Universidad de Alcalá

Departamento de Automática

ARQUITECTURA DE REDES
Laboratorio

Práctica 7: Protocolos de transporte en TCP/IP

Grado en Ingeniería Informática
Curso 2022/23

1. OBJETIVO

Conocer las características de los protocolos de transporte de la arquitectura TCP/IP: TCP y UDP.

2. PROTOCOLOS DE TRANSPORTE TCP/IP

TCP y UDP son los protocolos de transporte de la arquitectura TCP/IP. Se trata de dos protocolos muy diferentes entre sí, lo que permite que se pueda seleccionar en cada caso el más adecuado a las necesidades de la aplicación. Mientras que TCP es un protocolo orientado a conexión y confiable, UDP ofrece un transporte poco fiable pero rápido y con poca carga adicional en la red. En la figura 1 se muestra un esquema con los protocolos de transporte utilizados por diferentes aplicaciones conocidas.

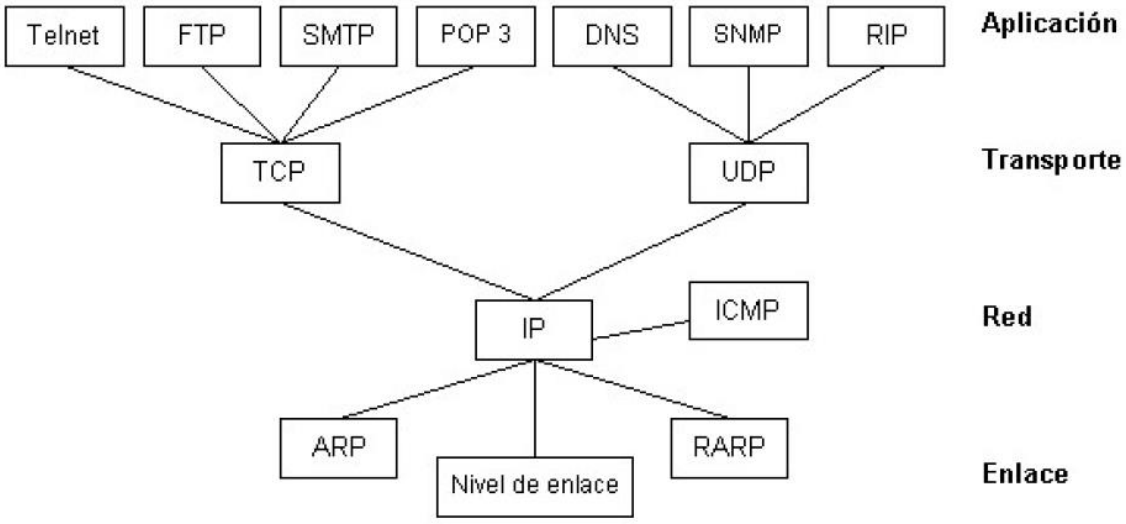

Figura 1: Protocolos en la arquitectura TCP/IP

En la capa de transporte se utilizan como direcciones valores de 16 bits conocidos como **puertos**, que son identificadores de los buffers presentes en las máquinas involucradas en la comunicación. Estos buffers son el interfaz entre la capa de aplicación y la de red, de forma que cada aplicación o proceso de la capa de aplicación tiene asignado un buffer a través del cual intercambia información con la capa de transporte. Una vez establecidos estos buffers la capa de transporte envía esta información en bloques de tamaño adecuado a la capa de red. De esta manera una misma máquina puede tener varios procesos independientes que emitan o reciban paquetes a nivel de transporte, tal como se muestra en la figura 2. Al conjunto formado por una dirección IP y el puerto del proceso origen, más la IP y el puerto del proceso en el destino empleando un cierto protocolo de transporte (TCP o UDP) se le conoce como socket.

En la página http://www.iana.org/assignments/port-numbers se puede obtener una lista completa los puertos reconocidos oficialmente por IANA (*Internet Assigned Numbers Authority*).

