PRÁCTICA 3

Protocolos de nivel de Transporte en TCP/IP

REDES (9359)

ING. TÉCNICA EN INFORMÁTICA DE SISTEMAS

Curso 2010/2011

(*Este documento es una versión en papel de la versión completa en formato web-SCORM publicada a través de la plataforma Moodle-UA*)

Pablo Gil Vázquez (Pablo.Gil@ua.es)
Grupo de Innovación Educativa en Automática

Universitat d'Alacant
Universidad de Alicante

3.1. Introducción

Los objetivos de la práctica 3 de la asignatura Redes es profundizar en el funcionamiento de los protocolos de transporte en la arquitectura de red. En particular, se pretende conocer las características del nivel de transporte, así como el formato de las estructuras de datos que circulan en este nivel. La información de uso y funcionamiento de los protocolo de transporte más habituales UDP y TCP se puede encontrar en las RFC 768 Y 793 respectivamente (*http://www.rfc-es.org/rfc/*).

3.2. Revisión de los niveles de la arquitectura TCP/IP

Los protocolos de nivel de transporte en TCP/IP proporcionan una comunicación extremo a extremo. Dentro de los servicios deseables de un protocolo de transporte cabe mencionar el proveer un servicio fiable, asegurando que los datos llegan a su destino sin errores (realizando comprobaciones utilizando sumas de verificación) y en el orden correcto. Además, el nivel de transporte debe ser capaz de coordinar los distintos procesos a nivel de aplicación de forma que se asegure que los datos llegan de forma correcta a la aplicación destino. También puede garantizar la ausencia de duplicados así como una sincronización entre los extremos de una conexión posibilitando realizar el envío de los datos independientemente de la longitud de los mismos. En la siguiente figura se aprecia la arquitectura de protocolos TCP/IP en donde se sitúan los protocolos de transporte TCP y UDP.

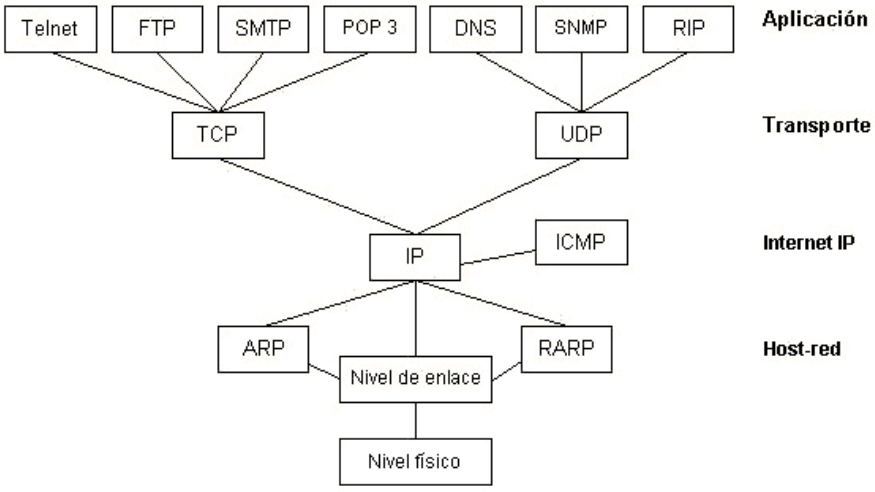

Figura 1. Conjunto de protocolos TCP/IP.

Los protocolos de transporte de la arquitectura TCP/IP que van a ser objeto de estudio en la presente práctica son el TCP (protocolo de control de transporte) y UDP (protocolo de datagramas de usuario). TCP/IP ofrece dos opciones de protocolo de transporte al nivel de aplicación cada uno de ellos con unas características determinadas. La aplicación de un protocolo u otro va a depender directamente del uso al que esté orientado, de forma que es posible encontrar aplicaciones que utilizan TCP y otras que usan UDP.

En general, no se puede afirmar que un protocolo (TCP o UDP) sea mejor que el otro. Como características principales, TCP es un protocolo orientado a conexión que ofrece un servicio muy fiable aunque implica

bastante tráfico adicional en la red. En cambio UDP no está orientado a conexión y ofrece un servicio poco fiable, aunque rápido y con poca carga adicional en la red.

Los paquetes que emplean los protocolos TCP y UDP (también conocidos como segmentos) se encapsulan dentro del campo de datos de paquetes IP tal como se aprecia en la figura 2. Habrá que tener en cuenta el MSS, que es el tamaño máximo de los datos de aplicación que el nivel de transporte acepta para evitar la fragmentación que, evidentemente, dependerá del valor de la MTU.

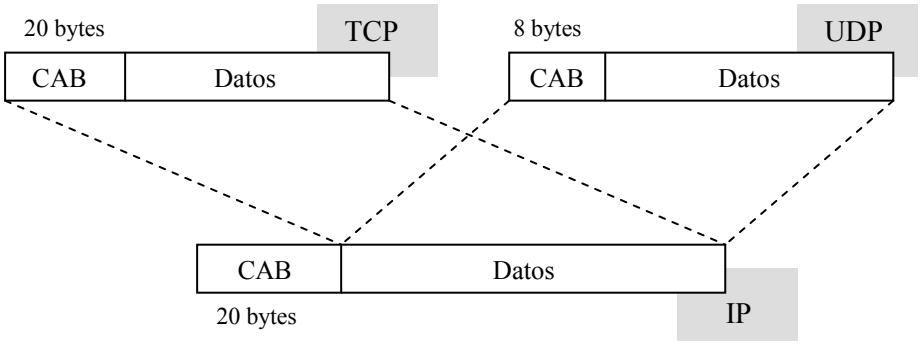

Figura 2. Encapsulación de protocolos de transporte y aplicación en IP.

