Arquitectura TCP/IP

Transmission Control Protocol / Internet Protocol

Capítulo
1

Protocolo Internet (IP)

Introducción

La arquitectura TCP/IP esta hoy en día ampliamente difundida, a pesar de ser una arquitectura de facto, en lugar de ser uno de los estándares definidos por la ISO, IICC, etc...
Esta arquitectura se empezó a desarrollar como base de la ARPANET (red de comunicaciones militar del gobierno de los EE.UU), y con la expansión de la INTERNET se ha convertido en una de las arquitecturas de redes más difundida.

Antes de continuar, pasemos a ver la relación de esta arquitectura con respecto al modelo de referencia OSI (Open Systems Interconnection) de la ISO.
Así como el modelo de referencia OSI posee siete niveles (o capas), la arquitectura TCP/IP viene definida por 4 niveles : el **nivel de subred** [enlace y físico], el **nivel de interred** [Red, IP], **el protocolo proveedor de servicio** [Transporte, TCP o UDP] , y el **nivel de aplicación**.

El Protocolo Internet (*Internet Protocol - IP*)

El protocolo IP es el principal del modelo OSI, así como parte integral del TCP/IP. Las tareas principales del IP son el direccionamiento de los datagramas de información y la administración del proceso de fragmentación de dichos datagramas.
El datagrama es la unidad de transferencia que el IP utiliza, algunas veces identificada en forma más específica como datagrama Internet o datagrama IP
Las características de este protocolo son :

- NO ORIENTADO A CONEXIÓN
- Transmisión en unidades denominadas **datagramas**.
- Sin corrección de errores, ni control de congestión.
- No garantiza la entrega en secuencia.

La entrega del datagrama en IP no está garantizada porque ésta se puede retrasar, enrutar de manera incorrecta o mutilar al dividir y reensamblar los fragmentos del mensaje. Por otra parte, el IP no contiene suma de verificación para el contenido de datos del datagrama, solamente para la información del encabezado.

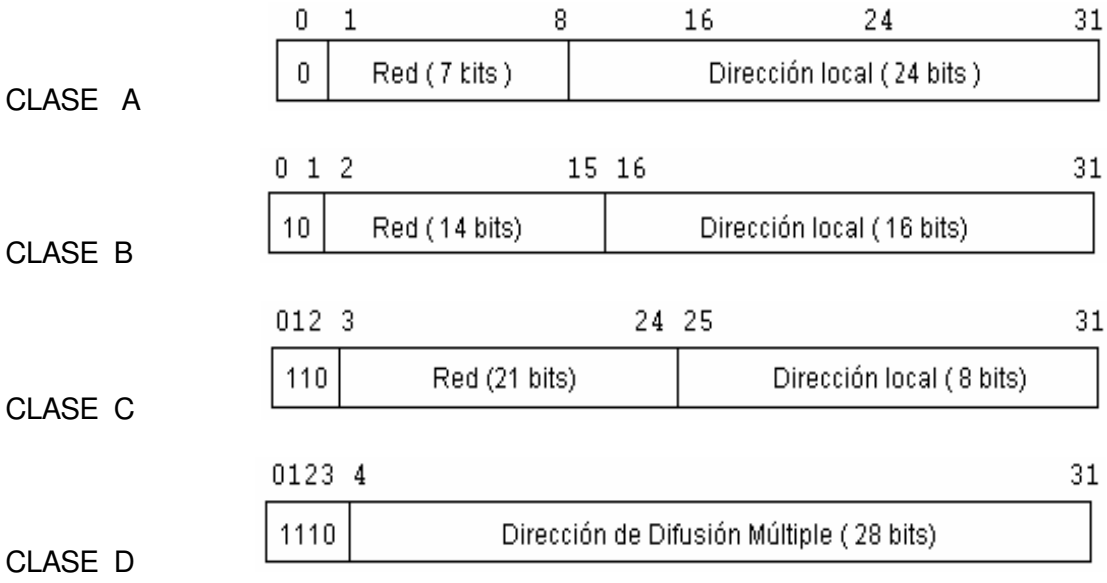
En cuanto al ruteo (encaminamiento) este puede ser :

- Paso a paso a todos los nodos
- Mediante tablas de rutas estáticas o dinámicas

Direccionamiento IP

El TCP/IP utiliza una dirección de 32 bits para identificar una máquina y la red a la cual está conectada. Unicamente el NIC (Centro de Información de Red) asigna las direcciones IP (o Internet), aunque si una red no está conectada a Internet, dicha red puede determinar su propio sistema de numeración.

Hay cuatro formatos para la dirección IP, cada uno de los cuales se utiliza dependiendo del tamaño de la red. Los cuatro formatos, Clase A hasta Clase D (aunque últimamente se ha añadido la Clase E para un futuro) aparecen en la figura :



Conceptualmente, cada dirección está compuesta por un par (RED (netid), y Dir. Local (hostid)) en donde se identifica la red y el host dentro de la red.

La clase se identifica mediante las primeras secuencias de bits, a partir de los 3 primeros bits (de orden más alto).

Las direcciones de Clase A corresponden a redes grandes con muchas máquinas. Las direcciones en decimal son 0.1.0.0 hasta la 126.0.0.0 (lo que permite hasta 1.6 millones de hosts).

Las direcciones de Clase B sirven para redes de tamaño intermedio, y el rango de direcciones varía desde el 128.0.0.0 hasta el 191.255.0.0. Esto permite tener 16320 redes con 65024 host en cada una.

