Arquitectura de Sistemas

Planificación multiprocesador y de tiempo real

Gustavo Romero López

Updated: 14 de marzo de 2025

Departamento de Ingeniería de Computadores, Automática y Robótica

Índice

Lecturas recomendadas

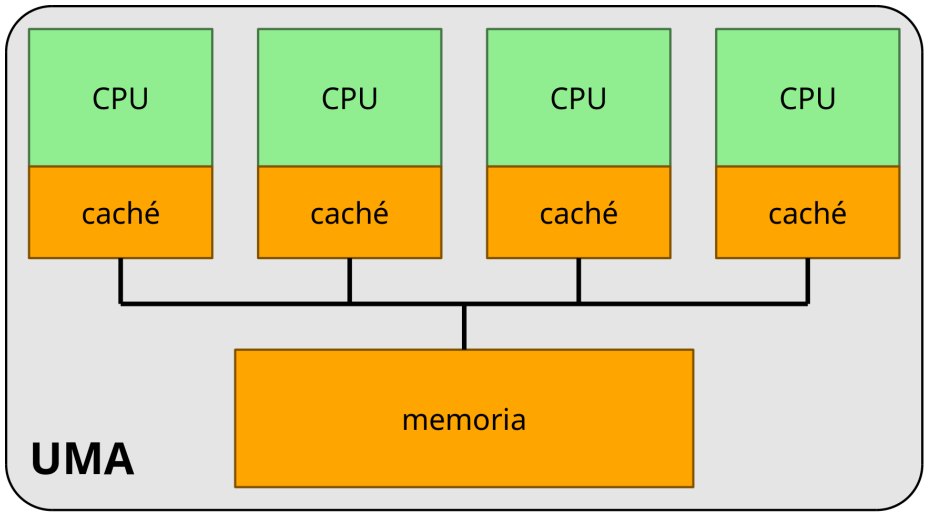
Silberschatz	Fundamentos de Sistemas Operativos (5, 19.5)
Stallings	Sistemas Operativos (9, 10)
Tanuenbaum	Sistemas Operativos Modernos (2.5, 7.4, 8.1.4, 8.2.6)

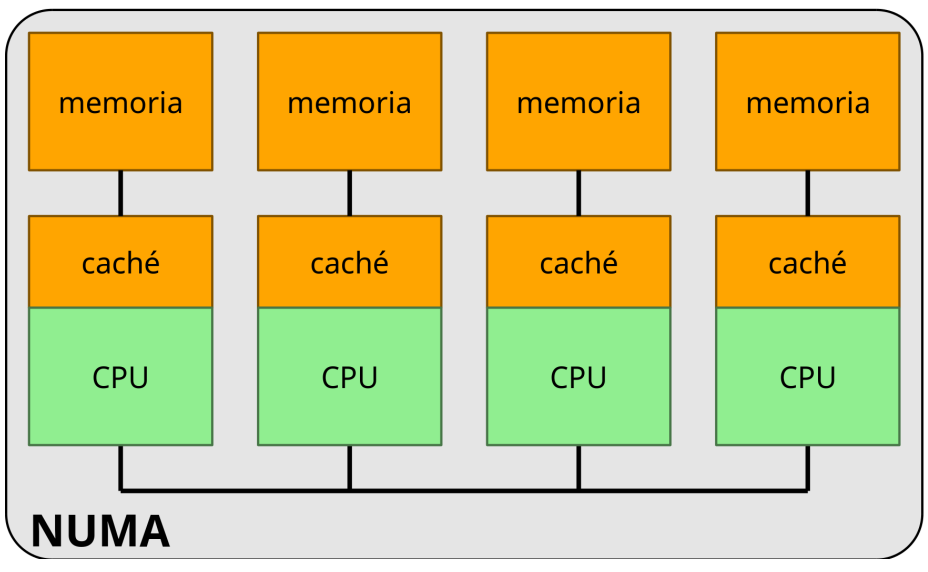
Clasificación de multiprocesadores

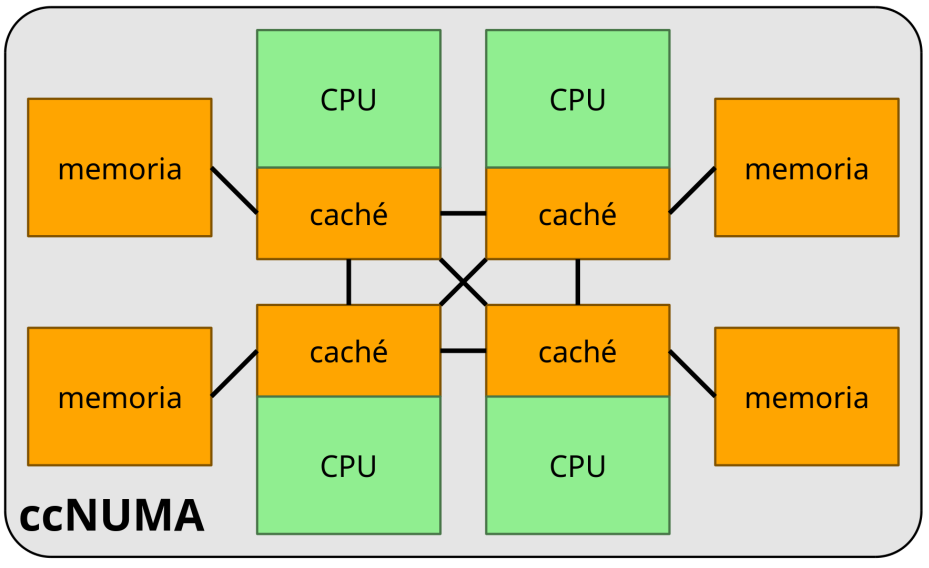
- **Débilmente acoplados**:
  - Cada procesador tiene su propia memoria y dispositivos de E/S.
  - Ejemplo: un conjunto de estaciones de trabajo (*"cluster"*).
  - Cada uno controlado por propio sistema operativo propio.

- **Fuertemente acoplados**:
  - Los procesadores comparten la memoria principal:
    - Acceso uniforme a memoria (**UMA**).
    - Acceso no uniforme a memoria (**NUMA**).
  - Controlados por un único sistema operativo.
    - Planificación centralizada.
    - Planificación descentralizada.

Diferencia UMA/NUMA/ccNUMA







Multiprocesadores fuertemente acoplados

- Multiprocesamiento **asimétrico**
  - Relación maestro/esclavo:
    - El maestro gestiona la planificación, interrupciones,...
    - Los esclavos ejecutan aplicaciones.
  - Inconveniente: baja robustez/tolerancia a fallos en caso de que falle el maestro.

- Multiprocesamiento **simétrico (SMP)**
  - Cualquier procesador puede ejecutar cualquier tarea.
  - Durante su tiempo de vida las hebras pueden ejecutarse...
    - en cualquier procesador.
    - siempre en el mismo procesador (afinidad estricta).
  - Las peticiones de interrupciones pueden llegar a cualquier procesador.

Motivación (1)

- En sistemas con un **único procesador** todo se ejecuta sobre este y la única decisión que es necesario tomar es **cuándo** ejecutar **qué** hebra/proceso.
- Simplificando... en cada instante, como mucho, se está realizando una actividad en el procesador.
- Otras tareas a las que prestar atención en un SMP:
  - Actividades de las aplicaciones.
    - Relativas a procesos.
    - Relativas a las hebras de un proceso multihebra.
  - Actividades del sistema.
    - ¿Cómo asignar CPU a un cliente? ¿Y a un servidor?
    - ¿Cómo planificar actividades periódicas?
  - Actividades del núcleo.
    - ¿Cómo prevenir interferencias mutuas de rutas críticas?

Motivación (2)

La planificación SMP es complicada porque...

- Puede interesar que un procesador quede desocupado incluso aunque haya hebras preparadas.
- Existen SMP heterogéneos en los que habrá que considerar...
  - procesadores con diferentes velocidades.
  - procesadores con diferentes conjuntos de instrucciones.
- Pueden aparecer anomalías $\simeq$ comportamientos que van en contra del sentido común.