3.3. Puertos y sockets

Es posible encontrar en un momento dado distintos servicios a nivel de transporte ejecutándose de forma simultánea. Este hecho va a propiciar la necesidad de crear un método para identificar cada uno de los procesos de aplicación que está atendiendo el protocolo a nivel de transporte. Para realizar esta identificación ambos protocolos de transporte utilizan identificadores de 16 bits denominados puertos que permiten identificar los procesos origen y destino dentro de las máquinas origen y destino respectivamente. Así, cuando un proceso a nivel de aplicación desea establecer una conexión con un servidor, es necesario de alguna manera conocer un identificador de dicho servidor. Si el cliente que desea establecer la conexión conoce únicamente la dirección IP del servidor, pero no el puerto, no se podrá realizar la solicitud del servicio. Por lo tanto, los puertos van a permitir identificar los procesos a nivel de aplicación, sin embargo, es de destacar que los puertos TCP son independientes de los UDP de forma que es posible encontrar dos procesos utilizando el puerto 1150, uno TCP y otro distinto para UDP.

De lo visto se puede afirmar que una misma máquina puede tener varios procesos independientes que emitan o reciban paquetes a nivel de transporte como se muestra en la figura 3.

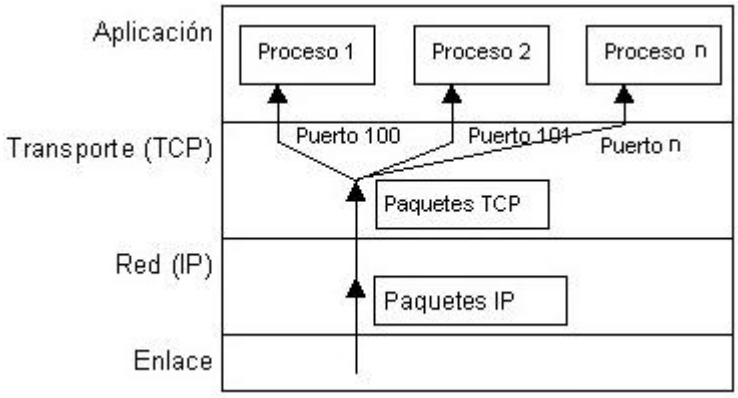

Figura 3. Procesos en ejecución en una máquina con conexión TCP/IP.

Para que dos procesos a nivel de aplicación puedan comunicarse entre sí, es necesario establecer un canal de comunicación. Esto es lo que se llama un socket. Para establecer dicho canal, se necesita una dirección IP junto con un puerto origen, así como una dirección IP y un puerto destino. A la combinación de estas dos direcciones y estos dos puertos con un cierto protocolo de transporte (TCP o UDP) es lo que se le denomina un socket. En muchas ocasiones el par dirección IP-puerto se suele indicar con el siguiente formato 195.23.12.12:80 para hacer referencia al puerto 80 de la máquina con dirección IP 195.23.12.12.

Servicio	Puerto
Servidor Eco (devuelve lo que se le envía).	UDP 7
Servidor Fecha y hora (devuelve la hora y fecha del sistema).	UDP 13
Servidor FTP (protocolo de transferencia de ficheros).	TCP 21
Servidor SSH (protocolo de seguridad del Shell). El servicio admite aplicaciones remotas de consola, transferencia de ficheros, ejecución de comandos, etc.	TCP 22
Servidor Telnet (protocolo de terminal remoto sin seguridad).	TCP 23
Servidor SMTP (protocolo simple de trasferencia de correo).	TCP 25
Servidor DNS (Domain Name System). Devuelve IP para nombres de dominio.	TCP 53
HTTP (Servidor Web) y HTTPS (Servidor web seguro).	TCP 80 y 443

Tabla 1. Procesos servidores y números de puerto.

Con el uso de sockets se diferencian dos tipos de procesos: servidores y clientes. Los primeros son los que ofrecen algún tipo de servicio a otros procesos, llamados clientes. Es el proceso cliente el que solicita establecer un socket con un servidor. Un servidor normalmente puede atender a varios clientes simultáneamente. Un cliente emplea un número de puerto desde 1024 hasta 5000, y dentro de una misma máquina el puerto cliente se va asignando incrementalmente. En cambio, para los procesos servidores se han definido unos puertos concretos de forma que sean conocidos de antemano por los clientes, como son los que

se muestran en la tabla 1. En la dirección web *http://www.iana.org/assignments/port-numbers* se proporciona el listado de puertos de IANA para diferentes aplicaciones.

3.4. Protocolo UDP

El protocolo UDP (user datagram protocol, protocolo de datagramas de usuario) es utilizado para conexiones que no precisan de corrección de errores o continuidad en el flujo. La función de UDP se fundamenta en mandar paquetes sueltos (denominados datagramas) al puerto destino ofreciendo un servicio no fiable y no orientado a conexión. Este protocolo deja al programa de aplicación la responsabilidad de una transmisión fiable. Las características principales de este protocolo son:

- No orientado a conexión. No emplea ninguna sincronización origen – destino.
- Trabaja con paquetes o datagramas enteros, no con bytes individuales como veremos posteriormente para TCP. Una aplicación de envío o recepción utilizando UDP por parte de una aplicación implica el envío o recepción de un paquete completo.
- 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 IP para ser enviado dentro de más de un paquete IP si resulta necesario.
- Un paquete UDP admite utilizar como dirección IP de destino la dirección de Broadcast de la red IP ya que no emplea conexiones.

Una aplicación típica de UDP son los streams de vídeo bajo demanda. En la figura 4 se muestra el formato de un paquete UDP:

Figura 4. Formato de una estructura de datos UDP.

