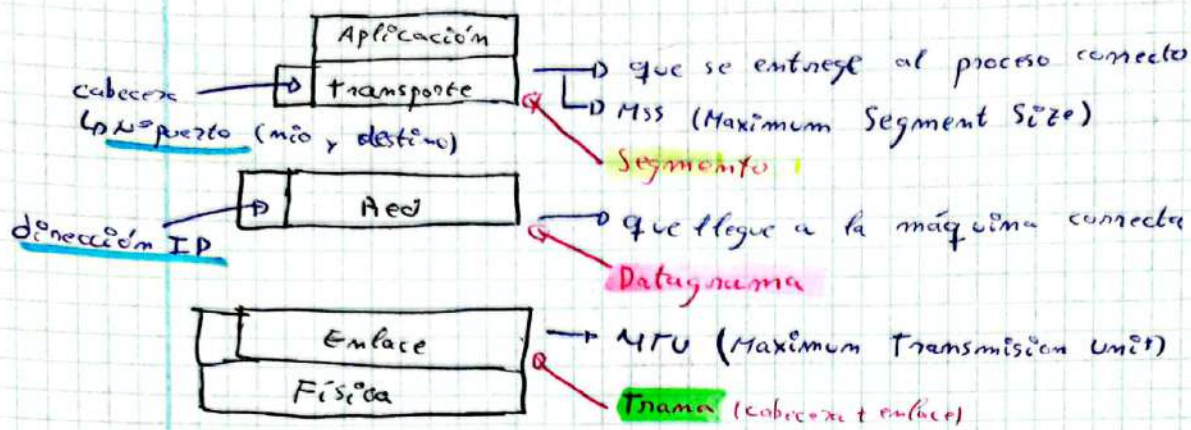


Tema 3

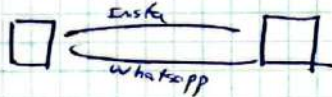
• Redes de datagramas



Transporte (se encarga de)

- 1.- Dividir el mensaje en trozos de tamaño "adecuado"
- 2.- Multiplexar las distintas "sesiones"

Ej 2 comunicaciones entre 2 hosts

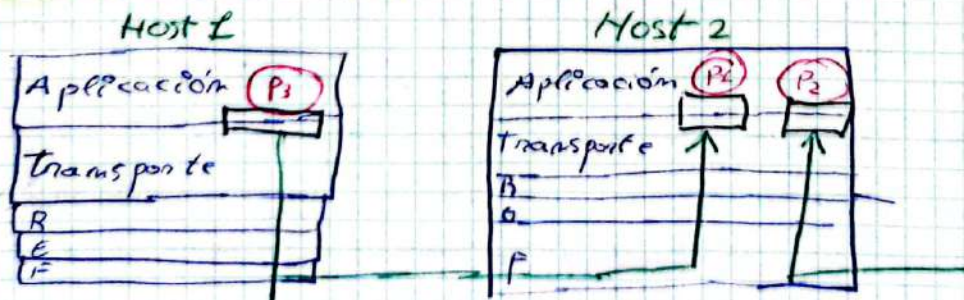


- 3.- Proporcionar una transmisión fiable
- 4.- Controlar el flujo
- 5.- Comprobar la congestión

Multiplexación y demultiplexación

• El N° de puerto se lo asocia el S.O. entre $[0, 2^{16}-1]$

- Multiplexación en el emisor: Reunir datos de múltiples sockets, empaquetarlos con el encabezado
- Demultiplexación en el destino: Entregar segmentos recibidos al socket correcto



○ Procesos
□ Sockets

• El protocolo de Internet para la capa de red se llama IP

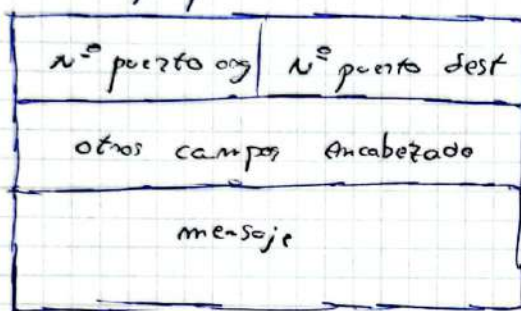
- Se encarga de dar una conexión lógica entre hosts
- Entrega datagramas de un host a otro, pero sin garantías

- ¿cómo funciona la desmultiplexación?

• El host recibe datagramas IP

- Cada datagrama contiene una IP origen y destino
- Cada datagrama lleva un segmento de la capa de transporte
- Cada datagrama contiene un puerto origen y destino

• El host usa IP y n° de puerto para dirigir el segmento al socket apropiado.



Formato de segmento TCP/UDP

• Cuando un host recibe un segmento UDP

↳ comprueba el n° de puerto destino

↳ redirige el segmento al socket con ese n° de puerto

- Desmultiplexación orientada a conexión

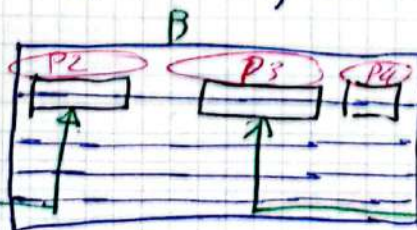
• Un socket TCP se identifica por una 4-upla

- IP origen
- n° puerto origen
- IP destino
- n° puerto destino

• El receptor usa los 4 valores para redirigir el segmento al socket adecuado

• El host servidor debe soportar varios sockets TCP simultáneos

↳ cada uno con sus 4 identificadores



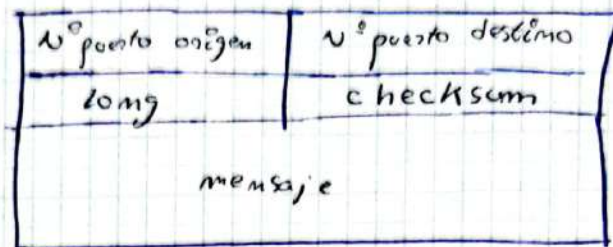
Po:	5775
Pd:	80
IP-O:	C
IP-D:	B

Transporte sin conexión: UDP

• UDP: User Datagram Protocol

- carece de establecimiento de conexión entre emisor y receptor
- cada segmento se trata de forma independiente
- usado para

- streaming
- DNS
- SNMP
- ...



