Scheduling: WFQ

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

Grado en Ingeniería en Tecnologías de Telecomunicación, 3º

Weighted Fair Queueing

WFQ

- *Weighted Fair Queueing*
- Aproximación de GPS (*Generalized Processor Sharing*) para el caso de paquetes
- Equivalente a PGPS (*Packet-by-packet Generalized Processor Sharing)*
- No requiere conocer el tamaño medio de paquete
- Emplea un reloj virtual
- Calcula el final virtual en que se enviaría cada paquete en el caso ideal GPS
- Se envían en orden de tiempo final virtual
- Más complejo de implementar
- Puede ofrecer *worst-case bounds*

WFQ

- Se simulan “turnos” (*rounds*)
- Supongamos que no hay pesos
- Supongamos que GPS no sirve fluido perfecto sino bit-a-bit
- En cada turno se envía 1 bit de cada flujo
- El número de turno (*round number*) es el número de turnos bit-a-bit que se han completado en un instante
- Así, cuantos más flujos activos simultáneos hay, más despacio se incrementa el turno con el tiempo
- Ignoramos el servir bit-a-bit si definimos el round number como un valor que crece inversamente proporcional al número de flujos activos

WFQ

- Un paquete **k** del flujo **i** que llega en el instante **t**
- Su *finish number F(i,k,t)* :
  - Si flujo está inactivo: el *round number* actual + el tamaño en bits
  - Si flujo está activo: máx[F(i,k-1,t), round_number] + tamaño en bits
- Una vez calculado para un paquete no hay que recalcularlo ante nuevas llegadas
- Si llega a una cola llena se descartan paquetes en orden decreciente de finish number hasta que quepa

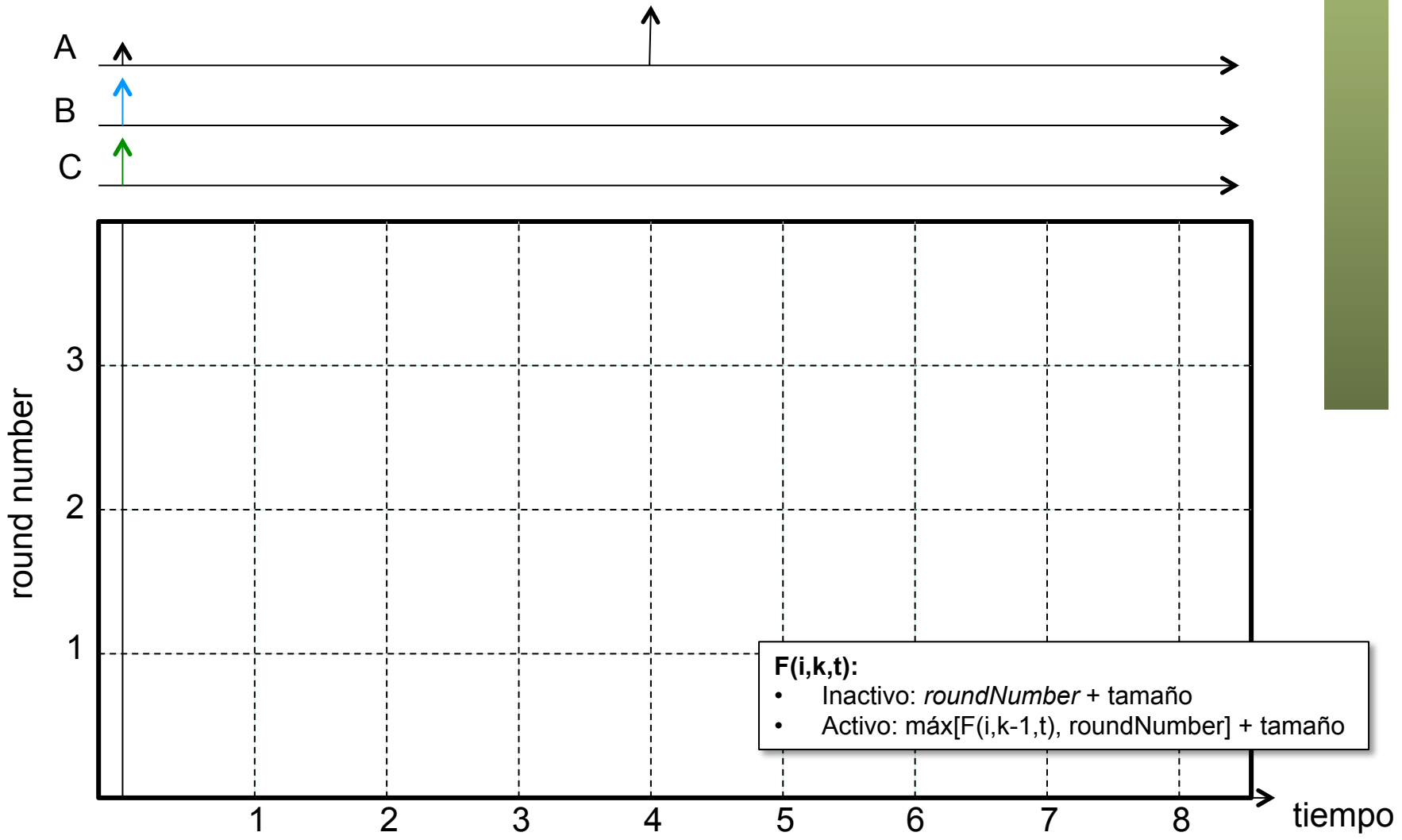
WFQ

- Calcular el round number es complejo
- Hay que hacerlo para cada paquete que llega y por cada uno que se envía
- En el caso con pesos a la hora de calcular el finish number:
  - Si flujo inactivo: el *round number* actual + tamaño / peso
  - Si flujo activo: máx[F(i,k-1,t), round_number] + tamaño / peso
- y el round number se incrementa con el inverso de la suma de los pesos
- Existen variantes para simplificar este cálculo:
  - Self-Clocked Fair Queuing (SCFQ)
  - Start-Time Fair Queuing

