El nivel de transporte

PID_00218445

Ramon Musach Pi

Universitat Oberta de Catalunya

Índice

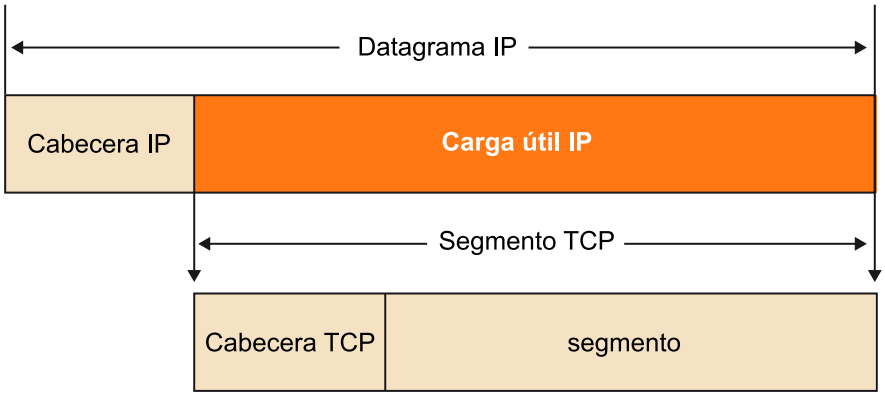
Introducción

En este módulo detallaremos las características y el funcionamiento del nivel de transporte.

La **capa de transporte** es la que se encarga de proveer la comunicación **extremo a extremo** entre procesos de aplicaciones ubicadas en diferentes equipos finales. Por lo tanto, permite que las aplicaciones no deban preocuparse de la infraestructura de red subyacente. El hecho de que sea extremo a extremo hace que este protocolo solo se deba aplicar en estos anfitriones o equipos finales, y no en los equipos intermedios o routers.

La información transmitida por las aplicaciones se convierte en paquetes de la capa de transporte, conocidos como **segmentos de la capa de transporte**. Cada segmento construido dispone de una cabecera y la información que se quiere transmitir (que seguramente se ha repartido en diferentes segmentos). Los segmentos se envían a la capa de red, donde serán incluidos en los paquetes del nivel de red, llamados **datagramas**. Estos datagramas van circulando por diferentes conmutadores (*switches*) o enrutadores (routers), en el nivel de la capa de red, hasta llegar al equipo destinatario o receptor. Será este el que en el nivel de su capa de red extraerá el segmento de transporte y lo pasará a su capa de transporte para poder procesarlo y enviarlo a la aplicación correspondiente.

Figura 1



Los segmentos TCP se encapsulan y envían dentro de datagramas IP, tal y como se muestra en esta figura.

Las funciones de la capa de transporte son las siguientes:

- Garantizar la transmisión sin errores, extremo a extremo, independientemente de los tipos de red por los que circule la información. Esto lo hace asegurando que los datos lleguen sin pérdidas, sin errores y sin duplicidades.

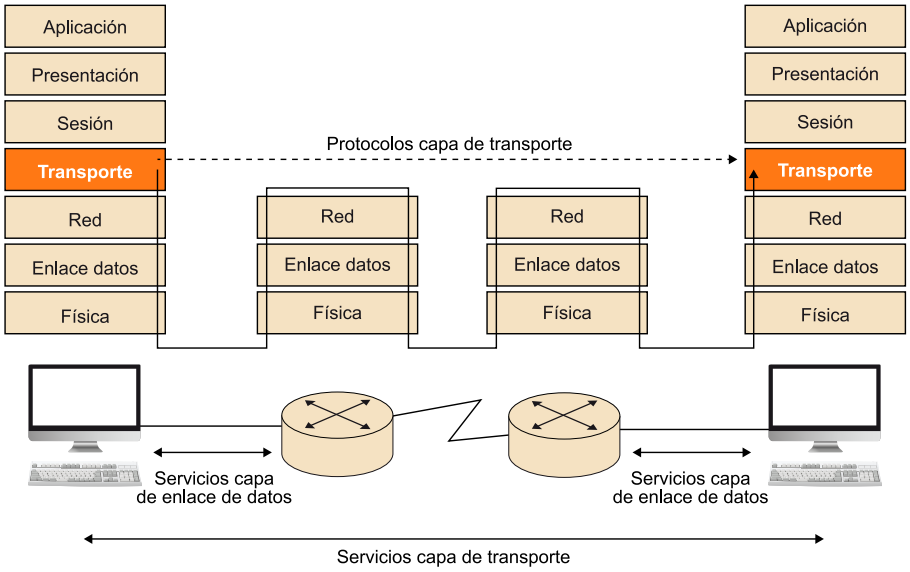
- Controlar la transmisión extremo a extremo, del flujo de datos y de la congestión de la red. Se encargará de ordenar los paquetes que vayan llegando; y de fragmentar o recomponer los mensajes, según convenga.

- Responsabilizarse de establecer, mantener y finalizar las conexiones entre dos equipos finales, o entre un equipo final y un servidor en una red.

- Permitir la multiplexación de varias conexiones de transporte sobre una misma conexión de red.

Los servicios de la capa de transporte son implementados por un protocolo de la misma capa que es utilizado para comunicar dos *hosts* o equipos finales.

Figura 2



La capa de transporte es la primera capa extremo a extremo.

