

Paradigmas de conmutación

Area de Ingeniería Telemática
<http://www.tlm.unavarra.es>

Arquitectura de Redes, Sistemas y Servicios
Grado en Ingeniería en Tecnologías de
Telecomunicación, 2º

Temario

1. Introducción
2. **Arquitecturas de conmutación y protocolos**
 - Elementos, protocolos y arquitecturas de protocolos
 - Arquitecturas OSI y TCP/IP
 - Servicios, interfaces, funcionalidades
 - Conmutación de circuitos y de paquetes
 - **Retardos de transmisión, propagación, procesado, cola**
 - **Variación del retardo, pérdidas y throughput**
3. Introducción a las tecnologías de red
4. Control de acceso al medio
5. Conmutación de circuitos
6. Transporte fiable
7. Encaminamiento
8. Programación para redes y servicios

Objetivos

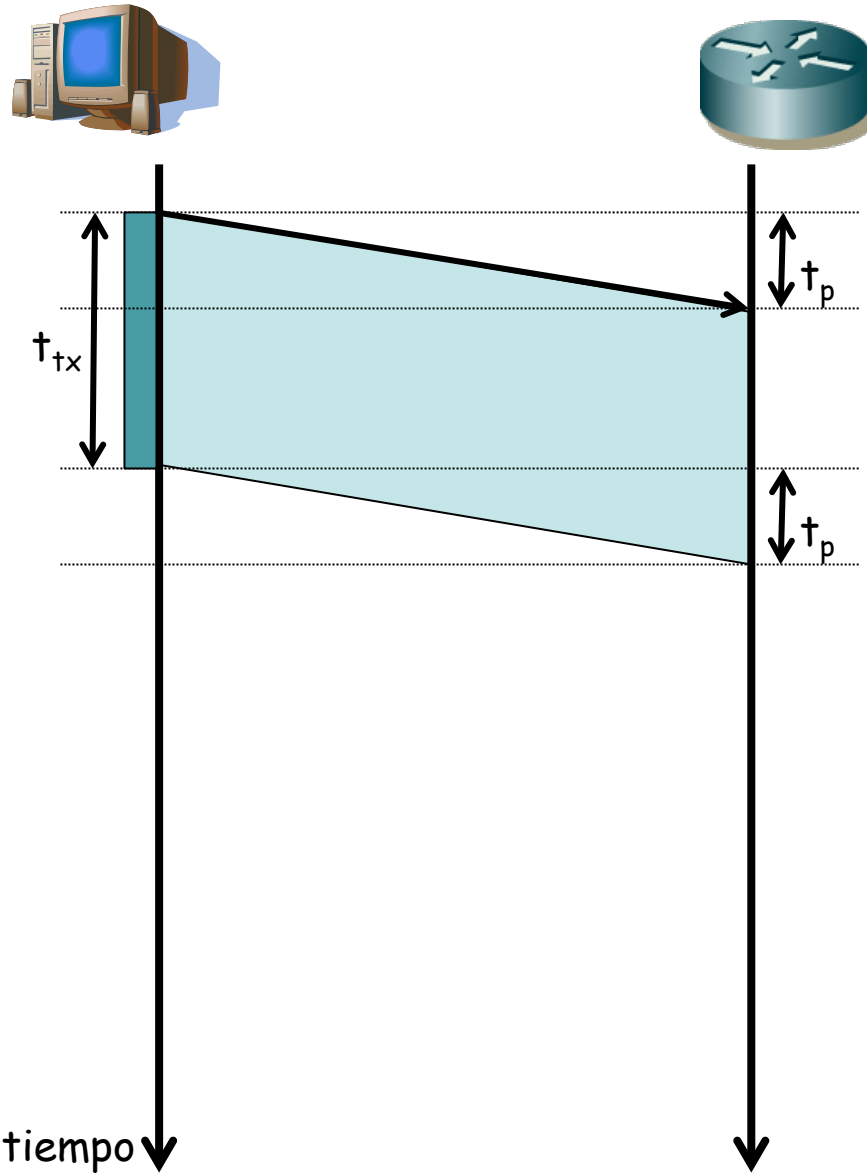
- Diferenciar y saber trabajar con **retardos** de propagación y transmisión en redes con almacenamiento y reenvío
- Comprender el origen y comportamiento general del retardo en cola
- Saber que existe la **variación del retardo** en redes de conmutación de paquetes, a qué se debe y qué efectos tiene
- Conocer la existencia y los motivos de las **pérdidas** en redes de conmutación de paquetes
- Entender qué es un cuello de botella

Contenido

- Retardos
 - Retardo de procesado
 - Retardo en cola
- Efectos del tamaño del paquete
- Throughput
- Packet Delay Variation
- Pérdidas
- Problemas de circuitos y paquetes

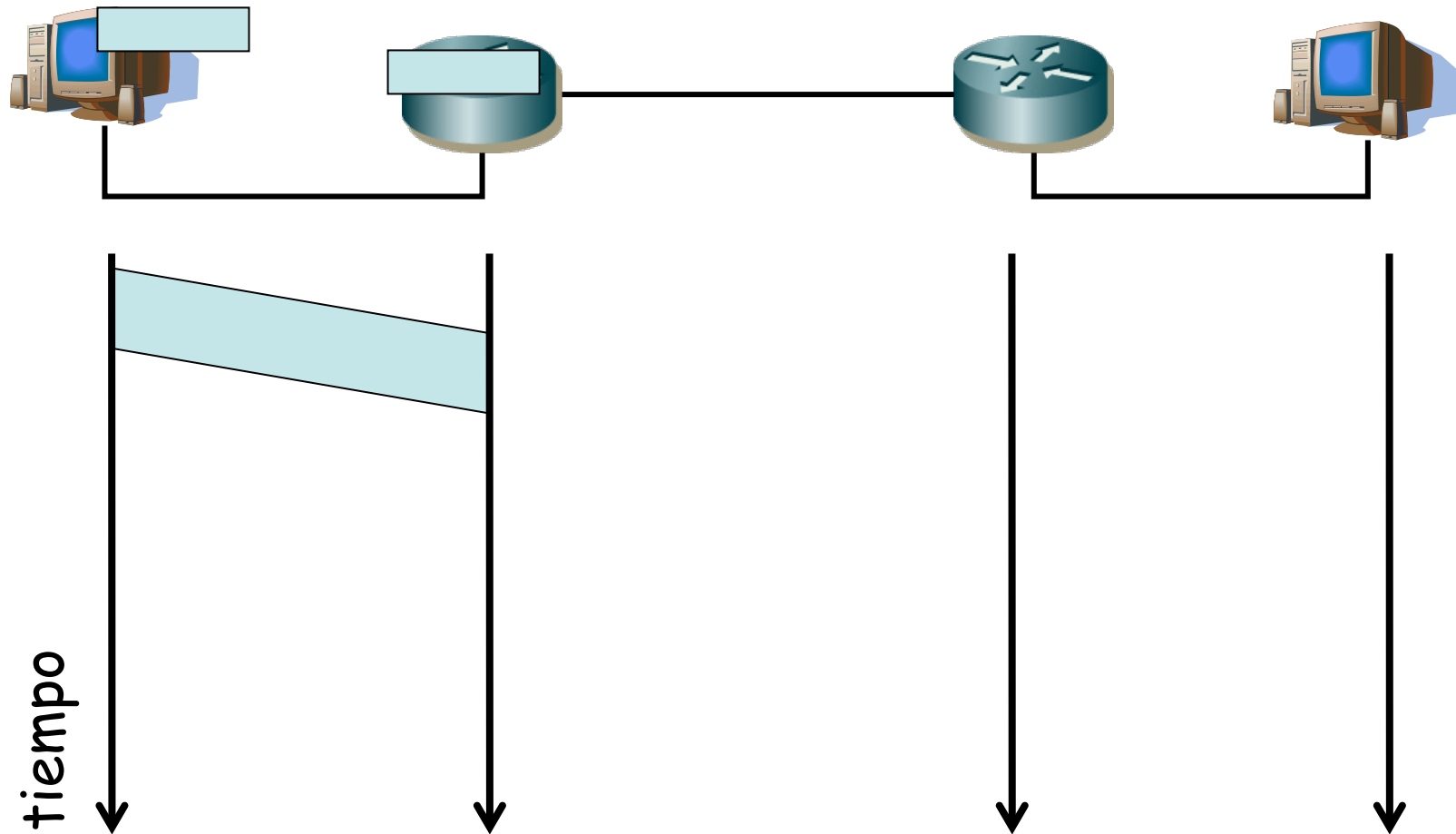
Retardos en conmutación de paquetes

Transmisión y propagación



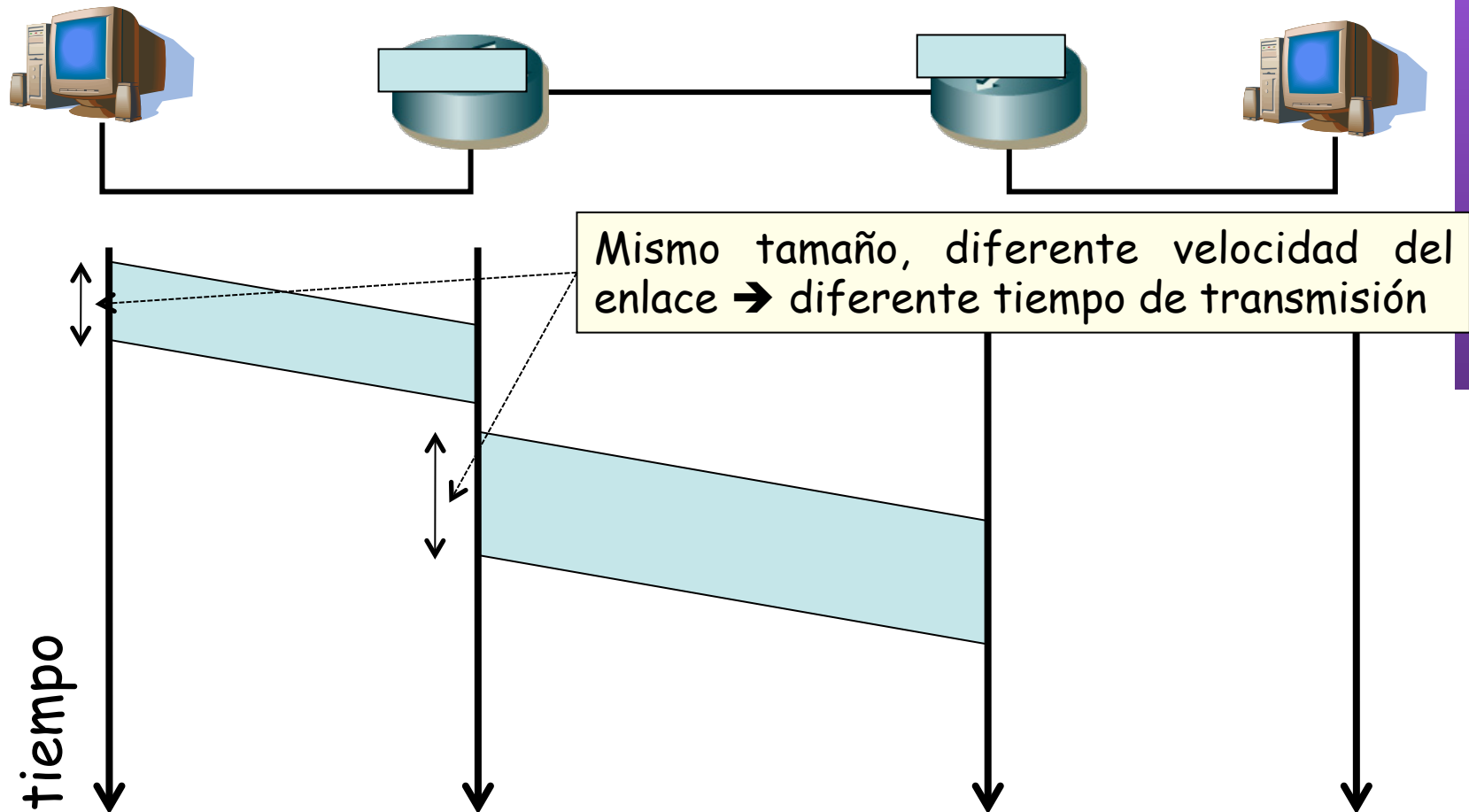
Store-and-forward

- El paquete completo debe llegar al conmutador de paquetes antes de que lo pueda retransmitir (. . .)



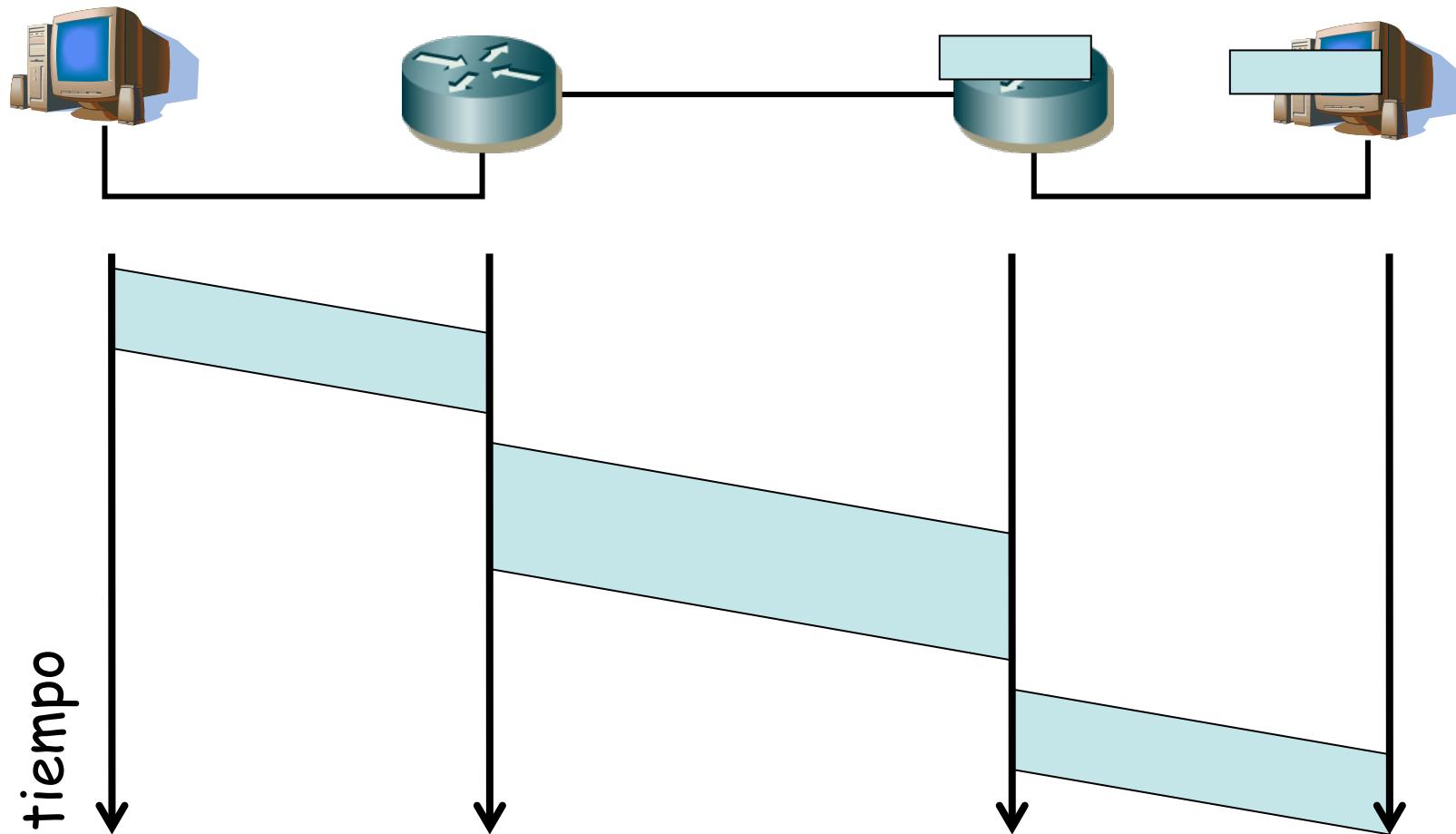
Store-and-forward

- El paquete completo debe llegar al conmutador de paquetes antes de que lo pueda retransmitir (. . .)