Las direcciones de Clase C tienen sólo 8 bits para la dirección local o de anfitrión (host) y 21 bits para red. Las direcciones de esta clase están comprendidas entre 192.0.1.0 y 223.255.255.0, lo que permite cerca de 2 millones de redes con 254 hosts cada una.

Por último, las direcciones de Clase D se usan con fines de multidifusión, cuando se quiere una difusión general a más de un dispositivo. El rango es desde 224.0.0.0 hasta 239.255.235.255.

Cabe decir que, las direcciones de clase E (aunque su utilización será futura) comprenden el rango desde 240.0.0.0 hasta el 247.255.255.255.

Por tanto, las direcciones IP son cuatro conjuntos de 8 bits, con un total de 32 bits. Por comodidad estos bits se representan como si estuviesen separados por un punto, por lo que el formato de dirección IP puede ser red.local.local.local para Clase A hasta red.red.red.local para clase C.

A partir de una dirección IP, una red puede determinar si los datos se enviarán a través de una compuerta (GTW, ROUTER). Obviamente, si la dirección de la red es la misma que la dirección actual (enrutamiento a un dispositivo de red local, llamado *host directo*), se evitará la compuerta ; pero todas las demás direcciones de red se enrutarán a una compuerta para que salgan de la red local. La compuerta que reciba los datos que se transmitirán a otra red, tendrá entonces que determinar el enrutamiento can base en la dirección IP de los datos y una tabla interna que contiene la información de enrutamiento.

Otra de las ventajas que ofrece el direccionamiento IP es el uso de **direcciones de difusión** (*broadcast addresses*), que hacen referencia a todos los host de la misma red. Según el estándar, cualquier dirección local (hostid) compuesta toda por 1s está reservada para difusión (broadcast). Por ejemplo, una dirección que contenga 32 1s se considera un mensaje difundido a todas las redes y a todos los dispositivos. Es posible difundir en todas las máquinas de una red alterando a 1s toda la dirección local o de anfitrión (hostid), de manera que la dirección 147.10.255.255 para una red de Clase B se recibiría en todos los dispositivos de dicha red ; pero los datos no saldrían de dicha red.

Ejemplos prácticos :

EJEMPLO I

Consideremos la siguiente dirección IP en binario:

```
11001100.00001000.00000000.10101010 (204.8.0.170)
```

La dirección de la máscara (MASK) es en binario :

```
11111111.11111111.11100000.00000000 (255.255.224.0)
```

Según lo visto anteriormente, para hallar la dirección se SubRED (SubNet) tomamos la IP y considerando que todo lo que tenga 1s en la máscara se queda como esta en la IP, y todo lo que tenga 0s en la mascara se pone a 0 en la IP. Entonces, la dirección de SubRed es :

```
11001100.00001000.00000000.00000000 (204.8.0.0)
```

EJEMPLO II

Sea la dirección IP en binario :

```
00001001.01000011.00100110.00000000 (9.67.38.0)
```

Cuya máscara de red es :

```
11111111.11111111.11111111.11000000 (255.255.255.192)
```

Siguiendo el criterio anterior, tenemos que la dirección de SubNet es :

```
00001001.01000011.00100110.00000000 (9.67.38.0)
```

En la dirección de la máscara de red, los último 6 bits han quedado a 0. Estos bits son los que definen las máquinas de la SubRed (2^6=64). De estas 64 máquinas quitamos la última de ellas (será para el Broadcast). Por tanto tendremos :

```
9.67.38.0      SubNet Address
├─9.67.38.1   (1ª máquina de la SubRed)
├─9.67.38.2      (2ª máquina de la SubRed)
 .........
├─9.67.38.62  (última máquina de la SubRed)
└─9.67.38.63   BROADCAST
```

EJEMPLO III

Sea la dir.IP la `201.222.5.121`, la dirección de máscara `255.255.255.248`,

entonces , haciendo los correspondientes cálculos en binario tenemos que :

```
201.222.5.121        (IP address)

255.255.255.248    (NET MASK)

201.222.5.120      (SubNet addr.)
```

En la dirección de máscara, el 248 es 0111000, por tanto los últimos 3 bits a 0 son destinados para las máquinas de red (2^3=8), por tanto habrá 6 máquinas :

```
201.222.5.120          SubNet address
   ├── 201.222.5.121   1ª máquina de la SubNet
   ├── 201.222.5.122   2ª máquina de la SubNet
   ..............
   ├── 201.222.5.126   última máquina de la SubNet
   └── 201.222.5.127   BROADCAST
```

EJEMPLO IV

`15.16.193.6 (IP addr.)`

`255.255.248.0 (Net MASK), el SubNet addr. Será :`

`15.16.192.0` y como en la máscara de red 248.0 es 11111000.00000000

tendremos por tanto 2^11=2048, lo que implica que tenemos 2046 máquinas en la SubRed :

```
15.16.192.0      SubNet address
   ├── 15.16.192.1 1ª máquina de la SubRed
   ├── 15.16.192.2 2ª máquina de la SubRed
   ............
   ├── 15.16.200.254     última máquina de la SubRed
   └── 15.16.200.255     BROADCAST
```

DIRECCIONES DE RED Y DE DIFUSIÓN

La mayor ventaja de la codificación de información de red en las direcciones de red en IP tiene una ventaja importante: hacer posible que exista un ruteo eficiente. Otra ventaja es que las direcciones de red IP se pueden referir tanto a redes como a anfitriones (hosts). Por regla, nunca se asigna un campo hostID igual a 0 a un anfitrión individual. En vez de eso, una dirección IP con campo hostID a 0 se utiliza para referirse a la red en sí misma. En resumen:

Las direcciones IP se pueden utilizar para referirse a redes así como a anfitriones individuales. Por regla, una dirección que tiene todos los bits del campo hostID a 0, se reserva para referirse a la red en sí misma.

