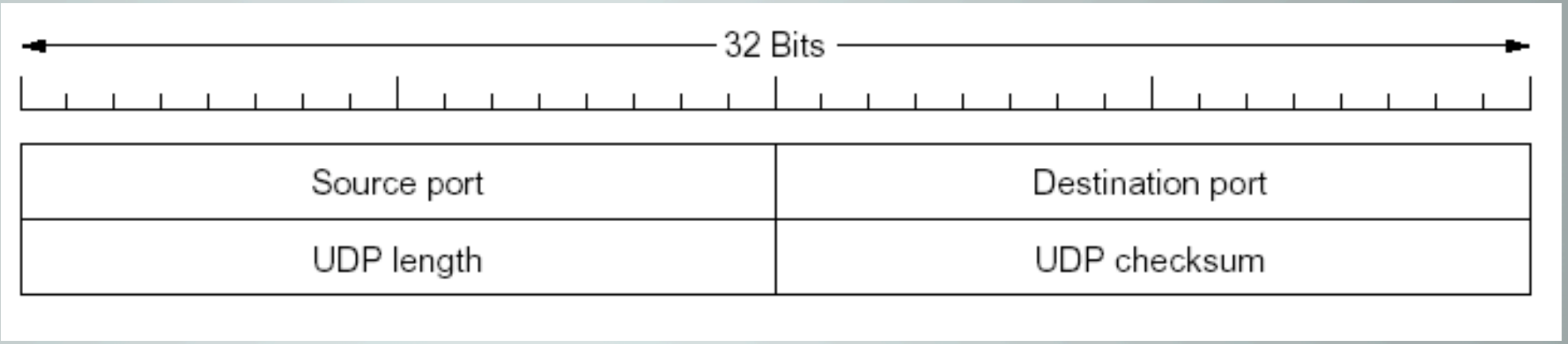
Transporte en Internet

UDP

El User Datagram Protocol (UPD) es esencialmente una versión en la capa de transporte de IP.