Los campos de la cabecera tienen las siguientes funciones:

- 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 cabecera y 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 fiables (protocolo TFTP).
- Operaciones de sondeo. Transmisión de paquetes de datos pequeños o esporádicos.
- Streams de vídeo bajo demanda.

3.5. Protocolo TCP

TCP (transmission control protocol, protocolo de control de transmisión) permite que grandes volúmenes de información lleguen a su destino correctamente, pudiendo recuperar la pérdida esporádica de paquetes.

Se trata un protocolo de transferencia fiable que asocia un contador cada vez que un paquete es enviado, de forma que al expirar un tiempo máximo establecido sin haber recibido la confirmación el paquete se reenvía. TCP emplea un protocolo de ventana deslizante en el que se define un tamaño de ventana que indicará el número de paquetes a enviar sin necesidad de recibir su confirmación. Según se va recibiendo la confirmación de los primeros paquetes enviados, la ventana avanza posibilitando el envío de los siguientes paquetes.

Las características principales de este protocolo son:

- Trabaja con un flujo de bytes. El nivel de aplicación entrega o recibe desde el de transporte bytes individuales. TCP agrupa esos bytes en paquetes de tamaño adecuado para mejorar el rendimiento y evitar a la vez la fragmentación a nivel IP.
- Transmisión orientada a conexión. Se requiere una secuencia de conexión previa al envío - recepción de datos entre cliente y servidor, y una desconexión final.
- Fiable. Emplea control de flujo mediante ventana deslizante de envío continuo y asentimientos positivos o ACKs para confirmar las tramas válidas recibidas. La ventana deslizante se aplica a los bytes: se numeran y confirman bytes y no paquetes.
- Flujo de bytes ordenado. Aunque IP trabaja con datagramas, un receptor TCP ordena los paquetes que recibe para entregar los bytes al nivel superior en orden.

El control de flujo de TCP se puede resumir de la siguiente forma. Un receptor TCP envía un ACK del siguiente byte que desea recibir para confirmar una recepción correcta hasta ese byte. Si el receptor recibe errores en un paquete no se envía ACK. En el emisor se usa un tiempo máximo de espera, transcurrido el cual sin recibir ACK reenvía de nuevo el paquete.

Un paquete TCP tiene el siguiente formato, en donde destaca la complejidad de la cabecera (figura 5).

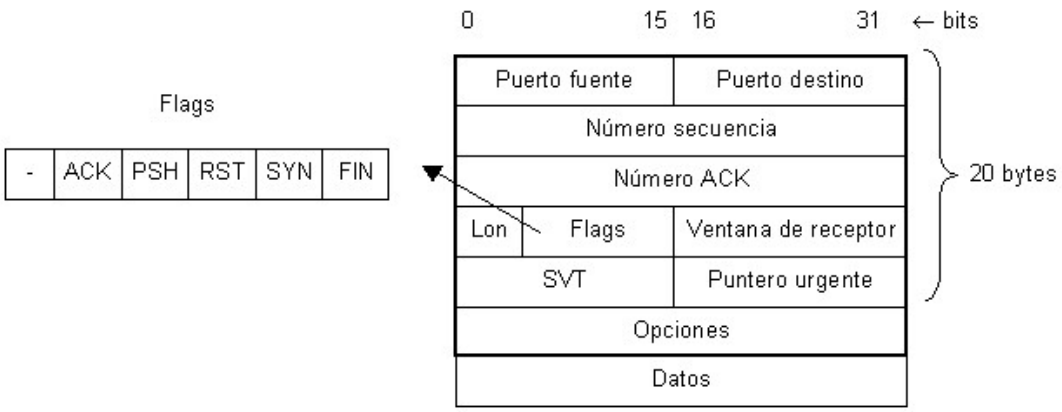

Figura 5. Formato de la cabecera de un segmento TCP.

La utilidad de los diferentes campos de la cabecera es la siguiente:

- *Puerto fuente* y *puerto destino*. Valores de 16 bits correspondientes a los puertos de nivel de transporte.
- *Número de secuencia*. Número del primer byte del campo de datos del paquete.
- *Número de ACK*. Número del siguiente byte que se espera recibir en un próximo paquete.
- *Lon (4 bits)*. Número de palabras de 32 bits (4 bytes) que forman la cabecera TCP. Como mínimo son 5 palabras ó 20 bytes cuando no existe el campo de opciones.
- *Flags*. Campo con bits con significado propio, del cual se usan sólo 6. Los más importantes de estos 6 se describen a continuación:
    - *ACK*. A 1 indica que el número de ACK es válido y debe interpretarse, es decir, el paquete tiene ACK.
    - *PSH* (push). A 1 indica que el receptor debe pasar los datos a la aplicación sin que pasen por un buffer intermedio.
    - *RST (restart)*. Solicita un reinicio de la conexión. 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 (synchronice). Se utiliza para solicitar establecimiento de una conexión.
    - *FIN*. Se utiliza para solicitar la liberación de una conexión.
- *Ventana*. Sirve para informar sobre el número de bytes que el emisor del paquete es capaz de aceptar al recibir. Este valor depende de la porción que queda libre en su buffer de recepción. Si vale 0 indica que no se van a aceptar datos (aunque si los 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.

Seguidamente, en la figura 6 se muestra un ejemplo de secuencia de conexión y desconexión de TCP. Hay que destacar que la desconexión se puede efectuar en un solo sentido aunque normalmente se efectúa en ambos.