Cada servicio puede emplear uno de los dos protocolos disponibles dependiendo del nivel de fiabilidad que cada aplicación necesite. Estos dos protocolos son:

- **UDP (*user datagram protocol*)**: protocolo no orientado a conexión.
- **TCP (*transmission control protocol*)**: protocolo orientado a conexión.

TCP proporciona fiabilidad completa, dado que garantiza que los paquetes lleguen correctamente y en orden, mientras que UDP solo proporciona un simple control de errores.

Los servicios de la capa de transporte se pueden dividir en cinco grandes categorías, que a continuación pasamos a describir:

1) Transmisión de mensajes de extremo a extremo. La capa de red trata cada uno de los paquetes enviados de manera independiente; en cambio, la capa de transporte asegura la transmisión correcta de todo el mensaje, formado por diferentes paquetes individuales que habrá que ordenar en el destino.

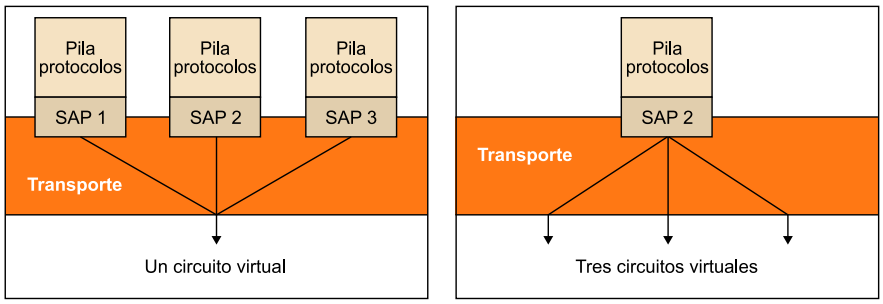
2) Direccionamiento. Teniendo en cuenta que en un mismo **segmento** puede estar dando servicio a diferentes aplicaciones, es necesario que los protocolos de la capa de transporte conozcan qué protocolos de capas superiores se están comunicando. Por lo tanto, además de conocer el direccionamiento físico y lógico, hay que conocer un tercer nivel de direccionamiento: las ubicaciones conocidas con el nombre de **puntos de acceso al servicio** (*service access points* o **SAP**).

3) Fiabilidad de las transmisiones. El protocolo debe entregar a la aplicación de destino exactamente la misma información que le ha entregado la aplicación de origen, por lo tanto, ha de estar libre de errores. Además, debe garantizar que toda la información transmitida por la aplicación de origen se entregue a la aplicación de destino, garantizando la secuencia en la transmisión, sin pérdidas y sin duplicidades.

4) Control del flujo. Corresponde a un conjunto de procedimientos que permiten indicar al emisor qué cantidad de datos (paquetes o bytes) ha de transmitir antes de ponerse a esperar un acuse de recibo o ACK, por parte del receptor. El flujo de datos se debe realizar de manera que nunca sature el *buffer* del receptor donde se van dejando los datos para su posterior tratamiento.

5) Multiplexación. La multiplexación de aplicaciones consiste en tomar los datos de las diferentes aplicaciones que se están usando en la capa de aplicación, etiquetarlas con un número de puerto (UDP o TCP) que identifica a cada aplicación del emisor y enviar este paquete con la información de todas las aplicaciones a la capa de red. Esta multiplexación puede ser de dos tipos: **multiplexación ascendente** (múltiples conexiones de la capa de transporte utilizan la misma conexión de red), cuando la capa de transporte permite que diferentes aplicaciones puedan estar compartiendo un mismo circuito virtual, y **multiplexación descendente** (una conexión de transporte utilizando múltiples conexiones de red) cuando la capa de transporte divide una única conexión en diferentes circuitos virtuales para mejorar la velocidad de transmisión.

Figura 3



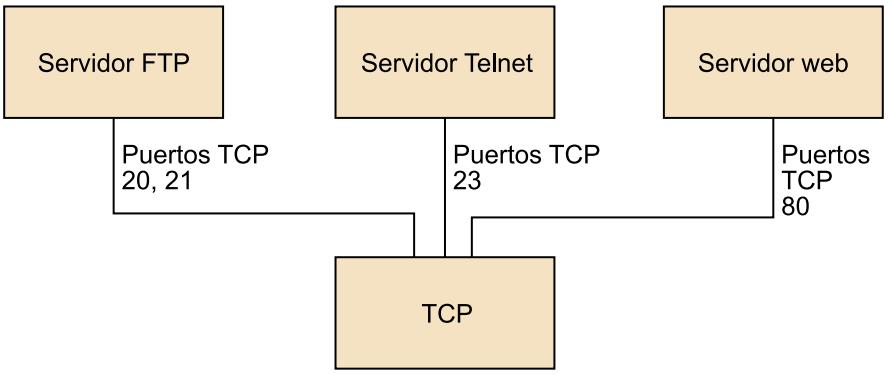
Multiplexación ascendente Multiplexación descendente

En estos esquemas podemos apreciar la diferencia entre los dos tipos de multiplexación. Recordemos que los servicios están disponibles en los llamados *service access point* (SAP), identificados con una dirección única.