Otra ventaja significativa del esquema de direccionamiento IP es que éste incluye una dirección de difusión (BROADCAST) que se refiere a todos los anfitriones de la red. De acuerdo con el estándar, cualquier campo hostID consistente solamente en 1s, esta reservado para la difusión (BROADCAST). Esto permite que un sistema remoto envíe un sólo paquete que será publidifundido en la red especificada.

RESUMEN DE REGLAS ESPECIALES DE DIRECCIONAMIENTO :

En la práctica, el IP utiliza sólo unas cuantas combinaciones de ceros ("está") o unos ("toda"). Las posibilidades son las siguientes :

TODOS 0 - Éste anfitrión (permitido solamente en el arranque del sistema, pero nunca es una dirección válida de destino.
TODOS 0 | ANFITRIÓN - Anfitrión en ésta RED (solo para arranque, no como dir. válida)
TODOS 1 - Difusión limitada (red local) (Nunca es una dirección válida de origen)
RED | TODOS 1 - Difusión dirigida para RED (" ")
127 | NADA (a menudo 1) - LOOPBACK (nunca de be aparecer en una red

Como se menciona arriba, la utilización de todos los ceros para la red sólo está permitida durante el procedimiento de iniciación de la maquina. Permite que una máquina se comunique temporalmente. Una vez que la máquina "aprende" su red y dir. IP correctas, no debe utilizar la red 0.

PROTOCOLOS DE RUTEO (nivel IP).

A dos routers dentro de un sistema autónomo se les denomina "interiores" con respecto a otro.
¿Cómo pueden los routers en un sistema autónomo aprender acerca de redes dentro del sistema y redes externas?
En redes como InterNet que tienen varias rutas físicas, los administradores por lo general seleccionan una de ellas como ruta primaria. Los ruteadores interiores normalmente se

comunican con otros, intercambian información de accesibilidad a red o información de ruteo de red, a partir de la cual la accesibilidad se puede deducir.
A diferencia de esto, en la comunicación de un router exterior no se ha desarrollado un solo protocolo que se utilice con los sistemas autónomos.

Protocolo de Información de Ruteo (RIP).

Uno de los I.G.P. (Interior Gateway Protocol) más ampliamente utilizados es el RIP, también conocido con el nombre de un programa que lo implementa (el routeD o Route Daemon).
El protocolo RIP es consecuencia directa de la implantación del ruteo de vector-distancia para redes locales. En principio, divide las máquinas participantes en activas o pasivas (silenciosas). Los routers activos anuncian sus rutas a los otros; las máquinas pasivas listan y actualizan sus rutas con base a estos anuncios. Sólo un router puede correr RIP en modo activo de modo que un anfitrión deberá correr el RIP en modo pasivo.
Un router con RIP en activo difunde un mensaje cada 30 segundos, éste mensaje contiene información tomada de la base de datos de ruteo actualizada. Cada mensaje consiste en pares, donde cada par contiene una dirección IP y un entero que representa la distancia hacia esta red (el IP address).
El RIP por tanto hace uso de un vector de distancias, con una métrica por número de saltos donde se considera que 16 saltos o más es infinito. De esta manera, el número de saltos (hops number) o el contador de saltos (hop count) a lo largo de una trayectoria desde una fuente dada hacia un destino dado hace referencia al número de routers que un datagrama encontrará a lo largo de su trayectoria. Por tanto lo que se hace es utilizar el conteo de saltos para calcular la trayectoria óptima (aunque esto no siempre produce resultados buenos).
Para prevenir que dos routers oscilen entre dos o más trayectorias de costos iguales, RIP especifica que se deben conservar las rutas existentes hasta que aparezca una ruta nueva con un costo estrictamente menor.
Si falla el primer router que anuncia la ruta RIP especifica que todas las escuchas deben asociar un tiempo límite a las rutas que aprenden por medio de RIP. Cuando un router instala una ruta en su tabla, inicia un temporizador para tal ruta. Este tiempo debe iniciarse cada vez que el router recibe otro mensaje RIP anunciando la ruta. La ruta queda invalidada si transcurren 180 segundos sin que el router haya recibido un anuncio nuevamente.
RIP debe manejar tres tipos de errores ocasionados por los algoritmos subyacentes. En primer lugar, dado que el algoritmo no especifica detección de ciclos de ruteo, RIP debe asumir que los participantes son confiables o deberá tomar precauciones para prevenir los ciclos. En segundo lugar, para prevenir inestabilidades, RIP debe utilizar un valor bajo para la distancia máxima posible (RIP utiliza 16 saltos como medida máxima). Esto implica que para una red como Internet, los administradores deben dividirla en secciones o utilizar un protocolo alternativo. En tercer y último lugar, el algoritmo vector-distancia empleado por RIP crea un problema de convergencia lenta o conteo al infinito, problema en el cual aparecerán inconsistencias, debido a que los mensajes de actualización de ruteo se difunden lentamente a través de la red. Seleccionando un infinito pequeño (16) se ayuda a limitar la convergencia lenta, pero NO se elimina.

La inconsistencia en la tabla de ruteo no es exclusiva de RIP, éste es un problema fundamental que se presenta en todo protocolo con algoritmos vector-distancia, en el que los mensajes de actualización transportan únicamente pares de redes de destino y distancias hacia estas redes.

