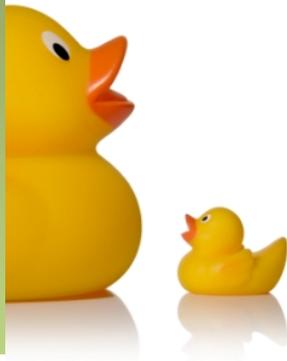


Max-min *Weighted* Fair Share

- En ocasiones se desea una asignación preferente a unos flujos frente a otros
- Se asocia esta preferencia con unos pesos w_1, w_2, \dots, w_n
- Extensión:
 - Los recursos se asignan en orden de demanda creciente, normalizada por el peso
 - Ningún cliente recibe más de lo que solicita
 - Aquellos cuya demanda no se pueda satisfacer se reparten el remanente del recurso en proporción a sus pesos



Max-min WFS (Ejemplo)

- Recurso: 20. Demandas: 4, 2, 10 y 8. Pesos: 2.5, 4, 0.5 y 1
- Normalización de los pesos:
 - Que el menor valga 1
 - Pesos normalizados: 5, 8, 1 y 2
- En vez de 4 clientes es como si hubiera $5+8+1+2 = 16$
- $C/n = 20/16 = 1.25$
- (...)

upna

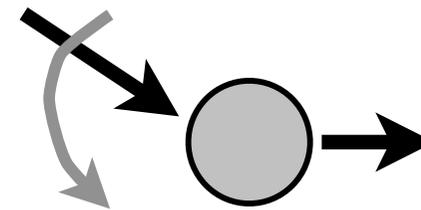
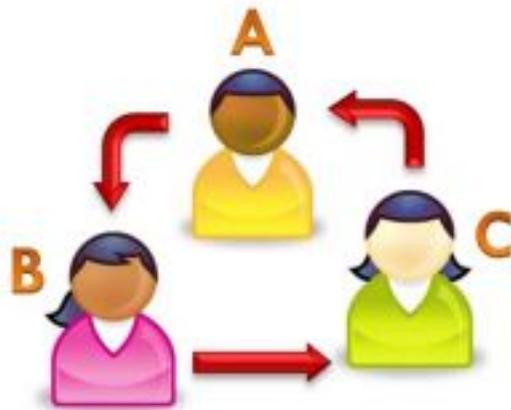
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Tecnologías Avanzadas de Red
Área de Ingeniería Telemática

Weighted Round Robin

Weighted Round Robin (WRR)

- Opera por “turnos”. Un peso para cada clase
- Se normaliza el peso por el tamaño medio de paquete en la clase $\frac{\phi(i)}{s_i}$
- Normaliza el resultado para que sean enteros
- En cada turno visita cada cola (en RR) y sirve tantos paquetes como su peso normalizado
- Ejemplo:
 - (...)



Weighted Round Robin (WRR)

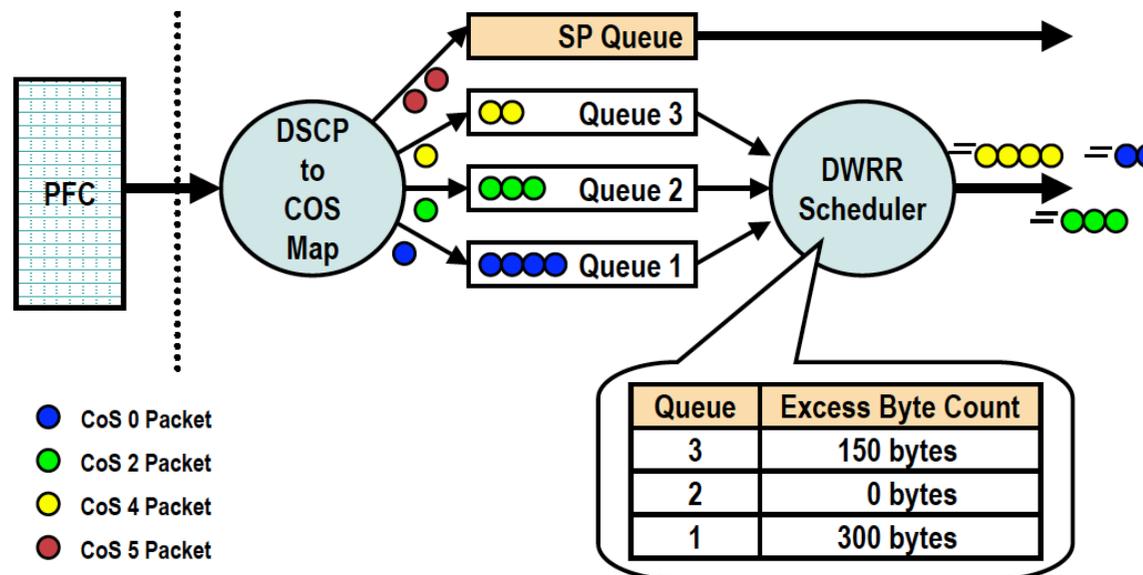
Problemas

- Necesita saber el tamaño medio de paquete de cada clase
- Más sencillo si los paquetes son de tamaño constante
- Es justo solo por encima de la escala de la duración del turno
- Ejemplo:
 - (...)



