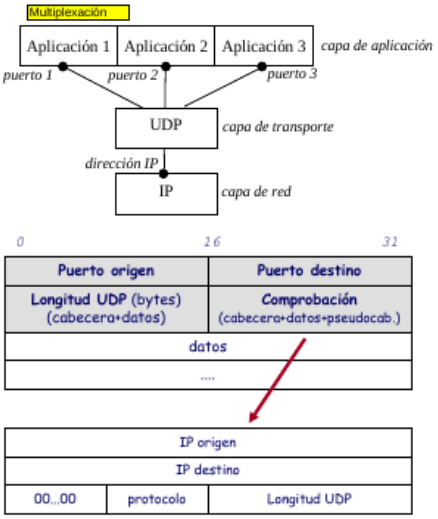
TEMA 3: CAPA DE TRANSPORTE EN INTERNET

La capa de transporte realiza comunicaciones extremo a extremo (end-to-end) y la multiplexación y demultiplexación de aplicaciones → puerto

1. Protocolo de datagrama de usuario (UDP)

El protocolo UDP (*User Datagram Protocol*) utiliza la funcionalidad '*best effort*'. Es un servicio no orientado a conexión, por tanto, es no fiable, lo cual quiere decir que no se sabe si el paquete va llegar ni cuándo lo hará. No hay garantía de entrega ordenada ni control de congestión (entrega tan rápida como se pueda), pero sí hace comprobación de errores, luego si un paquete está mal, lo desecha.



Se suele usar para aplicaciones multimedia, ya que son tolerantes a fallos y sensibles a retardos, pues con este protocolo disminuye el *delay*. Cada segmento UDP se encapsula en un datagrama IP.

El *checksum* está inicialmente a 0, y se autocalcula sumando el resto de los campos de la cabecera, haciendo al final el complemento a uno. Esto se hace así porque es más rápido en la comprobación, ya que al sumar con el destino dará una cadena de dieciséis 1s.

Realiza multiplexación y demultiplexación de aplicaciones (transportar las TPDU(1) al proceso correcto).

(1) Transport Protocol Data Unit: mensaje enviado de una entidad de transporte a otra, contenidos en paquetes (intercambiados por capa de red), a su vez contenidos en tramas (intercambiadas por capa de enlace de datos).

2. Protocolo de control de transmisión (TCP)

El protocolo TCP (*Transmission Control Protocol*) es un servicio orientado a conexión (fiable - *handshaking*). TCP se encarga de mantener la conexión y es inherentemente complejo. Garantiza entrega ordenada (simplifica la parte de aplicación) y es full-duplex. Realiza control de errores (ARQ) y de flujo (ACKs, *piggybacking*, *timeouts*, ventanas adaptables), control de la conexión y control de congestión. Por todo esto es un servicio fiable.

ARQ: si hay un envío secuencial de paquetes y en un punto hay un error en el envío de alguno de ellos, se para el envío hasta que llegue el que dio error. Para esto los paquete cuando llegan, responden al origen; si llega bien manda una respuesta positiva, y si llega mal, una negativa. Pero este mecanismo necesita otros apoyos extra, como es la incorporación de un temporizador para saber cuánto esperar para recibir respuesta y así volver a mandar el paquete si fuese necesario.

La conexión TCP se identifica por: puerto e IP del origen y del destino y, opcionalmente, el protocolo. Cada **segmento** TCP se encapsula en un datagrama IP. La cabecera de TCP es más compleja, dado que el protocolo por sí mismo lo es, y su longitud es variable: U_{rgent} A_{ck} P_{ush} R_{eset} S_{ync} $F_{inalize}$

2.1 Control de la conexión

El número de secuencia es un campo de 32 bits (2^{32} valores). Se inicializa al ISN (*Initial Sequence Number*) elegido por el sistema, que protege de coincidencias, pero no de sabotajes.

Para el control de la conexión el intercambio de información tiene tres fases (three-way handshake):

1. Establecimiento de la conexión (sincronizar número de secuencia y reservar recursos): se inicia el número de secuencia aleatoriamente. El servidor genera otro número de secuencia aleatorio.

   El SYN es un paquete de control sin información y sin número de secuencia → x
   El SYN-ACK es otro paquete de control igual que el anterior → x+1
   El ACK es el último paquete de control de todos con los flags a 0, menos ACK.
2. Intercambio de datos (full-duplex).
3. Cierre de la conexión: se liberan los recursos. El bit de fin está activo (el fin es una **solicitud** de cierre de la conexión). Puede haber algún ACK. Si quedan paquetes pendientes, se envían después durante MSL (Maximum Sequence Lenght).

2.2 Control de errores y de flujo

Control de errores

El control de errores se realiza mediante un esquema ARQ (*Automatic Repeat reQuest*) con confirmaciones positivas y acumulativas, es decir, se requiere de confirmaciones positivas por parte del receptor. Se denominan acumulativas porque no es necesario confirmar uno a uno todos los segmentos recibidos, sino que es posible confirmar más de uno de forma simultánea.

Los campos involucrados son:

- Campo **secuencia**: offset (en bytes) dentro del mensaje que indica el primer octeto del campo de datos. Permite la entrega ordenada.
- Campo **acuse**: número de byte esperado en el receptor. Tiene carácter acumulativo.
- Bit **A** (ACK) del campo de **control**. Indica la validez o no de la confirmación de acuse.
- Campo de **comprobación**: checksum de todo el segmento y uso de pseudo-cabecera.