Solución al problema de la convergencia lenta:

Es posible resolver el problema de la convergencia lente mediante una técnica conocida como actualización de horizonte separado (split horizon update). Cuando se utilizan horizontes separados, un router registra la interfaz por la que ha recibido una ruta particular y no difunde la información acerca de la ruta de regreso sobre la misma interfaz. Con esto evitamos que la información "negativa" no sea difundida con rapidez.
Una de las técnicas finales para resolver el problema de la convergencia lenta se conoce como Poison Reverse. Una vez que una conexión desaparece, el router anuncia la conexión conservando la entrada de información por varios periodos de actualización e incluye un costo infinito en la difusión. Para hacer el Poison Reverse más efectivo, se debe combinar con las Triggered Updates (actualizaciones activadas) que obligan al router a que envíe una difusión inmediatamente al recibir "malas noticias", en lugar de esperar el próximo periodo de difusión. Al enviar una actualización inmediatamente, un router minimiza el tiempo en que es vulnerable por recibir "buenas noticias".

Protocolo SPF abierto (OSPF).

El algoritmo de propagación de rutas abierto (OSPF) propone los siguientes objetivos:

- Tecnología de estado de enlaces
- Soporta tipos de servicio (los administradores pueden instalar múltiples rutas hacia un destino dad, uno por cada tipo de servicio).
- Proporciona un balance de cargas entre rutas de igual peso (Si un administrador especifica múltiples rutas hacia un destino con el mismo costo, el OSPF distribuye el tráfico entre todas las rutas de la misma manera. Nótese que el RIP calcula una sola ruta para cada destino).
- Partición en áreas.
- Propagación de modificaciones entre los enlaces.
- Localización automática de routers vecinos.
- Propagación de rutas aprendidas de fuentes externas.
- Routers designados en redes multiacceso.

PROTOCOLOS DE RESOLUCION DE DIRECCIONES.

El objetivo es diseñar un software de bajo nivel que oculte las direcciones físicas (MAC) y permita que programas de un nivel más alto trabajen sólo con direcciones IP. La transformación de direcciones se tiene que realizar en cada fase a lo largo del camino, desde la fuente original hasta el destino final. En particular, surgen dos casos. Primero, en la última fase de entrega de un paquete, éste se debe enviar a través de una red física hacia su destino final. La computadora que envía el paquete tiene que transformar la dirección IP de destino final en su dirección física (MAC). Segundo, en cualquier punto del camino, de la fuente al destino, que no sea la fase final, el paquete se debe enviar hacia un router intermedio. Por lo tanto, el transmisor tiene que transformar la dirección IP del router en una dirección física.

El problema de transformar direcciones de alto nivel en direcciones físicas se conoce como *problema de asociación de direcciones* (Address Resolution Problem). Este problema se suele resolver, normalmente, mediante tablas en cada máquina que contienen pares de direcciones, de alto nivel y físicas.
En el problema de asociación de direcciones en TCP/IP para redes con capacidad de difusión como Ethernet, se utiliza un protocolo de bajo nivel para asignar direcciones en forma dinámica y evitar así la utilización de una tabla de conversiones. Este protocolo es conocido como **Protocolo de Asociación de Direcciones** (**ARP** - *Address Resolution Protocol*). La idea detrás de la asociación dinámica con ARP es muy sencilla: cuando un host A quiere definir la dirección IP (IPb), transmite por difusión (broadcast) un paquete especial que pide al anfitrión (host) que posee la dirección IP (IPb), que responda con su dirección física (Pb). Todos los anfitriones reciben la solicitud, incluyendo a B, pero sólo B reconoce su propia dirección IP y envía una respuesta que contiene su dirección física. Cuando A recibe la respuesta, utiliza la dirección física para enviar el paquete IP directamente a B. En resumen:

El ARP permite que un anfitrión encuentre la dirección física de otro anfitrión dentro de la misma red física con sólo proporcionar la dirección IP de su objetivo.

La información se guarda luego en una tabla ARP de orígenes y destinos.

***Protocolo de Asociación de Direcciones por Réplica** (**RARP**):*
Una máquina sin disco utiliza un protocolo TCP/IP para internet llamado RARP (Protocolo Inverso de Asociación de Direcciones) o Reverse Address Resolution Protocol, a fin de obtener su dirección IP desde un servidor.
En el arranque del sistema, una máquina de estas características (sin HDD permanente) debe contactar con un servidor para encontrar su dirección IP antes de que se pueda comunicar por medio del TCP/IP. El protocolo RARP utiliza el direccionamiento físico de red para obtener la dirección IP de la máquina. El mecanismo RARP proporciona la dirección hardware física de la máquina de destino para identificar de manera única el procesador y transmite por difusión la solicitud RARP. Los servidores en la red reciben el mensaje, buscan la transformación en una tabla (de manera presumible en su almacenamiento secundario) y responden al transmisor. Una vez que la máquina obtiene su dirección IP, la guarda en memoria y no vuelve a utilizar RARP hasta que se inicia de nuevo.

MENSAJES DE ERROR Y CONTROL en IP (ICMP).

Como hemos visto anteriormente, el Protocolo Internet (IP) proporciona un servicio de entrega de datagramas, no confiable y sin conexión, al hacer que cada router direccione datagramas. Si un router no puede, por ejemplo, rutear o entregar un datagrama, o si el router detecta una condición anormal que afecta su capacidad para direccionarlo (v.q., congestionamiento de la red), necesita informar a la fuente original para que evite o corrija el problema.
Para permitir que los routers de una red reporten los errores o proporcionen información sobre circunstancias inesperadas, se agregó a la familia TCP/IP un mecanismo de mensajes de propósito especial, el *Protocolo de Mensajes de Control Internet* (**ICMP**). El ICMP permite que los routers envíen mensajes de error o de control hacia otros routers o anfitriones, proporcionando una comunicación entre el software de IP en una máquina y el mismo software en otra.

