 11111111

11000000 en notación de bits). Tenga en cuenta que, por convención, la <dirección de red> también se incluye en la máscara.

Debido a las restricciones de todos los bits 0 y todos los bits 1, esto define 2subredes válidas 18-2 (de 1 a 262143). Esta división proporciona 262142 subredes, cada una con un máximo de 26-2 (62) hosts.

El valor aplicado al número de subred toma el valor del octeto completo con bits no significativos establecidos en cero. Por ejemplo, el valor hexadecimal 01 de esta máscara de subred supone un valor de 8 bits 01000000. Esto proporciona un valor de subred de 64.

La aplicación de 255.255.255.192 a la dirección de clase A de muestra de 9.67.38.1 proporciona la siguiente información:

00001001 01000011 00100110 00000001 = 9.67.38.1 (dirección Clase A)

11111111 11111111 11111111 11------ 255.255.255.192 (máscara de subred)

===================================== logical\_AND 00001001 01000011 00100110 00------ = 9.67.38.0 (dirección base de subred)

Esto deja una dirección de host de:

-------- -------- -------- --000001 = 1 (dirección de host)

IP reconocerá todas las direcciones de host como si estuvieran en la red local para la cual la operación de logical\_AND descrita anteriormente produce el mismo resultado. Esto es importante para el enrutamiento de datagramas IP en entornos de subred (consulte 3.1.3, "Enrutamiento IP" en la página 77).

El número de subred es:

-------- 01000011 00100110 00------ = 68760 (número de subred)

Este número de subred es un número relativo. Es decir, es la subred número 68760 de la red 9 con la máscara de subred dada. Este número no se parece en nada a la dirección IP real que se le ha asignado a este host (9.67.38.1). No tiene ningún significado en términos de enrutamiento IP.

El administrador de red elige la división de la <dirección de host> original en <subred><host>. Los valores de todos los ceros y todos los unos del campo <subred> están reservados.

##### Ejemplo de subredes de longitud variable

Considere una corporación a la que se le ha asignado la red de Clase C

165.214.32.0. La corporación tiene el requisito de dividir este rango de direcciones en cinco redes separadas, cada una con el siguiente número de hosts:

Subred 1: 50 hosts

Subred 2: 50 hosts

Subred 3: 50 hosts

Subred 4: 30 hosts

Subred 5: 30 hosts

Esto no se puede lograr con subredes estáticas. En este ejemplo, las subredes estáticas dividen la red en cuatro subredes, cada una con 64 hosts, u ocho subredes, cada una con 32 hosts. Esta asignación de subred no cumple los requisitos establecidos.

Para dividir la red en cinco subredes, es necesario definir varias máscaras. Con una máscara de 255.255.255.192, la red se puede dividir en cuatro subredes, cada una con 64 hosts. La cuarta subred se puede dividir en dos subredes, cada una con 32 hosts, mediante una máscara de 255.255.255.224. Habrá tres subredes, cada una con 64 hosts, y dos subredes con 32 hosts cada una. Esto satisface los requisitos establecidos y elimina la posibilidad de un gran número de direcciones de host desperdiciadas.

##### Determinación de la máscara de subred

Por lo general, los hosts almacenarán la máscara de subred en un archivo de configuración. Sin embargo, a veces esto no se puede hacer, por ejemplo, como en el caso de una estación de trabajo sin disco. El protocolo ICMP incluye dos mensajes: solicitud de máscara de dirección y respuesta de máscara de dirección. Estos permiten a los hosts obtener la máscara de subred correcta de un servidor (consulte "Solicitud de máscara de dirección (17) y respuesta de máscara de dirección (18)" en la página 116).

##### Direccionamiento de enrutadores y hosts multihost

Cada vez que un host tiene una conexión física a varias redes o subredes, se describe como *multihost.* De forma predeterminada, todos los enrutadores son multihost porque su propósito es unir redes o subredes. Un host multihost tiene diferentes direcciones IP asociadas a cada adaptador de red. Cada adaptador se conecta a una subred o red diferente.

#### 3.1.3 Encaminamiento IP

Una función importante de la capa IP es el *enrutamiento IP*. Esto proporciona el mecanismo básico para que los enrutadores interconecten diferentes redes físicas. Un dispositivo puede funcionar simultáneamente como un host normal y un enrutador.

Un router de este tipo se denomina router con información de ruteo parcial. El router solo tiene información sobre cuatro tipos de destinos:

Hosts que están conectados directamente a una de las redes físicas a las que está conectado el enrutador.

Hosts o redes para los que el router ha recibido definiciones explícitas.

Hosts o redes para los que el router ha recibido un mensaje de redireccionamiento ICMP.

Un valor predeterminado para todos los demás destinos.

Se necesitan protocolos adicionales para implementar un enrutador con todas las funciones. Este tipo de routers son imprescindibles en la mayoría de las redes, ya que pueden intercambiar información con otros routers del entorno. Revisamos los protocolos utilizados por estos routers en el Capítulo 5, "Protocolos de enrutamiento" en la página 171.

Hay dos tipos de enrutamiento IP: directo e indirecto.

##### Enrutamiento directo

Si el host de destino está conectado a la misma red física que el host de origen, los datagramas IP se pueden intercambiar directamente. Esto se hace encapsulando el datagrama IP en la trama de red física. Esto se denomina entrega directa y se conoce como enrutamiento directo.

##### Enrutamiento indirecto

El enrutamiento indirecto se produce cuando el host de destino no está conectado a una red conectada directamente al host de origen. La única forma de llegar al destino es a través de una o más puertas de enlace IP. (Tenga en cuenta que en la terminología TCP/IP, los términos puerta de enlace y enrutador se utilizan indistintamente. Esto describe un sistema que realiza las funciones de un enrutador). La dirección de la primera puerta de enlace (el primer salto) se denomina ruta indirecta en el algoritmo de enrutamiento IP. La dirección de la primera puerta de enlace es la única información que necesita el host de origen para enviar un paquete al host de destino.

En algunos casos, puede haber varias subredes definidas en la misma red física. Si los hosts de origen y destino se conectan a la misma red física pero están definidos en diferentes subredes, se utiliza el enrutamiento indirecto para comunicarse entre el par de dispositivos. Se necesita un enrutador para reenviar el tráfico entre subredes.

La Figura 3-5 muestra un ejemplo de rutas directas e indirectas. Aquí, el host C tiene una ruta directa a los hosts B y D, y una ruta indirecta al host A a través de la puerta de enlace B.

Anfitrión D

Anfitrión C

Anfitrión B

Anfitrión A

*Figura 3-5 IP: Rutas directas e indirectas*

##### Tabla de enrutamiento IP

La determinación de las rutas directas se deriva de la lista de interfaces locales. Se compone automáticamente mediante el proceso de enrutamiento IP en la inicialización. Además, se puede configurar una lista de redes y pasarelas asociadas (rutas indirectas). Esta lista se utiliza para facilitar el enrutamiento IP. Cada host mantiene el conjunto de asignaciones entre lo siguiente:

Direcciones de red IP de destino

Rutas a las siguientes puertas de enlace

Esta información se almacena en una tabla denominada tabla de enrutamiento IP. En esta tabla hay tres tipos de asignaciones:

Las rutas directas que describen las redes conectadas localmente

Las rutas indirectas que describen las redes a las que se puede llegar a través de una o más Gateways

La ruta predeterminada que contiene la ruta (directa o indirecta) utilizada cuando la red IP de destino no se encuentra en las asignaciones de los tipos anteriores de tipo 1 y 2

La Figura 3-6 presenta una red de ejemplo.

*Figura 3-6 IP: Escenario de tabla de enrutamiento*

Anfitrión D

Anfitrión C

Anfitrión E

Anfitrión F

Anfitrión B

Anfitrión A

128.15

129.7

128.10

La tabla de enrutamiento del host D puede contener las siguientes entradas (simbólicas) (Tabla 3-2).

*Tabla 3-2 Entradas de ejemplo del host D*

|  |  |  |
| --- | --- | --- |
| **Destino** | **Enrutador** | **Interfaz** |
| 129.7.0.0 | E | lan0 |
| 128.15.0.0 | D | lan0 |
| 128.10.0.0 | B | lan0 |
| predeterminado | B | lan0 |
| 127.0.0.1 | Loopback | retrete |

Dado que D está conectado directamente a la red 128.15.0.0, mantiene una ruta directa para esta red. Sin embargo, para llegar a las redes 129.7.0.0 y 128.10.0.0, debe tener una ruta indirecta a través de E y B, respectivamente, porque estas redes no están conectadas directamente a ella.

La tabla de enrutamiento del host F puede contener las siguientes entradas (simbólicas) (Tabla 3-3).

*Tabla 3-3 Entradas de ejemplo del host F*

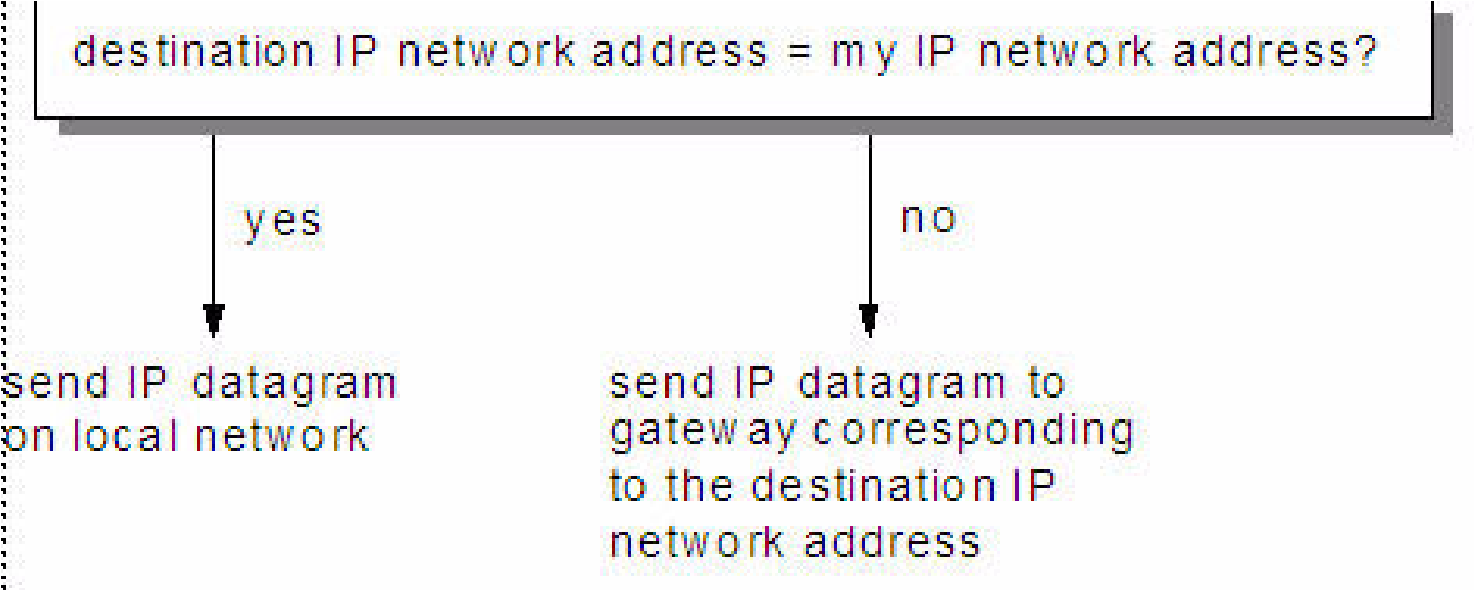
|  |  |  |
| --- | --- | --- |
| **Destino** | **Enrutador** | **Interfaz** |
| 129.7.0.0 | F | wan0 |
| predeterminado | E | wan0 |
| 127.0.0.1 | Loopback | lo |

Dado que se debe llegar a todos los hosts que no están en la red 129.7.0.0 a través del host E, el host F simplemente mantiene una ruta predeterminada a través de E.

##### Algoritmo de enrutamiento IP

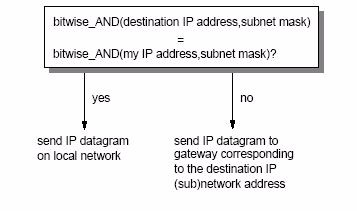
IP utiliza un algoritmo único para enrutar datagramas, como se ilustra en la Figura 3-7.

*Figura 3-7 IP: Enrutamiento sin subredes*



Para diferenciar entre subredes, se actualiza el algoritmo de enrutamiento IP, como se muestra en la Figura 3-8.

*Figura 3-8 IP: Enrutamiento con subredes*



Algunas implicaciones de este cambio incluyen:

Este algoritmo representa un cambio en el algoritmo general de IP. Por lo tanto, para poder operar de esta manera, la puerta de enlace en particular debe contener el nuevo algoritmo. Es posible que algunas implementaciones sigan utilizando el algoritmo general y no funcionen dentro de una red con subredes, aunque pueden seguir comunicándose con hosts de otras redes que tienen subredes.

Como el enrutamiento IP se utiliza en todos los hosts (y no solo en los enrutadores), todos los hosts de la subred deben tener:

* Un algoritmo de enrutamiento IP que admite subredes
* La misma máscara de subred (a menos que se formen subredes dentro de la subred)

Si la implementación de IP en cualquiera de los hosts no admite la creación de subredes, ese host podrá comunicarse con cualquier host de su propia subred, pero no con ningún equipo de otra subred dentro de la misma red. Esto se debe a que el host solo ve una red IP y su enrutamiento no puede diferenciar entre un datagrama IP dirigido a un host en la subred local y un datagrama que debe enviarse a través de un enrutador a una subred diferente.

En caso de que uno o más hosts no admitan la creación de subredes, existe una forma alternativa de lograr el mismo objetivo en forma de *proxy-ARP*. Esto no requiere ningún cambio en el algoritmo de enrutamiento IP para hosts de un solo host. Requiere cambios en los routers entre subredes de la red (consulte 3.4.4, "Proxy-ARP o subredes transparentes" en la página 123).

La Figura 3-9 ilustra todo el algoritmo de enrutamiento IP.

Interfaz(es) bit a bit Y local(es)

con local\_subnet\_mask(s)

Sí

Sí

Sí

Tomar IP de destino

dirección

No

No

No

Enviar mensaje de error ICMP

"Red inalcanzable"

Bit a bit Y dest\_IP\_addr

con local\_subnet\_mask(s)

¿Hay una coincidencia?

¿Existe una ruta indirecta?

¿entrada?

Es una ruta predeterminada

¿especificado?

Entregar indirectamente a la

Dirección IP predeterminada del router

Entrega directa usando el

local correspondiente

interfaz

Entregar indirectamente a la

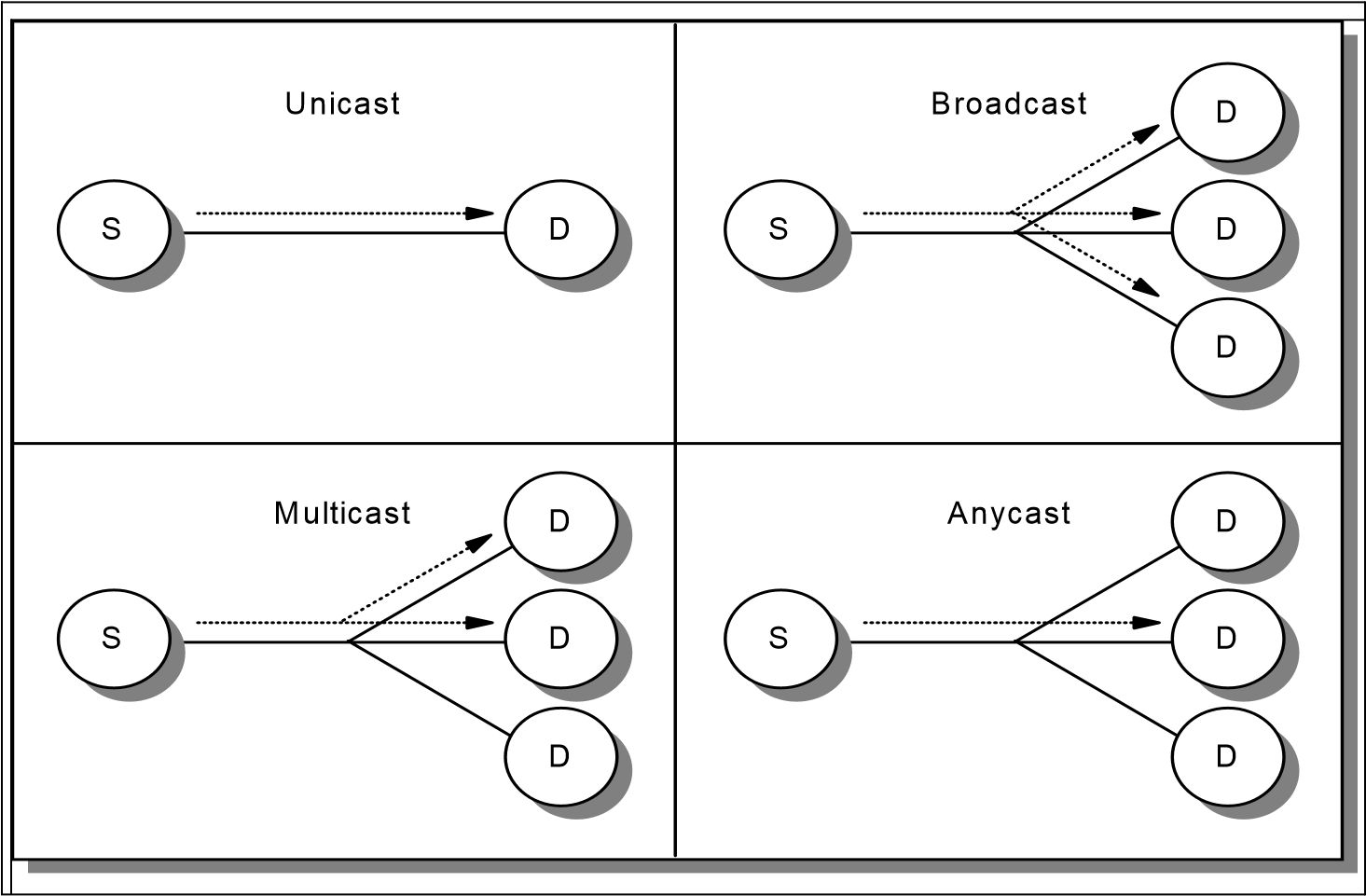
IP del router correspondiente

dirección

*Figura 3-9 IP: Algoritmo de enrutamiento (con subredes)*

#### 3.1.4 Métodos de distribución: unidifusión, radiodifusión, multidifusión y difusión por proximidad

La mayoría de las direcciones IP se refieren a un único destinatario, esto se denomina *unidifusión* dirección. Las conexiones de unidifusión especifican una relación uno a uno entre un único origen y un único destino. Además, hay tres tipos especiales de direcciones IP que se utilizan para dirigirse a varios destinatarios: direcciones de difusión, direcciones de multidifusión y direcciones de difusión por proximidad. La figura 3-10 muestra su funcionamiento.



*Figura 3-10 IP: Modos de entrega de paquetes*

Un  *protocolo sin conexión* puede enviar mensajes de unidifusión, difusión, multidifusión o difusión por proximidad. Un  *protocolo orientado a la conexión* solo puede utilizar direcciones de unidifusión (debe existir una conexión entre un par específico de hosts).

##### Radiodifusión

Las direcciones de difusión nunca son válidas como dirección de origen. Deben especificar la dirección de destino. Los diferentes tipos de direcciones de difusión incluyen:

Dirección de difusión limitada: Utiliza la dirección 255.255.255.255 (todos los bits 1 en todas las partes de la dirección IP). Hace referencia a todos los hosts de la subred local. Esto es reconocido por todos los anfitriones. Los hosts no necesitan ninguna configuración de IP

información. Los routers no reenvían este paquete.

Una excepción a esta regla se denomina *reenvío BOOTP*. El protocolo BOOTP utiliza la dirección de difusión limitada para permitir que una estación de trabajo sin disco se ponga en contacto con un servidor de arranque. El reenvío BOOTP es una opción de configuración disponible en algunos routers. Sin esta función, se requiere un servidor BOOTP independiente en cada subred (consulte 3.6, "Protocolo de arranque (BOOTP)" en la página 125).

Dirección de difusión dirigida a la red: se utiliza en un entorno sin subredes. El número de red es un número de red válido y el número de host es todos unos (por ejemplo, 128.2.255.255). Esta dirección se refiere a todos los hosts de la red especificada. Los routers deben reenviar estos mensajes de difusión. Esto se utiliza en solicitudes ARP (consulte 3.4, "Protocolo de resolución de direcciones (ARP)" en la página 119) en redes sin subred.

Dirección de difusión dirigida a la subred: si el número de red es un número de red válido, el número de subred es un número de subred válido y el número de host es todos unos, la dirección hace referencia a todos los hosts de la subred especificada. Dado que la subred del remitente y la subred de destino pueden tener una máscara de subred diferente, el remitente debe determinar de alguna manera la máscara de subred en uso en el destino. La difusión la realiza el router que recibe el datagrama en la subred.

Dirección de difusión dirigida a todas las subredes: si el número de red es un número de red válido, la red está dividida en subredes y la parte local es todos unos (por ejemplo, 128.2.255.255), la dirección hace referencia a todos los hosts de todas las subredes de la red especificada. En principio, los enrutadores pueden propagar difusiones para todas las subredes, pero no están obligados a hacerlo. En la práctica, no lo hacen. Hay muy pocas circunstancias en las que sea deseable una emisión de este tipo. Si está mal configurado, puede dar lugar a problemas. Considere el host 9.180.214.114 mal configurado en una red de clase A con subredes. Si el dispositivo se configuró con la dirección 9.255.255.255 como dirección de difusión local en lugar de 9.180.214.255, todos los routers de la red reenviarán la solicitud a todos los clientes.

Si los enrutadores respetan la dirección de difusión dirigida a todas las subredes, utilizan un algoritmo llamado *reenvío de ruta inversa* para evitar que los mensajes de difusión se multipliquen fuera de control. Consulte RFC 922 para obtener más detalles sobre este algoritmo.

##### Multidifusión

Si un datagrama IP se difunde a una subred, lo reciben todos los hosts de la subred. Cada host procesa el paquete para determinar si el protocolo de destino está activo. Si no está activo, el datagrama IP se descarta. La multidifusión evita esto mediante la selección de grupos de destino.

Cada grupo está representado por una dirección IP de clase D. Para cada dirección de multidifusión, un conjunto de cero o más hosts están escuchando los paquetes dirigidos a la dirección. Este conjunto de hosts se denomina *grupo de hosts*. Los paquetes enviados a una dirección de multidifusión se reenvían solo a los miembros del grupo host correspondiente. La multidifusión permite conexiones de uno a varios (consulte el Capítulo 6, "Multidifusión IP" en la página 237).

##### Anycasting

A veces, los mismos servicios IP son proporcionados por diferentes hosts. Por ejemplo, un usuario desea descargar un archivo mediante FTP y el archivo está disponible en varios servidores FTP. Los hosts que implementan el mismo servicio proporcionan una dirección de difusión por proximidad a otros hosts que requieren el servicio. Las conexiones se realizan con el primer host del grupo de direcciones de difusión por proximidad que responde. Este proceso se utiliza para garantizar que el servicio sea proporcionado por el host con la mejor conexión con el receptor.

El servicio anycast se incluye en IPV6 (consulte 9.2.2, "Direccionamiento IPv6" en la página 339).

#### 3.1.5 El problema del agotamiento de la dirección IP

El número de redes en Internet se ha duplicado aproximadamente anualmente durante varios años. Sin embargo, el uso de las redes de clase A, B y C difiere mucho. Casi todas las nuevas redes asignadas a finales de la década de 1980 eran de Clase B, y en 1990 se hizo evidente que si esta tendencia continuaba, el último número de red de Clase B se asignaría durante 1994. Sin embargo, las redes de clase C apenas se utilizaban.

La razón de esta tendencia fue que la mayoría de los usuarios potenciales encontraron que una red de Clase B era lo suficientemente grande para sus necesidades anticipadas, ya que tiene capacidad para hasta 65534 hosts, mientras que una red de Clase C, con un máximo de 254 hosts, restringe severamente el crecimiento potencial incluso de una pequeña red inicial. Además, la mayoría de las redes de Clase B asignadas eran pequeñas. Hay relativamente pocas redes que necesitarían hasta 65.534 direcciones de host, pero muy pocas para las que 254 hosts sería un límite adecuado. En resumen, aunque las divisiones de Clase A, Clase B y Clase C de la dirección IP son lógicas y fáciles de usar (porque se producen en los límites de bytes), en retrospectiva, no son las más prácticas porque las redes de Clase C son demasiado pequeñas para ser útiles para la mayoría de las organizaciones, mientras que las redes de Clase B son demasiado grandes para ser densamente pobladas por cualquiera que no sea la organización más grande.

En mayo de 1996, todas las direcciones de Clase A fueron asignadas o asignadas, así como el 61,95 por ciento de las direcciones de red IP de Clase B y el 36,44 por ciento de las de Clase C.

Los términos asignados y asignados en este contexto tienen los siguientes significados:

Asignado: el número de números de red en uso. Las cifras de la Clase C son algo inexactas, porque las cifras no incluyen muchas de las de la Clase C

redes en Europa, que se asignaron a RIPE y posteriormente se asignaron, pero que todavía se registran como asignadas.

Asignadas: Esto incluye todas las redes asignadas y, además, aquellas redes que han sido reservadas por la IANA (por ejemplo, las 63 redes de Clase A están reservadas por la IANA) o que han sido asignadas a registros regionales por la IANA y posteriormente serán asignadas por esos registros.

Otra forma de ver estos números es examinar la proporción del espacio de direcciones que se ha utilizado. Por ejemplo, el espacio de direcciones de Clase A es tan grande como el resto combinado, y una sola red de Clase A puede tener teóricamente tantos hosts como 66.000 redes de Clase C.

Desde 1990, el número de redes asignadas de Clase B ha aumentado a un ritmo mucho menor que el número total de redes asignadas y aún no se ha producido el agotamiento previsto de los números de red de Clase B. La razón de ello es que a finales de 1990 se modificaron las políticas sobre atribución de números de red para preservar el espacio de direcciones existente, en particular para evitar el agotamiento del espacio de direcciones de Clase B. Las nuevas políticas se pueden resumir de la siguiente manera:

La mitad superior del espacio de direcciones de clase A (números de red 64 a 127) se reserva indefinidamente para permitir la posibilidad de utilizarlo para la transición a un nuevo esquema de numeración.

Las redes de clase B solo se asignan a organizaciones que pueden demostrar claramente su necesidad. Lo mismo ocurre, por supuesto, con las redes de clase A. Los requisitos para las redes de Clase B son que la organización solicitante:

* Tiene un plan de subredes que documenta más de 32 subredes dentro de su red organizativa
* Tiene más de 4096 anfitriones

Cualquier requisito para una red de Clase A se maneja en función de cada caso.

A las organizaciones que no cumplen con los requisitos para una red de Clase B se les asigna un bloque numerado consecutivamente de números de red de Clase C.

La mitad inferior del espacio de direcciones de Clase C (números de red 192.0.0 a 207.255.255) se divide en ocho bloques, que se asignan a las autoridades regionales de la siguiente manera:

|  |  |  |
| --- | --- | --- |
| **192.0.0 - 193.255.255** | Multirregional | |
| **194.0.0 - 195.255.255** | Europa | |
| **196.0.0 - 197.255.255** | Otros | |
| **198.0.0 - 199.255.255** | América del Norte | |
| **200.0.0 - 201.255.255** | América Central y del Sur | |
| **202.0.0 - 203.255.255** | Cuenca del Pacífico | |
| **204.0.0 - 205.255.255 206.0.0 - 207.255.255 208.0.0 - 209.255.255**  **210.0.0 - 211.255.255** | Otros  Otros  ARIN1  APNIC | |
| **212.0.0 - 213.255.255** | MADURO NCC |
| **214.0.0 - 215.255.255** | Departamento de Defensa de EE. UU. |
| **216.0.0 - 216.255.255** | ARIN |
| **217.0.0 - 217.255.255** | MADURO NCC |
| **218.0.0 - 218.255.255** | APNIC |
| **219.0.0 - 222.255.255** | APNIC |

Los rangos definidos como Otros deben ser aquellos en los que se requiere flexibilidad fuera de las limitaciones de los límites regionales. El rango definido como multirregional incluye las redes de Clase C que se asignaron antes de que se adoptara este nuevo esquema. Las 192 redes fueron asignadas por el InterNIC y las 193 redes fueron previamente asignadas a RIPE en Europa.

Cuando una organización tiene un rango de números de red de Clase C, el rango proporcionado se asigna como un  *rango contiguo bit a bit* de números de red, y el número de redes en el rango es una potencia de 2. Es decir, todas las direcciones IP del intervalo tienen un prefijo común y todas las direcciones con ese prefijo están dentro del intervalo. Por ejemplo, a una organización europea que requiera 1500 direcciones IP se le asignarían ocho números de red de Clase C (2048 direcciones IP) del espacio numérico reservado para las redes europeas (194.0.0 a 195.255.255) y el primero de estos números de red sería divisible por ocho. Un rango de direcciones que satisfacen estas reglas es 194.32.136 a 194.32.143, en cuyo caso el rango consta de todas las direcciones IP con el prefijo de 21 bits 194.32.136, o B '110000100010000010001'.

El número máximo de números de red asignados de forma contigua es 64, lo que corresponde a un prefijo de 18 bits. Una organización que requiere más de 4096 direcciones pero menos de 16.384 direcciones puede solicitar una dirección de Clase B o un rango de direcciones de Clase C. En general, el número de redes de Clase C asignadas es el mínimo requerido para proporcionar el número necesario de direcciones IP para la organización sobre la base de una perspectiva de dos años. Sin embargo, en algunos casos, una organización puede solicitar que varias redes se traten por separado. Por ejemplo, a una organización con 600 hosts normalmente se le asignan cuatro redes de clase C. Sin embargo, si esos hosts se distribuyeran en 10 LAN con entre 50 y 70 hosts por LAN, dicha asignación puede causar problemas graves, ya que la organización tendría que encontrar 10 subredes dentro de un rango de direcciones locales de 10 bits. Esto significa que al menos algunas de las LAN tienen una máscara de subred de 255.255.255.192, lo que permite solo 62 hosts por LAN. La intención de las reglas no es forzar a la organización a crear subredes complejas

de redes pequeñas, y la organización debe solicitar 10 números diferentes de Clase C, uno para cada LAN.

1 La información de este y los siguientes números de esta lista proviene de:

<http://www.iana.org/assignments/ipv4-address-space>

Las reglas actuales se encuentran en RFC 2050, que actualiza RFC 1466. Las razones de las normas para la atribución de números de red de clase C se pondrán de manifiesto en las secciones siguientes. El uso de números de red de Clase C de esta manera ha evitado el agotamiento del espacio de direcciones de Clase B, pero no es una solución permanente a las limitaciones generales del espacio de direcciones que son fundamentales para IP. Discutimos una solución a largo plazo en el Capítulo 9, "IP versión 6" en la página 327.

#### 3.1.6 Intranets: Direcciones IP privadas

Otro enfoque para conservar el espacio de direcciones IP se describe en RFC 1918. Este RFC relaja la regla de que las direcciones IP deben ser únicas globalmente. Reserva parte del espacio global de direcciones para su uso en redes que no requieren conectividad a Internet. Normalmente, estas redes son administradas por una sola organización. Para ello se han reservado tres rangos de direcciones:

10.0.0.0: Una sola red de clase A

172.16.0.0 a 172.31.0.0: 16 redes contiguas de clase B

192.168.0.0 a 192.168.255.0: 256 redes contiguas de Clase C

Cualquier organización puede utilizar cualquier dirección de estos intervalos. Sin embargo, debido a que estas direcciones no son únicas globalmente, no se definen para ningún enrutador externo. Se espera que los enrutadores en redes que no utilizan direcciones privadas, particularmente aquellas operadas por proveedores de servicios de Internet, descarten silenciosamente toda la información de enrutamiento relacionada con estas direcciones. Se espera que los enrutadores de una organización que utilizan direcciones privadas limiten todas las referencias a direcciones privadas a los enlaces internos. No deben anunciar externamente rutas a direcciones privadas ni reenviar datagramas IP que contengan direcciones privadas a enrutadores externos.

Los hosts que solo tienen una dirección IP privada no tienen conectividad directa de capa IP a Internet. Toda la conectividad a hosts de Internet externos debe proporcionarse con puertas de enlace de aplicaciones (consulte "Puerta de enlace de nivel de aplicación (proxy)" en la página 798), SOCKS (consulte 22.5, "SOCKS" en la página 846) o traducción de direcciones de red (NAT), que se describe en la sección siguiente.

#### 3.1.7 Traducción de direcciones de red (NAT)

En esta sección se explica la traducción tradicional de direcciones de red (NAT), la NAT básica y la traducción de puertos de direcciones de red (NAPT). NAT también se conoce como enmascaramiento de IP. Proporciona una asignación entre direcciones IP internas y Direcciones externas asignadas oficialmente.

Originalmente, NAT se sugirió como una solución a corto plazo para el problema de agotamiento de direcciones IP. Además, muchas organizaciones han utilizado, en el pasado, direcciones IP asignadas localmente, sin esperar requerir conectividad a Internet.

Hay dos variaciones de NAT tradicional, NAT básica y NAPT. La NAT tradicional se define en RFC 3022 y se analiza en RFC 2663. En las siguientes secciones se proporciona una breve explicación de la NAT tradicional, la NAT básica y la NAPT basadas en RFC 3022.

##### NAT tradicional

La idea de NAT tradicional (en lo sucesivo, NAT) se basa en el hecho de que solo un pequeño número de hosts en una red privada se comunican fuera de esa red. Si a cada host se le asigna una dirección IP del grupo de direcciones IP oficial solo cuando necesita comunicarse, solo se requiere un pequeño número de direcciones oficiales.

NAT puede ser una solución para redes que tienen intervalos de direcciones privadas o direcciones no oficiales y desean comunicarse con hosts en Internet. Cuando un servidor proxy, un servidor SOCKS o un firewall no están disponibles o no cumplen requisitos específicos, se puede usar NAT para administrar el tráfico entre la red interna y externa sin anunciar las direcciones de host internas.

##### NAT básica

Considere una red interna que se basa en el espacio de direcciones IP privadas y los usuarios desean usar un protocolo de aplicación para el que no hay ninguna puerta de enlace de aplicaciones; la única opción es establecer conectividad de nivel IP entre los hosts de la red interna y los hosts de Internet. Debido a que los enrutadores en Internet no sabrían cómo enrutar paquetes IP a una dirección IP privada, no tiene sentido enviar paquetes IP con direcciones IP privadas como direcciones IP de origen a través de un enrutador a Internet.

Como se muestra en la Figura 3-11, la NAT básica toma la dirección IP de un paquete saliente y la traduce dinámicamente a una dirección global asignada oficialmente. En el caso de los paquetes entrantes, traduce la dirección asignada a una dirección interna.



No seguro

a.b.1.0/24

a.b.1.1

10.0.1.1

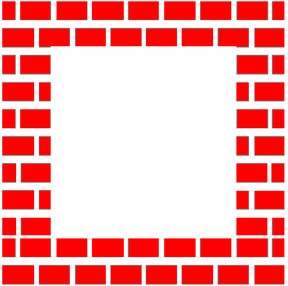
Seguro

src=a.b.1.1 dest=a.b.2.1

src=a.b.1.1 dest=10.0.1.1



10.0.0.0/8



NAT

Filtrado

TCP/UDP

IP/ICMP



Configuración de NAT

**RESERVA a.b.2.0 255.255.255.0**

**TRADUCIR 10.0.0.0 255.0.0.0**



Reglas de filtrado

Basado en no traducido

Direcciones IP (10.x.x.x)

*Figura 3-11 Traducción básica de direcciones de red (NAT)*

Desde el punto de dos hosts que intercambian paquetes IP entre sí, uno en la red interna y otro en la red externa, la NAT en sí es transparente (consulte la Figura 3-12).



No seguro

a.b.1.0/24

a.b.1.1

Parece un

Enrutador normal

src=a.b.1.1 dest=a.b.2.1

a.b.2.0/24

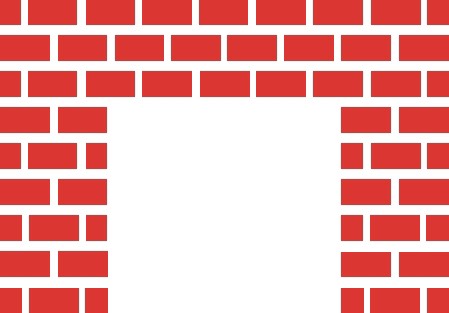
A.B.2.1

Seguro

*Figura 3-12 NAT vista desde la red externa*

##### Mecanismo básico de traducción NAT

Para cada paquete IP saliente, las reglas de configuración de NAT comprueban la dirección de origen. Si una regla coincide con la dirección de origen, la dirección se traduce a una dirección global desde el grupo de direcciones. El grupo de direcciones predefinido contiene las direcciones que NAT puede utilizar para la traducción. Para cada paquete entrante, se comprueba la dirección de destino si NAT la utiliza. Cuando esto es cierto, la dirección se traduce a la dirección interna original. La Figura 3-13 muestra la configuración básica de NAT.



Pendiente de traducción

Excluir

Red segura

Red no segura

Reserva

N

Un

T

Excluir

Cortafuegos

Mapa

Piscina

*Figura 3-13 Configuración básica de NAT*

Cuando la NAT básica traduce una dirección para un paquete IP, también se ajusta la suma de comprobación. En el caso de los paquetes FTP, la tarea es aún más difícil, ya que los paquetes pueden contener direcciones en los datos del paquete. Por ejemplo, el comando FTP PORT contiene una dirección IP en ASCII. Estas direcciones también deben estar traducidas correctamente; Las actualizaciones de la suma de comprobación y las actualizaciones de secuencia y reconocimiento TCP también deben realizarse en consecuencia.

Para que las tablas de enrutamiento funcionen, el diseño de la red IP debe elegir las direcciones como si se conectaran dos o más redes o subredes IP a través de un enrutador. Las direcciones IP NAT deben provenir de redes o subredes separadas, y las direcciones deben ser inequívocas con respecto a otras redes o subredes de la red no segura. Si la red externa es Internet, las direcciones NAT deben provenir de una red o subred pública; en otras palabras, las direcciones NAT deben ser asignadas por la IANA.

Las direcciones asignadas deben reservarse en un grupo para poder usarlas cuando sea necesario. Si las conexiones se establecen desde la red interna, NAT puede simplemente elegir la siguiente dirección pública libre en el grupo NAT y asignarla a la Solicitando host interno. El servicio NAT realiza un seguimiento de qué direcciones IP internas se asignan a qué direcciones IP externas en un momento dado, por lo que podrá asignar una respuesta que reciba de la red externa a la dirección IP segura correspondiente.

Cuando el servicio NAT asigna direcciones IP en función de la demanda, necesita saber cuándo devolver la dirección IP externa al grupo de direcciones IP disponibles. No hay configuración de conexión ni desmontaje en el nivel de IP, por lo que no hay nada en el propio protocolo IP que el servicio NAT pueda usar para determinar cuándo ya no es necesaria una asociación entre una dirección IP interna y una dirección IP externa NAT. Debido a que TCP es un protocolo orientado a la conexión, es posible obtener la información del estado de la conexión desde el encabezado TCP (ya sea que la conexión finalice o no), mientras que UDP no incluye dicha información. Por lo tanto, configure un valor de tiempo de espera que indique a NAT cuánto tiempo debe mantener una asociación en estado inactivo antes de devolver la dirección IP externa al grupo TDR libre. Por lo general, el valor predeterminado de este parámetro es de 15 minutos.

Los administradores de red también deben indicar a NAT si todos los hosts internos pueden usar NAT o no. Esto se puede hacer mediante el uso de los comandos de configuración correspondientes. Si los hosts de la red externa necesitan iniciar conexiones con los hosts de la red interna, NAT debe configurarse de antemano en cuanto a qué dirección NAT externa coincide con qué dirección IP interna. Por lo tanto, se debe definir una asignación estática para permitir conexiones desde redes externas a un host específico en la red interna. Tenga en cuenta que las direcciones NAT externas asignadas estáticamente a direcciones IP internas no deben superponerse con las direcciones especificadas como pertenecientes al grupo de direcciones externas que el servicio NAT puede utilizar según la demanda.

El servidor de nombres externo puede, por ejemplo, tener una entrada para una puerta de enlace de correo que se ejecuta en un equipo de la red interna. El servidor de nombres externo resuelve el nombre de host público de la puerta de enlace de correo interna en la dirección IP asignada estáticamente (la dirección externa) y el servidor de correo remoto envía una solicitud de conexión a esta dirección IP. Cuando esa solicitud llega al servicio TDR en la interfaz externa, el servicio TDR examina sus reglas de asignación para ver si tiene una asignación estática entre la dirección IP pública externa especificada y una dirección IP interna. Si es así, traduce la dirección IP y reenvía el paquete IP a la red interna a la puerta de enlace de correo.

##### Traducción de puertos de direcciones de red (NAPT)

La diferencia entre NAT básica y NAPT es que NAT básica se limita a traducir solo direcciones IP, mientras que NAPT se amplía para incluir la dirección IP y el identificador de transporte (como el puerto TCP/UDP o el ID de consulta ICMP).

Como se muestra en la Figura 3-14, la traducción de puertos de direcciones de red puede traducir muchas direcciones de red y sus identificadores de transporte en una sola dirección de red con muchos identificadores de transporte o, más específicamente, puertos.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tabla de transición  10.10.10.11:80 = a.b.65.1:8000  10.10.10.12:80 = a.b.65.1:8001 |  |  |
| Externo  A.B.65.0 /30  A.B.65.3 /30 |  | Interno  10.10.10.0 /24 | 10.10.10.11 /24  10.10.10.12 /24 |
| NAPT |

*Figura 3-14 Traducción del puerto de dirección de red*

NAPT asigna direcciones privadas a una única dirección única a nivel mundial. Por lo tanto, el enlace es desde la dirección privada y el puerto privado a la dirección asignada y al puerto asignado. NAPT permite que varios nodos de una red local accedan simultáneamente a redes remotas utilizando la única dirección IP asignada a su enrutador.

En NAPT, las modificaciones en el encabezado IP son similares a las de NAT básica. Sin embargo, para las sesiones TCP/UDP, las modificaciones deben ampliarse para incluir la traducción del puerto de origen para los paquetes salientes y el puerto de destino para los paquetes entrantes en el encabezado TCP/UDP. Además de las sesiones TCP/UDP, los mensajes ICMP, con la excepción del tipo de mensaje REDIRECT, también pueden ser monitoreados por el servicio NAPT que se ejecuta en el router. Los paquetes de tipo de consulta ICMP se traducen de forma similar a la de los paquetes TCP/UDP en el sentido de que el campo de identificador en el encabezado del mensaje ICMP se asignará de forma única a un identificador de consulta de la dirección IP registrada.

##### Limitaciones de NAT

Las limitaciones de NAT se mencionan en RFC 3022 y RFC2663. Aquí discutimos algunas de las limitaciones.

NAT funciona bien para las direcciones IP en el encabezado IP. Algunos protocolos de aplicación intercambian información de direcciones IP en la parte de datos de la aplicación de un paquete IP y, por lo general, NAT no podrá controlar la traducción de direcciones IP en el protocolo de aplicación. Actualmente, la mayoría de las implementaciones manejan el FTP protocolo. Cabe señalar que la implementación de NAT para aplicaciones específicas que tienen información de IP en los datos de la aplicación es más sofisticada que las implementaciones NAT estándar.

NAT es de cálculo intensivo incluso con la ayuda de un sofisticado algoritmo de ajuste de suma de comprobación, ya que cada paquete de datos está sujeto a búsquedas y modificaciones de NAT.

Es obligatorio que todas las solicitudes y respuestas relacionadas con una sesión se enruten a través del mismo enrutador que ejecuta el servicio NAT.

La traducción de fragmentos TCP/UDP salientes (es decir, los que se originan en hosts privados) en una configuración NAPT no funcionará (consulte "Fragmentación" en la página 104). Esto se debe a que solo el primer fragmento contiene el encabezado TCP/UDP necesario para asociar el paquete a una sesión con fines de traducción. Los fragmentos posteriores no contienen información de puerto TCP/UDP, sino que simplemente llevan el mismo identificador de fragmentación especificado en el primer fragmento. Cuando el host de destino recibe los dos datagramas no relacionados, que llevan el mismo ID de fragmentación y de la misma dirección de host asignada, no puede determinar a cuál de las dos sesiones pertenecen los datagramas. En consecuencia, ambas sesiones se dañarán.

NAT cambia parte de la información de la dirección en un paquete IP. Esto se convierte en un problema cuando se utiliza IPSec. Consulte 22.4, "Arquitectura de seguridad IP (IPSec)" en la página 809 y 22.10, "Descripción general de las redes privadas virtuales (VPN)" en la página 861. Cuando se utiliza la autenticación IPSec de extremo a extremo, un paquete cuya dirección se ha cambiado siempre fallará en su comprobación de integridad bajo el protocolo de encabezado de autenticación, ya que cualquier cambio en cualquier bit del datagrama invalidará el valor de comprobación de integridad generado por el origen. Debido a que los protocolos IPSec ofrecen algunas soluciones a los problemas de direccionamiento que anteriormente eran manejados por NAT, no hay necesidad de NAT cuando todos los hosts que componen una red privada virtual determinada usan direcciones IP (públicas) únicas globalmente. La ocultación de direcciones se puede lograr mediante el modo de túnel IPSec. Si una empresa utiliza direcciones privadas dentro de su intranet, el modo de túnel IPSec puede evitar que aparezcan en texto sin cifrar desde la Internet pública, lo que elimina la necesidad de NAT.

#### 3.1.8 Enrutamiento entre dominios sin clases (CIDR)

El enrutamiento IP estándar solo comprende direcciones de red de clase A, B y C. Dentro de cada una de estas redes, se pueden utilizar subredes para proporcionar una mejor granularidad. Sin embargo, no hay forma de especificar que varias redes de clase C estén relacionadas. El resultado de esto se denomina problema de explosión de la tabla de *enrutamiento* : una red de clase B de 3000 hosts requiere una entrada de tabla de enrutamiento en cada red troncal

enrutador. El mismo entorno, si se aborda como un rango de redes de clase C, requiere 16 entradas.

La solución a este problema se denomina Classless Inter-Domain Routing (CIDR).

El CIDR se describe en las RFC 1518 a 1520. El CIDR no enruta de acuerdo con el

clase del número de red (de ahí el término sin clases). Se basa únicamente en los bits de orden superior de la dirección IP. Estos bits se denominan prefijo IP.

Cada entrada de la tabla de ruteo CIDR contiene una dirección IP de 32 bits y una máscara de red de 32 bits, que juntas proporcionan la longitud y el valor del prefijo IP. Esto se representa como la <IP\_address network\_mask> de la tupla. Por ejemplo, para direccionar un bloque de ocho direcciones de Clase C con una sola entrada de tabla de enrutamiento, la siguiente representación es suficiente: <192.32.136.0 255.255.248.0>. Esto se refiere, desde el punto de vista de la red troncal, al rango de red de Clase C de 192.32.136.0 a 192.32.143.0 como una sola red. Esto se ilustra en la Figura 3-15.

11000000 00100000 10001000 00000000 = 192.32.136.0 (dirección de clase C)

11111111 11111111 11111--- -------- 255.255.248.0 (máscara de red)

===================================== logical\_AND

11000000 00100000 10001--- -------- = 192.32.136 (prefijo IP)

11000000 00100000 10001111 00000000 = 192.32.143.0 (dirección de clase C)

11111111 11111111 11111--- -------- 255.255.248.0 (máscara de red)

===================================== logical\_AND

11000000 00100000 10001--- -------- = 192.32.136 (mismo prefijo IP)

*Figura 3-15 Enrutamiento entre dominios sin clases: ejemplo de superred IP*

Este proceso de combinación de varias redes en una sola entrada se conoce como *superred.* El enrutamiento se basa en máscaras de red que son más cortas que la máscara de red natural de una dirección IP. Esto contrasta con las subredes (consulte 3.1.2, "Subredes IP" en la página 72) donde las máscaras de subred son más largas que la máscara de red natural.

Las políticas actuales de asignación de direcciones de Internet y los supuestos en los que se basaron esas políticas se describen en RFC 1518. Se pueden resumir de la siguiente manera:

La asignación de direcciones IP refleja la topología física de la red y no la topología organizativa. Siempre que los límites organizativos y administrativos no coincidan con la topología de red, *no* deben utilizarse para la asignación de direcciones IP.

En general, la topología de red seguirá de cerca los límites continentales y nacionales. Por lo tanto, las direcciones IP deben asignarse sobre esta base.

Habrá un conjunto relativamente pequeño de redes que transportan una gran cantidad de tráfico entre dominios de enrutamiento. Estas redes estarán interconectadas de una manera no jerárquica que trascienda las fronteras nacionales. Estas redes se denominan *dominios de enrutamiento de tránsito (TRD)* Cada TRD tendrá un prefijo IP único. Los TRD no se organizarán de forma jerárquica cuando no exista una jerarquía adecuada. Sin embargo, siempre que una TRD se encuentre totalmente dentro de un límite continental, su prefijo IP debe ser una extensión del prefijo IP continental.

Habrá muchas organizaciones que tengan vínculos con otras organizaciones que sean para el uso privado de esas dos organizaciones. Los archivos adjuntos no transportan tráfico destinado a otros dominios (tráfico de tránsito). Estas conexiones privadas no tienen un efecto significativo en la topología de enrutamiento y se pueden ignorar.

La gran mayoría de los dominios de enrutamiento serán de un solo hogar. Es decir, se adjuntarán a un solo TRD. Se les deben asignar direcciones que comiencen con el prefijo IP de ese TRD. Por lo tanto, todas las direcciones de todos los dominios de un solo hogar conectados a un TRD se pueden agregar en una sola entrada de tabla de enrutamiento para todos los dominios fuera de ese TRD.

Hay una serie de esquemas de asignación de direcciones que se pueden utilizar para dominios multihost. Entre ellas se encuentran:

* El uso de un único prefijo IP para el dominio. Los enrutadores externos deben tener una entrada para la organización que se encuentre parcial o totalmente fuera de la jerarquía normal. Cuando un dominio es multihost, pero todos los TRD conectados están topológicamente cerca, es apropiado que el prefijo IP del dominio incluya los bits comunes a todos los TRD conectados. Por ejemplo, si todas las TRD se encuentran totalmente dentro de los Estados Unidos, es apropiado un prefijo IP que implique un dominio exclusivamente norteamericano.
* El uso de un prefijo IP para cada TRD conectado con hosts en el dominio que tienen direcciones IP que contienen el prefijo IP del TRD más apropiado. La organización parece ser un conjunto de dominios de enrutamiento.
* Asignación de un prefijo IP desde uno de los TRD conectados. Este TRD se convierte en un TRD predeterminado para el dominio, pero otros dominios pueden enrutarse explícitamente por uno de los TRD alternativos.
* El uso de prefijos IP para hacer referencia a conjuntos de dominios multihost que tienen los adjuntos TRD. Por ejemplo, puede haber un prefijo IP para referirse a dominios de un solo host conectados a la red A, uno para hacer referencia a

dominios de un solo hogar conectados a la red B, y uno para hacer referencia a los dominios de doble alojamiento conectados a las redes A y B.

Cada uno de estos tiene varias ventajas, desventajas y efectos secundarios. Por ejemplo, el primer enfoque tiende a dar lugar a que el tráfico entrante entre en el dominio de destino más cerca del host de envío que el segundo enfoque.

Por lo tanto, la organización receptora incurre en una mayor proporción de los costos de red.

Debido a que los dominios multihost varían mucho en carácter, ninguno de estos esquemas es adecuado para todos los dominios. No hay una sola política que sea la mejor. RFC 1518 no especifica ninguna regla para elegir entre ellos.

##### Implementación de CIDR

La implementación de CIDR en Internet se basa principalmente en la versión 4 del protocolo de puerta de enlace fronteriza (consulte 5.9, "Protocolo de puerta de enlace fronteriza (BGP)" en la página 215). La estrategia de implementación, descrita en RFC 1520, implica un proceso por etapas a través de la jerarquía de enrutamiento que comienza con los enrutadores troncales. Los proveedores de servicios de red se dividen en cuatro tipos:

Tipo 1: Aquellos proveedores que no pueden emplear ningún enrutamiento entre dominios predeterminado.

Tipo 2: Aquellos proveedores que utilizan el enrutamiento entre dominios predeterminado pero requieren rutas explícitas para una proporción sustancial de los números de red IP asignados.

Tipo 3: Aquellos proveedores que utilizan el enrutamiento entre dominios predeterminado y lo complementan con un pequeño número de rutas explícitas.

Tipo 4: Aquellos proveedores que realizan enrutamiento entre dominios utilizando solo rutas predeterminadas.

La implementación de CIDR comenzó con los proveedores de red de Tipo 1, luego con los proveedores de Tipo 2 y finalmente con los proveedores de Tipo 3. El CIDR ya se ha desplegado ampliamente en la red troncal y más de 190.000 rutas basadas en clases han sido reemplazadas por aproximadamente 92.000 rutas basadas en CIDR (a través de agregados únicos anunciados).

#### 3.1.9 Datagrama IP

La unidad de transferencia en una red IP se denomina datagrama IP. Consta de un encabezado IP y datos relevantes para protocolos de nivel superior. Consulte la Figura 3-16 para obtener más detalles.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | encabezado | datos |   datagrama IP base...   |  |  | | --- | --- | | Encabezado de red física | Datagrama IP como datos |   encapsulado dentro del marco de la red física |

*Figura 3-16 IP: Formato de un datagrama IP base*

IP puede proporcionar fragmentación y reensamblaje de datagramas. La longitud máxima de un datagrama IP es de 65.535 octetos. Todos los hosts IP deben admitir datagramas de 576 octetos sin fragmentación.

Cada uno de los fragmentos de un datagrama tiene un encabezado. El encabezado se copia del datagrama original. Un fragmento se trata como un datagrama IP normal mientras se transporta a su destino. Sin embargo, si uno de los fragmentos se pierde, el datagrama completo se considera perdido. Dado que IP no proporciona ningún mecanismo de reconocimiento, el host de destino descarta los fragmentos restantes.

##### Formato de datagrama IP

El encabezado del datagrama IP tiene una longitud mínima de 20 octetos, como se ilustra en la Figura 3-17.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 1 2 3  0 4 8 6 9 4 1   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | VERS |  | HLEN | | Tipo de servicio | Total | | Largura | |  |  | IDENTIFICACIÓN | | | FLG | | Desplazamiento de fragmento | |  | TTL |  | Protocolo | |  | | Suma de comprobación de encabezado | |  |  | Anuncio IP de origen | | | | | vestido | |  |  | IP de destino | | | | | Dirección | |  |  | Opciones de IP | | | |  | Relleno | |  |  | Datos...  …  … | | | | |  | |

*Figura 3-17 IP: Formato de un encabezado de datagrama IP*

Dónde:

VERS: El campo contiene la versión del protocolo IP. La versión actual es la 4. La versión 5 es una versión experimental. La versión 6 es la versión para IPv6 (consulte 9.2, "El formato de encabezado IPv6" en la página 330).

HLEN: La longitud del encabezado IP contada en cantidades de 32 bits. Esto no incluye el campo de datos.

Tipo de servicio: El tipo de servicio es una indicación de la calidad de servicio solicitada para este datagrama IP. Este campo contiene la información que se ilustra en la Figura 3-18.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 1 2 3 4 5 6 7   |  |  |  | | --- | --- | --- | | precedencia | TÉRMINOS DE Servicio | MBZ | |

*Figura 3-18 IP: Tipo de servicio*

Dónde:

* Precedencia: Este campo especifica la naturaleza y la prioridad del datagrama:
  + 000: Rutina
  + 001: Prioridad
  + 010: Inmediato
  + 011: Destello
  + 100: Anulación de flash
  + 101: Crítico
  + 110: Control de la interconexión
  + 111: Control de red
* TOS: Especifica el tipo de valor de servicio:
  + 1000: Minimizar el retraso
  + 0100: Maximizar el rendimiento
  + 0010: Maximice la confiabilidad
  + 0001: Minimizar el costo monetario
  + 0000: Servicio normal

Una descripción detallada del tipo de servicio se encuentra en el RFC 1349 (consulte 8.1, "¿Por qué QoS?" en la página 288).

* MBZ: Reservado para uso futuro.

Longitud total: La longitud total del datagrama, el encabezado y los datos.

Identificación: Un número único asignado por el remitente para ayudar a volver a ensamblar un datagrama fragmentado. Cada fragmento de un datagrama tiene el mismo número de identificación.

Indicadores: Este campo contiene indicadores de control que se ilustran en la Figura 3-19.

0

0

1

2

D

F

M

F

*Figura 3-19 IP: Banderas*

Dónde:

* 0: Reservado, debe ser cero.
* DF (No fragmentar): 0 significa permitir la fragmentación; 1 significa que no permiten la fragmentación.
* MF (Más fragmentos): 0 significa que este es el último fragmento del datagrama; 1 significa que seguirán fragmentos adicionales.

Desplazamiento de fragmentos: Se utiliza para ayudar a volver a ensamblar el datagrama completo. El valor de este campo contiene el número de segmentos de 64 bits (no se cuentan los bytes de encabezado) contenidos en fragmentos anteriores. Si se trata del primer (o único) fragmento, este campo contiene un valor de cero.

Tiempo de vida: Este campo especifica el tiempo (en segundos) que el datagrama puede viajar. Teóricamente, se supone que cada router que procesa este datagrama resta su tiempo de procesamiento de este campo. En la práctica, un router procesa el datagrama en menos de 1 segundo. Por lo tanto, el router resta uno del valor de este campo. El TTL se convierte en una métrica de recuento de saltos en lugar de una métrica de tiempo. Cuando el valor llega a cero, se supone que este datagrama ha estado viajando en un bucle cerrado y se descarta. El valor inicial debe ser establecido por el protocolo de nivel superior que crea el datagrama.

Número de protocolo: Este campo indica el protocolo de nivel superior al que IP debe entregar los datos en este datagrama. Entre ellas se encuentran:

* 0: Reservado
* 1: Protocolo de mensajes de control de Internet (ICMP)
* 2: Protocolo de administración de grupos de Internet (IGMP)
* 3: Protocolo de puerta de enlace a puerta de enlace (GGP)
* 4: IP (encapsulación IP)
* 5: Transmisión
* 6: Protocolo de control de transmisión (TCP)
* 8: Protocolo de puerta de enlace exterior (EGP)
* 9: Protocolo de puerta de enlace interior (IGP)
* 17: Protocolo de datagramas de usuario (UDP)
* 41: Protocolo simple de Internet (SIP)
* 50: Carga útil de seguridad (ESP) de SIPP Encap
* 51: Encabezado de autenticación SIPP (AH)
* 89: Abrir primero el IGP de la ruta más corta (OSPF)

La lista completa se encuentra en STD 2 – Números de Internet asignados.

Suma de comprobación del encabezado: Este campo es una suma de comprobación de la información contenida en el encabezado. Si la suma de comprobación del encabezado no coincide con el contenido, el datagrama se descarta.

Dirección IP de origen: La dirección IP de 32 bits del host que envía este datagrama. Dirección IP de destino: la dirección IP de 32 bits del host de destino para este datagrama.

Opciones: No se requiere una implementación de IP para ser capaz de generar opciones en un datagrama. Sin embargo, todas las implementaciones de IP son necesarias para poder procesar datagramas que contienen opciones. El campo Opciones tiene una longitud variable (puede haber cero o más opciones). Hay dos formatos de opción. El formato de cada uno depende del valor del número de opción que se encuentra en el primer octeto:

* En la Figura 3-20 se ilustra un octeto de tipo solo.

|  |  |
| --- | --- |
| |  | | --- | | tipo |   1 byte |

*Figura 3-20 IP: Un byte de tipo*

* Un octeto de tipo, un octeto de longitud y uno o más octetos de datos de opción, como se ilustra en la Figura 3-21.

tipo

largura

datos de la opción...

/ /

/ /

1

byte

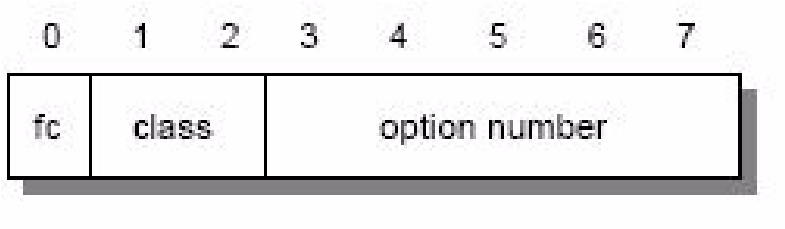
Longitud - 2 bytes

1

byte

*Figura 3-21 IP: Un byte de tipo, un byte de longitud y uno o más bytes de datos de opción*

El byte de tipo tiene la misma estructura en ambos casos, como se ilustra en la figura 3-22.



*Figura 3-22 IP: La estructura de bytes de tipo*

Dónde:

* fc (Flag copy): Este campo indica si (1) o no (0) se copia el campo de opción cuando el datagrama está fragmentado.
* class: La clase de opción es un entero sin signo de 2 bits:
  + 0: Control
  + 1: Reservado
  + 2: Depuración y medición
  + 3: Reservado
* número de opción: El número de opción es un entero sin signo de 5 bits:
  + 0: Fin de la lista de opciones. Tiene una clase de 0, el bit fc se establece en cero y no tiene byte de longitud ni datos. Es decir, la lista de opciones termina con un byte X'00'. Solo es necesario si la longitud del encabezado IP (que es un múltiplo de 4 bytes) no coincide con la longitud real de las opciones.
  + 1: Sin operación. Tiene una clase de 0, el bit fc no está establecido y no hay byte de longitud ni datos. Es decir, un byte X'01' es un NOP. Se puede utilizar para alinear campos en el datagrama.
  + 2: Seguridad. Tiene una clase de 0, se establece el bit fc y hay un byte de longitud con un valor de 11 y 8 bytes de datos). Se utiliza para la información de seguridad necesaria para los requisitos del Departamento de Defensa de EE. UU.
  + 3: Enrutamiento de origen suelto. Tiene una clase de 0, el bit fc está establecido y hay un campo de datos de longitud variable. Discutiremos esta opción con más detalle más adelante.
  + 4: Marca de tiempo de Internet. Tiene una clase de 2, el bit fc no está configurado y hay un campo de datos de longitud variable. La longitud total puede ser de hasta 40 bytes. Discutiremos esta opción con más detalle más adelante.
  + 7: Ruta de grabación. Tiene una clase de 0, el bit fc no está establecido y hay un campo de datos de longitud variable. Discutiremos esta opción con más detalle más adelante.
  + 8: ID de transmisión. Tiene una clase de 0, el bit fc está establecido y hay un byte de longitud con un valor de 4 y un byte de datos. Se utiliza con el sistema SATNET.
  + 9: Enrutamiento estricto de la fuente. Tiene una clase de 0, el bit fc está establecido y hay un campo de datos de longitud variable. Discutiremos esta opción con más detalle más adelante.
* length: Este campo cuenta la longitud (en octetos) de la opción, incluidos los campos type y length.
* datos de opción: Este campo contiene datos relevantes para la opción específica.

Relleno: Si se utiliza una opción, el datagrama se rellena con octetos de cero hasta el siguiente límite de 32 bits.

Datos: Los datos contenidos en el datagrama. Se pasa al protocolo de nivel superior especificado en el campo protocolo.

##### Fragmentación

Cuando un datagrama IP viaja de un host a otro, puede pasar a través de diferentes redes físicas. Cada red física tiene un tamaño máximo de fotograma. Esto se denomina *unidad máxima de transmisión* (MTU). Limita la longitud de un datagrama que se puede colocar en un marco físico.

IP implementa un proceso para fragmentar datagramas que exceden la MTU. El proceso crea un conjunto de datagramas dentro del tamaño máximo. El host receptor vuelve a ensamblar el datagrama original. IP requiere que cada enlace admita una MTU mínima de 68 octetos. Es la suma de la longitud máxima del encabezado IP (60 octetos) y la longitud mínima posible de los datos en un fragmento no final (8 octetos). Si alguna red proporciona un valor inferior a este, la fragmentación y el reensamblaje deben implementarse en la capa de interfaz de red. Esto debe ser transparente para la P.I. Las implementaciones IP no son necesarias para manejar datagramas no fragmentados de más de 576 bytes. En la práctica, la mayoría de las implementaciones se adaptarán a valores más grandes.

Un datagrama no fragmentado tiene un campo de información de fragmentación totalmente cero. Es decir, el bit de indicador de más fragmentos es cero y el desplazamiento del fragmento es cero. Los siguientes pasos fragmentan el datagrama:

1. El bit de indicador DF se comprueba para ver si se permite la fragmentación. Si se establece el bit, el datagrama se descartará y se devolverá un error ICMP al originador.
2. En función del valor de MTU, el campo de datos se divide en dos o más partes. Todas las porciones de datos recién creadas deben tener una longitud que sea un múltiplo de 8 octetos, con la excepción de la última porción de datos.
3. Cada parte de datos se coloca en un datagrama IP. Los encabezados de estos datagramas son modificaciones menores del original:
   * El bit de indicador de más fragmentos se establece en todos los fragmentos excepto en el último.
   * El campo de desplazamiento de fragmento de cada uno se establece en la ubicación que ocupaba esta parte de datos en el datagrama original, en relación con el principio del datagrama original no fragmentado. El desplazamiento se mide en unidades de 8 octetos.
   * Si las opciones se incluyeron en el datagrama original, el bit de orden superior del byte de tipo de opción determina si esta información se copia en todos los datagramas de fragmento o solo en el primer datagrama. Por ejemplo, las opciones de ruta de origen se copian en todos los fragmentos.
   * Se establece el campo de longitud del encabezado del nuevo datagrama.
   * Se establece el campo de longitud total del nuevo datagrama.
   * Se vuelve a calcular el campo de suma de comprobación del encabezado.
4. Cada uno de estos datagramas fragmentados ahora se reenvía como un datagrama IP normal. IP maneja cada fragmento de forma independiente. Los fragmentos pueden atravesar diferentes enrutadores hasta el destino previsto. Pueden estar sujetos a una mayor fragmentación si pasan a través de redes que especifican una MTU más pequeña.

En el host de destino, los datos se vuelven a ensamblar en el datagrama original. El campo de identificación establecido por el host de envío se utiliza junto con las direcciones IP de origen y destino en el datagrama. La fragmentación no altera este campo.

Para volver a ensamblar los fragmentos, el host receptor asigna un búfer de almacenamiento cuando llega el primer fragmento. El anfitrión también inicia un temporizador. Cuando llegan los fragmentos posteriores del datagrama, los datos se copian en el almacenamiento del búfer en la ubicación indicada por el campo de desplazamiento del fragmento. Cuando han llegado todos los fragmentos, se restaura el datagrama original completo sin fragmentar. El procesamiento continúa como para los datagramas no fragmentados.

Si se excede el temporizador y quedan fragmentos pendientes, el datagrama se descarta. El valor inicial de este temporizador se denomina valor de tiempo de vida (TTL) del datagrama IP. Depende de la implementación. Algunas implementaciones permiten configurarlo.

El comando **netstat** se puede utilizar en algunos hosts IP para enumerar los detalles de la fragmentación.

##### Opciones de enrutamiento de datagramas IP

El campo Opciones de datagrama IP proporciona dos métodos para que el originador de un datagrama IP proporcione explícitamente información de enrutamiento. También proporciona un método para que un datagrama IP determine la ruta que recorre.

###### Enrutamiento de origen suelto

La opción de enrutamiento de origen suelto, también denominada opción de enrutamiento de origen suelto y registro (LSRR), proporciona un medio para que el origen de un datagrama IP proporcione información de enrutamiento explícita. Esta información es utilizada por los routers al reenviar el datagrama al destino. También se utiliza para registrar la ruta, como se ilustra en la Figura 3-23.

10000011

largura

puntero

Datos de ruta

//

/

/

*Figura 3-23 IP: Opción de enrutamiento de origen suelto*

Los campos de este encabezado incluyen:

**1000011(decimal 131)** Este es el valor del tipo de opción octeto para el enrutamiento de origen suelto.

**Largura** Este campo contiene la longitud de este campo de opción,

incluidos los campos de tipo y longitud.

|  |  |
| --- | --- |
| **Puntero** | Este campo apunta a los datos de la opción en la siguiente dirección IP que se va a procesar. Se cuenta en relación con el inicio de la opción, por lo que su valor mínimo es cuatro. Si el puntero es mayor que la longitud de la opción, se alcanza el final de la ruta de origen y el enrutamiento posterior se basará en la dirección IP de destino (como en el caso de los datagramas sin esta opción). |
| **Datos de ruta** | Este campo contiene una serie de direcciones IP de 32 bits. |

Cuando un datagrama llega a su destino y la ruta de origen no está vacía (puntero < longitud), el host receptor:

1. Toma la siguiente dirección IP en el campo de datos de ruta (la indicada por el campo de puntero) y la coloca en el campo de dirección IP de destino del datagrama.
2. Coloca la dirección IP local en la lista de origen en la ubicación a la que apunta el campo de puntero. La dirección IP para esto es la dirección IP local correspondiente a la red en la que se reenviará el datagrama. (Los enrutadores están conectados a varias redes físicas y, por lo tanto, tienen varias direcciones IP).
3. Incrementa el puntero en 4.
4. Transmite el datagrama a la nueva dirección IP de destino.

Este procedimiento garantiza que la ruta de retorno se registre en los datos de ruta (en orden inverso) para que el destinatario final utilice estos datos para construir una ruta de origen suelta en la dirección inversa. Esta es una ruta de origen flexible porque el router de reenvío puede utilizar cualquier ruta y cualquier número de routers intermedios para llegar a la siguiente dirección de la ruta.

###### Enrutamiento estricto de origen

La opción de enrutamiento de origen estricto, también denominada opción de ruta de origen y registro estricto (SSRR), utiliza el mismo principio que el enrutamiento de origen suelto, excepto que el enrutador intermedio *debe* enviar el datagrama a la siguiente dirección IP en la ruta de origen a través de una red conectada directamente. No puede utilizar un router intermedio. Si esto no se puede hacer, se emite un mensaje de error ICMP Destination Unreachable. La Figura 3-24 ofrece una descripción general de la opción SSRR.

*Figura 3-24 IP: Opción de enrutamiento de origen estricto*

100001001

largura

puntero

Datos de ruta

//

/

/

Dónde:

**1001001 (Decimal 137)** El valor del byte de tipo de opción para el enrutamiento de origen estricto.

**Longitud** Esta información se describe en "Enrutamiento de origen suelto" en la página 105.

**Puntero** Esta información se describe en "Enrutamiento de origen suelto" en la página 105.

**Datos de ruta** Una serie de direcciones IP de 32 bits.

###### Ruta de registro

Esta opción proporciona un medio para registrar la ruta recorrida por un datagrama IP. Funciona de manera similar a la opción de enrutamiento de origen. Sin embargo, esta opción proporciona un campo de datos de enrutamiento vacío. Este campo se rellena a medida que el datagrama atraviesa la red. El host de origen debe proporcionar suficiente espacio para esta información de enrutamiento. Si el campo de datos se rellena antes de que el datagrama llegue a su destino, el datagrama se reenvía sin más registro de la ruta. La Figura 3-25 ofrece una descripción general de la opción de ruta de registro.

00000111

largura

puntero

Datos de ruta

//

/

/

*Figura 3-25 IP: Opción de ruta de registro*

Dónde:

|  |  |
| --- | --- |
| **0000111 (Decimal 7)** | El valor del tipo de opción byte para la ruta de registro |
| **Largura** | Esta información se describe en "Enrutamiento de origen suelto" en la página 105. |
| **Puntero** | Esta información se describe en "Enrutamiento de origen suelto" en la página 105. |
| **Datos de ruta** | Una serie de direcciones IP de 32 bits. |

##### Marca de tiempo de Internet

Una marca de tiempo es una opción que obliga a algunos (o todos) de los routers a lo largo de la ruta al destino a poner una marca de tiempo en los datos de la opción. Las marcas de tiempo son

se mide en segundos y se puede utilizar con fines de depuración. No se pueden utilizar para la medición del rendimiento por dos razones:

Debido a que la mayoría de los datagramas IP se reenvían en menos de un segundo, las marcas de tiempo no son precisas.

Debido a que no se requiere que los enrutadores IP tengan relojes sincronizados, es posible que no sean precisos.

La Figura 3-26 ofrece una descripción general de la opción de marca de tiempo de Internet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 8 16 24 28   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 01000100 | largura |  | puntero | OFLW | bandera | |  | Dirección IP | |  |  | | |  | Tiempo st | amperio |  |  | | |  | ... |  |  |  | | |

*Figura 3-26 IP: Opción de marca de tiempo de Internet*

Dónde:

|  |  |  |
| --- | --- | --- |
| **01000100 (Decimal 68)** | | Este campo es el valor del tipo de opción para la opción de marca de tiempo de Internet. |
| **Largura** |  | Este campo contiene la longitud total de esta opción, incluidos los campos de tipo y longitud. |
| **Puntero** |  | Este campo apunta a la siguiente marca de tiempo que se va a procesar (primera marca de tiempo libre). |
| **Oflw (desbordamiento)** |  | Este campo contiene el número de dispositivos que no pueden registrar marcas de tiempo debido a la falta de espacio en el campo de datos. |
| **Bandera** |  | Este campo es un valor de 4 bits que indica cómo se deben registrar las marcas de tiempo:   1. Solo marcas de tiempo, almacenadas en palabras consecutivas de 32 bits. 2. Cada marca de tiempo va precedida de la dirección IP del dispositivo de registro. 3. Los campos de dirección IP están preespecificados; un dispositivo IP solo se registra cuando encuentra su propia dirección en la lista. |
| **Marca de tiempo** | Marca de tiempo de 32 bits registrada en milisegundos desde la medianoche UT (GMT). | |

El host de origen debe componer esta opción con un área de datos suficiente para contener todas las marcas de tiempo. Si el área de marca de tiempo se llena, no se agregan más marcas de tiempo.

### 3.2 Protocolo de mensajes de control de Internet (ICMP )

ICMP es un protocolo estándar con STD número 5. Esa norma también incluye la propiedad intelectual

(consulte 3.1, "Protocolo de Internet (IP)" en la página 68) e IGMP (consulte 6.2, "Grupo de Internet")

Protocolo de Gestión (IGMP)" en la página 241). Su estado es obligatorio. Se describe en RFC 792 con actualizaciones en RFC 950. ICMPv6 utilizado para IPv6 se describe en 9.3, "Protocolo de mensajes de control de Internet versión 6 (ICMPv6)" en la página 352.

Path MTU Discovery es un borrador de protocolo estándar con un estado de optativo. Se describe en RFC 1191.

ICMP Router Discovery es un protocolo estándar propuesto con un estado de optativo. Se describe en RFC 1256.

Cuando un router o un host de destino debe informar al host de origen sobre errores en el procesamiento de datagramas, utiliza el Protocolo de mensajes de control de Internet (ICMP). ICMP se puede caracterizar de la siguiente manera:

ICMP utiliza IP como si ICMP fuera un protocolo de nivel superior (es decir, los mensajes ICMP se encapsulan en datagramas IP). Sin embargo, ICMP es una parte integral de IP y debe ser implementado por todos los módulos de IP.

ICMP se utiliza para informar de errores, *no* para hacer que la IP sea fiable. Los datagramas aún pueden ser anulados sin ningún informe sobre su pérdida. La confiabilidad debe ser implementada por los protocolos de nivel superior que utilizan servicios IP.

ICMP no se puede utilizar para notificar errores con mensajes ICMP. Esto evita repeticiones infinitas. Las respuestas ICMP se envían en respuesta a los mensajes de consulta ICMP (tipos ICMP 0, 8, 9, 10 y 13 a 18).

En el caso de los datagramas fragmentados, los mensajes ICMP solo se envían sobre errores con el primer fragmento. Es decir, los mensajes ICMP nunca hacen referencia a un datagrama IP con un campo de desplazamiento de fragmento distinto de cero.

Los mensajes ICMP nunca se envían en respuesta a datagramas con una dirección de destino de difusión o multidifusión.

Los mensajes ICMP nunca se envían en respuesta a un datagrama que no tiene una dirección IP de origen que represente un host único. Es decir, la dirección de origen no puede ser cero, una dirección de bucle invertido, una dirección de difusión o una dirección de multidifusión.

RFC 792 establece que los mensajes ICMP *enlatar* para informar de errores de procesamiento de datagramas IP. Sin embargo, esto no es obligatorio. En la práctica, los routers casi siempre generan mensajes ICMP para errores. En el caso de los hosts de destino, la generación de mensajes ICMP depende de la implementación.

#### 3.2.1 Mensajes ICMP

Los mensajes ICMP se describen en RFC 792 y RFC 950, pertenecen a STD 5 y son obligatorias.

Los mensajes ICMP se envían en datagramas IP. El encabezado IP tiene un número de protocolo de 1 (ICMP) y un tipo de servicio de cero (rutina). El campo de datos IP contiene el mensaje ICMP que se muestra en la Figura 3-27.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  |  |  | | --- | --- | --- | --- | | identificador | Número de secuencia |  | checksum | | Datos ICMP (dependiendo del tipo de | | sabio) |  | |

*Figura 3-27 ICMP: Formato del mensaje*

El mensaje contiene los siguientes componentes:

**Tipo** Especifica el tipo de mensaje:

**0** Respuesta de eco

1. Destino inalcanzable
2. Enfriamiento en fuente
3. Redirigir
4. Eco
5. Anuncio del router
6. Solicitud de enrutador
7. Tiempo excedido
8. Problema de parámetros
9. Solicitud de marca de tiempo
10. Respuesta de marca de tiempo
11. Solicitud de máscara de dirección
12. Respuesta de máscara de dirección

**30** Traceroute

1. Solicitud de nombre de dominio)
2. Respuesta de nombre de dominio)

Es necesario mencionar las siguientes RFC para algunos de los tipos de mensajes ICMP: RFC 1256, RFC 1393 y RFC 1788.

|  |  |
| --- | --- |
| **Código** | Contiene el código de error para el datagrama notificado por este  Mensaje ICMP. La interpretación depende de la |
| **Checksum** | tipo de mensaje.    Contiene la suma de comprobación del mensaje ICMP que comienza con el campo Tipo ICMP. Si la suma de comprobación no coincide con el contenido, el datagrama se descarta. |

**Datos** Contiene información para este mensaje ICMP. Normalmente,

contendrá la parte del mensaje IP original para el que se generó este mensaje ICMP.

Cada uno de los mensajes ICMP se describe individualmente.

##### Eco (8) y Eco Respuesta (0)

El eco se utiliza para detectar si hay otro host activo en la red. Es utilizado por el comando Ping (consulte "Ping" en la página 117). El remitente inicializa el identificador, el número de secuencia y el campo de datos. A continuación, el datagrama se envía al host de destino. El destinatario cambia el tipo a Respuesta de eco y devuelve el datagrama al remitente. Consulte la Figura 3-28 para obtener más detalles.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  | | --- | --- | | identificador | Número de secuencia | |  | datos... | |

*Figura 3-28 Eco y respuesta de eco*

##### Destino inalcanzable (3)

Si este mensaje se recibe de un router intermedio, significa que el router considera que la dirección IP de destino es inalcanzable.

Si este mensaje se recibe del host de destino, significa que el protocolo especificado en el campo de número de protocolo del datagrama original no está activo o que el puerto especificado está inactivo. (Consulte 4.2, "Protocolo de datagramas de usuario (UDP)" en la página 146 para obtener información adicional sobre los puertos). Consulte la Figura 3-29 para obtener más detalles.

|  |  |  |
| --- | --- | --- |
| 0 8 16 31   |  | | --- | | sin usar (cero) | | Encabezado IP - 64 bits de datos originales del datagrama | |

*Figura 3-29 ICMP: Destino inalcanzable*

El campo de código de encabezado ICMP contiene uno de los siguientes valores:

1. Red inalcanzable
2. Host inalcanzable
3. Protocolo inalcanzable
4. Puerto inalcanzable
5. Se necesita fragmentación, pero se estableció el bit No fragmentar
6. Error en la ruta de origen
7. Red de destino desconocida
8. Host de destino desconocido
9. Host de origen aislado (obsoleto)
10. La red de destino está prohibida administrativamente
11. Host de destino prohibido administrativamente
12. Red inaccesible para este tipo de servicio
13. Host inaccesible para este tipo de servicio
14. Comunicación prohibida administrativamente por filtrado
15. Violación de la precedencia del host
16. Límite de precedencia en vigor

Estos se detallan en RFC 792, RFC 1812 actualizado por RFC 2644, RFC 1122, actualizado por RFC 4379 y forma parte de STD 3 - Requisitos de host.

Si un router implementa el protocolo de detección de MTU de trayectoria, el formato del mensaje de destino inalcanzable se cambia para el código 4. Esto incluye la MTU del enlace que no aceptó el datagrama. Consulte la Figura 3-30 para obtener más información.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  | | --- | --- | | sin usar (cero) | enlace MTU | | Encabezado IP - 64 bits de datos originales del datagrama | | |

*Figura 3-30 ICMP: Fragmentación requerida con MTU de link*

##### Fuente Quench (4)

Si este mensaje se recibe de un router intermedio, significa que el router no tenía el espacio de búfer necesario para poner en cola el datagrama.

Si este mensaje se recibe del host de destino, significa que los datagramas entrantes están llegando demasiado rápido para ser procesados.

El campo de código de encabezado ICMP siempre es cero.

Consulte la Figura 3-31 para obtener más detalles.

|  |  |  |
| --- | --- | --- |
| 0 8 16 31   |  | | --- | | sin usar (cero) | | Encabezado IP - 64 bits de datos originales del datagrama | |

*Figura 3-31 ICMP: Enfriamiento de fuente*

##### Redirección (5)

Si este mensaje se recibe de un router intermedio, significa que el host debe enviar datagramas futuros para la red al router cuya dirección IP se especifica en el mensaje ICMP. Este enrutador preferido siempre estará en la misma subred que el host que envió el datagrama y el enrutador que devolvió el datagrama IP. El router reenvía el datagrama a su destino de salto siguiente. Este mensaje no se enviará si el datagrama IP contiene una ruta de origen.

El campo de código de encabezado ICMP tendrá uno de los siguientes valores:

1. Redireccionamiento de red
2. Redireccionamiento de host
3. Redireccionamiento de red para este tipo de servicio **3** Redireccionamiento de host para este tipo de servicio Consulte la Figura 3-32 para obtener más detalles.

|  |  |  |
| --- | --- | --- |
| 0 8 16 31   |  | | --- | | Dirección IP del router | | Encabezado IP - 64 bits de datos originales del datagrama | |

*Figura 3-32 ICMP: Redireccionamiento*

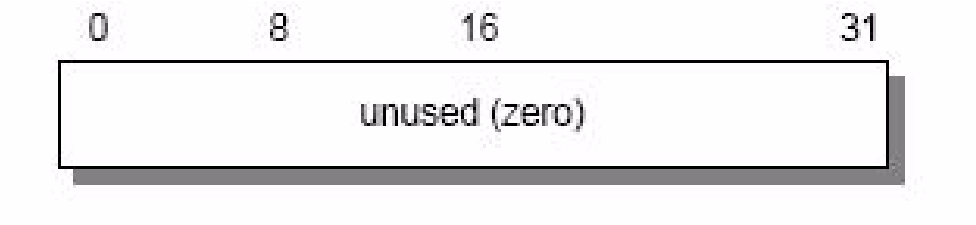
##### Anuncio de enrutador (9) y solicitud de enrutador (10)

Los mensajes ICMP 9 y 10 son opcionales. Se describen en RFC 1256, que es optativa. Consulte la Figura 3-33 y la Figura 3-34 en la página 114 para obtener más detalles.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 8 16 31 | | | | | | | |
|  |  | |  |  | |  | |
|  | número | Longitud de entrada | TTL | |  |  |
|  |  | | | |
| Dirección del router 1 | | | |  |
|  |  | | | |
| Nivel de preferencia 1 | | | |  |
|  |  | | | |
|  | | | |  |
| / | / |  | | | / | / |
|  |  |  |  |
|  | | | | |
|  | dirección del router m— | | | |  |
|  | nivel de preferencia m— | | | |
|  |  | | | |
|  | | | | |

*Figura 3-33 ICMP: Anuncio del router*

*Figura 3-34 ICMP: Solicitud de enrutador*



Dónde:

|  |  |
| --- | --- |
| **Número** | El número de entradas en el mensaje. |
| **Longitud de entrada** | Longitud de una entrada en unidades de 32 bits. Esto es 2 (32 bits para la dirección IP y 32 bits para el valor de preferencia). |
| **TTL** | El número de segundos que una entrada se considerará válida. |
| **Dirección del router** | Una de las direcciones IP del remitente. |
| **Nivel de preferencia** | Un nivel de 32 bits con signo que indica la preferencia que se asignará a esta dirección al seleccionar un router predeterminado. Cada enrutador de una subred es responsable de anunciar su propio nivel de preferencia. Los valores más grandes implican una preferencia más alta; Los valores más pequeños implican más bajos. El valor predeterminado es cero, que se encuentra en el medio del rango posible. Un valor de X'80000000' (-231) indica que el router nunca debe utilizarse como router predeterminado. |

El campo de código de encabezado ICMP es cero para ambos mensajes.

Estos dos mensajes se utilizan si un host o un router admite el protocolo de detección de routers. Los enrutadores anuncian periódicamente sus direcciones IP en las subredes en las que están configurados para hacerlo. Los anuncios se realizan en la dirección de multidifusión de todos los sistemas (224.0.0.1) o en la dirección de difusión limitada

(255.255.255.255). El comportamiento predeterminado es enviar anuncios cada 10 minutos con un valor TTL de 1800 (30 minutos). Los routers también responden a los mensajes de solicitud que reciben. Pueden responder directamente al host solicitante, o pueden esperar un breve intervalo aleatorio y responder con una multidifusión.

Los anfitriones pueden enviar mensajes de solicitud. Los mensajes de solicitud se envían a la dirección de multidifusión de todos los routers (224.0.0.2) o a la dirección de difusión limitada (255.255.255.255). Normalmente, se envían tres mensajes de solicitud a intervalos de 3 segundos. Alternativamente, un host puede esperar anuncios periódicos. Cada vez que un host recibe un anuncio con un valor de preferencia más alto, actualiza su enrutador predeterminado. El host también establece el temporizador TTL para que la nueva entrada coincida con el valor del anuncio. Cuando el host recibe un nuevo anuncio para su enrutador predeterminado actual, restablece el valor TTL al del nuevo anuncio.

Este proceso también proporciona un mecanismo para que los routers se declaren no disponibles. Envían un anuncio con un valor TTL de cero.

##### Tiempo Excedido (11)

Si este mensaje se recibe de un router intermedio, significa que el campo de tiempo de vida de un datagrama IP ha expirado.

Si este mensaje se recibe del host de destino, significa que el temporizador de tiempo de reensamblado del fragmento IP para vivir ha expirado mientras el host espera un fragmento del datagrama. El campo de código de encabezado ICMP puede tener uno de los siguientes valores:

1. TTL de tránsito excedido
2. TTL de reensamblaje excedido

Consulte la Figura 3-35 para obtener más detalles.

|  |  |  |
| --- | --- | --- |
| 0 8 16 31   |  | | --- | | sin usar (cero) | | Encabezado IP - 64 bits de datos originales del datagrama | |

*Figura 3-35 ICMP: Tiempo excedido*

##### Problema de parámetros (12)

Este mensaje indica que se ha encontrado un problema durante el procesamiento de los parámetros de cabecera IP. El campo de puntero indica el octeto en el datagrama IP original donde se encontró el problema. El campo de código de encabezado ICMP puede tener uno de los siguientes valores:

1. Error no especificado
2. Falta la opción requerida

Consulte la Figura 3-36 para obtener más detalles.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  | | --- | --- | | puntero | sin usar (cero) | | Encabezado IP - 64 bits de datos originales del datagrama | | |

*Figura 3-36 ICMP: Problema de parámetros*

##### Solicitud de marca de tiempo (13) y respuesta de marca de tiempo (14)

Estos dos mensajes son para la depuración y las mediciones de rendimiento. No se utilizan para la sincronización del reloj.

El remitente inicializa el identificador y el número de secuencia (que se utiliza si se envían varias solicitudes de marca de tiempo), establece la marca de tiempo de origen y envía el datagrama al destinatario. El host receptor rellena las marcas de tiempo de recepción y transmisión, cambia el tipo a respuesta de marca de tiempo y lo devuelve al remitente original. El datagrama tiene dos marcas de tiempo si hay una diferencia de tiempo perceptible entre los tiempos de recepción y transmisión. En la práctica, la mayoría de las implementaciones realizan los dos (recepción y respuesta) en una sola operación. De este modo, las dos marcas de tiempo se establecen en el mismo valor. Las marcas de tiempo son el número de milisegundos transcurridos desde la medianoche UT (GMT).

Consulte la Figura 3-37 para obtener más detalles.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  | | --- | --- | | identificador | Número de secuencia | | Marca de tiempo de origen | | | Recibir marca de tiempo | | | Marca de tiempo de transmisión | | |

*Figura 3-37 ICMP: Solicitud de marca de tiempo y respuesta de marca de tiempo*

##### Solicitud de máscara de dirección (17) y respuesta de máscara de dirección (18)

Un host utiliza una solicitud de máscara de dirección para determinar la máscara de subred utilizada en una red conectada. La mayoría de los hosts se configuran con su máscara o máscaras de subred. Sin embargo, algunas, como las estaciones de trabajo sin disco, deben obtener esta información de un servidor. Un host utiliza RARP (consulte 3.5, "Protocolo de resolución de direcciones inversas (RARP)" en la página 124) para obtener su dirección IP. Para obtener una máscara de subred, el host difunde una solicitud de máscara de dirección. Cualquier host de la red que se haya configurado para enviar respuestas de máscara de dirección rellenará la máscara de subred, convertirá el paquete en una respuesta de máscara de dirección y lo devolverá al remitente. El campo de código de encabezado ICMP es cero.

Consulte la Figura 3-38 en la página 117 para obtener más detalles.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  | | --- | --- | | identificador | Número de secuencia | | máscara de dirección de subred | | |

*Figura 3-38 ICMP: Solicitud y respuesta de máscara de dirección*

#### 3.2.2 Aplicaciones ICMP

Existen dos aplicaciones sencillas y ampliamente utilizadas basadas en ICMP: Ping y Traceroute. Ping utiliza los mensajes ICMP Echo y Echo Reply para determinar si se puede acceder a un host. Traceroute envía datagramas IP con valores TTL bajos para que caduquen en el camino a un destino. Utiliza los mensajes ICMP Time Exceeded resultantes para determinar en qué parte de Internet caducaron los datagramas y reconstruye una vista de la ruta a un host. Analizamos estas aplicaciones en las siguientes secciones.

##### Señal

Ping es la más simple de todas las aplicaciones TCP/IP. Envía datagramas IP a un host de destino especificado y mide el tiempo de ida y vuelta para recibir una respuesta. La palabra *ping*, que se utiliza como sustantivo y verbo, se toma de la operación de sonar para localizar un objeto submarino. También es una abreviatura de *Packet InterNet Groper*.

Por lo general, la primera prueba de accesibilidad de un host es intentar hacer ping a él. Si puede hacer ping correctamente a un host, otras aplicaciones como Telnet o FTP deberían poder llegar a ese host. Sin embargo, con el advenimiento de las medidas de seguridad en Internet, en particular los cortafuegos (véase 22.3, "Cortafuegos" en la página 794), que controlan el acceso a las redes por protocolo de aplicación o número de puerto, o ambos, esto ya no es necesariamente cierto. El protocolo ICMP se puede restringir en el firewall y, por lo tanto, no se puede hacer ping correctamente al host.

La sintaxis que se utiliza en las diferentes implementaciones de ping varía de una plataforma a otra. Un formato común para usar el comando ping es: ping host

Donde host es el destino, ya sea un nombre simbólico o una dirección IP.

La mayoría de las plataformas permiten especificar los siguientes valores:

|  |  |
| --- | --- |
| **Tamaño** | El tamaño de la parte de datos del paquete. |
| **Paquetes** | El número de paquetes que se van a enviar. |
| **Contar** | El número de solicitudes de eco que se van a enviar. |
| **Rutas de registro** | Registre la ruta por salto de recuento. | |
| **Marca de tiempo** | Marca de tiempo cada salto de recuento. | |
| **Ping infinito** | Haga ping hasta que se detenga manualmente. | |
| **Resolver dirección** | Resuelva la dirección de host en el nombre de host. | |
| **Tiempo de vida (TTL)** | El tiempo (en segundos) que el datagrama puede viajar. | |

**Tipo de servicio (TOS)** El tipo de calidad del servicio de Internet.

**Lista de anfitriones** Ruta de origen suelta o ruta de origen estricta de listas de hosts.

**Interrupción** El tiempo de espera para esperar cada respuesta. **Sin fragmentación** El indicador de fragmento no está establecido.

Ping utiliza los mensajes ICMP Echo y Echo Reply (consulte "Echo (8) y Echo Reply (0)" en la página 111). Debido a que ICMP es necesario en todas las implementaciones de TCP/IP, los hosts no requieren un servidor independiente para responder a las solicitudes de ping.

El ping es útil para verificar una instalación IP. Cada una de las siguientes variaciones del comando requiere el funcionamiento de una parte diferente de una instalación IP:

ping loopback: Verifica el funcionamiento del software TCP/IP base.

ping my-IP-address: Verifica si se puede direccionar el dispositivo de red físico.

ping a-remote-IP-address: Verifica si se puede acceder a la red.

ping a-remote-host-name: Verifica el funcionamiento del servidor de nombres (o del solucionador de espacio de nombres plano, según la instalación).

##### Traceroute (Ruta de seguimiento)

El programa Traceroute se utiliza para determinar la ruta que siguen los datagramas IP a través de la red.

Traceroute se basa en ICMP y UDP. Envía un datagrama IP con un TTL de 1 al host de destino. El primer router disminuye el TTL a 0, descarta el datagrama y devuelve un mensaje ICMP Time Exceeded al origen. De esta manera, se identifica el primer router de la ruta. Este proceso se repite con valores TTL sucesivamente mayores para identificar la serie exacta de enrutadores en la ruta al host de destino.

Traceroute envía datagramas UDP al host de destino. Estos datagramas hacen referencia a un número de puerto fuera del rango estándar. Cuando se recibe un mensaje de puerto ICMP inalcanzable, el origen determina que se ha alcanzado el host de destino.

### 3.3 Protocolo de administración de grupos de Internet (IGMP )

IGMP es un protocolo estándar con número STD 5. Esa norma también incluye IP (véase 3.1, "Protocolo de Internet (IP)" en la página 68) e ICMP (véase 3.2, "Internet

Protocolo de mensajes de control (ICMP)" en la página 109). Se recomienda su estado. Se describe en RFC 1112 con actualizaciones en RFC 2236.

Al igual que ICMP, el Protocolo de administración de grupos de Internet (IGMP) también es una parte integral de IP. Permite a los hosts participar en multidifusiones IP. IGMP proporciona además a los routers la capacidad de comprobar si algún host de una subred local está interesado en una multidifusión concreta.

Consulte 6.2, "Protocolo de administración de grupos de Internet (IGMP)" en la página 241 para obtener una revisión detallada de IGMP.

### 3.4 Protocolo de resolución de direcciones (ARP)

El protocolo de resolución de direcciones (ARP) es un protocolo estándar específico de la red. El protocolo de resolución de direcciones es responsable de convertir las direcciones de protocolo de nivel superior (direcciones IP) en direcciones de red físicas. Se describe en

Artículo 826 del Código Penal.

#### 3.4.1 Descripción general de ARP

En una sola red física, los hosts individuales se conocen en la red por su dirección de hardware físico. Los protocolos de nivel superior se dirigen a los hosts de destino en forma de dirección simbólica (dirección IP en este caso). Cuando un protocolo de este tipo quiere enviar un datagrama a la dirección IP de destino w.x.y.z, el controlador del dispositivo no entiende esta dirección.

Por lo tanto, se proporciona un módulo (ARP) que traducirá la dirección IP a la dirección física del host de destino. Utiliza una tabla de búsqueda (a veces denominada *caché ARP*) para realizar esta traducción.

