Universitat d'Alacant
Universidad de Alicante

Dep. de Física, Enginyeria de Sistemas i Teoria del Senyal
Dpto. de Física, Ingeniería de Sistemas y Teoría de la Señal

REDES DE COMPUTADORES

Grado en Ingeniería Informática

Curso académico 2016/2017

Práctica 3. Protocolos de Transporte TCP y UDP

CONTENIDOS

1. Objetivos básicos

- Conocer los protocolos de transporte de la arquitectura TCP/IP: TCP y UDP. Estudiar sus principales características y utilidades.
- Conocer el funcionamiento del recurso proporcionado por la capa de transporte a la aplicación: los **sockets**, así como el proceso de conexión-datos-desconexión de TCP.
- Evaluar el rendimiento real a nivel de transporte comparando la cadencia eficaz a este nivel frente a la velocidad de transmisión de enlace de la red.

2. Protocolos de transporte en TCP/IP

Los protocolos de transporte de la arquitectura TCP/IP son el TCP (Protocolo de Control de la Transmisión - *Transmision Control Protocol*) y UDP (Protocolo de Datagramas de Usuario - *User Datagram Protocol*). Es decir, TCP/IP ofrece dos opciones de protocolo de transporte al nivel superior, el de aplicación. Ambos protocolos trabajan de forma muy diferente, y están orientados a distintos usos. No hay que pensar que TCP es mejor que UDP en general o viceversa. Así, existen aplicaciones que utilizan TCP y otras que usan UDP (ver figura siguiente).

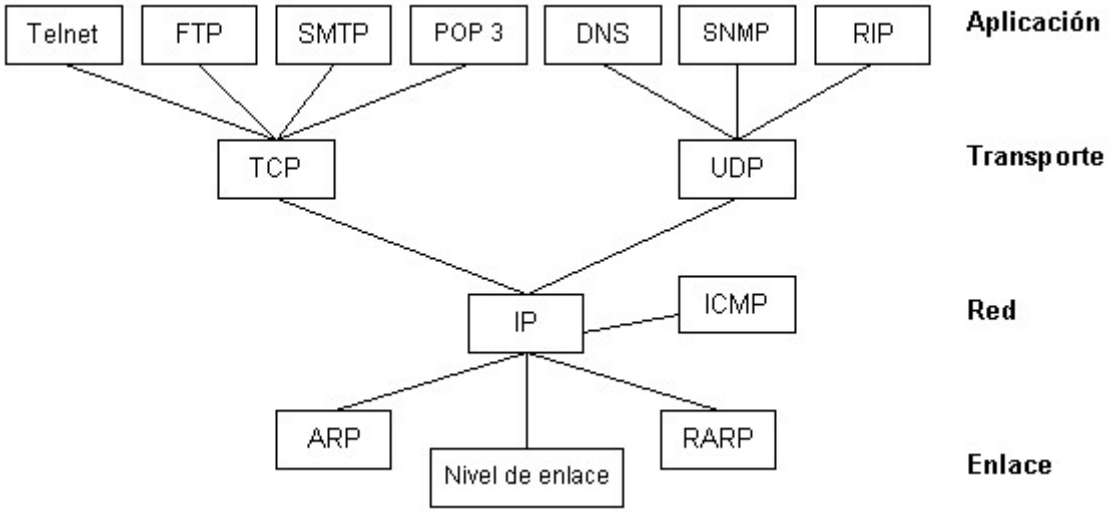


Figura 1. Conjunto de protocolos en la arquitectura de red TCP/IP.

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 de la siguiente forma:

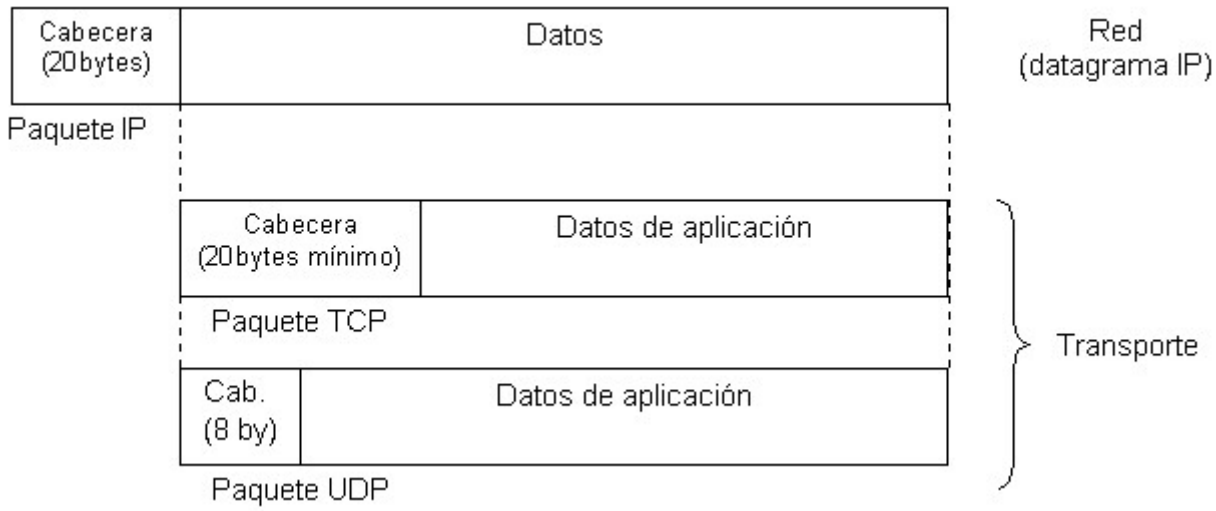


Figura 2. Encapsulación de paquetes de la capa de transporte en datagramas IP.

2.1 Puertos y *sockets*

A la vez que IP trabaja con direcciones de máquina para identificar los interfaces de red origen y destino de un paquete, a nivel de transporte TCP y UDP emplean valores de 16 bits conocidos como puertos, que son unos identificadores de buffers en las máquinas origen y destino respectivamente. 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. Posteriormente 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, como ilustra el ejemplo siguiente.