Anomalías SMP (R. Graham)

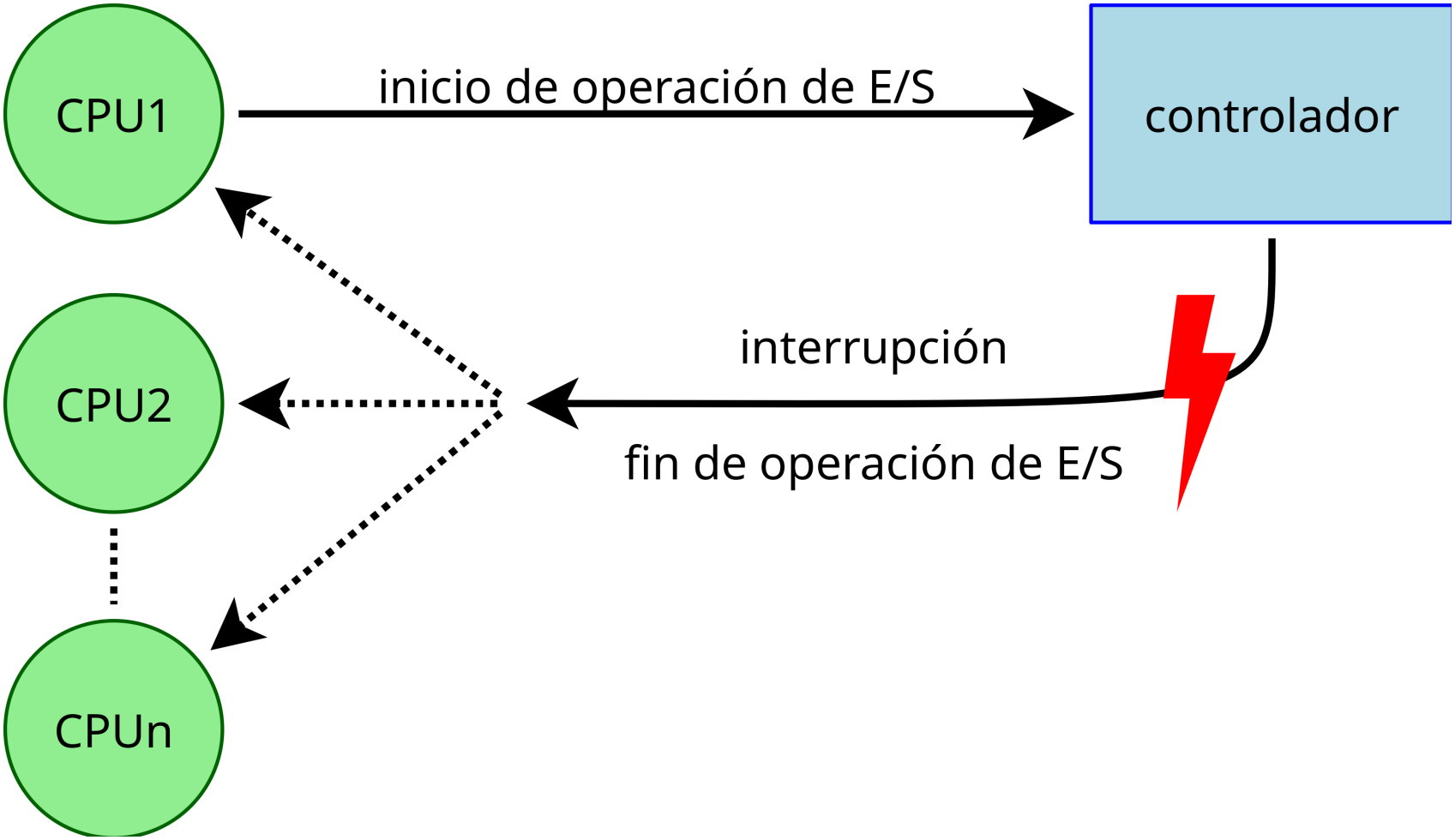
- Consideremos por ejemplo $p > 1$ procesadores y $h > 1$ hebras y ciertas restricciones de precedencia.
- R. Graham [1] demostró la existencia de anomalías respecto al tiempo de estancia máximo (TE_{max}).
  - Añadir otro procesador puede incrementar TE_{max}.
  - Eliminar restricciones de precedencia puede incrementar TE_{max}.
  - Reducir los tiempos de ejecución los procesos puede incrementar TE_{max}.

[1] R.L. Graham: Bounds on Multiprocessor Timing Anomalies. SIAM, J. Applied Mathematics. 1969.

Problemas adicionales de la planificación multiprocesador

- ¿Quién manejará las interrupciones?
- ¿Cuando se crea una nueva hebra debería ejecutarse en el mismos procesador o en otro?
- ¿Cómo planificar las hebras de un proceso multihebra?
- ¿Debe una hebra permanecer en el mismo procesador en que comenzó a ejecutarse o puede cambiar?
- ¿Cuáles son los criterios habituales para migrar o fijar hebras entre procesadores?
- ¿Cuándo conseguir la información de planificación y cómo recolectarla y almacenarla?

Gestión de interrupciones en un SMP



Planificación del manejador de interrupción (1)

Hay cuatro posibilidades:

1. La interrupción puede ser manejada en **cualquier** procesador.

2. La interrupción debe ser manejada en el procesador que **inicio** la E/S porque...
   - La hebra que ha iniciado la E/S podría ser **afín** a dicho procesador.
   - La hebra podría conservar su huella en la **caché** del procesador.
   - La hebra podría **seguir** ejecutándose sobre el procesador en caso de E/S asíncrona.

Planificación del manejador de interrupción (2)

3. La interrupción podría gestionarse en un procesador que **ya** esté ejecutando otro manejador de interrupción.
   - \+ **Ahorra** un cambio de modo (usuario → núcleo) + planificar.
   - \- Puede **retrasar** el manejo de interrupciones (si no pueden anidarse → procesamiento en cola).
   - \- Se **penalizando** a la "inocente" hebra actual.

4. La interrupción puede ser manejada en el procesador con la actividad de **menor prioridad**, por ejemplo, la hebra ociosa.

Ejecución de una nueva hebra

Dependiendo de la aplicación puede ser conveniente planificar las nuevas hebras...

- En el procesador que las crea.
  - la hebra nueva y su creadora podría cooperar compartiendo datos, ej: varias hebras de un mismo proceso.
- Sobre un procesador específico.
  - Un procesador cercano para mejorar la cooperación en máquinas NUMA.
  - Para balancear la carga del sistema o de la aplicación.
  - Cuando el programador conoce las diferencias entre los procesadores del sistema (SMP heterogéneo).
- Sobre cualquier procesador para mejorar el paralelismo.

Planificación de hebras y procesos

Procesos **monohebra**:

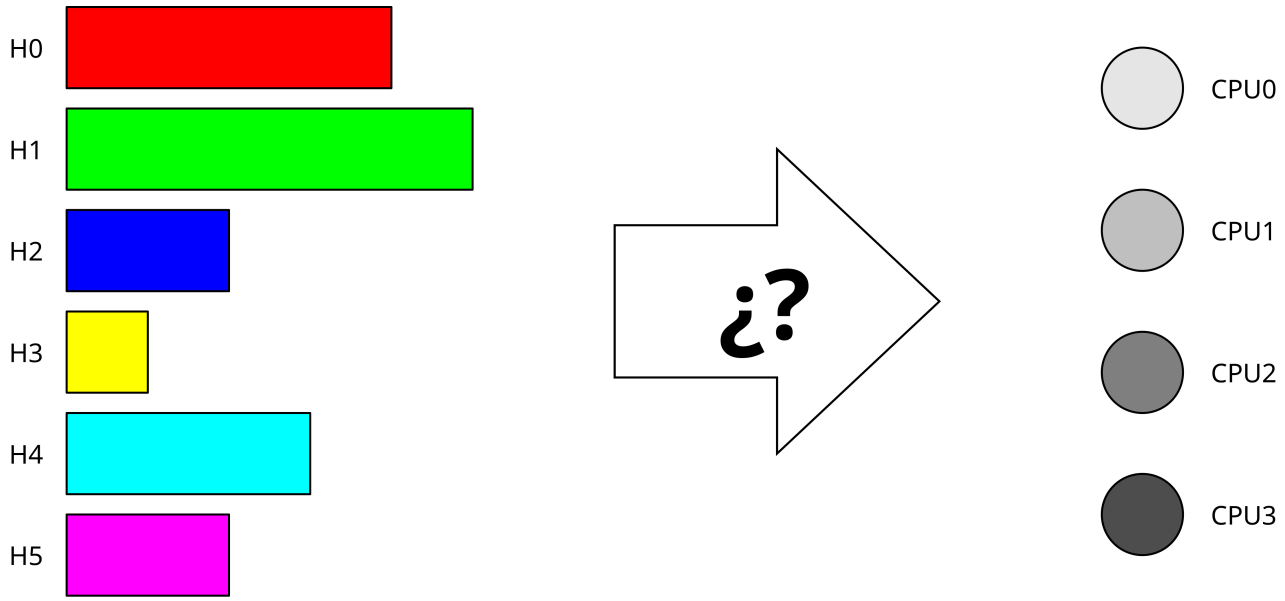
- Planificar jutos procesos que **comparten** código/datos...
  - reduce el tiempo de carga por compartir la **caché**.
  - reduce el tiempo de traducción por compartir la **TLB**.
- La **planificación anónima** puede **reducir** el tiempo de **estancia**.

Procesos **multihebra**:

- Usar un único procesador permite **ahorrar**...
  - tiempo de carga de **caché** y **TLB**
  - pero **elimina** el **paralelismo** por completo.
- Usar muchos procesadores mejora el **paralelismo** pero incrementa los tiempos de carga.
- Planificar **las hebras** de un proceso al **juntas** (planificación en grupo) puede **mejorar** su ejecución **paralela**.