Cuando la dirección no se encuentra en la memoria caché ARP, se envía una difusión en la red con un formato especial llamado *Solicitud ARP*. Si una de las máquinas de la red reconoce su propia dirección IP en la solicitud, enviará un mensaje *Respuesta de ARP* de vuelta al host solicitante. La respuesta contendrá la dirección de hardware físico del host y la información de la ruta de origen (si el paquete ha cruzado puentes a su paso). Tanto esta dirección como la información de la ruta de origen se almacenan en la memoria caché ARP del host solicitante. Todos los datagramas posteriores a esta dirección IP de destino ahora se pueden traducir a una dirección física, que es utilizada por el controlador de dispositivo para enviar el datagrama en la red.

Una excepción a la regla es la tecnología de modo de transferencia asíncrono (ATM), en la que ARP no se puede implementar en la capa física como se describió anteriormente. Por lo tanto, cada host, tras la inicialización, debe registrarse en un servidor ARP para poder resolver las direcciones IP en direcciones de hardware (consulte también 2.10, "Modo de transferencia asíncrono (ATM)" en la página 47).

ARP fue diseñado para ser utilizado en redes que soportan la transmisión de hardware.

Esto significa, por ejemplo, que ARP no funcionará en una red X.25.

#### 3.4.2 Concepto detallado de ARP

ARP se utiliza en redes IEEE 802, así como en las redes Ethernet DIX más antiguas para asignar direcciones IP a direcciones de hardware físicas (consulte 2.1, "Ethernet y redes de área local (LAN) IEEE 802" en la página 30). Para ello, está estrechamente relacionado con el controlador de dispositivo de esa red. De hecho, las especificaciones ARP en RFC 826 solo describen su funcionalidad, no su implementación. La implementación depende en gran medida del controlador de dispositivo para un tipo de red y, por lo general, se codifican juntos en el *microcódigo del adaptador*.

##### Generación de paquetes ARP

Si una aplicación desea enviar datos a una determinada dirección IP de destino, el mecanismo de enrutamiento IP determina primero la dirección IP del siguiente salto del paquete (puede ser el propio host de destino o un enrutador) y el dispositivo de hardware en el que se debe enviar. Si se trata de una red IEEE 802.3/4/5, se debe consultar el módulo ARP para asignar el tipo <protocolo, la dirección del protocolo de destino> a una dirección física.

El módulo ARP intenta encontrar la dirección en esta caché ARP. Si encuentra el par coincidente, devuelve la dirección física de 48 bits correspondiente a la persona que llama (el controlador del dispositivo), que luego transmite el paquete. Si no encuentra el par en su tabla, descarta *el paquete* (se supone que un protocolo de nivel superior retransmitirá) y genera una *difusión* de red de una solicitud ARP. Consulte la Figura 3-39 en la página 121 para obtener más detalles.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | |  |  | Encabezado de capa física | | | Un  R  P  P a c k e t | espacio de direcciones de hardware | | | | Espacio de direcciones de protocolo | | | | Longitud de bytes de la dirección de hardware (N) | | Longitud de bytes de la dirección de protocolo (m) | | Código de operación | | | | Dirección de hardware del remitente | | | | Dirección de protocolo del remitente | | | | Dirección de hardware del destino | | | | Dirección de protocolo del destino | | |   x bytes  2 bytes  2 bytes 2bytes  2 bytes n bytes m bytes n bytes  m bytes |

*Figura 3-39 ARP: Paquete de solicitud/respuesta*

Dónde:

Espacio de direcciones de hardware: especifica el tipo de hardware; algunos ejemplos son Ethernet o Packet Radio Net.

Espacio de direcciones de protocolo: Especifica el tipo de protocolo, igual que el campo EtherType en el encabezado IEEE 802 (IP o ARP).

Longitud de la dirección de hardware: Especifica la longitud (en bytes) de las direcciones de hardware de este paquete. Para IEEE 802.3 e IEEE 802.5, es 6.

Longitud de la dirección de protocolo: Especifica la longitud (en bytes) de las direcciones de protocolo de este paquete. En el caso de IP, es 4.

Código de operación: Especifica si se trata de una solicitud ARP (1) o de una respuesta (2).

Dirección de hardware de origen/destino: contiene las direcciones de hardware de red física. Para IEEE 802.3, se trata de direcciones de 48 bits.

Dirección de protocolo de origen/destino: Contiene las direcciones de protocolo. Para TCP/IP, estas son las direcciones IP de 32 bits.

Para el paquete de solicitud ARP, la dirección de hardware de destino es el único campo indefinido en el paquete.

##### Recepción de paquetes ARP

Cuando un host recibe un paquete ARP (ya sea una solicitud de difusión o una respuesta punto a punto), el controlador del dispositivo receptor pasa el paquete al módulo ARP, que lo trata como se muestra en la Figura 3-40.

Sí

Sí

Sí

Sí

Sí

Sí

No

(

descartar

)

¿Tengo el

¿Tipo de hardware?

No

(

descartar

)

No

No

)

descartar

(

No

No

(

descartar

)

Fin

¿Hablo lo especificado?

¿protocolo?

Es el par <tipo protocolo,

Dirección del protocolo del remitente>

¿Ya estás en mi mesa?

¿Soy el protocolo objetivo?

¿dirección?

¿Flag = false?

¿Es el código de operación una solicitud?

Agregue el triplete <protocolo

tipo, protocolo del remitente y

hardware emisor> a

mesa.

Establecer bandera = false.

Actualice la tabla con el atributo

Dirección de hardware del remitente.

Establezca flag=true.

Origen y destino de intercambio

direccionesen el ARP

paquete. Pon mi local

Direcciones en el origen

Campos de dirección. Devolver

Paquete ARP como ARP

Respuesta a la solicitud

anfitrión.

*Figura 3-40 ARP: Recepción de paquetes*

El host solicitante recibirá esta respuesta ARP y seguirá el mismo algoritmo para tratarla. Como resultado de esto, el triplete <tipo de protocolo, dirección de protocolo, dirección de hardware> para el host deseado se agregará a su tabla de búsqueda (caché ARP). La próxima vez que un protocolo de nivel superior quiera enviar un paquete a ese host, el módulo ARP encontrará la dirección de hardware de destino y el paquete se enviará a ese host.

Tenga en cuenta que debido a que la solicitud ARP original era una difusión en la red, todos los hosts de esa red habrán actualizado la dirección de hardware del remitente en su tabla (solo si ya estaba en la tabla).

#### 3.4.3 ARP y subredes

El protocolo ARP permanece sin cambios en presencia de subredes. Recordar que cada datagrama IP pasa primero por el algoritmo de enrutamiento IP. Este algoritmo Selecciona el controlador de dispositivo de hardware que debe enviar el paquete. Solo entonces, se consulta el módulo ARP asociado a ese controlador de dispositivo.

#### 3.4.4 Proxy-ARP o subredes transparentes

Proxy-ARP se describe en RFC 1027, que es un subconjunto del método propuesto en RFC 925. Es otro método para construir subredes locales, sin necesidad de una modificación en el algoritmo de enrutamiento IP, pero con modificaciones en los routers que interconectan las subredes.

##### Concepto Proxy-ARP

Considere una red IP que está dividida en subredes e interconectada por enrutadores. Utilizamos el algoritmo de enrutamiento IP "antiguo", lo que significa que ningún host conoce la existencia de múltiples redes físicas. Considere los hosts A y B, que se encuentran en diferentes redes físicas dentro de la misma red IP, y un enrutador R entre las dos subredes, como se ilustra en la Figura 3-41.

Un

R

B

*Figura 3-41 ARP: Hosts interconectados por un router*

Cuando el host A desea enviar un datagrama IP al host B, primero tiene que determinar la dirección de red física del host B mediante el uso del protocolo ARP.

Debido a que el host A no puede diferenciar entre las redes físicas, su algoritmo de enrutamiento IP cree que el host B está en la red física local y envía una solicitud ARP de difusión. El host B no recibe esta difusión, pero el router R hace. El router R entiende las subredes, es decir, ejecuta la versión de subred del algoritmo de ruteo IP y podrá ver que el destino de la solicitud ARP (desde el campo de dirección del protocolo de destino) está en otra red física. Si las tablas de ruteo del router R especifican que el siguiente salto a esa otra red es a través de un dispositivo físico diferente, responderá al ARP como si fuera el host B, diciendo que la dirección de red del host B es la del propio router R.

El host A recibe esta respuesta ARP, la coloca en su caché y enviará futuros paquetes IP para el host B al router R. El router reenviará dichos paquetes a la subred correcta.

El resultado es una subred transparente:

Los hosts normales (como A y B) no conocen las subredes, por lo que utilizan el algoritmo de enrutamiento IP "antiguo".

Los enrutadores entre subredes deben:

* Utilice el algoritmo de enrutamiento IP de subred.
* Utilice un módulo ARP modificado, que puede responder en nombre de otros hosts.

Consulte la Figura 3-42 para obtener más detalles.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | | |  | |
|  | Un |  |
|  |
| "o ld " IP ro u tin g |  | B |
|  |
|  |  |  |  |
|  | R |  |
| IP s u b n e t ro u tin g a n d m o d ifie d A R P |

*Figura 3-42 ARP: Enrutador Proxy-ARP*

### 3.5 Protocolo de resolución inversa de direcciones (RARP)

El protocolo de resolución inversa de direcciones (RARP) es un protocolo estándar específico de la red. Se describe en RFC 903.

Algunos hosts de red, como las estaciones de trabajo sin disco, no conocen su propia dirección IP cuando se inician. Para determinar su propia dirección IP, utilizan un mecanismo similar a ARP, pero ahora la dirección de hardware del host es el parámetro conocido y la dirección IP el parámetro consultado. Difiere más fundamentalmente de ARP en el hecho de que debe existir un servidor RARP en la red que mantenga que una base de datos de asignaciones de la dirección de hardware a la dirección de protocolo debe estar preconfigurada.

#### 3.5.1 Concepto de RARP

La resolución inversa de direcciones se realiza de la misma manera que la dirección ARP

resolución. Se utiliza el mismo formato de paquete (consulte la Figura 3-39 en la página 121) que para

ARP.

Una excepción es el campo de código de operación que ahora toma los siguientes valores:

1. Para la solicitud de RARP
2. Para la respuesta del RARP

Y, por supuesto, el encabezado físico de la trama ahora indicará RARP como el protocolo de nivel superior (8035 hexadecimal) en lugar de ARP (0806 hexadecimal) o IP (0800 hexadecimal) en el campo EtherType.

Algunas diferencias surgen del propio concepto de RARP:

ARP solo asume que cada host conoce la asignación entre su propia dirección de hardware y la dirección de protocolo. RARP requiere que uno o más hosts de servidor en la red mantengan una base de datos de asignaciones entre direcciones de hardware y direcciones de protocolo para que puedan responder a las solicitudes de los hosts de cliente.

Debido al tamaño que puede tener esta base de datos, parte de la función del servidor suele implementarse fuera del microcódigo del adaptador, con opcionalmente una pequeña caché en el microcódigo. La parte de microcódigo solo es responsable de la recepción y transmisión de las tramas RARP, el mapeo RARP en sí mismo es atendido por el software de servidor que se ejecuta como un proceso normal en la máquina host.

La naturaleza de esta base de datos también requiere algún software para crear y actualizar la base de datos manualmente.

Si hay varios servidores RARP en la red, el solicitante de RARP solo utiliza la primera respuesta RARP recibida en su solicitud RARP de difusión y descarta las demás.

### 3.6 Protocolo de arranque (BOOTP)

El protocolo Bootstrap (BOOTP) permite que una estación de trabajo cliente se inicialice con una pila IP mínima y solicite su dirección IP, una dirección de puerta de enlace y la dirección de un servidor de nombres de un servidor BOOTP. Si se va a utilizar BOOTP en la red, el servidor y el cliente suelen estar en el mismo segmento de LAN física. BOOTP solo se puede utilizar a través de segmentos puenteados cuando los puentes de enrutamiento de origen son o a través de subredes, si tiene un router capaz de reenviar BOOTP.

BOOTP es un borrador de protocolo estándar. Se recomienda su estado. Las especificaciones de BOOTP se encuentran en RFC 951, que ha sido actualizado por RFC1542 y RFC 2132.

También hay actualizaciones de BOOTP, algunas relacionadas con la interoperabilidad con DHCP (consulte 3.7, "Protocolo de configuración dinámica de host (DHCP)" en la página 130), descritas en RFC 1542, que actualiza RFC 951 y RFC 2132. Las actualizaciones de BOOTP son proyectos de normas con un estatus de optativas y recomendadas, respectivamente.

El protocolo BOOTP se desarrolló originalmente como un mecanismo para permitir que los hosts sin disco se iniciaran de forma remota a través de una red como estaciones de trabajo, enrutadores, concentradores de terminales, etc. Permite una pila mínima de protocolos IP sin información de configuración para obtener suficiente información para comenzar el proceso de descarga del código de arranque necesario. BOOTP no define cómo se realiza la descarga, pero este proceso normalmente utiliza TFTP (consulte también 14.2, "Protocolo trivial de transferencia de archivos (TFTP)" en la página 529), como se describe en RFC 906. Aunque todavía se usa ampliamente para este propósito por los hosts sin disco, BOOTP también se usa comúnmente únicamente como un mecanismo para entregar información de configuración a un cliente que no se ha configurado manualmente.

El proceso BOOTP implica los siguientes pasos:

1. El cliente determina su propia dirección de hardware; esto normalmente está en una ROM en el hardware.
2. Un cliente BOOTP envía su dirección de hardware en un datagrama UDP al servidor. La figura 3-43 de la página 127 muestra el contenido completo de este datagrama. Si el cliente conoce su dirección IP o la dirección del servidor, debe usarlas, pero en general, los clientes BOOTP no tienen ningún dato de configuración de IP. Si el cliente no conoce su propia dirección IP, utiliza 0.0.0.0. Si el cliente no conoce la dirección IP del servidor, utiliza la dirección de difusión limitada (255.255.255.255). El número de puerto UDP es 67.
3. El servidor recibe el datagrama y busca la dirección de hardware del cliente en su archivo de configuración, que contiene la dirección IP del cliente. El servidor rellena los campos restantes en el datagrama UDP y los devuelve al cliente mediante el puerto UDP 68. Para ello, se puede utilizar uno de estos tres métodos:
   * Si el cliente conoce su propia dirección IP (se incluyó en la solicitud BOOTP), el servidor devuelve el datagrama directamente a esta dirección. Es probable que la memoria caché ARP en la pila de protocolos del servidor no conozca el

dirección de hardware que coincida con la dirección IP. Se utilizará ARP para determinarlo como normal.

* + Si el cliente no conoce su propia dirección IP (era 0.0.0.0 en la solicitud BOOTP), el servidor debe ocuparse de su propia caché ARP.
  + ARP en el servidor no se puede utilizar para encontrar la dirección de hardware del cliente porque el cliente no conoce su dirección IP y, por lo tanto, no puede responder a una solicitud ARP. A esto se le llama el problema del "huevo y la gallina". Hay dos soluciones posibles:
    - Si el servidor tiene un mecanismo para actualizar directamente su propia caché ARP sin usar ARP en sí, lo hace y luego envía el datagrama directamente.
    - Si el servidor no puede actualizar su propia caché ARP, debe enviar una respuesta de difusión.

1. Cuando reciba la respuesta, el cliente BOOTP registrará su propia dirección IP (lo que le permitirá responder a las solicitudes ARP) y comenzará el proceso de arranque.

La Figura 3-43 ofrece una descripción general del formato de mensaje BOOTP.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 8 16 24 31   |  |  |  |  |  | | --- | --- | --- | --- | --- | | código |  | Tipo H/W | largura | lúpulo | |  |  | ID de transacción | |  | | sobras | |  | Banderas | campo | |  | | Dirección IP del cliente | |  | |  | | su dirección IP | |  | |  | | Dirección IP del servidor | |  | |  | | Dirección IP del router | |  | |  | | Dirección de hardware del cliente (16 bytes) | |  | |  | | Nombre de host del servidor (64 bytes) | |  | |  | | Nombre del archivo de arranque (128 bytes) | |  | |  | | Área específica del proveedor (64 bytes) | |  | |

*Figura 3-43 Formato de mensaje BOOTP* Donde:

|  |  |
| --- | --- |
| **Código** | Indica una solicitud o una respuesta:   1. Pedir 2. Respuesta |
| **Tipo H/W** | El tipo de hardware, por ejemplo: |

**1** Ethernet

**6** Redes IEEE 802

Consulte STD 2 – Números de Internet asignados para obtener una lista completa.

|  |  |
| --- | --- |
| **Largura** | Longitud de la dirección de hardware en bytes. Ethernet y token ring usan 6, por ejemplo. |
| **Lúpulo** | El cliente lo establece en 0.  Se incrementa mediante un enrutador que retransmite la solicitud a otro servidor y se utiliza para identificar bucles. RFC 951 sugiere que un valor de 3 indica un bucle. |
| **ID de transacción** | Un número aleatorio que se utiliza para hacer coincidir esta solicitud de arranque con la respuesta que genera. |
| **Sobras** | Establecido por el cliente. Es el tiempo transcurrido en segundos desde que el cliente inició su proceso de arranque. |
| **Campo Banderas** | La parte más significativa del campo de banderas se utiliza como bandera de difusión. Todos los demás bits deben ponerse a cero; Están reservados para uso futuro. Normalmente, los servidores BOOTP intentan entregar mensajes BOOTREPLY directamente a un cliente mediante la entrega de unidifusión. La dirección de destino en el encabezado IP se establece en BOOTP *su dirección IP* y la dirección MAC se establece en la *dirección de hardware del cliente BOOTP*. Si un host no puede recibir un datagrama IP de unidifusión hasta que conozca su dirección IP, este bit de difusión debe establecerse para indicar al servidor que BOOTREPLY debe enviarse como una difusión IP y MAC. De lo contrario, este bit debe ponerse a cero. |
| **Dirección IP del cliente** | Establecido por el cliente, ya sea en su dirección IP conocida o 0.0.0.0. |
| **Su dirección IP** | Establecido por el servidor si el campo de dirección IP del cliente era 0.0.0.0. |
| **Dirección IP del servidor** | Establecido por el servidor. |
| **Dirección IP del router** | Esta es la dirección de un agente de retransmisión BOOTP, *no* un enrutador IP general para ser utilizado por el cliente. Lo establece el agente de reenvío cuando se utiliza el reenvío BOOTP (consulte 3.6.1, "Reenvío BOOTP" en la página 129). |

#### Dirección de hardware del cliente

Establecido por el cliente y utilizado por el servidor para identificar qué cliente registrado está arrancando.

|  |  |
| --- | --- |
| **Nombre de host del servidor** | Nombre de host de servidor opcional terminado por X'00'. |
| **Nombre del archivo de arranque** | El cliente deja este valor nulo o especifica un nombre genérico, como router, que indica el tipo de archivo de arranque que se va a utilizar. El servidor devuelve el nombre de archivo completo de un archivo de arranque adecuado para el cliente. El valor termina en X'00'. |

**Área específica del proveedor** Área opcional específica del proveedor. Los clientes siempre deben llenar los primeros cuatro bytes con una "cookie mágica". Si no se utiliza una cookie mágica específica del proveedor, el cliente debe utilizar 99.130.83.99 seguido de una etiqueta final (255) y establecer los bytes restantes en cero. El área específica del proveedor también puede contener *Extensiones de proveedor de BOOTP*. Estas son opciones que se pueden pasar al cliente en el momento del arranque junto con su dirección IP. Por ejemplo, el cliente también puede recibir la dirección de un enrutador predeterminado, la dirección de un servidor de nombres de dominio y una máscara de subred. BOOTP comparte las mismas opciones que DHCP, con la excepción de varias opciones específicas de DHCP. Consulte RFC 2132 para obtener todos los detalles.

Una vez que el cliente BOOTP ha procesado la respuesta, puede continuar con la transferencia del archivo de arranque y ejecutar el proceso de arranque completo. Consulte RFC 906 para obtener la especificación de cómo se hace esto con TFTP. En el caso de un host sin disco, el proceso de arranque completo normalmente reemplazará la pila de protocolo IP mínima, cargada desde ROM y utilizada por BOOTP y TFTP, con una pila de protocolo IP normal transferida como parte del archivo de arranque y que contiene la personalización correcta para el cliente.

#### 3.6.1 Reenvío BOOTP

El cliente BOOTP utiliza la dirección de difusión limitada para las solicitudes BOOTP, lo que requiere que el servidor BOOTP esté en la misma subred que el cliente. El reenvío BOOTP es un mecanismo para que los enrutadores reenvíen solicitudes BOOTP a través de subredes. Es una opción de configuración disponible en la mayoría de los routers. El router configurado para reenviar solicitudes BOOTP se conoce como *agente de retransmisión BOOTP*.

Un router normalmente descartará cualquier datagrama que contenga direcciones de origen ilegales, como 0.0.0.0, que es utilizado por un cliente BOOTP. Un router también descartará generalmente los datagramas con la dirección de destino de difusión limitada. Sin embargo, un

El agente de retransmisión BOOTP aceptará dichos datagramas de los clientes BOOTP en el puerto 67.

El proceso llevado a cabo por un agente de retransmisión BOOTP al recibir una solicitud BOOTP es el siguiente:

1. Cuando el agente de retransmisión BOOTP recibe una solicitud BOOTP, primero comprueba el campo de saltos para comprobar el número de saltos ya completados con el fin de decidir si se reenvía la solicitud. Normalmente, el umbral para el número permitido de saltos es configurable.

1. Si el agente de retransmisión decide retransmitir la solicitud, comprueba el contenido del campo de dirección IP del router. Si este campo es cero, rellena este campo con la dirección IP de la interfaz en la que se recibió la solicitud BOOTP. Si este campo ya tiene una dirección IP de otro agente de retransmisión, no se toca.
2. El valor del campo hops se incrementa.
3. A continuación, el agente de retransmisión reenvía la solicitud BOOTP a uno o varios servidores BOOTP. La dirección del servidor o servidores BOOTP está preconfigurada en el agente de retransmisión. Normalmente, BOOTPREQUEST se reenvía como una trama de unidifusión, aunque algunas implementaciones utilizan el reenvío de difusión.
4. Cuando el servidor BOOTP recibe la solicitud BOOTP con el campo de dirección IP del enrutador distinto de cero, envía una BOOTREPLY de unidifusión IP al agente de retransmisión BOOTP en la dirección de este campo en el puerto 67.
5. Cuando el agente de retransmisión BOOTP recibe BOOTREPLY, los campos de tipo H/W, longitud y dirección de hardware del cliente en el mensaje proporcionan suficiente información de la capa de enlace para devolver la respuesta al cliente. El agente de retransmisión comprueba el indicador de difusión. Si se establece este indicador, el agente reenvía BOOTPREPLY al cliente como una difusión. Si no se establece el indicador de difusión, el agente de retransmisión envía una respuesta como unidifusión a la dirección especificada en su dirección IP.

Cuando un router se configura como agente de retransmisión BOOTP, la tarea de reenvío BOOTP es considerablemente diferente de la tarea de conmutación de datagramas entre subredes que normalmente realiza un router. El reenvío de mensajes BOOTP se puede considerar como la recepción de mensajes BOOTP como destino final y, a continuación, la generación de nuevos mensajes BOOTP para reenviarlos a otro destino.

#### 3.6.2 Consideraciones sobre BOOTP

El uso de BOOTP permite la configuración centralizada de múltiples clientes. Sin embargo, requiere que se mantenga una tabla estática con una dirección IP preasignada para cada cliente que probablemente se conecte al servidor BOOTP, incluso si el cliente rara vez está activo. Esto significa que no hay alivio en el número de direcciones IP requeridas. Existe una medida de seguridad en un entorno que utiliza BOOTP, ya que el servidor solo asignará una dirección IP a un cliente si tiene una dirección MAC válida.

### 3.7 Protocolo de configuración dinámica de host (DHCP)

DHCP es un borrador de protocolo estándar. Su estatus es electivo. Las especificaciones DHCP actuales se encuentran en RFC 2131 con actualizaciones en RFC 3396 y RFC 4361. Las especificaciones también se encuentran en RFC 2132 con actualizaciones en RFC3442, RFC3942 y RFC4361.

El protocolo de configuración dinámica de host (DHCP) proporciona un marco para pasar información de configuración a los hosts de una red TCP/IP. DHCP se basa en el protocolo BOOTP, lo que agrega la capacidad de asignación automática de direcciones de red reutilizables y opciones de configuración adicionales. Para obtener información sobre BOOTP, consulte 3.6, "Protocolo de arranque (BOOTP)" en la página 125. Los mensajes DHCP utilizan el puerto UDP 67, el puerto conocido del servidor BOOTP y el puerto UDP 68, el puerto conocido del cliente BOOTP. Los participantes de DHCP pueden interoperar con los participantes de BOOTP. Consulte 3.7.8, "Interoperabilidad BOOTP y DHCP" en la página 140 para obtener más detalles.

DHCP consta de dos componentes:

Un protocolo que entrega parámetros de configuración específicos del host desde un servidor DHCP a un host

Un mecanismo para la asignación de direcciones de red temporales o permanentes a los hosts

IP requiere la configuración de muchos parámetros dentro del software de implementación del protocolo. Debido a que IP se puede usar en muchos tipos diferentes de hardware de red, los valores de esos parámetros no se pueden adivinar ni asumir que tienen valores predeterminados correctos. El uso de un esquema de asignación de direcciones distribuidas basado en un mecanismo de sondeo/defensa, para el descubrimiento de direcciones de red que ya están en uso, no puede garantizar direcciones de red únicas porque es posible que los hosts no siempre puedan defender sus direcciones de red.

DHCP admite tres mecanismos para la asignación de direcciones IP:

Asignación automática

DHCP asigna una dirección IP permanente al host.

Asignación dinámica

DHCP asigna una dirección IP durante un período de tiempo limitado. Esta dirección de red se denomina *arrendamiento*. Este es el único mecanismo que permite la reutilización automática de direcciones que ya no son necesarias para el host al que se asignó.

Asignación manual

La dirección del host es asignada por un administrador de red.

#### 3.7.1 El formato del mensaje DHCP

El formato de un mensaje DHCP se muestra en la Figura 3-44.

*Figura 3-44 Formato de mensaje DHCP*

código

largura

31

0

8 16 24

Tipo HW

lúpulo

Identificación de la cción

Transa

Dirección IP del cliente

su dirección IP

Dirección IP del servidor

sobras

Campo de banderas

Dirección IP del router

Dirección de hardware del cliente

(16

Bytes

)

Nombre de host del servidor

(64

Bytes

)

Nombre del archivo de arranque

(128

Bytes

)

Opciones

(312

Bytes

)

Dónde:

|  |  |
| --- | --- |
| **Código** | Indica una solicitud o una respuesta:   1. Pedir 2. Respuesta |
| **Tipo HW** | El tipo de hardware, por ejemplo:  **1** Ethernet |
| **Largura** | **6** Redes IEEE 802    Consulte STD 2 – Números de Internet asignados para obtener una lista completa.  Longitud de la dirección de hardware en bytes. |
| **Lúpulo** | El cliente lo establece en 0. Se incrementa mediante un enrutador que retransmite la solicitud a otro servidor y se utiliza para identificar bucles. RFC 951 sugiere que un valor de 3 indica un bucle. | |
| **ID de transacción** | Un número aleatorio que se utiliza para hacer coincidir esta solicitud de arranque con la respuesta que genera. | |
| **Sobras** | Establecido por el cliente. Es el tiempo transcurrido en segundos desde que el cliente inició su proceso de arranque. | |
| **Campo Banderas** | La parte más significativa del campo de banderas se utiliza como bandera de difusión. Todos los demás bits deben ponerse a cero y se reservan para su uso futuro. Normalmente, los servidores DHCP intentan entregar mensajes DHCP directamente a un cliente mediante la entrega de unidifusión. La dirección de destino en el encabezado IP se establece en el DHCP*, su dirección IP*, y la dirección MAC se establece en la *dirección de hardware del cliente DHCP*. Si un host no puede recibir un datagrama IP de unidifusión hasta que conozca su dirección IP, este bit de difusión debe establecerse para indicar al servidor que la respuesta DHCP debe enviarse como una difusión IP y MAC.  De lo contrario, este bit debe ponerse a cero. | |
| **Dirección IP del cliente** | Establecido por el cliente. Su dirección IP conocida o 0.0.0.0. | |
| **Su dirección IP** | Establecido por el servidor si el campo de dirección IP del cliente era 0.0.0.0. | |
| **Dirección IP del servidor** | Establecido por el servidor. | |
| **Dirección IP del router** | Esta es la dirección de un agente de retransmisión BOOTP, *no* un enrutador IP general para ser utilizado por el cliente. Lo establece el agente de reenvío cuando se utiliza el reenvío BOOTP (consulte 3.6.1, "Reenvío BOOTP" en la página 129). | |

##### Dirección de hardware del cliente

Establecido por el cliente. DHCP define una opción de identificador de cliente que se utiliza para la identificación del cliente. Si no se utiliza esta opción, el cliente se identifica por su dirección MAC.

|  |  |
| --- | --- |
| **Nombre de host del servidor** | Nombre de host de servidor opcional terminado por X'00'. |
| **Nombre del archivo de arranque** | El cliente deja este valor nulo o especifica un nombre genérico, como router, que indica el tipo de archivo de arranque que se va a utilizar. En una solicitud DHCPDISCOVER, se establece en  nulo. El servidor devuelve un nombre de ruta de acceso de directorio completo en una solicitud DHCPOFFER. El valor termina en X'00'. |

**Opciones** Los primeros cuatro bytes del campo de opciones del DHCP

contiene la cookie mágica (99.130.83.99). El resto del campo de opciones consta de parámetros etiquetados que se denominan *opciones*. Consulte RFC 2132, con actualizaciones en RFC3942, para obtener más detalles.

#### 3.7.2 Tipos de mensajes DHCP

Los mensajes DHCP se clasifican en una de las siguientes categorías:

DHCPDISCOVER: Difusión por parte de un cliente para encontrar servidores DHCP disponibles.

DHCPOFFER: Respuesta de un servidor a un DHCPDISCOVER y a la dirección IP de oferta y otros parámetros.

DHCPREQUEST: Mensaje de un cliente a los servidores que realiza una de las siguientes acciones:

* Solicita los parámetros ofrecidos por uno de los servidores y rechaza todas las demás ofertas.
* Verifica una dirección asignada previamente después de un cambio en el sistema o en la red (un reinicio, por ejemplo).
* Solicita la extensión de un contrato de arrendamiento en una dirección en particular.

DHCPACK: Acuse de recibo del servidor al cliente con parámetros, incluida la dirección IP.

DHCPNACK: Acuse de recibo negativo del servidor al cliente, que indica que la concesión del cliente ha expirado o que una dirección IP solicitada es incorrecta.

DHCPDECLINE: Mensaje del cliente al servidor que indica que la dirección ofrecida ya está en uso.

DHCPRELEASE: Mensaje del cliente al servidor que cancela el resto de una concesión y renuncia a la dirección de red.

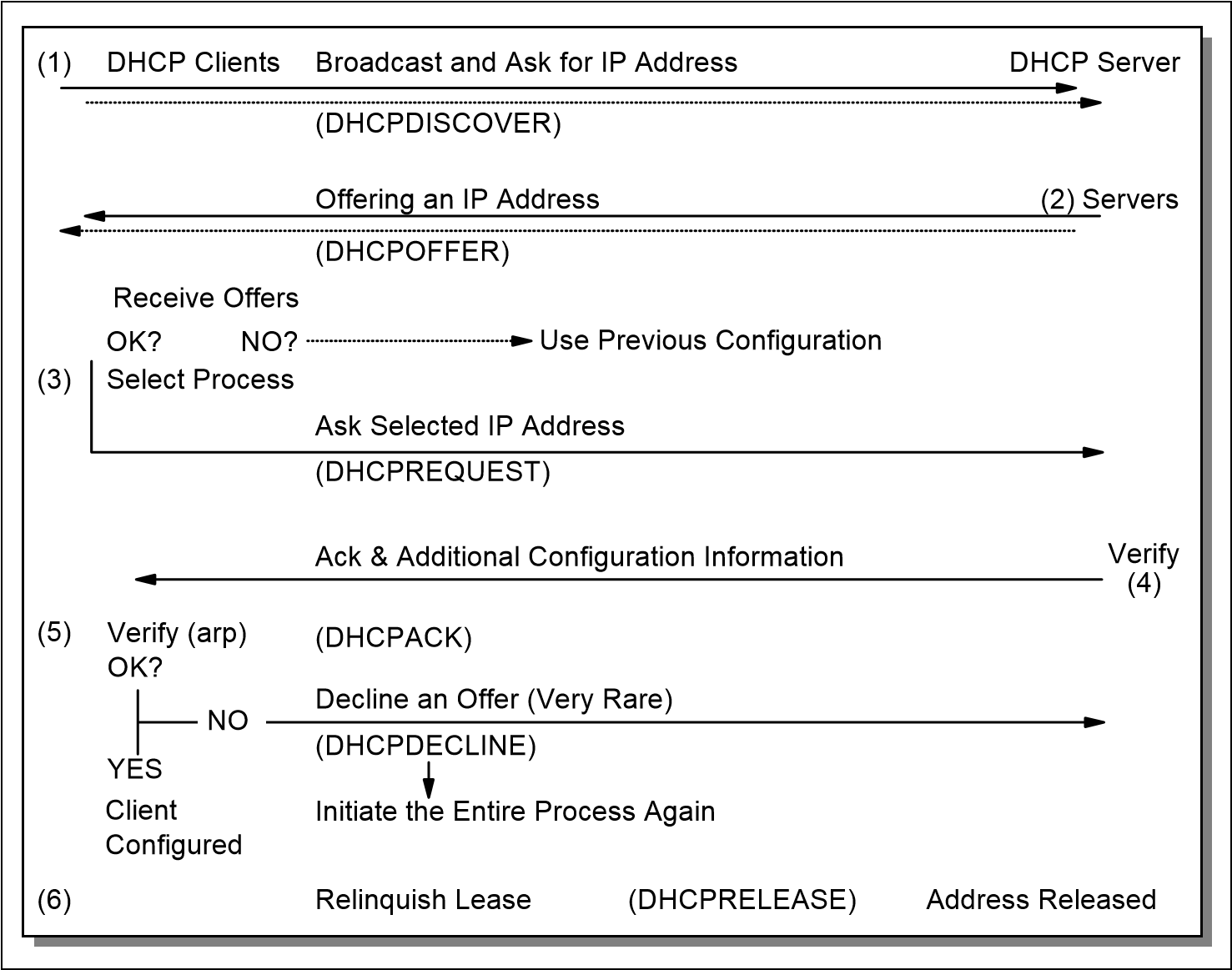
DHCPINFORM: Mensaje de un cliente que ya tiene una dirección IP (configurada manualmente, por ejemplo), solicitando más parámetros de configuración al servidor DHCP.

#### 3.7.3 Asignación de una nueva dirección de red

En esta sección se describe la interacción cliente/servidor si el cliente no conoce su dirección de red. Supongamos que el servidor DHCP tiene un bloque de red

direcciones desde las que puede satisfacer las solicitudes de nuevas direcciones. Cada servidor también mantiene una base de datos de direcciones asignadas y concesiones en almacenamiento local permanente.

Los pasos de interacción cliente/servidor DHCP se ilustran en la Figura 3-45.



*Figura 3-45 Interacción entre el cliente DHCP y el servidor DHCP*

El siguiente procedimiento describe los pasos de interacción cliente/servidor DHCP que se ilustran en la Figura 3-45:

1. El cliente difunde un mensaje DHCPDISCOVER en su subred física local. En este momento, el cliente se encuentra en el estado INIT. El mensaje DHCPDISCOVER puede incluir algunas opciones, como la sugerencia de dirección de red o la duración de la concesión.
2. Cada servidor responde con un mensaje DHCPOFFER que incluye una dirección de red disponible (su dirección IP) y otras opciones de configuración. Los servidores registran la dirección tal como se ofrece al cliente para evitar que la misma dirección se ofrezca a otros clientes en caso de que se produzcan más

DHCPDISCOVER que se reciben antes de que el primer cliente tenga

completó su configuración.

1. El cliente recibe uno o varios mensajes DHCPOFFER de uno o varios servidores. El cliente elige uno en función de los parámetros de configuración ofrecidos y difunde un mensaje DHCPREQUEST que incluye la opción de identificador de servidor para indicar qué mensaje ha seleccionado y la opción de dirección IP solicitada tomada de su dirección IP en la oferta seleccionada.
2. En el caso de que no se reciban ofertas, si el cliente tiene conocimiento de una dirección de red anterior, el cliente puede reutilizar esa dirección si su concesión sigue siendo válida hasta que expire la concesión.
3. Los servidores reciben la difusión DHCPREQUEST del cliente. Los servidores no seleccionados por el mensaje DHCPREQUEST utilizan el mensaje como

Notificación de que el cliente ha rechazado la oferta de ese servidor. El servidor seleccionado en el mensaje DHCPREQUEST confirma el enlace del cliente al almacenamiento persistente y responde con un mensaje DHCPACK que contiene los parámetros de configuración para el cliente solicitante. La combinación del hardware del cliente y la dirección de red asignada constituyen un identificador único para la concesión del cliente y son utilizados tanto por el cliente como por el servidor para identificar una concesión a la que se hace referencia en cualquier mensaje DHCP. El campo de su dirección IP en los mensajes DHCPACK se rellena con la dirección de red seleccionada.

1. El cliente recibe el mensaje DHCPACK con los parámetros de configuración. El cliente realiza una comprobación final de los parámetros, por ejemplo, con ARP para la dirección de red asignada, y anota la duración de la concesión y la cookie de identificación de concesión especificada en el mensaje DHCPACK. En este punto, el cliente está configurado.
2. Si el cliente detecta un problema con los parámetros del mensaje DHCPACK (la dirección ya está en uso en la red, por ejemplo), el cliente envía un mensaje DHCPDECLINE al servidor y reinicia el proceso de configuración. El cliente debe esperar un mínimo de diez segundos antes de reiniciar el proceso de configuración para evitar un tráfico de red excesivo en caso de bucle. Al recibir un DHCPDECLINE, el servidor debe marcar la dirección ofrecida como no disponible (y posiblemente informar al administrador del sistema de que hay un problema de configuración).
3. Si el cliente recibe un mensaje DHCPNAK, el cliente reinicia el proceso de configuración.
4. El cliente puede optar por renunciar a su arrendamiento en una dirección de red enviando un mensaje DHCPRELEASE al servidor. El cliente identifica la concesión que se va a liberar incluyendo su dirección de red y su dirección de hardware.

#### 3.7.4 Proceso de renovación de arrendamiento DHCP

En esta sección se describe la interacción entre los servidores DHCP y los clientes que

ya se han configurado y el proceso que garantiza el vencimiento de la concesión y renovación.

El proceso consta de los siguientes pasos:

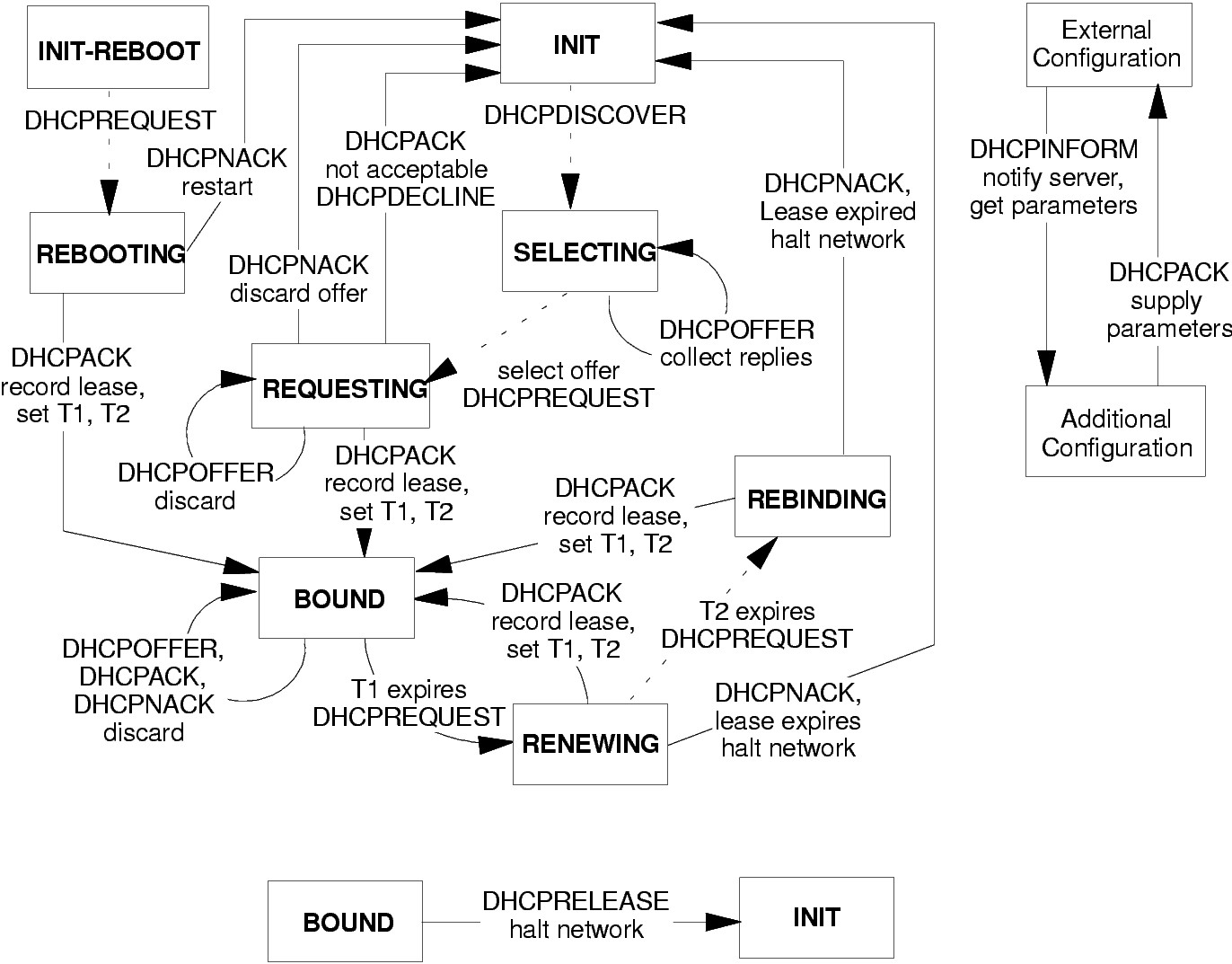
1. Cuando un servidor envía el DHCPACK a un cliente con una dirección IP y parámetros de configuración, también registra el inicio del tiempo de concesión para esa dirección. Este tiempo de concesión se pasa al cliente como una de las opciones del mensaje DHCPACK, junto con dos valores de temporizador, T1 y T2. El cliente tiene derecho a utilizar la dirección dada durante el tiempo de arrendamiento. Al aplicar la configuración recibida, el cliente también inicia los temporizadores T1 y T2. En este momento, el cliente se encuentra en el estado BOUND. Los tiempos T1 y T2 son opciones configurables por el servidor, pero T1 debe ser menor que T2 y T2 debe ser menor que el tiempo de concesión. De acuerdo con RFC 2132, T1 tiene un valor predeterminado de (0,5 \* tiempo de concesión) y T2 de forma predeterminada es (0,875 \* tiempo de concesión).
2. Cuando expira el temporizador T1, el cliente enviará una solicitud DHCP (unidifusión) al servidor que ofreció la dirección, solicitando extender la concesión para la configuración dada. El cliente se encuentra ahora en el estado RENEWING. Por lo general, el servidor responde con un mensaje DHCPACK que indica el nuevo tiempo de concesión, y los temporizadores T1 y T2 se restablecen en el cliente en consecuencia. El servidor también restablece su registro del tiempo de concesión. En circunstancias normales, un cliente activo renueva continuamente su contrato de arrendamiento de esta manera indefinidamente, sin que el contrato de arrendamiento expire nunca.
3. Si no se recibe ningún DHCPACK hasta que expire el temporizador T2, el cliente ingresa el

Estado REVINCULANTE. Ahora transmite un mensaje DHCPREQUEST para extender su concesión. Esta solicitud se puede confirmar mediante un mensaje DHCPACK desde cualquier servidor DHCP de la red.

1. Si el cliente no recibe un mensaje DHCPACK después de que haya expirado su concesión, tiene que dejar de usar su configuración TCP/IP actual. A continuación, el cliente puede volver al estado INIT, emitiendo una difusión DHCPDISCOVER para intentar obtener cualquier dirección válida.

La Figura 3-46 muestra el proceso DHCP y el cambio de estado del cliente durante ese proceso.

*Figura 3-46 Estado del cliente DHCP y proceso DHCP*



#### 3.7.5 Reutilización de una dirección de red previamente asignada

Si el cliente recuerda y quiere reutilizar una dirección de red previamente asignada, se llevan a cabo los siguientes pasos:

1. El cliente difunde un mensaje DHCPREQUEST en su subred local. El mensaje DHCPREQUEST incluye la dirección de red del cliente.
2. Un servidor con conocimiento de los parámetros de configuración del cliente responde con un mensaje DHCPACK al cliente (siempre que la concesión siga vigente), renovando la concesión al mismo tiempo.
3. Si la concesión del cliente ha expirado, el servidor con conocimiento del cliente responde con DHCPNACK.
4. El cliente recibe el mensaje DHCPACK con los parámetros de configuración. El cliente realiza una comprobación final de los parámetros y anota la duración de la concesión y la cookie de identificación de concesión especificada en el mensaje DHCPACK. En este punto, se configura el cliente y se restablecen sus temporizadores T1 y T2.
5. Si el cliente detecta un problema con los parámetros del mensaje DHCPACK, el cliente envía un mensaje DHCPDECLINE al servidor y reinicia el proceso de configuración solicitando una nueva dirección de red. Si el cliente recibe un mensaje DHCPNAK, no puede volver a utilizar su dirección de red recordada. En su lugar, debe solicitar una nueva dirección reiniciando el proceso de configuración como se describe en 3.7.3, "Asignación de una nueva dirección de red" en la página 134.

Para obtener más información, consulte las RFC mencionadas anteriormente.

#### 3.7.6 Repositorio de parámetros de configuración

DHCP proporciona almacenamiento persistente de parámetros de red para clientes de red. Un servidor DHCP almacena una entrada clave-valor para cada cliente, siendo la clave un identificador único, por ejemplo, un número de subred IP y un identificador único dentro de la subred (normalmente una dirección de hardware), y el valor contiene los parámetros de configuración asignados por última vez a este cliente en particular.

Un efecto de esto es que un cliente DHCP tenderá a ser siempre asignado a la misma dirección IP por el servidor, siempre que el grupo de direcciones no esté sobresuscrito y la dirección anterior no haya sido ya asignada a otro cliente.

#### 3.7.7 Consideraciones sobre DHCP

La asignación dinámica DHCP de direcciones IP y parámetros de configuración libera al administrador de red de una gran cantidad de trabajo de configuración manual. La capacidad de mover un dispositivo de una red a otra y obtener automáticamente parámetros de configuración válidos para la red actual puede ser de gran beneficio para los usuarios móviles. Además, debido a que las direcciones IP solo se asignan cuando los clientes están realmente activos, es posible, mediante el uso de tiempos de arrendamiento razonablemente cortos y el hecho de que a los clientes móviles no es necesario asignar más de una dirección, reducir el número total de direcciones en uso en una organización. Sin embargo, tenga en cuenta los siguientes puntos cuando se implemente DHCP:

DHCP se basa en UDP, que es inherentemente inseguro. En funcionamiento normal, un cliente no autorizado puede conectarse a una red y obtener una dirección IP y una configuración válidas. Para evitar esto, es posible preasignar direcciones IP a direcciones MAC particulares (similar a BOOTP), pero esto aumenta la carga de trabajo de administración y elimina el beneficio del reciclaje de direcciones. También se pueden configurar servidores DHCP no autorizados, enviando información falsa y potencialmente perjudicial a los clientes.

En un entorno DHCP en el que se utiliza la asignación automática o dinámica de direcciones, generalmente no es posible predeterminar la dirección IP de un cliente en un momento determinado. En este caso, si también se utilizan servidores DNS estáticos, es probable que los servidores DNS no contengan asignaciones válidas de nombre de host a dirección IP para los clientes. Si tener entradas de cliente en el DNS es importante para la red, puede usar DHCP para asignar manualmente direcciones IP a esos clientes y, a continuación, administrar las asignaciones de cliente en el DNS según corresponda.

#### 3.7.8 Interoperabilidad BOOTP y DHCP

El formato de los mensajes DHCP se basa en el formato de los mensajes BOOTP, lo que permite que los clientes BOOTP y DHCP interoperen en determinadas circunstancias. Cada mensaje DHCP contiene una opción de tipo de mensaje DHCP (51). Se supone que cualquier mensaje sin esta opción proviene de un cliente BOOTP.

El soporte para clientes BOOTP en un servidor DHCP debe ser configurado por un administrador del sistema, si es necesario. El servidor DHCP responde a los mensajes BOOTPREQUEST con BOOTPREPLY, en lugar de DHCPOFFER. Cualquier servidor DHCP que no esté configurado de esta manera descartará cualquier trama BOOTPREQUEST que se le envíe. Un servidor DHCP puede ofrecer direcciones estáticas, o direcciones automáticas (de su grupo de direcciones no asignadas), a un cliente BOOTP (aunque no todas las implementaciones de BOOTP entenderán las direcciones automáticas). Si se ofrece una dirección automática a un cliente BOOTP, esa dirección debe tener un tiempo de concesión infinito, ya que el cliente no comprenderá el mecanismo de concesión DHCP.

Los routers configurados como agentes de retransmisión BOOTP pueden reenviar los mensajes DHCP.

### 3.8 RFC relevantes para este capítulo

Las siguientes RFC proporcionan información detallada sobre los protocolos y arquitecturas de conexión presentados a lo largo de este capítulo:

[RFC 791 – Protocolo de Internet (septiembre de 1981)](http://www.ietf.org/rfc/rfc791.txt)

[RFC 792 – Protocolo de mensajes de control de Internet (septiembre de 1981)](http://www.ietf.org/rfc/rfc792.txt)

[RFC 826 - Protocolo de resolución de direcciones Ethernet: o conversión de red](http://www.ietf.org/rfc/rfc826.txt)   [direcciones de protocolo a la dirección Ethernet de 48.bit para la transmisión en hardware Ethernet (noviembre de 1982)](http://www.ietf.org/rfc/rfc826.txt)

[RFC 903 – Un protocolo de resolución de direcciones inversas (junio de 1984)](http://www.ietf.org/rfc/rfc903.txt)

[RFC 906 – Carga de arranque mediante TFTP (junio de 1984)](http://www.ietf.org/rfc/rfc906.txt)

[RFC919 – Difusión de datagramas de Internet (octubre de 1984)](http://www.ietf.org/rfc/rfc919.txt)

[RFC922 – Difusión de datagramas de Internet en presencia de subredes (octubre de 1984)](http://www.ietf.org/rfc/rfc922.txt)

[RFC 925 - Resolución de direcciones multi-LAN (octubre de 1984)](http://www.ietf.org/rfc/rfc925.txt)

[RFC 950 – Procedimiento de subredes estándar de Internet (agosto de 1985)](http://www.ietf.org/rfc/rfc950.txt)

[RFC 951 – Protocolo Bootstrap (septiembre de 1985)](http://www.ietf.org/rfc/rfc951.txt)

[RFC 1027: uso de ARP para implementar puertas de enlace de subred transparentes](http://www.ietf.org/rfc/rfc1027.txt)

[(Octubre de](http://www.ietf.org/rfc/rfc1027.txt) [1987)](http://www.ietf.org/rfc/rfc1027.txt)

[RFC 1112 - Extensiones de host para multidifusión IP (agosto de 1989)](http://www.ietf.org/rfc/rfc1112.txt)

[RFC 1122 – Requisitos para Hosts de Internet – Capas de Comunicación (Octubre de 1989)](http://www.ietf.org/rfc/rfc1122.txt)

[RFC 1166 – Números de Internet (julio de 1990)](http://www.ietf.org/rfc/rfc1166.txt)

[RFC 1191 – Detección de MTU de ruta (noviembre de 1990)](http://www.ietf.org/rfc/rfc1191.txt)

[RFC 1256 - Mensajes de detección de enrutador ICMP (septiembre de 1991)](http://www.ietf.org/rfc/rfc1256.txt)

[RFC 1349 – Tipo de servicio en el conjunto de protocolos de Internet (julio de 1992)](http://www.ietf.org/rfc/rfc1349.txt)

[RFC 1393 Traceroute mediante una opción IP G (enero de 1993)](http://www.ietf.org/rfc/rfc1393.txt)

[RFC 1466 - Directrices para la gestión del espacio de direcciones IP (mayo de 1993)](http://www.ietf.org/rfc/rfc1466.txt)

[RFC 1518 - Una arquitectura para la asignación de direcciones IP con CIDR](http://www.ietf.org/rfc/rfc1518.txt)

[(Septiembre de](http://www.ietf.org/rfc/rfc1518.txt) [1993)](http://www.ietf.org/rfc/rfc1518.txt)

[RFC 1519 - Enrutamiento entre dominios sin clases (CIDR): una asignación de direcciones (septiembre de 1993)](http://www.ietf.org/rfc/rfc1519.txt)

[RFC 1520 - Intercambio de información de enrutamiento a través de los límites del proveedor en el entorno CIDR (septiembre de 1993)](http://www.ietf.org/rfc/rfc1520.txt)

[RFC 1542 – Aclaraciones y extensiones para el protocolo Bootstrap](http://www.ietf.org/rfc/rfc1524.txt)

[(Octubre de](http://www.ietf.org/rfc/rfc1524.txt) [1993)](http://www.ietf.org/rfc/rfc1524.txt)

[RFC 1788 – Mensajes de nombres de dominio ICMP (abril de 1995)](http://www.ietf.org/rfc/rfc1788.txt)

[RFC 1812 - Requisitos para enrutadores IP versión 4 (junio de 1995)](http://www.ietf.org/rfc/rfc1812.txt)

[RFC 1918 – Asignación de direcciones para Internets privadas (febrero de 1996)](http://www.ietf.org/rfc/rfc1918.txt)

[RFC 2050 - Pautas de asignación de IP del Registro de Internet (noviembre de 1996)](http://www.ietf.org/rfc/rfc2050.txt)

[RFC 2131 – Protocolo de configuración dinámica de host (marzo de 1997)](http://www.ietf.org/rfc/rfc2131.txt)

[RFC 2132 - Opciones DHCP y extensiones de proveedor BOOTP (marzo de 1997)](http://www.ietf.org/rfc/rfc2132.txt)

[RFC 2236 – Protocolo de administración de grupos de Internet, versión 2](http://www.ietf.org/rfc/rfc2236.txt)

[(Noviembre de](http://www.ietf.org/rfc/rfc2236.txt) [1997)](http://www.ietf.org/rfc/rfc2236.txt)

[RFC 2474 – Definición del campo de servicios diferenciados (campo DS) en el](http://www.ietf.org/rfc/rfc2474.txt)

[Encabezados IPv4 e IPv6 (diciembre de 1998)](http://www.ietf.org/rfc/rfc2474.txt)

[RFC 2644 - Cambio del valor predeterminado para las difusiones dirigidas en el router](http://www.ietf.org/rfc/rfc2644.txt)

[(Agosto de](http://www.ietf.org/rfc/rfc2644.txt) [1999)](http://www.ietf.org/rfc/rfc2644.txt)

[RFC 2663 - Terminología del traductor de direcciones de red IP (NAT) y](http://www.ietf.org/rfc/rfc2663.txt)

[Consideraciones (agosto de 1999)](http://www.ietf.org/rfc/rfc2663.txt)

[RFC 3022 - Traductor de direcciones de red IP tradicional (NAT tradicional) (enero de 2001)](http://www.ietf.org/rfc/rfc3022.txt)

[RFC 3168 - La adición de la notificación de congestión explícita (ECN) a IP (septiembre de 2001)](http://www.ietf.org/rfc/rfc3168txt)

[RFC 3260 – Nueva terminología y aclaraciones para Diffserv (abril de 2002)](http://www.ietf.org/rfc/rfc3260.txt)

[RFC 3330 – Direcciones IPv4 de uso especial (septiembre de 2002)](http://www.ietf.org/rfc/rfc3330.txt)

[RFC 3396 – Codificación de opciones largas en la configuración dinámica del host](http://www.ietf.org/rfc/rfc3396.txt)

[Protocolo (DHCPv4) (noviembre de 2002)](http://www.ietf.org/rfc/rfc3396.txt)

[RFC 3442: la opción de ruta estática sin clases para el host dinámico](http://www.ietf.org/rfc/rfc3442.txt)

[Protocolo de configuración (DHCP) versión 4 (diciembre de 2002)](http://www.ietf.org/rfc/rfc3442.txt)

[RFC 3942 - Reclasificación del protocolo de configuración dinámica de host versión 4](http://www.ietf.org/rfc/rfc3942txt)

[(DHCPv4) Opciones (noviembre de 2004)](http://www.ietf.org/rfc/rfc3942txt)

[RFC 4361 - Identificadores de cliente específicos del nodo para la configuración dinámica del host](http://www.ietf.org/rfc/rfc4361.txt)

[Versión cuatro del protocolo (DHCPv4) (febrero de 2006)](http://www.ietf.org/rfc/rfc4361.txt)

[RFC 4379 - Detección del plano de datos conmutado por etiquetas multiprotocolo (MPLS)](http://www.ietf.org/rfc/rfc479.txt)

[Fallos (febrero de 2006)](http://www.ietf.org/rfc/rfc479.txt)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **4** | |

## Capítulo 4. Protocolos de capa de transporte

Este capítulo proporciona una visión general de los protocolos más importantes y comúnmente utilizados de la capa de transporte TCP/IP. Entre ellas se encuentran:

Protocolo de datagramas de usuario (UDP)

Protocolo de control de transmisión (TCP)

Al basarse en la funcionalidad proporcionada por el Protocolo de Internet (IP), los protocolos de transporte entregan datos a las aplicaciones que se ejecutan en Internet. Esto se hace haciendo uso de puertos, como se describe en 4.1, "Puertos y zócalos" en la página 144. Los protocolos de transporte pueden proporcionar funcionalidades adicionales, como el control de la congestión, la entrega fiable de datos, la supresión de datos duplicados y el control de flujo, tal y como lo hace TCP.

© Derechos de autor IBM Corp. 1989-2006. Todos los derechos reservados.

### 4.1 Puertos y zócalos

En esta sección se presentan los conceptos de puerto y socket, que son necesarios para determinar qué proceso local en un host determinado se comunica realmente con qué proceso, en qué host remoto y mediante qué protocolo. Si esto suena confuso, considere los siguientes puntos:

A un proceso de solicitud se le asigna un número de identificador de proceso (ID de proceso), que es probable que sea diferente cada vez que se inicie ese proceso.

Los identificadores de proceso difieren entre las plataformas del sistema operativo, por lo que no son uniformes.

Un proceso de servidor puede tener varias conexiones a varios clientes a la vez, por lo que los identificadores de conexión simples no son únicos. El concepto de puertos y sockets proporciona una forma de identificar de manera uniforme y única las conexiones y los programas y hosts que participan en ellas, independientemente de los ID de proceso específicos.

El concepto de puertos y sockets proporciona una forma de identificar de manera uniforme y única las conexiones y los programas y hosts que participan en ellas, independientemente de los ID de proceso específicos.

#### 4.1.1 Puertos

Cada proceso que desea comunicarse con otro proceso se identifica en el conjunto de protocolos TCP/IP mediante uno o más puertos. Un puerto es un número de 16 bits utilizado por el protocolo de host a host para identificar a qué protocolo de nivel superior o programa de aplicación (proceso) debe entregar mensajes entrantes. Hay dos tipos de puertos:

Conocidos: Los puertos conocidos pertenecen a servidores estándar, por ejemplo, Telnet utiliza el puerto 23. Los números de puerto conocidos oscilan entre 1 y 1023 (antes de 1992, el rango entre 256 y 1023 se usaba para servidores específicos de UNIX). Los números de puerto conocidos suelen ser impares, porque los primeros sistemas que utilizaban el concepto de puerto requerían un par de puertos pares/impares para las operaciones dúplex. La mayoría de los servidores requieren un solo puerto. Las excepciones son el servidor BOOTP, que utiliza dos: 67 y 68 (consulte 3.6, "Protocolo de arranque (BOOTP)" en la página 125) y el servidor FTP, que utiliza dos: 20 y 21 (consulte (14.1, "Protocolo de transferencia de archivos (FTP)" en la página 514).

Los puertos conocidos son controlados y asignados por la Autoridad de Números Asignados de Internet (IANA) y, en la mayoría de los sistemas, solo pueden ser utilizados por procesos del sistema o por programas ejecutados por usuarios privilegiados. Los puertos conocidos permiten a los clientes encontrar servidores sin información de configuración. Los números de puerto conocidos se definen en STD 2 – Números de Internet asignados.

Efímero: Algunos clientes no necesitan números de puerto conocidos porque inician la comunicación con los servidores, y el número de puerto que están utilizando está contenido en los datagramas UDP/TCP enviados al servidor. A cada proceso de cliente se le asigna un número de puerto, durante el tiempo que necesite, por el host en el que se está ejecutando. Los números de puerto efímeros tienen valores superiores a 1023, normalmente en el rango de 1024 a 65535.

Los puertos efímeros no están controlados por la IANA y pueden ser utilizados por programas ordinarios desarrollados por el usuario en la mayoría de los sistemas.

La confusión, debido a que dos aplicaciones diferentes intentan usar los mismos números de puerto en un host, se evita escribiendo esas aplicaciones para solicitar un puerto disponible de TCP/IP. Dado que este número de puerto se asigna dinámicamente, puede diferir de una invocación de una aplicación a la siguiente.

UDP, TCP e ISO TP-4 utilizan el mismo principio de puerto. En la medida de lo posible, se utilizan los mismos números de puerto para los mismos servicios además de UDP, TCP e ISO TP-4.

**Nota:** Normalmente, un servidor usará TCP o UDP, pero hay excepciones. Por ejemplo, los servidores de nombres de dominio (consulte 12.1, "Sistema de nombres de dominio (DNS)" en la página 426) utilizan tanto el puerto UDP 53 como el puerto TCP 53.

#### 4.1.2 Zócalos

La interfaz de socket es una de las varias interfaces de programación de aplicaciones para los protocolos de comunicación (consulte 11.2, "Interfaces de programación de aplicaciones (API)" en la página 410). Diseñadas para ser una interfaz de programación de comunicación genérica, las API de socket fueron introducidas por primera vez por 4.2 Berkeley Software Distribution (BSD). Aunque no se ha estandarizado, la API de socket de Berkeley se ha convertido en una abstracción estándar de facto de la industria para la implementación de sockets TCP/IP de red.

Tenga en cuenta las siguientes terminologías:

Un *socket* es un tipo especial de *identificador de archivo*, que es utilizado por un proceso para solicitar servicios de red del sistema operativo.

Una *dirección de socket* es el triple:

<protocolo, dirección local, puerto local>

Por ejemplo, en el conjunto TCP/IP (versión 4):

<TCP, 192.168.14.234, 8080>

Una *conversación* es el vínculo de comunicación entre dos procesos.

Una *asociación* es la tupla de 5 que especifica completamente los dos procesos que componen una conexión:

<protocolo, dirección local, puerto local, dirección extranjera, puerto extranjero>

En el conjunto TCP/IP (versión 4), lo siguiente podría ser una asociación válida:

<tcp, 192.168.14.234, 1500, 192.168.44, 22>

Una *asociación a medias* es una de las siguientes, cada una de las cuales especifica la mitad de una conexión:

<protocolo, dirección local, proceso local> O bien:

<protocolo, dirección extranjera, proceso extranjero>

La media asociación también se denomina *socket* o *dirección de transporte*. Es decir, un socket es un punto final para la comunicación que se puede nombrar y direccionar en una red.

Dos procesos se comunican a través de sockets TCP. El modelo de socket proporciona un proceso con una conexión de flujo de bytes dúplex completo a otro proceso. No es necesario que la aplicación se ocupe de la gestión de este flujo; estas instalaciones son proporcionadas por TCP.

TCP utiliza el mismo principio de puerto que UDP para proporcionar multiplexación. Al igual que UDP, TCP utiliza puertos conocidos y efímeros. Cada lado de una conexión TCP tiene un socket que se puede identificar por el triple <TCP, la dirección IP, el número de puerto>. Si dos procesos se comunican a través de TCP, tienen una conexión lógica que es identificable de forma única por los dos sockets involucrados, es decir, por la combinación <TCP, dirección IP local, puerto local, dirección IP remota, puerto remoto>. Los procesos del servidor pueden gestionar varias conversaciones a través de un único puerto. Consulte 11.2.1, "La API de socket" en la página 410 para obtener más información sobre las API de socket.

### 4.2 Protocolo de datagramas de usuario (UDP)

UDP es un protocolo estándar con número STD 6. UDP se describe mediante RFC 768 - Protocolo de datagramas de usuario. Su estado es estándar y casi todas las implementaciones TCP/IP destinadas a la transferencia de pequeñas unidades de datos o aquellas que pueden permitirse perder una pequeña cantidad de datos (como la transmisión multimedia) incluirán UDP.

UDP es básicamente una interfaz de aplicación para IP. No agrega confiabilidad, control de flujo o recuperación de errores a IP. Simplemente sirve como un multiplexor/demultiplexor para enviar y recibir datagramas, utilizando puertos para dirigir los datagramas, como se muestra en la Figura 4-1. Para obtener una explicación más detallada de los puertos, consulte 4.1, "Puertos y zócalos" en la página 144.

Puerto B

Proceso 2

Puerto Z

...

UDP - De-Multiplexación de Puertos

IP

proceso m—

Proceso 1

Puerto A

*Figura 4-1 UDP: Demultiplexación basada en puertos*

UDP proporciona un mecanismo para que una aplicación envíe un datagrama a otra. La capa UDP puede considerarse extremadamente delgada y, en consecuencia, muy eficiente, pero requiere que la aplicación asuma la responsabilidad de la recuperación de errores, etc.

Las aplicaciones que envían datagramas a un host necesitan identificar un destino que sea más específico que la dirección IP, ya que los datagramas normalmente se dirigen a ciertos procesos y no al sistema en su conjunto. UDP proporciona esto mediante el uso de puertos. Discutimos el concepto de puerto en 4.1, "Puertos y zócalos" en la página 144.

#### 4.2.1 Formato de datagrama UDP

Cada datagrama UDP se envía dentro de un único datagrama IP. Aunque el datagrama IP puede fragmentarse durante la transmisión, la implementación IP receptora lo volverá a ensamblar antes de presentarlo a la capa UDP. Todas las implementaciones IP deben aceptar datagramas de 576 bytes, lo que significa que, teniendo en cuenta que un encabezado IP de tamaño máximo de 60 bytes, un datagrama UDP de 516 bytes es aceptable para todas las implementaciones. Muchas implementaciones aceptarán datagramas más grandes, pero esto no está garantizado.

El datagrama UDP tiene un encabezado de 8 bytes, como se describe en la Figura 4-2 de la página 148.

*Figura 4-2 UDP: Formato de datagrama*

Puerto de destino

Puerto de origen

Da

gracias...

Checksum

Largura

Dónde:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Puerto de origen** | | Indica el puerto del proceso de envío. Es el puerto al que se dirigen las respuestas. | | | |
| **Puerto de destino** | | Especifica el puerto del proceso de destino en el host de destino. | | | |
| **Largura** | | La longitud (en bytes) de este datagrama de usuario, incluido el encabezado. | | | |
| **Checksum** | | Un complemento opcional de 16 bits de la suma del complemento de uno de un encabezado pseudo-IP, el encabezado UDP y los datos UDP. En la Figura 4-3, vemos un encabezado pseudo-IP. Contiene las direcciones IP de origen y destino, el protocolo y la longitud UDP. | | | |
|  | | Dirección IP de origen |  |
|  | | Dirección IP de destino | Ess |
| Cero | | Protocolo | Longitud de TCP |

*Figura 4-3 UDP: Encabezado Pseudo-IP*

El encabezado pseudo-IP amplía eficazmente la suma de comprobación para incluir el datagrama IP original (no fragmentado).

#### 4.2.2 Interfaz de programación de aplicaciones UDP

La interfaz de aplicación ofrecida por UDP se describe en RFC 768. Establece:

La creación de nuevos puertos de recepción

Operación de recepción que devuelve los bytes de datos y una indicación del puerto de origen y la dirección IP de origen

La operación de envío que tiene, como parámetros, los puertos y direcciones de datos, origen y destino

La forma en que se implementa esta interfaz se deja a discreción de cada proveedor.

Tenga en cuenta que UDP e IP no proporcionan entrega garantizada, control de flujo o recuperación de errores, por lo que deben ser proporcionados por la aplicación.

Las aplicaciones estándar que utilizan UDP incluyen:

Protocolo trivial de transferencia de archivos (consulte 14.2, "Protocolo trivial de transferencia de archivos (TFTP)" en la página 529).

Servidor de nombres del Sistema de Nombres de Dominio (consulte 12.2, "Sistema de Nombres de Dominio Dinámico" en la página 453).

Llamada a procedimiento remoto, utilizada por el sistema de archivos de red (consulte 11.2.2, "Llamada a procedimiento remoto (RPC)" en la página 415 y 14.4, "Sistema de archivos de red (NFS)" en la página 538).

Protocolo simple de administración de red (consulte 17.1, "El protocolo simple de administración de red (SNMP)" en la página 624).

Protocolo ligero de acceso a directorios (consulte 12.4, "Protocolo ligero de acceso a directorios (LDAP)" en la página 459).

### 4.3 Protocolo de control de transmisión (TCP)

TCP es un protocolo estándar con número STD 7. TCP se describe mediante RFC 793 - Protocolo de control de transmisión. Su estado es estándar y, en la práctica, todas las implementaciones de TCP/IP que no se utilicen exclusivamente para el enrutamiento incluirán TCP.

TCP proporciona considerablemente más facilidades para las aplicaciones que UDP. Específicamente, esto incluye la recuperación de errores, el control de flujo y la confiabilidad. TCP es un  *protocolo orientado a la conexión*, a diferencia de UDP, que no tiene *conexión*. La mayoría de los protocolos de aplicación de usuario, como Telnet y FTP, utilizan TCP. Los dos procesos se comunican entre sí a través de una conexión TCP (comunicación entre procesos o IPC), como se muestra en la Figura 4-4. En la figura, los procesos 1 y 2 se comunican a través de una conexión TCP transportada por datagramas IP. Consulte 4.1, "Puertos y zócalos" en la página 144 para obtener más detalles sobre los puertos y zócalos.

Puerto M

fidedigno

Conexión TCP

...

TCP

IP

...

puerto n

Proceso 2

...

TCP

IP

...

Anfitrión A

poco fiable

Datagramas IP

Proceso 1

anfitrión B

*Figura 4-4 TCP: Conexión entre procesos*

#### 4.3.1 Concepto TCP

Como se señaló anteriormente, el propósito principal de TCP es proporcionar un circuito lógico confiable o un servicio de conexión entre pares de procesos. No asume la confiabilidad de los protocolos de nivel inferior (como IP), por lo que TCP debe garantizarlo por sí mismo.

TCP se puede caracterizar por las siguientes facilidades que proporciona para las aplicaciones que lo utilizan:

Transferencia de datos de flujo: desde el punto de vista de la aplicación, TCP transfiere un flujo contiguo de bytes a través de la red. La aplicación no tiene que molestarse en cortar los datos en bloques básicos o datagramas. TCP hace esto agrupando los bytes en segmentos TCP, que se pasan a la capa IP para su transmisión al destino. Además, el propio TCP decide cómo segmentar los datos y puede reenviarlos a su conveniencia. A veces, una aplicación necesita estar segura de que todos los datos pasados a TCP se han transmitido realmente al destino. Por esa razón, se define una función push. Empujará todos los segmentos TCP restantes que aún estén almacenados a

el host de destino. La función de conexión de cierre normal también envía los datos al destino.

Confiabilidad: TCP asigna un número de secuencia a cada byte transmitido y espera un acuse de recibo positivo (ACK) de la capa TCP receptora. Si la confirmación no se recibe dentro de un intervalo de tiempo de espera, los datos se retransmiten. Debido a que los datos se transmiten en bloques (segmentos TCP), solo se envía al host de destino el número de secuencia del primer byte de datos del segmento.

El TCP receptor utiliza los números de secuencia para reorganizar los segmentos cuando llegan desordenados y para eliminar los segmentos duplicados.

Control de flujo: El TCP receptor, al enviar un ACK de vuelta al remitente, también indica al remitente el número de bytes que puede recibir (más allá del último segmento TCP recibido) sin causar saturación y desbordamiento en sus búferes internos. Esto se envía en el ACK en forma del número de secuencia más alto que puede recibir sin problemas. Este mecanismo también se conoce como mecanismo de ventana, y lo discutiremos con más detalle más adelante en este capítulo.

Multiplexación: Se logra mediante el uso de puertos, al igual que con UDP.

Conexiones lógicas: Los mecanismos de confiabilidad y control de flujo que se describen aquí requieren que TCP inicialice y mantenga cierta información de estado para cada flujo de datos. La combinación de este estado, incluidos los sockets, los números de secuencia y los tamaños de ventana, se denomina conexión lógica. Cada conexión se identifica de forma única por el par de sockets utilizados por los procesos de envío y recepción.

Dúplex completo: TCP proporciona flujos de datos simultáneos en ambas direcciones.

##### El principio de la ventana

Un protocolo de transporte simple podría usar el siguiente principio: enviar un paquete y luego esperar un acuse de recibo del receptor antes de enviar el siguiente paquete. Si el ACK no se recibe dentro de un cierto período de tiempo, retransmita el paquete. Consulte la Figura 4-5 para obtener más detalles.

|  |  |
| --- | --- |
| Remitente | Receptor |
| Enviar paquete 1  Recibir confirmación  Enviar paquete 2 | Reciba el paquete 1 y responda con un ACK 1 |

*Figura 4-5 TCP: El principio de la ventana*

Aunque este mecanismo garantiza la fiabilidad, solo utiliza una parte del ancho de banda de red disponible.

Ahora, considere un protocolo en el que el remitente agrupa sus paquetes para ser transmitidos, como en la Figura 4-6, y utiliza las siguientes reglas:

El remitente puede enviar todos los paquetes dentro de la ventana sin recibir una confirmación, pero debe iniciar un temporizador de tiempo de espera para cada uno de ellos.

El receptor debe acusar recibo de cada paquete recibido, indicando el número de secuencia del último paquete bien recibido.

El remitente desliza la ventana en cada confirmación recibida.

*Figura 4-6 TCP: Paquetes de mensajes*

ventana

Paquetes

3

4

5

6

7

8

9

...

2

1

Como se muestra en la Figura 4-7, el remitente puede transmitir los paquetes 1 a 5 sin esperar ningún acuse de recibo.

*Figura 4-7 TCP: Principio de ventana*

Remitente

Red

Enviar paquete 1

Enviar paquete 2

Enviar paquete 3

Enviar paquete 4

Enviar paquete 5

ACK para el paquete 1 recibido

ACK 1

Como se muestra en la Figura 4-8, en el momento en que el remitente recibe ACK 1

(acuse de recibo para el paquete 1), puede deslizar su ventana un paquete hacia la derecha.

|  |  |
| --- | --- |
| Remitente | Receptor |
| Enviar paquete 1  Recibir confirmación  Enviar paquete 2 | Reciba el paquete 1 y responda con un ACK 1 |

*Figura 4-8 TCP: Paquetes de mensajes*

En este punto, el remitente también puede transmitir el paquete 6.

Imaginemos algunos casos especiales:

El paquete 2 se pierde: El remitente no recibirá la confirmación 2, por lo que su ventana permanecerá en la posición 1 (como en la Figura 4-8 en la página 153). De hecho, debido a que el receptor no recibió el paquete 2, reconocerá los paquetes 3, 4 y 5 con un ACK 1, porque el paquete 1 fue el último recibido en secuencia. En el lado del remitente, eventualmente se producirá un tiempo de espera para el paquete 2 y se retransmitirá. Tenga en cuenta que la recepción de este paquete por parte del receptor generará ACK 5, porque ahora ha recibido con éxito todos los paquetes 1 a 5, y la ventana del remitente se deslizará cuatro posiciones al recibir este ACK 5.

El paquete 2 llegó, pero el acuse de recibo se pierde: el remitente no recibe ACK 2, pero recibirá ACK 3. ACK 3 es un acuse de recibo para *todos los* paquetes hasta 3 (incluido el paquete 2) y el remitente ahora puede deslizar su ventana al paquete 4.

Este mecanismo de ventana garantiza:

Transmisión fiable.

Mejor uso del ancho de banda de la red (mejor rendimiento).

Control de flujo, porque el receptor puede retrasar la respuesta a un paquete con un acuse de recibo, sabiendo que sus búferes libres están disponibles y el tamaño de la ventana de la comunicación.

##### El principio de ventana aplicado a TCP

El principio de ventana discutido anteriormente se usa en TCP, pero con algunas diferencias:

Dado que TCP proporciona una conexión de secuencia de bytes, los números de secuencia se asignan a cada byte de la secuencia. TCP divide esta secuencia de bytes contigua en segmentos TCP para transmitirlos. El principio de ventana se utiliza a nivel de bytes, es decir, los segmentos enviados y los ACK recibidos llevarán números de secuencia de bytes y el tamaño de la ventana se expresa como un número de bytes, en lugar de un número de paquetes.

El tamaño de la ventana lo determina el receptor cuando se establece la conexión y es variable durante la transferencia de datos. Cada mensaje ACK incluirá el tamaño de la ventana con la que el receptor está listo para tratar en ese momento en particular.

El flujo de datos del remitente ahora se puede ver de la siguiente manera en la Figura 4-9.

5

4

2

1

...

1

14

13

12

11

10

9

8

7

3

6

5

Un

B

C

D

ventana (tamaño expresado en bytes)

Bytes

*Figura 4-9 TCP: Principio de ventana aplicado a TCP*

Dónde:

1. Bytes que se transmiten y se han reconocido
2. Bytes que se envían pero que aún no se reconocen
3. Bytes que se pueden enviar sin esperar ningún acuse de recibo
4. Bytes que aún no se pueden enviar

Recuerde que TCP bloqueará los bytes en segmentos, y un segmento TCP solo lleva el número de secuencia del primer byte del segmento.

##### Formato de segmento TCP

La Figura 4-10 muestra el formato del segmento TCP.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | | 1 | | | 2 | | | | | | 3 |
| 0 1 2 3 4 5 6 7 8 9 | | 0 1 2 3 4 5 6 7 8 9 | | | 0 1 2 3 4 5 6 7 8 9 | | | | | | 0 1 2 3 4 5 6 7 8 9 |
| Puerto de origen | | | | | Puerto de destino | | | | | | |
| Número de secuencia | | | | | | | | | | | |
| Número de acuse de recibo | | | | | | | | | | | |
| Datos  Compensar | Reservado | | U  R  G | Un  C  K | | P  S  H | R  S  T | S  Y N | F  Yo  N | Ventana | |
| Checksum | | | | | | | | Puntero urgente | | | |
| Opciones …|… Relleno | | | | | | | | | | | |
| Bytes de datos | | | | | | | | | | | |

*Figura 4-10 TCP: Formato de segmento*

Dónde:

**Puerto de origen** El número de puerto de origen de 16 bits, utilizado por el receptor para

respuesta.

**Puerto de destino** El número de puerto de destino de 16 bits.

**Número de secuencia** Número de secuencia del primer byte de datos de este segmento. Si se activa el bit de control SYN, el número de secuencia es el número de secuencia inicial (n) y el primer byte de datos es n+1.

###### Número de acuse de recibo

Si se activa el bit de control ACK, este campo contiene el valor del siguiente número de secuencia que el receptor espera recibir.

|  |  |  |
| --- | --- | --- |
| **Desplazamiento de datos** | El número de palabras de 32 bits en el encabezado TCP. Indica dónde comienzan los datos. | |
| **Reservado** | Seis bits reservados para uso futuro; debe ser cero. | |
| **URG** | Indica que el campo de puntero urgente es significativo en este segmento. | |
| **ACK**  **PSH**  **RST** | Indica que el campo de confirmación es significativo en este segmento.  Función de empuje.  Restablece la conexión. | |
| **SYN** | Sincroniza los números de secuencia. |
| **ALETA** | No más datos del remitente. |
| **Ventana** | Se utiliza en segmentos ACK. Especifica el número de bytes de datos, empezando por el indicado en el campo de número de acuse de recibo que el receptor (el emisor de este segmento) está dispuesto a aceptar. |
| **Checksum** | El complemento de 16 bits del complemento de uno suma de todas las palabras de 16 bits en un pseudoencabezado, el encabezado TCP y los datos TCP. Al calcular la suma de comprobación, el campo de suma de comprobación en sí se considera cero. |

El pseudoencabezado es el mismo que el utilizado por UDP para calcular la suma de comprobación. Es un encabezado pseudo-IP, que solo se usa para el cálculo de la suma de comprobación, con el formato que se muestra en la Figura 4-11.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Dirección IP de origen |  |  |
|  |  | Dirección IP de destino | Ess |
|  | Cero | Protocolo | Longitud de TCP |

*Figura 4-11 TCP: Encabezado pseudo-IP*

**Puntero urgente** Apunta al primer octeto de datos después de los datos urgentes. Solo es significativo cuando se establece el bit de control URG.

**Opciones** Al igual que en el caso de las opciones de datagramas IP, las opciones pueden ser

cualquiera de los dos:

* Un solo byte que contiene el número de opción
* Una opción de longitud variable en el siguiente formato, como se muestra en la Figura 4-12

3

3

opción

largura

datos de la opción...

*Figura 4-12 TCP: Opción de datagrama IP, opción de longitud variable*

Actualmente hay siete opciones definidas, como se muestra en la Tabla 4-1.

*Tabla 4-1 TCP: Opciones de datagramas IP*

|  |  |  |
| --- | --- | --- |
| **Amable** | **Largura** | **Significado** |
| 0 | - | Fin de la lista de opciones |
| 1 | - | Sin operación |
| 2 | 4 | Tamaño máximo del segmento |
| 3 | 3 | Escala de ventana |
| 4 | 2 | Saco permitido |
| 5 | X | Saco |
| 8 | 10 | Marcas de tiempo |

###### Opción de tamaño máximo de segmento

Esta opción sólo se utiliza durante el establecimiento de la conexión (conjunto de bits de control SYN) y se envía desde el lado que va a recibir los datos para indicar la longitud máxima del segmento que puede manejar. Si no se utiliza esta opción, se permite cualquier tamaño de segmento. Consulte la Figura 4-13 para obtener más detalles.

3

3

2

máximo.

Tamaño seg

4

*Figura 4-13 TCP: Tamaño máximo de segmento*

###### Opción de escala de ventana

Esta opción no es obligatoria. Ambos lados deben enviar la opción Escala de ventana en sus segmentos SYN para habilitar el escalado de ventanas en su dirección. La escala de ventana expande la definición de la ventana TCP a 32 bits. Define el tamaño de la ventana de 32 bits mediante el factor de escala en el segmento SYN sobre el tamaño de ventana estándar de 16 bits. El receptor reconstruye el tamaño de ventana de 32 bits utilizando el tamaño de ventana de 16 bits y el factor de escala. Esta opción se determina durante el protocolo de enlace. No hay forma de cambiarlo después de que se haya establecido la conexión. Consulte la Figura 4-14 para obtener más detalles.

3

3

shift.cnt

*Figura 4-14 TCP: Opción de escala de ventana*

###### Opción permitida por SACK

Esta opción se establece cuando se utiliza el acuse de recibo selectivo en esa conexión TCP. Consulte la Figura 4-15 para obtener más detalles.

4

2

*Figura 4-15 TCP: Opción permitida por SACK*

**Opción SACK** El acuse de recibo selectivo (SACK) permite al receptor informar al remitente sobre todos los segmentos que se reciben correctamente. Por lo tanto, el remitente solo enviará los segmentos que se perdieron. Si el número de segmentos que se han perdido desde el último SACK es demasiado grande, la opción SACK será demasiado grande. Como resultado, el número de bloques que se pueden notificar mediante la opción SACK se limita a cuatro. Para reducir esto, se debe utilizar la opción SACK para los datos recibidos más recientes. Consulte la Figura 4-16 para obtener más detalles.

*Figura 4-16 TCP: Opción SACK*

Largura

5

Borde izquierdo del 1er bloque

Borde derecho del 1er bloque

-

- - - - -

-

Borde izquierdo del enésimo bloque

Borde derecho del enésimo bloque

k

/

/

/

/

**Opción de marcas de tiempo** La opción timestamps envía un valor de marca de tiempo que

indica el valor actual del reloj de marca de tiempo del TCP que envía la opción. El valor de eco de marca de tiempo solo se puede utilizar si el bit ACK está establecido en el encabezado TCP. Consulte la Figura 4-17 para obtener más detalles.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | | |
|  | |  |  |  |  | |
|  | 8 | 10 | Válvula TS | Respuesta de TS Echo |  |  |
|  |  |  |  |  |
| 1 | 1 | 4 | 4 | |

*Figura 4-17 TCP: Opción de marcas de tiempo*

**Relleno** Todos los bytes cero se utilizan para llenar el encabezado TCP hasta una longitud total que es un múltiplo de 32 bits.

##### Agradecimientos y retransmisiones

TCP envía datos en segmentos de longitud variable. Los números de secuencia se basan en un recuento de bytes. Las confirmaciones especifican el número de secuencia del siguiente byte que el receptor espera recibir.

Tenga en cuenta que un segmento se pierde o se corrompe. En este caso, el receptor acusará recibo de todos los segmentos adicionales bien recibidos con un acuse de recibo que se refiere al primer byte del paquete que falta. El remitente dejará de transmitir cuando haya enviado todos los bytes de la ventana. Eventualmente, se agotará el tiempo de espera y se retransmitirá el segmento que falta.

La figura 4-18 ilustra un ejemplo en el que se utiliza un tamaño de ventana de 1500 bytes y segmentos de 500 bytes.

Remitente

Receptor

Segmento 1 (seq. 1000)

Recibe 1000, envía ACK 1500

Segmento 2 (seq. 1500)

\\\

se pierde

Segmento 3 (seq. 2000)

Recibe el ACK 1500,

¿Qué ventana de deslizamiento?

Segmento 4 (seq. 2500)

Recibe uno de los fotogramas

y responde con ACK 1500

(

El receptor todavía está esperando

byte 1500)

tamaño de ventana alcanzado,

esperando ACK

Recibe el ACK 1500,

que no desliza el

ventana

Tiempo de espera para el segmento 2

Retransmisión

*Figura 4-18 TCP: Proceso de acuse de recibo y retransmisión*

Ahora surge un problema, porque el remitente sabe que el segmento 2 se ha perdido o dañado, pero no sabe nada sobre los segmentos 3 y 4. El emisor debería al menos retransmitir el segmento 2, pero también podría retransmitir los segmentos 3 y 4 (porque están dentro de la ventana actual). Es posible que:

Se ha recibido el segmento 3 y no conocemos el segmento 4. Podría recibirse, pero ACK aún no nos ha llegado, o podría perderse.

El segmento 3 se perdió, y recibimos el ACK 1500 en la recepción del segmento 4.

Cada implementación de TCP es libre de reaccionar a un tiempo de espera como lo deseen quienes la implementan. Puede retransmitir solo el segmento 2, pero en el segundo caso, estaremos esperando nuevamente hasta que el segmento 3 agote el tiempo de espera. En este caso, perdemos todas las ventajas de rendimiento del mecanismo de ventana. O TCP podría volver a enviar inmediatamente todos los segmentos de la ventana actual.

Cualquiera que sea la elección, se pierde el máximo rendimiento. Esto se debe a que el ACK no contiene un segundo número de secuencia de acuse de recibo que indique la trama real recibida.

###### Intervalos de tiempo de espera variables

Cada TCP debe implementar un algoritmo para adaptar los valores de tiempo de espera que se utilizarán para el tiempo de ida y vuelta de los segmentos. Para ello, TCP registra la hora a la que se envió un segmento y la hora a la que se recibe la confirmación. Se calcula un promedio ponderado en varios de estos tiempos de ida y vuelta, que se utilizará como valor de tiempo de espera para el siguiente segmento o segmentos que se enviarán.

Esta es una característica importante, porque los retrasos pueden variar en la red IP, dependiendo de múltiples factores, como la carga de una red intermedia de baja velocidad o la saturación de una puerta de enlace IP intermedia.

##### Establecimiento de una conexión TCP

Antes de que se pueda transferir cualquier dato, se debe establecer una conexión entre los dos procesos. Uno de los procesos (normalmente el servidor) emite una  *llamada OPEN pasiva*, el otro una llamada *OPEN activa*. La llamada pasiva OPEN permanece inactiva hasta que otro proceso intenta conectarse a ella mediante un OPEN activo.

Como se muestra en la Figura 4-19, en la red, se intercambian tres segmentos TCP.

Iniciar

)

cliente

(

Capa TCP

Escucha

(

servidor

)

Capa TCP

1)

SYN SEQ:999 ACK

:

2)

SYN ACK SEQ:4999 ACK

:1000

3)

ACK SEQ:1000 ACK

:5000

*Figura 4-19 TCP: Establecimiento de la conexión*

Todo este proceso se conoce como apretón de manos a tres bandas. Tenga en cuenta que los segmentos TCP intercambiados incluyen los números de secuencia iniciales de ambos lados, que se utilizarán en transferencias de datos posteriores.

El cierre de la conexión se realiza implícitamente mediante el envío de un segmento TCP con el bit FIN (no más datos) establecido. Debido a que la conexión es dúplex completo (es decir, hay dos flujos de datos independientes, uno en cada dirección), el segmento FIN solo cierra la transferencia de datos en una dirección. El otro proceso ahora enviará los datos restantes que aún tiene que transmitir y también termina con un segmento TCP donde se establece el bit FIN. La conexión se elimina (información de estado en ambos lados) después de que el flujo de datos se cierra en ambas direcciones.

A continuación se muestra una lista de los diferentes estados de una conexión TCP:

LISTEN: A la espera de una solicitud de conexión de otra capa TCP.

SYN-SENT: Se ha enviado un SYN y TCP está esperando el SYN de respuesta.

SYN-RECEIVED: Se ha recibido un SYN, se ha enviado un SYN y TCP está esperando un ACK.

ESTABLECIDO: Se ha completado el protocolo de enlace a tres bandas.

FIN-WAIT-1: La aplicación local ha emitido un CLOSE. TCP ha enviado un FIN y está a la espera de un ACK o un FIN.

FIN-WAIT-2: Se ha enviado un FIN y se ha recibido un ACK. TCP está esperando un FIN de la capa TCP remota.

CLOSE-WAIT: TCP ha recibido un FIN y ha enviado un ACK. Está a la espera de una solicitud cercana de la aplicación local antes de enviar un FIN.

CIERRE: Se ha enviado un FIN, se ha recibido un FIN y se ha enviado un ACK. TCP está esperando un ACK para el FIN que se envió.

LAST-ACK: Se ha recibido un FIN y se han enviado un ACK y un FIN.

TCP está a la espera de un ACK.

TIME-WAIT: Los FIN se han recibido y confirmado, y TCP está esperando dos MSL para eliminar la conexión de la tabla.

CERRADO: Imaginario, indica que se ha eliminado una conexión de la tabla de conexiones.

#### 4.3.2 Interfaz de programación de aplicaciones TCP

La interfaz de programación de aplicaciones TCP no está completamente definida. Solo algunas funciones básicas que debe proporcionar se describen en RFC 793 - Protocolo de control de transmisión. Como es el caso de la mayoría de las RFC en el conjunto de protocolos TCP/IP, se deja un gran grado de libertad a los implementadores, lo que permite implementaciones óptimas dependientes del sistema operativo, lo que resulta en una mejor eficiencia y un mayor rendimiento.

Las siguientes llamadas de función se describen en el RFC:

Abierto: Para establecer una conexión se necesitan varios parámetros, tales como:

* Activo/Pasivo
* Enchufe externo
* Número de puerto local
* Valor de tiempo de espera (opcional)

Esto devuelve un nombre de conexión local, que se utiliza para hacer referencia a esta conexión concreta en todas las demás funciones.

Enviar: hace que los datos de un búfer de usuario al que se hace referencia se envíen a través de la conexión. Opcionalmente, puede establecer el indicador URGENT o el indicador PUSH.

Recibir: copia los datos TCP entrantes en un búfer de usuario.

Cerrar: Cierra la conexión; provoca una inserción de todos los datos restantes y un segmento TCP con el indicador FIN establecido.

Estado: una llamada dependiente de la implementación que puede devolver información, como:

* Enchufe local y foráneo
* Tamaños de ventana de envío y recepción
* Estado de conexión
* Nombre de la conexión local

Anular: hace que se anulen todas las operaciones de envío y recepción pendientes y que se envíe un RESET al TCP externo.

Para obtener más información, consulte RFC 793 - Protocolo de control de transmisión.

#### 4.3.3 Algoritmos de control de congestión TCP

Una gran diferencia entre TCP y UDP es el algoritmo de control de congestión. El algoritmo de congestión TCP evita que un remitente sobrepase la capacidad de la red (por ejemplo, enlaces WAN más lentos). TCP puede adaptar la velocidad del remitente

a la capacidad de la red e intentar evitar posibles situaciones de congestión. Para comprender la diferencia entre TCP y UDP, es muy útil comprender los algoritmos básicos de control de congestión TCP.

A lo largo de los años, se han agregado y sugerido varias mejoras en el control de la congestión a TCP. Esta es todavía un área de investigación activa y en curso, pero las implementaciones modernas de TCP contienen cuatro algoritmos entrelazados como estándares básicos de Internet:

Arranque lento

Evitar la congestión

Retransmisión rápida

Recuperación rápida

##### Arranque lento

Las implementaciones antiguas de TCP inician una conexión con el remitente inyectando múltiples segmentos en la red, hasta el tamaño de ventana anunciado por el receptor. Aunque esto está bien cuando los dos hosts están en la misma LAN, si hay enrutadores y enlaces más lentos entre el remitente y el receptor, pueden surgir problemas. Algunos enrutadores intermedios no pueden manejarlo, los paquetes se descartan y los resultados y el rendimiento de la retransmisión se degradan.

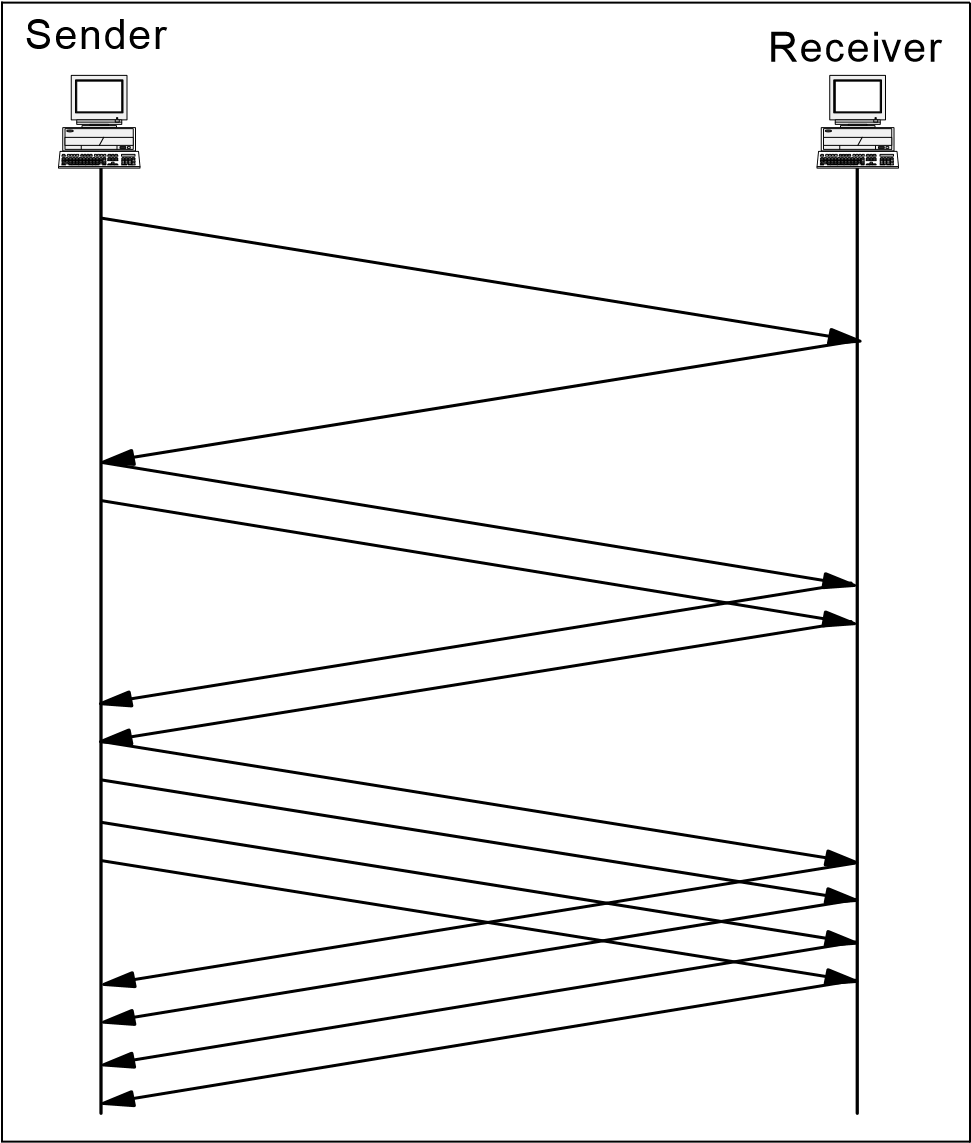
El algoritmo para evitar esto se llama inicio lento. Funciona observando que la velocidad a la que se deben inyectar nuevos paquetes en la red es la velocidad a la que los acuses de recibo son devueltos por el otro extremo. El inicio lento agrega otra ventana al TCP del remitente: la ventana de congestión, llamada cwnd. Cuando se establece una nueva conexión con un host en otra red, la ventana de congestión se inicializa en un segmento (por ejemplo, el tamaño de segmento anunciado por el otro extremo, o el valor predeterminado, normalmente 536 o 512).