1. Puertos

Un concepto muy importante en el protocolo TCP es el de los puertos, que permiten identificar de manera unívoca todos los programas que funcionan en el nivel de aplicación y que reciben los datos que provienen de TCP. Recordemos que vienen a ser como canales de entrada/salida de información, pero que no existen como tales porque la entrada física solo es una: la conexión de red. Así, por ejemplo, FTP utiliza el puerto 21, Telnet el 23, SMTP (para correo electrónico) el 25, o http (para www) el puerto 80. De este modo, si se quiere contactar con una aplicación en una máquina determinada, será suficiente escribir la dirección IP de la máquina seguida de su número de puerto.

Figura 4



Asociación de algunas aplicaciones y los puertos que utilizan.

En definitiva, un ordenador conectado a la red se identifica unívocamente con su número IP, lo que le permite comunicarse con otros ordenadores. Pero además de identificarse cada ordenador en la red, deberemos poder llegar a diferenciar los servicios que podemos utilizar. Por lo tanto, la dirección completa, en el caso de internet, será la IP acompañada del número de puerto.

En el caso de la jerarquía TCP/IP se definen dos direcciones que relacionan el nivel de transporte con los niveles superior e inferior. Para el nivel inferior tenemos la **dirección IP**, que identifica un subsistema dentro de una red, y para el nivel superior el denominado **puerto**, que identifica la aplicación que requiere la comunicación. Se denomina **zócalo** o ***socket*** de un proceso al par formado por la dirección IP del *host* en el que el proceso o aplicación se está ejecutando y el número de puerto de este proceso.

Ejemplo

Si la dirección IP de un *host* en el que se está ejecutando un servicio HTTP es 180.24.32.15, y HTTP es el servicio (con número de puerto predeterminado 80), entonces el *socket* del servicio HTTP en este *host* sería 180.24.32.15:80. Con este número el *host* identificará el servicio solicitado, en este caso HTTP.

Dado que algunas aplicaciones son ya de por sí protocolos estandarizados, como Telnet y FTP, emplean el mismo número de puerto en todas las implementaciones. Estos puertos "asignados" se conocen como puertos bien conocidos, y sus aplicaciones, como aplicaciones bien conocidas. Estos números de puerto son controlados y asignados por **IANA** (*Internet Assigned Numbers Authority*), por lo que deben ser respetados por los fabricantes. En la mayoría de los sistemas solo los pueden utilizar los procesos del sistema o los programas que ejecutan usuarios privilegiados. Estos números de puerto van del **0 al 1023**.

Los fabricantes tienen la posibilidad de requerir a **IANA** números de puerto, que serán oficialmente asignados para protocolos propios (del **1024 al 49151**) –se conocen como puertos registrados–. El resto de los puertos, del **49152 al 65535** –puertos dinámicos y/o privados– no los controla **IANA** y en la mayor parte de los sistemas los pueden usar los programas de usuario.

Rango de puertos	Denominación
0 al 1023	Puertos bien conocidos
1024 al 49151	Puertos registrados
49152 al 65535	Puertos dinámicos y/o privados

Rango de puertos y denominación.

Enlace

En el siguiente enlace se puede encontrar la lista de puertos actualizada por IANA (*Internet Assigned Numbers Authority*, www.iana.org)

http://www.iana.org/assignments/port-numbers.

Existen 65.535 puertos diferentes empleados por las conexiones de red.

2. *User datagram protocol* (UDP)

UDP (*user datagram protocol*) es un protocolo **no orientado a conexión** utilizado en la capa de transporte como alternativa al protocolo TCP (que veremos más adelante). El hecho de ser no orientado a conexión hace que en el equipo emisor no se compruebe si el *host* de destino está disponible.

Se caracteriza por un servicio rápido de transmisión, con una mínima carga de la red, pero por el contrario no ofrece fiabilidad ni control de flujo. La no fiabilidad está relacionada con el hecho de que el emisor no tiene garantías de que los datos enviados lleguen correctamente al destino. En cuanto a no realizar control de flujo, hay que decir que en los campos de sus segmentos no encontramos un número de secuencia; por lo tanto, no se controla el orden con el que llegan los segmentos al destino.

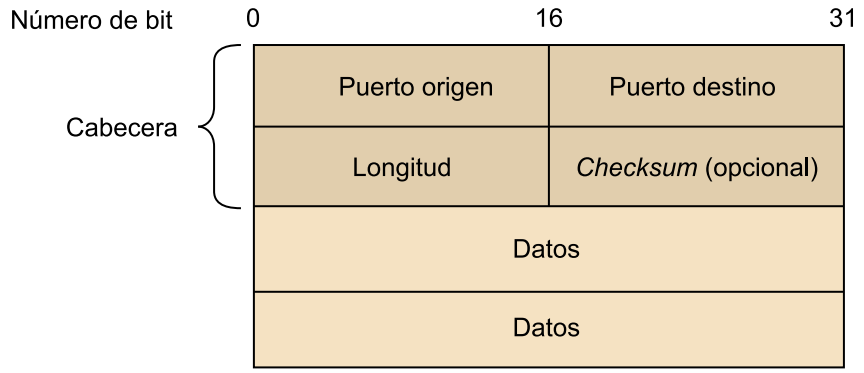
Otra característica de este protocolo es que ofrece la multiplexación de aplicaciones sobre la misma conexión de red gestionada por el protocolo IP. Así, un mismo protocolo de transporte lleva información de diferentes aplicaciones, siendo identificadas por el puerto.

