

Redes e comunicacións

Tema 3: Capa de transporte

Oscar García Lorenzo

Escola Politécnica Superior de Enxeñería

Índice

- 1 Introducción
- 2 UDP: protocolo de datagramas de usuario
- 3 Fundamentos da transmisión fiábel
- 4 TCP: protocolo de control de transmisión
- 5 Control de conxestión

Índice

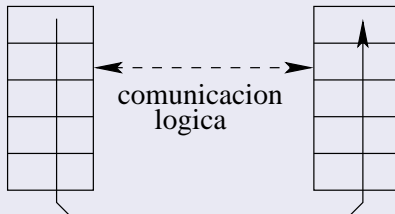
- 1 **Introducción**
- 2 UDP: protocolo de datagramas de usuario
- 3 Fundamentos de la transmisión fiable
- 4 TCP: protocolo de control de transmisión
- 5 Control de congestión

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 aplicación
Capa de transporte
Capa de red
Capa de enlace
Capa física



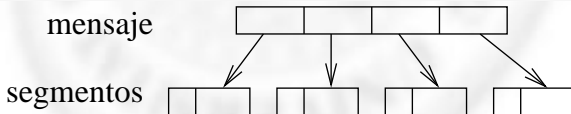
Capa de transporte

En TCP/IP

- Prepara as mensaxes para transmitirlas por un canal no fiable (rede de datagramas, IP, servizo de mellor esforzo)
- Protocolos de transporte TCP e UDP

TCP en orixe

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



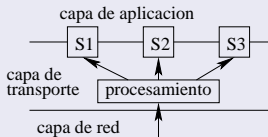
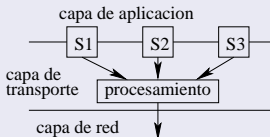
UDP en orixe

- Engadir-lles a cabeceira da capa de transporte

Multiplexación e demultiplexación

Interacción entre as aplicacións e a capa de transporte

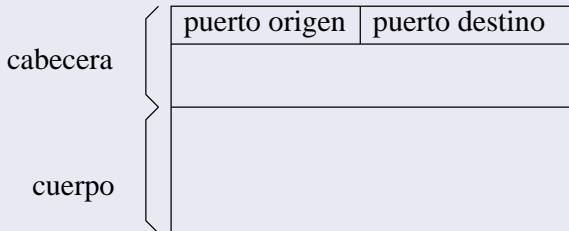
- As mensaxes pasan da capa de aplicación á capa de transporte a través dun socket
 - Os procesos escriben e len do socket
 - A capa de transporte recolle as mensaxes do socket e as traslada ao socket destino
- Multiplexación: percorrer todos os sockets abertos, procesar os mensaxes e envialos á capa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



Multiplexación e demultiplexación

Interacción entre as aplicaciónes e a capa de transporte

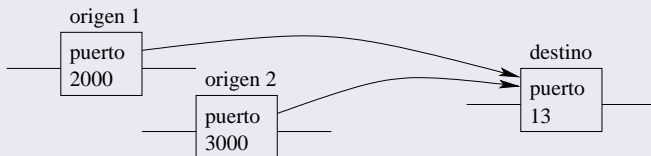
- Identificación do socket destino: a través dos números de porto da cabeceira do segmento
 - Enteiros de 16 bits
 - De 0–1023 usados polo administrador para os servizos ben coñecidos



Multiplexación e demultiplexación

Multiplexación con sockets sin 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

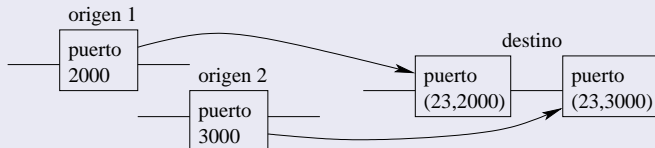


Multiplexación e demultiplexación

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

⇒ varias conexións ao mesmo porto poidan ser atendidas por procesos ou fíos diferentes

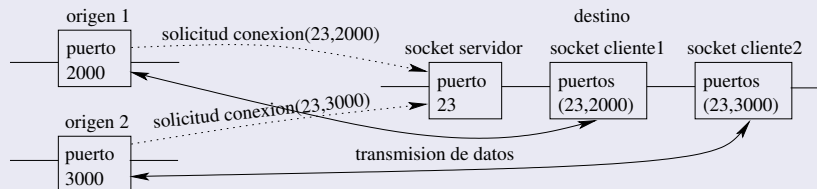


Multiplexación con sockets orientados a conexión TCP

Conexiones TCP

Dous tipos de sockets:

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



Índice

- 1 Introducción
- 2 UDP: protocolo de datagramas de usuario**
- 3 Fundamentos de la transmisión fiable
- 4 TCP: protocolo de control de transmisión
- 5 Control de congestión

Transporte no orientado a conexión: UDP

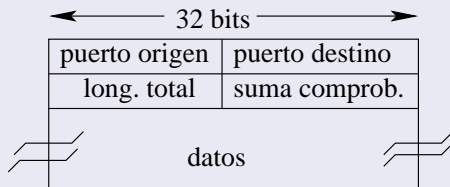
Protocolo de datagramas de usuario

- 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 incluíndo a pseudo-cabeceira



Transporte non orientado a conexión: UDP

Características

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

Transporte no orientado a conexión: UDP

Aplicaciones que usan TCP

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

Aplicaciones 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 más TCP en aplicaciones multimedia, ¿?

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

Índice

- 1 Introducción
- 2 UDP: protocolo de datagramas de usuario
- 3 Fundamentos de transmisión fiable**
- 4 TCP: protocolo de control de transmisión
- 5 Control de congestión

Transmisión fiable

Fundamentos

- Basanse nos protocolos de retransmisión: que retransmiten os paquetes con erros
- Protocolos ARQ *Automatic Repeat reQuest* (solicitud 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 $0, 1, \dots, 2^n - 1$, n número de bits para numerarlos

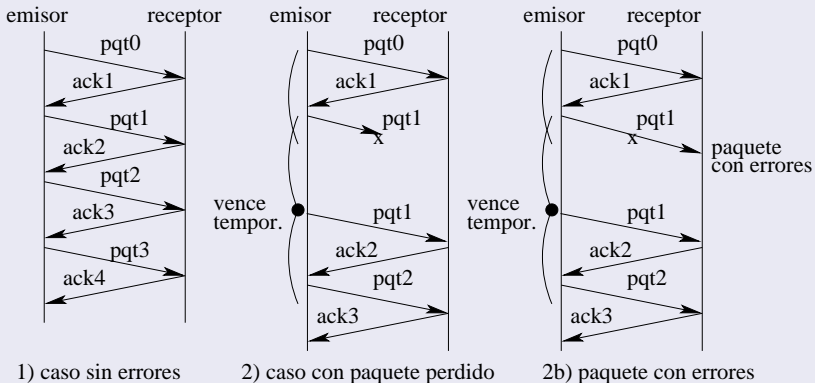
Protocolo ARQ parar e esperar

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 \Rightarrow paquetes e ACKs duplicados
 - Paquetes duplicados: coma antes
 - ACKs duplicados: ignóranse

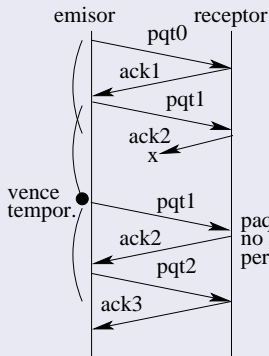
Protocolo ARQ parar e esperar

Casos

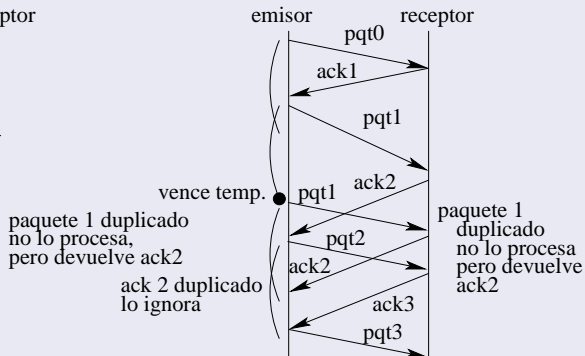


Protocolo ARQ parar e esperar

Casos



3) ACK perdido



4) vencimiento del temporizador

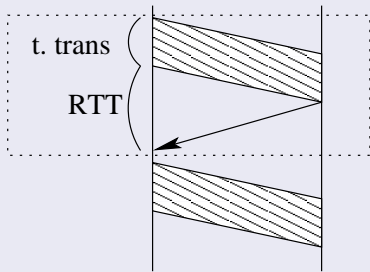
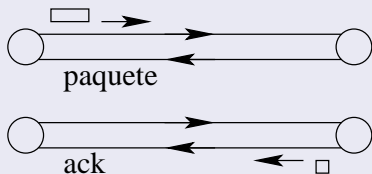
Protocolo ARQ parar e esperar