Cuando un datagrama causa un error, el ICMP sólo puede reportar la condición del error a la fuente original del datagrama; la fuente debe relacionar el error con un programa de aplicación individual o debe tomar alguna otra acción para corregir el problema.

Formato de los mensajes ICMP:

Aunque cada mensaje ICMP tiene su propio formato, todos comienzan con los mismos tres campos; un campo TYPE (TIPO) de mensaje, de 8 bits y números enteros, que identifica el mensaje; un campo CODE (CODIGO), de 8 bits, que proporciona más información sobre el tipo de mensaje, y una campo CHECKSUM (SUMA DE VERIFICACIÓN), de 16 bits. Además, los mensajes ICMP que reportan errores siempre incluyen el encabezado y los primeros 64 bits de datos del datagrama que causó el problema.

La razón de regresar más que el encabezado del datagrama únicamente es para permitir que el receptor determine de manera más precisa qué protocolo(s) y qué programa de aplicación son responsables del datagrama.

El campo TYPE de ICMP define el significado del mensaje así como su formato. Los tipos incluyen:

CAMPO TYPE	Tipo de Mensaje ICMP
0	Respuesta de ECO
3	Destino inaccesible
4	Disminución de origen (source quench - datagrama eliminado por congestión)
5	Redireccionar (cambiar una ruta)
8	Solicitud de ECO
11	Tiempo excedido para un datagrama
12	Problema de parámetros de un datagrama
13	Solicitud de TIMESTAMP
14	Respuesta de TIMESTAMP
15	Solicitud de Información (obsoleto)
16	Respuesta de Información (obsoleto)
17	Solicitud de Máscara de dirección
18	Respuesta de máscara de dirección

Una de las herramientas de depuración más utilizadas incluye los mensajes ICMP de *echo request (8)* y *echo reply (0)*. En la mayoría de los sistemas, el comando que llama el usuario para enviar solicitudes de eco ICMP se conoce como **ping**.

PROTOCOLO DE DATAGRAMA DE USUARIO (UDP).

La mayoría de los Sistemas Operativos actuales soportan multiprogramación. Puede parecer natural decir que un proceso es el destino final de un mensaje. Sin embargo, especificar que un proceso en particular en una máquina en particular es el destino final para un datagrama es un poco confuso. Primero, por que los procesos se crean y se destruyen dinámicamente, los transmisores rara vez saben lo suficiente para identificar un proceso en otra máquina. Segundo, nos gustaría poder reemplazar los procesos que reciben datagramas, sin tener que informar a todos los transmisores (v.q. reiniciar la máquina puede cambiar todos los PID de los procesos). Tercero, necesitamos identificar los destinos de las funciones que implantan sin conocer el proceso que implanta la función (v.q. permitir que un transmisor contacte un servidor de ficheros sin saber qué proceso en la máquina de destino implanta la función de FS).

En vez de pensar en un proceso como destino final, imaginaremos que cada máquina contiene un grupo de puntos abstractos de destino, llamados ***puertos de protocolo***. Cada puerto de protocolo se identifica por medio de un número entero positivo.
Para comunicarse con un puerto externo, un transmisor necesita saber tanto la dirección IP de la máquina de destino como el número de puerto de protocolo del destino dentro de la máquina.
El UDP proporciona el mecanismo primario que utilizan los programas de aplicación para enviar datagramas a otros programas de aplicación. El UDP proporciona puertos de protocolo utilizados para distinguir entre muchos programas que se ejecutan en la misma máquina. Esto es, además de los datos, cada mensaje UDP contiene tanto en número de puerto de destino como el número de puerto origen, haciendo posible que el software UDP en el destino entregue el mensaje al receptor correcto y que éste envíe una respuesta.
El UDP utiliza el Protocolo Internet subyacente para transportar un mensaje de una máquina a otra y proporciona la misma semántica de entrega de datagramas, sin conexión y no confiable que el IP. No emplea acuses de recibo para asegurarse de que llegan mensajes, no ordena los mensajes entrantes, ni proporciona retroalimentación para controlar la velocidad del flujo de información entre las máquinas. Por tanto, los mensajes UDP se pueden perder, duplicar o llegar sin orden. Además, los paquetes pueden llegar más rápido de lo que el receptor los puede procesar. En resumen:

El UDP proporciona un servicio de entrega sin conexión y no confiable, utilizando el IP para transportar mensajes entre máquinas. Emplea el IP para llevar mensajes, pero agrega la capacidad para distinguir entre varios destinos dentro de la computadora anfitrión.

Formato de los mensajes UDP:

Cada mensaje UDP se conoce como *datagrama de usuario.* Conceptualmente, un datagrama de usuario consiste en dos partes: un encabezado UDP y un área de datos UDP. El encabezado se divide en cuatro campos de 16 bits, que especifican el puerto desde el que se envió el mensaje, el puerto para el que se destina el mensaje, la longitud del mensaje y una suma de verificación UDP.

2

Protocolo de Control de Transmisión (TCP)

Servicio de Transporte de Flujo Confiable

En las secciones anteriores hemos visto el servicio de entrega de paquetes sin conexión y no confiable, que forma la base para toda comunicación en InterNet, así como el protocolo IP que lo defina.

Ahora veremos el segundo servicio más importante y mejor conocido a nivel de red, la entrega de flujo confiable (Reliable Stream Transport), así como el Protocolo de Control de Transmisión (TCP) que lo define.

