Redes

Ingeniería Informática (9185)

Manual de la Práctica 3:

Protocolos de Transporte TCP y UDP

Francisco Andrés Candelas Herías

Jorge Pomares Baeza

Grupo de **I**nnovación **E**ducativa en **A**utomática

© 2009 GITE – IEA

1. Objetivos

- 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 capa de 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 (*Transmision Control Protocol*: protocolo de control de la transmisión) y UDP (*User Datagram Protocol*: protocolo de datagramas de usuario). 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 1).

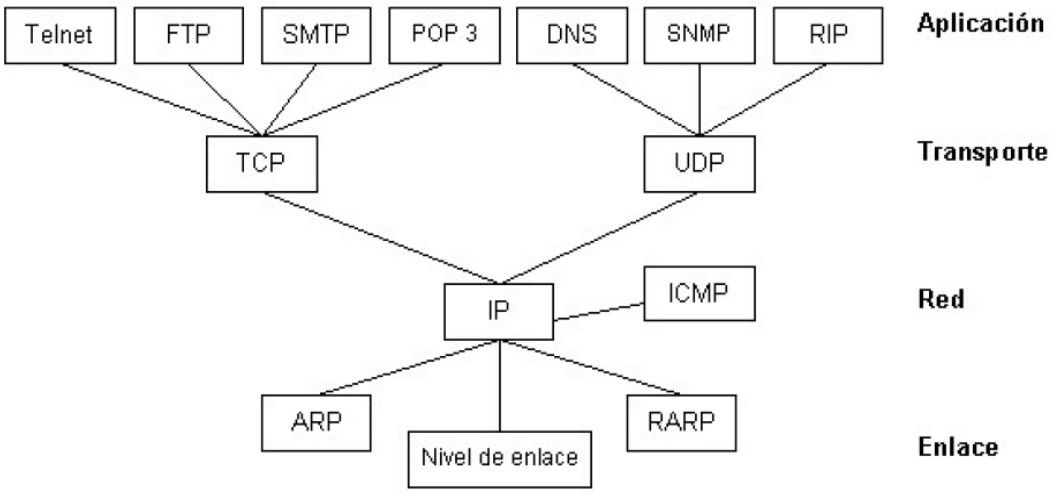

Figura 1. Conjunto de protocolos en la arquitectura de red TCP/IP.

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 o PDUs empleados por los protocolos TCP y UDP (también conocidos como segmentos) se encapsulan dentro del campo de datos de paquetes IP como muestra la figura 2.

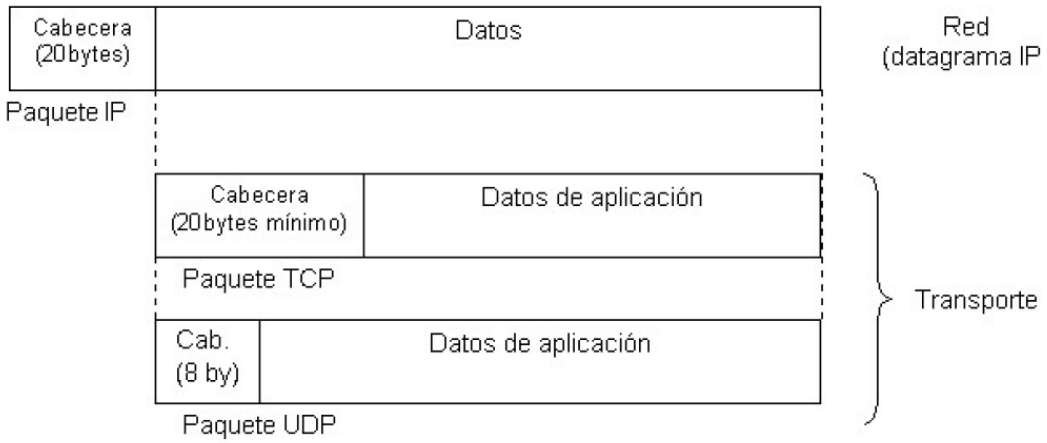

Figura 2. Encapsulación de paquetes de transporte en datagramas IP

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 de la figura 3.