WFQ: Ejemplo

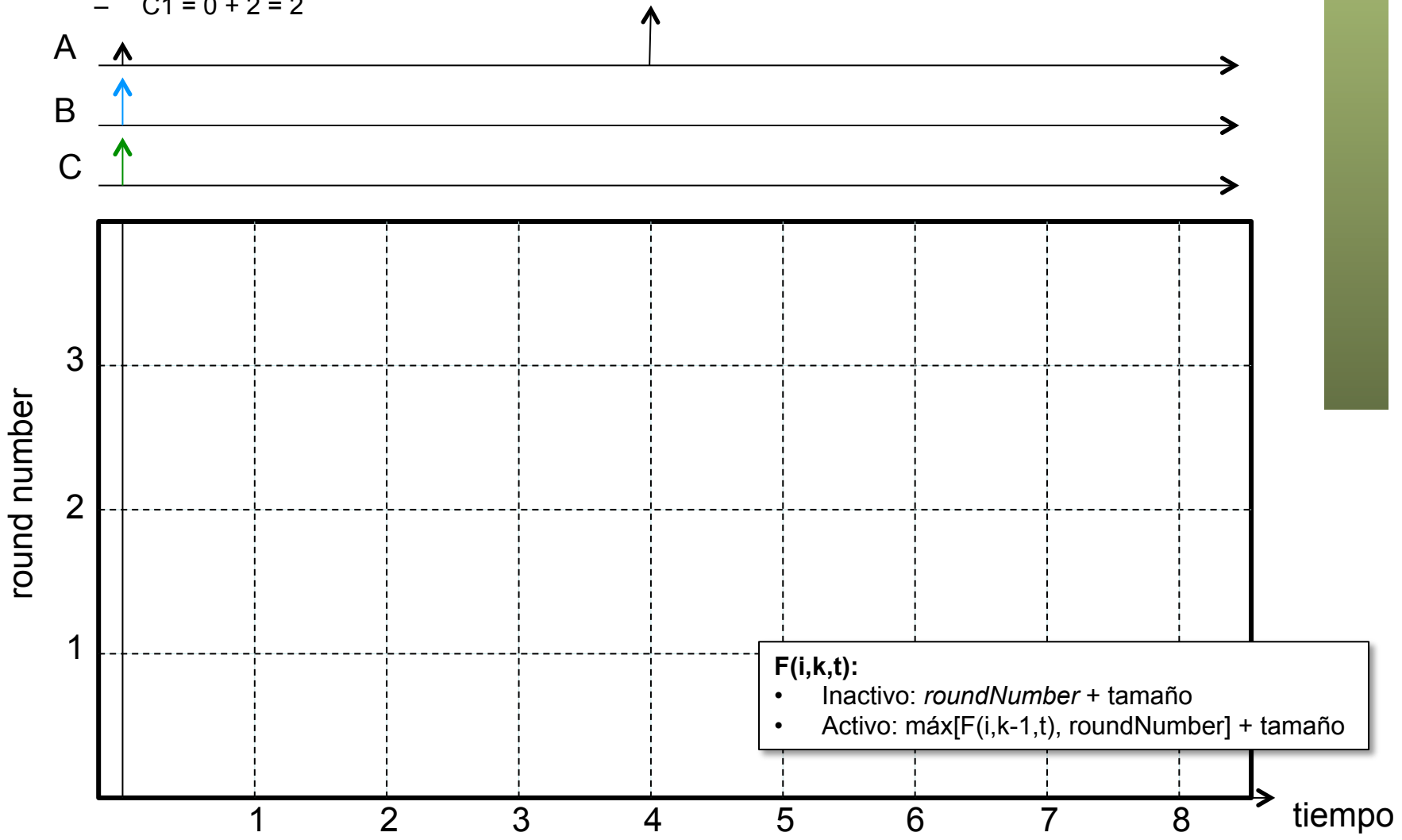
WFQ (Ejemplo)

- Enlace a 1 unidad/s
- Llegadas de tamaños 1, 2 y 2 unidades en t=0 y de tamaño 2 unidades en t=4



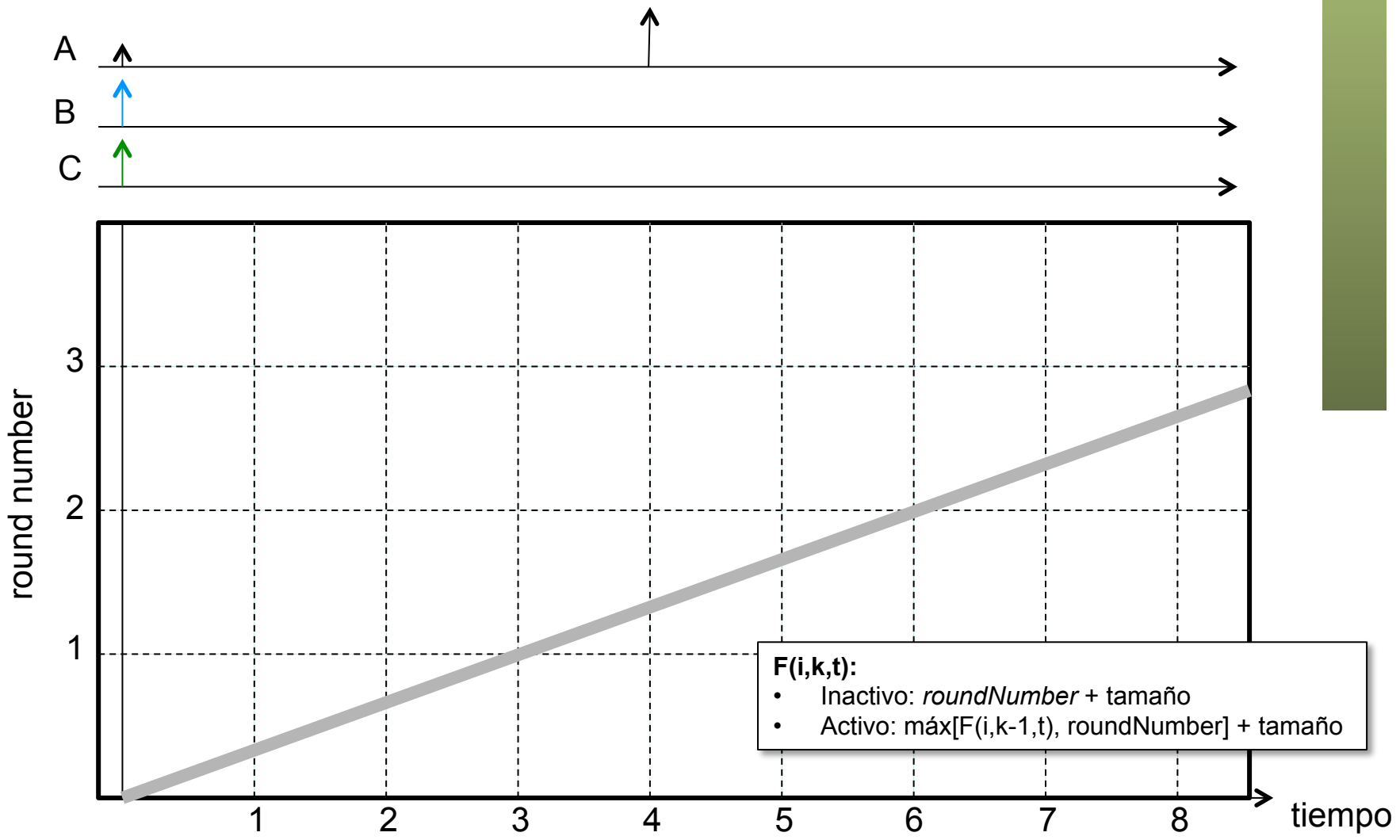
WFQ (Ejemplo)