Formato de segmento UDP

Checksum (udp)

objetivo: detectar "errores" en el segmento transmitido
(bits alterados)

- Emisor:

checksum = suma (en C_{256}) del contenido del segmento
↳ se mete en el campo UDP correspondiente

- Receptor:

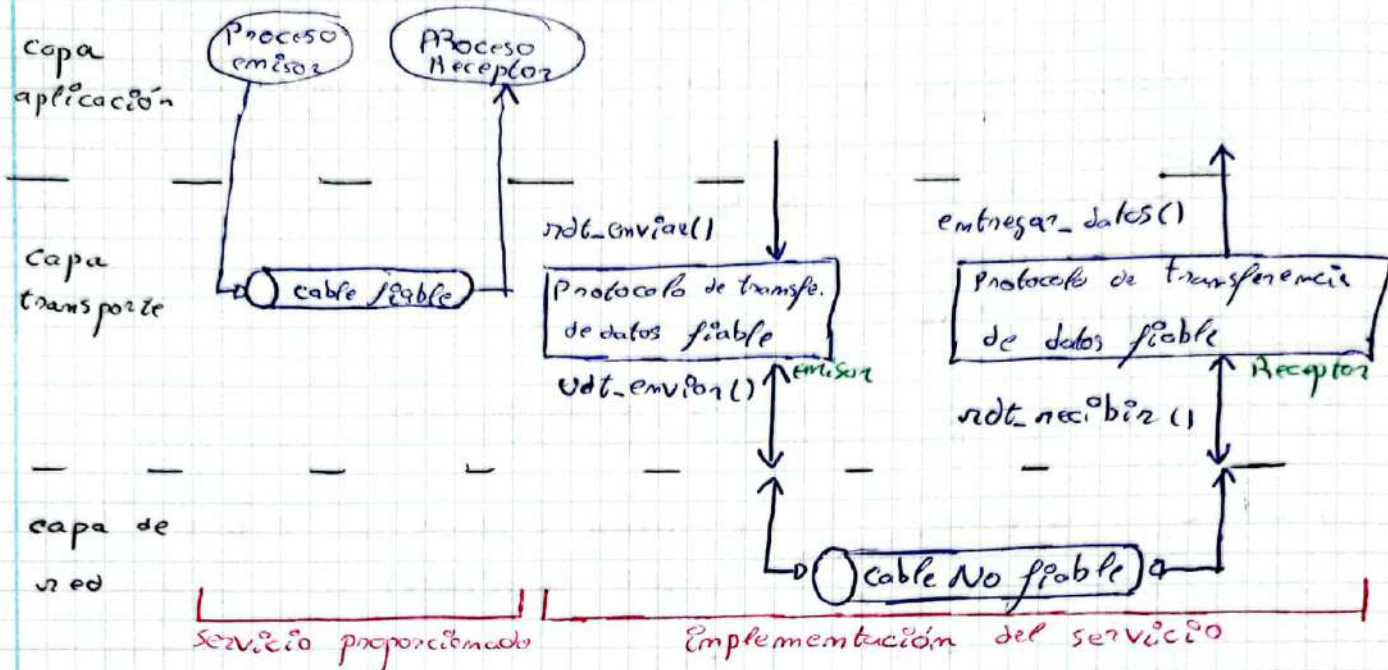
- Calcula el checksum del segmento recibido
 - Comprueba si el valor es el mismo que el del campo
- { No \Rightarrow error detectado
 { Sí \Rightarrow error no detectado

ej

$$\begin{array}{r} 1100110 \\ + 1010101 \\ \hline 10111011 \\ + 1 \quad \quad \quad \rightarrow 1 \\ \hline 0111100 \rightarrow \text{checksum} \end{array}$$

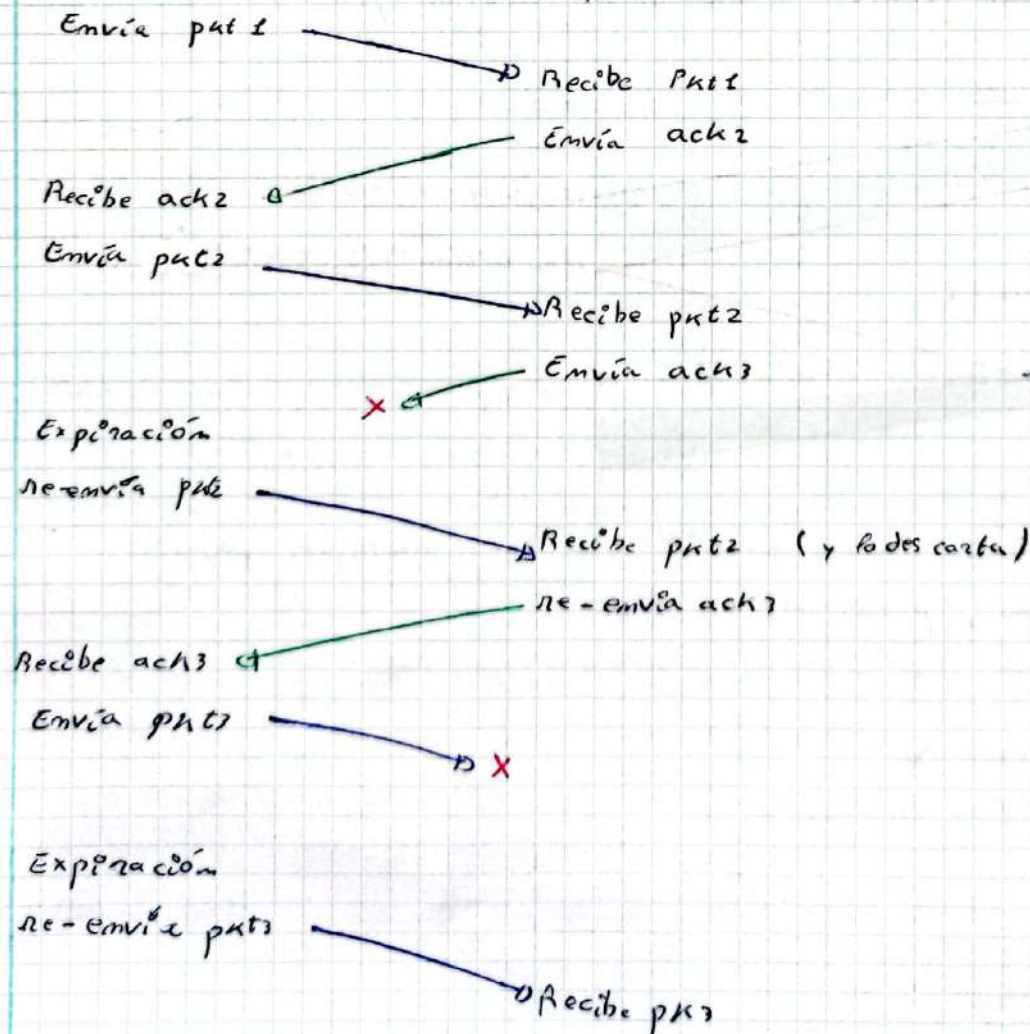
Principio transferencia de datos fiables