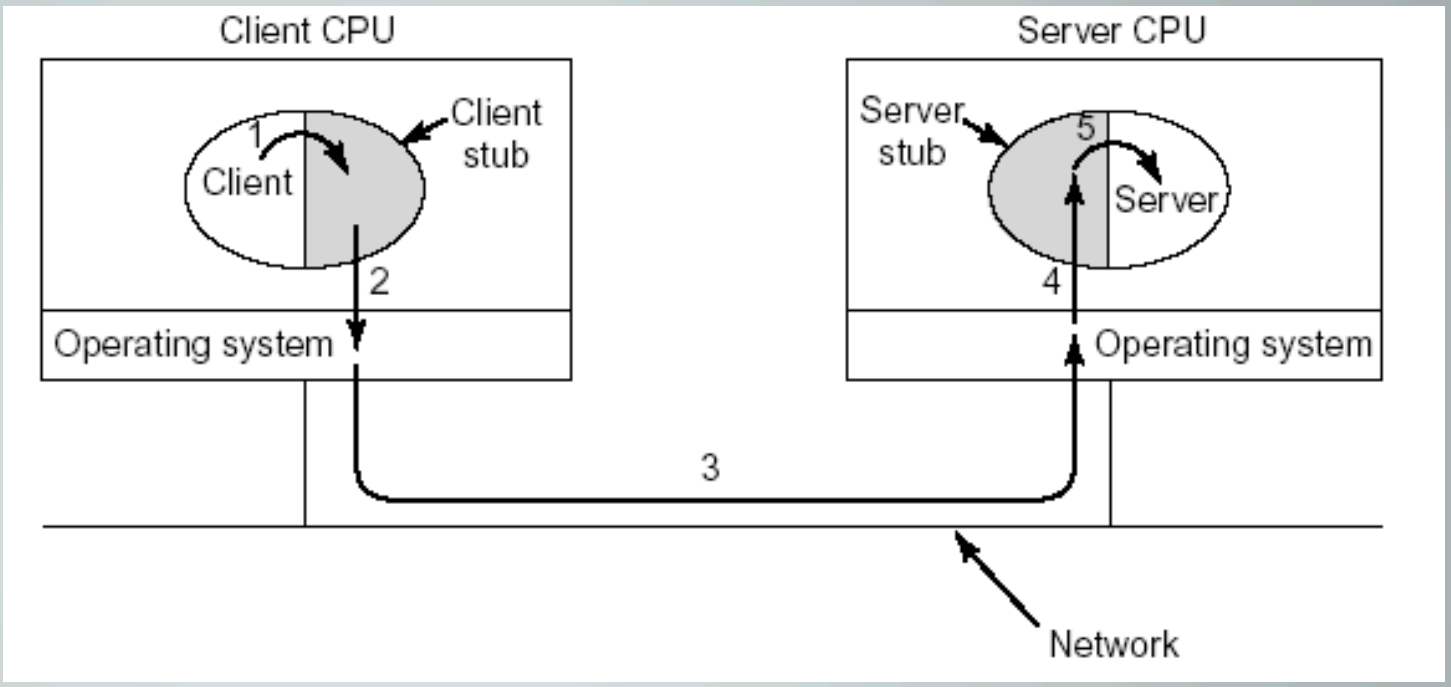


Observación: UDP es simple: sin control de flujo, sin control de errores, sin retransmisiones.

Pregunta: Por que no usar IP directamente?

RPC

UDP se utiliza para hacer accesible un procedimiento a clientes remotos



1) Clientes llaman al procedimiento en un stub local

2) el stub del cliente empaqueta el pedido: todo en un paquete UDP

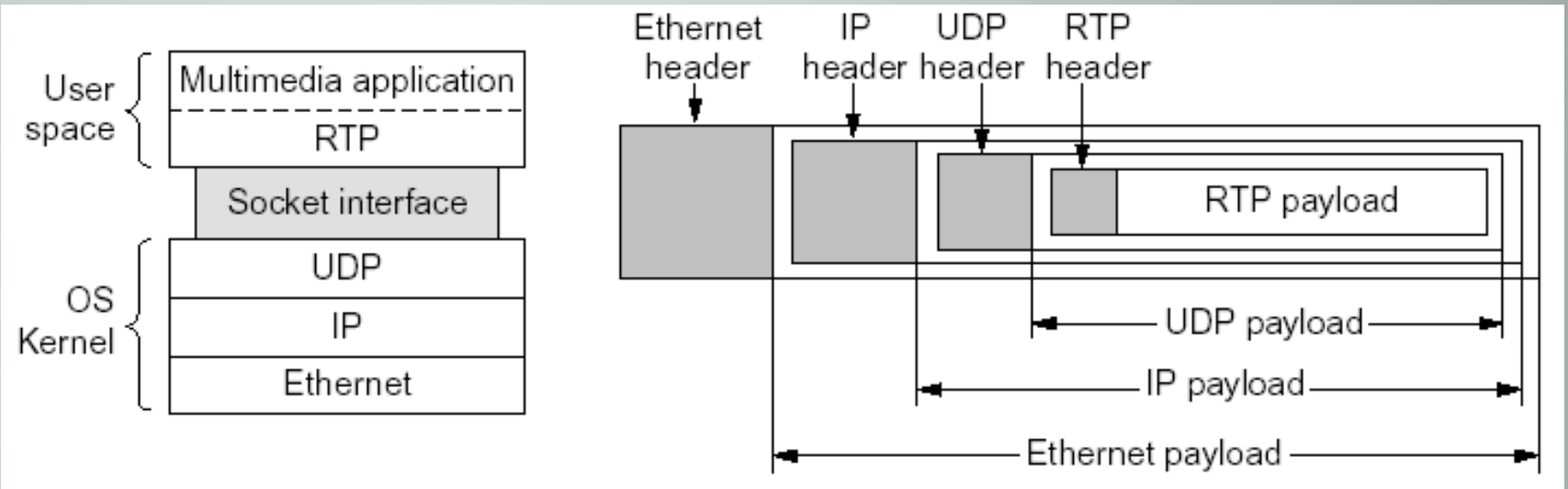
3) el mensaje se transmite a través de la red

4) el stub del servidor desempaqueta el paquete y llama a la implementación local

RTP

Problema: Podemos soportar multimedia streaming?