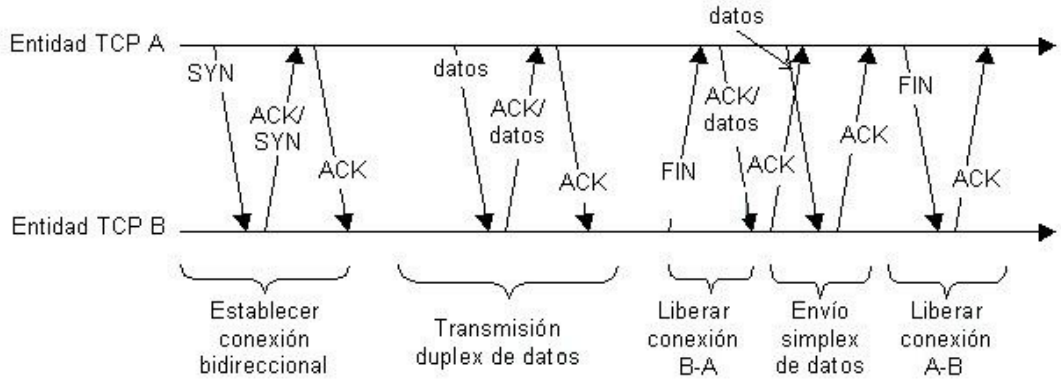

Figura 6. Ejemplo de conexión y desconexión en TCP.

3.6. Intercambio de paquetes con TCP

Cuando dos máquinas A y B, emplean el protocolo TCP para intercambiar mensajes de datos se pueden producir diferentes situaciones desde el punto de vista de los paquetes TCP encontrados durante el envío y recepción. Así, el estado de las máquinas A y B determina el tipo de paquetes TCP que se generan. El tipo de paquetes TCP viene definido por el tipo de flags que se activan en la cabecera del paquete TCP. Así, algunas de las situaciones más comunes son:

- La estación A envía paquetes a la estación B y la máquina B en el momento que A realiza la petición tiene el puerto sobre el que se solicita la petición no disponible.
- La estación A envía paquetes a la estación B y la máquina B en el momento que A realiza la petición está preparada para recibir.
- La estación A envía paquetes a la estación B, ambas se encuentran en distintas redes interconectadas y la máquina B en el momento que A realiza la petición se encuentra apagada.

El diagrama de flujo refleja el envío de los segmentos de datos TCP que se intercambian en esas comunicaciones A y B. Así, en la figura 7 se representan los diagramas de flujo que intercambian A y B en las tres situaciones anteriormente comentadas. En dicha figura se puede observar que flags se activan en las fases de establecimiento y de liberación de la conexión.

Además, en la figura 8, se puede observar un ejemplo de intercambio de datos entre las estaciones A y B en la fase de transferencia de datos. En esta figura se puede observar un envío sin errores de tres segmentos de datos entre A y B, y una confirmación para ese grupo de tres segmentos. El valor de nSeq indica el número de secuencia de cada paquete de transporte TCP enviado, el valor de nACK indica el número de confirmación de cada paquete, y el valor DATOS el contenido del campo de información o de datos del paquete o segmento TCP.

En este ejemplo, el campo de DATOS tiene un valor de 64bytes porque el tamaño máximo del segmento (tamaño máximo de MSS) es 64bytes. El MSS viene determinado por el valor de MTU y se puede definir como el máximo tamaño de datos de TCP.

El nSeq de un paquete n+1 se puede calcular sumando el nSeq del paquete anterior, paquete n, y del tamaño de los datos que éste último transporta. El valor nACK del paquete de confirmación que envía el receptor se puede calcular sumando el valor del nSeq del último paquete de datos que se recibe en el receptor y del tamaño de los datos que éste transporta.

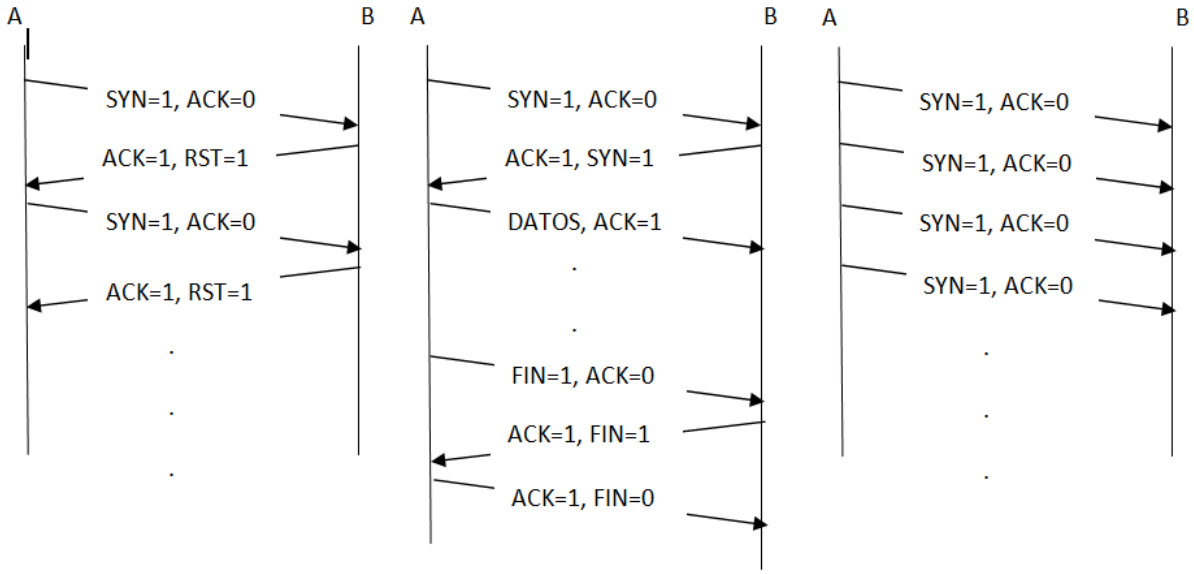

Figura 7. Diagramas de flujo TCP.

