Redes e comunicacións

Tema 3: Capa de transporte

Oscar García Lorenzo

Escola Politécnica Superior de Enxeñería

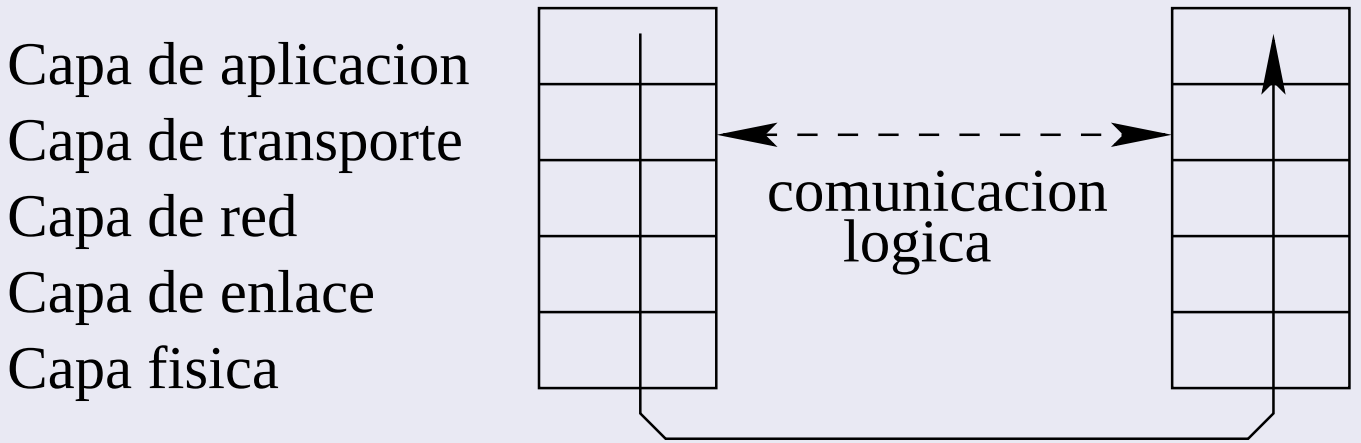
Índice

Índice

Introducción

Capa de transporte

- Prepara as mensaxes das aplicacións para ser transmitidos
- En destino, recupera as mensaxes e as entrega ás aplicacións
- So está implementada nos sistemas finais
- Proporciona unha comunicación lóxica entre procesos



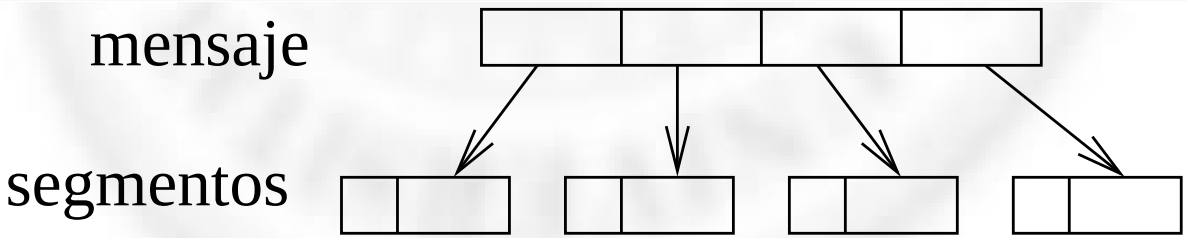
Capa de transporte

En TCP/IP

- Prepara as mensaxes para transmitirlas por unha canle non fiabel (rede de datagramas, IP, servizo de mellor esforzo)
- Protocolos de transporte TCP e UDP

TCP en orixe

- Fragmentar as mensaxes en segmentos
- Engadirlles a cabeceira da capa de transporte



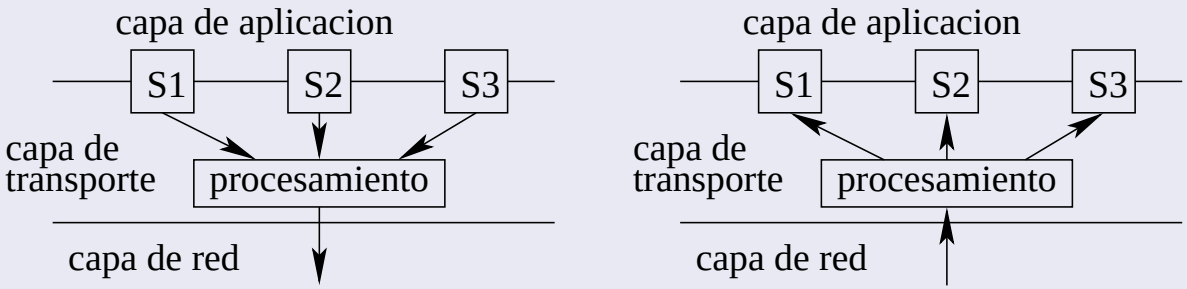
UDP en orixe

- Engadirlles a cabeceira da capa de transporte

Multiplexación e demultiplexación

Interacción entre as aplicacións e a capa de transporte

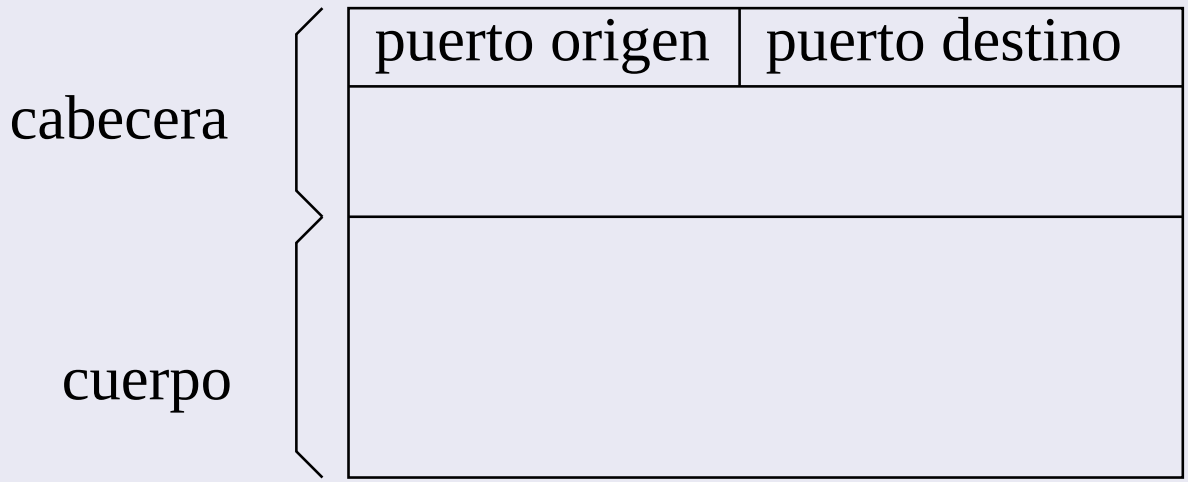
- As mensaxes pasan da capa de aplicación á capa de transporte a través dun socket
  - Os procesos escriben e len do socket
  - A capa de transporte recolle as mensaxes do socket e as traslada ao socket destino
- Multiplexación: percorrer todos os sockets abertos, procesar os mensaxes e envialos á capa de rede
- Demultiplexación: recoller os segmentos que chegan da capa de rede, reconstruir as mensaxes e colocarlos nos sockets destino



Multiplexación e demultiplexación

Interacción entre as aplicaciones e a capa de transporte

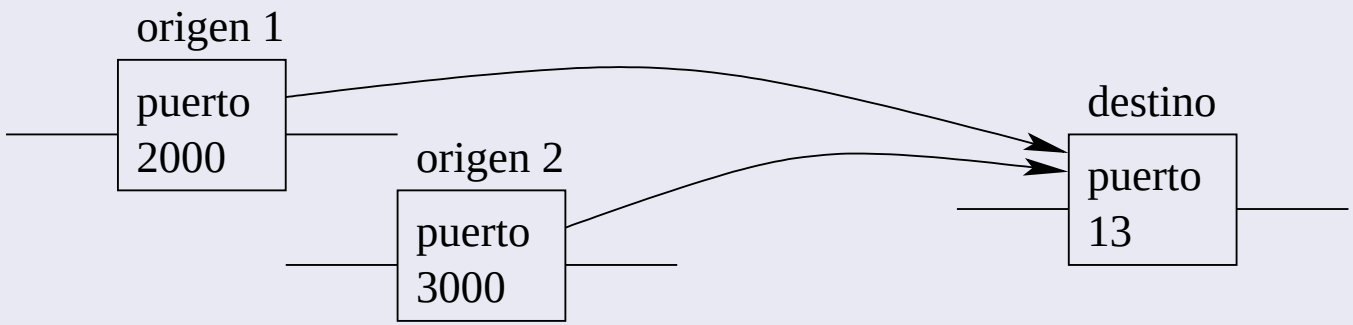
- Identificación do socket destino: a través dos números de porto da cabeceira do segmento
  - Enteiros de 16 bits
  - De 0–1023 usados polo administrador para os servizos ben coñecidos