RTP es el mejor esfuerzo para lograrlo



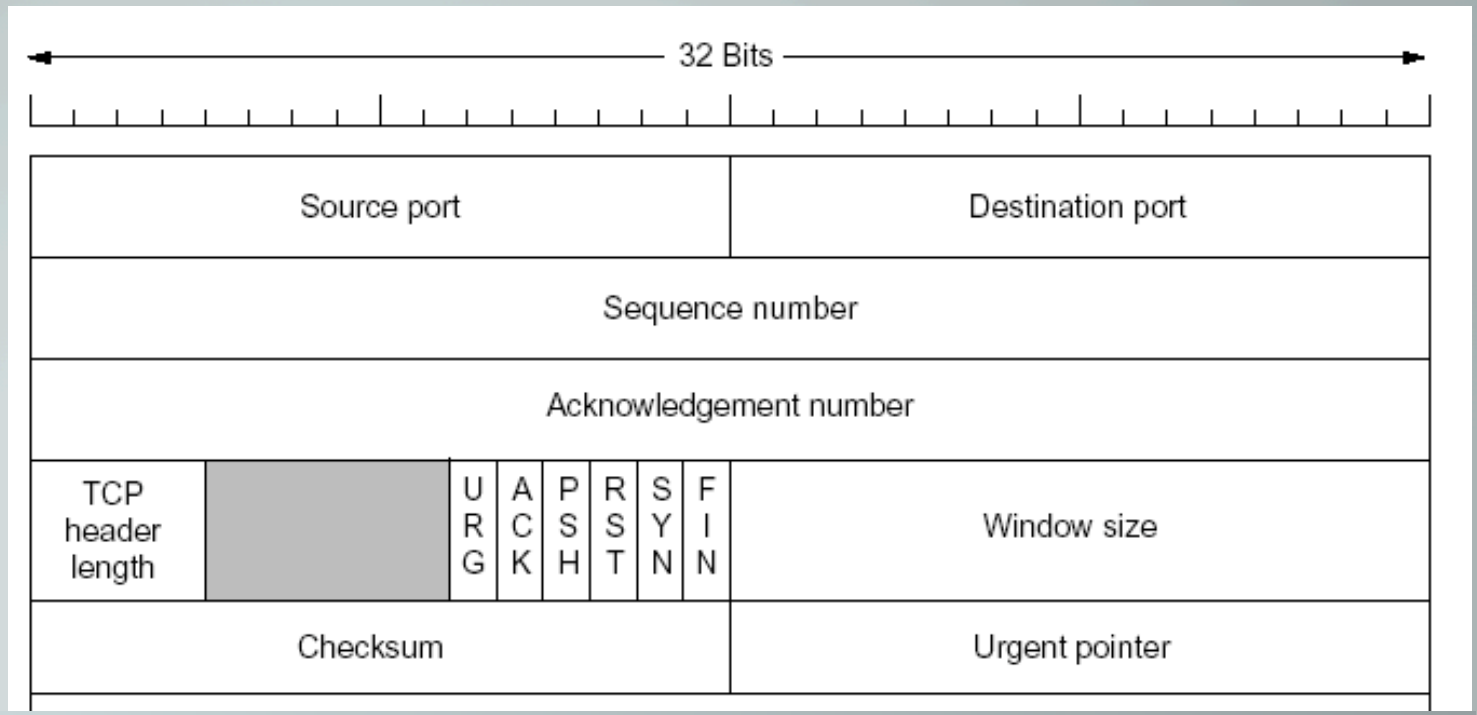
RTP multiplexa un número de de stream multimedia en stream UDP. El receptor es responsable de los paquetes perdidos

Tiempo Real: Paquetes RTP pueden marcarse con tiempo; cada etiqueta indica cuan lejos estan de su predecesor. Este procedimiento permite disminuir "jitter". Ademas, las etiquetas pueden usarse para sincronizar mutliples substreams.

TCP

- Servicio orientado a la conexión que soporta byte streams (no message streams). Un emisor puede enviar ocho paquetes de 512-bytes que son recibidos como dos chunks de 1024 bytes, y uno de 2048 bytes.
- Las direcciones consisten en numero de puertos de 16-bit
- TCP asegura reliablity, conexiones punto a punto. No implementa multicasting ni multiplexing
- Un TPDU de TCP se llama “segment”, que consiste (minimo) un encabezado de 20 bytes y una longitud maxima de 65.535. Un segmento es fragmentado por la capa de red cuando es mas grande que el MTU de la red.