Deficit Round Robin (DRR)

- Permite hacer un RR con pesos sin conocer tamaños medios de paquetes (veamos primero versión **sin pesos**)
- Cada clase mantiene un contador de *déficit* inicializado a 0
- En cada turno se añade q (el *quantum*) al contador de cada clase si tiene paquetes por servir, si no se resetea
- El planificador visita cada clase y sirve el primer paquete de la cola si su tamaño es menor que su contador de déficit
- y decrementa el contador en el tamaño del paquete
- (...)

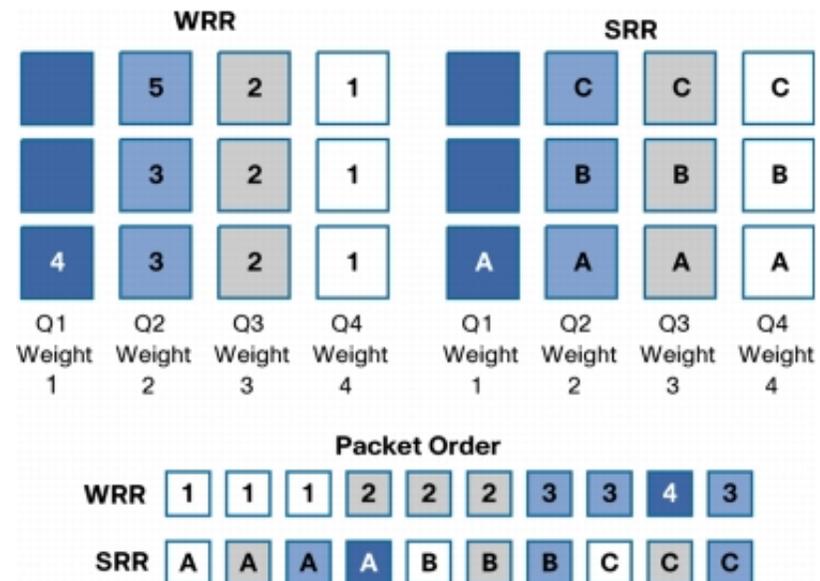


Deficit Round Robin (DRR)

- En la versión con pesos el *quantum* es el peso de cada clase
- El *quantum* debería ser al menos del tamaño máximo de paquete para servir alguno en todos los turnos
- Es sencillo de implementar

Shaped Round Robin (SRR)

- Modo *Shaped*: limita BW consumido por cada clase
- Modo *Shared*: el tiempo no utilizado por una clase lo usan el resto
- A partir de una cierta escala sirven los mismos paquetes de cada cola WRR y SRR
- SRR los envía con un orden diferente, más entremezclados de las diferentes clases



upna

Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Tecnologías Avanzadas de Red
Área de Ingeniería Telemática

Weighted Fair Queueing

GPS

- Partiendo de *Processor Sharing* se asocia un peso $\phi(i)$ a cada cola y entonces la cantidad de servicio es proporcional al mismo
- Ofrece *weighted max-min fairness* y lo llamamos *Generalized Processor Sharing (GPS)*
- En cualquier caso, en la realidad no podemos servir fluidos sino que servimos paquetes así que solo podremos aproximarlos
- Round Robin es una aproximación a PS
- WRR es una aproximación a GPS



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

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*
- En resumen
 - Simular GPS
 - Enviar los paquetes en el mismo orden que GPS



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: $\max[F(i,k-1,t), \text{round_number}] + \text{tamaño en bits}$
- 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: $\max[F(i,k-1,t), \text{round_number}] + \text{tamaño} / \text{peso}$
- 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