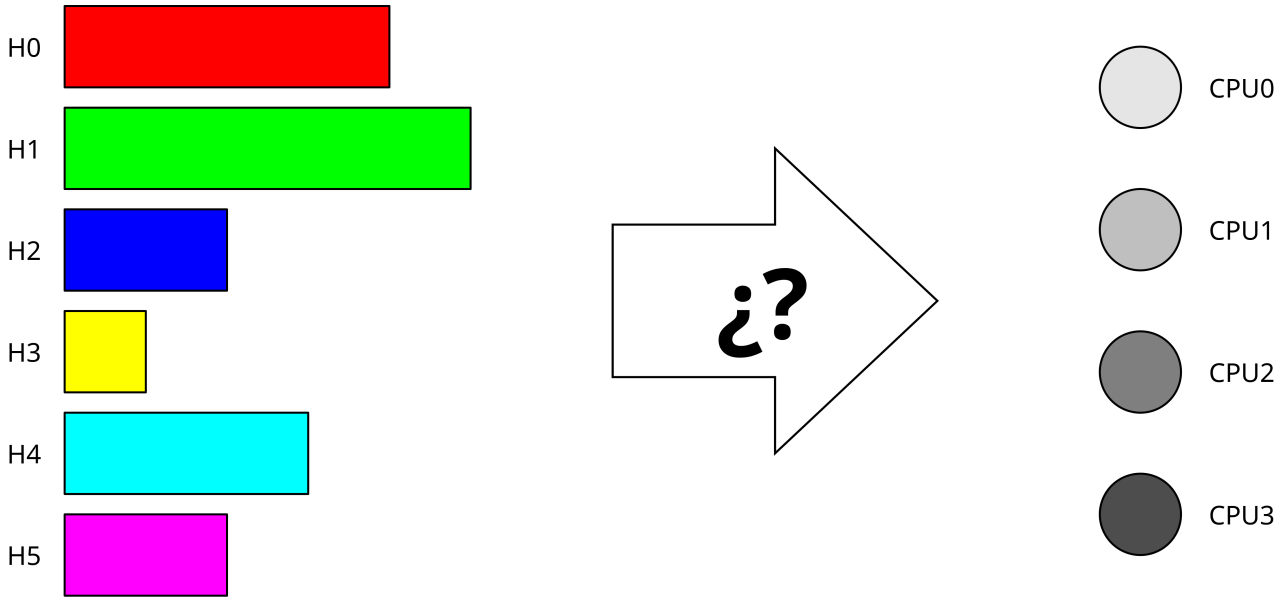
Parámetros de planificación adicionales (1)

Supongamos un proceso multihebra que hay que planificar sobre un sistema multiprocesador fuertemente acoplado, inicialmente vacio.



1. Número de procesadores involucrados.
2. Relaciones de precedencia.
3. Coste de comunicación.

Parámetros de planificación adicionales (2)

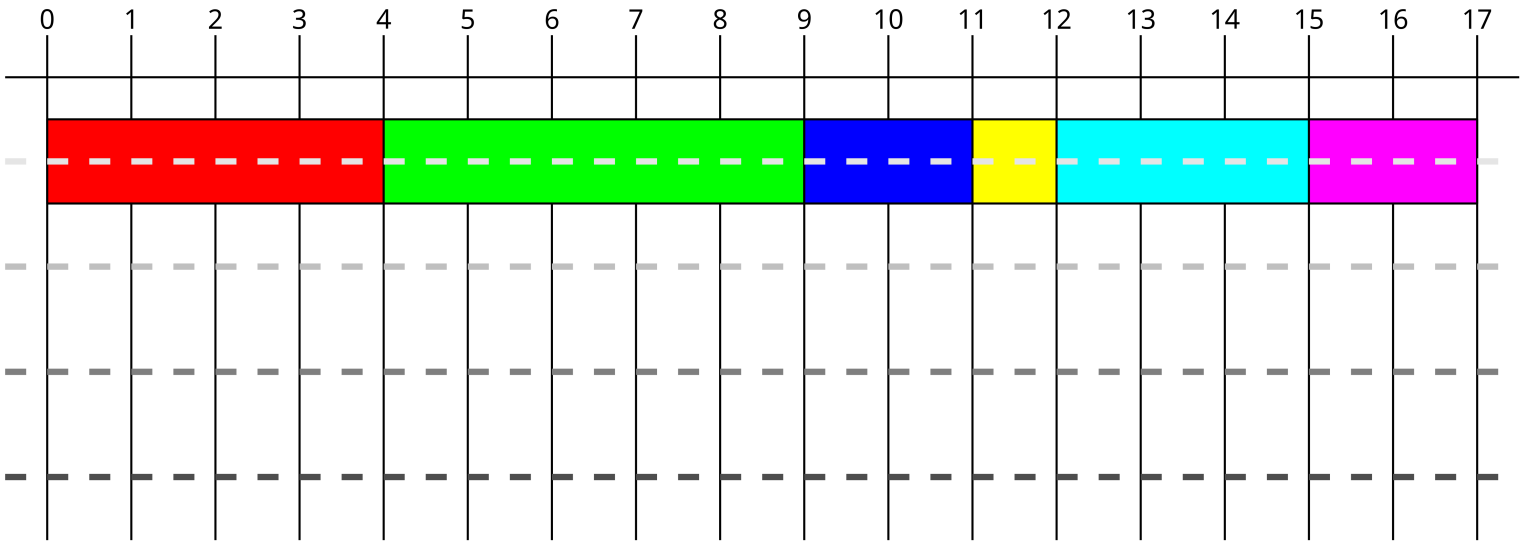


Número de procesadores involucrados:

- 1 procesador.
- $p < n$ procesadores.
- todos los procesadores.

¿Ventajas e inconvenientes tiene cada una de estas opciones?

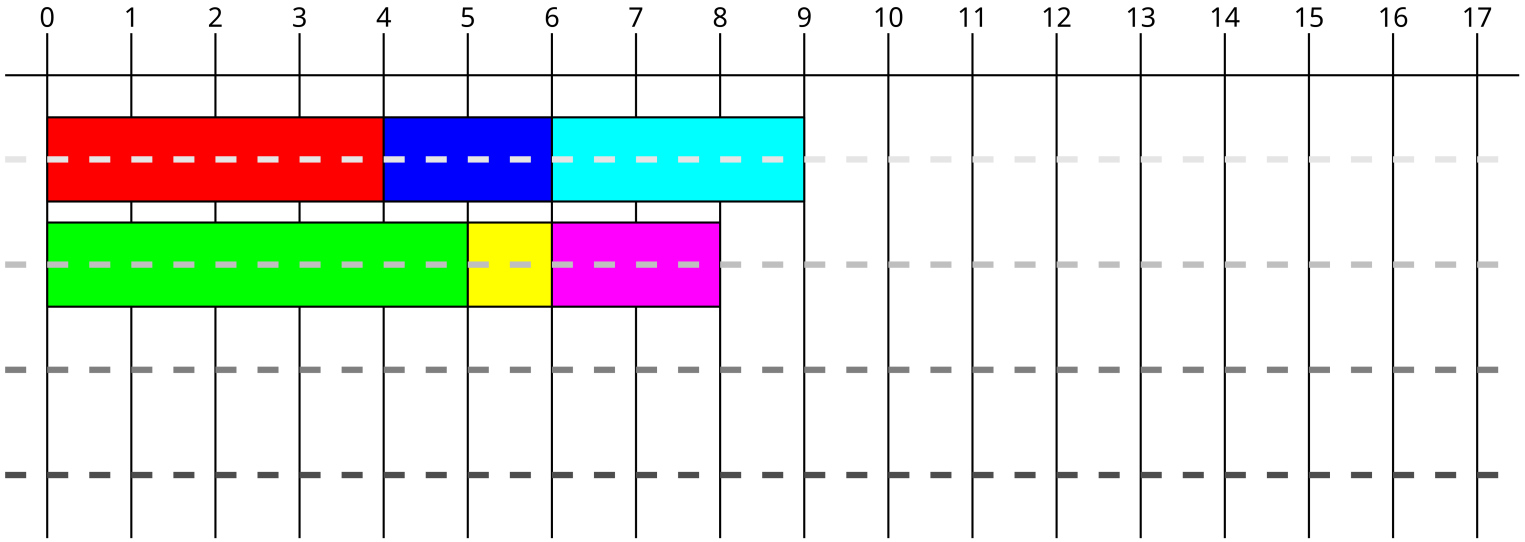
Parámetros de planificación adicionales (3)



1 procesador

- Ventajas:
  - Solución idéntica a un sistema uniprocesador.
  - Los procesadores adicionales pueden dejarse para otras aplicaciones.
- Inconvenientes:
  - Se desaprovecha el paralelismo ofrecido por la máquina.
  - El tiempo de estancia es innecesariamente elevado.

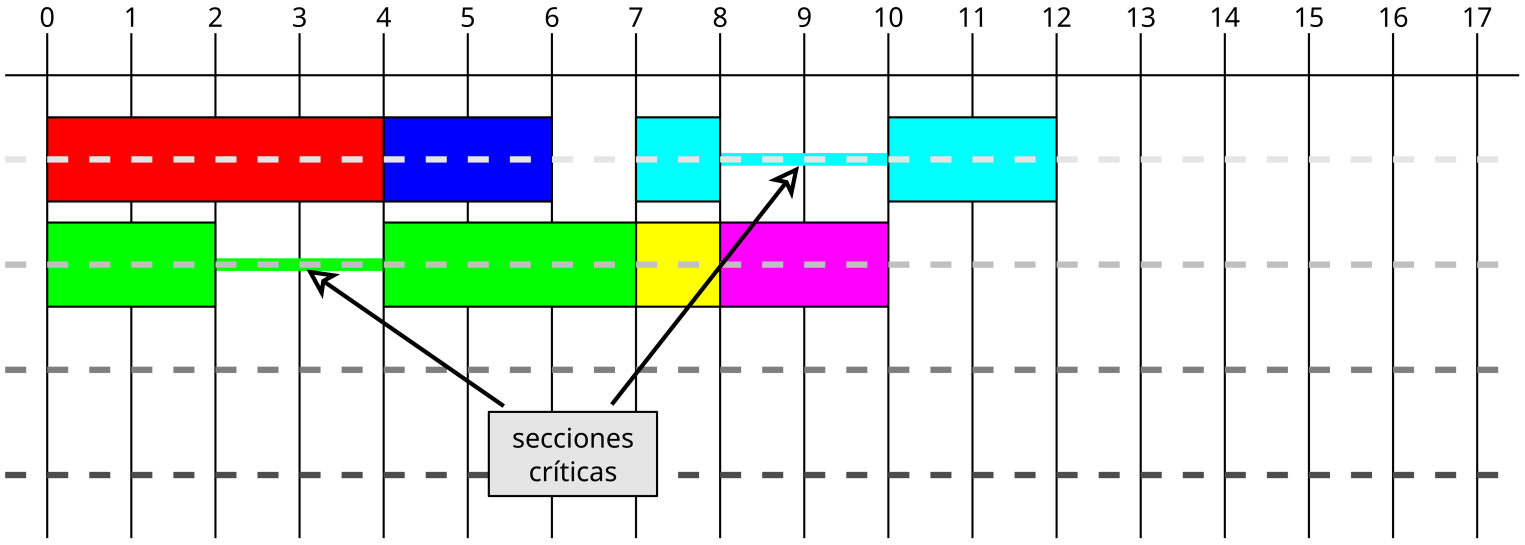
Parámetros de planificación adicionales (4)