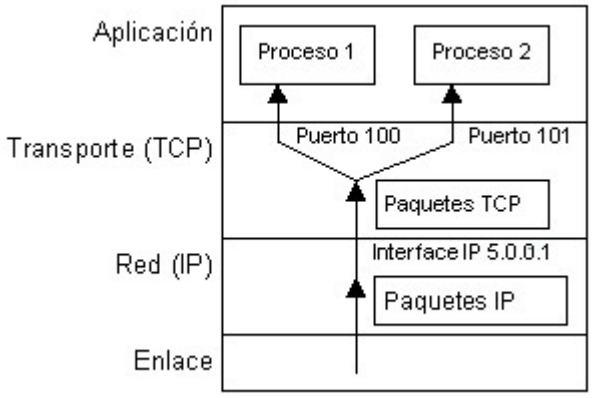


Figura 3. Multiplexación de conexiones en la capa de transporte de TCP/IP.

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 denomina *socket*. Los *sockets* permiten la transmisión bidireccional de datos en modo full-duplex. Así lo ilustra el siguiente ejemplo.

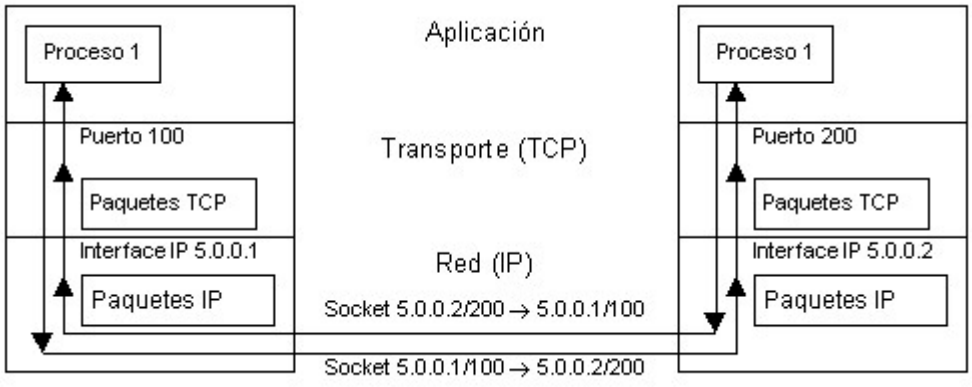


Figura 4. Intercambio de información entre aplicaciones empleando la capa de transporte de TCP/IP

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 una conexión (*socket*) con un servidor. Un servidor normalmente puede atender a varios clientes simultáneamente. Un cliente emplea un número de puerto generalmente en el rango desde 1024 hasta 5000, y dentro de una misma máquina el valor del puerto cliente se va incrementando conforme se establecen nuevas conexiones. En cambio, para los procesos servidores se han definido unos puertos concretos de forma que sean conocidos de antemano por los clientes para poder establecer las conexiones. Ejemplos de números de puertos reservados a servidores son:

- Servidor Eco (devuelve una copia de la información que se le envía al servidor): puerto UDP 7.
- Servidor Fecha y hora (devuelve la hora y fecha del sistema): puerto UDP 13.
- Servidor FTP (protocolo de transferencia de ficheros): puerto TCP 21.
- Servidor Telnet (protocolo de terminal remoto): puerto TCP 23.
- Servidor SMTP (protocolo simple de trasferencia de correo): puerto TCP 25.
- Servidor Web: puerto TCP 80.
- Servidor POP3 (protocolo de oficina de correos para acceso a un buzón de correos): puerto TCP 110.
- Servidor IMAP (protocolo de acceso de mensajes de Internet, un protocolo alternativo a POP3 para acceso a un buzón de correo) : puerto TCP 143.

Mientras que las direcciones IP van en la cabecera de un datagrama IP, los números de puerto se especifican dentro de las cabeceras de los paquetes TCP o UDP.

2.2 Protocolo TCP

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 del 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.

Un paquete TCP tiene el siguiente formato, donde destaca la complejidad de la cabecera:

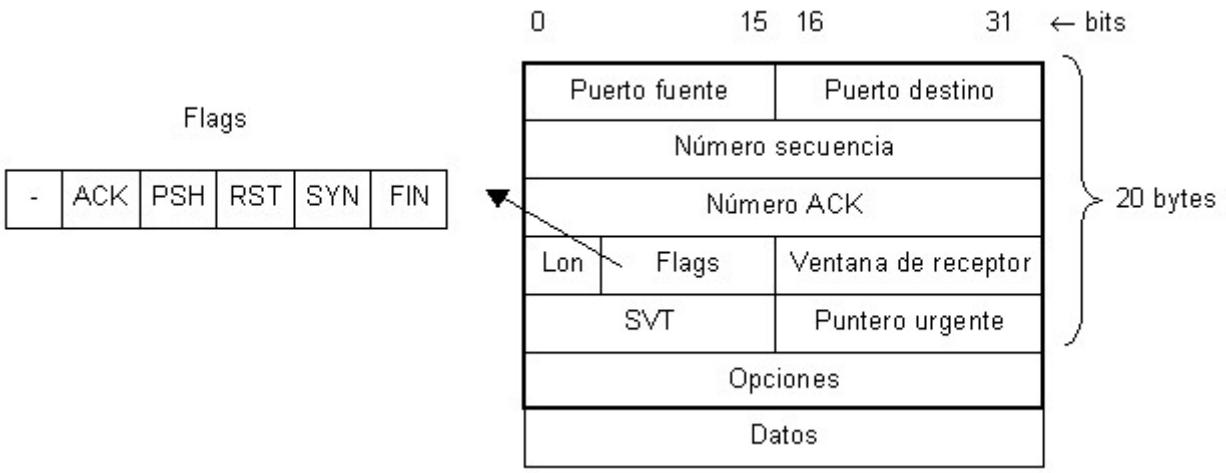


Figura 5. Formato de la cabecera del protocolo TCP.

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

- **Puerto fuente y puerto destino**. Valores de 16 bits correspondientes a los identificadores de los puertos de nivel de transporte.
- **Número de secuencia**. Número de secuencia de numeración del primer byte del campo de datos del paquete.
- **Número de ACK**. Número de la siguiente secuencia de numeración de los bytes del campo de datos 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. Cuando toma el valor 1 indica que el número de ACK es válido y debe interpretarse, es decir, el paquete tiene información de asentimiento.

