



SISTEMAS OPERATIVOS

Semestre 2010-1

4. Planificación de procesos



Objetivo

El alumno explicará las diferentes técnicas de asignación del procesador a los procesos.



Planificación

- El Objetivo de la planificación de procesos y procesos ligeros es el reparto del tiempo del procesador entre los procesos que pueden ejecutar





Planificación

En un sistema multiprogramado, múltiples procesos son mantenidos en memoria principal. *Multiprogramación* surgió con la idea de tener algún proceso ejecutándose en todo momento con la finalidad de maximizar el uso del CPU.



Planificación

Tiempo Compartido fue concebido con la idea de conmutar o cambiar continuamente el CPU entre procesos de forma tal que los usuarios puedan interactuar con sus programas mientras están corriendo



Planificación

- Con multiprogramación, un proceso ejecuta hasta esperar por algún evento. En un sistema simple sin multiprogramación el CPU estaría ocioso



Planificación

Cuando un proceso entra al sistema es colocado en una cola de trabajos. Una vez que el proceso se encuentra en memoria principal y está listo para ejecutar, este es colocado en la cola de procesos listos (*ready*).



Planificación

- Cuando al proceso se le asigna el CPU, ejecuta por un tiempo y eventualmente el proceso terminará, o será interrumpido o esperará por la ocurrencia de algún evento.



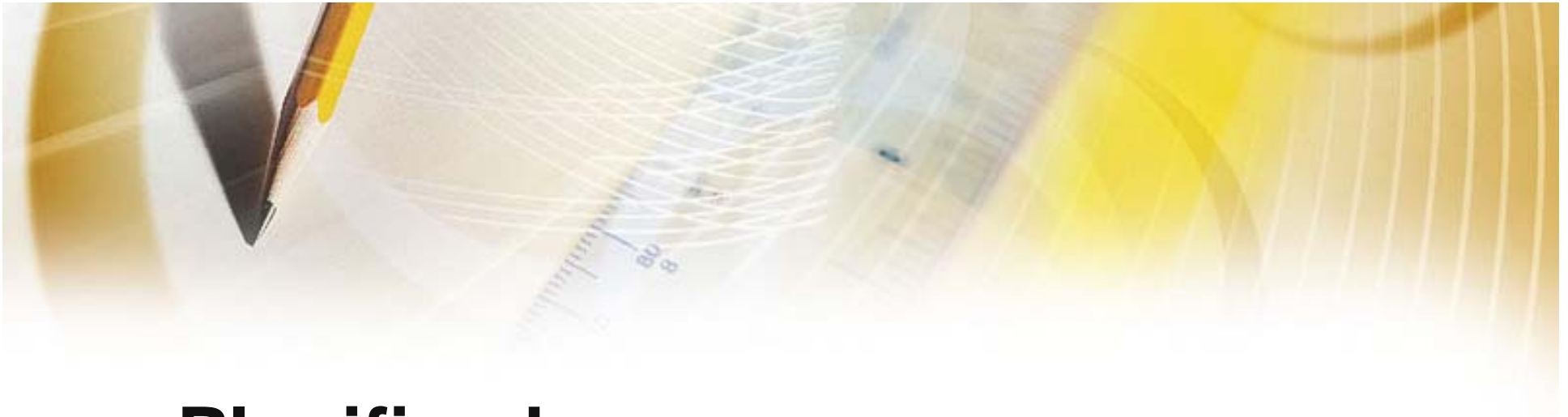
Planificación

- Con frecuencia nos encontraremos con situaciones en las que dos o más procesos son ejecutables desde el punto de vista lógico. En estos casos el sistema de operación debe decidir cuál de ellos debe ejecutarse primero.