Observemos que emplear UDP puede llegar a ser muy beneficioso, sobre todo cuando se quieren transmitir pequeñas cantidades de información, o cuando nuestras aplicaciones necesitan una respuesta por parte del destinatario de la información, como por ejemplo una consulta en una base de datos.

Otro caso donde UDP resulta muy eficiente, a pesar de que no proporcione una transmisión fiable, es cuando se quiere transferir una gran cantidad de datos en muy poco tiempo, como por ejemplo en las transmisiones de vídeo o audio por internet, y en tiempo real. Es un protocolo sencillo que permite velocidades elevadas de transmisión.

Los campos de un segmento UDP se representan en esta figura, con su cabecera y la parte de datos:

Figura 5



Cabecera y campos de datos del protocolo UDP.

Pasemos a describir estos campos:

- **Puerto origen y puerto destino**. Indican el número del puerto que inicia la transmisión en el *host* origen, y el número del puerto destino de la transmisión. Cada uno de estos campos tienen una longitud de 16 bits.

- **Longitud**. Indica la longitud total del segmento en *bytes*. Este campo tiene un tamaño de 16 bits.

- **Cheksum**. Este campo, opcional en UDP, solo permite al *host* destino que compruebe si está libre de errores, para descartarlo y no enviarlo a la aplicación si detecta errores. Su longitud es de 16 bits.

- **Datos**. Contiene los datos enviados por el emisor. El tamaño de este campo es variable.

Como hemos comentado, un ejemplo concreto de conexión UDP es el caso de *streaming* de vídeo o audio. En el caso de visualizar un vídeo o escuchar un fichero de audio que tenemos en la red internet, este generalmente se descarga mientras lo estamos visualizando o escuchando, por lo tanto en tiempo real. Bajar en ***streaming*** se asocia a que llega sin interrupciones. La conexión empleada utiliza UDP en lugar de TCP, dado que no es tan importante que llegue sin ningún error como que la velocidad de llegada sea suficientemente alta.

3. *Transmission control protocol* (TCP)

TCP, *transmission control protocol*, es un protocolo de nivel de transporte que se utiliza en internet cuando las aplicaciones necesitan una transmisión fiable y libre de errores. De hecho, es el protocolo más complejo e importante de la pila de protocolos de internet.

Es un estándar definido en el RFC 793, *Transmission Control Protocol* (TCP).

TCP envía un segmento a la red y hasta que su emisor no recibe otro segmento confirmando que ha llegado al destino, no lo da por enviado correctamente. Si no llega esta confirmación, después de un periodo de tiempo determinado, el emisor vuelve a retransmitir el segmento TCP. Para verificar que llega a destino libre de errores, todos los segmentos enviados contienen una suma de comprobación. Es el campo **checksum**, que corresponde a ***sum****mation* ***check***, abreviado. Hasta que no llega al destino, y en este se comprueba su corrección, no se envía el paquete de confirmación al emisor.

TCP también proporciona control del flujo garantizando que el receptor no se colapsa con más datos de los que su *buffer* puede llegar a almacenar. Entendemos por *buffer* aquella área del *host* de destino que se utiliza para almacenar temporalmente los datos que le llegan a raíz de la transmisión, hasta que llegan a ser procesados por la aplicación destino.

Como ya hemos comentado, TCP es un protocolo **orientado a conexión**, lo que implica que el emisor y el receptor de la transmisión acuerdan previamente esta transmisión, y cómo será el inicio y el final de esta. Por lo tanto, TCP ve los bytes que envía como un flujo constante y no como segmentos separados. En este sentido, todos los segmentos presentan un número de secuencia para controlar su orden.

3.1. Funcionamiento del protocolo TCP

Pasamos ahora a detallar algo más el funcionamiento del protocolo TCP. Antes de que dos *hosts* puedan empezar a intercambiarse información, deben establecer una sesión previa en la transmisión. Esta sesión se inicializa con un protocolo llamado ***three-way handshake protocol***. Con este protocolo se sincronizan los números de secuencia y se provee el control de la información necesario para establecer la conexión virtual entre los dos *hosts*.

Una vez ha finalizado el protocolo *three-way handshake*, los segmentos son enviados y confirmados secuencialmente entre el *host* emisor y el receptor. La conexión TCP es *full-duplex*, por lo tanto, los datos fluyen en los dos sentidos (emisor-receptor y receptor-emisor).

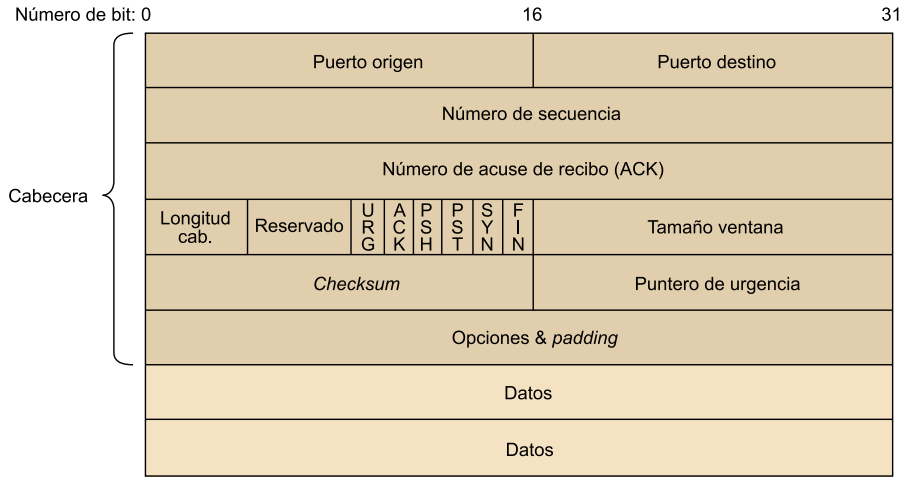
Un proceso similar, denominado ***handshake*** (que se puede traducir como apretón de manos), se lleva a cabo antes de cerrar la conexión, verificando que los dos *hosts* han terminado de enviar y recibir toda la información.

