Tema 3. Capa de transporte en Internet

Introducción

Funciones y servicios de la capa de transporte

- Comunicación extremo a extremo (end-to-end).
- Multiplexación/demultiplexación de aplicaciones →puerto.

Protocolo UDP

- Multiplexación/demultiplexación de aplicaciones.
- Servicio no orientado a conexión y no fiable.
- La TPDU (Transport Protocol Data Unit) se denomina datagrama de usuario o datagrama UDP.

Protocolo TCP

- Multiplexación/demultiplexación de aplicaciones.
- Servicio orientado a conexión, fiable: control de errores y de flujo. Control de la conexión. Control de congestión.
- Forma, junto a IP, el corazón del modelo de referencia seguido en Internet, al cual da su nombre: TCP/IP.

Protocolo de datagrama de usuario (UDP)

El “User Datagram Protocol” está definido en el RFC 768.

Funcionalidad “best-effort”

- Servicio no orientado a conexión.
- Servicio no fiable.
- No hay garantías de entrega ordenada.
- No hay control de congestión: entrega tan rápida como se pueda.
- Multiplexación/demultiplexación: transportar las TPDU al proceso correcto. A través de ella se hace posible diferenciar entre diversas aplicaciones Internet simultáneas en un mismo host.

Multiplexación/demultiplexación

Se basa en transportar las TPDU al proceso correcto. UDP se usa frecuentemente para aplicaciones multimedia: tolerantes a fallos y sensibles a retardos. Cada segmento UDP se encapsula en un datagrama IP.

Puerto	Aplicación/Servicio	Descripción
53	DNS	Servicio de nombres de domino
69	TFTP	Transferencia simple de ficheros
123	NTP	Protocolo de tiempo de red
161	SNMP	Protocolo simple de administración de red
520	RIP	Protocolo de información de encaminamiento

Para el servicio DNS la mayoría de paquetes usan UDP pero para el control usan TCP.

Composición del datagrama UDP

- Puerto origen: campo de 16 bits que especifica el puerto origen.
- Puerto destino: campo de 16 bits que especifica el puerto destino.
- Longitud UDP: 16 bits que indican el número de octetos que forman el datagrama UDP completo.
- Comprobación (campo checksum): suma de comprobación usual considerada en Internet, en la cual no se contempla el campo *Datos*.
- Datos: PDU (Protocol Data Unit) de capa superior encapsulada en el datagrama UDP.

Los cuatro primeros campos constituyen la cabecera del datagrama UDP, la cual tiene una longitud fija de 64 bits.

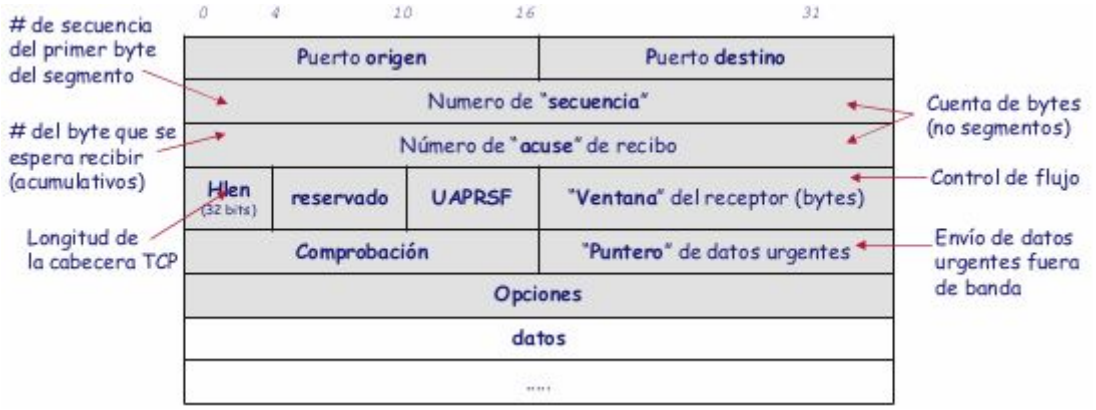
Protocolo de control de transmisión (TCP)

El “Transmission Control Protocol” está definido en el RFC 793.