2 procesadores

- Ventajas:
  - Menor tiempo de estancia (esperado)
- Inconvenientes:
  - Debido a la existencia de **secciones críticas** y **relaciones de precedencia** los tiempos de estancia pueden ser mayores de lo esperado.

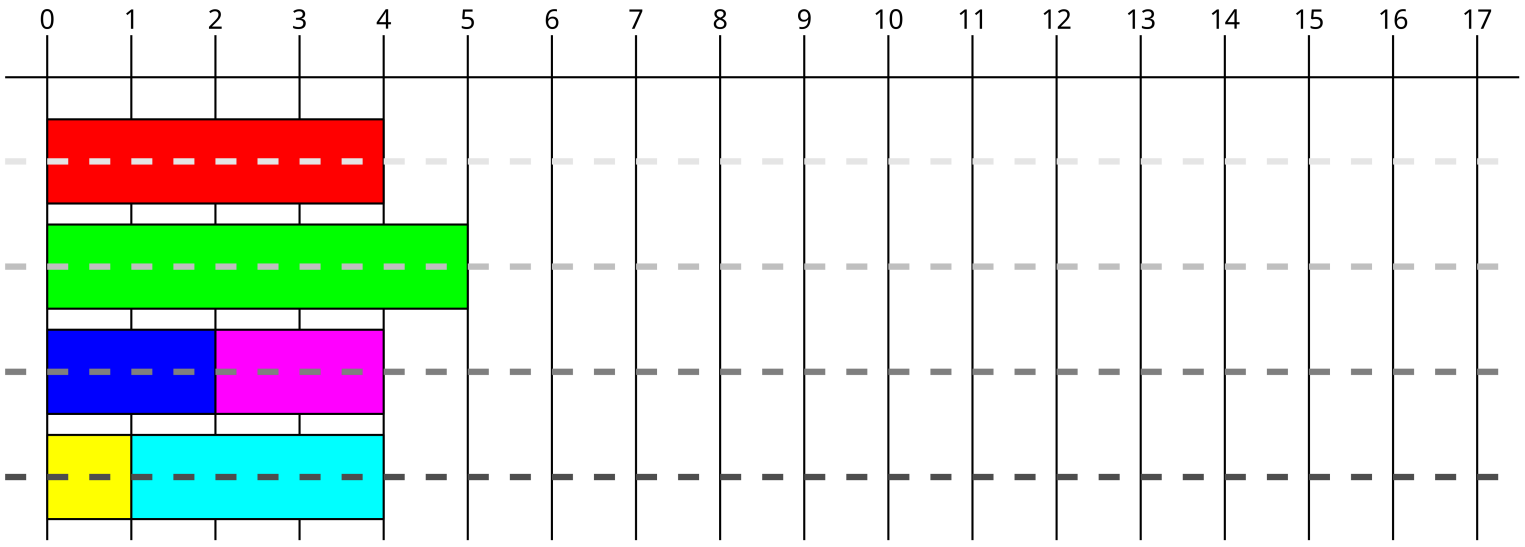
Parámetros de planificación adicionales (5)



2 procesadores

¿Hay otras restricciones que eleven el tiempo de planificación?

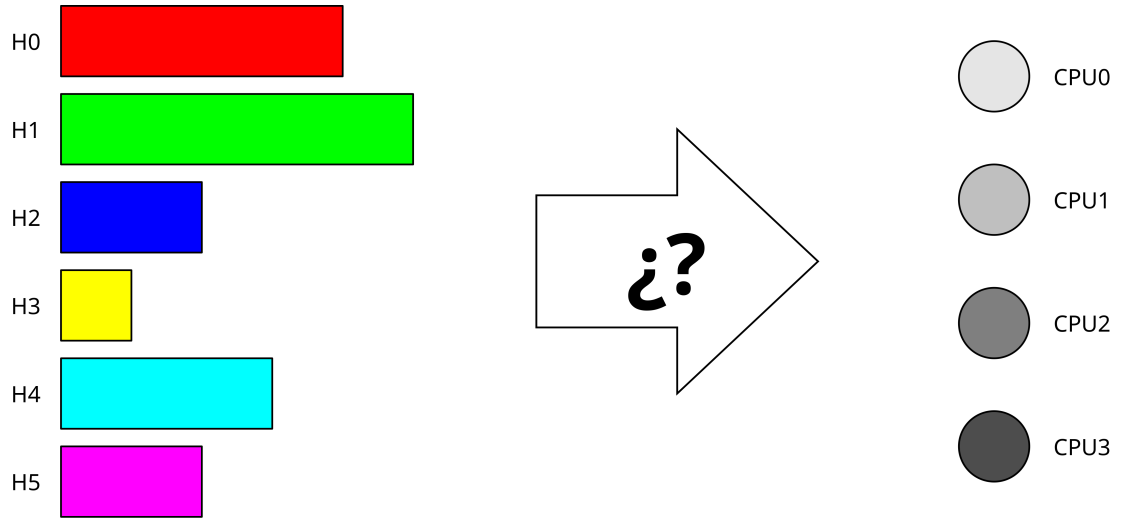
Parámetros de planificación adicionales (6)



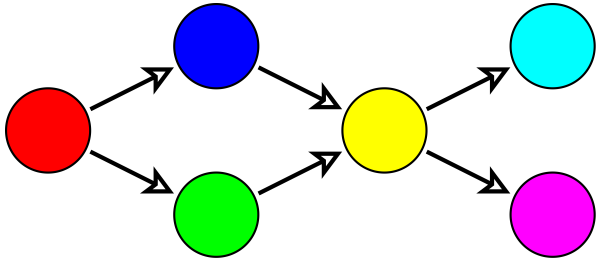
4 procesadores (máximo)

- Ventajas:
  - Menor tiempo de estancia posible (esperado)
- Inconvenientes:
  - Debido a las secciones críticas los tiempos de estancia pueden ser mayores de lo esperado.

Parámetros de planificación adicionales (7)

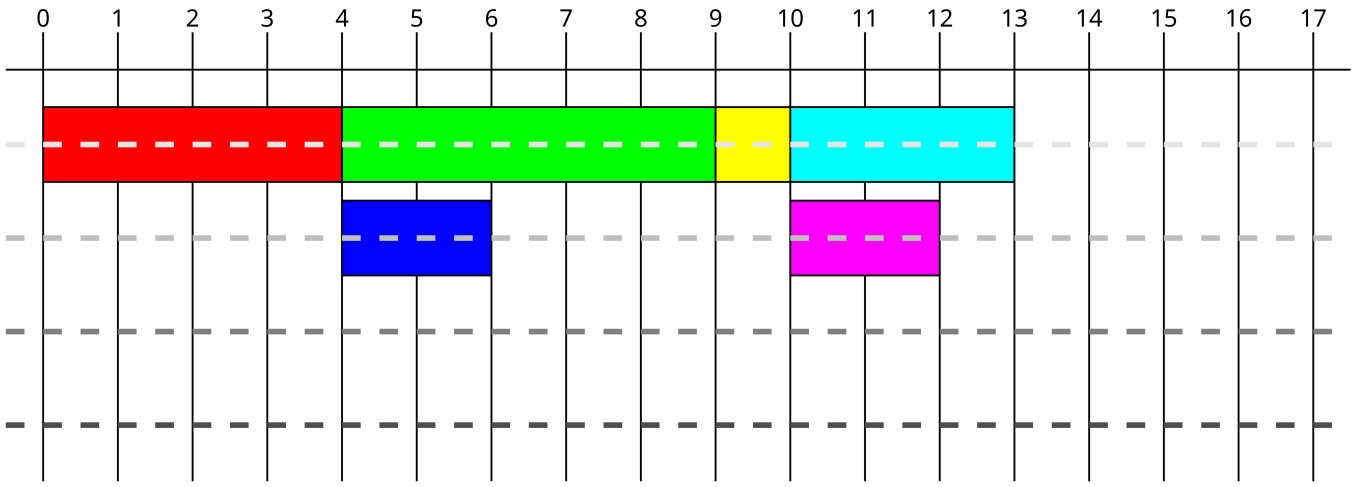


Restricciones de precedencia: ciertas hebras deben finalizarse antes de que otras puedan iniciar su ejecución.