En definitiva, TCP proporciona fiabilidad a la aplicación, garantizando la entrega de toda la información en el mismo orden en el que la ha transmitido la aplicación de origen. Lo consigue proporcionando un servicio orientado a conexión con control de flujo y errores.

3.1.1. El segmento TCP por dentro

El formato del segmento TCP se representa en esta figura con el detalle de los campos de la cabecera:

Figura 6



Cabecera y campos de datos del protocolo TCP.

Pasemos a describir algunos de los principales campos:

- **Puerto origen y puerto destino**. Indican el número del puerto que inicia la transmisión al *host* origen y el número del puerto destino de la transmisión. Cada uno de estos campos tiene una longitud de 16 bits.

- **Número de secuencia**. Identifica el primer byte del campo de datos. De hecho, TCP no numera segmentos, sino que numera bytes. A partir del establecimiento de la conexión se asigna un número inicial denominado **ISN** (*initial sequence number*), y a partir de este se van numerando los bytes que se envían. Este campo tiene 32 bits de longitud.

- **Número de acuse de recibo ACK (*acknowledgment number*)** o **número reconocido**. Se utiliza para enviar un acuse de recibo. Cuando el bit ACK está activo (vale 1), este campo indica el número de secuencia del segmento que se espera recibir a continuación. Su longitud es de 32 bits.

- **Longitud de la cabecera**. Igual que en el protocolo IP, la cabecera es variable, dado que puede haber opciones o no. Este campo indica la longitud en múltiplos de 4 bytes.

- **Checksum** o **suma de verificación**. Este campo actúa del mismo modo que hemos visto en el protocolo IP, con la diferencia de que aquel solo se aplicaba a la cabecera. Este se aplica a todo el segmento TCP y permite al *host* de destino determinar si el segmento recibido tiene errores. Su longitud es de 16 bits.

- **Puntero de urgencia**. Es el último campo obligatorio de un encabezamiento TCP. Cuando el bit de control denominado URG está activo (vale 1), este campo de 16 bits indica que hay datos urgentes en el campo datos y su localización.

- **Opciones y carácter de relleno (*padding*)**. El tamaño de este campo es variable y puede ir de 0 a 32 bits, en función de si el segmento contiene información adicional (como por ejemplo el tamaño máximo de segmento que puede soportar una red) o no. Si se usan opciones, la parte de caracteres de relleno (*padding*) contiene bits de cumplimentación para que el tamaño total del campo llegue hasta los 32 bits.

- **Datos**. Contiene los datos enviados por el *host* origen. El tamaño de este campo es variable.

3.2. Los puertos en el protocolo TCP

Cada proceso que desea comunicarse con otro se identifica en la pila de protocolos TCP/IP con uno o más puertos. Un puerto es un número de 16 bits, empleado por un protocolo extremo a extremo para identificar a qué protocolo del nivel superior o programa de aplicación se han de entregar los datos recibidos.

La confusión que se produce cuando dos aplicaciones diferentes intentan usar los mismos puertos en un ordenador se evita haciendo que soliciten un puerto disponible a TCP/IP. Como este número se asigna dinámicamente, puede ser diferente en cada ejecución de una misma aplicación.

El protocolo TCP identifica los extremos de una conexión por las direcciones IP de los dos nodos implicados (servidor y cliente) y el número de los puertos de cada nodo.

Por lo tanto, con los puertos, como canales de entrada/salida que son, deberemos adoptar algunas medidas básicas de seguridad. Es importante conocer cuáles están abiertos y cuál es el motivo por el que lo están. Así podríamos llegar a cerrar los puertos que no se utilicen, teniendo siempre en cuenta que si cerramos un puerto de un servicio que estemos utilizando, este dejará de funcionar.

Es importante conocer los puertos que tenemos abiertos en nuestro sistema. Para saber qué puertos tenemos abiertos en nuestro ordenador, se puede utilizar algún programa que escanee los puertos. También existen escáneres de puertos en línea, que comprueban si están abiertos, cerrados o no existen. En todo caso, si detectamos puertos abiertos, esto no necesariamente tiene que atribuirse a un troyano, sino a las propias necesidades del sistema. Y si tenemos dudas de algún tipo de infección debemos pasar sin dudarlo un buen antivirus. También puede llegar a darse la posibilidad de que nuestro ordenador se vea afectado por un escaneo de puertos (tantear las entradas y salidas para observar cómo se encuentran) sin nuestra autorización.

Una vez detectados los puertos abiertos, podemos o bien bloquear puertos (con algún programa diseñado para ello), o bien utilizar un cortafuegos. La opción más utilizada es instalar un cortafuegos. Con él determinaremos a través de qué puertos permitimos el acceso a nuestro sistema.