upna

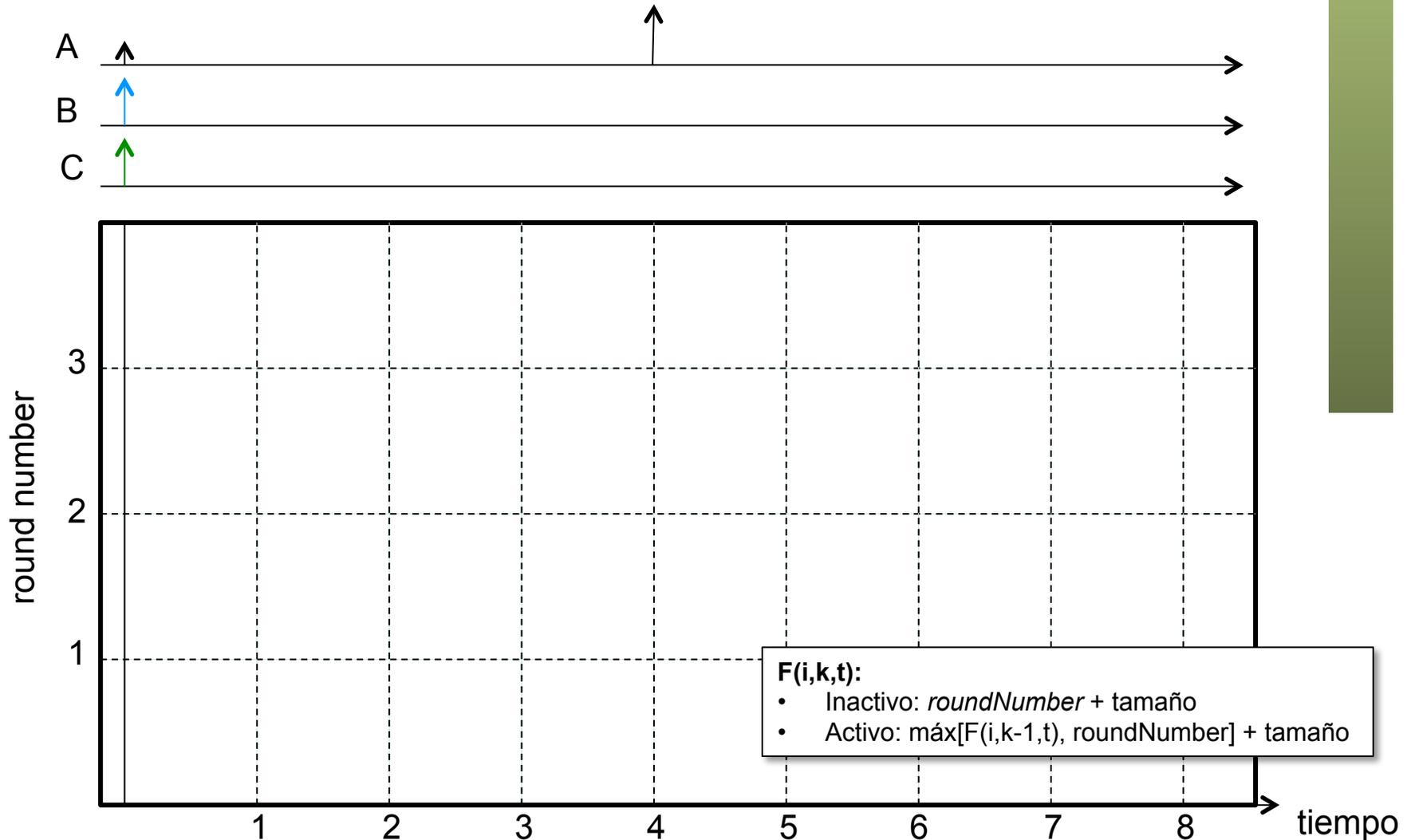
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Tecnologías Avanzadas de Red
Área de Ingeniería Telemática

WFQ: Ejemplo

WFQ (Ejemplo)