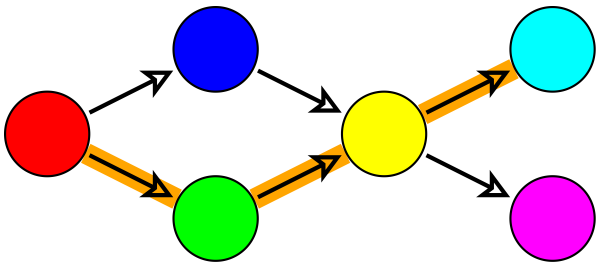
Las flechas indican las relaciones de precedencia.

Parámetros de planificación adicionales (8)

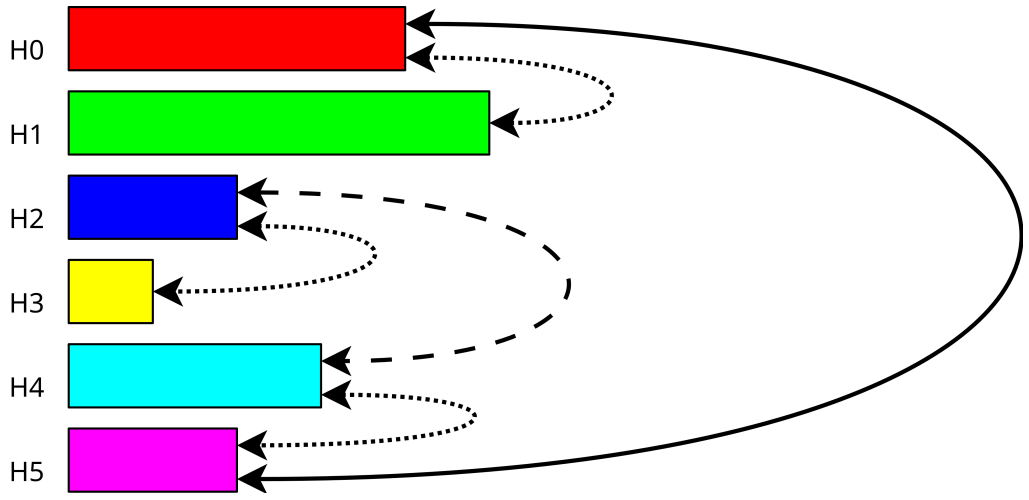


Camino crítico: secuencia de actividades de mayor longitud que determina el **mínimo tiempo** de ejecución posible.

- Del gráfico se puede deducir que como máximo podremos emplear dos procesadores.
- Asignar procesadores de acuerdo a las relaciones de precedencia.



Parámetros de planificación adicionales (9)



Coste de comunicación entre hebras:

- La comunicación entre hebras en diferentes procesadores tiene que hacerse a través de memoria principal.
- La comunicación entre hebras en el mismo procesador puede hacerse a través de registros o caché.

Conclusión: Lo que se gana mediante el paralelismo real puede perderse debido al incremento del coste de las comunicaciones.

Planificación de procesos multihebra

- Planificación (dinámica) **anónima**:
  - Asignar una hebra al primer procesador disponible $\Longrightarrow$ las hebras pueden ejecurse sobre diferentes procesadores.
- Planificación **dedicada**:
  - Cada hebra de un proceso es asignada a un procesador específico.
- Planificación **adaptativa**:
  - El planificador coloca las hebras, estática o dinámicamente, de acuerdo a la carga de los procesadores.
- Planificación basada en **usuarios/programadores**:
  - El usuario puede fijar hebras a procesadores.

Planificación de procesos

- Procesos monohebra:
  - La planificación justa entre procesos es sencilla.
  - Los procesos que comparten espacio de direcciones tienen preferencia para evitar el intercambio.
- Procesos multihebra:
  - Planificación justa entre procesos o hebras.
  - Intercambiar procesos completos para mejorar el paralelismo.

Observación: Algunos sistemas operativos antiguos (VAX VMS) utilizaban otra unidad de planificación, la sesión, que es relativa al usuario y permite establecer planificación justa en base a dichas sesiones.

Planificación en pandilla (*"gang scheduling"*)

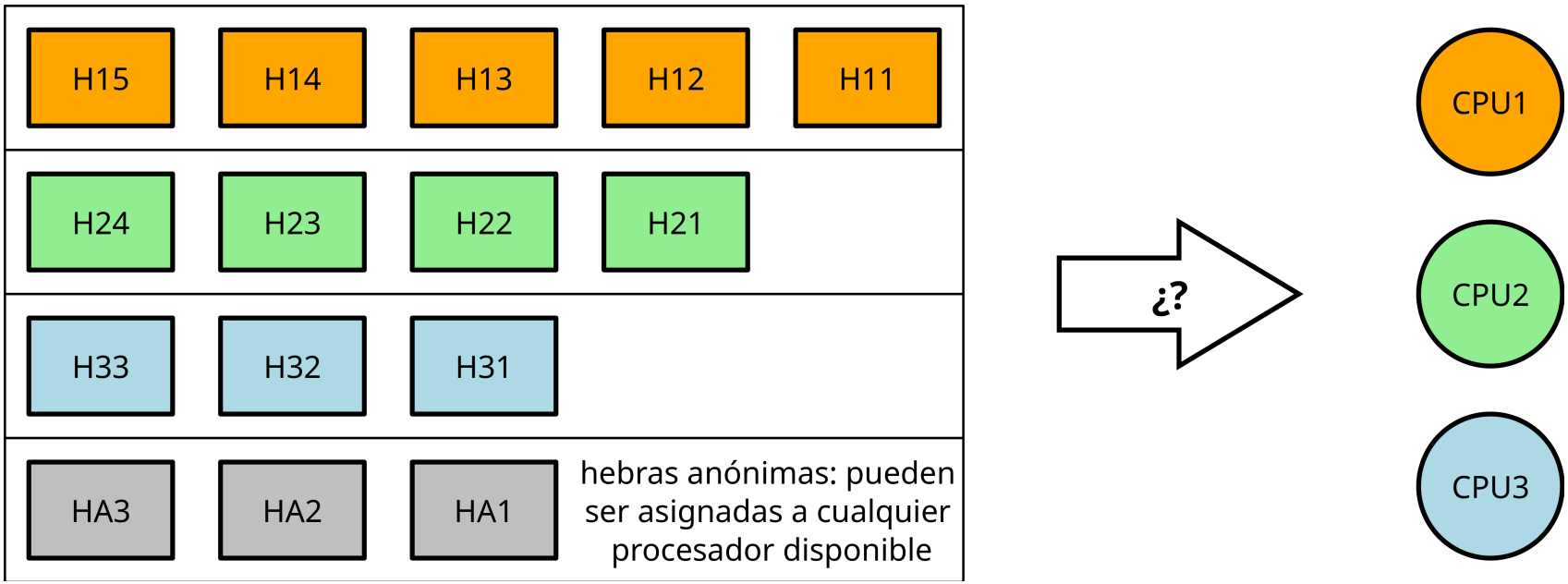
- Planificación simultánea de las hebras de un proceso.
- Útil para aplicaciones en las que el rendimiento se degrada de forma severa cuando alguna de sus partes no se está ejecutando.
- Las hebras a menudo deben sincronizarse entre ellas, ej: una nueva iteración en la resolución de una ecuación diferencial (a través de una barrera).

Planificación SMP centralizada/descentralizada

- La planificación SMP **centralizada** puede utilizar **una** cola **global** de hebras preparadas:
  - \+ Es fácil implementar una política consistente.
  - \- No es escalable para un gran número de procesadores.

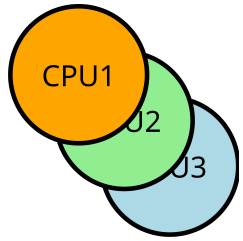
- La planificación SMP **descentralizada** utiliza una cola **global** compartida más una cola **local** de hebras preparadas por procesador:
  - \+ Menos conflictos en el acceso.
  - \- Aparece el problema del balanceo de carga.
  - \- ¿Cómo rellenar cada una de las colas?

Asignación de hebras dedicadas y anónimas



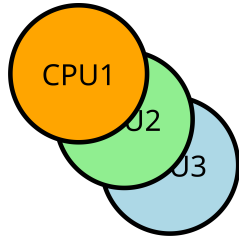
¿Qué estructura de datos sería más eficiente para representar las colas de hebras preparadas?