En el nivel más bajo, las redes de comunicación proporcionan una entrega de paquetes no confiable. Los paquetes se pueden perder o destruir debido a errores (falla el hardware, sobrecarga de la red,...). Las redes que rutean dinámicamente los paquetes pueden entregarlos en desorden, con retraso o duplicados. En el nivel más alto, los programas de aplicación a menudo necesitan enviar grandes volúmenes de datos de una computadora a otra. Utilizar un sistema de entrega de conexión y no confiable para transferencias de grandes volúmenes de información resulta ser la peor opción. Debido a esto, el TCP se ha vuelto un protocolo de propósito general para estos casos.

La interfaz entre los programas de aplicación y la entrega confiable (es, decir, las características del TCP) se caracterizan por cinco funciones :

- **Servicio Orientado a Conexión :** El servicio de entrega de flujo en la máquina destino pasa al receptor exactamente la misma secuencia de bytes que le pasa el transmisor en la máquina origen.

- **Conexión de Circuito Virtual :** Durante la transferencia, el software de protocolo en las dos máquinas continúa comunicándose para verificar que los datos se reciban correctamente. Si la comunicación no se logra por cualquier motivo (v.q. falla el hardware de red), ambas máquinas detectarán la falla y la reportarán a los programas apropiados de aplicación. Se utiliza el término *circuito virtual* para describir dichas conexiones porque aunque los programas de aplicación visualizan la conexión como un circuito dedicado de hardware, la confiabilidad que se proporciona depende del servicio de entrega de flujo.

- **Transferencia con Memoria Intermedia :** Los programas de aplicación envían un flujo de datos a través del circuito virtual pasando repetidamente bytes de datos al software de protocolo. Cuando se transfieren datos, cada aplicación utiliza piezas del tamaño que encuentre adecuado, que pueden ser tan pequeñas como un byte. En el extremo receptor, el software de protocolo entrega bytes del flujo de datos en el mismo orden en que se enviaron, poniéndolos a disposición del programa de aplicación receptor tan pronto como se reciben y se verifican. El software de protocolo puede dividir el flujo en paquetes, independientemente de las piezas que transfiera el programa de aplicación. Para hacer eficiente la transferencia y minimizar el tráfico de red, las implantaciones por lo general recolectan datos suficientes de un flujo para llenar un datagrama razonablemente largo antes de enviarlo. Por lo tanto, inclusive si el programa de aplicación genera el flujo un byte a la vez, la transferencia a través de la red puede ser sumamente eficiente. De forma similar, si el programa de aplicación genera bloques de datos muy largos, el software de protocolo puede dividir cada bloque en partes más pequeñas para su transmisión. Para aplicaciones en las que los datos de deben entregar aunque no se llene una memoria intermedia, el servicio de flujo proporciona un mecanismo de *empuje* o *push* que las aplicaciones utilizan para forzar una transferencia. En el extremo transmisor, el push obliga al software de protocolo a transferir todos los datos generados sin tener que esperar a que se llene una memoria intermedia. Sin embargo, la función de push sólo garantiza que los datos se transferirán, por tanto, aún cuando la entrega es forzada, el software de protocolo puede dividir el flujo en formas inesperadas (v.q. el transmisor puede reducirlo en caso de congestión).

- **Flujo no estructurado :** Posibilidad de enviar información de control junto a datos.

- **Conexión Full Duplex :** Se permite la transferencia concurrente en ambas direcciones. Desde el punto de vista de un proceso de aplicación, una conexión full duplex permite la existencia de dos flujos independientes que se mueven en direcciones opuestas, sin ninguna interacción aparente. Esto ofrece una ventaja : el software subyacente de protocolo puede enviar datagramas de información de control de flujo al origen, llevando datos en la dirección opuesta. Este procedimiento de carga, transporte y descarga REDUCE EL TRAFICO en la red.

La “Contradicción”

Hemos visto que el servicio de entrega de flujo confiable garantiza la entrega de los datos enviados de una máquina a otra sin pérdida o duplicación. Surge ahora la pregunta contradictoria “del millón” : *¿ Cómo puede el software subyacente de protocolo proporcionar una transferencia confiable si el sistema subyacente de comunicación sólo ofrece una entrega NO confiable de paquetes ?.*

La respuesta es complicada, pero la mayor parte de los protocolos confiables utilizan una técnica fundamental conocida como ***acuse de recibo positivo con retransmisión***. La técnica requiere que un receptor se comunique con el origen y le envíe un mensaje de acuse de recibo (***ACK***) conforme recibe los datos (ver los primeros temas para una descripción más detallada). El transmisor guarda un registro de cada paquete que envía y espera un ACK antes de enviar el siguiente paquete. El transmisor también arranca un temporizador cuando envía un paquete y lo retransmite si dicho temporizador expira antes de que llegue un ACK.

El problema final de la confiabilidad surge cuando un sistema subyacente de entrega de paquetes los duplica. Los duplicados también pueden surgir cuando las redes tienen grandes retrasos que provocan la retransmisión prematura. Para evitar la confusión causada por ACKs retrasados o duplicados, los protocolos de acuses de recibo positivos envían los números de secuencia dentro de los ACKs, para que el receptor pueda asociar correctamente los acuses de recibo con los paquetes.

Pero, como casi todo en esta vida es un problema tras otro, el TCP no iba a ser menos ; uno de los problemas que acarrea lo anterior es que un protocolo simple de acuses de recibo positivos ocupa una cantidad sustancial de ancho de banda de red debido a que debe retrasar el envío de un nuevo paquete hasta que reciba un ACK del paquete anterior.