  PSH (*Push*). Cuando toma el valor 1 indica que la capa de transporte debe pasar los datos a la capa de aplicación sin esperar a recibir más datos.

RST *R*eset*).* Indica un rechazo 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 (*Finish*). 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 recibir en su buffer de recepción. Si vale 0 indica que no puede recibir datos (aunque sí 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 inicio del campo de datos del paquete TCP que contiene información urgente y se ha de enviar directamente a la capa de aplicación.
- **Opciones**. Permite campos adicionales.

Seguidamente 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.

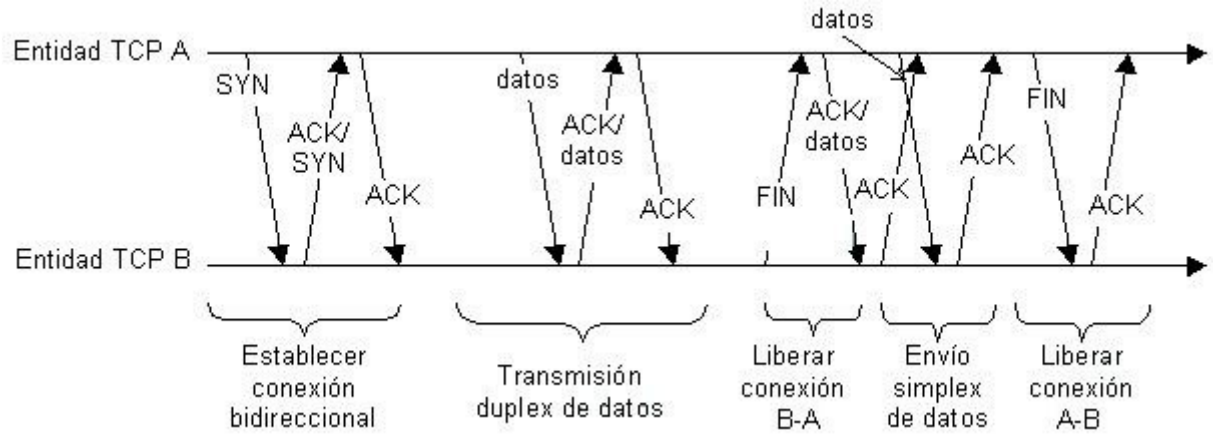


Figura 6. Secuencia de intercambio de paquetes en una conexión TCP.

En el establecimiento de conexión se realiza la gestión de ciertos parámetros de la comunicación empleando el campo de opciones de la cabecera TCP. Uno de estos parámetros es la cantidad máxima de datos en cada paquete TCP o MSS (*Maximun segment size* o tamaño máximo de segmento). Ya se estudió en la práctica 2 de la asignatura que la fragmentación de información del nivel superior en varios datagramas provoca congestión en los routers que deben realizar la función de encaminamiento, ya que estos necesitarán encaminar más paquetes.

TCP intenta gestionar una comunicación fiable y fluida, por lo que negocia entre los dos extremos de la comunicación qué tamaño máximo de datos se van a intercambiar en los paquetes TCP. Este valor depende del valor de MTU que exista en cada extremo de la comunicación.

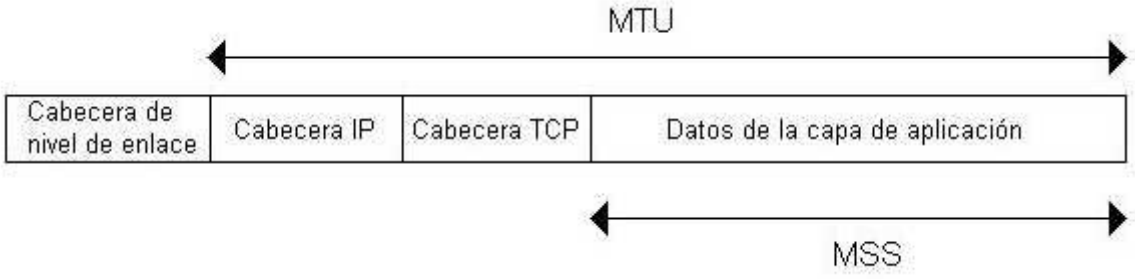


Figura 7. MSS de una conexión TCP.

Del diagrama anterior puede comprobarse cómo cada extremo de la comunicación puede calcular su valor de MSS que puede emplear para que no se produzca fragmentación.

MSS = MTU - 20 bytes (longitud cabecera TCP) - 20 bytes (longitud cabecera IP).

Este valor de MSS es informado por cada extremo de la comunicación durante el establecimiento de la conexión introduciéndolo en el campo opciones de la cabecera TCP de los paquetes SYN. Una vez establecida la conexión, se empleará el menor de los valores de MSS intercambiados, para evitar que la capa IP tenga que fragmentar la información.

Sin embargo, es posible que intercambiar el valor de MSS entre los dos extremos de la comunicación no sea suficiente para evitar que los datagramas IP necesiten ser fragmentados en su camino al destino. Para evitar este problema se introdujo la norma RFC 1191, cuyo objetivo es evitar que los datagramas IP que contienen paquetes de una conexión TCP sean fragmentados en la red.

Cuando dos equipos remotos establecen una conexión TCP, determinan como valor de MSS a emplear el menor de los de las dos estaciones. Sin embargo, es posible que entre las dos estaciones exista una red intermedia donde el valor de MSS sea menor, por lo que los paquetes TCP serán fragmentados por IP y se producirá un descenso en la fluidez de la conexión. Para ello, la norma RFC 1191 establece que los dos extremos de la comunicación intercambien paquetes TCP en los que la cabecera IP esté activo el bit *don't fragment* y que sean capaces de interpretar paquetes ICMP *fragmentation needed and the bit don't fragment was set*. De esta forma, cuando una estación envía un paquete TCP que tiene que ser fragmentado al atravesar una red intermedia, el router que tiene que realizar la fragmentación no puede hacerlo debido al bit *don't fragment* activo y envía al origen del paquete TCP un mensaje ICMP d*estination unreachable*. Este mensaje ICMP es el de código *fragmentation needed and the bit don't fragment was se*t. En la cabecera ICMP de este mensaje se indica cuál es el valor del MTU de la red que necesita fragmentar el paquete TCP, indicándolo en el campo *Next Hop MTU*. Con este valor de MTU intermedio, la estación origen determina el nuevo valor de MSS de la conexión y reenvía el paquete TCP que no llegó al destino, introduciendo en el mismo una cantidad de datos menor o igual al MSS.