A continuación presentamos un listado con la asociación de puertos, protocolos y aplicaciones realizada por **IANA**:

Número de puerto	Protocolo	Aplicación o proceso
7	TCP y UDP	Echo
20	TCP	FTP data
21	TCP	FTP control
22	TCP	SSH
23	TCP	Telnet
25	TCP	SMTP
53	UDP, TCP	DNS
67,68	UDP	DHCP
69	UDP	TFTP
80	TCP	HTTP

Tabla de puertos más utilizados, con el protocolo concreto que utilizan (TCP o UDP). Los detalles de funcionamiento de estos procesos o aplicaciones los veremos en el próximo módulo, en el que trataremos la capa de aplicación.

Número de puerto	Protocolo	Aplicación o proceso
110	TCP	POP3
119	TCP	NNTP
143	TCP	IMAP
161	UDP	SNMP
443	TCP	SSL
16.384 - 32.767	UDP	RTP - para voz y vídeo

Tabla de puertos más utilizados, con el protocolo concreto que utilizan (TCP o UDP). Los detalles de funcionamiento de estos procesos o aplicaciones los veremos en el próximo módulo, en el que trataremos la capa de aplicación.

Enlace

La lista oficial con todas las asignaciones de nombres de servicios, puertos empleados y protocolo asociado (TCP, UDP, DCCP y SCTP, estos dos últimos protocolos los describiremos en un próximo apartado) la podemos encontrar en:

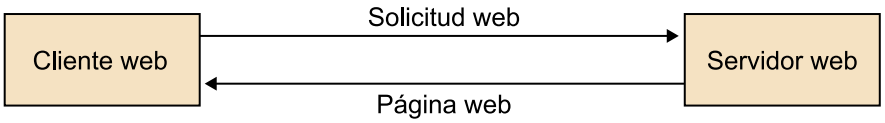
http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

3.3. Ejemplos de conexión TCP

a) Ejemplo 1 - HTTP

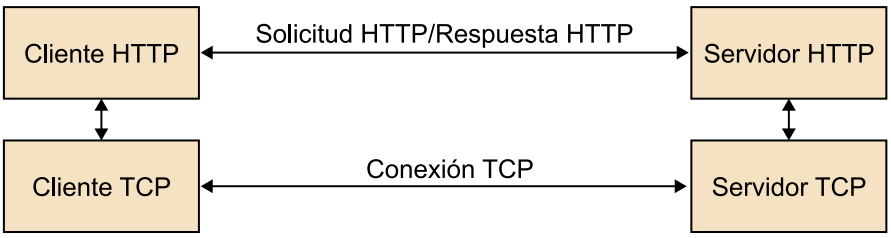
Un usuario desde un navegador de un *host* A realiza una petición de acceso a la dirección URL http://www.uoc.edu, que se localiza en otro *host* remoto B. Veamos con detalle lo que sucede en la red:

Figura 7



Esquema básico de la petición web y respuesta dada por el servidor.

Figura 8



Detalle de la actuación entre capas (aplicación y transporte).

Cuando un navegador web solicita una página a un servidor remoto, le pasa el nivel TCP una serie de bytes que contienen una petición HTTP y delante una cabecera TCP donde, entre otros campos, tenemos: una dirección IP (del servidor) y un número de puerto que, en el caso de HTTP, es el puerto 80 (el puerto donde los servidores web esperan las conexiones, "están escuchando el

puerto 80"). Como puerto de salida, TCP elige al azar uno que esté libre, por ejemplo el 1448. Este puerto de salida es, en realidad, el puerto por el que el *host* origen esperará la respuesta del *host* destino.

Así, el segmento será:

Figura 9

Segmento TCP creado por el *host* A, con la cabecera TCP (con los campos descritos) y el campo de datos que contiene la petición HTTP.

Este segmento pasará a IP, con las direcciones del *host* A y del *host* remoto B en su cabecera, junto con otros campos.

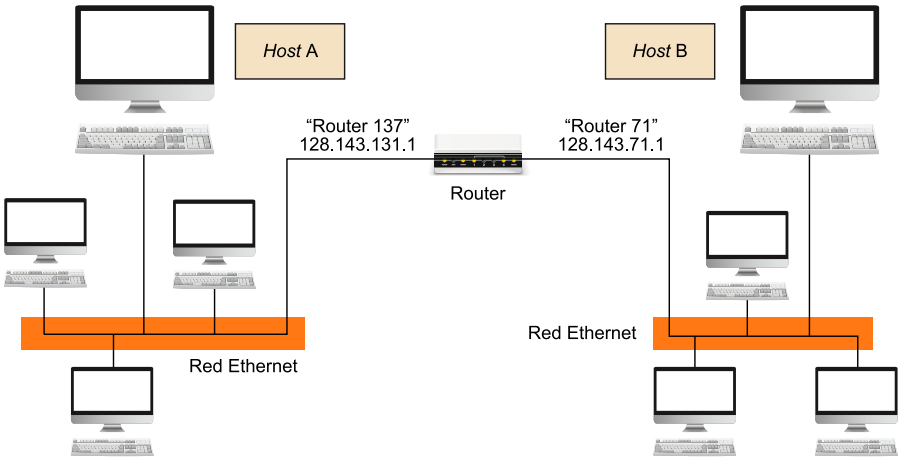
De este modo, tendremos el siguiente paquete de datos IP:

Figura 10

Paquete de datos IP, con la cabecera IP (con los campos descritos) y el campo de datos que contiene el segmento TCP anterior.