- El problema de implementar servicios de transferencia de datos fiables aparece tanto en la capa de transporte, de enlace y de aplicación



- Es responsabilidad del Protocolo de transferencia de datos fiables implementar la abstracción del servicio

ndt 3.0



- Dado que los números de secuencia de los paquetes alternan entre 0 y 1, el protocolo rdt 3.0 se denomina en ocasiones protocolo de bit alternante

- rdt 3.0 funciona, pero el rendimiento es muy malo

ej:

- Enlace de $1 \text{ Gb/s} = 10^9 \text{ b/s}$
- 15 ms retardo prop
- Paquete 8 kb

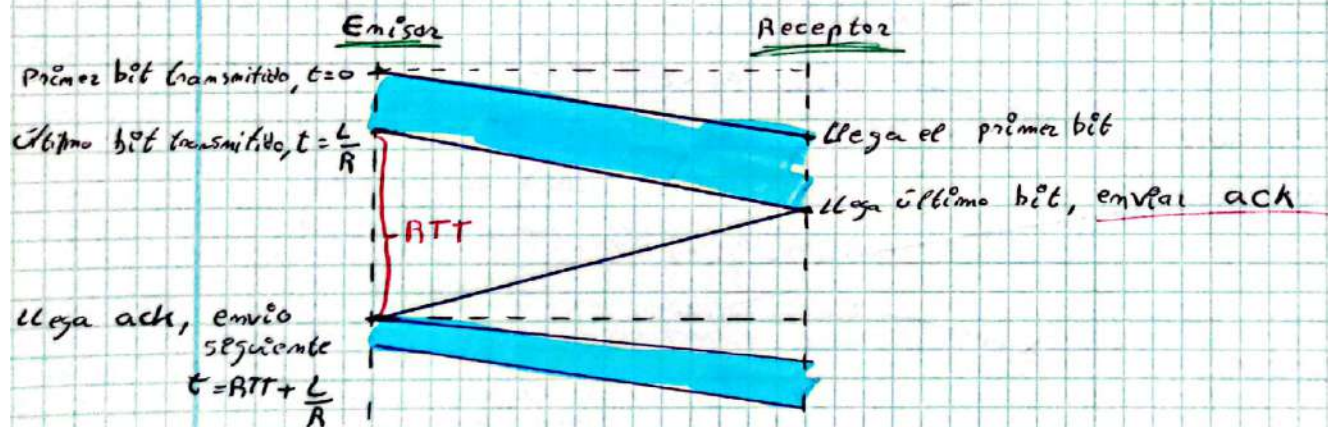
$$d_{\text{trans}} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ b/s}} = 8 \text{ microsegundos}$$

$U_{\text{emisor}} = \text{Utilización emisor}$ (Fracción de tiempo que el emisor está ocupado emitiendo)

$$U_{\text{emisor}} = \frac{L/R}{RTT + \frac{L}{R}} = \frac{0.008}{30.008} = 0.00027$$

• Si $RTT = 30 \text{ ms}$, 1 paquete de KB cada 30 seg $\Rightarrow 33 \text{ kb/s}$ de 1 Gb/s

(El protocolo limita el uso de los recursos físicos)



- Segmentación: El emisor permite que haya múltiples paquetes "en camino", pendientes de ACK
- Hay 2 formas gemélicas de protocolos segmentados!

- Retroceder N
- Repetición selectiva

• Al segmentar

$$U_{emisor} = \frac{3 \times (L/R)}{RTT + \frac{L}{R}} = \frac{0.024}{30.008} = 0.0008$$

(0 Mejora)

Retroceder N

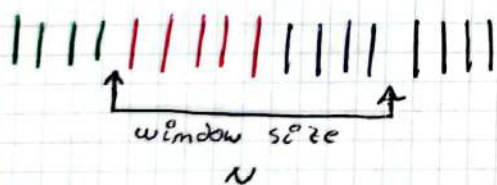
- El emisor puede tener hasta N paquetes pendientes de ack
- El receptor sólo envía ack's acumulativos
 - No lo envía para un paquete si hay una laguna
- Emisor tiene un temporizador para el paquete más antiguo sin ack
 - si llega a 0, se retransmiten paquetes

Repetición selectiva

- El emisor puede tener hasta N paquetes pendientes de ack
- El receptor envía ack para cada paquete
- El emisor mantiene un temporizador para cada paquete sin ack
 - si llega a 0, retransmite solo paquete sin ack

• Retroceder N (GBN)

- N° de secuencia de k bits en cabecera del paquete
- Ventana de hasta N paquetes consecutivos sin ack



- | enviado, sin ack aún
- | enviado, con ack
- | sin enviar
- | sin definir

• temporizador para cada paquete en camino

• timeout(m): Retransmitir paquete m y siguientes en la ventana

Repetición selectiva (SR)

- El receptor envía ack individual para cada paquete
- El emisor sólo reenvía los faltantes de ack
 - Un temporizador para cada paquete