Variantes

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

Protocolo ARQ parar e esperar

Inconviente 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

Requerimentos

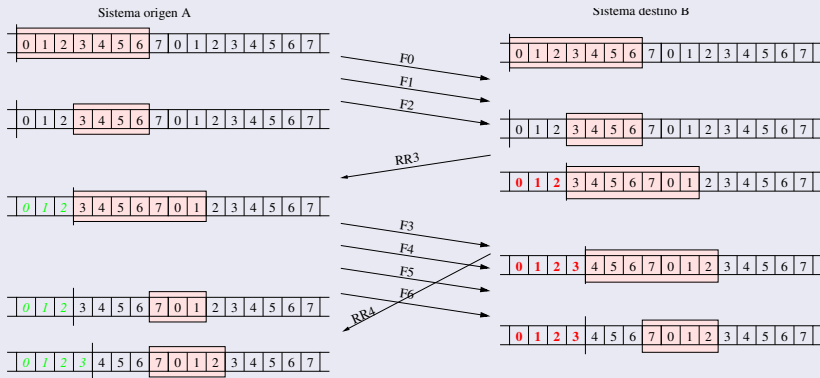
- O rango de números de secuencia debe abarcar ao menos o dobre do tamaño da xanela emisora
- Emisor e receptor deben almacenar máis dun paquete

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 só acepta paquetes en orden
 - Se un paquete chega con erros ou non chega, descártanse os seguintes
 - Se expira o temporizador dun paquete, retransmítense tamén os seguintes
 - ACKs acumulativos: un ACK implica un ACK a todos os paquetes previos
- Repetición selectiva: o receptor acepta paquetes fora de orden
 - Só se retransmíten os paquetes erróneos ou que non chegan
 - Hai que enviar os ACKs para cada un dos paquetes recibidos

Índice

- 1 Introducción
- 2 UDP: protocolo de datagramas de usuario
- 3 Fundamentos da transmisión fiábel
- 4 TCP: protocolo de control de transmisión**
- 5 Control de conxestión

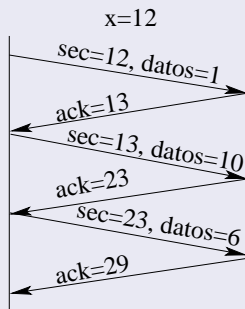
TCP: protocolo de control de transmisión

Transmisión fiable

- Aplicación dos principios de transmisión fiable

Números de secuencia

- Números de 32 bits que identifican bytes (non segmentos)
 - Empiezas por x (aleatorio)
 - Incrementase en $x + n\text{bytes}$
 - Os ACKs indican o seguinte byte que se desexa recibir

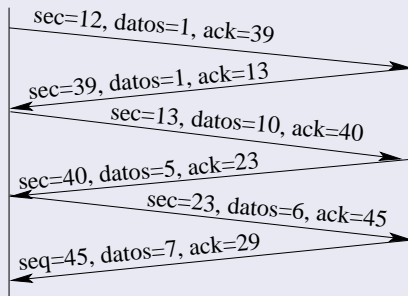


TCP: protocolo de control de transmisión

Superposición (*piggybacking*)

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

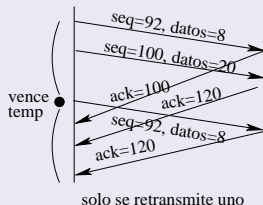
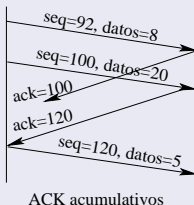
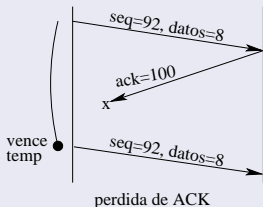
$x=12, y=39$



