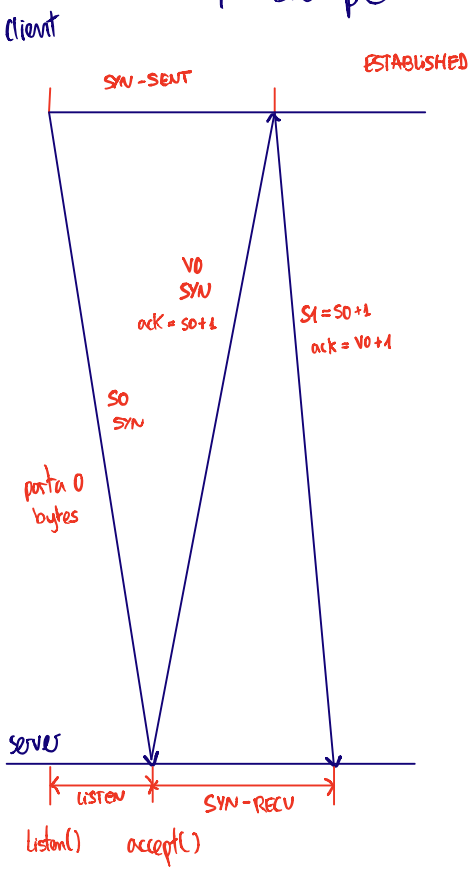
ACK : 0 primer envío

1 siempre

FLAG ACK → indica si se usa ack



MSS : Maximum Segment Size [B]

TCP header size : 20 ~ 60 Bytes

MSS = MTU - 40

TWH

Handshaking (3-way):

- payload = 0
- SYN and FIN consume 1 seq.num
- Initial seq. num is RANDOM

Options:

IPv4 + TCP headers ↓

- Max Segment Size (MSS): MSS = MTU - 40 ← (Solo se envía durante TWH)
- Window Scale Factor: awnd · 2^{WSF} WSF = #bits (if (awnd > 2^{16}) usamos WS para poder llegar)
- Timestamp: used to compute RTT. (envía hora ⇒ envía echo ⇐) 10 bytes
- Selective ACK (SACK): envía acks solicitando segmentos perdidos y evitar repetidos

TCP Sequence Numbers:

- Sequence number points the first payload byte.
- SYN and FIN consume 1 byte.
- Ack number points the next missing byte.

TCP congestion control

finestra: #bytes que es poden enviar sense confirmar

↓

wnd = min(awnd, cwnd)

- awnd: espacio libre en el buffer del receptor (en las cabeceras va enviando awnd)
- cwnd: ventana en función de la congestión (interna en cada socket y gestionada por el SO)

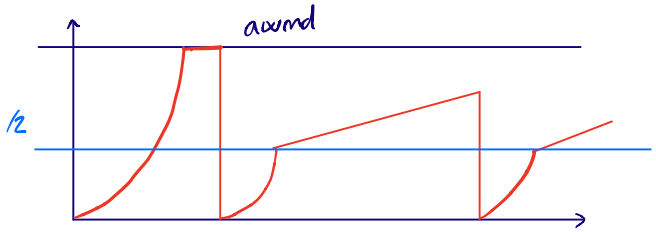
  [segmentos] $wnd(seg) = \frac{cwnd(B)}{MSS}$

cwnd = 1

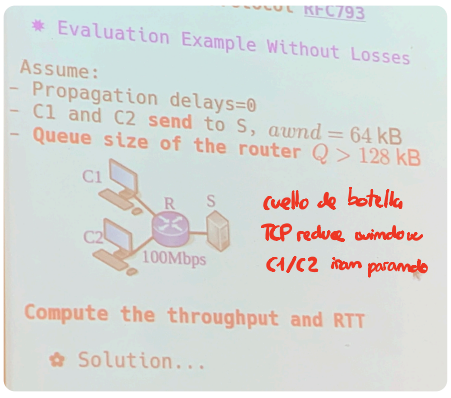
↓ Tx recibe ACK de nuevos datos

```
if( cwnd < ssth) ++cwnd;
else  cwnd += 1/cwnd;

if ( RTO ==0) // ha habido pérdidas, falta algun ACK por recibir
   Retx // Se reenvia el último segmento de la cola (no confirmado)
   cwnd = 1
   ssth = max ( cwnd/2 , 2),
```



EJERCICIO ↴



Throughput $C_x \overset{[v_1 = v_2]}{=} \frac{100}{2} = 50$ Mbps

RTT: tiempo entre enviar segmento y recibir confirmación de este
- en régimen estacionario: cola router ≈ 128 kB (full)
  - entra en cola
  - cola se vacía (128 kB)
  - envía ack