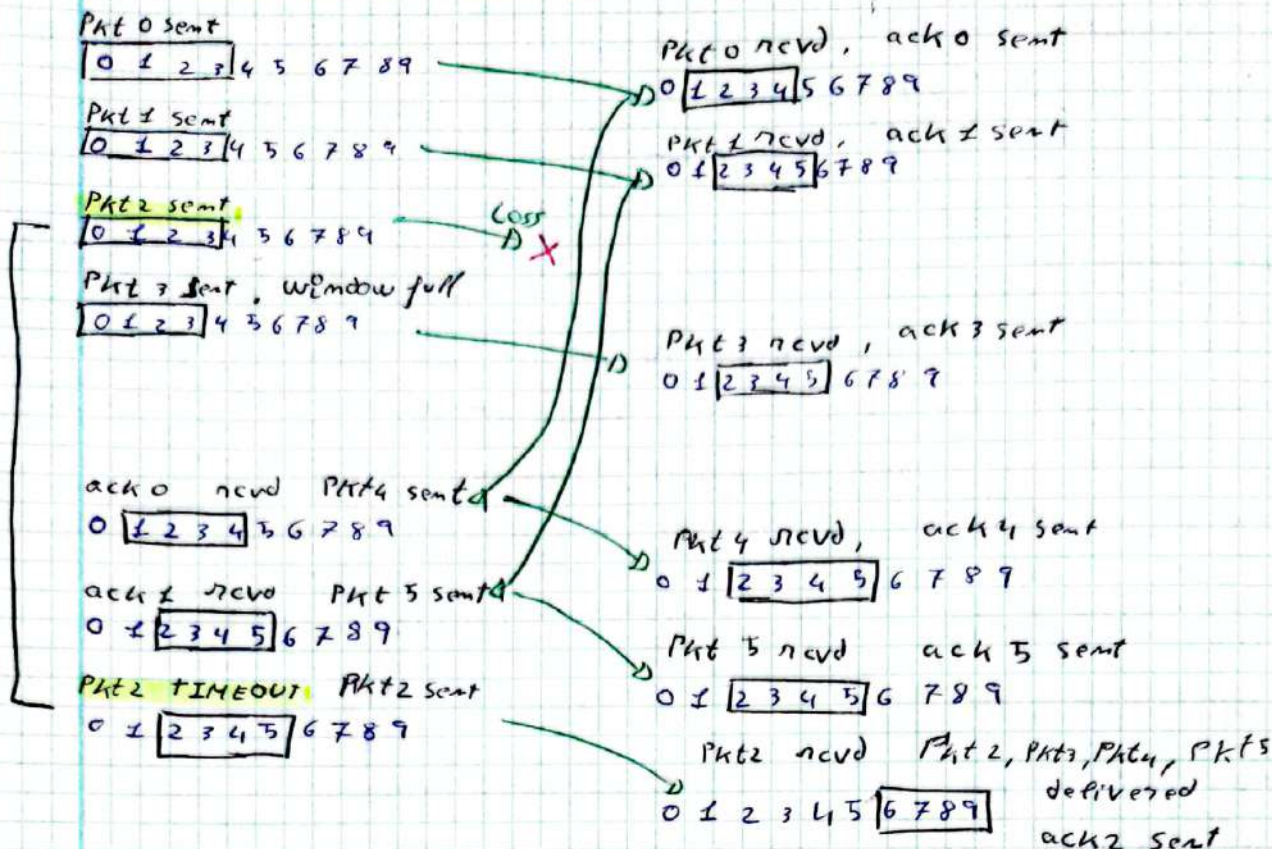
• Emisor

- timeout(m)
- ack(m)

(Marca paquete m como recibido)

• Receptor

- paquete m en $[rbase, rbase + N - 1]$
- Envía ack(m)



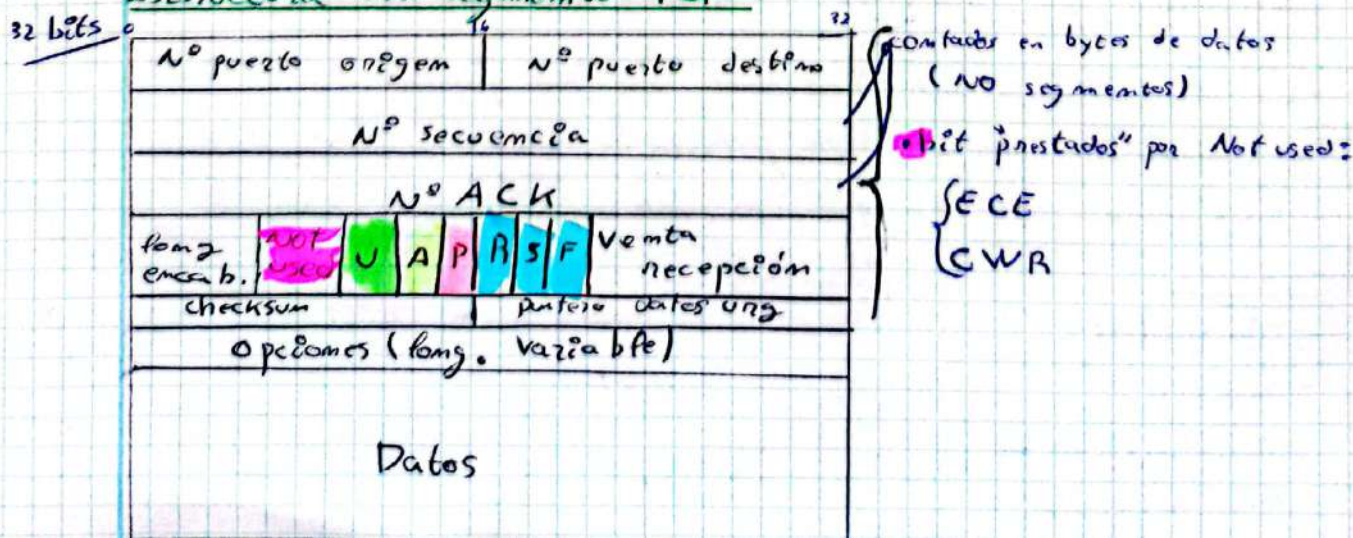
Transporte orientado a conexión: TCP

- Punto a punto (un emisor y un receptor)
- Flujo de bytes fiable, en orden
- Segmentado

El control de flujo y congestión de TCP fijan el tamaño de la ventana

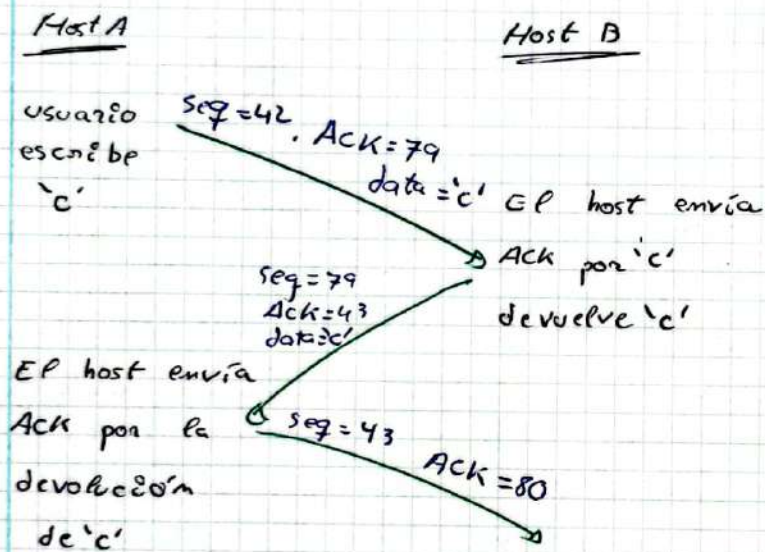
- Buffers de emisión y recepción
- Datos full duplex
 - Flujo de datos bidireccional en la misma conexión
 - MSS (Máximo tamaño segmento)
- Orientado a conexión
 - Inicializa estados antes del intercambio de datos
- Control de flujo
 - Emisor NO satura al receptor