Encabezado TCP



- ACKs son "piggybacked" cuando ACK=1
- SYN es para el setup de al conexión (ACK=0: request - ACK=1: accepted)
- FIN es para liberar conexión. Los datos mandados antes de la liberación no se pierden
- URG indica procesamiento y retransmisión inmediato: el receptor se notifica

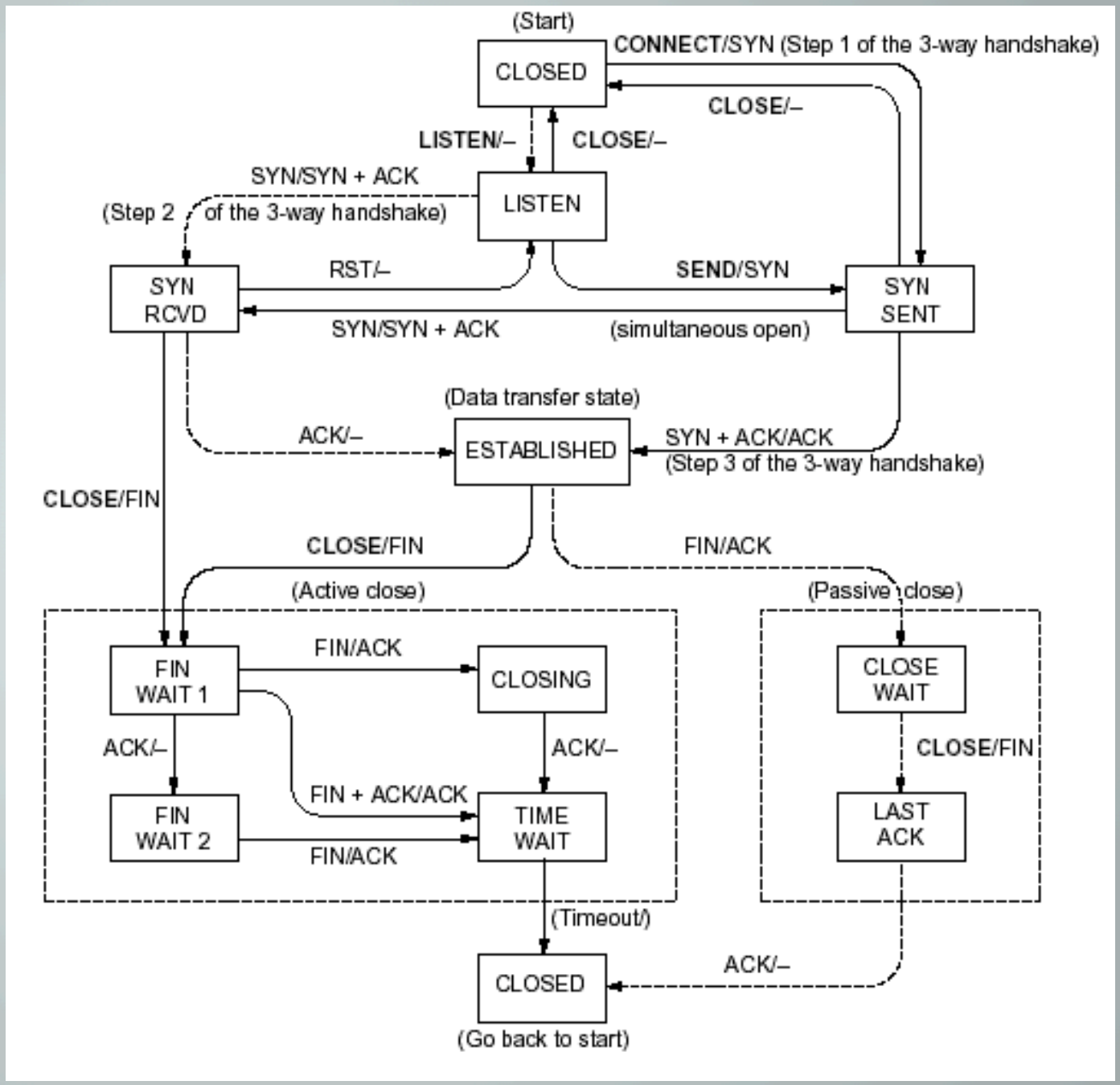
TCP Connection Managment

Establecimiento de la conexión: Usa three-way handshake

Liberación de la conexión: A ser pensado como la liberación de dos conexiones simples independientes.