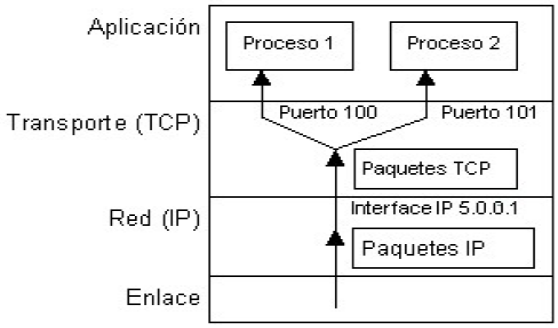

Figura 2: Comunicación entre la capa de transporte y la de aplicación

Los puertos en el rango 1-1023 se denominan puertos bien conocidos y están reservados para servicios clásicos; los del rango 1024-49151 son puertos registrados; el rango 49152-65535 son puertos privados o dinámicos y son empleados para servicios temporales.

3. El PROTOCOLO DE TRANSPORTE TCP

Las principales características del protocolo TCP son las siguientes:

- Trabaja con un flujo de bytes ordenado.
- Transmisión orientada a conexión, lo que implica un procedimiento previo de conexión y posterior de desconexión
- Fiable, ya que utiliza procedimientos de reconocimiento explícito (ACKs o *Acknowledgements*) para confirmar las tramas válidas recibidas.
- Realiza control de flujo de manera que en receptor envía un ACK del siguiente byte que puede recibir como indicador de confirmación de la correcta recepción del byte anterior. Ante la posible ausencia de ACKs, existe un tiempo máximo de espera transcurrido el cual se reenvía de nuevo el paquete no confirmado.

El formato del segmento TCP se muestra en la figura 3.

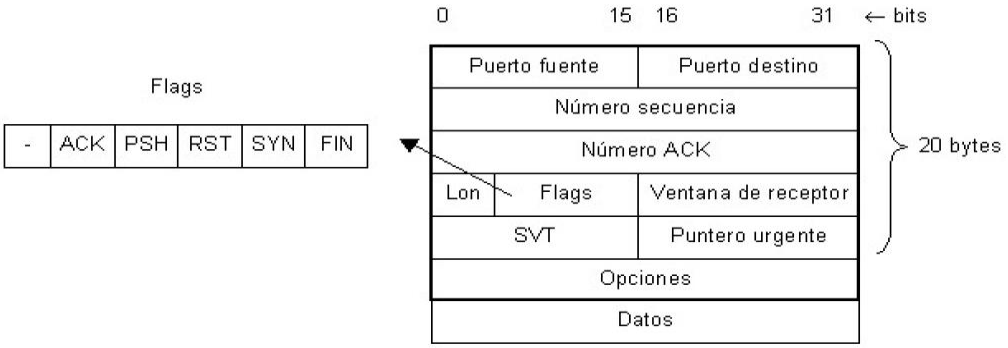

Figura 3: Segmento TCP

En la cabecera del segmento TCP (figura 3) se incluyen los siguientes campos:

- **Puertos origen y destino**. Valores de 16 bits correspondientes a los identificadores de los puertos de nivel de transporte.
- **Número de secuencia**. Coincide con la numeración del primer byte del campo de datos.
- **Número de ACK**. Número del primer byte de datos que se espera recibir en el próximo segmento.
- **Lon** (cuatro bits). Número de palabras de 32 bits (cuatro bytes) que forman la cabecera TCP.
- **Flags**. Seis bits con significado propio:
  - *ACK*. Cuando toma el valor 1 indica que el número de ACK es válido y debe interpretarse como un reconocimiento válido
  - *PSH*. Cuando toma el valor 1 indica que el proceso TCP del receptor debe pasar los datos que tenga almacenados a la capa de aplicación sin esperar a recibir más datos.
  - *RST*. Se usa cuando ha habido un problema en la secuencia de bytes, cuando falla un intento de iniciar conexión o para rechazar paquetes no válidos.
  - *SYN*. Se utiliza para solicitar establecimiento de una conexión.
  - *FIN*. Se utiliza para solicitar la liberación de una conexión.
  - *URG*. Indica que el valor del campo *Urgent Pointer* debe ser tenido en cuenta porque el segmento TCP incluye información urgente.
- **Ventana**. Sirve para informar sobre el número de bytes que el proceso TCP del emisor del paquete es capaz de recibir en su buffer de recepción. Si vale cero indica que no se pueden recibir datos (aunque sí se puede interpretar los paquetes con flags ACK, RST, FIN...).
- **SVT**. Suma de verificación, aplicada a la cabecera y datos TCP, además de a algún campo de la cabecera IP.
- **Puntero urgente**. Desplazamiento en bytes desde el número de secuencia indicado, a partir del cual hay información urgente.
- **Opciones**. Permite campos adicionales.

El proceso de conexión y desconexión en TCP se muestra en la figura 4, donde cualquiera de los hosts implicados en la comunicación puede iniciar el proceso.

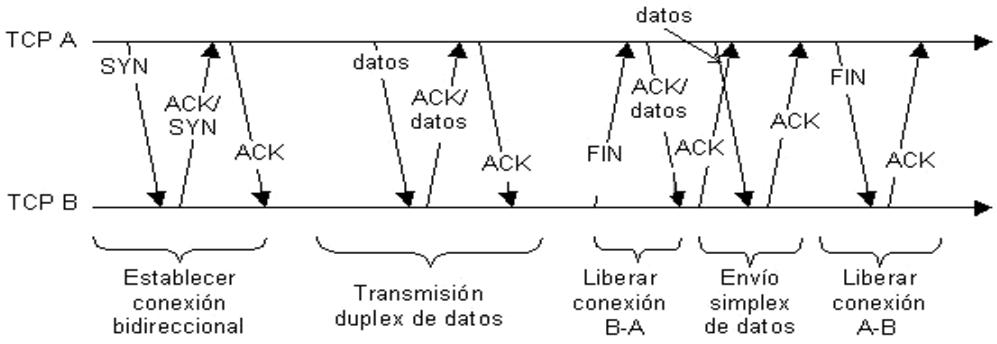

Figura 4: Proceso de conexión y desconexión en TCP

Habitualmente es el cliente (A) el que envía un primer paquete TCP con el bit SYC activado a un servidor (B) para solicitar el inicio de una conexión. El servidor puede aceptar esa conexión contestando con un paquete TCP con el bit ACK activado, o rechazar

la conexión enviando un TCP con el bit RST activado. Si el servidor acepta la conexión, aprovecha el paquete TCP de ACK para enviar también su solicitud de conexión en sentido inverso, activando el bit SYN del mismo paquete. El cliente aceptará enviando un paquete TCP con ACK.

En el establecimiento de conexión se realiza la gestión de ciertos parámetros de la comunicación empleando el campo de opciones de la cabecera TCP. El cliente y el servidor pueden negociar el valor de ciertos parámetros, como el MTU a utilizar, o informar sobre los datos que pueden a recibir sin sobrepasar sus buffers. Tras realizar la conexión se pueden intercambiar datos con paquetes TCP.

Puesto que se usa un sistema de envío continuo, un equipo puede activar el bit de ACK de un paquete TCP con el que envía datos para confirmar al mismo tiempo los datos previamente recibidos.

Para la desconexión, el equipo que quiere desconectar envía un paquete TCP con el bit FIN activado, y el otro equipo responde con un TCP con ACK activado. El que equipo que responde puede seguir enviando datos, pero lo habitual es que también solicite desconexión con otro paquete TCP con FIN activado.

