Tema 4: CONTROL DE FLUJO

Ingeniería de Protocolos y Servicios

4.1. Introducción

Para qué sirve el control de flujo:

- Asegurar que los datos no se transmiten más rápido de lo que se puedan procesar.
- Optimizar el uso del canal.
- Evitar la saturación de ciertos enlaces del canal.

Los puntos 2 y 3 son complementarios.

DTIC
Área Ingeniería Telemática
francesc.burrull@upct.es

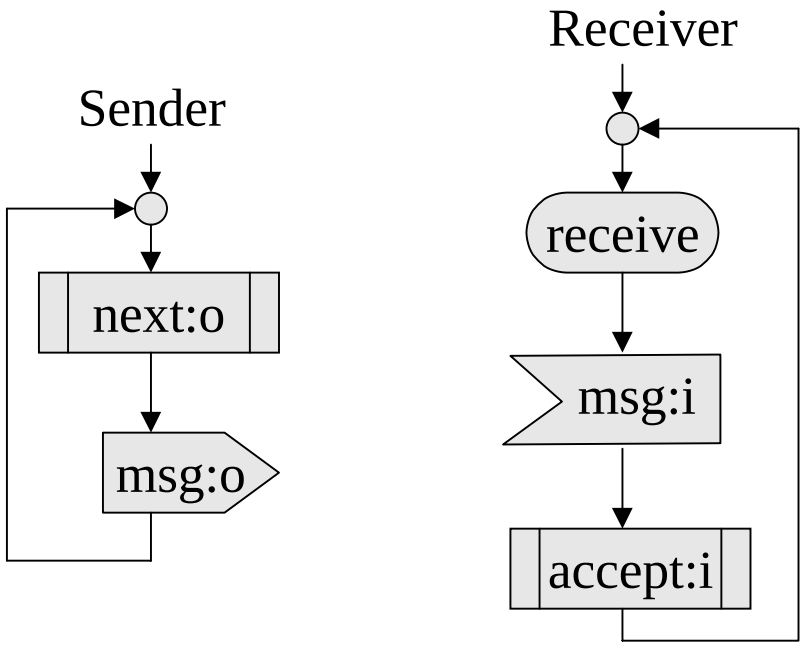
Tema 4: CONTROL DE FLUJO

Ingeniería de Protocolos y Servicios

4.1. Introducción

MODELO BÁSICO: Protocolo sin control de flujo.

-simplex-



DTIC
Área Ingeniería Telemática
francesc.burrull@upct.es

Tema 4: CONTROL DE FLUJO

Ingeniería de Protocolos y Servicios

4.1. Introducción

MODELO BÁSICO

• Funciona si el receptor es más rápido que el receptor.

• Diseño de sistemas concurrentes: No hacer suposiciones de las velocidades relativas de procesos concurrentes.

DTIC
Área Ingeniería Telemática

francesc.burrull@upct.es

4.1. Introducción

X-ON X-OFF

• Utiliza 2 mensajes de control: *resume* y *suspend*

• No requiere negociación previa

Así, suponiendo un canal sin errores y el vocabulario:

V = { *msg*, *suspend*, *resume* }

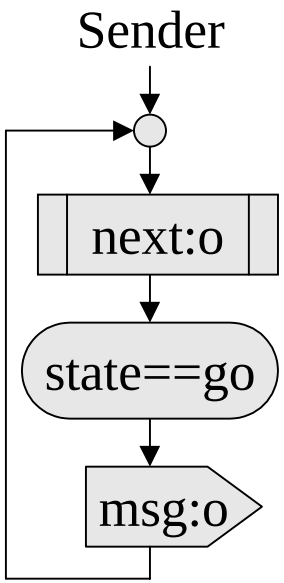
Las reglas de procedimiento para el emisor y para el receptor serán:

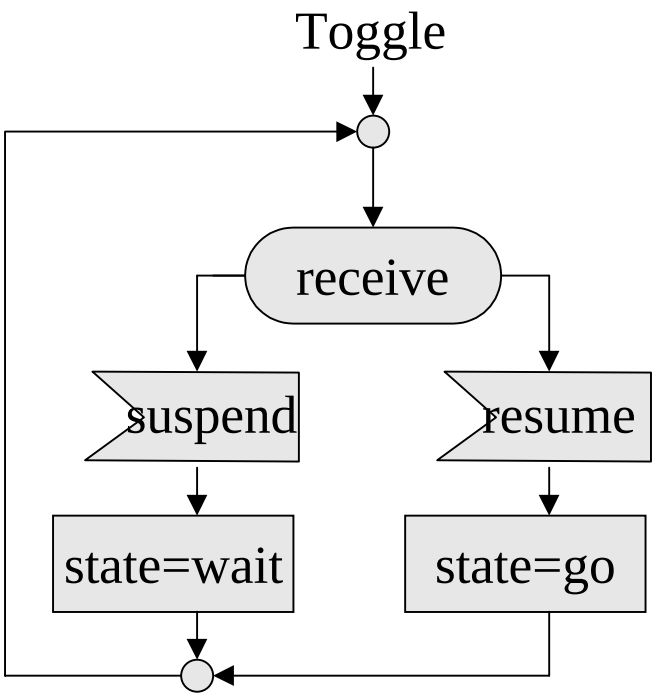
Tema 4: CONTROL DE FLUJO

Ingeniería de Protocolos y Servicios

4.1. Introducción

X-ON X-OFF: **Emisor**





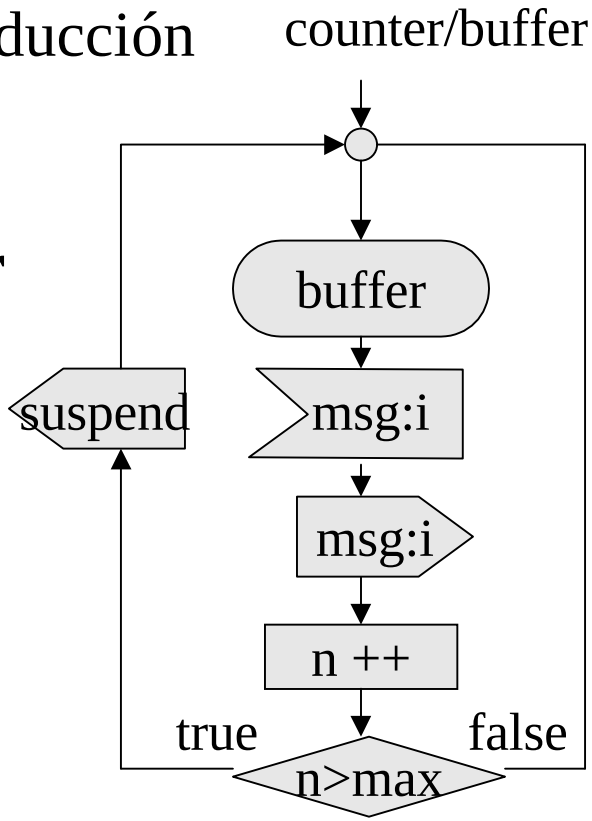
DTIC
Área Ingeniería Telemática

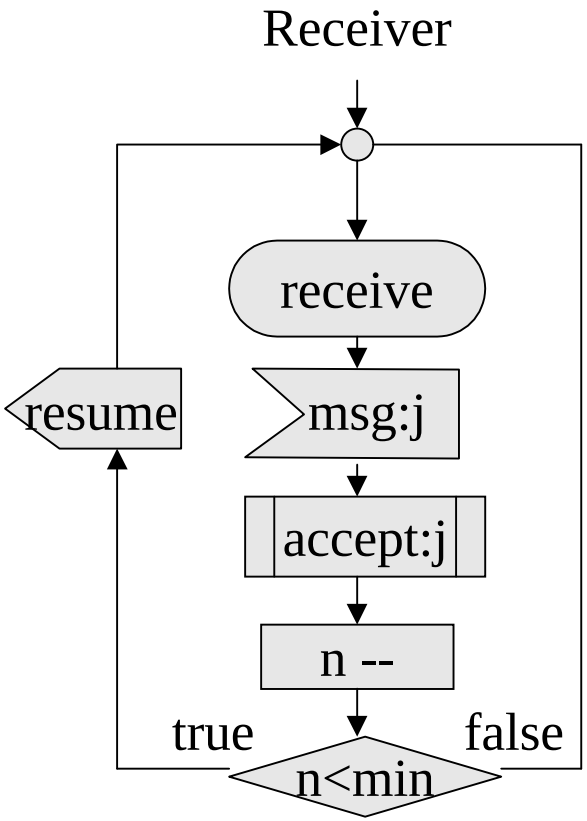
francesc.burrull@upct.es

Tema 4: CONTROL DE FLUJO

Ingeniería de Protocolos y Servicios

4.1. Introducción
X-ON
X-OFF
Receptor





DTIC
Área Ingeniería Telemática
francesc.burrull@upct.es

Tema 4: CONTROL DE FLUJO

Ingeniería de Protocolos y Servicios

4.1. Introducción

X-ON X-OFF

Los mensajes de datos pasan de *counter/buffer* a *receiver* a través de una cola interna.

Pegas: El protocolo depende de las propiedades del canal:

- Si un mensaje *suspend* se retrasa o pierde hay *overflow*.
- Si un mensaje *resume* se pierde los 4 procesos colapsan.

DTIC
Área Ingeniería Telemática
francesc.burrull@upct.es