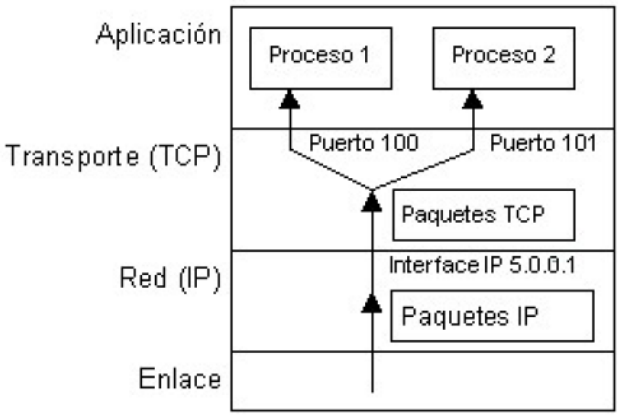

Figura 3. Multiplexación de conexiones en la capa de transporte de TCP/IP

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 conoce como *socket*. Los *sockets* permiten la transmisión bidireccional de datos, que puede ser en modo *full-duplex* si el sistema operativo lo permite. La figura 4 muestra un ejemplo de *socket* TCP.

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 los segundos. Es el proceso cliente el que siempre solicita establecer una conexión (crear un *socket*) con un servidor para acceder a un determinado servicio. Un servidor normalmente puede atender a varios clientes simultáneamente. Un cliente emplea un número de puerto que generalmente comienza en

1024. El sistema operativo de un equipo es el que asigna este número, comenzando por un valor e incrementando ese valor conforme se establecen nuevas conexiones.

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

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. Mientras que los servicios en los puertos 1 a 1023 están claramente definidos para servicios clásicos, también se definen puertos, para muchos otros servicios en el rango 1024 a 49151. Finalmente, el rango de puertos 49152 a 65535 se reserva para puertos dinámicos, es decir, empleados para servicios temporales. Cabe destacar que un mismo número de puerto puede ser utilizado para dos procesos con servicios diferentes, uno que use TCP y otro que use UDP, si bien lo común es que un puerto se asigna para un solo servicio, emplee sólo TCP, sólo UDP o ambos.

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

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*). Al final de la práctica, se listan los puertos oficiales más comunes.

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. El proceso TCP del emisor 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. La conexión implica que solo hay dos equipos involucrados en el intercambio de datos (un cliente y un servidor)
- Fiable. Emplea control de flujo mediante ventana deslizante de envío continuo y asentimientos positivos (ACKs o *Acknowledgements*) 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, el proceso de TCP en el receptor 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 se reenvía de nuevo el paquete.

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

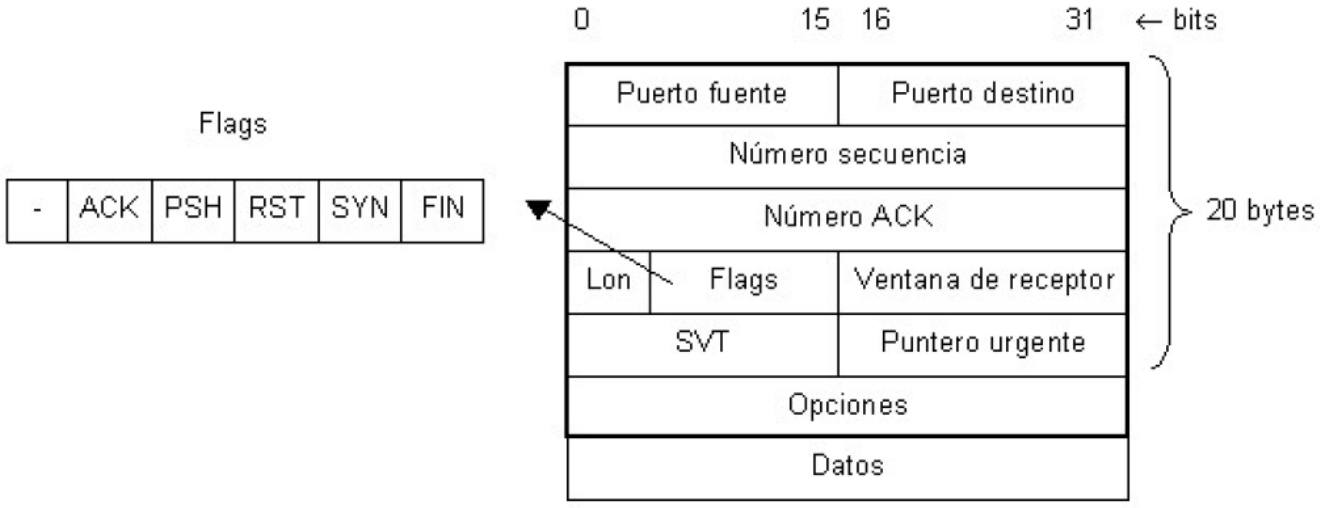

Figura 5. Formato de la cabecera del protocolo TCP.