Multiplexación e demultiplexación

Multiplexación con sockets sen conexión (en UDP)

- Identificación do socket: a parella dirección IP destino e porto destino
- Segmentos de distintos hosts que se entreguen no mesmo porto son recollidos polo mesmo proceso
- O porto orixe usase para que o proceso sepa a quen respostar

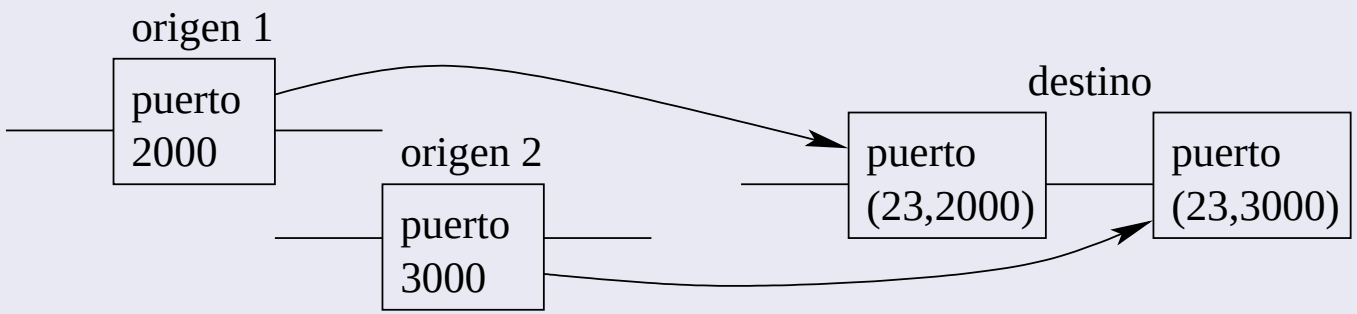


Multiplexación e demultiplexación

Multiplexación con sockets orientados a conexión (en TCP)

- Identificación do socket: a tupla de direccións IP orixe e destino e portos orixe e destino
- Segmentos de distintos hosts (ou distintos portos orixe) que se entreguen no mesmo porto van a sockets distintos

$\Longrightarrow$ varias conexións ao mesmo porto poidan ser atendidas por procesos ou fíos diferentes

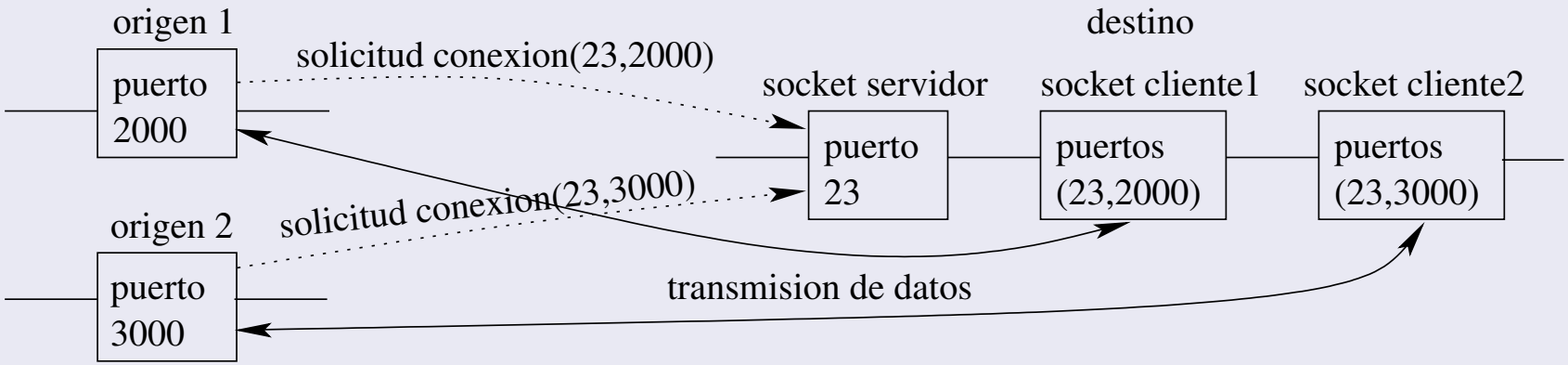


Multiplexación con sockets orientados a conexión TCP

Conexións TCP

Dous tipos de sockets:

- Un socket de servidor: espera conexións de clientes
- Varios sockets de conexión: encárganse da transmisión de datos



Índice

Transporte non orientado a conexión: UDP

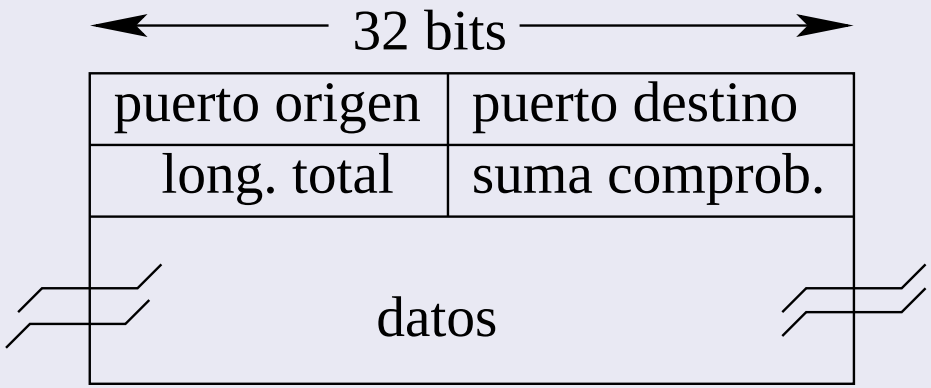
Protocolo de datagramas de usuario

- Protocolo de transporte simple e pouco sofisticado
- Fai case o mínimo que debería facer un protocolo de transporte
  - Multiplexación/demultiplexación
  - Mecanismo de comprobación de erros
- Descrito no RFC 768

Transporte non orientado a conexión: UDP

Protocolo de datagramas de usuario

- En orixe, engadelle unha cabeceira á mensaxe para formar un segmento
- En destino, comproba se o paquete chegou sen erros
  - Sen erros: entrega o paquete ao socket
  - Con erros: en xeral descartase
- Cabeceira de 4 campos (8 bytes):
  - Portos orixe e destino
  - Lonxitude total do segmento (cabeceira + datos) en bytes
  - Suma de comprobación incluindo a pseudo-cabeceira



Transporte non orientado a conexión: UDP

Características

- Sen conexión
  - Non hai acordo previo entre emisor e receptor $\Longrightarrow$ non hai retardo no establecemento da conexión
  - Non hai retransmisións en caso de erros
- Sen estado: cada segmento envíase con independencia dos demáis
  - Non hai segmentación (non hai información para reconstruir as mensaxes)
  - Procesamiento máis rápido e con menos recursos (máis clientes)
- Cabeceiras máis pequenas
- Máis control da aplicación
- Para aplicacións que prefieren velocidade frente a fiabilidade: transmisión de audio, vídeo, DNS...

Transporte non orientado a conexión: UDP

Aplicacións que usan TCP

- Correo electrónico (SMTP)
- Web (HTTP) (Cambiando, HTTP3 QUIC)
- Acceso a terminais remotos (telnet, SSH)
- Transferenza de arquivos (FTP, SFTP)

Aplicacións que normalmente usan UDP

- Traducción de nomes (DNS)
- Protocolos de encamiñamento (RIP)
- Administración de rede (SNMP)
- Servidor de arquivos remoto (NFS)

Aplicaciones que usan TCP o UDP

- Fluxos multimedia
- Telefonía por Internet

Cada vez se usa máis TCP en aplicaciones multimedia, ¿?