- Finish numbers:
  - A1 = 0 + 1 = 1
  - B1 = 0 + 2 = 2
  - C1 = 0 + 2 = 2



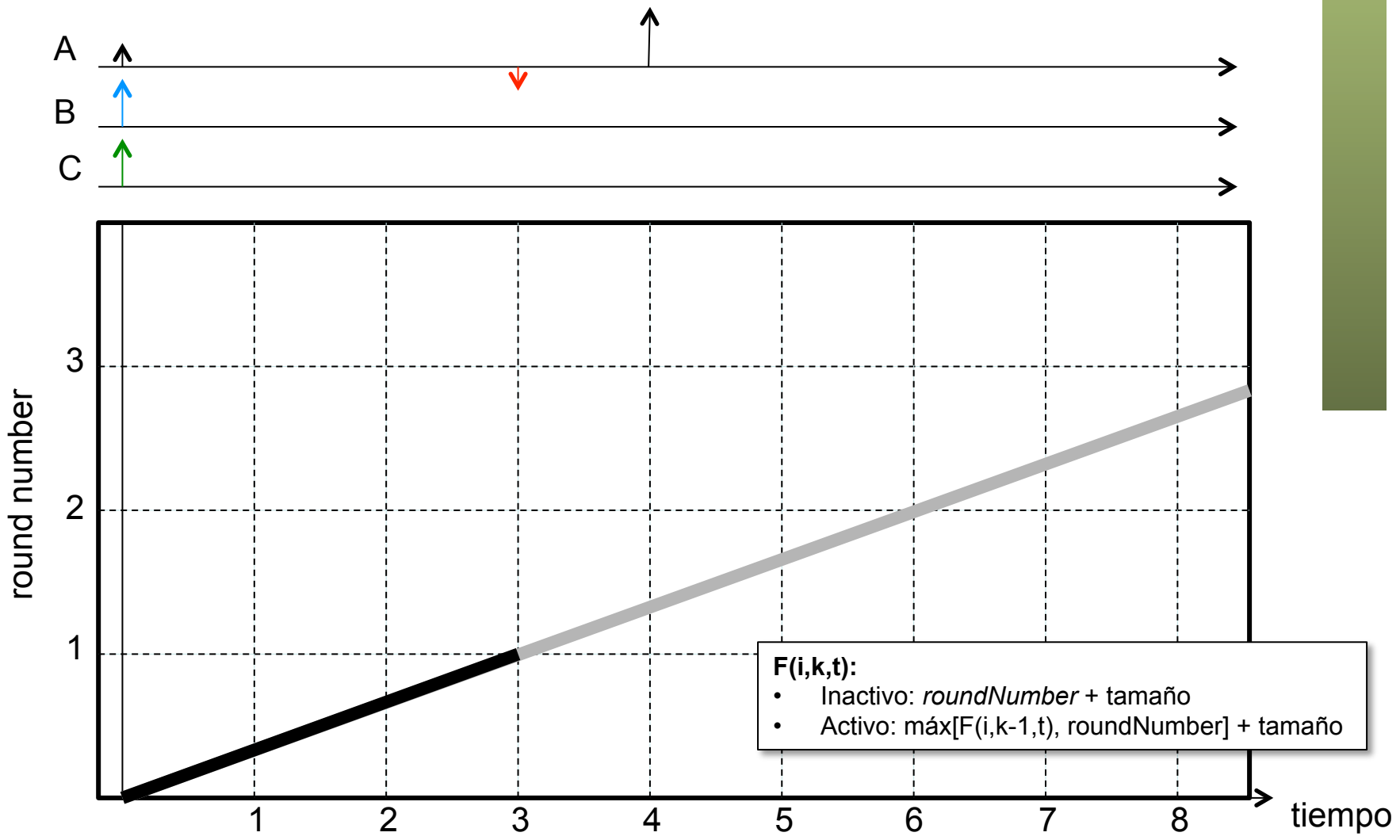
WFQ (Ejemplo)

- Hay 3 flujos a enviar simultáneamente
- El round number se incrementa a C/3 = 1/3 por unidad de tiempo



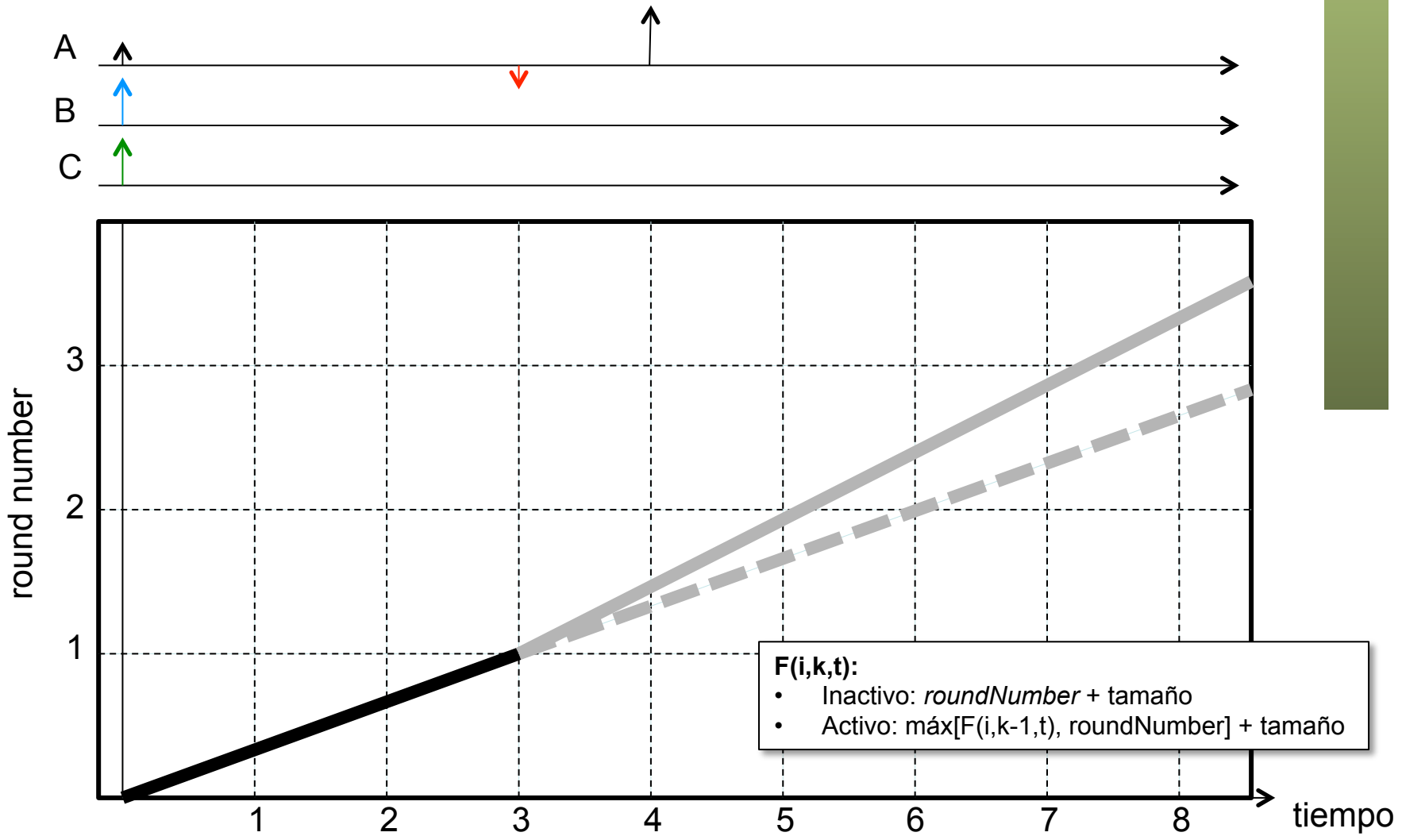
WFQ (Ejemplo)

- En el instante t=3 se han servido 3 bits, eso es uno por flujo y por lo tanto termina el round 1 y termina de enviarse A1