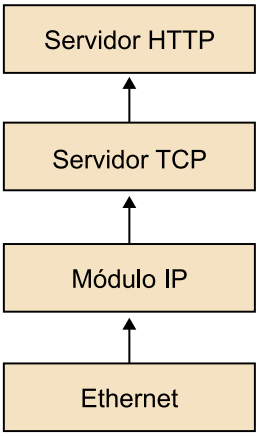
Este paquete será el que pasará por algunos routers, llegando al *host* B en el que se halla el servidor web.

Figura 11



En este esquema los paquetes circularán por dos redes que están enlazadas por un router. Fijémonos en que las dos redes tienen rangos (128.143.137.x y 128.143.71.y) diferenciados.

Figura 12



Cuando el paquete llega al *host* B, las diferentes capas irán extrayendo la información que les corresponde, desde la capa física hasta la capa de aplicación.

Mediante el nivel IP se conocerá la dirección IP del *host* que le ha enviado la petición, sin ninguna otra información. Cuando el segmento llega al nivel TCP del *host* remoto, ya detecta que va dirigido a la aplicación asociada al puerto 80 (HTTP). Además, ve el puerto origen de la petición, a quien responde con un mensaje de acuse de recibo. Los datos del segmento TCP (los que hemos denominado petición HTTP) se pasan a la aplicación (tres niveles por encima de la de transporte, según el modelo OSI), que las procesará. Esta toma el fichero que contiene la página web solicitada y la entrega a su nivel IP:

Figura 13

Segmento TCP creado por el *host* remoto B, con la cabecera TCP (ahora el puerto origen es el 80 y el destino 1448, por lo tanto, se intercambian) y el campo de datos que contiene la respuesta HTTP (página web solicitada).

Este segmento se pasa al nivel IP (también intercambiando las direcciones IP):

Figura 14

Paquete de datos IP creado en el *host* remoto B, con la cabecera IP (con los campos descritos) y el campo de datos que contiene el segmento TCP anterior.

Este paquete hace el camino de regreso por la red, muy posiblemente pasando por routers diferentes, y llega al *host* A, que está esperando la respuesta por el puerto que ha enviado la petición (en nuestro ejemplo, el 1448).

b) Ejemplo 2 - Telnet

Telnet es también un servicio que utiliza TCP/IP para conectarse a un *host* remoto. El protocolo correspondiente, que tiene el mismo nombre, pertenece a la capa de aplicación. Un cliente realiza una solicitud de comunicación a un servidor por el puerto 23, entonces el servidor reconoce que el cliente está

interesado en iniciar una sesión Telnet. En ese momento, el servidor conecta con el puerto Telnet del cliente –el 23 por defecto– y establece un circuito virtual.

3.4. La aplicación netstat

La mayoría de los sistemas operativos ofrecen herramientas que permiten mostrar las conexiones de red que tenemos en cada momento. Entre ellas nos encontramos con la aplicación netstat. Para ejecutarla, introduciremos en la pantalla de sistema: **netstat -an**.

Para entender mejor qué conexiones tenemos abiertas, lo mejor es que antes de ejecutar esta orden cerremos todos los programas a excepción de la pantalla de sistema, para así ir desde el principio comprobando qué conexiones tenemos y cuáles se van abriendo.

Si queremos que se actualice automáticamente la información, podemos escribir **netstat -an 5** (poned el número en segundos del intervalo que queramos que actualice la información).

Para obtener una pequeña ayuda con relación a netstat ejecutaremos: **netstat /help**.

Un ejemplo de la información que nos facilita **netstat** es la que se muestra en esta imagen:

Figura 15

```
Conexiones activas

  Proto  Dirección local        Dirección remota       Estado
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3022           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3284           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3286           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3287           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5000           0.0.0.0:0              LISTENING
  TCP    127.0.0.1:3001         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:3002         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:3003         0.0.0.0:0              LISTENING
  TCP    193.153.12.193:3284    207.46.106.87:1863     ESTABLISHED
  TCP    193.153.12.193:3286    213.73.40.217:80       CLOSE_WAIT
  TCP    193.153.12.193:3287    213.73.40.217:80       CLOSE_WAIT
  TCP    193.153.12.193:3289    213.73.40.68:80        TIME_WAIT
  TCP    193.153.12.193:9396    0.0.0.0:0              LISTENING
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:1026           *:*
  UDP    0.0.0.0:3006           *:*
  UDP    0.0.0.0:3014           *:*
  UDP    0.0.0.0:3031           *:*
  UDP    127.0.0.1:123          *:*
  UDP    127.0.0.1:1900         *:*
  UDP    127.0.0.1:3013         *:*
  UDP    127.0.0.1:3077         *:*
  UDP    127.0.0.1:3285         *:*
  UDP    193.153.12.193:123     *:*
  UDP    193.153.12.193:1900    *:*
  UDP    193.153.12.193:9591    *:*
  UDP    193.153.12.193:57996   *:*
```

Pantalla en la que se muestra el detalle de la información que nos facilita el comando **netstat**. En la primera columna, el protocolo empleado (TCP o UDP). Podemos también comprobar que al principio, al no estar conectados a internet, tenemos direcciones locales, pero no tenemos dirección remota (equipos a los que nos conectamos). Todos los estados están en *listening*, lo que significaría que el puerto está disponible. Al conectarnos a internet vemos que se nos abren nuevos puertos, entre los

que se halla el 3284, que sería la conexión con el servidor. Cuando utilizamos el navegador para intentar conectarnos a una dirección web, podemos ver que se nos abre automáticamente el puerto 80, que sería el de www.