$RTT = \frac{128 \cdot 10^3 \cdot 8 bits}{100 \cdot 10^6} \cdot 10^3 \overset{(ms)}{=} 10'24$ ms

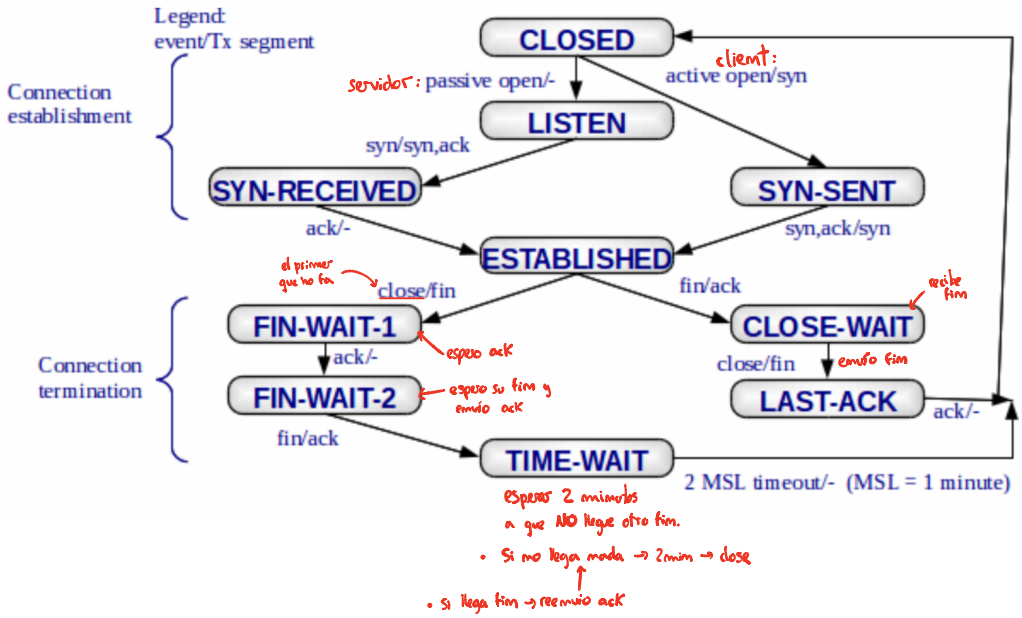
$V_{ef} = \frac{\overline{W}}{RTT} = \frac{64 \cdot 10^3 \cdot 8\ bit}{10'24 \cdot 10^{-3}} = 50$ Mbps

coincidencia?

RTO: Retransmission Time-Out

- Actiu des de l'últim ack confirmat.
- Només hi ha un temporitzador per socket
- Càlcul:
  - servidor estima RTT amb l'últim a ack de confirmació.
  - RTO = srtt + 4·rttvar
    - srtt: mitjana
    - rttvar: variança
  - Si es retransmet algun paquet (RTO = 2·RTO)

3.4.14 TCP State diagram



UDP

AKA: Puta basura → no reliable, no error recovery, no ack, connectionless

Header.sizeof() = 8 bytes

Sí tiene CHECKSUM → pero la modifica NAT-PAT → actualizar IP

- header
- pseudoheader
- payload

ARQ

protocol { · reliable · efficient } ⇒ [error { detection / recovery } + flow control]

Requisitos:
- Tx/Rx buffers
- acks
- RTO
- Seq. Number

DATOS

distancia: D [m]

bitrate: v_t [bps]

propagation { speed: $v_p \approx C$ [m/s] ; delay: $t_p = D/v_p$

PDU { information: I_K ; ack: A_K

PDU length (info/ack): L_I / L_A Bits

transmission time: $t_t = L_I / v_t$; $t_a = L_A / v_t$

ARQ Protocols

Stop & Wait

- $T_c = t_t + t_p + t_a + t_p$
- $E_f(\%) = 100 \cdot \frac{v_{ef}}{v_t} = \frac{t_t}{T_c}$

- Continuous Tx

$E(\%) = 100\%$ (without losses)

↳ Si hay errores { Go Back N → si pérdida, no envías acks (hasta recibir el que esperas) (mientras, descartas todo) ; Selective ReTx → como go back n, pero sin descartar.

- Flow Control

Adapt Tx to Rx rate. → Tx window

- Stop & Wait → Tx window = 1.

Optimal Window → { W_{opt} [PDU] $= \lceil \frac{RTT}{t_t} \rceil = \lceil v_{ef_{max}} \cdot RTT \rceil$; W_{opt} [byte] $\approx v_{ef_{max}}$ [B/s] $\cdot$ RTT