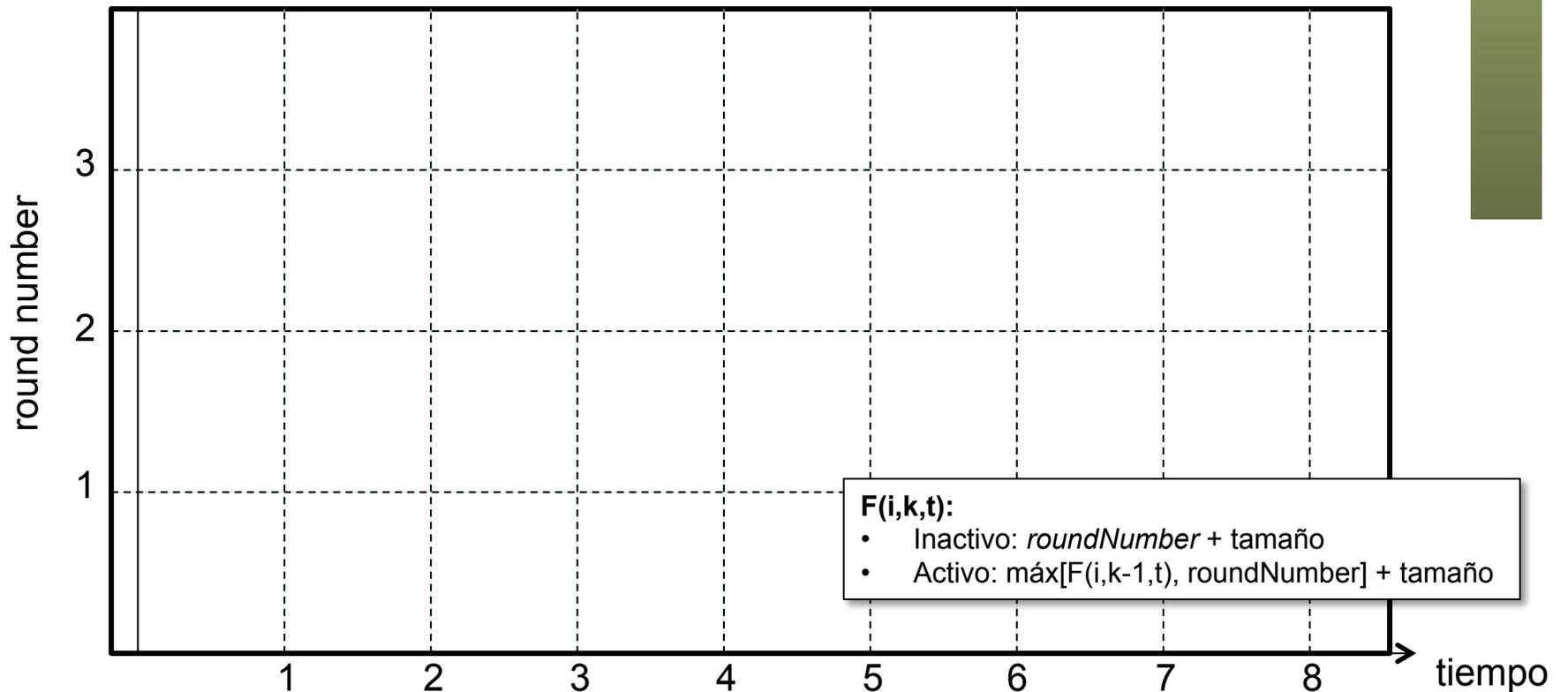
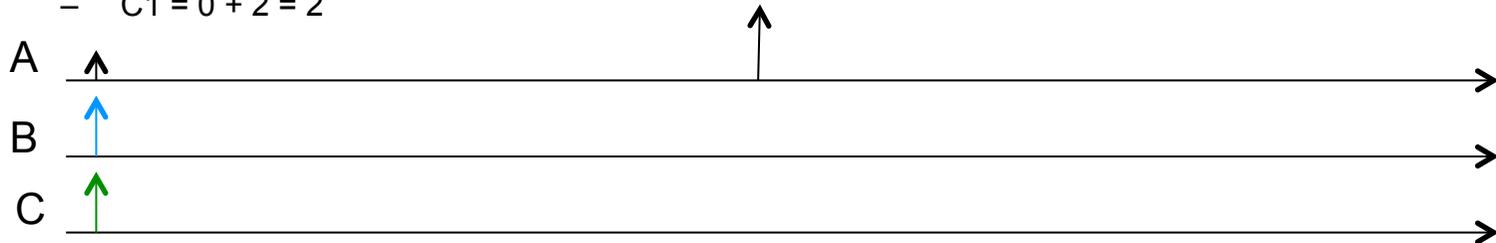
- Enlace a 1 unidad/s
- Llegadas de tamaños 1, 2 y 2 unidades en $t=0$ y de tamaño 2 unidades en $t=4$



WFQ (Ejemplo)