Con este procedimiento se consigue ajustar de forma dinámica la cantidad de datos introducida en cada paquete TCP para evitar que los paquetes tengan que ser fragmentados y afecten al rendimiento de la comunicación. Para que la norma RFC 1191 sea implementada se precisa que los equipos remotos activen el bit *don't fragment* en la cabecera IP de los paquetes TCP enviados y que interpreten los mensajes ICMP *fragmentation needed and the bit don't fragment was set*, además de que los routers intermedios entre los dos extremos sean capaces de enviar este mensaje ICMP.

Control del flujo con TCP

El control del flujo en TCP se fundamenta en un protocolo de ventana deslizante. En TCP el elemento emisor de datos dispone de una serie de secuencias que identifican la cantidad de

bytes transmitidos y el elemento receptor confirma (ACK) qué secuencias de datos espera recibir.

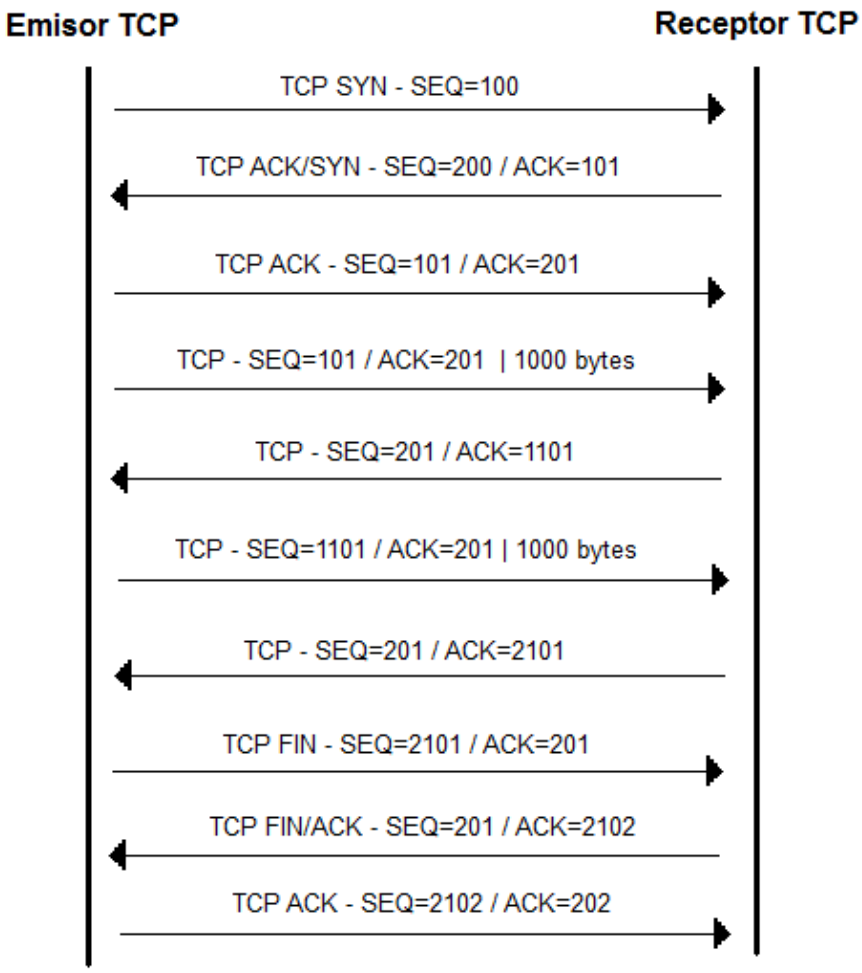
La ventana deslizante permite transmitir información por parte del emisor sin que lleguen confirmaciones (ACK) por parte del receptor, hasta un valor máximo de bytes (**ventana del emisor TCP**). Por otro lado, el receptor puede recibir una cantidad máxima de datos por parte del emisor sin enviar ACK (**ventana del receptor**).

El envío de ACKs por parte del receptor TCP es variable y depende del retardo en la red de comunicaciones, congestión del receptor, etc. **En general, el receptor TCP enviará un paquete ACK por cada paquete de datos recibido,** aunque en ciertas condiciones el receptor podrá enviar un ACK confirmando varios bloques de datos recibidos.

Los valores de números de secuencia y ACK de los paquetes TCP permiten coordinar la cantidad de información intercambiada entre un emisor y un receptor. Los números de secuencia de los paquetes transmitidos por el emisor se incrementan en la **cantidad de datos** que transporta el paquete (**los paquetes TCP de control SYN, RST o FIN se consideran de tamaño 1 byte**).

El valor de la secuencia ACK asociada a un paquete de datos se corresponde con el **valor de secuencia de los datos + cantidad de datos del paquete**.

El siguiente ejemplo ilustra los valores de secuencia y ACK de paquetes TCP en el intercambio de 2000 bytes de datos en una red con MSS de 1000 bytes.



En este esquema, si el receptor TCP recibe errores en un paquete no envía el ACK correspondiente. En el emisor TCP se usa un tiempo máximo de espera de ACK para cada

paquete de datos transmitido, transcurrido el cual sin recibir ACK se reenvía de nuevo el paquete.

Un receptor TCP puede recibir paquetes de datos con secuencias no esperadas pero no enviará el ACK correspondiente. Estos datos se almacenan en la ventana del receptor siempre que tenga espacio disponible. Cuando se recibe la secuencia esperada, se vacía la información consecutiva que se haya almacenado en la ventana del receptor.

2.3 Protocolo UDP

Las características principales de este protocolo son:

- Sin conexión. No emplea ninguna sincronización origen – 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 en varios paquetes 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.