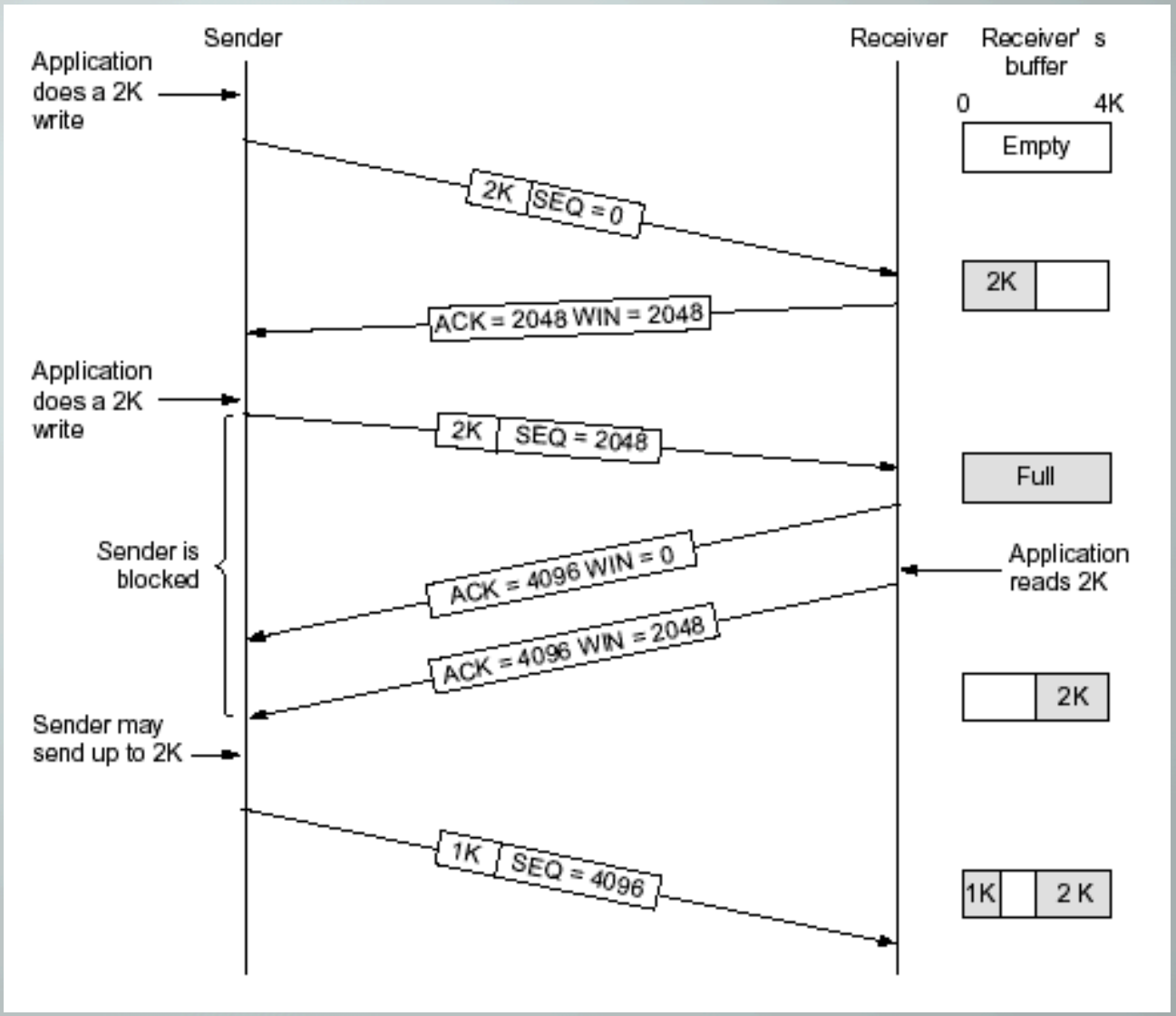
State	Textlist
CLOSED	No connection active or pending
LISTEN	Server waiting for conn. request
SYN RCVD	Conn. request has arrived; wait for ACK
SYN SENT	Conn. request just sent; wait for SYN+ACK
ESTABLISHED	Data can be sent and received
FIN WAIT 1	Client just sent conn. release
FIN WAIT 2	Server just agreed to release connection
TIMED WAIT	Wait for all packets to die
CLOSING	Client & server both tried to close
CLOSE WAIT	Other side initiated release
LAST ACK	Wait for all packets to die

Liberación de conexión



Manejo de Ventana

Idea Básica: El receptor envia un ACK para el siguiente byte que puede ser enviado en stream actual y el número máximo de bytes que se pueden mandar



Manejo de Ventana

Importante: La entidad de TCP no esta obligada a enviar los datos que recibe inmediatamente: puede hacer tanto buffering como quiera.