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 correspondiente al primer byte del campo de datos del paquete.
- **Número de ACK**. Número del primer byte de datos que se espera recibir en un próximo paquete, con lo que se representa también que todos los bytes previos al indicado se han recibido correctamente.
- **Lon** (4 bits). Longitud como número de palabras de 32 bits (4 bytes) que forman la cabecera TCP. Como mínimo son 5 palabras o 20 bytes, lo que corresponde a que no existe el campo de opciones.
- **Flags**. Campo con bits con significado propio, del cual se usan sólo 6:

  ACK (*Acknowledgement*). 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 el proceso TCP del receptor debe pasar los datos que tenga almacenados a la capa de aplicación sin esperar a recibir más datos.

RST (*Reset*). 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 (*Finalize*). Se utiliza para solicitar la liberación de una conexión.

URG (Urgent). 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 puede recibir datos (aunque si 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.

Conexión y desconexión con TCP

La figura 6 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.

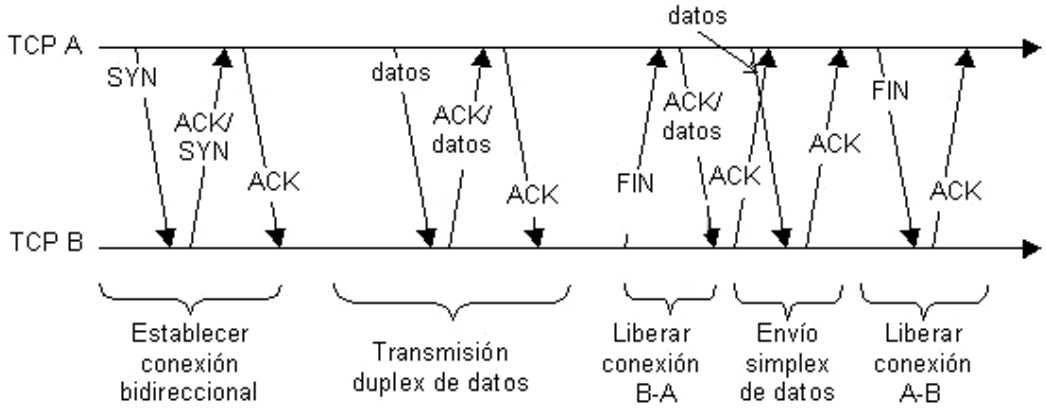

Figura 6. Secuencia de intercambio de paquetes en una conexión TCP.

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 (ver siguiente punto), 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.

La desconexión puede ser iniciada por cualquiera de los equipos. En ella 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.

TCP y La fragmentación

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, al comienzo de una conexión, negocia entre los dos extremos de la comunicación qué tamaño máximo de datos se va a utilizar para los paquetes TCP de esa conexión de forma que se evite su fragmentación. Este tamaño máximo de datos de TCP se conoce como MSS (M*áximun Segment Size* o tamaño máximo de segmento), y depende del valor de MTU que exista en cada extremo de la comunicación, como muestra la figura 7.

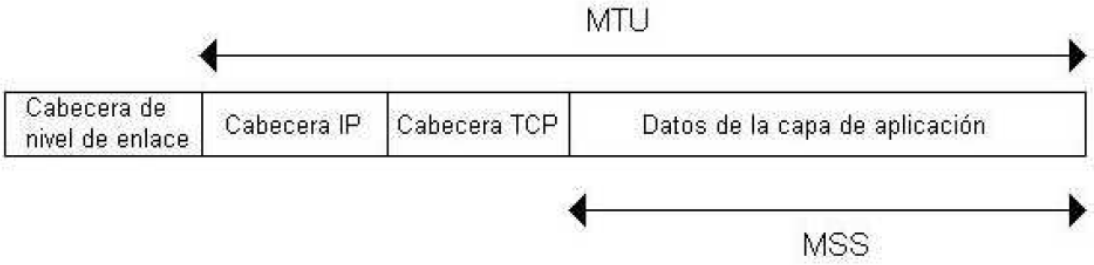

Figura 7. MSS de una conexión TCP.