4.1. Introducción

STOP AND WAIT

Hay que solucionar 2 problemas:

- Problemas de *overflow* del mecanismo X-ON/X-OFF
- Protección contra pérdidas de mensajes.

Propuesta: Protocolo ping-pong (*Stop and wait*)

- Emisor espera un reconocimiento del mensaje enviado.
- Ya no hay *overflow*, pero la pérdida de tan solo un mensaje -datos o control- bloquea el sistema.

Tema 4: CONTROL DE FLUJO

Ingeniería de Protocolos y Servicios

4.1. Introducción

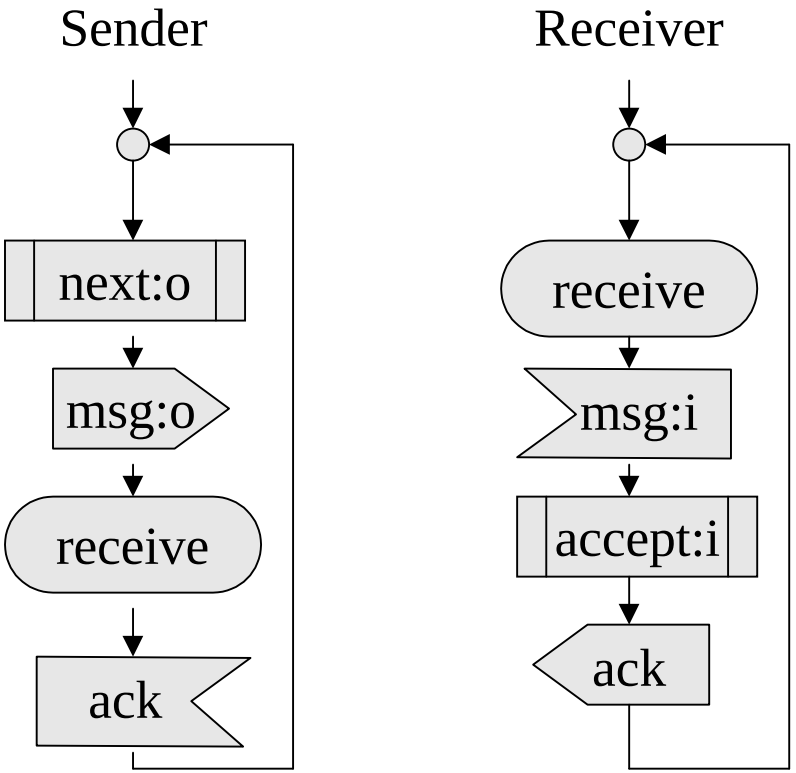
PING-PONG

2t+a+p

t: tiempo propagación

p: tiempo producir

a: tiempo aceptar



DTIC
Área Ingeniería Telemática

francesc.burrull@upct.es

4.2. Protocolos de ventana

El mensaje de control *ack* del protocolo *Stop and Wait* puede verse como un mecanismo de crédito.

Así, al iniciar la comunicación, el receptor puede indicar el buffer disponible dando cierto crédito al emisor.