Estructura del segmento TCP



- **URG** (datos urgentes)
- **ACK** (nº ACK válido)
- **PSH** (Push)
enviar estos datos ya a la aplicación
(El programador no puede programarlo)
- **RST, SYN, FIN**
establecer conexión
- ventana recepción
nº bytes que receptor está dispuesto a aceptar

c/u



• Gestión de conexión

Establecimiento en 3 pasos

• Paso 1

- El cliente envía segmento **SYN** al servidor
- (Especificar nº secuencia inicial) (sin datos)
↳ Aleatorio

• Paso 2

- Servidor recibe SYN responde con segmento **SYNACK**
- Servidor crea buffers
- Especifica nº secuencia inicial servidor
↳ Aleatorio

• Paso 3

Cliente recibe SYNACK, responde con segmento ACK, (puede contener datos)

CONEXIÓN

(cliente socket, server);

• Paso 1

El cliente envía el segmento de control TCP FIN al servidor

• Paso 2

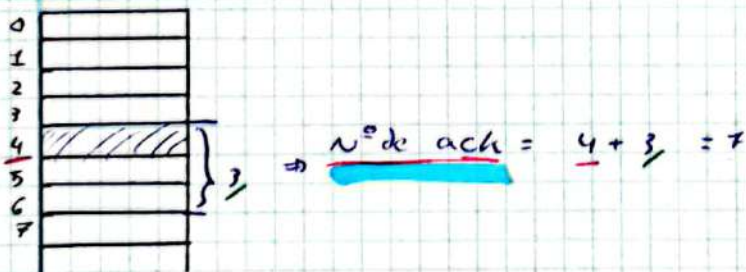
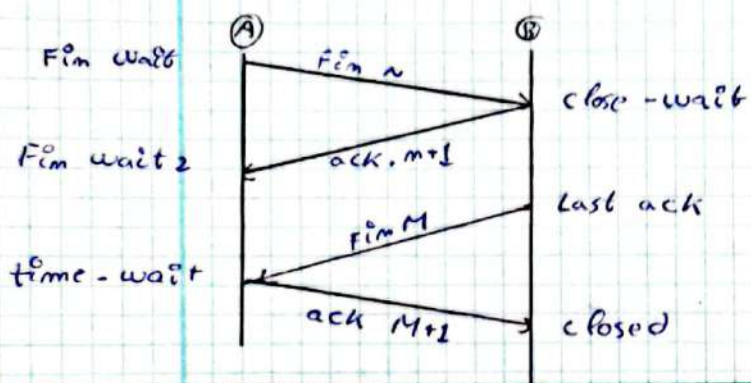
Servidor recibe FIN, responde con ACK,
Cierra conexión y envía FIN

• Paso 3

Cliente recibe FIN, responde ACK
(espera tiempo)

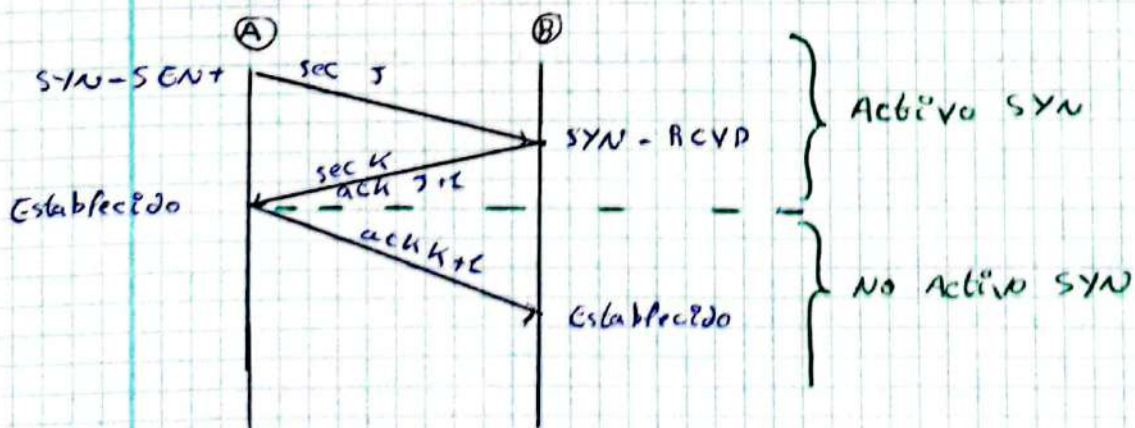
• Paso 4

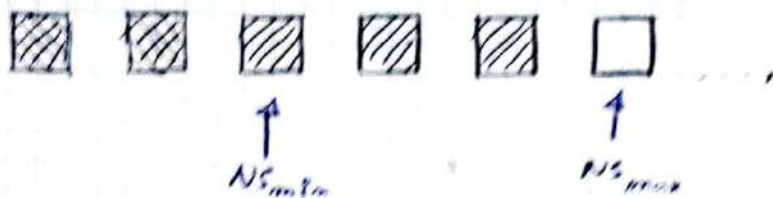
Recibe ACK, conexión cerrada



• N° de secuencia al inicio de una conexión es aleatorio

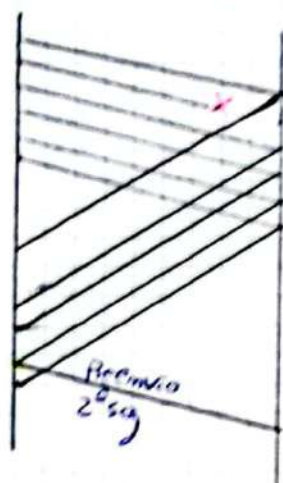
• Acuerdo en 2 pasos





- Ns_{min} : Núm. secuencia más baja sin reconocer
 send base
 Ns_{max} : Next seq Num (primer segmento no usado)

Recepción de pds



• Recvto 1, entonces se que tengo que recibir

- Generación de ack (Receptor)
- Llegada segmento ordenado. ACK retardado. Espera 500ms
 N^o sec esperando. → al siguiente segmento.
 Anterior, ya tienen ack. Si no llega, envía ACK
- Llega segmento ordenado. → Inmediatamente envía ACK
 hay otro segmento a falta de ACK acumulatorio para ambos
- Llega segmento fuera de orden mayor que el esperado (fuera). → Inmediatamente envía ACK duplicado
 Indicando el byte esperado
- Llega segmento que completa parcialmente una figura. → Inmediatamente envía ACK

Algoritmo de Nagle

- Recibir varios datos de una aplicación y mandarlos todos juntos enviados

- Llega datos, ACK's pendientes → Acumula datos en buffer emisor
- Llega ACK pendiente → Inmediato envía datos del buffer acumulados
- Llega datos, No ACK's pendientes → Inmediato envía datos
- Llega datos, No hay espacio en el buffer → Inmediatamente envía datos si lo permite la ventana.