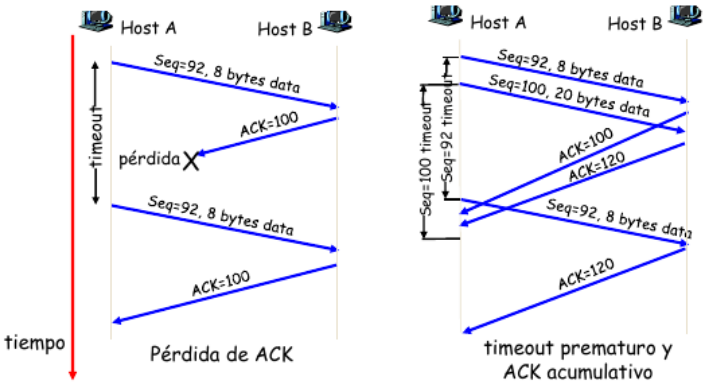


Gráfico 1: Escenarios de retransmisión

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 ms. 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.

Cómo estimar los "***timeouts***": es mayor que el tiempo de ida y vuelta (RTT); si es demasiado pequeño (timeouts prematuros); si es demasiado grande (reacción lenta ante pérdida de segmentos). Para situaciones cambiantes la mejor solución es la adaptable, es decir, según el emisor estime la velocidad de la red. Tiene que inferir y modificar en tiempo real el tiempo del *timeout*.

RTT_{medido}: tiempo desde la emisión de un segmento hasta la recepción del ACK

Control de flujo

Para evitar que el emisor sature al receptor se utiliza un esquema crediticio donde el receptor avisa al emisor de lo que puede aceptar. Para ello se utiliza el campo "ventana" (window) en el segmento TCP para establecer la ventana ofertada. La "ventana" del receptor dice el tamaño de la ventana en cada momento utilizando heurísticas.

- El receptor especifica al emisor un tamaño de ventana máximo (en bytes) autorizado para transmitir → ventana ofertada.
- El emisor transmite datos de acuerdo con lo que se denomina **ventana útil**, que es igual a la *ventana ofertada – bytes en tránsito*, donde los bytes en tránsito se refieren a los octetos ya transmitidos hacia el receptor y de los cuales aún no se ha recibido confirmación.

Control de congestión

Parecido al control de flujo. Controla, sobre todo, el envío del emisor para no congestionar la red ni al receptor. El único mecanismo para comprobar que hay una ralentización en la red es un sistema heurístico: se limita el tráfico generado, que es igual al control del flujo, solo que no hay ningún "ente" que avise de lo que se puede enviar, para ello se usa una medida indirecta: pérdidas y/o retrasos en los ACKs. De este modo, el emisor puede controlar la cantidad de envíos.

Para controlar la velocidad también se utiliza una ventana de un tamaño característico que funciona como limitador (TCP Tahoe). Tahoe, en el emisor, empieza con una ventana pequeña y un umbral para intentar no congestionar la red. El umbral permite saber en qué momento se pasa de de acelerar mucho (en el inicio)[(1)] a mantener el crecimiento[(2)]. Inicialmente VCongestión = MSS (Maximum Segment Size) y el umbral a un cierto valor:

Si VCongestión < Umbral, **por cada ACK recibido**[(1) Inicio lento]
→ VCongestión += MSS (crecimiento exponencial)
Si Vcongestión > Umbral, **por cada ventana completada** (todos ACKs recibidos) [(2) Prevención de congestión]
→ Vcongestión += MSS (crecimiento lineal)
Si hay timeouts[(3)] → Umbral = VCongestión/2; VCongestión = MSS

(3) Si hay timeouts en cualquier situación, utiliza ese retraso como alarma de que la red se puede estar congestionando, de modo que el umbral será igual a la mitad del tamaño de la ventana de congestión y ésta pasará a ser igual al MSS. Por tanto, Tahoe cae al principio en cada timeout, lo cual no puede ser muy desado. A esto se le denomina decremento multiplicativo.

Combinación del control de flujo y congestión

No se quiere congestionar la red ni el receptor, por lo que el emisor selecciona la ventana más pequeña de las dos para saber los bytes permitidos a enviar; a esto se le resta los bytes en tránsito.

```
Bytes permitidos a enviar = min{ VCongestión, Ventana ofertada en el receptor }
Ventana útil del emisor = bytes permitios a enviar – bytes en tránsito
```

3. Extensiones TCP

TCP se define con múltiples "sabores", cada uno de los cuales no afectan a la interoperabilidad entre los extremos.

- **Ventana escalada:** Opción TCP → factor de escala en segmentos SYN (hasta $2^{14} \cdot 2^{16}$ B = 1GB autorizados).
- **Estimación RTT:** Opción TCP de cálculo del RTT para cada segmento enviado mediante un sello de tiempo, de forma que cuando se genera un segmento TCP se incluye dicha opción, que será devuelta en un paquete ACK.
- **PAWS** ("*Protect Against Wrapped Sequence numbers*")**:** Protección contra números de secuencia ya recibidos, que hace uso del sello de tiempo de TCP para el rechazo de segmentos duplicados.
- **SACK:** Confirmaciones selectivas. Dice qué paquetes reenviar ante la pérdida aislada de un segmento.