**Nota:** El control de congestión se define en RFC 2581. Además, RFC 3390 actualiza RFC 2581 de modo que las implementaciones de TCP puedan inicializar la ventana de congestión entre dos y cuatro segmentos, con un límite superior de 4 K.

Cada vez que se recibe una ACK, la ventana de congestión aumenta en un segmento. El remitente puede transmitir el valor más bajo de la ventana de congestión o la ventana anunciada. La ventana de congestión es el control de flujo impuesto por el emisor, mientras que la ventana anunciada es el control de flujo impuesto por el receptor. El primero se basa en la evaluación del remitente de la congestión percibida de la red; Esto último está relacionado con la cantidad de espacio de búfer disponible en el receptor para esta conexión.

El emisor comienza transmitiendo un segmento y esperando su ACK. Cuando se recibe esa confirmación, la ventana de congestión se incrementa de uno a dos y se pueden enviar dos segmentos. Cuando se reconoce cada uno de esos dos segmentos, la ventana de congestión aumenta a cuatro. Esto proporciona un crecimiento exponencial, aunque no es exactamente exponencial, porque el receptor puede retrasar sus ACK, enviando normalmente un ACK por cada dos segmentos que recibe.

En algún momento, se puede alcanzar la capacidad de la red IP (por ejemplo, enlaces WAN más lentos) y un enrutador intermedio comenzará a descartar paquetes. Esto le dice al remitente que su ventana de congestión se ha vuelto demasiado grande. Consulte la Figura 4-20 para obtener una descripción general del inicio lento en acción.



*Figura 4-20 TCP: Inicio lento en acción*

##### Evitar la congestión

La suposición del algoritmo es que la pérdida de paquetes causada por daños es muy pequeña (mucho menos del 1%). Por lo tanto, la pérdida de un paquete indica congestión en algún lugar de la red entre el origen y el destino. Hay dos indicaciones de pérdida de paquetes:

Se agota el tiempo de espera.

Se reciben confirmaciones duplicadas.

La evitación de la congestión y el arranque lento son algoritmos independientes con objetivos diferentes. Pero cuando se produce una congestión, TCP debe reducir su velocidad de transmisión de paquetes a la red e invocar un inicio lento para que las cosas vuelvan a funcionar. En la práctica, se implementan conjuntamente.

La evitación de la congestión y el arranque lento requieren que se mantengan dos variables para cada conexión:

Una ventana de congestión, cwnd Un tamaño de umbral de inicio lento, ssthresh

El algoritmo combinado funciona de la siguiente manera:

1. La inicialización de una conexión determinada establece cwnd en un segmento y ssthresh en 65535 bytes.
2. La rutina de salida TCP nunca envía más que el valor inferior de cwnd o la ventana anunciada del receptor.
3. Cuando se produce una congestión (tiempo de espera o confirmación duplicada), la mitad del tamaño de la ventana actual se guarda en ssthresh. Además, si la congestión se indica mediante un tiempo de espera, cwnd se establece en un segmento.
4. Cuando el otro extremo reconoce nuevos datos, aumente cwnd, pero la forma en que aumenta depende de si TCP realiza un inicio lento o evita la congestión. Si cwnd es menor o igual que ssthresh, TCP está en inicio lento; de lo contrario, TCP está realizando la evitación de la congestión.

El inicio lento continúa hasta que TCP está a mitad de camino de donde estaba cuando se produjo la congestión (ya que registró la mitad del tamaño de la ventana que causó el problema en el paso 2) y, a continuación, la evitación de la congestión toma el control. El inicio lento tiene que comenzar en un segmento y se incrementa en un segmento cada vez que se recibe una confirmación. Como se mencionó anteriormente, esto abre la ventana exponencialmente: envíe un segmento, luego dos, luego cuatro, y así sucesivamente.

La prevención de la congestión dicta que cwnd se incremente en segsize\*segsize/cwnd cada vez que se recibe un ACK, donde segsize es el tamaño del segmento y cwnd se mantiene en bytes. Este es un crecimiento lineal de cwnd, en comparación con el crecimiento exponencial de inicio lento. El aumento en cwnd debe ser como máximo un segmento en cada tiempo de ida y vuelta (independientemente de cuántos ACK se reciban en ese tiempo de ida y vuelta), mientras que el inicio lento incrementa cwnd por el número de ACK recibidos en un tiempo de ida y vuelta. Muchas implementaciones agregan incorrectamente una pequeña fracción del tamaño del segmento (normalmente el tamaño del segmento dividido por 8) durante la prevención de la congestión. Esto es incorrecto y no debe emularse en futuras versiones. Consulte la Figura 4-21 para ver un ejemplo de arranque lento de TCP y prevención de congestión en acción.

Tiempos de ida y vuelta

CWND

cwnd

ssthresh

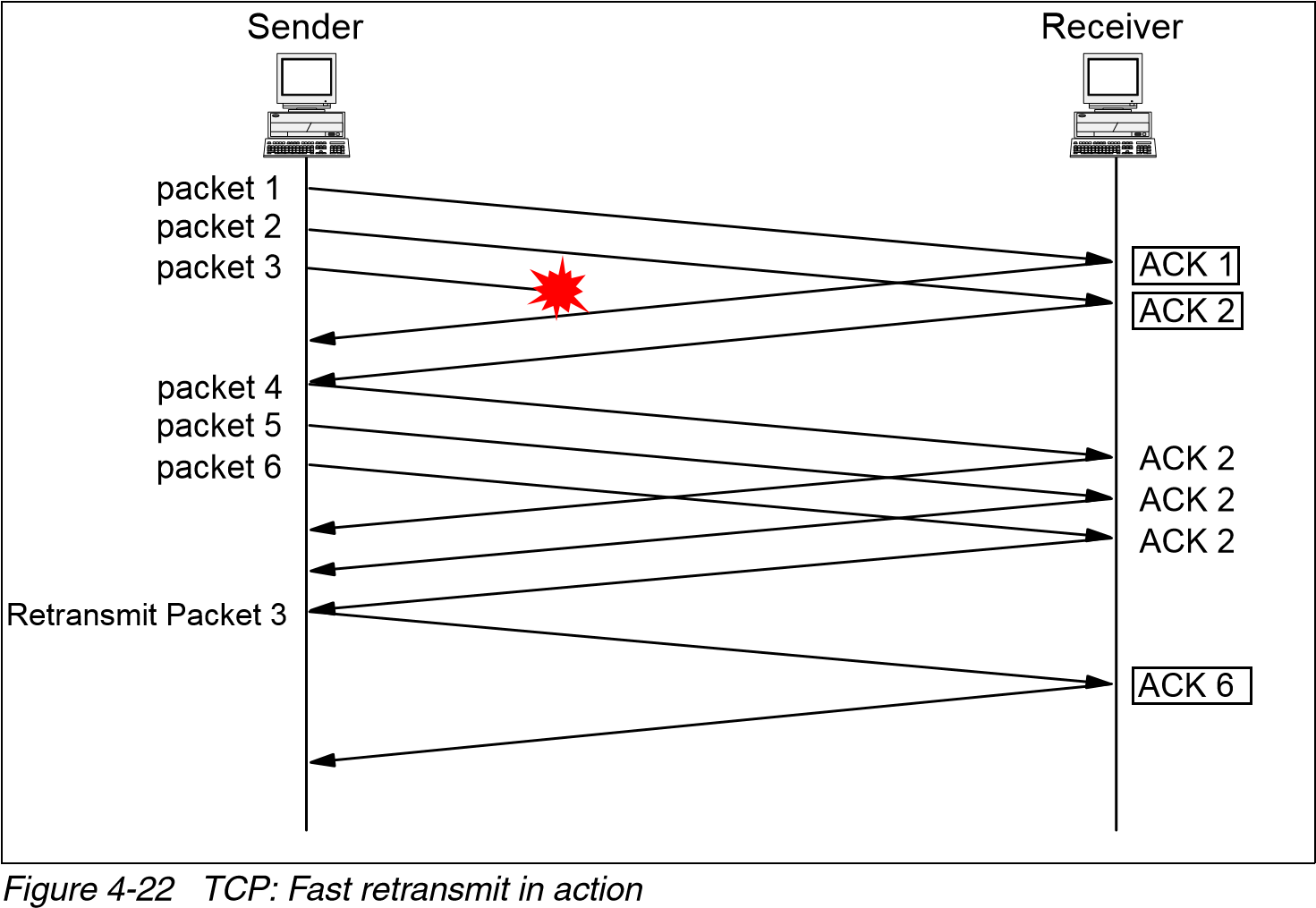
*Figura 4-21 TCP: Comportamiento de arranque lento y evitación de congestión en acción*

##### Retransmisión rápida

La retransmisión rápida evita que TCP espere un tiempo de espera para reenviar los segmentos perdidos.

En 1990 se propusieron modificaciones al algoritmo para evitar la congestión. Antes de describir el cambio, tenga en cuenta que TCP puede generar una confirmación inmediata (una confirmación duplicada) cuando se recibe un segmento fuera de servicio. Esta confirmación duplicada no debe retrasarse. El propósito de este ACK duplicado es informar al otro extremo que se recibió un segmento fuera de orden y decirle qué número de secuencia se espera.

Debido a que TCP no sabe si una confirmación duplicada se debe a un segmento perdido o simplemente a un reordenamiento de segmentos, espera a que se reciba una pequeña cantidad de confirmaciones duplicadas. Se supone que si solo hay un reordenamiento de la segmentos, solo habrá uno o dos ACK duplicados antes de que se procese el segmento reordenado, que luego generará un nuevo ACK. Si se reciben tres o más confirmaciones duplicadas seguidas, es una fuerte indicación de que se ha perdido un segmento. A continuación, TCP realiza una retransmisión de lo que parece ser el segmento que falta, sin esperar a que expire un temporizador de retransmisión. Consulte la Figura 4-22 para obtener una descripción general de la retransmisión rápida TCP en acción.



##### Recuperación rápida

Después de que la retransmisión rápida envía lo que parece ser el segmento que falta, se evita la congestión, pero no se realiza un inicio lento. Este es el algoritmo de recuperación rápida. Es una mejora que permite un alto rendimiento bajo una congestión moderada, especialmente para ventanas grandes.

La razón para no realizar un inicio lento en este caso es que la recepción de los confirmadores duplicados le dice a TCP que se ha perdido algo más que un paquete. Debido a que el receptor solo puede generar el ACK duplicado cuando se recibe otro segmento, ese segmento ha abandonado la red y está en el búfer del receptor. Es decir, todavía hay datos que fluyen entre los dos extremos, y TCP no quiere reducir el flujo abruptamente entrando en un inicio lento. Los algoritmos de retransmisión rápida y recuperación rápida generalmente se implementan juntos de la siguiente manera:

1. Cuando se recibe el tercer ACK duplicado seguido, establezca ssthresh en la mitad de la ventana de congestión actual, cwnd, pero no menos de dos segmentos. Retransmita el segmento que falta. Establezca cwnd en ssthresh más tres veces el tamaño del segmento. Esto infla la ventana de congestión por el número de segmentos que han salido de la red y el otro extremo se ha almacenado en caché (3).
2. Cada vez que llegue otro ACK duplicado, incremente cwnd por el tamaño del segmento. Esto infla la ventana de congestión para el segmento adicional que ha abandonado la red. Transmita un paquete, si lo permite el nuevo valor de cwnd.
3. Cuando llegue la siguiente confirmación que confirme nuevos datos, establezca cwnd en ssthresh (el valor establecido en el paso 1). Este ACK es el acuse de recibo de la retransmisión desde el paso 1, un tiempo de ida y vuelta después de la retransmisión. Además, este ACK reconoce todos los segmentos intermedios enviados entre el paquete perdido y la recepción del primer ACK duplicado. Este paso evita la congestión, ya que TCP se ha reducido a la mitad de la velocidad a la que estaba cuando se perdió el paquete.

### 4.4 RFC relevantes para este capítulo

Las siguientes RFC proporcionan información detallada sobre los protocolos y arquitecturas de conexión presentados a lo largo de este capítulo:

RFC 761 - Protocolo de control de transmisión estándar del Departamento de Defensa (enero de 1980)

RFC 768 – Protocolo de datagramas de usuario (agosto de 1980)

RFC 793 - Actualizado por RFC 3168 - La adición de congestión explícita

Notificación (ECN) a IP (septiembre de 2001)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **5** | |

## Capítulo 5. Protocolos de enrutamiento

En este capítulo se proporciona una descripción general del enrutamiento IP y se analizan los diversos protocolos de enrutamiento utilizados.

Una de las funciones básicas que proporciona el protocolo IP es la capacidad de formar conexiones entre diferentes redes físicas. Un sistema que realiza esta función se denomina *router IP*. Este tipo de dispositivo se conecta a dos o más redes físicas y reenvía datagramas entre las redes.

Al enviar datos a un destino remoto, un host pasa datagramas a un router local. El router reenvía los datagramas hacia el destino final. Viajan de un router a otro hasta llegar a un router conectado al segmento LAN del destino. Cada enrutador a lo largo de la ruta de extremo a extremo selecciona el dispositivo de *salto siguiente* utilizado para llegar al destino. El siguiente salto representa el siguiente dispositivo a lo largo de la ruta para llegar al destino. Se encuentra en una red física conectada a este sistema intermedio. Dado que esta red física difiere de aquella en la que el sistema recibió originalmente el datagrama, el host intermedio ha *reenviado* (es decir, enrutado) el datagrama IP de una red física a otra.

© Derechos de autor IBM Corp. 1989-2006. Todos los derechos reservados.

La Figura 5-1 muestra un entorno en el que el host C está posicionado para reenviar paquetes entre la red X y la red Y.

*Figura 5-1 Operaciones de enrutamiento IP*

Aplicación

TCP

IP

Interfaz X

Anfitrión A

Aplicación

TCP

IP

Interfaz Y

Anfitrión

B

Host C que actúa como

Enrutador

Enrutamiento IP

Interfaz X

Interfaz Y

Red Y

Red X

La tabla de enrutamiento IP de cada dispositivo se utiliza para reenviar paquetes entre segmentos de red. La tabla básica contiene información sobre las redes conectadas localmente de un router. La configuración del dispositivo se puede ampliar para contener información detallada de redes remotas. Esta información proporciona una visión más completa del entorno general.

Un protocolo de enrutamiento robusto proporciona la capacidad de crear y administrar dinámicamente la información en la tabla de enrutamiento IP. A medida que se producen cambios en la topología de la red, las tablas de enrutamiento se actualizan con una intervención manual mínima o nula. En este capítulo se detallan varios protocolos de enrutamiento IP y cómo cada protocolo administra esta información.

**Nota:** En otras secciones de este libro, se muestra la posición de cada protocolo dentro del modelo en capas de la pila de protocolos OSI. La función de enrutamiento se incluye como parte de la capa entre redes. Sin embargo, la función principal de un protocolo de enrutamiento es intercambiar información de enrutamiento con otros enrutadores. En este sentido, los protocolos de enrutamiento se comportan más como un protocolo de aplicación. Por lo tanto, este capítulo no intenta representar la posición de estos protocolos dentro de la pila general de protocolos.

**Nota:** La documentación de enrutamiento IP anterior a menudo se refería a un enrutador IP como una *puerta de enlace IP.*

### 5.1 Sistemas autónomos

La definición de un sistema autónomo (SA) es fundamental para comprender la función y alcance de un protocolo de enrutamiento. Un AS se define como una parte lógica de

una red IP más grande. Normalmente, un AS consiste en una red intermedia dentro de una organización. Es administrado por una única autoridad de gestión. Como se muestra en la Figura 5-2, un AS puede conectarse a otros sistemas autónomos administrados por la misma organización. Alternativamente, puede conectarse a otras redes públicas o privadas.

Enrutador

Enrutador

IGPs

Sistema autónomo A

Enrutador

IGPs

Sistema Autónomo C

Enrutador

Enrutador

Enrutador

Autoridad Única de Gestión

EGP

Enrutador

Enrutador

Enrutador

Sistema Autónomo B

IGPs

Internet

*Figura 5-2 Sistemas autónomos*

Algunos protocolos de enrutamiento se utilizan para determinar las rutas de enrutamiento dentro de un AS. Otros se utilizan para interconectar un conjunto de sistemas autónomos:

Protocolos de puerta de enlace interior (IGP): Los protocolos de puerta de enlace interior permiten a los routers intercambiar información dentro de un AS. Ejemplos de estos protocolos son: Open

Short Path First (OSPF) y Routing Information Protocol (RIP).

Protocolos de puerta de enlace exterior (EGP): Los protocolos de puerta de enlace exterior permiten el intercambio de información resumida entre sistemas autónomos. Un ejemplo de este tipo de protocolo de enrutamiento es el protocolo de puerta de enlace fronteriza (BGP).

La Figura 5-2 en la página 173 muestra la interacción entre los protocolos de puerta de enlace interior y exterior. Muestra los protocolos de puerta de enlace interior utilizados para mantener la información de ruteo dentro de cada AS. La figura también muestra los protocolos de puerta de enlace exterior que mantienen la información de enrutamiento entre sistemas autónomos.

Dentro de un AS, se pueden utilizar varios procesos de enrutamiento interior. Cuando esto ocurre, el AS debe aparecer a los demás sistemas autónomos como si tuviera un único plan de enrutamiento interior coherente. El AS debe presentar una visión coherente de los destinos internos.

### 5.2 Tipos de encaminamiento IP y algoritmos de encaminamiento IP

Los algoritmos de enrutamiento crean y mantienen la tabla de enrutamiento IP en un dispositivo. Hay dos métodos principales que se usan para crear la tabla de enrutamiento:

Enrutamiento estático: El enrutamiento estático utiliza definiciones preprogramadas que representan rutas a través de la red.

Enrutamiento dinámico: Los algoritmos de enrutamiento dinámico permiten que los enrutadores descubran y mantengan automáticamente el conocimiento de las rutas a través de la red. Esta detección automática puede utilizar una serie de protocolos de enrutamiento dinámico disponibles actualmente. La diferencia entre estos protocolos es la forma en que descubren y calculan nuevas rutas a las redes de destino. Se pueden clasificar en cuatro grandes categorías:

* Protocolos de vectores de distancia
* Protocolos de estado de enlace
* Protocolos de vectores de ruta
* Protocolos híbridos

En el resto de esta sección se describe el funcionamiento de cada algoritmo.

Hay varias razones para la multiplicidad de protocolos:

El enrutamiento dentro de una red y el enrutamiento entre redes suelen tener diferentes requisitos de seguridad, estabilidad y escalabilidad. Se han desarrollado diferentes protocolos de enrutamiento para abordar estos requisitos.

Se han desarrollado nuevos protocolos para abordar las deficiencias observadas en los protocolos establecidos.

Las redes de diferentes tamaños pueden utilizar diferentes algoritmos de enrutamiento. Pequeño a Las redes de tamaño mediano a menudo utilizan protocolos de enrutamiento que reflejan la simplicidad del entorno.

Sin embargo, estos protocolos no se escalan para admitir grandes redes interconectadas. Se requieren algoritmos de enrutamiento más complejos para admitir estos entornos.

#### 5.2.1 Encaminamiento estático

El enrutamiento estático lo realiza manualmente el administrador de red. El administrador es responsable de descubrir y propagar rutas a través de la red. Estas definiciones se programan manualmente en todos los dispositivos de enrutamiento del entorno.

Una vez que se ha configurado un dispositivo, simplemente reenvía los paquetes a los puertos predeterminados. No hay comunicación entre los routers con respecto a la topología actual de la red.

En redes pequeñas con redundancia mínima, este proceso es relativamente sencillo de administrar. Sin embargo, este enfoque tiene varias desventajas para mantener las tablas de enrutamiento IP:

Las rutas estáticas requieren una cantidad considerable de coordinación y mantenimiento en entornos de red no triviales.

Las rutas estáticas no pueden adaptarse dinámicamente al estado operativo actual de la red. Si una subred de destino se vuelve inalcanzable, las rutas estáticas que apuntan a esa red permanecen en la tabla de enrutamiento. El tráfico continúa siendo redirigido hacia ese destino. A menos que el administrador de red actualice las rutas estáticas para reflejar la nueva topología, el tráfico no podrá utilizar ninguna ruta alternativa que pueda existir.

Normalmente, las rutas estáticas solo se utilizan en topologías de red simples. Sin embargo, hay circunstancias adicionales en las que el enrutamiento estático puede ser atractivo. Por ejemplo, se pueden utilizar rutas estáticas:

Para definir manualmente una ruta predeterminada. Esta ruta se utiliza para reenviar el tráfico cuando la tabla de enrutamiento no contiene una ruta más específica al destino.

Definir una ruta que no se anuncia automáticamente dentro de una red.

Cuando la utilización o las tarifas de línea hacen que no sea deseable enviar tráfico de anuncio de enrutamiento a través de conexiones WAN de menor capacidad.

Cuando se requieren políticas de enrutamiento complejas. Por ejemplo, las rutas estáticas se pueden utilizar para garantizar que el tráfico destinado a un host específico atraviese una ruta de red designada.

Para proporcionar un entorno de red más seguro. El administrador conoce todas las subredes definidas en el entorno. El administrador autoriza específicamente todas las comunicaciones permitidas entre estas subredes.

Proporcionar una utilización más eficiente de los recursos. Este método de administración de tablas de enrutamiento no requiere ancho de banda de red para anunciar rutas entre dispositivos vecinos. También utiliza menos memoria del procesador y ciclos de CPU para calcular las rutas de red.

#### 5.2.2 Encaminamiento vectorial de distancia

Los algoritmos de vectores de distancia son ejemplos de protocolos de enrutamiento dinámico. Estos algoritmos permiten que cada dispositivo de la red construya y mantenga automáticamente una tabla de enrutamiento IP local.

El principio detrás del enrutamiento vectorial de distancia es simple. Cada router de la red interconectada mantiene la *distancia* o el *coste* desde sí mismo hasta cada destino conocido. Este valor representa la conveniencia general de la ruta. Las rutas asociadas a un valor de coste menor son más atractivas de usar que las rutas asociadas a un valor mayor. La ruta representada por el menor coste se convierte en la ruta preferida para llegar al destino.

Esta información se mantiene en una *tabla de vectores de distancia.* La tabla se actualiza periódicamente

anunciados a cada router vecino. Cada router procesa estos anuncios para determinar las mejores rutas a través de la red.

La principal ventaja de los algoritmos de vectores de distancia es que suelen ser fáciles de implementar y depurar. Son muy útiles en redes pequeñas con redundancia limitada. Sin embargo, existen varias desventajas con este tipo de protocolo:

Durante una condición adversa, el tiempo que tarda cada dispositivo de la red en producir una tabla de enrutamiento precisa se denomina *tiempo de convergencia*. En interredes grandes y complejas que utilizan algoritmos de vectores de distancia, este tiempo puede ser excesivo. Mientras las tablas de enrutamiento convergen, las redes son susceptibles a un comportamiento de enrutamiento inconsistente. Esto puede causar bucles de enrutamiento u otros tipos de reenvío de paquetes inestables.

Para reducir el tiempo de convergencia, a menudo se establece un límite en el número máximo de saltos contenidos en una sola ruta. Las rutas válidas que superen este límite no se pueden utilizar en redes vectoriales de distancia.

Las tablas de enrutamiento de vectores de distancia se transmiten periódicamente a los dispositivos vecinos. Se envían incluso si no se han realizado cambios en el contenido de la tabla. Esto puede provocar períodos notables de mayor utilización en entornos de capacidad reducida.

Se han desarrollado mejoras en el algoritmo básico del vector de distancia para reducir las exposiciones de convergencia e inestabilidad. Describimos estas mejoras en 5.3.5, "Convergencia y conteo hasta el infinito" en la página 185.

RIP es un ejemplo popular de un protocolo de enrutamiento de vector de distancia.

#### 5.2.3 Encaminamiento del estado del enlace

El crecimiento del tamaño y la complejidad de las redes en los últimos años ha requirió el desarrollo de algoritmos de enrutamiento más robustos. Estos Los algoritmos abordan las deficiencias observadas en los protocolos de vectores de distancia.

Estos algoritmos utilizan el principio de un *estado de enlace* para determinar la topología de la red. Un estado de enlace es la descripción de una interfaz en un router (por ejemplo, dirección IP, máscara de subred, tipo de red) y su relación con los routers vecinos. La colección de estos estados de vínculo forma una base de datos de estado de vínculo.

El proceso utilizado por los algoritmos de estado de enlace para determinar la topología de red es sencillo:

1. Cada router identifica todos los demás dispositivos de enrutamiento en las redes conectadas directamente.
2. Cada router anuncia una lista de todos los enlaces de red conectados directamente y el coste asociado de cada enlace. Esto se realiza a través del intercambio de anuncios de estado de enlace (LSA) con otros enrutadores de la red.
3. Con estos anuncios, cada router crea una base de datos que detalla la topología de red actual. La base de datos de topología de cada router es idéntica.
4. Cada router utiliza la información de la base de datos de topología para calcular las rutas más deseables a cada red de destino. Esta información se utiliza para actualizar la tabla de enrutamiento IP.

##### Algoritmo Shortest-Path First (SPF)

El algoritmo SPF se utiliza para procesar la información de la base de datos de topologías. Proporciona una representación en árbol de la red. El dispositivo que ejecuta el algoritmo SPF es la raíz del árbol. El resultado del algoritmo es la lista de rutas más cortas a cada red de destino. La Figura 5-3 en la página 178 proporciona un ejemplo del algoritmo de ruta más corta ejecutado en el router A.

*Figura 5-3 Ejemplo de la ruta más corta primero (SPF)*

Un

B

C

D

Estado del vínculo

Base de datos

4

2

1

D

1

3

3

E

Un

B

C

D

B-2

C-1

A-2

D-4

A-1

D-1

E-3

C-1

B-4

E-3

C-3

D-3

Un

B

C

D

E

Dado que cada enrutador procesa el mismo conjunto de LSA, cada enrutador crea una base de datos de estado de vínculo idéntica. Sin embargo, debido a que cada dispositivo ocupa un lugar diferente en la topología de la red, la aplicación del algoritmo SPF produce un árbol diferente para cada enrutador.

El protocolo OSPF es un ejemplo popular de un protocolo de ruteo de estado de vínculo.

#### 5.2.4 Encaminamiento vectorial de trayecto

El enrutamiento vectorial de ruta se describe en RFC 1322; los siguientes párrafos se basan en el RFC.

El algoritmo de enrutamiento de vector de ruta es algo similar al algoritmo de vector de distancia en el sentido de que cada enrutador de borde anuncia los destinos a los que puede llegar a su enrutador vecino. Sin embargo, en lugar de anunciar las redes en términos de un destino y la distancia a ese destino, las redes se anuncian como direcciones de destino y descripciones de rutas para llegar a ellos Destinos.

Una ruta se define como un emparejamiento entre un destino y los atributos de la ruta a ese destino, de ahí el nombre, enrutamiento vectorial de ruta, donde los routers reciben un vector que contiene rutas a un conjunto de destinos.

La ruta, expresada en términos de los dominios (o confederaciones) atravesados hasta el momento, se transporta en un atributo de ruta especial que registra la secuencia de dominios de enrutamiento a través de los cuales ha pasado la información de accesibilidad. La ruta representada por el menor número de dominios se convierte en la ruta preferida para llegar al destino.

La principal ventaja de un protocolo de vector de ruta es su flexibilidad. Hay varias otras ventajas relacionadas con el uso de un protocolo de vector de ruta:

La complejidad computacional es menor que la del protocolo de estado de enlace. El cálculo del vector de ruta consiste en evaluar una ruta recién llegada y compararla con la existente, mientras que el cálculo del estado del enlace convencional requiere la ejecución de un algoritmo SPF.

El enrutamiento vectorial de ruta no requiere que todos los dominios de enrutamiento tengan políticas homogéneas para la selección de rutas; Las directivas de selección de rutas utilizadas por un dominio de enrutamiento no son necesariamente conocidas por otros dominios de enrutamiento. El soporte para políticas de selección de rutas heterogéneas tiene serias implicaciones para la complejidad computacional. El protocolo de vector de ruta permite que cada dominio realice su selección de ruta de forma autónoma, basándose únicamente en políticas locales. Sin embargo, el enrutamiento vectorial de ruta puede acomodar la selección de rutas heterogéneas con poco costo adicional.

Solo los dominios cuyas rutas se ven afectadas por los cambios tienen que volver a calcularse.

La supresión de bucles de enrutamiento se implementa a través del atributo de ruta, en contraste con el estado del enlace y el vector de distancia, que utilizan una métrica monótona definida globalmente, lo que aumenta la selección de rutas. Por lo tanto, se acomodan diferentes definiciones de confederación porque se evita el bucle mediante el uso de información de ruta completa.

El cálculo de rutas precede a la difusión de información de enrutamiento. Por lo tanto, solo la información de enrutamiento asociada con las rutas seleccionadas por un dominio se distribuye a los dominios adyacentes.

El enrutamiento vectorial de ruta tiene la capacidad de ocultar información de forma selectiva.

Sin embargo, este enfoque tiene desventajas, entre las que se incluyen: Los cambios de topología solo dan lugar a la recomputación de las rutas afectadas por estos

cambios, que es más eficiente que el recálculo completo. Sin embargo, debido a la inclusión de información de ruta completa con cada vector de distancia, el efecto de un cambio de topología puede propagarse más lejos que en los algoritmos tradicionales de vectores de distancia.

A menos que la topología de red esté completamente mallada o pueda parecerlo, los bucles de enrutamiento pueden convertirse en un problema.

BGP es un ejemplo popular de un protocolo de enrutamiento de vector de ruta.

#### 5.2.5 Encaminamiento híbrido

La última categoría de protocolos de enrutamiento son los protocolos híbridos. Estos protocolos intentan combinar los atributos positivos de los protocolos de vector de distancia y de estado de enlace. Al igual que el vector de distancia, los protocolos híbridos utilizan métricas para asignar una preferencia a una ruta. Sin embargo, las métricas son más precisas que los protocolos convencionales de vectores de distancia. Al igual que los algoritmos de estado de vínculo, las actualizaciones de enrutamiento en protocolos híbridos se basan en eventos en lugar de ser periódicas. Las redes que utilizan protocolos híbridos tienden a converger más rápidamente que las redes que utilizan protocolos de vectores de distancia. Por último, estos protocolos reducen potencialmente los costes de las actualizaciones del estado de los enlaces y los anuncios de vectores de distancia.

Aunque existen protocolos híbridos abiertos, esta categoría se asocia casi exclusivamente con el algoritmo EIGRP propietario. EIGRP fue desarrollado por Cisco Systems, Inc.

### 5.3 Protocolo de información de enrutamiento (RIP)

RIP es un ejemplo de un protocolo de puerta de enlace interior diseñado para su uso en pequeños sistemas autónomos. RIP se basa en el protocolo de enrutamiento XNS de Xerox. Las primeras implementaciones de RIP fueron fácilmente aceptadas porque el código se incorporó en el sistema operativo basado en UNIX de Berkeley Software Distribution (BSD). RIP es un protocolo de vector de distancia.

A mediados de 1988, el IETF emitió el RFC 1058 con actualizaciones en RFC2453, que describe las operaciones estándar de un sistema RIP. Sin embargo, el RFC se emitió después de que se completaran muchas implementaciones de RIP. Por esta razón, algunos sistemas RIP no admiten todo el conjunto de mejoras del algoritmo básico de vector de distancia (por ejemplo, inverso de veneno y actualizaciones activadas).

#### 5.3.1 Tipos de paquetes RIP

El protocolo RIP especifica dos tipos de paquetes. Estos paquetes pueden ser enviados por cualquier dispositivo que ejecute el protocolo RIP:

Paquetes de solicitud: Un paquete de solicitud consulta a los dispositivos RIP vecinos para obtener su tabla de vectores de distancia. La solicitud indica si el vecino debe devolver un subconjunto específico o todo el contenido de la tabla.

Paquetes de respuesta: Un dispositivo envía un paquete de respuesta para anunciar la información mantenida en su tabla de vectores de distancia local. La tabla se envía durante las siguientes situaciones:

* La tabla se envía automáticamente cada 30 segundos.
* La tabla se envía como respuesta a un paquete de solicitud generado por otro nodo RIP.
* Si se admiten actualizaciones activadas, la tabla se envía cuando hay un cambio en la tabla de vectores de distancia local. Discutimos las actualizaciones activadas en "Actualizaciones activadas" en la página 188.

Cuando un dispositivo recibe un paquete de respuesta, la información contenida en la actualización se compara con la tabla de vectores de distancia local. Si la actualización contiene una ruta de menor costo a un destino, la tabla se actualiza para reflejar la nueva ruta.

#### 5.3.2 Formato de paquete RIP

RIP utiliza un formato de paquete específico para compartir información sobre las distancias a destinos de red conocidos. Los paquetes RIP se transmiten mediante datagramas UDP. RIP envía y recibe datagramas utilizando el puerto UDP 520.

Los datagramas RIP tienen un tamaño máximo de 512 octetos. Las actualizaciones que superen este tamaño deben anunciarse en varios datagramas. En entornos LAN, los datagramas RIP se envían utilizando la dirección de difusión MAC de todas las estaciones y una dirección de difusión de red IP. En entornos punto a punto o sin difusión, los datagramas se dirigen específicamente al dispositivo de destino.

El formato del paquete RIP se muestra en la Figura 5-4.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Número de octetos  Solicitud=1  1Respuesta=2  Versión = 1  1  2  Dirección Familia  2Identificador de IP  2   |  |  | | --- | --- | | Mandar |  | | Versión | | Reservado | | AFI: X'0002' | | Reservado | | Dirección IP | | Reservado | | Métrico | |  |   4 }Entrada de enrutamiento: Se puede repetir  }  8  4 |

*Figura 5-4 Formato de paquete RIP*

Un tamaño de paquete de 512 bytes permite incluir un máximo de 25 entradas de enrutamiento en un solo anuncio RIP.

#### 5.3.3 Modos de funcionamiento RIP

Los hosts RIP tienen dos modos de funcionamiento:

Modo activo: los dispositivos que funcionan en modo activo anuncian su tabla de vectores de distancia y también reciben actualizaciones de enrutamiento de hosts RIP vecinos. Por lo general, los dispositivos de enrutamiento están configurados para funcionar en modo activo.

Modo pasivo (o silencioso): Los dispositivos que funcionan en este modo simplemente reciben actualizaciones de enrutamiento de los dispositivos RIP vecinos. No anuncian su tabla de vectores de distancia. Por lo general, las estaciones finales están configuradas para funcionar en modo pasivo.

#### 5.3.4 Cálculo de vectores de distancia

La tabla de vectores de distancia describe cada red de destino. Las entradas de esta tabla contienen la siguiente información:

La red de destino (vector) descrita por esta entrada en la tabla.

El costo asociado (distancia) del camino más atractivo para llegar a este destino. Esto proporciona la capacidad de diferenciar entre varias rutas a un destino. En este contexto, los términos distancia y costo pueden inducir a error. No tienen relación directa con la distancia física o el costo monetario.

La dirección IP del dispositivo de salto siguiente utilizado para llegar a la red de destino.

Cada vez que un dispositivo recibe un anuncio de tabla de enrutamiento, se procesa para determinar si se puede llegar a algún destino mediante una ruta de menor costo. Esto se hace utilizando el algoritmo de vector de distancia RIP. El algoritmo se puede resumir en:

En la inicialización del enrutador, cada dispositivo contiene una tabla de vectores de distancia que enumera cada red conectada directamente y el costo configurado. Normalmente, a cada red se le asigna un coste de 1. Esto representa un solo salto a través de la red. El número total de saltos en una ruta es igual al costo total de la ruta. Sin embargo, el costo se puede cambiar para reflejar otras medidas, como la utilización, la velocidad o la confiabilidad.

Cada router transmite periódicamente (normalmente cada 30 segundos) su tabla de vectores de distancia a cada uno de sus vecinos. El router también puede transmitir la tabla cuando se produce un cambio de topología. Cada router utiliza esta información para actualizar su tabla de vectores de distancia local:

* El coste total para cada destino se calcula sumando el coste notificado en la tabla de vectores de distancia de un vecino al coste del vínculo a ese vecino. La ruta con el menor coste se almacena en la tabla de vectores de distancia.
* Todas las actualizaciones reemplazan automáticamente la información anterior en la tabla de vectores de distancia. Esto permite que RIP mantenga la integridad de las rutas en la tabla de enrutamiento.

La tabla de enrutamiento IP se actualiza para reflejar la ruta de menor costo a cada destino.

La Figura 5-5 ilustra las tablas de vectores de distancia para tres routers dentro de una interred simple.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N2  R1 | | | | N3  N4  N5  N6  R2  R3  R4  R5 |
| N1  Enrutador R2  Vector de distancia  Mesa | | | | Enrutador R3 Enrutador R4  Vector de distancia Vector de distancia  Mesa Mesa   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Red | Próximo salto | Métrico |  | Red | Próximo  Lúpulo | Métrico | | N1 | R2 | 3 | N1 | R3 | 4 | | N2 | R2 | 2 | N2 | R3 | 3 | | N3 | Directo | 1 | N3 | R3 | 2 | | N4 | Directo | 1 | N4 | Directo | 1 | | N5 | R4 | 2 | N5 | Directo | 1 | | N6 | R4 | 3 | N6 | R5 | 2 | |
|  | Red | Próximo salto | Métrico |
|  | N1 | R1 | 2 |
|  | N2 | Directo | 1 |
|  | N3 | Directo | 1 |
|  | N4 | R3 | 2 |
|  | N5 | R3 | 3 |
|  | N6 | R3 | 4 |
|  | | | |

*Figura 5-5 Tabla de enrutamiento de vectores de distancia de muestra*

#### 5.3.5 Convergencia y conteo hasta el infinito

Con el tiempo suficiente, este algoritmo calculará correctamente el vector de distancia en cada dispositivo. Sin embargo, durante este tiempo de convergencia, las rutas erróneas puede propagarse a través de la red. La figura 5-6 muestra este problema.

*Figura 5-6 Conteo hasta el infinito de la red de muestras*

Blanco

Red

Un

B

C

D

(

n) = Costo de red

(1)

(1)

(1)

(1)

(1)

(10)

Esta red contiene cuatro routers interconectados. Cada enlace tiene un costo de 1, excepto el enlace que conecta el router C y el router D; Este enlace tiene un coste de 10. Los costos se han definido de modo que el reenvío de paquetes en el link que conecta el router C y el router D no sea deseable. Una vez que la red ha convergido, cada dispositivo tiene información de enrutamiento que describe todas las redes.

Por ejemplo, para llegar a la red de destino, los routers tienen la siguiente información:

Router D a la red de destino: Red conectada directamente. La métrica es 1.

Enrutador B a la red de destino: El siguiente salto es el enrutador D. La métrica es 2.

Router C a la red de destino: El siguiente salto es el router B. La métrica es 3. Enrutador A a la red de destino: El siguiente salto es el enrutador B. La métrica es 3.

Considere una condición adversa en la que falla el link que conecta el router B y el router D. Una vez que la red ha vuelto a converger, todas las rutas utilizan el enlace que conecta el router C y el router D para llegar a la red de destino. Sin embargo, este tiempo de reconvergencia puede ser considerable. La Figura 5-7 ilustra cómo se actualizan las rutas a la red de destino a lo largo del período de reconvergencia. Para simplificar, esta figura supone que todos los routers envían actualizaciones al mismo tiempo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tiempo D: | Directo | 1 | Directo | 1 | Directo 1 | Directo | 1 | -- | Directo | 1 | Directo | 1 |
| B: | Inalcanzable |  | C | 4 | C 5 | C | 6 |  | C | 11 | C | 12 |
| C: | B | 3 | Un | 4 | Un 5 | Un | 6 |  | Un | 11 | D | 11 |
| Un: | B | 3 | C | 4 | C 5 | C | 6 | -- | C | 11 | C | 12 |

*Figura 5-7 Secuencia de convergencia de red*

La reconvergencia comienza cuando el router B se da cuenta de que la ruta al router D no está disponible. El router B puede eliminar inmediatamente la ruta fallida porque se ha agotado el tiempo de espera del link. Sin embargo, pasa una cantidad considerable de tiempo antes de que los otros routers eliminen sus referencias a la ruta fallida. Esto se describe en la secuencia de actualizaciones que se muestra en la Figura 5-7:

1. Antes de que ocurra la condición adversa, el router A y el router C tienen una ruta a la red de destino a través del router B.
2. La condición adversa se produce cuando falla el link que conecta el router D y el router B. El enrutador B reconoce que su ruta preferida a la red de destino ahora no es válida.
3. El enrutador A y el enrutador C continúan enviando actualizaciones que reflejan la ruta a través del enrutador B. Esta ruta no es válida porque el link que conecta el router D y el router B ha fallado.
4. El enrutador B recibe las actualizaciones del enrutador A y el enrutador C. El enrutador B cree que ahora debe enrutar el tráfico a la red de destino a través del enrutador A o el enrutador C. En realidad, esta no es una ruta válida, porque las rutas en el router A y el router C son vestigios de la ruta anterior a través del router B.
5. Utilizando el anuncio de ruteo enviado por el router B, el router A y el router C pueden determinar que la ruta a través del router B ha fallado. Sin embargo, el router A y el router C ahora creen que la ruta preferida existe a través del partner.

La convergencia de la red continúa a medida que el router A y el router C se involucran en un período prolongado de engaño mutuo. Cada dispositivo afirma ser capaz de llegar a la red de destino a través del dispositivo asociado. La ruta para llegar a la red de destino ahora contiene un bucle de enrutamiento.

La manera en que los costos en la tabla de vectores de distancia incrementan da lugar al término *Contar hasta el infinito*. Los costos siguen aumentando, teóricamente hasta el infinito. Para minimizar esta exposición, siempre que una red no esté disponible, el El incremento de métricas a través de actualizaciones de enrutamiento debe detenerse tan pronto como se práctico para hacerlo. En un entorno RIP, los costes siguen aumentando hasta que alcanzan un valor máximo de 16. Este límite se define en RFC 1058.

Un efecto secundario del límite métrico es que también limita el número de saltos que un paquete puede atravesar desde la red de origen a la red de destino. En un entorno RIP, cualquier ruta que supere los 15 saltos se considera inválida. El algoritmo de enrutamiento descartará estas rutas.

Hay dos mejoras en el algoritmo básico de vectores de distancia que pueden minimizar el problema de contar hasta el infinito:

Horizonte dividido con veneno inverso Actualizaciones activadas

Estas mejoras no afectan al límite máximo de métricas.

##### Horizonte dividido

El tiempo de convergencia excesivo causado por el conteo hasta el infinito se puede reducir con el uso del horizonte dividido. Esta regla dicta que se impide que la información de ruteo salga del router en una interfaz a través de la cual se recibió la información.

La regla básica de horizonte dividido no se admite en RFC 1058. En su lugar, la norma especifica el horizonte de división mejorado con el algoritmo de inversión de veneno. La regla básica se presenta aquí para los antecedentes y la integridad. El algoritmo mejorado se revisa en la siguiente sección.

La incorporación del horizonte dividido modifica la secuencia de actualizaciones de enrutamiento que se muestran en la Figura 5-7 en la página 186. La nueva secuencia se muestra en la Figura 5-8. Las tablas muestran que la convergencia se produce considerablemente más rápido utilizando la regla del horizonte dividido.

Hora

D:

Directo

1

Directo

1

Directo

1

Directo

1

B:

Inalcanzable

Inalcanzable

Inalcanzable

C

12

C:

B

3

Un

4

D

11

D

11

Un:

B

3

C

4

Inalcanzable

C

12

Nota: Convergencia más rápida de la tabla de ruteo

*Figura 5-8 Convergencia de red con horizonte dividido*

La limitación de esta regla es que cada nodo debe esperar a que se agote el tiempo de espera de la ruta al destino inalcanzable antes de que la ruta se elimine de la tabla de vectores de distancia. En entornos RIP, este tiempo de espera es de al menos tres minutos después de la interrupción inicial. Durante ese tiempo, el dispositivo continúa proporcionando información errónea a otros nodos sobre el destino inalcanzable. Esto propaga bucles de enrutamiento y otras anomalías de enrutamiento.

##### Horizonte dividido con veneno inverso

Poison reverse es una mejora de la implementación estándar del horizonte dividido. Es compatible con RFC 1058. Con la inversión de veneno, todas las redes conocidas se anuncian en cada actualización de enrutamiento. Sin embargo, las redes aprendidas a través de una interfaz específica se anuncian como inalcanzables en los anuncios de enrutamiento enviados a esa interfaz.

Esto mejora drásticamente el tiempo de convergencia en entornos complejos y altamente redundantes. Con la inversión de veneno, cuando una actualización de enrutamiento indica que no se puede acceder a una red, las rutas se eliminan inmediatamente de la tabla de enrutamiento. Esto rompe las rutas erróneas y en bucle antes de que puedan propagarse a través de la red. Este enfoque difiere de la regla básica de horizonte dividido, en la que las rutas se eliminan mediante tiempos de espera.

La inversión de veneno no tiene ningún beneficio en redes sin redundancia (redes de ruta única).

Una desventaja de la inversión envenenada es que podría aumentar significativamente el tamaño de los anuncios de enrutamiento intercambiados entre vecinos. Esto se debe a que todas las rutas de la tabla de vectores de distancia se incluyen en cada anuncio. Aunque esto generalmente no es un problema en las redes de área local, puede causar períodos de mayor utilización en conexiones WAN de menor capacidad.

##### Actualizaciones activadas

Al igual que el horizonte dividido con veneno inverso, los algoritmos que implementan actualizaciones activadas están diseñados para reducir el tiempo de convergencia de la red. Con las actualizaciones activadas, cada vez que un router cambia el costo de una ruta, envía inmediatamente la tabla de vectores de distancia modificada a los dispositivos vecinos. Este mecanismo garantiza que las notificaciones de cambio de topología se propaguen rápidamente, en lugar de en el intervalo periódico normal.

Las actualizaciones desencadenadas son compatibles con RFC 1058.

#### 5.3.6 Limitaciones de RIP

Hay una serie de limitaciones observadas en los entornos RIP:

Límites de coste de trayectoria: La resolución del problema de conteo hasta el infinito impone un Coste máximo de una ruta de acceso de red. Esto establece un límite superior en el diámetro máximo de la red. Las redes que requieren trayectorias superiores a 15 saltos deben utilizar un protocolo de enrutamiento alternativo.

Actualizaciones de tablas con uso intensivo de la red: La difusión periódica de la tabla de vectores de distancia puede dar lugar a una mayor utilización de los recursos de red. Esto puede ser una preocupación en segmentos de capacidad reducida.

Convergencia relativamente lenta: RIP, al igual que otros protocolos de vectores de distancia, es relativamente lento para converger. Los algoritmos se basan en temporizadores para iniciar anuncios de tabla de enrutamiento.

No se admite el enmascaramiento de subred de longitud variable: los anuncios de ruta en un entorno RIP no incluyen información de enmascaramiento de subred. Esto hace que sea imposible que las redes RIP implementen máscaras de subred de longitud variable.

### 5.4 Protocolo de información de enrutamiento versión 2 (RIP-2)

El IETF reconoce dos versiones de RIP:

RIP versión 1 (RIP-1): Este protocolo se describe en RFC 1058.

RIP versión 2 (RIP-2): RIP-2 también es un protocolo de vector de distancia diseñado para su uso dentro de un AS. Fue desarrollado para abordar las limitaciones observadas en RIP-1. RIP-2 se describe en RFC 2453. La norma (STD 56) se publicó a finales de 1994.

En la práctica, el término RIP se refiere a RIP-1. Siempre que encuentre el término RIP en la literatura TCP/IP, es seguro asumir que la referencia es a la versión 1 de RIP a menos que se indique lo contrario. Esta misma convención se utiliza en este documento. Sin embargo, cuando se comparan las dos versiones, se utiliza el término RIP-1 para evitar confusiones.

RIP-2 es similar a RIP-1. Fue desarrollado para extender la funcionalidad RIP-1 en redes pequeñas. RIP-2 proporciona estos beneficios adicionales que no están disponibles en RIP-1:

Compatibilidad con CIDR y VLSM: RIP-2 admite la superred (es decir, CIDR) y el enmascaramiento de subredes de longitud variable. Este apoyo fue la razón principal por la que se desarrolló el nuevo estándar. Esta mejora posiciona el estándar en acomodar un grado de complejidad de direccionamiento no admitido en RIP-1.

Compatibilidad con la multidifusión: RIP-2 admite el uso de la multidifusión en lugar de la simple difusión de anuncios de enrutamiento. Esto reduce la carga de procesamiento en los hosts que no escuchan los mensajes RIP-2. Para garantizar la interoperabilidad con entornos RIP-1, esta opción se configura en cada interfaz de red.

Soporte para autenticación: RIP-2 admite la autenticación de cualquier nodo que transmita anuncios de ruta. Esto evita que las fuentes fraudulentas dañen la tabla de enrutamiento.

Compatibilidad con RIP-1: RIP-2 es totalmente interoperable con RIP-1. Esto proporciona compatibilidad con versiones anteriores entre los dos estándares.

Como se señaló en la sección RIP-1, una deficiencia notable en el estándar RIP-1 es la implementación del campo métrico. RIP-1 especifica la métrica como un valor entre 0 y 16. Para garantizar la compatibilidad con las redes RIP-1, RIP-2 conserva esta definición. En ambos estándares, las rutas de red con un número de saltos superior a 15 se interpretan como inalcanzables.

#### 5.4.1 Formato de paquete RIP-2

La especificación original de RIP-1 fue diseñada para soportar futuras mejoras. El estándar RIP-2 fue capaz de capitalizar esta característica. Los desarrolladores de RIP-2 notaron que un paquete RIP-1 ya contiene un campo de versión y que el 50% de los octetos no se utilizan.

La Figura 5-9 ilustra el contenido de un paquete RIP-2. El paquete se muestra con información de autenticación. La primera entrada de la actualización contiene una entrada de enrutamiento o una entrada de autenticación. Si la primera entrada es una entrada de autenticación, se pueden incluir 24 entradas de enrutamiento adicionales en el mensaje. Si no hay información de autenticación, se pueden proporcionar 25 entradas de enrutamiento.

*Figura 5-9 Formato de paquete RIP-2*

Mandar

Versión

Reservado

AFI: X'FFFF'

Tipo de autenticación

Datos de autenticación

AFI:2

Etiqueta de ruta

Dirección IP

Máscara de subred

Próximo salto

Métrico

Número de octetos

Solicitud=1

Respuesta=2

0=

Sin autenticación

2=

Datos de contraseña

Contraseña si se selecciona el tipo 2

Entrada de enrutamiento: Es posible que no sea

repetido

}

}

1

1

2

2

2

16

2

2

4

4

4

4

}

}

Autenticación

Entrada

El uso del campo de comando, el campo de dirección IP y el campo de métrica en un mensaje RIP-2 es idéntico al uso en un mensaje RIP-1. De lo contrario, los cambios implementados en un paquete RIP-2 incluyen:

**Versión** El valor contenido en este campo debe ser dos. Esto indica a los routers RIP-1 que ignoren cualquier información contenida en los campos no utilizados anteriormente.

**AFI (familia de direcciones)** Un valor de x'0002' indica que la dirección contenida en el campo de dirección de red es una dirección IP. Un valor de x'FFFF' indica una entrada de autenticación.

**Tipo de autenticación** : este campo define los 16 bytes restantes de la

entrada de autenticación. Un valor de 0 indica *que no hay* autenticación. Un valor de dos indica que el campo de datos de autenticación contiene datos de contraseña.

**Datos de autenticación** Este campo contiene una contraseña de 16 bytes.

**Etiqueta de ruta** Este campo está destinado a diferenciar entre rutas internas y externas. Las rutas internas se aprenden a través de RIP-2 dentro de la misma red o AS.

**Máscara de subred** Este campo contiene la máscara de subred de la red a la que se hace referencia.

**Próximo salto** Este campo contiene una recomendación sobre el salto siguiente que el router debe utilizar al enviar datagramas a la red a la que se hace referencia.

#### 5.4.2 Limitaciones de RIP-2

RIP-2 se desarrolló para abordar muchas de las limitaciones observadas en RIP-1. Sin embargo, los límites de coste del trayecto y la lenta convergencia inherentes a las redes RIP-1 también son motivo de preocupación en los entornos RIP-2.

Además de estas preocupaciones, existen limitaciones en el proceso de autenticación RIP-2. El estándar RIP-2 no cifra la contraseña de autenticación. Se transmite en texto claro. Esto hace que la red sea vulnerable a los ataques de cualquier persona con acceso físico directo al entorno.

### 5.5 RIPng para IPv6

RIPng se desarrolló para permitir que los enrutadores dentro de una red basada en IPv6 intercambien información utilizada para calcular rutas. Está documentado en RFC 2080. Proporcionamos información adicional sobre IPv6 en 9.1, "Introducción a IPv6" en la página 328.

Al igual que los otros protocolos de la familia RIP, RIPng es un protocolo de vector de distancia diseñado para su uso dentro de un pequeño sistema autónomo. RIPng utiliza los mismos algoritmos, temporizadores y lógica que se utilizan en RIP-2.

RIPng tiene muchas de las mismas limitaciones inherentes a otros vectores de distancia

protocolos. Las restricciones de coste del trayecto y el tiempo de convergencia siguen siendo motivo de preocupación

RIPng redes.

#### 5.5.1 Diferencias entre RIPng y RIP-2

Hay dos distinciones importantes entre RIP-2 y RIPng:

Compatibilidad con la autenticación: El estándar RIP-2 incluye compatibilidad con Autenticación de un nodo que transmite información de enrutamiento. RIPng no incluye ningún soporte de autenticación nativa. Más bien, RIPng utiliza las características de seguridad inherentes a IPv6. Además de la autenticación, estas características de seguridad proporcionan la capacidad de cifrar cada paquete RIPng. Esto puede controlar el conjunto de dispositivos que reciben la información de enrutamiento. Una consecuencia del uso de las características de seguridad de IPv6 es que se elimina el campo AFI dentro del paquete RIPng. Ya no es necesario distinguir entre las entradas de autenticación y las entradas de enrutamiento dentro de un anuncio.

Compatibilidad con formatos de direccionamiento IPv6: Los campos contenidos en los paquetes RIPng se actualizaron para admitir el formato de dirección IPv6 más largo.

#### 5.5.2 Formato de paquete RIPng

Los paquetes RIPng se transmiten mediante datagramas UDP. RIPng envía y recibe datagramas utilizando el puerto UDP número 521.

El formato de un paquete RIPng es similar al formato RIP-2. Específicamente, ambos paquetes contienen un encabezado de comando de 4 octetos seguido de un conjunto de entradas de ruta de 20 octetos. El formato del paquete RIPng se muestra en la Figura 5-10.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Número de octetos  Solicitud=1   |  |  | | --- | --- | | Mandar |  | | Versión | | Reservado | | Entrada de tabla de rutas (RTE) | |  |   1 {Respuesta=2  1  2  20 {Se puede repetir |

*Figura 5-10 Formato de paquete RIPng*

El uso del campo de comando y el campo de versión es idéntico al uso en un paquete RIP-2. Sin embargo, los campos que contienen información de enrutamiento se han actualizado para dar cabida a la dirección IPv6 de 16 octetos. Estos campos se utilizan de forma diferente a los campos correspondientes en un paquete RIP-1 o RIP-2. El formato del RTE se muestra en la Figura 5-11.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Número de octetos   |  |  | | --- | --- | | Prefijo IPv6 |  | | Etiqueta de ruta | | Longitud del prefijo | | Métrico | |  |   16  2  1  1 |

*Figura 5-11 Entrada de tabla de rutas (RTE)*

En RIPng, la combinación del prefijo IP y la longitud del prefijo identifica la ruta que se va a anunciar. La métrica permanece codificada en un campo de 1 octeto. Esta longitud es suficiente porque RIPng utiliza un número máximo de saltos de 16.

Otra diferencia entre RIPng y RIP-2 es el proceso utilizado para determinar el siguiente salto. En RIP-2, cada entrada de la tabla de rutas contiene un campo de salto siguiente. En RIPng, incluir esta información en cada RTE habría duplicado el tamaño del anuncio. Por lo tanto, en RIPng, el siguiente salto se incluye en un tipo especial de RTE. El salto siguiente especificado se aplica a cada entrada posterior de la tabla de enrutamiento en el anuncio. El formato de un RTE utilizado para especificar el salto siguiente se muestra en la Figura 5-12.

16

2

1

1

Dirección IPv6 Next Hop

Reservado

Reservado

Métrica 0x'FF'

Número de octetos

{

Se utiliza para distinguir un

Entrada de salto siguiente

*Figura 5-12 Entrada de tabla de rutas de salto siguiente (RTE)*

El RTE del salto siguiente se identifica mediante un valor de 0x'FF' en el campo métrico. Este valor reservado está fuera del rango válido de métricas.

El uso de RTE y RTE de salto siguiente se muestra en la Figura 5-13.

*Figura 5-13 Uso del RIPng RTE*

4

20

20

20

20

20

20

20

20

Número de octetos

Mandar

Entrada de enrutamiento #1

Entrada de enrutamiento #2

Entrada de enrutamiento #3

Próximo salto RTE A

Entrada de enrutamiento #4

Entrada de enrutamiento #5

Próximo salto RTE B

Entrada de enrutamiento #6

En este ejemplo, las tres primeras entradas de enrutamiento no tienen un RTE de salto siguiente correspondiente. Los prefijos de dirección especificados por estas entradas se enrutarán a través del enrutador de publicidad. Los prefijos incluidos en las entradas de enrutamiento 4 y 5 se enrutarán a través de la dirección de salto siguiente especificada en el RTE A de salto siguiente. El prefijo incluido en la entrada de enrutamiento 6 se enrutará a través de la dirección de salto siguiente especificada en el RTE B del salto siguiente.

### 5.6 Abrir primero el camino más corto (OSPF )

El protocolo Open Shortest Path First (OSPF) es otro ejemplo de protocolo de puerta de enlace interior. Fue desarrollado como una alternativa de enrutamiento no propietaria a abordar las limitaciones de RIP. El desarrollo inicial comenzó en 1988 y finalizó en 1991. Se siguen publicando actualizaciones posteriores del protocolo. La versión actual del estándar está documentada en RFC 2328.

OSPF proporciona una serie de características que no se encuentran en los protocolos de vectores de distancia. La compatibilidad con estas características ha convertido a OSPF en un protocolo de enrutamiento ampliamente implementado en grandes entornos de red. De hecho, RFC 1812 - Requisitos para enrutadores IPv4, enumera OSPF como el único protocolo de enrutamiento dinámico requerido. Las siguientes características contribuyen a la aceptación continua del estándar OSPF:

Equilibrio de carga de igual costo: el uso simultáneo de múltiples rutas puede proporcionar una utilización más eficiente de los recursos de red.

Partición lógica de la red: Esto reduce la propagación de la información de interrupciones durante condiciones adversas. También proporciona la capacidad de agregar anuncios de enrutamiento que limitan el anuncio de información de subred innecesaria.

Compatibilidad con la autenticación: OSPF admite la autenticación de cualquier nodo que transmita anuncios de ruta. Esto evita que las fuentes fraudulentas dañen las tablas de enrutamiento.

Tiempo de convergencia más rápido: OSPF proporciona una propagación instantánea de los cambios de enrutamiento. Esto acelera el tiempo de convergencia necesario para actualizar las topologías de red.

Compatibilidad con CIDR y VLSM: Esto permite al administrador de red asignar de manera eficiente los recursos de direcciones IP.

OSPF es un protocolo de estado de enlace. Al igual que con otros protocolos de estado de enlace, cada router OSPF ejecuta el algoritmo SPF ("Algoritmo SPF (Shortest-Path First" en la página 177) para procesar la información almacenada en la base de datos de estado de enlace. El algoritmo produce un árbol de ruta más corta que detalla las rutas preferidas a cada red de destino.

#### 5.6.1 Terminología OSPF

OSPF utiliza terminología específica para describir el funcionamiento del protocolo.

##### Áreas OSPF

Las redes OSPF se dividen en una colección de *áreas*. Un área consiste en una agrupación lógica de redes y enrutadores. El área puede coincidir con límites geográficos o administrativos. A cada área se le asigna un identificador de área de 32 bits.

La subdivisión de la red proporciona las siguientes ventajas:

Dentro de un área, cada router mantiene una base de datos de topología idéntica que describe los dispositivos de enrutamiento y los enlaces dentro del área. Estos routers no tienen conocimiento de topologías fuera del área. Solo conocen las rutas a estos destinos externos. Esto reduce el tamaño de la base de datos de topología mantenida por cada enrutador.

Las áreas limitan el crecimiento potencialmente explosivo en el número de actualizaciones de estado de vínculo. La mayoría de los LSA se distribuyen solo dentro de un área.

Las áreas reducen el procesamiento de CPU necesario para mantener la base de datos de topología. El algoritmo SPF se limita a gestionar los cambios dentro del área.

###### Área de la columna vertebral y área 0

Todas las redes OSPF contienen al menos un área. Esta área se conoce como área 0 o área troncal. Se pueden crear áreas adicionales en función de la topología de red u otros requisitos de diseño.

En las redes que contienen varias áreas, la red troncal se conecta físicamente a todas las demás áreas. OSPF espera que todas las áreas anuncien la información de enrutamiento directamente en la red troncal. A continuación, la red troncal anuncia esta información en otras áreas.

La figura 5-14 de la página 198 muestra una red con un área troncal y cuatro áreas adicionales.

##### Enrutadores dentro del área, borde de área y límite de AS

Hay tres clasificaciones de routers en una red OSPF. La Figura 5-14 ilustra la interacción de estos dispositivos.

Área 4

Área 2

Área 1

ASBR

ABR

ABR

ABR

Área 3

ABR

ASBR

IA

IA

AS Enlaces externos

Área 0

COMO 10

AS Enlaces externos

Llave

ASBR - Enrutador de borde AS

ABR - Enrutador de borde de área

IA - Router intra-área

*Figura 5-14 Tipos de routers OSPF*

Dónde:

|  |  |
| --- | --- |
| **Enrutadores dentro del área** | Esta clase de router se encuentra lógicamente completamente dentro de un área OSPF. Los enrutadores dentro del área mantienen una base de datos de topología para su área local. |
| **Enrutadores de borde de área (ABR)** | Esta clase de router está conectada lógicamente a dos o más áreas. Un área debe ser el área de la columna vertebral. Un ABR se utiliza para interconectar áreas. Mantienen una base de datos de topología independiente para cada área asociada. Los ABR también ejecutan  instancias separadas del algoritmo SPF para cada área. |

**Routers de límite AS (ASBR)** Esta clase de router se encuentra en la periferia de una interred OSPF. Funciona como una puerta de enlace que intercambia accesibilidad entre la red OSPF y otros entornos de enrutamiento.

Los ASBR son responsables de anunciar los anuncios de enlaces externos de AS a través del AS. Proporcionamos más información sobre los anuncios de enlaces externos en 5.6.4, "Redistribución de rutas OSPF" en la página 208.

A cada enrutador se le asigna un *ID de enrutador* (RID) de 32 bits. El RID identifica de forma única el dispositivo. Una implementación popular asigna el RID desde la dirección IP con el número más bajo configurado en el enrutador.

##### Tipos de redes físicas

OSPF clasifica los segmentos de red en tres tipos. La frecuencia y los tipos de comunicación que se producen entre los dispositivos OSPF conectados a estas redes se ven afectados por el tipo de red:

Punto a punto: Las redes punto a punto enlazan directamente dos routers.

Acceso múltiple: Las redes de acceso múltiple admiten la conexión de más de dos enrutadores.

Se subdividen a su vez en dos tipos:

* Las redes de difusión tienen la capacidad de dirigir simultáneamente un paquete a todos los routers conectados. Esta capacidad utiliza una dirección reconocida por todos los dispositivos. Las LAN Ethernet y token-ring son ejemplos de redes de acceso múltiple de difusión OSPF.
* Las redes que no son de radiodifusión no tienen capacidad de radiodifusión. Cada paquete debe estar dirigido específicamente a cada router de la red. Las redes X.25 y de retransmisión de tramas son ejemplos de redes OSPF de acceso múltiple sin difusión.

Punto a multipunto: Las redes punto a multipunto son un caso especial de redes de acceso múltiple que no son de difusión. En una red punto a multipunto, no es necesario que un dispositivo tenga una conexión directa con todos los demás dispositivos. Esto se conoce como un entorno parcialmente mallado.

##### Routers vecinos y adyacencias

Los enrutadores que comparten un segmento de red común establecen una relación de vecino

en el segmento. Los routers deben acordar la siguiente información para convertirse en vecinos:

ID de área: Los routers deben pertenecer a la misma área OSPF.

Autenticación: Si se define la autenticación, los routers deben especificar la misma contraseña.

Intervalos de saludo e inactividad: Los routers deben especificar los mismos intervalos de temporizador utilizados en el protocolo de saludo. Describimos este protocolo con más detalle en "Tipos de paquetes OSPF" en la página 203.

Indicador de área de código auxiliar: Los routers deben aceptar que el área esté configurada como un área de código auxiliar. Describimos las áreas de stub con más detalle en 5.6.5, "Áreas stub OSPF" en la página 210.

Después de que dos routers se hayan convertido en vecinos, se puede formar una relación de adyacencia entre los dispositivos. Los enrutadores vecinos se consideran adyacentes cuando han sincronizado sus bases de datos de topología. Esto ocurre a través del intercambio de información sobre el estado del enlace.

##### Enrutador designado y designado de respaldo

El intercambio de información sobre el estado del vínculo entre vecinos puede crear cantidades significativas de tráfico de red. Para reducir el ancho de banda total necesario para sincronizar bases de datos y anunciar información sobre el estado del vínculo, un router no necesariamente desarrolla adyacencias con todos los dispositivos vecinos:

Redes de acceso múltiple: Las adyacencias se forman entre un router individual y el router designado (de respaldo).

Redes punto a punto: Se forma una adyacencia entre ambos dispositivos.

Cada red de acceso múltiple elige un enrutador designado (DR) y un enrutador designado de respaldo (BDR). El DR realiza dos funciones clave en el segmento de red:

Forma adyacencias con todos los routers de la red de acceso múltiple. Esto hace que el DR se convierta en el punto focal para el reenvío de LSA.

Genera anuncios de enlace de red que enumeran cada enrutador conectado a la red de acceso múltiple. Para obtener información adicional sobre los anuncios de enlaces de red, consulte "Anuncios de estado de enlaces e inundaciones" en la página 201.

El BDR forma las mismas adyacencias que el router designado. Asume la funcionalidad de recuperación ante desastres cuando se produce un error en la recuperación ante desastres.

A cada router se le asigna una prioridad de 8 bits, lo que indica su capacidad para seleccionarse como DR o BDR. Una prioridad de router de cero indica que el router no es elegible para ser seleccionado. La prioridad se configura en cada interfaz del router.

La figura 5-15 ilustra la relación entre vecinos. No se forman adyacencias entre los routers que no están seleccionados para ser el DR o BDR.

DR

Otro

Otro

BDR

Adyacente

Vecinos

Vecinos

*Figura 5-15 Relación entre adyacencias y vecinos*

##### Base de datos de estado de vínculos

La base de datos de estado de vínculo también se denomina *base de datos de topología*. Contiene el conjunto de anuncios de estado de enlace que describen la red OSPF y las conexiones externas. Cada router dentro del área mantiene una copia idéntica de la base de datos de estado del link.

**Nota:** RFC 2328 utiliza el término base de datos de estado de vínculo con preferencia a base de datos de topología. El primer término tiene la ventaja de que describe el contenido de la base de datos. Este último término es más descriptivo de la finalidad de la base de datos. Este libro ha utilizado anteriormente el término base de datos de topología por esta razón. Sin embargo, para el resto de la sección OSPF, nos referimos a ella como la base de datos de estado de vínculo.

##### Vincular anuncios de estado e inundaciones

El contenido de un LSA describe un componente de red individual (es decir, enrutador, segmento o destino externo). Los LSA se intercambian entre enrutadores OSPF adyacentes. Esto se hace para sincronizar la base de datos de estado de vínculo en cada dispositivo.

Cuando un router genera o modifica un LSA, debe comunicar este cambio a través de la red. El router inicia este proceso reenviando el LSA a cada dispositivo adyacente. Al recibir el LSA, estos vecinos almacenan la información en su base de datos de estado de vínculo y comunican el LSA a sus vecinos. Esta actividad de almacenamiento y reenvío continúa hasta que todos los dispositivos reciben la actualización. Este proceso se denomina *inundación confiable*. Se toman dos pasos para garantizar que esta inundación transmita los cambios de manera efectiva sin sobrecargar la red con cantidades excesivas de tráfico LSA:

Cada router almacena el LSA durante un período de tiempo antes de propagar la información a sus vecinos. Si, durante ese tiempo, llega una nueva copia del LSA, el router reemplaza la versión almacenada. Sin embargo, si la nueva copia está desactualizada, se descarta.

Para garantizar la confiabilidad, se debe reconocer cada anuncio de estado de vínculo. Se pueden agrupar varias confirmaciones en un solo paquete de confirmación. Si no se recibe un acuse de recibo, se retransmite el paquete de actualización del estado del link original.

Los anuncios de estado de vínculo contienen cinco tipos de información. Juntos, estos anuncios proporcionan la información necesaria para describir toda la red OSPF y cualquier entorno externo:

LSA del router: Este tipo de anuncio describe el estado de las interfaces (links) del router dentro del área. Son generados por cada router OSPF. Los anuncios están inundados por toda la zona.

LSA de red: este tipo de anuncio enumera los routers conectados a una red de acceso múltiple. Son generados por el DR en un segmento de acceso múltiple. Los anuncios están inundados por toda la zona.

LSAs de resumen (Tipo-3 y Tipo-4): Este tipo de anuncio es generado por un ABR. Hay dos tipos de anuncios de enlaces de resumen:

* Los LSA de resumen de tipo 3 describen rutas a destinos en otras áreas dentro de la red OSPF (destinos entre áreas).
* Los LSA de resumen de tipo 4 describen las rutas a los ASBR. Los LSA de resumen se utilizan para intercambiar información de accesibilidad entre áreas. Normalmente, la información se anuncia en el área de la red troncal. A continuación, la red troncal inyecta esta información en otras áreas.

LSA externos de AS: Este tipo de anuncio describe rutas a destinos externos a la red OSPF. Son generados por un ASBR. Los anuncios se inundan en todas las áreas de la red OSPF.

La figura 5-16 ilustra los diferentes tipos de anuncios de estado de vínculo.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Enlaces de enrutador | Enlaces de red | | Enrutador  - Anunciado por el router - Describe el estado/costo de los enlaces del router | DR  - Anunciado por el router designado: describe todos los routers conectados a la red | | Enlaces de resumen  ABR  Área X  Área 0   * Anunciado por router * Describe el estado/costo de | Enlaces externos  ASBR  Área 0  Área X   * Anunciado por router * Describe el estado/costo de |   Enlaces del router Enlaces del router |

*Figura 5-16 Anuncios de estado de link OSPF*

##### Tipos de paquetes OSPF

Los paquetes OSPF se transmiten en datagramas IP. No están encapsulados dentro de paquetes TCP o UDP. El encabezado IP utiliza el identificador de protocolo 89. Los paquetes OSPF se envían con un ToS de IP de 0 y una precedencia de IP de control entre redes. Esto se utiliza para obtener un procesamiento preferencial para los paquetes. Discutimos los términos de servicio y la precedencia IP más adelante en "Servicios integrados" en la página 288.

Siempre que sea posible, OSPF utiliza instalaciones de multidifusión para comunicarse con los dispositivos vecinos. En entornos de difusión y punto a punto, los paquetes se envían a la dirección de multidifusión reservada 224.0.0.5. RFC 2328 se refiere a esto como la dirección AllSPFRouters. En entornos que no son de difusión, los paquetes se dirigen a la dirección IP específica del vecino.

Todos los paquetes OSPF comparten el encabezado común que se muestra en la Figura 5-17. El encabezado proporciona información general, incluido el identificador de área, el RID, la suma de comprobación y la información de autenticación.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Número de octetos  Versión = 2  1  1= Hola   |  |  | | --- | --- | | Versión |  | | Tipo de paquete | | Longitud del paquete | | Router ID | | Identificación de área | | Checksum | | Tipo de autenticación | | Datos de autenticación | |  |   1 {2=Descripción de la base de datos  3=Solicitud de estado de enlace  4=Actualización del estado del enlace  25=Acuse de recibo del estado del enlace  4  4  2  20=Sin autenticación  {1=Contraseña simple  8Password si se selecciona el tipo 1 |

*Figura 5-17 Encabezado común OSPF*

El campo type identifica el paquete OSPF como uno de los cinco tipos posibles:

|  |  |
| --- | --- |
| **Hola** | Este tipo de paquete detecta y mantiene relaciones de vecindad. |
| **Descripción de la base de datos** | Este tipo de paquete describe el conjunto de LSA contenidos en la base de datos de estado de link del router. |
| **Solicitud de estado de enlace** | Este tipo de paquete solicita una instancia más actual de un LSA a un vecino. |
| **Actualización del estado del vínculo** | Este tipo de paquete proporciona una instancia más actual de un LSA a un vecino. |

###### Acuse de recibo del estado del vínculo

Este tipo de paquete acusa recibo de un LSA recién recibido.

Describimos el uso de estos paquetes en la siguiente sección.

#### 5.6.2 Comunicación con vecinos

OSPF es responsable de determinar el conjunto óptimo de rutas a través de una red. Para lograr esto, cada router intercambia LSA con otros routers de la red. El protocolo OSPF define una serie de actividades para lograr este intercambio de información:

Descubriendo vecinos

Elección de un router designado

Establecimiento de adyacencias y sincronización de bases de datos

Los cinco tipos de paquetes OSPF se utilizan para soportar estos intercambios de información.

##### Descubriendo vecinos: el protocolo OSPF Hello

El protocolo Hello detecta y mantiene relaciones con routers vecinos. Los paquetes de saludo se envían periódicamente a cada interfaz de enrutador. El paquete contiene el RID de otros routers cuyos paquetes de saludo ya se han recibido a través de la interfaz.

Cuando un dispositivo ve su propio RID en el paquete de saludo generado por otro enrutador, estos dispositivos establecen una relación de vecino.

El paquete hello también contiene la prioridad del router, el identificador DR y el identificador BDR. Estos parámetros se utilizan para elegir el DR en redes de acceso múltiple.

##### Elección de un router designado

Todas las redes de acceso múltiple deben tener un DR. También se puede seleccionar un BDR. La copia de seguridad garantiza que no haya una pérdida prolongada de la capacidad de enrutamiento si se produce un error en la recuperación ante desastres.

El DR y el BDR se seleccionan utilizando la información contenida en los paquetes de saludo. El dispositivo con la prioridad de enrutador OSPF más alta en un segmento se convierte en el DR para ese segmento. El mismo proceso se repite para seleccionar el BDR. En caso de empate, se selecciona el router con el RID más alto. Un router declarado como DR no es elegible para convertirse en BDR.

Una vez elegidos, el DR y el BDR proceden a establecer adyacencias con todos los routers en el segmento de acceso múltiple.

##### Establecimiento de adyacencias y sincronización de bases de datos

Los routers vecinos se consideran adyacentes cuando han sincronizado sus bases de datos de estado de vínculo. Un router no desarrolla una adyacencia con todos los dispositivos vecinos. En las redes de acceso múltiple, las adyacencias se forman solo con el DR y el BDR. Este es un proceso de dos pasos.

###### Paso 1: Proceso de intercambio de bases de datos

La primera fase de la sincronización de la base de datos es el proceso de intercambio de la base de datos.

Esto ocurre inmediatamente después de que dos vecinos intentan establecer una adyacencia. El proceso consiste en un intercambio de paquetes de descripción de la base de datos. Los paquetes contienen una lista de los LSA almacenados en la base de datos local.

Durante el proceso de intercambio de bases de datos, los routers forman una relación maestro/subordinado. El maestro es el primero en transmitir. Cada paquete se identifica mediante un número de secuencia. Con este número de secuencia, el subordinado reconoce cada paquete de descripción de base de datos del maestro. El subordinado también incluye su propio conjunto de encabezados de estado de vínculo en las confirmaciones.

###### Paso 2: Carga de la base de datos

Durante el proceso de intercambio de la base de datos, cada router anota los encabezados de estado de link para los que el vecino tiene una instancia más actual (todos los anuncios tienen una marca de tiempo). Una vez completado el proceso, cada router solicita la información más actual del vecino. Esta solicitud se realiza con un paquete de solicitud de estado de vínculo.

Cuando un router recibe una solicitud de estado de enlace, debe responder con un conjunto de paquetes de actualización de estado de enlace que proporcionan el LSA solicitado. Cada LSA transmitido es reconocido por el receptor. Este proceso es similar al procedimiento de inundación confiable utilizado para transmitir cambios de topología a través de la red.

Cada LSA contiene un campo de antigüedad que indica el tiempo en segundos desde el origen del anuncio. La antigüedad sigue aumentando después de instalar el LSA en la base de datos de topología. También aumenta durante cada salto del proceso de inundación. Cuando se alcanza la antigüedad máxima, el LSA ya no se utiliza para determinar la información de enrutamiento y se descarta de la base de datos de estado del vínculo. Esta edad también se utiliza para distinguir entre dos copias idénticas de un anuncio.

#### 5.6.3 Máquina de estado vecina OSPF

La especificación OSPF define un conjunto de estados vecinos y los eventos que pueden hacer que un vecino pase de un estado a otro. Se utiliza una máquina de estados para describir estas transiciones:

Abajo: Este es el estado inicial. Indica que no se ha recibido información reciente de ningún dispositivo del segmento.

Intento: este estado se utiliza en redes que no son de difusión. Indica que un vecino parece estar inactivo. Continúan los intentos de restablecer el contacto.

Init: se ha iniciado la comunicación con el vecino, pero no se ha establecido la comunicación bidireccional. Específicamente, se recibió un paquete de saludo del vecino, pero el router local no apareció en el paquete de saludo del vecino.

2 vías: Se ha establecido una comunicación bidireccional entre los dos routers. Se pueden formar adyacencias. Los vecinos son elegibles para ser elegidos como enrutadores designados.

ExStart: Los vecinos están empezando a formar una adyacencia.

Intercambio: los dos vecinos están intercambiando sus bases de datos de topología.

Cargando: los dos vecinos están sincronizando sus bases de datos de topología.

Completo: los dos vecinos son totalmente adyacentes y sus bases de datos están sincronizadas.

Los eventos de red hacen que cambie el estado OSPF de un vecino. Por ejemplo, cuando un router recibe un paquete de saludo de un dispositivo vecino, el estado del vecino OSPF cambia de Down a Init. Cuando se ha establecido la comunicación bidireccional, el estado vecino cambia de Init a 2-Way. RFC 2328 contiene una descripción completa de los eventos que causan un cambio de estado.

##### Enlaces virtuales OSPF y zonas de tránsito

Los vínculos virtuales se utilizan cuando una red no admite la topología de red OSPF estándar. Esta topología define un área troncal que se conecta directamente a cada área OSPF adicional. El vínculo virtual aborda dos condiciones:

Puede conectar lógicamente el área de la red troncal cuando no es contigua.

Puede conectar un área a la red troncal cuando no existe una conexión directa.

Se establece un enlace virtual entre dos ABR que comparten un área común que no es la red troncal. El vínculo se trata como un vínculo punto a punto. La zona común se conoce como *zona de tránsito*. La Figura 5-18 de la página 208 ilustra la interacción entre los enlaces virtuales y las áreas de tránsito cuando se utiliza para conectar un área a la red troncal.

*Figura 5-18 Áreas de tránsito y enlace virtual OSPF*

Área 0

Área 1

Área 2

Área de tránsito

ABR

ABR

Enlace virtual

Este diagrama muestra que el área 1 no tiene una conexión directa con la red troncal. El área 2 se puede utilizar como área de tránsito para proporcionar esta conexión. Se establece un enlace virtual entre las dos ABR ubicadas en el área 2. El establecimiento de este enlace virtual amplía lógicamente el área troncal para conectarse con el área 1.

Un vínculo virtual solo se utiliza para transmitir información de enrutamiento. No transporta tráfico regular entre el área remota y la red troncal. Este tráfico, además del tráfico de vínculo virtual, se enruta mediante el enrutamiento estándar dentro del área dentro del área de tránsito.

#### 5.6.4 Redistribución de rutas OSPF

La redistribución de rutas es el proceso de introducir rutas externas en una red OSPF. Estas rutas pueden ser rutas estáticas o rutas aprendidas a través de otro protocolo de enrutamiento. Son anunciados en la red OSPF por un ASBR. Estas rutas se convierten en rutas externas OSPF. El ASBR anuncia estas rutas inundando los LSA externos de OSPF AS en toda la red OSPF.

Las rutas describen una ruta de extremo a extremo que consta de dos partes:

Porción externa: Esta es la porción de la trayectoria externa a la red OSPF. Cuando estas rutas se distribuyen en OSPF, el ASBR asigna un costo inicial. Este coste representa el *coste externo* asociado a la travesía de la parte externa de la ruta.

Parte interna: Esta es la parte de la trayectoria interna a la red OSPF. Los costos de esta parte de la red se calculan utilizando algoritmos OSPF estándar.

OSPF diferencia entre dos tipos de rutas externas. Se diferencian en la forma en que se calcula el costo de la ruta. El ASBR está configurado para redistribuir la ruta como:

Tipo externo 1: El costo total de la ruta es la suma del costo externo y los costos internos de OSPF.

Tipo externo 2: El coste total de la ruta es siempre el coste externo. Esto ignora los costos internos de OSPF necesarios para alcanzar el ASBR.

La Figura 5-19 ilustra un ejemplo de los tipos de rutas externas OSPF.

*Figura 5-19 Redistribución de ruta OSPF*

**Tabla de enrutamiento R1**

10.99.5.0/24

E1: Coste 60

o

E2: Coste 50

**Tabla de ruteo R2**

10.99.5.0/24

E1: Coste 65

o

E2: Coste 50

R1

R2

(20)

(10)

(15)

OSPF

Red

ASBR

10.99.5.0/24

Redistribuido

con coste externo 50

Externo

Interno

RASGADURA

Red

10.99.5.0/24

En este ejemplo, el ASBR está redistribuyendo la ruta 10.99.5.0/24 en la red OSPF. Esta subred se encuentra dentro de la red RIP. La ruta se anuncia en OSPF con un costo externo de 50. Esto representa el costo de la parte de la ruta que atraviesa la red RIP:

Si el ASBR redistribuyó la ruta como una ruta E1, R1 contendrá una ruta externa a esta subred con un costo de 60 (50 + 10). R2 tendrá una ruta externa con un coste de 65 (50 + 15).

Si el ASBR redistribuyó la ruta como una ruta E2, tanto R1 como R2 contendrán una ruta externa a esta subred con un costo de 50. Los costos asociados con la travesía de segmentos dentro de la red OSPF no se incluyen en el costo total para llegar al destino.

#### 5.6.5 Zonas auxiliares OSPF

OSPF permite que ciertas áreas se definan como un área auxiliar. Un área de stub se crea cuando el ABR que se conecta a un área de stub excluye los LSA externos de AS para que no se inunden en el área. Esto se hace para reducir el tamaño de la base de datos de estado de vínculo mantenida dentro de los enrutadores de área auxiliar. Debido a que no hay rutas específicas a redes externas, el enrutamiento a estos destinos se basa en una ruta predeterminada generada por el ABR. Las bases de datos de estado de link mantenidas dentro del área stub contienen solo la ruta predeterminada y las rutas desde dentro del entorno OSPF (por ejemplo, rutas dentro del área y entre áreas).

Dado que un área de código auxiliar no permite LSA externos, un área de código auxiliar no puede contener un ASBR. No se pueden generar rutas externas desde dentro del área de código auxiliar.

Las áreas de código auxiliar se pueden implementar cuando hay un único punto de salida que conecta el área con la red troncal. Un área con varios puntos de salida también puede ser un área de código auxiliar. Sin embargo, no hay garantía de que los paquetes que salen del área sigan una ruta óptima. Esto se debe al hecho de que cada ABR genera una ruta predeterminada. No hay capacidad para asociar el tráfico con rutas predeterminadas específicas.

Todos los routers dentro del área deben configurarse como routers stub. Esta configuración se verifica a través del intercambio de paquetes de saludo.

##### Zonas no tan rechonchos

Una extensión del concepto de área stub es el *área no tan stubby* (NSSA). Esta alternativa está documentada en RFC 3101. Una NSSA es similar a un área stub en el sentido de que el ABR que da servicio a la NSSA no inunda ninguna ruta externa en la NSSA. Las únicas rutas inundadas en la NSSA son la ruta predeterminada y cualquier otra ruta desde el entorno OSPF (por ejemplo, dentro del área y entre áreas).

Sin embargo, a diferencia de un área de código auxiliar, un ASBR se puede ubicar dentro de un NSSA. Éste

ASBR puede generar rutas externas. Por lo tanto, las bases de datos de estado de vínculo

mantenidas dentro de la NSSA contienen la ruta predeterminada, las rutas desde dentro del entorno OSPF (por ejemplo, rutas dentro del área y entre áreas) y las rutas externas generadas por el ASBR dentro del área.

El ABR que da servicio a la NSSA inunda las rutas externas desde dentro de la NSSA a través del resto de la red OSPF.

#### 5.6.6 Resumen de rutas OSPF

El resumen de rutas es el proceso de consolidar varias entradas de enrutamiento contiguas en un solo anuncio. Esto reduce el tamaño de la base de datos de estado de vínculo y la tabla de enrutamiento IP. En una red OSPF, el resumen se realiza en un router de borde. Hay dos tipos de resumen:

Resumen de rutas entre áreas: El ABR realiza el resumen entre áreas para un área. Se utiliza para resumir los anuncios de rutas que se originan en la zona. La ruta resumida es el anuncio en la columna vertebral. La red troncal recibe la ruta agregada y anuncia el resumen en otras áreas.

Resumen de rutas externas: Este tipo de resumen se aplica específicamente a las rutas externas inyectadas en OSPF. Esto lo realiza el ASBR distribuyendo las rutas en la red OSPF. La Figura 5-20 ilustra un ejemplo de resumen de rutas OSPF.

*Figura 5-20 Resumen de la ruta OSPF*

**OSPF**

**Área 1**

ASBR

10.99.192.0/24

a través de

10.99.254.0/24

RASGADURA

Red

10.99.0.0/24

a través de

10.99.83.0/24

ABR

R1

**OSPF Área 2**

10.99.0.0/28

10.99.192.0/28

**OSPF**

**Área 0**

Resumen externo

10.99.0.0/28

Resumen inter-área

10.99.192.0/28

En esta figura, el ASBR anuncia una única ruta de resumen para las 64 subredes ubicadas en el entorno RIP. Esta única ruta de resumen se inunda en toda la red OSPF. Además, el ABR está generando una única ruta de resumen para las 64 subredes ubicadas en el área 1. Esta ruta de resumen se inunda a través del área 0 y el área 2. Dependiendo de la configuración del ASBR, la ruta de resumen entre áreas también se puede redistribuir en la red RIP.

### 5.7 Protocolo de enrutamiento de puerta de enlace interior mejorado (EIGRP)

El protocolo de enrutamiento de puerta de enlace interior mejorado (EIGRP) se clasifica como un protocolo de enrutamiento híbrido. De manera similar a un algoritmo de vector de distancia, EIGRP utiliza métricas para determinar las rutas de red. Sin embargo, al igual que un protocolo de estado de vínculo, las actualizaciones de topología en un entorno EIGRP se basan en eventos.

EIGRP, como su nombre lo indica, es un protocolo de puerta de enlace interior diseñado para su uso dentro de un AS. En redes correctamente diseñadas, EIGRP tiene el potencial de mejorar la escalabilidad y una convergencia más rápida sobre algoritmos de vectores de distancia estándar. EIGRP también está mejor posicionado para soportar redes complejas y altamente redundantes.

EIGRP es un protocolo propietario desarrollado por Cisco Systems, Inc. En el momento de escribir este artículo, no es un protocolo estándar del IETF.

#### 5.7.1 Características de EIGRP

EIGRP tiene varias capacidades. Algunas de estas capacidades también están disponibles en vectores de distancia o algoritmos de estado de vínculo.

EIGRP mantiene una lista de rutas alternativas que se pueden utilizar si falla una ruta preferida. Cuando se produce un error en la ruta, la nueva ruta se instala inmediatamente en la tabla de enrutamiento IP. No se realiza ningún recálculo de ruta.

EIGRP permite actualizaciones parciales de enrutamiento. Cuando EIGRP descubre un router vecino, cada dispositivo intercambia toda su tabla de ruteo. Después del intercambio de información inicial, solo se propagan los cambios en la tabla de enrutamiento. No hay retransmisión periódica de toda la tabla de enrutamiento.

EIGRP utiliza una cantidad baja de ancho de banda. Durante las operaciones normales de red, solo los paquetes de saludo se transmiten a través de una red estable.

EIGRP admite superredes (CIDR) y máscaras de subred de longitud variable (VLSM). Esto permite al administrador de red asignar de manera eficiente los recursos de la dirección IP.

EIGRP admite la capacidad de resumir los anuncios de enrutamiento. Esto limita el anuncio de información de subred innecesaria.

EIGRP puede proporcionar enrutamiento de capa de red para múltiples protocolos, como AppleTalk, IPX y redes IP.

EIGRP admite el uso simultáneo de múltiples rutas de costo desigual a un destino. Cada ruta se instala en la tabla de enrutamiento IP. EIGRP también equilibra de forma inteligente la carga del tráfico a través de las múltiples rutas.

EIGRP utiliza una tabla de topología para instalar rutas en la tabla de ruteo IP. La tabla de topología enumera todas las redes de destino anunciadas actualmente por los routers vecinos. La tabla contiene toda la información necesaria para construir un conjunto de distancias y vectores a cada destino.

EIGRP mantiene una tabla para realizar un seguimiento del estado de cada vecino adyacente. Esto se denomina tabla de vecinos.

EIGRP puede garantizar la entrega ordenada de paquetes a un vecino. Sin embargo, no todos los tipos de paquetes deben transmitirse de forma fiable. Por ejemplo, en una red que admite la multidifusión, no es necesario enviar paquetes de saludo individuales y reconocidos a cada vecino. Para proporcionar un funcionamiento eficiente, la confiabilidad se proporciona solo cuando es necesario. Esto mejora el tiempo de convergencia en redes que contienen conexiones de velocidad variable.

##### Detección y recuperación de vecinos

EIGRP puede aprender dinámicamente sobre otros routers en redes conectadas directamente. Esto es similar al protocolo Hello utilizado para la detección de vecinos en un entorno OSPF.

Los dispositivos de una red EIGRP intercambian paquetes de saludo para verificar que cada vecino esté operativo. Al igual que OSPF, la frecuencia utilizada para intercambiar paquetes se basa en el tipo de red. Los paquetes se intercambian en un intervalo de cinco segundos en enlaces de gran ancho de banda (por ejemplo, segmentos LAN). De lo contrario, los paquetes de saludo en conexiones de ancho de banda inferior se intercambian cada 60 segundos.

Al igual que OSPF, EIGRP utiliza un temporizador de espera para eliminar vecinos inactivos. Este temporizador indica la cantidad de tiempo que un dispositivo seguirá considerando a un vecino activo sin recibir un paquete de saludo del vecino.

##### Algoritmo de enrutamiento EIGRP

EIGRP no se basa en actualizaciones periódicas para converger en la topología. En su lugar, crea una tabla de topología que contiene cada uno de los anuncios de su vecino. A diferencia de un protocolo de vector de distancia, estos datos no se descartan.

EIGRP procesa la información de la tabla de topología para determinar las mejores rutas a cada red de destino. EIGRP implementa un algoritmo conocido como Diffusing Update ALgorithm (DUAL).

##### Recálculo de rutas

Para un destino específico, el sucesor es el router vecino que se utiliza actualmente para el reenvío de paquetes. Este dispositivo tiene la ruta de menor costo al destino y se garantiza que no participará en un bucle de enrutamiento. Un sucesor factible asume la responsabilidad de reenvío cuando falla el router sucesor actual. El conjunto de sucesores factibles representa los dispositivos que pueden convertirse en sucesores sin necesidad de volver a calcular la ruta ni introducir bucles de enrutamiento.

Un nuevo cálculo de ruta se produce cuando no hay un sucesor factible conocido para el destino. El sucesor es el router vecino que se utiliza actualmente para el reenvío de paquetes. El proceso comienza con un router que envía un paquete de consulta de multidifusión para determinar si algún vecino conoce un sucesor factible para el destino. Un vecino responde si tiene un sucesor factible.

Si el vecino no tiene un sucesor factible, el vecino puede devolver una consulta que indica que también está realizando un recálculo de ruta. Cuando se produce un error en el vínculo a un vecino, todas las rutas que utilizaron ese vecino como único sucesor factible requieren un nuevo cálculo de ruta.

#### 5.7.2 Tipos de paquetes EIGRP

EIGRP utiliza cinco tipos de paquetes para establecer relaciones de vecino y anunciar información de enrutamiento:

Hola/acuse de recibo: Estos paquetes se utilizan para la detección de vecinos. Se anuncian en multidifusión en cada segmento de red. Se devuelven las respuestas de unidifusión al paquete de saludo. Un paquete de saludo sin ningún dato se considera un acuse de recibo.

Actualizaciones: Estos paquetes se utilizan para transmitir información de accesibilidad para cada destino. Cuando se detecta un nuevo vecino, se intercambian paquetes de actualización de unidifusión para permitir que cada vecino construya su tabla de topología. Otros tipos de anuncios (por ejemplo, cambios de métricas) utilizan paquetes de multidifusión. Los paquetes de actualización siempre se transmiten de forma fiable.

Consultas y respuestas: Estos paquetes se intercambian cuando un destino entra en un estado activo. Se envía un paquete de consulta de multidifusión para determinar si algún vecino contiene un sucesor factible para el destino. Los paquetes de respuesta de unidifusión se envían para indicar que el vecino no necesita entrar en un estado activo porque se ha identificado un sucesor factible. Los paquetes de consulta y respuesta se transmiten de forma fiable.

Solicitud: Estos paquetes se utilizan para obtener información específica de un vecino. Estos paquetes se utilizan en aplicaciones de servidor de rutas.

### 5.8 Protocolo de puerta de enlace exterior (EGP)

EGP es un protocolo de puerta de entrada exterior de mérito histórico. Fue uno de los primeros protocolos desarrollados para la comunicación entre sistemas autónomos. Se describe en RFC 904.

EGP asume que la red contiene una sola red troncal y que existe una sola ruta entre dos sistemas autónomos cualesquiera. Debido a esta limitación, el uso actual de EGP es mínimo. En la práctica, EGP ha sido reemplazado por BGP.