Cada extremo de la comunicación puede calcular el valor de MSS que puede emplear para que no se produzca fragmentación como:

MSS = MTU - 20 bytes cabecera TCP - 20 bytes de 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 "*destination unreachable*", *con código* "*fragmentation needed and the bit don't fragment was set*". En la cabecera ICMP de este mensaje se indica dentro del campo en el campo "*Next Hop MTU*" cuál es el valor del MTU de la red que necesita fragmentar el paquete TCP. 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* funcione 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 tipo de mensaje ICMP.

Ventana de TCP y numeración de los bytes

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.

Algoritmo de Nagle de TCP

El Algoritmo de Nagle es un mecanismo utilizado por TCP cuyo propósito es evitar la aparición de paquetes de pequeño tamaño conocidos como *tinygrams* ocasionados por conexiones interactivas como las que suceden con los protocolos de aplicación de Telnet, de acceso remoto a un sistema.

En la aplicación Telnet cada pulsación de teclado se envía individualmente como el carácter ASCII correspondiente dentro de un paquete TCP al servidor, y se espera recibir del servidor el mismo carácter en un paquete TCP de vuelta (eco). Sobre una red WAN esto

podría ocasionar un exceso de paquetes pequeños muy perjudicial, ya que aumentaría la congestión en los *routers* de la red WAN debido al gran número de paquetes a encaminar.

Como solución, el protocolo TCP calcula el tiempo entre cada envío de paquete y su confirmación recibida, conocido como tiempo de ida y vuelta o RTT. Si este tiempo supera cierto valor, TCP entiende que la conexión es a través de alguna WAN, y en caso contrario supone una conexión a través de LANs. Dependiendo del RTT, TCP espera hasta obtener un número determinado de caracteres pulsados, guardándolos en un buffer, para después enviarlos en un único paquete y evitar los *tinygrams*.

A pesar de que el algoritmo de Nagle es adecuado para aplicaciones de terminal interactivas como Telnet cuando son usadas a través de redes lentas, para otras aplicaciones puede ser conveniente desactivar este algoritmo.

2.3. Protocolo 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.

El formato de un paquete UDP es el representado en la figura 8.

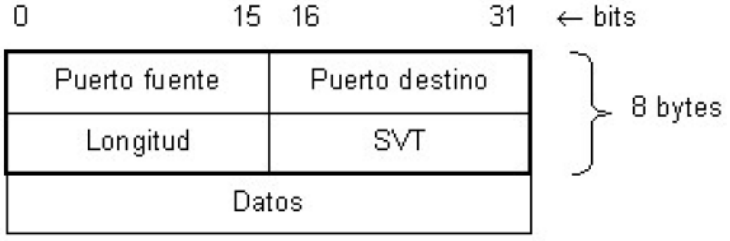

Figura 8. Formato de la cabecera del protocolo UDP.

Los campos de la cabecera de UDP 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 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 (Domain Name System), RIP (Routing Information protocol) o SNMP (Simple Network Management Protocol).
- Transmisiones multicast de video o audio. UDP es usado por aplicaciones de VoIP (*Voice over IP*), 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.

3. Evaluación del rendimiento a nivel de transporte.

3.1. Protocolo de nivel de enlace PPP

El protocolo PPP (*Point to Point Protocol*, Protocolo Punto a Punto), es un protocolo de nivel de enlace especificado para el implementar protocolos de red sobre enlaces entre pares de estaciones que está normalizado 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 muy utilizado.

Como muestra la figura 9, 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). Hay que tener en cuenta que en un enlace punto a punto entre un par de estaciones no resulta necesario usar direcciones de enlace o MAC, ya que solo hay dos equipos, y lo que uno envía llena directamente al otro. Así mismo, tampoco se requiere usar ARP para resolver las direcciones de enlace. 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. Después de los datos aparece una secuencia de verificación de la trama *(Frame Control Secuence)* que es un código CRC de 16 o 32.

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

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.

Además, hay que tener en cuenta qué tipo de nivel físico se emplea para la transmisión de los paquetes PPP. Por ejemplo, en el laboratorio existe una conexión PPP entre el router CISCO 2513 y el equipo Linux 1 (10.3.7.0), que está basada en una interfaz serie asíncrona RS-232, 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, tal y como muestra la figura 10. En esta figura también se puede observar como los bits se transmiten con lógica negativa.