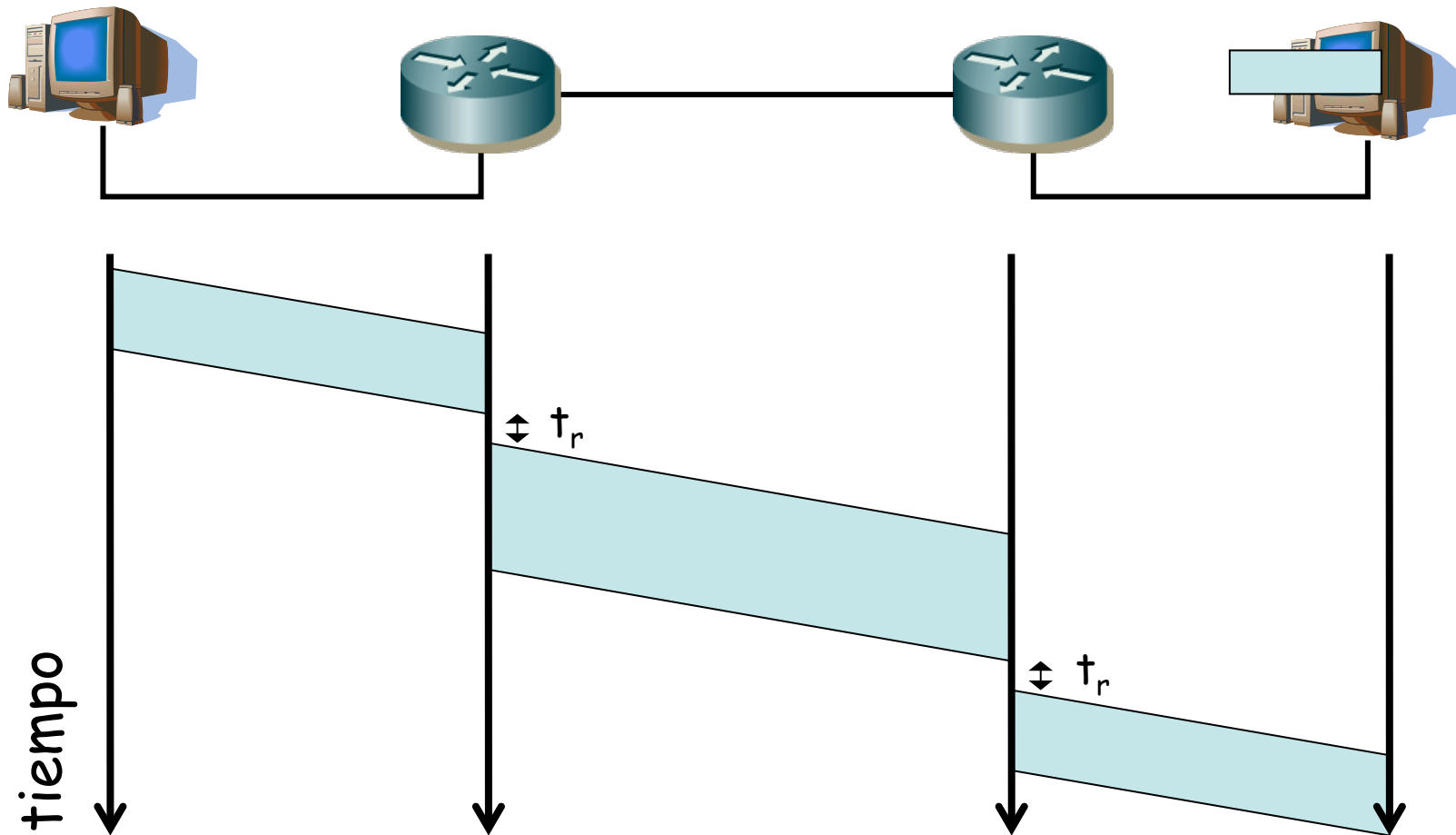
Store-and-forward

- El paquete completo debe llegar al conmutador de paquetes antes de que lo pueda retransmitir (. . .)



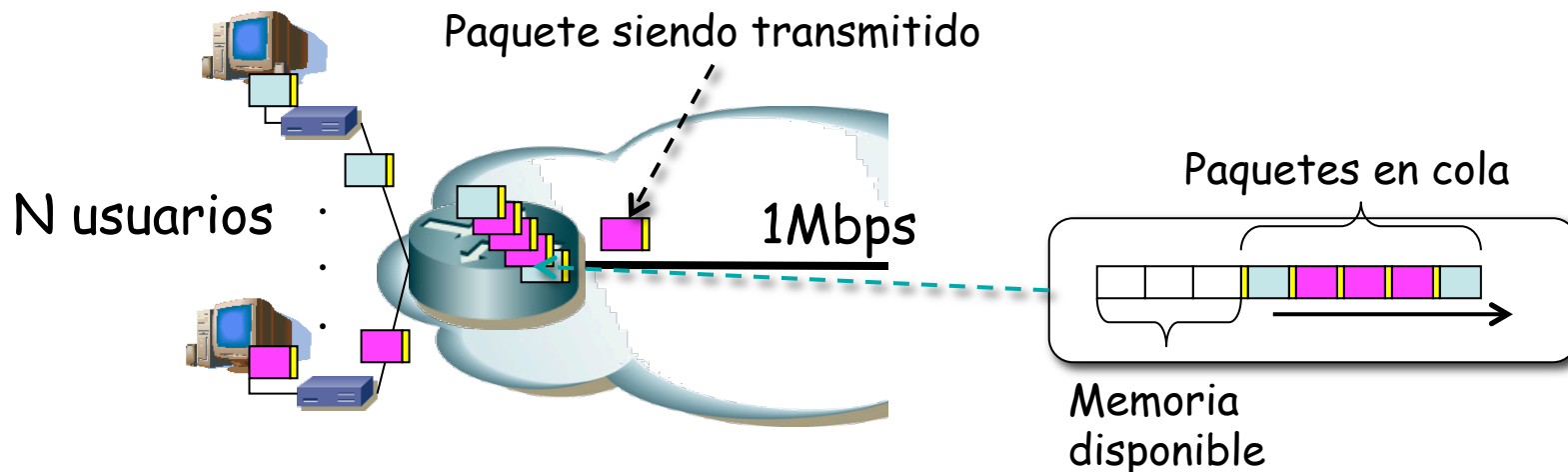
Tiempo de procesamiento

- El conmutador debe tomar una decisión para cada paquete, la cual lleva tiempo (t_r)



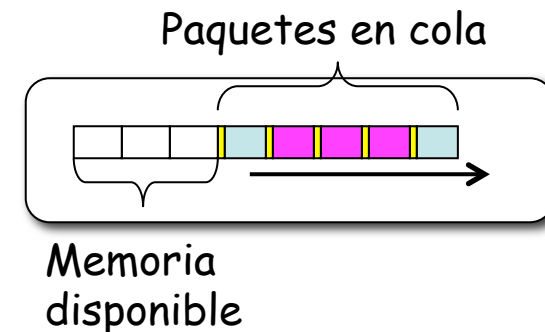
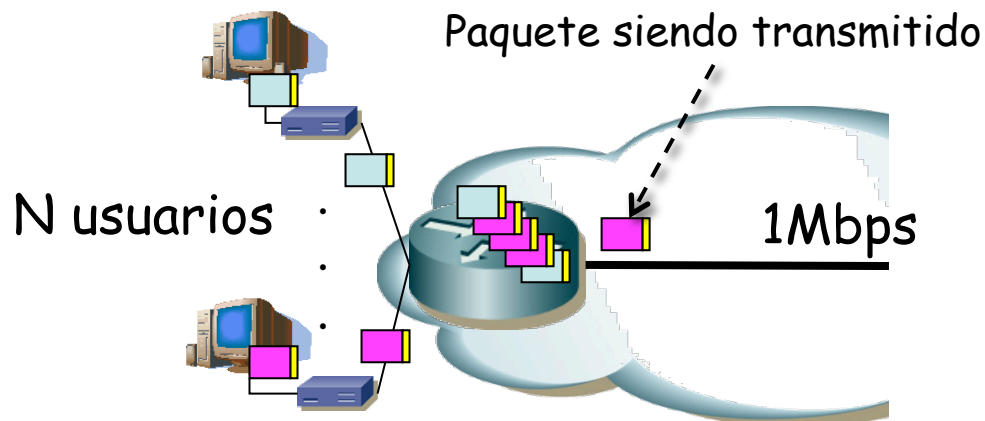
Retardo en cola

- Los paquetes pueden llegar al router a una velocidad mayor que la capacidad del enlace de salida
- O pueden llegar varios simultáneamente por enlaces diferentes pero solo puede salir uno a la vez
- El router los almacena en memoria hasta poder enviarlos
- Esperan en una *cola* (normalmente en el interaz de salida)
- Si no queda espacio en memoria para almacenar un paquete, normalmente éste se pierde (*drop-tail policy*)



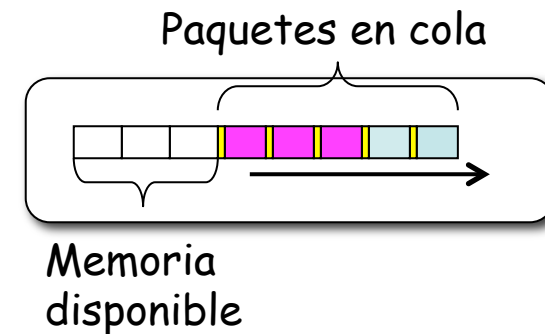
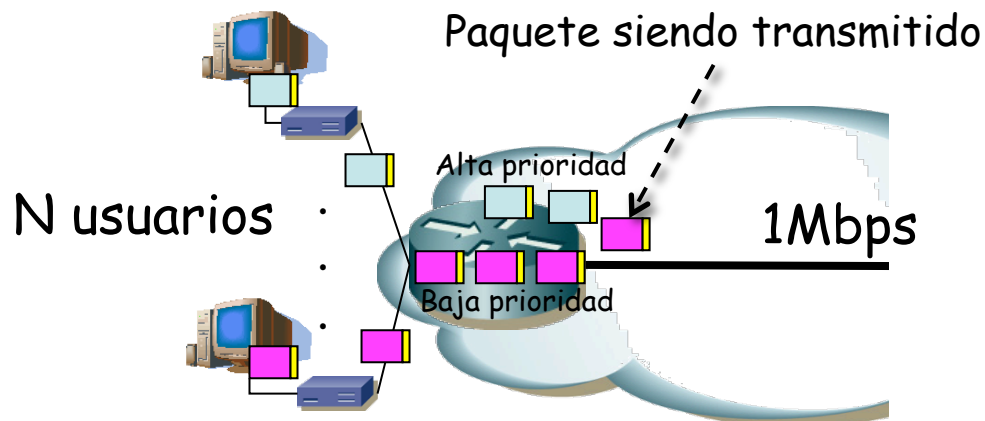
Planificación

- ¿ En qué orden se atiende a los paquetes que hay en la cola ?
- FCFS: *First Come First Served*
 - También llamado FIFO (First In First Out)
 - Trato equitativo a diferentes flujos/usuarios/aplicaciones
 - Un paquete que requiera bajo retardo (voz) tiene que esperar a que se sirvan todos los anteriores en la cola
 - Asegurar límites en el retardo requiere caracterizar todas las fuentes de tráfico



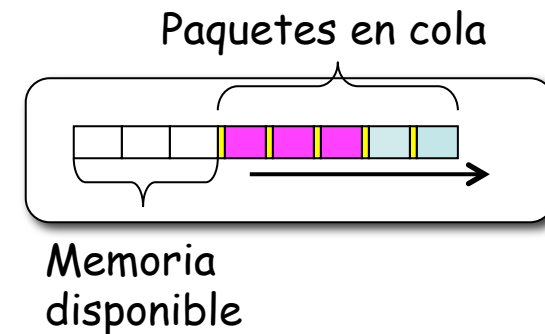
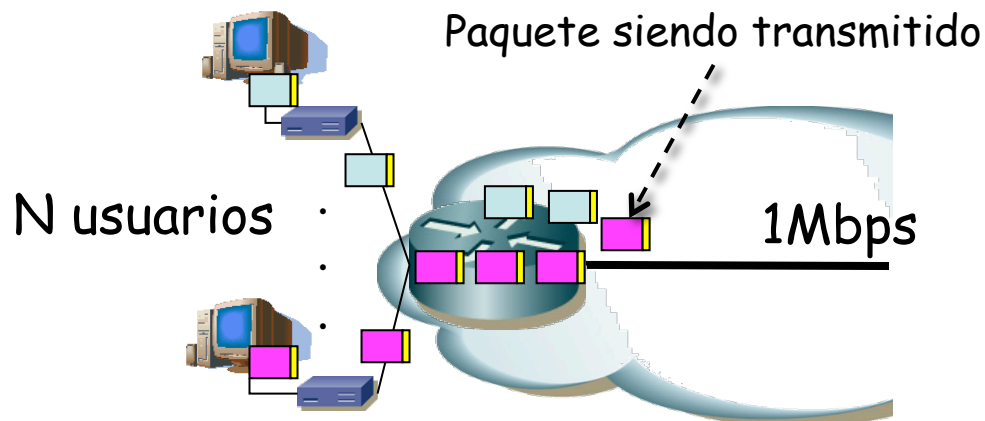
Planificación

- ¿ En qué orden se atiende a los paquetes que hay en la cola ?
- ¿Otras alternativas?
- Prioridades:
 - Clasificar los paquetes de entrada
 - Cada clase tiene una prioridad diferente
 - Solo se envían paquetes de una clase si las clases de prioridad superior no tienen paquetes en la memoria del router



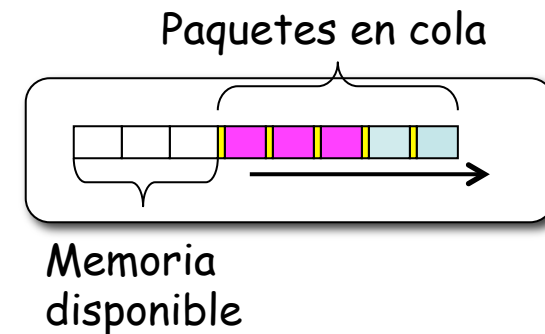
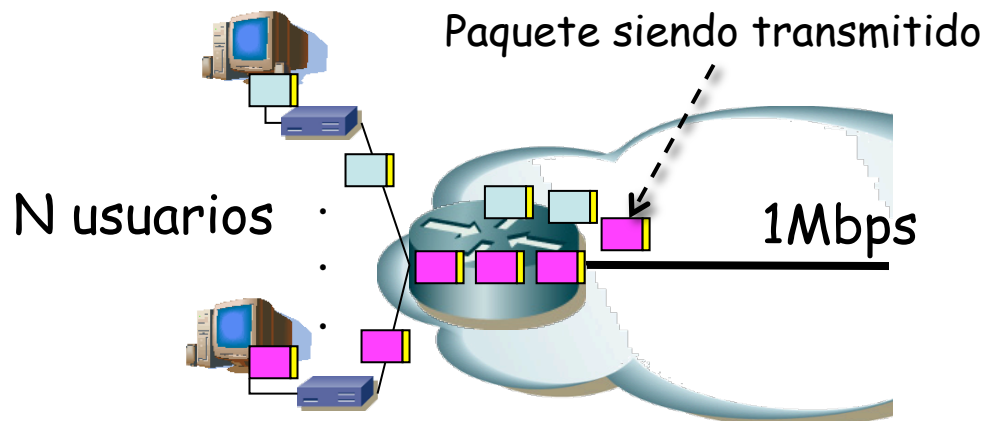
Planificación

- ¿ En qué orden se atiende a los paquetes que hay en la cola ?
- ¿Otras alternativas?
 - Round Robin
 - Weighed Round Robin
 - Deficit Round Robin
 - Generalized Processor Sharing
 - Weighed Fair Queueing
 - ...



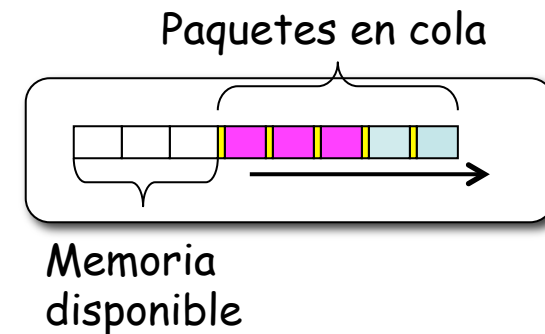
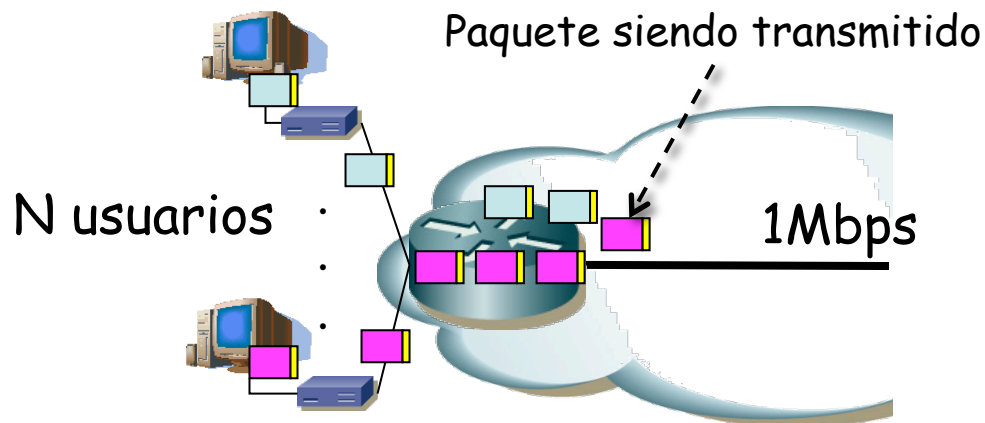
Planificación

- ¿ En qué orden se atiende a los paquetes que hay en la cola ?
- ¿Otras alternativas?
- **Buscan:**
 - Hacer un reparto “justo” (*max-min fair*)
 - Protección: un flujo no pueda acaparar todos los recursos
 - Asegurar límites (al retardo, jitter, pérdidas...) predecibles
 - Simplicidad de implementación

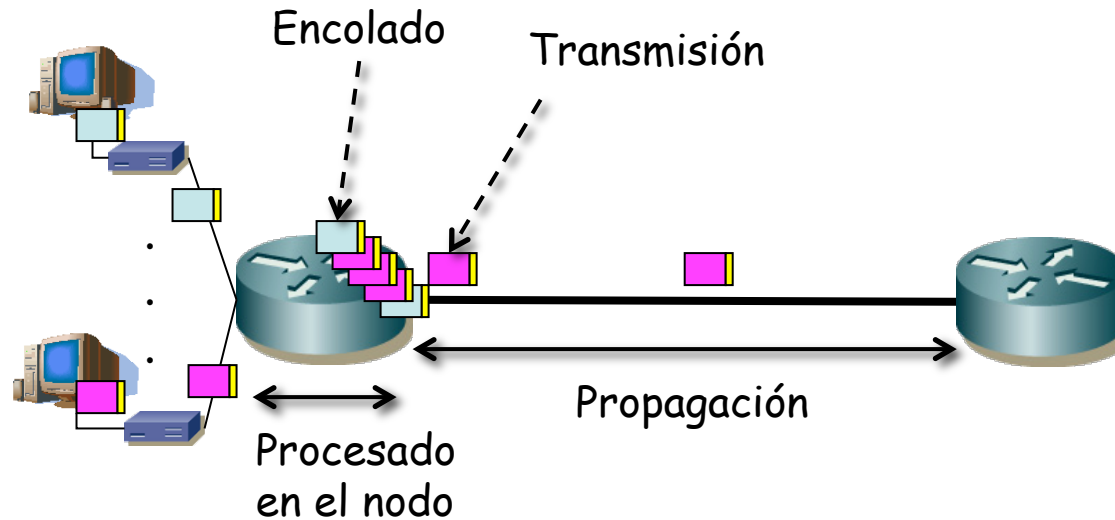


Planificación

- ¿ En qué orden se atiende a los paquetes que hay en la cola ?
- ¿Otras alternativas?
- **Necesario para:**
 - Ofrecer Calidad de Servicio (QoS, *Quality of Service*)