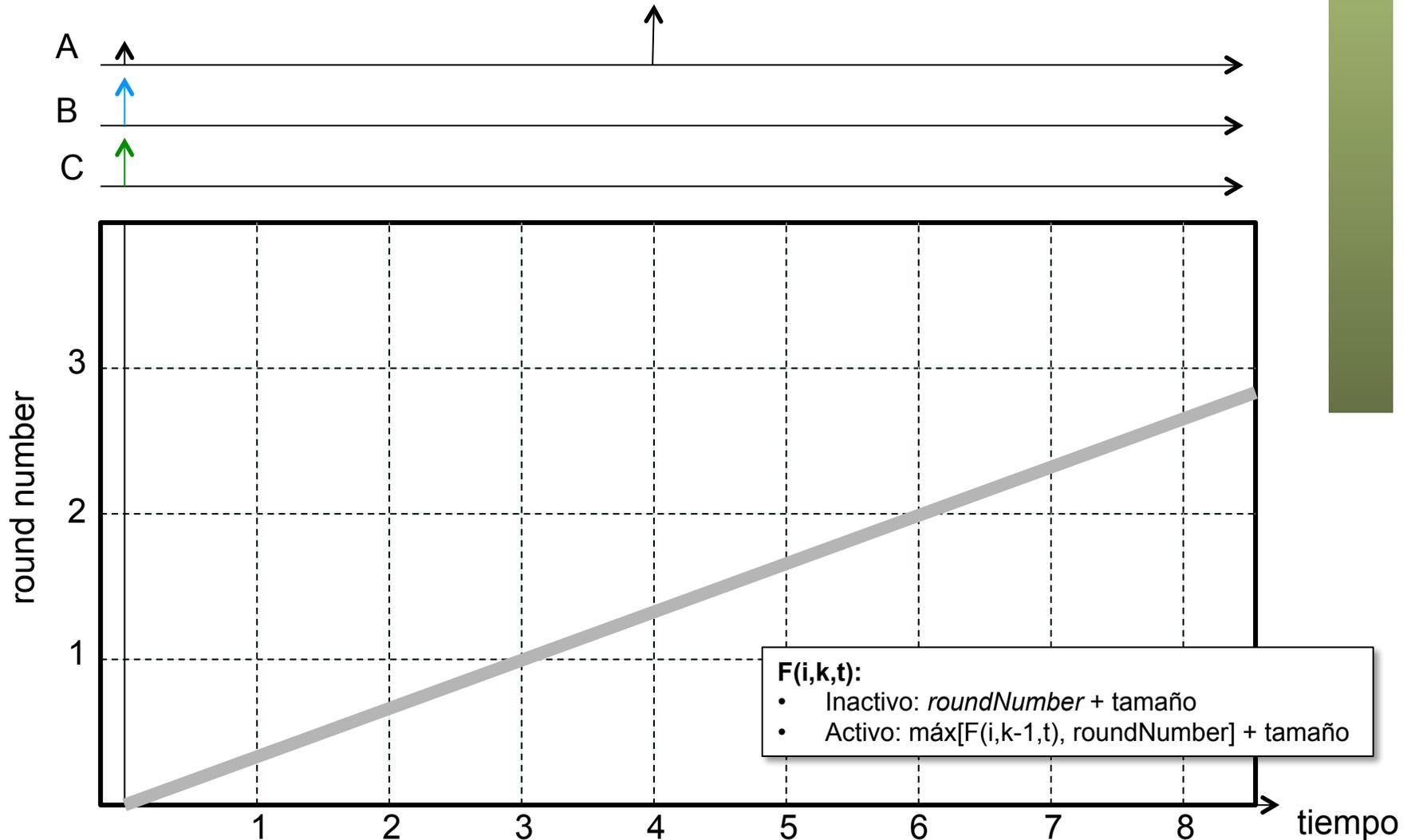
- Finish numbers:

- $A1 = 0 + 1 = 1$
- $B1 = 0 + 2 = 2$
- $C1 = 0 + 2 = 2$



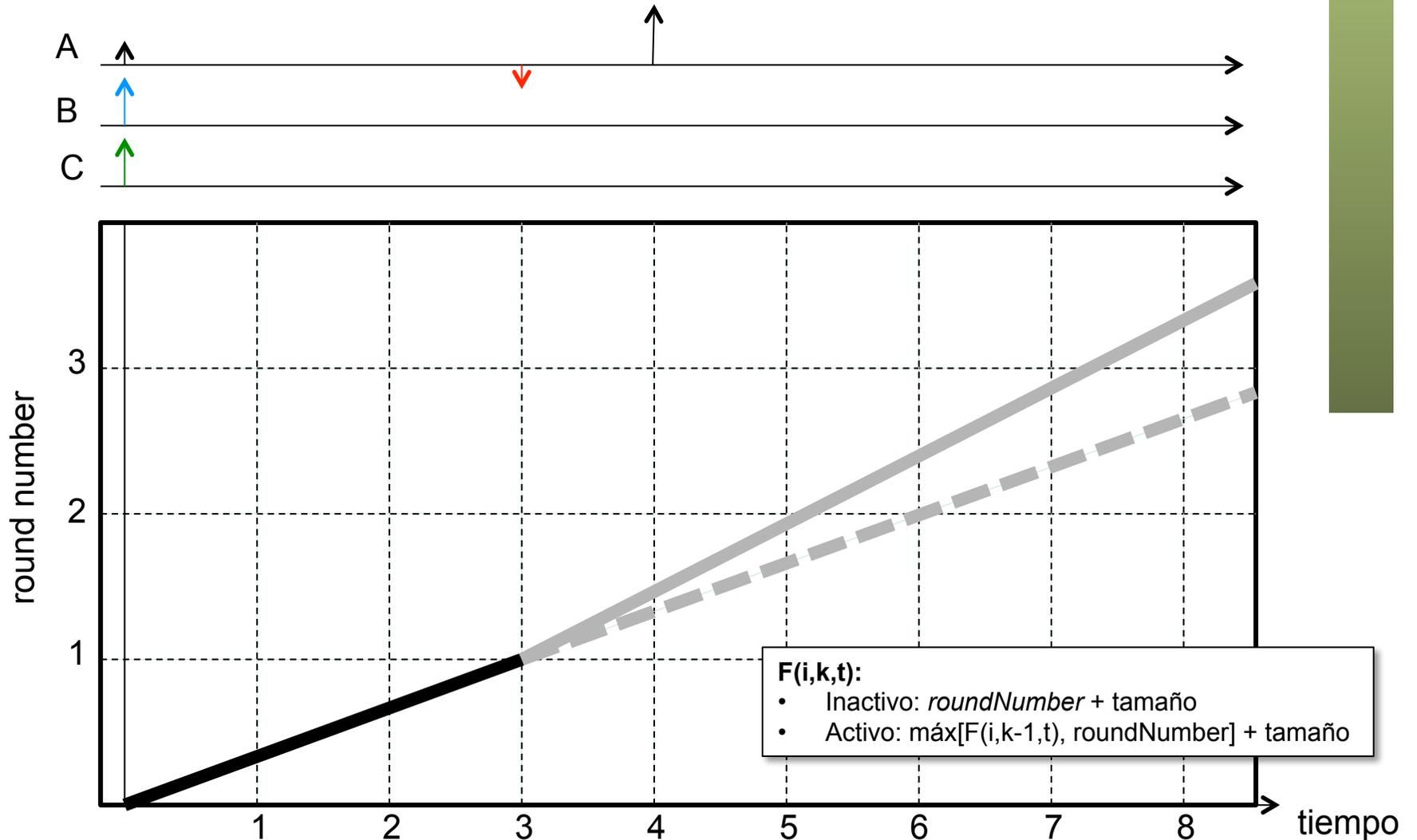
WFQ (Ejemplo)

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



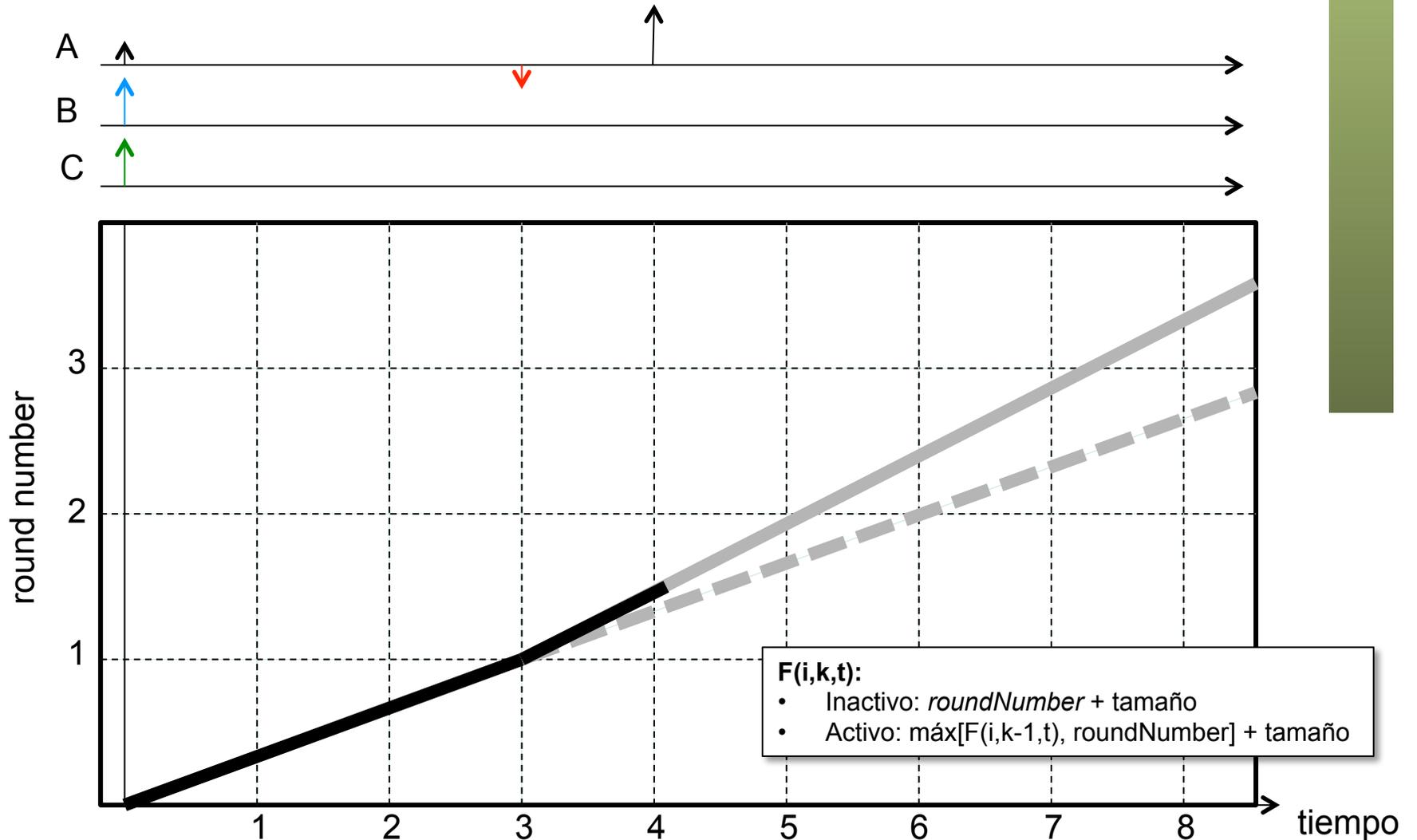
WFQ (Ejemplo)