Suponiendo un canal sin pérdidas, el emisor:

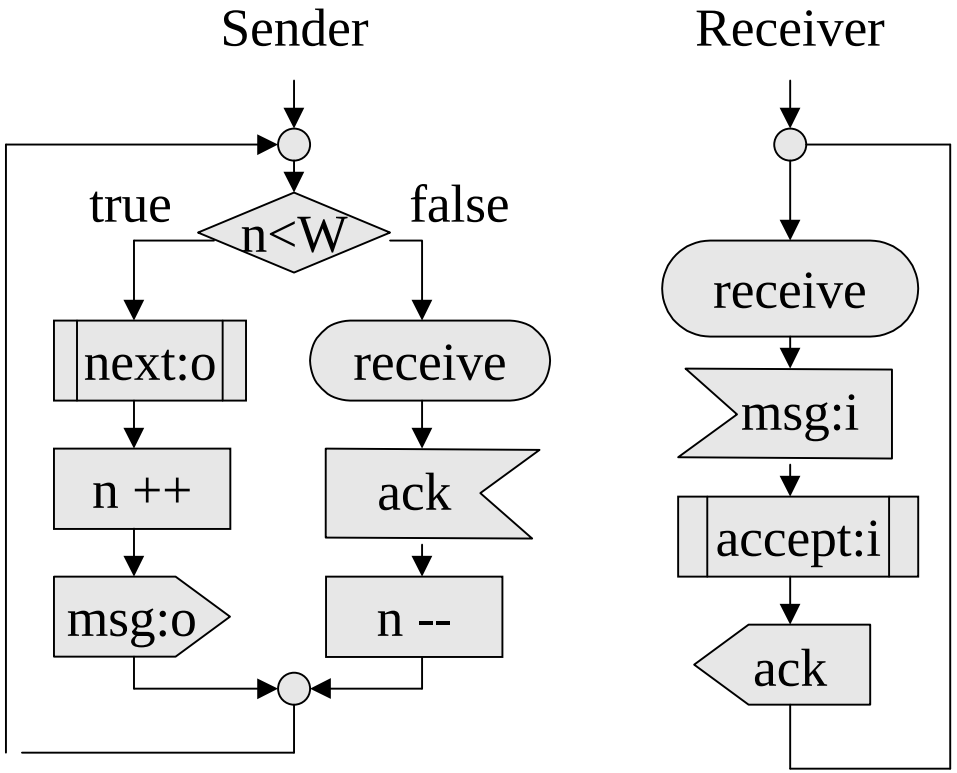
- Por cada mensaje transmitido decrementa 1 crédito
- Por cada ack recibido incrementa 1 crédito

Suponiendo un número inicial de créditos W:

Tema 4: CONTROL DE FLUJO

Ingeniería de Protocolos y Servicios

4.2. Protocolos de ventana



DTIC
Área Ingeniería Telemática
francesc.burrull@upct.es

4.2. Protocolos de ventana

La cantidad W-n indica el número de créditos sin usar.

Siendo a(t) el número de créditos recibidos por el emisor, m(t) el número de mensajes enviados, y n(t) el valor de n en el instante t, el número máximo de mensajes que el emisor tiene pendientes de recibir *ack* es:

W-n(t)+m(t)-a(t)

Que debe ser <=W (demostrar)

El valor de W se llama “tamaño de la ventana”.

4.2. Protocolos de ventana

PÉRDIDAS DE MENSAJES - *TIMEOUTS*

Si se pierden suficientes *ack* el sistema se bloquea: Es necesario considerar el tiempo transcurrido desde la transmisión de un mensaje hasta la recepción del *ack*.

Otro problema que surge es hacer corresponder cada *ack* con su mensaje: aparición de los números de secuencia

4.3. Números de secuencia

EJEMPLO: Protocolo de bit alternado

Si al protocolo de ping-pong se le añade la posibilidad de pérdidas -uso de timeouts- y el uso de números de secuencia (1 bit) se convierte en el protocolo de bit alternado con timeouts.