Características

- Servicio orientado a conexión (“hand-shaking”). Se hace preciso arbitrar mecanismos que permitan el establecimiento y el cierre de conexión antes y después, respectivamente, de la transmisión de datos.
- Entrega ordenada. Permite el envío secuencial de datos.
- Full-duplex (a través del mismo canal de comunicación se puede enviar y recibir información) mediante la técnica piggybacking que consiste en la consideración simultánea de campos para el envío de datos y para la confirmación de las transmisiones en sentido contrario, dentro la misma PDU; en otras palabras, cuando se envían paquetes se informa de los paquetes recibidos.
- Mecanismo de control de flujo de detección y recuperación de errores (ARQ - Automatic Repeat reQuest).
- Mecanismo de control de congestión. Los mensajes ACK se envían con temporizadores (timeouts) que funcionan como Stop and Wait. Estos timeouts son adaptables para evitar la congestión.
- Servicio fiable →control de congestión y control de flujo.
- Multiplexación de varias aplicaciones simultáneas sobre un mismo host.

Segmento TCP: TPDU TCP



A la TPDU utilizada en TCP se le denomina segmento TCP. Para explicar los campos que forman el segmento TCP se utiliza como apoyo las funciones anteriormente descritas. Primero, se detallan los campos menos importantes para poder poner la atención en los campos principales.

- Hlen: campo de 4 bits utilizado para indicar la longitud de la cabecera TCP en palabras de 32 bits.
- Reservado: campo de 6 bits sin uso en la actualidad.
- Datos: campo donde se encapsula el mensaje de usuario. No se indica explícitamente.
- Opciones: este campo, de longitud variable, puede presentar dos formatos distintos: (a) 1 único octeto de tipo de opción o (b) 1 octeto de tipo de opción más 1 octeto que indica la longitud de los datos correspondientes a la opción, más los datos como tales.

Cada segmento TCP se encapsula en un datagrama IP.

Multiplexación/demultiplexación de aplicaciones

Puerto	Aplicación/Servicio	Descripción
20	FTP-DATA	Transferencia de ficheros: datos
21	FTP	Transferencia de ficheros: control
22	SSH	Terminal Seguro
23	TELNET	Acceso remoto
25	SMTP	Correo electrónico
53	DNS	Servicio de nombres de domino
80	HTTP	Acceso hipertexto (web)
110	POP3	Descarga de correo

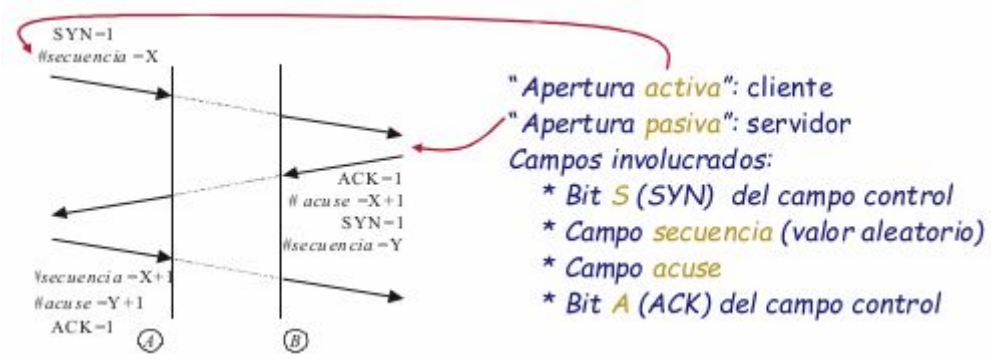
Al igual que en UDP, el puerto de origen y el puerto destino están formados por campos de 16 bits. El puerto destino viene con el servicio, por ejemplo el puerto 20 con SSH.