- Aplicaciones multimedia: toleran pérdidas e responden mal a mecanismos de control de conxestión
- Hai organizacións que bloquean o tráfico UDP, porque non ten mecanismos de control de conxestión
  - Desbordamento de paquetes nos routers
  - Estrangulamento do tráfico TCP, que si ten mecanismos de control de conxestión

Índice

Transmisión fiábel

Fundamentos

- Basanse nos protocolos de retransmisión: que retransmiten os paquetes con erros
- Protocolos ARQ *Automatic Repeat reQuest* (solicitude automática de repetición)
  - Parar e esperar: o emisor envía un paquete e espera a confirmación do receptor
  - Xanela deslizante: podense enviar varios paquetes antes de recibir as confirmacións
    - Retroceder *N*
    - Repetición selectiva
- Consideranse enlaces bidireccionais (*full duplex*)
- Numeración dos paquetes 0, 1, ..., $2^n - 1$, n número de bits para numerarlos

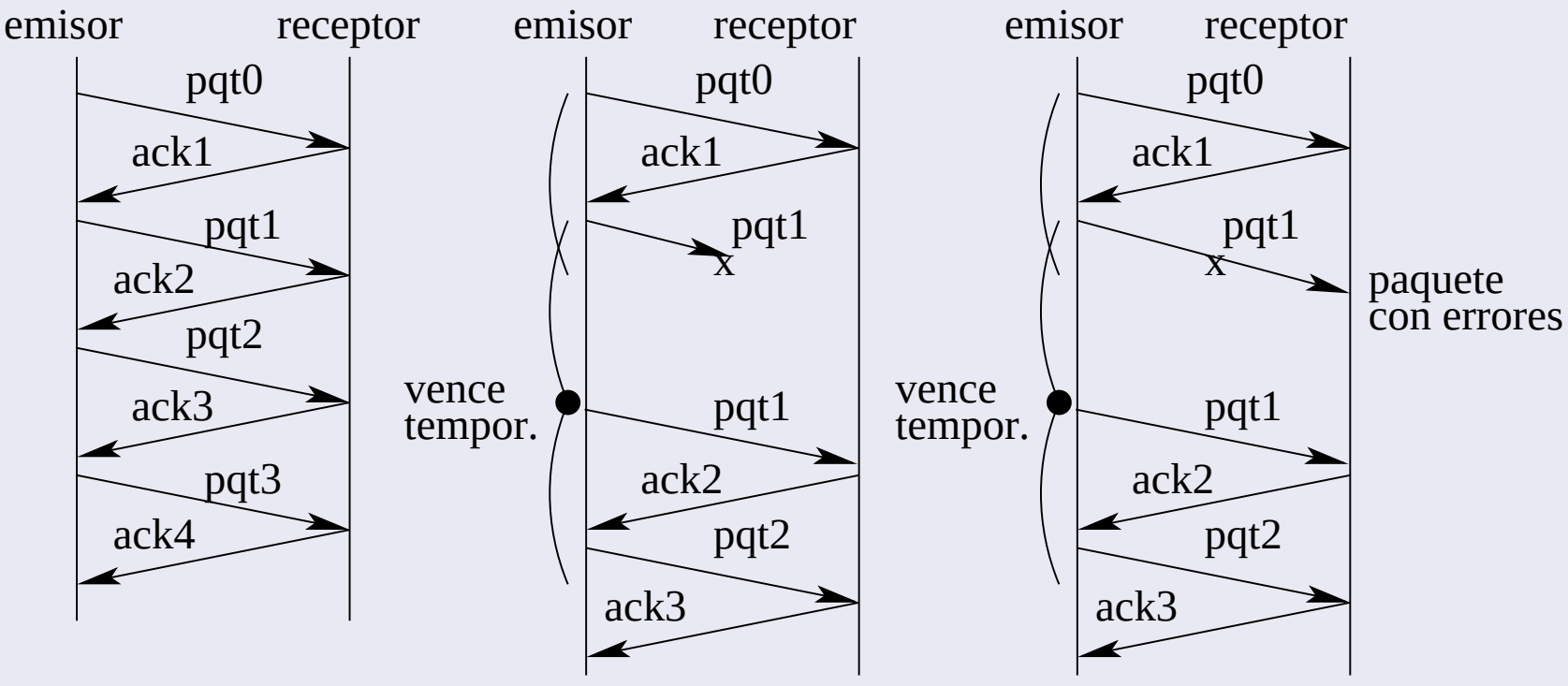
Protocolo ARQ parar e esperar

Casos

- Sen erros: o receptor devolve un ACK co número do seguinte paquete
- Pérdida de paquetes: o receptor non devolve o ACK (timeout no emisor) e o emisor retransmite o paquete
- Paquete con erros: similar ao anterior
- Pérdida dun ACK: o emisor retransmite o paquete (timeout). O receptor recibe un duplicado, o descarta e devolve o ACK
- Timeout: paquetes e ACKs chegan con retraso $\Longrightarrow$ paquetes e ACKs duplicados
  - Paquetes duplicados: coma antes
  - ACKs duplicados: ignóranse