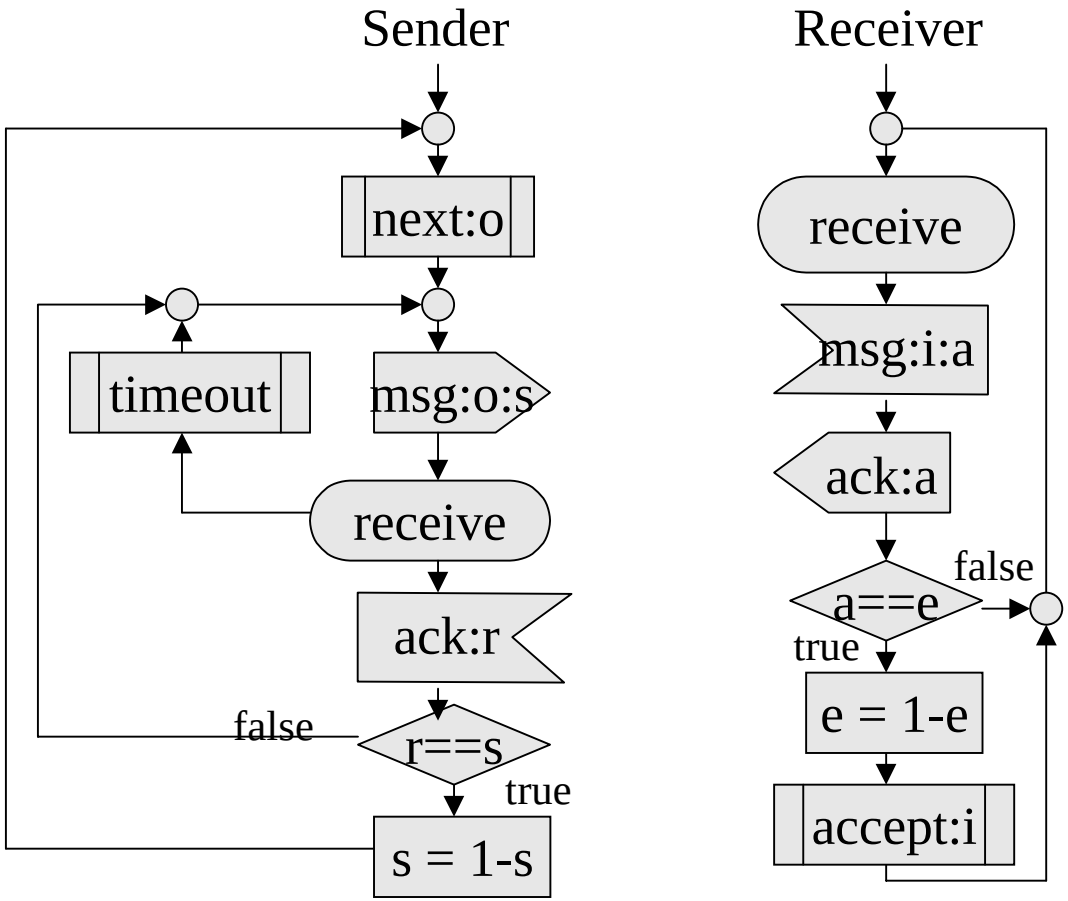
Se usan 2 tipos de mensaje:

- msg { msg, dato, nº secuencia }
- ack { ack, nº secuencia }

Tema 4: CONTROL DE FLUJO

Ingeniería de Protocolos y Servicios

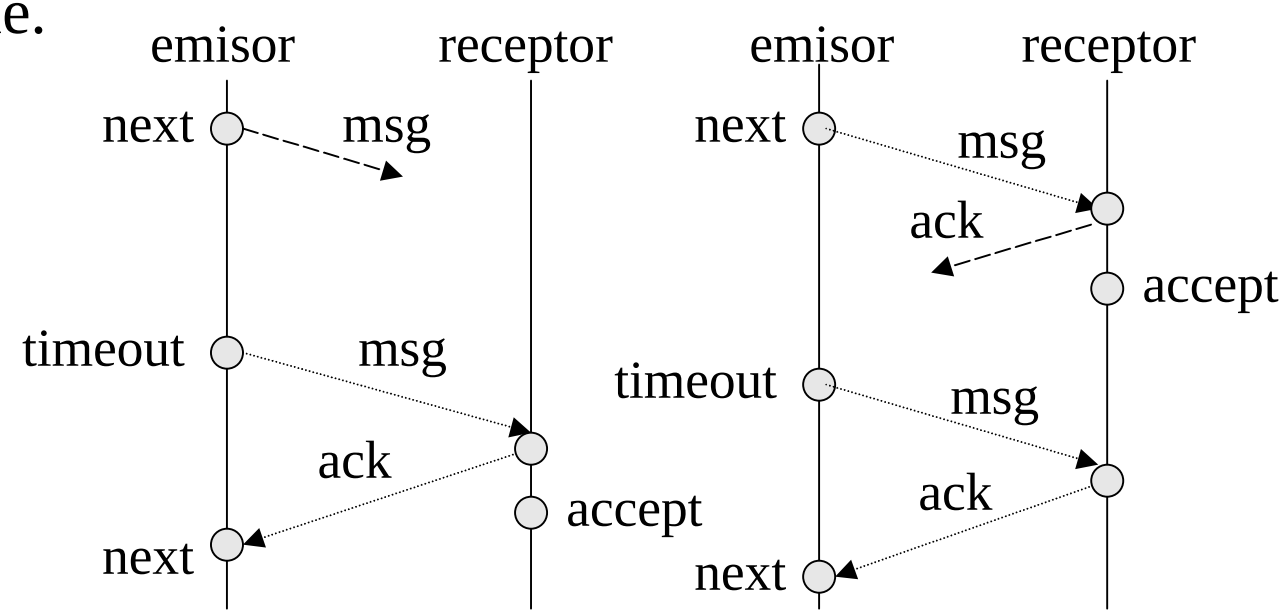
4.3. Núm. de sec.