Al contrario que otros protocolos, TCP numera los bytes que envía, no los segmentos.. Así, un proceso TCP emisor comenzará a numerar los bytes que recibe de su aplicación, y cuando coloque una cantidad de estos bytes es un segmento, el número de secuencia corresponderá al número del primer byte. La numeración no tiene porque empezar por cero para el primer byte de un flujo de datos. Desde ese momento, el proceso TCP emisor esperará recibir un segmento TCP con ACK, y un número de ACK correspondiente al siguiente byte que debe enviar. Pero puede ser que el proceso receptor tuviese problemas, en cuyo caso el TCP ACK que envía indicará el número del byte a partir del cual el proceso emisor debe reenviar datos.

Por otra parte, un proceso TCP receptor tiene un buffer en donde va acumulando los bytes recibidos hasta que su aplicación los lee. Este buffer es limitado, y si la aplicación no lee su contenido en durante la recepción de datos, se puede llenar. El receptor puede informar a su emisor en todo momento de la capacidad que queda libre en ese buffer, mediante el campo de "ventana de receptor". Si un proceso emisor recibe un valor 0 en "ventana de receptor", debe interpretar que el proceso TCP receptor no puede atender más datos de momento, y debe parar la emisión de bytes hasta que reciba un segmento con un valor de "ventana de receptor" mayor a cero.

En el momento de establecer una conexión, los valores de "ventana de receptor" que aparecen en los paquetes TCP SYN reflejan el tamaño máximo de los buffers de recepción para esa conexión. Es habitual que un proceso cliente emplee un tamaño más grande que su servidor, ya que el servidor debe repartir su memoria para atender a muchos clientes.

4. El PROTOCOLO DE TRANSPORTE UDP

Las características principales de este protocolo son:

- Sin conexión. No emplea ninguna sincronización entre origen y destino.
- Trabaja con paquetes o datagramas enteros, no con bytes individuales como TCP.
- Una aplicación que emplea el protocolo UDP intercambia información en forma de bloques de bytes, de forma que por cada bloque de bytes enviado de la capa de aplicación a la capa de transporte, se envía un paquete UDP.
- No es fiable. No emplea control del flujo ni ordena los paquetes.
- Su gran ventaja es que provoca poca carga adicional en la red, ya que es sencillo y emplea cabeceras muy simples.
- Un paquete UDP puede ser fragmentando por el protocolo IP para ser enviado fragmentado en varios paquetes IP si resulta necesario.
- Puesto que no hay conexión, un paquete UDP admite utilizar como dirección IP de destino la dirección de broadcast o de multicast de IP. Esto permite enviar un mismo paquete a varios destinos de forma simultánea.

En la figura 4 se muestra el formato del segmento UDP.

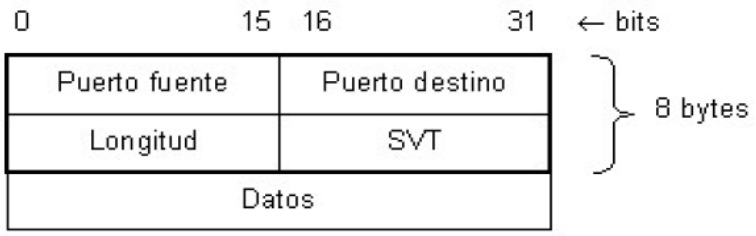

Figura 4: Segmento UDP

Los campos de la cabecera UDP son los siguientes:

- **Puerto fuente y puerto destino**. Valores de 16 bits correspondientes a los puertos de nivel de transporte.
- **Longitud**. Número total de bytes en el paquete UDP original incluye la cabecera y los datos), antes de ser fragmentado en paquetes IP.
- **SVT**. Suma de verificación, aplicada a la cabecera y datos UDP, además de a algún campo de la cabecera IP.