Control de la conexión

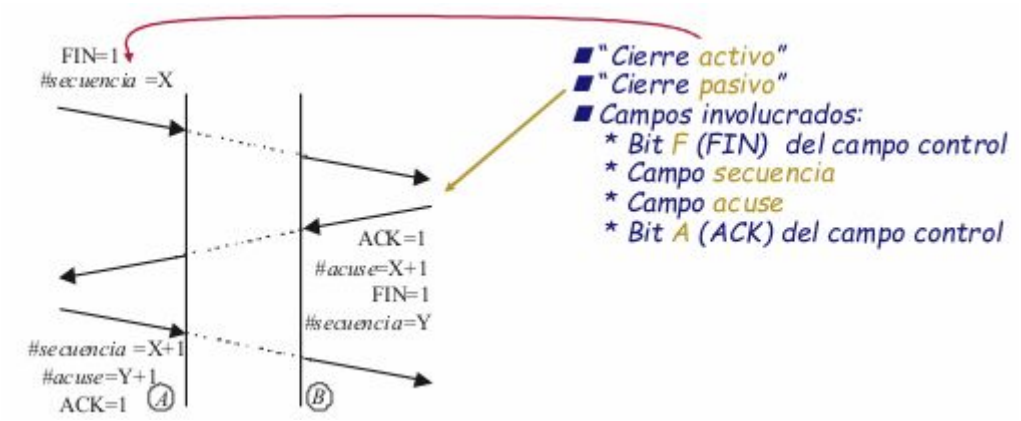
El intercambio de información tiene tres fases: three-way handshake.

1. Establecimiento de la conexión (sincronizar # de secuencia y reservar recursos). uno de los dos extremos activa el bit S. Simultáneamente, a través del campo secuencia se especifica un número de bytes arbitrario (0 o nº) a partir del cuál se comenzarán a numerar los segmentos en la forma ya conocida. Esto reduce la probabilidad de interferencia entre comunicaciones y disminuye el número de errores en éstas. En caso de aceptación de conexión, el otro extremo responde con un segmento TCP en el que se activa el bit ACK del campo control y se hace acuse = X + 1, donde X es el número de secuencia especificado por el emisor en el campo secuencia del emisor. La respuesta enviada no consiste solo en la aceptación de ésta como se ha indicado antes, sino que además se activa el bit SYN y se elige un número de secuencia Y, arbitrario e independiente del especificado por el emisor, para la transmisión en sentido contrario.
2. Intercambio de datos (full-duplex).
3. Cierre de la conexión (liberar recursos). Es análogo al procedimiento seguido para el establecimiento, excepto que el bit del campo de control utilizado no es SYN, ahora es FIN. Primero se envía un segmento TCP con FIN activado y secuencia = X cuando no existen más datos que transmitir en uno de los dos sentidos. Luego el otro extremo envía la confirmación de la solicitud recibida, activa ACK y acuse = X + 1, y solicitud (si procede) de cierre en el otro sentido de transmisión, activa FIN y secuencia = Y. Finalmente, el extremo que originó la finalización genera un segmento TCP con acuse = Y + 1 y ACK activado en respuesta al segmento del receptor.