DTIC
Área Ingeniería Telemática
francesc.burrull@upct.es

4.3. Números de secuencia

Ahora si hay pérdidas de mensajes la recuperación es posible.



DTIC
Área Ingeniería Telemática
francesc.burrull@upct.es

4.3. Números de secuencia

REORDENACIÓN DE MENSAJES

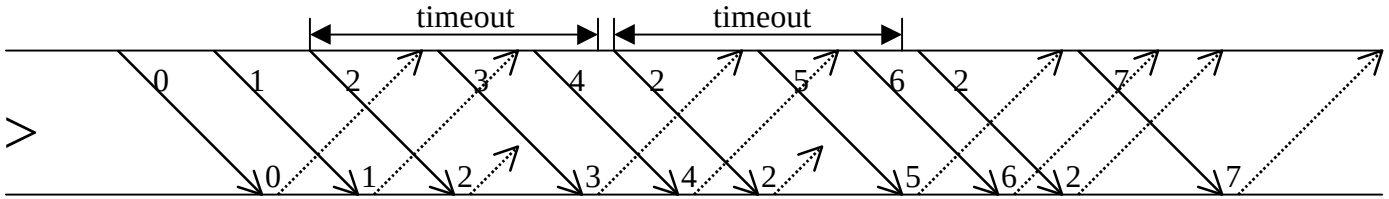
Los números de secuencia tienen un rango limitado (número finito de bits en una cabecera). ⇨ Reciclaje

¿Qué relación hay entre el tamaño de ventana W y los M números de secuencia disponibles?

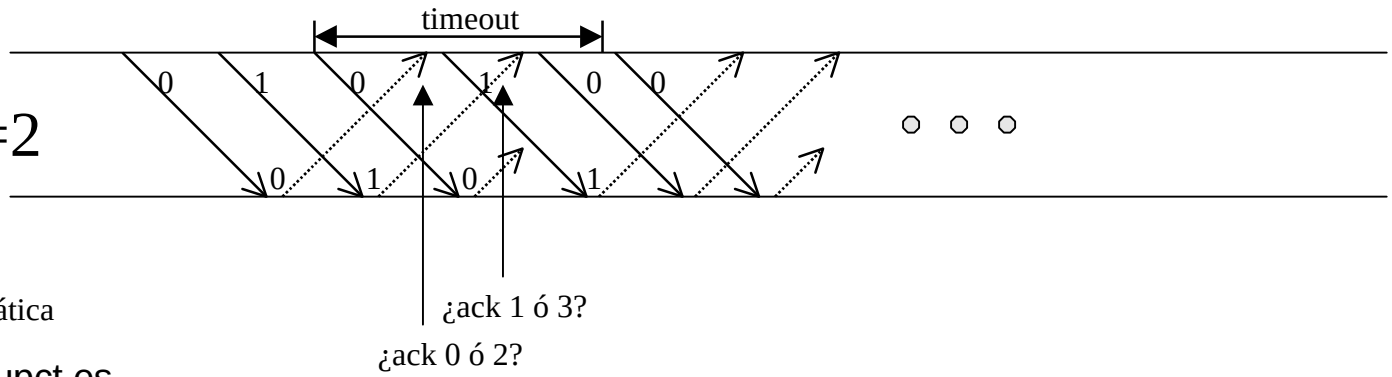
• Intuitivamente M tiene que ser suficientemente mayor que W para aparear cada mensaje con su *ack*.