Control de flujo (TCP)

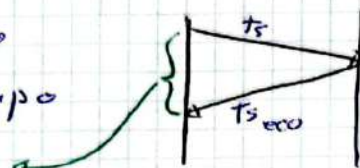
- El receptor tiene un buffer de recepción
- Equilibra la velocidad de envío a la que la aplicación vacía el buffer de recepción
- El receptor anuncia el espacio libre en "RcvWindow" en los segmentos
- El emisor limita los datos sin ACK a "RcvWindow"

El emisor no satura el buffer del receptor a base de enviar mucho muy seguido

Estimar RTT

T_s = sello de tiempo

$T_{s\text{eco}}$ = sello de tiempo



RTT muestra (i)

$$RTT_{\text{estimado}}(i+1) = RTT_{\text{muestra}}(i) \cdot \alpha + RTT_{\text{estimado}}(i) \cdot (1-\alpha)$$

$$(0 \leq \alpha \leq 1) \quad \alpha = \frac{1}{8} \text{ normalmente}$$

$$RTT_{\text{diferencia}}(i+1) = |RTT_M(i) - RTT_E(i)| \cdot \beta + RTT_D(i) \cdot (1-\beta)$$

$$(0 \leq \beta \leq 1) \quad \beta = \frac{1}{4}$$

$$RTT = RTT_E + 4 \cdot RTT_D$$

- Cuando está activo SYN

ws = tamaño de ventana

↳ Indica por cuanto hay que multiplicar la ventana de recepción

$$V_{\text{recepción}} = 2^{ws} \cdot V_{\text{cabecera}}$$

- Receptor anuncia espacio en la ventana si su tamaño es M_s o igual a la mitad de su memoria

Control de congestión

- Demasiadas fuentes envían demasiados datos demasiado deprisa para que la red lo pueda asimilar

• Síntomas

- Pérdida de paquetes
- Grandes retardos

• 2 formas de abordarlo

⊖ Control de terminal a terminal

- Se deduce la congestión
- No hay realimentación explícita de la red

⊕ Control asistido por la red

- Routers proporcionan realimentación a los terminales ^(del estado de la congestión)
 - Un bit indica la congestión

- indicación explícita de la tasa a la que el emisor debería enviar

- caso ABR (Available Bit Rate)

• "servicio elástico"


• si la ruta está "infra-cargada"

- Emisor se debe usar ancho de banda disponible

• si está sobrecargada

- Emisor limita a la tasa garantizada

- Célelos RM (Resource Management)

• Envío por el emisor intercalado en datos 

• Los bits de RM lo relleman los switches

- bit NI: No hay mejora en la velocidad (congestión suave)

- bit CI: Indica congestión

• bit CFI = 1 si switch está congestionado

⌊
el receptor pone CI en RM=1 y se lo devuelve al emisor

control congestión TCP

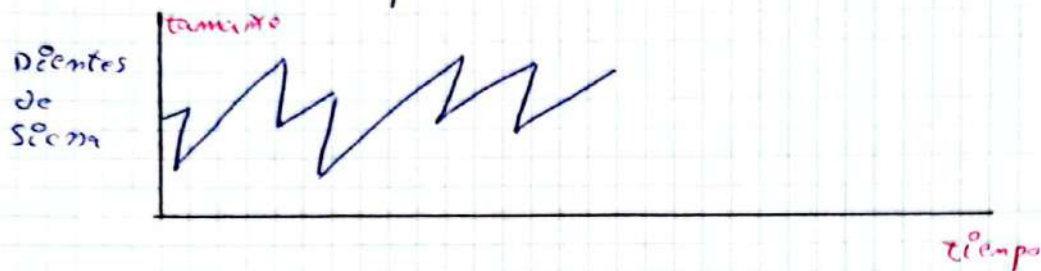
- Incremento aditivo, decremento multiplicativo
- Sondar el ancho de banda accesible, hasta que hay pérdidas (Incrementos léase de transferencia [tamaño Ventum])

Incremento aditivo

- Incrementar $cwnd$ en 1 MSS cada RTT hasta haya pérdidas

Decremento multiplicativo

- Dividir $cwnd$ por 2 cuando las haya



- emisor limita su transmisión:

$$\text{Last Byte Sent} - \text{Last Byte Acked} \leq \underbrace{\min \{ \text{Vent}_{\text{congestión}}, \text{Vent}_{\text{recepción}} \}}_{cwnd}$$

- a grosso modo

$$\text{tasa} = \frac{cwnd}{RTT} \text{ Bytes/s}$$

- $cwnd$ es dinámica en función de la congestión de la red

¿cómo percibe el emisor la congestión?

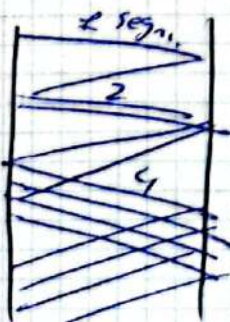
- evento de pérdida = expiración o 3ACK duplicados
- Emisor reduce la tasa ($cwnd$) tras un evento de pérdida

3 Mecanismos

- Arranque lento

Empezamos la conexión y la vamos incrementando hasta la primera pérdida

- Al inicio $cwnd = 1 \text{ MSS}$



- 3 acks duplicados
 $\frac{cwnd}{2} \Rightarrow$ la ventana ya crece lentamente
- pero, tras una expiración
 $cwnd = 1$
 La ventana crece exponencialmente
 Luego lineal

¿Cuándo pasamos de exponencial a lineal?

Cuando $cwnd$ llegue a $\frac{1}{2}$ de su valor antes de la expiración

eficiencia TCP

¿Cuál es la tasa media de TCP en función del tamaño de ventana y RTT?