Protocolo ARQ parar e esperar

Casos



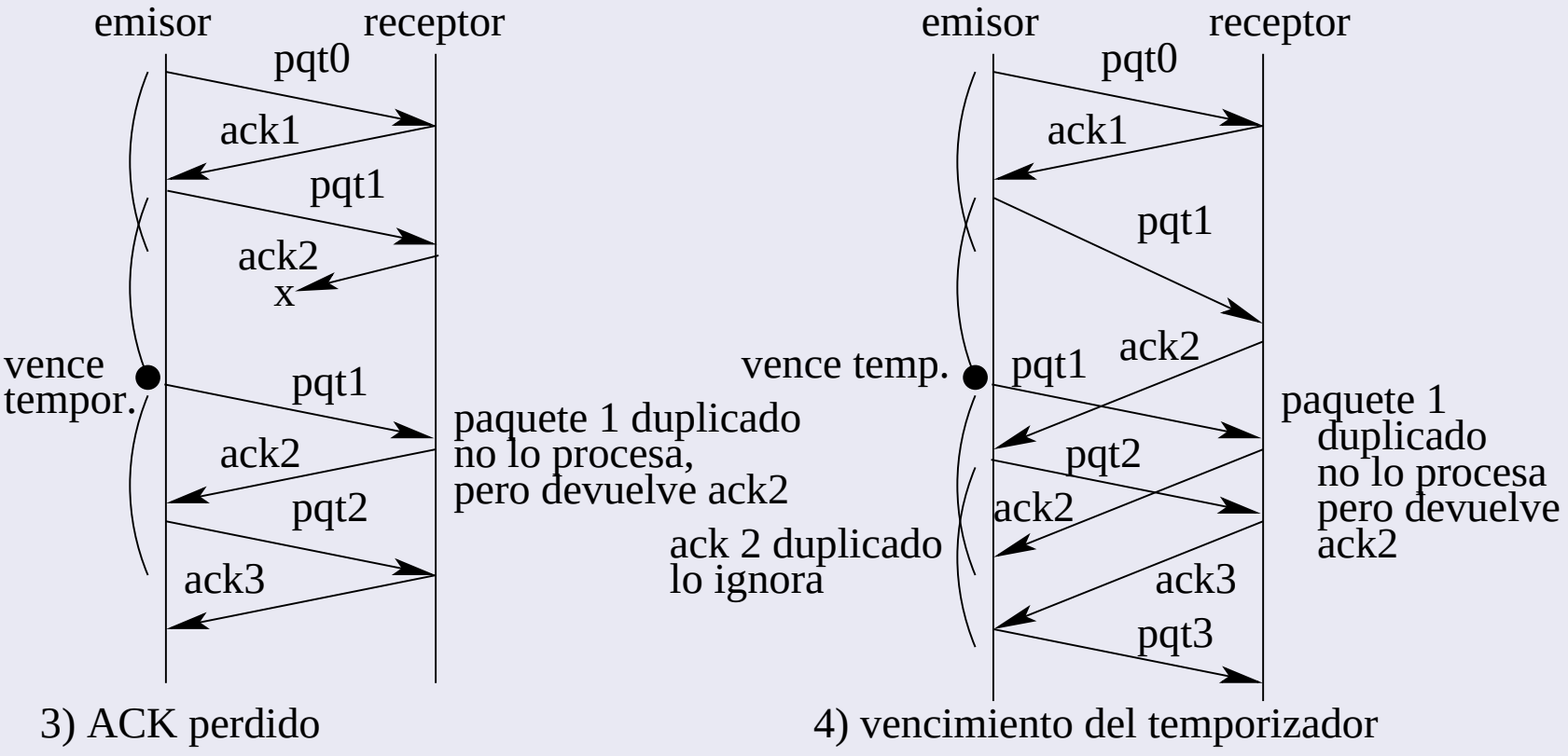
1) caso sin errores

2) caso con paquete perdido

2b) paquete con errores

Protocolo ARQ parar e esperar

Casos



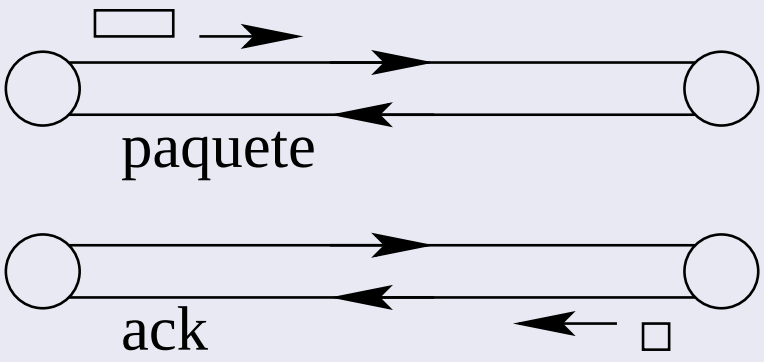
Protocolo ARQ parar e esperar

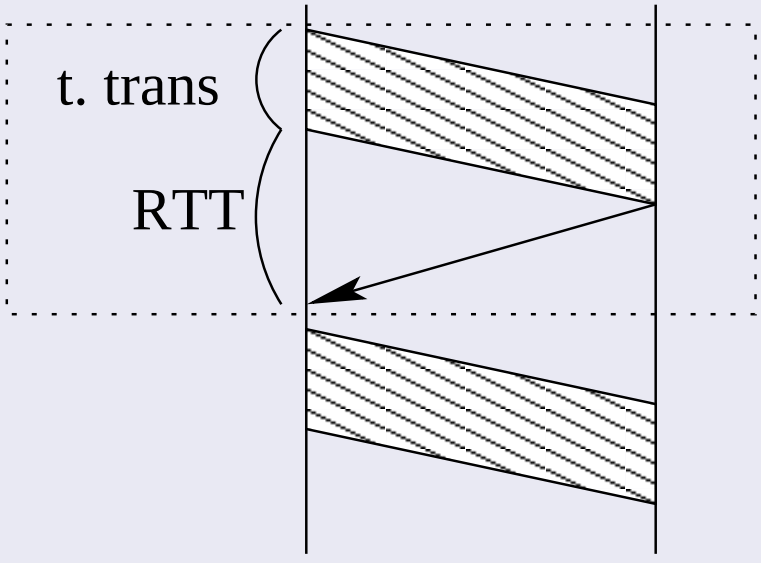
Variantes

- NAK: indica a recepción dun paquete con erros $\Longrightarrow$ non se espera ao timeout
- Dous ACKs iguais equivalen a un NAK: á recepción dun paquete con erros devolvese o ACK do último paquete correcto
  - Se vence o temporizador, envíanse continuamente duplicados de paquetes
- Tres ACKs iguais equivalen a un NAK: resolve inconvintes da anterior
  - TCP usao con algunha variación

Protocolo ARQ parar e esperar

Inconvinte principal: pouca utilización do enlace





Utilización do enlace por parte do emisor: $U = \frac{t_{trans}}{RTT + t_{trans}}$

Exemplo: paquetes de 4.000 bits, por un enlace a 1 Gbps de 1.000 km cunha velocidade de propagación de 200.000 km/s

Protocolo ARQ de xanela deslizante

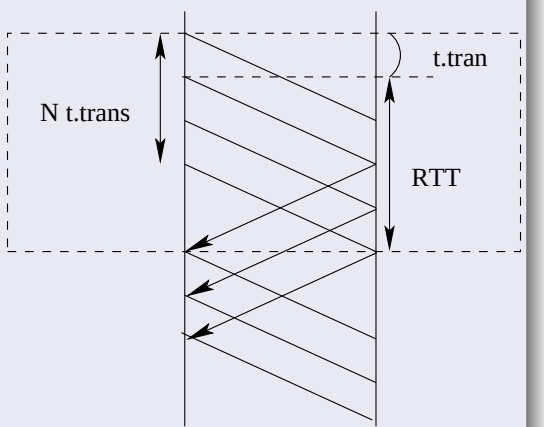
Entubamento

- O emisor envía un determinado número N de paquetes antes de recibir as confirmacións

Utilización do enlace

- Tempo útil no emisor: $N \cdot t_{trans}$
- Tempo total: $t_{trans} + RTT$
- $U = 1$ cando $N \cdot t_{trans} \geq t_{trans} + RTT$

$$\Longrightarrow N \geq 1 + \frac{RTT}{t_{trans}}$$



Protocolo ARQ de xanela deslizante

Requerimentos

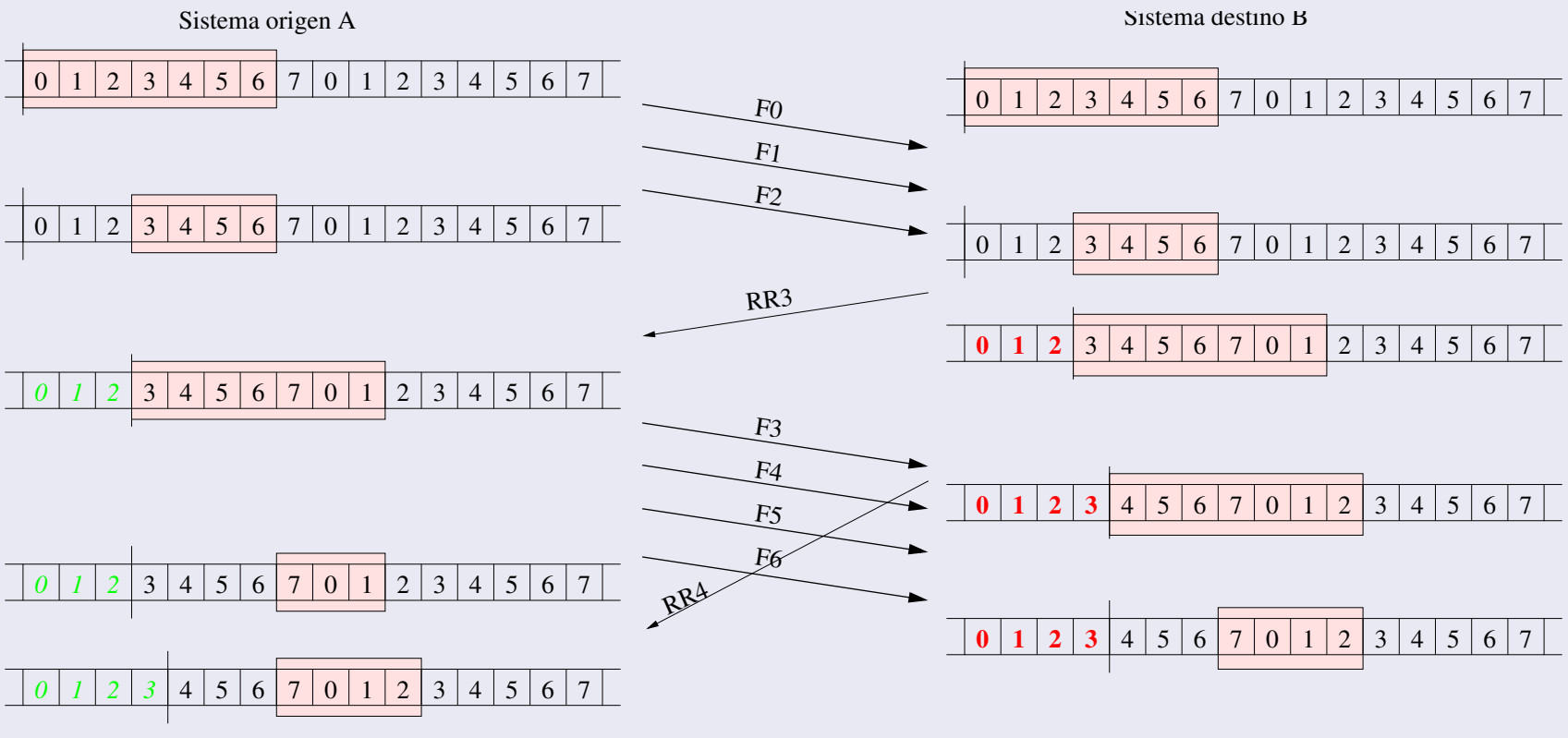
- O rango de números de secuencia debe abarcar ao menos o dobre do tamaño da xanela emisora
- Emisor e receptor deben almacenar máis dun paquete

Xanelas

- Xanela emisora: é o conxunto de N paquetes que o emisor pode enviar ou están pendentes de confirmación
- Xanela receptora: é o conxunto de N paquetes que o receptor pode aceptar ou está procesando
- Estas xanelas vanse desprazando a medida que o emisor (receptor) reciba (devolva) as confirmacións