Retardos



$$d_{\text{nodo}} = d_{\text{proc}} + d_{\text{cola}} + d_{\text{trans}} + d_{\text{prop}}$$

d_{proc} = tiempo de procesado

- Unos μs

d_{cola} = retardo en cola

- Depende de la congestión

d_{trans} = retardo transmisión

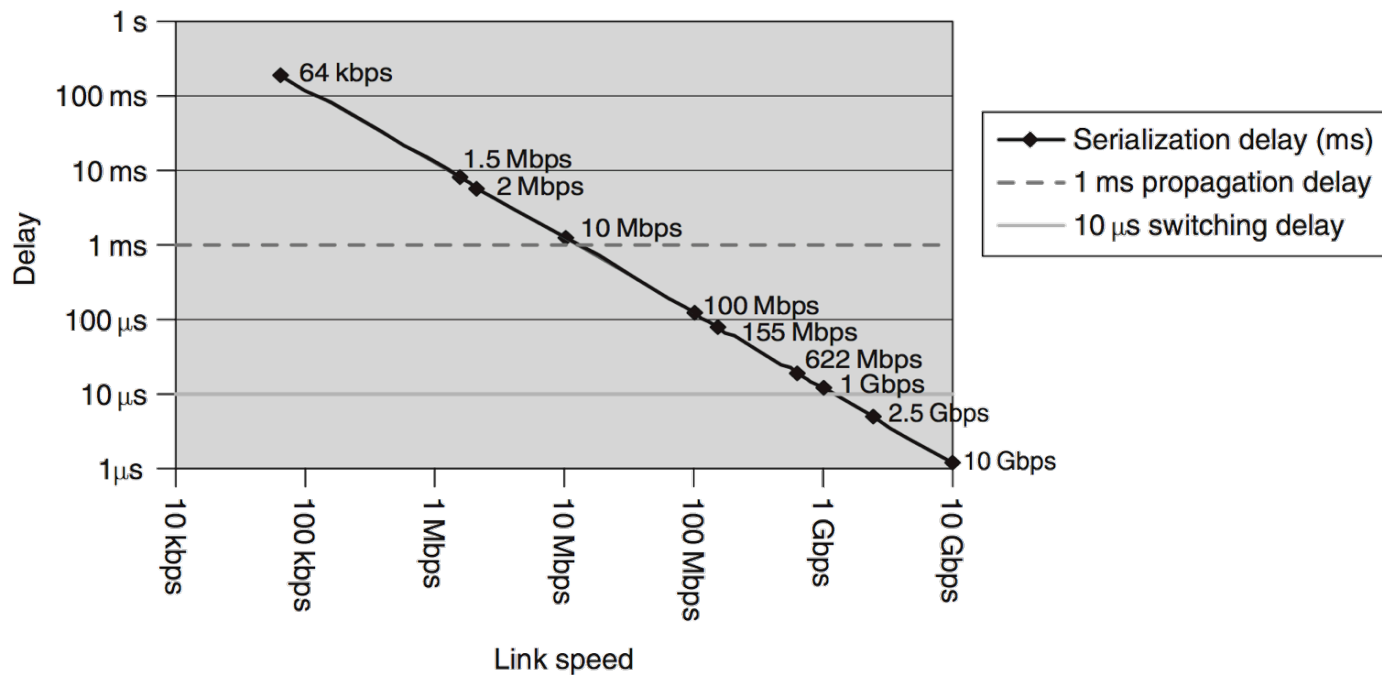
- = L/R , significativo en enlaces de baja velocidad

d_{prop} = retardo propagación

- De unos μs a centenares de ms

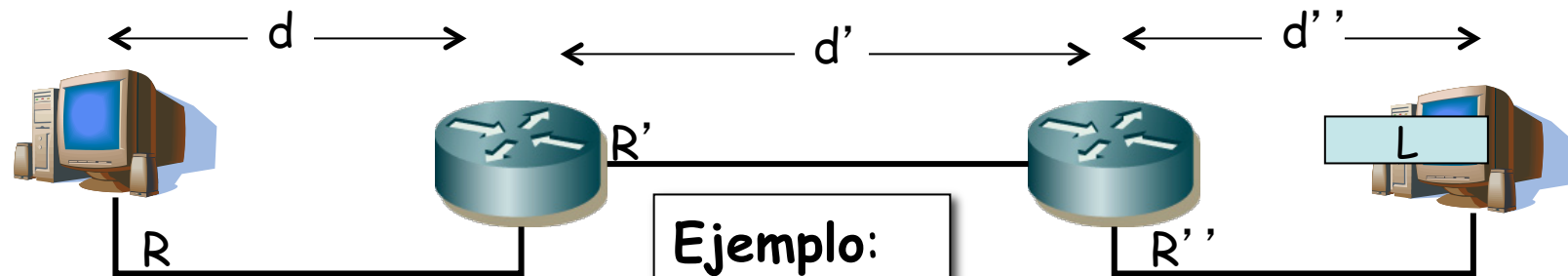
Retardos constantes

- Por cada salto intermedio habrá
 - Tiempo de transmisión (serialización)
 - Tiempo de propagación
 - Tiempo de procesado/conmutación
- Ejemplo comparativo
 - Paquete de 1500 bytes
 - Unos 200Km de fibra : 1ms de propagación



Ejemplo

- Conmutación de paquetes



Ejemplo:

- $R = R'' > R'$
- $s = s' = s''$
- $t_r = t_r'$
- no encola

$$\text{Delay} = L/R + d/s + t_r + L/R' + d'/s' + t_r' + L/R'' + d''/s'' =$$

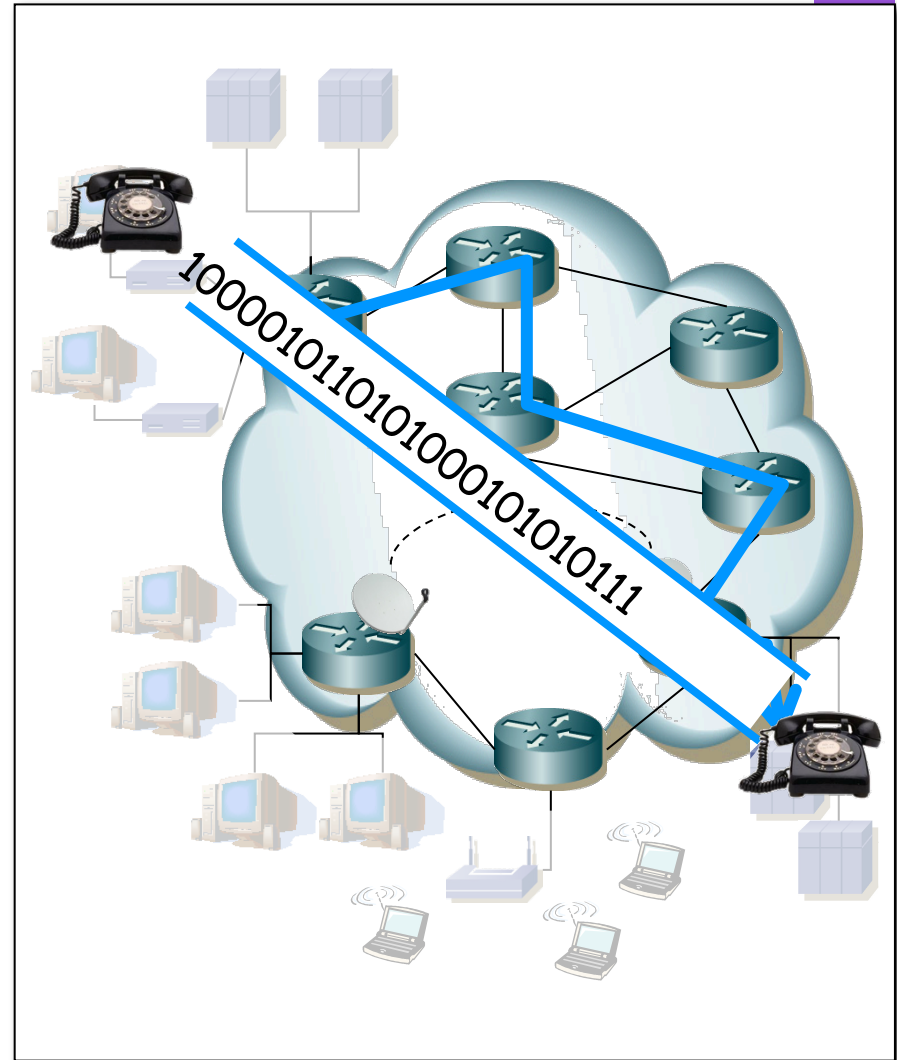
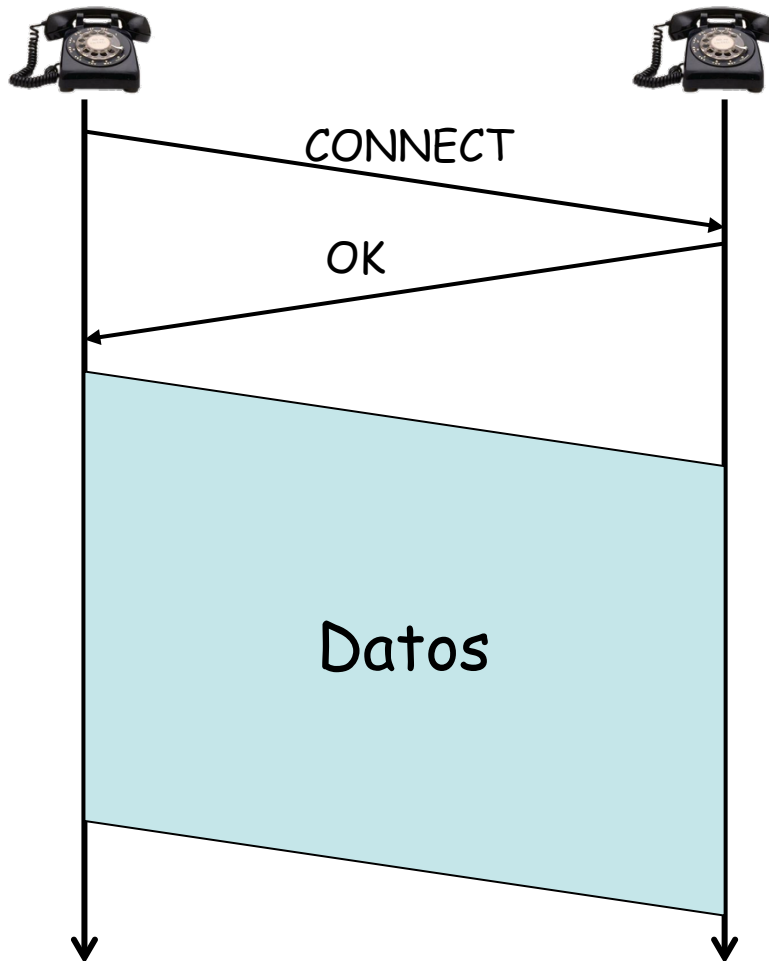
$$= 2L/R + L/R' + (d+d'+d'')/s + 2t_r$$

tiemp



Comparar con...

- Conmutación de circuitos



Contenido

- Retardos
 - Retardo de procesado
 - Retardo en cola
- Efectos del tamaño del paquete
- Throughput
- Packet Delay Variation
- Pérdidas
- Problemas de circuitos y paquetes

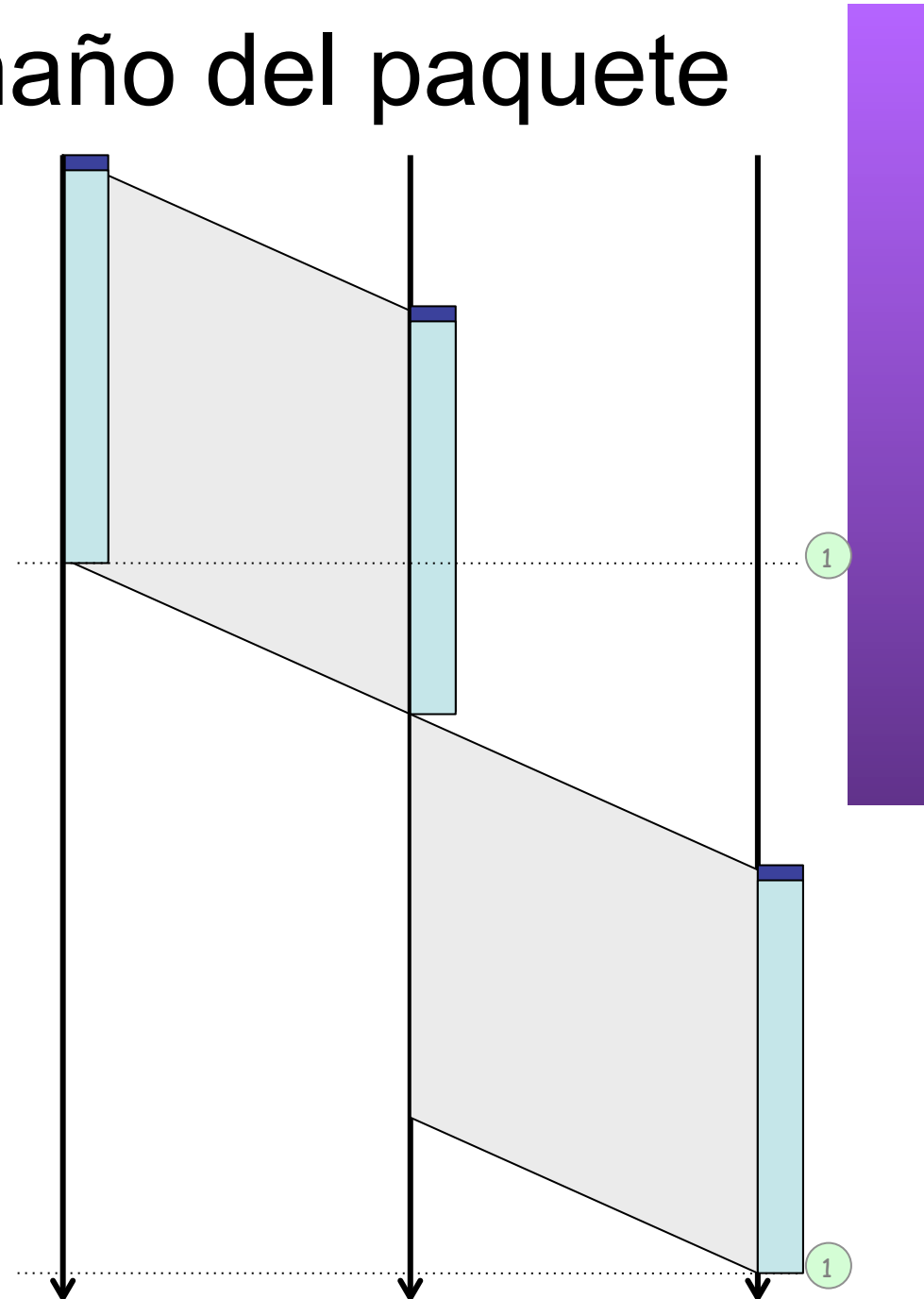
Efecto del tamaño del paquete

Mayor tamaño:

- Menos cabeceras, más eficiencia

Menor tamaño:

- (...)



Efecto del tamaño del paquete

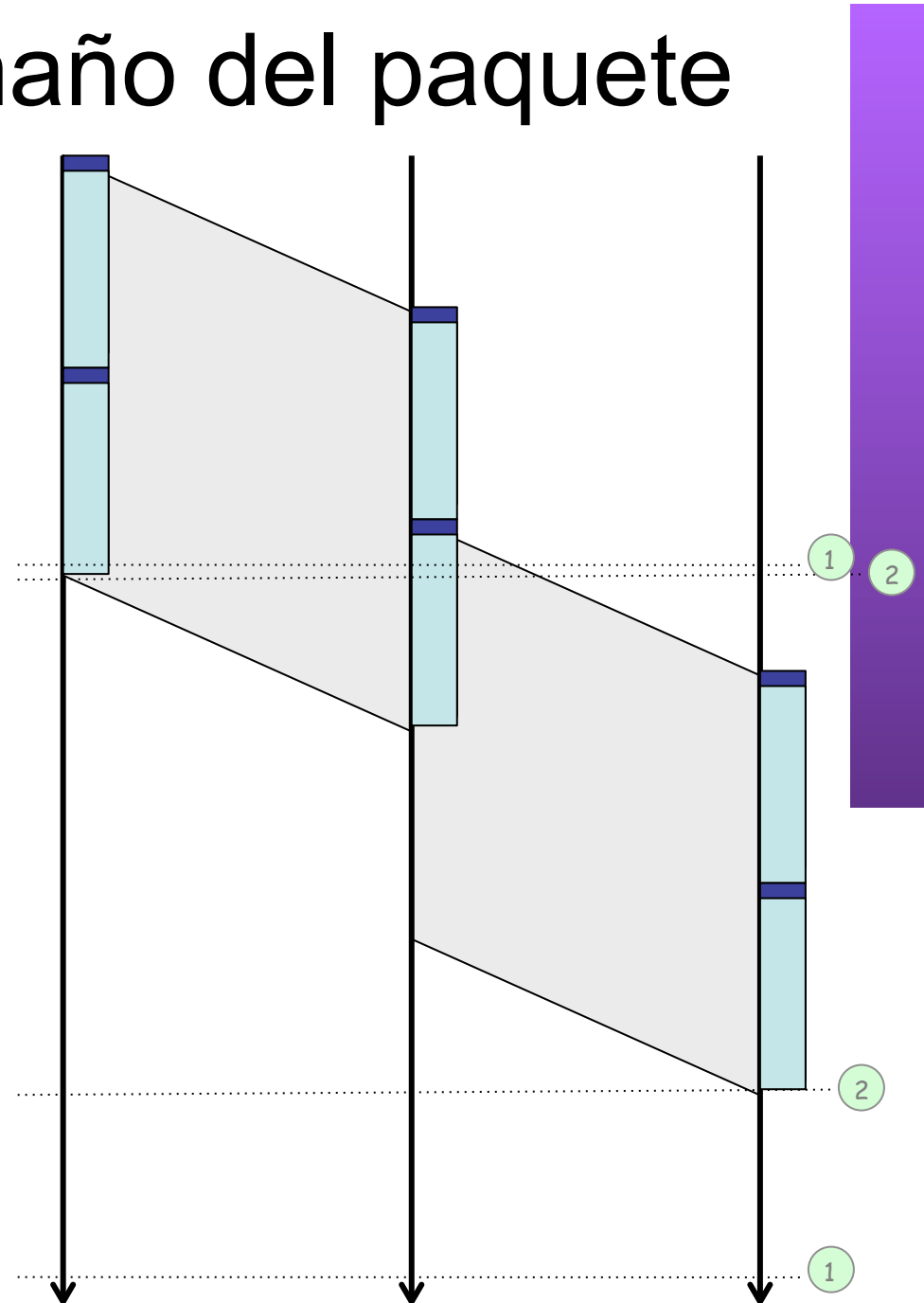
Mayor tamaño:

- Menos cabeceras, más eficiencia

Menor tamaño:

- Menos tiempo a esperar por *store and forward*

(...)