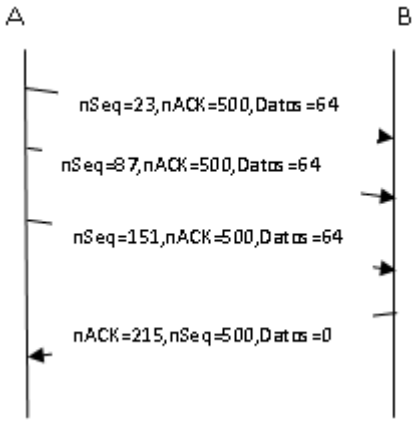

Figura 8. Diagramas de flujo TCP durante la fase de transferencia de información.

3.7. Cálculo del MTU en conexión

La RFC 1191 define un mecanismo para el cálculo del MTU de camino en una conexión TCP, válida sólo para este protocolo. Consiste en enviar todos los segmentos TCP con el bit 'don't fragment' de la cabecera IP a 1. Ello provoca la aparición de un mensaje ICMP 'Destination unreachable, fragment needed' ¾ donde los 2 últimos bytes de la cabecera ICMP contienen el MTU del siguiente salto, siempre que el enrutador que envía el paquete ICMP cumpla con la RFC1191. En caso contrario, ocurre exactamente lo mismo salvo que el dato del MTU del salto siguiente no aparece en la cabecera ICMP devuelta por el enrutador. El host

receptor de dicho mensaje ICMP aprende el MTU del salto siguiente reajustando su MSS para evitar la fragmentación.

Para ilustrar el procedimiento, veamos varios ejemplos y situaciones que pueden producirse. Supóngase que se dispone de dos estaciones A y B que desean intercambiar paquetes de datos TCP. Además, se sabe que A funciona como emisor y B como receptor y que el camino que define el enrutamiento de los paquetes TCP para ir de A hacia B es el que se indica en la figura 9. Por lo tanto, para ir desde A hasta B se tiene que pasar por 4 subredes: 10.1.2.0/24, 10.5.1.0/16, 10.5.2.0/16 y 10.1.3.0/24. Se sabe que la MTU de la red 10.1.2.0/24 es de 1500bytes y la MTU de la red 10.1.3.0/24 es de 1000bytes. Analicemos la segmentación que se produce en los siguientes casos, cuando A manda 2000 bytes de datos de transporte a B:

a) La MTU x>= min(1500,1000) e y>=min(1500,1000).

b) La MTU x< min(1500,1000) e y>=x.

c) La MTU x<min(1500,1000) e y<x.

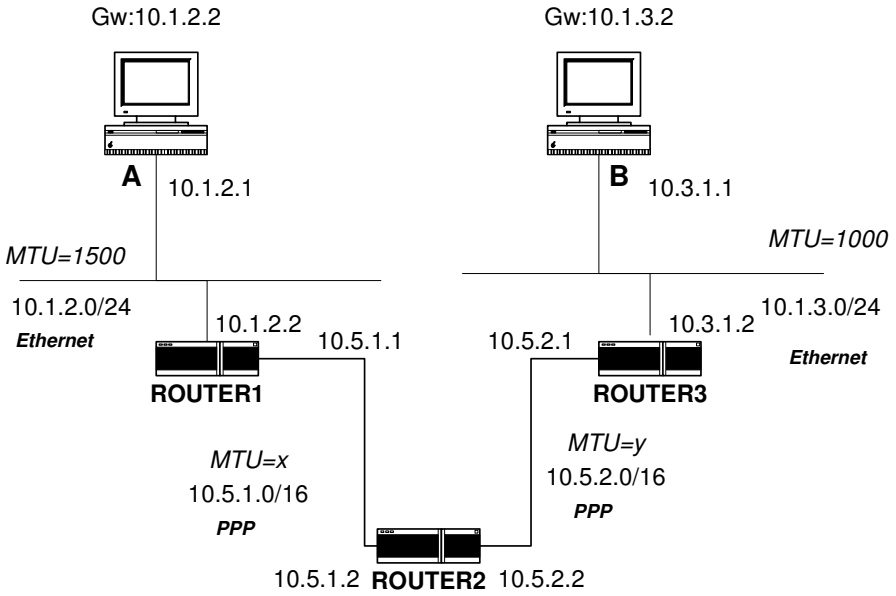

Figura 9. Diagramas de flujo TCP durante la fase de transferencia de información.

Caso a): A envía un paquete TCP, con el flag SYN=1 y con el bit don't fragment de la cabecera IP a 1, que negocia con B la MSS=1460 (ya que la MTU del segmento al que está conectado A es de 1500bytes). Y la máquina B responde con un paquete TCP, con flag ACK=1 y con valor MSS=960 (ya que la MTU del segmento al que está conectado B es de 1000bytes). Con estos dos paquetes de datos A y B negocian el tamaño de los paquetes para realizar la comunicación y el intercambio de información a la mínima MSS entre los dos extremos. Una vez negociada esa comunicación, A segmentará los 2000bytes de datos, construyendo paquetes TCP con 960 bytes de datos que es la mínima MSS de ambos extremos. Así, el resultado de paquetes a su paso por cada uno de los segmentos de red es:

IP:20 bytes+ TCP: 20 bytes+ DATOS: 960bytes
IP:20 bytes+ TCP: 20 bytes+ DATOS: 960bytes