4. Otros protocolos de la red de transporte

Además de los protocolos mencionados TCP y UDP empleados en la capa de transporte, existen otros protocolos de esta capa de transporte que se utilizan en diferentes ámbitos y situaciones en sustitución de TCP y UDP. A continuación pasaremos a describir los protocolos **DCCP** (*datagram congestion control protocol*) y **SCTP** (*stream control transmission protocol*).

4.1. DCCP

El protocolo **DCCP**, acrónimo de *datagram congestion control protocol* (DCCP) o **protocolo de control de congestión de datagramas**, es un protocolo de nivel de transporte orientado al mensaje, orientado a conexión.

Lo utilizan aquellas aplicaciones que tienen necesidad de un envío rápido de los datos, pero con un correcto control de la congestión. Por ejemplo, la **telefonía en internet** y **multimedia en tiempo real**, entre otras.

El hecho de emplear DCCP en esta capa hace que las aplicaciones no necesiten implementar el control de la congestión dentro del nivel de aplicación.

Por lo tanto, da buena respuesta a aplicaciones que, siguiendo la semántica basada en flujo de TCP, no necesitan ni la fiabilidad ni la entrega ordenada de la información que ofrece TCP, o incluso que quieren un control de la congestión diferente y más uniforme para las aplicaciones.

Antes de que surgiera este protocolo, si no se quería emplear TCP, a menudo se empleaba UDP, dotándolo de mecanismos adicionales de control de la congestión. DCCP fue publicado como RFC 4340, en la categoría de estándar propuesto por el **IETF** en marzo del 2006.

4.2. SCTP

El protocolo SCTP, acrónimo de *stream control transmission protocol*, es un protocolo de la capa de transporte definido por el IETF en el año 2000. El protocolo está especificado en la RFC 2960 y la RFC 3286.

No deja de ser una alternativa a los protocolos de transporte TCP y UDP, dado que ofrece fiabilidad, control de flujo y secuencia como TCP. Aun así, de manera opcional SCTP permite el envío de mensajes desordenadamnete, y a diferencia de TCP es un protocolo orientado al mensaje (similar a UDP).

5. Aplicaciones específicas para redes locales: analizadores de protocolos

En el caso de que queramos analizar el estado de la red, necesitamos saber el tipo de datos que circula por ella. Para hacerlo, podemos utilizar **analizadores de protocolos**, que permiten obtener muestras del tráfico de la red en tiempo real y posteriormente analizarlas.

Los sistemas operativos Linux integran diferentes herramientas para llevarlo a cabo (tcpdump...). En otros sistemas operativos, como Windows, se pueden instalar. En este sentido, a continuación, destacamos algunas de las **aplicaciones específicas** que permiten reconocer y obtener algunas características de la red local, de los equipos conectados, del tráfico, etc.

1) Overlook fing:

Figura 16

Disponible para ordenadores con Windows, OS X y Linux. También es una app (para Android o iOS) que permite el análisis de la red en la que estemos. Nace a partir de la antigua LOOK@LAN, una aplicación que permitía realizar un completo análisis de la red local, de manera sencilla y totalmente automatizada. La interfaz principal de la aplicación muestra toda la información recopilada como resultado del análisis: dirección IP, estado, grupo de red, sistema operativo, nombre del *host*, usuario, etc.

Este analizador lo podéis descargar en: http://www.overlooksoft.com/

Para detalles más concretos de su funcionamiento, podéis visualizar el siguiente vídeo: http://www.youtube.com/embed/0EkiPPwv010?rel=0&controls=0&showinfo=0

2) SoftPerfect Network Scanner (aplicación *freeware*):

Figura 17

Realiza un escaneo de la red para poder facilitar sus parámetros. Lo podéis obtener en: http://www.softperfect.com/products/networkscanner/.

3) NetworkView 3.50 (aplicación *shareware*):

Figura 18

Esta es una aplicación útil para realizar estudios de redes locales y crear diagramas con la distribución de los ordenadores y las conexiones existentes entre ellos. Con NetworkView se puede trazar en poco tiempo un completo esquema de vuestra red local, detectar todos los nodos de TCP/IP y sus rutas utilizando la información facilitada por los DNS, SNMP y los puertos TCP. Esta herramienta se encuentra disponible en www.networkview.com.

4) Wireshark:

Figura 19

Esta es otra utilidad también muy interesante para analizar protocolos y capturar paquetes que circulan por la red. Antes era conocida como **Ethereal**.

Wireshark es un analizador de protocolos, utilizado, entre otros, para el análisis y la solución de problemas en redes. Sus funcionalidades son parecidas a las de *tcpdump* pero incorporando una interfaz gráfica.

Tiene una versión para Windows y para Mac OS X de Apple. Las podéis encontrar en: http://www.wireshark.org/. Si queréis más información, podéis utilizar el manual *Análisis de tráfico con Wireshark*, de **INTECO-CERT** (https://www.incibe.es/CERT/guias_estudios/guias/guia_wireshark).