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, retransmítese so ese segmento non confirmado, reiniciase o temporizador e esperase un ACK



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

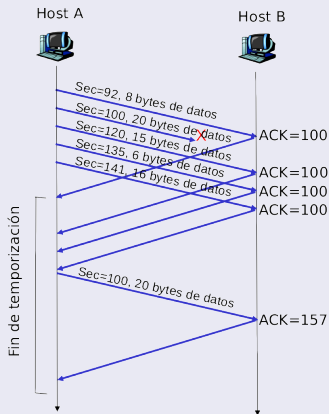
- Cando se produce a expiración do temporizador, o emisor duplica o tempo

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)
- Recíbese un segmento con maior número do esperado \Rightarrow o receptor envía un ACK duplicado
- Repítese a situación \Rightarrow o receptor envía outro ACK repetido \Rightarrow 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 \implies *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

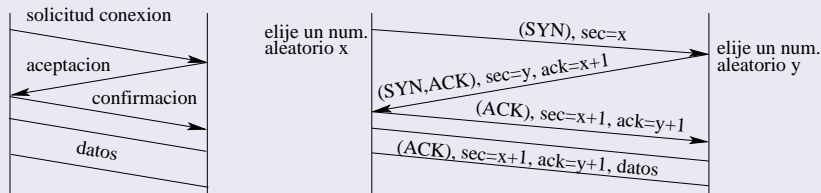
Control de fluxo

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

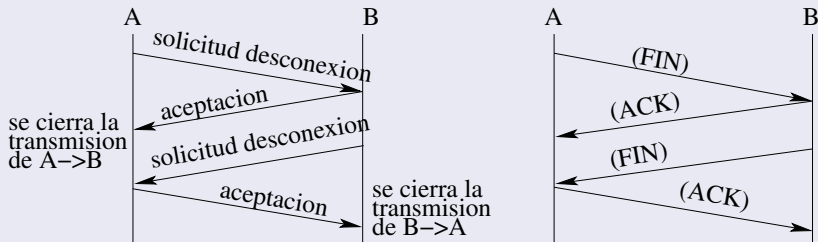
Conexión: acuerdo en tres fases



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

TCP: protocolo de control de transmisión

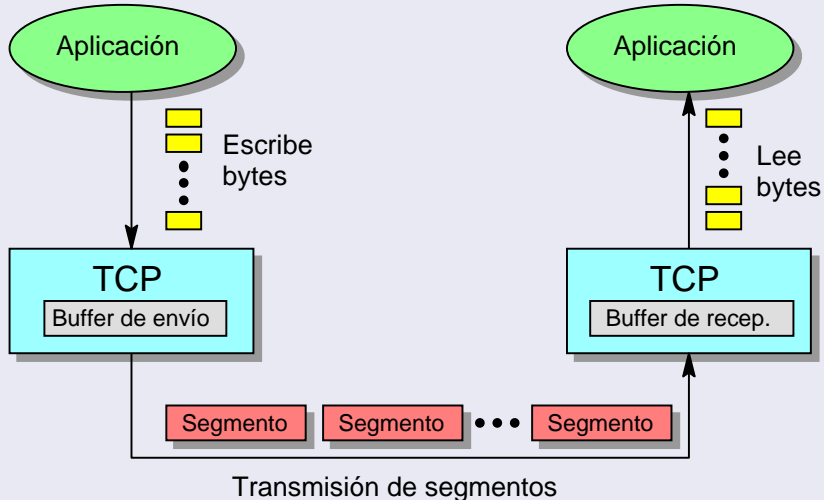
Desconexión



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

TCP: protocolo de control de transmisión

Transmisión de datos

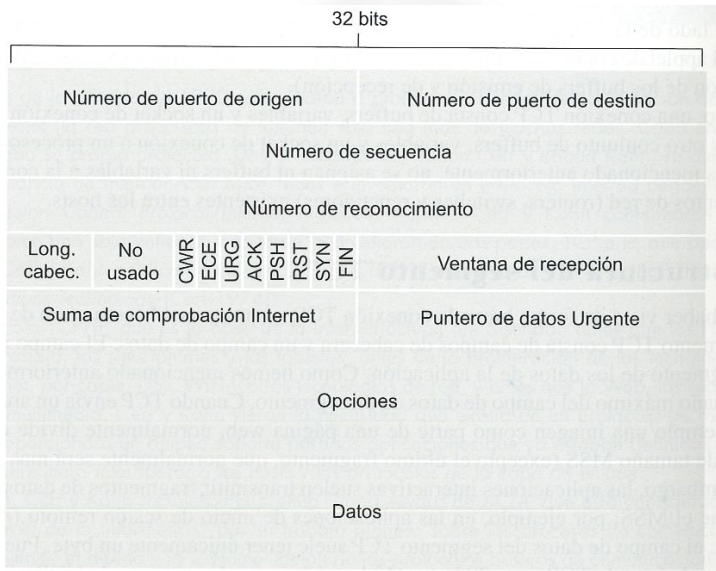


TCP: protocolo de control de transmisión

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 forza el envío (*push*)
 - Cando un temporizador llega a 0
- TCP xerase o segmento e se llo pasa a IP

TCP: cabeza TCP



Índice

- 1 Introducción
- 2 UDP: protocolo de datagramas de usuario
- 3 Fundamentos de la transmisión fiable
- 4 TCP: protocolo de control de transmisión
- 5 Control de congestión**

Control de congestión

Asignación de recursos

- Os recursos da rede débense repartir entre as diferentes peticións

Conxestión

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

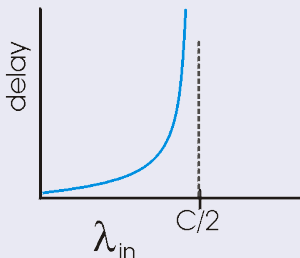
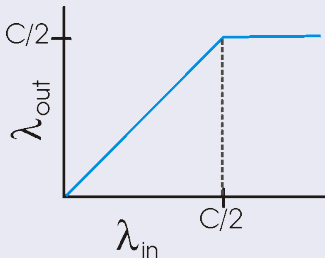
Control de congestión

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

Control de congestión