- Sea W el tamaño de la ventana cuando ocurre la pérdida

La tasa es $\frac{W}{RTT}$

Después la pérdida

$$ventana = \frac{W}{2}$$

$$tasa = \frac{W}{2RTT}$$

$$tasa\ media = 0.75 \frac{W}{RTT}$$

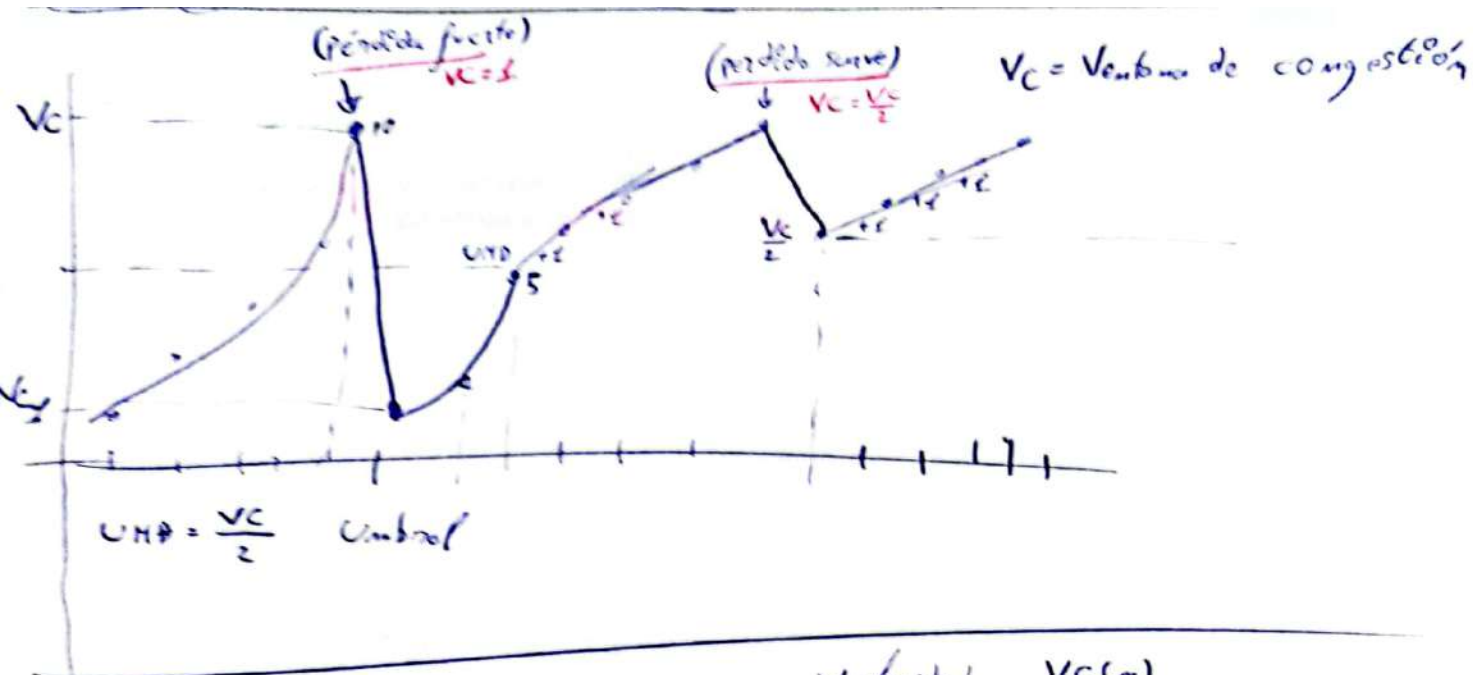
ej

- Segmentos de 1500 bytes
- RTT 400ms
- Se quiere tasa de 10 Gb/s

- Requiere tamaño ventana $W = 83.33$ segmentos en tránsito

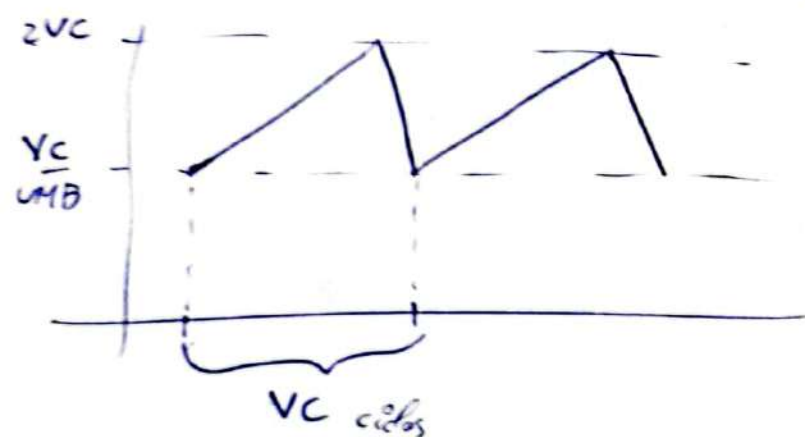
$$tasa\ transferencia = \frac{1.22 \cdot MSS}{RTT \sqrt{2}}$$

$$L = 2 \cdot 10^{-10} \text{ (tasa de pérdida muy baja)}$$



$$\text{Velocidad en 1 ciclo} = \frac{VC(m)}{RTT} \cdot MSS$$

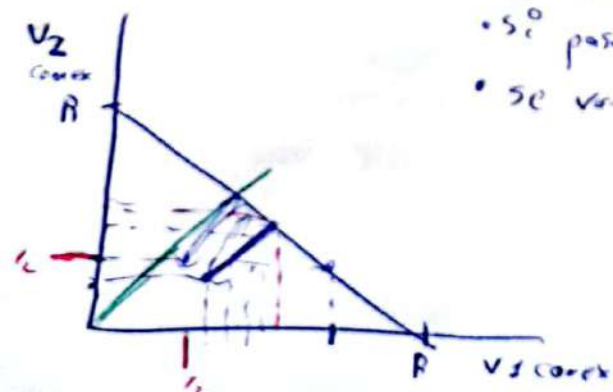
$$\bar{V} = \frac{\bar{VC} \cdot MSS}{RTT} = \frac{3VC}{2} \cdot \frac{MSS}{RTT}$$



$$L = \text{tasa de pérdidas} = TP = \frac{1}{VC \cdot RTT}$$

Equidad TCP

- R velocidad máxima posible
- Si pasan los \, se produce pérdida
- se van equilibrando hasta llegar a la mitad de \