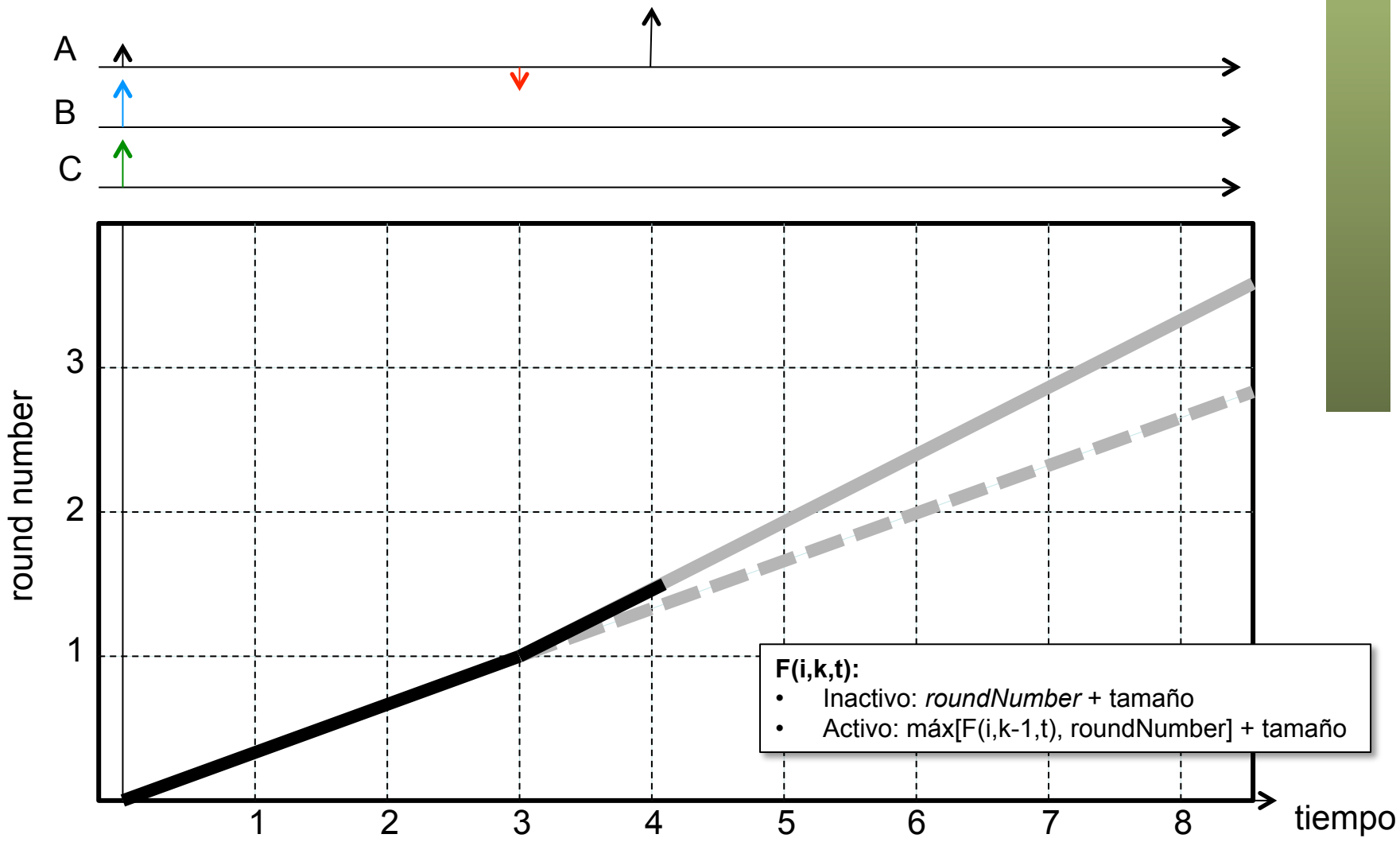
WFQ (Ejemplo)

- A partir de ahí se siguen sirviendo B1 y C1 con finish number = 2
- Al haber dos flujos activos crece el round number a 1/2



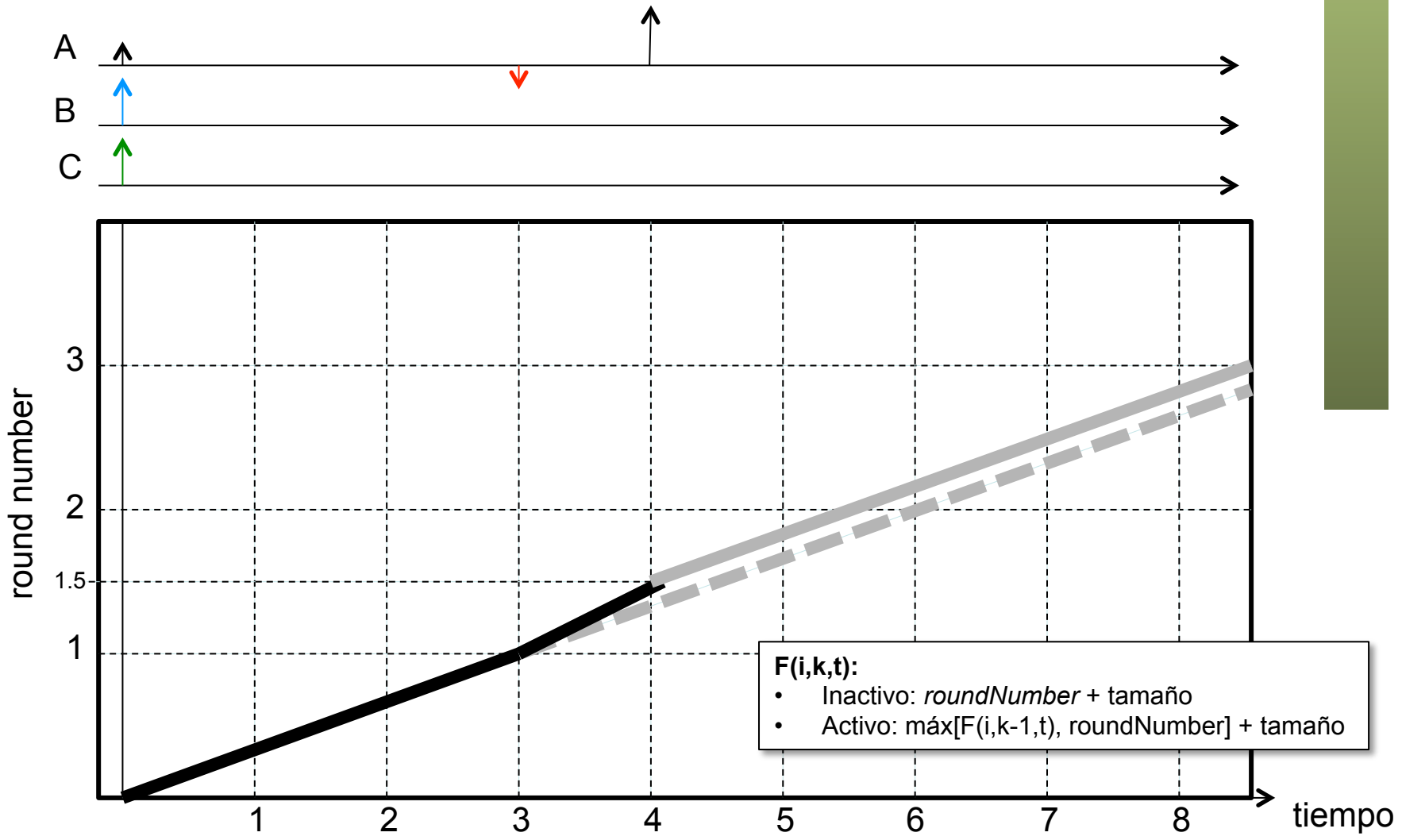
WFQ (Ejemplo)

- B1 y B2 terminarían de enviarse al alcanzar round number = 2 (t = 5) pero llega antes A2