EGP se basa en sondeos periódicos mediante un intercambio de mensajes de saludo/te escucho.

Se utilizan para supervisar la accesibilidad de los vecinos y solicitar respuestas de actualización.

La puerta de enlace que se conecta a un AS solo puede anunciar las redes de destino a las que se puede acceder dentro del AS local. No anuncia información de accesibilidad sobre sus vecinos EGP fuera del AS.

### 5.9 Protocolo de puerta de enlace fronteriza (BGP)

El protocolo de puerta de enlace fronteriza (BGP) es un protocolo de puerta de enlace exterior. Originalmente se desarrolló para proporcionar un método sin bucles para intercambiar información de enrutamiento entre sistemas autónomos. Desde entonces, BGP ha evolucionado para admitir la agregación y el resumen de la información de enrutamiento.

BGP es un borrador de protocolo estándar de IETF descrito en RFC 4271. La versión descrita en este RFC es BGP versión 4. Siguiendo la convención estándar, este documento utiliza el término BGP cuando se hace referencia a la versión 4 de BGP.

#### 5.9.1 Conceptos y terminología de BGP

BGP utiliza terminología específica para describir el funcionamiento del protocolo. La figura 5-21 ilustra esta terminología.

AS1

AS2

AS3

ASX

OSPF/RIP

OSPF/RIP

OSPF/RIP

OSPF/RIP

OSPF/RIP

OSPF/RIP

OSPF/RIP

Altavoz BGP

OSPF/RIP

Altavoz BGP

OSPF/RIP

Altavoz BGP

OSPF/RIP

Altavoz BGP

OSPF/RIP

Altavoz BGP

IBGP

IBGP

EBGP

EBGP

EBGP

*Figura 5-21 Componentes de una red BGP*

BGP utiliza los siguientes términos:

Altavoz BGP: Un router configurado para admitir BGP.

Vecinos BGP (peers): Un par de altavoces BGP que intercambian información de enrutamiento. Hay dos tipos de vecinos BGP:

* Vecino interno (IBGP): Un par de altavoces BGP dentro del mismo AS.
* Vecino externo (EBGP): un par de vecinos BGP, cada uno en un

COMO. Estos vecinos suelen compartir una red conectada directamente.

Sesión BGP: una sesión TCP que conecta dos vecinos BGP. La sesión se utiliza para intercambiar información de enrutamiento. Los vecinos monitorean el estado de la

sesión mediante el envío de mensajes keepalive.1

Tipo de tráfico: BGP define dos tipos de tráfico:

* Local: el tráfico local a un AS se origina o termina dentro del AS. La dirección IP de origen o de destino reside en el AS.
* Tránsito: Cualquier tráfico que no sea tráfico local es tráfico de tránsito. Uno de los objetivos de BGP es minimizar la cantidad de tráfico de tránsito.

Tipo de AS: BGP define tres tipos de sistemas autónomos:

* Stub: Un AS stub tiene una sola conexión con otro AS. Un código auxiliar de AS solo transporta tráfico local.
* Multihost: Un AS multihost tiene conexiones a dos o más sistemas autónomos. Sin embargo, se ha configurado un AS multihost para que no reenvíe el tráfico de tránsito.
* Tránsito: Un AS de tránsito tiene conexiones a dos o más sistemas autónomos y transporta tráfico local y de tránsito. El AS puede imponer restricciones de política sobre los tipos de tráfico de tránsito que se reenviarán.

Dependiendo de la configuración de los dispositivos BGP dentro del AS 2 en la Figura 5-21 en la página 216, este sistema autónomo puede ser un AS multihost o un AS de tránsito.

Número de AS: Un número de 16 bits que identifica de forma única a un AS.

Ruta AS: Una lista de números AS que describen una ruta a través de la red. Un vecino BGP comunica rutas a sus pares.

Política de enrutamiento: Conjunto de reglas que restringen el flujo de paquetes de datos a través de la red. Las políticas de enrutamiento no están definidas en el protocolo BGP. Más bien, se utilizan para configurar un dispositivo BGP. Por ejemplo, un dispositivo BGP se puede configurar para que:

* Un AS con múltiples conexiones puede negarse a actuar como un AS de tránsito. Esto se logra anunciando solo aquellas redes contenidas dentro del AS.
* Un AS multihost puede realizar el enrutamiento de AS de tránsito para un conjunto restringido de sistemas autónomos adyacentes. Para ello, adapta los anuncios de enrutamiento enviados a los pares EBGP.
* Un AS puede optimizar el tráfico para utilizar una ruta de AS específica para determinadas categorías de tráfico.

Información de accesibilidad de la capa de red (NLRI): BGP utiliza NLRI para anunciar rutas. Consiste en un conjunto de redes representadas por la tupla

<longitud,prefijo>. Por ejemplo, la tupla <14,220.24.106.0> representa la ruta CIDR 220.24.106.0/14.

1 Este mensaje de mantenimiento se implementa en la capa de aplicación. Es independiente del mensaje keepalive disponible en muchas implementaciones de TCP.

Rutas y caminos: una ruta asocia un destino con una colección de atributos que describen la ruta al destino. El destino se especifica en formato NRLI. La ruta de acceso se notifica como una colección de atributos de ruta de acceso. Esta información se anuncia en los mensajes de ACTUALIZACIÓN. Para obtener información adicional que describe el mensaje UPDATE, consulte 5.9.3, "Descripción del protocolo" en la página 220.

#### 5.9.2 Comunicación entre el IBGP y el EBGP

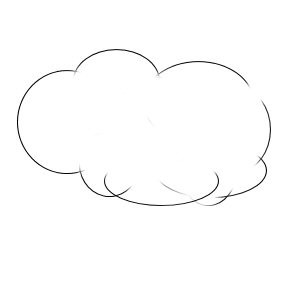
BGP no reemplaza al IGP que opera dentro de un AS. En su lugar, coopera con el IGP para establecer la comunicación entre sistemas autónomos. BGP dentro de un AS se utiliza para anunciar las rutas IGP locales. Estas rutas se anuncian a los pares BGP en otros sistemas autónomos. La figura 5-22 de la página 219 ilustra la comunicación que se produce entre los pares BGP. En este ejemplo se muestran cuatro sistemas autónomos. AS 2, AS 3 y AS 4 tienen cada uno una conexión EBGP a AS 1. Existe una malla completa de sesiones IBGP entre los dispositivos BGP dentro de AS 1.

La red 10.0.0.0/8 se encuentra dentro de AS 3. Utilizando BGP, la existencia de esta red se anuncia al resto del entorno:

R4 en AS 3 utiliza su conexión EBGP para anunciar la red a AS 1.

R1 en AS 1 utiliza sus conexiones IBGP para anunciar la red a R2 y R3.

R2 en AS 1 utiliza su sesión EBGP para anunciar la red en AS 2. R3 en AS 1 utiliza su sesión EBGP 5 para anunciar la red en AS 4.



BGP

R6

**COMO 2**

BGP

R5

**COMO 4**

BGP

R4

10.0.0.0/8

**COMO 3**

BGP

R2

**COMO 1**

BGP

R1

BGP

R3

IGP

Interconexión

IBGP

IBGP

IBGP

EBGP

EBGP

EBGP

*Figura 5-22 Comunicación EBGP e IBGP*

En la Figura 5-22 se muestran varios problemas operativos adicionales:

Función del BGP y el IGP: El diagrama muestra que, si bien el BGP por sí solo transporta información entre sistemas autónomos, tanto el BGP como el IGP se utilizan para transportar información a través de un AS.

Establecimiento de la sesión TCP entre pares: Antes de establecer una sesión BGP, un dispositivo verifica que la información de enrutamiento esté disponible para llegar al par:

* Pares EBGP: los pares EBGP suelen compartir una red conectada directamente. La información de enrutamiento necesaria para intercambiar paquetes BGP entre estos pares es trivial.
* Pares IBGP: Los pares IBGP se pueden ubicar en cualquier lugar dentro del AS. No es necesario que estén conectados directamente. BGP se basa en el IGP para localizar un par.

El reenvío de paquetes entre pares IBGP utiliza rutas aprendidas por IGP.

Malla completa de sesiones BGP dentro de un AS: los hablantes IBGP asumen que se ha establecido una malla completa de sesiones BGP entre pares en el mismo AS. En la Figura 5-22 de la página 219, los tres pares BGP en AS 1 están interconectados con sesiones BGP.

Cuando un altavoz BGP recibe una actualización de ruta de un par IBGP, el altavoz receptor utiliza EBGP para propagar la actualización a pares externos. Dado que el orador receptor asume que se ha establecido una malla completa de sesiones del IBGP, no propaga la actualización a otros pares del IBGP.

Por ejemplo, supongamos que no hubo ninguna sesión del IBGP entre R1 y R3 en Figure\_82. R1 recibe la actualización alrededor de 10.0.0.0/8 de AS 3. R1 reenvía la actualización a sus pares BGP, es decir, R2. R2 recibe la actualización de IBGP y la reenvía a sus pares de EBGP, es decir, R6. No se envía ninguna actualización a R3. Si R3 necesita recibir esta información, R1 y R3 deben configurarse para ser pares BGP.

#### 5.9.3 Descripción del protocolo

BGP establece una conexión TCP confiable entre pares. Las sesiones se establecen mediante el puerto TCP 179. BGP asume que la conexión de transporte administrará la fragmentación, la retransmisión, el reconocimiento y la secuenciación.

Cuando dos altavoces forman inicialmente una sesión BGP, intercambian toda su tabla de enrutamiento. Esta información de enrutamiento contiene la ruta de AS completa utilizada para llegar a cada destino. La información evita los bucles de enrutamiento y el comportamiento de conteo hasta el infinito observados en las redes RIP. Una vez que se ha intercambiado toda la tabla, los cambios en la tabla se comunican como actualizaciones incrementales.

##### Tipos de paquetes BGP

Todos los paquetes BGP contienen un encabezado estándar. El encabezado especifica el tipo de paquete BGP. Los tipos de paquetes BGP válidos incluyen:

OPEN: [[2]](#footnote-2)Este tipo de mensaje establece una sesión BGP entre dos nodos del mismo nivel.

ACTUALIZACIÓN: Este tipo de mensaje transfiere información de enrutamiento entre pares de GP.

NOTIFICACIÓN: Este mensaje se envía cuando se detecta una condición de error.

KEEPALIVE: Este mensaje determina si se puede acceder a los pares.

La Figura 5-23 muestra el flujo de estos tipos de mensajes entre dos sistemas autónomos.

BGP

BGP

ASY

BGP

ASX

Abrir

Mantente vivo

Actualizar

Notificación

*Figura 5-23 Flujos de mensajes BGP entre altavoces BGP*

##### Apertura y confirmación de una conexión BGP

Una vez que se ha establecido una sesión TCP entre dos nodos pares, cada router envía un mensaje OPEN al vecino. El mensaje OPEN incluye:

El número de AS y el identificador del router BGP del router de origen.

Un valor sugerido para el temporizador de espera. Discutimos la función de este temporizador en la siguiente sección.

Parámetros opcionales. Esta información se utiliza para autenticar a un par.

Un mensaje OPEN contiene soporte para autenticar la identidad de un par BGP. Sin embargo, el estándar BGP no especifica un mecanismo de autorización específico. Esto permite a los pares BGP seleccionar cualquier esquema de autorización compatible.

Un mensaje OPEN es reconocido por un mensaje KEEPALIVE. Una vez que los routers del mismo nivel han establecido una conexión BGP, pueden intercambiar información adicional.

##### Mantenimiento de la conexión BGP

BGP no utiliza ninguna señal de mantenimiento basada en transporte para determinar si se puede acceder a los pares. En su lugar, los mensajes BGP se intercambian periódicamente entre pares. Si no se reciben mensajes del par durante la duración especificada por el temporizador de espera, el router de origen asume que se ha producido un error. Cuando esto sucede, se envía una notificación de error al par y se cierra la conexión.

RFC 4271 recomienda un temporizador de espera de 90 segundos y un temporizador de mantenimiento de 30 segundos.

##### Envío de información de accesibilidad

La información de accesibilidad se intercambia entre pares en mensajes UPDATE.

BGP no requiere una actualización periódica de toda la tabla de enrutamiento BGP. Por lo tanto, cada altavoz BGP debe conservar una copia de la tabla de ruteo BGP actual utilizada por cada par. Esta información se mantiene durante la duración de la conexión. Una vez que los vecinos han realizado el intercambio inicial de información de enrutamiento completa, solo se intercambian actualizaciones incrementales de esa información.

Un mensaje UPDATE se utiliza para anunciar rutas factibles o retirar rutas no viables. El mensaje puede anunciar simultáneamente una ruta factible y retirar del servicio varias rutas no viables. La Figura 5-24 muestra el formato de un mensaje UPDATE:

Información de accesibilidad de la capa de red (NLRI).

Atributos de ruta (discutimos los atributos de ruta en "Atributos de ruta" en la página 223.

Rutas retiradas.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Número de octetos   |  |  | | --- | --- | | Encabezado común |  | | Longitud de rutas inviables | | Rutas retiradas | | Longitud total del atributo de ruta | | Atributos de ruta | | Información de accesibilidad de la capa de red | |  |   19Tipo = 2  2  Variable  2  Variable  Variable |

*Figura 5-24 Mensaje BGP UPDATE*

Se pueden utilizar varios atributos de ruta para describir una ruta.

###### Rutas retiradas

El campo de longitud de rutas inviables indica la longitud total del campo de rutas retiradas.

El campo de rutas retiradas proporciona una lista de prefijos de direcciones IP que no son factibles o que ya no están en servicio. Estas direcciones deben retirarse de la tabla de enrutamiento BGP. Las rutas retiradas se representan en el mismo formato de tupla que el NLRI.

##### Notificación de condiciones de error

Un dispositivo BGP puede observar condiciones de error que afectan a la conexión con un par. Los mensajes de NOTIFICACIÓN se envían al vecino cuando se detectan estas condiciones. Una vez enviado el mensaje, se cierra la conexión de transporte BGP. Esto significa que se desasignan todos los recursos para la conexión BGP. Las entradas de la tabla de enrutamiento asociadas con el par remoto se marcan como no válidas. Por último, se notifica a otros pares que estas rutas no son válidas.

Los mensajes de notificación incluyen un código de error y un subcódigo de error. Los códigos de error proporcionados por BGP incluyen:

Error de encabezado del mensaje

Error de mensaje OPEN

Error del mensaje UPDATE

Temporizador de espera caducado

Error de máquina de estado finito

Cesar

El subcódigo de error califica aún más el error específico. Cada código de error puede tener varios subcódigos asociados.

#### 5.9.4 Selección de trayecto

BGP es un protocolo de vector de ruta. En el enrutamiento vectorial de ruta, la ruta se expresa en términos de los dominios (o confederaciones) atravesados hasta el momento. La mejor ruta se obtiene comparando el número de dominios de cada ruta factible. Sin embargo, el enrutamiento entre AS complica este proceso. No existen métricas universalmente acordadas que se puedan utilizar para evaluar las rutas externas. Cada AS tiene su propio conjunto de criterios para la evaluación de rutas.

##### Atributos de ruta

Los atributos de ruta se utilizan para describir y evaluar una ruta. Los pares intercambian atributos de ruta junto con otra información de enrutamiento. Cuando un dispositivo anuncia una ruta,

Puede agregar o modificar los atributos de ruta antes de anunciar la ruta a un par. La combinación de atributos se utiliza para seleccionar la mejor ruta.

Cada atributo de ruta se coloca en una de las cuatro categorías separadas:

Obligatorio conocido: el atributo debe ser reconocido por todas las implementaciones de BGP. Debe enviarse en cada mensaje de ACTUALIZACIÓN.

Discrecional conocido: el atributo debe ser reconocido por todas las implementaciones de BGP. Sin embargo, no es necesario que se envíe en cada mensaje de ACTUALIZACIÓN.

Transitivo opcional: no es necesario que todas las implementaciones de BGP reconozcan este tipo de atributo. Una ruta con un atributo transitivo opcional no reconocido se acepta y simplemente se reenvía a otros pares BGP.

No transitivo opcional: no es necesario que todas las implementaciones de BGP reconozcan este tipo de atributo. Estos atributos se pueden ignorar y no pasar a otros pares BGP.

BGP define siete tipos de atributos para definir una ruta anunciada:

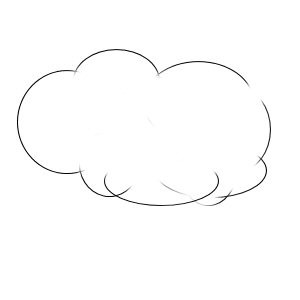
ORIGIN: Este atributo define el origen de la información de la ruta. Las selecciones válidas son IGP (interior del AS), EGP o INCOMPLETO. Este es un atributo obligatorio bien conocido.

AS\_PATH: Este atributo define el conjunto de sistemas autónomos que se deben atravesar para llegar a la red anunciada. Cada dispositivo BGP antepone su número de AS en la secuencia de ruta del AS antes de enviar la información de ruteo a un par EBGP. Utilizando la red de ejemplo que se muestra en la Figura 5-22 de la página 219, R4 anuncia la red 10.0.0.0 con un AS\_PATH de 3. Cuando la actualización atraviesa el AS 1, R2 le antepone su propio número de AS. Cuando la actualización de enrutamiento alcanza R6, el atributo AS\_PATH para la red 10.0.0.0 es <1 3>. Este es un atributo obligatorio bien conocido.

NEXT\_HOP: Este atributo define la dirección IP del siguiente salto utilizado para llegar al destino. Este es un atributo obligatorio bien conocido.

Para las actualizaciones de enrutamiento recibidas a través de conexiones EBGP, el salto siguiente suele ser la dirección IP del vecino EBGP en el AS remoto. BGP especifica que este salto siguiente se pasa sin modificación a cada vecino IBGP. Como resultado, cada vecino IBGP debe tener una ruta para llegar al vecino en el AS remoto. La figura 5-25 de la página 225 ilustra esta interacción.

*Figura 5-25 NEXT\_HOP atributo*



COMO 1

COMO 3

BGP

R1

BGP

R3

BGP

R4

172.16.1.2

172.16.1.1

10.0.0.0/8

EBGP

IBGP

En este ejemplo, cuando se envía una actualización de enrutamiento para la red 10.0.0.0/8 desde AS 3, R1 recibe la actualización con el atributo NEXT\_HOP establecido en 172.16.1.1.

Cuando esta actualización se reenvía a R3, la dirección del salto siguiente sigue siendo 172.16.1.1. R3 debe tener la información de enrutamiento adecuada para llegar a esta dirección. De lo contrario, R3 descartará los paquetes destinados a AS 3 si no se puede acceder al siguiente salto.

MULTI\_EXIT\_DISC (discriminador de salida múltiple, MED): este atributo se utiliza para discriminar entre varios puntos de salida a un AS vecino. Si esta información se recibe de un par EBGP, se propaga a cada par IBGP. Este atributo no se propaga a los pares de otros sistemas autónomos. Si todos los demás atributos son iguales, se prefiere el punto de salida con el valor MED más bajo. Se trata de un atributo no transitivo opcional. MED se analiza con más detalle en RFC 4451.

LOCAL\_PREF (preferencia local): Un altavoz BGP utiliza este atributo para informar a otros hablantes dentro del AS del grado de preferencia del hablante de origen por la ruta anunciada. A diferencia de MED, este atributo solo se utiliza dentro de un AS. El valor de la preferencia local no se distribuye fuera de un AS. Si todos los demás atributos son iguales, se prefiere la ruta con el mayor grado de preferencia. Este es un atributo discrecional bien conocido.

ATOMIC\_AGGREGATE: Este atributo se utiliza cuando un par BGP recibe anuncios para el mismo destino identificado en varias rutas no coincidentes (es decir, rutas superpuestas). Una ruta describe un conjunto más pequeño de destinos (un prefijo más específico), otras rutas describen un conjunto más grande de destinos (un prefijo menos específico). Este atributo es utilizado por el altavoz BGP para informar a los pares que ha seleccionado la ruta menos específica sin seleccionar la ruta más específica. Este es un atributo discrecional bien conocido. Una ruta con este atributo incluido puede atravesar sistemas autónomos que no figuran en la AS\_PATH.

AGREGADOR: Este atributo indica el último número de AS que formó la ruta agregada, seguido de la dirección IP del altavoz BGP que formó la ruta agregada. Para obtener más información sobre la agregación de rutas, consulte

5.9.6, "Agregación BGP" en la página 228. Se trata de un atributo transitivo opcional.

##### Proceso de decisión

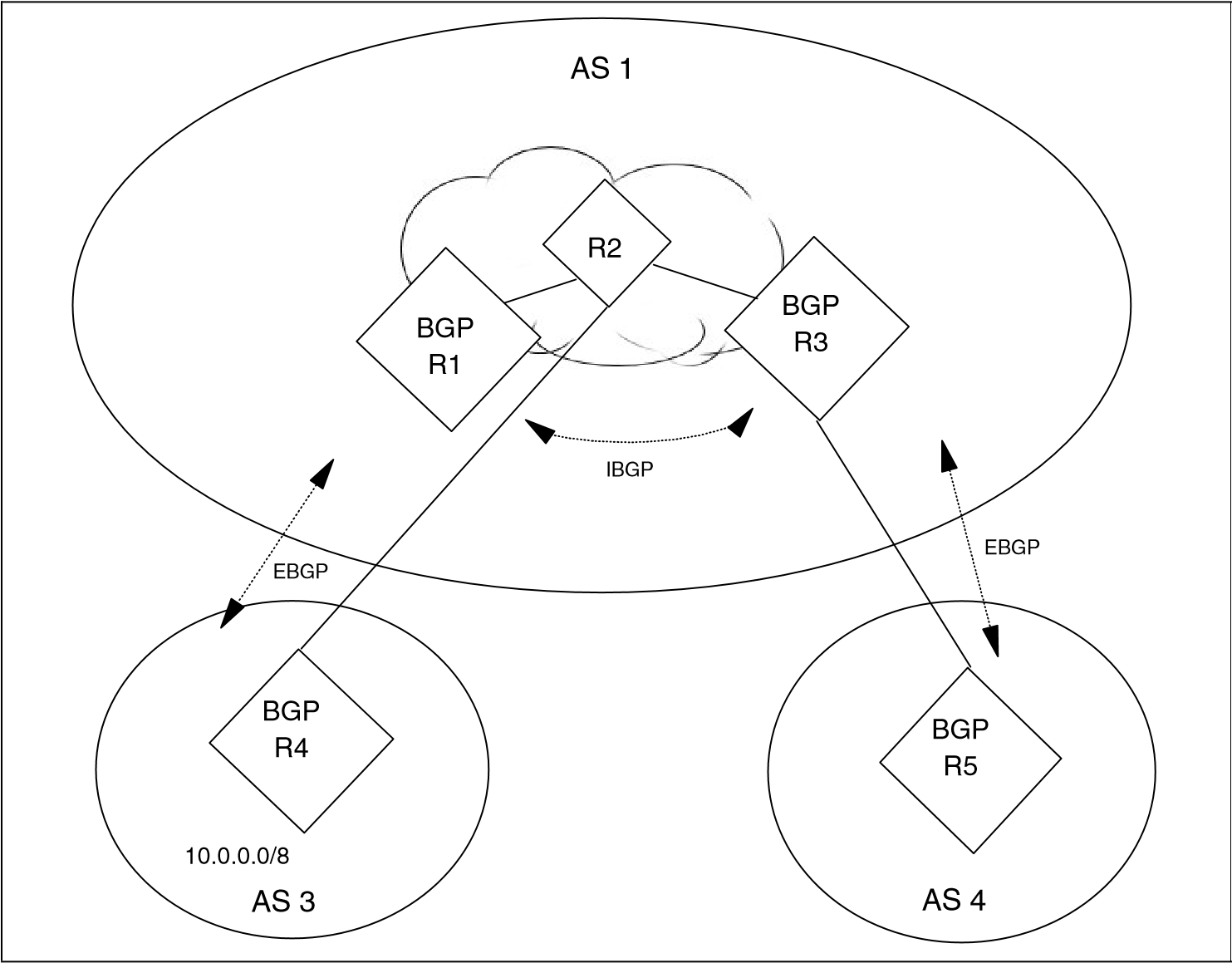
El proceso para seleccionar la mejor ruta utiliza los atributos de ruta que describen cada ruta. Se analizan los atributos y  *se asigna un grado de preferencia*. Debido a que puede haber varias rutas a un destino determinado, el proceso de selección de rutas determina el grado de preferencia para cada ruta factible. La ruta con el mayor grado de preferencia se selecciona como la mejor ruta. Esta es la ruta anunciada para cada vecino BGP. La agregación de rutas también se puede realizar durante este proceso. Cuando hay varias rutas a un destino, BGP realiza un seguimiento de cada ruta individual. Esto permite una convergencia más rápida a la ruta alternativa cuando se produce un error en la ruta principal.

#### 5.9.5 Sincronización BGP

La figura 5-26 de la página 227 muestra un ejemplo de un AS que proporciona servicio de tránsito. En este ejemplo, AS 1 se utiliza para transportar tráfico entre AS 3 y AS 4. Dentro de AS 1, R2 no está configurado para BGP. Sin embargo, R2 se utiliza para la comunicación entre R1 y R3. El tráfico entre estos dos nodos BGP atraviesa físicamente a través de R2.

Utilizando el flujo de actualización de enrutamiento descrito anteriormente, la red 10.0.0.0/8 se anuncia mediante la conexión EBGP entre R4 y R1. R1 pasa el anuncio de red a R3 mediante su conexión IBGP existente. Debido a que R2 no está configurado para BGP, no es consciente de ninguna red en AS 3. Se produce un problema si R3 necesita comunicarse con un dispositivo en AS 3. R3 pasa el tráfico a R2. Sin embargo, debido a que R2 no tiene ninguna ruta a las redes AS 3, el tráfico se descarta.

Si R3 anuncia la red 10.0.0.0/8 a AS 4, el problema continúa. Si AS 4 necesita comunicarse con un dispositivo en AS 3, los paquetes se reenvían de R5 a R3. R3 reenvía los paquetes a R2, donde se descartan.



*Figura 5-26 Sincronización de BGP*

Esta situación se soluciona mediante la regla de sincronización de BGP. La regla establece que un AS de tránsito no anunciará una ruta antes de que todos los routers dentro del AS hayan aprendido sobre la ruta. En este ejemplo, R3 no anunciará la existencia de las redes en AS 3 hasta que R2 haya construido una tabla de ruteo adecuada.

Hay tres métodos para implementar la regla de sincronización:

Habilite BGP en todos los dispositivos dentro del AS de tránsito. En esta solución, R2 tiene una sesión IBGP con R1 y R3. R2 se entera de la red 10.0.0.0/8 al mismo tiempo que se anuncia a R3. En ese momento, R3 anuncia las rutas a su par en AS 4.

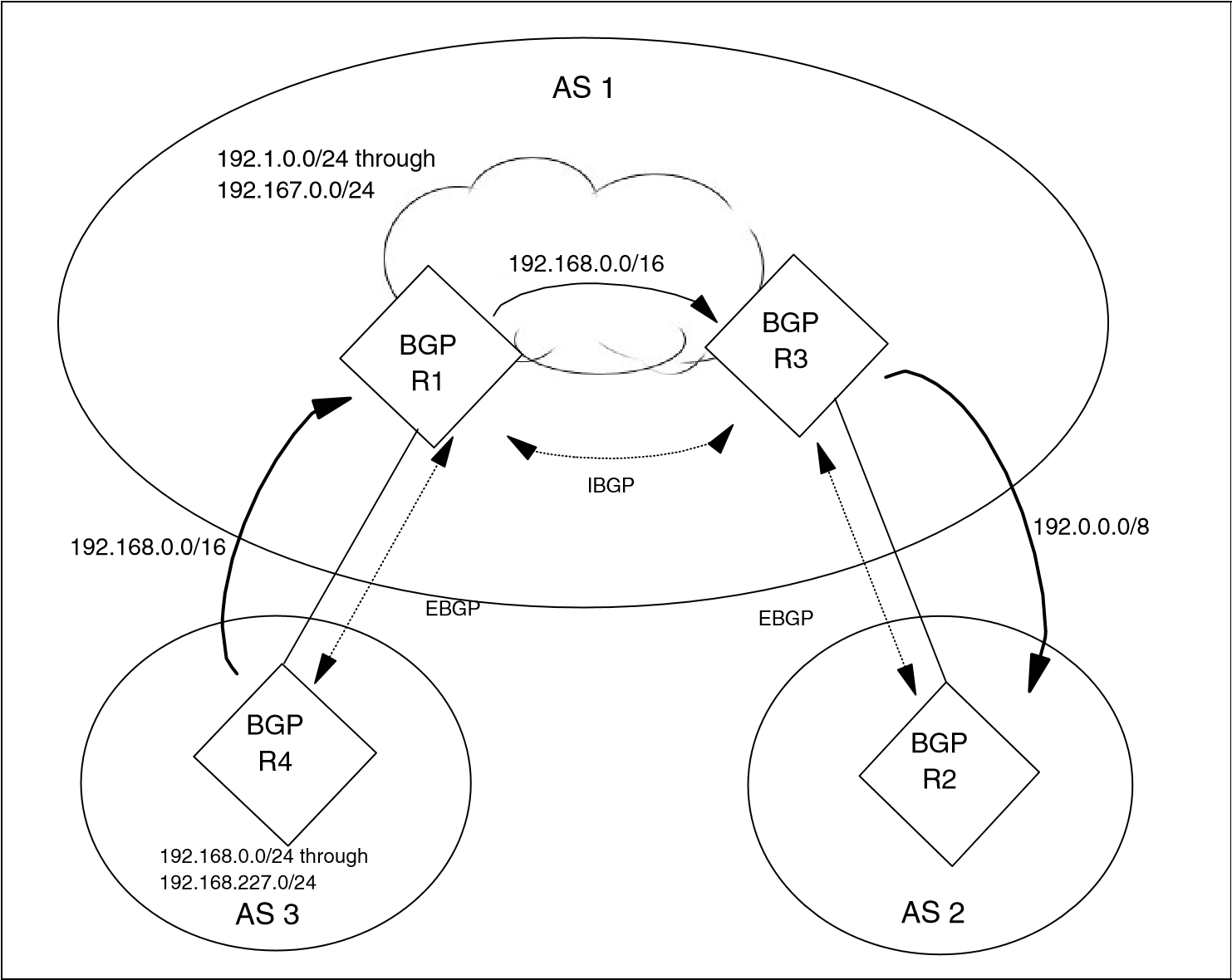
Redistribuya las rutas en el IGP utilizado dentro del área de tránsito. En esta solución, R1 redistribuye la red 10.0.0.0/8 en el IGP dentro de AS 1. R3 aprende de la red a través de dos protocolos de enrutamiento: BGP e IGP. Después

R3 se entera de la red a través del IGP, es seguro que otros routers dentro del AS también han aprendido de las rutas. En ese momento, R3 anuncia las rutas a su par en AS 4.

Encapsule el tráfico de tránsito a través del AS. En esta solución, el tráfico de tránsito se encapsula dentro de datagramas IP dirigidos a la puerta de enlace de salida. Debido a que esto no requiere que el IGP lleve información de ruteo exterior, no se requiere sincronización entre BGP y el IGP. R3 puede anunciar inmediatamente las rutas a su par en AS 4.

#### 5.9.6 Agregación BGP

La principal mejora introducida en la versión 4 de BGP fue la compatibilidad con CIDR y la agregación de rutas. Estas características permiten a los pares BGP consolidar varias entradas de enrutamiento contiguas en un solo anuncio. Mejora significativamente la escalabilidad de BGP en grandes entornos de trabajo en red. La figura 5-27 de la página 229 ilustra estas funciones.



*Figura 5-27 Agregación de rutas BGP*

Estos diagramas muestran tres sistemas autónomos interconectados por BGP. En este ejemplo, las redes 192.168.0.0 a 182.168.227.0 se encuentran dentro de AS 3. Para reducir el tamaño de los anuncios de enrutamiento, R4 agrega estas redes individuales en una sola entrada de ruta antes de anunciarlas en AS 1. La entrada única 192.168.0.0/16 representa una superred CIDR válida aunque sea una red ilegal de Clase C.

Las rutas agregadas de BGP contienen información adicional dentro del atributo de ruta de AS\_PATH. Cuando se generan entradas agregadas a partir de un conjunto de rutas más específicas, se combinan los atributos AS\_PATH de las rutas más específicas. Por ejemplo, en la Figura 5-27, la ruta agregada 192.0.0.0/8 se anuncia desde el AS 1 al AS 2. Este agregado representa el conjunto de rutas más específicas desplegadas dentro de AS 1 y AS 3. Cuando esta ruta agregada se envía a AS 2, el

AS\_PATH atributo consta de <1 3>. Esto se hace para evitar la información de enrutamiento

Bucles. Se puede producir un bucle si AS 1 generó un agregado con un atributo AS\_PATH de <1>. Si el AS 2 tuviera una conexión directa con el AS 3, la ruta con el AS\_PATH menos específico anunciado desde el AS 1 puede generar un bucle. Esto se debe a que AS 2 no sabe que este agregado contiene redes ubicadas dentro de

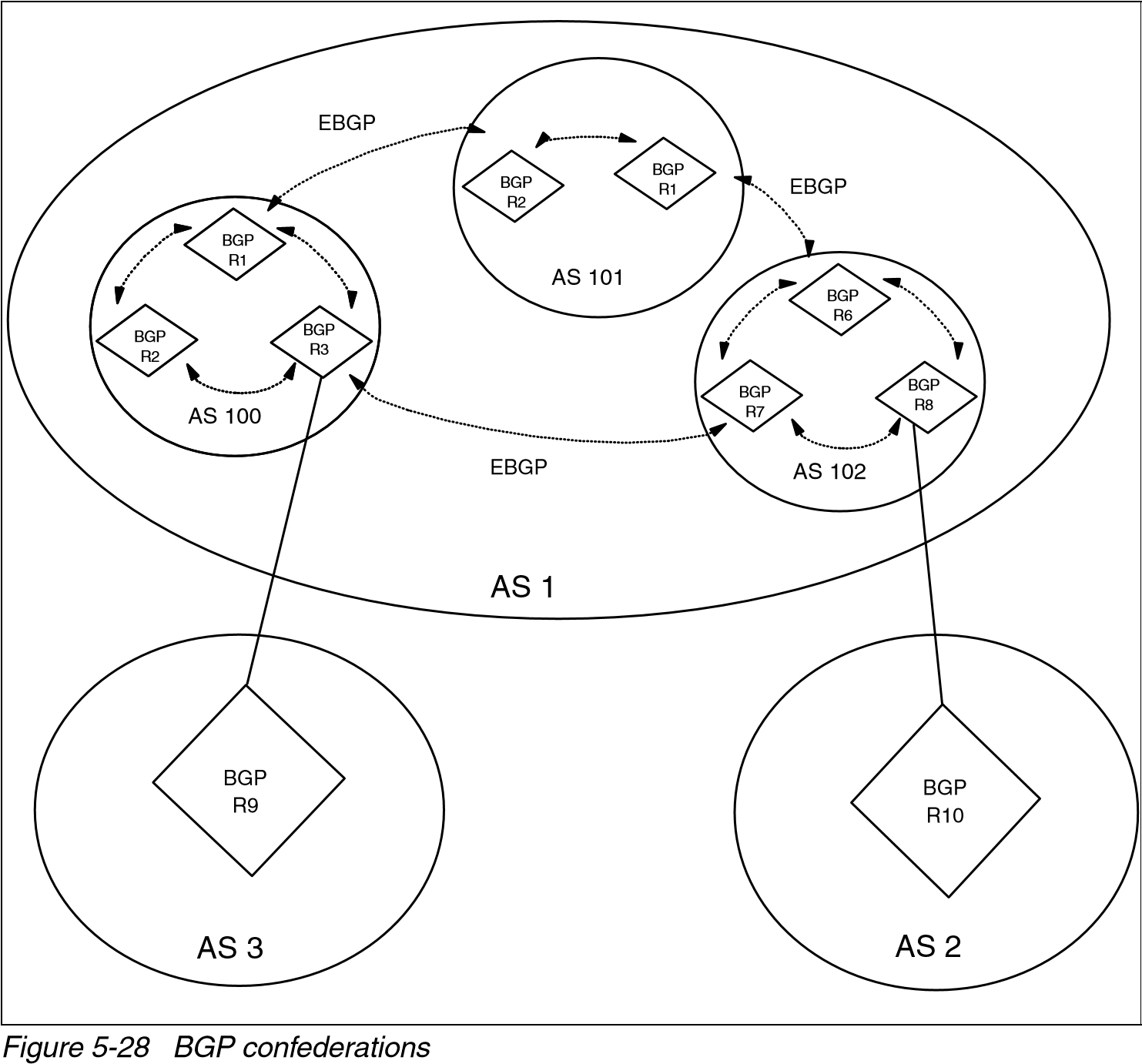
COMO 3.

#### 5.9.7 Confederaciones BGP

BGP requiere que todos los altavoces dentro de un solo AS tengan un conjunto completamente mallado de conexiones IBGP. Esto puede ser un problema de escalado en redes que contienen un gran número de pares IBGP. El uso de confederaciones BGP soluciona este problema.

Una confederación BGP crea un conjunto de sistemas autónomos que representan un único AS a los pares externos a la confederación. Esto elimina el requisito de malla completa y reduce la complejidad de la administración.

La figura 5-28 ilustra el funcionamiento de una confederación BGP. En esta red de ejemplo, AS 1 contiene ocho altavoces BGP. Una red BGP estándar requeriría 28 sesiones IBGP para engranar completamente los altavoces.



Una confederación divide el AS en un conjunto de dominios. En este ejemplo, AS 1 contiene tres dominios. Los dispositivos dentro de un dominio tienen un conjunto completamente mallado de conexiones IBGP. Cada dominio también tiene una conexión EBGP con otros dominios dentro de la confederación. En la red de ejemplo, R1, R2 y R3 tienen sesiones IBGP completamente maltratadas. R1 tiene una sesión EBGP dentro de la confederación a R4. R3 tiene una sesión EBGP fuera de la confederación a R9.

A cada router de la confederación se le asigna un ID de confederación. Un miembro de la confederación utiliza este ID en todas las comunicaciones con dispositivos fuera de la confederación. En este ejemplo, a cada router se le asigna un ID de confederación de AS 1.

Todas las comunicaciones de AS 1 a AS 2 o AS 3 parecen haberse originado en el ID de confederación de AS 1. Aunque la comunicación entre dominios dentro de una confederación se produce con EBGP, los dominios intercambian actualizaciones de enrutamiento como si estuvieran conectados por IBGP. En concreto, la información contenida en los atributos NEXT\_HOP, MULTI\_EXIT\_DESC y LOCAL\_PREF se conserva entre dominios. La confederación parece ser un único AS frente a otros sistemas autónomos.

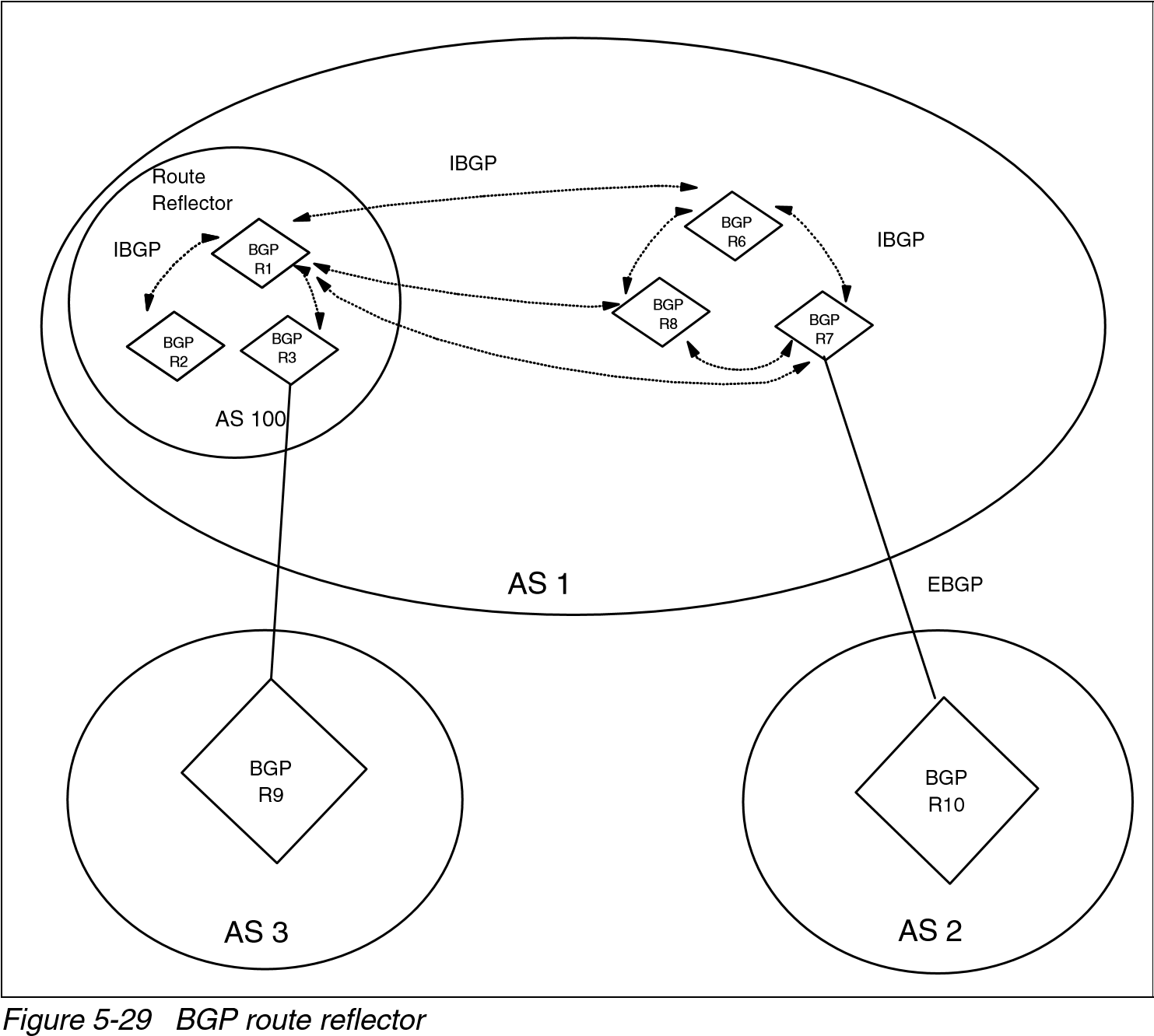
Las confederaciones BGP se describen en RFC 3065. En el momento de escribir este artículo, se trata de una norma propuesta. En cualquier caso, las confederaciones BGP se han desplegado ampliamente en Internet. Numerosos proveedores admiten esta función.

#### 5.9.8 Reflectores de ruta BGP

Los reflectores de ruta son otra solución para abordar el requisito de una malla completa de sesiones IBGP entre pares en un AS. Como se indicó anteriormente, cuando un altavoz BGP recibe una actualización de un par IBGP, el altavoz receptor propaga la actualización solo a los pares EBGP. El orador receptor no reenvía la actualización a otros pares del IBGP.

Los reflectores de ruta relajan esta restricción. Los hablantes de BGP pueden anunciar rutas aprendidas del IBGP a ciertos pares del IBGP. La figura 5-29 de la página 232 muestra un entorno utilizando reflectores de ruta. R1 está configurado como un reflector de ruta para R2 y R3. R2 y R3 son clientes de reflector de ruta de R1. No se define ninguna sesión IBGP entre R2 y R3. Cuando R3 recibe una actualización de EBGP de AS 3, se pasa a R1 mediante IBGP. Dado que R1 está configurado como reflector, R1 reenvía la actualización del IBGP a R2.

La Figura 5-29 también ilustra la interacción entre los reflectores de ruta y los altavoces BGP convencionales dentro de un AS.



En la Figura 5-29, R1, R2 y R3 se encuentran en el dominio del reflector de ruta. R6, R7 y R8 son altavoces BGP convencionales que contienen una malla completa de conexiones de pares IBGP. Además, cada uno de estos altavoces se empareja con el reflector de ruta. Esta configuración permite la comunicación completa del IBGP dentro de AS 1.

Aunque no se muestra en la Figura 5-29, un AS puede contener más de un reflector de ruta. Cuando esto ocurre, cada reflector trata a los demás reflectores como un par IBGP convencional.

Los reflectores de ruta se describen en RFC 4456. En el momento de escribir este artículo, se trata de una norma propuesta.

### 5.10 Selección del protocolo de encaminamiento

La elección de un protocolo de enrutamiento es una decisión importante para el administrador de la red.

Tiene un gran impacto en el rendimiento general de la red. La selección depende de la complejidad, el tamaño y las políticas administrativas de la red.

Es posible que el protocolo elegido para un tipo de red no sea adecuado para otros tipos de redes. Cada entorno único debe evaluarse en función de una serie de requisitos de diseño fundamentales:

Escalabilidad a grandes entornos: El crecimiento potencial de la red dicta la importancia de este requisito. Si se necesita compatibilidad con redes grandes y altamente redundantes, considere el estado del vínculo o los algoritmos híbridos. Los algoritmos de vector de distancia no se escalan en estos entornos.

Estabilidad durante las interrupciones: los algoritmos de vector de distancia pueden introducir inestabilidad de la red durante los períodos de interrupción. Los problemas de conteo hasta el infinito (5.3.5, "Convergencia y conteo hasta el infinito" en la página 185) pueden causar bucles de enrutamiento u otras rutas de enrutamiento no óptimas. Los algoritmos híbridos o de estado de enlace reducen la posibilidad de que se produzcan estos problemas.

Velocidad de convergencia: las actualizaciones activadas proporcionan la capacidad de iniciar inmediatamente la convergencia cuando se detecta un error. Los tres tipos de protocolos admiten esta característica. Un factor que contribuye a la convergencia es el tiempo necesario para detectar un fallo. En las redes OSPF y EIGRP, se debe perder una serie de paquetes de saludo antes de que comience la convergencia. En entornos RIP, los anuncios de ruta posteriores deben perderse antes de que se inicie la convergencia. Estos tiempos de detección aumentan el tiempo necesario para restablecer la comunicación.

Métricas: Las métricas proporcionan la capacidad de preparar las rutas de enrutamiento adecuadas a través de la red. Los algoritmos de estado de enlace tienen en cuenta el ancho de banda al calcular las rutas. EIGRP mejora esto para incluir el retardo de la red en el cálculo de la ruta.

Compatibilidad con VLSM: la disponibilidad de intervalos de direcciones IP dicta la importancia de este requisito. En entornos con un suministro limitado de direcciones, el administrador de red debe desarrollar un esquema de direccionamiento que se superponga de forma inteligente a la red. VLSM es un componente importante de este plan. El uso de rangos de direcciones privadas también puede abordar este problema.

Interoperabilidad de proveedores: Los tipos de dispositivos implementados en una red indican la importancia de este requisito. Si la red contiene equipos de varios proveedores, utilice protocolos de enrutamiento estándar. El IETF ha dictado las políticas operativas para los algoritmos de vector de distancia y estado de enlace descritos en este documento. La implementación de estos algoritmos evita cualquier problema de interoperabilidad que se encuentre con protocolos no estándar.

Facilidad de implementación: Los protocolos de vector de distancia son el protocolo de enrutamiento más simple de configurar y mantener. Debido a esto, estos protocolos tienen la mayor base de implementación. Se requiere una capacitación limitada para realizar la resolución de problemas en estos entornos.

En entornos pequeños y no cambiantes, las rutas estáticas también son fáciles de implementar. Estas definiciones solo cambian cuando se agregan o quitan sitios de la red. El administrador debe evaluar la importancia de cada uno de estos requisitos a la hora de determinar el protocolo de enrutamiento adecuado para un entorno.

### 5.11 Funciones adicionales realizadas por el router

Las principales funciones realizadas por un router se relacionan con la gestión de la tabla de enrutamiento IP y el reenvío de datos. Sin embargo, el router debe ser capaz de proporcionar información que alerte a otros dispositivos sobre posibles problemas de red.

Esta información es proporcionada por el protocolo ICMP descrito en 3.2, "Protocolo de mensajes de control de Internet (ICMP)" en la página 109. La información incluye:

Destino ICMP inalcanzable: La dirección de destino especificada en el paquete IP hace referencia a una red IP desconocida.

Redireccionamiento ICMP: Redirecciona el tráfico a un router más adecuado a lo largo de la ruta hacia el destino.

Extinción de origen ICMP: se han encontrado problemas de congestión (por ejemplo, demasiados datagramas entrantes para el espacio de búfer disponible) en un dispositivo a lo largo de la ruta hacia el destino.

Tiempo de vida de ICMP excedido: el campo de tiempo de vida de un datagrama IP ha llegado a cero. El paquete no se puede entregar en el destino final.

Además, cada enrutador IP debe admitir las siguientes operaciones y mensajes ICMP base:

Problema de parámetros: Este mensaje se devuelve al origen del paquete si se encuentra un problema con el encabezado IP. El mensaje indica el tipo y la ubicación del problema. El router descarta el paquete con errores.

Solicitud/respuesta de máscara de dirección: Un router debe implementar el soporte para recibir mensajes de solicitud de máscara de dirección ICMP y responder con mensajes de respuesta de máscara de dirección ICMP.

Marca de tiempo: El router debe devolver una respuesta de marca de tiempo a cada mensaje de marca de tiempo que se reciba. Debe diseñarse para una variabilidad mínima en el retardo. Para sincronizar el reloj en el router, se puede utilizar el protocolo de servidor de hora UDP o el protocolo de hora de red (NTP).

Solicitud/respuesta de eco: Un router debe implementar una función de servidor ICMP Echo que reciba las solicitudes enviadas al router y envíe las respuestas correspondientes. El router puede ignorar las solicitudes de eco ICMP dirigidas a direcciones IP broadcast o IP multicast.

### 5.12 Procesos de enrutamiento en sistemas basados en UNIX

Este capítulo se centra en los protocolos disponibles en los routers IP estándar. Sin embargo, varios de estos protocolos también están disponibles en sistemas basados en UNIX.

Estos protocolos a menudo se implementan mediante uno de estos dos procesos:

Enrutado (pronunciado route-D): Este es un proceso de enrutamiento básico para el enrutamiento interior. Se suministra con la mayoría de las implementaciones TCP/IP. Implementa el protocolo RIP.

Puerta (pronunciado puerta-D): Este es un proceso más sofisticado que permite el enrutamiento interior y exterior. Puede implementar una serie de protocolos, incluidos OSPF, RIP-2 y BGP-4.

### 5.13 RFC relevantes para este capítulo

Las siguientes RFC proporcionan información detallada sobre los protocolos y arquitecturas de conexión presentados a lo largo de este capítulo:

[RFC 904 - Especificación formal del protocolo de puerta de enlace exterior (abril de 1984)](http://www.ietf.org/rfc/rfc904.txt)

[RFC 1058 - Protocolo de información de enrutamiento (junio de 1988)](http://www.ietf.org/rfc/rfc1058.txt)

[RFC 1322 - Un enfoque unificado para el enrutamiento entre dominios (mayo de 1992)](http://www.ietf.org/rfc/rfc1322.txt)

[RFC 1812 - Requisitos para enrutadores IP versión 4. (Junio de 1995)](http://www.ietf.org/rfc/rfc1812.txt)

[RFC 2080 – RIPng para IPv6 (enero de 1997)](http://www.ietf.org/rfc/rfc2080.txt)

[RFC 2328 – OSPF Versión 2 (abril de 1998)](http://www.ietf.org/rfc/rfc2328.txt)

[RFC 2453 – RIP Versión 2 (noviembre de 1998)](http://www.ietf.org/rfc/rfc2453.txt)

[RFC 3065 - Confederaciones de sistemas autónomos para BGP (febrero de 2001)](http://www.ietf.org/rfc/rfc3065.txt)

[RFC 3101 - La opción OSPF Not-So-Stubby Area (NSSA) (enero de 2003)](http://www.ietf.org/rfc/rfc3101.txt)

[RFC 4271 - Un protocolo de puerta de enlace fronteriza 4 (BGP-4) (enero de 2006)](http://www.ietf.org/rfc/rfc4271.txt)

[RFC 4451 - Consideraciones sobre BGP MULTI\_EXIT\_DISC (MED) (marzo de 2006)](http://www.ietf.org/rfc/rfc4451.txt)

[RFC 4456 - Reflexión de ruta BGP: una alternativa al BGP interno de malla completa (IBGP) (abril de 2006)](http://www.ietf.org/rfc/rfc4456.txt)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **12** | |

## Capítulo 12. Protocolos de directorio y nomenclatura

Un problema inherente a la disponibilidad de recursos a través de una red es la creación de una forma fácil de acceder a estos recursos. En redes pequeñas, puede ser bastante fácil simplemente recordar o escribir la información necesaria para acceder de forma remota a los recursos. Sin embargo, esta solución no se escala a medida que las redes y el número de recursos disponibles siguen creciendo. Esto se vuelve cada vez más complejo a medida que los recursos están disponibles fuera de las redes individuales, a múltiples redes, o incluso a través de Internet, en múltiples plataformas y a través de una variedad de hardware diferente. Para superar esto, se diseñaron métodos de directorio y nomenclatura para proporcionar un método uniforme de obtener la información necesaria para acceder a un recurso en red. En este capítulo se revisan cuatro de estos métodos:

Domain Name System (DNS)

Dynamic Domain Name System (DDNS)

Network Information System (NIS)

Lightweight Directory Access Protocol (LDAP)

© Copyright IBM Corp. 1989-2006. All rights reserved.

### 12.1 Domain Name System (DNS)

The Domain Name System is a standard protocol with STD number 13, and its status is recommended. It is described in RFC 1034 and RFC 1035. This section explains the implementation of the Domain Name System and the implementation of name servers.

The early Internet configurations required the use of only numeric IP addresses. Because this was burdensome and much harder to remember than just the name of a system, this evolved into the use of symbolic host names. For example, instead of typing: TELNET 10.12.7.14

You can type:

TELNET MyHost

MyHost is then translated in some way to the IP address 10.12.7.14. Though using host names makes the process of accessing a resource easier, it also introduces the problem of maintaining the mappings between IP addresses and high-level machine names in a coordinated and centralized way.

Initially, host names to address mappings were maintained by the Network Information Center (NIC) in a single file (HOSTS.TXT), which was fetched by all hosts using FTP. This is called a *flat namespace*. But due to the explosive growth in the number of hosts, this mechanism became too cumbersome (consider the work involved in the addition of just one host to the Internet) and was replaced by a new concept: *Domain Name System*. Hosts on smaller networks can continue to use a local flat namespace (the HOSTS.LOCAL file) instead of or in addition to the Domain Name System. Outside of small networks, however, the Domain Name System is essential. This system allows a program running on a host to perform the mapping of a high-level symbolic name to an IP address for any other host without requiring every host to have a complete database of host names.

#### 12.1.1 The hierarchical namespace

Consider the typical internal structure of a large organization. Because the chief executive cannot do everything, the organization will probably be partitioned into divisions, each of them having autonomy within certain limits. Specifically, the executive in charge of a division has authority to make direct decisions, without permission from the chief executive.

Domain names are formed in a similar way, and will often reflect the hierarchical delegation of authority used to assign them. For example, consider the name: myHost.myDept.myDiv.myCorp.com

In this example, we know that there is a single host name *myHost*, which exists within the *myDept.myDiv.myCorp* subdomain. The *myDept.myDiv.myCorp* subdomain is one of the subdomains of *myDiv.myCorp.com* subdomain, which is in turn one of the subdomains of *myCorp.com.* Finally, *myCorp.com* is a subdomain of *com.* This hierarchy is better illustrated in Figure 12-1.

(

root

)

du

e

ov

g

mil

m

co

Pentagon

RPA

DA

Corp

my

SF

N

OT

D

me

Na

Ti

tle

Name

Title

my

Di

v

y

ourDiv

yourDept

myDept

*Figure 12-1 DNS hierarchical namespace*

We discuss this hierarchical structure at greater length in the following sections.

**12.1.2**  **Fully qualified domain names (FQDNs)**  When using the Domain Name System, it is common to work with only a part of the domain hierarchy, such as the myDivision.myCorp.com domain. The Domain Name System provides a simple method of minimizing the typing necessary in this circumstance. If a domain name ends in a dot (for example,

myDept.myDiv.myCorp.com.), it is assumed to be complete. This is called a *fully qualified domain name (FQDN)* or an *absolute domain name*. However, if it does not end in a dot (for example, myDept.myDiv), it is incomplete and the DNS resolver may complete this by appending a suffix such as .myCorp.com to the domain name. The rules for doing this are implementation-dependent and locally configurable.

#### 12.1.3 Generic domains

The top-level names are called the generic top-level domains (gTLDs), and can be three characters or more in length. Table 12-1 shows some of the top-level domains of today's Internet domain namespace.

*Table 12-1 Current generic domains*

|  |  |
| --- | --- |
| **Domain name** | **Meaning** |
| aero | The air transport industry |
| biz | Business use |
| cat | The Catalan culture |
| com | Commercial organizations |
| coop | Cooperatives |
| edu | Educational organizations |
| gov | U.S. governmental agencies |
| info | Informational sites |
| int | International organizations |
| jobs | Employment-related sites |
| mil | The U.S. military |
| mobi | Mobile devices sites |
| museum | Museums |
| **Domain name** | **Meaning** |
| name | Family and individual sites |
| net | Network infrastructures |
| org | Non-commercial organizations |
| pro | Professional sites |
| travel | The travel industry |

These names are registered with and maintained by the Internet Corporation for Assigned Names and Numbers (ICANN). For current information, see the ICANN Web site at:

[http://www.icann.org](http://www.icann.org/)

#### 12.1.4 Country domains

There are also top-level domains named for the each of the ISO 3166 international 2-character country codes (from *ae* for the United Arab Emirates to *zw* for Zimbabwe). These are called the *country* domains or the *geographical* domains. Many countries have their own second-level domains underneath which parallel the generic top-level domains. For example, in the United Kingdom, the domains equivalent to the generic domains .*com* and .*edu* are .*co*.*uk* and .*ac*.*uk* (ac is an abbreviation for academic). There is a .us top-level domain, which is organized geographically by state (for example, .ny.us refers to the state of New York). See RFC 1480 for a detailed description of the .us domain.

#### 12.1.5 Mapping domain names to IP addresses

The mapping of names to addresses consists of independent, cooperative systems called name servers. A name server is a server program that holds a master or a copy of a name-to-address mapping database, or otherwise points to a server that does, and that answers requests from the client software, called a name resolver.

Conceptually, all Internet domain servers are arranged in a tree structure that corresponds to the naming hierarchy in Figure 12-1 on page 427. Each leaf represents a name server that handles names for a single subdomain. Links in the conceptual tree do not indicate physical connections. Instead, they show which other name server a given server can contact.

#### 12.1.6 Mapping IP addresses to domain names: Pointer queries

The Domain Name System provides for a mapping of symbolic names to IP addresses and vice versa. While the hierarchical structure makes it easy in principle to search the database for an IP address using its symbolic name, the

process of mapping an IP address to a symbolic name cannot use the same process. Therefore, there is another namespace that facilitates the reverse mapping of IP address to symbolic name. It is found in the domain in-addr.arpa (arpa is used because the Internet was originally the *ARPAnet*).

Not including IPv6, IP addresses are normally written in dotted decimal format, and there is one layer of domain for each hierarchy. Contrary to domain names, which have the least-significant parts of the name first, the dotted decimal format has the most significant bytes first. Therefore, in the Domain Name System, the dotted decimal address is shown in reverse order.

For example, consider the following IPv4 address:

129.34.139.30

The in-add.arpa address for this is:

30.139.34.129.in-addr.arpa.

This is handled slightly different for IPv6 addresses. Because of the IPv6 address’ structure, the reverse order is done in nibbles in stead of octets. Also, the in-addr.arpa domain does not include IPv6. Instead, the domain used is IP6.ARPA. For example, consider the following IPv6 address:

4321:0:1:2:3:4:567:89ab

Breaking this into nibbles, reversing the odder, and appending the domain yields:

b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.IP6.ARPA

Given an IP address, the Domain Name System can be used to find the matching host name. A domain name query to do this is called a *pointer query*.

#### 12.1.7 The distributed name space

The Domain Name System uses the concept of a *distributed name space*. Symbolic names are grouped into *zones of authority*, more commonly referred to as *zones*. In each of these zones, one or more hosts has the task of maintaining a database of symbolic names and IP addresses within that zone, and provides a server function for clients who want to translate between symbolic names and IP addresses. These local name servers are then (through the internetwork on which they are connected) logically interconnected into a hierarchical tree of *domains*. Each zone contains a part or a *subtree* of the hierarchical tree, and the names within the zone are administered independently of names in other zones. Authority over zones is vested in the name servers.

Normally, the name servers that have authority for a zone will have domain names belonging to that zone, but this is not required. Where a domain contains a subtree that falls in a different zone, the name server or servers with authority over the superior domain are said to *delegate authority* to the name server or servers with authority over the subdomain. Name servers can also delegate authority to themselves; in this case, the domain name space is still divided into zones moving down the domain name tree, but authority for two zones is held by the same server. The division of the domain name space into zones is accomplished using resource records stored in the Domain Name System.

At the top-level root domain there is an exception to this. There is no higher system to which authority can be delegated, but it is not desirable to have all queries for fully qualified domain names to be directed to just one system. Therefore, authority for the top-level zones is shared among a set of *root name servers*[[3]](#footnote-3) coordinated by the ICANN.

To better illustrate the process of resolving a symbolic name to an IP address, consider a query for myHost.myDept.myCorp.com, and let us assume that our name server does not have the answer already in its cache. The query goes to the .com root name server, which in turn forwards the query to a server with an NS record for myCorp.com. At this stage, it is likely that a name server has been reached that has cached the needed answer. However, the query could be further delegated to a name server for myDept.myCorp.com

As a result of this scheme:

Rather than having a central server for the database, the work that is involved in maintaining this database is off-loaded to hosts throughout the name space.

Authority for creating and changing symbolic host names and responsibility for maintaining a database for them is delegated to the organization owning the zone (within the name space) containing those host names.

From the user's standpoint, there is a single database that deals with these address resolutions. The user might be aware that the database is distributed, but generally need not be concerned about this.

**Note:** Although domains within the namespace will frequently map in a logical fashion to networks and subnets within the IP addressing scheme, this is not a requirement of the Domain Name System. Consider a router between two subnets. It has two IP addresses, one for each network adapter, but it would not normally have two symbolic names.

#### 12.1.8 Domain name resolution

The domain name resolution process can be summarized in the following steps:

1. A user program issues a request such as the **gethostbyname()** system call (this particular call asks for the IP address of a host by passing the host name) or the **gethostname()** system call (which asks for a host name of a host by passing the IP address).
2. The resolver formulates a query to the name server. (Full resolvers have a local name cache to consult first; stub resolvers do not. See “Domain name full resolver” and “Domain name stub resolver” on page 434.)
3. The name server checks to see if the answer is in its local authoritative database or cache, and if so, returns it to the client. Otherwise, it queries other available name servers, starting down from the root of the DNS tree or as high up the tree as possible.
4. The user program is finally given a corresponding IP address (or host name, depending on the query) or an error if the query could not be answered. Normally, the program will not be given a list of all the name servers that have been consulted to process the query.

Domain name resolution is a client/server process (see 11.1.1, “The client/server model” on page 408). The client function (called the *resolver* or *name resolver*) is transparent to the user and is called by an application to resolve symbolic high-level names into real IP addresses or vice versa. The name server (also called a *domain name server*) is the server application providing the translation between high-level machine names and the IP addresses. The query/reply messages can be transported by either UDP or TCP.

##### Domain name full resolver

Figure 12-2 shows a program called a *full resolver*, which is distinct from the user program, that forwards all queries to a name server for processing. Responses are cached by the name server for future use.

User

Program

Full

Resolver

Cache

Name

Server

Foreign

Name

Server

C

a

c

h

e

Database

user

query

user

response

query

response

r

q

*Figure 12-2 DNS: Using a full resolver for domain name resolution*

##### Domain name stub resolver

Figure 12-3 shows a *stub resolver*, a routine linked with the user program, that forwards the queries to a name server for processing. Responses are cached by the name server, but not usually by the resolver, although this is implementation dependent. On most platforms, the stub resolver is implemented by two library routines (or by some variation of these routines): **gethostbyname()** and **gethostbyaddr()**. These are used for converting host names to IP addresses and vice versa. Stub resolvers are much more common than full resolvers.

User

Program

Stub

Resolver

Name

Server

Database

Foreign

Name

Server

C

a

c

h

e

query

response

r

q

*Figure 12-3 DNS: Using a stub resolver for domain name resolution*

##### Domain name resolver operation

Domain name queries can be one of two types: *recursive* or *iterative* (also called *non-recursive*). A flag bit in the domain name query specifies whether the client desires a recursive query, and a flag bit in the response specifies whether the server supports recursive queries. The difference between a recursive and an iterative query arises when the server receives a request for which it cannot supply a complete answer by itself. A recursive query requests that the server issues a query itself to determine the requested information and returns the complete answer to the client. An iterative query means that the name server

returns what information it has available and also a list of additional servers for the client to contact to complete the query.

Domain name responses can be one of two types: *authoritative* and *non-authoritative*. A flag bit in the response indicates which type a response is. When a name server receives a query for a domain in a zone over which it has authority, it returns all of the requested information in a response with the authoritative answer flag set. When it receives a query for a domain over which it does not have authority, its actions depend on the setting of the recursion desired flag in the query:

If the recursion desired flag is set and the server supports recursive queries, it will direct its query to another name server. This will either be a name server with authority for the domain given in the query, or it will be one of the root name servers. If the second server does not return an authoritative answer (for example, if it has delegated authority to another server), the process is repeated.

When a server (or a full resolver program) receives a response, it will cache it to improve the performance of repeat queries. The cache entry is stored for a maximum length of time specified by the originator in a 32-bit *time-to-live (TTL)* field contained in the response. A typical TTL value is 86,400 seconds (one day).

If the recursion desired flag is not set or the server does not support recursive queries, it will return whatever information it has in its cache and also a list of additional name servers to be contacted for authoritative information.

##### Domain name server operation

Each name server has *authority* for zero or more zones. There are three types of

|  |  |
| --- | --- |
| name servers: |  |
| **Primary** | A primary name server loads a zone's information from disk and has authority over the zone. |
| **Secondary** | A secondary name server has authority for a zone, but obtains its zone information from a primary server using a process called *zone transfer*. To remain synchronized, the secondary name servers query the primary on a regular basis (typically three hours) and re-execute the zone transfer if the primary has been updated. A name server can operate as a primary or a secondary name server for multiple domains, or a primary for some domains and as a secondary for others. A primary or secondary name server performs all of the functions of a caching-only name server. |
| **Caching-only** | A name server that does not have authority for any zone is called a caching-only name server. A caching-only name server obtains all of its data from primary or secondary name servers as required. It requires at least one NS record to point to a name server from which it can initially obtain information. |

When a domain is registered with the root and a separate zone of authority established, the following rules apply:

The domain must be registered with the root administrator.

There must be an identified administrator for the domain.

There must be at least two name servers with authority for the zone that are accessible from outside and inside the domain to ensure no single point of failure.

We also recommend that name servers that delegate authority apply these rules, because the delegating name servers are responsible for the behavior of name servers under their authority.

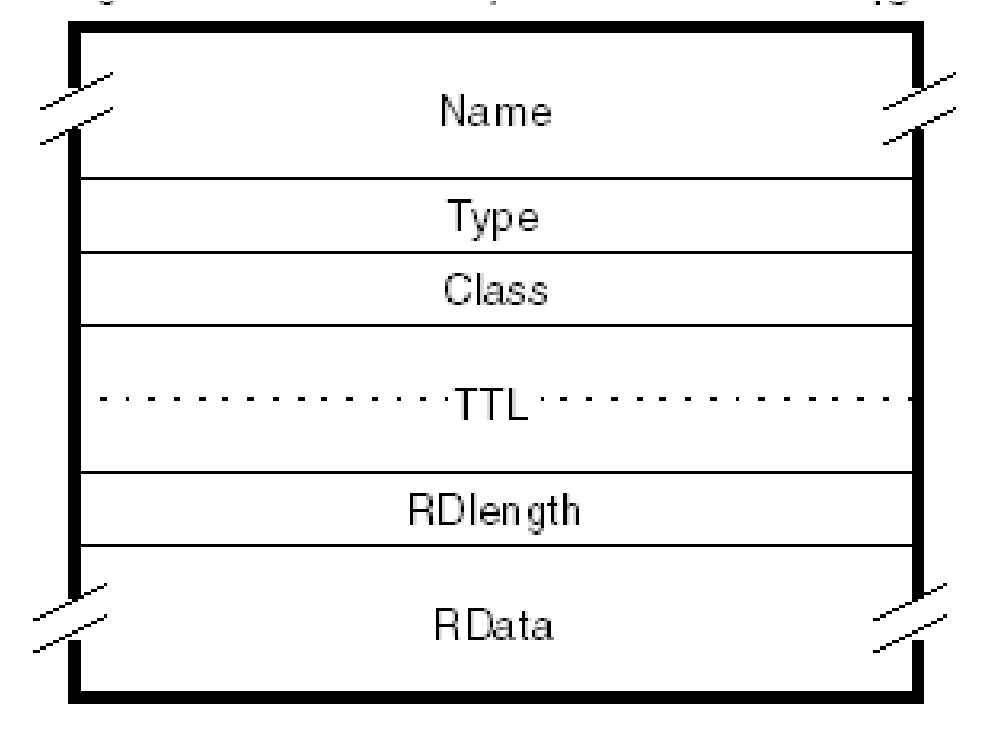
#### 12.1.9 Domain Name System resource records

The Domain Name System's distributed database is composed of *resource records* (RRs), which are divided into classes for different kinds of networks. We only discuss the Internet class of records. Resource records provide a mapping between domain names and *network objects*. The most common network objects are the addresses of Internet hosts, but the Domain Name System is designed to accommodate a wide range of different objects.

A zone consists of a group of resource records, beginning with a Start of

Authority (SOA) record. The SOA record identifies the domain name of the zone. There will be a name server (NS) record for the primary name server for this zone. There might also be NS records for the secondary name servers. The NS records are used to identify which of the name servers are authoritative (see “Domain name resolver operation” on page 434). Following these records are the resource records, which might map names to IP addresses or aliases to names.

The following figure shows the general format of a resource record (Figure 12-4).



*Figure 12-4 DNS general resource record format*

Where:

|  |  |
| --- | --- |
| **Name** | The domain name to be defined. The Domain Name System is very general in its rules for the composition of domain names. However, it recommends a syntax for domain names that minimizes the likelihood of applications that use a DNS resolver (that is, nearly all TCP/IP applications) from misinterpreting a domain name. A domain name adhering to this recommended syntax will consist of a series of labels consisting of alphanumeric characters or hyphens, each label having a length of between 1 and 63 characters, starting with an alphabetic character. Each pair of labels is separated by a dot (period) in human-readable form, but not in the form used within DNS messages. Domain names are not case-sensitive. |
| **Type** | Identifies the type of the resource in this record. There are numerous possible values, but some of the more common ones, along with the RFCs which define them, are listed in Table 12-2 on page 438. |
| **Class** | Identifies the protocol family. The only commonly used value is IN (the Internet system), though other values are defined by  RFC 1035 and include:   * CS (value 2): The CSNET class. This has been obsoleted. * CH (value 3): The CHAOS class. * HS (value 4): The Hesiod class. |
| **TTL** | The time-to-live (TTL) time in seconds for which this resource record will be valid in a name server cache. This is stored in the DNS as an unsigned 32-bit value. A typical value for records pointing to IP addresses is 86400 (one day). |
| **RDlength** | An unsigned 16-bit integer that specifies the length, in octets, of the RData field. |
| **RData** | A variable length string of octets that describes the resource. The format of this information varies according to the Type and Class of the resource record. |

*Table 12-2 Some of the possible resource record types*

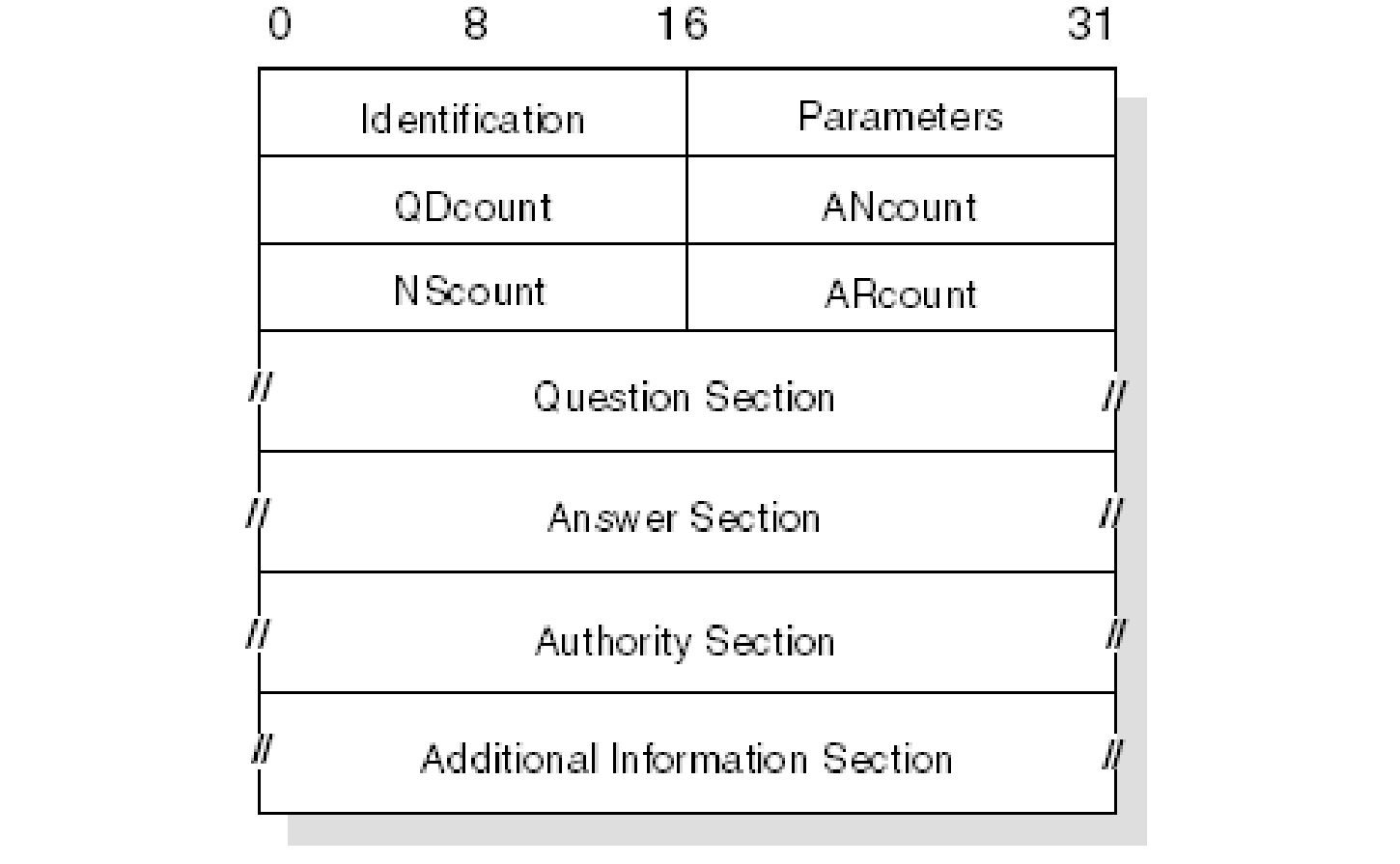
|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Value** | **Meaning** | **RFC def** |
| A | 1 | A host address | 1035 |
| NS | 2 | An authoritative name server | 1035 |
| CNAME | 5 | The canonical name for an alias | 1035 |
| SOA | 6 | Marks the start of a zone of authority | 1035 |
| MB | 7 | A mailbox domain name (experimental) | 1035 |
| MG | 8 | A mail group member (experimental) | 1035 |
| MR | 9 | A mail rename domain name (experimental) | 1035 |
| NULL | 10 | A NULL resource record (experimental) | 1035 |
| WKS | 11 | A well-known service description | 1035 |
| PTR | 12 | A domain name pointer | 1035 |
| HINFO | 13 | Host information | 1035 |
| MINFO | 14 | Mailbox or mail list information | 1035 |
| MX | 15 | Mail exchangea | 1035 |
| TXT | 16 | Text strings | 1035 |
| RP | 17 | Responsible person record | 1183 |
| AFSDB | 18 | Andrew File System database | 1183 |
| X25 | 19 | X.25 resource record | 1183 |
| ISDN | 20 | ISDN resource record | 1183 |
| RT | 21 | Route Through resource record | 1183 |
| NSAP | 22 | Network Service Access Protocol record | 1348 |
| **Type** | **Value** | **Meaning** | **RFC def** |
| NSAP-PTR | 23 | NSAP Pointer record | 1348 |
| KEY | 25 | The public key associated with a DNS name | 2535 |
| AAAA | 28 | An IPv6 address record | 3596 |
| LOC | 29 | GPS resource record | 1876 |
| SRV | 33 | Defines the services available in a zone | 2872 |
| CERT | 37 | Certificate resource records | 4398 |
| A6 | 38 | Forward mapping of an IPv6 address | 2874 |
| DNAME | 39 | Delegation of IPv6 reverse addresses | 2672 |
| DS | 39 | Delegated Signer record (DNS security) | 4034 |
| RRSIG | 46 | Resource record digital signature | 4034 |
| NSEC | 47 | Next Secure record (DNS security) | 4034 |
| DNSKEY | 48 | Public Key record (DNS security | 4034 |

a. The MX Type obsoletes Types MD (value 3, Mail destination) and MF (value 4, Mail forwarder).

#### 12.1.10 Domain Name System messages

All messages in the Domain Name System protocol use a single format. This format is shown in Figure 12-5 on page 440. This frame is sent by the resolver to the name server. Only the header and the question section are used to form the query. Replies and forwarding of the query use the same frame, but with more sections filled in (the answer/authority/additional sections).

*Figure 12-5 DNS message format*



##### Header format

The header section is always present and has a fixed length of 12 bytes. The other sections are of variable length.

**ID** A 16-bit identifier assigned by the program. This identifier is copied in the corresponding reply from the name server and can be used for differentiation of responses when multiple queries are outstanding at the same time.

**Parameters** A 16-bit value in the following format (Table 12-3).

*Table 12-3 Parameters*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 10 | 11 | 12 13 | 14 | 15 |
| Q  R | Op code |  |  | A  A | T  C | R  D | R  A | Zero |  | Rcode |  |  |

Where:

|  |  |
| --- | --- |
| **QR** | Flag identifying a query (0) or a response(1). |
| **Op code** | 4-bit field specifying the kind of query: |

1. Standard query (QUERY)
2. Inverse query (IQUERY)
3. Server status request (STATUS)

Other values are reserved for future use:

|  |  |
| --- | --- |
| **AA** | Authoritative answer flag. If set in a response, this flag specifies that the responding name server is an authority for the domain name sent in the query. |
| **TC** | Truncation flag. Set if message was longer than permitted on the physical channel. |
| **RD** | Recursion desired flag. This bit signals to the name server that recursive resolution is asked for. The bit is copied to the response. |
| **RA** | Recursion available flag. Indicates whether the name server supports recursive resolution. |
| **Zero** | 3 bits reserved for future use. Must be zero. |
| **Rcode** | 4-bit response code. Possible values are:   1. No error. 2. Format error. The server was unable to interpret the message. 3. Server failure. The message was not processed because of a problem with the server. 4. Name error. The domain name in the query does not exist.   This is only valid if the AA bit is set in the response.   1. Not implemented. The requested type of query is not implemented by name server. 2. Refused. The server refuses to respond for policy reasons.   Other values are reserved for future use. |
| **QDcount** | An unsigned 16-bit integer specifying the number of entries in the question section. |
| **ANcount** | An unsigned 16-bit integer specifying the number of RRs in the answer section. |
| **NScount** | An unsigned 16-bit integer specifying the number of name server RRs in the authority section. |
| **ARcount** | An unsigned 16-bit integer specifying the number of RRs in the additional records section. |

##### Question section

The next section contains the queries for the name server. It contains QDcount (usually 1) entries, each in the format shown in Figure 12-6.

*Figure 12-6 DNS question format*

*2*



Where:

|  |  |
| --- | --- |
| **Length** | A single byte giving the length of the next label. |
| **Label** | One element of the domain name characters (for example, ibm from ral.ibm.com). The domain name referred to by the question is stored as a series of these variable length labels, each preceded by a 1-byte length. |
| **00** | X'00' indicates the end of the domain name and represents the null label of the root domain. |
| **Type** | 2 bytes specifying the type of query. It can have any value from the Type field in a resource record. |
| **Class** | 2 bytes specifying the class of the query. For Internet queries, this will be IN. |

2 Note that all of the fields are byte-aligned. The alignment of the Type field on a 4-byte boundary is for example purposes and is not required by the format.

For example, the domain name mydiv.mycorp.com is encoded with the following fields:

X'05'

"mydiv"

X'06'

"mycorp"

X'03'

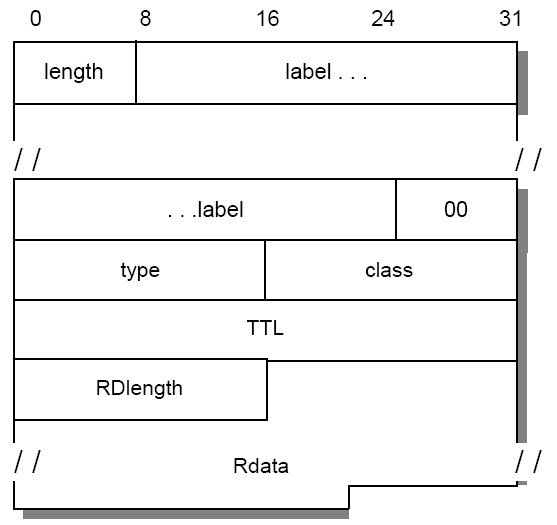
"com"

X'00'

Therefore, the entry in the question section for mydiv.mycorp.com requires 22 bytes: 18 to store the domain name and 2 each for the Qtype and Qclass fields.

##### Answer, authority, and additional resource sections

These three sections contain a variable number of resource records. The number is specified in the corresponding field of the header. The resource records are in the format shown in Figure 12-7.



*Figure 12-7 DNS: Answer Record Entry format [[4]](#footnote-4)*

Where the fields before the TTL field have the same meanings as for a question entry and:

|  |  |
| --- | --- |
| **TTL** | A 32-bit time-to-live value in seconds for the record. This defines how long it can be regarded as valid. |
| **RDlength** | A 16-bit length for the Rdata field. |
| **Rdata** | A variable length string whose interpretation depends on the Type field. |

##### Message compression

In order to reduce the message size, a compression scheme is used to eliminate the repetition of domain names in the various RRs. Any duplicate domain name or list of labels is replaced with a pointer to the previous occurrence. The pointer has the form of a 2-byte field as shown in Figure 12-8.



*Figure 12-8 DNS message compression*

Where:

The first 2 bits distinguish the pointer from a normal label, which is restricted to a 63-byte length plus the length byte ahead of it (which has a value of <64).

The offset field specifies an offset from the start of the message. A zero offset specifies the first byte of the ID field in the header.

If compression is used in an Rdata field of an answer, authority, or additional section of the message, the preceding RDlength field contains the real length after compression is done.

Refer to 12.2, “Dynamic Domain Name System” on page 453 for additional message formats.

##### Using the DNS Uniform Resource Identifiers (URI)

A DNS can also be queried using a Uniform Resource Identifier This is defined in RFC 4501. Strings are not case sensitive, and adhere to the following format:

“dns:” + [ “//” + dnsauthority + “:” + port + ”/” ] + dnsname +

[ “?” + dnsquery ]

|  |  |  |
| --- | --- | --- |
| Where: |  | |
| **dnsauthority** | The DNS server to which the query should be sent. If this is left blank, the query is sent to the default DNS server. | |
| **dnsname** | The name or IP address to be queried. |
| **dnsquery** | The type of the query to be performed. This can be any combination, separated by a semicolon (;), of: |

**CLASS** Usually IN for internet, the class of the query **TYPE** The type of resource record desired

For example, a request using the URI to resolve www.myCorp.com to an IP address might appear as follows: dns:www.mycorp.com

Additionally, the same request can be sent to the server at 10.1.2.3 on port 5353 using the following: dns://10.1.2.3:5353/www.mycorp.com

Finally, this same query can be made specifying a CLASS of IN and a TYPE of A:

dns://10.1.2.3:5353/www.mycorp.com?class=IN;type=A

#### 12.1.11 A simple scenario

Consider a stand-alone network (no outside connections), consisting of two physical networks:

One has an Internet network address of 129.112.

One has a network address of 194.33.7.

They are interconnected by an IP gateway (VM2). See Figure 12-9for more details.



*Figure 12-9 DNS: A simple configuration: Two networks connected by an IP gateway*

Assume the name server function has been assigned to VM1. Remember that the domain hierarchical tree forms a logical tree, completely independent of the physical configuration. In this simple scenario, there is only one level in the domain tree, which will be referred to as *test.example*.

The zone data for the name server appears as shown in Figure 12-10 and continued in Figure 12-11 on page 448.

;note: an SOA record has no TTL field

;

$origin test.example. ;note 1 @ IN SOA VM1.test.example. ADM.VM1.test.example.

(870611 ;serial number for data

1800 ;secondary refreshes every 30 mn

300 ;secondary reties every 5 mn

604800 ;data expire after 1 week

86400) ;minimum TTL for data is 1 week

;

@ 99999 IN NS VM1.test.example. ;note 2 ;

VM1 99999 IN A 129.112.1.1 ;note 3

99999 IN WKS 129.112.1.1 TCP (SMTP ;note 4

FTP

TELNET

NAMESRV)

;

RT1 99999 IN A 129.112.1.2

IN HINFO IBM RT/PC-AIX ;note 5

RT2 99999 IN A 129.112.1.3

IN HINFO IBM RT/PC-AIX

PC1 99999 IN A 129.112.1.11

PC2 99999 IN A 194.33.7.2

PC3 99999 IN A 194.33.7.3

;

;VM2 is an IP gateway and has 2 different IP addresses

;

VM2 99999 IN A 129.112.1.4

99999 IN A 194.33.7.1

99999 IN WKS 129.112.1.4 TCP (SMTP FTP) IN HINFO IBM-3090-VM/CMS

*Figure 12-10 Zone data for the name server, continued in Figure 12-11 on page 448*

|  |
| --- |
| ;  4.1.112.129.in-addr.arpa. IN PTR VM2 ;note 6  ;  ;Some mailboxes  ;  central 10 IN MX VM2.test.example. ;notes 7 and 8  ;  ;a second definition for the same mailbox, in case VM2 is down  ; central 20 IN MX VM1.test.example. waste 10 IN MX VM2.test.example. |

*Figure 12-11 Zone data for the name server, continued from Figure 12-10 on page 447*

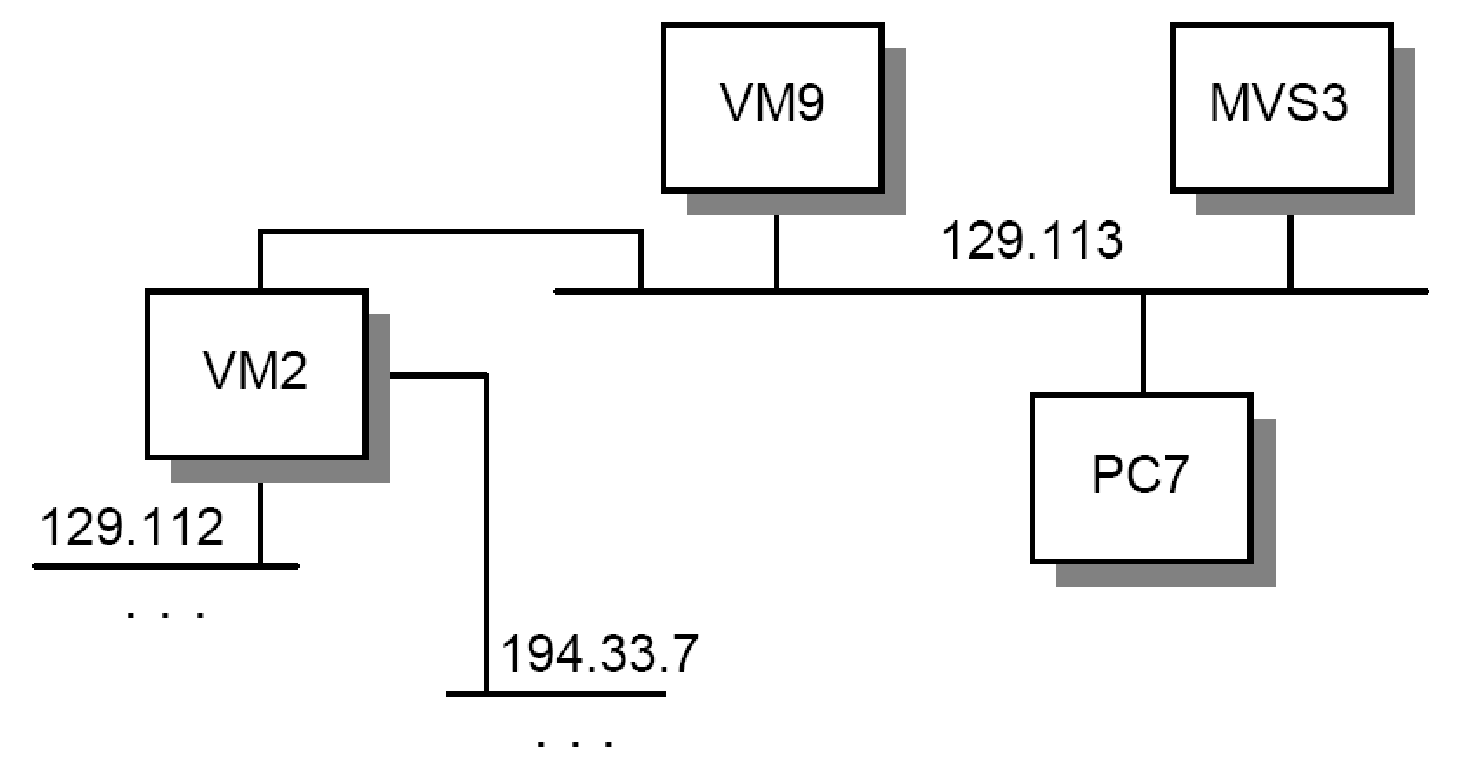
|  |
| --- |
| **Notes for Figure 12-10 on page 447 and Figure 12-11:**   1. The $origin statement sets the @ variable to the zone name   (test.example.). Domain names that do not end with a period are suffixed with the zone name. Fully qualified domain names (those ending with a period) are unaffected by the zone name.   1. Defines the name server for this zone. 2. Defines the Internet address of the name server for this zone. 3. Specifies well-known services for this host. These are expected to always be available. 4. Gives information about the host. 5. Used for inverse mapping queries (see 12.1.6, “Mapping IP addresses to domain names: Pointer queries” on page 430). 6. Will allow mail to be addressed to user@central.test.example. 7. See 15.1.2, “SMTP and the Domain Name System” on page 565 for the use of these definitions. |

#### 12.1.12 Extended scenario

Consider the case where a connection is made to a third network (129.113),

which has an existing name server with authority for that zone (see

Figure 12-12).



*Figure 12-12 DNS: Extended configuration - Third network connected to existing configuration*

Let us suppose that the domain name of the other network is tt.ibm.com and that its name server is located in VM9. All we have to do is add the address of this name server to our own name server database (in the named.ca initial cache file) and reference the other network by its own name server. The following two lines are all that is needed to do that:

tt.ibm.com. 99999 IN NS VM9.tt.ibm.com. VM9.tt.ibm.com. 99999 IN A 129.13.1.9

This simply indicates that VM9 is the authority for the new network and that all queries for that network will be directed to that name server.

#### 12.1.13 Transport

Domain Name System messages are transmitted either as datagrams (UDP) or through stream connection (TCP):

UDP usage: Server port 53 (decimal).

Messages carried by UDP are restricted to 512 bytes. Longer messages are truncated and the truncation (TC) bit is set in the header. Because UDP frames can be lost, a retransmission strategy is required.

TCP usage: Server port 53 (decimal).

In this case, the message is preceded by a 2-byte field indicating the total message frame length.

STD 3 – Host Requirements requires that:

* A Domain Name System resolver or server that is sending a non-zone-transfer query *must* send a UDP query first. If the answer section of the response is truncated and if the requester supports TCP, it tries the query again using TCP. UDP is preferred over TCP for queries because UDP queries have much lower overall processing cost, and the use of UDP is essential for a heavily loaded server. Truncation of messages is rarely a problem given the current contents of the Domain Name System database, because typically 15 response records can be accommodated in the datagram, but this might change as new record types continue to be added to the Domain Name System.
* TCP must be used for zone transfer activities because the 512-byte limit for a UDP datagram will always be inadequate for a zone transfer.
* Name servers must support both types of transport.

As IPv6 becomes more pervasive throughout the Internet community, some problems are forecasted as a result of mixed IPv4/IPv6 network segments. Primarily, if a resolver that can only use IPv4 is forwarded to across a network segment that supports only IPv6, the resolver and the name server will be unable to communicate. As a result, the hierarchical namespace becomes fragmented into two sets of segments: those that support IPv4 and IPv6, and those that support only IPv6. This impending issue has been named the Problem of Name Space Fragmentation, and documented in RFC 3901. In order to preserve namespace continuity, RFC 3901 recommends the following:

Every recursive name server should be either IPv4 only, or dual IPv4 and IPv6.

Every DNS zone should be served by at least one IPv4-reachable authoritative name server.

Additional suggestions about configuring IPv6 DNS servers are in RFC 4339.

##### Dynamic DNS (DDNS)

The Dynamic Domain Name System (DDNS) is a protocol that defines extensions to the Domain Name System to enable DNS servers to accept requests to add, update, and delete entries in the DNS database dynamically. Because DDNS offers a functional superset to existing DNS servers, a DDNS server can serve both static and dynamic domains at the same time, a welcome feature for migration and overall DNS design.

DDNS is currently available in a non-secure and a secure flavor, defined in RFC

2136 and RFC 3007, respectively. Rather than allowing any host to update its DNS records, the secure version of DDNS uses public key security and digital signatures to authenticate update requests from DDNS hosts.

Without client authentication, another host could impersonate an unsuspecting host by remapping the address entry for the unsuspecting host to that of its own. After the remapping occurs, important data, such as logon passwords and mail intended for the host would, unfortunately, be sent to the impersonating host instead.

See 12.2, “Dynamic Domain Name System” on page 453for more information about how DDNS works together with DHCP to perform seamless updates of reverse DNS mapping entries, and see 9.4, “DNS in IPv6” on page 367for more information about DNS with IPv6.

#### 12.1.14 DNS applications

Three common utilities for querying name servers are provided with many DNS implementations: **host**, **nslookup**, and **dig**. Though specifics details about each utility vary slightly by platform, most platforms provide a common set of options.

##### host

The **host** command obtains an IP address associated with a host name, or a host name associated with an IP address. The typical syntax for **host** command is: host [options] name [server]

Where:

**options** Valid options typically include:

**-c class** The query class. By default, this is IN (Internet), but other valid class names include CS (CSNET), CH (CHAOS), HS (Hesiod), and ANY (a wildcard encompassing all four classes).