El formato de un paquete UDP es el siguiente:

Figura 8 Formato de la cabecera del protocolo 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 son:

- Transmisión de datos en LANs fiables: Protocolo TFTP, 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, como es el caso del protocolo SNMP, protocolo simple de gestión de red.
- Operaciones consulta/respuesta breve. Intercambio de información con poco volumen de datos, como es el caso del protocolo DNS de asociación de nombre de dominio con direcciones IP.

3. Evaluación del rendimiento a nivel de transporte.

3.1 Protocolo de nivel de enlace PPP

El protocolo PPP (Protocolo Punto a Punto - *Point to Point Protocol*), es un protocolo de nivel de enlace especificado para el acceso a redes TCP/IP. Está normalizado por el IETF (Fuerza de Ingeniería en Internet - *Internet Engineering Task Force*) dependiente del IAB (Comité de Arquitectura de Internet - *Internet Architecture Board*) en el documento RFC 1661.

PPP proporciona sobre una línea punto a punto detección de errores, verificación de autenticación en el enlace, reconocimiento de varios protocolos de nivel de red (IP, IPX, OSI CLNP, etc.) y negociación de direcciones de red IP, lo que le convierte en el protocolo de nivel de enlace empleado por los proveedores de acceso a Internet a través de líneas telefónicas.

El protocolo PPP emplea para delimitar cada trama la secuencia de bits 01111110 al principio y final de la misma. En la cabecera del protocolo se introducen los campos de dirección, control y protocolo. Los campos dirección y control están a un valor fijo y sirven para mantener la compatibilidad con otros protocolos de nivel de enlace existentes (por ejemplo, HDLC). En el campo protocolo se especifica el tipo de paquete que hay en el cuerpo de datos del paquete PPP: un paquete IP, IPX, etc. Al final de los datos aparece una secuencia de verificación de la trama (un código CRC de 16 o 32 bits) y el delimitador de fin de la misma.

8 bits	8 bits	8 bits	8 o 16 bits	Longitud variable (1500 bytes por defecto)	16 o 32 bits	8 bits
01111110	11111111	00000011	Protocolo	Datos del protocolo de nivel de red o de LCP o NCP	FCS	01111110
Delimitador	Dirección	Control	NCP, LCP, IP, IPX			Delimitador

Figura 9. Formato del paquete de nivel de enlace PPP.

De esta forma, cada paquete PPP incorpora una cantidad total de 10 bytes redundantes en la cabecera y la cola.

Además, hay que tener en cuenta qué tipo de interfaz se emplea en la transmisión de los paquetes PPP. En el laboratorio, existe una conexión PPP entre el router CISCO 2513 y el equipo Linux 1 (10.3.7.0). Esta comunicación se realiza empleando un interfaz serie asíncrono, en el que por cada 8 bits que se transmiten por la línea se añade un bit de inicio y otro de stop para delimitar el conjunto de 8 bits.

Figura 10. Formato de transmisión en una línea serie asíncrona.

3.2 Cadencia eficaz de una comunicación

Frente a la velocidad de transmisión (Vt) que es capaz de ofrecer un medio físico en bits por segundo se puede definir la cadencia eficaz (Ce) como la velocidad real en bits por segundo de envío de los datos, y que es inferior a la Vt en función de la carga adicional en la red que supongan los protocolos de la arquitectura TCP/IP. Es decir, hay que tener en cuenta que al transmitir información en una red se añade información redundante a los datos transmitidos, es decir, se añaden las cabeceras de los protocolos de la arquitectura de red.

Para un protocolo de comunicación en el que no se producen errores ni reenvíos de paquetes, se puede obtener la cadencia eficaz como:

Ce = n / To = (bps)

Donde **n** es el número de bits de datos transmitidos y **To** es el tiempo en segundos que tarda el paquete en ser transmitido por el canal. Este tiempo **To** puede aproximarse como el tiempo de ida de un paquete al destino, o lo que es lo mismo, la mitad del tiempo de ida y vuelta (desde el envío de un paquete hasta recibir su confirmación) para facilitar su medición.

Sin embargo, ¿ qué se entiende por datos ? Para cada nivel de la arquitectura de red se considera como datos la información procedente del nivel superior. De esta forma, la cadencia eficaz es diferente para cada nivel de la arquitectura de red. Si el formato de un paquete genérico en la arquitectura TCP/IP tiene la siguiente forma:

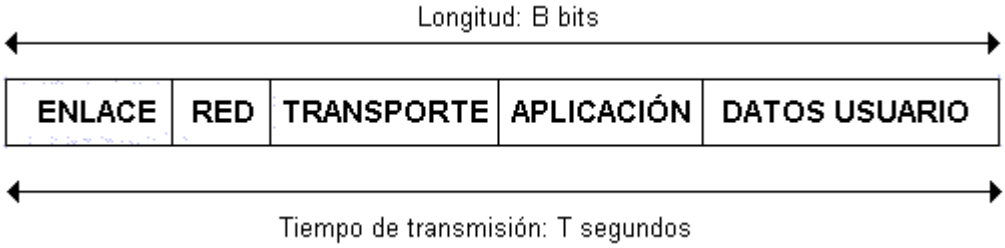


Figura 11. Formato genérico de un paquete en la arquitectura TCP/IP.

La velocidad de transmisión en el medio físico donde se realiza el envío de este paquete será:

Vt = B / T bps

El cálculo de la cadencia eficaz para cada nivel de la arquitectura de red será:

Ce_enlace = (B - longitud cabecera de enlace) / T bps
Ce_red = (B - lon. cab. enlace - long. cab. red) / T bps
Ce_transporte = (B - long. cab. enlace - long. cab. red - long. cab. transporte) / T bps
Ce_aplicación = (B - long. cab. enlace - long. cab. red - long. cab. transporte - log. cab. aplicación) / T bps

De estas expresiones se deduce claramente que Vt > Ce_enlace > Ce_red > Ce_transporte > Ce_aplicación. Es decir, que los protocolos de cualquier nivel en la arquitectura de red presentan una velocidad de transferencia efectiva para datos menor que la Vt del medio físico, y que además será menor cuanto mayor sea el nivel del protocolo dentro de la arquitectura.