- 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



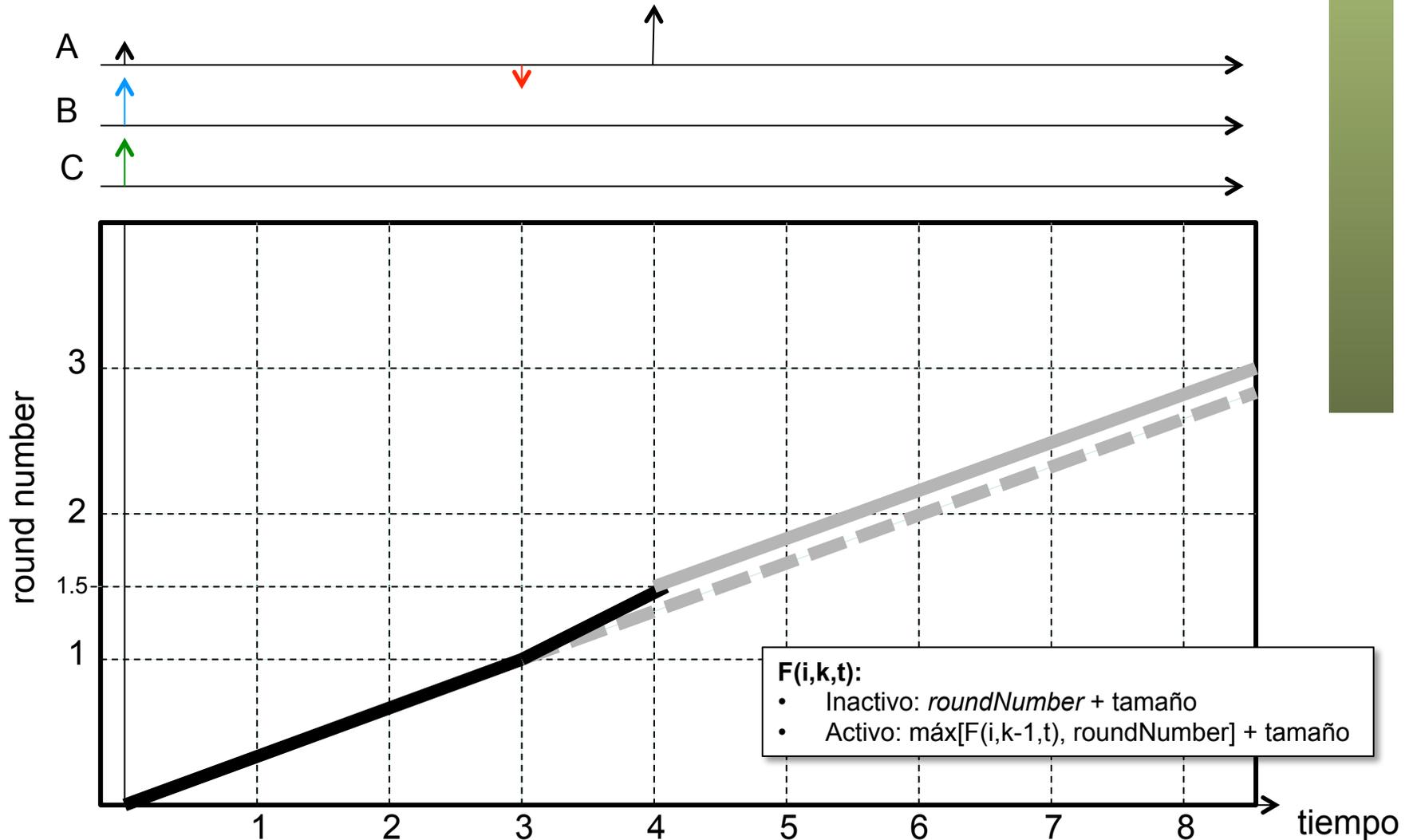
WFQ (Ejemplo)