Efecto del tamaño del paquete

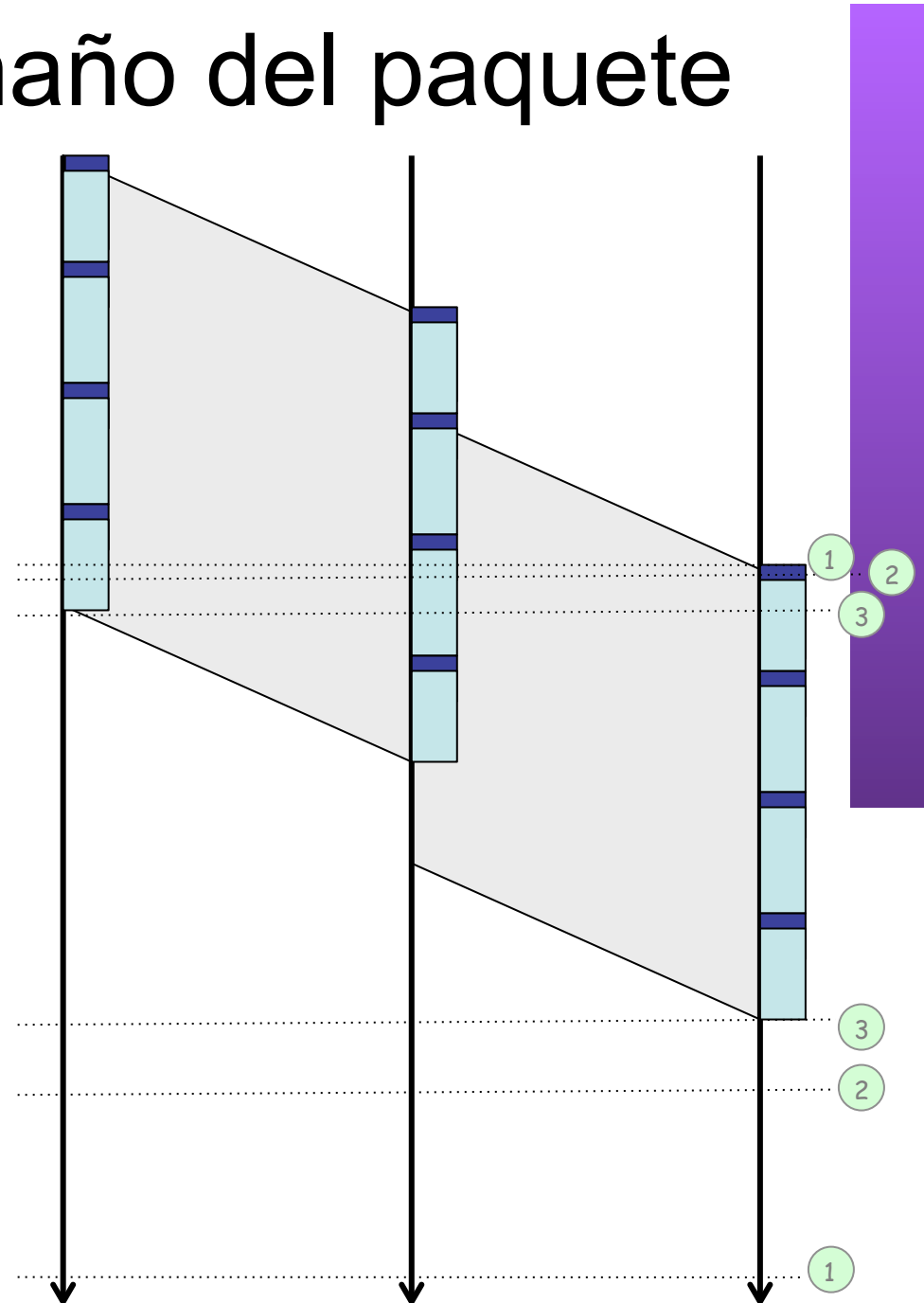
Mayor tamaño:

- Menos cabeceras, más eficiencia

Menor tamaño:

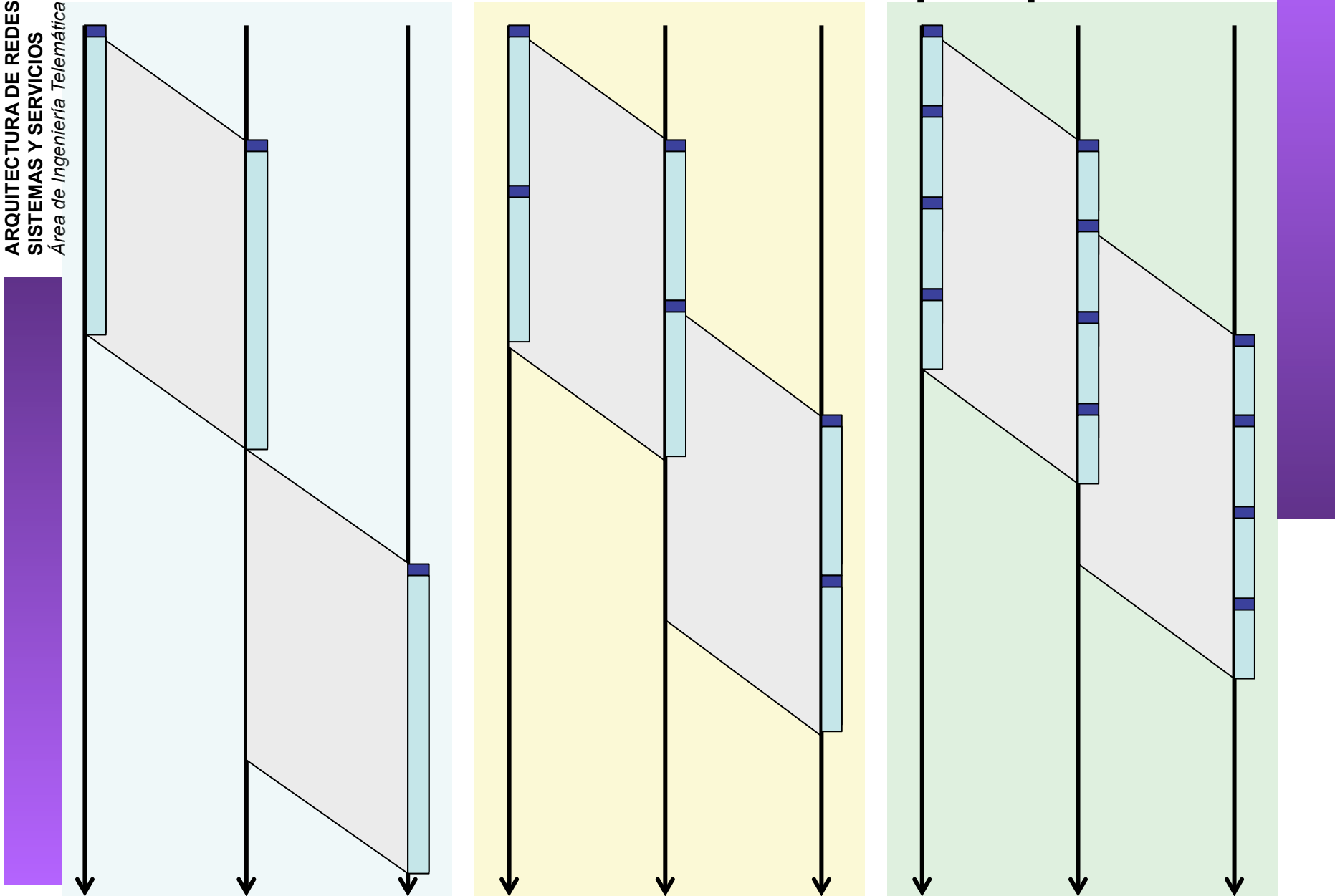
- Menos tiempo a esperar por *store and forward*

(...)



Efecto del tamaño del paquete

ARQUITECTURA DE REDES,
SISTEMAS Y SERVICIOS
Área de Ingeniería Telemática

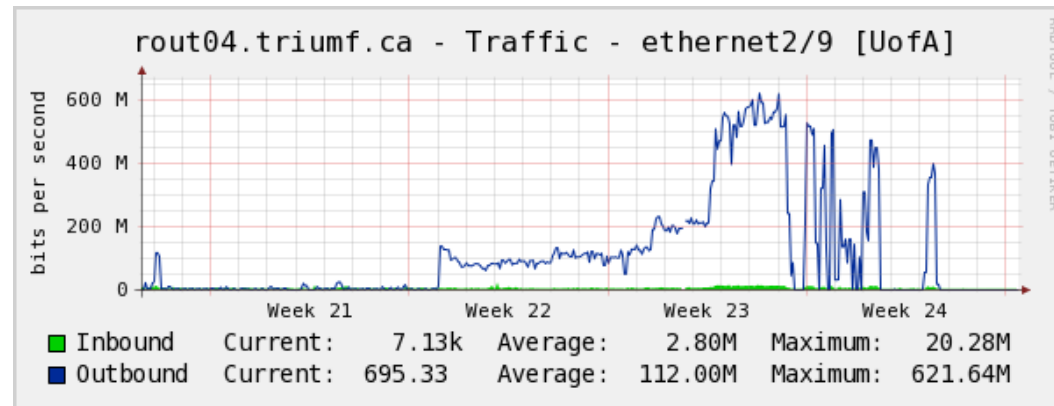


Contenido

- Retardos
 - Retardo de procesado
 - Retardo en cola
- Efectos del tamaño del paquete
- Throughput
- Packet Delay Variation
- Pérdidas
- Problemas de circuitos y paquetes

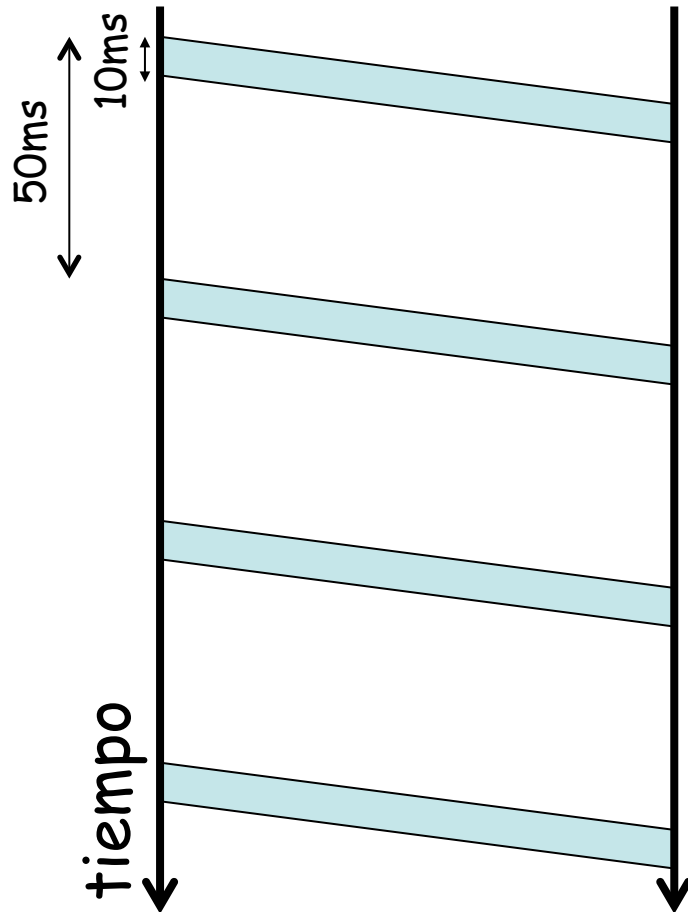
Throughput

- Throughput instantáneo: tasa a la cual se transmiten o transfieren o reciben datos
- Throughput medio: cantidad de datos transferidos en un intervalo de tiempo divididos por ese tiempo
- Ejemplo: transferencia de fichero de tamaño F bits en un tiempo T segundos ha sido en media a F/T bps
- En realidad, throughput instantáneo medido por debajo del tiempo de un paquete es el bit rate del enlace
- Medido por encima de esa escala es un throughput medio

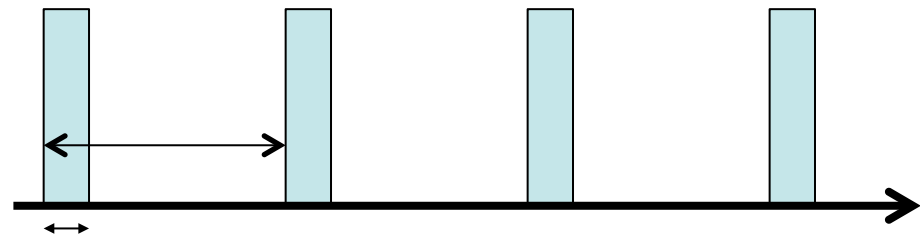


Throughput medio

- Supongamos un paquete de 1250 bytes exactamente cada 50 ms en un enlace a 1 Mbps
- Tiempo de transmisión = $1250 \times 8 / 1.000.000 = 10 \text{ ms}$

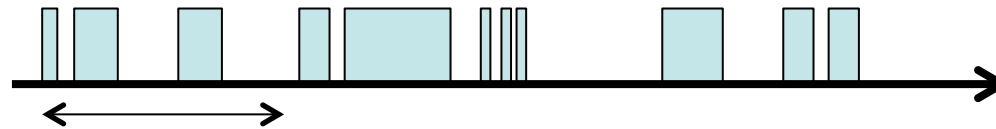


- Por ejemplo cada segundo se habrían transmitido:
 $1000 \text{ ms} / 50 \text{ ms/pkt} = 20 \text{ pkts}$
- 20 pkts/s
- $1250 \text{ bytes/pkt} \times 20 \text{ pkts/s} =$
 $= 200.000 \text{ bps} = 200 \text{ Kbps}$
- Es independiente del tiempo de propagación

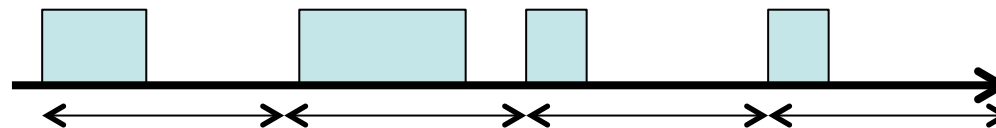


Throughput medio

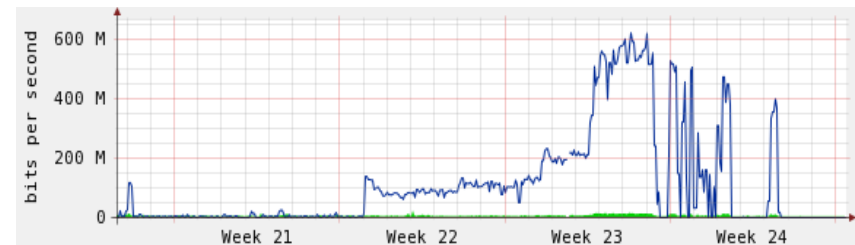
- Supongamos unas llegadas más irregulares (y más habituales)
- Paquetes de diferentes tamaños
- Paquetes con separaciones variables



- ¿Cuál es el throughput medio?
- Podemos tomar un intervalo “grande” y agregar los bytes enviados o recibidos en ese intervalo
- En cada intervalo es la cantidad de bytes transmitidos entre la anchura del intervalo



- Según anchura del intervalo
- Promedios más “groseros”



[Un apunte: unidades]