Establecimiento de la conexión



Cierre de la conexión



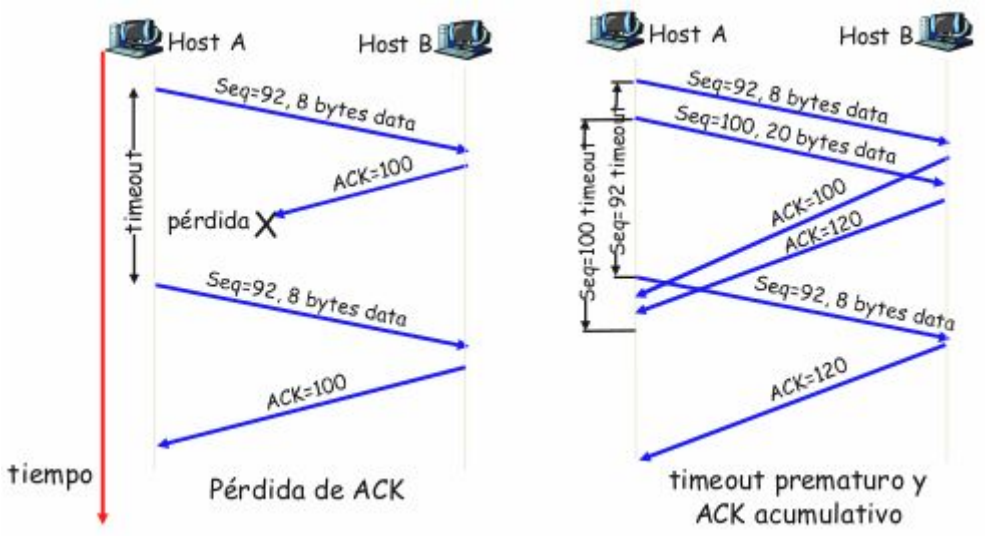
Números de secuencia

- Campo de 32 bits→2^{32} valores.
- Inicialización (#secuencia = ISN).
    - Empieza en el ISN (Initial Sequence Number) elegido por el sistema.
    - Para el ISN, el estándar sugiere utilizar un contador entero incrementado en 1 cada 4 μs aproximadamente→ciclo al cabo de 4 horas 46 min.
    - Protege de coincidencias, pero no de sabotajes.
- Incremento (#secuencia = #secuencia + Nbytes).
    - Se incrementa según los bytes de carga útil (payload).
    - Los flags SYN y FIN incrementan en 1 el número de secuencia.

Control de errores y de flujo

- Mejorar rendimiento→ventana deslizante: este campo de 16 bits indica el número de octetos que se autoriza al emisor de forma consecutiva. Las ventanas tienen un tamaño dinámico, por lo que se intentan adaptar a las capacidades de la red.
- Control de errores: esquema ARQ con confirmaciones positivas y acumulativas. La recepción de una confirmación positiva implica la suposición automática de que todos los datos transmitidos con anterioridad a los confirmados se han recibido también correctamente
- Campos involucrados:
    - Campo secuencia: indica el offset (en bytes) dentro del mensaje completo al que corresponde el primero de los octetos del campo datos del segmento.
    - Campo acuse: número de byte esperado en el receptor. campo de confirmación de 32 bits que indica el número de byte esperado en el receptor. Dicho byte hace referencia al campo secuencia especificado en el segmento confirmado.
    - Bit A (ACK) del campo de control: segundo bit del campo de 6 denominado control. Indica la validez o no de la confirmación especificada en el campo acuse.
    - Campo comprobación: a partir de este campo el receptor del segmento determina si éste se ha recibido correctamente o no. Es un checksum de todo el segmento y hace uso de pseudo-cabecera.

Escenarios de retransmisión



Se corresponde con un esquema ARQ tradicional: (1) envío de un segmento de datos por parte del emisor, (2) comprobación del campo de redundancia y confirmación en consecuencia por parte del receptor y, en caso de concluirse que la recepción ha resultado errónea, (3) reenvío de los datos correspondientes por parte del emisor.

Generación de ACKs

Evento	Acción del TCP receptor
Llegada ordenada de segmento, sin discontinuidad, todo lo anterior ya confirmado.	Retrasar ACK. Esperar recibir al siguiente segmento hasta 500 mseg. Si no llega, enviar ACK.
Llegada ordenada de segmento, sin discontinuidad, hay pendiente un ACK retrasado.	Inmediatamente enviar un único ACK acumulativo.
Llegada desordenada de segmento con # de sec. mayor que el esperado, discontinuidad detectada.	Enviar un ACK duplicado, indicando el # de sec. del siguiente byte esperado.
Llegada de un segmento que completa una discontinuidad parcial o totalmente.	Confirmar ACK inmediatamente si el segmento comienza en el extremo inferior de la discontinuidad.

Estimación de los timeouts

Los timeouts se pueden estimar de las siguientes formas:

- Mayor que el tiempo de ida y vuelta (RTT).
- Si es demasiado pequeño: timeouts prematuros.
- Si es demasiado grande: reacción lenta a pérdida de segmentos.
- Para situaciones cambiantes la mejor solución es la que mejor se adapte.