Algunas aplicaciones de UDP pueden ser:

- Transmisión de datos en LANs fiable, como el protocolo TFTP (*Trivial File Transfer Protocol*), que es una variante del protocolo FTP que emplea como protocolo de transporte UDP.
- Operaciones de sondeo. Transmisión de paquetes de datos pequeños o esporádicos para informar del estado de los equipos de la red, o para intercambiar información de encaminamiento, como es el caso de los protocolos DNS, RIP (Routing Information protocol) o SNMP (Simple Network Management Protocol).
- Transmisiones multicast de video o audio. UDP es usado por aplicaciones de VoIP, difusión de video y multiconferencia. En la transmisión de señales digitales suele ser más importante una respuesta rápida de los protocolos que un envío completamente fiable. No importa que se pierdan algunos datos: lo importante es

que se mantenga un flujo constante de información. Además con UDP es posible que una misma fuente envíe la señal a múltiples destinos, sin repetir paquetes de datos en la red.

- Otra aplicación es el envío de transacciones rápidas a BB.DD a través de redes LAN fiables. En este caso también premia la rapidez de respuesta, y dado que la red ofrece una alta calidad, no es necesario el complejo control de flujo de TCP.

5. NOTAS SOBRE WIRESHARK

WireShark dispone de una serie de opciones relativas a las conexiones y paquetes de TCP que conviene verificar. Estas opciones están situadas en *Edit →Preferences→ Protocols→ TCP*:

- *Validate TCP Checksum if possible*. Si está activada, WireShark mostrará cuando las sumas de chequeo de TCP son correctas o no.
- *Allow subdissector to reassemble TCP streams*. Conviene desactivarla, porque sino WireShark intentará juntar algunos segmentos TCP de una misma conexión.
- *Analyze TCP sequence numbers*. Se puede dejar activada para que WireShark pueda seguir el estado de las conexiones TCP.
- *Relative sequence numbers and window scaling*. Si se activa esta opción, WireShark mostrará los números de secuencia de bytes y de ACK de una conexión en relación a los valores de secuencia que había al iniciar la conexión, con lo que se verán las numeraciones desde cero. Esto puede facilitar la compresión de cómo funcionan los números de secuencia.

6. ACTIVIDADES

- Inicie una ventana de consola e inicie el programa Wireshark para realizar una captura.
- Capture todo el tráfico que se produce en el siguiente proceso:
  - Conéctese por FTP al servidor ftp://130.206.13.2/ como usuario anónimo
  - Cambie al directorio: /sites/apache.org/
  - Utilice el comando get para descargar el archivo HEADER.html presente en ese directorio.
  - Cierre la conexión FTP, pare la captura de paquetes y almacene el archivo con la captura.
- Analice los siguientes parámetros:
  - Tiempo de respuesta del servidor DNS a la consulta que se le realiza mediante UDP
  - Contenido de los campos de los paquetes UDP de consulta y respuesta.
  - Tiempo requerido para la conexión y desconexión del canal de control de TCP
  - Tiempo requerido para la conexión y desconexión del canal de datos de TCP
  - Rendimiento de la transferencia del archivo descargado, entendiendo por tal la relación entre carga adicional transmitida por el protocolo (carga de pago) y carga útil o carga de datos del archivo.
  - Evolución de los tiempos de reconocimiento o respuesta (RTT).

- Realice el mismo análisis anterior sobre la captura almacenada en el archivo capturaftpP7.libcap y que se corresponde con un proceso similar realizado en otro equipo, en otra ubicación, y con otro servidor y fichero.
- Repita el proceso de análisis para la descarga del archivo README.txt , ubicado en la carpeta /debían del servidor ftp.debian.org y justifique las diferencias.

7. EVALUACIÓN

En la evaluación de esta práctica se plantearán cuestiones de análisis de conexiones TCP y UDP

8. BIBLIOGRAFÍA

La señalada como básica para el laboratorio.

Para esta práctica se ha tomado como base el Manual de Prácticas de la asignatura de Redes del Grupo de Innovación Educativa en Automática de la Universidad de Alicante.