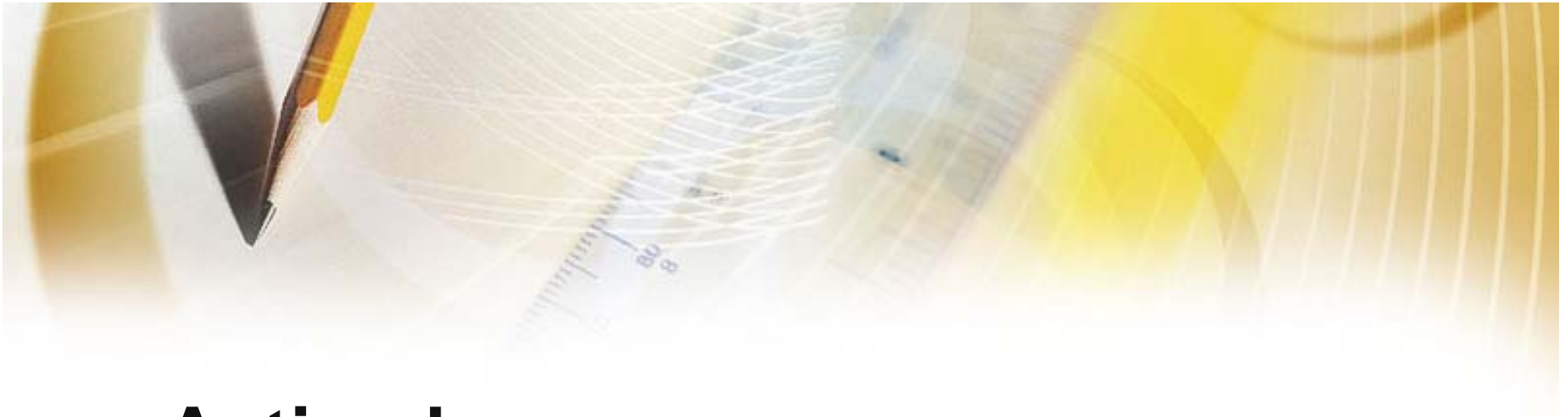
Planificación

- La parte del sistema de operación que lleva a cabo esta decisión se llama *Planificador* y el algoritmo que se utiliza se conoce como *Algoritmo de Planificación*.



Planificador

- El planificador es el módulo del sistema operativo que realiza la función de seleccionar el proceso en estado de listo que pasa a estado de ejecución



Activador

- Mientras que el activador es el módulo que pone en ejecución el proceso planificado



Tipos de planificación

- La **planificación a largo plazo** tiene por objeto añadir nuevos procesos al sistema, tomándolos de la lista de espera. Estos procesos son de tipo “lotes”, en los que no importa el instante preciso en el que se ejecuten, siempre que cumplan los límites de espera.



Tipos de planificación

- La **planificación a mediano plazo** trata la suspensión de procesos. Es la que decide que procesos pasan a suspendido y cuales dejan de estar suspendidos. Añade o elimina procesos de memoria principal modificando por tanto el grado de multiprogramación.



Tipos de planificación

- La **planificación a corto plazo** se encarga de seleccionar el proceso en estado de listo que pasa a estado de ejecución. Es por tanto la que asigna el procesador.



Tipos de planificación

- También es importante la planificación de entrada-salida. Esta planificación decide el orden en que se ejecutan las operaciones de entrada-salida que están encoladas para cada periférico.



Expulsión

- La planificación puede ser con expulsión o sin ella. En un sistema sin expulsión un proceso conserva el procesador mientras lo desee, es decir, mientras no solicite del sistema operativo un servicio que lo bloquee.