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

3.2. Cadencia eficaz de una comunicación

Frente a la velocidad de transmisión (V_t) que es capaz de ofrecer un medio físico en bits por segundo se puede definir la cadencia eficaz (C_e) como la velocidad real en bits por segundo de envío de los datos, y que es inferior a la V_t 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:

$$C_e = n / T_o = (bps)$$

Donde n es el número de bits de datos transmitidos y T_o es el tiempo que tarda el paquete en ser transmitido por el canal. Este tiempo T_o 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.

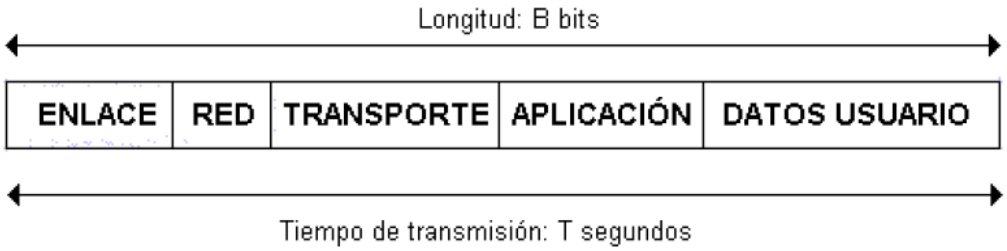

Figura 11. Formato genérico de un paquete en la arquitectura TCP/IP

Por ejemplo, si el formato de un paquete genérico en la arquitectura TCP/IP tiene la forma mostrada en la figura 11, la velocidad de transmisión en el medio físico por donde se realiza el envío del paquete será:

$$V_t = B / T = (bps)$$

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

$$C_{e\_enlace} = (B - longitud\ cabecera\ de\ enlace) / T = (bps)$$

$$C_{e\_red} = (B - lon.\ cab.\ enlace - long.\ cab.\ red) / T = (bps)$$

$$C_{e\_transporte} = (B - long.\ cab.\ enlace - long.\ cab.\ red - long.\ cab.\ transporte) / T = (bps)$$

$$C_{e\_aplicación} = (B - long.\ cab.\ enlace - long.\ cab.\ red - long.\ cab.\ transporte - \\ - log.\ cab.\ aplicación) / T = (bps)$$

De estas expresiones se deduce claramente que

$$V_t > C_{e\_enlace} > C_{e\_red} > C_{e\_transporte} > C_{e\_aplicación}$$

Es decir, que los protocolos de cualquier nivel en la arquitectura de red presentan una velocidad de transferencia efectiva para datos menor que la V_t del medio físico, y que además será menor cuanto mayor sea el nivel del protocolo dentro de la arquitectura.

4. Herramientas utilizadas

Rexec

Este 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 para MS. Windows llamado "*rexec.exe*" (figura 12), desarrollado por el profesor Luis Miguel Crespo Martínez, y que actúa como cliente del servicio *rexec*. 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.