4. Herramientas utilizadas

- **Rexec**. Rexec es el nombre de un servicio presente en sistemas operativos UNIX, sistemas que implementan en su diseño la arquitectura TCP/IP. Su objetivo es permitir la ejecución de comandos UNIX desde equipos remotos clientes. Para ello, el servidor

que atiende las peticiones de ejecución emplea el puerto 512 del protocolo TCP. Los clientes que desean ejecutar un comando Unix de forma remota establecerán una conexión TCP a este número de puerto y enviarán como datos el comando a ejecutar con un formato determinado. El servidor contesta enviando a través del socket el resultado que la ejecución del comando produce en la salida estándar.

Para las prácticas se dispondrá de un programa que actúa como cliente del servicio rexec (rexec.exe), desarrollado en Visual Basic para Windows 98 por el profesor **Luis Miguel Crespo Martínez**. La aplicación solicita la dirección IP del servidor rexec donde se desea ejecutar el comando, un nombre de usuario y contraseña en el servidor y el comando a ejecutar. El botón '**Ejecutar**' enviará el comando al servidor y nos mostrará el resultado en el panel inferior de la aplicación. En una sesión de rexec el cliente envía una cadena de datos con el nombre de usuario, contraseña y comando a ejecutar. Si el usuario tiene permiso de ejecución del comando solicitado, el servidor ejecuta la orden y el resultado de la salida estándar es enviado como datos al cliente. Este funcionamiento nos servirá para estudiar una conexión TCP.

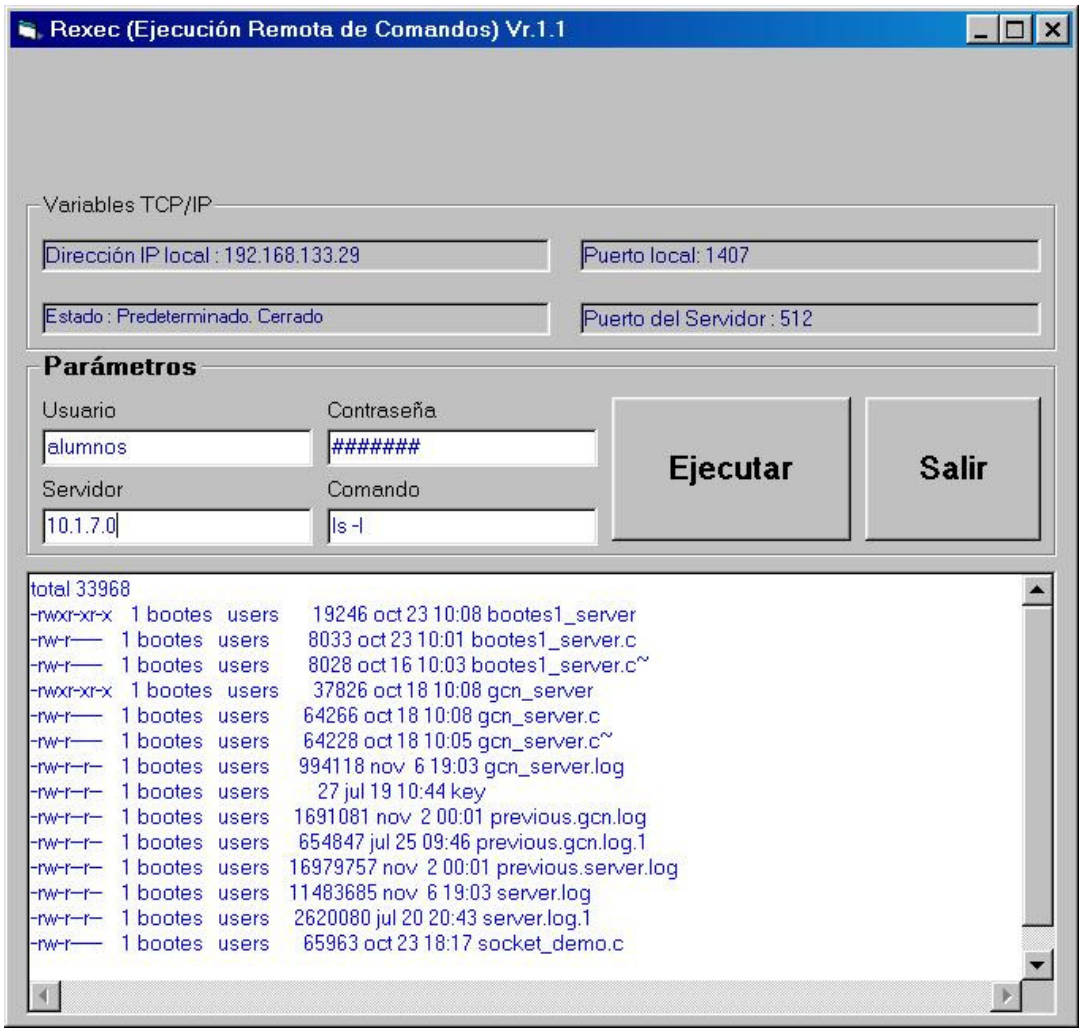


Figura 12. Aplicación Rexec.exe.

En una máquina con sistema operativo UNIX existe el comando **rsh** para ejecutar comandos en equipos remotos. La sintaxis de empleo básica de este comando es:

rsh <IP_SERVIDOR> <COMANDO_A_EJECUTAR>

- **Udp**. Esta aplicación para Windows 98, desarrollada también en Visual Basic por Luis Miguel Crespo Martínez, permite enviar y recibir paquetes UDP. Para ello se indica en el panel inferior el texto que se incorpora en el paquete UDP a enviar, el número de puerto en el servidor donde se enviarán los paquetes (RemotePort) y la dirección IP del destinatario del paquete (RemoteHost). Pulsando el botón '**Envía UDP**' se enviará al destinatario el paquete UDP. Dependiendo de a qué número de puerto se envíen los datos, el destinatario podrá contestar con cierta información. Si es así, esta información de respuesta se visualiza en el panel inferior.