**-r** Disables recursive processing.

|  |  |
| --- | --- |
|  | **-t type** The type of query required. This can be any of the standard resource record types (see Table 12-2 on page 438).  **-w** Instructs the **host** command to wait forever  for a reply. |
| **name** | The name of the host or the address to be resolved. |
| **server** | The name server to query. |

##### nslookup

The **nslookup** command enables you to locate information about network nodes, examine the contents of a name server database, and establish the accessibility of name servers. The typical syntax for the **nslookup** command is: nslookup [options] [host] [-nameserver] Where:

|  |  |
| --- | --- |
| **options** | These options vary widely by platform. Refer to the documentation for a specific implementation for information about what options are available. |
| **host** | The host name or IP address to be located. |
| **-nameserver** | The name server to which the query is to be directed. |

##### dig

**dig** stands for Domain Internet Groper, and enables you to exercise name servers, gather large volumes of domain name information, and execute simple domain name queries. The typical syntax for the **dig** command is: dig @server [options] [name] [type] [class] [queryopt]

Where:

|  |  |  |
| --- | --- | --- |
| **@server** | The DNS name server to be queried. | |
| **options** | Valid options typically include: | |
| **-b address** | The source IP address of the query-to address. |
|  | **-c class** | The query class. By default, this is IN  (Internet), but other valid class names |
|  | **-f filename** | include CS (CSNET), CH (CHAOS), and HS    (Hesiod).  Causes **dig** to operate in batch mode, and specifies the file from which the batch commands can be found. |

**-p port** Specifies that **dig** should send the query to a port other than well-known DNS port 53.

**-x address** Instructs **dig** to do a reverse lookup on the specified address.

**name** The name of the resource record to be looked up.

**type** The type of query required. This can be any of the standard resource record types (see Table 12-2 on page 438).

### 12.2 Dynamic Domain Name System

The Domain Name System described in 12.1, “Domain Name System (DNS)” on page 426 is a static implementation without recommendations with regard to security. In order to implement DNS dynamically, take advantage of DHCP, and still to be able to locate any specific host by means of a meaningful label (such as its host name), the following extensions to DNS are required:

A method for the host name to address a mapping entry for a client in the domain name server to be updated after the client has obtained an address from a DHCP server

A method for the reverse address to host name mapping to take place after the client obtains its address

Updates to the DNS to take effect immediately, without the need for intervention by an administrator

Updates to the DNS to be authenticated to prevent unauthorized hosts from accessing the network and to stop imposters from using an existing host name and remapping the address entry for the unsuspecting host to that of its own

A method for primary and secondary DNS servers to quickly forward and receive changes as entries are being updated dynamically by clients

However, implementation of a Dynamic Domain Name System (DDNS) can introduce problems if the environment is not secure. One method of security employed by DNS is the use of Secret Key Transaction Authentication (TSIG), defined in RFC 2845. This can be used to authenticate dynamic updates from clients, or authenticate responses coming from a recursive server. Additionally, these messages can now be protected for integrity and confidentiality through using TSIG over the Generic Security Service (GSS-TSIG). This extension, and the associated algorithms needed to implement GSS-TSIG, are defined in RFC 3645.

In addition to TSIG, and GSS-TSIG, several RFCs extended the functionality of DNS such that it incorporated additional security methods. These additions, defined in RFC 4033 and referred to as the DNS Security Extensions (DNSSEC), allow DNS to authenticate the origin of data as well as negative responses to DNS queries. However, they do not provide confidentiality, access control lists, or protection against denial-of-service-attacks. New resource records relating to security were added by RFCs 4034 and 4398, and include:

DNSKEY (public key)

DS (delegation signer)

RRSIG (resource record digital signature)

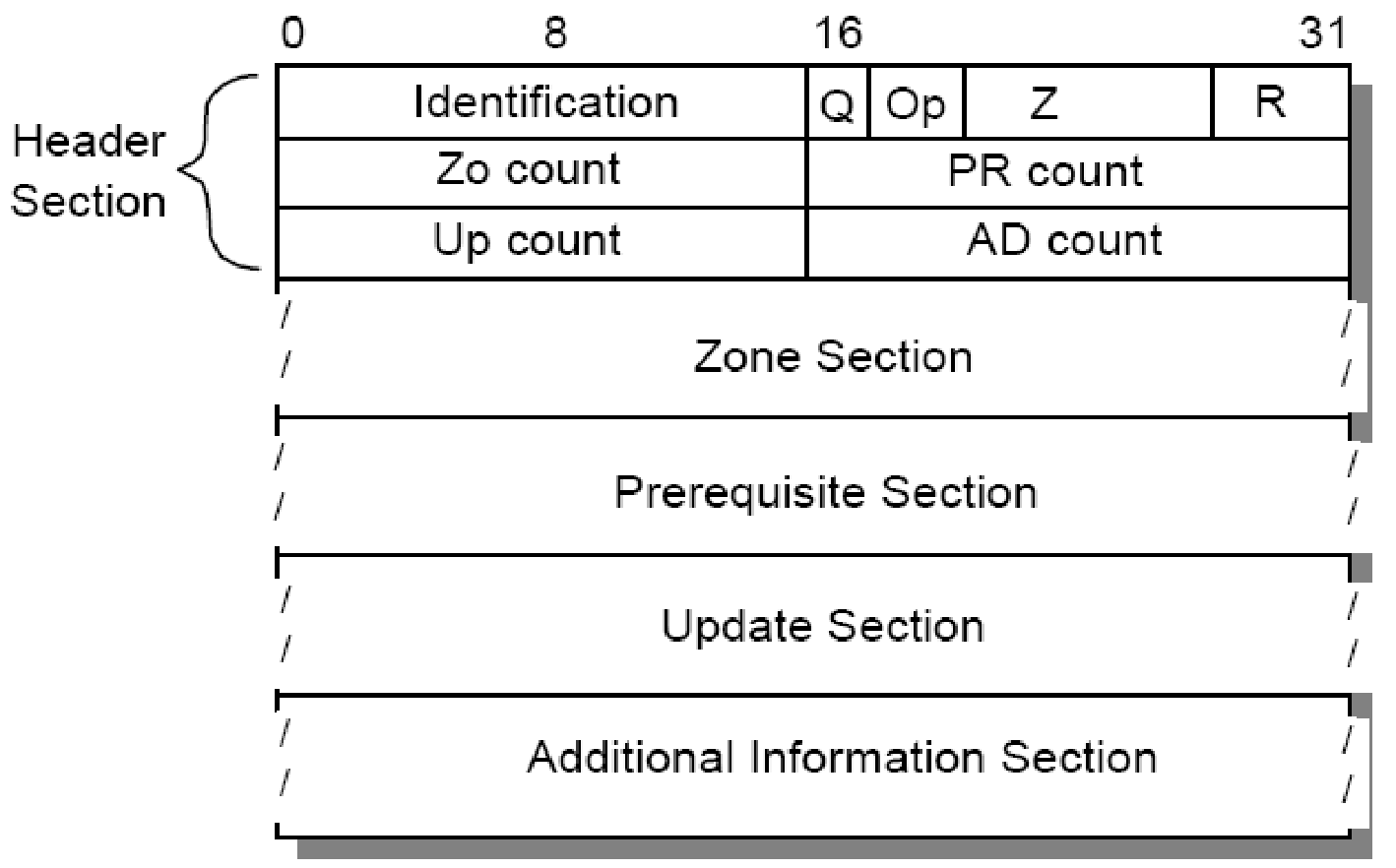
NSEC (authenticated denial of existence)

CERT (public key certificates)

Note that these RRs are also listed in Table 12-2 on page 438. Specific details about how the DNS protocol was modified to take advantage of these additions is in RFC 4035.

#### 12.2.1 Dynamic updates in the DDNS

The DNS message format (shown in Figure 12-5 on page 440) was designed for the querying of a static DNS database. RFC 2136 defines a modified DNS message for updates, called the UPDATE DNS message, illustrated in Figure 12-13 on page 455. This message adds or deletes resource records in the DNS, and allows updates to take effect without the DNS having to be reloaded.



*Figure 12-13 DDNS UPDATE message format*

The header section is always present and has a fixed length of 12 bytes. The other sections are of variable length. They are:

**Identification** A 16-bit identifier assigned by the program. This identifier is copied in the corresponding reply from the name server and can be used for differentiation when multiple queries/updates are outstanding at the same time.

1. Flag identifying an update request (0) or a response (1).

**Op** Opcode. The value 5 indicates an UPDATE message. **z** 7-bit field set to 0 and reserved for future use.

1. Response code (undefined in update requests). Possible

values are:

* 1. No error.
  2. Format error. The server was unable to interpret the message.
  3. Server failure. The message was not processed due to a problem with the server. **3** Name error. A name specified does not exist.

* + 1. Not implemented. The type of message specified in Opcode is not supported by this server.
    2. Refused. The server refuses to perform the UPDATE requested for security or policy reasons.
    3. Name error. A name exists when it should not.
    4. RRset error. A resource record set exists when it should not.
    5. RRset error. A resource record set specified does not exist.
    6. Zone Authority error. The server is not authoritative for the zone specified.
    7. Zone error. A name specified in the Prerequisite or Update sections is not in the zone specified.

|  |  |
| --- | --- |
| **ZO count** | The number of RRs in the Zone section. |
| **PR count** | The number of RRs in the Prerequisite section. |
| **UP count** | The number of RRs in the Update section. |
| **AD count** | The number of RRs in the Additional information section. |
| **Zone section** | This section is used to indicate the zone of the records that are to be updated. As all records to be updated must belong to the same zone, the zone section has a single entry specifying the zone name, zone type (which must be SOA), and zone class. |

##### Prerequisite section

This section contains RRs or RRsets that either must, or must not, exist, depending on the type of update.

**Update section** This section contains the RRs, RRsets, or both that are to be added to or deleted from the zone.

##### Additional information section

This section can be used to pass additional RRs that relate to the update operation in process.

For further information about the UPDATE message format, refer to RFC 2136.

#### 12.2.2 Incremental zone transfers in DDNS

RFC 1995 introduces the IXFR DNS message type, which allows incremental transfers of DNS zone data between primary and secondary DNS servers. In other words, when an update has been made to the zone data, only the change has to be copied to the other DNS servers that maintain a copy of the zone data, rather than the whole DNS database (as is the case with the AXFR DNS message type).

The format of an IXFR query is exactly that of a normal DNS query, but with a query type of IXFR. The response’s answer section, however, is made up of *difference sequences*.

Each list difference sequences is preceded by the server’s current version of the SOA and represents one update to the zone. Similarly, each difference sequence is preceded by an SOA version (indicating in which versions correspond to each change, and the difference sequences are ordered oldest to newest. Upon receiving this message, a server can update its zone by tracking the version history listed in the IXFR answer section.

For example, assume a server has the following zone:

|  |  |
| --- | --- |
| MYZONE.MYDIV.MYCORP | IN SOA MYHOST.MYDIV.MYCORP (  1 600 600 3600000 614800)  IN NS MYHOST.MYDIV.MYCORP |
| MYHOST.MYDIV.MYCORP | IN A 10.1.2.3 |
| OTHERHOST.MYDIV.MYCORP | IN A 10.2.3.4 |

Otherhost.mydiv.mycorp is removed, and in version 2, thishost.mydiv.mycorp is added, leaving the zone as:

MYZONE.MYDIV.MYCORP IN SOA MYHOST.MYDIV.MYCORP (

2 600 600 3600000 614800)

IN NS MYHOST.MYDIV.MYCORP

MYHOST.MYDIV.MYCORP IN A 10.1.2.3

THISHOST.MYDIV.MYCORP IN A 10.2.3.5

If the server receives an IXFR query, it sends back the following answer section:

|  |  |
| --- | --- |
| MYZONE.MYDIV.MYCORP | IN SOA serial=2 |
| MYZONE.MYDIV.MYCORP | IN SOA serial=1 |
| OTHERHOST.MYDIV.MYCORP | IN A 10.1.2.4 |
| MYZONE.MYDIV.MYCORP | IN SOA serial=2 |
| THISHOST.MYDIV.MYCORP | IN A 10.2.3.5 |
| MYZONE.MYDIV.MYCORP | IN SOA serial=2 |

**Note:** If a server received an IXFR query, but incremental zone transfers are not available, it will send back the entire zone in the reply.

#### 12.2.3 Prompt notification of zone transfer

RFC 1996 introduces the NOTIFY DNS message type, which is used by a master server to inform subordinate servers that an update has taken place and that they should initiate a query to discover the new data. The NOTIFY message uses the DNS message format, but only a subset of the available fields (unused

fields are filled with binary zeros). The message is similar to a QUERY message, and can contain the name of the RRs that have been updated. Upon receipt of a NOTIFY message, the subordinate returns a response. The response contains no useful information, and only serves to alert the master server of receipt of the NOTIFY. Based on the RRs contained in the notify, subordinate servers might then send an update query to the server to obtain the new changes.

### 12.3 Network Information System (NIS)

The Network Information System (NIS) is not an Internet standard. It was developed by Sun Microsystems, Inc. It was originally known as the Yellow Pages (YP) and many implementations continue to use this name.

NIS is a distributed database system that allows the sharing of system information in UNIX-based environment. Examples of system information that can be shared include the /etc/passwd, /etc/group, and /etc/hosts files. NIS has the following advantages:

Provides a consistent user ID and group ID name space across a large number of systems

Reduces the time and effort by users in managing their user IDs, group IDs, and NFS file system ownerships

Reduces the time and effort by system administrators in managing user IDs, group IDs, and NFS ownerships

NIS is built on the RPC, and employs the client/server model. Most NIS implementations use UDP. However, because it uses RPC, it is also possible for it to be implemented over TCP. A NIS domain is a collection of systems consisting of:

**NIS master server** Maintains *maps*, or databases, containing the

system information, such as passwords and host names. These are also referred to as Database Maps (DBMs).

**NIS subordinate server(s)** Can be defined to offload the processing from the master NIS server or when the NIS master server is unavailable.

**NIS client(s)** The remaining systems that are served by the

NIS servers.

The NIS clients do not maintain NIS maps; they query NIS servers for system information. Any changes to an NIS map is done only to the NIS master server (through RPC). The master server then propagates the changes to the NIS subordinate servers.

Note that the speed of a network determines the performance and availability of the NIS maps. When using NIS, the number of subordinate servers should be tuned in order to achieve these goals.

Because NIS is not standardized by the IETF, implementations vary by platform.

However, most platforms make available the following common NIS commands:

|  |  |
| --- | --- |
| **makedbm** | Generate a DBM file from an input file. |
| **ypcat** | Display the contents of a DBM file. |
| **ypinit** | Set up an NIS master or subordinate server. |
| **ypmake** | Performs the same function as **makedbm**, but provides the option to push the resulting DBMs to subordinate servers. |
| **ypmatch** | Prints the values associated with one or more keys in a DBM. |
| **yppasswd** | Change a login password stored in a DBM. |
| **yppush** | Pushes DBMs to subordinate servers. |
| **ypwhich** | Indicates what NIS server a client is using. |
| **ypxfr** | Pulls a DBM from the master server. |

### 12.4 Lightweight Directory Access Protocol (LDAP)

When implementing a Distributed Computing Environment (DCE), directory services are automatically included because they are an intrinsic part of the DCE architecture. However, though widely used, implementation of a DCE is not a practical solution for every company needing directory services because it is an “all-or-nothing” architecture. As such, if the other services provided by a DCE are not required, or if implementation of the DCE model is not feasible (for example, if it is not feasible to install the client software on every workstation within the network), other directory service alternatives must be identified.

One such alternative is the Lightweight Directory Access Protocol (LDAP), which is an open industry standard that has evolved to meet these needs. LDAP defines a standard method for accessing and updating information in a directory, and is gaining wide acceptance as the directory access method of the Internet. It is supported by a growing number of software vendors and is being incorporated into a growing number of applications.

For further information about LDAP, refer to the IBM Redbook *Understanding*  *LDAP - Design and Implementation*, SG24-4986.

#### 12.4.1 LDAP: Lightweight access to X.500

The OSI directory standard, X.500, specifies that communication between the directory client and the directory server uses the Directory Access Protocol

(DAP). However, as an application layer protocol, DAP requires the entire OSI protocol stack to operate, which requires more resources than are available in many small environments. Therefore, an interface to an X.500 directory server using a less resource-intensive or lightweight protocol was desired.

LDAP was developed as a *lightweight* alternative to DAP, because it requires the more popular TCP/IP protocol stack rather than the OSI protocol stack. LDAP also simplifies some X.500 operations and omits some esoteric features. Two precursors to LDAP appeared as RFCs issued by the IETF, RFC 1202 – Directory Assistance Service and RFC 1249 – DIXIE Protocol Specification.

These were both informational RFCs which were not proposed as standards. The directory assistance service (DAS) defined a method by which a directory client communicates to a proxy on an OSI-capable host, which issues X.500 requests on the client's behalf. DIXIE is similar to DAS, but provides a more direct translation of the DAP.

The first version of LDAP was defined in RFC 1487 – X.500 Lightweight Access, which was replaced by RFC 1777 – Lightweight Directory Access Protocol. Much of the work on DIXIE and LDAP was carried out at the University of Michigan, which provides reference implementations of LDAP and maintains LDAP-related Web pages and mailing lists. Since then, LDAPv2 has been replaced by LDAP Version 3. LDAPv3 is summarized in RFC 4510, but the technical specifications are divided into multiple subsequent RFCs listed in Table 12-4.

*Table 12-4 LDAP-related RFCs*

|  |  |
| --- | --- |
| **RFC number** | **Content** |
| 4510 | Technical Specification Road Map |
| 4511 | The Protocol |
| 4512 | Directory Information Models |
| 4513 | Authentication Methods and Security Mechanisms |
| 4514 | String Representation of Distinguished Names |
| 4515 | String Representation of Search Filters |
| 4516 | Uniform Resource Locator |
| 4517 | Syntaxes and Matching Rules |
| 4518 | Internationalized String Preparation |
| **RFC number** | **Content** |
| 4519 | Schema for User Applications |
| 4520 | Internet Assigned Numbers Authority (IANA) Considerations for  LDAP |
| 4521 | Considerations for LDAP |
| 4522 | The Binary Encoding Option |
| 4523 | Schema Definitions for X.509 Certificates |
| 4524 | COSINE/ LDAP X.500 Schema |
| 4525 | Modify-Increment Extension |
| 4526 | Absolute True and False Filters |
| 4527 | Read Entry Controls |
| 4528 | Assertion Control |
| 4529 | Requesting Attributes by Object Class in LDAP |
| 4530 | entryUUID Operational Attribute |
| 4531 | Turn Operation |
| 4532 | “Who Am I” Operation |
| 4533 | Content Synchronization Operation |

Though an application program interface (API) for previous versions of LDAP was limited to specifications in RFC 1823, the LDAPv3 provides both a C API and a Java™ Naming and Directory Interface (JNDI).

#### 12.4.2 The LDAP directory server

LDAP defines a communication protocol. That is, it defines the transport and format of messages used by a client to access data in an X.500-like directory. LDAP does not define the directory service itself. An application client program initiates an LDAP message by calling an LDAP API. But an X.500 directory server does not understand LDAP messages. In fact, the LDAP client and X.500 server even use different communication protocols (TCP/IP versus OSI). The LDAP client actually communicates with a gateway process (also called a proxy or front end) that forwards requests to the X.500 directory server (see Figure 12-14 on page 462), known as an LDAP server, which fulfils requests from the LDAP client. It does this by becoming a client of the X.500 server. The LDAP server must communicate using both TCP/IP (with the client) and OSI (with the X.500 server).

LDAP

Client

LDAP

Server

X.500

Server

Directory

TCP/IP

OSI

*Figure 12-14 LDAP server acting as a gateway to an X.500 directory server*

As the use of LDAP grew and its benefits became apparent, people who did not have X.500 servers or the environments to support them wanted to build directories that could be accessed by LDAP clients. This requires that the LDAP server store and access the directory itself instead of only acting as a gateway to X.500 servers (see Figure 12-15). This eliminates any need for the OSI protocol stack but, of course, makes the LDAP server much more complicated, because it must store and retrieve directory entries. These LDAP servers are often called stand-alone LDAP servers because they do not depend on an X.500 directory server. Because LDAP does not support all X.500 capabilities, a stand-alone LDAP server only needs to support the capabilities required by LDAP.

*Figure 12-15 Stand-alone LDAP server*

LDAP

Client

LDAP

Server

Directory

TCP/IP

The concept of the LDAP server being able to provide access to local directories supporting the X.500 model, rather than acting only as a gateway to an X.500 server, is discussed in RFC 4511 (see Table 12-4 on page 460). From the client's

point of view, any server that implements the LDAP protocol is an LDAP directory server, whether the server actually implements the directory or is a gateway to an X.500 server. The directory that is accessed can be called an LDAP directory, whether the directory is implemented by a stand-alone LDAP server or by an X.500 server.

#### 12.4.3 Overview of LDAP architecture

LDAP defines the content and format of messages exchanged between an LDAP client and an LDAP server. The messages specify the operations requested by the client (search, modify, delete, and so on), the responses from the server, and the format of data carried in the messages. LDAP messages are carried over TCP/IP, a connection-oriented protocol, so there are also operations to establish and disconnect a session between the client and server.

The general interaction between an LDAP client and an LDAP server takes the following form:

1. The client establishes a session with an LDAP server. This is known as *binding* to the server. The client specifies the host name or IP address and TCP/IP port number where the LDAP server is listening. The client can provide a user name and a password to properly authenticate with the server, or the client can establish an anonymous session with default access rights. The client and server can also establish a session that uses stronger security methods, such as encryption of data (see 12.4.5, “LDAP security” on page 471).
2. The client then performs operations on directory data. LDAP offers both read and update capabilities. This allows directory information to be managed as well as queried. LDAP supports searching the directory for data meeting arbitrary user-specified criteria. Searching is the most common operation in LDAP. A user can specify what part of the directory to search and what information to return. A search filter that uses Boolean conditions specifies which directory data matches the search.
3. When the client has finished making requests, it closes the session with the server. This is also known as *unbinding*.

Because LDAP was originally intended as a lightweight alternative to DAP for accessing X.500 directories, the LDAP server follows an X.500 model. The directory stores and organizes data structures known as entries. A directory entry usually describes an object such as a person, a printer, a server, and so on. Each entry has a name called a distinguished name (DN) that uniquely identifies it. The DN consists of a sequence of parts called relative distinguished names (RDNs), much like a file name can consist of a path of directory names. The entries can be arranged into a hierarchical tree-like structure based on their distinguished names. This tree of directory entries is called the directory information tree (DIT).

LDAP defines operations for accessing and modifying directory entries, such as:

Searching for entries meeting user-specified criteria

Adding an entry

Deleting an entry

Modifying an entry

Modifying the distinguished name or relative distinguished name of an entry (move)

Comparing an entry

#### 12.4.4 LDAP models

LDAP can be better understood by considering the four models upon which it is based:

|  |  |
| --- | --- |
| **Information** | Describes the structure of information stored in an LDAP directory. |
| **Naming** | Describes how information in an LDAP directory is organized and identified. |
| **Functional** | Describes the operations that can be performed on the information stored in an LDAP directory. |
| **Security** | Describes how the information in an LDAP directory can be protected from unauthorized access. |

The following sections discuss the first three LDAP models. We describe LDAP security in 12.4.5, “LDAP security” on page 471.

##### The information model

The basic unit of information stored in the directory is an entry, which represents an object of interest in the real world such as a person, server, or organization. Each entry contains one or more attributes that describe the entry. Each attribute has a type and one or more values. For example, the directory entry for a person might have an attribute called telephoneNumber. The syntax of the telephoneNumber attribute specifies that a telephone number must be a string of numbers that can contain spaces and hyphens. The value of the attribute is the person's telephone number, such as 123-456-7890 (a person might have multiple telephone numbers, in which case this attribute would have multiple values).

In addition to defining what data can be stored as the value of an attribute, an attribute syntax also defines how those values behave during searches and other directory operations. This is done using syntax and matching rules. The attribute telephoneNumber, for example, might have a syntax that specifies:

Lexicographic ordering.

Case, spaces, and dashes are ignored during the comparisons. Values must be character strings.

For example, using the correct definitions, the telephone numbers

123-456-7890, 123456-7890, and 1234567890 are considered to be the same. A few of the common syntaxes and matching rules, defined in RFC 4517, are listed in Table 12-5.

*Table 12-5 Examples of LDAP syntaxes*

|  |  |
| --- | --- |
| **Syntaxes and matching rules** | **Description** |
| Bit String | A sequence of binary digits |
| Postal Address | A sequence of strings that form an address in a physical mail system |
| caseExactMatch | A matching rule requiring that string comparisons are case-sensitive |
| caseIgnoreMatch | A matching rule that does not require case-sensitive comparisons |

Table 12-6 lists some common attributes defined by RFC 4519. Some attributes have alias names that can be used wherever the full attribute name is used.

*Table 12-6 Examples of LDAP syntaxes*

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute, alias** | **Syntax** | **Description** | **Example** |
| commonName, cn | cis | Common name of an entry | John Smith |
| surname, sn | cis | A person’s last name | Smith |
| initials | cis | A person’s initials | JS |
| telephoneNumber | tel | A person’s telephone number | 123-456-7890 |

An object class is a general description, sometimes called a template, of an object type, as opposed to the description of a specific object of that type. For example, the object class *person* has a surname attribute, while the object describing John Smith has a surname attribute with the value *Smith*. The object classes that a directory server can store and the attributes they contain are described by a *schema*. A schema defines where object classes are allowed in the directory, which attributes they must contain, which attributes are optional, and the syntax of each attribute. For example, a schema might define a *person* object class that requires that a person has a character-string surname attribute, can optionally have a number-string telephoneNumber attribute, and so on. Schema-checking ensures that all required attributes for an entry are present before an entry is stored. Schemas also define the inheritance and subclassing of objects, and where in the DIT structure (hierarchy) objects can appear.

Though an implementation can define any schema to meet its needs, RFC 4519 defines a few standard schemas. Table 12-7 lists a few of the common schema (object classes and their required attributes). In many cases, an entry can consist of more than one object class.

*Table 12-7 Examples of object classes and required attributes*

|  |  |  |
| --- | --- | --- |
| **Object class** | **Description** | **Attributes** |
| country | Defines a country | Required: countryCode Optional: searchGuide description |
| locality | Defines a place in the physical world | Required:  None  Optional: street seeAlso searchGuide description |
| person | Defines a person | Required:  surname commonName  Optional:  userPassword telephoneNumber seeAlso description |

Because servers can define their own schema, LDAP includes the functionality of allowing a client to query a server for the contents of the supported schema.

##### The naming model

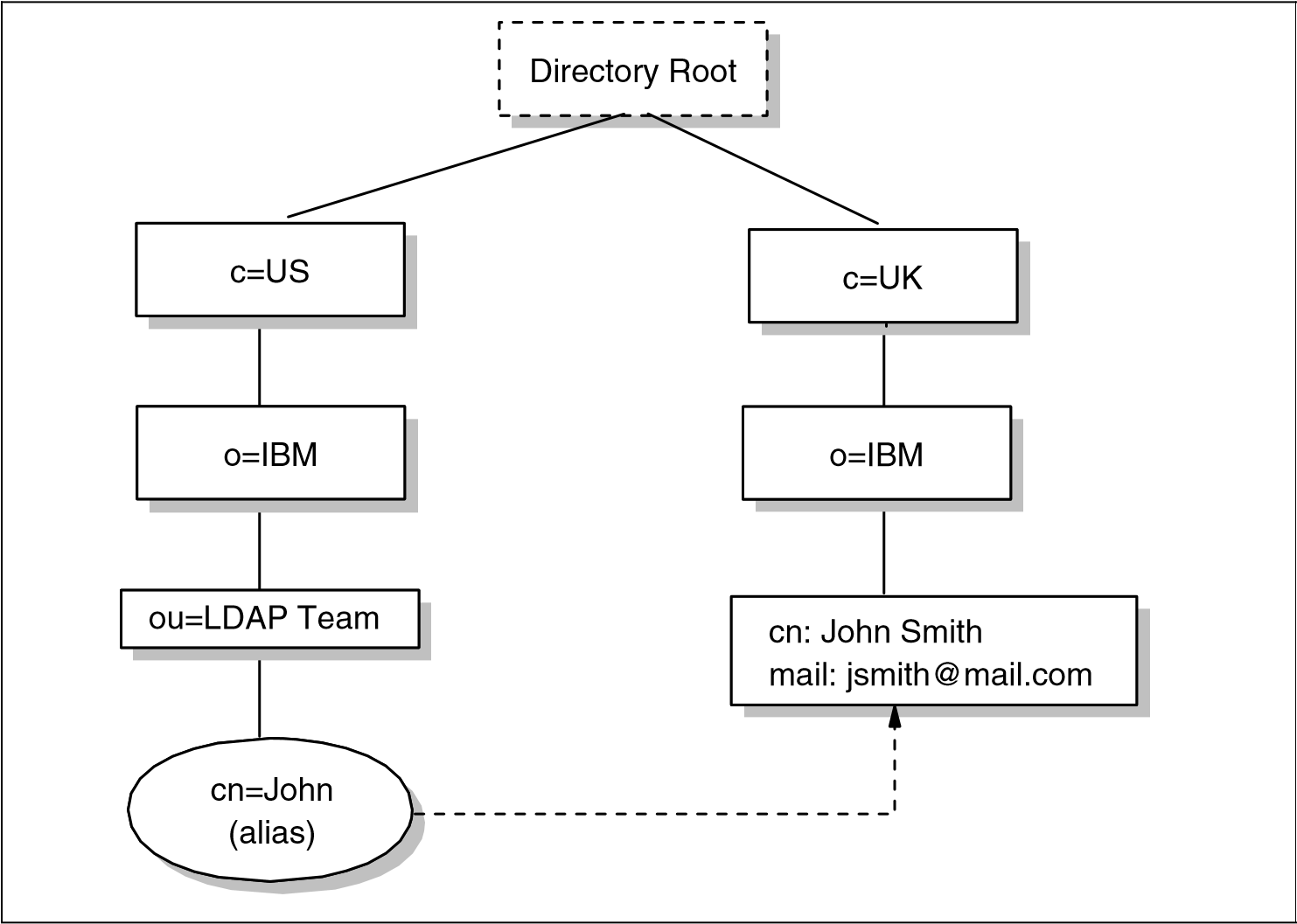
The LDAP naming model defines how entries are identified and organized.

Entries are organized in a tree-like structure called the directory information tree

(DIT). Entries are arranged within the DIT based on their distinguished name (DN). A DN is a unique name that unambiguously identifies a single entry. DNs are made up of a sequence of relative distinguished names (RDNs). Each RDN in a DN corresponds to a branch in the DIT leading from the root of the DIT to the directory entry.

Each RDN is derived from the attributes of the directory entry. In the simple and common case, an RDN has the form <attribute-name>=<value>. A DN is composed of a sequence of RDNs separated by commas. These relationships are defined in RFC 4514.

An example of a DIT is shown in Figure 12-16. The example is very simple, but can be used to illustrate some basic concepts. Each box represents a directory entry. The root directory entry is conceptual and does not actually exist. Attributes are listed inside each entry. The list of attributes shown is not complete. For example, the entry for the country UK (c=UK) could have an attribute called description with the value United Kingdom.



*Figure 12-16 Example of a directory information tree (DIT)*

It is usual to follow either a geographical or an organizational scheme to position entries in the DIT. For example, entries that represent countries would be at the top of the DIT. Below the countries would be national organizations, states, and provinces, and so on. Below this level, entries might represent people within those organizations or further subdivisions of the organization. The lowest layers of the DIT entries can represent any object, such as people, printers, application servers, and so on. The depth or breadth of the DIT is not restricted and can be designed to suit application requirements.

Entries are named according to their position in the DIT. The directory entry in the lower-right corner of Figure 12-16 on page 467 has the DN cn=John Smith,o=IBM,c=UK.

**Note:** DNs read from leaf-to-root, as opposed to names in a file system directory, which usually read from root-to-leaf.

The DN is made up of a sequence of RDNs. Each RDN is constructed from an attribute (or attributes) of the entry it names. For example, the DN cn=John Smith,o=IBM,c=UK is constructed by adding the RDN cn=John Smith to the DN of the ancestor entry o=IBM,c=UK.

The DIT is described as being tree-like, implying that it is not a tree. This is because of aliases. Aliases allow the tree structure to be circumvented. This can be useful if an entry belongs to more than one organization or if a commonly used DN is too complex. Another common use of aliases is when entries are moved within the DIT and you want access to continue to work as before. In Figure 12-16 on page 467, cn=John,ou=LDAP Team,o=IBM,c=US is an alias for cn=John Smith,o=IBM,c=UK.

Because an LDAP directory can be distributed, an individual LDAP server might not store the entire DIT. Instead, it might store the entries for a particular department but not the entries for the ancestors of the department. For example, a server might store the entries for the Accounting department at Yredbookscorp. The highest node in the DIT stored by the server would be ou=Accounting,o=Yredbookscorp,c=US. The server would store entries ou=Accounting,o=Yredbookscorp,c=US but not for c=US or for o=Yredbookscorp,c=US. The highest entry stored by a server is called a *suffix*. Each entry stored by the server ends with this suffix, so in this case, the suffix is the entire ou=Accounting,o=Yredbookscorp,c=US.

A single server can support multiple suffixes. For example, in addition to storing information about the Accounting department, the same server can store information about the Sales department at MyCorp. The server then has the suffixes ou=Accounting,o=Yredbookscorp,c=US and ou=Sales,o=MyCorp,c=US. Because a server might not store the entire DIT, servers need to be linked together in some way in order to form a distributed directory that contains the entire DIT. This is accomplished with *referrals*. A referral acts as a pointer to an entry on another LDAP server where requested information is stored. A referral is an entry of objectClass *referral*. It has an attribute, *ref*, whose value is the LDAP URL of the referred entry on another LDAP server. See 12.4.6, “LDAP URLs” on page 474 for further information. Referrals allow a DIT to be partitioned and distributed across multiple servers. Portions of the DIT can also be replicated. This can improve performance and availability.

**Note:** When an application uses LDAP to request directory information from a server, but the server only has a referral for that information, the LDAP URL for that information is passed to the client. It is then the responsibility of that client to contact the new server to obtain the information. This is unlike the standard mechanisms of both DCE and X.500, where a directory server, if it does not contain the requested information locally, will always obtain the information from another server and pass it back to the client.

##### The functional model

LDAP defines operations for accessing and modifying directory entries. LDAP operations can be divided into the following three categories:

**Query** Includes the search and compare operations used to

retrieve information from a directory.

**Update** Includes the add, delete, modify, modify RDN, and

unsolicited notification operations used to update stored information in a directory. These operations will normally be carried out by an administrator.

**Authentication** Includes the bind, unbind, abandon, and startTLS operations used to connect and disconnect to and from an LDAP server, establish access rights, and protect information. For further information, see 12.4.5, “LDAP security” on page 471.

###### The search operation

The most common operation is the search. This operation is very flexible and therefore has some of the most complex options. The search operation allows a client to request that an LDAP server search through some portion of the DIT for information meeting user-specified criteria in order to read and list the results.

The search can be very general or very specific. The search operation allows the specification of the starting point within the DIT, how deep within the DIT to search, the attributes an entry must have to be considered a match, and the attributes to return for matched entries.

Some example searches expressed informally are:

Find the postal address for cn=John Smith,o=IBM,c=UK.

Find all the entries that are children of the entry ou=ITSO,o=IBM,c=US.

Find the e-mail address and phone number of anyone in an organization whose last name contains the characters “miller” and who also has a fax number.

To perform a search, the following parameters must be specified:

**Base** A DN that defines the starting point, called the base

object, of the search. The base object is a node within the DIT.

**Scope** Specifies how deep within the DIT to search from the base object. There are three choices: **baseObject** Only the base object is examined.

**singleLevel** Only the immediate children of the base object are examined; the base object itself is not examined.

**wholeSubtree** The base object and all of its

descendants are examined.

**Alias dereferencing** Specifies if aliases are dereferenced. That is, the actual object of interest, pointed to by an alias entry, is examined. Not dereferencing aliases allows the alias entries themselves to be examined. This parameter must be one of the following:

**neverDerefAliases**

Do not deference aliases.

derefInSearching

Dereference aliases only when searching subordinates of the base object.

derefFindingBaseObj

Dereference aliases only when searching for the base object, but not when searching subordinates of the base object.

**derefAlways** Always dereference aliases.

**Size Limit** The maximum number of entries that should be returned as a result of the search.

**Time Limit** The maximum number of seconds allowed to perform the

search. Specifying zero indicates that there is no time limit.

**Types Only** This parameter has two possible values:

|  |  |
| --- | --- |
| **TRUE** | Only attribute descriptions are returned. |
| **FALSE** | Attribute descriptions and values are returned. |

**Search filter** Specifies the criteria an entry must match to be returned from a search. The search filter is a Boolean combination of attribute value assertions. An attribute value assertion tests the value of an attribute for equality, less than or equal, and so on.

**Attributes to return** Specifies which attributes to retrieve from entries that match the search criteria. Because an entry can have many attributes, this allows the user to only see the attributes in which they are interested.

#### 12.4.5 LDAP security

Security is of great importance in the networked world of computers, and this is true for LDAP as well. When sending data over insecure networks, internally or externally, sensitive information might need to be protected during transportation. There is also a need to know who is requesting the information and who is sending it. This is especially important when it comes to the update operations on a directory. RFC 4513 discusses the authentication methods and security mechanisms available in LDAPv3, which can be divided into the following sections:

|  |  |
| --- | --- |
| **Authentication** | Assurance that the opposite party (machine or person) really is who he/she/it claims to be. |
| **Integrity** | Assurance that the information that arrives is really the same as what was sent. |
| **Confidentiality** | Protection against information disclosure, by means of data encryption, to those who are not intended to receive it. |
| **Authorization** | Assurance that a party is really allowed to do what it is requesting to do, usually checked after user  authentication. Authorization is achieved by assigning access controls, such as read, write, or delete, for user IDs or common names to the resources being accessed. Because these attributes are usually platform-specific,  LDAP does not provide specific controls. Instead, it has |

built-in mechanisms to allow the use of the platform-specific controls.

Because the use of authorization controls is platform-specific, the following sections describe only the authentication, integrity, and confidentiality. There are several mechanisms that can be used for this purpose; the most important ones are discussed here. These are:

No authentication

Basic authentication

Simple Authentication and Security Layer (SASL)

Transport Layer Security (TLS)

The security mechanism to be used in LDAP is negotiated when the connection between the client and the server is established.

##### No authentication

No authentication should only be used when data security is not an issue and when no special access control permissions are involved. This might be the case, for example, when your directory is an address book browsable by anybody. No authentication is assumed when you leave the password and DN field empty in the bind API call. The LDAP server then automatically assumes an anonymous user session and grants access with the appropriate access controls defined for this kind of access (not to be confused with the SASL anonymous user discussed in “Simple Authentication and Security Layer (SASL)”).

##### Basic authentication

Basic authentication is also used in several other Web-related protocols, such as HTTP. When using basic authentication with LDAP, the client identifies itself to the server by means of a DN and a password, which are sent in the clear over the network (some implementations might use Base64 encoding instead). The server considers the client authenticated if the DN and password sent by the client matches the password for that DN stored in the directory. Base64 encoding is defined in the Multipurpose Internet Mail Extensions, or MIME (see 15.3, “Multipurpose Internet Mail Extensions (MIME)” on page 571). Base64 is a relatively simple encryption, and it is not hard to break after the data has been captured in the network.

##### Simple Authentication and Security Layer (SASL)

SASL is a framework for adding additional authentication mechanisms to connection-oriented protocols, and is defined in RFC 4422. It has been added to LDAPv3 to overcome the authentication shortcomings of Version 2. SASL was originally devised to add stronger authentication to the IMAP protocol (see 15.5, “Internet Message Access Protocol (IMAP4)” on page 591), but has since evolved into a more general system for mediating between protocols and authentication systems.

In SASL, connection protocols, such as LDAP, IMAP, and so on, are represented by profiles; each profile is considered a protocol extension that allows the protocol and SASL to work together. A complete list of SASL profiles can be obtained from the Information Sciences Institute (ISI). Among these are IMAP, SMTP, POP, and LDAP. Each protocol that intends to use SASL needs to be extended with a command to identify an authentication mechanism and to carry out an authentication exchange. Optionally, a security layer can be negotiated to encrypt the data after authentication and ensure confidentiality. LDAPv3 includes such a command (**ldap\_sasl\_bind()** or **ldap\_sasl\_bind\_s()**). The key parameters that influence the security method used are:

|  |  |
| --- | --- |
| **dn** | This is the distinguished name of the entry which is to bind. This can be thought of as the user ID in a normal user ID and password authentication. |
| **mechanism** | This is the name of the security method to use. Valid security mechanisms are, currently:  **OTP** The One Time Password mechanism  (defined in RFC 2444).  **GSSAPI** The Generic Security Services Application Program Interface (defined in RFC 2743).  **CRAM-MD5** The Challenge/Response Authentication Mechanism (defined in RFC 2195), based on the HMAC-MD5 MAC algorithm.  **DIGEST-MD5** An HTTP Digest-compatible CRAM based on the HMAC -MD5 MAC algorithm.  **EXTERNAL** An external mechanism. Usually this is TLS, discussed in “Transport Layer Security (TLS)” on page 474.  **ANONYMOUS** Unauthenticated access. |
| **credentials** | This contains the arbitrary data that identifies the DN. The format and content of the parameter depends on the mechanism chosen. If it is, for example, the ANONYMOUS  mechanism, it can be an arbitrary string or an e-mail address that identifies the user. |

SASL provides a high-level framework that lets the involved parties decide on the particular security mechanism to use. The SASL security mechanism negotiation between client and server is done in the clear. After the client and the server agree on a common mechanism, the connection is secure against modifying the authentication identities. However, an attacker might try to eavesdrop the mechanism negotiation and cause a party to use the least secure mechanism. In order to prevent this from happening, configure clients and servers to use a minimum security mechanism, provided they support such a configuration option. As stated earlier, SSL and its successor, TLS, are the mechanisms commonly used in SASL for LDAP. For details about these protocols, refer to 22.7, “Secure Sockets Layer (SSL)” on page 854.

Because no data encryption method was specified in LDAPv2, some vendors added their own SSL calls to the LDAP API. A potential drawback of such an approach is that the API calls might not be compatible among different vendor implementations. The use of SASL, as specified in LDAPv3, assures compatibility, although it is likely that vendor products will support only a subset of the possible range of mechanisms (possibly only SSL).

##### Transport Layer Security (TLS)

Transport Layer Security (TLS) is available through the SASL EXTERNAL method, described earlier. An LDAP client can opt to secure a session using TLS at any point during a transaction with an LDAP server, except when:

The session is already TLS protected.

A multi-stage SASL negotiation is in progress.

The client is awaiting a response from an operation request.

To request that TLS be set up on the session, the client sends a *StartTLS* message to the server. This enables the client and server to exchange certificates. RFC 4513 requires that, in addition to this, the client verifies the server’s identity using the DNS or IP address presented in the server’s certificate. This prevents a client’s attempt to connect to a server from being intercepted by malicious user, who might then stage a *man-in-the-middle* attack. After this has occurred, the client and server can negotiate a ciphersuite.

#### 12.4.6 LDAP URLs

Because LDAP has become an important protocol on the Internet, a URL format for LDAP resources has been defined in RFC 4516. LDAP URLs begin with ldap:// or ldaps://, if the LDAP server communicates using SSL. LDAP URLs can simply name an LDAP server, or can specify a complex directory search.

Some examples help make the format of LDAP URLs clear. The following example refers to the LDAP server on the host ldpserv.mydiv.mycorp.com (using the well-known port 389):

ldap://ldpserv.mydiv.mycorp.com/

Additionally, search options can be specified in the URL. The following example retrieves all the attributes for the DN ou=Accounting,c=US from the LDAP server on host ldpserv.mydiv.mycorp.com. In this case, nonstandard port 4389 is explicitly specified here as an example. ldap://ldpserv.mydiv.mycorp.com:4389/ou=Accounting,c=US

The following example retrieves all the attributes for the DN cn=JohnSmith,ou=Sales,o=myCorp,c=US. Note that some characters are considered unsafe in URLs because they can be removed or treated as delimiters by some programs. Unsafe characters such as space, comma, brackets, and so forth need to be represented by their hexadecimal value preceded by the percent sign: ldap://ldpserv.mydiv.mycorp.com/cn=John%20Smith,o=myCorp,c=US

In this example, %20 is a space. More information about unsafe characters and URLs in general are in RFC 4516.

In addition to options, the URL can specify what values attributes are to be returned using the ? symbol. For example, assume we want to find the U.S. address of myCorp. We use the following URL: ldap://ldpserv.mydiv.mycorp.com:4389/o=myCorp,c=US?postalAddress

#### 12.4.7 LDAP and DCE

DCE has its own Cell Directory Service, or CDS (see 13.3.1, “DCE directory service” on page 498). If applications never access resources outside of their DCE cell, only CDS is required. However, if an application needs to communicate with resources in other DCE cells, the Global Directory Agent (GDA) is required. The GDA accesses a global (that is, non-CDS) directory where the names of

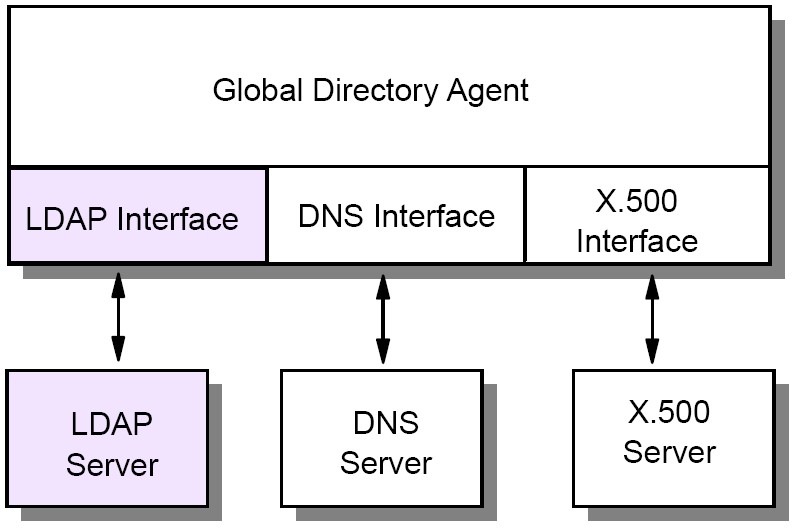
DCE cells can be registered. This global directory (GDS) can be either a Domain Name System (DNS) directory or an X.500 directory. The GDA retrieves the address of a CDS server in the remote cell. The remote CDS can then be contacted to find DCE resources in that cell. Using the GDA enables an organization to link multiple DCE cells together using either a private directory on an intranet or a public directory on the Internet.

In view of LDAP's strong presence in the Internet, two LDAP projects have been sponsored by The Open Group to investigate LDAP integration with DCE technology.

##### LDAP interface for the GDA

One way LDAP is being integrated into DCE is to allow DCE cells to be registered in LDAP directories. The GDA in a cell that wants to connect to remote cells is configured to enable access to the LDAP directory (see Figure 12-17).

*Figure 12-17 The LDAP interface for GDA*



DCE originally only supported X.500 and DNS name syntax for cell names. LDAP and X.500 names both follow the same hierarchical naming model, but their syntax is slightly different. X.500 names are written in reverse order and use a slash (/) rather than a comma (,) to separated relative distinguished names. When the GDA is configured to use LDAP, it converts cell names in X.500 format into the LDAP format.

##### LDAP interface for the CDS

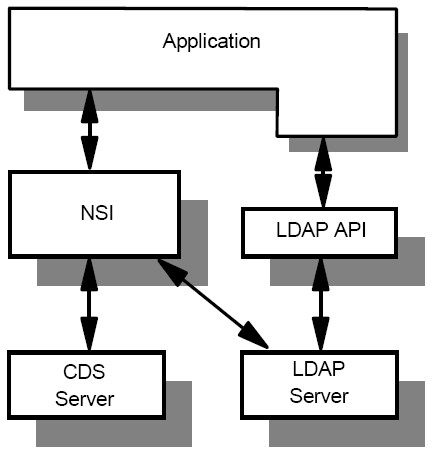
DCE provides two programming interfaces to the Directory Service; Name

Service Interface (NSI) and the X/Open Directory Service (XDS). XDS is an X.500-compatible interface used to access information in the GDS, and it can also be used to access information in the CDS. However, the use of NSI is much more common in DCE applications. The NSI API provides functionality that is

specifically tailored for use with DCE client and server programs that use RPC. NSI allows servers to register their address and the type of RPC interface they support. This address/interface information is called an RPC binding, and is needed by clients that want to contact the server. NSI allows clients to search the CDS for RPC binding information.

NSI was designed to be independent of the directory where the RPC bindings are stored. However, the only supported directory to date has been CDS. NSI will be extended to also support adding and retrieving RPC bindings from an LDAP directory. This will allow servers to advertise their RPC binding information in either CDS or an LDAP directory. Application programs can use either the NSI or the LDAP API when an LDAP directory is used (see Figure 12-18).

*Figure 12-18 The LDAP interface for NSI*



#### 12.4.8 The Directory-Enabled Networks (DEN) initiative

In September 1997, Cisco Systems Inc. and Microsoft® Corp. announced the so-called Directory-Enabled Networks (DEN) initiative as a result of a collaborative work. Many companies, such as IBM, either support this initiative or

actively participate in ad hoc working groups (ADWGs). DEN represents an information model specification for an integrated directory that stores information about people, network devices, and applications. The DEN schema defines the object classes and their related attributes for those objects. As such, DEN is a key piece to building intelligent networks, where products from multiple vendors can store and retrieve topology and configuration-related data.

Of special interest is that the DEN specification defines LDAPv3 as the core protocol for accessing DEN information, which makes information available to LDAP-enabled clients or network devices, or both.

DEN makes use of the Common information Model (CIM). CIM details a way of integrating different management models such as SNMP MIBs and DMTF MIFs. At the time of writing, the most current CIM schema was version 2.12, released in April of 2006.

More information about the DEN initiative can be found on the founder’s Web at:

<http://www.dmtf.org/standards/wbem/den/><http://www.dmtf.org/standards/cim/>

#### 12.4.9 Web-Based Enterprise Management (WBEM)

WBEM is a set of standards designed to deliver an integrated set of management tools for the enterprise. By making use of XML and CIM, it becomes possible to manage network devices, desktop systems, telecom systems and application systems, all from a Web browser. For further information, see: <http://www.dmtf.org/standards/wbem/>

### 12.5 RFCs relevant to this chapter

The following RFCs provide detailed information about the directory and naming protocols and architectures presented throughout this chapter:

[RFC 1032 – Domain administrators guide (November 1987)](http://www.ietf.org/rfc/rfc1032.txt)

[RFC 1033 – Domain administrators operations guide (November 1987)](http://www.ietf.org/rfc/rfc1033.txt)

[RFC 1034 – Domain names - concepts and facilities (November 1987)](http://www.ietf.org/rfc/rfc1034.txt)

[RFC 1035 – Domain names - implementation and specifications](http://www.ietf.org/rfc/rfc1035.txt)

[(November](http://www.ietf.org/rfc/rfc1035.txt) [1987)](http://www.ietf.org/rfc/rfc1035.txt)

[RFC 1101 – DNS encoding of network names and other types (April 1989)](http://www.ietf.org/rfc/rfc1101.txt)

[RFC 1183 – New DNS RR Definitions (October 1990)](http://www.ietf.org/rfc/rfc1183.txt)

[RFC 1202 – Directory Assistance service (February 1991)](http://www.ietf.org/rfc/rfc1202.txt)

[RFC 1249 – DIXIE Protocol Specification (August 1991)](http://www.ietf.org/rfc/rfc1249.txt)

[RFC 1348 – DNS NSAP RRs (July 1992)](http://www.ietf.org/rfc/rfc1348.txt)

[RFC 1480 – The US Domain (June 1993)](http://www.ietf.org/rfc/rfc1480.txt)

[RFC 1706 – DNS NSAP Resource Records (October 1994)](http://www.ietf.org/rfc/rfc1706.txt)

[RFC 1823 – The LDAP Application Programming Interface (August 1995)](http://www.ietf.org/rfc/rfc1823.txt)

[RFC 1876 – A Means for Expressing Location Information in the Domain](http://www.ietf.org/rfc/rfc1876.txt)

[Name System (January 1996)](http://www.ietf.org/rfc/rfc1876.txt)

[RFC 1995 – Incremental Zone Transfer in DNS (August 1996)](http://www.ietf.org/rfc/rfc1995.txt)

[RFC 1996 – A Mechanism for Prompt Notification of Zone Changes (DNS](http://www.ietf.org/rfc/rfc1996.txt)

[NOTIFY) (August 1996)](http://www.ietf.org/rfc/rfc1996.txt)

[RFC 2136 – Dynamic Updates in the Domain Name System (DNS UPDATE) (April 1997)](http://www.ietf.org/rfc/rfc2136.txt)

[RFC 2444 – The One-time-Password SASL Mechanism (October 1998)](http://www.ietf.org/rfc/rfc2444.txt)

RFC 4592 – The Role of Wildcards in the Domain Name System (July 2006)

[RFC 2743 – Generic Security Service Application Program Interface Version](http://www.ietf.org/rfc/rfc2743.txt)

[2, Update 1 (January 2000)](http://www.ietf.org/rfc/rfc2743.txt)

[RFC 2874 – DNS Extensions to Support IPv6 Address Aggregation and](http://www.ietf.org/rfc/rfc2874.txt)

[Renumbering (July 2000)](http://www.ietf.org/rfc/rfc2874.txt)

[RFC 3007 – Secure Domain Name Systems (DNS) Dynamic Update](http://www.ietf.org/rfc/rfc3007.txt)

[(November 2000)](http://www.ietf.org/rfc/rfc3007.txt)

[RFC 3494 – Lightweight Directory Access protocol version 2 (LDAPv2) (March 2003)](http://www.ietf.org/rfc/rfc3494.txt)

[RFC 3596 – DNS Extensions to Support IP Version 6 (October 2003)](http://www.ietf.org/rfc/rfc3596.txt)

[RFC 3645 – Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG) (October 2003)](http://www.ietf.org/rfc/rfc3645.txt)

[RFC 3901 – DNS IPv6 Transport Operational Guidelines (September 2004)](http://www.ietf.org/rfc/rfc3901.txt)

[RFC 4033 – DNS Security Introduction and Requirements (March 2005)](http://www.ietf.org/rfc/rfc4033.txt)

[RFC 4034 – Resource Records for the DNS Security Extensions](http://www.ietf.org/rfc/rfc4034.txt)

[(March](http://www.ietf.org/rfc/rfc4034.txt) [2005)](http://www.ietf.org/rfc/rfc4034.txt)

[RFC 4035 – Protocol Modifications for the DNS Security Extensions](http://www.ietf.org/rfc/rfc4035.txt)

[(March](http://www.ietf.org/rfc/rfc4035.txt) [2005)](http://www.ietf.org/rfc/rfc4035.txt)

[RFC 4339 – IPv6 Host Configuration of DNS Server Information Approaches (February 2006)](http://www.ietf.org/rfc/rfc4339.txt)

[RFC 4398 – Storing Certificates in the Domain Name System (DNS)](http://www.ietf.org/rfc/rfc4398.txt)

[(March](http://www.ietf.org/rfc/rfc4398.txt) [2006)](http://www.ietf.org/rfc/rfc4398.txt)

[RFC 4422 – Simple Authentication and Security Layer (SASL) (June 2006)](http://www.ietf.org/rfc/rfc4422.txt)

[RFC 4501 – Domain Name System Uniform Resource Identifiers (May 2006)](http://www.ietf.org/rfc/rfc4501.txt) [RFC 4505 – Anonymous Simple Authentication and Security Layer (SASL) (June 2006)](http://www.ietf.org/rfc/rfc4505.txt)

[RFC 4510 – Lightweight Directory Access Protocol (LDAP): Technical](http://www.ietf.org/rfc/rfc4510.txt)

[Specification Road Map (June 2006)](http://www.ietf.org/rfc/rfc4510.txt)

[RFC 4511 – Lightweight Directory Access Protocol (LDAP): The Protocol (June 2006)](http://www.ietf.org/rfc/rfc4511.txt)

[RFC 4512 – Lightweight Directory Access Protocol (LDAP): Directory](http://www.ietf.org/rfc/rfc4512.txt)

[Information Models (June 2006)](http://www.ietf.org/rfc/rfc4512.txt)

[RFC 4513 – Lightweight Directory Access Protocol (LDAP): Authentication](http://www.ietf.org/rfc/rfc4513.txt)

[Methods and Security Mechanisms (June 2006)](http://www.ietf.org/rfc/rfc4513.txt)

[RFC 4514 – Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names (June 2006)](http://www.ietf.org/rfc/rfc4514.txt)

[RFC 4515 – Lightweight Directory Access Protocol (LDAP): String](http://www.ietf.org/rfc/rfc4515.txt)

[Representation of Search Filters (June 2006)](http://www.ietf.org/rfc/rfc4515.txt)

[RFC 4516 – Lightweight Directory Access Protocol (LDAP): Uniform](http://www.ietf.org/rfc/rfc4516.txt)

[Resource Locator (June 2006)](http://www.ietf.org/rfc/rfc4516.txt)

[RFC 4517 – Lightweight Directory Access Protocol (LDAP): Syntaxes and](http://www.ietf.org/rfc/rfc4517.txt)

[Matching Rules (June 2006)](http://www.ietf.org/rfc/rfc4517.txt)

[RFC 4518 – Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation (June 2006)](http://www.ietf.org/rfc/rfc4518.txt)

[RFC 4519 – Lightweight Directory Access Protocol (LDAP): Schema for User Applications (June 2006)](http://www.ietf.org/rfc/rfc4519.txt)

[RFC 4520 – Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP) (June 2006)](http://www.ietf.org/rfc/rfc4520.txt)

[RFC 4521 – Considerations for Lightweight Directory Access Protocol](http://www.ietf.org/rfc/rfc4521.txt)

[(LDAP) (June 2006)](http://www.ietf.org/rfc/rfc4521.txt)

[RFC 4522 – Lightweight Directory Access Protocol (LDAP): The Binary](http://www.ietf.org/rfc/rfc4522.txt)

[Encoding Option (June 2006)](http://www.ietf.org/rfc/rfc4522.txt)

[RFC 4523 – Lightweight Directory Access Protocol (LDAP): Schema](http://www.ietf.org/rfc/rfc4523.txt)

[Definitions for X.509 Certificates (June 2006)](http://www.ietf.org/rfc/rfc4523.txt)

[RFC 4524 – Lightweight Directory Access Protocol (LDAP): COSINE/LDAP X.500 Schema (June 2006)](http://www.ietf.org/rfc/rfc4524.txt)

[RFC 4525 – Lightweight Directory Access Protocol (LDAP): Modify-Increment](http://www.ietf.org/rfc/rfc4525.txt)

[Extension (June 2006)](http://www.ietf.org/rfc/rfc4525.txt)

[RFC 4526 – Lightweight Directory Access Protocol (LDAP): Absolute True and False Filters (June 2006)](http://www.ietf.org/rfc/rfc4526.txt)

[RFC 4527 – Lightweight Directory Access Protocol (LDAP): Read Entry Controls (June 2006)](http://www.ietf.org/rfc/rfc4527.txt)

[RFC 4528 – Lightweight Directory Access Protocol (LDAP): Assertion Control (June 2006)](http://www.ietf.org/rfc/rfc4528.txt)

[RFC 4529 – Requesting Attributes by Object Class in the Lightweight](http://www.ietf.org/rfc/rfc4529.txt)

[Directory Access Protocol (LDAP) (June 2006)](http://www.ietf.org/rfc/rfc4529.txt)

[RFC 4530 – Lightweight Directory Access Protocol (LDAP): entryUUID](http://www.ietf.org/rfc/rfc4530.txt)

[Operational Attribute (June 2006)](http://www.ietf.org/rfc/rfc4530.txt)

[RFC 4531 – Lightweight Directory Access Protocol (LDAP): Turn Operation (June 2006)](http://www.ietf.org/rfc/rfc4531.txt)

[RFC 4532 – Lightweight Directory Access Protocol (LDAP): “Who Am I?”](http://www.ietf.org/rfc/rfc4532.txt)

[Operation (June 2006)](http://www.ietf.org/rfc/rfc4532.txt)

[RFC 4533 – Lightweight Directory Access Protocol (LDAP): Content](http://www.ietf.org/rfc/rfc4533.txt)

[Synchronization Operation (June 2006)](http://www.ietf.org/rfc/rfc4533.txt)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **17** | |

## Chapter 17. Network management

With the growth in size and complexity of the TCP/IP-based internetworks the need for network management became very important. To address this, in 1988

the Internet Architecture Board (IAB) issued RFC 1052 detailing its recommendation to achieve this management. The suggestion adopted two different approaches:

The Simple Network Management Protocol (SNMP)

ISO Common Management Information Services/Common Management

Information Protocol (CMIS/CMIP)

The original plan detailed by RFC 1052 involved using SNMP as a short-term solution to the network management problem. The development of SNMP was to be kept simple, facilitating rapid deployment of the protocol throughout the Internet community. After the immediate management needs were met, albeit temporarily, by SNMP, thorough research and development could be performed on CMIS/CMIP. Ultimately, this protocol would then be deployed as a permanent solution, replacing SNMP.

However, this plan allowed SNMP to gain widespread usage throughout the

Internet community. As a result, CMIS/CMIP was never fully deployed. In fact, only two RFCs describing CMIS/CMIP were released. RFC 1095, which described the CMIS protocol over TCP/IP (CMOT) was released in 1989. It was ultimately obsoleted by RFC 1189, released in 1990, which proposed the standard for CMOT and CMIP. The status of RFC 1189 is currently historic, and is not widely used in today’s networks.

© Copyright IBM Corp. 1989-2006. All rights reserved.

Given these circumstances, this chapter discusses the SNMP model and implementation. Also included is a discussion on the NETSTAT utility. Though this command is not RFC defined, it is available on most platforms and is useful in both monitoring and managing local connections on a system.

### 17.1 The Simple Network Management Protocol (SNMP)

The fundamental use of the Simple Network Management Protocol (SNMP) is to manage all aspects of a network, as well as applications related to that network. To do this, SNMP has been architected to perform two services:

Monitor: SNMP implementations allow network administrators to monitor their networks in order to--among other things--ensure the health of the network, forecast usage and capacity, and in problem determination. Aspects which can be monitored vary in granularity, and can be something as global as the total amount of IP traffic experienced on a single host, or can be as minute as the current status of a single TCP connection.

Additionally, the SNMP architecture allows components of the SNMP model to notify network administrators should a problem occur on a network. For example, if a link were to break or an interface to deactivate for some reason, SNMP can send a message to alert network administrators that this has occurred.

Manage: In addition to monitoring a network, SNMP provides the capability for network administrators to affect aspects with the network. Values which regulate network operation can be altered, allowing administrators to quickly respond to network problems, dynamically implement new network changes, and to perform real-time testing on how changes may affect their network.

SNMP implements a manager/agent/subagent model, which conforms very closely to the client/server model (11.1.1, “The client/server model” on page 408). RFC 1157 defines the components and interactions involved in an SNMP community, which include:

A Management Information Base (MIB, discussed in 17.1.1, “The

Management Information Base (MIB)” on page 625)

An SNMP agent (discussed in 17.1.2, “The SNMP agent” on page 630)

An manager (discussed in 17.1.3, “The SNMP manager” on page 631) SNMP subagents (discussed in 17.1.4, “The SNMP subagent” on page 632)

#### 17.1.1 The Management Information Base (MIB)

The management information base defines a set of objects which can be monitored or managed using an SNMP implementation. The current MIB, MIB-II, is defined in RFC 1213 and replaces the MIB-I definition outlined in RFC 1156. It

is updated by RFCs 4022, 4113, and 4293. MIB-II defines the groups of information which should be made available in any SNMP implementation in a TCP/IP based network. These groups are as shown in Table 17-1.

*Table 17-1 Group names and descriptions available in an SNMP implementation*

|  |  |  |
| --- | --- | --- |
| **Group name** | **Description of objects within the group** | **RFC defining**  **the group’s MIB** |
| System | Basic system information, such as the system’s name, description, and how much time has passed since the last time the system was restarted. | RFC 3418 |
| Interfaces | Information about network interfaces, including a list of interfaces, and statistics specific to these interfaces. | RFC 2863 |
| IP | Information and statistics on IP traffic. | RFC 4293 |
| ICMP | Statistics on ICMP input and output. | RFC 4293 |
| TCP | General information about the TCP layer (such as timeout values and the total number of TCP connections) as well as information about specific TCP connections (such as the connection’s state, addresses, and ports). | RFC 4022 |
| UDP | General information about the UDP layer (such as the number of UDP packets sent and delivered) as well as information about specific UDP usage (such as addresses and ports). | RFC 4113 |
| EGP | Statistics on external gateway protocol traffic. | RFC 1156  RFC 1213 |
| Transmission | This group is not yet implemented, but was created as a placeholder for when Internet-standard definitions for managing transmission media emerge. | RFC 1213 |

|  |  |  |
| --- | --- | --- |
| **Group name** | **Description of objects within the group** | **RFC defining**  **the group’s MIB** |
| SNMP | Information and statistics related to the SNMP environment. | RFC 3411  RFC 3412  RFC 3413  RFC 3414  RFC 3415  RFC 3418 |

Note that these constitute only the minimum implementation. An implementation can also create it’s own groups and objects which are application or platform-specific. In such cases, the MIB must be built into the SNMP agent itself, or be made available to the agent using the distributed programming interface (DPI, see 11.2.3, “The SNMP distributed programming interface (SNMP DPI)” on page 419). Additionally, each managed node supports only those groups that are appropriate. For example, if there is no gateway, the EGP group need not be supported. If a group is appropriate, all objects in that group must be supported.

Objects within a MIB are defined using the Structure of Management Information Version 2 (SMIv2), defined in RFC 2578.

##### Structure of Management Information Version 2 (SMIv2)

SMIv2 defines the rules for how managed objects are described and how management protocols can access these objects. The description of managed objects is made using a subset of the Abstract Syntax Notation One (ASN.1, ISO standard 8824), a data description language. The object type definition uses the following syntax:

|  |  |
| --- | --- |
| objectName | OBJECT-TYPE |
| SYNTAX | *syntax* |
| UNITS | “*units*” |
| MAX-ACCESS | *access* |
| STATUS | *status* |
| DESCRIPTION | “*descriptiveText*” |
| REFERENCE | “*referenceText*” |
| INDEX | {*indexTypes*} |
| DEFVAL | {*defaultValue*} |

::= { *group #* }

These fields are defined as follows:

|  |  |
| --- | --- |
| **OBJECT-TYPE** | A textual name, called the *object descriptor*, for the object type along with its corresponding *object identifier* defined later. |
| **SYNTAX** | The abstract syntax for the object type. It can be a choice of SimpleSyntax (Integer, Octet String, Object Identifier, Null) or an ApplicationSyntax (NetworkAddress, Counter, Gauge, TimeTicks, Opaque) or other application-wide types (see RFC 2578 for more details). |
| **UNITS** | This field is optional and applies only when the object maintains a value that is unit specific. For example, an object that monitors time might specify UNIT “seconds” or UNIT “minutes”. |
| **MAX-ACCESS** | Defines the access that a user has to this object. Valid values include accessible-for-notify, read-only, read-write, read\_create, and not-accessible. |
| **STATUS** | Defines the status of the object. Valid values include current, deprecated, and obsolete. |
| **DESCRIPTION** | A textual description of the semantics of the object type. |
| **REFERENCE** | This field is optional and can be used to include a text string referencing some other document. |
| **INDEX** | This field is optional and is only used when the object is part of a conceptual row of objects. |
| **DEFVAL** | This field is optional and is only used when the object should have a default value. |
| **group** | The RFC 1213-defined group to which the object belongs. |
| **#** | The object’s position within the RFC 1213-defined group. |

As an example, consider the object ifType, defined by RFC 2863. ifType is part of the Interface group defined by RFC 1213. Its SMIv2 definitions is illustrated in Figure 17-1.

|  |
| --- |
| ifType OBJECT-TYPE  SYNTAX IANAifType  MAX-ACCESS read-only  STATUS current  DESCRIPTION  “The type of interface. Additional values for ifType are assigned by the Internet Assigned Numbers Authority  (IANA), through updating the syntax of the IANAifType textual convention.”  ::= { ifEntry 3 } |

*Figure 17-1 SMIv2 definition of sysUpTime*

##### Object identifiers (OIDs)

A managed object not only has to be described but identified, too. This is done using the ASN.1 object identifier (OID). The object identifier reserves a set of numbers for different groups. Each object is identified by a string of numbers indicating the hierarchy to which it belongs. Referring back to the example of ifType, this object has an OID of 1.3.6.1.2.1.2.2.1.3. This can initially be broken into two parts:

**ifEntry** 1.3.6.1.2.1.2.2.1

###### ifType 3

Note that the terms ifType, 1.3.6.1.2.1.2.2.1.3, and ifEntry.3 are functionally interchangeable. However, ifType’s OID can be further broken down as follows:

|  |  |
| --- | --- |
| **ifTable** | 1.3.6.1.2.1.2.2 |
| **ifEntry** | 1 |
| **ifType** | 3 |

Again, the terms ifType, 1.3.6.1.2.1.2.2.1.3, ifTable.1.3, and ifEntry.3 are all functionally interchangeable. The OID can continue to be broken down because each digit has a specific meaning. The significance of each digit adheres to the following rules:

The first digit defines the node administrator:

* 1 for ISO
* 2 for CCITT
* 3 for the joint ISO-CCITT

The possible values for the second digit are determined by the first digit. In this case, the ISO node administrator defines *3* for use by other organizations.

The third digit’s potential values again depend on the first and second digits.

But if the first two digits are 1.3, *6* is defined for the use of the U.S. Department of Defense.

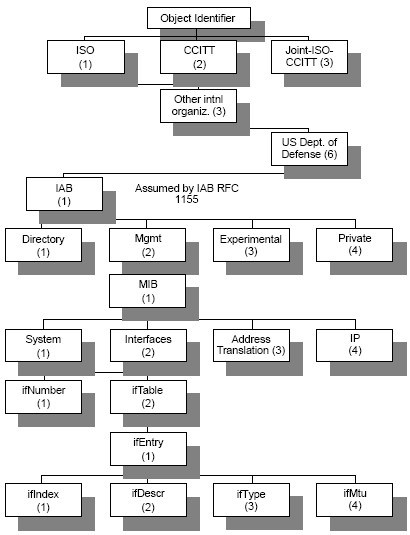
In the fourth group, the Department of Defense has not indicated how it will manage its group, so the Internet community assumed *1* for its own.

The fifth group was approved by IAB to be:

* 1 for the use of OSI directory in the Internet
* 2 for object identification for management purposes
* 3 for object identification for experimental purposes
* 4 for object identification for private use

This if further illustrated in Figure 17-2, which shows a mapping of how the OID number for ifType is determined.

*Figure 17-2 Finding ifType’s OID number*



#### 17.1.2 The SNMP agent

The SNMP agent acts as a server in the client/server model and listens on well-known UDP port 161 for requests from SNMP managers (we discuss managers in 17.1.3, “The SNMP manager” on page 631). Additionally, the SNMP agent is typically responsible for supporting both the system and the SNMP groups defined by RFC 1213.

Upon receiving a request from a manager, the agent determines if the requesting management station has the authority to access the SNMP community. If so, the agent obtains the value of the requested objects and returns them to the manager.

Finally, the agent can accept connections from subagents (discussed in the next section) to make MIBs available to SNMP managers. Upon receiving a request from a manager for an object supported by a subagent, the agent forwards the request to the appropriate subagent. This is illustrated later in Figure 17-5 on page 636.

#### 17.1.3 The SNMP manager

The SNMP manager, also referred to as a Network Management Station (NMS), provides a user interface through which network administrators can monitor and manage their network. The manager fulfills the role of a client in the client/server model and is available in a variety of formats including command-line interfaces, graphical user interface (GUI) applications, and fully automated applications.

The SNMP manager is responsible for issuing requests to the SNMP agent. These requests can be queries to obtain the value of a MIB object, or they can be requests to set the value of a MIB object. SNMP managers also can listen for notifications or alerts, called traps, generated by components in the SNMP community. For additional information about traps, see 17.1.6, “SNMP traps” on page 638.

**Note:** Communication between an SNMP manager and the SNMP agent occurs using the communication structure outlined by the Simple Network Management Protocol. SNMP usually employs UDP as a transport.

An SNMP manager can make the following types of requests to the SNMP agent:

|  |  |
| --- | --- |
| **getRequest** | Requests that the agent return the value of the specified object. |
| **getNextRequest** | Requests that the agent return the first valid value following the specified object. For example, assume a  **getNext** is executed for ifType (1.3.6.1.2.1.2.2.1.3). Assuming the first valid instance of ifType is ifType.1 (1.3.6.1.2.1.2.2.1.3.1), this is the value that the SNMP agent will return. |
| **getBulkRequest** | Performs the same function as the **get** request, but allows the manager to query more than one object per request. This is only valid using the SNMPv2c security model (see “The GetBulkRequest” on page 642). | |
| **setRequest** | Requests that the SNMP agent set the value of the specified object. | |
| **walk** | Implements a series of **getNext** requests such that an entire sequence of objects is returned to the manager. In each iteration of the **getNext** series, the last object returned becomes the next object on which a **getNext** is executed. The **walk** ends when an object is returned that is beyond the scope of the request. An example of this is provided in Figure 17-6 on page 637. | |

Note that the **walk** request is not architected in the SNMP communication that occurs between the SNMP manager and SNMP agent. Instead, it is a convention widely used by most SNMP managers.

#### 17.1.4 The SNMP subagent

An SNMP subagent supports its own MIB, which might be an RFC-architected MIB, or might be a proprietary (referred to as enterprise-specific) MIB. For example, a TCP/IP subagent would most likely support the IP, ICMP, TCP, UDP, and Interface groups defined in RFC 1213. However, an individual software company might want use SNMP to make available information specific to their software. To do this, they can create a subagent that supports their enterprise-specific MIB.

The subagent, upon initializing, opens a DPI connection to the SNMP agent. This occurs by first querying the agent, as though the subagent were a manager, for information about the agent’s DPI ports. Note that this information is maintained by the SNMP agent in the following two objects: dpiPortForTCP.0 1.3.6.1.4.1.2.2.1.1.1.0 dpiPortForUDP.0 1.3.6.1.4.1.2.2.1.1.2.0

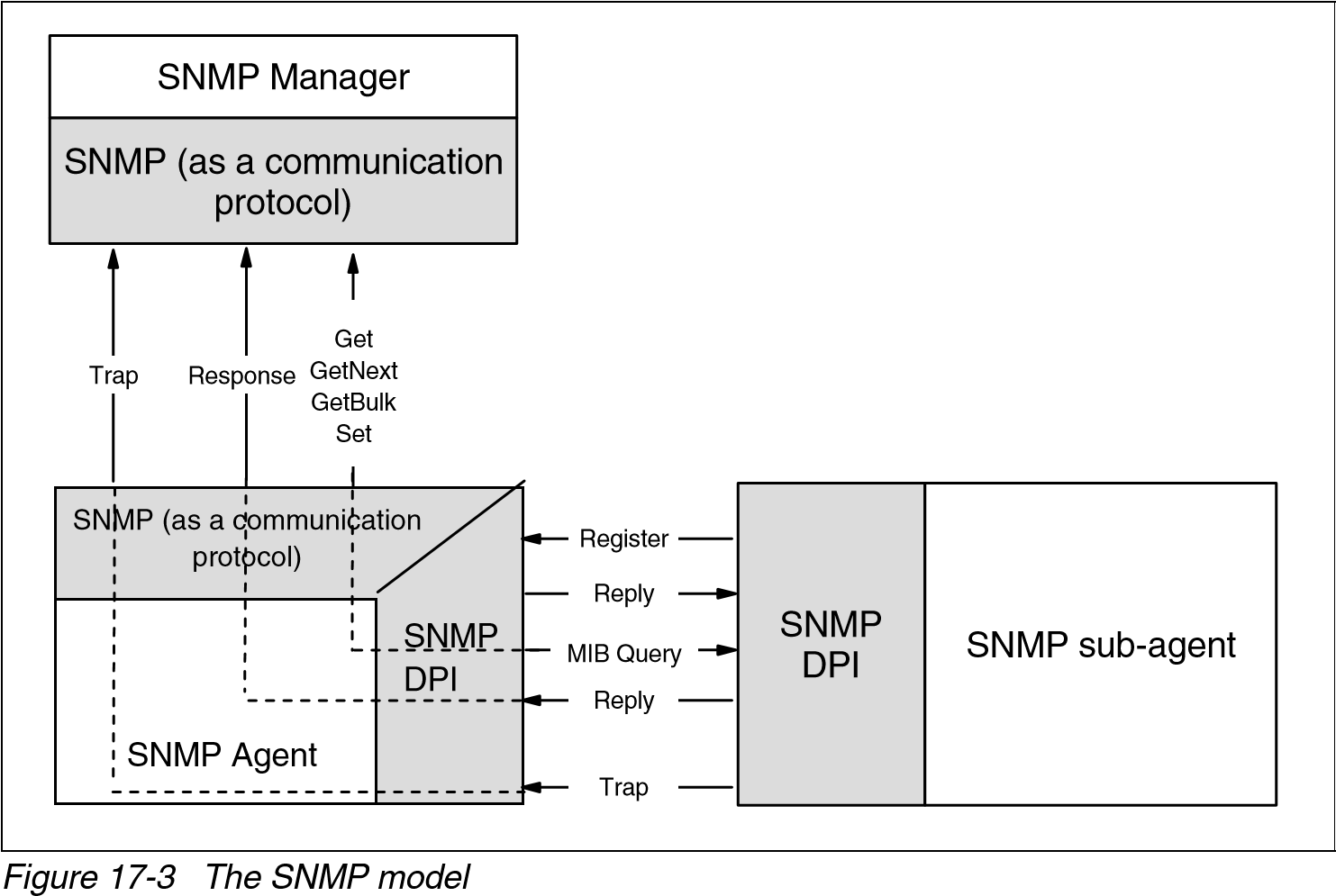
The agent’s response directs the subagent to the correct port over which a DPI connection can be opened. With this information, the subagent can interact with the agent, as described in 11.2.3, “The SNMP distributed programming interface (SNMP DPI)” on page 419.

#### 17.1.5 The SNMP model

The interaction between SNMP components is agent-restrictive: A manager can communicate only with an agent, and a subagent can communicate only with an agent. In no aspects of the model will a subagent communicate with a manager. If a manager needs to obtain or set the value of an object supported by a subagent, the request is delivered to the agent.

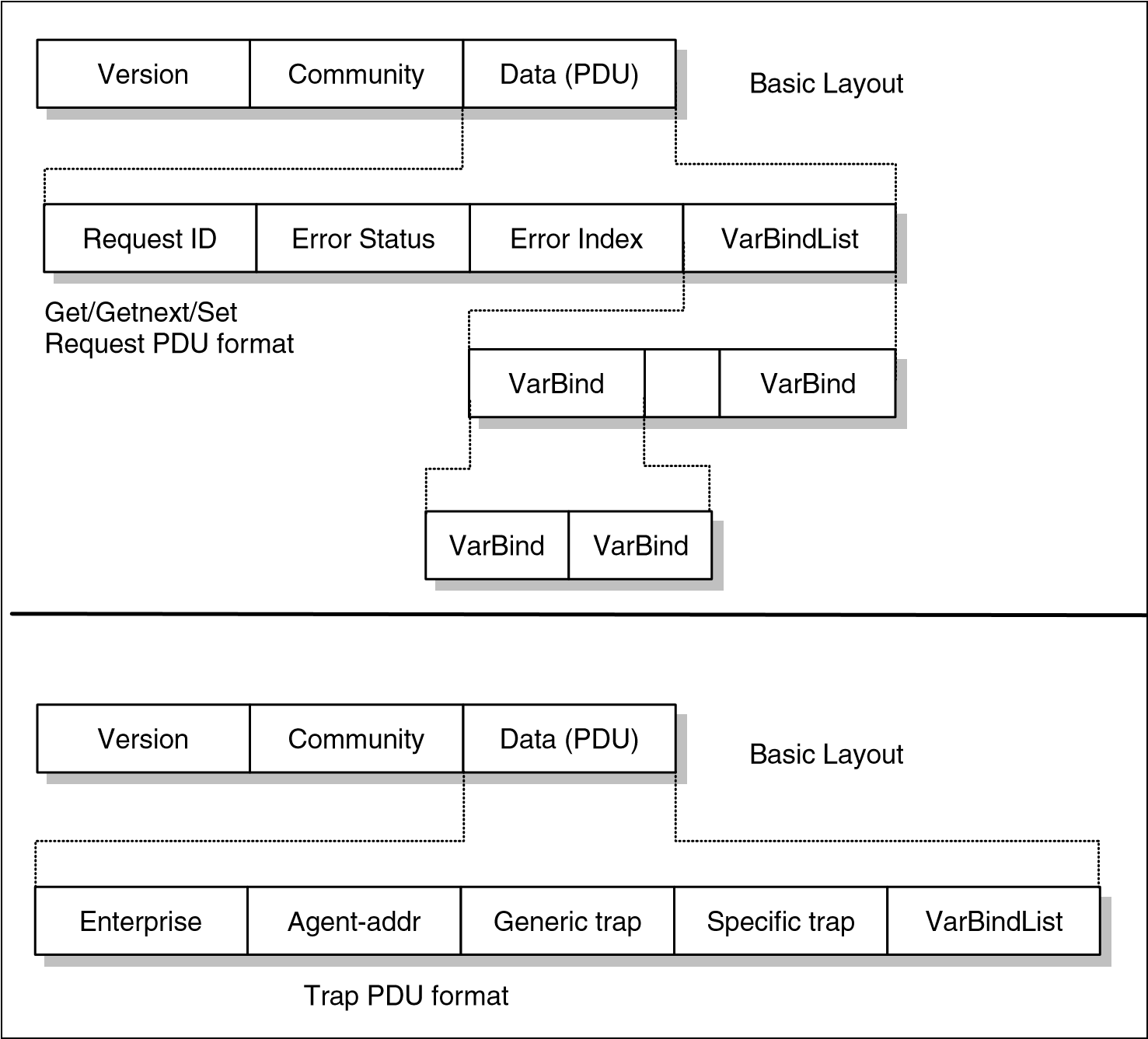
The agent, upon realizing that the request is for an object in a MIB other than one it supports, attempts to find the object in one of the MIBs registered by a subagent. Upon finding the correct MIB, the agent passes the request to the subagent.

The subagent then locates the correct value and passes it back to the SNMP agent. The agent then forwards this value back to the manager. This process, as well as the relationship that exists between manager, agents, and subagents, is illustrated in Figure 17-5 on page 636.



Messages sent between SNMP agents and managers use a Protocol Data Unit

(PDU) preceded by the SNMP header. The header specifies the version of SNMP being used, as well as authentication credentials. The PDU contains information regarding the type of request or response contained in the PDU, and in the case of a response, the actual value of the queried objects. The SNMP message format is illustrated in Figure 17-4.



*Figure 17-4 The SNMP message format*

This format is defined in RFC 1157, and the fields are defined as follows:

|  |  |
| --- | --- |
| **Version** | Indicates the version of SNMP being used (valid values are 1, 2, and 3). |
| **Community** | The SNMP community to which the request is directed, or from which the response originated. |
| **Request ID** | Serializes request/response iterations. A manager uses this to determine which responses correspond to a specific request. |
| **Error Status** | Specified on responses to inform the manager of any errors encountered with the request. On request PDUs, this value is set to 0. Valid values are: **noError** No error was encountered.  **tooBig** The response was too large to deliver in a PDU.  **noSuchName** The requested object does not exist.  **badValue** The returned value does not adhere to ASN.1 encoding standards.  **readOnly** A set request was executed for an object which is read-only.  **genErr** An error not covered by the other types occurred. |
| **Error Index** | An integer indicating which object in a list of objects caused the error reported by the errorStatus. |
| **VarBind** | The list of objects requested and, for a response, their values. |
| **Enterprise** | The entity (defined by the sysObjectID MIB object) generating an enterprise-specific trap. |
| **Agent-addr** | The address of the SNMP agent from which the trap was generated. |
| **Generic trap** | The generic trap type that indicates the type of trap sent. If this is an enterprise-specific trap, this is set to 6 (see 17.1.6, “SNMP traps” on page 638). |
| **Specific trap** | The value of an enterprise-specific trap (see 17.1.6, “SNMP traps” on page 638). |

To illustrate the process of querying an object using SNMP, refer back to the example of the ifType object. Figure 17-5 illustrates the sequence of events occurring when executing a **getNext** request for the ifType object.

getNext ifType

161)

UDP,Port

(

SNMP

Manager

SNMP

Agent

ifType belongs to the

Interface Group,

which was registered

by the TCP/IP

sub-agent

Forward this value

to the requesting

manager

TCPIP

Sub-agent

Retrieve from the

Interface Group MIB

ifType.1 = 6

ifType.1 = 6

UDP, manager's

(

ephemeral port)

getNext ifType

(

DPI socket

)

IPGroup MIB

UDP Group MIB

TCP Group MIB

ifType.1 = 6

)

DPI socket

(

SNMP Group

MIB

System Group

MIB

MIBs registered

by sub-agents

Interface Group

MIB

ifType.1 = 6

ICMP Group MIB

*Figure 17-5 An example of SNMP manager, agent, and subagent interaction*

The sequence of events is as follows:

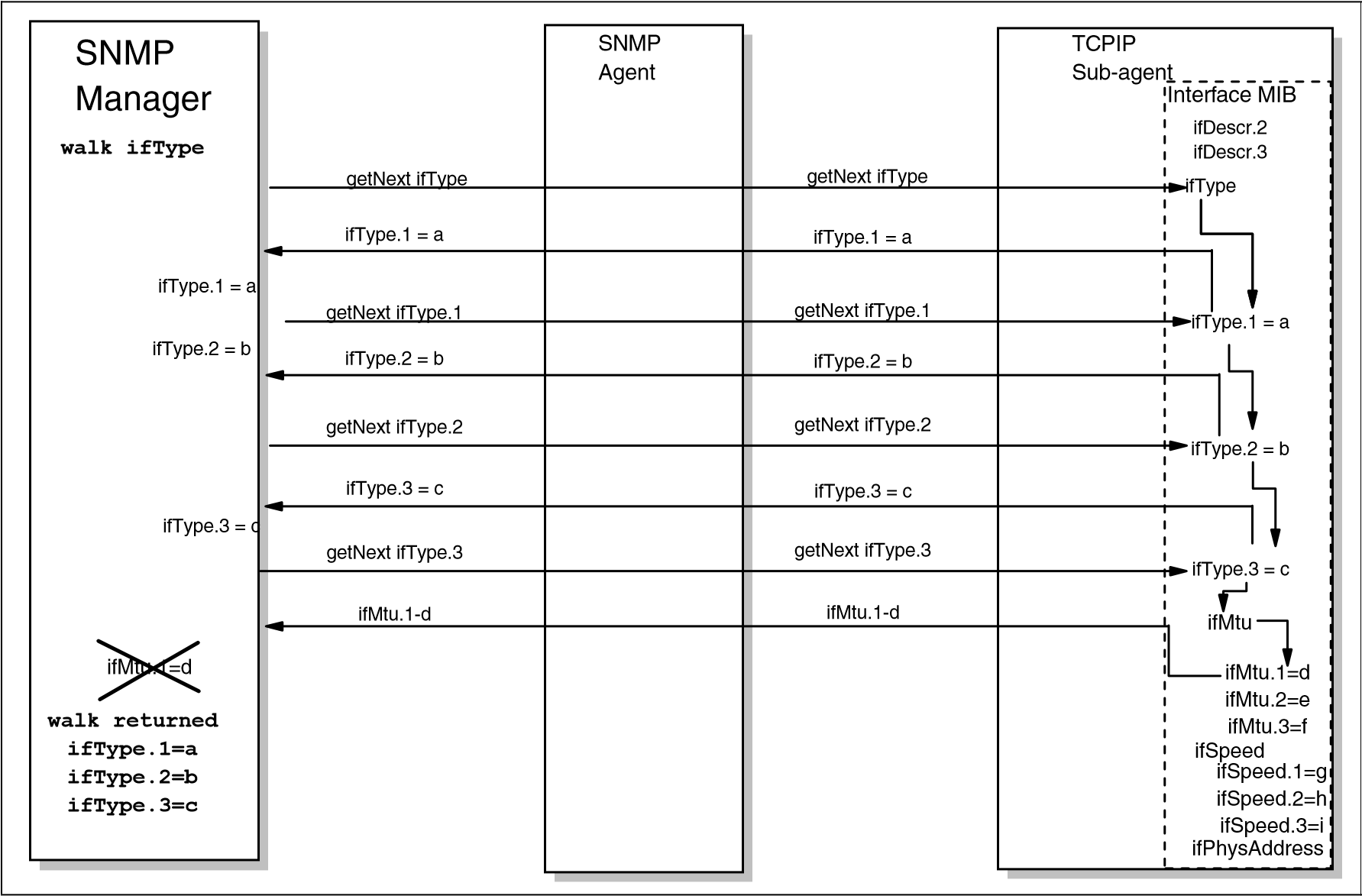
1. In this example, a **getNext** request is executed by the SNMP manager for ifType.
2. The SNMP agent does not recognize ifType as an object from the System Group or SNMP Group MIBs. It does, however, recognize ifType as an object from a MIB registered by a subagent. In this case, the MIB is the Interface Group MIB, and was registered by the TCP/IP subagent.
3. The SNMP agent passes the **getNext** over the DPI socket to the TCP/IP subagent.
4. The subagent recognizes ifType as an object in the Interface MIB and obtains the next object with a valid value: ifType.1.
5. The subagent sends a response back to the agent with the object ifType.1

and the value 6 (which, by RFC definition, is Ethernet).

1. The agent then replies back to the manager, again indicating that the object is ifType.1 and the value is 6.

Note that, had the **getNext** request been for an object in the System Group or SNMP Group MIBs, the SNMP agent would have recognized the object and responded directly to the manager. No subagent would have been involved in this circumstance.

While the concept of the **get**, **getNext**, **getBulk**, and **set** processes are somewhat simple, the process of executing a **walk** warrants additional discussion. As noted previously, a **walk** request is never actually sent to the SNMP agent. Instead, the walk is a manager convention, and is implemented as a series of **getNext** requests. In each iteration of the series, the object returned previously by the agent becomes the next object specified on the **getNext** request. The first time the agent returns a value outside of the range specified on the **walk** request, the **walk** ends. To demonstrate this, assume that there are three instances of ifType at the time a **walk** is issued for the ifType object: ifType.1 through ifType.3. The processing of the walk proceeds as illustrated in Figure 17-6.



*Figure 17-6 An example of a walk on ifType*

In this illustration, we see the following progression:

1. The user executes a **walk** on ifType, which the manager implements by first executing a **getNext** on ifType. This is forwarded to the TCP/IP subagent by the SNMP agent.
2. The subagent obtains the next valid value following ifType, ifType.1, and returns this value to the agent. The agent passes this information back to the manager.
3. The manager takes the ifType.1 response and executes a **getNext** for it.
4. The subagent locates ifType.1 and returns the next valid value: ifType.2.
5. The manager executes a **getNext** for ifType.2 6. The subagent returns ifType.3
6. The manager executes a **getNext** for ifType.3
7. The subagent returns ifMtu.1
8. The manager recognizes that ifMtu.1 is outside the scope of ifType, and assumes that the **walk** is complete.

#### 17.1.6 SNMP traps

Traps are asynchronous notifications of events occurring within an SNMP community. They can be generated both by SNMP agents and SNMP subagents. Additionally, they can be RFC architected (called a generic trap type) or they can be proprietary (called enterprise-specific). Architected traps, defined in RFC 1215, are as follows:

|  |  |
| --- | --- |
| **coldStartEvent** | Notifies managers that the SNMP agent is reinitializing and that the configuration might have been altered. This trap belongs to the RFC 1213-architected System group, and is therefore supported by the SNMP agent. |
| **warmStartEvent** | Notifies managers that the SNMP agent is reinitializing, but there has been no alteration of the configuration. This trap belongs to the RFC 1213-architected System group, and is therefore supported by the SNMP agent. |
| **linkDownEvent** | Notifies managers that an interface has been deactivated. Information identifying the interface is included in the trap. This trap belongs to the RFC 1213-architected Interface group and is usually supported by a TCP/IP specific  subagent. |

**linkUpEvent** Notifies managers that an interface has been activated. Information identifying the interface is included in the trap. This trap belongs to the RFC 1213-architected Interface group and is usually supported by a TCP/IP-specific subagent, or by an SNMP agent that manages its own TCP/IP MIBs.

##### snmpAuthenFailureEvent

Notifies managers that a user attempting to access the SNMP community did not provide the credentials needed to gain authorization by the SNMP agent. This trap belongs to the RFC 1213-architected SNMP group, and therefore is supported by the SNMP agent.

##### egpNeighborLossEvent

Notifies manages that a relationship with an EGP neighbor no longer exists. Information identifying the EGP neighbor is included in the trap. This trap belongs to the RFC 1213-architected EGP group, and therefore is usually supported by an EGP-specific subagent or a TCP/IP specific-subagent.

**entSpecificEvent** This trap is a placeholder that allows SNMP agent or subagent implementations to create enterprise-specific traps.

Traps generated by an SNMP agent are usually delivered to managers using well-known UDP port 162. However, SNMP implementations might provide a configuration option to allow traps to be sent to other user-determined ports. If a subagent generates a trap, the trap is not sent directly from the subagent to a manager. Instead, the trap is passed over the DPI connection to the agent, who then sends out the trap to the managers (see Figure 17-3 on page 633).

#### 17.1.7 SNMP versions

There are three versions of SNMP available, usually referred as SNMPv1 (RFC 1157), SNMPv2 (RFC 1901), and SNMPv3 (RFC 3414). Additionally, RFC 3584 was created to specify how all three versions can coexist with a single SNMP community. The security functions provided by the SNMP protocols are categorized into the following two models:

Community-based Security Model, whose data is protected by nothing more than a password, referred to as the community name (see Figure 17-4 on page 634). Community-based security allows the SNMP agent to authenticate a request based on the community name used and the IP address from which the request originated. This level of security is provided by the SNMPv1 and SNMPv2 Community-based Security Models.

User-based Security Model (USM), which provides different levels of security, based on the user accessing the managed information. To support this security level, the SNMPv3 framework defines several security functions, such as USM for authentication and privacy and vIew-based Access Control Model (VACM, defined in RFC 3415), which provides the ability to limit access to different MIB objects on a per-user basis and the use of authentication and data encryption for privacy.

##### SNMPv1

Version 1 of SNMP incorporates the basics of SNMP already covered in this chapter. Therefore, there is no need for additional discussion except to note that SNMPv1 does not allow **getBulk** requests. Such requests executed in an

SNMPv1 community, depending on the SNMP manager implementation, will result either in an error or in downgrading the request to a series of **get** requests.

##### SNMPv2

The framework of Version 2 of the Simple Network Management Protocol (SNMPv2) was published in April 1993 and consists of 12 RFCs, including the first, RFC 1441, which is an introduction. In August 1993, all 12 RFCs became a proposed standard with the status elective.

This framework consists of the following disciplines:

Structure of Management Information (SMI)

Definition of the OSI ASN.1 subset for creating MIB modules. See RFC 2578 for a description.

Textual conventions

Definition of the initial set of textual conventions available to all MIB modules. See RFC 2579 for a description.

Protocol operations

Definition of protocol operations with respect to the sending and receiving of PDUs. See RFC 3416 for a description.

Transport mappings

Definition of mapping SNMPv2 onto an initial set of transport domains because it can be used over a variety of protocol suites. The mapping onto UDP is the preferred mapping. The RFC also defines OSI, DDP, IPX, and so on. See RFC 3417 for a description.

Protocol instrumentation

Definition of the MIB for SNMPv2. See RFC 3418 for a description.

Administrative framework

Definition of the administrative infrastructure for SNMPv2, the User-based Security Model for SNMPv2 and the Community-based SNMPv2. See RFCs 1901, 1909, and 1910 for descriptions.

Conformance statements

Definition of the notation compliance or capability of agents. See RFC 2580 for a description.

The following sections describe the major differences and improvements from SNMPv1 to SNMPv2.

###### SNMPv2 entity

An SNMPv2 entity is an actual process that performs network management operations by generating or responding, or both, to SNMPv2 protocol messages by using the SNMPv2 protocol operations. All possible operations of an entity can be restricted to a subset of all possible operations that belong to a particular administratively defined party (refer to “SNMPv2 party”). An SNMPv2 entity can be member of multiple SNMPv2 parties. The following local databases are maintained by an SNMPv2 entity:

One database for all parties known by the SNMPv2 entity that can be:

* Operation realized locally
* Operation realized by proxy interactions with remote parties or devices
* Operation realized by other SNMPv2 entities

Another database that represents all managed object resources that are known to that SNMPv2 entity.

And at least a database that represents an access control policy that defines the access privileges accorded to known SNMPv2 parties.

An SNMPv2 entity can act as an SNMPv2 agent or manager.

###### SNMPv2 party

An SNMPv2 party is a conceptual, virtual execution environment whose operation is restricted, for security or other purposes, to an administratively defined subset of all possible operations of a particular SNMPv2 entity (refer to “SNMPv2 entity”). Architecturally, each SNMPv2 party consists of: A single, unique party identity.