La solución está en otra técnica conocida como ***ventana deslizante***, que es una forma más compleja de acuse de recibo positivo y retransmisión. Los protocolos de ventana deslizante utilizan el ancho de banda de red de mejor forma al permitir que el transmisor envíe varios paquetes sin esperar el ACK (remitirse a capítulos anteriores para una descripción de éste método).

Puertos, conexiones y puntos extremos.

Al igual que el UDP, el TCP reside sobre el IP en el esquema de estratificación por capas de protocolos. El TCP permite que varios programas de aplicación en una máquina se comuniquen de manera concurrente y realiza el demultiplexado del tráfico TCP entrante entre los programas de aplicación. Así mismo, al igual que el UDP, el TCP utiliza números de ***puerto de protocolo*** para identificar el destino final dentro de una máquina. Cada puerto tiene asignado un número entero pequeño utilizado para identificarlo.

Para comprender el significado de un puerto hay que pensar de cada puerto como en una cola de salida en la que el software de protocolo coloca los datagramas entrantes, aunque en realidad los puertos TCP son más complejos, ya que un número de puerto no corresponde a un sólo objeto. El TCP utiliza la conexión, no el puerto de protocolo, como su abstracción fundamental ; las conexiones se identifican por medio de un par de puntos extremos.

¿Qué es exactamente un punto extremo en TCP ?

Un punto extremo es un par de números enteros **(host, puerto)**, en donde *host* es la dirección IP de un anfitrión y *puerto* es el un puerto TCP en dicho anfitrión.

Las conexiones vienen definidas por dos puntos extremos, y es más : la abstracción de la conexión para TCP permite que varias conexiones compartan un punto extremo (por ejemplo, varias conexiones en los mismos puertos). Esto es posible a que el TCP identifica una conexión por medio de un par de puntos extremos, y por eso varias conexiones en la misma máquina pueden compartir un número de puerto TCP.

El TCP combina la asignación dinámica y estática de puertos mediante un conjunto de *asignación de puertos bien conocidos* para programas llamados con frecuencia, pero la salida de la mayor parte de los números disponibles para el sistema se asigna conforme los programas lo necesitan.

Capítulo
3

La siguiente tabla muestra un ejemplo de números de puerto TCP asignados actualmente.

DECIMAL	CLAVE	CLAVE UNIX	DESCRIPCIÓN
0			Reservado
1	TCPMUX		Multiplexor TCP
5	RJE		Introducción de función remota
7	ECHO	echo	Eco
9	DISCARD	discard	Abandonar
11	USERS	systat	Usuarios activos
13	DAYTIME	daytime	Fecha, hora
15		netstat	Estado de red
17	QUOTE	qotd	Cita del día
19	CHARGEN	chargen	Generador de caracteres
20	FTP-DATA	ftp-data	Datos para FTP
21	FTP	ftp	File Transfer Protocol
23	TELNET	telnet	Conexión por terminal
25	SMTP	smtp	Protocolo de Transporte de Correo Sencillo
42	NAMESERVER	name	Nombre del host servidor
43	NICNAME	whois	Comando whois
53	DOMAIN	nameserver	Servidor de nombre de dominio (DNS)
79	FINGER	finger	Comando finger
93	DCP		Protocolo de Control de Dispositivo
101	HOSTNAME	hostnames	Servidor de Nombre de Anfitrión NIC
103	X400	x400	Servicio de correo X400
104	X400-SND	x400-snd	Envío de coreo X400

La *Interface* SOCKET

El Paradigma de E/S de UNIX y la E/S de la Red

En primer lugar hemos de distinguir entre los protocolos de interface y el TCP/IP, debido a que los estándares no especifican exactamente cómo es que interactúan los programas de aplicación con el software de protocolo.

A pesar de la carencia de un estándar, veremos la interface del UNIX BSD como se emplea el TCP/IP en programación. En particular, la interface ***Winsock*** proporciona la funcionalidad socket para MsWindows.

Veamos pues cómo empezó todo este "jaleo":

Unix fue desarrollado y diseñado como un sistema operativo de tiempo compartido para computadoras uniprocesador. Se trata, como ya es sabido, de un S.O. orientado a proceso, en el que cada programa de aplicación se ejecuta como un proceso de nivel de usuario. Derivados de los MULTICS, los primitivos sistemas de E/S de UNIX siguen un paradigma conocido como "*Open-Read-Write-Close*": antes de que un proceso de usuario pueda ejecutar operaciones de E/S, llama a *Open* para especificar el archivo o dispositivo que se va a utilizar (recuerdese la independencia de dispositivo de UNIX) y obtiene el permiso. La llamada a *Open* devuelve un pequeño entero (el descriptor de archivo) que el proceso utiliza al ejecutar las operaciones de E/S en el archivo abierto. Una vez abierto un objeto, se pueden hacer las llamadas a *Read* y/o *Write*. Tanto Read como Write toman tres argumentos (descriptor de archivo, dirección del buffer y nº de bytes a transferir). Una vez completadas estas operaciones el proceso llama a *Close*.

Originalmente, todas las operaciones UNIX se agrupaban como se ha descrito anteriormente, y una de las primeras implementaciones de TCP/IP también utilizó éste paradigma. Pero el grupo que añadió los protocolos TCP/IP al BSD decidió que, como los protocolos de red eran más complejos que los dispositivos convencionales de E/S, la interacción entre los programas de usuaio y los protocolos de red debías ser más compleja. En particular, la interface de protocolo debía permitir a los programadores crear un código de servidor que esperaba las conexiones pasivamente, así como también un código cliente que formara activamente las conexiones. Para manejas datagramas, se decidió abandonar este paradigma.

La abstracción de SOCKET

La base para la E/S de red en UNIX se centra en una abstracción conocida como ***socket***.