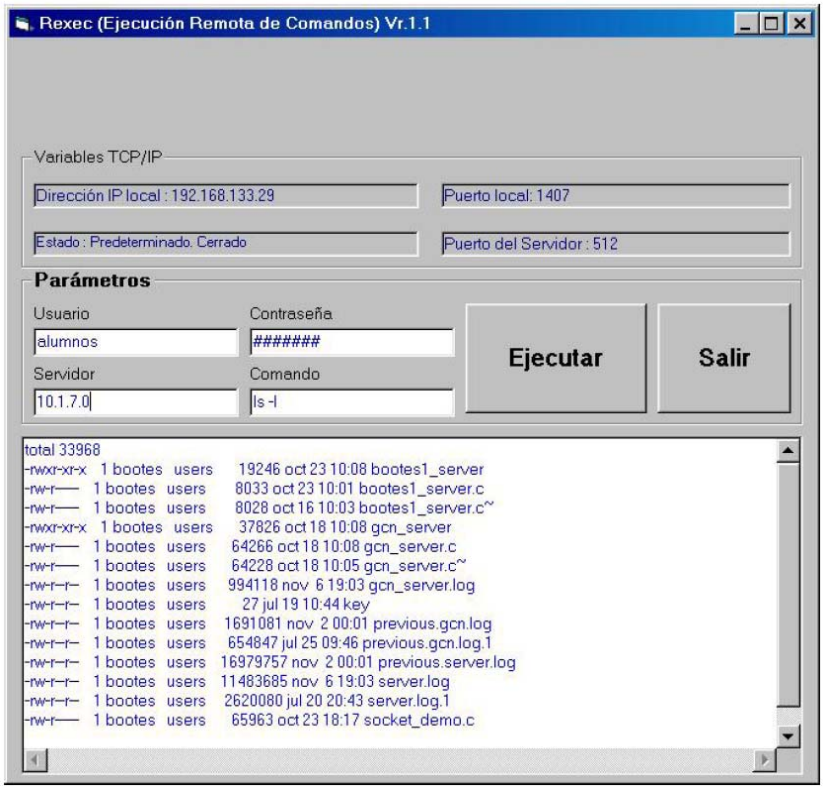

Figura 12. Aplicación rexec.exe

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

La aplicación "*udp.exe*" para el sistema operativo MS. Windows (figura 13), desarrollada también por Luis Miguel Crespo Martínez, permite enviar y recibir paquetes UDP. Para ello se indica en el panel inferior el texto de la aplicación en contenido del 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. Para que se generen paquetes es necesario colocar antes algún texto en la ventana. 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.

También se puede configurar el programa para recibir paquetes UDP. Solamente es necesario pulsar el botón "Abrir UDP a la escucha" para que el programa muestre los datos de los paquetes UDP dirigidos al puerto y a la dirección IP especificados.

Puesto que el protocolo UDP puede trabajar con direcciones de broadcast y de multicast, también se pueden utilizar estas con el programa "*udp.exe*".

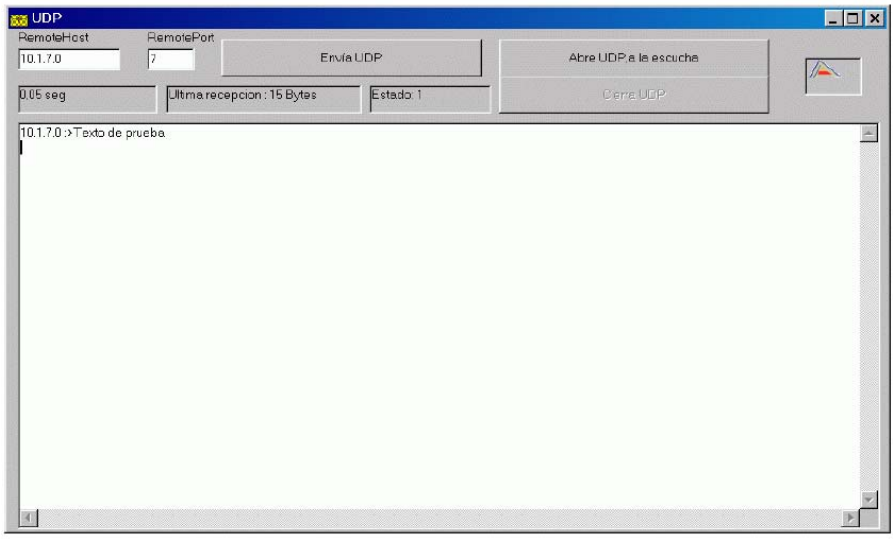

Figura 13. Aplicación udp.exe

Notas sobre el monitor Wireshark

El monitor de red 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.

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. Utilizar el usuario "alumnos" y contraseña "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 enunciado. 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 113 (0071 en

hexadecimal) del servicio "*Authentication service*" (RFC 912), 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 preguntando a su servicio en el puerto 113, y como el cliente no soporta la autenticación rechaza la conexión.

- Comprobar también los valores de puertos utilizados (número asignado de puerto cliente y número de puerto utilizado por el servidor), como varían los número de secuencia de byte y de ACK, y los *flags* activados en las cabeceras TCP.
- Analizar 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 RedesII_p2_51.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 RedesII_p2_51.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).

5.2. TCP y la norma RFC 1191

5.2.1. Descarga desde la carpeta "Prácticas" en la sección de materiales de Campus Virtual de la UA el fichero "RFC1191.zip" y extrae el contenido "RFC1191.txt" en la carpeta "Mis documentos" del PC.

En una ventana de línea de comandos ejecuta "route add 10.3.7.0 mask 255.255.255.255 172.20.43.231". Con esto se especifica al equipo que la puerta de enlace para llegar al destino 10.3.7.0 (Linux 1) es el *router* Cisco 1601. De esta forma, los paquetes dirigidos al destino 10.3.7.0 pasarán por la red 10.4.2.0/30 en vez de la 10.4.2.4/30. Puedes comprobar que la nueva puerta de enlace se ha configurado bien ejecutando el comando "netstat –rn".