A logical network location at which the party executes, characterized by a transport protocol domain and transport addressing information.

A single authentication protocol and associated parameters by which all protocol messages originated by the party are authenticated as to origin and integrity.

A single privacy protocol and associated parameters by which all protocol messages received by the party are protected from disclosure.

###### The GetBulkRequest

The **GetBulkRequest** is defined in RFC 3416 and is thus part of the protocol operations. A **GetBulkRequest** is generated and transmitted as a request of an SNMPv2 application. The purpose of the **GetBulkRequest** is to request the transfer of a potentially large amount of data, including, but not limited to, the efficient and rapid retrieval of large tables. The **GetBulkRequest** is more efficient than the **GetNext** request in case of retrieval of large MIB object tables. The syntax of the **GetBulkRequest** is:

GetBulkRequest [ non-repeaters = N, max-repetitions = M ]

( RequestedObjectName1,

RequestedObjectName2, RequestedObjectName3 )

Where:

**RequestedObjectName1, 2, 3**

MIB object identifier, such as sysUpTime. The objects are in a lexicographically ordered list. Each object identifier has a binding to at least one variable. For example, the object identifier ipNetToMediaPhysAddress has a variable binding for each IP address in the ARP table and the content is the associated MAC address.

**N** Specifies the non-repeaters value, which means that you request

only the contents of the variable next to the object specified in your request of the first N objects named between the parentheses. This is the same function as provided by the **GetNextRequest**.

**M** Specifies the max-repetitions value, which means that you request from the remaining (number of requested objects - N) objects the contents of the M variables next to your object specified in the request. Similar to an iterated **GetNextRequest** but transmitted in only one request.

With the **GetBulkRequest**, you can efficiently get the contents of the next variable or the next M variables in only one request.

Assume the following ARP table in a host that runs an SNMPv2 agent:

Interface-Number Network-Address Physical-Address Type

1 10.0.0.51 00:00:10:01:23:45 static

1. 9.2.3.4 00:00:10:54:32:10 dynamic
2. 10.0.0.15 00:00:10:98:76:54 dynamic

An SNMPv2 manager sends the following request to retrieve the sysUpTime and the complete ARP table:

GetBulkRequest [ non-repeaters = 1, max-repetitions = 2 ]

( sysUpTime, ipNetToMediaPhysAddress, ipNetToMediaType )

The SNMPv2 agent responds with a response PDU:

Response (( sysUpTime.0 = "123456" ),

( ipNetToMediaPhysAddress.1.9.2.3.4 =

"000010543210" ),

( ipNetToMediaType.1.9.2.3.4 = "dynamic" ),

( ipNetToMediaPhysAddress.1.10.0.0.51 =

"000010012345" ),

( ipNetToMediaType.1.10.0.0.51 = "static" ))

The SNMPv2 manager continues with:

GetBulkRequest [ non-repeaters = 1, max-repetitions = 2 ]

( sysUpTime, ipNetToMediaPhysAddress.1.10.0.0.51, ipNetToMediaType.1.10.0.0.51 )

The SNMPv2 agent responds with:

Response (( sysUpTime.0 = "123466" ),

( ipNetToMediaPhysAddress.2.10.0.0.15 =

"000010987654" ), ( ipNetToMediaType.2.10.0.0.15 =

"dynamic" ), ( ipNetToMediaNetAddress.1.9.2.3.4 =

"9.2.3.4" ),

( ipRoutingDiscards.0 = "2" ))

This response signals the end of the table to the SNMPv2 manager. Using the **getNextRequest**, this same result requires four iterations of queries.

###### InformRequest

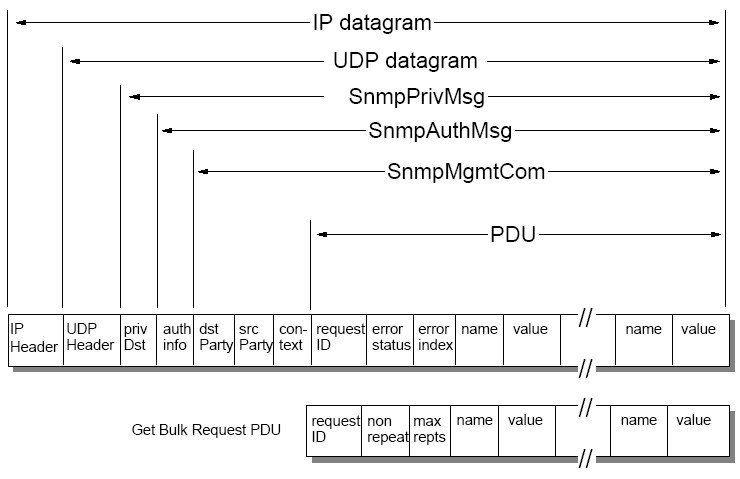
An **InformRequest** is generated and transmitted as a request from an application in an SNMPv2 manager entity that wants to notify another application, acting also in an SNMPv2 manager entity, of information in the MIB view[[5]](#footnote-5) of a party local to the sending application. The packet is used as an indicative assertion to the manager of another party about information accessible to the originating party (manager-to-manager communication across party boundaries). The first two variables in the variable binding list of an **InformRequest** are sysUpTime.0 and snmpEventID.i[[6]](#footnote-6), respectively. Other variables can follow.

###### The new administrative model

It is the purpose of the administrative model for SNMPv2 to define how the administrative framework is applied to realize effective network management in a variety of configurations and environments.

The model entails the use of distinct identities for peers that exchange SNMPv2 messages. Therefore, it represents a departure from the community-based administrative model of the original SNMPv1. By unambiguously identifying the source and intended recipient of each SNMPv2 message, this new strategy improves on the historical community scheme both by supporting a more convenient access control model and allowing for effective use of asymmetric (public key) security protocols in the future. Figure 17-7 on page 645 illustrates the new message format.

*Figure 17-7 The SNMPv2 message format*



In this figure, the fields are defined as follows:

**PDU** Includes one of the following protocol data units:

* GetNextRequest
* GetRequest
* Inform
* Report
* Response
* SNMPv2-Trap
* SetRequest

The **GetBulkRequest** has a different PDU format, as shown earlier (refer to “The GetBulkRequest” on page 642).

SnmpMgmtCom (SNMP Management Communication)

Adds the source party ID (srcParty), the destination party ID (dstParty), and the context to the PDU. The context specifies the SNMPv2 context containing the management information referenced by the communication.

**SnmpAuthMsg** This field is used as authentication information from the authentication protocol used by that party. The SnmpAuthMsg is serialized according to ASN.1 BER[[7]](#footnote-7) and can then be encrypted.

SnmpPrivMsg SNMP Private Message

An SNMPv2 private message is an SNMPv2 authenticated management communication that is (possibly) protected from disclosure. A private destination (privDst) is added to address the destination party.

The message is then encapsulated in a normal UDP/IP datagram and sent to the destination across the network.

For further information, refer to the previously mentioned RFCs.

##### SNMPv3

SNMPv3 is described in RFCs 3410 through 3415. SNMPv3 is not a replacement of SNMPv1 or SNMPv2, but rather is an extension to the existing SNMP architecture.

SNMPv3 supports the following:

A new SNMP message format

Authentication for messages

Security for messages

Access control

Continued support for SNMPv2

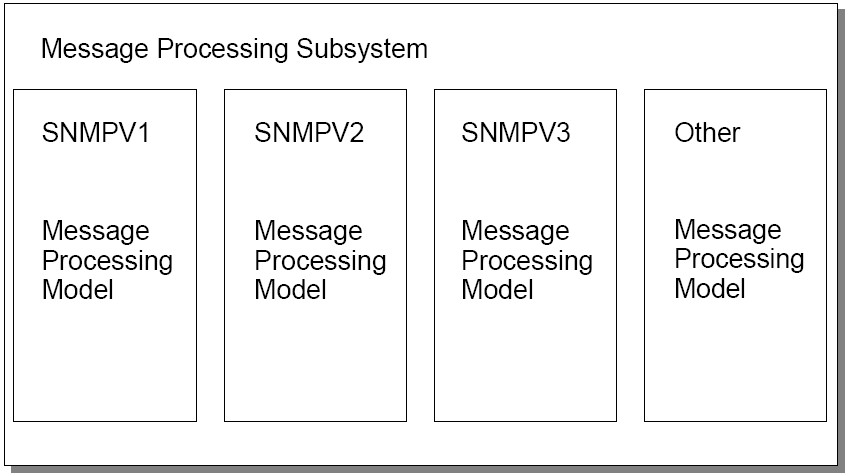
The User-based Security Model (USM), described in RFC 3414, specifies using MD5 and hashing algorithms. This provides data integrity, security, and privacy. There is support for the authentication protocols HMAC-MD5-96,

HMAC-SHA-96, and optional support for the encryption protocol CBC-DES.

The View-based Access Control Model (VACM), defined in RFC 3415, shows how to define views that are subsets of the full MIB tree. Access control on a per-user basis can then be implemented for these views.

Because SNMP has a modular structure, changes to individual modules do not impact the other modules directly. This allows you to easily define SNMPv3 over the existing model. For example, to add a new SNMP message format, it is

sufficient to upgrade the message processing model. Furthermore, because it is needed to support SNMPv1 and SNMPv2 messages as well, it can be achieved by adding the new SNMPv3 message module into the message processing subsystem. Figure 17-8 illustrates this structure.



*Figure 17-8 The SNMP message processing subsystem*

#### 17.1.8 Single authentication and privacy protocol

The authentication protocol provides a mechanism by which SNMPv3 management communications, transmitted by a party, can be reliably identified as having originated from that party.

The privacy protocol provides a mechanism by which SNMPv3 management communications transmitted to a party are protected from disclosure.

Principal threats against which the SNMPv3 security protocol provides protection are:

Modification of information

Masquerade

Message stream modification

Disclosure

The following security services provide protection against these threats:

**Data integrity** Provided by the MD5 message digest algorithm. A 128-bit digest is calculated over the designated portion of a SNMPv3 message and included as part of the message sent to the recipient.

|  |  |
| --- | --- |
| **Data origin authentication** | Provided by prefixing each message with a secret value shared by the originator of that message and its intended recipient before digesting. |
| **Message delay or replay** | Provided by including a time stamp value in each message. |
| **Data confidentiality** | Provided by the symmetric privacy protocol that encrypts an appropriate portion of the message according to a secret key known only to the originator and recipient of the message. This protocol is used in conjunction with the symmetric encryption algorithm, in the cipher block chaining mode, which is part of the Data Encryption Standard (DES). The designated portion of an SNMPv3 message is encrypted and included as part of the message sent to the recipient. |

### 17.2 The NETSTAT utility

The NETSTAT utility is a command available on most platforms that enables a user to list the sockets in use on a system. The information returned by the command is only for the local host, and there is no provision for monitoring remote hosts using this utility.

The most common uses for NETSTAT are:

Determining how many sockets are currently open on a system

Determining what application owns a particular socket

Diagnosing TCP/IP problems

Diagnosing routing problems

The NETSTAT command can be issued with or without parameters. Without parameters, the output generated by the command typically lists all of the active UDP and TCP connections in the system’s connection table. Options can be added to filter the output, or to request additional information. Because NETSTAT is not RFC defined, the specific options employed by different implementations vary. However, there is a common set of options that remain constant among most NETSTAT implementations.

#### 17.2.1 Common NETSTAT options

Common NETSTAT options include:

|  |  |
| --- | --- |
| **-r** / **-route** | Displays the routing table currently used by the TCP/IP |
| **-i** / **-interface**  **-l** / **-listening**  **-a** / **-all**   1. / **-statistics** 2. / **-timer**   **-v** / **-verbose**  **-f** / **-family** | application.  Displays a list of interfaces, and their states.  Displays only sockets on which an application is listening.  Displays all connections (typically, this is the default).  Displays the statistics for each protocol.  Displays timer information.  Displays the output in verbose mode.  Displays the address family of the connections. |

#### 17.2.2 Sample NETSTAT report output

Example 17-1 is a sample of a NETSTAT -all command and illustrates what is usually output by the default implementation of the utility.

*Example 17-1 NETSTAT -all command output*

:\> NETSTAT -a

|  |  |
| --- | --- |
| TCPIP Name: TCPIP | 13:11:51 |
| User Id Conn Local Socket Foreign Socket | State |
| ------- ---- ------------ --------------  FTPD1 00064A00  10.44.36.163..21 10.76.141.227..1780 Establsh | ----- |
| FTPD1 00000039 0.0.0.0..21 0.0.0.0..0 | Listen |
| PSF06A 00064B75 10.44.36.163..1384 10.27.172.17..9100 | SynSent |
| SMTP 00000037 0.0.0.0..25 0.0.0.0..0 | Listen |
| SNMPD 00000031 0.0.0.0..1026 0.0.0.0..0  TCPIP 0006421F  10.44.36.163..23 10.27.204.195..1055 Establsh | Listen |
| SMTP 00000038 0.0.0.0..1028 \*..\* | UDP |
| SNMPD 00000030 0.0.0.0..161 \*..\* | UDP |

The columns of the output, as well as in most implementations, are defined as

|  |  |
| --- | --- |
| follows: |  |
| **User Id** | The application or user that is using the socket. |
| **Conn** | The connection identification number. |
| **Local Socket** | The local IP address and port over which the connection is active. | |
| **Foreign Socket** | The remote IP address and port over which the connection is active. | |
| **State** | The state of the connection. Most implementations use some form of the following values for state: | |

* CloseWait
* Closed
* Established
* FinWait\_1
* FinWait\_2
* LastAck
* Listen
* SynReceived
* SynSent
* TimeWait
* UDP

(Because UDP is a connectionless protocol, they cannot be listed in a particular state. As such, NETSTAT simply indicates that they are UDP sockets.)

Additional information about these states can be found in RFC 0793.

Additionally, Example 17-2 is a sample routing table generated by NETSTAT.

*Example 17-2 Sample routing table*

:\> NETSTAT -r

TCPIP Name: TCPIP 13:25:04

Destination Gateway Flags Refcnt Interface

----------- ------- ----- ------ ---------

Default 10.44.36.129 UGS 001504 INTRF1 Default 10.44.36.129 UGS 000006 INTRF2

10.44.36.128 0.0.0.0 US 000003 INTRF1 10.44.36.128 0.0.0.0 US 000000 INTRF2

10.44.36.129 0.0.0.0 UHS 000000 INTRF1

10.44.36.129 0.0.0.0 UHS 000000 INTRF2

10.44.36.163 0.0.0.0 UH 000000 VIPAL1

10.44.36.164 0.0.0.0 UH 000000 INTRF1

10.44.36.165 0.0.0.0 UH 000000 INTRF2

127.0.0.1 0.0.0.0 UH 000002 LOOPBACK

Again, the columns above are defined as follows:

|  |  |
| --- | --- |
| **Destination** | The route described by the report. |
| **Gateway** | The gateway (if any) used to reach this route. |
| **Flags** | Attributes of the route. Possible values include:   * G: The route uses a gateway. * U: The interface over which the route travels is up. * H: Only a single host can be reached through this route. * D: The route was dynamically created. * M: The route’s table entry was modified by an ICMP redirect message. * !: The route is a reject route; datagrams will be dropped. |
| **Refcnt** | The number of connections using this route. |
| **Interface** | The interface used by the route. |

### 17.3 RFCs relevant to this chapter

The following RFCs provide detailed information about the management protocols and architectures presented throughout this chapter:

[RFC 0793 – Transmission Control Protocol (September 1981)](http://www.ietf.org/rfc/rfc0793.txt)

[RFC 1028 – Simple Gateway Monitoring Protocol (November 1987) Historic](http://www.ietf.org/rfc/rfc1028.txt)

[RFC 1052 – IAB recommendations for the development of Internet network management standards (April 1988)](http://www.ietf.org/rfc/rfc1052.txt)

[RFC 1085 – ISO presentation services on top of TCP/IP based internets (December 1988)](http://www.ietf.org/rfc/rfc1085.txt)

[RFC 1095 – Common Management Information Services and Protocol over TCP/IP (CMOT) (April 1989)](http://www.ietf.org/rfc/rfc1095.txt)

[RFC 1155 – Structure and identification of management information for](http://www.ietf.org/rfc/rfc1155.txt)

[TCP/IP-based internets (May 1990)](http://www.ietf.org/rfc/rfc1155.txt)

[RFC 1156 – Management Information Base for network management of TCP/IP-based internets (May 1990)](http://www.ietf.org/rfc/rfc1156.txt)  [RFC 1157 – Simple Network Management Protocol (SNMP) (May 1990)](http://www.ietf.org/rfc/rfc1157.txt)

[RFC 1189 – Common Management Information Services and Protocol for the](http://www.ietf.org/rfc/rfc1189.txt)

[Internet (CMOT and CMIP) (October 1990)](http://www.ietf.org/rfc/rfc1189.txt)

[RFC 1213 – Management Information Base for Network Management of](http://www.ietf.org/rfc/rfc1213.txt)

[TCP/IP-based internets: MIB-II (March 1991)](http://www.ietf.org/rfc/rfc1213.txt)

[RFC 1215 – Convention for defining traps for use with the SNMP](http://www.ietf.org/rfc/rfc1215.txt)

[(March1991)](http://www.ietf.org/rfc/rfc1215.txt)

[RFC 1239 – Reassignment of experimental MIBs to standard MIBs](http://www.ietf.org/rfc/rfc1239.txt)

[(June](http://www.ietf.org/rfc/rfc1239.txt) [1991)](http://www.ietf.org/rfc/rfc1239.txt)

[RFC 1351 – SNMP Administrative Model (July 1992)](http://www.ietf.org/rfc/rfc1351.txt)

[RFC 1352 – SNMP Security Protocols (July 1992)](http://www.ietf.org/rfc/rfc1352.txt)

[RFC 1441 – Introduction to version 2 of the Internet-standard Network](http://www.ietf.org/rfc/rfc1441.txt)

[Management Framework (April 1993)](http://www.ietf.org/rfc/rfc1441.txt)

[RFC 1592 – Simple Network Management Protocol Distributed Protocol](http://www.ietf.org/rfc/rfc1592.txt)

[Interface Version 2.0 (March 1994)](http://www.ietf.org/rfc/rfc1592.txt)

[RFC 1748 – IEEE 802.5 MIB using SMIv2 (December 1994)](http://www.ietf.org/rfc/rfc1748.txt)

[RFC 1901 – Introduction to Community-based SNMPv2 (January 1996)](http://www.ietf.org/rfc/rfc1901.txt)

[RFC 1909 – An Administrative Infrastructure for SNMPv2 (February 1996)](http://www.ietf.org/rfc/rfc1909.txt)

[RFC 1910 – User-based Security Model for SNMPv2 (February 1996)](http://www.ietf.org/rfc/rfc1910.txt)

[RFC 2578 – Structure of Management Information Version 2 (SMIv2) (April](http://www.ietf.org/rfc/rfc2578.txt) [1999)](http://www.ietf.org/rfc/rfc2578.txt)

[RFC 2579 – Textual Conventions for SMIv2 (April 1999)](http://www.ietf.org/rfc/rfc2579.txt)

[RFC 2580 – Conformance Statements for SMIv2 (April 1999)](http://www.ietf.org/rfc/rfc2580.txt)

[RFC 2742 – Definitions of Managed Objects for Extensible SNMP Agents (January 2000)](http://www.ietf.org/rfc/rfc2742.txt)

[RFC 2863 – The Interfaces Group MIB (June 2000)](http://www.ietf.org/rfc/rfc2863.txt)

[RFC 3410 – Introduction and Applicability Statements for Internet-Standard](http://www.ietf.org/rfc/rfc3410.txt)

[Management Framework (December 2002)](http://www.ietf.org/rfc/rfc3410.txt)

[RFC 3411 – An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks (December 2002)](http://www.ietf.org/rfc/rfc3411.txt)

[RFC 3412 – Message Processing and Dispatching for the Simple Network](http://www.ietf.org/rfc/rfc3412.txt)

[Management Protocol (SNMP) (December 2002)](http://www.ietf.org/rfc/rfc3412.txt)

[RFC 3413 – Simple Network Management Protocol (SNMP) Applications](http://www.ietf.org/rfc/rfc3413.txt)

[(December 2002)](http://www.ietf.org/rfc/rfc3413.txt)

[RFC 3414 – User-based Security Model (USM) for version 3 of the Simple](http://www.ietf.org/rfc/rfc3414.txt)

[Network Management Protocol (SNMPv3) (December 2002)](http://www.ietf.org/rfc/rfc3414.txt)

[RFC 3415 – View-based Access Control Model (VACM) for the Simple](http://www.ietf.org/rfc/rfc3415.txt)

[Network Management Protocol (SNMP) (December 2002)](http://www.ietf.org/rfc/rfc3415.txt)

[RFC 3416 – Version 2 of the Protocol Operations for the Simple Network](http://www.ietf.org/rfc/rfc3416.txt)

[Management Protocol (SNMP) (December 2002)](http://www.ietf.org/rfc/rfc3416.txt)

[RFC 3417 – Transport Mappings for the Simple Network Management](http://www.ietf.org/rfc/rfc3417.txt)

[Protocol (SNMP) (December 2002)](http://www.ietf.org/rfc/rfc3417.txt)

[RFC 3418 – Management Information Base (MIB) for the Simple Network](http://www.ietf.org/rfc/rfc3418.txt)

[Management Protocol (SNMP) (December 2002)](http://www.ietf.org/rfc/rfc3418.txt)

[RFC 3584 – Coexistence between Version 1, Version 2, and Version 3 of the](http://www.ietf.org/rfc/rfc3584.txt)

[Internet-standard Network Management Framework (August 2003)](http://www.ietf.org/rfc/rfc3584.txt)

[RFC 4022 – Management Information Base for the Transmission Control](http://www.ietf.org/rfc/rfc4022.txt)

[Protocol (TCP) (March 2005)](http://www.ietf.org/rfc/rfc4022.txt)

[RFC 4113 – Management Information Base for the User Datagram Protocol (UDP) (June 2005)](http://www.ietf.org/rfc/rfc4113.txt)

[RFC 4293 – Management Information Base for the Internet Protocol (IP) (April 2006)](http://www.ietf.org/rfc/rfc4293.txt)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **22** | |

## Chapter 22. TCP/IP security

This chapter discusses security issues regarding TCP/IP networks and provides an overview of solutions to prevent security exposures or problems before they occur. The field of network security in general and of TCP/IP security in particular is too wide to be dealt within an all encompassing way in this book, so the focus of this chapter is on the most common security exposures and measures to counteract them. Because many, if not all, security solutions are based on cryptographic algorithms, we also provide a brief overview of this topic for the better understanding of concepts presented throughout this chapter.

© Copyright IBM Corp. 1989-2006. All rights reserved.

### 22.1 Security exposures and solutions

This section gives an overview of some of the most common attacks on computer security, and it presents viable solutions to those exposures and lists actual implementations thereof.

#### 22.1.1 Common attacks against security

For thousands of years, people have been guarding the gates to where they store their treasures and assets. Failure to do so usually resulted in being robbed, victimized by society, or even killed. Though things are usually not as dramatic anymore, they can still become very bad. Modern day IT managers have realized that it is equally important to protect their communications networks against intruders and saboteurs from both inside and outside. One does not have to be overly paranoid to find some good reasons as to why this is the case:

Packet sniffing: To gain access to cleartext network data and passwords

Impersonation: To gain unauthorized access to data or to create unauthorized e-mails by impersonating an authorized entity

Denial-of-service: To render network resources non-functional

Replay of messages: To gain access to information and change it in transit

Password cracking: To gain access to information and services that would normally be denied (dictionary attack)

Guessing of keys: To gain access to encrypted data and passwords (brute-force attack)

Viruses: To destroy data

Port scanning: To discover potential available attack points

Though these attacks are not exclusively specific to TCP/IP networks, they must be considered potential threats to anyone who is going to base their network on TCP/IP, which is the most prevalent protocol used. TCP/IP is an open protocol, and therefore, hackers find easy prey by exploiting vulnerabilities using the previous methods.

#### 22.1.2 Solutions to network security problems

Network owners need to try to protect themselves with the same zealousness that intruders use to search for a way to get into the network. To that end, we provide some solutions to effectively defend a network from an attack, specifically against the attacks mentioned earlier. It has to be noted that any of these solutions only solve a single (or a very limited number) of security problems. Therefore, consider a combination of several such solutions to guarantee a certain level of safety and security. These solutions include:

Encryption: To protect data and passwords

Authentication by digital signatures and certificates: To verify who is sending data over the network

Authorization: To prevent improper access

Integrity checking and message authentication codes: To protect against improper alteration of messages

Non-repudiation: To make sure that an action cannot be denied by the person who performed it

One-time passwords and two-way random number handshakes: To mutually authenticate parties of a conversation

Frequent key refresh, strong keys, and prevention of deriving future keys: To protect against breaking of keys (cryptanalysis)

Address concealment: To protect against denial-of-service attacks

Disable unnecessary services: To minimize the number of attack points

Table 22-1 matches common problems and security exposures to the previous solutions.

*Table 22-1 Security exposures and protections*

|  |  |
| --- | --- |
| **Problem/exposure** | **Remedy** |
| How to prevent a packet sniffer from reading messages? | Encrypt messages, typically using a shared secret key (secret keys offer a tremendous performance advantage over public/private keys). |
| How to distribute the keys in a secure way? | Use a different encryption technique, typically public/private key. |
| How to prevent keys from becoming stale, and how to protect against guessing of future keys by cracking current keys? | Refresh keys frequently and do not derive new keys from old ones (use perfect forward secrecy). |
| How to prevent retransmission of messages by an impostor (replay attack)? | Use sequence numbers (time stamps are usually unreliable for security purposes). |
| How to ensure that a message has not been altered in transit? | Use message digests (hash or one-way functions). |
| How to ensure that the message digest has not also been compromised? | Use digital signatures by encrypting the message  digest with a secret or private key (origin authentication, non-repudiation). |
| **Problem/exposure** | **Remedy** |
| How to ensure that the message and signature    originated from the desired partner? | Use two-way handshakes involving encrypted random numbers (mutual authentication). |
| How to ensure that handshakes are exchanged with the right partners (man-in-the-middle attack)? | Use digital certificates (binding of public keys to permanent identities). |
| How to prevent improper use of services by otherwise properly authenticated users? | Use a multilayer access control model. |
| How to protect against viruses? | Restrict access to outside resources; run anti-virus software on every server and workstation that has contact to outside data, and update that software frequently. |
| How to protect against unwanted or malicious messages (denial of service attacks)? | Restrict access to internal network using filters, firewalls, proxies, packet authentication, conceal internal address and name structure, and so on. |
| How to minimize the number of attack points? | Close all unnecessary services. Use encryption and encapsulation to run many services over a smaller number of ports. |

In general, keep your network tight toward the outside, but also keep a watchful eye on the inside because most attacks are mounted from inside a corporate network.

#### 22.1.3 Implementations of security solutions

The following protocols and systems are commonly used to provide various degrees of security services in a computer network. They are discussed at length throughout the rest of this chapter.

IP filtering

Network Address Translation (NAT)

IP Security Architecture (IPSec)

SOCKS

Secure Shell (SSH)

Secure Sockets Layer (SSL)

Application proxies

Firewalls

Kerberos and other authentication systems (AAA servers)

Secure Electronic Transactions (SET)

Figure 22-1 illustrates where these security solutions fit within the TCP/IP layers.

*Figure 22-1 Security solutions in the TCP/IP layers*

Applications

TCP/UDP

)

Transport

(

IP

)

(

Internetwork

Network Interface

(

Data Link

)

-

S-MIME

-

Kerberos

Proxies

-

SET

-

)

IPSec (ISAKMP

-

-

SOCKS

SSL, TLS

-

-

IPSec (AH, ESP

)

Packet filtering

-

-

Tunneling protocols

-

CHAP, PAP, MS-CHAP

Table 22-2 summarizes the characteristics of some of the security solutions mentioned earlier and compares them to each other. This should help anyone who needs to devise a security strategy to determine what combination of solutions achieves a desired level of protection.

*Table 22-2 Security solution implementations: A comparison*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Access control** | **Encryption** | **Authentication** | **Integrity checking** | **Perfect forward security** | **Address concealment** | **Session monitoring** |
| IP filtering | Y | N | N | N | N | N | N |
| NAT | Y | N | N | N | N | Y | Y  (connection) |
| IPSec | Y | Y (packet) | Y (packet) | Y (packet) | Y | Y | N |
| SOCKS | Y | N | Y (client/ user) | N | N | Y | Y  (connection) |
| SSL | Y | Y (data) | Y (system/ user) | y |  | n | y |
|  | **Access control** | **Encryption** | **Authentication** | **Integrity checking** | **Perfect forward security** | **Address concealment** | **Session monitoring** |
| Application proxy | Y | Normally no | Y (user) | Y | Normally no | Y | Y  (connection and data) |
| AAA servers | y (user) | N | Y (user) | N | N | N | N |

An overall security solution can, in most cases, only be provided by a combination of the listed options. Your particular security requirements need to be specified in a security policy and should be, for example, enforced by using firewalls and validated by using security health checking tools and vulnerability scanners.

#### 22.1.4 Network security policy

An organization's overall security policy must be determined according to security and business needs analysis and based on security best practices. Because a firewall relates to network security only, a firewall has little value unless the overall security policy is properly defined.

A network security policy defines those services that will be explicitly allowed or denied, how these services will be used, and the exceptions to these rules. Every rule in the network security policy should be implemented on a firewall, remote access server (RAS), or both. Generally, a firewall uses one of the following methods.

##### Everything not specifically permitted is denied

This approach blocks all traffic between two networks except for those services and applications that are permitted. Therefore, each desired service and application is implemented one by one. No service or application that might be a potential hole on the firewall is permitted. This is the most secure method, denying services and applications unless explicitly allowed by the administrator. However, from the point of users, it might be more restrictive and less convenient.

##### Everything not specifically denied is permitted

This approach allows all traffic between two networks except for those services and applications that are denied. Therefore, each untrusted or potentially harmful service or application is denied one by one. Although this is a flexible and convenient method for the users, it can potentially cause some serious security problems, especially as new applications are introduced into the environment.

Remote access servers should provide authentication of users and should ideally also provide for limiting certain users to certain systems and networks within the corporate intranet (authorization). Remote access servers must also determine if a user is considered roaming (can connect from multiple remote locations) or stationary (can connect only from a single remote location), and if the server should use callback for particular users after they are properly authenticated.

Generally, anonymous access should at best, be granted to servers in a demilitarized zone (DMZ, see “Screened subnet firewall (demilitarized zone)” on page 808). All services within a corporate intranet should require at least password authentication and appropriate access control. Direct access from the outside should always be authenticated and accounted.

### 22.2 A short introduction to cryptography

The purpose of this chapter is to introduce the terminology and give a brief overview of the major cryptographic concepts that relate to TCP/IP security implementations. The information presented here only scratches the surface. Some issues are left open or not mentioned at all.

#### 22.2.1 Terminology

Let us start with defining some very basic concepts.

##### Cryptography

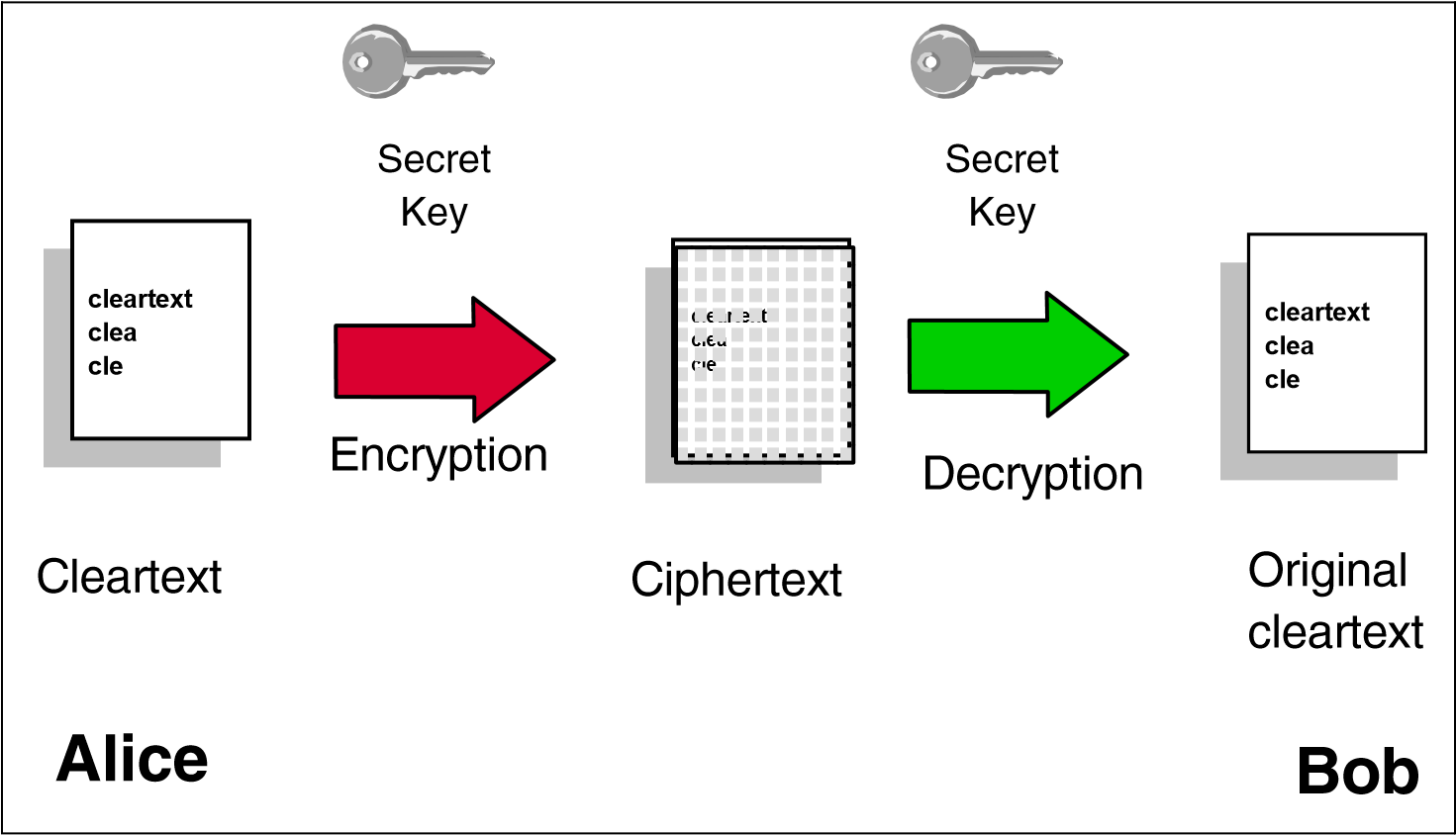
Put simply, *cryptography* is the science of altering the appearance of data in an effort to keep data and data communications secure. To achieve this goal, techniques such as *encryption*, *decryption*, and *authentication* are used. With the recent advances in this field, the frontiers of cryptography have become blurred. Every procedure consisting of transforming data based on methods that are difficult to reverse can be considered cryptography. The key factor to strong cryptography is the difficulty of reverse engineering. You might be amazed to know that simple methods, such as password-scrambled word processor documents or compressed archives, can be broken in a matter of minutes by a hacker using an ordinary PC. *Strong* cryptography means that the computational effort needed to retrieve your cleartext messages without knowing the proper keys makes the retrieval infeasible. In this context, infeasible means something like this: If all the computers in the world were assigned to the problem, they would have to work tens of thousands of years until the solution was found. The process of retrieval is called *cryptanalysis*. An attempted cryptanalysis is an *attack*.

###### Encryption and decryption: Cryptographic algorithms

*Encryption* is the transformation of a cleartext message into an unreadable form in order to hide its meaning. The opposite transformation, which retrieves the original cleartext, is the *decryption*. The mathematical function used for encryption and decryption is the *cryptographic algorithm* or *cipher*.

The security of a cipher might be based entirely on keeping its functionality a secret, in which case it is a *restricted cipher*. There are many drawbacks to restricted ciphers. It is very difficult to keep an algorithm a secret when it is used by many people. If it is incorporated in a commercial product, it is only a matter of time and money before it is reverse engineered. For these reasons, the currently used algorithms are *keyed*, that is, the encryption and decryption makes use of a parameter, known as the *key*. The key can be chosen from a set of possible values, called the *keyspace*. The keyspace usually is huge, the bigger the better. The security of these algorithms rely entirely on the key, not on their internal secrets. In fact, the algorithms themselves are usually public and are extensively analyzed for possible weaknesses. The principle of keyed ciphers is shown in Figure 22-2.

**Note:** Do not trust new, unknown, or unpublished algorithms.



*Figure 22-2 Keyed encryption and decryption*

**Note:** It is common in cryptographic literature to denote the first participant in a protocol as Alice and the second one as Bob. They are the “crypto couple.”

###### Authentication, integrity, and non-repudiation

Encryption provides confidentiality to messages. When communicating over an untrusted medium, such as the Internet, you might also need, in addition to confidentiality:

Authentication: A method for verifying that the sender of a message is really who he or she claims to be. Any intruder masquerading as someone else is detected by authentication.

Integrity checking: A method for verifying that a message has not been altered along the communication path. Any tampered message sent by an intruder is detected by an integrity check. As a side effect, communication errors are also detected.

Non-repudiation: The possibility to prove that the sender has really sent the message. When algorithms providing non-repudiation are used, the sender is not able to later deny the fact that he or she sent the message in question.

#### 22.2.2 Symmetric or secret-key algorithms

Symmetric algorithms are keyed algorithms where the decryption key is the same as the encryption key. These are conventional cryptographic algorithms where the sender and the receiver must agree on the key *before* any secured communication can take place between them. Figure 22-2 on page 778 illustrates a symmetric algorithm. There are two types of symmetric algorithms: *block algorithms*, which operate on the cleartext in blocks of bits, and *stream algorithms*, which operate on a single bit (or byte) of cleartext at a time.

Block ciphers are used in several modes. Electronic Codebook Mode (ECB) is the simplest; each block of cleartext is encrypted independently. Given a block length of 64 bits, there are 264 possible input cleartext blocks, each of them corresponding to exactly one out of 264 possible ciphertext blocks. An intruder might construct a codebook with known cleartext-ciphertext pairs and mount an attack. Because of this vulnerability, the Cipher Block Chaining (CBC) mode is often used, where the result of the encryption of the previous block is used in the encryption of the current block, thus each ciphertext block is dependent not just on the corresponding plaintext block, but on all previous plaintext blocks.

The algorithms often make use of initialization vectors (IVs). These are variables independent of the keys and are good for setting up the initial state of the algorithms.

A well-known block algorithm is the Data Encryption Standard (DES), which was a worldwide standard cipher developed by IBM. DES operates on 64-bit blocks and has a key length of 56 bits, often expressed as a 64-bit number, with every eighth bit serving as parity bit. From this key, 16 subkeys are derived, which are used in the 16 rounds of the algorithm.

DES produces ciphertexts the same length as the cleartext and the decryption algorithm is exactly the same as the encryption, the only difference being the subkey schedule. These properties make it very suitable for hardware implementations.

DES is becoming obsolete (its origins date back to the early 1970s) and is no longer sufficient as a standard. The most practical attack against it is *brute-force* decryption, with all possible keys, looking for a meaningful result. The problem with DES is the key length. Given enough time and computers, a brute-force attack against the 56-bit key might be feasible. That is why newer modes of DES, called triple-DES, or 3DES, have become popular. With triple-DES, the original DES algorithm is applied in three rounds, with two or three different keys.

Today, DES is still widely used in many forms but has been replaced as a standard by the Advanced Encryption Standard (AES), which is based on a block cipher named Rijndael. The Rijndael cipher is based on a block cipher Square. The Rijndael key length and block size are both variable and can be set to 128, 192, or 256 bits, but the official block size is 128 bits.

Another, block algorithm is the International Data Encryption Algorithm (IDEA).

This cipher uses 64-bit blocks and 128-bit keys. It was developed in the early

1990s and aimed to replace DES. It is cryptographically strong and faster than DES. The most significant use of IDEA is in the freeware secure e-mail package Pretty Good Privacy (PGP).

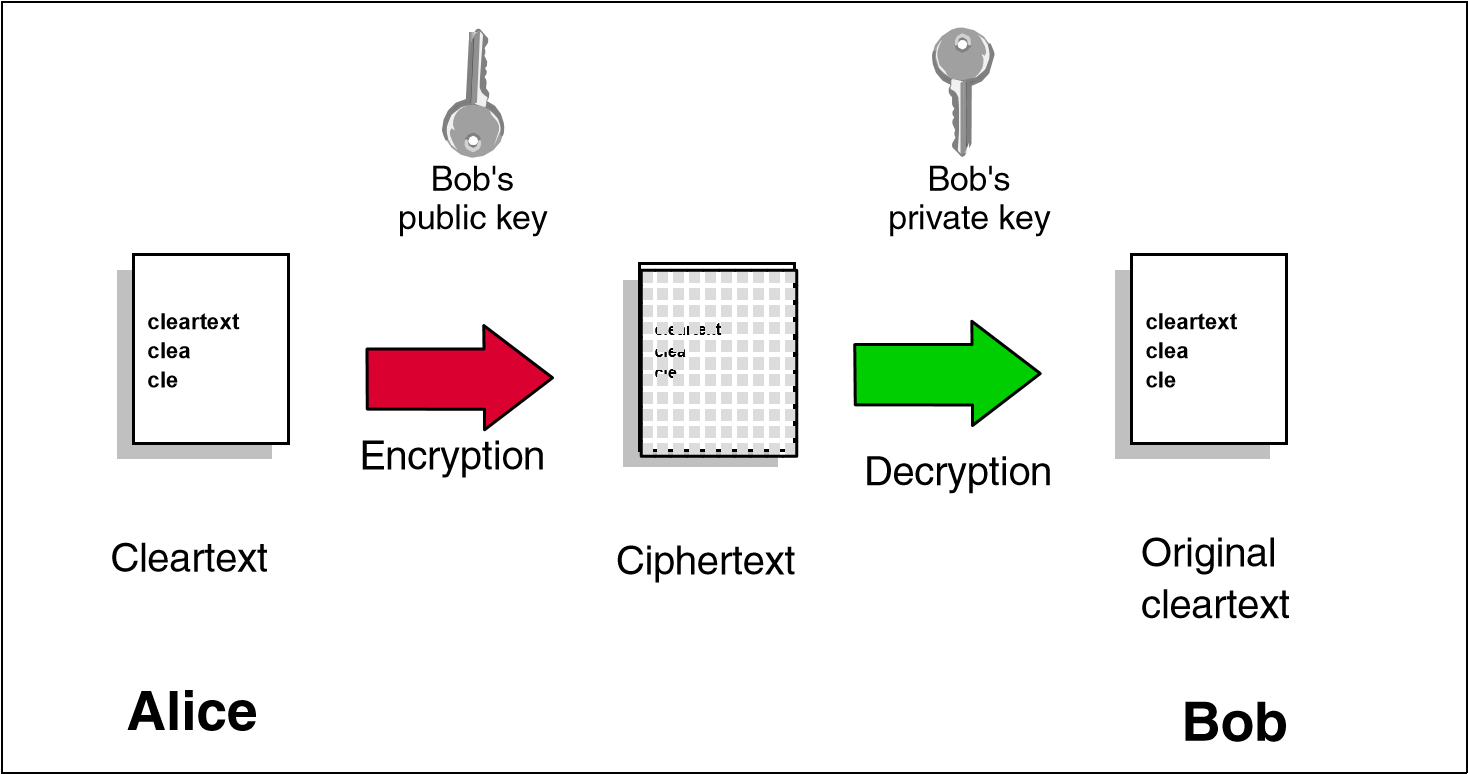
An example of a stream algorithm is A5, which is used to encrypt digital cellular telephony traffic in the GSM standard, widely used in Europe.

The advantage of the symmetric algorithms is their efficiency. They can be easily implemented in hardware. A major disadvantage is the difficulty of key management. A secure way of exchanging the keys must exist, which is often very hard to implement.

#### 22.2.3 Asymmetric or public key algorithms

These algorithms address the major drawback of symmetric ciphers, the requirement of the secure key-exchange channel. The idea is that two different keys should be used: a public key, which, as the name implies, is known to

everyone, and a private key, which is to be kept in tight security by the owner. The private key cannot be determined from the public key. A cleartext encrypted with the public key can only be decrypted with the corresponding private key. A cleartext encrypted with the private key can only be decrypted with the corresponding public key. Therefore, if someone sends a message encrypted with the recipient's public key, it can be read by the intended recipient only. The process is shown in Figure 22-3, where Alice sends an encrypted message to Bob.

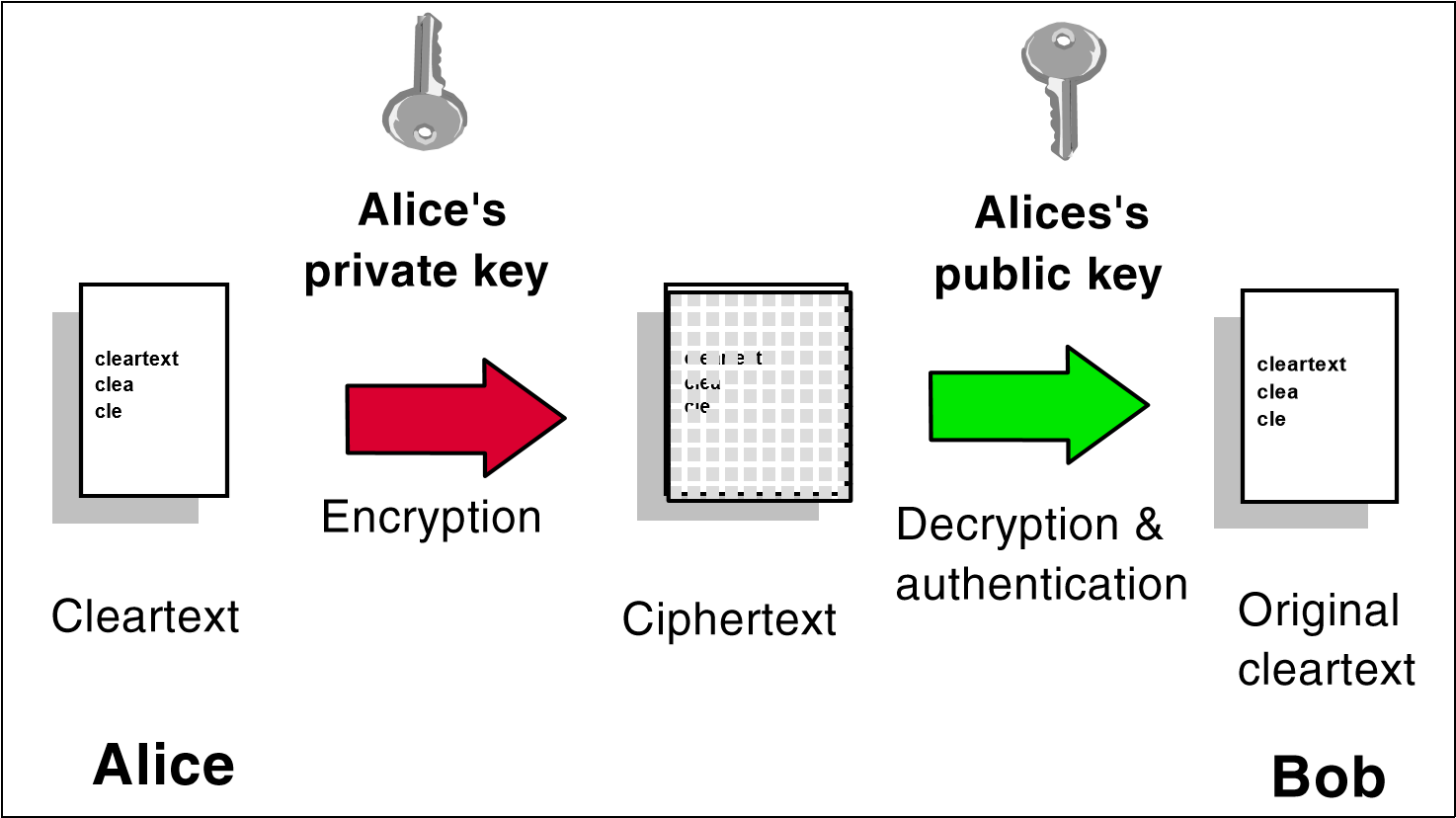


*Figure 22-3 Encryption using the recipient's public key*

As the public key is available to anyone, privacy is assured without the need for a secure key-exchange channel. Parties that want to communicate retrieve each other's public key.

##### Authentication and non-repudiation

An interesting property of the public key algorithms is that they can provide authentication. The private key is used for encryption. Because anyone has access to the corresponding public key and can decrypt the message, this provides no privacy. However, it authenticates the message. If you can successfully decrypt it with the claimed sender's public key, the message has been encrypted with the corresponding private key, which is known by the real sender only. Therefore, the sender's identity is verified. Encryption with the private key is used in *digital signatures*. The principle is shown in Figure 22-4 on page 782. Alice encrypts her message with her private key (“signs” it), in order to enable Bob to verify the authenticity of the message.



*Figure 22-4 Authentication by encrypting with a private key*

Going a step further, encrypting with the private key gives non-repudiation, too. The mere existence of such an encrypted message testifies that the originator has really sent it, because only he or she could have used the private key to generate the message. Additionally, if a time stamp is included, the exact date and time can also be proven. There are protocols involving trusted third parties that prevent the sender from using phony time stamps.

##### Examples of public key algorithms

Algorithms based on public keys can be used for a variety of purposes. Two common applications are:

Encryption (see “RSA public key algorithm” on page 783).

Generation of shared keys for use with symmetric key algorithms (see “Diffie-Hellman key exchange” on page 784).

The most popular public key algorithm is the *de facto* standard RSA, named after its three inventors: Ron Rivest, Adi Shamir, and Leonard Adleman. The security of RSA relies on the difficult problem of factoring large numbers. The public and private keys are functions of two very large (200 digits or even more) prime numbers. Given the public key and the ciphertext, an attack is successful if it can factor the product of the two primes. RSA has resisted many years of extensive attacks. As computing power grows, keeping RSA secure is a matter of increasing the key length (unlike DES, where the key length is fixed).

Another public key algorithm, the very first ever invented, is *Diffie-Hellman*. This is a key exchange algorithm; that is, it is used for securely establishing a shared secret over an insecure channel. The communicating parties exchange public information from which they derive a key. An eavesdropper cannot reconstruct the key from the information that went through the insecure channel. More precisely, the reconstruction is computationally infeasible. The security of Diffie-Hellman relies on the difficulty of calculating discrete logarithms in finite fields. After the shared secret has been established, it can then be used to derive keys for use with symmetric key algorithms such as DES.

Diffie-Hellman makes the secure derivation of a shared secret key possible, but it does not authenticate the parties. For authentication, another public key algorithm must be used, such as RSA.

Unfortunately, public key algorithms, while providing for easier key management, privacy, authentication, and non-repudiation, also have some disadvantages. The most important one is that they are slow and difficult to implement in hardware. For example, RSA is 100 to 10,000 times slower than DES, depending on implementation. Because of this, public key algorithms generally are not used for bulk encryption. Their most important use is key exchange and authentication. Another notable disadvantage is that they are susceptible to certain cryptanalytic attacks to which symmetric algorithms are resistant.

Therefore, a good cryptographic system (*cryptosystem*) makes use of both worlds. It uses public key algorithms in the session establishment phase for authentication and key exchange, and then a symmetric one for encrypting the consequent messages.

For the interested reader, we give more detailed information of the two most important asymmetric algorithms, which involve modular arithmetic. An arithmetic operation modulo m means that the result of that operation is divided by m and the remainder is taken. For example: 3 \* 6 mod 4 = 2, since 3 \* 6 = 18 and dividing 18 by 4 gives us 2 as the remainder.

###### RSA public key algorithm

RSA is used in the ISAKMP/Oakley framework as one of the possible authentication methods. The principle of the RSA algorithm is as follows:

1. Take two large primes, p and q.
2. Find their product n = pq; n is called the modulus.
3. Choose a number, e, less than n and relatively prime to (p-1)(q-1), which means that e and (p-1)(q-1) have no common factor other than 1.
4. Find its inverse, d mod (p-1)(q-1), which means that ed = 1 mod (p-1)(q-1).

e and d are called the public and private exponents, respectively. The public key is the pair (n,e); the private key is d. The factors p and q must be kept secret or destroyed.

A simplified example of RSA encryption is:

1. Suppose Alice wants to send a private message, m, to Bob. Alice creates the ciphertext c by exponentiating:

c = me mod n

Where e and n are Bob's public key.

1. Alice sends c to Bob.
2. To decrypt, Bob exponentiates:

m = cd mod n

And recovers the original message; the relationship between e and d ensures that Bob correctly recovers m. Because only Bob knows d, only Bob can decrypt the ciphertext.

A simplified example of RSA authentication is:

1. Suppose Alice wants to send a signed message, m, to Bob. Alice creates a digital signature s by exponentiating:

s = md mod n

Where d and n belong to Alice's private key.

1. She sends s and m to Bob.
2. To verify the signature, Bob exponentiates and checks if the result, compares to m:

m = se mod n

Where e and n belong to Alice's public key.

###### Diffie-Hellman key exchange

The Diffie-Hellman key exchange is a crucial component of the ISAKMP/Oakley framework. In the earliest phase of a key negotiation session, there is no secure channel in place. The parties derive shared secret keys using the Diffie-Hellman algorithm. These keys will be used in the next steps of the key negotiation protocol. The following steps outline the algorithm:

1. The parties (Alice and Bob) share two public values, a modulus m and an integer g. m is a large prime number.
2. Alice generates a large random number a and computes: X = ga mod m
3. Bob generates a large random number b and computes: Y = gbmod m
4. Alice sends X to Bob.
5. Bob computes:

K1 = Xb mod m

1. Bob sends Y to Alice.
2. Alice computes: K2 = Ya mod m

Both K1 and K2 are equal to gab mod m. This is the shared secret key. No one is able to generate this value without knowing a or b. The security of the exchange is based on the fact that is extremely difficult to inverse the exponentiation performed by the parties. (In other words, to calculate discrete logarithms in finite fields of size m.) Similar to RSA, advances in adversary computing power can be countered by choosing larger initial values, in this case a larger modulus m.

See 22.4.5, “Internet Key Exchange (IKE) protocol” on page 829 for more details about how ISAKMP/Oakley uses Diffie-Hellman exchanges.

#### 22.2.4 Hash functions

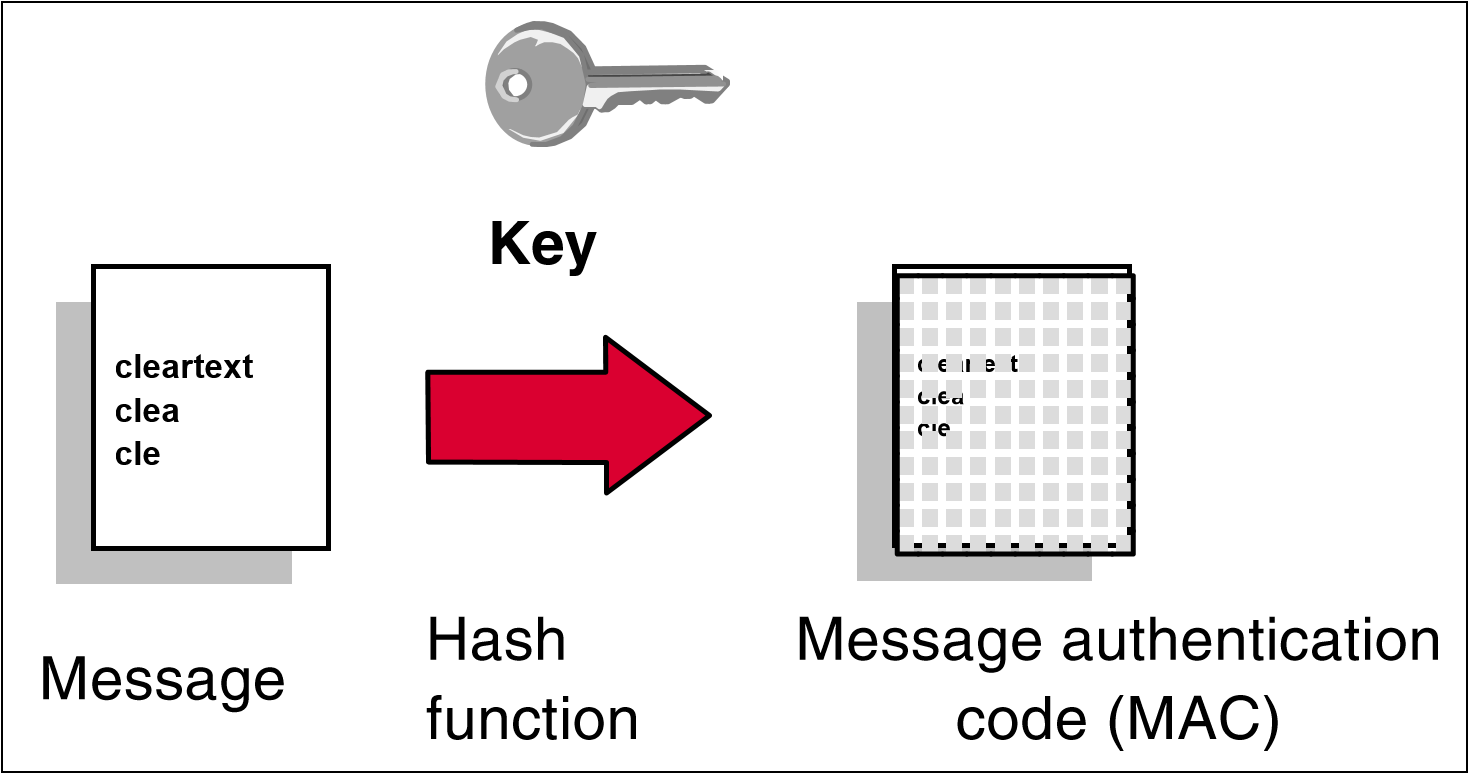
Hash functions (also called message digests) are fundamental to cryptography. A hash function is a function that takes variable-length input data and produces fixed length output data (the hash value), which can be regarded as the “fingerprint” of the input. That is, if the hashes of two messages match, it is highly probable that the messages are the same.

Cryptographically useful hash functions must be *one-way*, which means that they should be easy to compute, but infeasible to reverse. An everyday example of a one-way function is mashing a potato; it is easy to do, but once mashed, reconstructing the original potato is rather difficult.

A good hash function must also be *collision-resistant*. It must be hard to find two different inputs that hash to the same value. Because any hash function maps an input set to a smaller output set, theoretically it is possible to find collisions. The

point is to provide a unique digital “fingerprint” of the message that identifies it with high confidence, much like a real fingerprint identifying a person.

A hash function that takes a key as a second input parameter and its output depends on both the message and the key is called a *message authentication code (MAC)*, as shown in Figure 22-5.



*Figure 22-5 Generating a message authentication code (MAC)*

Put simply, if you encrypt a hash, it becomes a MAC. If you add a secret key to a message, and then hash the concatenation, the result is a MAC. Both symmetric and asymmetric algorithms can be used to generate MACs.

Hash functions are primarily used to assure integrity and authentication:

The sender calculates the hash of the message and appends it to the message.

The recipient calculates the hash of the received message and then compares the result with the transmitted hash.

If the hashes match, the message was not tampered with.

If the encryption key (symmetric or asymmetric) is only known by a trusted sender, a successful MAC decryption indicates that the claimed and actual senders are identical.

See Figure 22-6 for an illustration of the procedure. The Message\* and MAC\* notations reflect the fact that the message might have been altered while crossing the untrusted channel.

**Key**

Message

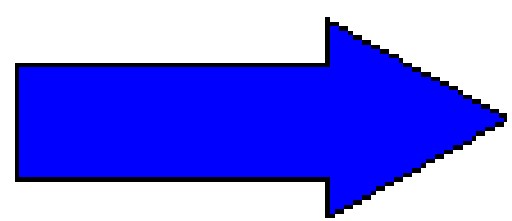
MAC

+



Hash

function



Message\*



Original

MAC



MAC\*

**Untrusted**

**channel**

**Alice**

Match?

Compare

Accept

Reject

NO

YES

**Bob**



*Figure 22-6 Checking integrity and authenticity with MAC*

You might argue that the same result can be obtained with any kind of encryption, because if an intruder modifies an encrypted message, the decryption will result in nonsense, thus tampering can be detected. The answer is that many times only integrity, authentication, or both are needed, maybe with encryption on some of the fields of the message. Also encryption is very processor-intensive. Examples include the personal banking machine networks, where only the PINs are encrypted. However, MACs are widely used. Encrypting all the messages in their entirety would not yield noticeable benefits and performance would dramatically decrease.

The encryption of a hash with the private key is called a *digital signature*. It can be thought of as a special MAC. Using digital signatures instead of encrypting the whole message with the private key leads to considerable performance gains and a remarkable new property. The authentication part can be decoupled from the document itself. This property is used, for example, in the Secure Electronic

Transactions (SET) protocol.

The encryption of a secret key with a public key is called a *digital envelope*. This is a common technique used to distribute secret keys for symmetric algorithms.

##### Examples of hash functions

The most widely used hash functions are MD5 and Secure Hash Algorithm 1 (SHA-1). MD5 was designed by Ron Rivest (co-inventor of RSA). SHA-1 is largely inspired from MD5 and was designed by the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA) for use with the Digital Signature Standard (DSS). MD5 produces a 128-bit hash, while SHA-1 produces a 160-bit hash. Both functions encode the message length in their output. SHA-1 is regarded as more secure, because of the larger hashes it produces.

Neither MD5 nor SHA-1 takes a key as an input parameter. Therefore, in their original implementation, they cannot be used for MAC calculation. However, for this purpose, it is easy to concatenate a key with the input data and apply the function to the result.

**Note:** In practice, for example, in IPSec, more sophisticated schemes are often used.

###### Keyed MD5 and keyed SHA-1

Using MD5 and SHA-1 in keyed mode is simple. The shared secret key and the data to be protected are both input to the hash algorithm. In the following IPSec example, the datagram is combined with the key, and the output hash value is placed in the Authentication Data field of the AH header, as shown in Figure 22-7.

*Figure 22-7 Keyed MD5 processing*

IP Hdr

AH

Payload

(

Pad

)

MD5

Shared key

(128

bits

)

(128)

Keyed SHA-1 operates in the same way, the only difference being the larger 160-bit hash value.

###### HMAC-MD5-96 and HMAC-SHA-1-96

A stronger method is the Hashed Message Authentication Code (HMAC), proposed by IBM. HMAC itself is not a hash function, rather a cryptographically strong way to use a specific hash function for MAC calculation.

To show how HMAC works, consider MD5 as an example. The base function is applied twice in succession. In the first round, the input to MD5 is the shared secret key and the datagram. The 128-bit output hash value and the key are input again to the hash function in the second round. The left-most 96 bits of the resulting hash value are used as the MAC for the datagram. See Figure 22-8 for an illustration.

*Figure 22-8 HMAC-MD5-96 processing*

IP Hdr

AH

Payload

)

(

Pad

MD5

Shared key

(128

)

bits

(128)

MD5

(128)

(96)

HMAC-SHA-1-96 operates in the same way, except that the intermediary results are 160 bits long.

###### Digital Signature Standard (DSS)

As mentioned previously, a hash value encrypted with the private key is called a *digital signature* and is illustrated in Figure 22-9.



**Private key**

Message



Encryption



Message

digest

(

hash

)

Hash

function

Digital

signature

*Figure 22-9 Generating a digital signature*

One authentication method that can be used with ISAKMP/Oakley is DSS, which was selected by NIST and NSA to be the digital authentication standard of the U.S. government. The standard describes the Digital Signature Algorithm (DSA) used to sign and verify signatures of message digests produced with SHA-1.

The following steps provide a brief description of DSA:

1. Choose a large prime number, p, usually between 512 and 1024 bits long.
2. Find a prime factor q of (p-1), 160 bits long.
3. Compute:

g=h(p-1)/q mod p

Where h is a number less than (p-1) and the following is true:

h(p-1)/q>1

1. Choose another number x, less than q, as the sender's private key.
2. Compute: y=gx mod p

And use that as the sender's public key. The pair (x,y) is sometimes referred to as the long-term key pair.

1. The sender signs the message as follows:
   1. Generate a random number, k, less than q.
   2. Compute:

r=(gk mod p) mod q s=(k-1(SHA1(m)+xr)) mod q

The pair (k,r) is sometimes referred to as the per-session key pair, and the signature is represented by the pair (r,s).

1. The sender sends (m,r,s).
2. The receiver verifies the signature as follows:

Compute:

w=s-1 mod q u1=(SHA1(m)\*w) mod q u2=(rw) mod q v=((gu1yu2) mod p) mod q

1. If v=r, the signature is verified.

#### 22.2.5 Digital certificates and certification authorities

As mentioned in “Authentication and non-repudiation” on page 781, with public key cryptography, the parties retrieve each other's public key. However, there are security exposures here. An intruder can replace some real public keys with his or her own public key, and then mount a so-called *man-in-the-middle attack*.

For example, the intruder places himself between Alice and Bob. He can trick Bob by sending him one of his own public keys as though it were Alice's. The same applies to Alice. She thinks she uses Bob's public key, but she actually uses the intruder's. So, the clever intruder can decrypt the confidential traffic between the two and remain undetected. For example, a message sent by Alice and encrypted with “Bob's” public key arrives at the intruder, who decrypts it, learns its content, then re-encrypts it with Bob's real public key. Bob has no way to realize that Alice is using a phony public key.

An intruder can also use impersonation, claiming to be somebody else, for example, an online shopping mall, fooling innocent shoppers.

The solution to these serious threats is the *digital certificate*. A digital certificate is a file that binds an identity to the associated public key. This binding is validated by a trusted third party, the *certification authority (CA)*. A digital certificate is signed with the private key of the certification authority, so it can be authenticated. It is only issued after a verification of the applicant. Apart from the public key and identification, a digital certificate usually contains other information too, such as:

Date of issue

Expiration date

Miscellaneous information from the issuing CA (for example, serial number)

**Note:** There is an international standard in place for digital certificates: The ISO X.509 protocols.

The parties retrieve each other's digital certificate and authenticate it using the public key of the issuing certification authority. They have confidence that the public keys are real, because a trusted third party vouches for them. This helps protect against both man-in-the-middle and impersonation attacks.

It is easy to imagine that one CA cannot cover all needs. What happens when Bob's certificate is issued by a CA unknown to Alice? Can she trust that unknown authority? Well, this is entirely her decision, but to make life easier, CAs can form a hierarchy, often referred to as the *trust chain*. Each member in the chain has a certificate signed by its superior authority. The higher the CA is in the chain, the tighter security procedures are in place. The root CA is trusted by everyone and its private key is top secret.

Alice can traverse the chain upward until she finds a CA that she trusts. The traversal consists of verifying the subordinate CA's public key and identity using the certificate issued to it by the superior CA.

When a trusted CA is found in the chain, Alice is assured that Bob's issuing CA is trustworthy. This is all about delegation of trust. We trust your identity card if somebody who we trust signs it. And if the signer is unknown to us, we can go upward and see who signs for the signer, and so on.

An implementation of this concept is in the SET protocol, where the major credit card brands operate their own CA hierarchies that converge to a common root. Lotus® Notes® authentication, as another example, is also based on certificates, and it can be implemented using hierarchical trust chains. PGP also uses a similar approach, but its trust chain is based on persons and it is a distributed Web rather than a strict hierarchical tree.

#### 22.2.6 Random-number generators

An important component of a cryptosystem is the random-number generator. Many times random session keys and random initialization variables (often referred to as initialization vectors) are generated. For example, DES requires an explicit initialization vector and Diffie-Hellman relies on picking random numbers which serve as input for the key derivation.

The quality, that is the randomness of these generators, is more important than you might think. The ordinary random function provided with most programming language libraries is good enough for games, but not for cryptography. Those random-number generators are rather predictable; if you rely on them, be prepared for happy cryptanalysts finding interesting correlations in your encrypted output.

The fundamental problem faced by the random-number generators is that the computers are ultimately deterministic machines, so real random sequences cannot be produced. As John von Neumann ironically said: “Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.” That's why the term *pseudorandom generator* is more appropriate.

Cryptographically strong pseudorandom generators must be unpredictable. It must be computationally infeasible to determine the next random bit, even with total knowledge of the generator.

A common practical solution for pseudorandom generators is to use hash functions. This approach provides sufficient randomness and it can be efficiently implemented. Military-grade generators use specialized devices that exploit the inherent randomness in physical phenomena. An interesting solution can be found in the PGP software. The initial seed of the pseudorandom generator is derived from measuring the time elapsed between the keystrokes of the user.

#### 22.2.7 Export/import restrictions on cryptography

U.S. export regulations changed on January 14, 2000 with the publication of new regulations in the Federal Register. These regulations make it easier for United States companies and individuals to export strong encryption. Some of the changes include:

“Retail” encryption products are widely exportable to all but certain “terrorist” nations though still subject to a government review and reporting requirements.

Non-retail products are also exportable, subject to similar requirements, to most non-government users. Encryption products with less than 64-bits are freely exportable.

Some non-proprietary source code is exportable to most countries after notice to the government.

In September 1998, the White House announced further liberalization of U.S. export restrictions on cryptographic material and key recovery requirements, which can be summarized as follows:

The key recovery requirement for export of 56-bit DES and equivalent products is eliminated. This includes products that use 1024-bit asymmetric key exchanges together with 56-bit symmetric key algorithms.

Export of unlimited strength encryption (for example, 3DES) under license exceptions (with or without key recovery) is now broadened to include others besides the financial industry for 45 countries. This includes subsidiaries of U.S firms, insurance, health and medical (excluding biochemical and pharmaceutical manufacturers), and online merchants for the purpose of securing online transactions (excluding distributors of items considered munitions).

For the latter, recoverable products will be granted exceptions world wide (excluding terrorist countries) without requiring a review of foreign key recovery agents.

Export of recoverable products will be granted to most commercial firms, for a broad range of countries, in the major commercial markets (excluding items on the U.S. munitions list).

Export licenses to end users may be granted on a case-by-case basis.

More information can be obtained from the U.S. Department of Commerce: <http://www.bis.doc.gov/Encryption/Default.htm>

According to the law in France, any product capable of enciphering/deciphering user data must be granted a license from the French government before being marketed. Clients need to be authorized to use such products on a case-by-case basis. In reality, two major and useful exceptions exist:

Routinely, licenses are granted that allow banks to use DES products on a global basis (no case-by-case authorization required).

Routinely, global licenses are granted that allow anybody to use weak encryption (RC2/RC4 with 40-bit keys).

### 22.3 Firewalls

Firewalls have significant functions in an organization's security policy. Therefore, it is important to understand these functions and apply them to the network properly. This chapter explains the firewall concept, network security, firewall components, and firewall examples.

#### 22.3.1 Firewall concept

A firewall is a system (or group of systems) that enforces a security policy between a secure internal network and an untrusted network such as the

Internet. Firewalls tend to be seen as a protection between the Internet and a

private network. But generally speaking, a firewall should be considered as a means to divide the world into two or more networks: one or more secure networks and one or more non-secure networks. See Figure 22-10.



Secure internal

network

Company A

Untrusted

network

(

Internet

)

Secure internal

network

Company B

Firewall

Firewall

*Figure 22-10 A firewall illustration*

A firewall can be a PC, a router, a midrange, a mainframe, a UNIX workstation, or a combination of these that determines which information or services can be accessed from the outside and who is permitted to use the information and services from outside. Generally, a firewall is installed at the point where the secure internal network and untrusted external network meet, which is also known as a *choke point*.

In order to understand how a firewall works, consider the network to be a building to which access must be controlled. The building has a lobby as the only entry point. In this lobby, receptionists welcome visitors, security guards watch visitors, video cameras record visitor actions, and badge readers authenticate visitors who enter the building.

Although these procedures can work well to control access to the building, if an unauthorized person succeeds in entering, there is no way to protect the building against this intruder's actions. However, if the intruder's movements are monitored, it can be possible to detect any suspicious activity.

Similarly, a firewall is designed to protect the information resources of the organization by controlling the access between the internal secure network and the untrusted external network (see Figure 22-11 on page 796). However, it is important to note that even if the firewall is designed to permit the trusted data to pass through, deny the vulnerable services, and prevent the internal network from outside attacks, a newly created attack can penetrate the firewall at any time. The network administrator must examine all logs and alarms generated by the firewall on a regular basis. Otherwise, it is generally not possible to protect the internal network from outside attacks.

Client1

Client2

Internet

Untrusted network

organization.com

Secure

network

priv

ate.organization.com

Production server

**Stop**

*Figure 22-11 A firewall controls traffic between the secure network and the Internet*

#### 22.3.2 Components of a firewall system

As mentioned previously, a firewall can be a PC, a midrange, a mainframe, a UNIX workstation, a router, or combination of these. Depending on the requirements, a firewall can consist of one or more of the following functional components:

Packet-filtering router

Application-level gateway (proxy)

Circuit-level gateway

Each of these components has different functions and shortcomings. Generally, in order to build an effective firewall, these components are used together.

##### Packet-filtering router

Most of the time, packet-filtering is accomplished by using a router that can forward packets according to filtering rules. When a packet arrives at the packet-filtering router, the router extracts certain information from the packet header and makes decisions according to the filter rules as to whether the packet will pass through or be discarded (see Figure 22-12). The following information can be extracted from the packet header:

Source IP address

Destination IP address

TCP/UDP source port

TCP/UDP destination port

ICMP message type

Encapsulated protocol information (TCP, UDP, ICMP, or IP tunnel)

The packet-filtering rules are based on the network security policy (see 22.1.4, “Network security policy” on page 776). Therefore, packet-filtering is done by using these rules as input. When determining the filtering rules, outside attacks must be taken into consideration, as well as service level restrictions and source/destination level restrictions.

*Figure 22-12 Packet-filtering router*

Filter

Client1

Client2

Client3

Client4

Trusted network

Untrusted network

###### Service level filtering

Because most services use well-known TCP/UDP port numbers, it is possible to allow or deny services by using related port information in the filter. For example, an FTP server listens for connections on TCP port 21, and for a non-passive mode client, makes outbound data connections from port 20. Therefore, to permit FTP connections to pass through to a secure network, the router can be configured to permit packets that contain 20 and 21 as the TCP port in its header. However, there are some applications, such as NFS, that use RPC and use different ports for each connection. Allowing these kind of services might cause security problems.

###### Source/destination level filtering

The packet-filtering rules allow a router to permit or deny a packet according to the destination or the source information in the packet header. In most cases, if a service is available, only that particular server is permitted to outside users. Other packets that have another destination or no destination information in their headers are discarded.

###### Advanced filtering

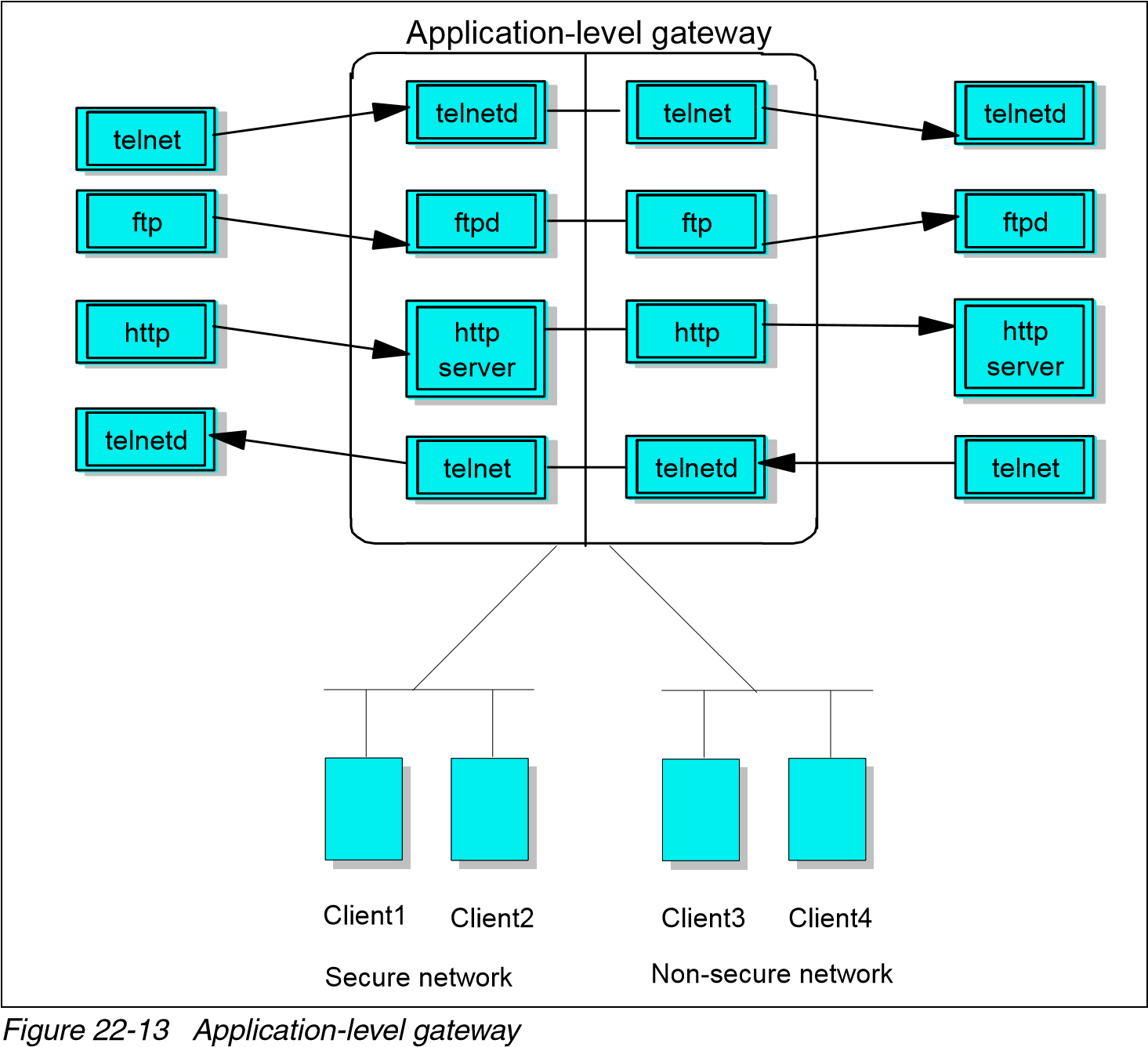
As mentioned previously (see 22.1.1, “Common attacks against security” on page 772), there are different types of attacks that threaten the privacy and network security. Some of them can be discarded by using advanced filtering rules such as checking IP options, fragment offset, and so on.

###### Packet-filtering limitations

Packet-filtering rules are sometimes very complex. When there are exceptions to existing rules, it becomes much more complex. Although there are a few testing utilities available, it is still possible to leave some holes in the network security. Packet filters do not provide an absolute protection for a network. For some cases, it might be necessary to restrict some set of information (for example, a command) from passing through to the internal secure network. It is not possible to control the data with packet filters because they are not capable of understanding the contents of a particular service. For this purpose, an application-level control is required.

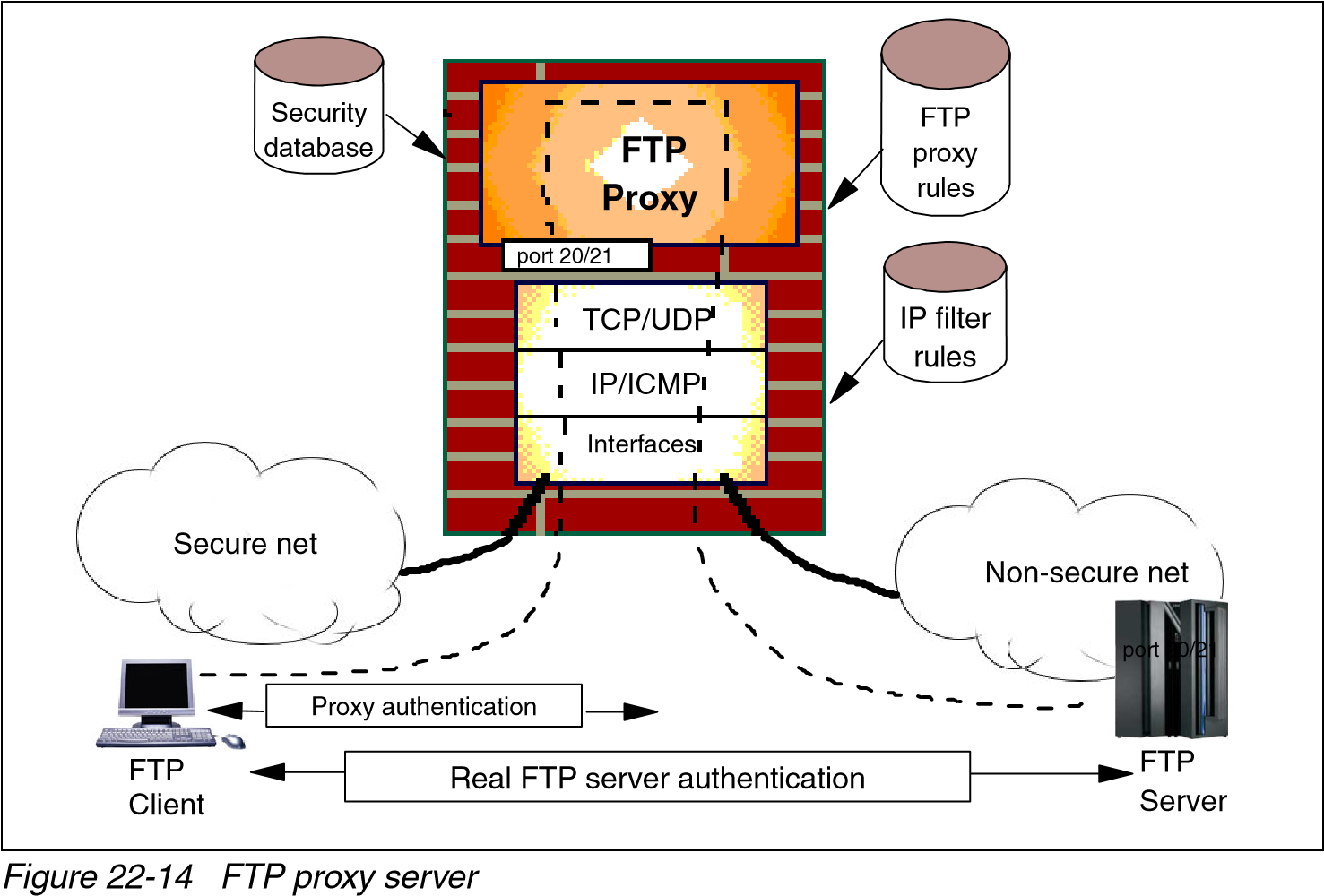
##### Application-level gateway (proxy)

An application-level gateway is often referred to as a *proxy*. An application-level gateway provides higher-level control on the traffic between two networks in that the contents of a particular service can be monitored and filtered according to the network security policy. Therefore, for any desired application, the corresponding proxy code must be installed on the gateway in order to manage that specific service passing through the gateway (see Figure 22-13).



A proxy acts as a server to the client and as a client to the destination server. A virtual connection is established between the client and the destination server. Though the proxy seems to be *transparent* from the point of view of the client and the server, the proxy is capable of monitoring and filtering any specific type of data, such as commands, before sending it to the destination. For example, an FTP server is permitted to be accessed from outside. In order to protect the server from any possible attacks, the FTP proxy in the firewall can be configured to deny PUT and MPUT commands.

A proxy server is an application-specific relay server that runs on the host that connects a secure and a non-secure network. The purpose of a proxy server is to control exchange of data between the two networks at an application level instead of an IP level. By using a proxy server, it is possible to disable IP routing between the secure and the non-secure network for the application protocol the proxy server is able to handle, but still be able to exchange data between the networks by relaying it in the proxy server. Figure 22-14 shows an FTP proxy server.



Note that in order for any client to be able to access the proxy server, the client software must be specifically modified. In other words, the client and server software must support the proxy connection. In the previous example, the FTP client must authenticate itself to the proxy first. If it is successfully authenticated, the FTP session starts based on the proxy restrictions. Most proxy server implementations use more sophisticated authentication methods such as security ID cards. This mechanism generates a unique key that is not reusable for another connection. Two security ID cards are supported by IBM Firewall: the SecureNet card from Axent and the SecureID card from Security Dynamics.

Compared with IP filtering, application-level gateways provide much more comprehensive logging based on the application data of the connections. For example, an HTTP proxy can log the URLs visited by users. Another feature of application-level gateways is that they can use strong user authentication. For example, when using FTP and Telnet services from the unsecure network, users can be forced to authenticate themselves to the proxy. Figure 22-15 shows a proxy server TCP segment flow example.

*Figure 22-15 Proxy server TCP segment flow*

Client host

Proxy server host

Server host

Secure

network

Non-secure

network

epn: Ephemeral port number n

sss: Server port number sss

ssp: Proxy server port number

Proxy Server

Client

Real

Server

ep1

ssp

ep2

sss

###### Application-level gateway limitations

A disadvantage of application-level gateways is that, in order to achieve a connection through a proxy server, the client software must be changed to support that proxy service. This can sometimes be achieved by some modifications in user behavior rather than software modification. For example, to connect to a Telnet server over a proxy, the user usually has to be authenticated by the proxy server then by the destination Telnet server. This requires two user steps to make a connection rather than one. However, a modified Telnet client can make the proxy server transparent to the user by specifying the destination host rather than proxy server in the Telnet command.

###### An example: FTP proxy server

Most of the time, in order to use the FTP proxy server, users must have a valid user ID and password. On UNIX systems, users also must be defined as users of the UNIX system.

FTP can be used in one of two modes:

Normal mode

Passive mode

In normal mode, the FTP client first connects to the FTP server port 21 to establish a control connection. When data transfer is required (for example, as the result of a DIR, GET, or PUT command), the client sends a PORT command to the server instructing the server to establish a data connection from the server's data port (port 20) to a specified ephemeral port number on the client host.

In an FTP proxy server situation, normal mode means that we have to allow inbound TCP connections from the non-secure network to the FTP proxy host. Notice in Figure 22-16 how a connection is established from the FTP server port 20 in the non-secure network to the FTP proxy server's ephemeral port number. To allow this to happen, IP filtering rules are used that allow inbound connection requests from port 20 to an ephemeral port number on the FTP proxy host. This is normally not an IP filter rule. It is sometimes better to add a custom filter rule configuration, because it would allow a cracker to run a program on port 20 and scan all the port numbers above 1023, which, in its simplest form, might result in a denial-of-service situation. Some firewalls handle this correctly by building a table of outgoing FTP requests and matching up the corresponding incoming data transfer request.

Proxy Server

FTP proxy

FTP

client

Client host

FTP proxy server host

FTP

server

Server host

Secure

network

Non-secure

network

epn: Ephemeral port number n

ep1

21

21

ep2

ep4

20

20

ep3

Incoming

*Figure 22-16 Normal mode FTP proxy*

A much more firewall-friendly mode is the passive mode of operation, as shown in Figure 22-17. This mode has been dubbed a firewall-friendly FTP and is described in RFC 1579 – Firewall-Friendly FTP.

Proxy Server

FTP proxy

FTP

client

Client host

FTP proxy server host

FTP

server

Server host

Secure

network

Non-secure

network

epn: Ephemeral port number n

ep1

21

21

ep2

ep4

ep6

ep5

ep3

Outbound

*Figure 22-17 Passive mode FTP proxy (firewall-friendly FTP)*

In passive mode, the FTP client again establishes a control connection to the server's port 21. When data transfer has to start, the client sends a PASV command to the server. The server responds with a port number for the client to contact, in order to establish the data connection, and the client then initiates the data connection.

In this setup, to establish connections to both port 21 and any ephemeral port number in the non-secure network, an ephemeral port number is used on the FTP proxy host. Here, we do not need a rule that allows inbound connections to ephemeral port numbers, because we are now connecting outward.

##### Circuit-level gateway

A circuit-level gateway relays TCP connections and does not provide any extra packet processing or filtering. Some circuit-level gateways can handle UDP packets. A circuit-level gateway can be said to be a special type of application-level gateway. This is because the application-level gateway can be configured to pass all information after the user is authenticated, just as the circuit-level gateway (see Figure 22-18 on page 805). However, in practice, there are significant differences between them, such as:

Circuit-level gateways can handle several TCP/IP applications, as well as UDP applications, without any extra modifications on the client side for each application. Therefore, this makes circuit-level gateways a good choice to satisfy user requirements.

Circuit-level gateways do not provide packet processing or filtering. Therefore, a circuit-level gateway is generally referred to as a *transparent* gateway.

Application-level gateways have a lack of support for UDP.

Circuit-level gateways are often used for outbound connections, while application-level gateways (proxy) are used for both inbound and outbound connections. Generally, when using both types combined, circuit-level gateways can be used for outbound connections and application-level gateways can be used for inbound connections to satisfy both security and user requirements.

Circuit-level gateways can sometimes handle incoming UDP packets or TCP connections. However, a client on the secure side must inform the gateway to expect such packets. SOCKS v5 has this capability.

A well-known example of a circuit-level gateway is SOCKS (refer to 22.5,

“SOCKS” on page 846 for more information). Because the data that flows over SOCKS is not monitored or filtered, a security problem can arise. To minimize security problems, trusted services and resources need to be used on the outside network (untrusted network).

*Figure 22-18 Circuit-level gateway*

Client1

Client3

Client2

Client4

Non-secure network

SOCKS

server

SOCKS-enabled

client program

Unmodified

server program

Secure network

#### 22.3.3 Types of firewalls

A firewall consists of one or more software elements that run on one or more hosts. The hosts can be general purpose computer systems or specialized such as routers. There are four important examples of firewalls. These are:

Packet-filtering firewall

Dual-homed gateway firewall

Screened host firewall

Screened subnet firewall

##### Packet-filtering firewall

The packet-filtering firewall is commonly used because it is inexpensive (see Figure 22-19 on page 806). The firewall is just a router sitting between the external network and the internal secure network. Packet-filtering rules are defined to permit or deny traffic (see “Packet-filtering router” on page 797).

Generally, a packet-filtering firewall is configured to deny any service if it is not explicitly permitted. Although this approach prevents some potential attacks, the firewall is still open to attacks that result from improper filter rule configurations.

Internal

DNS and

Mail server

Router

Packet

filter

Client1

Client2



Internet

Untrusted network

Secure network

organization.com

*Figure 22-19 Packet-filtering firewall*

The filter will allow some of the hosts on the internal network to be directly accessed from the external network. Such hosts need their own authorization mechanism and need to be updated regularly in case of any attacks.

##### Dual-homed gateway firewall

A dual-homed host has at least two network interfaces and therefore at least two IP addresses. IP forwarding is disabled in the firewall, thus all IP traffic between the two interfaces is broken at the firewall (see Figure 22-20 on page 807). Therefore, there is no way for a packet to pass the firewall except through the related proxy or SOCKS service. Unlike the packet-filtering firewalls, dual-homed gateway firewalls make sure that any attack that comes from an unknown service will be blocked. A dual-homed gateway implements the method in which everything not specifically permitted is denied.

*Figure 22-20 Dual-homed firewall*

Internal

DNS and

mail server

Client1

Client2

Packet

filter

Secure network

private.organization.com

Proxy

servers

SOCKS

server

External

DNS

Router



Internet

Untrusted network

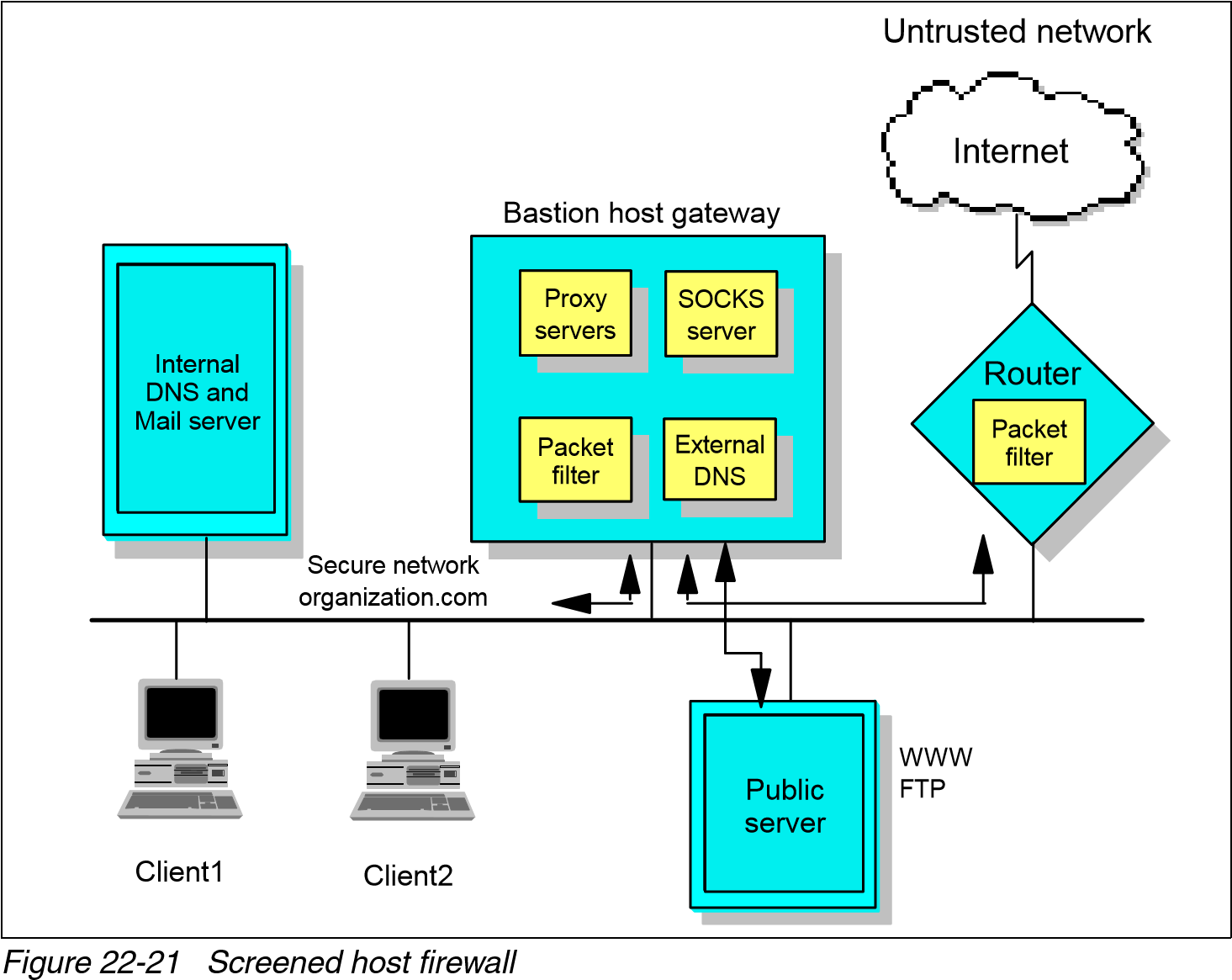
Non-secure network

organization.com

If an information server (such as a Web or FTP server) needs to be located to give access to both inside and outside users, it can either be installed inside the protected network or it can be installed between the firewall and the router, which is relatively insecure. If it is installed beyond the firewall, the firewall must have the related proxy services to give access to the information server from inside the secure network. If the information server is installed between the firewall and the router, the router must be capable of packet filtering and configured accordingly. This type of firewall is called a screened host firewall and discussed in the following section.

##### Screened host firewall

This type of firewall consists of a packet-filtering router and an application-level gateway. The host containing the application-level gateway is known as a bastion host. The router is configured to forward all untrusted traffic to the bastion host and in some cases also to the information server (see Figure 22-21 on page 808). Because the internal network is on the same subnet as the bastion host, the security policy can allow internal users to access outside networks directly or force them to use proxy services to access the outside network. This can be achieved by configuring the router filter rules so that the router only accepts outbound traffic originating from the bastion host.