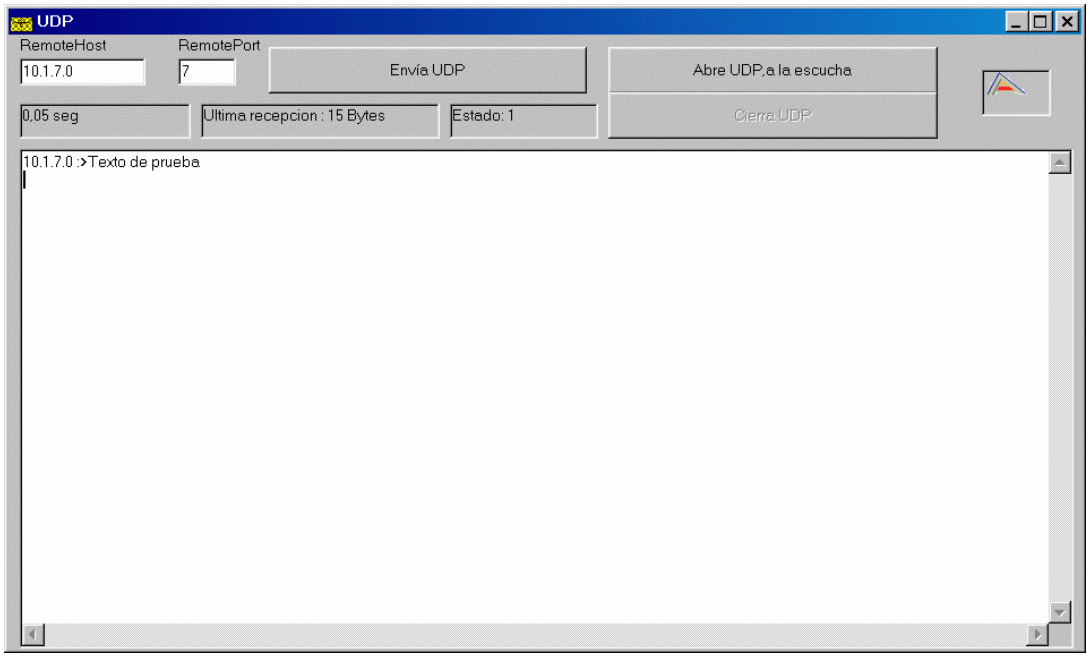


Figura 13. Aplicación Udp.exe.

5. Cuestiones a realizar

5.1 PROTOCOLO TCP

5.1.1 Utiliza el programa **Rexec** para ejecutar comandos UNIX (ls -a, pwd, who, etc.) en la máquina 172.20.43.232. Utiliza el usuario "**alumnos**" y password "**alumnos**" para ello.

- Con el monitor de red, analizar y estudiar la secuencia de paquetes TCP que se desencadenan. Comprobar que las secuencias de conexión y desconexión TCP que aparecen son las comentadas en el apartado 2.2 del documento de la práctica. Determina cuál es el valor de MSS que se negocia en los paquetes TCP SYN.
- Una vez establecida la conexión, el alumno detectará que el servidor 172.20.43.232 envía un paquete TCP SYN al PC del alumno dirigido al puerto **Authentication service** 113 (0071 en hexadecimal), y el PC del alumno contesta con un paquete TCP RST. Estos dos paquetes no deben ser considerados por el alumno y se generan debido a que el servidor trata de autenticar al cliente, y éste no soporta la autenticación, rechazando la conexión.
- Comprueba también los valores de puertos utilizados (número asignado de puerto cliente y número de puerto utilizado por el servidor), como varían los números de secuencia de byte y de ACK, y los flags activados en las cabeceras TCP.

- Analiza los valores de las ventanas de receptor anunciadas. ¿Son iguales en el cliente y en el servidor? ¿Cuál es más grande y por qué piensas que son diferentes?

5.1.2 Utiliza el programa **Rexec** para ejecutar el comando 'cat RC-GII.txt' en el servidor 172.20.43.232.

- La información recibida es de varios miles de bytes y se enviará en paquetes TCP de gran tamaño. ¿ Fragmenta el protocolo IP estos paquetes TCP grandes ? ¿ Por qué no ?

5.1.3 Utiliza el programa **Rexec** para ejecutar el comando 'cat RC-GII.txt' en el servidor 10.3.7.0.

- ¿ Qué valor de MSS se negocia entre los extremos de la comunicación ?
- Comparar la diferencia en el tamaño de los paquetes TCP en este caso respecto del anterior (5.1.2).

5.1.4 Norma RFC 1191

Descarga desde la carpeta 'Prácticas' en la sección de materiales de campus virtual el fichero 'rfc1191.txt' en el directorio raíz C:\ de tu PC.

En una ventana de MS-DOS y dentro del directorio raíz emplear el programa ftp para enviar el fichero rfc1191.txt al servidor 172.20.41.241. Previamente, inicia el monitor de red para capturar toda la secuencia de paquetes intercambiados con el PC del alumno.

Para realizar la transferencia del archivo se empleará la siguiente secuencia de comandos:

C:\route delete 172.20.41.241

C:\ftp 172.20.41.241
Conectado a 172.20.41.241.
220 Linux1.disc.ua.es FTP server (Version wu-2.4.2-academ[BETA-18-VR12](1) Wed Jan 27 22:19:46 CST 1999) ready.
Usuario (172.20.41.241:(none)):alumnos
331 Password required for alumnos.
Contraseña:alumnos
230 User alumnos logged in.
ftp> bin
200 Type set to I
ftp> put rfc1191.txt
200 PORT command successful.
150 Opening BINARY mode data connection for rfc1191.txt
226 Transfer complete.
ftp: 48730 bytes sent in 24.28 segundos 2.01 KB/s
ftp> quit
221-You have transferred 48730 bytes in 1 files.
221-Total traffic for this session was 49154 bytes in 1 transfers.
221-Thank you for using the FTP service on Linux1.disc.ua.es.
221 Goodbye.

- Determina con el monitor de red qué valor de MSS se ha negociado en la conexión TCP.

- ¿ Aparecen paquetes ICMP fragmentation needed and the bit don't fragment was set ? ¿ Quién envía el mensaje ICMP de error ?
- ¿ Como afecta este mensaje ICMP al tamaño de los paquetes TCP intercambiados entre tu PC y el servidor 172.20.41.241 ?
- ¿ Reenvía tu PC algún paquete TCP al servidor ?
- ¿ Fragmenta IP algún paquete TCP ?