Protocolo ARQ de xanela deslizante

Exemplo



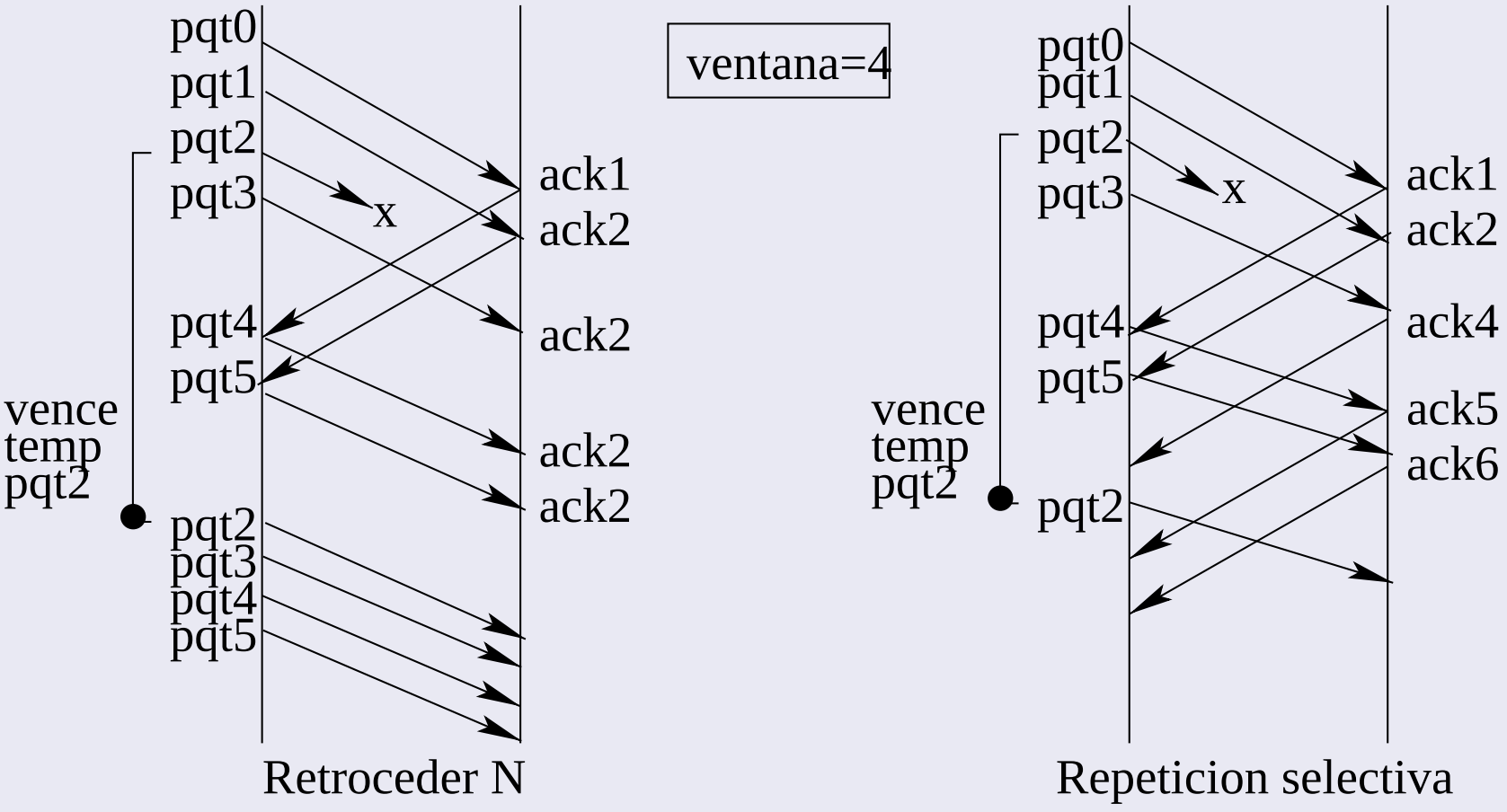
Protocolo ARQ de xanela deslizante

Tipos

- Retroceder *N* (*Go Back N*): o receptor so acepta paquetes en orden
  - Se un paquete chega con erros ou non chega, descartanse os seguintes
  - Se expira o temporizador dun paquete, retransmitense tamén os seguintes
  - ACKs acumulativos: un ACK implica un ACK a todos os paquetes previos
- Repetición selectiva: o receptor acepta paquetes fora de orden
  - So se retransmiten os paquetes erróneos ou que non chegan
  - Hai que enviar os ACKs para cada un dos paquetes recibidos

Protocolo ARQ de xanela deslizante

Retroceder *N* e repetición selectiva



Índice

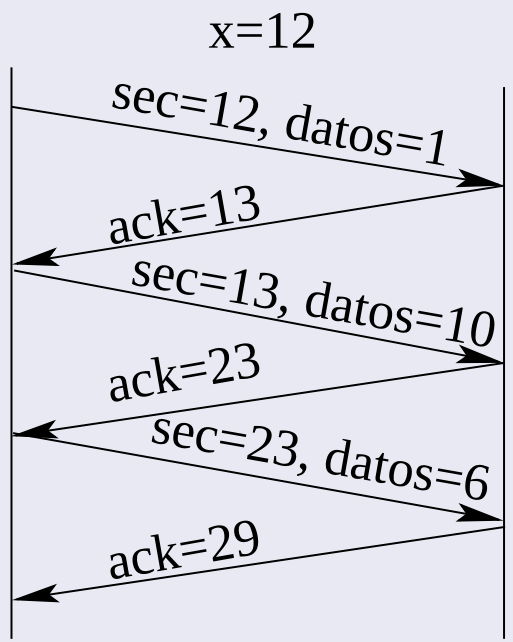
TCP: protocolo de control de transmisión

Transmisión fiábel

- Aplicación dos principios de transmisión fiábel

Números de secuencia

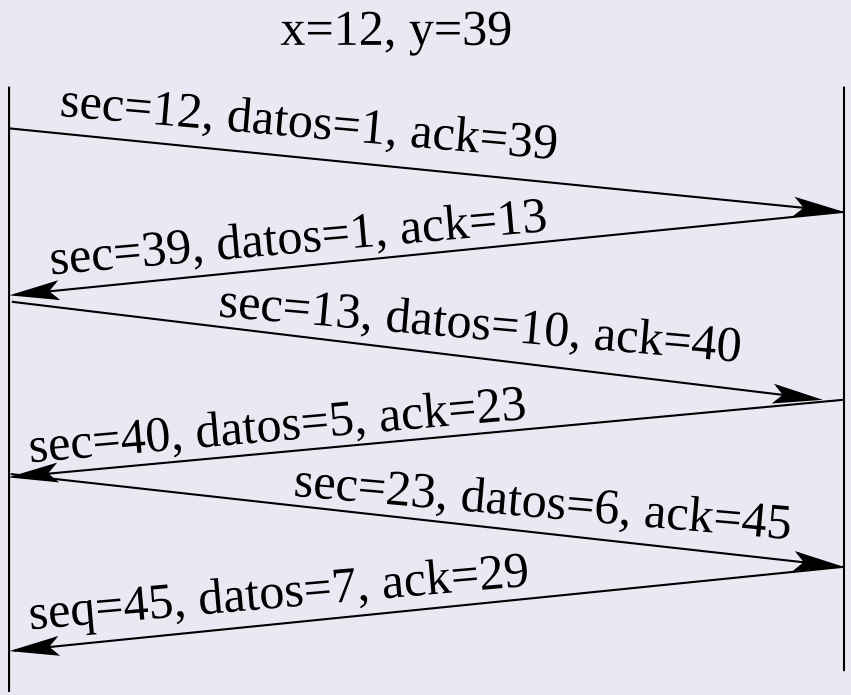
- Números de 32 bits que identifican bytes (non segmentos)
  - Empiezase por *x* (aleatorio)
  - Incrementase en $x + nbytes$
  - Os ACKs indican o siguinte byte que se desexa recibir



TCP: protocolo de control de transmisión

Superposición (*piggybacking*)

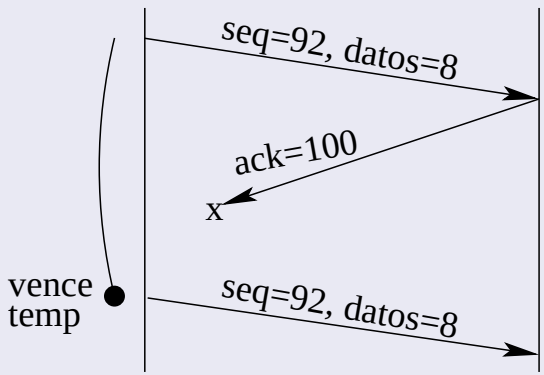
- Nun mesmo segmento ACK, transmitese tamén datos
- Tempo de espera máximo: se non ten datos, transmite so o ACK



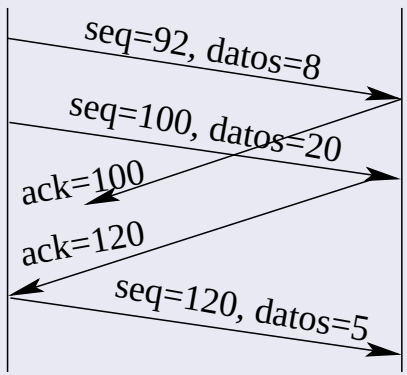
TCP: protocolo de control de transmisión