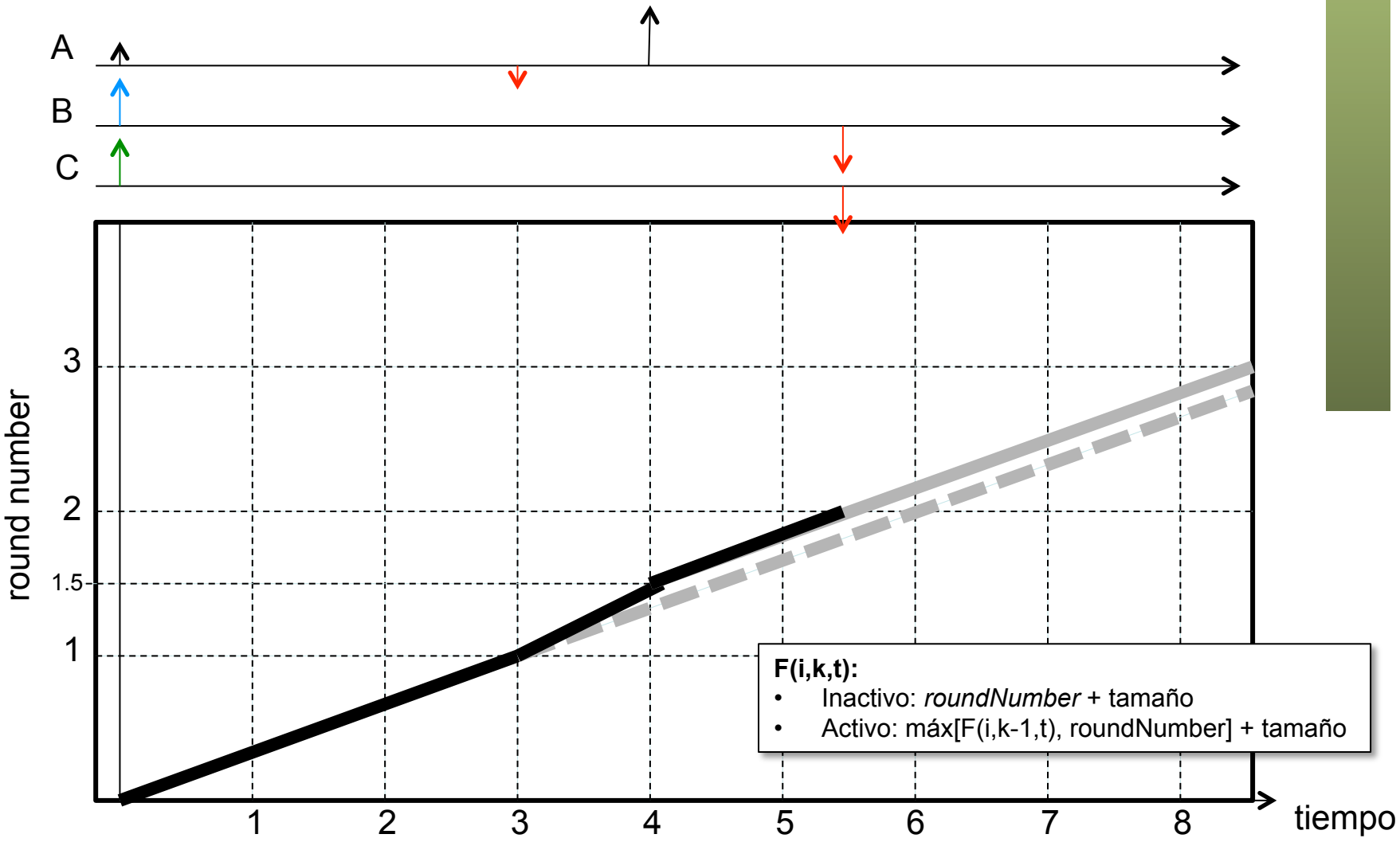
WFQ (Ejemplo)

- Finish number de A2 es 1.5 + 2 = 3.5
- A partir de t=4 vuelve a haber 3 flujos simultáneos



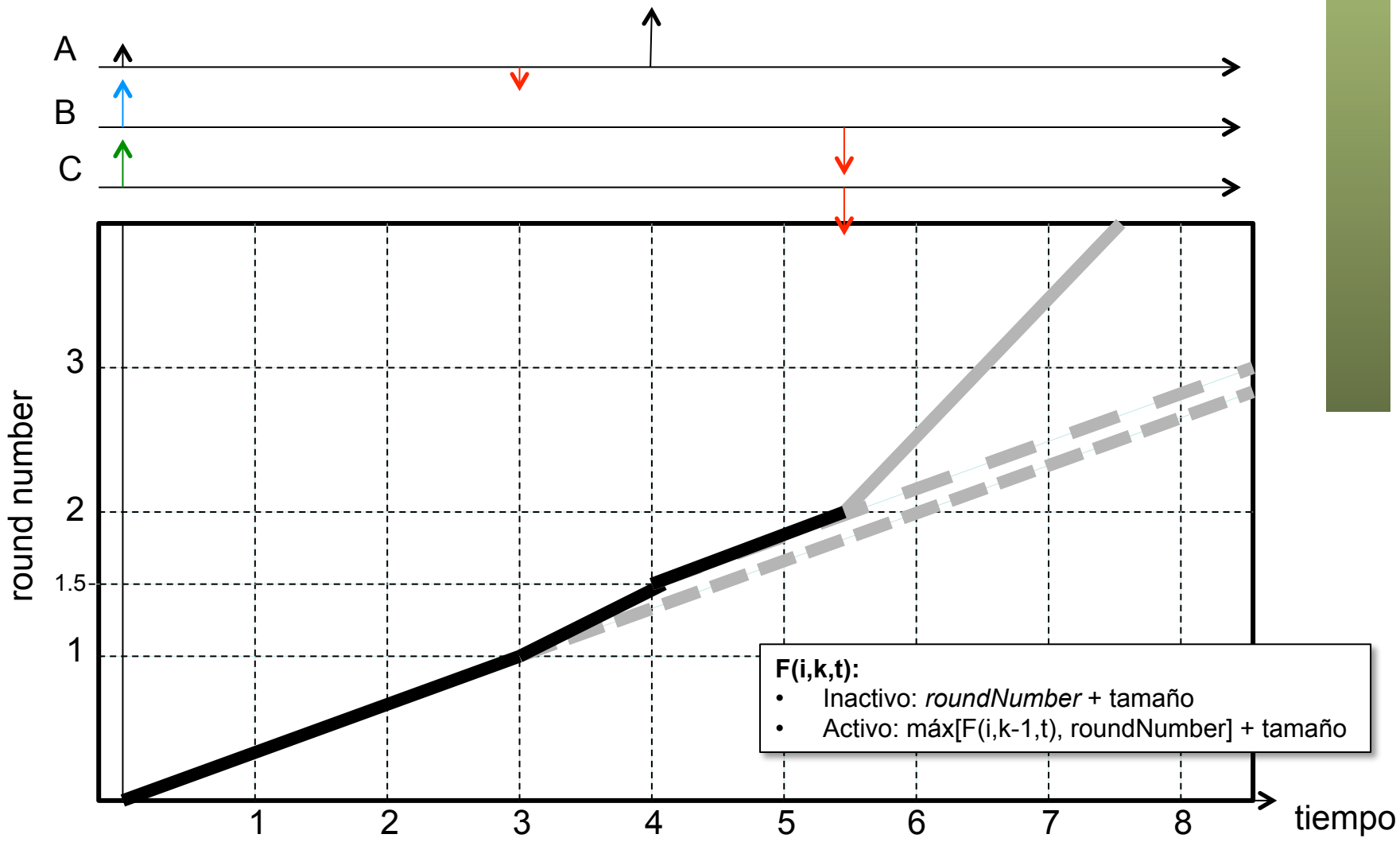
WFQ (Ejemplo)

- Se alcanza el round number 2 en t = 4 + 0.5/(1/3) = 5.5
- Entonces se completaría el envío GPS de B1 y C1