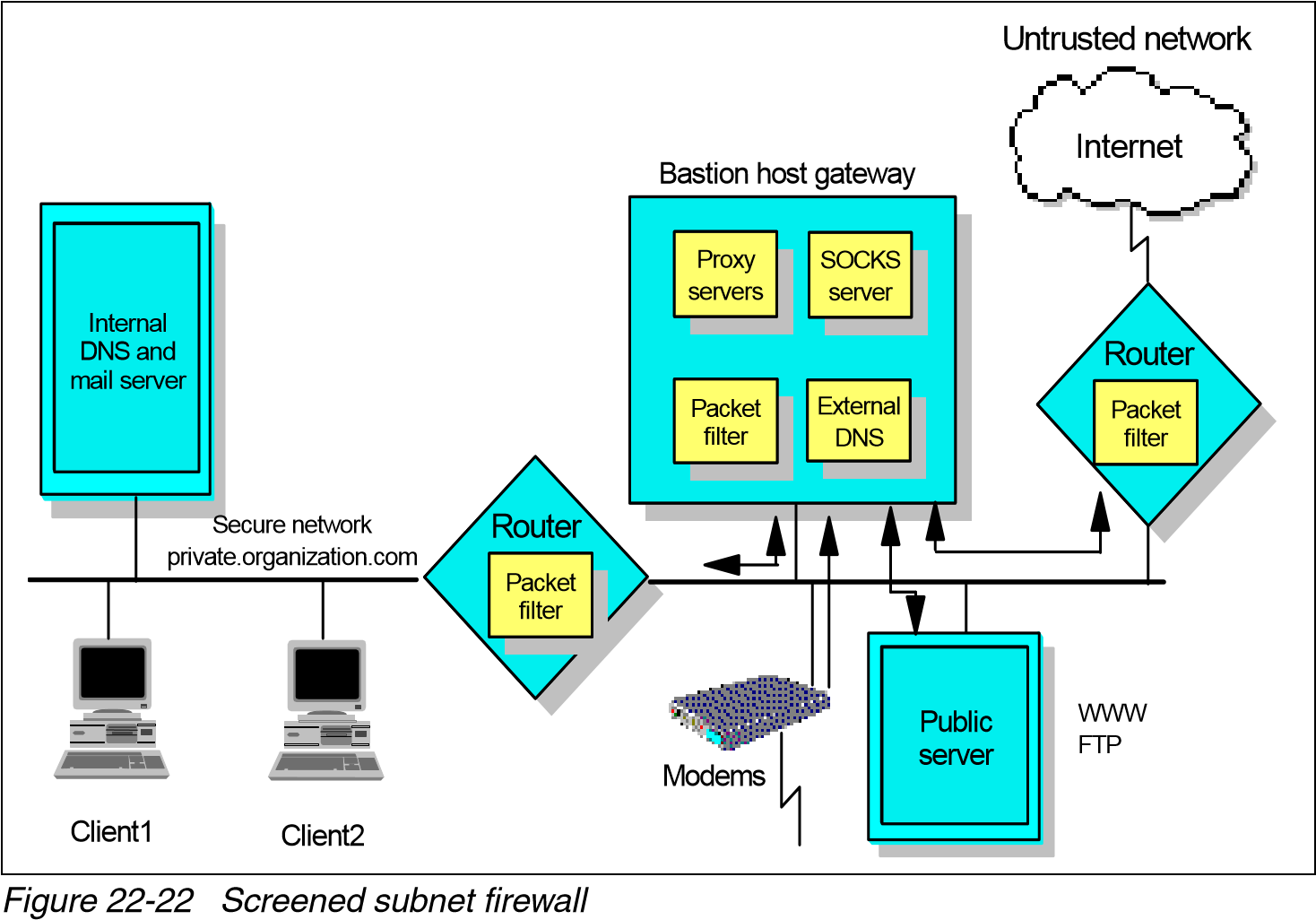


This configuration allows an information server to be placed between the router and the bastion host. Again, the security policy determines whether the information server will be accessed directly by either outside users or internal users, or if it will be accessed through the bastion host. If strong security is needed, traffic from both the internal network to the information server and from outside to the information server can go through the bastion host.

In this configuration, the bastion host can be a standard host or, if a more secure firewall system is needed, it can be a dual-homed host. In this case, all internal traffic to the information server and to the outside through the router is automatically forced to pass the proxy server on the dual-homed host. The bastion host is then the only system that can be accessed from the outside. No one should be permitted to log on to the bastion host; otherwise, an intruder might log on the system and change the configuration to bypass the firewall.

##### Screened subnet firewall (demilitarized zone)

This type of firewall consists of two packet-filtering routers and a bastion host. Screened subnet firewalls provide the highest level security among the different firewall types (see Figure 22-22 on page 809). This is achieved by creating a demilitarized zone (DMZ) between the external and internal network so that the outer router only permits access from the outside to the bastion host (possibly to the information server) and the inner router only permits access from the internal network to the bastion host. The routers force all inbound and outbound traffic through the bastion host. This provides strong security because an intruder has to penetrate three separate systems to reach the internal network.



One of the significant benefits of the DMZ is that because the routers force the systems on both external and internal networks to use the bastion host, there is no need for the bastion host to be a dual-homed host. This provides much faster throughput than achieved by a dual-homed host. Of course, this is complicated and some security problems might be caused by improper router configurations.

### 22.4 IP Security Architecture (IPSec)

This section examines, in detail, the IPSec framework and its three main components, Authentication Header (AH), Encapsulated Security Payload (ESP), and Internet Key Exchange (IKE). We discuss the header formats, the specific cryptographic features, and the different modes of application.

IPSec adds integrity checking, authentication, encryption, and replay protection to IP packets. It is used for end-to-end security and also for creating secure tunnels between gateways.

IPSec was designed for interoperability. When correctly implemented, it does not affect networks and hosts that do not support it. IPSec is independent of the current cryptographic algorithms; it can accommodate new ones as they become available. It works both with IPv4 and IPv6. In fact, IPSec is a mandatory component of IPv6.

IPSec uses state-of-the-art cryptographic algorithms. The specific implementation of an algorithm for use by an IPSec protocol is often called a *transform*. For example, the DES algorithm used by ESP is called the ESP DES-CBC transform. The transforms, like the protocols, are published in the RFCs.

#### 22.4.1 Concepts

Two major IPSec concepts need to be clarified: Security Associations and tunneling. We describe these concepts in the following sections.

##### Security Associations

The concept of a Security Association (SA) is fundamental to IPSec. An SA is a unidirectional (simplex) logical connection between two IPSec systems, uniquely identified by the following triple:

<Security Parameter Index, IP destination address, security protocol>

The definition of the members is as follows:

Security parameter index (SPI)

This is a 32-bit value used to identify different SAs with the same destination address and security protocol. The SPI is carried in the header of the security protocol (AH or ESP). The SPI has only local significance, as defined by the creator of the SA. SPI values in the range 1 to 255 are reserved by the Internet Assigned Numbers Authority (IANA). The SPI value of 0 must be used for local implementation-specific purposes only. RFC 2406 states that a value of 0 must not be transmitted. Generally, the SPI is selected by the destination system during SA establishment.

IP destination address

This address can be a unicast, broadcast, or multicast IP address. However, currently SA management mechanisms are defined only for unicast addresses.

Security protocol

This can be either AH or ESP.

An SA can be in either of two modes, transport or tunnel, depending on the mode of the protocol in that SA. You can find the explanation of these protocol modes later in this chapter.

SAs are simplex, thus, for bidirectional communication between two IPSec systems, there must be two SAs defined, one in each direction.

A single SA gives security services to the traffic carried by it either by using AH or ESP, but not both. In other words, for a connection that needs to be protected by both AH and ESP, two SAs must be defined for each direction. In this case, the set of SAs that define the connection is referred to as an *SA bundle*. The SAs in the bundle do not have to terminate at the same endpoint. For example, a mobile host can use an AH SA between itself and a firewall and a nested ESP SA that extends to a host behind the firewall.

An IPSec implementation maintains two databases related to SAs:

Security Policy Database (SPD)

The Security Policy Database specifies what security services are to be offered to the IP traffic, depending on factors such as source, destination, whether it is inbound, outbound, and so on. It contains an ordered list of policy entries, separate for inbound and outbound traffic. These entries might specify that some traffic must bypass the IPSec processing, some must be discarded, and the rest must be processed by the IPSec module. Entries in this database are similar to firewall rules or packet filters.

Security Association Database (SAD)

The Security Association Database contains parameter information about each SA, such as AH or ESP algorithms and keys, sequence numbers, protocol mode, and SA lifetime. For outbound processing, an SPD entry points to an entry in the SAD. That is, the SPD determines which SA is to be used for a given packet. For inbound processing, the SAD is consulted to determine how the packet must be processed.

**Note:** The user interface of an IPSec implementation usually hides or presents these databases in a friendlier way.

##### Tunneling

Tunneling or encapsulation is a common technique in packet-switched networks. It consists of wrapping a packet in a new one. That is, a new header is attached to the original packet. The entire original packet becomes the payload of the new one, as shown in Figure 22-23.

*Figure 22-23 IP tunneling*

New IP header

IP header

Payload

Original (encapsulated) datagram is

the payload for the new IP header

In general, tunneling is used to carry traffic of one protocol over a network that does not support that protocol directly. For example, NetBIOS or IPX can be encapsulated in IP to carry it over a TCP/IP WAN link. In the case of IPSec, IP is tunneled through IP for a slightly different purpose: To provide total protection, including the header of the encapsulated packet. If the encapsulated packet is encrypted, an intruder cannot figure out, for example, the destination address of that packet. (Without tunneling, the intruder could.) The internal structure of a private network can be concealed in this way.

Tunneling requires intermediate processing of the original packet while en-route. The destination specified in the outer header, usually an IPSec firewall or router, receives the tunneled packet, extracts the original packet, and sends it to the ultimate destination. The processing cost is compensated by the extra security.

A notable advantage of IP tunneling is the possibility to exchange packets with private IP addresses between two intranets over the public Internet, which requires globally unique addresses. Because the encapsulated header is not processed by the Internet routers, only the endpoints of the tunnel (the gateways) need to have globally assigned addresses; the hosts in the intranets behind them can be assigned private addresses (for example, 10.x.x.x). Because globally unique IP addresses are becoming a scarce resource, this interconnection method gains importance.

**Note:** IPSec tunneling is modeled after RFC 2003 – IP Encapsulation within IP. It was originally designed for Mobile IP, an architecture that allows a mobile host to keep its home IP address even if attached to remote or foreign subnets. See 7.1, “Mobile IP overview” on page 276.

#### 22.4.2 Authentication Header (AH)

AH is used to provide integrity and authentication to IP datagrams. Replay protection is also possible. Although its usage is optional, the replay protection service must be implemented by any IPSec-compliant system. The services are connectionless, that is, they work on a per-packet basis. AH is used in two modes, transport mode and tunnel mode.

AH authenticates as much of the IP datagram as possible. In transport mode, some fields in the IP header change en-route and their value cannot be predicted by the receiver. These fields are called *mutable* and are not protected by AH. The mutable IPv4 fields are:

Type of service (TOS)

Flags

Fragment offset

Time to live (TTL)

Header checksum

When protection of these fields is required, tunneling must be used. The payload of the IP packet is considered immutable and is always protected by AH.

AH is identified by protocol number 51, assigned by the IANA. The protocol header (IPv4, IPv6, or extension) immediately preceding the AH contains this value in its protocol (IPv4) or Next header (IPv6, extension) field.

AH processing is applied only to non-fragmented IP packets. However, an IP packet with AH applied can be fragmented by intermediate routers. In this case, the destination first reassembles the packet and then applies AH processing to it. If an IP packet that appears to be a fragment (offset field is non-zero, or the More Fragments bit is set) is input to AH processing, it is discarded. This prevents the so-called *overlapping fragment attack*, which misuses the fragment reassembly algorithm in order to create forged packets and force them through a firewall.

Packets that fail authentication are discarded and never delivered to upper layers. This mode of operation greatly reduces the chances of successful denial-of-service attacks, which aim to block the communication of a host or gateway by flooding it with bogus packets.

##### AH format

The AH format is described in RFC 2402. Figure 22-24 shows the position of the Authentication Header fields in the IP packet.

*Figure 22-24 AH format*

IP Hdr

AH

Payload

Next header

Payld length

Reserved

Security parameter index (SPI)

Sequence number

Authentication data (variable size)

Integrity check value

)

(

32

bits

The fields are as follows:

|  |  |
| --- | --- |
| **Next header** | The next header *t* is an 8-bit field that identifies the type of what follows. The value of this field is chosen from the set of IP protocol numbers defined in the most recent *Assigned Numbers* RFC from the Internet Assigned Numbers Authority (IANA). In other words, the IP header protocol field is set to 51, and the value that would have gone in the protocol field goes in the AH next header field. |
| **Payload length** | This field is 8 bits long and contains the length of the AH header expressed in 32-bit words, minus 2. It does not relate to the actual payload length of the IP packet as a whole. If default options are used, the value is 4 (three  32-bit fixed words plus three 32-bit words of authentication data minus two). |
| **Reserved** | This field is reserved for future use. Its length is 16 bits and it is set to zero. |

###### Security parameter index (SPI)

This field is 32 bits in length. See “Security Associations” on page 810 for a definition.

**Sequence number** This 32-bit field is a monotonically increasing counter, which is used for replay protection. Replay protection is optional; however, this field is mandatory. The sender always includes this field, and it is at the discretion of the receiver to process it or not. At the establishment of an SA, the sequence number is initialized to zero. The first packet transmitted using the SA has a sequence number of 1. Sequence numbers are not allowed to repeat. Therefore, the maximum number of IP packets that can be transmitted on any given SA is 232-1. After the highest sequence number is used, a new SA, and consequently a new key, are established. Anti-replay is enabled at the sender by default. If upon SA establishment the receiver chooses not to use it, the sender need not be concerned with the value in this field anymore.

**Notes:** Typically, the anti-replay mechanism is not used with manual key management. The original AH specification in RFC 1826 did not discuss the concept of sequence numbers. Older IPSec implementations that are based on that RFC can therefore not provide replay protection.

**Authentication data** This is a variable-length field containing the Integrity Check Value (ICV), and is padded to 32 bits for IPv4 or 64 bits for IPv6. The ICV for each packet is calculated with the algorithm selected at SA initialization. As its name implies, it is used by the receiver to verify the integrity of the incoming packet.

In theory, any MAC algorithm can be used to calculate the

ICV. The specification requires that HMAC-MD5-96 and HMAC-SHA-1-96 must be supported. The old RFC 1826 requires Keyed MD5. In practice, Keyed SHA-1 is also used. Implementations usually support two to four algorithms.

When doing the ICV calculation, the mutable fields are considered to be filled with zero.

##### Ways of using AH

AH can be used in two ways: transport mode and tunnel mode.

###### AH in transport mode

In this mode, the authentication header is inserted immediately after the IP header, as shown in Figure 22-25. If the datagram already has IPSec headers, the AH is inserted before them.

IP Hdr

AH

Payload

IP Hdr

Payload

Original IP datagram

Datagram with AH

in transport mode

Authenticated

(

except mutable fields

)

*Figure 22-25 Authentication Header in transport mode*

Transport mode is used by hosts, not by gateways. Gateways are not required to support transport mode.

The advantage of transport mode is fewer processing costs. The disadvantage is that mutable fields are not authenticated.

###### AH in tunnel mode

With this mode, the tunneling concept is applied, a new IP datagram is constructed and the original IP datagram is made the payload of it. AH in transport mode is applied to the resulting datagram. See Figure 22-26 for an illustration.

AH

Authenticated

)

(

except mutable fields

Dest

options\*

Payload

IP Hdr

hop, dest\*,

routing, frag

Ext. Hdr(s)

*Figure 22-26 Authentication Header in tunnel mode*

Tunnel mode is used whenever either end of a Security Association is a gateway. Therefore, between two firewalls, tunnel mode is always used.

Gateways often also support transport mode. This mode is allowed when the gateway acts as a host, that is, in cases when traffic is destined to the gateway itself. For example, SNMP commands can be sent to the gateway using transport mode.

In tunnel mode, the outer headers' IP addresses do not need to be the same as the inner headers' addresses. For example, two security gateways can operate an AH tunnel that is used to authenticate all traffic between the networks they connect together. This is a very typical mode of operation.

The advantages of tunnel mode include total protection of the encapsulated IP datagram and the possibility of using private addresses. However, there are extra processing costs associated with this mode.

**Note:** The original AH specification in RFC 1825 only mentions tunnel mode in passing, not as a requirement. Because of this, there are IPSec implementations based on that RFC that do not support AH in tunnel mode.

##### IPv6 considerations

AH is an integral part of IPv6 (see 9.2.1, “Extension headers” on page 333). In an IPv6 environment, AH is considered an end-to-end payload and it appears after hop-by-hop, routing, and fragmentation extension headers. The destination options extension headers can appear either before or after the Authentication Header. Figure 22-27 illustrates the positioning of AH in transport mode for a typical IPv6 packet. The position of the extension headers marked with an asterisk (\*) is variable, if present at all.

*Figure 22-27 AH in transport mode for IPv6*

AH

Authenticated

(

except mutable fields

)

Dest

options\*

Payload

IP Hdr

hop, dest\*,

routing, frag

Ext. Hdr(s)

For a detailed description of AH in IPv6, refer to RFC 2402.

#### 22.4.3 Encapsulating Security Payload (ESP)

ESP is used to provide integrity check, authentication, and encryption to IP datagrams. Optional replay protection is also possible. These services are connectionless, in that they operate on a per-packet basis. The set of desired services are selectable upon SA establishment. However, some restrictions apply:

Integrity check and authentication are used together.

Replay protection is selectable only in conjunction with integrity check and authentication.

Replay protection can be selected only by the receiver.

Encryption can be selected independently of other services. It is highly recommended that, if encryption is enabled, integrity check and authentication be turned on. If only encryption is used, intruders can forge packets in order to mount cryptanalytic attacks.

Although both authentication (with integrity check) and encryption are optional, at least one of them is always selected; otherwise, you would not be using ESP.

ESP is identified by protocol number 50, as assigned by the IANA. The protocol header (IPv4, IPv6, or extension) immediately preceding the AH header will contain this value in its protocol (IPv4) or the next header field (IPv6, extension).

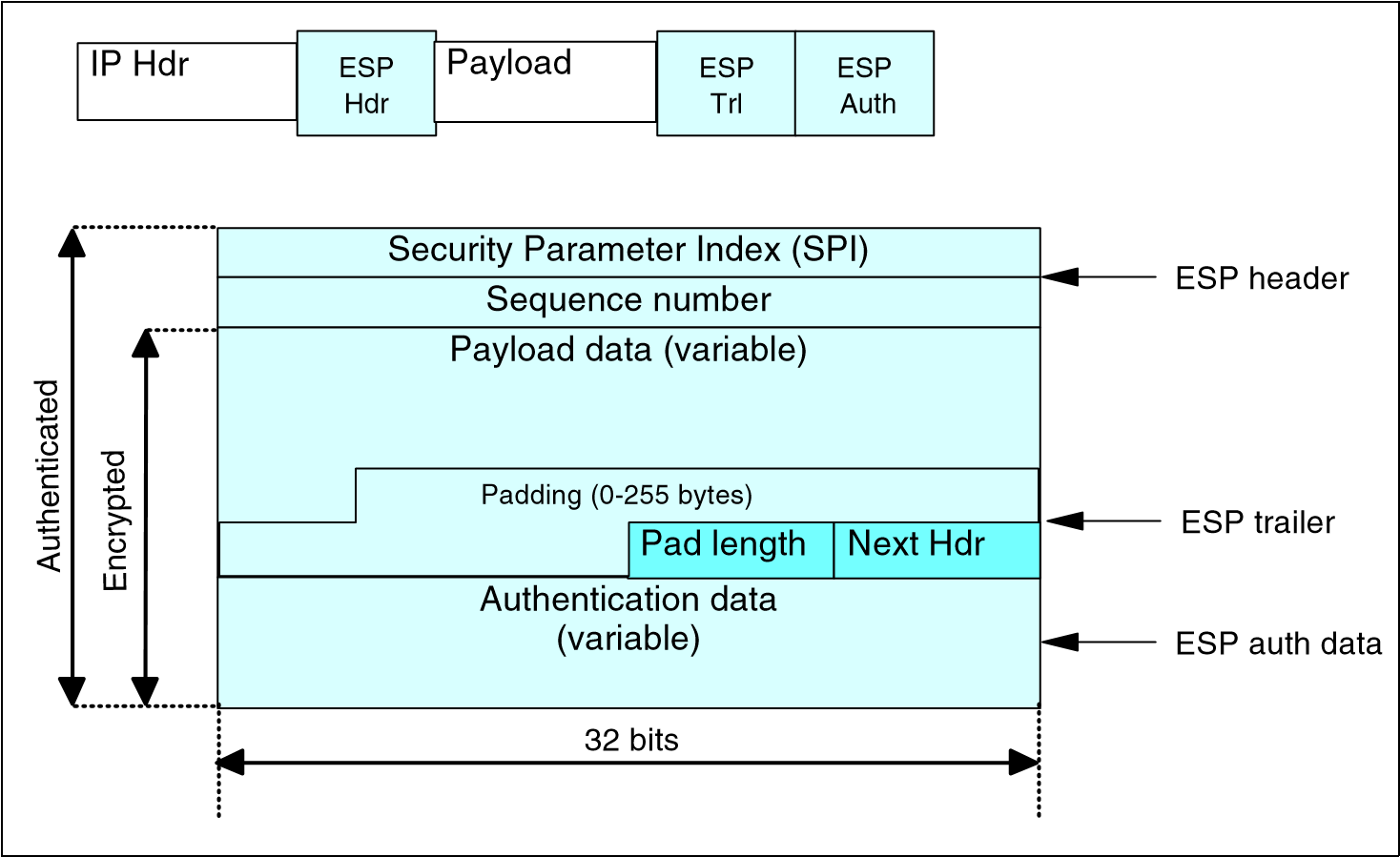
ESP processing is applied only to non-fragmented IP packets. However, an IP packet with ESP applied can be fragmented by intermediate routers. In this case, the destination first reassembles the packet and then applies ESP processing to it. If an IP packet that appears to be a fragment is input to ESP processing (offset field is non-zero, or the More Fragments bit is set), it is discarded. This prevents the overlapping fragment attack mentioned in 22.4.2, “Authentication Header (AH)” on page 813.

If both encryption and authentication with integrity check are selected, the receiver first authenticates the packet and, only if this step was successful, proceeds with decryption. This mode of operation saves computing resources and reduces the vulnerability to denial-of-service attacks.

##### ESP packet format

The current ESP packet format is described in RFC 2406. It contains important modifications compared to the previous ESP specification, RFC 1827. The information in this section is based on RFC 2406.

The format of the ESP packet is more complicated than that of the AH packet. There is not only an ESP header, but also an ESP trailer and ESP authentication data (see Figure 22-28 on page 819). The payload is located (encapsulated) between the header and the trailer, thus the name of the protocol.



*Figure 22-28 ESP header and trailer*

The following fields are part of an ESP packet:

###### Security Parameter Index (SPI)

This field is 32 bits in length. See “AH format” on page 814 for the definition.

**Sequence number** This 32-bit field is a monotonically increasing counter. See “AH format” on page 814 for the definition.

**Notes:** Typically, the anti-replay mechanism is not used with manual key management. The original ESP specification in RFC 1827 did not discuss the concept of sequence numbers. Older IPSec implementations that are based on that RFC can therefore not provide replay protection.

**Payload data** The payload data field is mandatory. It consists of a variable number of bytes of data described by the next header field. This field is encrypted with the cryptographic algorithm selected during SA establishment. If the

algorithm requires initialization vectors, these are also included here.

The ESP specification requires support for the DES algorithm in CBC mode (DES-CBC transform). Often, other encryption algorithms are also supported, such as triple-DES and CDMF, in the case of IBM products.

**Padding** Most encryption algorithms require that the input data must be an integral number of blocks. Also, the resulting ciphertext (including the padding, pad length, and next header fields) must terminate on a 4-byte boundary, so the next header field is right-aligned. For this reason, padding is included. It can also be used to hide the length of the original messages. However, this might adversely impact the effective bandwidth. Padding is an optional field (but needed for some algorithms).

**Note:** The encryption covers the payload data, padding, pad length and next header fields.

**Pad length** This 8-bit field contains the number of the preceding padding bytes. It is always present, and the value of 0 indicates no padding.

**Next header** The next header is an 8-bit mandatory field that shows the data type carried in the payload, for example, an upper-level protocol identifier such as TCP. The values are chosen from the set of IP protocol numbers defined by the IANA.

**Authentication data** This field is variable in length and contains the ICV calculated for the ESP packet from the SPI to the next header field inclusive. The authentication data field is optional. It is included only when integrity check and authentication have been selected at SA initialization time.

The ESP specifications require two authentication algorithms to be supported: HMAC with MD5 and HMAC with SHA-1. Often the simpler keyed versions are also supported by IPSec implementations.

**Notes:** The IP header is not covered by the ICV. The original ESP specification in RFC 1827 discusses the concept of authentication within ESP in conjunction with the encryption transform. That is, there is no authentication data field and it is left to the encryption transforms to eventually provide authentication.

##### Ways of using ESP

Like AH, ESP can be used in two ways: transport mode and tunnel mode.

###### ESP in transport mode

In this mode, the ESP header is inserted right after the IP header, as shown in Figure 22-29. If the datagram already has IPSec header or headers, the ESP header is inserted before any of those. The ESP trailer and the optional authentication data are appended to the payload.

O rig in a l IP d a ta g ra m

ESP

Hdr

D a ta g ra m w ith E S P

in tra n sp o rt m o d e

Encrypted

Authenticated

IP H d r

IP Hdr

P a ylo a d

Payload

ESP

Trl

ESP

Auth

*Figure 22-29 ESP in transport mode*

ESP in transport mode provides neither authentication nor encryption for the IP header. This is a disadvantage, because false packets might be delivered for ESP processing. The advantage of transport mode is the lower processing cost.

As in the case of AH, ESP in transport mode is used by hosts, not gateways. Gateways are not required to support transport mode.

###### ESP in tunnel mode

As expected, this mode applies the tunneling principle. A new IP packet is constructed with a new IP header. ESP is then applied, as in transport mode. This is illustrated in Figure 22-30. Because the original datagram becomes the payload data for the new ESP packet, it is completely protected if both encryption and authentication are selected. However, the new IP header is still not protected.

ESP

Hdr

New

IP Hdr

ESP

Trl

ESP

Auth

IP Hdr

Payload

New

IP Hdr

IP Hdr

Payload

Datagram with ESP

in transport mode

Original IP datagram

Encrypted

Authenticated

IP Hdr

Payload

Tunneled datagram

*Figure 22-30 ESP in tunnel mode*

The tunnel mode is used whenever either end of a Security Association is a gateway. Therefore, between two firewalls the tunnel mode is always used.

Gateways often also support transport mode. This mode is allowed when the gateway acts as a host, that is, in cases when traffic is destined to the gateway itself. For example, SNMP commands can be sent to the gateway using transport mode.

In tunnel mode the outer header's IP addresses does not need to be the same as the inner headers' addresses. For example, two security gateways can operate an ESP tunnel that is used to secure all traffic between the networks they connect together. Hosts are not required to support tunnel mode.

The advantages of tunnel mode are total protection of the encapsulated IP datagram and the possibility of using private addresses. However, there is an extra processing charge associated with this mode.

##### IPv6 considerations

As with AH, ESP is an integral part of IPv6 (see 9.2.1, “Extension headers” on page 333). In an IPv6 environment, ESP is considered an end-to-end payload and it appears after hop-by-hop, routing, and fragmentation extension headers. The destination options extension header(s) could appear either before or after the AH header. Figure 22-31 illustrates the positioning of the AH header in transport mode for a typical IPv6 packet. The position of the extension headers marked with an asterisk (\*) is variable, if present at all.

For more details, refer to RFC 2406.

ESP

Hdr

IP Hdr

hop, dest\*,

routing, frag

ESP

Trl

ESP

Auth

Dest

options\*

Payload

Encrypted

Authenticated

Ext. Hdr

*Figure 22-31 ESP in transport mode for IPv6*

##### Two authentication protocols

Knowing about the security services of ESP, you might ask if there is really a requirement for AH. Why does ESP authentication not cover the IP header as well? There is no official answer to these questions, but here are some points that justify the existence of two different IPSec authentication protocols:

ESP requires strong cryptographic algorithms to be implemented, whether it will actually be used or not. There are restrictive regulations on strong cryptography in some countries. It might be troublesome to deploy ESP-based solutions in such areas. However, authentication is not regulated and AH can be used freely around the world.

Often, only authentication is needed. AH is more performant compared to ESP with authentication only, because of the simpler format and lower processing costs. It makes sense to use AH in these cases.

Having two different protocols means finer-grade control over an IPSec network and more flexible security options. By nesting AH and ESP, for example, you can implement IPSec tunnels that combine the strengths of both protocols.

#### 22.4.4 Combining IPSec protocols

The AH and ESP protocols can be applied alone or in combination. Given the two modes of each protocol, there is quite a number of possible combinations. To make things more complicated, the AH and ESP SAs do not need to have identical endpoints. Luckily, out of the many possibilities, only a few make sense in real-world scenarios.

**Note:** RFC 2406 describes mandatory combinations that must be supported by each IPSec implementation. Other combinations may also be supported, but this might impact interoperability.

We mentioned in “Security Associations” on page 810 that the combinations of IPSec protocols are realized with SA bundles.

There are two approaches to creating an SA bundle:

Transport adjacency: Both security protocols are applied in transport mode to the same IP datagram. This method is practical for only one level of combination.

Iterated (nested) tunneling: The security protocols are applied in tunnel mode, in sequence. After each application, a new IP datagram is created and the next protocol is applied to it. This method has no limit in the nesting levels.

However, more than three levels are impractical.

These approaches can be combined. For example, an IP packet with transport adjacency IPSec headers can be sent through nested tunnels.

When designing a VPN, limit the number of IPSec processing stages. In our view, three stages is the limit beyond which further processing has no benefits. Two stages are sufficient for almost all cases.

Note that, in order to be able to create an SA bundle in which the SAs have different endpoints, at least one level of tunneling must be applied. Transport adjacency does not allow for multiple source/destination addresses, because only one IP header is present.

The practical principle of the combined usage is that, upon the receipt of a packet with both protocol headers, the IPSec processing sequence should be authentication followed by decryption. It is common sense not to bother with decryption of packets of uncertain origin.

Following this principle, the sender first applies ESP and then AH to the outbound traffic. In fact, this sequence is an explicit requirement for transport mode IPSec processing. When using both ESP and AH, a new question arises:

Should ESP authentication be turned on? AH authenticates the packet anyway. The answer is simple.

Turning on ESP authentication makes sense only when the ESP SA extends beyond the AH SA. For example, ESP can be used end-to-end, while AH only goes as far as the remote gateway. In this case, not only does it make sense to use ESP authentication, but we highly recommend doing so to avoid spoofing attacks within the intranet.

As far as the modes are concerned, transport mode is usually used between the endpoints of a connection and tunnel mode is usually used between two machines when at least one of them is a gateway.

Let us take a look at the different ways of using the IPSec protocols, from the simplest to the more complicated nested setups.

##### Case 1: End-to-end security

As shown in Figure 22-32, two hosts are connected through the Internet (or an intranet) without any IPSec gateway between them. They can use ESP, AH, or both. Either transport or tunnel mode can be applied.

*Figure 22-32 End-to-end security*



Internet/

intranet

H1

H2

Connection

IPSec tunnel

The following combinations are required to be supported by any IPSec implementation:

Transport mode

* AH alone
* ESP alone
* AH applied after ESP (transport adjacency)

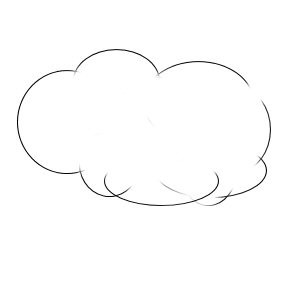
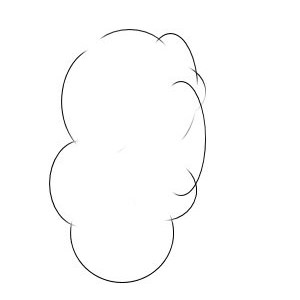
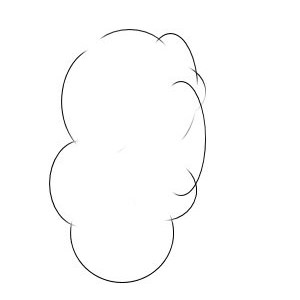
Tunnel mode

* AH alone
* ESP alone

##### Case 2: Basic VPN support

We describe virtual private networks (VPNs) in 22.10, “Virtual private networks (VPNs) overview” on page 861.

Figure 22-33 illustrates the simplest IPSec VPN. The gateways G1 and G2 run the IPSec protocol stack. The hosts in the intranets are not required to support IPSec.



Internet/

intranet

G1

G2

Connection

IPSec tunnel

H1

H2

intranet

intranet

*Figure 22-33 Basic VPN support*

In this case, the gateways are required to support only tunnel mode, either with AH or ESP.

###### Combined tunnels between gateways

Although gateways are required to support either an AH tunnel or ESP tunnel, it is often desirable to have tunnels between gateways that combine the features of both IPSec protocols.

The IBM IPSec implementations support this type of combined AH-ESP tunnels. The order of the headers is user selectable by setting the tunnel policy.

A combined tunnel between gateways does not mean that iterated tunneling takes place. Because the SA bundles comprising the tunnel have identical endpoints, it is inefficient to do iterated tunneling. Instead, one IPSec protocol is applied in tunnel mode and the other in transport mode, which can be conceptually thought of as a combined AH-ESP tunnel. An equivalent approach is to IP tunnel the original datagram and then apply transport adjacency IPSec processing to it. The result is that we have an outer IP header followed by the IPSec headers in the order set by the tunnel policy, and then the original IP packet, as shown in Figure 22-34 on page 827. This is the packet format in a combined AH-ESP tunnel between two IBM firewalls.

**Note:** ESP authentication data was not present in early implementations of the IBM firewall.

*Figure 22-34 Combined AH-ESP tunnel*

ESP

Hdr

IP Hdr

ESP

Trl

Inner

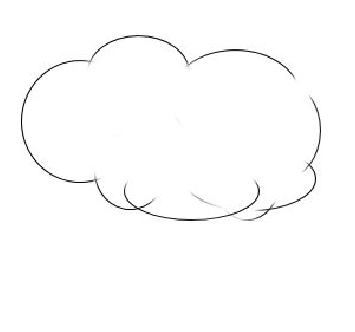
IP Hdr

Payload

AH

##### Case 3: End-to-end security with VPN support

This case is a combination of cases 1 and 2 and does not raise new IPSec requirements for the machines involved (see Figure 22-35). The big difference from case 2 is that now the hosts are also required to support IPSec.



Internet/

intranet

G1

G2

IPSec tunnels

Connection

H1

H2

intranet

intranet

*Figure 22-35 End-to-end security with VPN support*

In a typical setup, the gateways use AH in tunnel mode, while the hosts use ESP in transport mode. An enhanced security version might use a combined AH-ESP tunnel between the gateways. In this way, the ultimate destination addresses are encrypted; the whole packet traveling the Internet would be authenticated and the carried data double encrypted. This is the only case when three stages of IPSec processing might be useful, however, at a cost—the performance impact is considerable.

***AH tunneling of ESP transport*** Let us look in more detail at the common combination of using AH tunneling to protect ESP traffic in transport mode.

Figure 22-36 shows in detail how this combination is realized. Consider that host H1 in Figure 22-35 on page 827 sends an IP packet to host H2. Here is what happens:

1. Host H1 constructs the IP packet and applies ESP transport to it. H1 then sends the datagram to gateway G1, the destination address being H2.
2. Gateway G1 realizes that this packet should be routed to G2. Upon consulting its IPSec databases (SPD and SAD), G1 concludes that AH in tunnel mode must be applied before sending the packet out. It does the required encapsulation. Now the IP packet has the address of G2 as its destination, the ultimate destination H2 being encapsulated.
3. Gateway G2 receives the AH-tunneled packet. It is destined to itself, so it authenticates the datagram and strips off the outer header. G2 sees that the payload is yet another IP packet (that one sent by H1) with destination H2, so it forwards to H2. G2 does not care that this packet has an ESP header.
4. Finally H2 receives the packet. Because this is the destination, ESP-transport processing is applied and the original payload retrieved.

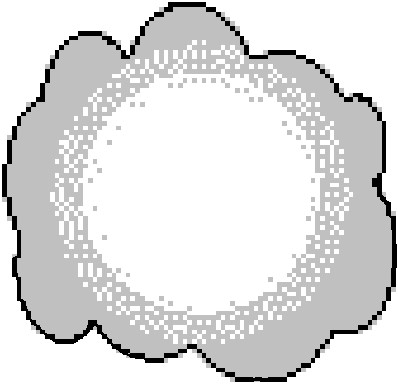
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | IP Hdr  Src:H1  Dest:H2 | | Payload | | | | | | | IP Hdr  Src:H1  Dest:H2 | | | ESP  Hdr | | | Payload | | | | | ESP  Trl | | | ESP  Auth | | | New IP hdr  Src:G1  Dest:G2 | | | AH | | | IP Hdr  Src:H1  Dest:H2 | | | | | ESP  Hdr | | | Payload | | | | | ESP  Trl | | ESP  Auth | | | IP Hdr  Src:H1  Dest:H2 | | | | ESP  Hdr | | | | Payload | | | | | ESP  Trl | | ESP  Auth | | | IP Hdr  Src:H1  Dest:H2 | | | | Payload | | | | | | |

*Figure 22-36 Nesting of IPSec protocols*

##### Case 4: Remote access

This case, shown in Figure 22-37, applies to remote hosts that use the Internet to reach a server in the organization protected by a firewall. The remote host typically uses a PPP dial-in connection to an ISP.

*Figure 22-37 Remote access*



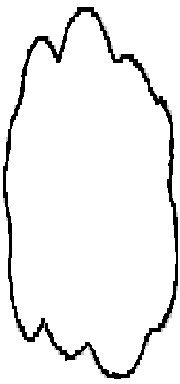
Internet/

intranet

G2

Connection

IPSec tunnels



H1

H2

intranet

Between the remote host H1 and the firewall G2, only tunnel mode is required. The choices are the same as in case 2. Between the hosts themselves, either tunnel mode or transport mode can be used, with the same choices as in case 1.

A typical setup is to use AH in tunnel mode between H1 and G2 and ESP in transport mode between H1 and H2. Older IPSec implementations that do not support AH in tunnel mode cannot implement this.

It is also common to create a combined AH-ESP tunnel between the remote host H1 and the gateway G2. In this case, H1 can access the whole intranet using just one SA bundle, while if it were using the setup shown in Figure 22-37, it only could access one host with one SA bundle.

#### 22.4.5 Internet Key Exchange (IKE) protocol

The Internet Key Exchange (IKE) framework, previously referred to as ISAKMP/Oakley, supports automated negotiation of Security Associations, and automated generation and refresh of cryptographic keys. The ability to perform these functions with little or no manual configuration of machines is a critical element to any enterprise-scale IPSec deployment.

Before describing the details of the key exchange and update messages, some explanations are necessary:

Internet Security Association and Key Management Protocol (ISAKMP)

A framework that defines the management of Security Associations (negotiate, modify, delete) and keys, and it also defines the payloads for exchanging key generation and authentication data. ISAKMP itself does not define any key exchange protocols, and the framework it provides can be applied to security mechanisms in the network, transport, or application layer, and also to itself.

Oakley

A key exchange protocol that can be used with the ISAKMP framework to exchange and update keying material for Security Associations.

Domain of Interpretation (DOI)

Definition of a set of protocols to be used with the ISAKMP framework for a particular environment; also a set of common definitions shared with those protocols regarding the syntax of SA attributes and payload contents, namespace of cryptographic transforms, and so on. In relation to IPSec, the DOI instantiates ISAKMP for use with IP.

Internet Key Exchange (IKE)

A protocol that uses parts of ISAKMP and parts of the Oakley and SKEME key exchange protocols to provide management of keys and Security

Associations for the IPSec AH and ESP protocols and for ISAKMP itself.

##### Protocol overview

ISAKMP requires that all information exchanges must be both encrypted and authenticated, so that no one can eavesdrop on the keying material. The keying material will be exchanged only among authenticated parties. This is required because the ISAKMP procedures deal with initializing the keys, so they must be capable of running over links where no security can be assumed to exist.

In addition, the ISAKMP methods have been designed with the explicit goals of providing protection against several well-known exposures:

Denial of service: The messages are constructed with unique *cookies* that can be used to quickly identify and reject invalid messages without the need to execute processor-intensive cryptographic operations. Man-in-the-middle: Protection is provided against the common attacks such

as deletion of messages, modification of messages, reflecting messages back to the sender, replaying of old messages, and redirection of messages to unintended recipients.

Perfect Forward Secrecy (PFS): Compromise of past keys provides no useful clues for breaking any other key, whether it occurred before or after the compromised key. That is, each refreshed key will be derived without any dependence on predecessor keys.

The following authentication methods are defined for IKE:

Pre-shared key

Digital signatures (DSS and RSA)

Public key encryption (RSA and revised RSA)

The robustness of any cryptography-based solution depends much more strongly on keeping the keys secret than it does on the actual details of the chosen cryptographic algorithms. Therefore, the IETF IPSec Working Group has prescribed a set of extremely robust Oakley exchange protocols. It uses a two-phase approach.

###### Phase 1

This set of negotiations establishes a master secret from which all cryptographic keys will be derived for protecting the users' data traffic. In the most general case, public key cryptography is used to establish an ISAKMP Security Association between systems and to establish the keys that will be used to protect the ISAKMP messages that will flow in the subsequent phase 2 negotiations. Phase 1 is concerned only with establishing the protection suite for the ISAKMP messages themselves, but it does not establish any Security Associations or keys for protecting user data.

In phase 1, the cryptographic operations are the most processor-intensive, but need only be done infrequently, and a single phase 1 exchange can be used to support multiple subsequent phase 2 exchanges. As a guideline, phase 1 negotiations are executed once a day or maybe once a week, while phase 2 negotiations are executed once every few minutes.

###### Phase 2

Phase 2 exchanges are less complex, because they are used only after the security protection suite negotiated in phase 1 has been activated. A set of communicating systems negotiate the Security Associations and keys that will protect user data exchanges. Phase 2 ISAKMP messages are protected by the ISAKMP Security Association generated in phase 1. Phase 2 negotiations generally occur more frequently than phase 1. For example, a typical application of a phase 2 negotiation is to refresh the cryptographic keys once every two to three minutes.

###### Permanent identifiers

The IKE protocol also offers a solution even when the remote host's IP address is not known in advance. ISAKMP allows a remote host to identify itself by a *permanent* identifier, such as a name or an e-mail address. The ISAKMP phase 1 exchanges will then authenticate the remote host's permanent identity using public key cryptography:

Certificates create a binding between the permanent identifier and a public key. Therefore, ISAKMP's certificate-based phase 1 message exchanges can authenticate the remote host's permanent identify.

Because the ISAKMP messages themselves are carried within IP datagrams, the ISAKMP partner (for example, a firewall or destination host) can associate the remote host's dynamic IP address with its authenticated permanent identity.

##### Initializing Security Associations with IKE

This section outlines how ISAKMP/Oakley protocols initially establish Security Associations and exchange keys between two systems that want to communicate securely.

In the remainder of this section, the parties involved are named Host-A and Host-B. Host-A is the initiator of the ISAKMP phase 1 exchanges, and Host-B is the responder. If needed for clarity, subscripts A or B are used to identify the source of various fields in the message exchanges.

##### IKE phase 1: Setting up ISAKMP Security Associations

The Security Associations that protect the ISAKMP messages themselves are set up during the phase 1 exchanges. Because we are starting “cold” (no previous keys or SAs have been negotiated between Host-A and Host-B), the phase 1 exchanges use the ISAKMP Identity Protect Exchange (also known as Oakley Main Mode). Six messages are needed to complete the exchange:

Messages 1 and 2 negotiate the characteristics of the Security Associations. Messages 1 and 2 flow in the clear for the initial phase 1 exchange, and they are unauthenticated.

Messages 3 and 4 exchange nonces (random values) and also execute a Diffie-Hellman exchange to establish a master key (SKEYID). Messages 3 and 4 flow in the clear for the initial phase 1 exchange, and they are unauthenticated.

Messages 5 and 6 exchange the required information for mutually authenticating the parties' identities. The payloads of messages 5 and 6 are protected by the encryption algorithm and keying material established with messages 1 through 4.

A detailed description of the phase 1 messages and exchanged information follows.

###### IKE phase 1, message 1

Because Host-A is the initiating party, it constructs a cleartext ISAKMP message (message 1) and sends it to Host-B. The ISAKMP message itself is carried as the payload of a UDP packet, which in turn is carried as the payload of a normal IP datagram (see Figure 22-38).

IP

Header

UDP

Header

ISAKMP

Header

SA

Proposal

#1

Transform

1)

for #

(

...

Proposal

#n

Transform

(

for #n

)

Host A

Host B

Offer alternatives

Accept one

*Figure 22-38 Message 1 of an ISAKMP phase 1 exchange*

The source and destination addresses to be placed in the IP header are those of Host-A (initiator) and Host-B (responder), respectively. The UDP header will identify that the destination port is 500, which has been assigned for use by the ISAKMP protocol. The payload of the UDP packet carries the ISAKMP message itself.

In message 1, Host-A, the initiator, proposes a set of one or more protection suites for consideration by Host-B, the responder. Therefore, the ISAKMP message contains at least the following fields in its payload:

|  |  |
| --- | --- |
| **ISAKMP header** | The ISAKMP header in message 1 indicates an exchange type of Main Mode and contains a Message ID of 0. Host-A sets the Responder Cookie field to 0 and fills in a random value of its choice for the Initiator Cookie, denoted as Cookie-A. |
| **Security Association** | The Security Association field identifies the Domain of  Interpretation (DOI). Because the hosts plan to run |
| **Proposal Payload** | IPSec protocols between themselves, the DOI is simply IP.  Host-A's Proposal Payload specifies the protocol PROTO\_ISAKMP and sets the SPI value to 0. |

**Note:** For ISAKMP phase 1 messages, the SPI field within the Proposal Payload is not used to identify the ISAKMP Security Association. During phase 1, the ISAKMP SA is identified instead by the pair of values <Initiator Cookie, Responder Cookie>, both of which must be non-zero values. Because the Responder Cookie has not yet been generated by Host-B, the ISAKMP SA is not yet unambiguously identified.

**Transform Payload** The Transform Payload specifies KEY\_OAKLEY. For the KEY\_OAKLEY transform, Host-A must also specify the relevant attributes: namely, the authentication method to be used, the pseudo-random function to be used, and the encryption algorithm to be used.

**Note:** Multiple proposals can be included in message 1.

###### IKE phase 1, message 2

In message 1, Host-A proposed one or more candidate protection suites to be used to protect the ISAKMP exchanges. Host-B uses message 2 to indicate which one, if any, it will support. If Host-A proposed just a single option, Host-B merely needs to acknowledge that the proposal is acceptable.

The source and destination addresses to be placed in the IP header are those of Host-B (responder) and Host-A (initiator), respectively. The UDP header identifies that the destination port is 500, which has been assigned for use by the ISAKMP protocol. The payload of the UDP packet carries the ISAKMP message itself.

The message contents are as follows:

|  |  |  |
| --- | --- | --- |
| **ISAKMP header** | The ISAKMP header in message 2 indicates an exchange type of Main Mode and contains a Message ID of 0. Host-B sets the Responder Cookie field to a random value, which we call Cookie-B, and copies into the Initiator Cookie field the value that was received in the Cookie-A field of message 1. The value pair <Cookie-A, Cookie-B> serves as the SPI for the ISAKMP Security Association. | |
| **Security Association** | The Security Association field identifies the Domain of  Interpretation (DOI). Because the hosts plan to run IPSec protocols between themselves, the DOI is simply IP. | |
| **Proposal Payload** | Host-B's Proposal Payload specifies the protocol PROTO\_ISAKMP and sets the SPI value to 0. |
| **Transform Payload** | The Transform Payload specifies KEY\_OAKLEY. For the KEY\_OAKLEY transform, the attributes that were accepted from the proposal offered by Host-A are copied into the appropriate fields. |

At this point, the properties of the ISAKMP Security Association have been agreed to by Host-A and Host-B. The identity of the ISAKMP SA has been set equal to the pair <Cookie-A, Cookie-B>. However, the identities of the parties claiming to be Host-A and Host-B have not yet been authoritatively verified.

###### IKE phase 1, message 3

The third message of the phase 1 ISAKMP exchange begins the exchange of the information from which the cryptographic keys will eventually be derived (see Figure 22-39 on page 836).

The ISAKMP payload exchanges two types of information.

**Important:** None of the messages themselves carry the actual cryptographic keys. Instead, they carry inputs that will be used by Host-A and Host-B to derive the keys locally.

Diffie-Hellman public value

The Diffie-Hellman public value gx from the initiator. The exponent x in the public value is the private value that must be kept secret.

|  |  |
| --- | --- |
| **Nonce** | The nonce Ni from the initiator. (Nonce is a name for a value that is considered to be random according to some very strict mathematical guidelines.) |
| **ID** | If the RSA public key is used for authentication, the nonces are encrypted with the public key of the other party. Likewise for the IDs of either party, which are then also exchanged at this stage. |

If authentication with revised RSA public key is used, the KE and ID payloads are encrypted with a secret key that is derived from the nonces and the encryption algorithm agreed to in messages 1 and 2, thus avoiding one CPU-intensive public key operation.

Certificates can optionally be exchanged in either case of public key authentication, as well as a hash value thereof.

These values are carried in the Key Exchange, and the Nonce and the ID fields, respectively.



IP

Header

UDP

Header

ISAKMP

Header

g

x

N

j

ID

Certificate

Signature

Host A

Host B

g

x

, N

j

g

y

, N

r

*Figure 22-39 Message 3 of an ISAKMP phase 1 exchange*

###### IKE phase 1, message 4

After receiving a Diffie-Hellman public value and a nonce from Host-A, Host-B responds by sending to Host-A its own Diffie-Hellman public value (gy from the responder) and its nonce (Nr from the responder).

###### Generating the keys (phase 1)

At this point, each host knows the values of the two nonces (Ni and Nr). Each host also knows its own private Diffie-Hellman value (x and y) and also knows its partner's public value (gx or gy). Therefore, each side can construct the composite value gxy. And finally, each side knows the values of the initiator cookie and the responder cookie.

Given all these bits of information, each side can then independently compute identical values for the following quantities:

SKEYID: This collection of bits is sometimes referred to as keying material, because it provides the raw input from which actual cryptographic keys will be derived later in the process. It is obtained by applying the agreed-to keyed pseudorandom function (prf) to the known inputs:

* For digital signature authentication:

SKEYID = prf(Ni, Nr, gxy)

* For authentication with public keys:

SKEYID = prf(hash(Ni, Nr), CookieA, CookieB) – For authentication with a pre-shared key: SKEYID = prf(pre-shared key, Ni, Nr)

Having computed the value SKEYID, each side then proceeds to generate two cryptographic keys and some additional keying material:

* SKEYID\_d is keying material that will be subsequently used in phase 2 to derive the keys that will be used in non-ISAKMP SAs for protecting user traffic:

SKEYID\_d = prf(SKEYID, gxy, CookieA, CookieB, 0)

* SKEYID\_a is the key used for authenticating ISAKMP messages:

SKEYID\_a = prf(SKEYID, SKEYID\_d, gxy, CookieA, CookieB, 1) – SKEYID\_e is the key used for encrypting ISAKMP exchanges:

SKEYID\_e = prf(SKEYID, SKEYID\_a, gxy, CookieA, CookieB, 2)

At this point in the protocol, both Host-A and Host-B have derived identical authentication and encryption keys that they will use to protect the ISAKMP exchanges. And they have also derived identical keying material from which they will derive keys to protect user data during phase 2 of the ISAKMP negotiations. However, at this point, the two parties' identities still have not been authenticated to one another.

###### IKE phase 1, message 5

At this point in the phase 1 flows, the two hosts exchange identity information with each other to authenticate themselves. As shown in Figure 22-40, the ISAKMP message carries an identity payload, a signature payload, and an optional certificate payload. Host-A uses message 5 to send information to Host-B that will allow Host-B to authenticate Host-A.

IP

Header

UDP

Header

ISAKMP

Header

Identity

Certificate

Signature

*Figure 22-40 Message 5 of an ISAKMP phase 1 exchange*

When an actual certificate is present in the Certificate Payload field, the receiver can use the information directly, after verifying that it has been signed with a valid signature of a trusted certificate authority. If there is no certificate in the message, it is the responsibility of the receiver to obtain a certificate using some implementation method. For example, it can send a query to a trusted certificate authority using a protocol such as LDAP, or it can query a secure DNS server, or it can maintain a secure local cache that maps previously used certificates to their respective ID values, or it can send an ISAKMP Certificate Request message to its peer, who must then immediately send its certificate to the requester.

**Note:** The method for obtaining a certificate is a local option, and is not defined as part of IKE. In particular, it is a local responsibility of the receiver to check that the certificate in question is still valid and has not been revoked.

There are several points to bear in mind:

At this stage of the process, all ISAKMP payloads, whether in phase 1 or phase 2, are encrypted, using the encryption algorithm (negotiated in messages 1 and 2) and the keys (derived from the information in messages 3 and 4). The ISAKMP header itself, however, is still transmitted in the clear.

In phase 1, IPSec's ESP protocol is not used; that is, there is no ESP header. The recipient uses the encryption bit in the Flags field of the ISAKMP header to determine if encryption has been applied to the message. The pair of values <CookieA, CookieB>, which serve as an SPI for phase 1 exchanges, provide a pointer to the correct algorithm and key to be used to decrypt the message.

The digital signature, if used, is not applied to the ISAKMP message itself. Instead, it is applied to a hash of information that is available to both Host-A and Host-B.

The identity carried in the identity payload does not necessarily bear any relationship to the source IP address; however, the identity carried in the identity payload must be the identity to which the certificate, if used, applies.

Host-A (the initiator) generates the following hash function, and then places the result in the Signature Payload field:

HASH\_I = prf(SKEYID, gx, gy, CookieA, CookieB, SAp, IDA)

If digital signatures were used for authentication, this hash will also be signed by Host-A.

IDA is Host-A's identity information that was transmitted in the identity payload of this message, and SAp is the entire body of the SA payload that was sent by Host-A in message 1, including all proposals and all transforms proposed by Host-A. The cookies, public Diffie-Hellman values, and SKEYID were explicitly carried in messages 1 through 4, or were derived from their contents.

###### IKE phase 1, message 6

After receiving message 5 from Host-A, Host-B verifies the identity of Host-A by validating the hash.

If digital signatures were used for authentication, the signature of this hash are verified by Host-B.

If this is successful, Host-B sends message 6 to Host-A to allow Host-A to verify the identity of Host-B.

The structure of message 6 is the same as that of message 5, with the obvious changes that the identity payload and the certificate payload now pertain to Host-B:

HASH\_R = prf(SKEYID, gy, gx, CookieB, CookieA, SAp, IDB)

Notice that the order in which Diffie-Hellman public values and the cookies appear has been changed, and the final term now is the identity payload that Host-B has included in message 6.

If digital signatures were used for authentication, this hash is also signed by Host-B, which is different from the one previously signed by Host-A.

When Host-A receives message 6 and verifies the hash or digital signature, the phase 1 exchanges are then complete. At this point, each participant has authenticated itself to its peer. Both have agreed on the characteristics of the ISAKMP Security Associations, and both have derived the same set of keys (or keying material).

###### Miscellaneous phase 1 facts

There are several miscellaneous facts worth noting:

Regardless of the specific authentication mechanism that is used, there will be six messages exchanged for the Oakley Main Mode. However, the content of the individual messages differs, depending on the authentication method.

Although Oakley exchanges make use of both encryption and authentication, they do not use either IPSec's ESP or AH protocol. ISAKMP exchanges are protected with application-layer security mechanisms, not with network-layer security mechanisms.

ISAKMP messages are sent using UDP. There is no guaranteed delivery for them.

The only way to identify that an ISAKMP message is part of a phase 1 flow rather than a phase 2 flow is to check the Message ID field in the ISAKMP header. For phase 1 flows, it must be 0, and (although not explicitly stated in the ISAKMP documents) for phase 2 flows, it must be non-zero.

##### IKE phase 2: Setting up protocol Security Associations

After completing the phase 1 negotiation process to set up the ISAKMP Security Associations, Host-A's next step is to initiate the Oakley phase 2 message exchanges (also known as Oakley Quick Mode) to define the Security Associations and keys that will be used to protect IP datagrams exchanged between the pair of users. (In the Internet drafts, these are referred to somewhat obtusely as “non-ISAKMP SAs.”)

Because the purpose of the phase 1 negotiations was to agree on how to protect ISAKMP messages, all ISAKMP phase 2 payloads, but not the ISAKMP header itself, must be encrypted using the algorithm agreed to by the phase 1 negotiations.

When Oakley Quick Mode is used in phase 2, authentication is achieved through the use of several cryptographically based hash functions. The input to the hash functions comes partly from phase 1 information (SKEYID) and partly from information exchanged in phase 2. Phase 2 authentication is based on certificates, but the phase 2 process itself does not use certificates directly. Instead, it uses the SKEYID\_a material from phase 1, which itself was authenticated through certificates.

Oakley Quick Mode comes in two forms:

Without a Key Exchange attribute, Quick Mode can be used to refresh the cryptographic keys, but does not provide the property of Perfect Forward Secrecy (PFS).

With a Key Exchange attribute, Quick Mode can be used to refresh the cryptographic keys in a way that provides PFS. This is accomplished by including an exchange of public Diffie-Hellman values within messages 1 and 2.

**Note:** PFS apparently is a property that is very much desired by cryptography experts, but strangely enough, the specifications treat PFS as optional. They mandate that a system must be capable of handling the Key Exchange field when it is present in a Quick Mode message, but do not require a system to include the field within the message.

A detailed description of the phase 2 messages and exchanged information follows.

###### IKE phase 2, message 1

Message 1 of a Quick Mode Exchange allows Host-A to authenticate itself, to select a nonce, to propose Security Associations to Host-B, to execute an exchange of public Diffie-Hellman values, and to indicate if it is acting on its own behalf or as a proxy negotiator for another entity. An overview of the format of message 1 is shown in Figure 22-41.

**Note:** Inclusion of a Key Exchange field is optional. However, when Perfect Forward Secrecy is used, it must be present.

IP

Header

UDP

Header

ISAKMP

Header

Hash

SA

Proposal

#1

Transform

#1

...

Proposal

#n

Transform

#n

N

K

E

IDs

Host A

Host B

Hash-1,SA(ESP & AH),g

x

, N

j

Hash-2,SA(ESP & AH),g

y

, N

r

Hash-3

*Figure 22-41 Message 1 of an ISAKMP phase 2 Quick Mode Exchange*

Because we assumed that Host-A and Host-B are each acting on their own behalf, the user identity fields illustrated in Figure 22-41 will not be present. The message will consist of:

**ISAKMP header** The ISAKMP header indicates an exchange type of Quick Mode, includes a non-zero Message ID chosen by Host-A, includes the initiator and responder cookie values chosen in phase 1 (that is, Cookie-A and Cookie-B), and turns on the encryption flag to indicate that the payloads of the ISAKMP message are encrypted according to the algorithm and key negotiated during phase 1.

**Hash** A Hash payload must immediately follow the ISAKMP header. HASH\_1 uses the keyed pseudo-random function that was negotiated during the phase 1 exchanges, and is derived from the following information:

HASH\_1 = prf(SKEYID\_a, M-ID, SA, Nqmi, KE, IDqmi, IDqmr)

* SKEYID\_a was derived from the phase 1 exchanges.
* M-ID is the message ID of this message.
* SA is the Security Association payload carried in this message, including all proposals that were offered.
* Nonce is a new value different from the one used in phase 1.
* KE is the public Diffie-Hellman value carried in this message. This quantity is chosen by Host-A, and is denoted as gqmx. Note that this is not the same quantity as gx that was used in the phase 1 exchanges.
* IDs, which can identify either the endpoints of the phase 1 exchange or endpoints on whose behalf the protocol SA should be negotiated (proxy IDs when IKE is used in client mode). These can subsequently be different from the IDs used in phase 1.

**Note:** The use of KE and ID is optional, depending if PFS is used.

**Security Association** Indicates IP as the Domain of Interpretation.

Proposal, Transform Pairs

There can be one or more of these pairs in this message. The first proposal payload is numbered 1, identifies an IPSec protocol to be used, and includes an SPI value that is randomly chosen by Host-A for use with that protocol. The proposal payload is followed by a single transform payload that indicates the cryptographic algorithm to be used with that protocol. The second proposal payload is numbered 2, and so on.

**Nonce payload** This contains the nonce Nqmi that was chosen

randomly by Host-A.

|  |  |
| --- | --- |
| **KE** | This is the key exchange payload that carries the public Diffie-Hellman value chosen by Host-A, gqmx. There is also a field called Group that indicates the  prime number and generator used in the Diffie-Hellman exchange. |
| **ID payload** | Specifies the endpoints for this SA. |

###### IKE phase 2, message 2

After Host-B receives message 1 from Host-A and successfully authenticates it using HASH\_1, it constructs a reply, message 2, to be sent back to Host-A. The Message ID of the reply is the same one that Host-A used in message 1.

Host-B chooses new values for the following:

**Hash** The hash payload now carries the value HASH\_2,

which is defined as:

HASH\_2 = prf(SKEYID\_a, Nqmi, M-ID, SA, Nqmr, KE, IDqmi, IDqmr)

|  |  |
| --- | --- |
| **Security Association** | The Security Association payload only describes the single chosen proposal and its associated transforms, not all of the protection suites offered by Host-A. Host-B also chooses an SPI value for the selected protocol. Host-B's SPI does not depend in any way on the SPI that Host-A assigned to that protocol when it offered the proposal. That is, it is not necessary that SPIA be the same as SPIB; it is only necessary that they each be non-zero and that they each be randomly chosen. |
| **Nonce** | Nonce payload now carries Nr, a random value chosen by Host-B. |
| **KE** | Key exchange payload now carries Host-B's public Diffie-Hellman value, gqmy. |

At this point, Host-A and Host-B have exchanged nonces and public Diffie-Hellman values. Each one can use this in conjunction with other information to derive a pair of keys, one for each direction of transmission.

###### Generating the keys (phase 2)

Using the nonces, public Diffie-Hellman values, SPIs, protocol code points exchanged in messages 1 and 2 of phase 2, and the SKEYID value from phase 1, each host now has enough information to derive two sets of keying material:

When PFS is used:

* For data generated by Host-A and received by Host-B, the keying material is:

KEYMATAB = prf(SKEYID\_d, gqmxy, protocol, SPIB, Nqmi, Nqmr)

* For data generated by Host-B and received by Host-A, the keying material is:

KEYMATBA = prf(SKEYID\_d, gqmxy, protocol, SPIA, Nqmi, Nqmr) When PFS is not used:

* For data generated by Host-A and received by Host-B, the keying material is:

KEYMATAB = prf(SKEYID\_d, protocol, SPIB, Nqmi, Nqmr)

* For data generated by Host-B and received by Host-A, the keying material is:

KEYMATBA = prf(SKEYID\_d, protocol, SPIA, Nqmi, Nqmr)

**Note:** Depending on the particular case, Host-A might need to derive multiple keys for the following purposes:

Generating the integrity check value for transmitted datagrams

Validating the integrity check value of received datagrams

Encrypting transmitted datagrams

Decrypting received datagrams

Likewise, Host-B needs to derive the mirror image of the same keys. For example, the key that Host-B uses to encrypt its outbound messages is the same key that Host-A uses to decrypt its inbound messages, and so on.

###### IKE phase 2, message 3

At this point, Host-A and Host-B have exchanged all the information necessary for them to derive the necessary keying material. The third message in the Quick Mode exchange is used by Host-A to prove its alive state, which it does by producing a hash function that covers the message ID and both nonces that were exchanged in messages 1 and 2. Message 3 consists only of the ISAKMP header and a hash payload that carries:

HASH\_3 = prf(SKEYID\_a, 0, M-ID, Nqmi, Nqmr)

When Host-B receives this message and verifies the hash, both systems can begin to use the negotiated security protocols to protect their user data streams.

##### Negotiating multiple Security Associations

It is also possible to negotiate multiple Security Associations, each with its own set of keying material, within a single three-message Quick Mode exchange.

The message formats are very similar to the previously illustrated ones, so we only highlight the differences:

Message 1 carries multiple Security Association payloads, each offering a range of protection suites.

HASH\_1 covers the entire set of all offered Security Associations carried in message 1. That is, each Security Association and all of its offered proposals are included.

In message 2, for each offered SA, Host-B selects a single protection suite. That is, if n SAs are open for negotiation, Host-B chooses n protection suites, one from each proposal.

As was the case for HASH\_1, HASH\_2 now covers the entire set of all offered Security Associations carried in message 1. That is, each Security Association and all of its offered proposals are included.

After messages 1 and 2 have been exchanged, Host-A and Host-B generate the keying material for each of the accepted protection suites, using the same formulas as in “Generating the keys (phase 2)” on page 844, applied individually for each accepted SA. Even though the nonces and the public Diffie-Hellman values are the same for all selected suites, the keying material derived for each selected protection suite is different because each proposal has a different SPI.

Because multiple Security Associations have been negotiated, it is a matter of local choice as to which one is used to protect a given datagram. A receiving system must be capable of processing a datagram that is protected by any SA that has been negotiated. That is, it is legal for a given source host to send two consecutive datagrams to a destination system, where each datagram was protected by a different SA.

##### Using IKE with remote access

The critical element in the remote access scenario is the use of Oakley to identify the remote host by name, rather than by its dynamically assigned IP address. After the remote host's identity has been authenticated and the mapping to its dynamically assigned IP address has been ascertained, the remainder of the processes are the same as we have described for the other scenarios. For example, if the corporate intranet is considered to be trusted, the remote host needs to establish a single SA between itself and the firewall. But if the corporate intranet is considered to be untrusted, it might be necessary for the remote host to set up two SAs: one between itself and the firewall, and a second between itself and the destination host.

Recall that a single ISAKMP phase 1 negotiation can protect several subsequent phase 2 negotiations. Phase 1 ISAKMP negotiations use computationally intensive public key cryptographic operations, while phase 2 negotiations use the less computationally intensive symmetric key cryptographic operations. Therefore, the heavy computational load only occurs in phase 1, which is only executed when the dial-up connection is first initiated.

The principal points that pertain to the remote access case are:

The remote host's dynamically assigned address is the one that is placed in the IP header of all ISAKMP messages.

The remote host's permanent identifier (such as an e-mail address) is the quantity that is placed in the ID field of the ISAKMP phase 1 messages.

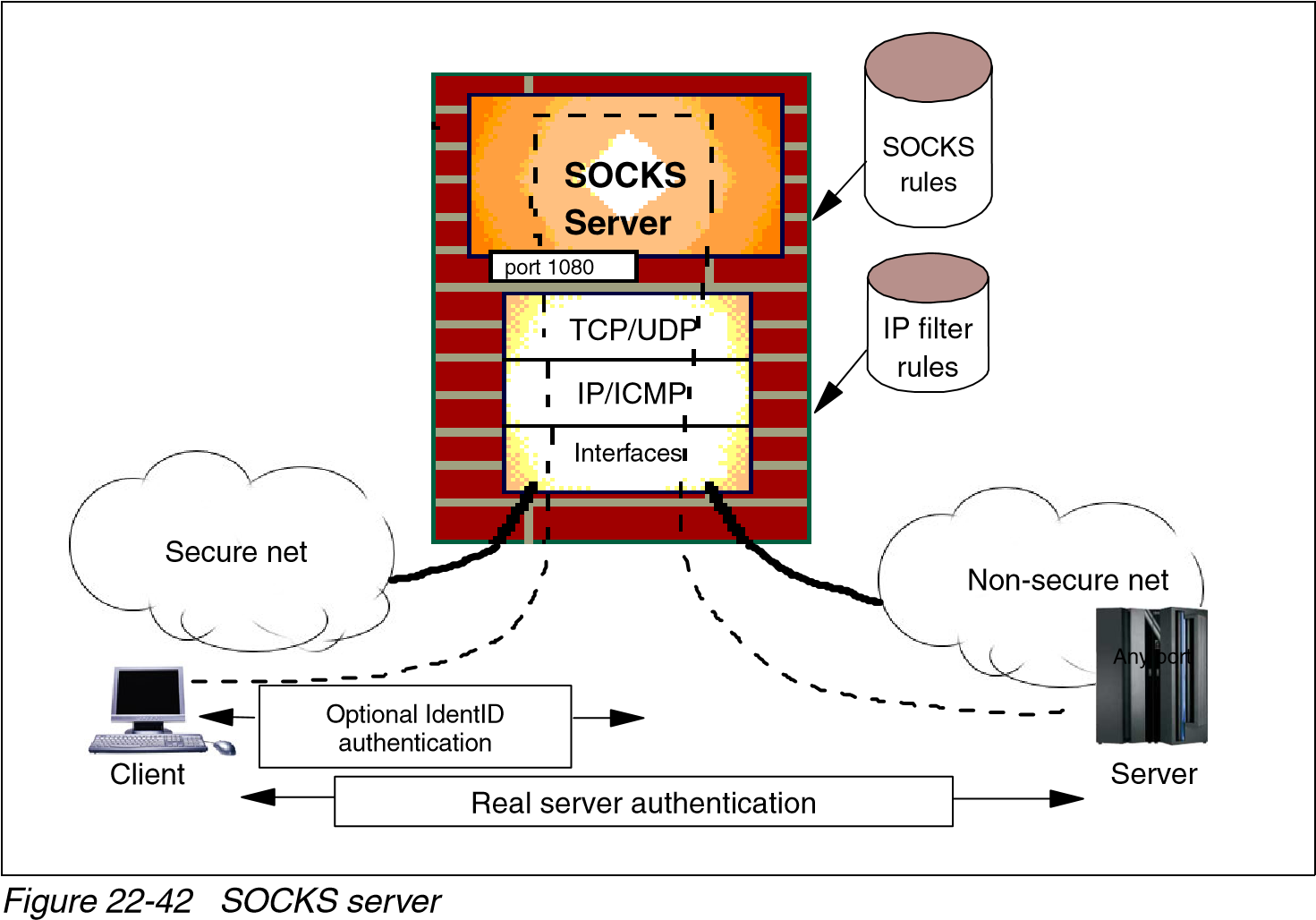
The remote host's certificate used in the ISAKMP exchange must be associated with the remote host's permanent identifier.

In traffic-bearing datagrams, the remote host's dynamically assigned IP address is used. This is necessary because the destination IP address that appears in the datagram's IP header is used in conjunction with the SPI and protocol type to identify the relevant IPSec Security Association for processing the inbound datagram.

### 22.5 SOCKS

SOCKS is a standard for circuit-level gateways. It does not require the processing costs associated with a more conventional proxy server where a user has to consciously connect to the firewall first before requesting the second connection to the destination (see Figure 22-42 on page 847).

The user starts a client application with the destination server IP address. Instead of directly starting a session with the destination server, the client initiates a session to the SOCKS server on the firewall. The SOCKS server then validates that the source address and user ID are permitted to establish an onward connection into the nonsecure network, and then creates the second session.



SOCKS needs to have new versions of the client code (called SOCKS-enabled clients) and a separate set of configuration profiles on the firewall. However, the server machine does not need modification; indeed, it is unaware that the session is being relayed by the SOCKS server. Both the client and the SOCKS server need to have SOCKS code. The SOCKS server acts as an application-level router between the client and the real application server. SOCKSv4 is for outbound TCP sessions only. It is simpler for the private network user, but does not have secure password delivery, so it is not intended for sessions between public network users and private network applications. SOCKSv5 provides for several authentication methods and can therefore be used for inbound connections as well, though these need to be used with caution. SOCKSv5 also supports UDP-based applications and protocols.

The majority of Web browsers are SOCKS-enabled and you can get SOCKS-enabled TCP/IP stacks for most platforms. For additional information, refer to RFC 1928, RFC 1929, RFC 1961, and the following URL: [http://www.socks.nec.com](http://www.socks.nec.com/)

#### 22.5.1 SOCKS Version 5 (SOCKSv5)

SOCKS version 5 is a proposed standard protocol with a status of elective. It is described in RFC 1928.

Application-level gateways provide secure connections for some applications such as Telnet, FTP, and SMTP. However, it is not easy to write proxy code for each new application. Generally, the proxy service becomes available after some time, even if the service can be used directly and application-level gateways do not allow UDP connections. SOCKSv5 satisfies all these shortcomings and requirements with a strong authentication mechanism and the hiding of addresses from a non-secure network. Although supporting UDP might seem to be vulnerable, it can be configured to pass UDP for particular users and particular applications only.

The SOCKSv5 concept is based on SOCKSv4 with some extensions such as UDP support, new and various sophisticated authentication methods, and extended addressing schemes to cover domain-name and IPv6. SOCKSv5 supports a range of authentication methods, including:

User name/password authentication

One-time password generators

Kerberos

Remote Authentication Dial-In User Services (RADIUS)

Password Authentication Protocol (PAP)

IPSec authentication method

SOCKSv5 also supports the following encryption standards:

DES

Triple DES

IPSec

The following tunneling protocols are supported:

PPTP

L2F

L2TP

The following key management systems are supported:

SKIP

ISAKMP/Oakley

*Figure 22-43 Socks TCP segment flow*

Proxy Server

SOCKS

Client

Client host

SOCKS host

Real

server

Server host

Secure

network

Non-secure

network

epn: Ephemeral port number n

sss: Server port number sss

ep1

1080

IdentD

113

sss

ep2

ep3

The SOCKSv5 server listens for connections on a given port, usually 1080. According to the connection type (TCP or UDP), the steps discussed in the following sections establish a connection.

##### SOCKSv5 TCP connection

To establish a connection using TCP, the client first sends a TCP packet that contains session request information through port 1080 to the server (see Figure 22-43). If the access permissions allow this operation and the connection request succeeds, the client enters an authentication negotiation. In this state, the authentication type is determined, after which the client sends a relay request. The SOCKSv5 server evaluates the request and either establishes the connection or rejects it. The client sends the following message, which contains a version identifier and method options (Figure 22-44).

ver

nmethods

methods

1

byte 1 byte 1 to 255 bytes

*Figure 22-44 SOCKSv5: Version identifier and method selection message format*

Where:

|  |  |
| --- | --- |
| **VER** | Indicates the version of SOCKS. For SOCKSv5, the value is hexadecimal X'05'. |
| **NMETHODS** | Indicates the number of the methods in the methods field. |
| **METHODS** | Indicates the supported authentication and encapsulation methods. |

The server responds by the following message (Figure 22-45).

ver

method

1

byte 1 byte

*Figure 22-45 SOCKSv5: Selected method message format*

The hexadecimal values for current standard methods are as follows:

|  |  |
| --- | --- |
| **X'00'** | No authentication required |
| **X'01'** | GSSAPI |
| **X'02'** | User name/password |
| **X'03' to X'7F'** | IANA assigned |
| **X'80' to X'FE'** | Reserved for private methods |
| **X'FF'** | No acceptable methods |

All implementations should support user name/password and GSSAPI authentication methods.

###### SOCKSv5 Connect

After authentication completes successfully, the client sends the request details. If an encapsulation method is negotiated during the method negotiation, the selected encapsulation method must be applied for the following messages. The detail request message format issued by the client is as shown in Figure 22-46.

ver

cmd

RSV

ATYP

DST ADDR

DST Port

1

byte 1 byte X'00' 1 byte variable 2 bytes

*Figure 22-46 SOCKSv5: Detail request message format*

Where:

|  |  |
| --- | --- |
| **VER** | Socks protocol version. For SOCKSv5, the value is hexadecimal X'05'. |
| **CMD** | SOCKS command in octets: |
| **X'01'** | Connect |
| **X'02'** | BIND |
| **X'03'** | UDP associate |
| **RSV** | Reserved for future use. |
| **ATYP** | Address types in octets: |
| **X'01'** | IPv4 address |
| **X'03'** | Domain-name |
| **X'04'** | IPv6 address |
| **DST.ADDR** | Desired destination address. |
| **DST.PORT** | Desired destination port in network octet order. |

An IPv4 address is stored as 4 bytes. An IPv6 address is stored as 16 bytes.

A domain name is stored as a length byte, and then a fully qualified domain name. There is no trailing null at the end of the domain name.

The server evaluates the request detail message and replies with one or more messages. Here is the reply message format issued by the server (Figure 22-47).

ver

rep

RSV

ATYP

BND.ADDR

BND.Port

1

byte 1 byte X'00' 1 byte variable 2 bytes

*Figure 22-47 SOCKSv5: Server reply message format*

Where:

|  |  |  |
| --- | --- | --- |
| **VER** | Socks protocol version. For SOCKSv5, the value is hexadecimal X'05'. | |
| **REP**  **X'00' X'01' X'02'** | Reply field:    Succeeded  General SOCKS server failure  Connection not allowed by ruleset | |
| **X'03'** | Network unreachable |
| **X'04'** | Host unreachable |
| **X'05'** | Connection refused |
| **X'06'** | TTL expired |
| **X'07'** | Command not supported |
| **X'08'** | Address type not supported |
| **X'09' to X'FF'** | Unassigned |
| **RSV** | Reserved for future use. |
| **ATYP** | Address types in octets: |
| **X'01'** | IPv4 address |
| **X'03'** | Domain name |
| **X'04'** | IPv6 address |
| **BND.ADDR** | Server bound address. |
| **BND.PORT** | Server bound port in network octet order. |

###### SOCKSv5 BIND

To accept an incoming connection from the Internet, use the same request and reply format as described earlier for SOCKSv5 Connect, setting the CMD field to BIND. However, you receive two reply packets.

The first reply contains the IP address and port number on which the SOCKS server has put a listener.

When the remote system calls into the SOCKS server, you get a second reply with the BND.ADDR and BIND.Port fields containing details of the remote server.

##### SOCKSv5 UDP connection

To be able use a UDP connection over a SOCKS server, the client first issues the UDP ASSOCIATE command to the SOCKSv5 server. The SOCKSv5 server then assigns a UDP port to which the client sends all UDP datagrams. Each UDP datagram has a UDP request header. The UDP request header format is as follows (Figure 22-48).

RSV

frag

ATYP

DST.ADDR

DST.Port

data

2

bytes 1 byte 1 byte variable 2 bytes variable

*Figure 22-48 SOCKSv5: UDP datagram request header format*

Where:

|  |  |
| --- | --- |
| **RSV** | Reserved for future use. All bytes are zero. |
| **FRAG** | Current fragment number. |
| **ATYP** | Address types in octets: |
| **X'01'** | IPv4 address |
| **X'03'** | Domain-name |
| **X'04'** | IPv6 address |
| **DST.ADDR** | Desired destination address. |
| **DST.PORT** | Desired destination port in network octet order. |
| **DATA** | User data. |

The UDP relay server gets the IP address of the client, which sends UDP datagrams to the port specified by DST.PORT. It then discards any datagram that comes from another source.

### 22.6 Secure Shell (1 and 2)

SSH can secure connections between systems. It allows application traffic, such as that generated by Telnet, FTP POP3, or even X Window System, to be both encrypted and compressed. Compression is useful over slow modem links. Implementations allow the user a choice of encryption methods.

Client software often offers both SSH1 and SSH2 support. The user is authenticated by password or public/private key.

SSH1 offers Blowfish, DES, 3DES, and RC4 encryption ciphers.

SSH2 offers 3DES, RC4, and Twofish encryption ciphers.

#### 22.6.1 SSH overview

SSH establishes a single TCP/IP connection from the client to the server. The traffic sent down this connection is encrypted, and optionally compressed using LempleZiv compression. Public/private keys can be used to verify both the user and the identity of the remote system.

##### SSH and X Window System

X Window System sessions can pass through the SSH connection. The SSH server generates a new DISPLAY variable (and xauth key) for the remote

X Window System’s clients. SSH forwards the X Window System traffic to the user’s local X Server. Users have to supply their own X Server applications; make sure it is listening on the local host.

##### SSH port forwarding

SSH offers the ability to map TCP/IP ports across systems. For example, you can configure SSH to copy data between a port on the client’s local host and the servers POP3 port. By running a POP3 client and pointing it at the local host, you establish a secure encrypted session over which to read e-mail.

### 22.7 Secure Sockets Layer (SSL)

SSL is a security protocol that was developed by Netscape Communications Corporation, along with RSA Data Security, Inc. The primary goal of the SSL protocol is to provide a private channel between communicating applications, which ensures privacy of data, authentication of the partners, and integrity.

#### 22.7.1 SSL overview

SSL provides an alternative to the standard TCP/IP socket API that has security implemented within it. Therefore, in theory, it is possible to run any TCP/IP application in a secure way without changing the application. In practice, SSL is only widely implemented for HTTP connections, but Netscape Communications Corp. has stated an intention to employ it for other application types, such as NNTP and Telnet, and there are several such implementations freely available on the Internet. IBM, for example, uses SSL to enhance security for TN3270 sessions in the IBM WebSphere Host On-Demand and eNetwork Communications Server products.

SSL is composed of two layers:

At the lower layer, a protocol for transferring data using a variety of predefined cipher and authentication combinations, called the *SSL Record Protocol*. Figure 22-49 on page 855 illustrates this and contrasts it with a standard HTTP socket connection. Note that this diagram shows SSL as providing a simple socket interface on which other applications can be layered. In reality, current implementations have the socket interface

embedded within the application and do not expose an API that other applications can use.

On the upper layer, a protocol for initial authentication and transfer of encryption keys, called the *SSL Handshake Protocol*.

API

socket

Client

Server

API

socket

**Session**

Standard HTTP

socket API

socket

API

Client

Server

socket

API

socket

API

**Session**

SSL

SSL Record Protocol

*Figure 22-49 SSL: Comparison of standard and SSL sessions*

An SSL session is initiated as follows:

1. On the client (browser), the user requests a document with a special URL that starts with https: instead of http:, either by typing it into the URL input field, or by clicking a link.
2. The client code recognizes the SSL request and establishes a connection through TCP port 443 to the SSL code on the server.
3. The client then initiates the SSL handshake phase, using the SSL Record Protocol as a carrier. At this point, there is no encryption or integrity checking built in to the connection.

The SSL protocol addresses the following security issues:

|  |  |
| --- | --- |
| **Privacy**  **Integrity** | After the symmetric key is established in the initial handshake, the messages are encrypted using this key.  Messages contain a message authentication code (MAC) ensuring the message integrity. |

**Authentication** During the handshake, the client authenticates the server using an asymmetric or public key. It can also be based on certificates.

SSL requires that each message is encrypted and decrypted and therefore has a high performance and resource cost.

##### Differences between SSL V2.0 and SSL V3.0

There is backward compatibility between SSL V2.0 and SSL V3.0. An SSL V3.0 server implementation should be able accept the connection request from an SSL V2.0 client. The main differences between SSL V2.0 and SSL V3.0 are as follows:

SSL V2.0 does not support client authentication.

SSL V3.0 supports more ciphering types in the CipherSpec.

#### 22.7.2 SSL protocol

The SSL protocol is located at the top of the transport layer. SSL is also a layered protocol itself. It simply takes the data from the application layer, reformats it, and transmits it to the transport layer. SSL handles a message as follows:

The sender performs the following tasks:

1. Takes the message from upper layer.
2. Fragments the data to manageable blocks.
3. Optionally compresses the data.
4. Applies a message authentication code (MAC).
5. Encrypts the data.
6. Transmits the result to the lower layer.

The receiver performs the following tasks:

1. Takes the data from lower layer.
2. Decrypts.
3. Verifies the data with the negotiated MAC key.
4. Decompresses the data if compression was used.
5. Reassembles the message.
6. Transmits the message to the upper layer.

An SSL session works in different states. These states are *session* and *connection* states. The SSL handshake protocol (see “SSL handshake protocol” on page 858) coordinates the states of the client and the server. In addition, there are read and write states defined to coordinate the encryption according to the change CipherSpec messages.

When either party sends a change CipherSpec message, it changes the pending write state to current write state. Again, when either party receives a change CipherSpec message, it changes the pending read state to the current read state.

The session state includes the following components:

|  |  |
| --- | --- |
| **Session identifier** | An arbitrary byte sequence chosen by the server to identify an active or resumable session state. |
| **Peer certificate** | Certificate of the peer. This field is optional; it can be empty. |
| **Compression method** | The compression algorithm. |
| **CipherSpec** | Specifies data encryption algorithm (such as null, DES) and a MAC algorithm. |
| **Master secret** | 48-byte shared secret between the client and the server. |
| **Is resumable** | A flag indicating whether the session can be used for new connections. |

The connection state includes the following components:

**Server and client random** An arbitrary byte sequence chosen by the client and server for each connection.

|  |  |
| --- | --- |
| **Server write MAC secret** | The secret used for MAC operations by the server. |
| **Client write MAC secret** | The secret used for MAC operations by the client. |
| **Server write key** | The cipher key for the server to encrypt the data and the client to decrypt the data. |
| **Client write key** | The cipher key for the client to encrypt the data and the server to decrypt the data. |
| **Initialization vectors** | Initialization vectors store the encryption information. |
| **Sequence numbers** | A sequence number indicates the number of the message transmitted since the last change  CipherSpec message. Both the client and the server maintain sequence numbers. |

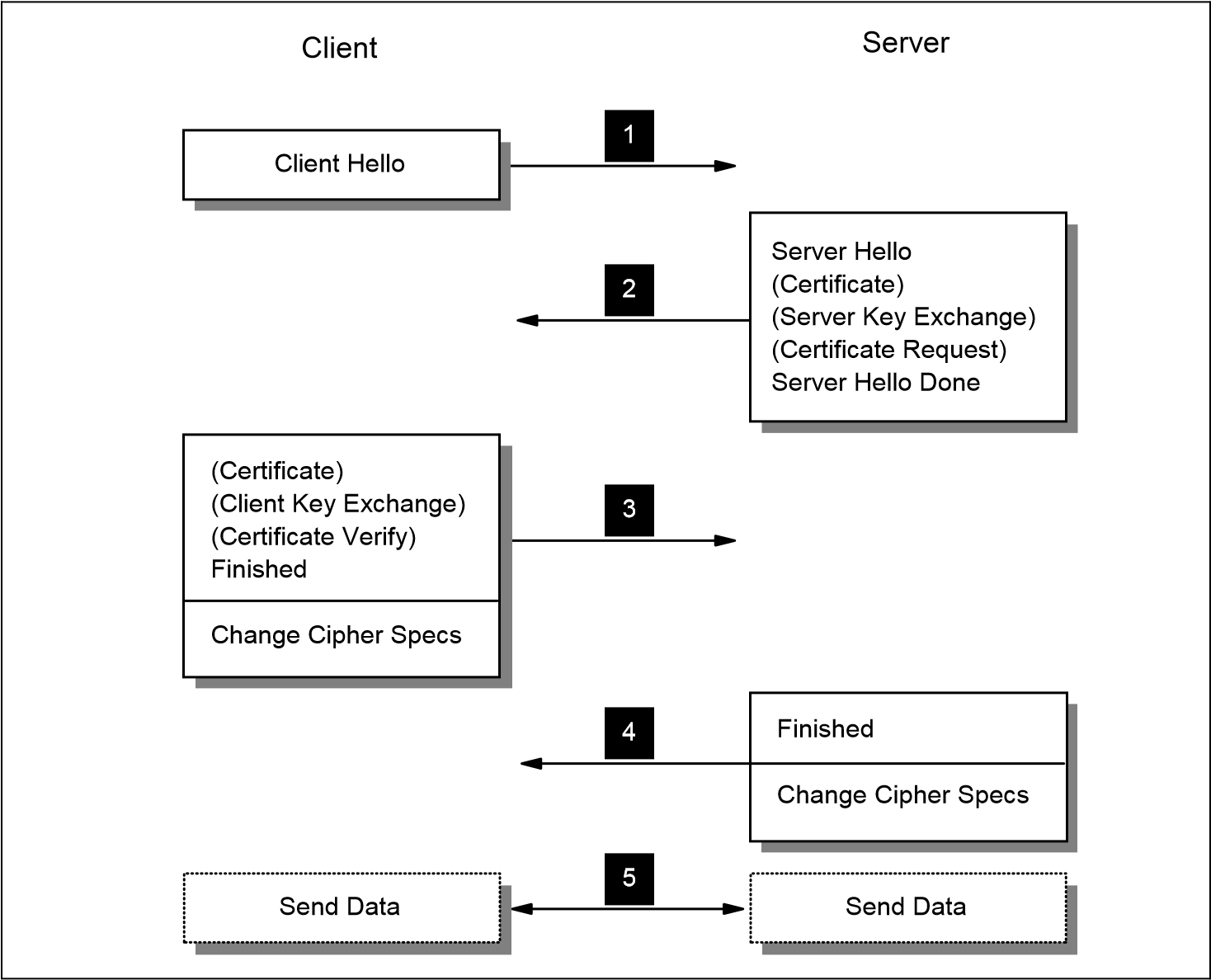
##### Change CipherSpec protocol

The change CipherSpec protocol is responsible for sending change CipherSpec

messages. At any time, the client can request to change current cryptographic parameters such as the handshake key exchange. Following the change CipherSpec notification, the client sends a handshake key exchange and if available, certificate verify messages, and the server sends a change CipherSpec message after processing the key exchange message. After that, the newly agreed keys will be used until the next change CipherSpec request. The change CipherSpec message is sent after the hello messages during the negotiation.

##### SSL handshake protocol

The SSL handshake protocol allows the client and server to determine the required parameters for an SSL connection such as protocol version, cryptographic algorithms, optional client or server authentication, and public key encryption methods to generate shared secrets. During this process, all handshake messages are forwarded to the SSL record layer to be encapsulated into special SSL messages. Figure 22-50 illustrates an SSL handshake process.



*Figure 22-50 SSL: Handshake process*

We explain the SSL handshake process detailed in Figure 22-50 in more detail:

1. The client sends a connection request with a client hello message. This message includes:
   * Desired version number.
   * Time information (the current time and date in standard UNIX 32-bit format).
   * Optionally, a session ID. If it is not specified the server will try to resume previous sessions or return an error message.
   * Cipher suites. (List of the cryptographic options supported by the client. These are authentication modes, key exchange methods, encryptions, and MAC algorithms.)
   * Compression methods supported by the client.
   * A random value.
2. The server evaluates the parameters sent by the client hello message and returns a server hello message that includes the following parameters that were selected by the server to be used for the SSL session:
   * Version number
   * Time information (the current time and date in standard UNIX 32-bit format)
   * Session ID
   * Cipher suite
   * Compression method
   * A random value

Following the server hello message, the server sends the following messages:

* + Server certificate if the server is required to be authenticated
  + A server key exchange message if there is no certificate available or the certificate is for signing only
  + A certificate request if the client is required to be authenticated

Finally, the server sends a server hello done message and begins to wait for the client response.

1. The client sends the following messages:
   * If the server has sent a certificate request, the client must send a certificate or a no certificate message.
   * If the server has sent a server key exchange message, the client sends a client key exchange message based on the public key algorithm determined with the hello messages.
   * If the client has sent a certificate, the client verifies the server certificate and sends a certificate verify message indicating the result.

The client then sends a finished message indicating the negotiation part is completed. The client also sends a change CipherSpec message to generate shared secrets. Note that this is not controlled by the handshake protocol; the change CipherSpec protocol manages this part of the operation.

1. The server sends a finished message indicating that the negotiation part is completed. The server then sends the change CipherSpec message.
2. Finally, the session partners separately generate an encryption key, in which they derive the keys to use in the encrypted session that follows from the master key. The handshake protocol changes the state to the connection state. All data taken from the application layer is transmitted as special messages to the other party.

There are significant additional processing costs associated with starting up an SSL session compared with a normal HTTP connection. The protocol avoids some of these costs by allowing the client and server to retain session key information and to resume that session without negotiating and authenticating a second time.

Following the handshake, both session partners have generated a master key. From that key, they generate other session keys, which are used in the symmetric-key encryption of the session data and in the creation of message digests. The first message encrypted in this way is the finished message from the server. If the client can interpret the finished message, it means:

Privacy has been achieved, because the message is encrypted using a symmetric-key bulk cipher (such as DES or RC4).

The message integrity is assured, because it contains a message authentication code (MAC), which is a message digest of the message itself plus material derived from the master key.

The server has been authenticated, because it was able to derive the master key from the pre-master key. Because this was sent using the server's public key, it can only be decrypted by the server (using its private key). Note that this relies on the integrity of the server's public key certificate.

##### SSL record protocol

After the master key has been determined, the client and server can use it to encrypt application data. The SSL record protocol specifies a format for these messages. In general, they include a message digest to ensure that they have not been altered and the whole message is encrypted using a symmetric cipher. Usually, this uses the RC2 or RC4 algorithm, although DES, triple-DES, and IDEA are also supported by the specification.

The U.S. National Security Agency (NSA), a department of the United States federal government, imposed restrictions on the size of the encryption key that can be used in software exported outside the U.S. These rules have been reviewed, but originally the key was limited to an effective size of 56 bits. The RC2 and RC4 algorithms achieved this by using a key in which all but 56 bits are set to a fixed value. International (export) versions of software products had this hobbled security built into them. SSL checks for mismatches between the export and nonexport versions in the negotiation phase of the handshake. For example, if a U.S. browser tries to connect with SSL to an export server, they will agree on export-strength encryption. See 22.2.7, “Export/import restrictions on cryptography” on page 793 for more information about recent changes of U.S. export regulations of cryptographic material.

### 22.8 Transport Layer Security (TLS)

The Transport Layer Security 1.0 protocol is based on SSL. The TLS 1.0 protocol is documented in RFC 2246. Two applications (without knowing each other’s code) can use TLS to communicate securely. There are no significant differences between SSL 3.0 and TLS 1.0. They can interoperate with some modifications of the message formats. A TLS 1.0 application can back down to an SSL 3.0 connection.

### 22.9 Secure Multipurpose Internet Mail Extension (S-MIME)

Secure Multipurpose Internet Mail Extension (S-MIME) can be thought of as a very specific SSL-like protocol. S-MIME is an application-level security construct, but its use is limited to protecting e-mail through encryption and digital signatures. It relies on public key technology, and uses X.509 certificates to establish the identities of the communicating parties. S-MIME can be implemented in the communicating end systems; it is not used by intermediate routers or firewalls.

### 22.10 Virtual private networks (VPNs) overview

The Internet has become a popular, low-cost backbone infrastructure. Its universal reach has led many companies to consider constructing a secure virtual private network (VPN) over the public Internet. The challenge in designing a VPN for today's global business environment will be to exploit the public Internet backbone for both intra-company and inter-company communication while still providing the security of the traditional private, self-administered corporate network.

In this chapter, we begin by defining a virtual private network (VPN) and explaining the benefits that clients can achieve from its implementation. After discussing the security considerations and planning aspects, we then describe the VPN solutions available in the market today.

#### 22.10.1 VPN introduction and benefits

With the explosive growth of the Internet, companies are beginning to ask: “How can we best exploit the Internet for our business?” Initially, companies were using the Internet to promote their company's image, products, and services by providing World Wide Web access to corporate Web sites. Today, however, the Internet potential is limitless, and the focus has shifted to e-business, using the global reach of the Internet for easy access to key business applications and data that reside in traditional IT systems. Companies can now securely, and cost-effectively, extend the reach of their applications and data across the world through the implementation of secure virtual private network (VPN) solutions.

A virtual private network (VPN) is an extension of an enterprise's private intranet across a public network such as the Internet, creating a secure private connection, essentially through a private *tunnel*. VPNs securely convey information across the Internet connecting remote users, branch offices, and business partners into an extended corporate network, as shown in Figure 22-51 on page 863. Internet service providers (ISPs) offer cost-effective access to the Internet (via direct lines or local telephone numbers), enabling companies to eliminate their current, expensive leased lines, long-distance calls, and toll-free telephone numbers.



A 1997 VPN Research Report, by Infonetics Research, Inc., estimates savings from 20% to 47% of wide area network (WAN) costs by replacing leased lines to remote sites with VPNs. And, for remote access VPNs, savings can be 60% to 80% of corporate remote access dial-up costs. Additionally, Internet access is available worldwide where other connectivity alternatives might not be available.

The technology to implement these virtual private networks, however, is just becoming standardized. Some networking vendors today are offering nonstandards-based VPN solutions that make it difficult for a company to incorporate all its employees and business partners/suppliers into an extended corporate network. However, VPN solutions based on Internet Engineering Task Force (IETF) standards will provide support for the full range of VPN scenarios with more interoperability and expansion capabilities.