Transferencia fiábel de datos

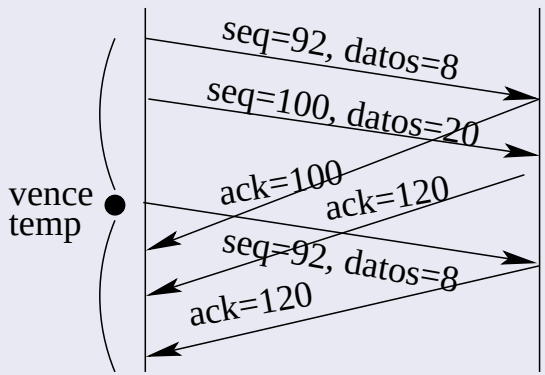
- O emisor usa temporizadores para a retransmisión
- Usase xanela deslizante e ACKs acumulativos
- Recomendase usar un temporizador único: cando chega un ACK reiniciase
- Se un ACK non chega a tempo, retransmitese so ese segmento non confirmado, reiniciase o temporizador e esperase un ACK



perdida de ACK



ACK acumulativos



solo se retransmite uno

TCP: protocolo de control de transmisión

Estimación do tempo de espera

$$\begin{aligned} \text{Temporizador} &= \text{EstimacionRTT} + 4\text{DevRTT} \\ \text{EstimacionRTT} &= (1-\alpha)\text{EstimacionRTT} + \alpha\text{MuestraRTT} \\ \text{DevRTT} &= (1-\beta)\text{DevRTT} + \beta|\text{MostraRTT} - \text{EstimacionRTT}| \end{aligned}$$

- DevRTT é unha medida da variación do RTT e, en xeral, $\alpha = 0,125$ e $\beta = 0,25$
- A estimación do RTT é en base a segmentos transmitidos e confirmados (non retransmitidos)

Duplicación do tempo de espera

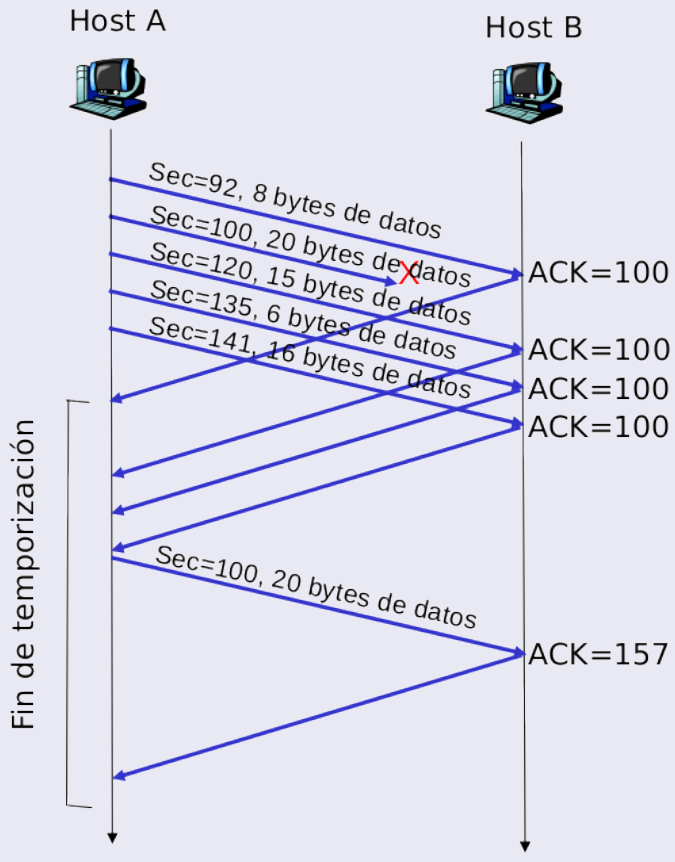
- Cando se produce a expiración do temporizador, o emisor duplica o tempo

TCP: protocolo de control de transmisión

Retransmisión rápida

Retransmisión dun paquete antes de que expire o temporizador

- 3 ACKs duplicados interpretanse como un NAK (o 4º)
- Recibese un segmento con maior número do esperado $\Longrightarrow$ o receptor envía un ACK duplicado
- Repitese a situación $\Longrightarrow$ o receptor envía outro ACK repetido $\Longrightarrow$ o emisor recibe 3 ACKs repetidos e realiza unha *retransmisión rápida*



TCP: protocolo de control de transmisión

Pérdidas en TCP

- Expiración do temporizador: problema grave, perdense os segmentos e os ACKs
- 3 ACKs duplicados: problema leve, ao menos chegan los ACKs $\Longrightarrow$ *retransmisión rápida*

ARQ intermedio entre retroceder *N* e repetición selectiva

- Retroceder *N*: ACKs acumulativos
- Repetición selectiva: aceptan segmentos fora de orden e retransmitense so os necesarios

TCP: protocolo de control de transmisión

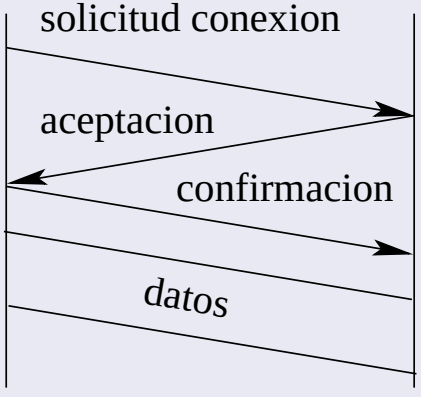
Control de fluxo

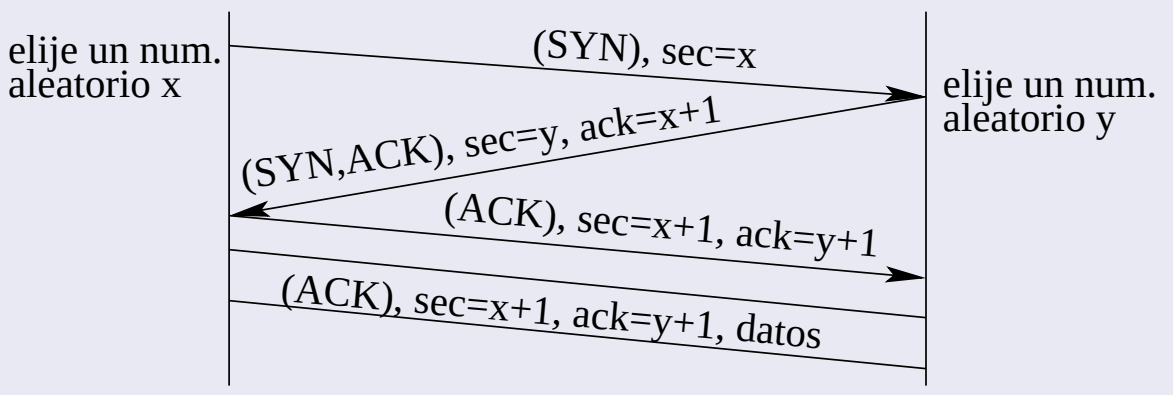
Mecanismo que permite ao receptor indicar ao emisor o ritmo ao que pode recibir datos

- No momento da conexión, o receptor indica o tamaño de su xanela de recepción
- O emisor fixa a sua xanela de envío a este valor
- Para elo existe un campo na cabeceira TCP
- O tamaño de xanela pódese modificar en cada transmisión

TCP: protocolo de control de transmisión

Conexión: acordo en tres fases

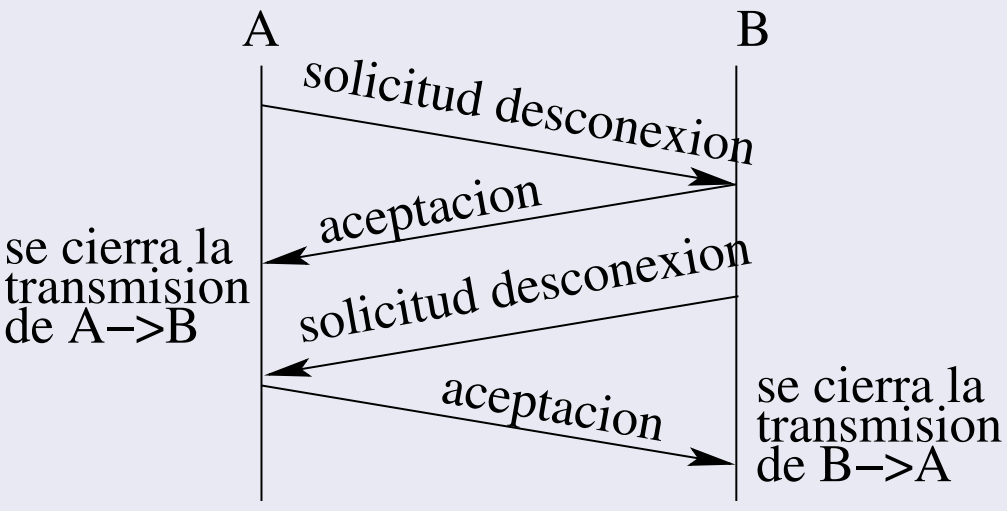


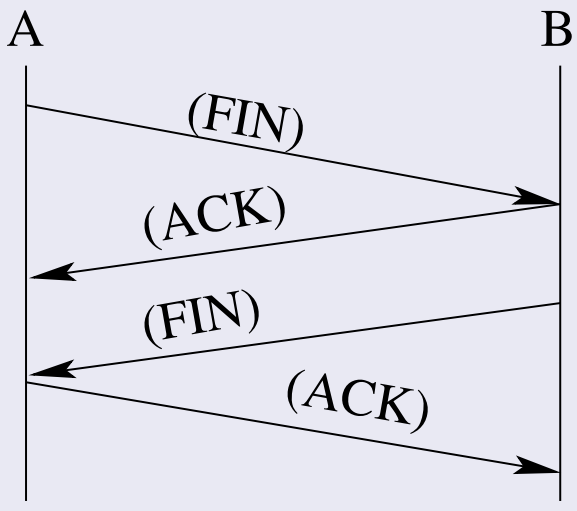


- SYN = 1, cando se envía por primeira vez *x* o *y*
- ACK = 1, cando se confirma un segmento
- Na terceira fase xa se poden enviar datos