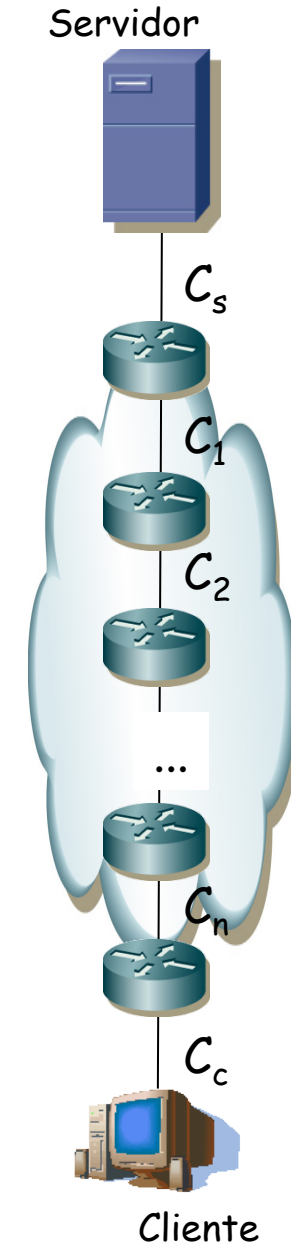
- 8 bits = 1 Byte
- Cuando hablamos de datos transferidos
 - Para mayores cantidades se usan potencias de 2 en vez de potencias de 10
 - 1 kByte = 1.024 Bytes = 2^{10} Bytes ... NO 1.000 Bytes
 - Se le suele llamar “kilobyte” aunque en realidad es incorrecto y oficialmente es un “kibibyte”
 - Igualmente decimos “megabyte” para 1 MByte = 2^{20} bytes cuando es en realidad un “mebibyte”, “gibibyte” (2^{30}) y “tebibyte” (2^{40})
- Cuando hablamos de velocidades de transmisión de datos
 - Usamos los prefijos del sistema internacional
 - 1 kbps = 10^3 = 1.000 bps, 1 Mbps = 10^6 = 1.000.000 bps, 1 Gbps = 10^9 bps



Prefix	Symbol for Prefix	Scientific Notation
exa	E	10^{18}
peta	P	10^{15}
tera	T	10^{12}
giga	G	10^9
mega	M	10^6
kilo	k	10^3
hecto	h	10^2
deka	da	10^1
---	--	10^0
deci	d	10^{-1}
centi	c	10^{-2}
milli	m	10^{-3}
micro	μ	10^{-6}
nano	n	10^{-9}
pico	p	10^{-12}
femto	f	10^{-15}
atto	a	10^{-18}

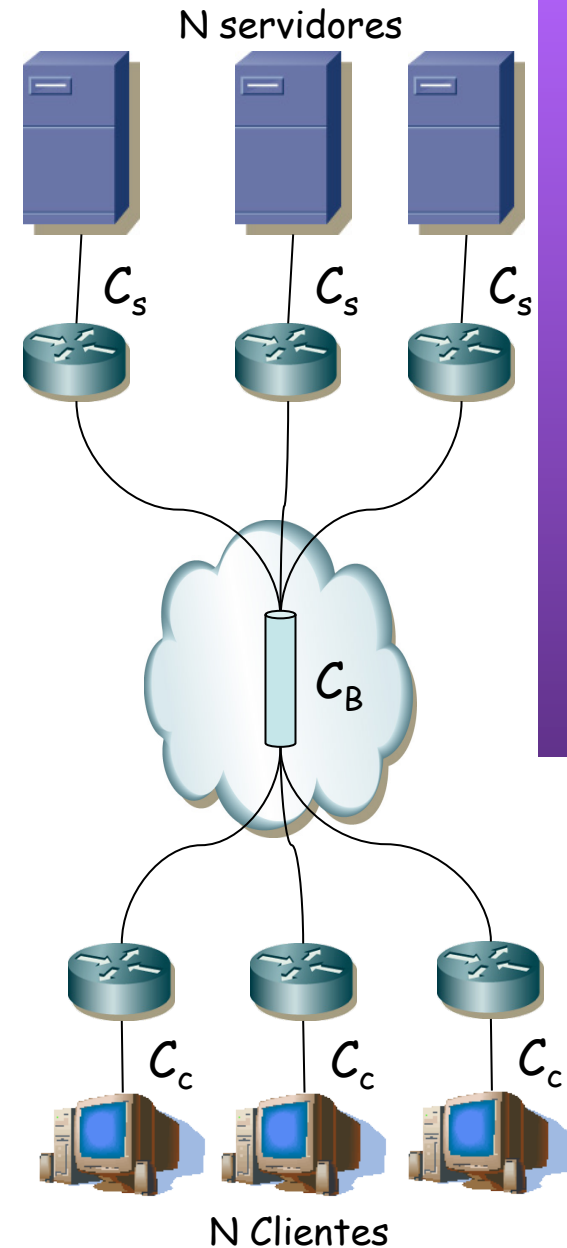
Ejemplo

- Cliente, servidor y un conjunto de conmutadores de paquetes intermedios
- Capacidades de los enlaces C_S , C_C y C_i $i=1..n$
- Cliente se descarga un fichero de tamaño F del servidor
- No hay más tráfico
- Máximo throughput: $\min\{C_S, C_C, C_i\}_{i=1..n}$
- Hoy en día los enlaces en el núcleo de la red son de mucha mayor capacidad que los del acceso
- Con eso de nuevo máximo throughput: $\min\{C_S, C_C\}$



Ejemplo 2

- N Clientes y servidores
- Cada cliente descarga un fichero de un servidor
- Comparten un enlace en la red de capacidad C_B
- La capacidad en el resto de enlaces en el núcleo de la red es superior
- No hay más tráfico
- ¿Dónde está ahora el cuello de botella?
- Supongamos $C_C < C_S < C_B$
- Supongamos que la capacidad C_B se reparte equitativamente entre los N flujos
- Máximo throughput: $\min\{C_C, C_B/N\}$
- El cuello de botella puede estar en el acceso o en el núcleo



Contenido

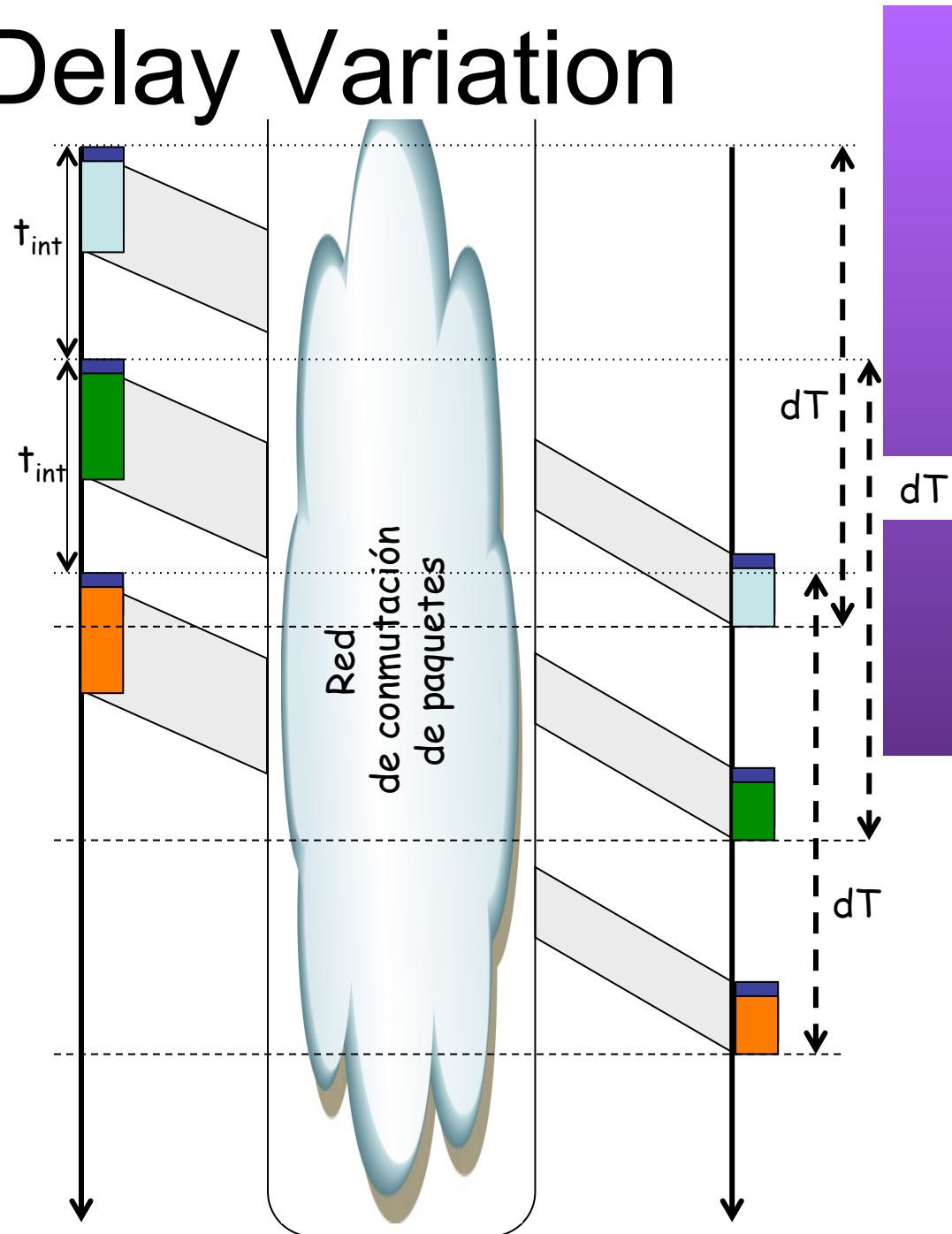
- Retardos
 - Retardo de procesado
 - Retardo en cola
- Efectos del tamaño del paquete
- Throughput
- Packet Delay Variation
- Pérdidas
- Problemas de circuitos y paquetes

Packet Delay Variation

- Variación en el retardo (*jitter*)

Ejemplo 1

- Paquetes equiespaciados
- Retardo medido entre el tiempo de inicio de envío de primer bit y tiempo de fin de recepción del último bit (dT)
- Todos sufren igual retardo hasta el punto de medida
- (...)

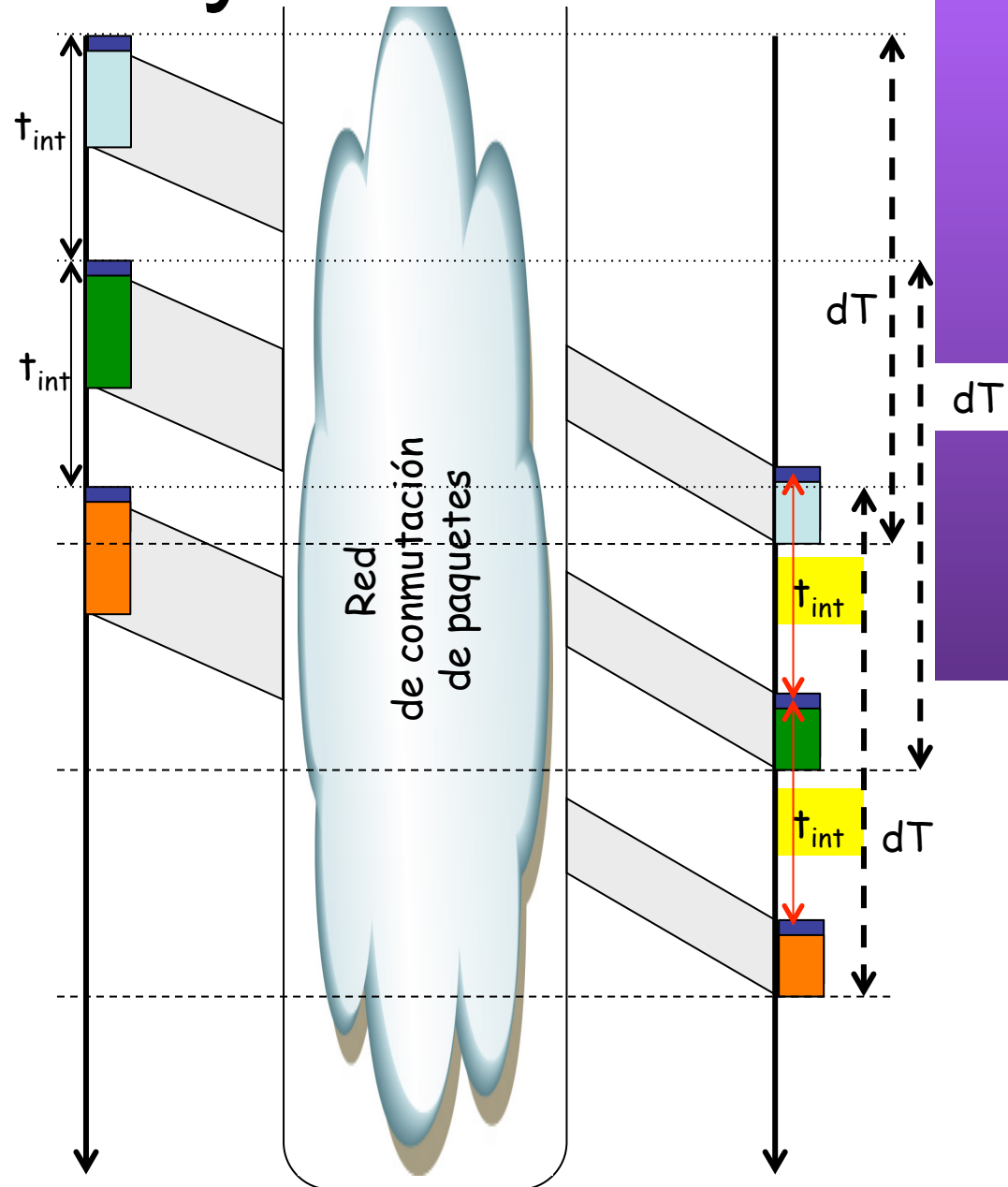


Packet Delay Variation

- Variación en el retardo (*jitter*)

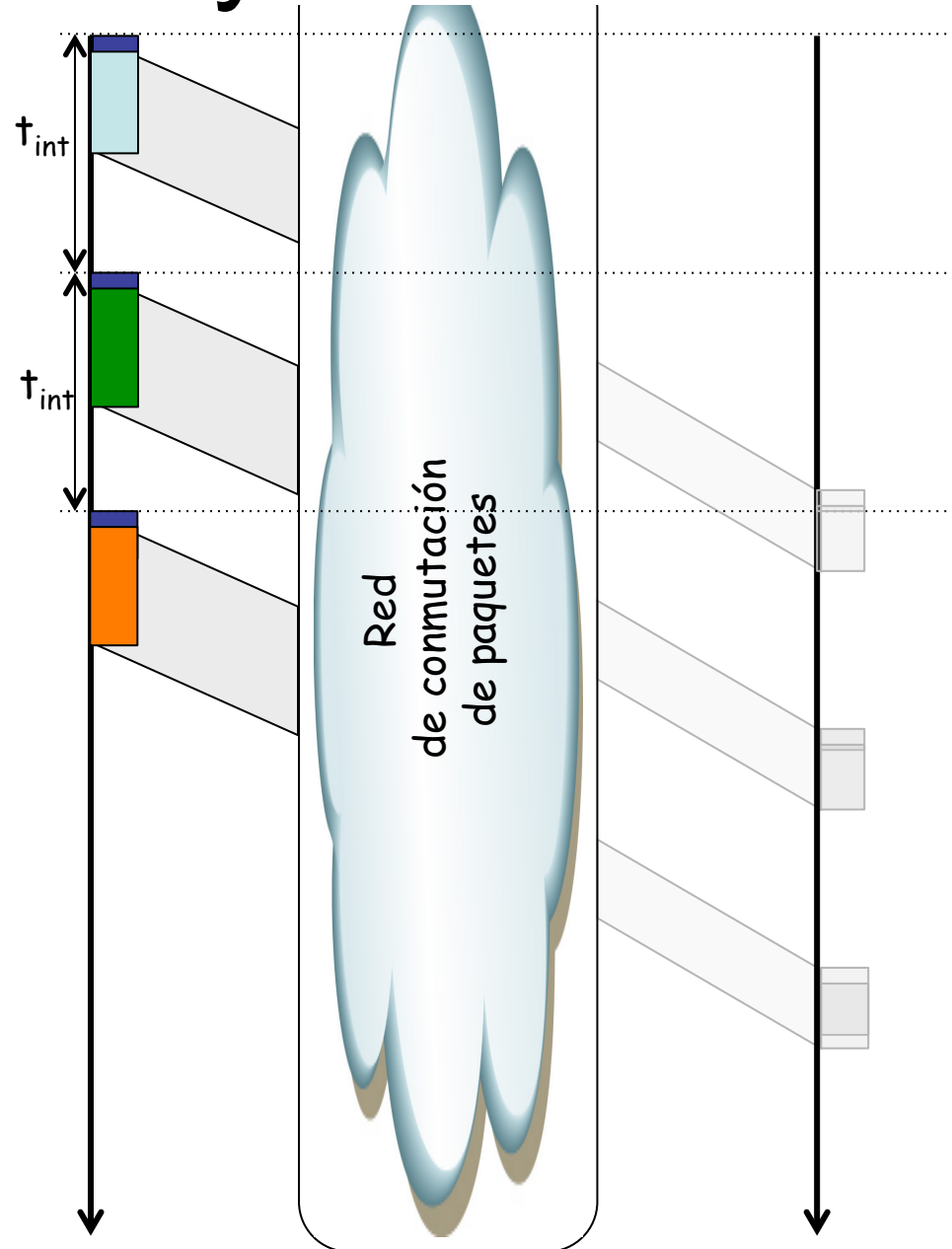
Ejemplo 1

- Paquetes equiespaciados
- Retardo medido entre el tiempo de inicio de envío de primer bit y tiempo de fin de recepción del último bit (dT)
- Todos sufren igual retardo hasta el punto de medida
- En ese otro extremo (o punto de medida) los paquetes están equiespaciados
- No hay variación en el retardo



Packet Delay Variation

- Variación en el retardo (*jitter*)
- ## Ejemplo 2
- Paquetes equiespaciados
 - (En gris los instantes del ejemplo anterior)
 - Sufren diferente retardo (dT_1 , dT_2 y dT_3) (...)