Cola de preparadas (orden natual FCFS)



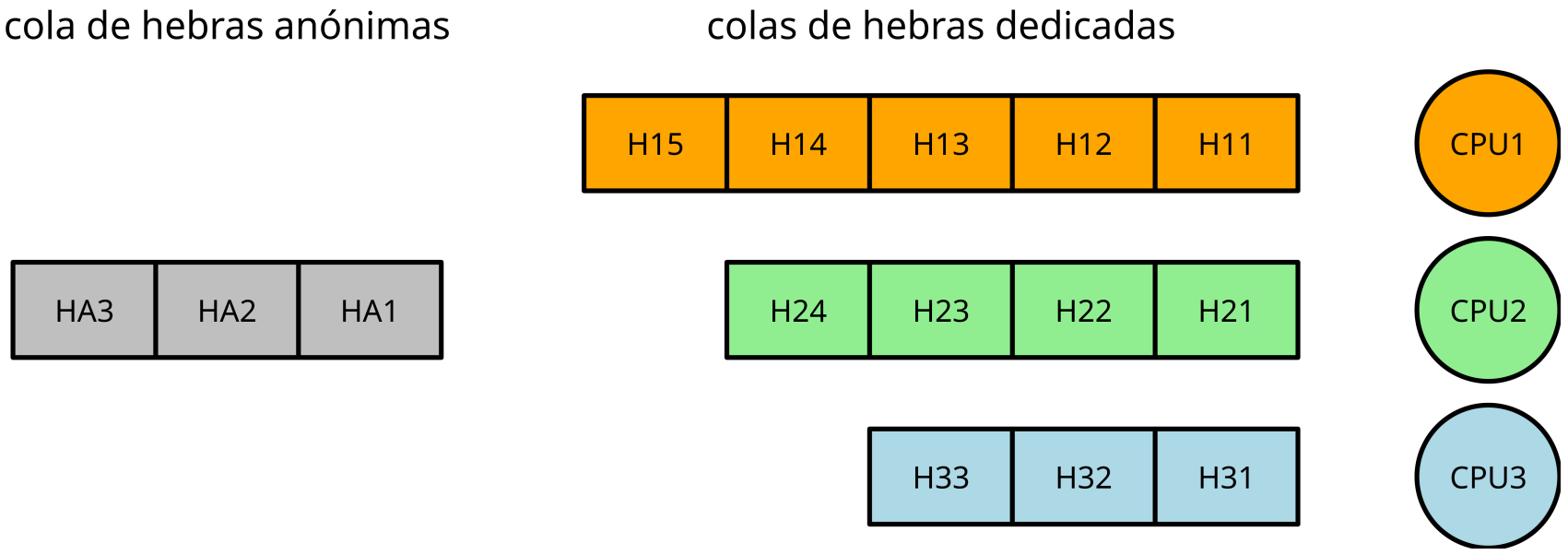
- Política: escoger la **primera** hebra.
- Inconvenientes:
  - No asignar el primero de la cola provocará sobrecarga.
  - Se puede asignar una hebra anónima a un procesador incluso aunque exista otra hebra preparada y dedicada al mismo.

Cola de preparadas (orden natural FCFS)



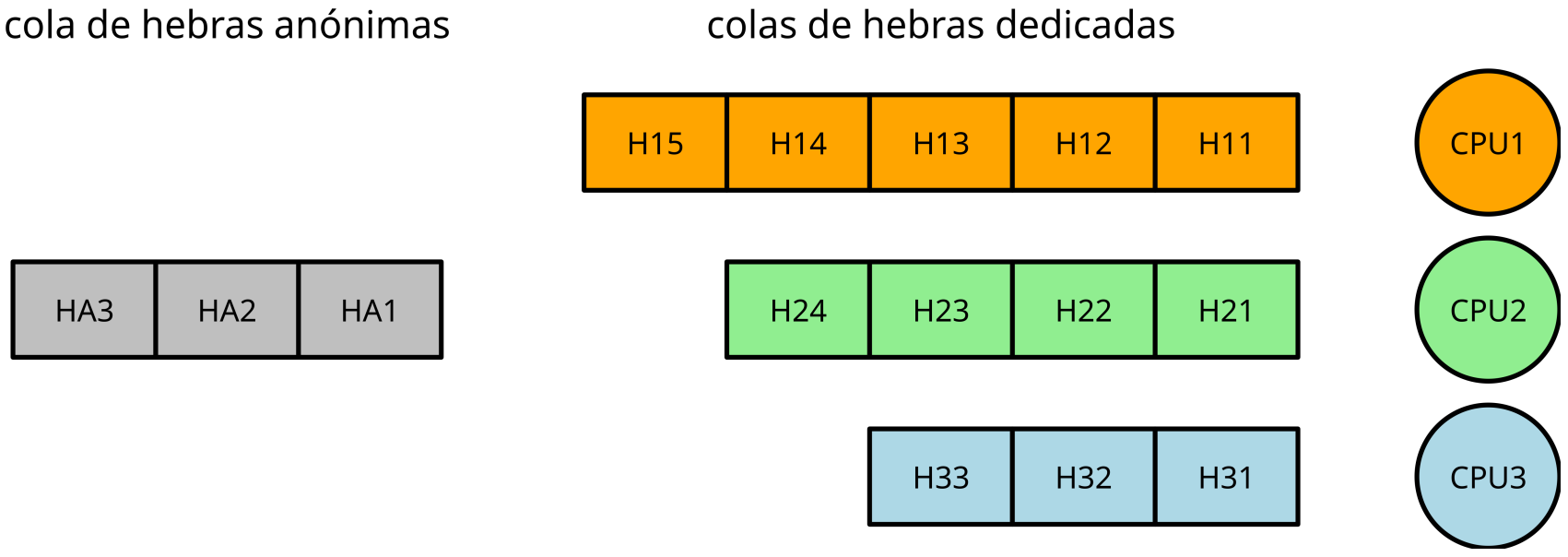
- Política: escoger la hebra que **mejor** se ajuste.
- Inconvenientes:
  - Puede ser necesario recorrer la cola entera.
  - La eficiencia de este algoritmo sería O(n).

Diferentes colas preparadas para anónimas y dedicadas



- Política: preferir hebras **dedicadas**.
- Primero buscar en la cola dedicada apropiada y si está vacía buscar en la cola anónima.

Diferentes colas preparadas para anónimas y dedicadas



- Política: preferir hebras de **mayor prioridad**.
- Comparar las cabeceras de la cola dedicada apropiada y la cola anónima y escoger aquella de mayor prioridad.

Introducción a la planificación de tiempo real (1)

- Disciplina de creciente importancia.
- Ejemplos: control de experimentos, robótica, control de tráfico aéreo, vehículos autónomos, telecomunicaciones, cirugía remota, multimedia,...
- El único objetivo del sistema es **cumplir plazos**.
- La **corrección** del sistema no sólo depende del **resultado** de un cálculo sino también de **cuándo** se obtiene.
- Se asocia un **plazo** de inicio y fin (o duración) a cada tarea.

Introducción a la planificación de tiempo real (2)

- Clasificación de *sistemas* de tiempo real:
  - Tiempo real **estricto**: han de cumplirse todos los plazos.
  - Tiempo real **flexible**: el incumplimiento ocasional de algún plazo, aunque inconveniente, puede ser aceptable.

- Clasificación de *tareas* de tiempo real:
  - Tareas **aperiódicas**: tiene un plazo de inicio o de fin.
  - Tareas **periódicas**: han de ejecutarse de forma periódica.

Requisitos (1)

- **Determinismo**: Realiza operaciones en instantes de tiempo fijos o dentro de intervalos de tiempo predeterminados.
  - Medida: retardo desde la llegada de una interrupción hasta que comienza a ser atendida.
  - Valores típicos: <1ms (SO no de tiempo real: 10-100ms).
- **Reactividad**: si el determinismo mide cuánto tarda el sistema en reconocer una interrupción, la reactividad se preocupa de cuánto tarda en darle servicio.
  - Duración: tiempo necesario para ejecutar la rutina de servicio de la interrupción más posibles anidaciones de interrupciones.
- **Tiempo de respuesta** = determinismo + reactividad.

Requisitos (2)

- **Control del usuario**: mucho mayor que en un SO normal.
  - Fino control de planificación: prioridad, tipo, intercambio,...
- **Fiabilidad**: muy importante/vital (no sirve reiniciar :)
- **Fallo suave**:
  - relacionado con la fiabilidad.
  - en caso de fallo que este sea lo menos perjudicial posible.
  - ej: no abortar procesos con fallos.
  - ej: incumplir los plazos de las tareas menos críticas.

Características típicas (1)

- ¿Cambio de proceso rápido?
- ¿Tamaño pequeño?
- ¿Capacidad para responder rápidamente a eventos?
- ¿Multitarea con mecanismos de comunicación entre procesos (semáforos, señales,...)?
- ¿Almacenamiento rápido de información en ficheros secuenciales?
- ¿Planificación expulsiva basada en prioridades?
- ¿Minimización de los intervalos de deshabilitación de interrupciones?
- ¿Primitivas para bloquear y desbloquear tareas muy precisas?
- ¿Alarmas y temporizadores muy precisos?

Características típicas (2)

- ¿Cambio de proceso rápido? En cualquier SO debería ser rápido.
- ¿Tamaño pequeño? Cualquier SO debería ser pequeño.
- ¿Capacidad para responder rápidamente a eventos? Retrasar el manejo de interrupciones nunca es aceptable.
- ¿Multitarea con mecanismos de comunicación entre procesos (semáforos, señales,...)? Útil para todos los sistemas con tareas que interaccionen.
- ¿Almacenamiento rápido de información en ficheros secuenciales? No.
- ¿Planificación expulsiva basada en prioridades? Si, incluso del SO.
- ¿Minimización de los intervalos de deshabilitación de interrupciones? Si.
- ¿Primitivas para bloquear y desbloquear tareas muy precisas? Si.
- ¿Alarmas y temporizadores muy precisos? Si.

Clasificación (1)

- Los algoritmos de planificación de tiempo real se caracterizan en función de varios factores:
  - ¿**Cuándo** se realiza el análisis de planificabilidad?
  - ¿**Cómo** se realiza el análisis, estática o dinámicamente?
  - ¿El resultado del análisis es un **plan** o no?