Ejemplo: Apliaciones orientadas a caracteres interactivas. En lugar de mandar un byte a la vez, bufferean tanto como pueden hasta que el batch anterior es ACK

Ejemplo: Eliminar el sindrome "silly window" donde el servidor lee un byte por vez (y ACKs uno a la vez). En lugar, el receptor debe esperar hasta que pueda recibir una cantidad razonable cantidad de bytes.

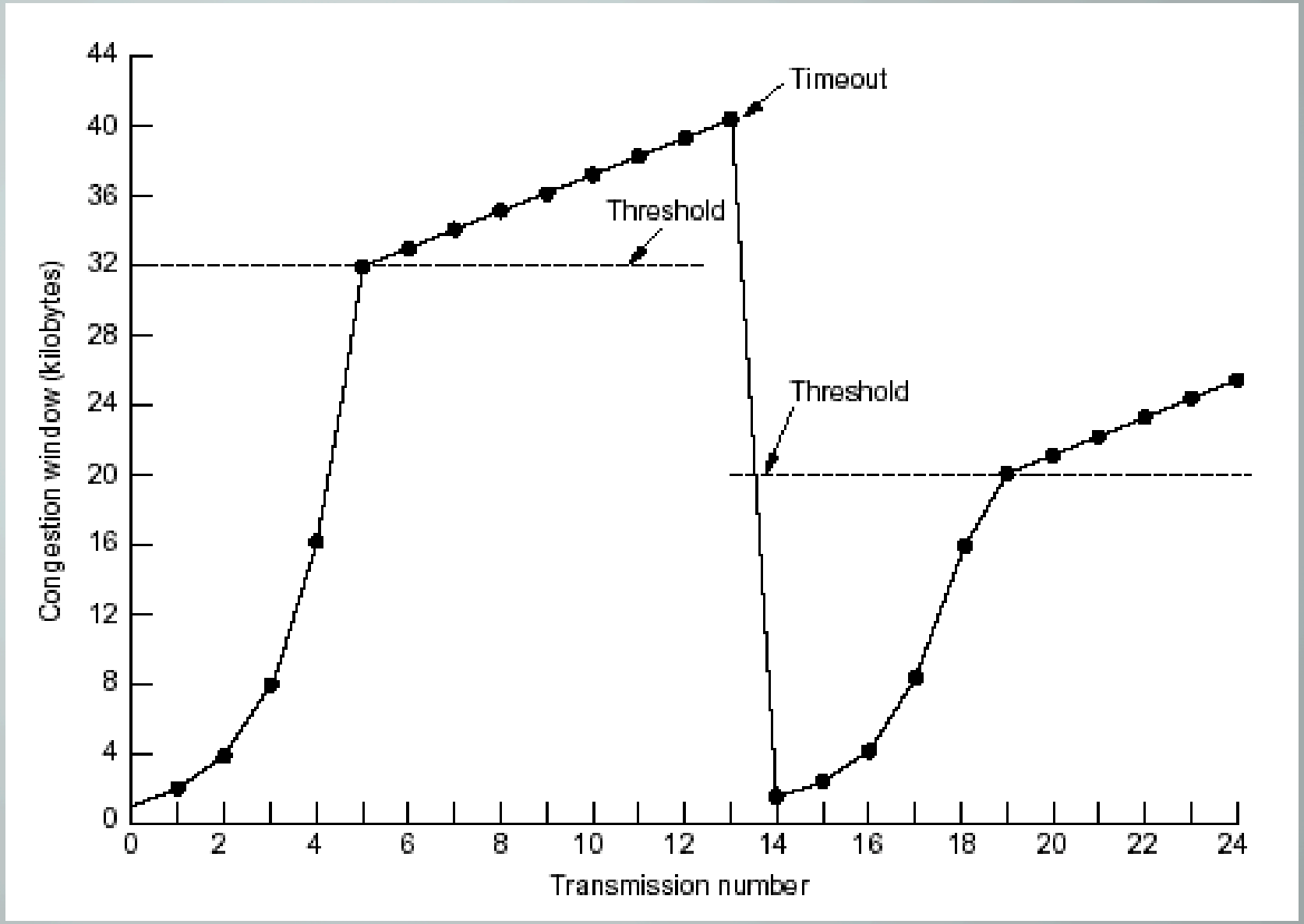
Control de Congestión

Problema: Como detectar congestión y reaccionar?