IP:20 bytes+ TCP: 20 bytes+ DATOS: 80bytes

Caso b): A envía un paquete TCP, con el flag SYN=1 y con el bit don't fragment de la cabecera IP a 1, que negocia con B la MSS=1460 (ya que la MTU del segmento al que está conectado A es de 1500bytes). Y la máquina B responde con un paquete TCP, con flag ACK=1 y con valor MSS=960 (ya que la MTU del segmento al que está conectado B es de 1000bytes). Con estos dos paquetes de datos A y B negocian el tamaño de los paquetes para realizar la comunicación y el intercambio de información a la mínima MSS entre los dos extremos. Una vez negociada esa comunicación, A segmentará los 2000bytes de datos, construyendo paquetes TCP con 960 bytes de datos que es la mínima MSS de ambos extremos. Se pone a enviar el primer paquete IP con el bit don't fragment a 1 (***IP:20 bytes+ TCP: 20 bytes+ DATOS: 960bytes***) y el paquete cuando llega al ROUTER1 no puede ser encaminado hasta el ROUTER2, porque la MTU del segmento 10.5.1.0/16 tiene un valor menor que 1000. Por lo tanto, el interfaz 10.1.2.2 del ROUTER1 enviará un mensaje ***ICMP (3/4) 'Destination Unreachable Fragment Needed'*** hasta 10.1.2.1 indicando en los dos últimos bytes de su cabecera la ***MTU=x*** del segmento por el que no ha podido pasar. Cuando este ICMP llegue a A, A adecuará la segmentación a la MTU devuelta por ese ICMP. Así, la nueva segmentación sería:

IP:20 bytes+ TCP: 20 bytes+ DATOS: (x-40)bytes
IP:20 bytes+ TCP: 20 bytes+ DATOS: 2000-(x-40)bytes
Etc.

Caso c): Se repite el caso b) y la máquina A comienza enviando el primer paquete IP con el bit don't fragment a 1 (***IP:20 bytes+ TCP: 20 bytes+ DATOS: (x-40)bytes***), y el paquete cuando llega al ROUTER2 no puede ser encaminado hasta el ROUTER3, porque la MTU del segmento 10.5.2.0/16 tiene un valor menor que x. Por lo tanto, el interfaz 10.5.1.2 del ROUTER2 enviará un mensaje ***ICMP (3/4) 'Destination Unreachable Fragment Needed'*** hasta 10.1.2.1 indicando en los dos últimos bytes de su cabecera la ***MTU=y*** del segmento por el que no ha podido pasar. Cuando este ICMP llegue a A, A adecuará la segmentación a la MTU devuelta por ese ICMP. Así, la nueva segmentación sería:

IP:20 bytes+ TCP: 20 bytes+ DATOS: (y-40)bytes
IP:20 bytes+ TCP: 20 bytes+ DATOS: 2000-(y-40)bytes
Etc.

3.8.Herramientas empleadas en la práctica

***Ping*.** Este programa presente en la mayoría de S.O. que soportan la arquitectura TCP/IP utiliza un mensaje de petición de eco para enviar un datagrama a su destinatario y espera el retorno de un mensaje echo reply del destinatario. El contenido enviado (por defecto 32 bytes) son los caracteres ASCII alfabéticos. De este modo es capaz de evaluar tiempos de respuesta promedios. Dispone de varias opciones, entre las que cabe destacar la posibilidad de modificar el tamaño del paquete enviado, el registro de ruta, y el control del número de paquetes enviados. En MS Windows las opciones son:

```
ping [-n <x>] [-l <y>] [-i <z>] [-f] <ipaddr>
        -n <x> Envía <x> paquetes ICMP
        -l <y> Envía paquetes de longitud <y>
        -i <z> Limita la vida del paquete (TTL) a <z>
        -f Activa el bit Don't fragment <ipaddr> Dirección IP de
        destino
```

Route. Se entiende por ruta a la secuencia de subredes IP que debe atravesar un paquete para llegar a su destino. El paso del paquete de una subred a otra se efectúa a través de un router, que es un elemento que une dos o más redes trabajando a nivel de red. Un router decide a que red de las que tiene acceso directo reenviar un paquete que le llega por uno de sus interfaces en función de una tabla de rutas que posee. El comando *route* del S.O. permite alterar dicha tabla.

Realmente no sólo los routers tienen tabla de rutas, sino también cualquier equipo con TCP/IP. Un equipo emplea dicha tabla cuando no puede enviar un paquete directamente al destino por que no se trata de él mismo o porque no está en su propia subred.

Básicamente la tabla consta de una serie de entradas, cada una de las cuales indica la dirección IP de la próxima máquina (gateway) en la red a la que enviar un paquete recibido en base a la dirección destino de dicho paquete. La dirección destino de una entrada puede ser una IP de máquina o una subred, y para distinguir de que se trata se requiere también la máscara IP en la entrada. También suele existir incluir una entrada "por defecto" (default gateway) a la que se envían los paquetes cuya dirección destino no encaja con ninguna entrada anterior de la tabla.

Las opciones básicas de este comando en Windows son:

```
route add <IP_DESTINO> MASK <MASCARA_DESTINO> <IP_GATEWAY>
route delete <IP_DESTINO>
```

netstat. Comando del S.O. que muestra información sobre TCP/IP, como puede ser las conexiones TCP actuales (sockets) o la tabla de rutas. Para ver la tabla de rutas en MS Windows:

```
netstat -rn
```

Traceroute / Tracert. Estos comandos, para Unix y Windows respectivamente, permiten averiguar la ruta que debe seguir un paquete IP para llegar a su destino, dando la secuencia de IPs de los routers que atraviesa. Su sintaxis típica es:

```
tracert -d <IP_DESTINO>
```

La versión de MS Windows tiene un modo muy peculiar de averiguar dicha ruta, haciendo uso de paquetes ICMP con distinto TTL de partida. El mecanismo es sencillo, se envían paquetes "Echo" con valores TTL que se incrementan a partir de 1 en sucesivos intentos hasta que se obtenga un mensaje "Echo Reply" de respuesta en vez de un "TTL-exceeded in transit". Para evaluar los tiempos de ida y vuelta medios cada intento se repite 3 veces.