4.3. Números de secuencia
REORDENACIÓN DE MENSAJES

W=3, M>>



W=3, M=2



DTIC
Área Ingeniería Telemática

francesc.burrull@upct.es

4.3. Números de secuencia

RELACIÓN TAMAÑO VENTANA - Nº SECUENCIA

M: rango de números de secuencia. W el tamaño de la ventana. H: Último mensaje reconocido por el receptor.

Se tiene que:

Mensaje más viejo que puede ser retransmitido:

- (H-W+1)%M

Mensaje más nuevo (futuro) que puede ser transmitido:

- (H+W)%M

Éstos mensajes deben ser distinguibles: M > 2W - 1

4.4. Reconocimiento negativo

Útil si la probabilidad de error es alta.

Cuando el emisor recibe un *nack* ya no debe esperar el *timeout* para saber que su mensaje llegó mal.

4.4. Reconocimiento negativo

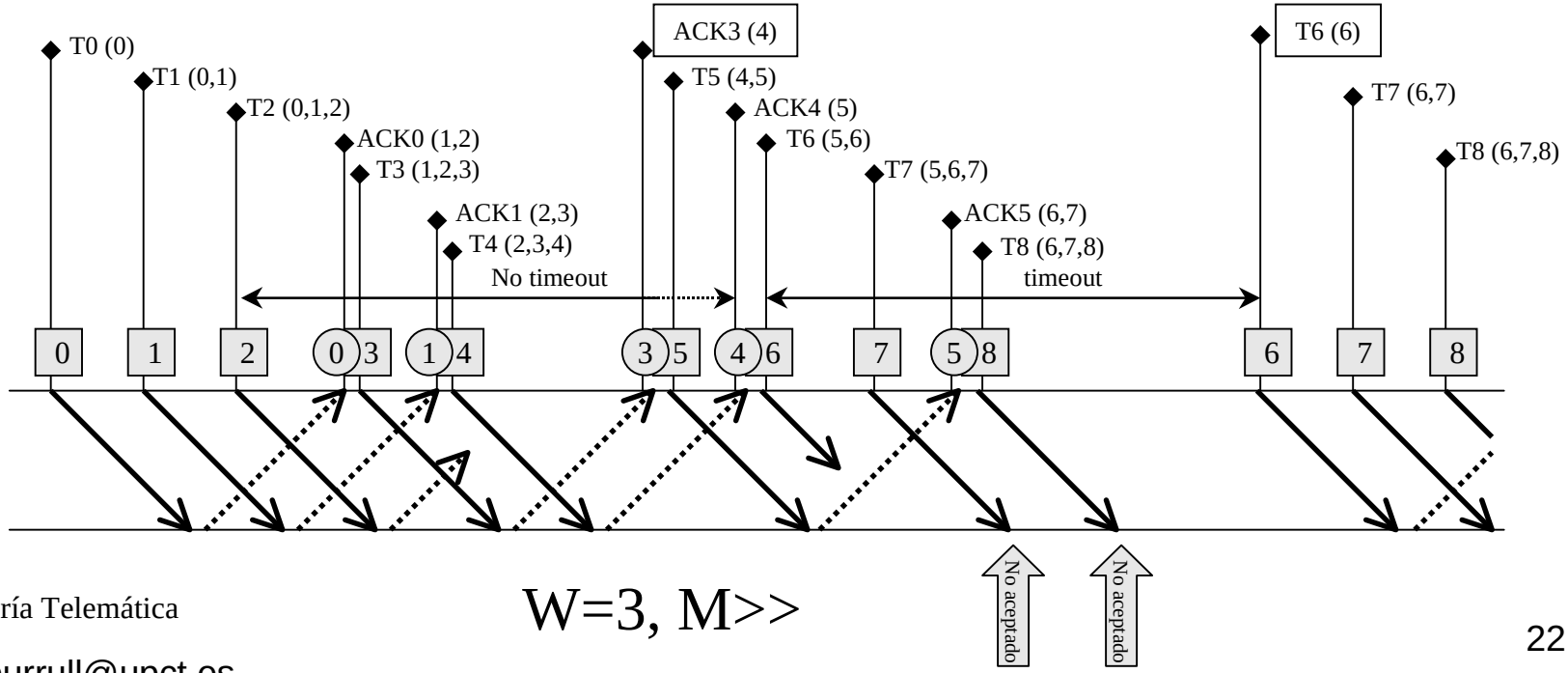
TERMINOLOGÍA

El uso de reconocimientos para la retransmisión de mensajes se conoce como método ARQ (*Automatic Repeat reQuest*). Las 3 variantes principales son:

- *Stop and Wait* ARQ
- *Selective repeat* ARQ
- *Go-back-N continuous* ARQ

4.4. Reconocimiento negativo

Ejemplo *Go-back-N continuous* ARQ



W=3, M>>

4.4. Reconocimiento negativo

RECONOCIMIENTO POR BLOQUES

Una variante consiste en enviar reconocimientos no para cada mensaje, sinó para bloques de mensajes.

En el caso de *Go-back-N*, el recibir un *ack* significa que todos los mensajes anteriores han llegado correctamente. Así, un ack puede significar que el rango anterior es correcto.

4.5. Prevención de la congestión

Para un cierto enlace, ¿Como se escoge W y M?

• Límite superior W: Saturación del canal

Ejemplo:

Tiempo Emisor-Receptor: 0.5 segundos.

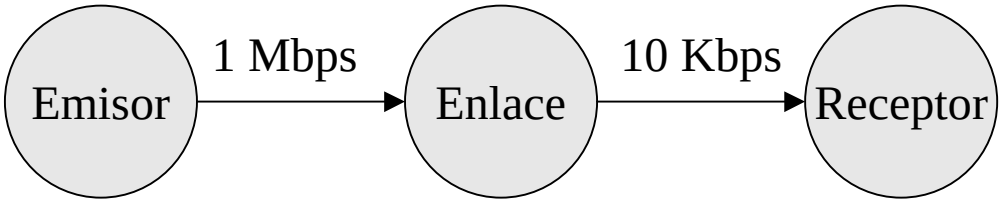
Velocidad de transmisión: 1200bps.

El emisor satura el canal si transmite durante 1 segundo.

Si los mensajes son de 8 bits, la ventana máxima seria de 150 mensajes (mayor sería un desperdicio).

4.5. Prevención de la congestión

Considerando la red siguiente:



Se puede definir el protocolo de control de flujo como:

- Nodo a nodo
- Extremo a extremo

DTIC
Área Ingeniería Telemática

francesc.burrull@upct.es

4.5. Prevención de la congestión

Nodo a nodo:

Calcular W por separado en ambos tramos (saturar ambos)

Aparece un problema mayor: Enlace recibe datos 100 veces mas deprisa que los transmite: Desbordamiento.

Un esquema de control de flujo debería optimizar el uso de:

- Espacio de buffer de los nodos intermedios.
- Ancho de banda de los enlaces entre nodos.

Se hace necesario algún mecanismo de *feedback* entre nodos

4.5. Prevención de la congestión

Extremo a extremo:

La capacidad del enlace es la del tramo más lento, y a partir de dicho tramo se calcula W.

Se desperdicia tiempo en los demás tramos.

Si la red evoluciona, se debería adaptar el tráfico ofrecido.

Una solución para reducir el tráfico ofrecido es reducir W.

4.5. Prevención de la congestión

CONTROL DE FLUJO DINÄMICO

El método de control de flujo de ventana dinámica hace al protocolo auto-adaptarse al canal.

El emisor decrementa el valor de W cuando se produce un *timeout*. Lo incrementa gradualmente hasta W máximo.

Tres variaciones usadas son:

- *Timeout* decrementa 1, *ack* incrementa 1
- *Timeout* decrementa a mitad, N *ack* incrementan 1
- *Timeout* decrementa a 1, N *ack* incrementan 1

4.6. Resumen

Un esquema de control de flujo puede usarse para coordinar la velocidad de transmisión de mensajes entre procesos de un sistema distribuido, para evitar cuellos de botella y para recuperación del sistema en caso de errores de transmisión.

Se han visto *timeouts*, números de secuencia, *ack* y *nack*.

Extremo a extremo sincronizan.

Nodo a nodo optimizan manejo de buffer y uso del ancho de banda