Clasificación (2)

Tipos de algoritmos:

- **Estáticos**:
  - **Dirigidos por tabla**: análisis estático de planificabilidad que da lugar a un plan de tiempos de inicio de ejecución.
  - **Dirigidos por prioridad**: análisis estático de planificabilidad que permite asignar prioridades a las tareas y que debe utilizarse junto a un algoritmo de planificación expulsivo basado en prioridades.
- **Dinámicos**:
  - **Basados en un plan**: una nueva tarea es aceptada sólo si es posible crear un plan de ejecución que satisfaga todas las restricciones temporales.
  - **De mejor esfuerzo**: intenta cumplir todos los plazos y en caso de no poder intenta provocar un "fallo suave".

Planificación estática dirigida por tabla

- Suele aplicarse a tareas **periódicas**.
- Datos de entrada: periodo de llegada, tiempo de ejecución, plazo de finalización y prioridad relativa.
- El planificador ejecuta un **plan** previamente diseñado que permite **cumplir los plazos** de todas las tareas.
- Enfoque predecible pero poco flexible dado que cualquier pequeño cambio obliga a rehacer el plan.
- Ejemplo: **planificación de tasa monótona** (*"Rate Monotonic Scheduling" - RMS*).

Planificación estática con expulsión dirigida por prioridad

- Utiliza mecanismos de planificación **expulsivos** dirigidos por **prioridades** como la mayoría de los sistemas operativos no de tiempo real.
- En sistemas no de tiempo real la prioridad suele asignarse en función del consumo de procesador o E/S.
- En sistemas de tiempo real la prioridad se asigna en función de las restricciones de tiempo de cada tarea: periodos de ejecución y tiempos de servicio.
- Ejemplo: El **plazo más cercano primero** (*"Earliest Deadline First" - EDF*).

Planificación dinámica basada en un plan

- Es necesario **modificar el plan** cada vez que llega una **nueva tarea**.
- La **nueva** tarea es **aceptada** sólo si se puede crear un nuevo **plan** que siga **cumpliendo** las **restricciones** temporales todas las tareas.

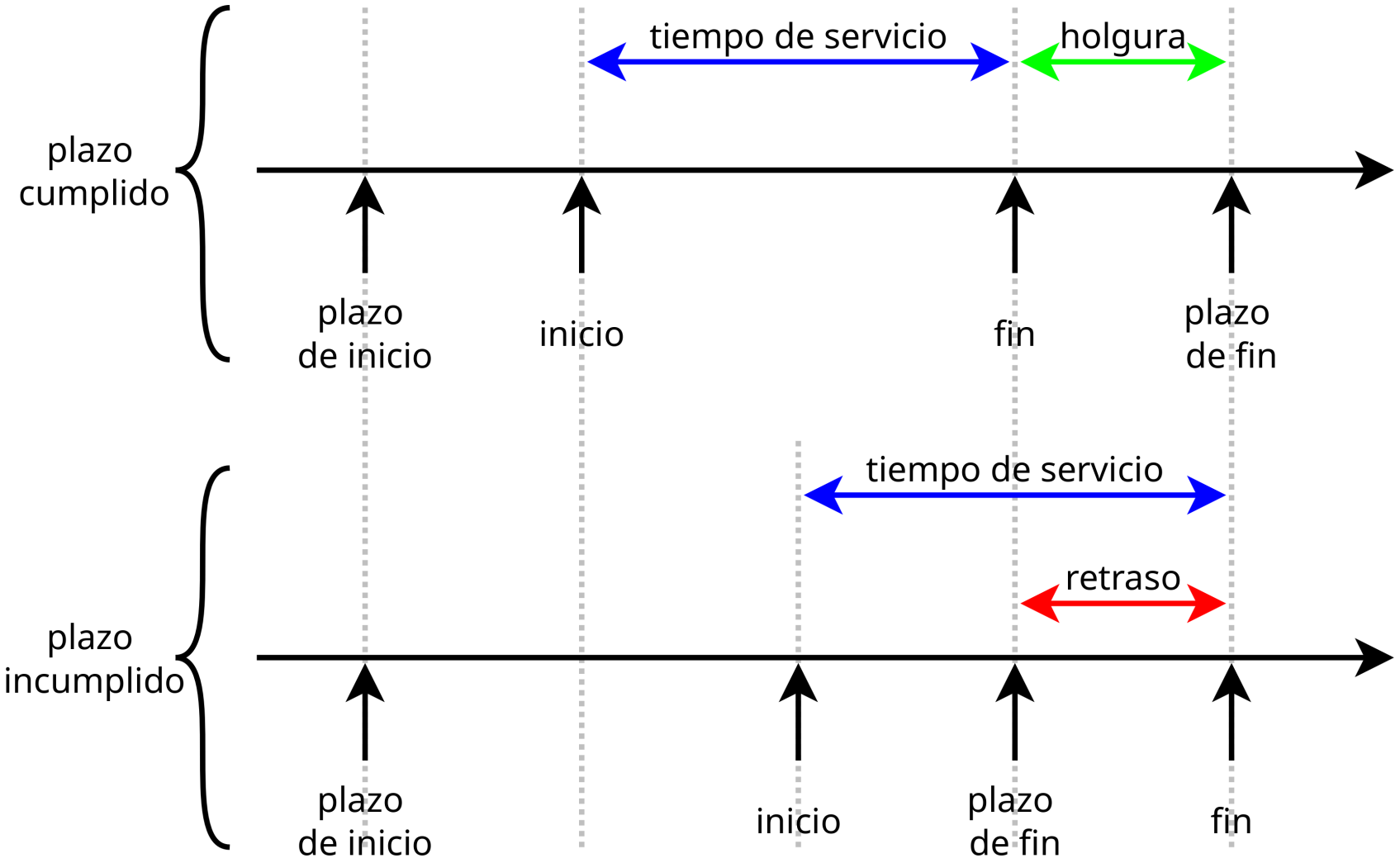
Planificación dinámica de mejor esfuerzo

- Enfoque, poco ambicioso pero práctico, muy utilizado en sistemas comerciales.
- Cuando llega una nueva tarea al sistema se le asigna una **prioridad** basada en sus características.
- Suele emplearse algún tipo de planificación basada en **plazos** (EDF).
- Como las **tareas no** son **periódicas** no puede realizarse un análisis de planificabilidad.
- Ventaja: fácil de implementar.
- Inconveniente: no sabremos si una restricción temporal se cumple hasta que venza su plazo o la tarea se complete.

Planificación por plazos

- El objetivo de gran parte de sistemas es **arrancar** las tareas de tiempo real tan **rápido** como sea posible.
- El verdadero objetivo no debería ser la velocidad sino completar, o comenzar, las tareas en los **momentos adecuados**.
- Las **prioridades** son un **burda aproximación** que no garantiza completar, o iniciar, una tarea en el momento adecuado.
- Las propuestas para implementar enfoques más apropiados se basan en tener **información adicional** sobre las tareas: tiempo de activación, plazo de comienzo, plazo de conclusión, tiempo de proceso, recursos necesarios, prioridad y estructura de hebras.
- Se ha demostrado que, de entre las políticas expulsivas, planificar mediante EDF minimiza la cantidad de tareas que **incumplen** sus plazos.

Eventos


plazo
cumplido
tiempo de servicio
holgura
plazo
de inicio
inicio
fin
plazo
de fin
plazo
incumplido
tiempo de servicio
retraso
plazo
de inicio
inicio
plazo
de fin
fin

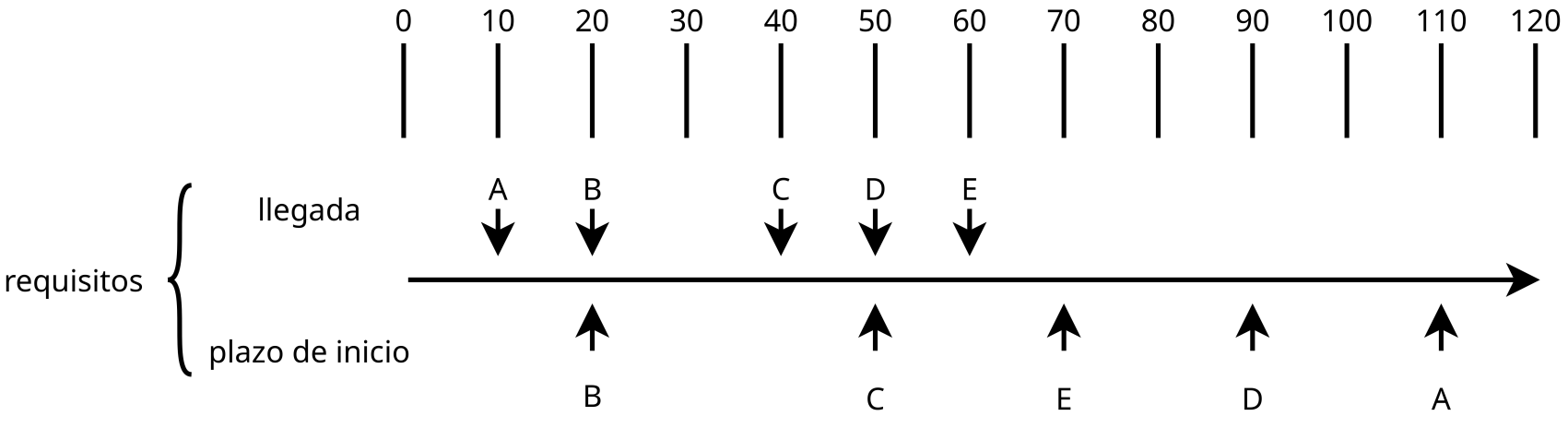
- Ordena las tareas en orden no decreciente según su plazo de fin:

  $d_1 \leq d_2 \leq d_3 \leq \cdots \leq d_n$

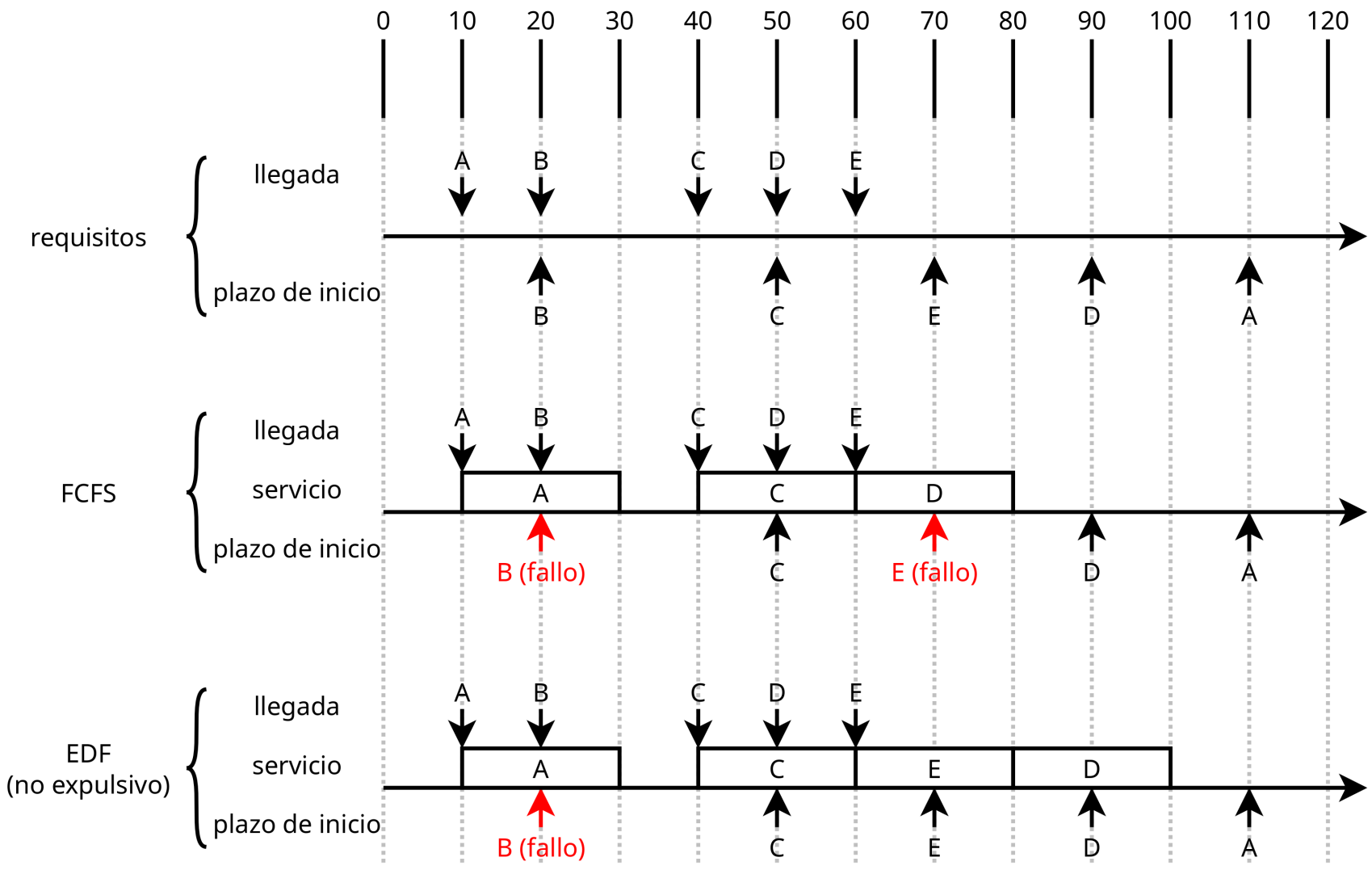
- Esta política es óptima con respecto al **máximo retraso** y a la **máxima tardanza**.
- La tardanza es el máximo retraso de entre todas las tareas.

Ejemplo de planificación aperiódica (1)

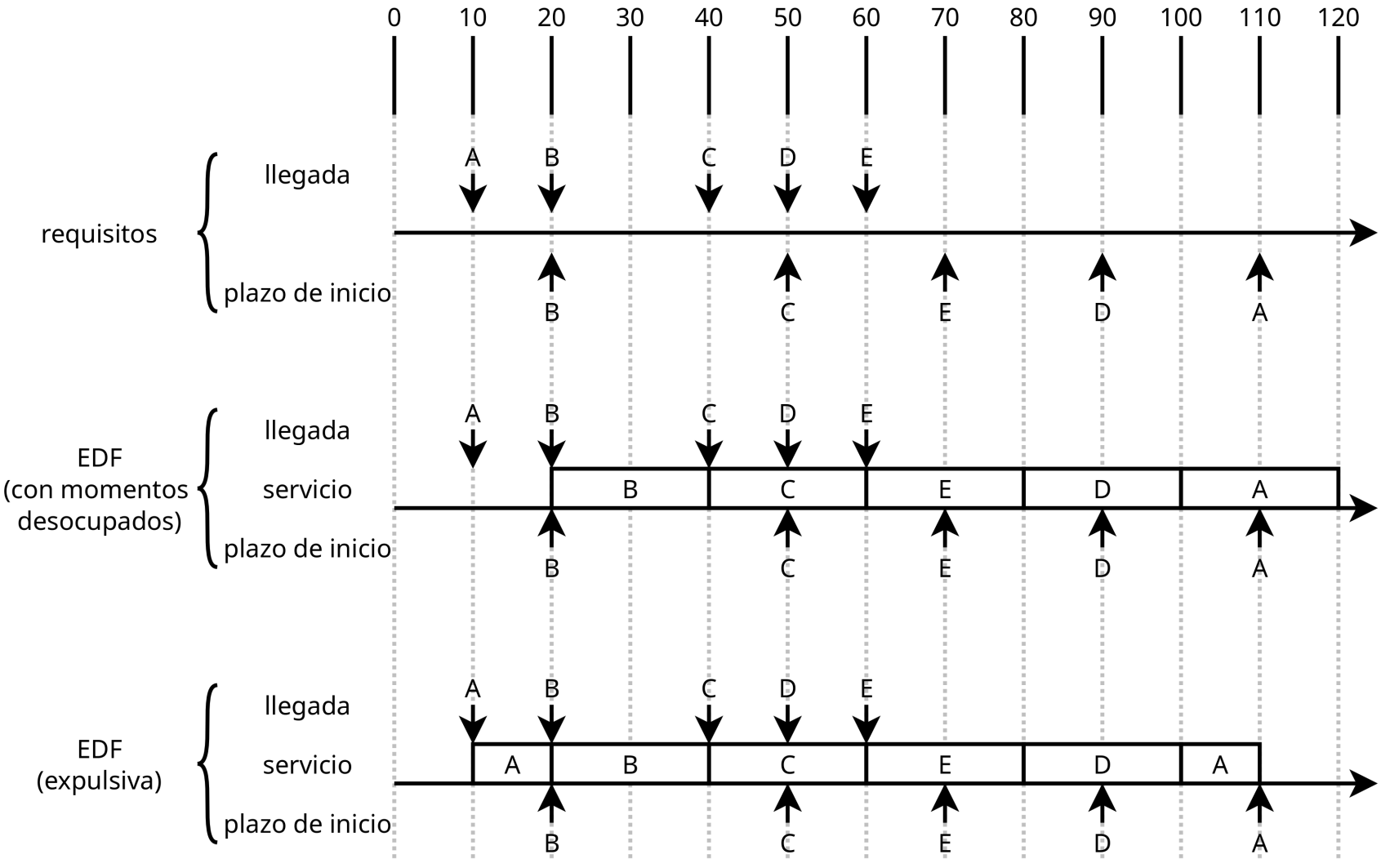
hebra	llegada	servicio	plazo de inicio
A	10	20	110
B	20	20	**20 (!!!)**
C	40	20	50
D	50	20	90
E	60	20	70



Ejemplo de planificación aperiódica (2)



Ejemplo de planificación aperiódica (3)



Planificación de tasa monótona (RMS)

- Uno de los métodos más empleados para tareas **periódicas**.
- Asigna **prioridades** a los procesos en función de su periodo: a menor periodo → mayor prioridad.
- Las tareas se ejecutan hasta concluir (mejora: expulsivo).
- $U = \frac{C}{P}$ U = utilización, C = cómputo, P = periodo
- Cómo garantizar el cumplimiento de los plazos para n tareas:
  $U = \sum_i \frac{C_i}{P_i} \leq 1$ límite **teórico** de RMS.
  $U = \sum_i \frac{C_i}{P_i} \leq n(2^{1/n} - 1)$ límite **práctico** de RMS.
- Diferencia de prestaciones practica con EDF pequeña, consiguen tasas de utilización de más del 90 %.
- Permite mezclar tareas de tiempo real estricto y flexible.
- RMS es más estable que EDF: podemos garantizar el cumplimiento de los plazos de las tareas más prioritarias.

Ejemplo de planificación de actividades periódicas

hebra	periodo	servicio	$U = C/P$
H_1	4	1	$1/4 = 0{,}25$
H_2	6	2	$2/6 = 0{,}33$
H_3	8	2	$2/8 = 0{,}25$