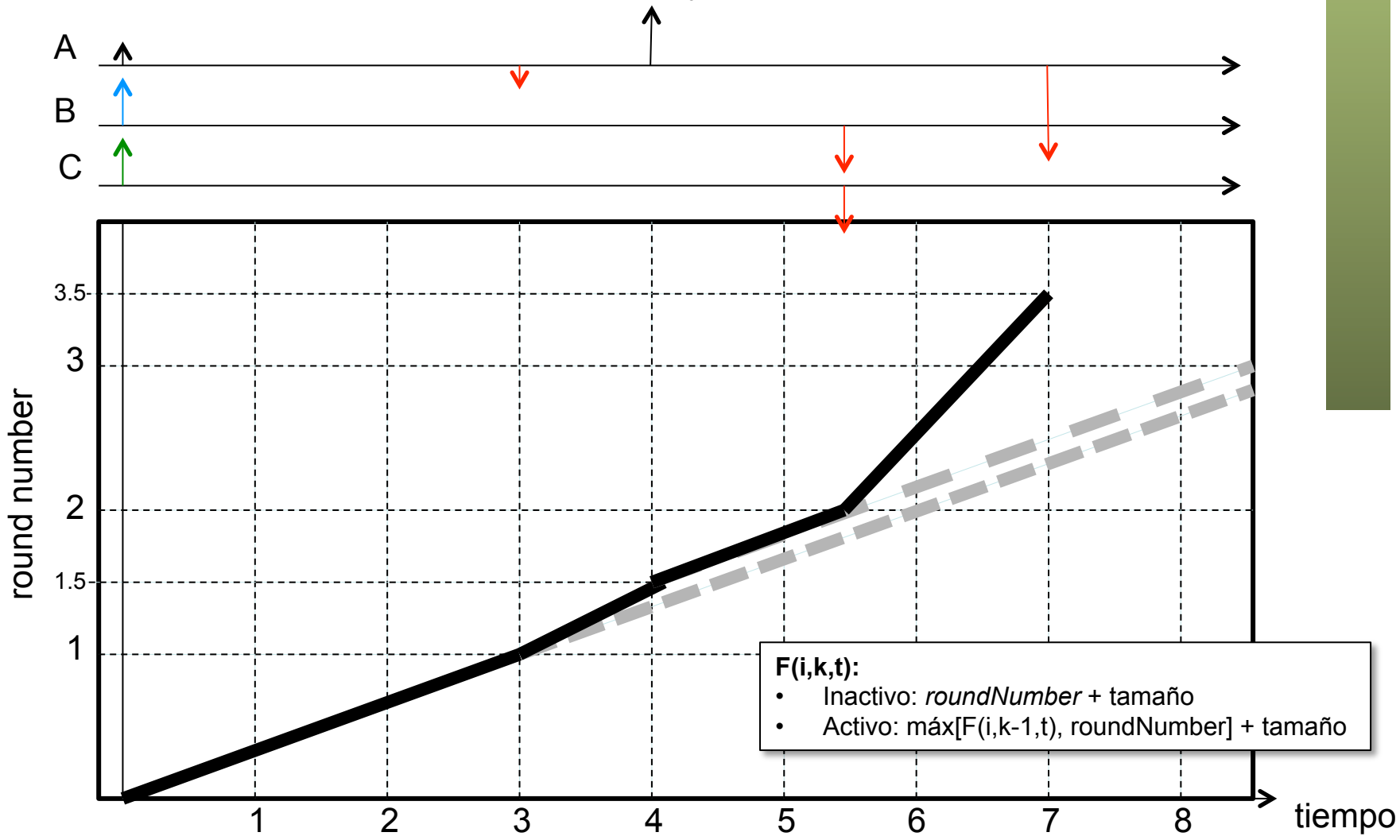
WFQ (Ejemplo)

- Queda solo una fuente activa luego ahora se avanza a 1 round por unidad de tiempo



WFQ (Ejemplo)

- Queda solo una fuente activa luego ahora se avanza a 1 round por unidad de tiempo
- En t = 7 se alcanza el round number 3.5 y termina de enviarse A2



WFQ: Cotas

Cotas (*bounds*) en WFQ

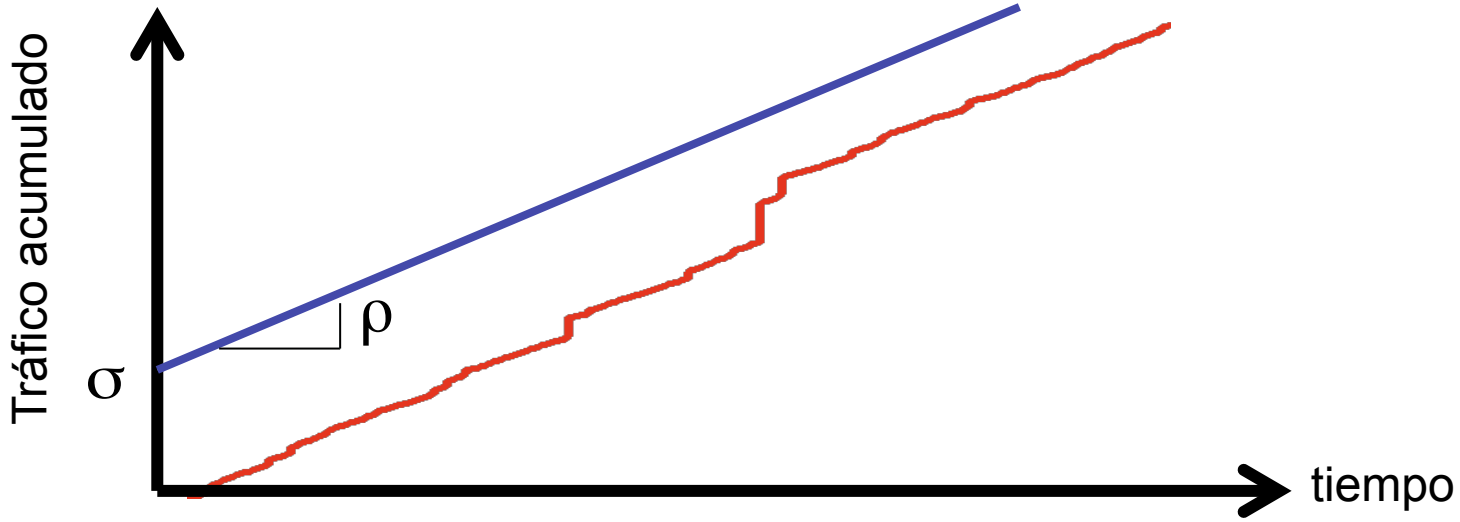
- WFQ garantiza reparto weighted max-min fair
- Eso quiere decir que cada flujo recibe una asignación proporcional a su peso

$$c_i = C\frac{\phi(i)}{\sum \phi(i)}$$

- Además pone una cota al retardo máximo
- ¿Cómo? (…)

Cotas (*bounds*) en WFQ

- Supongamos un flujo con una restricción "sigma-ro" (σ, ρ) :
  - En un intervalo t llegan como mucho $\sigma + \rho t$ bits
  - Es la salida de un *token bucket*
  - *Linear Bounded Arrival Process* (LBAP)
- Un flujo *i* con restricción ($\sigma(i)$, $\rho(i)$)
- El resto del tráfico puede no estar conformado
- (…)