5.1.5 Intentar acceder al servidor Web de la máquina 172.20.43.232 empleando el cliente navegador de tu PC con la dirección http://172.20.43.232/. Determinar qué secuencia de paquetes se intercambian en la conexión TCP y por qué.

5.2 PROTOCOLO UDP

5.2.1 Utiliza el programa **Udp** para realizar un envío de datos al puerto 7 (eco) o al puerto 13 (hora y día) del servidor 172.20.43.232. Para ello basta especificar la dirección IP y el puerto del servidor, colocar algún texto en el panel inferior de la aplicación 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 un texto de menos de 100 caracteres.
- Comparar los valores de los campos de tamaño de las cabeceras IP y UDP.

5.2.2 Realizar el mismo procedimiento que en el apartado anterior pero enviando un texto mucho más grande (sobre 2000 bytes) al puerto UDP 7. Para ello puede copiarse parte de algún fichero de texto en el panel inferior de la aplicación **Udp**

- ¿Se produce fragmentación IP de los paquetes UDP?.
- Analiza el valor del campo longitud de la cabecera UDP y del campo longitud total de la cabecera IP en los paquetes enviados y recibidos.

5.2.3 Realizar un nuevo envío de un texto corto a los puertos UDP 7 y 13, pero dirigido a la dirección de broadcast de la red local.

- ¿ Qué estaciones contestan en cada caso ?

5.2.4 Intenta enviar un texto al puerto 80 de la máquina 172.20.43.232 empleando el programa **Udp**. Determina la secuencia de paquetes que se intercambian entre tu equipo y el servidor 172.20.43.232.

5.3 RENDIMIENTO DE UNA COMUNICACIÓN

5.3.1 Determina de forma práctica las velocidades de transmisión en la subred local y en el enlace PPP entre el router CISCO 2513 y el Linux 1 (10.3.7.0) mediante el envío de tramas de un tamaño determinado con la aplicación **ping** y el tiempo que se obtiene de ida y vuelta. Para obtener medidas más exactas es conveniente enviar tramas de enlace que llenen el campo de datos hasta el MTU. Debe recordarse que PPP trabaja sobre una línea serie asíncrona con el formato de trama especificado en el punto 3.1, por lo que cada byte se transmite por el medio físico como 10 bits.

5.3.2 Determinar el MSS para TCP y UDP en la subred local ethernet y en el enlace PPP entre el router CISCO 2513 y el Linux 1 (10.3.7.0) en base a los MTUs que presentan estas redes. Determinar la cadencia eficaz para ambas subredes y compararlas con las velocidades de transmisión correspondientes. Para ello se considerará el tiempo medido en el apartado anterior, pero teniendo en cuenta que el formato del paquete será el de un paquete TCP o UDP. Es decir,

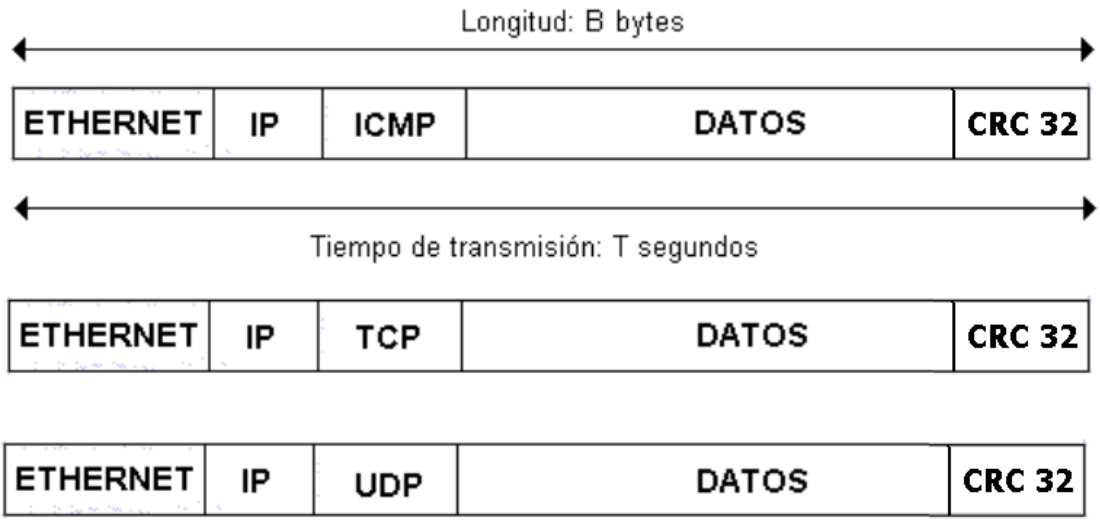


Figura 14. Cálculo de la Cadencia Eficaz en la capa de transporte.

Conocido el tiempo que tarda en ser trasmitido un paquete de B bytes es posible determinar la velocidad de transferencia del medio. Para ello nos valemos de la información que proporciona la aplicación **ping**. Si queremos conocer la cadencia eficaz para TCP y UDP debemos determinar qué cantidad bits de datos habría para cada protocolo en un paquete de B bytes que tarda T segundos en ser transmitido. **Hay que tener en cuenta que para el cálculo de la cadencia eficaz cada byte de datos son 8 bits,** ya que los bits redundantes sólo se tienen en cuenta para el cálculo de la velocidad de transferencia. Así, tendríamos:

Ce_TCP = (B - long. cab. Eth. – long. CRC – long. cab. IP - long. cab. TCP)*8 / T bps
Ce_UDP = (B - long. cab. Eth. – long. CRC – long. cab. IP - long. cab.UDP)*8 / T bps

6. Documentación complementaria

- RFC 793. Transmision Control protocol (TCP).
- RFC 768. User Datagram Protocol (UDP).
- RFC 896. Congestion Control in IP/TCP Internetworks.
- RFC 1700. Internet Assigned Numbers.
- RFC 1191. Path MTU discovery.
- RFC 1661. The Point-to-Point Protocol (PPP).