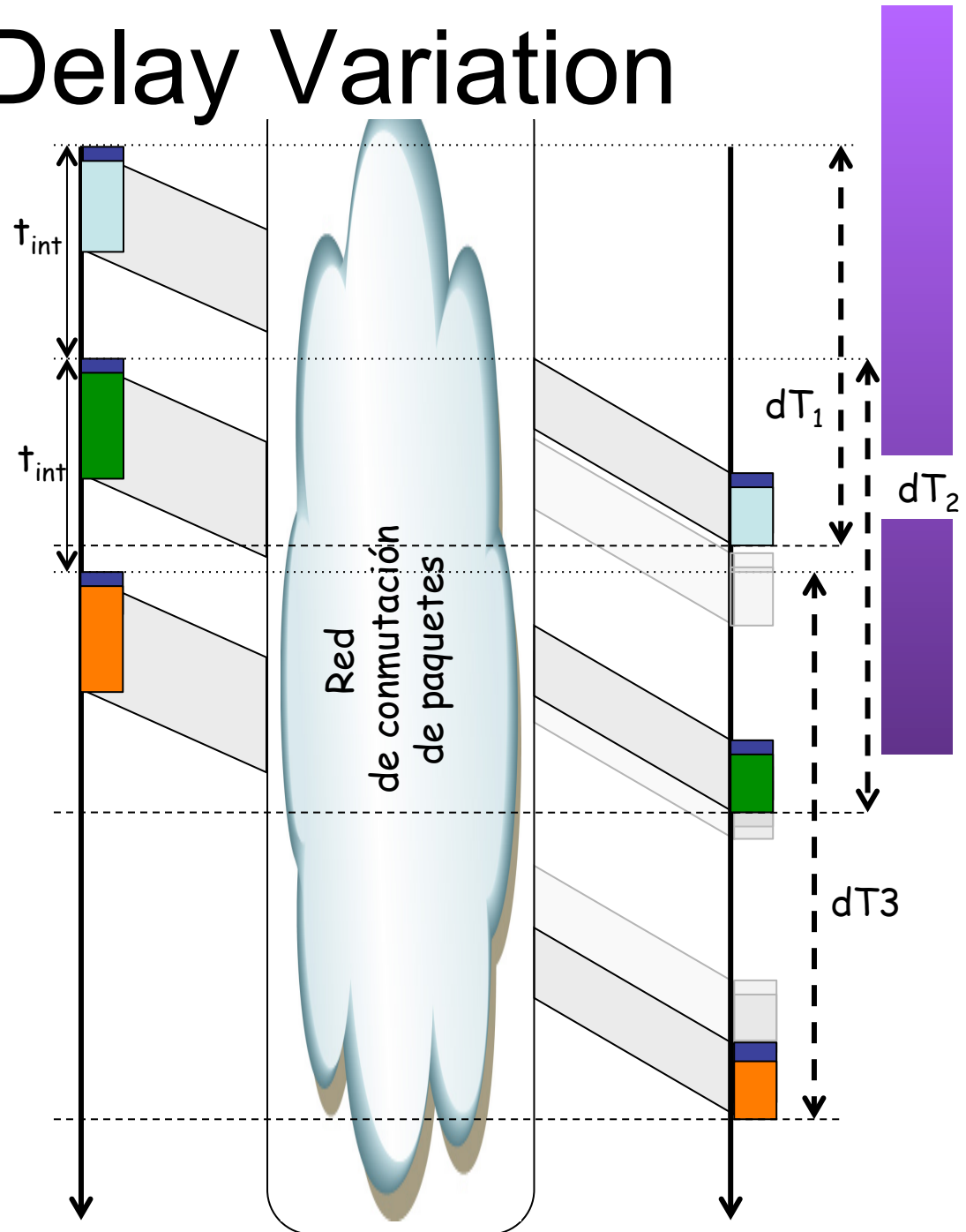


Packet Delay Variation

- Variación en el retardo (*jitter*)

Ejemplo 2

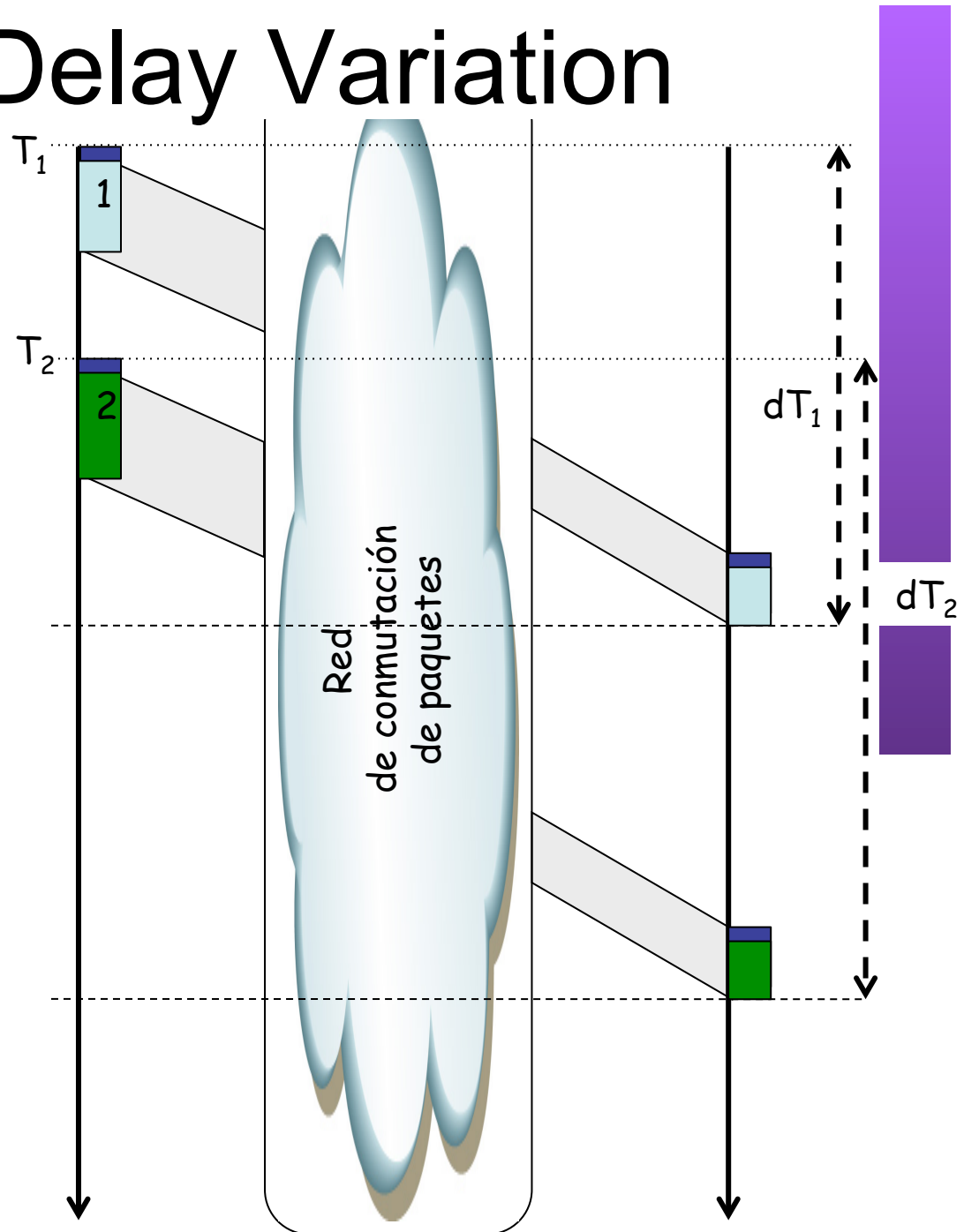
- Paquetes equiespaciados
- (En gris los instantes del ejemplo anterior)
- Sufren diferente retardo (dT_1 , dT_2 y dT_3)
- PDV mide la variación en el retardo



Packet Delay Variation

Cálculo

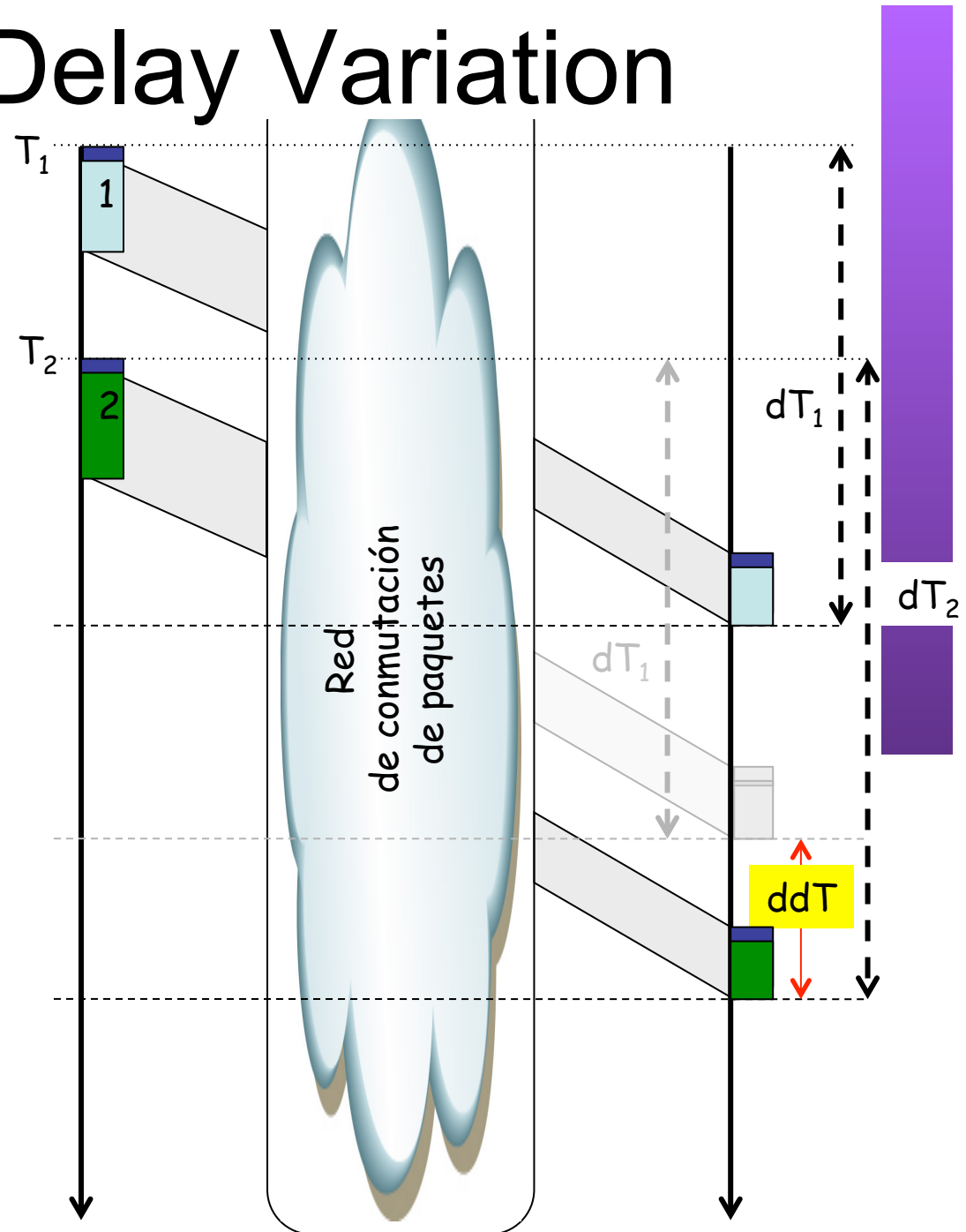
- Dos paquetes (1) y (2)
- Retardos dT_1 y dT_2
- $ddT = dT_2 - dT_1$
- Mide la diferencia entre cuándo ha llegado el segundo paquete y cuándo “debería” haber llegado
- El “debería” sería en el caso de mismo retardo ambos (...)



Packet Delay Variation

Cálculo

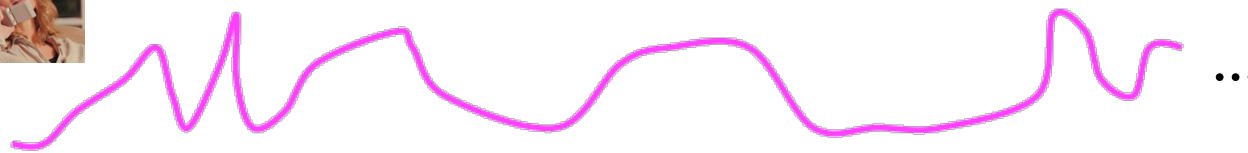
- Dos paquetes (1) y (2)
- Retardos dT_1 y dT_2
- $ddT = dT_2 - dT_1$
- Mide la diferencia entre cuándo ha llegado el segundo paquete y cuándo “debería” haber llegado
- El “debería” sería en el caso de mismo retardo ambos (paquete en gris)
- Diferencia puede ser positiva o negativa (atrasarse o adelantarse)



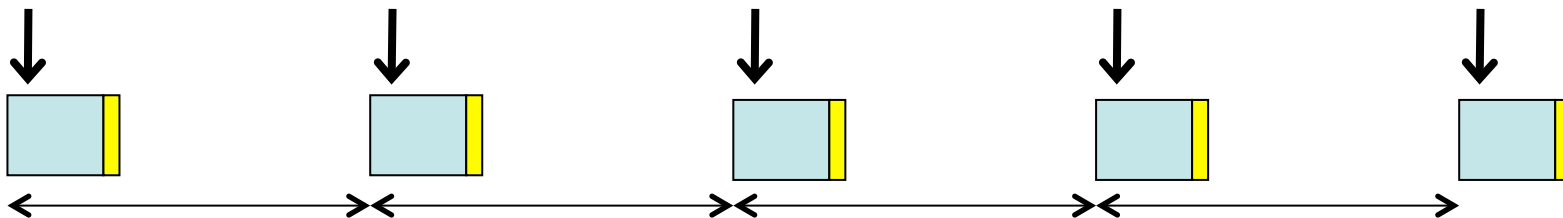
Efectos del PDV

Ejemplo

- Codec de voz que genera información digital a tasa constante
- Una vez paquetizada se convierte en paquetes equiespaciados
- (...)



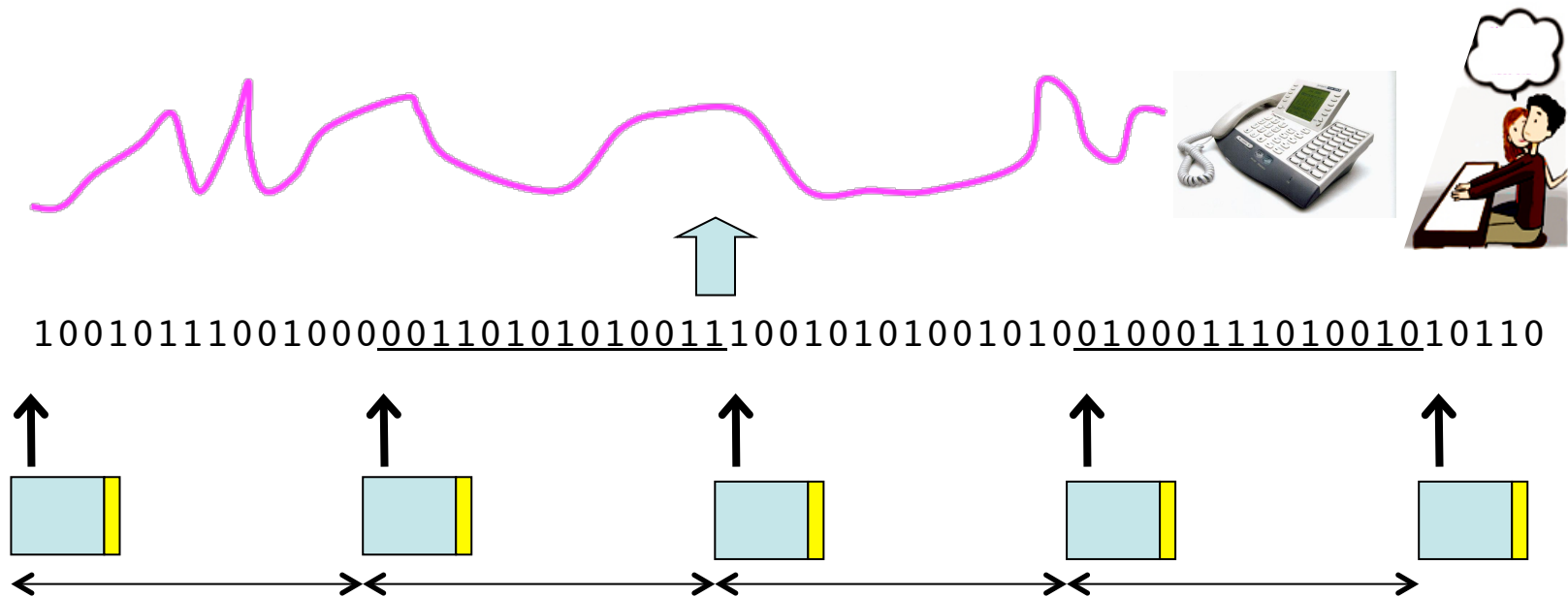
1001011100100000110101010011100101010010100100011101001010110



Efectos del PDV

Ejemplo

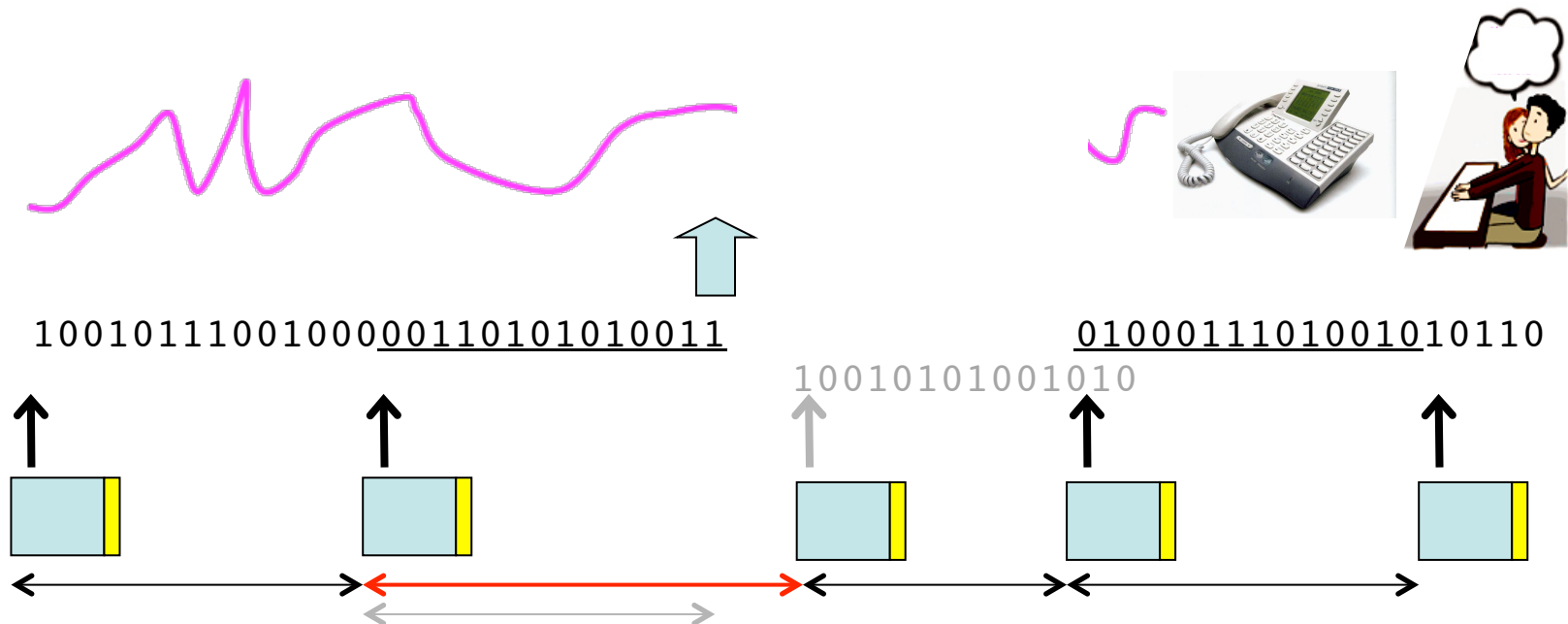
- Codec de voz que genera información digital a tasa constante
- Una vez paquetizada se convierte en paquetes equiespaciados
- En la decodificación se consumen a esa misma tasa
- (...)