Cada vez que se transmite un segmento, el emisor anota el instante de tiempo en que se produce el envío. También se anota el tiempo en que se recibe la confirmación

correspondiente a dicho segmento. La diferencia entre ambos tiempos es el tiempo de ida y vuelta (RTT del inglés Round-Trip Time). Para estimar los timeouts se utiliza este tiempo.

RTTmedido: *tiempo desde la emisión de un segmento hasta la recepción del ACK.*

$$RTT = (1-\alpha) \times RTT + \alpha \times RTTmedido\ ,\ \alpha\ \&\ \beta \in [0,1]$$

$$Desv = (1-\beta) \times Desv + \beta \times |\ RTTmedido - RTT\ |$$

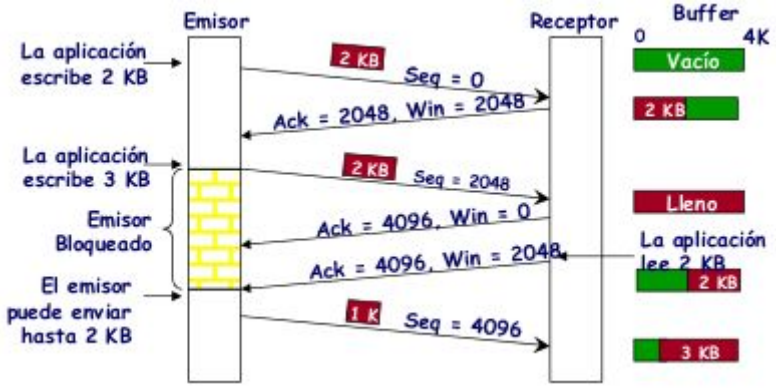
$$Timeout = RTT + 4 * Desv\ (Inicial\ 1s)$$

La técnica adaptable comentada presenta un problema. Cuando se recibe una confirmación no se sabe si ésta corresponde al segmento originalmente transmitido o a una potencial retransmisión posterior del mismo. A este problema se le conoce como ambigüedad en las confirmaciones. La solución es el algoritmo de Karn, que consiste en actualizar el RTT sólo para los segmentos no repetidos. Si hay que repetir un segmento se incrementa el timeout. El algoritmo de Karn presenta un inconveniente: el RTT no se adapta cuando se realizan retransmisiones. Surge así la técnica conocida como retroceso del temporizador (timer backoff), en la cual, cada vez que expira el timeout asociado a un segmento TCP, se hace: timeout nuevo = $\gamma \cdot$ timeout viejo.

Control de flujo

- Es un procedimiento para evitar que el emisor sature al receptor.
- Es un esquema crediticio: el receptor avisa al emisor de lo que puede aceptar
- Se utiliza el campo ventana (WINDOW) en el segmento TCP para establecer la ventana ofertada. Este campo de 16 bits indica el número de octetos que se autoriza a enviar al emisor de forma consecutiva.

En el emisor (paso 1). Transmite datos de acuerdo con lo que se denomina ventana útil, la cual se calcula como *ventana útil = ventana ofertada - bytes en tránsito,* donde, como es fácilmente comprensible, bytes en tránsito se refiere a los octetos ya transmitidos hacia el receptor y de los cuales aún no se ha recibido confirmación.
En el receptor (paso 2). Especifica al emisor un tamaño de ventana máximo (en bytes) autorizado para transmitir. Es lo que se conoce como ventana ofertada.



El posible problema que se presenta es el síndrome de la ventana tonta (RFC 813) si se utilizan segmentos muy pequeños. El problema que se plantea es el siguiente: en una transmisión de datos donde el tamaño de un segmento TCP es pequeño. Esto provoca la liberación de poco espacio en los buffers del receptor cuando sea confirmado dicho