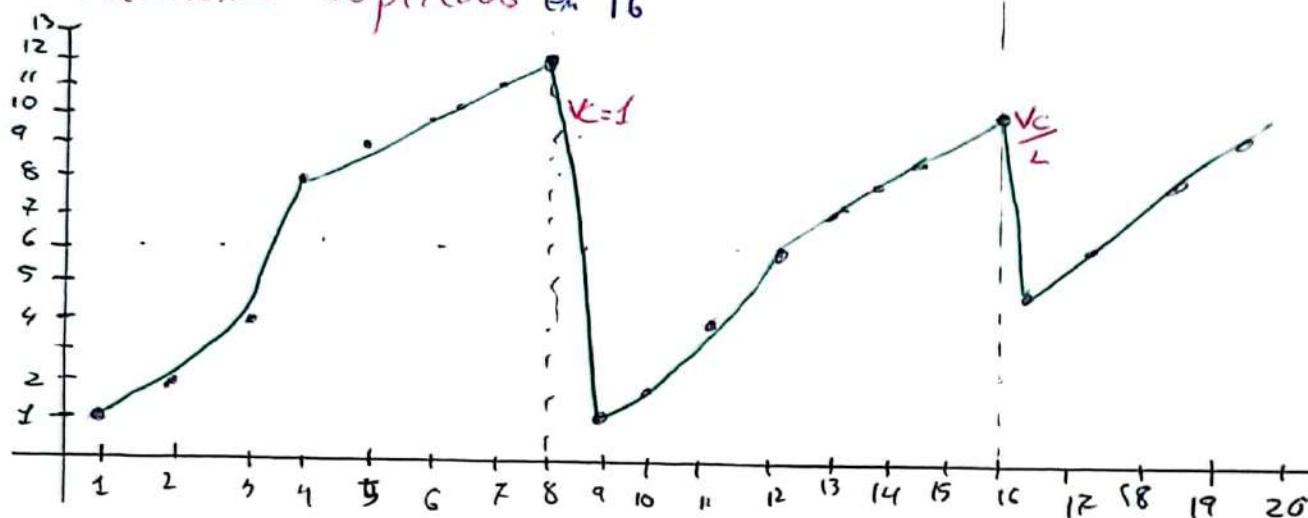


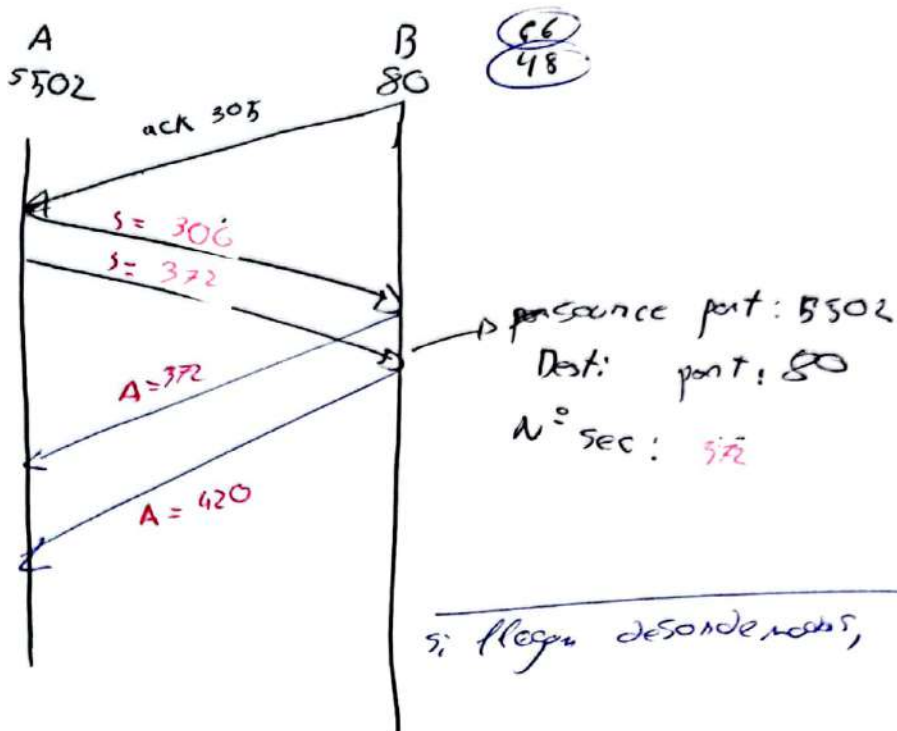
$$RTT_e = RTT_H(\rho) \cdot \alpha + RTT_E(\rho) \cdot (1 - \alpha)$$

$$0 \leq \alpha \leq 1 \quad \alpha = \frac{1}{8}$$

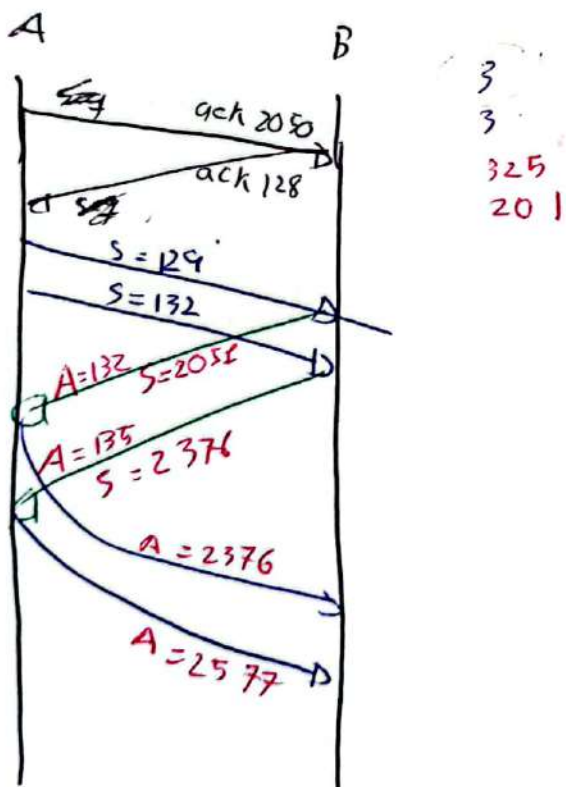
1

- Arranque lento
- Umbrel inicial = 8 = V_C
- Pêndula paquete en 8
- Reconocimiento duplicado en 16





Source 80
dest 5502
n° seq 306



A → B 8 segmentos consecutivos, se pierde el 2

GBN (Retraceder N)

- Envía 9 segmentos, 1 2 3 4 5, luego reenvía 2 3 4 5

- B envía 8 reconocimientos

 - Primeros van 4

 - luego los correspondientes a 2 3 4 5

- SR (Repetición selectiva)

- Envía 6 segmentos 1 2 3 4 5, luego el 2

- B envía 5 reconocimientos, 1 3, 4, 5

 - luego el 2

- TCP

- Envía 6 segmentos 1, 2, 3, 4, 5, luego el 2

- B Envía 5 reconocimientos, 4 con número de reconocimiento 2

- y luego el último con 6