TCP: protocolo de control de transmisión

Desconexión

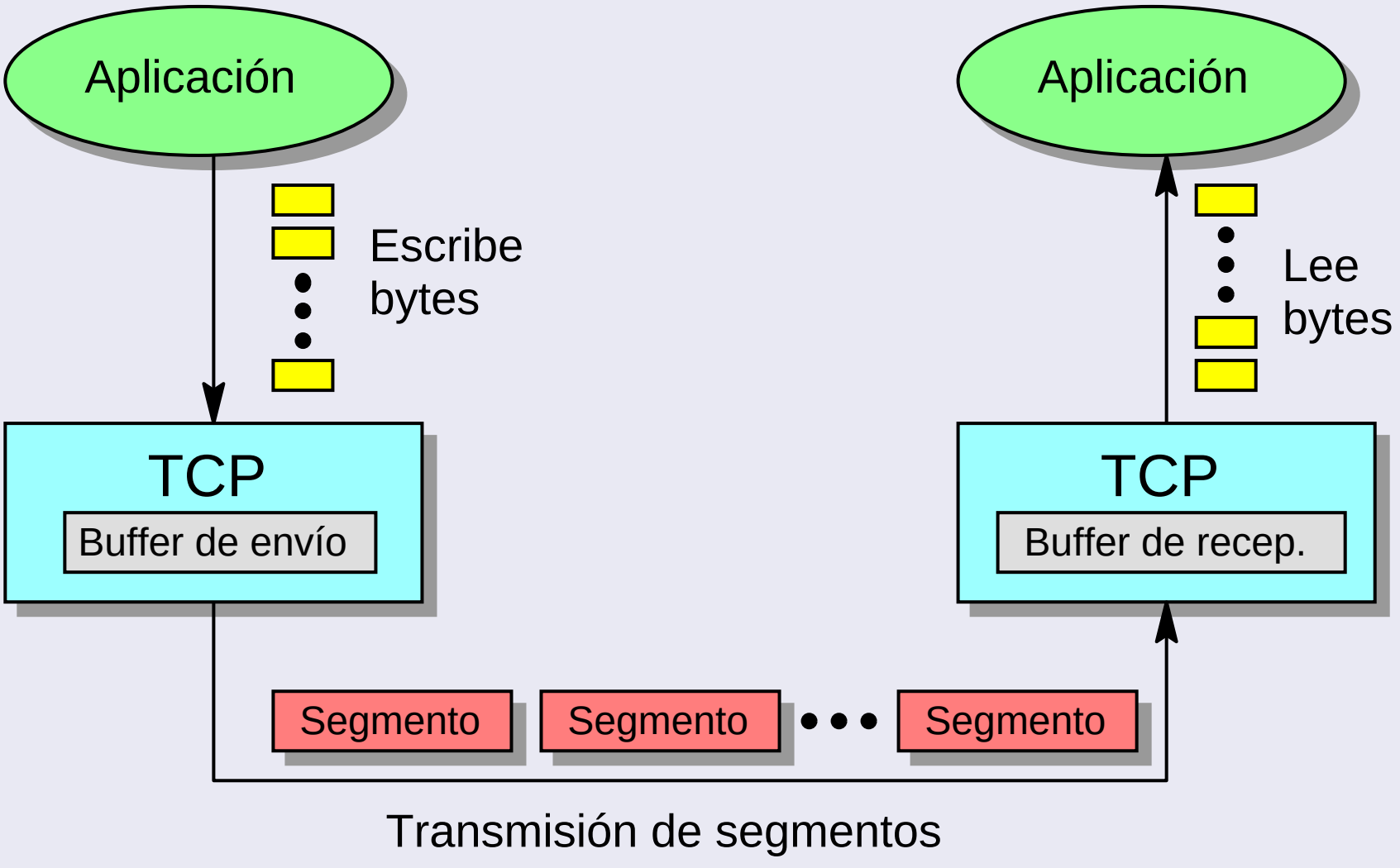




- En duas fases: cada unha para desconectar a transmisión nun sentido
- Podríase desconectar nun sentido e seguir transmitindo no otro
- FIN = 1, solicitude de desconexión
- ACK = 1, aceptación

TCP: protocolo de control de transmisión

Transmisión de datos

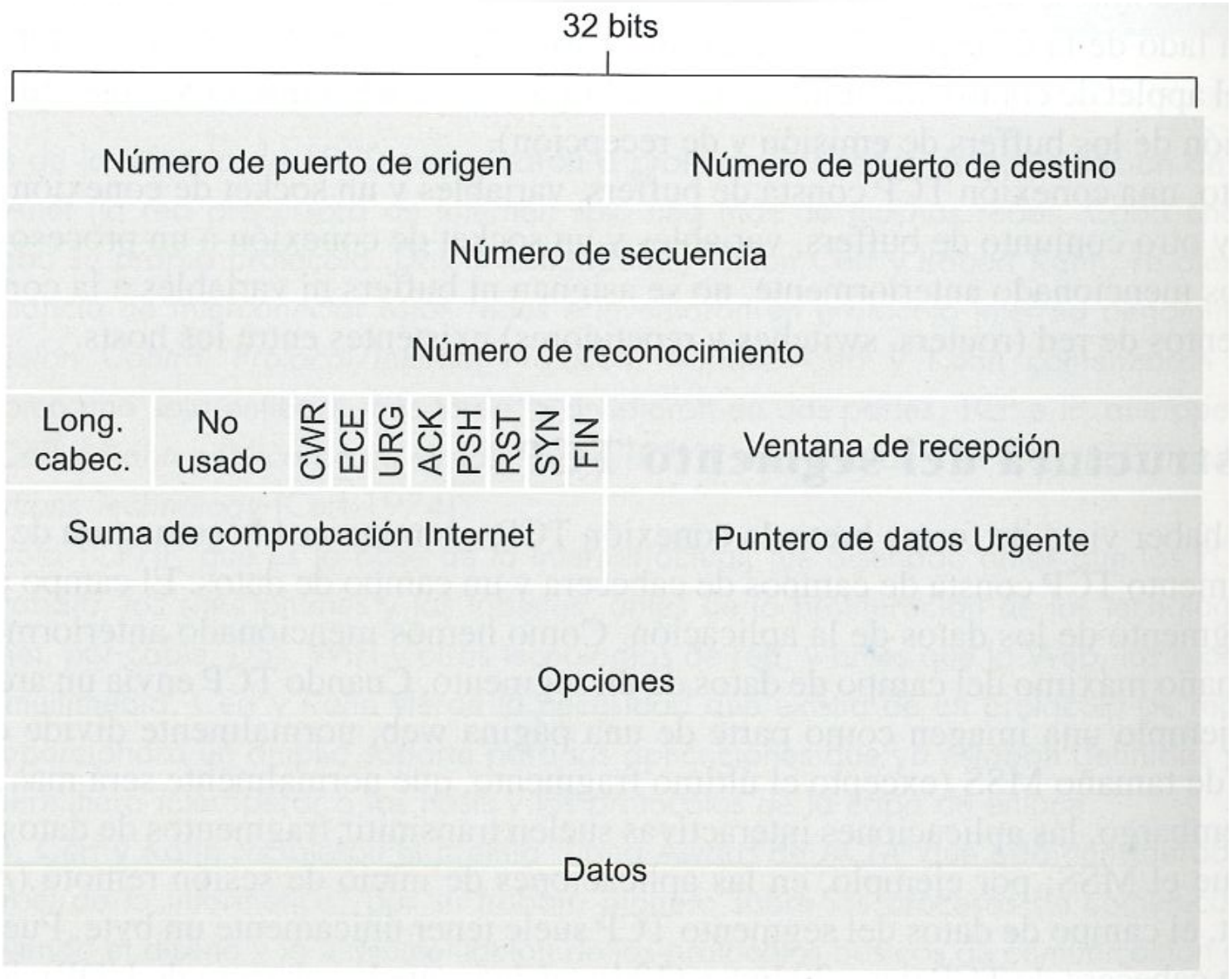


TCP: protocolo de control de transmisión

Transmisión de datos

- Unha vez establecida a conexión empeza a transmisión
- A aplicación vai pasando os datos á capa de transporte (escribindo no socket) que vainos acumulando
- Mecanismos para disparar a transmisión dun segmento:
  - Segmentación: cando o número de bytes supere ao MSS (*maximun segment size*)
  - Cando a aplicación fuerza el envío (*push*)
  - Cando un temporizador llega a 0
- TCP xerase o segmento e se llo pasa a IP