Rexec. Remote Shell es un servicio presente normalmente en un S.O. UNIX con TCP/IP que atiende el puerto TCP 512 en espera de peticiones de ejecución de comandos desde procesos remotos clientes. Utiliza

TCP, por lo que trabaja con conexión. Para las prácticas se dispondrá de un programa para MS Windows (rexec.exe) que actúa como cliente. En una sesión de *rexec.exe* se pide inicialmente un nombre de usuario y password en la máquina servidora, y tras introducir estos, se pueden ejecutar comandos UNIX en dicha máquina. Nos servirá para estudiar una conexión TCP. Dentro de una máquina UNIX, el cliente es un programa de línea de comandos con esta sintaxis básica:

```
rsh <IP_SERVIDOR> <COMANDO_A_EJECUTAR>
```

Udp.exe. Este sencillo programa para MS Windows nos permitirá enviar y recibir paquetes UDP, especificando también su contenido, a un número de puerto y una IP destinos especificados para comprobar el funcionamiento de este protocolo.

3.9. Ejercicios

Cuestión 1. UDP

a. Utilizar el programa *udp.exe* para realizar un envío de datos al puerto 7 (eco) o al puerto 13 (hora y día) del servidor Linux1. Para ello basta especificar la dirección IP y el puerto del servidor, colocar algún texto en la ventana y pulsar el botón "Envía UDP". Con el monitor de red analizar la secuencia de paquetes UDP que se desencadenan cuando se envía como datos una palabra corta. Utilizar para ello el filtro adecuado (direcciones y protocolos).

b. Probar de nuevo, pero enviando un texto mucho más grande (sobre 2Kbytes). Esto se puede hacer copiando parte de algún fichero de texto en la ventana de *udp.exe*. ¿Se produce fragmentación IP de los paquetes UDP? Estudiar las longitudes del paquete UDP y las de los paquetes IP que aparecen.

Cuestión 2. TCP y sus diagramas de secuencia de paquetes

a. Emplear el programa *rexec* para intentar ejecutar el comando 'ls –l' en la maquina con dirección 172.20.43.232. Utilizar para ello el usuario 'alumnos' y la clave 'alumnos'. Con el monitor de red, analizar y estudiar la secuencia de paquetes TCP intercambiados en el establecimiento de la conexión entre la máquina del alumno y la 172.20.43.232. Utilizar para ello el filtro adecuado (direcciones y protocolos).

a.1) Dibuja gráficamente, mediante un diagrama de envío y recepción de paquetes TCP, la fase de establecimiento de la conexión y la fase de liberación de la conexión. Indica en cada paquete los flags TCP que se activan, y los números de secuencia (nSeq, nACK, etc.)

a.2) Dibuja gráficamente, el intercambio de datos (fase de transferencia). Indica qué paquetes tienen el flag PUSH activo y trata de justificar por qué hay paquetes TCP de datos con ese flag activo y otros con él no activo.

b. Emplear el programa *ftp* para intentar ejecutar el comando 'ftp 172.20.43.X' siendo X la máquina de un compañero. Con el monitor de red, analizar y estudiar la secuencia de paquetes TCP intercambiados en el establecimiento de la conexión entre la máquina del alumno y la máquina del compañero 172.20.43.X. Utilizar para ello el filtro adecuado (direcciones y protocolos).

Cuestión 3. Fragmentación

a. Determinar el número de paquetes UDP que se generan (indicando el formato: cabeceras y datos), cuando el nivel de transporte envía 1000 bytes de datos en una red Ethernet con MTU de 500 bytes. Hacer lo mismo considerando que el nivel de transporte utilizado fuera TCP. Comparar el número y tipo de paquetes UDP y TCP, justificar la respuesta.

b. Determinar el número de paquetes UDP que se generan (indicando el formato: cabeceras y datos), cuando el nivel de transporte envía 2000 bytes de datos desde la máquina del alumno hasta la máquina 172.20.41.241.

c. Suponga que se transmite un fichero de 100Kbytes directamente sobre un protocolo de transporte UDP. Suponga, además, que no hay errores en la transmisión y que se atraviesa una red origen con una MTU de 1500 bytes y otra red con MTU de 500, ambas unidas por encaminadores IP. Determinar el número de bytes que llegarán a la red destino. (Se sabe que inicialmente la aplicación origen manda los paquetes sin necesidad de fragmentación adicional).

Cuestión 4. Cálculo de la MTU en conexión

a. En base a la topología que se muestra en la figura 10 y considerando que todos los equipos presentes en dicha topología cumplen la RFC 1191:

a.1) Determinar el número de paquetes TCP que se generan al mandar 1500 bytes de datos de transporte desde la máquina A hasta la máquina B. Indicar la MTU del camino. ¿Cuántos paquetes ICMP se generan?. Identifica sus direcciones IP origen y destino si fuera necesario, así como el tipo y código de dichos mensajes ICMP.

a.2) Determinar el número de paquetes TCP que se generan al mandar 1500 bytes de datos de transporte desde la máquina A hasta la máquina E. Indicar la MTU del camino. ¿Cuántos paquetes ICMP se generan?. Identifica sus direcciones IP origen y destino si fuera necesario, así como el tipo y código de dichos mensajes ICMP.