Cambia de directorio en la ventana de comandos a "C:\Documents and Settings\EPS\Mis Documentos" y ejecuta programa "ftp" para enviar el fichero "rfc1191.txt" al servidor FTP de 10.3.7.0. Para ello emplear la siguiente secuencia de comandos:

```
C:\Documents and Settings\EPS\Mis Documentos>ftp 10.3.7.0
  Usuario: alumnos
  Password: alumnos
ftp> bin
ftp> put rfc1191.txt
ftp> bye
```

- Determina con el monitor de red qué valor de MSS se ha negociado en la conexión TCP. Para ello visualiza los paquetes TCP-IP de la conexión de datos de FTP intercambiados entre tu PC y el servidor 172.20.41.241.
- ¿Aparecen paquetes ICMP "*fragmentation needed and the bit don't fragment was set*"? ¿Quién envía el mensaje ICMP de error? ¿Qué red causa el problema?

- ¿Cómo 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?

5.2.2. Intenta 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.

5.3. Protocolo UDP

5.3.1. Utiliza el programa "udp.exe" 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.
- Probar con los otros *routers* de la red 172.20.43.192/26.

5.3.2. 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.exe".

- ¿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.

5.3.3. 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 172.20.43.192/26.

- ¿Qué equipos contestan en cada caso?

5.3.4. Intentar enviar un texto al puerto 80 de la máquina172.20.43.232 empleando el programa "*udp.exe*".

- Determina la secuencia de paquetes que se intercambian entre tu equipo y el servidor172.20.43.232.

5.4. Rendimiento de una comunicación

5.4.1. 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 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.

5.4.2. 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 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,

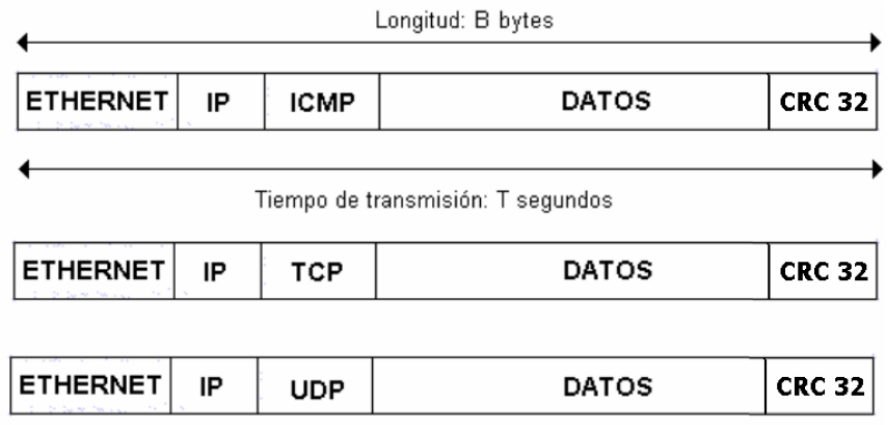

Figura 14. Cálculo de la Cadencia Eficaz en la capa de transporte

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 obtendríamos:

$$C_{e\_TCP} = (B - long.\ cab.\ Eth. - long.\ CRC - long.\ cab.\ IP - long.\ cab.\ TCP)\cdot 8 \,/\, T = (bps)$$

$$C_{e\_UDP} = (B - long.\ cab.\ Eth. - long.\ CRC - long.\ cab.\ IP - long.\ cab.UDP)\cdot 8 \,/\, T = (bps)$$

6. Documentación complementaria

6.1. Referencias

- RFC 793. Transmission Control protocol (TCP). En español en: http://www.rfc-es.org/rfc/rfc0793-es.txt.
- RFC 768. User Datagram Protocol (UDP). En español en: http://www.rfc-es.org/rfc/rfc0768-es.txt
- RFC 1191. Path MTU discovery. http://www.rfc-archive.org/getrfc.php?rfc=1191
- RFC 1661. The Point-to-Point Protocol (PPP). http://www.rfc-archive.org/getrfc.php?rfc=1661
- Listado de puertos de IANA. http://www.iana.org/assignments/port-numbers