TCP: cabeceira TCP



Índice

Control de conxestión

Asignación de recursos

- Os recursos da rede débense repartir entre as diferentes peticións

Conxestión

- Demasiados paquetes na rede producen retardos nas transmisións e pérdida de moitos paquetes
- Usualmente polo desbordamento da memoria dos routers

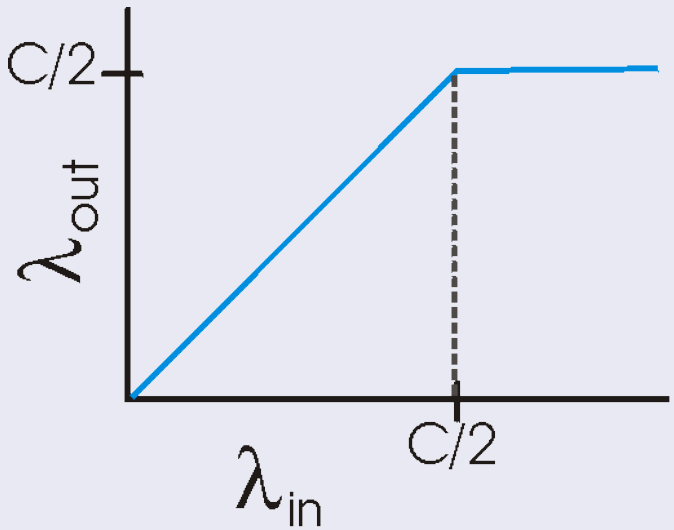
Control de conxestión

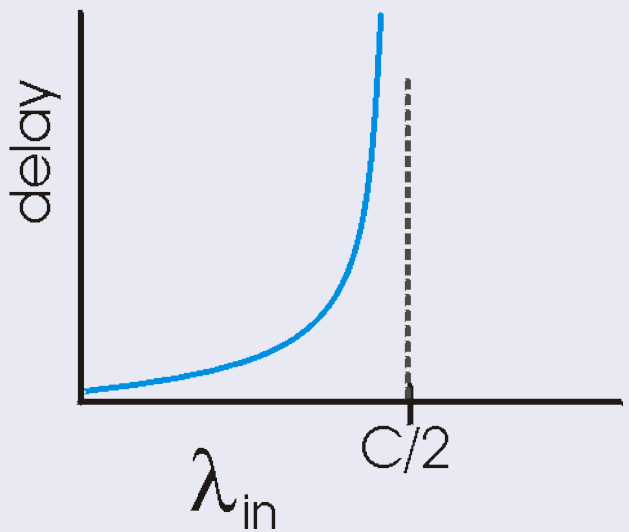
- Esforzos realizados polos elementos da rede para prevenir ou responder a situacións de conxestión
  - Pre-reservar recursos para evitar conxestión
  - Deixar que a conxestión ocorra e resolvela entón

Control de conxestión

Orige da conxestión

- Escenario 1: dous emisores, un router con capacidade infinita e enlace compartido de velocidade C
  - Tasa de transmisión entre 0 y C/2: todo se recibe OK e con retardo finito
  - Tasa de transmisión maior que C/2: o enlace non puede proporcionar paquetes a esa velocidade $\Rightarrow$ paquetes na cola do router e aumenta o retardo

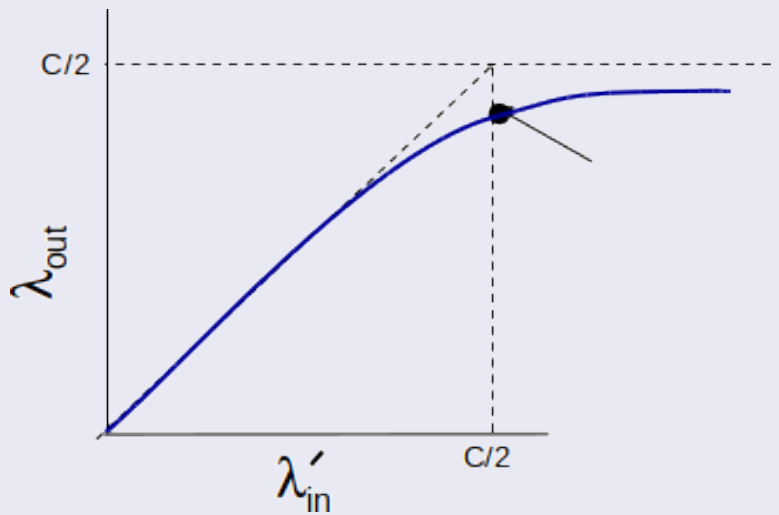




Control de conxestión

Orige da conxestión

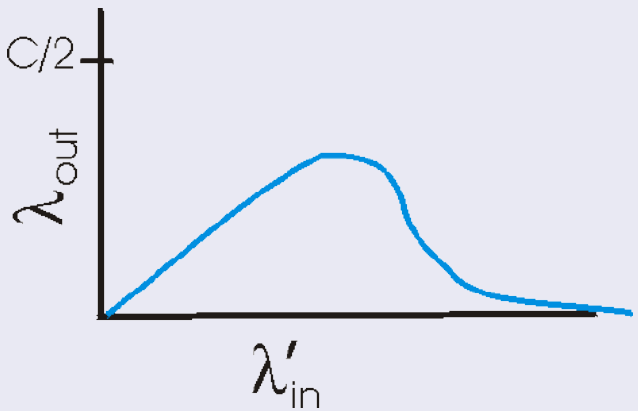
- Escenario 2: dous emisores, un router con memoria finita e enlace compartido de velocidade C
  - λ'_{in}: carga ofrecida, conten datos orixinais e retransmitidos
  - Tasa de transmisión entre 0 y C/2: todo se recibe OK e con retardo finito
  - Tasa de transmisión maior que C/2: a tasa entregada disminúe porque algúns son duplicados



Control de conxestión

Orixe da congestión

- Escenario 3: varios emisores, routers con memoria finita e varios enlaces
  - Tasa de transmisión pequena: todo se recibe OK e con retardo finito



- Tasa de transmisión elevada: os buffers dos routers enchense e a tasa entregada disminúe
  - No límite tende a cero

Necesidade de mecanismos de control de conxestión

En TCP/IP o control de conxestión recae principalmente en TCP

Control de conxestión en TCP

Mecanismo en TCP

- O emisor adecúa a sua tasa de envío en función da conxestión que percibe
- Considera que hai conxestión se:
  - Expira un temporizador
  - Recibense 3 ACKs duplicados

Procedimento

- No emisor definense: a *xanela de conxestión* e ol RTT
- O RTT estimase periódicamente (tempo desde o envío dun segmento ata que chega o seu ACK)
- Actualizase a xanela de conxestión según o caso

$$\text{tasa de envío} = \frac{\text{xanela de conxestión}}{RTT} \text{ bytes/segundo}$$

Control de conxestión en TCP

Mecanismos para actualizar a xanela de conxestión

- Inicio lento: determina a capacidade inicial da rede
- Incremento aditivo/decremento multiplicativo (AIMD): situación normal
- Recuperación rápida: evita a fase de inicio lento cando hai conxestión

Notificación explícita de conxestión (ECN)

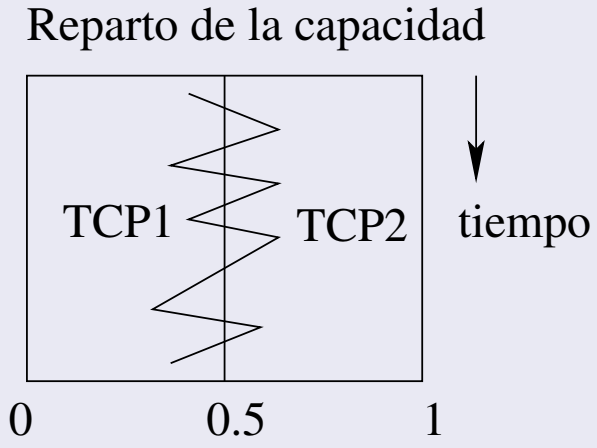
- Un router conxestionado activa uns bits da cabeceira IP
- O receptor TCP activa o bit ECE (Eco de ECN)
- O emisor TCP reduce a xanela de conxestión y activa o bit CWR

Control de conxestión en TCP

Imparcialidade

Dúas aplicacións compartindo un enlace de capacidade limitada

- Dúas conexións TCP repartense a capacidade do enlace



- Unha conexión TCP e unha UDP, a UDP acaparará a maior parte da capacidade
- Se unha aplicación usa unha conexión TCP e a outra 9 conexións TCP paralelas, a capacidade será 1/10, 9/10