Solución: Utilizamos una ventana de congestión al lado de la ventana dada por por el receptor. El tamaño real de la ventana es el minimo de la dos.

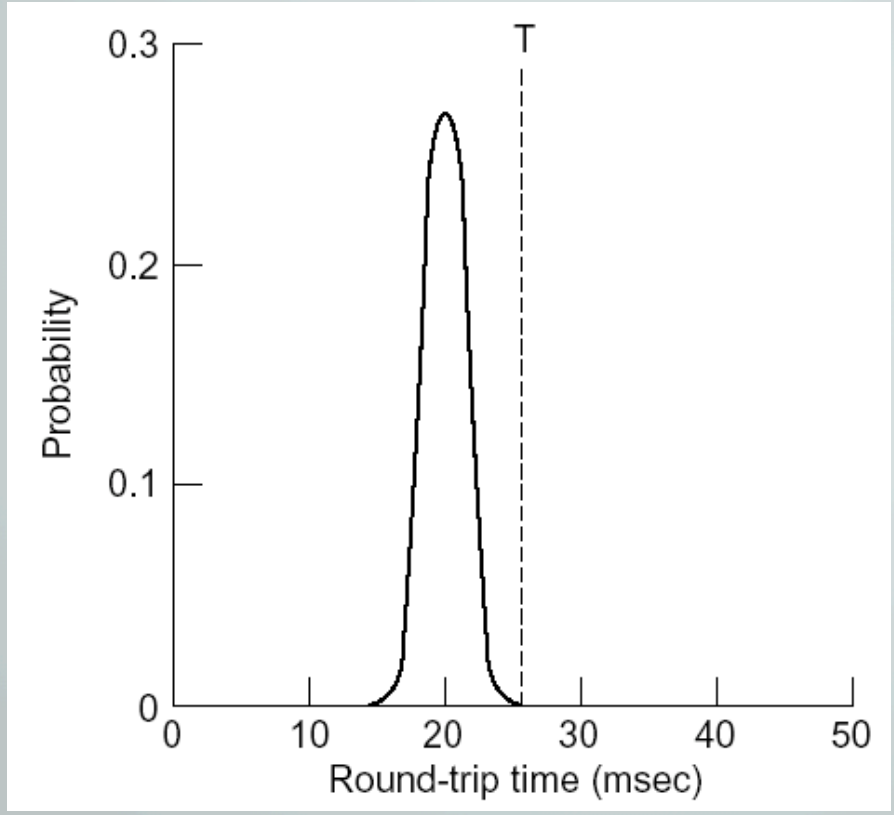
- Inicializar la ventana de congestión al tamaño maximo de segmento a usar. Mandarlo. Si recibimos ACK, duplicarlo. Repetir hasta que recibimos un error.
- Ademas usar un threshold. En un timeout, bajar el threshold al 50% del tamaño de la ventana de congestión, hacer un comienzo lento (exp) hasta nuevo threshold, y agregar maximo tamaño de segmento a la ventana de congestión despues (crecimientos lineal)