Cotas (*bounds*) en WFQ

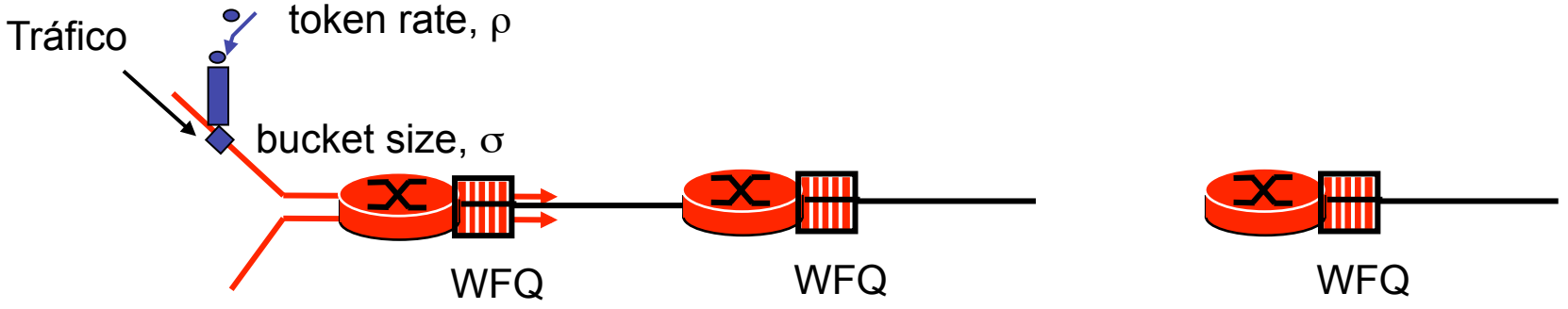
- Camino con *K* saltos (todos WFQ)
- Se le ha asignado una tasa g(i,k) en cada uno:

$$g(i,k) = r(k)\frac{\phi(i,k)}{\sum_j \phi(j,k)}$$

r(k) link rate en enlace k

- g(i) es el mínimo de g(i,k) y g(i) ≥ ρ(i)
- $P_{max}(i)$ es el mayor tamaño de paquete del flujo *i* y P_{max} en la red
- Entonces el retardo extremo a extremo (caso peor) debido a encolado y transmisión es:

$$D^*(i) \le \frac{\sigma(i)}{g(i)} + \sum_{k=1}^{K-1}\frac{P_{\max}(i)}{g(i,k)} + \sum_{k=1}^{K}\frac{P_{\max}}{r(k)}$$



Cotas (*bounds*) en WFQ

Ejemplo

- Flujo LBAP con parámetros (16 KiBytes, 150 Kbps)
- K = 10 saltos, todos a 45 Mbps, retardo de propagación total de 30ms
- Máximo tamaño de paquete de 8 KiBytes
- Queremos un retardo extremo-a-extremo máximo de 100 ms
- Entonces retardo máximo de 100 - 30 = 70 ms
  - (...)

Cotas (*bounds*) en WFQ

Ejemplo

- Flujo LBAP con parámetros (16 KiBytes, 150 Kbps)
- K = 10 saltos, todos a 45 Mbps, retardo de propagación total de 30ms
- Máximo tamaño de paquete de 8 KiBytes
- Queremos un retardo extremo-a-extremo máximo de 100 ms
- Entonces retardo máximo de 100 - 30 = 70 ms

$$D^*(i) \le \frac{\sigma(i)}{g(i)} + \sum_{k=1}^{K-1} \frac{P_{\max}(i)}{g(i,k)} + \sum_{k=1}^{K} \frac{P_{\max}}{r(k)}$$

$$\frac{16*1024*8}{g} + 9*\frac{8*1024*8}{g} + 10*\frac{8*1024*8}{45*10^6} \le 70*10^{-3}$$

- ¡¡ g > 13 Mbps !!
- El segundo término contribuye en 45.4 ms y el tercero en 14.5 ms
- El término σ/ρ solo contribuye en torno a 10.07 ms
- Al haber paquetes grandes tienen un gran efecto en el retardo de caso peor

Cotas (*bounds*) en WFQ

- Con planificadores WFQ en todo el camino
- Y que el flujo esté conformado por un leaky bucket
- Podemos garantizar un retardo máximo extremo a extremo
- No se imponen restricciones al resto de flujos en la red
- Solo hay que seleccionar los valores adecuados de reserva de BW en los enlaces

$$D^*(i) \leq \frac{\sigma(i)}{g(i)} + \sum_{k=1}^{K-1} \frac{P_{\max}(i)}{g(i,k)} + \sum_{k=1}^{K} \frac{P_{\max}}{r(k)}$$

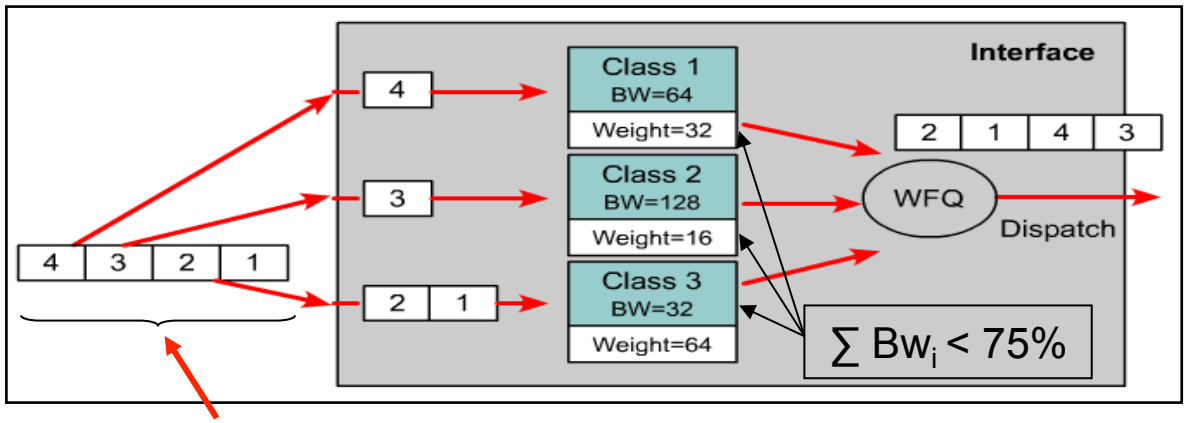
WFQ en la práctica

Típica implementación de WFQ

- Cada flujo es una "conversación" reconocida por info. de layer 3 (direcciones IP, precedencia) y de layer 4 (puertos)
- Pesos en función de los bits de precedencia (parte del TOS) de los paquetes
- No requiere configuración
- No escala (una cola por conversación)
- Alternativa: (…)

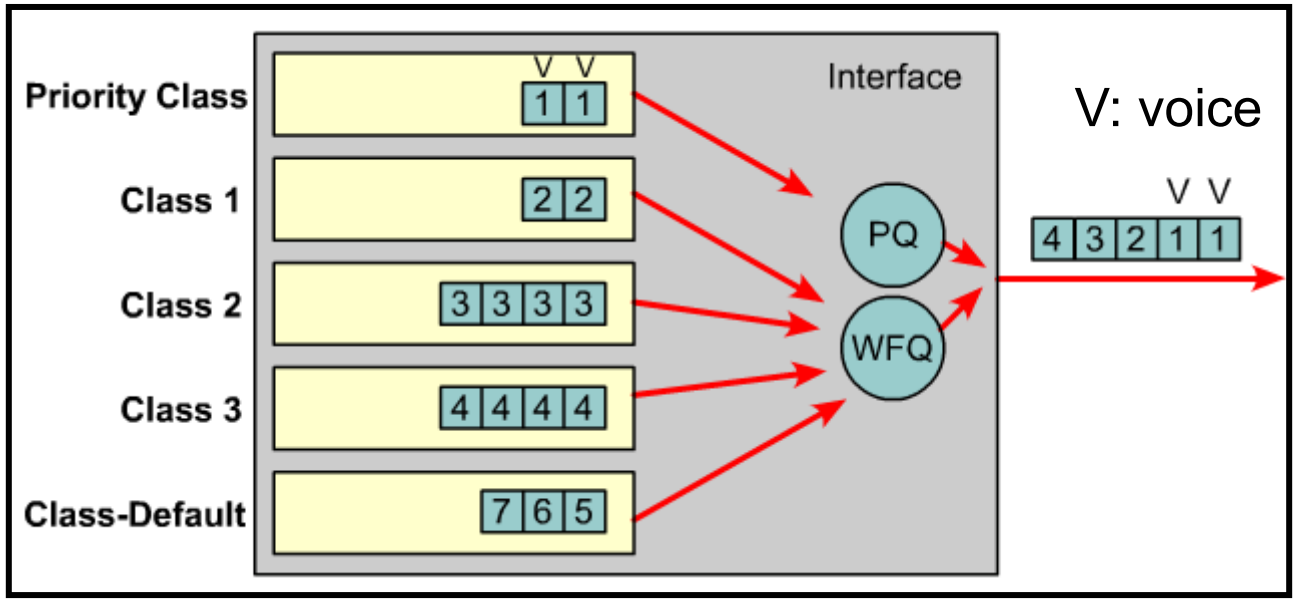
Típica implementación de WFQ

- Cada flujo es una "conversación" reconocida por info. de layer 3 (direcciones IP, precedencia) y de layer 4 (puertos)
- Pesos en función de los bits de precedencia (parte del TOS) de los paquetes
- No requiere configuración
- No escala (una cola por conversación)
- CBWFQ
  - *Class Based WFQ*
  - Especificar los filtros (clases) que determinan los paquetes que van a cada cola (una por clase, no por flujo)
  - Especificar peso para cada cola