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



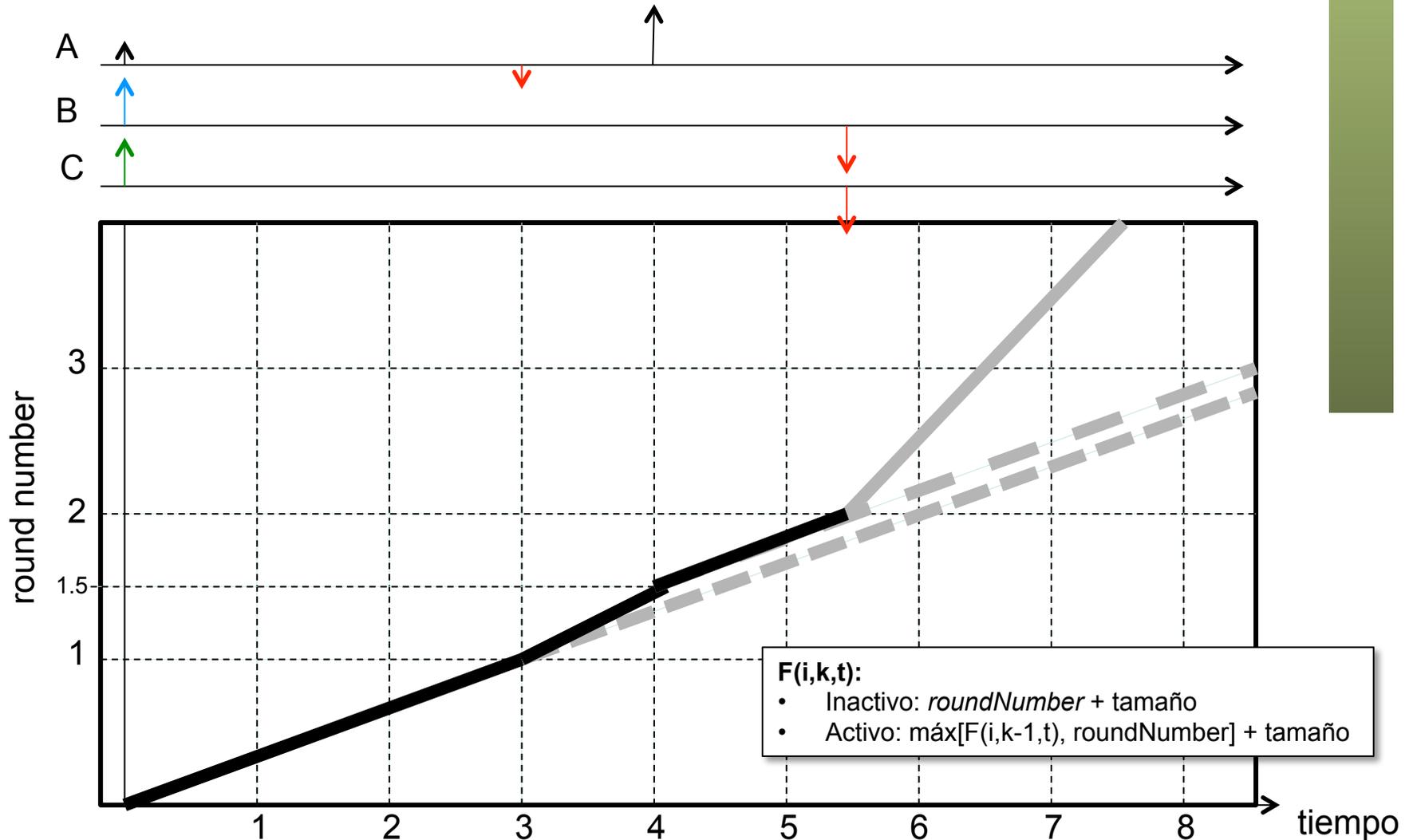
WFQ (Ejemplo)

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



WFQ (Ejemplo)

- 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