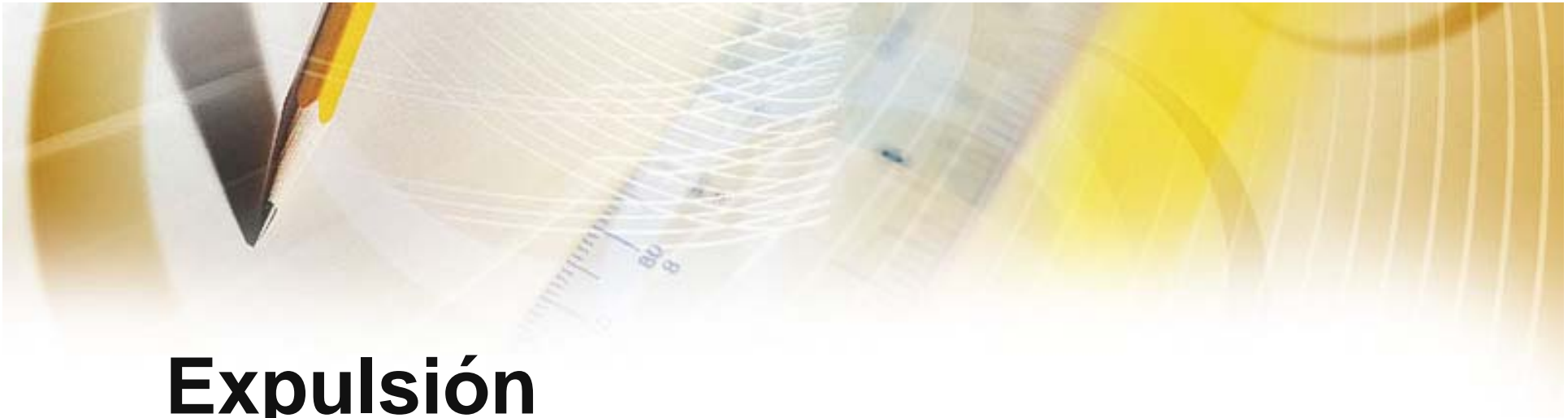
Expulsión

- Esta solución (sin expulsión) minimiza el tiempo que gasta el sistema operativo en planificar y activar procesos, pero tiene como inconveniente que un proceso puede monopolizar el procesador.



Expulsión

- En los sistemas con expulsión, el sistema operativo puede quitar a un proceso del estado de ejecución aunque este no lo solicite. Esta solución permite controlar el tiempo que está en ejecución un proceso, pero requiere que el sistema operativo
.....



Expulsión

- entre de forma sistemática a ejecutar para así poder comprobar si el proceso ha superado un límite de tiempo de ejecución.



Expulsión

- Como sabemos las interrupciones sistemáticas del reloj garantizan que el hardware soportado por el SO entre a ejecutar cada pocos milisegundos, pudiendo determinar en estos instantes si ha de producirse un cambio de proceso o no.



Objetivos de la planificación

El objetivo de la planificación es optimizar el comportamiento del sistema. Ahora bien el comportamiento de un sistema informático es muy complejo, por tanto el objetivo se deberá centrar en la faceta del comportamiento en el que se esté interesado. Los objetivos que se suelen perseguir son:



Objetivos de la planificación

- - Reparto equitativo del procesador
- - Eficiencia (optimizar el uso del procesador)
- - Menor tiempo de respuesta en uso interactivo.
- - Menor tiempo de espera.
- - Mayor número de trabajos por unidad de tiempo.
- - Cumplir los plazos de ejecución de un sistema en tiempo real.



Objetivos de la planificación

La mayoría de estos objetivos son incompatibles entre sí, por lo que hay que centrar la atención en aquel que sea de mayor interés. Por ejemplo una planificación que realice un reparto equitativo del procesador no conseguirá optimizar el uso del mismo.



Algoritmos de planificación

- Ciclica o Round-Robin
- El algoritmo cíclico está diseñado para hacer un reparto equitativo del tiempo del procesador, por lo que esta especialmente destinado a los sistemas de tiempo compartido. El algoritmo se basa en el concepto de rodaja (slot) de tiempo.



Algoritmos de planificación

- Ciclica o Round-Robin
- Los procesos están organizados en forma de cola circular, eligiéndose para su ejecución el proceso de cabecera de la cola. Un proceso permanecerá en ejecución hasta que ocurra una de las dos condiciones siguientes:



Algoritmos de planificación

- Ciclica o Round-Robin
 - - El proceso pasa a estado de bloqueado, porque solicita un servicio del sistema operativo.
 - - El proceso conduce su rodaja de tiempo, es decir, lleva ejecutando el tiempo estipulado de rodaja.



Algoritmos de planificación

- Ciclica o Round-Robin
- Un proceso que ha consumido su rodaja de tiempo es expulsado y pasa a ocupar el último lugar en la cola. De esta forma se consigue que todos los procesos pasan a ejecutar, repartiendo el tiempo del procesador de forma homogénea entre ellos.