Low Latency Queueing (LLQ)

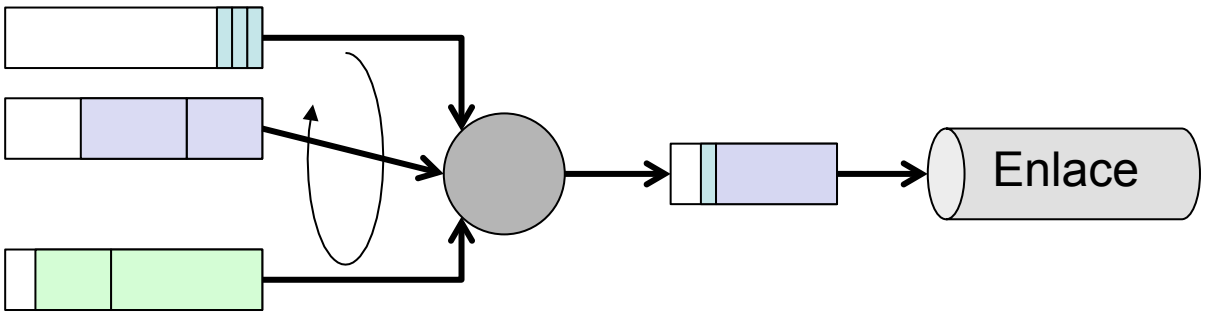
- Añade una PQ (*Priority Queue*) a CBWFQ = PQ-CBWFQ = LLQ
- Recomendable para tráfico multimedia (VoIP): bajo retardo y jitter.
- Aunque WFQ puede acotar retardo lo habitual para tráfico con requisitos estrictos de bajo retardo es usar una cola de prioridad
- Se puede configurar junto al resto de colas CBWFQ como una cola más asociada a una clase determinada.



Retardos a la salida y fragmentación

Cola en el interfaz

- Normalmente el planificador no envía los paquetes directamente al enlace
- Los envía a una cola del interfaz hardware
- Esta cola permite maximizar el throughput en el interfaz
- Es una cola FIFO: “interface FIFO” o “transmit ring buffer” o “tx-ring”
- Entre el scheduler y la cola se emplea un control de flujo para que no se desborde el tx-ring
- (…)

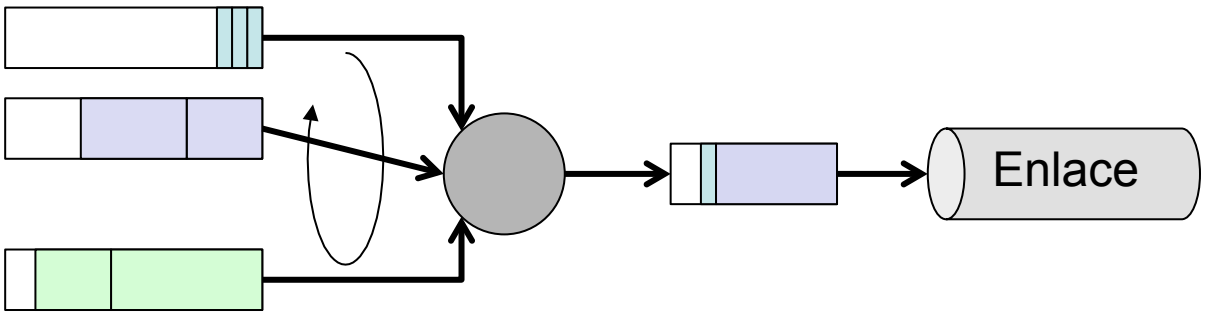


Cola en el interfaz

- Incluso un paquete con prioridad se encola ahí
- El tamaño suele estar en torno a 1-2 MTUs
- Puede introducir un retardo significativo en enlaces de muy baja velocidad
- Básicamente es el problema de que llegue un paquete de alta prioridad cuando se está transmitiendo uno de baja prioridad

$$D^*(i) \le \frac{\sigma(i)}{g(i)} + \sum_{k=1}^{K} \frac{P_{\max}(i)}{g(i,k)} + \sum_{k=1}^{K} \frac{P_{\max}}{r(k)}$$

WFQ: Efecto de encontrar paquete de otra clase delante

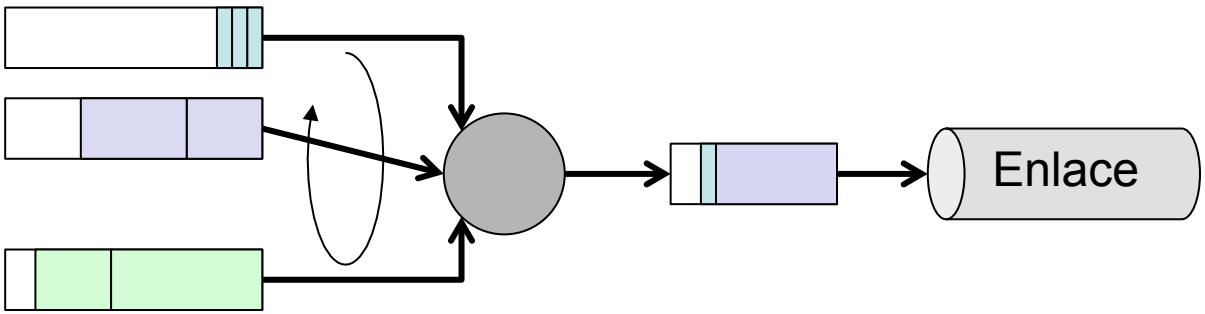


Problema

- Llega paquete IP a su cola de alta prioridad (estando ésta vacía)
- Mientras está saliendo otro paquete de clase con menor prioridad
- Retardo máximo producido si el paquete es de 1500 bytes y la línea de 256Kbps:

  (1500*8) bits / 256 Kbps = 46.8 ms!

	64Bytes	1024Bytes	1500Bytes
128Kbps	4 ms	64 ms	93 ms
256Kbps	2 ms	32 ms	46 ms
512Kbps	1 ms	16 ms	23 ms
768Kbps	0.64 ms	10.2 ms	15 ms
1Mbps	0.51 ms	8.19 ms	12 ms



Link Fragment and Interleaving (LFI)

Solución:

- Fragmentar los paquetes de datos
- Ej.: límite fragmentos "de 10ms"
- Es decir, tamaño de un paquete igual a máximo que se pueda enviar en 10 ms
- Insertar paquete de VoIP entre estos paquetes
- Asegura un retraso mucho menor
- Los paquetes VoIP no deben fragmentarse

	64Bytes	1024Bytes	1500Bytes
128Kbps	4 ms	64 ms	93 ms
256Kbps	2 ms	32 ms	46 ms
512Kbps	1 ms	16 ms	23 ms
768Kbps	0.64 ms	10.2 ms	15 ms
1Mbps	0.51 ms	8.19 ms	12 ms