Orígenes de la congestión

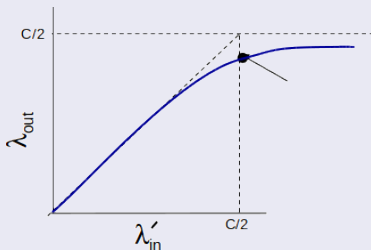
- Escenario 1: dos emisores, un router con capacidad infinita e enlace compartido de velocidad C
 - Tasa de transmisión entre 0 y $C/2$: todo se recibe OK e con retardo finito
 - Tasa de transmisión mayor que $C/2$: el enlace no puede proporcionar paquetes a esa velocidad \Rightarrow paquetes en la cola del router e aumenta el retardo



Control de congestión

Orígenes da congestión

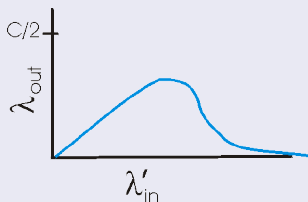
- Escenario 2: dous emisores, un router con memoria finita e enlace compartido de velocidade C
 - λ'_{in} : carga ofrecida, contén datos orixinais e retransmitidos
 - Tasa de transmisión entre 0 y $C/2$: todo se recibe OK e con retardo finito
 - Tasa de transmisión maior que $C/2$: a taxa entregada diminúe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



Control de congestión

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 taxa entregada diminúe
 - No límite tende a cero

Necesidade de mecanismos de control de congestión

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

Control de congestión en TCP

Mecanismo en TCP

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

Procedimento

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

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

Control de congestión en TCP

Mecanismos para actualizar a xanela de congestión

- 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 congestión

Notificación explícita de congestión (ECN)

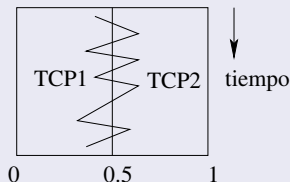
- Un router congestionado activa uns bits da cabeceira IP
- O receptor TCP activa o bit ECE (Eco de ECN)
- O emisor TCP reduce a xanela de congestión y activa o bit CWR

Control de congestión en TCP

Imparcialidad

Dúas aplicacións compartindo un enlace de capacidade limitada

- Dúas conexións TCP repartense a capacidade do enlace
Reparto de la capacidad



- 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