Efectos del PDV

Ejemplo

- Codec de voz que genera información digital a tasa constante
- Una vez paquetizada se convierte en paquetes equiespaciados
- En la decodificación se consumen a esa misma tasa
- Un primer paquete sufre un retardo mayor que el anterior y puede que cuando llegue “ya sea tarde”
- Es decir, ya no sirve decodificarlo pues ya se ha producido el corte en la reproducción



Efectos del PDV

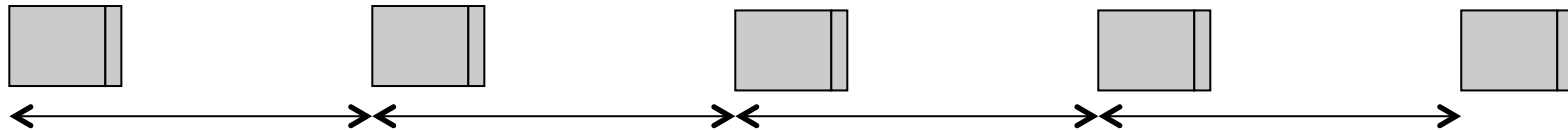
Solución

- Retrasar comienzo de la reproducción mediante *buffering* en el cliente
- Supongamos que en $t=0$ tiene el primer paquete y podría empezar a reproducir
- (...)

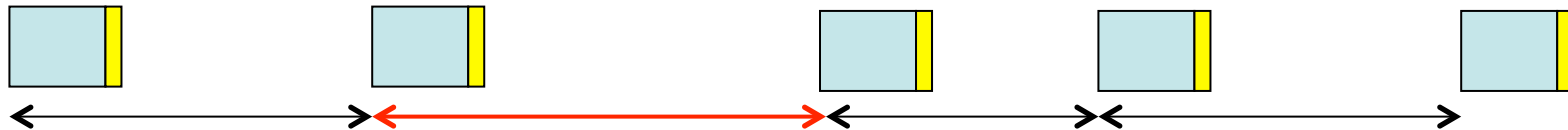
$t=0$
↓

1001011100100000110101010011100101010010100100011101001010110

ideal



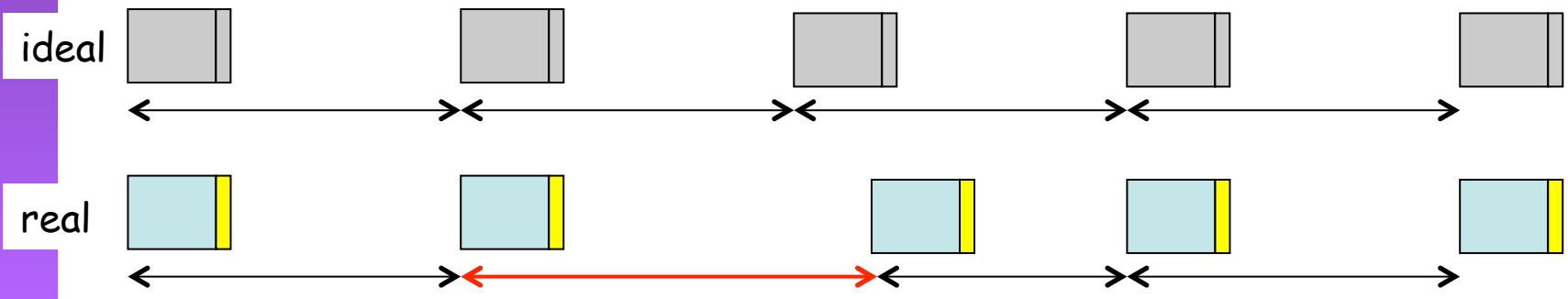
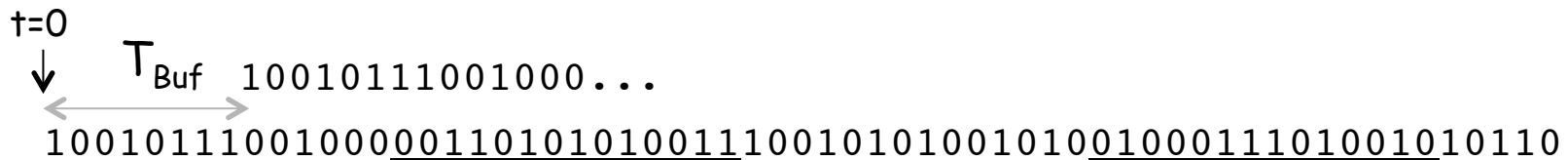
real



Efectos del PDV

Solución

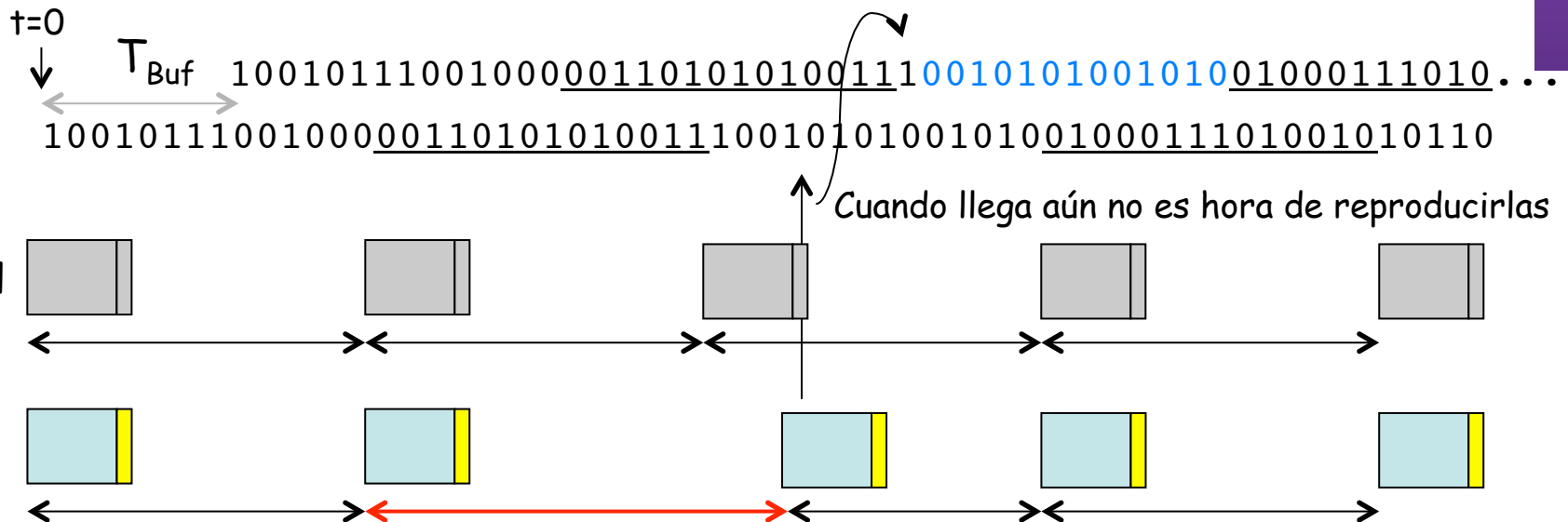
- Retrasar comienzo de la reproducción mediante buffering en el cliente
- Supongamos que en $t=0$ tiene el primer paquete y podría empezar a reproducir
- Se introduce en memoria durante T_{Buf} (mientras tanto pueden llegar más paquetes, según el tiempo que se desee y lo grande que sea T_{Buf})
- (...)



Efectos del PDV

Solución

- Retrasar comienzo de la reproducción mediante buffering en el cliente
- Supongamos que en $t=0$ tiene el primer paquete y podría empezar a reproducir
- Se introduce en memoria durante T_{Buf} (mientras tanto pueden llegar más paquetes, según lo grande que sea T_{Buf})
- El paquete muy retrasado entrará en el buffer y aún se estarán reproduciendo muestras de anteriores si su PDV es menor que T_{Buf}



Contenido

- Retardos
 - Retardo de procesado
 - Retardo en cola
- Efectos del tamaño del paquete
- Throughput
- Packet Delay Variation
- Pérdidas
- Problemas de circuitos y paquetes

Pérdidas

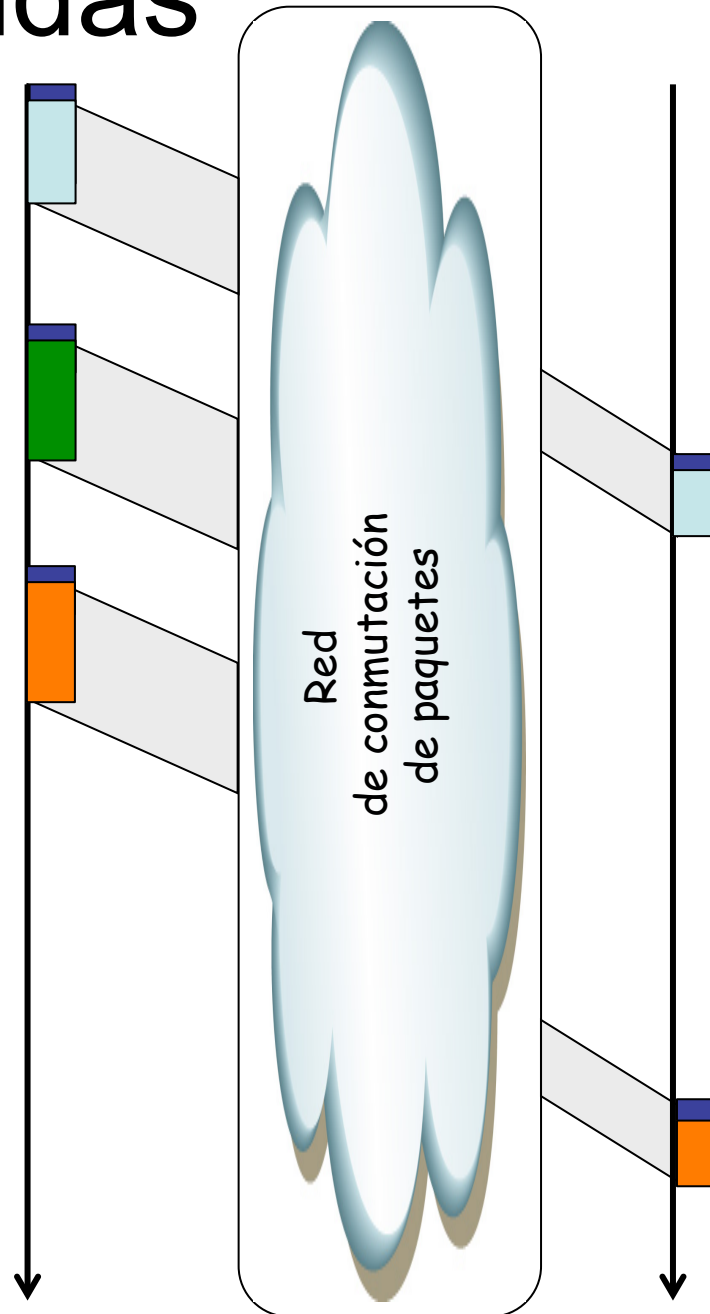
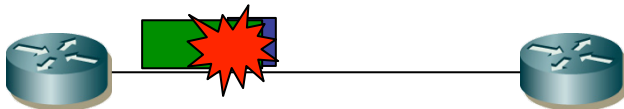
- Los paquetes podrían no llegar nunca a su destino

Posibles motivos

- Se corrompió y fue descartado en algún nodo de la red (CRCs)
- BER = Bit Error Rate
- Aproxima a la probabilidad de error de bit p_{err}
- Probabilidad de algún error en un paquete de N bits:

$$p_{epk} = 1 - (1 - p_{err})^N$$

- Asumiendo errores indep. (no ráfagas)
- Sin código “corrector” de errores
- Ejemplo: $p_{err} = 10^{-6}$, $N = 12.000 \rightarrow \rightarrow$
 $p_{epk} \approx 10^{-2}$
- (...)

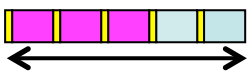
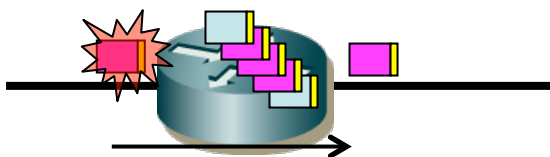


Pérdidas

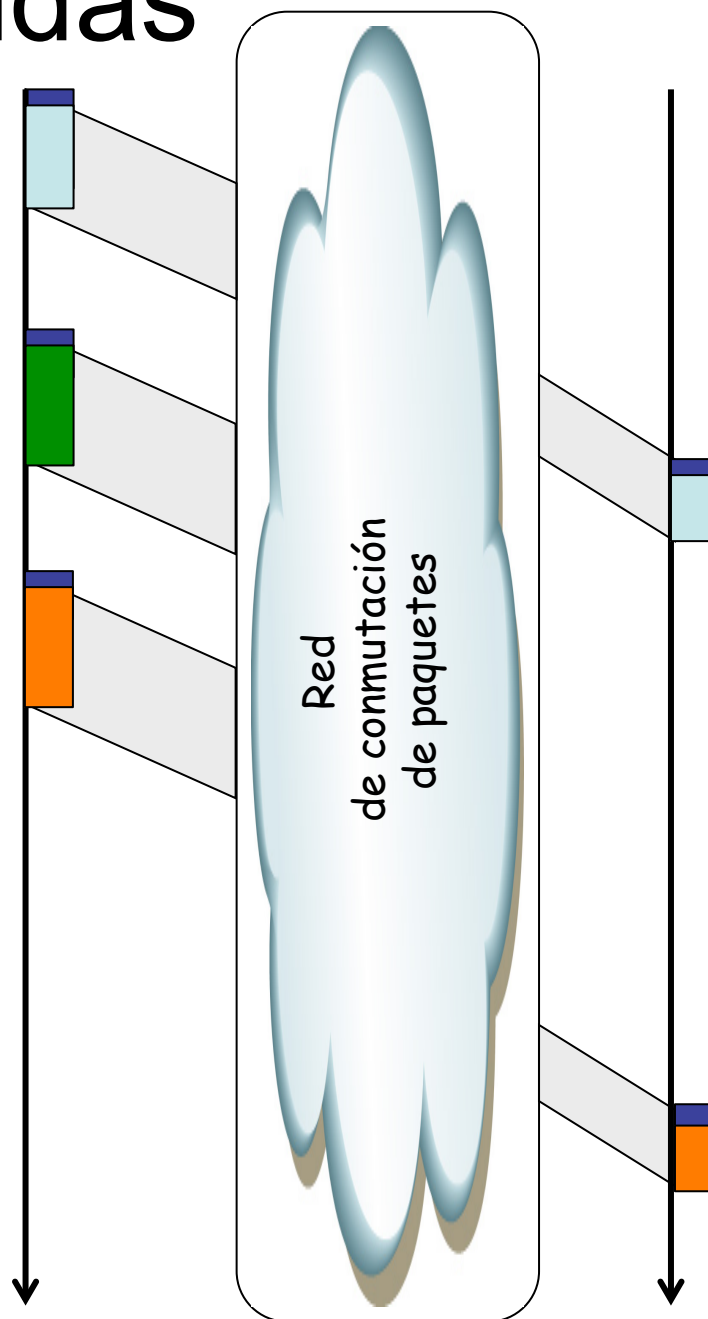
- Los paquetes podrían no llegar nunca a su destino

Posibles motivos

- Se descartó en un nodo de la red por desbordamiento de buffer
- (...)