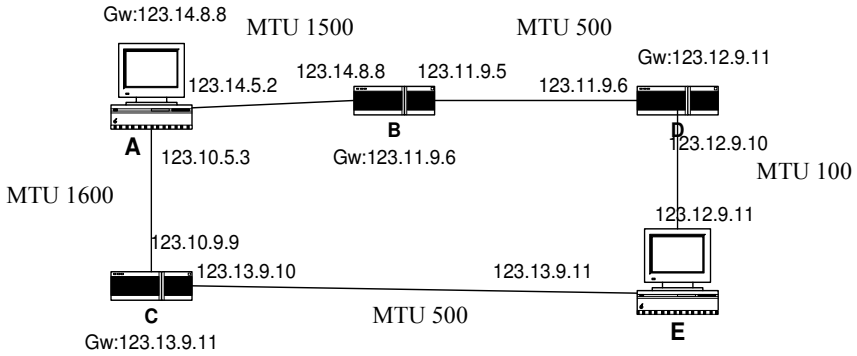

Figura 10.

b. En base a la topología de la Figura 11, determinar el conjunto de paquetes TCP que se generan, indicando el formato de los paquetes (tamaño de cabeceras y datos), cuando el nivel de transporte envía 2000 bytes de datos desde la máquina A hasta C. Para ello se facilitan los valores de MTU para las tres subredes de la topología y se sabe que se cumple la RFC 1191. Además, se sabe que las redes que atraviesan los datagramas IP para ir de A hasta C son: 10.1.2.0/24, 10.1.3.0/24 y 10.1.4.0/24. Indicar la MTU del camino. ¿Cuántos paquetes ICMP se generan?. Identifica sus direcciones IP origen y destino si fuera necesario, así como el tipo y código de dichos mensajes ICMP.

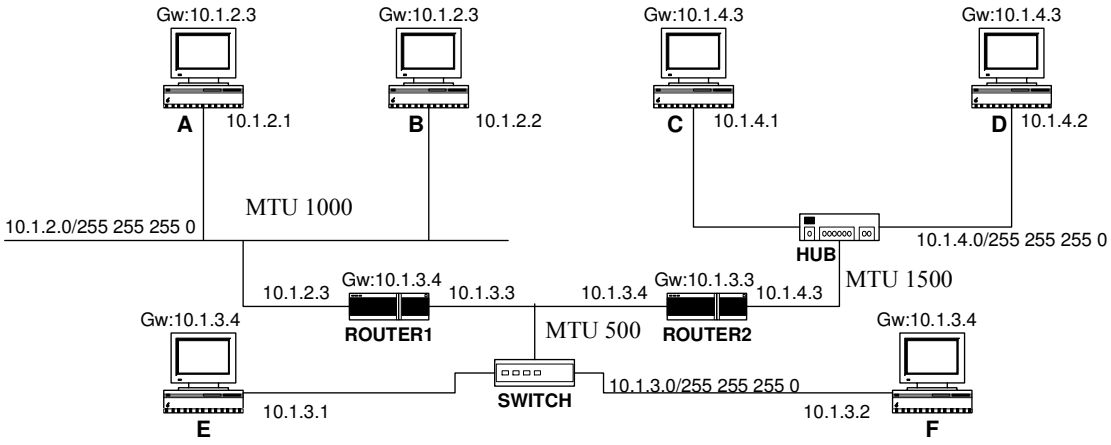

Figura 11.

Cuestión 5. Control de flujo en TCP

a. Emplear el programa *telnet* para intentar ejecutar el comando 'ls –l' en la maquina con dirección 172.20.43.232. Utilizar para ello el usuario 'alumnos' y la clave 'alumnos'. Con el monitor de red, analizar y estudiar la secuencia de paquetes TCP intercambiados en el establecimiento de la conexión entre la máquina del alumno y la 172.20.43.232. Utilizar para ello el filtro adecuado (direcciones y protocolos).

a.1) Dibuja gráficamente, mediante un diagrama de envío y recepción de paquetes TCP, la fase de establecimiento de la conexión y la fase de liberación de la conexión. Indica en cada paquete los flags TCP que se activan, y los números de secuencia (nSeq, nACK), así como el tamaño de datos que transporta cada paquete.

a.2) Dibuja gráficamente, el intercambio de datos (fase de transferencia). Indica qué paquetes tienen el flag PUSH activo y trata de justificar por qué hay paquetes TCP de datos con ese flag activo y otros con él no activo.

b. La capa de transporte de una máquina A desea establecer una conexión con la capa de transporte de una máquina B, separada de A por dos segmentos de red. La máquina A maneja una ventana de 512 bytes y un tamaño máximo de MSS de 128 bytes. Sin embargo, la máquina B trabaja con una ventana de 256bytes y un tamaño máximo de MSS de 64bytes.

b.1) ¿Es posible que A y B manejen diferentes ventanas y MSS? En caso afirmativo, dibuja gráficamente, mediante un diagrama de envío y recepción de paquetes TCP, la fase de establecimiento de la conexión. Indica en cada paquete los flags TCP que se activan, y los números de secuencia (nSeq, nACK), el tamaño de datos que transporta cada paquete, el valor de MSS y el valor de la ventana. Supóngase que la máquina A utiliza como número de secuencia inicial, el valor 32, y la máquina B, el valor 1024.

a.2) Posteriormente, la máquina A envía sin errores a la máquina B, un grupo de cuatro paquetes de datos de transporte. Y además se sabe, que la máquina B confirma esos cuatro paquetes de golpe con un único paquete de confirmación. Dibuja gráficamente, el intercambio de datos (fase de transferencia). Indica en cada paquete los flags TCP que se activan, y los números de secuencia (nSeq, nACK), el tamaño de datos que transporta cada paquete, el valor de la ventana y el valor de MSS.