segmento y, por tanto, la especificación posterior de un tamaño de ventana ofertada también reducido. La evolución de esta situación provoca la emisión de segmentos TCP cada vez más cortos, y ésto último que la especificación posterior de tamaños de ventana ofertada sean aún más pequeños.
Una posible mejora es utilizar la ventana optimista (RFC 813).
ventana útil emisor = ventana ofertada receptor
Consiste en el envío por parte del emisor de una cantidad de datos mayor que la realmente ofertada por el receptor. Esta idea se fundamenta en la suposición de que la transmisión se va a desarrollar con éxito, de forma que los bytes en tránsito se van a recibir correctamente y, así, se producirá una liberación de espacio adicional al especificado mediante la ventana ofertada. Por tanto la idea subyace en la igualdad entre la ventana útil y la ventana ofertada.
Es posible hacer entregas no ordenadas: bit U (URG). Se conoce como envío fuera de banda. El bit U significa que en la posición especificada por el campo puntero del segmento TCP finalizan los datos urgentes contenidos en el mismo (siempre comienzan en el byte 0).
Se puede solicitar una entrega inmediata: bit P (PSH). Fuerza el envío de un paquete y se usa en algunas aplicaciones para evitar retardos.

Control de congestión

El procedimiento para evitar que el emisor sature la red:

- ancho de banda de las líneas.
- buffer en dispositivos de interconexión.

La solución es limitar el tráfico generado (= control de flujo).
El problema es que no hay “un ente” que avise de lo que se puede enviar (≠ control de flujo). Como no hay solución se usa una medida indirecta que provoca pérdidas y/o retrasos en los ACKs.
En el emisor se utilizan una ventana y un umbral. Para la implementación se usa el decremento multiplicativo e inicio lento a nivel TCP. Previamente se definen los siguientes parámetros:

- Ventana permitida, correspondiente a la ventana de emisión o cantidad de datos que el emisor puede transmitir de forma consecutiva sin esperar confirmación.
- Ventana de recepción, correspondiente, como su nombre indica, a la cantidad de datos que el receptor autoriza a enviar de forma consecutiva al emisor.
- Ventana de congestión, la cual corresponde a la máxima cantidad de datos que se autoriza a transmitir en una situación de congestión.

El proceso seguido es:

1. Inicialmente VCongestión = MSS (maximum segment size)→Actualmente 2, 3, o 4. El umbral se establece a un cierto valor.
2. Dependiendo de VCongestión:
    2.1. Inicio lento: Una vez superada la situación de congestión se vuelve al funcionamiento normal del sistema. Para ello, el esquema de inicio lento consiste en hacer inicialmente la ventana de congestión a 1 segmento, incrementando su valor en 1 cada vez que se reciba una confirmación(crecimiento exponencial).
    Si VCongestión < Umbral, por cada ACK recibido. VCongestión += MSS

2.2. Prevención de la congestión: Se utiliza para evitar que el incremento asociado a la ventana de emisión pueda volver a provocar congestión en la subred y consiste en el incremento en un valor 1 de dicha ventana si, y sólo si, se han confirmado todos los segmentos de la misma. Si VCongestión > Umbral, por cada ventana completada (todos ACKs recibidos), entonces VCongestion += MSS (crecimiento lineal).

2.3. Decremento multiplicativo: consiste en reducir la ventana de congestión a la mitad de su valor anterior cada vez que expire el temporizador asociado a la retransmisión de un segmento. Además de reducir la ventana de emisión se incrementan de forma potencial los temporizadores asociados a las retransmisiones de los segmentos. Si hay timeout entonces
Umbral = VCongestion/2
VCongestion = MSS

Combinación del control de flujo y congestión

El emisor elige el mínimo de las ventanas correspondientes:

```
Bytes_permitidos_enviar = min{VCongestion, VentanaDeRecepcion}

ventana permitida emisor = Bytes permitidos enviar - bytes en
tránsito
```

Extensiones TCP

- TCP se define con múltiples “sabores”.
- Los diferentes sabores no afectan a la interoperabilidad entre los extremos.
- Desde cualquier versión de Linux con kernel mayor que la 2.6.19 se usa por defecto TCP CuBIC.
- Adaptación de TCP a las redes actuales (RFC 1323, 2018).
    - Ventana escalada: opción TCP en segmentos SYN: Hasta 1GB autorizados.
    - Estimación RTT: opción TCP de sello de tiempo en todos los segmentos.
    - PAWS (“Protect Against Wrapped Sequence numbers”): sello de tiempo y rechazo de segmentos duplicados
    - SACK: confirmaciones selectivas.