6.2. Lista de puertos y protocolos comunes

Puerto	Transporte	Protocolo de aplicación
7	TCP, UDP	**Eco**. Este servicio devuelve una copia del paquete UDP recibido.
13	TCP, UDP	**Daytime**. Servicio de fecha y hora, que devuelve la hora y fecha del sistema donde está el servidor.
20, 21	TCP	**FTP** (File Transfer Portocol). Protocolo estándar para transferencia de archivos. El puerto 21 se utiliza para solicitar conexiones de control del cliente al servidor, y el putero 20 se usa el envío de datos en modo activo.
22	TCP	**SSH** (Secure Shell). Permite que un proceso cliente se conecte a un equipo para realizar operaciones sobre el. El servicio admite aplicaciones de consola remota, transferencias de archivos, ejecución de comandos y otras funciones, todas ellas sobre una comunicación segura (encriptada y autentificada).
23	TCP	**Telnet**. Servicio clásico de terminal de consola de texto remota. A diferencia de SSH, no ofrece una conexión segura.
25	TCP	**SMTP** (Simple Mail transfer Protocol). Protocolo clásico para enviar y encaminar mensajes de email.
53	UDP	**DNS** (Domain Name System). Un servidor de nombres del dominio recibe peticiones con cadenas de nombre de dominio (tipo www.ua.es) y devuelve respuestas con las direcciones IP equivalentes.
67,68	UDP	**BOOTP** (BootStrap Protocol) y **DHCP** (Dynamic Host Configuration Protocol). Protocolos de arranque por red, y de asignación dinámica de direcciones de red.
69	UDP	**TFTP** (Trivial File Transfer Protocol). Protocolo de transferencia de archivos al estilo de FTP, pero más sencillo y rápido, adecuado para LANs.
80	TCP	**HTTP** (Hypertext Transfer Protocol). Protocolo para transferencia de páginas web.
110	TCP	**POP3** (Post Office Protocol, V.3). Protocolo de acceso a buzones de correo para recuperar emails del servidor.
113	TCP, UDP	**Authentication Service.** Protocolo que permite determinar la identidad de un equipo que quiere establecer una conexión TCP/IP.
123	UDP	**NTP** (Network Time Protocol). Utilizado apara actualizar la hora y fecha de un sistema de acuerdo a un servidor de tiempo fiable.
137,138, 139	TCP	**NetBIOS**. Puertos usados por los servicios Name Service , Datagram Service, y Session Service del protocolo de red NetBIOS de Microsoft, usado para compartir archivos y otros recursos en los sistemas Windows.
143	TCP	**IMAP4** (Internet Message Access Protocol ver.4). Protocolo alternativo a POP3 para acceso a un buzón de correo.
161	UDP	**SNMP** (Simple Network Management Protocol). Protocolo para gestionar redes TCP/IP soportado por muchos sistemas.
443	TCP	**HTTPS** (HTTP over TLS/SSL). Acceso seguro (encriptado y autentificado) a páginas web.
514	UDP	**Syslog**. Protocolo usado para enviar mensajes de diagnostico de procesos y S.O. a través de la red.

Puerto	Transporte	Protocolo de aplicación
520	UDP	**RIP** (Routing Information protocol). Sencillo protocolo de encaminamiento dinámico, para actualización automática de tablas de encaminamiento de routers.
1701	UDP	**L2TP** (Level 2 Tunneling Protocol). Protocolo para gestion de VPN (Virtual Private Network), que permite el acceso remoto a los recursos de LANs.
1723	TCP, UDP	**PPTP** (Point to Point Tunneling Protocol). Protocolo de VPN propio de Microsoft, anterior a L2TP.
1812, 1813	UDP	**RADIUS** (Remote Authentication Dial In User Service). Puertos usados por los servicios de autentificación y gestión de RADIUS, un protocolo para gestión y control de bases de datos de usuarios de sistemas.
1863	TCP	Windows Live **Messenger** y MSN. Puerto usado por los protocolos de estas aplicaciones de Microsoft.
2948, 2949	TCP	**WAP** (Wireless Application Protocol) y MMS (Multimedia Messaging Service). Protocolos para acceso a servicios básicos de Internet desde un teléfono móvil
3074	TCP, UDP	**Xbox Live**. Puerto usado por el protocolo de juegos on-line de la consola Xbox.
3306	TCP	**MySQL**. Protocolo usado por el servicio de acceso a BB.DD. de MySQL.
3389	TCP	**RDP** (Remote Desktop Protocol). Acceso remoto a escritorio de sistemas MS. Windows.
5004, 5005	UDP	**RTP** (Real-time Transport Protocol). Protocolo auxiliar para transmisión de señales multimedia de tiempo real sobre IP. Usado, entre otras, por aplicaciones de VoIP (Voice over IP).