The key to maximizing the value of a VPN is the ability for companies to evolve their VPNs as their business needs change and to easily upgrade to future TCP/IP technology. Vendors that support a broad range of hardware and software VPN products provide the flexibility to meet these requirements. VPN solutions today run mainly in the IPv4 environment, but it is important that they have the capability of being upgraded to IPv6 to remain interoperable with your business partner's and supplier's VPN solutions. Perhaps equally critical is the ability to work with a vendor that understands the issues of deploying a VPN. The implementation of a successful VPN involves more than technology. The vendor's networking experience plays heavily into this equation.

### 22.11 Kerberos authentication and authorization system

The Kerberos Network Authentication Service Version 5 is a *proposed standard protocol*. Its status is *elective* and described in RFC 1510.

According to *The Enlarged Devil's Dictionary*, by Ambrose Bierce, Kerberos is “the watchdog of Hades, whose duty it was to guard the entrance against whom or what does not clearly appear; Kerberos is known to have had three heads.”

A Kerberos service is normally run on its own system in a secure area. Users have to validate themselves to Kerberos before they are allowed to connect to other servers in the network. The server’s identities can also be checked against Kerberos.

The Kerberos Authentication and Authorization System is an encryption-based security system that provides mutual authentication between the users and the servers in a network environment. The assumed goals for this system are:

Authentication to prevent fraudulent requests/responses between users and servers that must be confidential and between groups of at least one user and one service.

Authorization can be implemented independently from the authentication by each service that wants to provide its own authorization system. The authorization system can assume that the authentication of a user/client is reliable.

Permits the implementation of an accounting system that is integrated, secure, and reliable, with modular attachment and support for “chargebacks” or billing purposes.

The Kerberos system is mainly used for authentication purposes, but it also provides the flexibility to add authorization information.

#### 22.11.1 Assumptions

Kerberos assumes the following:

The environment using this security system includes public and private workstations that can be located in areas with minimal physical security, a campus network without link encryption that can be composed of dispersed local networks connected by backbones or gateways, centrally operated servers in locked rooms with moderate physical security, and centrally operated servers with considerable physical security.

Confidential data or high-risk operations such as a bank transaction cannot be part of this environment without additional security, because after you have a workstation as a terminal, you can emulate certain conditions and normal data will be flowing without any encryption protection.

One of the cryptosystems used is the Data Encryption Standard (DES), which has been developed to be modular and replaceable by the Kerberos designers.

Kerberos assumes a loosely synchronized clock in the whole system, so the workstation has to have a synchronization tool such as the time server provided.

#### 22.11.2 Naming

A *principal identifier* is the name that identifies a client or a service for the Kerberos system.

In Version 4, the identifier consists of three components:

The *principal* name is unique for each client and service assigned by the Kerberos Manager.

The *instance* name used for distinct authentication is an added label for clients and services, which exist in several forms. For users, an instance can provide different identifiers for different privileges. For services, an instance usually specifies the host name of the machine that provides this service.

The *realm* name used to allow independently administered Kerberos sites. The principal name and the instance are qualified by the realm to which they belong, and are unique only within that realm. The realm is commonly the domain name.

In Version 4, each of the three components has a limit of 39 characters long. Due to conventions, the period (.) is not an acceptable character.

In Version 5, the identifier consists of two parts only, the *realm* and the *remainder*, which is a sequence of however many components are needed to name the principal. Both the realm and each component of the remainder are defined as ASN.1 (Abstract Syntax Notation One, ISO standard 8824) GeneralStrings. This puts few restrictions on the characters available for principal identifiers.

#### 22.11.3 Kerberos authentication process

In the Kerberos system, a client that wants to contact a server for its service, first has to ask for a *ticket* from a mutually trusted third party, the Kerberos Authentication Server (KAS). This ticket is obtained as a function where one of the components is a private key known only by the service and the Kerberos Authentication Server so that the service can be confident that the information on the ticket originates from Kerberos. The client is known to the KAS as a principal name (c). The private key (Kc) is the authentication key known only to the user and the Kerberos Authentication Server (KAS).

In this section, the symbol {X,Y} indicates a message containing information (or data) X and Y. {X,Y}Kz indicates that a message that contains the data X and Y has been enciphered using the key Kz.

Figure 22-52 shows the authentication process.

Client c

Kerberos

Authentication

Server (KAS)

Kerberos

Ticket

Granting

Server (TGS)

Kerberos

Database

1

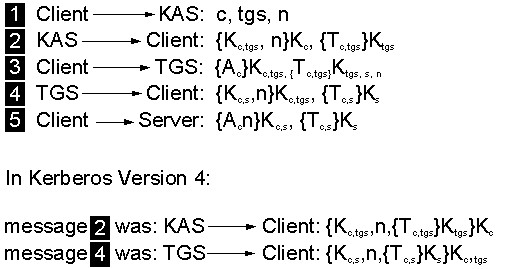
Server s

2

3

4

5



*Figure 22-52 Kerberos authentication scheme*

The authentication process consists of exchanging five messages (see Figure 22-52 on page 867):

1. Client → KAS

The client sends a message {c, tgs, n} to the KAS, containing its identity (c), a nonce (a time stamp or other means to identify this request), and requests for a ticket for use with the ticket-granting server (TGS).

1. KAS → client

The authentication server looks up the client name (c) and the service name (the ticket-granting server, tgs) in the Kerberos database and obtains an encryption key for each (Kc and Ktgs).

The KAS then forms a response to send back to the client. This response contains an initial ticket Tc,tgs, which grants the client access to the requested server (the ticket-granting server). Tc,tgs contains Kc,tgs, c, tgs, nonce, lifetime, and some other information. The KAS also generates a random encryption key Kc,tgs, called the session key. It then encrypts this ticket using the encryption key of the ticket-granting server (Ktgs). This produces what is called a *sealed ticket* {Tc,tgs}Ktgs. A message is then formed consisting of the sealed ticket and the TGS session key Kc,tgs.

|  |
| --- |
| **Note:** In Kerberos Version 4, the message is:  {Kc,tgs,n,{Tc,tgs}Ktgs}Kc  While in Kerberos Version 5, the message is of a simpler form: {Kc,tgs, n}Kc, {Tc,tgs}Ktgs  This simplifies the (unnecessary) double encryption of the ticket. |

1. Client → TGS

Upon receiving the message, the client decrypts it using its secret key Kc, which is only known to it and the KAS. It checks to see if the nonce (n) matches the specific request, and then caches the session key Kc,tgs for future communications with the TGS.

The client then sends a message to the TGS. This message contains the initial ticket {Tc,tgs}Ktgs, the server name (s), a nonce, and a new authenticator Ac containing a time stamp. Ac is {c, nonce}. The message is:

{Ac}Kc,tgs, {Tc,tgs}Ktgs, s, n

1. TGS → client

The ticket-granting server (TGS) receives the above message from the client (c), and first deciphers the sealed ticket using its TGS encryption key. (This ticket was originally sealed by the Kerberos authentication server in step 2 using the same key.) From the deciphered ticket, the TGS obtains the TGS-session-key. It uses this TGS session key to decipher the sealed authenticator. (Validity is checked by comparing the client name both in the ticket and in the authenticator, the TGS server name in the ticket, the network address that must be equal in the ticket, in the authenticator, and in the received message.)

Finally, it checks the current time in the authenticator to make certain the message is recent. This requires that all the clients and servers maintain their clocks within some prescribed tolerance. The TGS now looks up the server name from the message in the Kerberos database and obtains the encryption key (Ks) for the specified service.

The TGS forms a new random session key Kc,s for the benefit of the client (c) and the server (s), and then creates a new ticket Tc,s containing:

Kc,s, n, nonce, lifetime,

It then assembles and sends a message to the client.

|  |
| --- |
| **Note:** In Kerberos Version 4, the message is:  {Kc,s,n,{Tc,s}Ks}Kc,tgs  While in Kerberos Version 5, the message is of a simpler form: {Kc,s,n}Kc,tgs, {Tc,s}Ks  This simplifies the (unnecessary) double encryption of the ticket. |

5. Client → server

The client receives this message and deciphers it using the TGS session key that only it and the TGS share. From this message, it obtains a new session key Kc,s that it shares with the server (s) and a sealed ticket that it cannot decipher because it is enciphered using the server's secret key Ks.

The client builds an authenticator and seals it using the new session key Kc,s. At last, it sends a message containing the sealed ticket and the authenticator to the server (s) to request its service.

The server (s) receives this message and first deciphers the sealed ticket using its encryption key, which only it and KAS know. It then uses the new session key contained in the ticket to decipher the authenticator and does the same validation process that was described in step 4.

After the server has validated a client, an option exists for the client to validate the server. This prevents an intruder from impersonating the server. The client requires then that the server sends back a message containing the time stamp (from the client's authenticator, with one added to the time stamp value). This message is enciphered using the session key that was passed from the client to the server.

Let us summarize some of the central points in this scheme:

In order for the workstation to use any end server, a ticket is required. All tickets, other than the first ticket (also called the *initial ticket*), are obtained from the TGS. The first ticket is special; it is a ticket for the TGS itself and is obtained from the Kerberos authentication server.

Every ticket is associated with a session key that is assigned every time a ticket is allocated.

Tickets are reusable. Every ticket has a lifetime, typically eight hours. After a ticket has expired, you have to identify yourself to Kerberos again, entering your login name and password.

Unlike a ticket, which can be reused, a new authenticator is required every time the client initiates a new connection with a server. The authenticator carries a time stamp within it, and the authenticator expires a few minutes after it is issued. (This is the reason why clocks must be synchronized between clients and servers.)

A server maintains a history of previous client requests for which the time stamp in the authenticator is still valid. This way, a server can reject duplicate requests that might arise from a stolen ticket and authenticator.

#### 22.11.4 Kerberos database management

Kerberos needs a record of each user and service in its realm and each record keeps only the needed information, as follows:

Principal identifier (c,s)

Private key for this principal (Kc,Ks)

Date of expiration for this identity

Date of the last modification in this record

Identity of the principal that last modified this record (c,s)

Maximum lifetime of tickets to be given to this principal (lifetime)

Attributes (unused)

Implementation data (not visible externally)

The private key field is enciphered using a master key so that removing the database will not cause any problem because the master key is not in it.

The entity responsible for managing this database is the Kerberos Database Manager (KDBM). There is only one KDBM in a realm, but it is possible to have more than one Kerberos Key Distribution Server (KKDS), each one having a copy of the Kerberos database. This is done to improve availability and performance so that the user can choose one in a group of KKDSs to send its request to. The KKDS performs read-only operations, leaving the actualization to the KDBM, which copies the entire database a few times a day. This is done to simplify the operation using a Kerberos protected protocol. This protocol is basically a mutual authentication between KDBM and KKDS before a file transfer operation with checkpoints and checksum.

#### 22.11.5 Kerberos Authorization Model

The Kerberos Authentication Model permits only the service to verify the identity of the requester but it gives no information about whether the requester can use the service or not. The Kerberos Authorization Model is based on the principle that each service knows the user so that each one can maintain its own authorization information. However, the Kerberos Authentication System can be extended by information and algorithms that can be used for authorization purposes. (This is made easier in Version 5, as shown in the following section.) Kerberos can then check if a user/client is allowed to use a certain service.

Obviously, both the client and the server applications must be able to handle the Kerberos authentication process. That is, both the client and the server must be *kerberized*.

#### 22.11.6 Kerberos Version 5 enhancements

Kerberos Version 5 has a number of enhancements over Version 4. Some of the important ones are:

Use of encryption has been separated into distinct program modules which allows for supporting multiple encryption systems.

Network addresses that appear in protocol messages are now tagged with a type and length field. This allows support of multiple network protocols.

Message encoding is now described using the Abstract Syntax Notation 1 (ASN.1) syntax in accordance with ISO standards 8824 and 8825.

The Kerberos Version 5 ticket has an expanded format to support new features (for example, the inter-realm cooperation).

As mentioned in 22.11.2, “Naming” on page 865, the principal identifier naming has changed.

Inter-realm support has been enhanced.

Authorization and accounting information can now be encrypted and transmitted inside a ticket in the authorization data field. This facilitates the extension of the authentication scheme to include an authorization scheme as well.

A binding is provided for the Generic Security Service API (GSSAPI) to the Kerberos Version 5 implementation.

### 22.12 Remote access authentication protocols

Remote dial-in to the corporate intranet and to the Internet has made the remote access server a very vital part of today's internetworking services. More and more mobile users are requiring access not only to central-site resources, but to information sources on the Internet. The widespread use of the Internet and the corporate intranet has fueled the growth of remote access services and devices. There is an increasing demand for a simplified connection to corporate network resources from mobile computing devices, such as a notebook computer, or a palmtop device for e-mail access.

The emergence of remote access has caused significant development work in the area of security. The AAA (triple A) security model has evolved in the industry to address the issues of remote access security. Authentication, authorization, and accounting answers the questions who, what, and when, respectively. Here we provide a brief description of each of the three As in the AAA security model:

|  |  |
| --- | --- |
| **Authentication** | This is the action of determining who a user (or entity) is. Authentication can take many forms. Traditional authentication uses a name and a fixed password. Most computers work this way. However, fixed passwords have limitations, mainly in the area of security. Many modern authentication mechanisms utilize one-time passwords or a challenge-response query. Authentication generally takes place when the user first logs in to a machine or requests a service of it. |
| **Authorization** | This is the action of determining what a user is allowed to do. Generally, authentication precedes authorization, but again, this is not required. An authorization request might indicate that the user is not authenticated. (we do not know who they are.) In this case, it is up to the authorization agent to determine if an unauthenticated user is allowed the services in question. In current remote authentication protocols, authorization does not merely provide yes or no answers, but it can also customize the service for the particular user. |

**Accounting** This is typically the third action after authentication and authorization. But again, neither authentication nor authorization are required. Accounting is the action of recording what a user is doing and has done.

In the distributed client/server security database model, a number of communications servers, or clients, authenticate a dial-in user's identity through a single, central database, or authentication server. The authentication server stores all information about users, their passwords, and access privileges. Distributed security provides a central location for authentication data that is more secure than scattering the user information on different devices throughout a network. A single authentication server can support hundreds of communications servers, serving up to tens of thousand of users.

Communications servers can access an authentication server locally or remotely over WAN connections.

Several remote access vendors and the Internet Engineering Task Force (IETF) have been in the forefront of this remote access security effort, and the means whereby such security measures are standardized. Remote Authentication Dial In User Service (RADIUS) and Terminal Access Controller Access Control System (TACACS) are two such cooperative ventures that have evolved out of the Internet standardizing body and remote access vendors.

Remote Authentication Dial-In User Service (RADIUS) is a distributed security system developed by Livingston Enterprises. RADIUS was designed based on a previous recommendation from the IETF's Network Access Server Working Requirements Group. An IETF Working Group for RADIUS was formed in January 1996 to address the standardization of RADIUS protocol; RADIUS is now an IETF-recognized dial-in security solution (RFC 2058 and RFC 2138).

Similar to RADIUS, Terminal Access Controller Access Control System

(TACACS) is an industry standard protocol specification, RFC 1492. Similar to RADIUS, TACACS receives an authentication request from an NAS client and forwards the user name and password information to a centralized security server. The centralized server can either be a TACACS database or an external security database. Extended TACACS (XTACACS) is a version of TACACS with extensions that Cisco added to the basic TACACS protocol to support advanced features. TACACS+ is another Cisco extension that allows a separate access server (the TACACS+ server) to provide independent authentication, authorization, and accounting services.

Although RADIUS and TACACS authentication servers can be set up in a variety of ways, depending on the security scheme of the network they are serving, the basic process for authenticating a user is essentially the same. Using a modem, a remote dial-in user connects to a remote access server (also called the network access server or NAS) with a built-in analog or digital modem. After the modem connection is made, the NAS prompts the user for a name and password. The NAS then creates the so-called authentication request from the supplied data packet, which consists of information identifying the specific NAS device sending the authentication request, the port that is being used for the modem connection, and the user name and password.

For protection against eavesdropping by hackers, the NAS, acting as the RADIUS or TACACS client, encrypts the password before it sends it to the authentication server. If the primary security server cannot be reached, the security client or NAS device can route the request to an alternate server. When an authentication request is received, the authentication server validates the request and then decrypts the data packet to access the user name and password information. If the user name and password are correct, the server sends an authentication acknowledgment packet. This acknowledgement packet can include additional filters, such as information on the user's network resource requirements and authorization levels. The security server can, for instance, inform the NAS that a user needs TCP/IP or IPX using PPP, or that the user needs SLIP to connect to the network. It can include information about the specific network resource that the user is allowed to access.

To circumvent snooping in the network, the security server sends an authentication key, or signature, identifying itself to the security client. After the NAS receives this information, it enables the necessary configuration to allow the user the necessary access rights to network services and resources. If at any point in this log-in process all necessary authentication conditions are not met, the security database server sends an authentication reject message to the NAS device and the user is denied access to the network.

### 22.13 Extensible Authentication Protocol (EAP)

*Extensible Authentication Protocol* (EAP) is used for the exchange of authentication information. EAP is defined in RFC 2284 and is an extension to the Point-to-Point Protocol (PPP). EAP supports multiple authentication vehicles such as:

Kerberos

Public key authentication

Key tokens

One time passwords

EAP typically runs over the link layer and has a number of deployment solutions including:

EAP MD5

EAP-Tunneled TLS (EAP-TTLS)

Lightweight EAP (LEAP)

Protected EAP (PEAP)

When used in wireless communications, IEEE 802.1x defines how EAP is encapsulated in LAN frames. The wireless EAP solution is typically activated when a user connects to wireless access point (AP) and enters in authentication credentials. The AP verifies the identity of the user through a RADIUS server and, if the credentials are approved, access is granted to the user.

For further EAP details, refer to Chapter 23, “Port based network access control” on page 889.

### 22.14 Layer 2 Tunneling Protocol (L2TP)

L2TP permits the tunneling of PPP. Any protocol supported by PPP can be tunneled. This protocol extends the span of a PPP connection. Instead of beginning at the remote host and ending at a local ISP's point of presence (PoP), the *virtual PPP* link now extends from the remote host all the way back to the corporate gateway. L2TP tunneling is currently supported over IP/UDP. The specification is in RFC 2661.

L2TP is a consensus standard that came from the merging of two earlier tunneling protocols: Point-to-Point Tunneling Protocol (PPTP) and Layer 2 Forwarding (L2F, described in RFC 2341). These earlier protocols did not provide as complete a solution as the L2TP protocol; one addresses tunnels created by ISPs and the other addresses tunnels created by remote hosts. L2TP supports both host-created and ISP-created tunnels.

L2TP adds the ability to create a virtual private network where multiple protocols and privately addressed IP, IPX, and AppleTalk (AT) are allowed. In addition, L2TP gives remote users the ability to connect to a local ISP and tunnel through the Internet to a home network, avoiding long distance charges. It also provides a mechanism on which to solve the multiple box PPP multilink problem. (Calls connecting to different physical routers that are destined for the same MP bundle can be tunneled to the same endpoint where MP can be terminated for all links.)

#### 22.14.1 Terminology

Before describing the protocol, we provide a definition of some L2TP terminology

L2TP access concentrator (LAC)

A device attached to one or more public switched telephone network (PSTN)

or Integrated Services Digital Network (ISDN) lines capable of handling both the PPP operation and L2TP protocol. The LAC implements the media over which L2TP operates. L2TP passes the traffic to one or more L2TP servers (LNS).

L2TP network server (LNS)

An LNS operates on any platform that can be a PPP endstation. The LNS handles the server side of the L2TP protocol. Because L2TP relies only on the single media over which L2TP tunnels arrive, the LNS can have only a single LAN or WAN interface, yet is still able to terminate calls arriving from any PPP interfaces supported by a LAC, such as async, synchronous, ISDN, V.120, and so on.

Network access servers (NAS)

A device providing temporary, on demand network access to users. This access is point-to-point using PSTN or ISDN lines.

Session (Call)

L2TP creates a session when an end-to-end PPP connection is attempted between a dial-in user and the LNS, or when an outbound call is initiated. The datagrams for the session are sent over the tunnel between the LAC and the LNS. The LNS and LAC maintain the state information for each user attached to a LAC.

Tunnel

A tunnel is defined by an LNS-LAC pair. The tunnel carries PPP datagrams between the LAC and the LNS. A single tunnel can multiplex many sessions. A control connection operating over the same tunnel controls the establishment, release, and maintenance of all sessions and of the tunnel itself.

Attribute value air (AVP)

A uniform method of encoding message types and bodies. This method maximizes the extensibility while permitting interpretability of L2TP.

#### 22.14.2 Protocol overview

Because the host and the gateway share the same PPP connection, they can take advantage of PPP's ability to transport protocols other than just IP. For example, L2TP tunnels can support remote LAN access as well as remote IP access. Figure 22-53 outlines a basic L2TP configuration.

Internet

ISP

LNS

LAC

Dial

Connection

L2TP Tunnel

PPP Connection

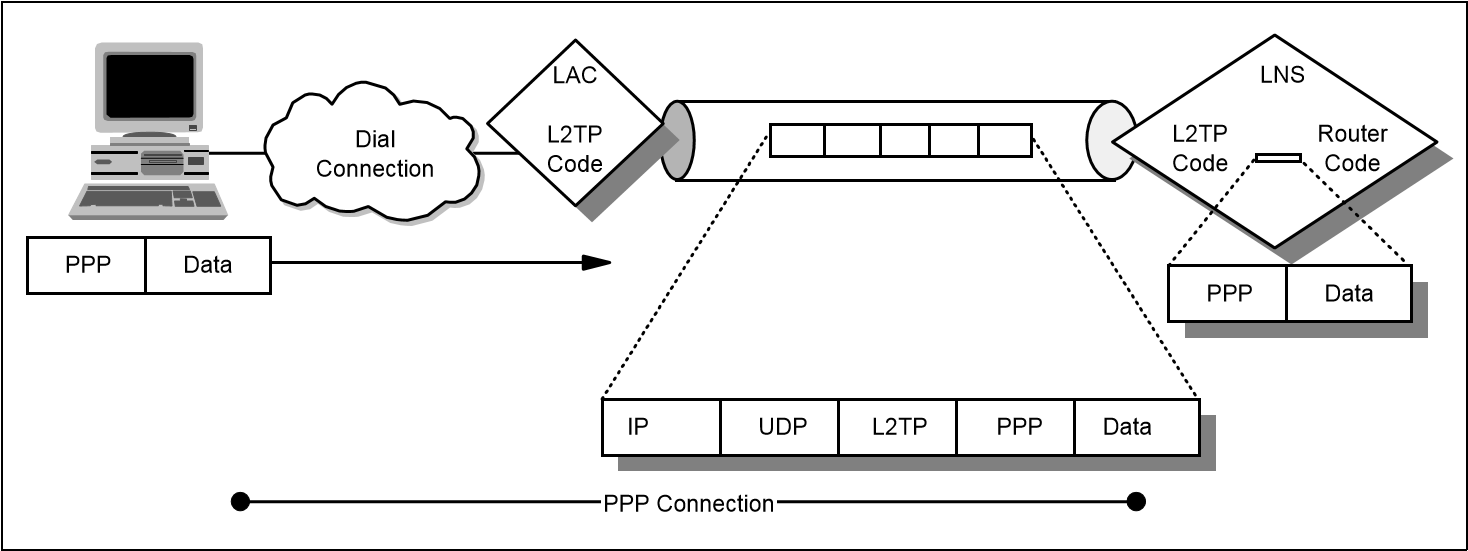
*Figure 22-53 Layer 2 Tunnel Protocol (L2TP) scenario*

Referring to Figure 22-53, the following actions occur:

1. The remote user initiates a PPP connection.
2. The NAS accepts the call.
3. The NAS identifies the remote user using an authorization server.
4. If the authorization is OK, the NAS/LAC initiates an L2TP tunnel to the desired LNS at the entry to the enterprise.
5. The LNS authenticates the remote user through its authentication server and accepts the tunnel.
6. The LNS confirms acceptance of the call and the L2TP tunnel.
7. The NAS logs the acceptance.
8. The LNS exchanges PPP negotiation with the remote user.
9. End-to-end data is now tunneled between the remote user and the LNS.

L2TP is actually another variation of an IP encapsulation protocol. As shown in Figure 22-54 on page 878, an L2TP tunnel is created by encapsulating an L2TP frame inside a UDP packet, which in turn is encapsulated inside an IP packet whose source and destination addresses define the tunnel's endpoints. Because the outer encapsulating protocol is IP, clearly IPSec protocols can be applied to

this composite IP packet, thus protecting the data that flows within the L2TP tunnel. AH, ESP, and ISAKMP/Oakley protocols can all be applied in a straightforward way.



*Figure 22-54 L2TP packet changes during transit*

L2TP can operate over UDP/IP and support the following functions:

Tunneling of single user dial-in clients

Tunneling of small routers, for example, a router with a single static route to set up based on an authenticated user's profile

Incoming calls to an LNS from a LAC

Multiple calls per tunnel

Proxy authentication for PAP and CHAP

Proxy LCP

LCP restart in the event that proxy LCP is not used at the LAC

Tunnel endpoint authentication

Hidden AVP for transmitting a proxy PAP password

Tunneling using a local realm (that is, user@realm) lookup table

Tunneling using the PPP user name lookup in the AAA subsystem (22.12,

“Remote access authentication protocols” on page 872)

UDP

IP

L2TP

PPP

Data

Data

L2

Net

L2TP

Code

IP

Code

IP Cloud

Data

L2

Net

L2TP

Code

IP

Code

*Figure 22-55 L2TP packet flow through any IP cloud*

#### 22.14.3 L2TP security issues

Although L2TP provides cost-effective access, multiprotocol transport, and remote LAN access, it does not provide cryptographically robust security features. For example:

Authentication is provided only for the identity of tunnel endpoints, but not for each individual packet that flows inside the tunnel. This can expose the tunnel to man-in-the-middle and spoofing attacks.

Without per-packet integrity, it is possible to mount denial-of-service attacks by generating bogus control messages that can terminate either the L2TP tunnel or the underlying PPP connection.

L2TP itself provides no facility to encrypt user data traffic. This can lead to embarrassing exposures when data confidentiality is an issue.

While the payload of the PPP packets can be encrypted, the PPP protocol suite does not provide mechanisms for automatic key generation or for automatic key refresh. This can lead to someone listening in on the wire to finally break that key and gain access to the data being transmitted.

Realizing these shortcomings, the PPP Extensions Working Group of the IETF considered how to remedy these shortfalls. Rather than duplicate work done

elsewhere, it was decided to recommend using IPSec within L2TP. This is described in RFC 2888.

In summary, Layer 2 Tunnel Protocols are an excellent way of providing cost-effective remote access. And when used in conjunction with IPSec, they are an excellent technique for providing secure remote access. However, without complementary use of IPSec, an L2TP tunnel alone does not furnish adequate security.

### 22.15 Secure Electronic Transaction (SET)

SET is the outcome of an agreement by MasterCard International and Visa International to cooperate on the creation of a single electronic credit card system. Prior to SET, each organization had proposed its own protocol and each had received support from a number of networking and computing companies. Now, most of the major players are behind the SET specification (for example, IBM, Microsoft, Netscape, and GTE).

The following sections describes at a high level the components and processes that make up the specification. Refer to the MasterCard and Visa home pages for more information about SET.

#### 22.15.1 SET roles

The SET specification defines several roles involved in the payment process:

|  |  |
| --- | --- |
| **The merchant** | This is any seller of goods, services, or information. |
| **The acquirer** | This is the organization that provides the credit card service and keeps the money flowing. The most widely known acquirers are MasterCard and Visa. |
| **The issuer** | This is the organization that issued the card to the purchaser in the first place. Usually, this is a bank or some other financial institution, which would know the purchaser best. |
| **The cardholder** | This is the Web surfer, who has been given a credit card by the issuer and now wants to exercise his or her purchasing power on the Web. |

##### The acquirer payment gateway

This provides an interface between the merchant and the bankcard network used by the acquirer and the issuer. It is important to remember that the bankcard network already exists. The acquirer payment gateway provides a well-defined, secure interface to that established network from the Internet. Acquirer payment gateways will be operated on behalf of the acquirers, but they might be provided by third-party organizations, such as Internet service providers (ISPs).

**The certificate authority**SET processing uses public key cryptography, so each element of the system need one or more public key certificates. Several layers of CA are described in the specification. (We discuss SET certificates in 22.15.3,

“The SET certificate scheme” on page 883.)

#### 22.15.2 SET transactions

The SET specification describes a number of transaction flows for purchasing, authentication, payment reversal, and so on. Figure 22-56 shows the transactions involved in a typical online purchase.

*Figure 22-56 Typical SET transaction sequence*

PInitReq

PInitRes

PReq

AuthReq

AuthRes

PRes

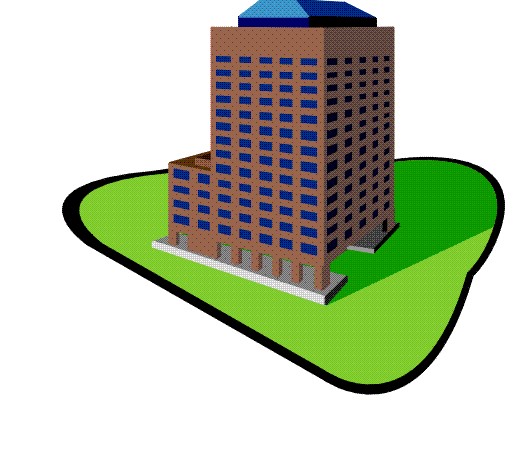
InqReq

InqRes

CapReq

CapRes

Cardholder



Merchant

*MasterCard*

*International*

*Mas*

*~~terC~~*

*ard*

Acquirer

Gateway

1

2

3

4

5

The diagram shows the following transactions (each transaction consists of a request/response pair):

1. PInit

This initializes the system, including details such as the brand of card being used and the certificates held by the cardholder. SET does not insist that cardholders have signing certificates, but it does recommend them. A cardholder certificate binds the credit card account number to the owner of a public key. If the acquirer receives a request for a given card number signed with the cardholder's public key, it knows that the request came from the real cardholder. To be precise, it knows that the request came from a computer where the cardholder's keyring was installed and available. It *could* still be a thief who had stolen the computer and cracked the keyring password.

1. Purchase order

This is the request from the cardholder to buy something. The request message is in fact two messages combined, the order instruction (OI), which is sent in the clear to the merchant, and the purchase instruction (PI), which the merchant passes on to the acquirer payment gateway. The PI is encrypted in the public key of the acquirer, so the merchant cannot read it. The merchant stores the message for later transmission to the acquirer. The PI also includes a hash of the OI, so the two messages can only be handled as a pair. Note that the card number is only placed in the PI portion of the request. This means that the merchant never has access to it, thereby preventing a fraudulent user from setting up a false store front to collect credit card information.

The purchase order has a response, which is usually sent (as shown here) after acquirer approval has been granted. However, the merchant can complete the transaction with the cardholder before authorization, in which case the cardholder would see a message that the request was accepted pending authorization.

1. Authorization

In this request, the merchant asks the acquirer, through the acquirer payment gateway, to authorize the request. The message includes a description of the purchase and the cost. It also includes the PI from the purchase order that the cardholder sent. In this way, the acquirer knows that the merchant and the cardholder both agree on what is being purchased and the amount.

When the acquirer receives the request, it uses the existing bank card network to authorize the request and sends back an appropriate response.

1. Inquiry

The cardholder might want to know how his or her request is proceeding. The SET specification provides an inquiry transaction for that purpose.

1. Capture

Up to this point, no money has changed hands. The capture request from the merchant tells the acquirer to transfer the previously authorized amount to its account.

In fact, capture can be incorporated as part of the authorization request/response (see the previous information). However, there are situations in which the merchant might want to capture the funds later. For example, most mail order operations do not debit the credit card account until the goods have been shipped.

There are several other transactions within the SET specification, but the previous summary shows the principles on which it is based.

#### 22.15.3 The SET certificate scheme

The SET specification envisions hundreds of thousands of participants worldwide. Potentially, each of these would have at least one public key

certificate. In fact, the protocol calls for an entity to have multiple certificates in some cases. For example, the acquirer payment gateways need one for signing messages and another for encryption purposes.

Key management on such a large scale requires something beyond a simple, flat certification structure. The organization of certifying authorities proposed for SET is shown in Figure 22-57.

Cardholder

CA

Cardholder

CA

Cardholder

CA

Cardholder

Cardholder

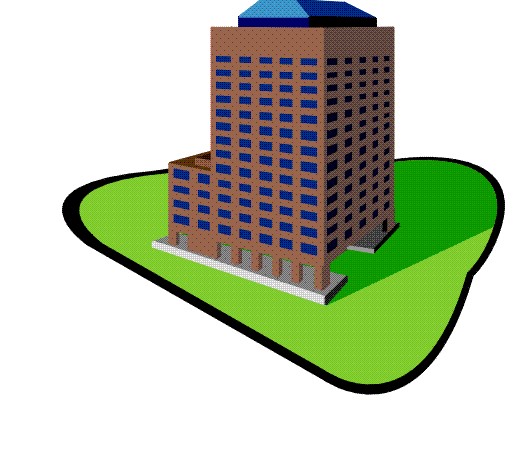
CA

Cardholder

CA

Merchant

CA



Merchant

Cardholder

CA

Cardholder

CA

Payment

CA

*ard*

*~~terC~~*

*Mas*

Acquirer

Gateway

Root

CA

Brand

CA

Geo-Political CA

optional

)

(

*Figure 22-57 SET certifying authorities*

At the top of the certificate chain, the root certifying authority is to be kept offline under extremely tight arrangements. It will only be accessed when a new credit card brand joins the SET consortium. At the next level in the hierarchy, the brand level CAs are also very secure. They are administered independently by each credit card brand.

There is some flexibility permitted under each brand for different operating policies. It would be possible to set up CAs based on region or country, for example. At the base of the CA hierarchy are the CAs that provide certificates for merchants, cardholders, and acquirer payment gateways. The SET specification provides protocols for merchants and cardholders to request certificates online. It is important to have a simple process because SET aims to encourage cardholders to have their own certificates. It envisions the cardholder surfing to the CA Web site, choosing a Request Certificate option to invoke the certificate request application on the browser, and then filling in a form to send and receive the certificate request.

Of course, if the system allows certificates to be created easily, it must also be able to revoke them easily in the event of a theft or other security breach. The SET specification includes some certificate update and revocation protocols for this purpose. Although the mechanism for requesting a certificate might be simple, there is still a need for user education. For example, it is obvious that a cardholder needs to notify the credit card company if his or her wallet is stolen, but less obvious that he or she also needs to notify them if his or her computer is stolen. However, if the computer includes his keyring file containing the private key and certificate, it might allow the thief to go shopping at the cardholder's expense.

### 22.16 RFCs relevant to this chapter

The following RFCs provide detailed information about the TCP/IP security solutions presented in this chapter:

RFC 1492 – An Access Control Protocol, Sometimes Called TACACS (July 1993)

RFC 1579 – Firewall-Friendly FTP (February 1994)

RFC 1928 – SOCKS Protocol Version 5 (March 1996)

RFC 1929 – Username/Password Authentication for SOCKS V5 (March 1996)

RFC 1961 – GSS-API Authentication Method for SOCKS Version 5 (June 1996)

RFC 2003 – IP Encapsulation within IP (October 1996)

RFC 2104 – HMAC: Keyed-Hashing for Message Authentication (February 1997)

RFC 2138 – Remote Authentication Dial In User Service (RADIUS) (April 1997)

RFC 2315 – PKCS 7: Cryptographic Message Syntax Version 1-5 (March 1998)

RFC 2403 – The Use of HMAC-MD5-96 within ESP and AH

(November 1998)

RFC 2404 – The Use of HMAC-SHA-1-96 within ESP and AH (November 1998)

RFC 2405 – The ESP DES-CBC Cipher Algorithm With Explicit IV (November 1998)

RFC 2407 – The Internet IP Security Domain of Interpretation for ISAKMP

(November 1998)

RFC 2410 – The NULL Encryption Algorithm and Its Use With IPSec

(November 1998)

RFC 2411 – IP Security Document Roadmap (November 1998)

RFC 2412 – The OAKLEY Key Determination Protocol (November 1998)

RFC 2661 – Layer Two Tunneling Protocol “L2TP” (August 1999)

RFC 2888 – Secure Remote Access with L2TP (August 2000)

RFC 2986 – PKCS #10: Certification Request Syntax Specification Version

1.7 (November 2000)

RFC 3022 – The IP Network Address Translator (NAT) (January 2001)

RFC 3162 – Radius and IPv6 (August 2001)

RFC 3174 – US Secure Hash Algorithm 1 (SHA1) (September 2001)

RFC 3207 – SMTP Service Extension for Secure SMTP over Transport Layer Security (February 2002)

RFC 3365 – Strong Security Requirements for Internet Engineering Task

Force Standard Protocols (August 2002)

RFC 3447 - Public-Key Cryptography Standards (PKCS) #1: RSA

Cryptography Specifications Version 2.1 (February 2003)

RFC 3514 – The Security Flag in the IPv4 Header (April 2003)

RFC 3586 – IP Security Policy (IPSP) Requirements (August 2003)

RFC 3686 – Using Advanced Encryption Standard (AES) Counter Mode With

IPSec Encapsulating Security Payload (ESP) (January 2004)

RFC 3711 – The Secure Real-time Transport Protocol (SRTP) (March 2004)

RFC 3715 – IPSec-Network Address Translation (NAT) Compatibility

Requirements (March 2004)

RFC 3748 – Extensible Authentication Protocol (EAP) (June 2004)

RFC 3749 – Transport Layer Security Protocol Compression Methods

(May 2004)

RFC 3750 – Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Certificate Handling (April 2004)

RFC 3751 – Secure/Multipurpose Internet Mail Extensions (S/MIME) Version

3.1 Message Specification (April 2004)

RFC 3852 – Cryptographic Message Syntax (CMS) (July 2004)

RFC 3871 – Operational Security Requirements for Large Internet Service

Provider (ISP) IP Network Infrastructure (September 2004)

RFC 4033 – DNS Security Introduction and Requirements (March 2005)

RFC 4050 – The Secure Shell (SSH) Protocol Assigned Numbers (April 2005)

RFC 4051 – The Secure Shell (SSH) Protocol Architecture (April 2005)

RFC 4052 – The Secure Shell (SSH) Authentication Protocol (April 2005)

RFC 4053 – The Secure Shell (SSH) Transport Layer Protocol (April 2005)

RFC 4054 – The Secure Shell (SSH) Connection Protocol (May 2005)

RFC 4055 – Using DNS to Securely Publish Secure Shell (SSH) Key

Fingerprints (June 2005)

RFC 4056 – Generic Message Exchange Authentication for the Secure Shell

Protocol (SSH) (June 2005)

RFC 4120 – The Kerberos Network Authentication Service (V5) (July 2005)

RFC 4301 – Security Architecture for the Internet Protocol (December 2005)

RFC 4302 – IP Authentication Header (December 2005)

RFC 4303 – IP Encapsulating Security Payload (ESP) (December 2005)

RFC 4306 – Internet Key Exchange (IKEv2) Protocol (December 2005)

RFC 4344 – The Secure Shell (SSH) Transport Layer Encryption Modes

RFC 4346 – The Transport Layer Security (TLS) Protocol Version 1.1 (April 2006)

RFC 4366 – Transport Layer Security (TLS) Extensions (April 2006)

RFC 4470 – Minimally Covering NSEC Records and DNSSEC On-line Signing (April 2006)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **24** | |

## Chapter 24. Availability, scalability, and load balancing

This chapter discusses the various availability, scalability, and load balancing techniques used within enterprises in an attempt to ensure continuous data flow and minimize outages.

This chapter describes the following topics:

Availability

Scalability

Load balancing

Clustering

Virtualization

Virtual Router Redundancy Protocol (VRRP)

Round-robin DNS

Alternate solutions to load balancing

© Copyright IBM Corp. 1989-2006. All rights reserved.

The Internet business has grown so rapidly that continuous availability of mission-critical data and applications residing on servers is a very important requirement for enterprises. The Internet challenges companies to develop new strategies for increasing revenue and providing detailed product and delivery information. The result raises client and business partner satisfaction. Also, the management of enterprises' internal business processes that are accessed through the Internet by the workforce must be better optimized.

Increasing demands from client, business partners, and employees for access to applications and data are challenges for the development of new server and networking strategies and services. Consider the following three main aspects:

How can availability to an enterprise's information be made available 24 hours a day and 7 days a week?

How can these services also be guaranteed even if the number of transactions increases very rapidly, for example, because of a spike in client or business partner inquiries?

How can the access to server applications and data be shared among parallel installed servers?

The answers are availability, scalability, and load balancing. In this chapter, we discuss techniques that can be employed to achieve availability, scalability, and load balancing. We discuss each technique at a fairly high level.

### 24.1 Availability

Application instances, network interfaces, and machines can fail (planned for maintenance or unplanned due to application or system error). In these cases, users must not lose their service. Recovering from application instance failure is fairly straightforward in that the application is simply restarted. Network interface failures can also be tolerated by making use of a virtual IP address, which is not tied to any particular physical interface and thus will never fail.

A *virtual IP address* can be given to a device that has one or more network interfaces. This allows the users’ machines to pick up a specific IP associated with a specific machine or device. However, the IP address given is not tied to the physical IP address of the device’s interfaces. Therefore, if one of the interfaces fails, the users are unaware of the failure.

Machine failure, however, is a bit more complex. Users must be able to immediately reconnect to the service without knowing that they now are using an alternate image of the application on another system. Users also must not be aware that the path to the other system has been automatically changed. The use of virtualization can be very advantageous with regards to increasing the availability of a system. Virtualization is discussed further later.

### 24.2 Scalability

Scalability means to provide a solution for a growing business that requires additional system capacity. When workload capacity becomes smaller due to many more new connection requests from clients or business partners, a nondisruptive growth of the current system environment must be made available.

In a traditional single system environment (no clustered systems), a nondisruptive upgrade of systems is relatively limited. In order to raise capacity, these systems have to be taken down to install new features. Therefore, they are not available for a certain time.

The implementation of clustered systems is a better approach (discussed in more depth later). Adding a new system to the cluster running equal applications instances does not impact the other systems in the cluster. This solution adds seamless capacity for a growing business. Compared to traditional systems, the user is not bound to a given system in a clustered server environment.

Therefore, the management of user connections to servers is more flexible. When a new system comes online, new connections are directed to that machine taking over a new workload.

Generic techniques to enhance scalability include clustering, virtualization, and the monitoring of devices to ensure that if certain resource thresholds are met, the resources are upgraded. We discuss clustering and virtualization in more detail later.

### 24.3 Load balancing

Assigning applications with user connections to a specific system can overload this system's capacity, while other systems with fewer connection requests to other applications might waste free capacity.

To reach the goal for an equal level of load of all systems, these systems must be organized in a clustered system group. All systems in this cluster can provide information about their workload to the load balancing device. This device will now be responsible for distributing connection requests from users to the systems of the application servers, based on workload information.

Users are not aware of such clusters. They try to connect to a service, assuming it is running in the machine of the load balancer. The load balancer forwards the connection request to the real service provider based on the current workload of all systems in the cluster. The information about the state of the workload can be provided by a function, such as a workload manager residing in every target system.

If there is no workload information from target systems, the load balancer can use distribution rules, such as:

A simple round-robin distribution

Number of distributed connections

We discuss techniques used to assist with or provide load balancing, scalability, and availability next.

### 24.4 Clustering

In order to provide the referenced availability requirement, another system organization has to be applied. This leads to running multiple application instances on multiple machines, including TCP/IP stacks with parallel connections to the TCP/IP network. This solution, called the clustering technique in general terms, is used for load balancing purposes but is also valid for solving high availability requirements.

The clustering technique dispatches connections to target servers, excluding failed servers, from a list of target servers that can receive connections. In this way, the dispatching function avoids routing connections to a server that is not capable of satisfying such a request.

The clustering technique requires the implementation of equal application instances running on different machines. If the application, the operating system with TCP/IP stack, or the machine fails, the dispatching technique immediately provides a backup.

A user requesting service from a particular server would no longer address an application in a particular server but now would address a group of servers. The connection request is now sent to the dispatcher, who decides to which available application server it is forwarded. Therefore, users are not aware to which application server (within the group) they are connected.

The clustering technique requires addresses that refer to groups of applications. This can be solved through virtual IP addresses. A virtual IP address (VIPA) is the IP address of a group of application servers, for example, a Telnet server. This VIPA is used for a connection request. The dispatcher is the receiver of the connection request from the user. It selects from a list of available servers a real server and forwards the request to this server.

The process of selecting an available application server may be extended by the dispatcher by using different kind distribution rules. The distribution of connection requests will be discussed in the load balancing section.

Another aspect of availability to consider is when the dispatcher fails. In this case, a backup dispatcher has to be implemented with the same IP address so that users can send their connection requests to the backup dispatcher. A backup dispatcher also propagates its IP address to the network. Therefore, routers use the new path that directs the user’s connection requests to the backup dispatcher.

If dispatchers maintain client/server connections, the backup dispatcher has to take over the currently running connections. A takeback process must be implemented to return running connections to the primary dispatcher.

Virtualization is also a technique used to provide availability and scalability. Virtualization has similarities to the clustering technique with regards to transparency shown to the users regarding which physical machine is being used as well as the there being no impact to the users if a machine is to fail. We discuss virtualization next.

### 24.5 Virtualization

Virtualization is the logical representation of resources not inhibited by physical

boundaries. The main objective of virtualization is to simplify the IT infrastructure. It simplifies access to resources and the management of those resources. A user accesses the required service through standard interfaces supported and maintained by the virtualized resource. The standard interfaces allows availability issues to be minimized when changes to the IT infrastructure occur.

There many types of virtualization, and we describe some in the following sections.

For additional information, refer to:

The IBM developerWorks® article “Virtualization in a nutshell: A pattern point of view” <http://www.ibm.com/developerworks/grid/library/gr-virt/>

The IBM Redpaper *Virtualization and the On Demand Business*, REDP-9115 <http://www.redbooks.ibm.com/redpapers/pdfs/redp9115.pdf>

#### *Server virtualization*

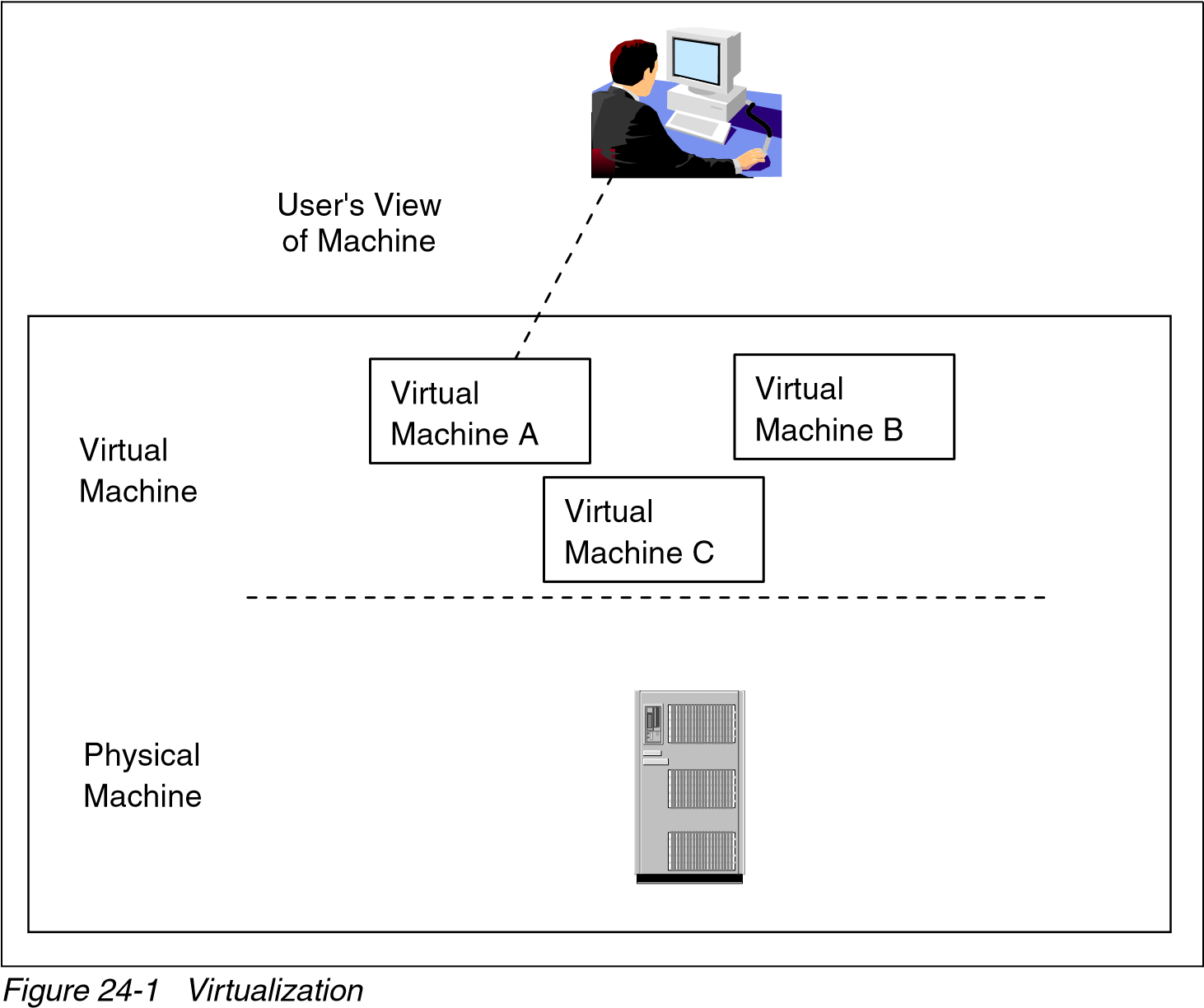
Many applications cannot be hosted on the same physical server due to resource conflicts. This creates issues regarding the number of servers deployed as well as the utilization of the existing resources. This lack of utilization is expensive, especially considering the cost of wasted storage space, server processing ability, and network utilization. Server virtualization is one way to resolve these issues.

Server virtualization is used to detach the applications from the physical configurations and limitations. Server virtualization is generally used as an IT optimization technique and has numerous benefits regarding availability and scalability.

Server virtualization provides the flexibility to dynamically change the allocation of system resources for the virtualized environments. The virtual servers can run on any of the physical machines. This means that the machine resources are fully shared. This makes it possible to run the physical server at high utilization levels. In addition, if any of the underlying physical resources need to be changed, it does not affect the virtualized servers. This enhances the level of scalability and availability associated with each virtual server.

Another aspect of availability to consider is if one of the virtual server instances fails. In such a case, it does not affect any of the other virtual servers currently residing on the same physical machine. Each instance of the virtual servers is completely isolated from each other. This also eliminates any security issues or concerns regarding data leakage. Each instance of the server is also kept as a file, which can then easily be copied onto a new virtual server if the instance fails. This assists with the time taken to recover the virtual servers and the overall availability of the service provided.

As shown in Figure 24-1, many virtual servers are running off of one physical server. This also illustrates how the users are unaware that the server being used is a virtual one as opposed to a physical one.



#### *Storage virtualization*

Storage virtualization is the combination of the capacity of multiple storage controllers into a single resource with a single view of the storage resources. This virtual layer between the physical storage devices and the users or host application provides the ability to conceal the physical infrastructure from the application and user. A benefit of storage virtualization is the ability to add, upgrade, or remove space and disks without the applications or the users service being affected.

#### *Network virtualization*

Network virtualization enables administrators to manage portions of a network that might be shared among different enterprises as virtual networks, while still

continuing to preserve the isolation of traffic and resource utilization. Network

virtualization also enables the administrator to prioritize traffic across the network

to ensure the optimum performance for vital business applications and processes. This includes technologies such as virtual private networks (VPNs), HiperSockets™, virtual networks, and virtual LANs.

### 24.6 Virtual Router Redundancy Protocol (VRRP)

Virtual Router Redundancy Protocol (VRRP) was issued to the IETF by IBM, Ascend Communications, Microsoft, and Digital Equipment Corporation in April 1998 and is documented in RFC 3768. Its status is a proposed standard.

#### 24.6.1 Introduction

The use of a statically configured default route is quite popular for host IP configurations. It minimizes configuration and processing inefficiencies on the end host and is supported by virtually every IP implementation. This mode of operation is likely where dynamic host configuration protocols (such as 3.7, “Dynamic Host Configuration Protocol (DHCP)” on page 130) are deployed, which typically provide configuration for an end host IP address and default gateway. However, this creates a single point of failure. Loss of the default router results in a catastrophic event, isolating all end hosts that are unable to detect any alternate path that may be available. Figure 24-2 illustrates VRRP.

Host

Host

Router

B

Router

Backbone

Network

Router

A

Server

Backup

Client

VRRP

*Figure 24-2 An illustration of VRRP*

VRRP is designed to eliminate the single point of failure inherent in the static, default, routed environment. VRRP specifies an election protocol that dynamically assigns responsibility for a virtual router to one of the VRRP routers on a LAN. The VRRP router controlling the IP addresses associated with a virtual router is called the master, and it forwards packets sent to these IP addresses. The election process provides dynamic fail-over in the forwarding responsibility if the master becomes unavailable. Any of the virtual router's IP addresses on a LAN can then be used as the default first hop router by end hosts. The advantage gained from using VRRP is a higher availability default path without requiring configuration of dynamic routing or router discovery protocols on every end host (see router discovery protocols in 3.2, “Internet Control Message Protocol (ICMP)” on page 109).

#### 24.6.2 VRRP definitions

Some terms used in VRRP are:

|  |  |
| --- | --- |
| **VRRP router** | A router running the Virtual Router Redundancy |
| **Virtual router**  **IP address owner**  **Primary IP address**  **Virtual router master**  **Virtual router backup** | Protocol. It can participate in one or more virtual routers.  An abstract object managed by VRRP that acts as a default router for hosts on a shared LAN. It consists of a virtual router identifier and a set of associated IP addresses depending on the definition, across a common LAN. A VRRP router can back up one or more virtual routers.  The VRRP router that has the virtual router's IP addresses as real interface addresses. This is the router that, when up, responds to packets addressed to one of these IP addresses for ICMP pings, TCP connections, and so on.  An IP address selected from the set of real interface addresses. One possible selection algorithm is to always select the first address. VRRP advertisements are always sent using the primary IP address as the source of the IP packet.  The VRRP router that is assuming the responsibility of forwarding packets sent to the IP addresses associated with the virtual router and answering ARP requests for these IP addresses. Note that if the IP address owner is available, it will always become the master.  The set of VRRP routers available to assume forwarding responsibility for a virtual router if the current master fails. |

#### 24.6.3 VRRP overview

VRRP specifies an election protocol to provide the virtual router function described earlier. All protocol messaging is performed using IP multicast datagrams (see Chapter 6, “IP multicast” on page 237), thus the protocol can operate over a variety of multiaccess LAN technologies supporting an IP multicast. Each VRRP virtual router has a single well-known MAC address allocated to it. The virtual router MAC address is used as the source in all periodic VRRP messages sent by the master router to enable bridge learning in an extended LAN.

A virtual router is defined by its virtual router identifier (VRID) and a set of IP addresses. A VRRP router can associate a virtual router with its real addresses on an interface and can also be configured with additional virtual router mappings and priority for virtual routers it is willing to back up. The mapping between VRID and addresses must be coordinated among all VRRP routers on a LAN. However, there is no restriction against reusing a VRID with a different address mapping on different LANs.

The scope of each virtual router is restricted to a single LAN. To minimize network traffic, only the master for each virtual router sends periodic VRRP advertisement messages. A backup router will not attempt to preempt the master unless it has higher priority. This eliminates service disruption unless a more preferred path becomes available. It is also possible to administratively prohibit all preemption attempts. The only exception is that a VRRP router will always become master of any virtual router associated with addresses it owns. If the master becomes unavailable, the highest priority backup will transition to master after a short delay, providing a controlled transition of the virtual router responsibility with minimal service interruption.

The VRRP protocol design provides rapid transition from master to backup to minimize service interruption and incorporates optimizations that reduce protocol complexity while guaranteeing controlled master transition for typical operational scenarios. The optimizations result in an election protocol with minimal runtime state requirements, minimal active protocol states, and a single message type and sender. The typical operational scenarios are defined to be two redundant routers or distinct path preferences among each router, or both. A side effect when these assumptions are violated (for example, more than two redundant paths all with equal preference) is that duplicate packets can be forwarded for a brief period during the master election. However, the typical scenario assumptions are likely to cover the vast majority of deployments, loss of the master router is infrequent, and the expected duration in master election convergence is quite small (< 1 second). Therefore, the VRRP optimizations represent significant simplifications in the protocol design while incurring an insignificant probability of brief network degradation.

#### 24.6.4 Sample configuration

Figure 24-3 shows a simple example network with two VRRP routers implementing one virtual router.

*Figure 24-3 VRRP simple configuration example*

Host-2

Host-1

Host-3

Host-4

RTR-1

VRID=1 (master)

RTR-2

VRID=1

9.180.20.4

9.180.20.3

This configuration shows a very simple VRRP scenario. In this configuration, the end hosts install a default route to the IP address of virtual router #1 (IP address 9.180.20.3) and both routers run VRRP. The router on the left becomes the master for virtual router #1 (VRID=1), and the router on the right is the backup for virtual router #1. If the router on the left fails, the other router takes over virtual router #1 and its IP addresses, and provides uninterrupted service for the hosts. Note that in this example, IP address 9.180.20.4 is not backed up by the router on the left. IP address 9.180.20.4 is only used by the router on the right as its interface address. In order to back up IP address 9.180.20.4, a second virtual router needs to be configured. This is shown in Figure 24-4.

Host-2

Host-1

Host-3

Host-4

RTR-1

VRID=1 (master)

RTR-2

VRID=1

9.180.20.4

9.180.20.3

*Figure 24-4 VRRP simple load-splitting configuration example*

Figure 24-4 on page 918 shows a configuration with two virtual routers with the hosts splitting their traffic between them. This example is expected to be very common in actual practice. In this configuration, half of the hosts install a default route to virtual router #1 (IP address of 9.180.20.3), and the other half of the hosts install a default route to virtual router #2 (IP address o f 9.180.20.4). This has the effect of load balancing the traffic from the hosts through the routers, while also providing full redundancy.

#### 24.6.5 VRRP packet format

The purpose of the VRRP packet is to communicate to all VRRP routers the priority and the state of the master router associated with the virtual router ID. VRRP packets are sent encapsulated in IP packets. They are sent to the IPv4 multicast address assigned to VRRP. The IP address, as assigned by the IANA for VRRP, is 224.0.0.18. This is a link local scope multicast address. Routers must not forward a datagram with this destination address regardless of its TTL (see 3.1, “Internet Protocol (IP)” on page 68). The TTL must be set to 255. A VRRP router receiving a packet with the TTL not equal to 255 must discard the packet. Figure 24-5 shows the VRRP packet format.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 4 8 16 24 31   |  |  |  |  |  | | --- | --- | --- | --- | --- | | vers | type | virtual router ID | priority | count IP addrs | | auth type | | advert int | checksum |  | | IP address (1) | | | | | | . . . | | | | | | authentication data (1) | | | | | | authentication data (2) | | | | | |

*Figure 24-5 VRRP packet format*

The fields of the VRRP header are defined as follows:

**Version** The version field specifies the VRRP protocol version of this packet. (In RFC 3768, the version is 2.)

**Type** The type field specifies the type of this VRRP packet. The only packet type defined in this version of the protocol is 1.

**Virtual router ID (VRID)** The virtual router identifier (VRID) field identifies the virtual router for which this packet is reporting the status.

**Priority** The priority field specifies the sending VRRP router's

priority for the virtual router. Higher values equal higher priorities. The priority value for the VRRP router that owns the IP addresses associated with the virtual router must be 255. VRRP routers backing up a virtual router must use priority values between 1-254. The default priority value for VRRP routers backing up a virtual router is 100. The priority value zero (0) has special meaning, indicating that the current master has stopped participating in VRRP. This is used to trigger backup routers to quickly transition to master without having to wait for the current master to time out.

**Count IP addrs** The number of IP addresses contained in this VRRP advertisement.

**Auth type** The authentication type field identifies the authentication method being utilized. Authentication type is unique on a per interface basis. The authentication type field is an 8-bit unsigned integer. A packet with unknown authentication type or that does not match the locally configured authentication method must be discarded. The authentication methods currently defined are: 0 - No authentication

1. - Simple text password
2. - IP authentication header

##### Advertisement interval (Adver Int)

The default is 1 second. This field can be used for

troubleshooting misconfigured routers.

**Checksum** This is used to detect data corruption in the VRRP message.

|  |  |
| --- | --- |
| **IP address(es)** | One or more IP addresses that are associated with the virtual router. |
| **Authentication data** | The authentication string is currently only used for simple text authentication. |

### 24.7 Round-robin DNS

Early solutions to address load balancing were often located at the point where host names are translated into actual IP addresses: the Domain Name System (see 12.1, “Domain Name System (DNS)” on page 426). By rotating through a table of alternate IP addresses for a specific service, some degree of load balancing is achieved. This approach is often called round-robin DNS. The advantages of this approach are that it is protocol-compliant and transparent both to the client and the destination host. Also, it is performed only once at the start of the transaction.

Unfortunately, this approach is sometimes defeated because intermediate name servers and client software (including some of the most popular browsers) cache the IP address returned by the DNS service and ignore an expressly specified time-to-live (TTL) value (see 3.1, “Internet Protocol (IP)” on page 68), particularly if the TTL is short or zero. As a result, the balancing function provided by the DNS is bypassed, because the client continues to use a cached IP address instead of resolving it again. Even if a client does not cache the IP address, basic round-robin DNS still has limitations:

It does not provide the ability to differentiate by port.

It has no awareness of the availability of the servers.

It does not take into account the workload on the servers.

RFC 1794 discusses DNS support for load balancing and mentions round-robin DNS.

### 24.8 Alternative solutions to load balancing

There are many vendors currently offering load balancing hardware or software products. The techniques used vary widely and have advantages and disadvantages.

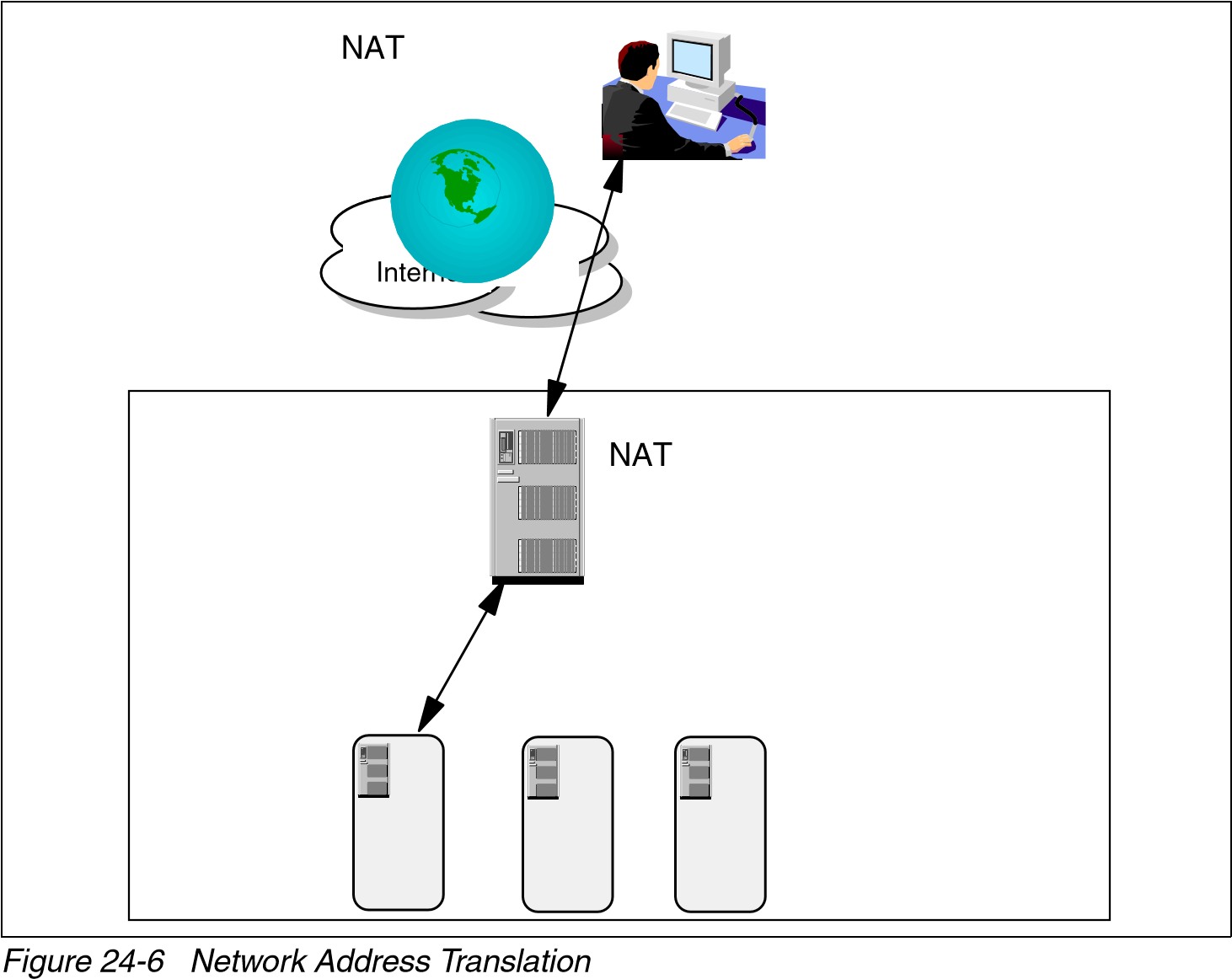
#### 24.8.1 Network Address Translation

Network Address Translation (NAT) works by modifying the source and target IP addresses in the inbound client-to-server packets and restoring the IP address to the original values in the outbound server-to-client packets. (Refer to 3.1.7,

“Network Address Translation (NAT)” on page 89.)

Note that if NAT is to be transparent to the server, eliminating the need for specialized agent code on the server, all packets sent back to the client must pass back through the load balancer in order to restore the IP addresses originally used by the client, as shown in Figure 24-6. This is a significant inefficiency, which will have a varying impact on the load balancer and the servers whose resources it manages.

This added processing charge and latency can mean network delay, and queuing delay in the load balancer itself. This drastically limits the potential scalability of NAT solutions. To overcome such delays, the capacity of the load balancer must not only be sufficient to handle both inbound and outbound packets, but also be able to cope with the disproportionately higher volume of outbound traffic.



As shown in Figure 24-6 on page 922, NAT offerings sometimes enforce the need to see both inbound and outbound requests by obliging the NAT device to be installed as a bridge (without permitting bridges of any other kind), thus forcing the servers on to what is essentially a private segment. This can complicate installation, because it requires a significant physical change to existing networks. All traffic for those servers must pass through the load balancer whether the traffic is to be load balanced or not.

The one advantage of NAT as originally conceived (the ability to forward packets to remote destinations across a wide area network) cannot be usefully deployed because the wide area network connection is behind the bridge and, therefore, can only be within the site's private network. Additionally, the same NAT device must still be the only exit from the wide area network link.

To attempt to overcome these limitations, some NAT solutions add to the overall inefficiency that is fundamental to NAT by providing add-ons. For example, the capability to map one port address to another. Refer to “Network Address Port Translation (NAPT)” on page 93.

To check if a server is up, NAT-based load balancing solutions need to sacrifice an actual client request, and so a server outage is typically perceived only as a result of a timeout of one of these real client requests.

NAT devices often only map affinity or *stickiness* based on the client's IP address, and not at the port level. This means that after a client has contacted a server, all traffic from that client that is intended for other applications is forwarded to the same server. This drastically restricts configuration flexibility, in many cases rendering the sticky capability unusable in the real world.

#### 24.8.2 Encapsulation

Another approach to load balancing is proxies that encapsulate packets rather than modifying them, and then pass them to the server. This approach has some merit, particularly because it permits the load balancer to forward traffic across a wide area network, unlike the bridging NAT solutions. But other implementations use encapsulation for all traffic, and this requires an agent of the load balancer to be installed on each server. This agent reverses the encapsulation process. As a result, the choice of server platform is, by definition, restricted to the platforms for which the server agent is available. Also, like NAT, it entails further processing of

the packet, which increases the likelihood that it will not be scalable to the levels required for major sites.

Encapsulation is discussed in further detail in RFC 1701, which appears to be updated by RFC 2784.

### 24.9 RFCs relevant to this chapter

The following RFCs provide detailed information about availability, scalability, and load balancing as presented throughout this chapter:

[RFC 1794 – DNS Support for Load Balancing (April 1995)](http://www.ietf.org/rfc/rfc1794.txt)

[RFC 3768 – Virtual Router Redundancy Protocol (VRRP) (April 2004)](http://www.ietf.org/rfc/rfc3768.txt)

[RFC 2784 – Generic Routing Encapsulation (GRE) (March 2000)](http://www.ietf.org/rfc/rfc2784.txt)

[RFC 1701 – Generic Routing Encapsulation (GRE) (October 1994)](http://www.ietf.org/rfc/rfc1701.txt)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **24** | |

## Appendix A. Multiprotocol Label Switching

This chapter provides an overview of the Multiprotocol Label Switching (MPLS) process and Generalized MPLS (GMPLS) architecture.

This chapter describes the following topics:

The ideas behind processing MPLS

An explanation of the differences between conventional, connectionless routing and flow routing

A definition of the terminology for MPLS concepts

A summary of the benefits of MPLS flow routing

A review of the details of MPLS protocols

An introduction to GMPLS

© Copyright IBM Corp. 1989-2006. All rights reserved.

### A.1 MPLS: An introduction

The idea behind MPLS was to emulate some property of circuit-switching network over a packet network, and to strike a happy middleground between extreme connection-oriented switching and pure connectionless routing service.

The theory for the idea of “mixing types to meet in the middle” was based on the observation that a sequence of correlated packets exist for stream and multimedia applications. We wanted to process them in the same routing path in a uniform fashion in order to guarantee quality of service (QoS). And we did not want to repeatedly examine and process those packet headers. The idea is feasible because we note that the headers in those related packets are the same or similar because those related packets in a stream desire consistent and similar processing treatment.

Multiprotocol Label Switching (MPLS) follows the same idea and comes up with new techniques to make a pseudo (and short-term) connection in a path (or subpath) for a sequence of correlated IP packets. The technology was proposed in RFC 3031. The Multiprotocol Label Switching (MPLS) standard represents the effort in the continued evolution of multilayer switching.

Generalized MPLS or GMPLS extends MPLS to encompass time-division (for example, SONET/SDH), wavelength (lambdas), and spatial switching (for example, incoming port or fiber to outgoing port or fiber). The focus of GMPLS is on the control plane of these various layers to dynamically provision resources and to provide network survivability using protection and restoration techniques.

#### A.1.1 Conventional routing versus MPLS forwarding mode

First, let us describe how the new paradigm shift might help in improving QoS performance by comparing conventional routing to the MPLS forwarding mode.

In an MPLS environment, conventional layer-3 or network-layer routing (that is, IP routing) is used to determine a path through the network. After the path is determined, data packets are then switched through each node as they traverse the network.

##### Conventional routing mode

In a traditional, connectionless network, every router runs a layer-3 routing algorithm. As a packet traverses through the network, each router along the path makes an independent forwarding decision for that packet. Using information contained in the packet header, as well as information obtained from the routing algorithm, the router chooses a “next hop” destination for the packet. In an IP network, this process involves matching the destination address stored in the IP header of each packet with the most specific route obtained from the IP routing table. This comparison process determines the next hop destination for the packet. This analysis and classification of the layer-3 header can be processor-intensive. In a traditional connectionless environment, this activity occurs at every node along the end-to-end path.

##### MPLS forwarding model

In an MPLS environment, optimum paths through the network are identified in advance. Then, as data packets enter the MPLS network, ingress devices use information in the layer-3 header to assign the packets to one of the predetermined paths. This assignment is used to append a *label* referencing the end-to-end path into the packet. The label accompanies the data packet as it traverses the network. Subsequent routers along the path use the information in the label to determine the next hop device. Because these devices only manipulate information in the label, processor-intensive analysis and classification of the layer-3 header occurs only at the ingress point.

#### A.1.2 Benefits

In additional to reducing the processing requirements on devices in the core of the network, MPLS has a number of additional advantages over conventional layer-3 routing, which we describe in the following sections.

##### Traffic engineering

Traffic engineering is the process of selecting network paths so that the resulting traffic patterns achieve a balanced utilization of resources.

Routing based on conventional Interior Gateway Protocol (IGP) algorithms might select network paths that result in unbalanced resource utilization. In these environments, some network resources are overused, while others are underused. A limited degree of engineering can be provided by manipulating the IGP metrics associated with network links. However, this effort is difficult to manage in environments with a large number of redundant paths.

To achieve the benefits of traffic engineering, MPLS can be used in-conjunction with IGP algorithms. MPLS provides the ability to specify the specific route data packets should use to traverse the network. This explicit routing of data packets ensures that a particular stream of data uses a specific path. By monitoring and managing these data streams, efficient utilization of network resources can be achieved. Explicit routing has been available through the source routing options of traditional IP routing. However, because this is a processor-intensive activity, its usage has been limited. MPLS makes the efficient use of explicit routing possible.

MPLS also provides the ability to analyze fields outside the IP packet header when determining the explicit route for a data packet. For example, the network administrator can develop traffic flow policies based on how or where a packet entered the network. In a traditional network, this information is only available at the ingress point. The additional analysis provides the administrator with a higher level of control, resulting in a more predictable level of service.

##### Quality of service routing

QoS routing is the ability to choose a route for a particular data stream so that the path provides a desired level of service. These levels of service can specify acceptable levels of bandwidth, delay, or packet loss in the network. This provides the intelligence to deliver different levels of service based on overall network policies.

Providing a network path delivering a desired QoS often requires the use of explicit routing. For example, it is straightforward to allocate a path for a particular stream requiring a specific bandwidth allocation. However, it is possible that the combined bandwidth of multiple streams may exceed existing capacity. In this scenario, individual streams, even those between the same ingress and egress nodes, might need to be individually routed. This requires a finer level of granularity than that provided by standard traffic engineering.

There are two approaches to providing QoS routing in an MPLS environment:

The MPLS label contains class of service (CoS) information. As traffic flows through the network, this information can be used to intelligently prioritize traffic at each network hop.

The MPLS network can provision multiple paths between ingress and egress devices. Each path is engineered to provide a different level of service. Traffic is then intelligently assigned to an appropriate path as it enters the network.

These approaches simply classify packets into a class of service category. Local network administration policies determine the service provided to each category.

##### Multiprotocol support

The Multiprotocol Label Switching standard provides support for existing network-layer protocols, including IPv4, IPv6, IPX, and AppleTalk. The standard also provides link layer support for Ethernet, token ring, FDDI, ATM, frame relay, and point-to-point links. Activities continue to extend this standard to other protocols and network types.

MPLS is not limited to a specific link layer technology; it can function on any media over which network layer packets can pass.

#### A.1.3 Terminology

The following sections define the terms that are used with MPLS.

##### Forwarding equivalency class (FEC)

An FEC is a group of layer-3 packets that are forwarded in the same manner. All packets in this group follow the same network path and have the same prioritization. Packets within an FEC can have different layer-3 header information. However, to simply make a forwarding decision, these packets are indistinguishable.

Common examples of FEC groups are:

A set of packets that have the same most specific route in the IP routing table.

A set of packets that have the same most specific route in the IP routing table and the same IP type of service setting.

In an MPLS network, an FEC is identified by a label.

##### Label and labeled packet

As stated previously, a label identifies a unique FEC. MPLS devices forward all identically labeled packets in the same way.

A label is locally significant between a pair of MPLS devices. It represents an agreement between the two devices describing the mapping between a label and an FEC. The fact that labels are locally significant enhances the scalability of MPLS into large environments, because the same label need not be used at every hop.

The MPLS label can be located at different positions in the data frame, depending on the layer-2 technology used for transport. If the layer-2 technology supports a label field, the MPLS label is encapsulated in the native label field. In an ATM network, the VPI/VCI fields can be used to store an MPLS label. Similarly, the DLCI field can be used to store an MPLS label in frame relay networks.

If the layer-2 technology does not natively support a label, the MPLS label resides in an encapsulation header appended specifically for this purpose. The header is located between the layer-2 header and the IP header. This use of a dedicated header permits MPLS service over any layer-2 technology (see

Figure A-1 on page 930, which depicts the 32-bit MPLS header).

TTL

CoS

Label

S

20

bits

8

bits 1 bit

3

bits

Layer 2

Header

MPLS

Header

IP

Header

Payload

*Figure A-1 The 32-bit MPLS header*

The contents of the MPLS header include:

A label field that contains the value of the MPLS label.

A CoS field that can be used to affect the queuing and discard algorithms applied to the packet as it traverses the network.

A S (stack) field that supports a hierarchical label stack.

A TTL (time-to-live) field that supports conventional IP TTL functionality.

A labeled packet is a packet into which a label has been encoded. To support enhanced MPLS functions, the packet might contain more than one label. This is known as a label stack. The stack establishes an ordered relationship between individual labels. The stack is implemented using the last-in, first-out model. This feature is further discussed in A.2.3, “Label stack and label hierarchies” on page 934.

##### Label stack router (LSR)

A label stack router is an MPLS node that is also capable of forwarding native layer-3 packets. There are two important types of LSRs in an MPLS network:

An *ingress node* connects the MPLS network with a node that does not execute MPLS functionality. The ingress node handles traffic as it enters the MPLS network.

An *egress node* connects the MPLS network with a node that does not execute MPLS functionality. The egress node handles traffic as it leaves the MPLS network.

##### Next hop label forwarding entry (NHLFE)

An NHLFE is used by an MPLS node to forward packets. There is at least one NHLFE for each FEC flowing through the node. Each node is responsible for maintaining an NHLFE information base containing the following information:

The packet’s next hop address

The operation performed on the label stack:

* Replace the label at the top of the stack with a specified new label. This is known as *popping* the old label and *pushing* a new label.
* Pop the label at the top of the stack.
* Replace the label at the top of the stack with a specified new label, and then push one or more specified new labels onto the label stack. When this action is complete, the stack will contain at least two MPLS labels.

The data link encapsulation used to transmit the packet (optional)

The label stack encoding used to transmit the packet (optional)

Any other information needed in order to properly process the packet

##### Incoming label map (ILM)

The ILM is used by an MPLS node to forward labeled packets. The label in an incoming packet is used as a reference to the ILM. The ILM information allows the node to select a set of NHLFEs containing forwarding instructions.

The ILM can map a label to a group of NHLFEs. This provides the ability to load balance over multiple equal-cost paths.

##### FEC-to-NHLFE map (FTN)

The FTN is used by an MPLS node to process packets that arrive unlabeled, but need to be labeled before forwarding. An unlabeled data packet is assigned a specific FEC at the ingress MPLS node. This FEC is used as a reference to the FTN. The FTN map allows the node to select a set of NHLFEs containing forwarding instructions. This activity is performed at the ingress node of the MPLS network.

The FTN can map a label to a group of NHLFEs. This provides the ability to load balance over multiple equal cost paths.

##### Label swapping

Label swapping is the process used by an MPLS node to forward a data packet to the next hop device. This process is used regardless of whether the packet arrives labeled or unlabeled. The process is similar to the method used in ATM and frame relay networks to forward traffic through a virtual circuit.

##### Label switched path (LSP)

An LSP represents a set of MPLS nodes traversed by packets belonging to a specific FEC. The set is an ordered, unidirectional list. Traffic flows from the node at the head-end of the list toward the node at the tail-end of the list.

##### Label stack and label hierarchies

A labeled packet can contain more than one label. The labels are maintained in a last-in, first-out stack. The stack implements an ordered hierarchy among the set of labels.

This hierarchy is used when an MPLS node delivers a packet to a partner MPLS node, but the nodes are not consecutive routers on the hop-by-hop path for the packet. In this situation, a tunnel is created between the two MPLS nodes. The tunnel is implemented as an LSP and label switching is used to forward traffic through the tunnel.

### A.2 MPLS network processing

The primary goal of MPLS is the integration of label swapping paradigms with traditional network layer routing. This integration bring efficiencies in data forwarding as well as positioning the network for advanced QoS functions.

#### A.2.1 Label swapping

Label swapping is the process used by an MPLS node to forward a data packet to the next hop device. This process is used regardless of whether the packet arrives labeled or unlabeled. The process is similar to the method used in ATM and frame relay networks to forward traffic through a virtual circuit.

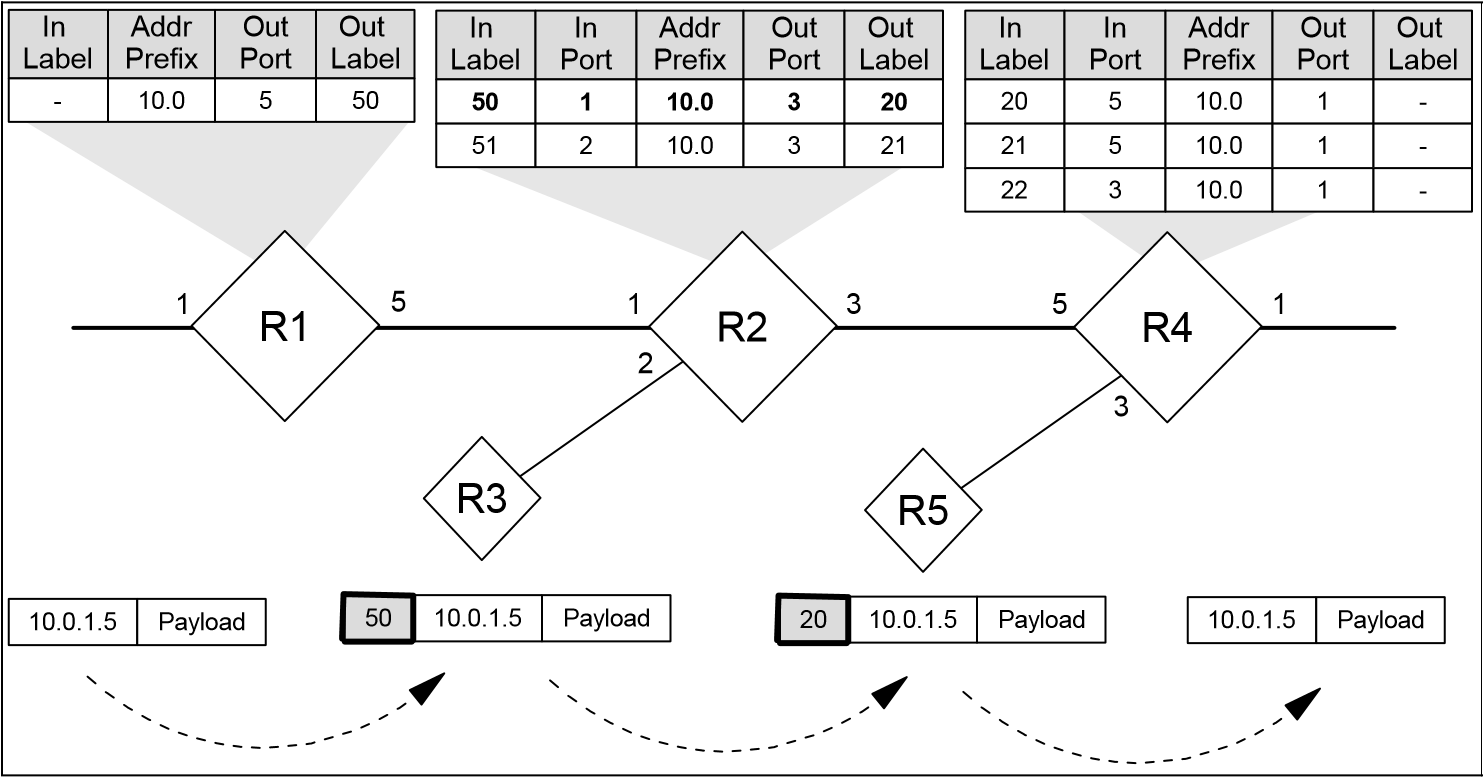
##### Forwarding a labeled packet

An MPLS node examines the label at the top of the stack of an incoming packet. It uses the ILM to map the label to an NHLFE. The NHLFE indicates where to forward the packet and the operation to perform on the label stack. Using this information, the node encodes a new label stack and forwards the resulting packet.

##### Forwarding an unlabeled packet

An MPLS node examines the network layer header and any other pertinent information required to determine an FEC. The node uses the FTN to map the FEC to an NHLFE. Processing is now identical to a labeled packet. The NHLFE indicates where to forward the packet and the operation to perform on the label stack. Using this information, the node encodes a new label stack and forwards the resulting packet.

Figure A-2 depicts label swapping in an MPLS environment.



*Figure A-2 Label swapping in an MPLS environment*

**Note:** In a label swapping environment, the next hop router is always determined from MPLS information. This might cause the packet to traverse a different path than the one obtained using conventional routing algorithms.

##### Penultimate hop popping

This is the ability to pop an MPLS label at the penultimate node rather than at the egress node. From an architectural perspective, this type of processing is permitted. The purpose of a label is to forward a packet through the network to the egress node. After the penultimate node has decided to send the packet to the egress node, the label no longer has any function. It does not need to be included in the packet.

The penultimate node pops the stack and forwards the packet based on the next hop address obtained from the NHLFE. When the egress node receives the packet, one of two activities occur:

The packet contains a label. This occurs when the penultimate node processed a packet with at least two labels. In this scenario, the label now at the top of the stack is the label the egress node needs to process to make a forwarding decision.

The packet does not contain a label. In this scenario, the LSP egress receives a standard network layer packet. The node uses the local IP routing table to make a forwarding decision.

#### A.2.2 Label switched path (LSP)

An LSP represents a set of MPLS nodes traversed by packets belonging to a specific FEC. The set is an ordered, unidirectional list. Traffic flows from the node at the head-end of the list toward the node at the tail-end of the list. The LSP for the traffic flow shown in Figure A-2 on page 933 is <R1, R2, R4>.

In an MPLS network, LSPs can be established in one of two ways:

Independent LSP control: Each LSR makes an independent decision to bind a label to an FEC. It then distributes the label to its peer nodes. This is similar to conventional IP routing; each node makes an independent decision as to how to forward a packet.

Ordered LSP control: An LSR binds a label to a particular FEC only if it is the egress LSR for that FEC, or if it has already received a label binding for that FEC from its next hop for that FEC. In an environment implementing traffic engineering policies, ordered LSP control is used to ensure that traffic in a particular FEC follows a specific path.

Section A.2.5, “Label distribution protocols” on page 938 describes the procedures used to exchange label information in an MPLS environment.

#### A.2.3 Label stack and label hierarchies

A label stack (FILO) is used in tunneling between the two MPLS nodes. The tunnel is implemented as an LSP and label switching is used to forward traffic through the tunnel.

The set of traffic sent through the tunnel constitutes an FEC. Each LSR in the tunnel must assign a label to that FEC.

To send a packet through the tunnel, the tunnel ingress node pushes a label understood by the tunnel egress node onto the label stack. The tunnel ingress node then pushes a label understood by the next hop node and forwards the data packet through the tunnel.

For example, a network might contain an LSP <R1, R2, R3, R4>. In this example, R2 and R3 are not directly connected, but are peers endpoints of an LSP tunnel. The actual sequence of LSRs traversed through the network is <R1, R2, R21, R22, R3, R4>. Figure A-3 shows this configuration.

*Figure A-3 LSP tunnels*

R1

R2

R21

R22

R3

R4

Level 1

Level 2

l

LSP Tunne

**Lb**

La

IP hdr

Payload

**Lc**

La

IP hdr

Payload

La

IP hdr

Payload

A packet traversing this network travels along a level-1 LSP: <R1, R2, R3, R4>. Then, when traveling from R2 to R3, uses a level-2 LSP: <R2, R21, R22, R3>. From the perspective of the level-1 LSP, R2’s peer devices are R1 and R3. From the level-2 perspective, R2’s peer device is R21.

Using this diagram, the following actions occur when a packet is sent through the LSP tunnel:

1. R2 receives a labeled packet from R1. The packet contains a single label. The depth of the label stack is one.
2. R2 pops this label and pushes a label understood by R3. This label is called La.
3. R2 must also include a label understood by R21. R2 pushes the label on top of the existing level-1 label. This label is called Lb. The label stack contains two entries.
4. R2 forwards the packet to R21.
5. R21 pops the level-2 label (Lb) appended by R2 and pushes a level-2 label understood by R22. This label is called Lc. R21 does not process the level-1 label. The label stack contains two entries.

1. R21 forwards the packet to R22.
2. R22 reviews the level-2 label appended by R21 and realizes it is the penultimate hop in the R2-R3 tunnel. R22 pops the level-2 label (Lc) and forwards the packet to R3. The label stack contains one entry.

#### A.2.4 MPLS stacks in a BGP environment

The network shown in Figure A-4 shows three autonomous systems. The environment contains two classes of IP routing:

Each autonomous system runs an IGP to maintain connectivity within the AS. For example, R2, R21, R22, and R3 use OSPF to maintain routes within AS 2.

Each autonomous systems runs BGP to maintain connectivity between autonomous systems. For example, border routers R1, R2, R3, and R4 use BGP to exchange inter-AS routing information.

**L3**

L1

IP hdr

Payload

AS 1

AS 2

AS 3

**L2**

L1

IP hdr

Payload

L1

IP hdr

Payload

EBGP

EBGP

IBGP

R1

IGP

R4

IGP

R21

R2

R3

R22

*Figure A-4 Connecting autonomous systems in an MPLS environment*

In this sample network, it is desirable to avoid distributing BGP-learned routes to devices which are not BGP border routers (for example, R21, R22). This minimizes the CPU processing required to maintain the IP routing table on these devices. It also eliminates the need to run a BGP routing algorithm on these devices.

An MPLS LSP stack can be used to implement this environment. In this configuration, BGP routes are distributed only to BGP peers, and not to interior

routers that lie along the hop-by-hop path between peers. LSP tunnels are configured so that:

Each peer distributes a label for each address prefix that it distributes through BGP. These labels are distributed to peers within the same AS.

The IGP maintains a host route for each BGP border router. Each interior router distributes a label for the host route to each IGP neighbor.

Consider a situation where R2 receives an unlabeled packet destined for a network connected through AS 3. The packet might have originated from a LAN segment locally connected to R2 or another LAN segment within AS 2. The packet would have been previously labeled if it had originated in AS1.

1. R2 searches the local IP forwarding table to determine the most specific route for the required destination address. The route will have be learned through BGP. The BGP next hop will be R3.
2. R3 has previously bound a label for the longest match and distributed this label to R2. This label is called L1.
3. Because all devices within AS 2 participate in the IGP, a route to R3 appears in the routing table of all devices within AS 2:
   * R22 has previously bound a label for R3 and distributed this label to R21. This label is called L2.
   * R21 has previously bound a label for R3 and distributed this label to R2. This label is called L3.
4. R2 prepares the data packet destined for AS 3 by creating a label stack. The initial entry on the stack is created by pushing the L1 label. The top entry on the stack is created by pushing the L3 label. The labeled packet is then sent to the next hop, R21.
5. R21 receives the labeled packet and reviews the top entry in the stack. Using the information in the NHLFE, R21 replaces the L3 label in the stack with the L2 label. The labeled packet is then sent to the next hop, R22.
6. R22 receives the labeled packet and reviews the top entry. Because R22 is the penultimate hop on the R2-R3 tunnel, R22 pops the L2 label on the stack and forwards the data packet to R3. The label stack now contains a single entry as it is forwarded to R3.
7. R3 receives the labeled data packet and reviews the L1 label on the stack. Using information in the NHLFE, R3 replaces the old label with a label bound by R4 and forwards the packet.

**Note:** Whenever an MPLS node pushes a label on to an already labeled packet, the new label must correspond to an FEC whose LSP egress is the node that assigned the new label.

#### A.2.5 Label distribution protocols

A label distribution protocol is a set of procedures that allows one MPLS node to distribute labels to other peer nodes. This specification is used by an LSR to notify another LSR of an assigned label and its associated meaning. This exchange establishes a common agreement between peers.

Each MPLS node participates in a local IGP to determine the network topology and populate the routing table. Label distribution protocols use this information to establish labels. After a distribution protocol has run in each node, the entire MPLS network has a complete set of paths and associated labels.

Label distribution protocols also encompass any negotiations between peers needed to learn the MPLS capabilities of each peer.

##### Types of label distribution protocols

The MPLS architecture does not specify a required distribution protocol nor does it assume there is only a single protocol. Because of this, there are a number of different standards under development. These standards can be placed into one of two categories.

###### Extensions to existing protocols

Proposals have been made to existing protocols so that label distribution information is included within existing data flows. Two examples of this are:

BGP extensions: In many cases, FECs are used to identify address prefixes distributed by BGP peers. It might be advantageous to have these same devices distribute MPLS labels. Further, the use of BGP route reflectors to distribute labels can provide significant scalability enhancements.

RSVP extensions: This proposal enhances the RSVP standard to include support for establishing and distributing LSP information. This enables the allocation of resources along the end-to-end path.

###### Development of new protocols

New protocols are also being developed with the sole purpose of distributing labels. These stand-alone protocols do not rely on the presence of specific routing protocols at every hop along the path. This is useful in situations in which an LSP must traverse nodes that do not support one of the existing protocols that has been extended to include label distribution functions.

##### Label distribution methods

There are two methods to initiate communication between MPLS nodes to exchange label information:

Downstream-on-demand: An LSR can request a label binding for a particular

FEC. The request is made to the next hop MPLS node for that FEC.

Unsolicited downstream: An LSR can distribute bindings to LSRs that have not explicitly requested the information.

Both of these distribution techniques can be used in the same network at the same time. For a given set of peers, the upstream LSR and the downstream LSR must agree on the technique to be used.

#### A.2.6 Stream merge

Stream merge is the aggregation of a large number of data flows into a single downstream flow. The device performing the merge consolidates the individual streams so that they are treated as a single stream by subsequent MPLS nodes. The merged stream is represented by a single label. After the merged packets are transmitted, any information that the packets arrived with different incoming labels is lost.

Stream merge is a major component of MPLS scalability.

### A.3 Emulating Ethernet over MPLS networks

As MPLS gaining popularity, there is a trend to carry non-IP traffic over MPLS network directly. One example is the encapsulation method for transport of Ethernet over MPLS networks.

Instead of using Ethernet cable or switch, we use MPLS as the underlying transport. The Ethernet service operates on top of the MPLS network layer as illustrated in Figure A-5 on page 940.

Emulated

Ethernet

MUX

MPLS

Emulated

Ethernet

MUX

MPLS

Physical

Packet

Tunnne

l

Etherne

t

service

*Figure A-5 Encapsulation of Ethernet over MPLS*

When emulating Ethernet over MPLS networks:

The Label Distribution Protocol (LDP) sets up a pseudo wire (as opposed to a real Ethernet cable wire, or a Ethernet switch) on MPLS.

The Ethernet pseudo wire carries the Ethernet/802.3 protocol data units (PDUs) over MPLS.

The pseudo wire emulation consists of the destination address, source address, length/type, MAC client data, and padding extracted from a MAC frame as a concatenated octet sequence in their original order.

Usually, only point-to-point Ethernet connections are emulated as the broadcast operation in a real Ethernet.

The emulation can provide multiplexing and demultiplexing at the consumer edge routers so that aggregrated traffic is between provider edge MPLS routers.

An Ethernet PW operates in one of two modes, raw mode or tagged mode:

In tagged mode, each frame *must* contain at least one 802.1Q Virtual LAN tag, a

1. Some of these protocols can be described as impractical at best. For instance, RFC 1149 (dated 1990 April 1) describes the transmission of IP datagrams by carrier pigeon and RFC 1437 (dated 1993 April 1) describes the transmission of people by electronic mail. [↑](#footnote-ref-1)
2. RFC 1771 uses uppercase to name BGP messages. The same convention is used in this section. [↑](#footnote-ref-2)
3. At the time of writing, there were 13 root servers. [↑](#footnote-ref-3)
4. Note that all of the fields are byte-aligned. The alignment of the Type field on a 4-byte boundary is for example purposes and is not required by the format. [↑](#footnote-ref-4)
5. A MIB view is a subset of the set of all instances of all object types defined according to SMI. [↑](#footnote-ref-5)
6. snmpEventID.i is an SNMPv2 manager-to-manager MIB object that shows the authoritative identification of an event. [↑](#footnote-ref-6)
7. ASN.1 BER specifies the Basic Encoding Rules for OSI Abstract Syntax Notation One, according to ISO 8825. [↑](#footnote-ref-7)