Tamaño del buffer

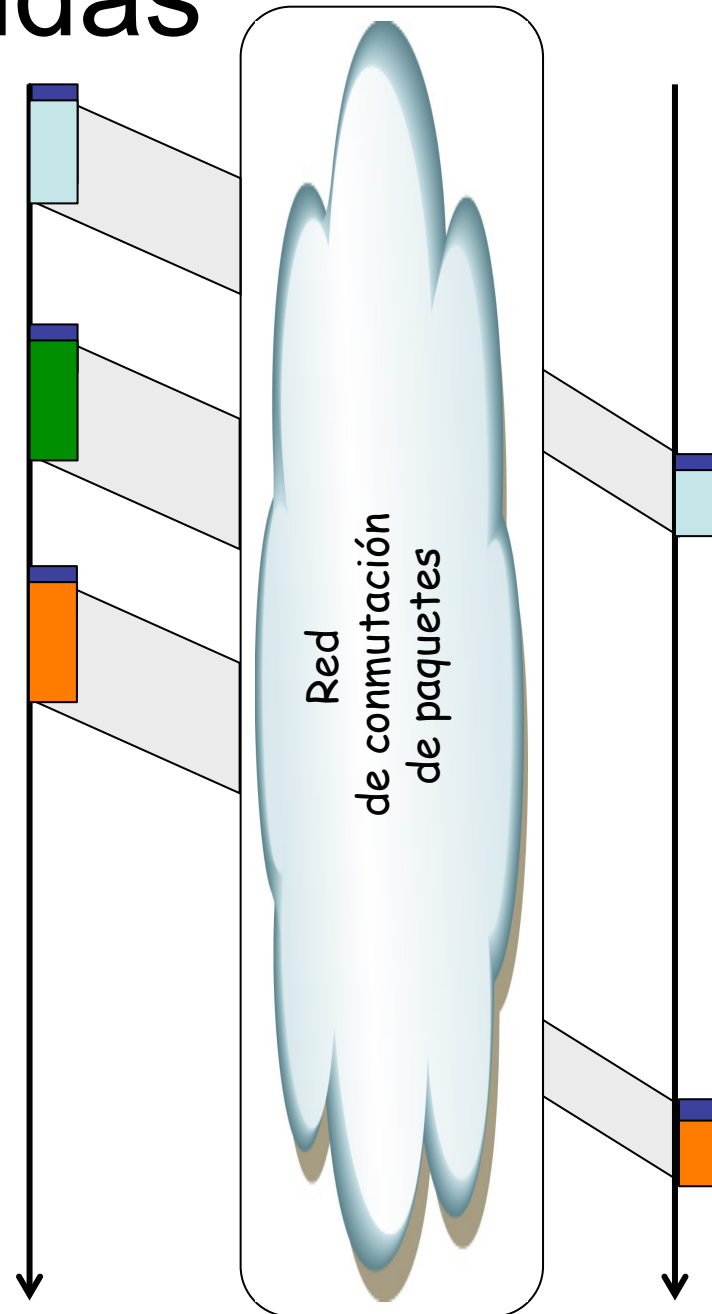
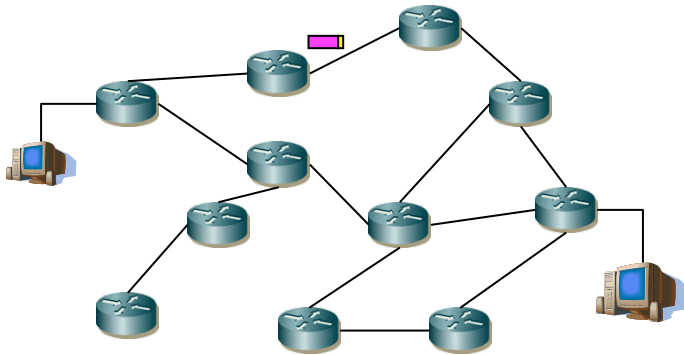


Pérdidas

- Los paquetes podrían no llegar nunca a su destino

Posibles motivos

- Se descartó por exceder el tiempo en la red
- (...)

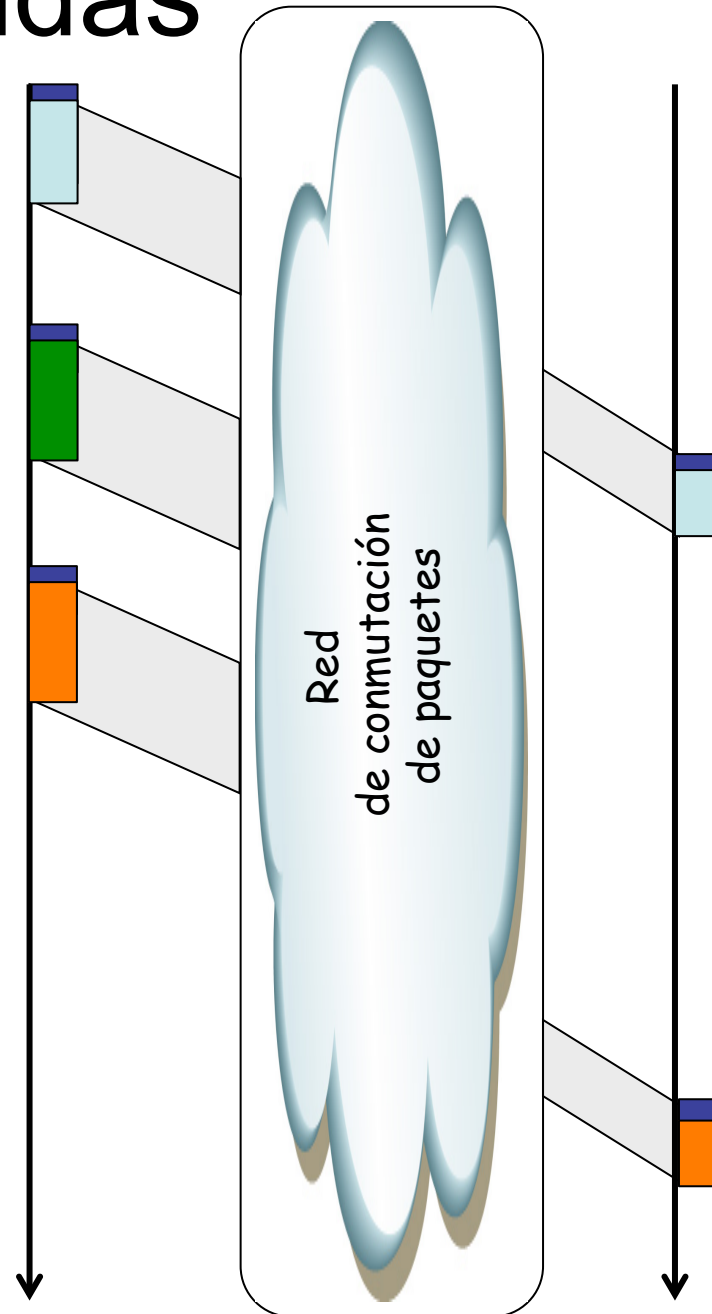
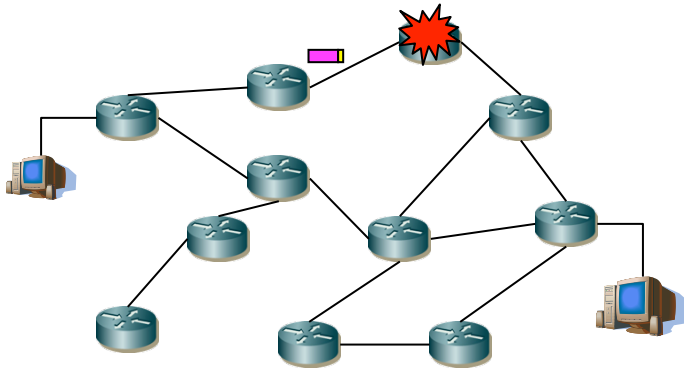


Pérdidas

- Los paquetes podrían no llegar nunca a su destino

Posibles motivos

- Fallo de un elemento de red
- Lleva un tiempo recalcular caminos
- (...)

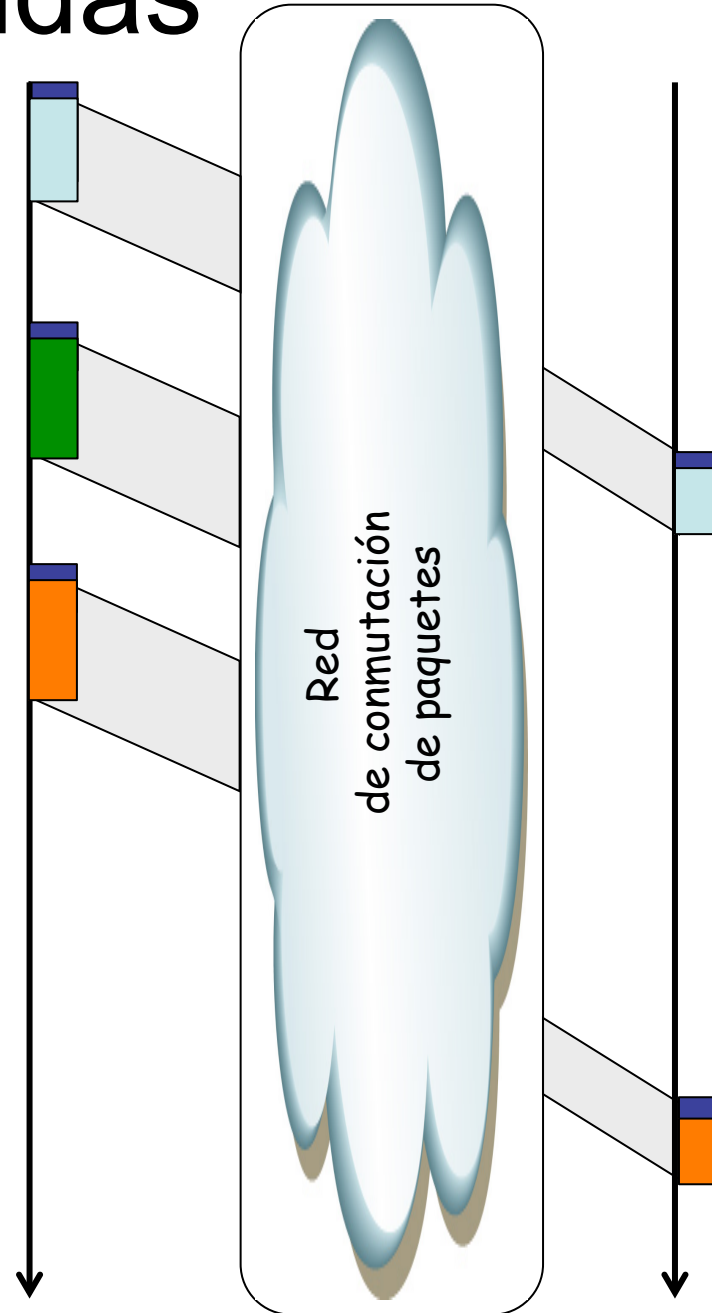


Pérdidas

- Los paquetes podrían no llegar nunca a su destino

Posibles motivos

- Descarte en nodo extremo por desbordamiento de buffer
- Puede ser culpa de la propia aplicación y del tiempo que le lleva procesar los datos recibidos



Problemas

Problemas de redes de circuitos

- **Encaminamiento**
 - Cuando se pide a la red establecer una llamada
 - A partir de la dirección de destino decidir por dónde reservar enlaces desde el origen al destino.
- **Bloqueo**
 - Si en algún punto la llamada necesita recursos no disponibles: no se establecerá y el usuario no recibe servicio.
 - Diseñar las redes de circuitos para que el bloqueo no se produzca o tenga una probabilidad baja

Problemas de redes de paquetes

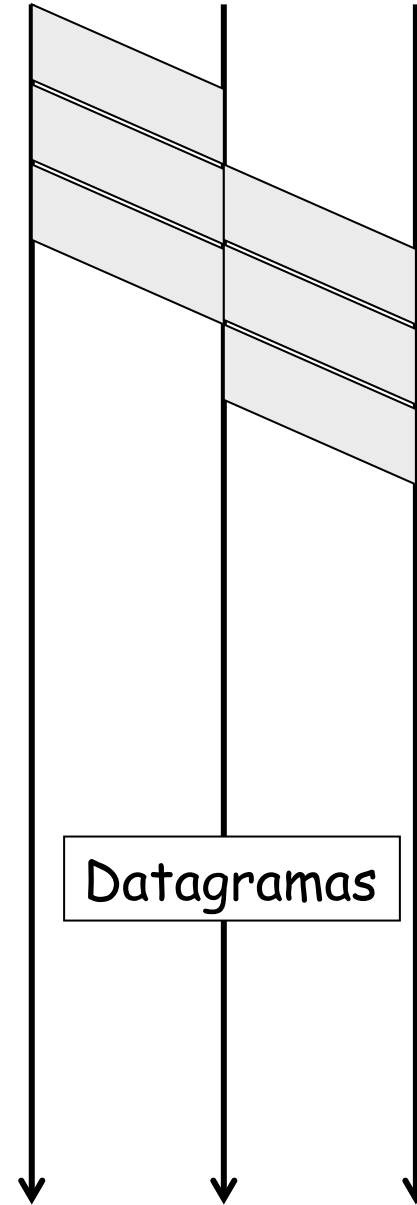
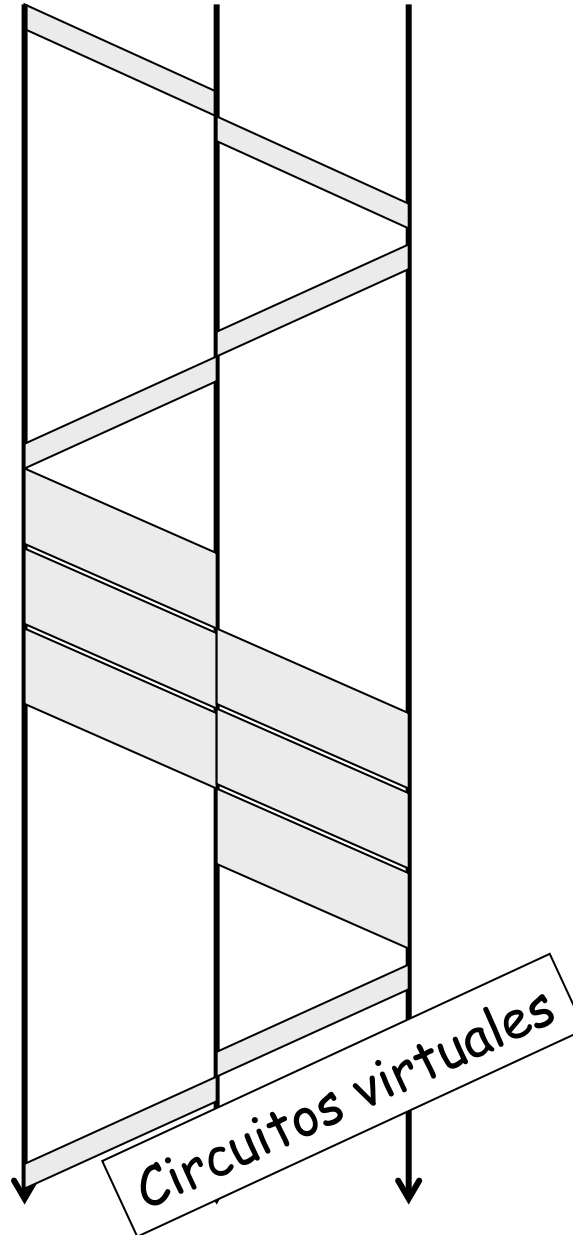
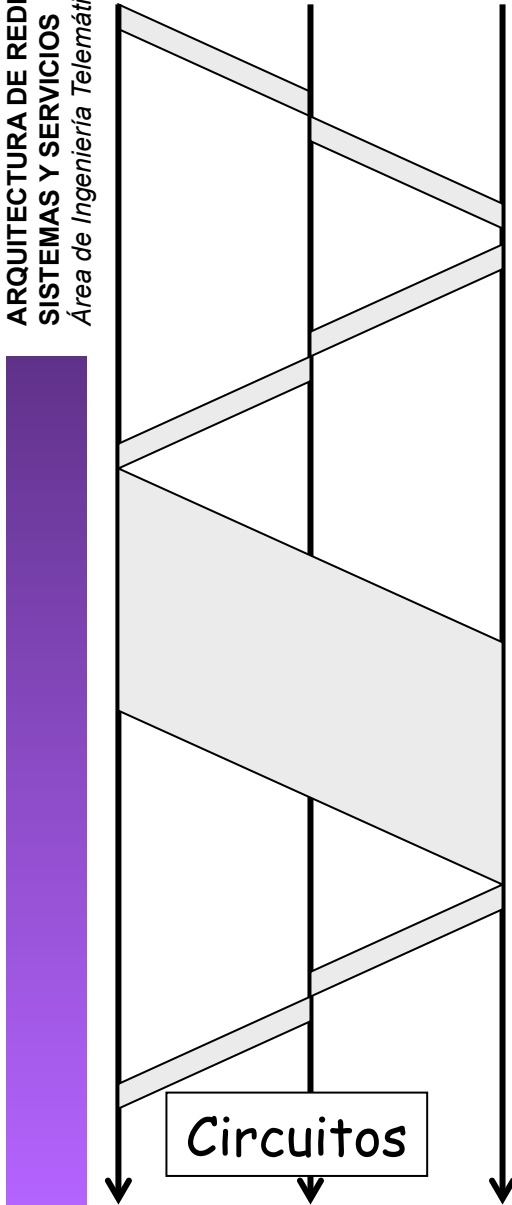
- **Encaminamiento**
 - Por cada paquete que debe reenviar un nodo debe decidir por qué camino reenviarlo (a qué vecino entregárselo)
- **Bloqueo:** No hay, la red acepta todos los paquetes.

Nuevos problemas:

- **Transporte fiable**
 - ¿Qué pasa si un paquete no se entrega?
- **Control de flujo**
 - ¿Qué pasa si llega un paquete a un destino que está muy ocupado para aceptarlo?
- **Congestión**
 - ¿Qué pasa si la red está aceptando demasiados paquetes y el retardo de entrega crece demasiado?

Tiempos

ARQUITECTURA DE REDES,
SISTEMAS Y SERVICIOS
Área de Ingeniería Telemática



Resumen

- En redes de conmutación de paquetes:
 - Retardo, pérdidas y variación del retardo
 - El retardo múltiples componente
- En redes de conmutación de circuitos:
 - Encaminamiento y bloqueo
- En redes de conmutación de datagramas problemas adicionales de control de flujo, congestión, etc.