El socket es la generalización del mecanismo de acceso a archivos de UNIX que proporciona un punto final para la comunicación. Al igual que con el acceso a archivos, los programas de aplicación requieren que el S.O. cree un socket cuando se necesite. El S.O. devuelve un

Capítulo
4

entero que el programa de aplicación utiliza para hacer referencia al socket recientemente creado. La diferencia principal entre los descriptores de archivo y los descriptores de socket es que el sistema operativo enlaza un descriptor de archivo a un archivo o dispositivo del sistemacuando la aplicación llama a Open, pero puede crear sockets sin enlazarlos a direcciones de destino específicas.

Básicamente, el socket es una API en la que el servidor espera en un puerto predefinido y el cliente puede utilizar sin embargo un puerto dinámico.

EJEMPLOS:

- Creación de un socket:

```
resultado = socket (pf, tipo, protocolo)
```

  El argumento PF especifica la familia de protocolo que se va utilizar con el socket (v.q. PF_INET para TCP/IP). El argumento tipo especifica el tipo de comunicación que se desea (v.q.SOCK_DGRAM para servicio de entrega de datagramas sin conexion, o SOCK_STREAM para servicio de entrega confiable de flujo).

- Envío de datos:

```
write (socket, buffer, lenght)
```

- Especificación de una dirección local:

```
bind (socket, localaddr, addrlen)
```

  Inicialmente, un socket se crea sin ninguna asociación hacia direcciones locales o de destino. Para los protocolos TCP/IP, esto significa que ningún número de puerto de protocolo local se ha asignado y que ningún puerto de destino o dirección IP se ha especificado. En muchos casos, los programas de aplicación no se preocupan por las direcciones locales que utilizan, ni están dispuestos a permitir que el software de protocolo elija una para ellos. Sin embargo, los procesos del servidor que operan en un puerto “bien conocido” deben ser capaces de especificar dicho puerto para el sistema. Una vez que se ha creao un socket, el servidor utiliza una llamada del sistema BIND (enlace) para establecer una dirección local para ello.

  BIND tiene la forma que se ha descrito arriba.

Sistema de Nombre de Dominio (DNS)

Introducción

Los protocolos descritos anteriormente utilizan enteros de 32 bits, llamados direcciones de protocolo internet (dir. IP) para identificar máquinas. Aún cuando cada dirección proporciona una representación compacta y conveniente para identificar la fuente y el destino en paquetes enviados a traves de la red, los usuarios prefieren asignar a las máquinas nombres fáciles de recordar.

El DNS tiene dos aspectos conceptualmente independientes. El primero es abstracto. Especifica la sintaxis del nombre y las reglas para delegar la autoridad respecto a los nombres. El segundo es concreto: especifica la implantación de un sistema de computación distribuido que transforma eficientemente los nombres en direcciones.

Resolución de nombres

Conceptualmente, la resolución de nombres de dominio procede de arriba hacia abajo, comenzando con el servidor de nombres raiz y siguiendo luego hacia los servidores localizados en las ramas del arbol de la red.

Hay dos forma de utilizar en sistema de nombres de dominio: contactar un servidor de nombres cada vez o solicitar al sistema de servidores de nombres que realice la traducción completa. En este caso, el software cliente forma una solicitud de nombres de dominio que contiene el nombre a resolver, una declaración sobre la clase del nombre, el tipo de respuesta deseada y un código que especifica si el servidor de nombres debe traducir el nombre completamente. Se envía la solicitud a un servidor de nombre para su resolución.

Cuando un servidor de nombres de dominio recibe una solicitud, verifica si el nombre señala un subdominio sobre el cual tenga autoridad. Si es así, traduce el nombre a una dirección de acuerdo con su base de datos y anexa una repuesta a la solicitud, antes de enviarla de regreso al cliente. Si el DNS no puede resolver el nombre completamente, verifica que tipo de interacción especificó el cliente. Si el cliente solicita una traducción completa (una *resolución recursiva* en la terminologia DNS), el servidor se pone en contacto con un servidor de nombres de dominio que pueda resolver el problema del nombre y devuelve la respuesta al cliente.

Si el cliente solicita una resolución no recursiva (resolución iterativa), el servidor de nombres no puede dar una respuesta. Se genera una réplica que especifica el nombre del servidor que el cliente deberá contactar la próxima vez para resolver el nombre.

Como encuentra un cliente un DNS para comenzar la búsqueda ?

Como encuentra un DNS a otros DNSs que puedan responder a las solicitudes que el no puede responder ?

La respuesta es sencilla: Un cliente debe saber como contactar al ultimo DNS para asegurarse de que el DNS puede alcanzar a otros, el sistema de dominio requiere que cada servidor conozca la dirección del último servidor en la raiz. Además, un servidor podría conocer la dirección de un servidor para el dominio de un nivel inmediatamente superior (llamado padre).

Los DNSs utilizan un puerto de protocolo bien conocido para toda comunicación, así, los clientes saben cómo comunicarse con un servidor una vez que conocen la dirección IP de la máquina que se conecta al servidor. No hay forma estándar que los anfitriones localicen una máquina en el entorno local, el cuaul corre un DNS; esto se encuentra abbierto para quien diseñe el software cliente.

El algunos sistemas, la dirección de la máquina que proporciona el servicio de nombres de dominio está dentro de la frontera de los programas de aplicación en el tiempo de compilación, mientras que en otros la dirección se encuentra configurada dentro del S.O. en el arranque. En otros mas, al administrador coloca la dirección de un servidor en un archivo en almacenamiento secundario (`/etc/hosts`).