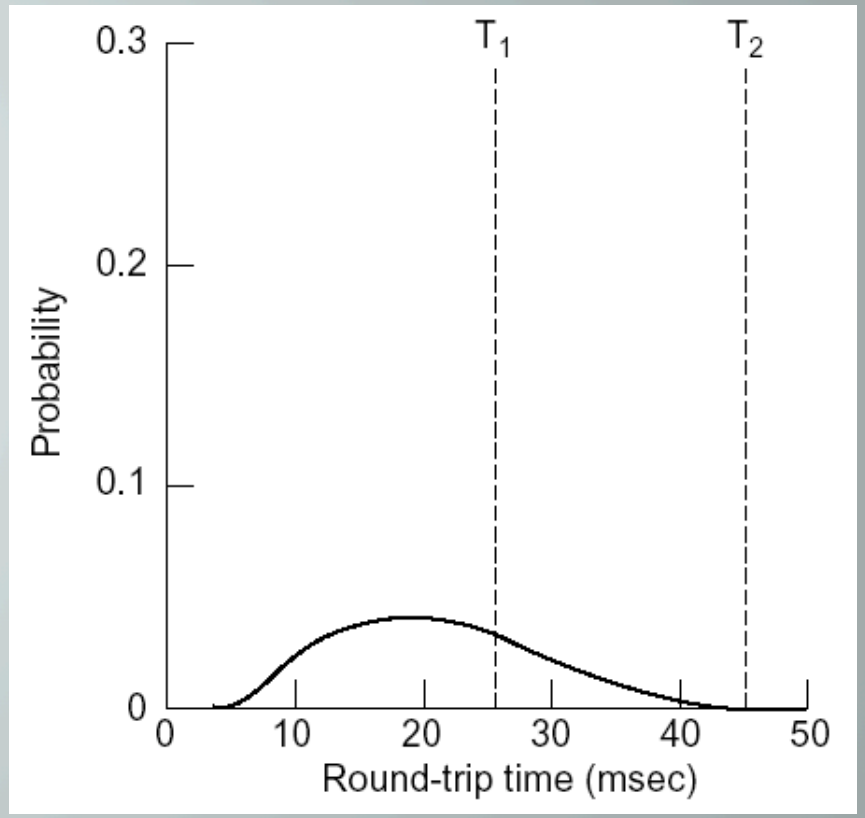
Control de Congestión



Manejo de timer enTCP

Problema Principal: Como determinar el mejor tiempo de timeout para retransmitir segmentos de cara a una gran desviación standard en demoras round-trip?





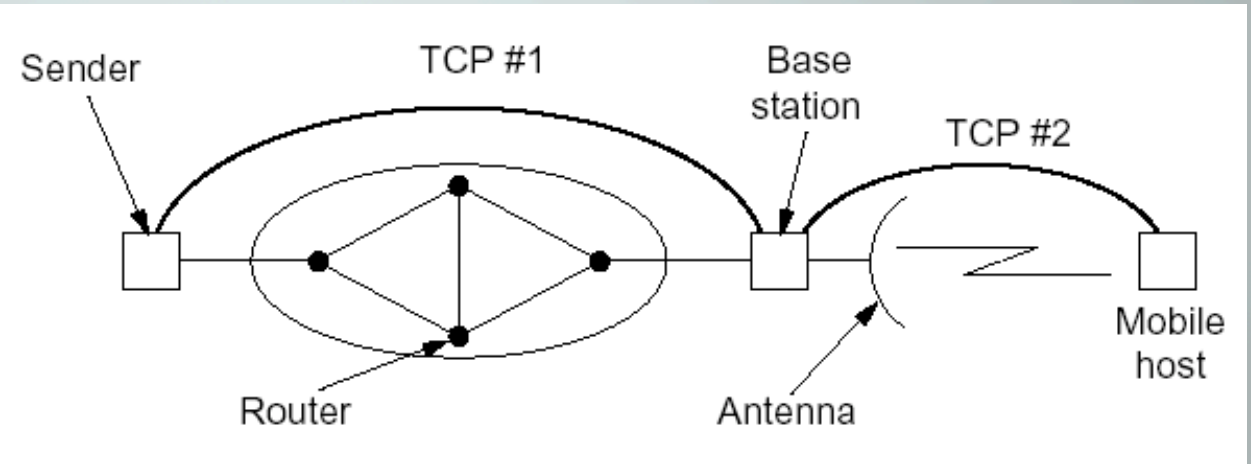
RTT	best current estimate of round-trip delay
D	estimate of deviation of round-trip delays
M	measured round-trip delay

$$
\begin{aligned}
RTT &= \alpha RTT + (1-\alpha)M \\
D &= \alpha D + (1-\alpha)|RTT - M| \\
timeout &= RTT + 4 \cdot D
\end{aligned}
$$

TCP Wireless

Problema: TCP supone que IP esta corriendo sobre cables. Cuando se pierden los paquetes, TCP supone que hay congestión y disminuye los envios. En wireless los paquetes se puede perder por otras causes. TCP deberia hacer lo opuesto, deberia mandar mas paquetes!

Solución 1: Dividir las conexiones TCP en wireless/cableado



Solución 2: Dejar que la base haga parte de la retransmision.