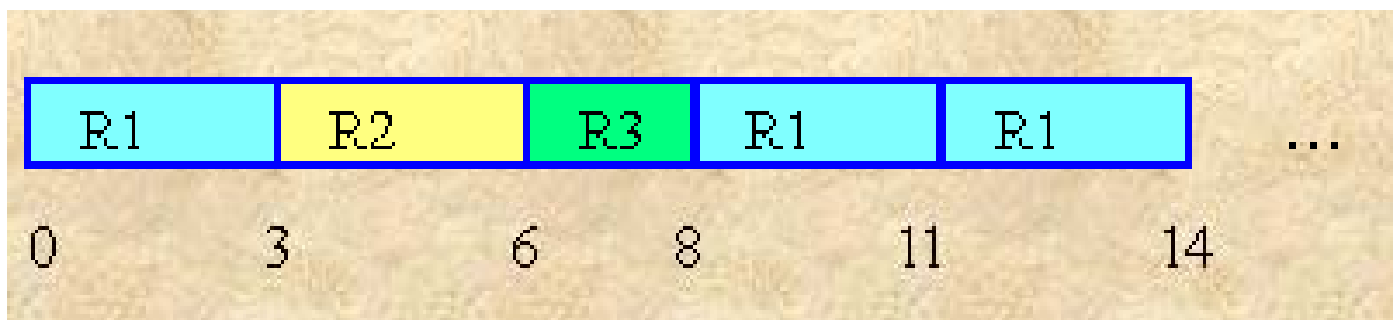
Round Robin

- Este algoritmo presupone la existencia de un *reloj en el sistema*. Un reloj es un dispositivo que genera periódicamente interrupciones. Esto es muy importante, pues garantiza que el sistema operativo (en concreto la rutina de servicio de interrupción del reloj) toma el mando del CPU periódicamente. El *quantum* de un proceso equivale a un número fijo de pulsos o ciclos de reloj. Al ocurrir una interrupción de reloj que coincide con la agotación del *quantum* se llama al dispatcher

Round Robin

- Ejemplo Ocupación del CPU

RAFAGA	TIEMPO LLEGADA	REQUERIMIENTOS DE CPU(ms)
R1	0	16
R2	1	3
R3	2	2





Round Robin

- Simulador
- http://wwdi.ujaen.es/~lina/TemasSO/PLANIFICACIONDEPROCESOS/PlanificadorProcesos/alg_planif_RR.html




Planificación de Plazo Fijo

- En la planificación de plazo fijo se programan ciertos trabajos para terminarse en un tiempo específico o plazo fijo. Estas tareas pueden tener un gran valor si se entregan a tiempo, y carecer de él si se entregan después del plazo. Esta planificación es compleja por varios motivos:



Planificación de Plazo Fijo

- El usuario debe informar por adelantado de las necesidades precisas de recursos del proceso. Semejante información rara vez está disponible.
- El sistema debe ejecutar el proceso en un plazo fijo sin degradar demasiado el servicio a los otros usuarios y debe planificar cuidadosamente sus necesidades de recursos dentro del plazo.



Planificación de Plazo Fijo

- Esto puede ser difícil por la llegada de nuevos procesos que impongan demandas imprevistas al sistema.
- Si hay muchas tareas a plazo fijo activas al mismo tiempo, la planificación puede ser tan compleja que se necesiten métodos de optimización avanzados para cumplir los plazos.



Planificación de Plazo Fijo

- La administración intensiva de recursos requerida por la planificación de plazo fijo puede producir un gasto extra substancial.



Planificación FIFO

- Primero en entrar primero en salir (FIFO, *First In First Out*)
- Cuando se tiene que elegir a qué proceso asignar el CPU se escoge al que llevara más tiempo listo. El proceso se mantiene en el CPU hasta que se bloquea voluntariamente.



Planificación FIFO

- La ventaja de este algoritmo es su fácil implementación, sin embargo, no es válido para entornos interactivos_ya que un proceso de mucho cálculo de CPU hace aumentar el tiempo de espera de los demás procesos .



Planificación FIFO

- Para implementar el algoritmo sólo se necesita mantener una cola con los procesos listos ordenada por tiempo de llegada. Cuando un proceso pasa de bloqueado a listo se sitúa el último de la cola.



Planificación FIFO

- Ejemplo
- Tenemos 4 procesos:
- P2, P4, P7, P8
- En un instante dado P7 está ejecutando en el procesador

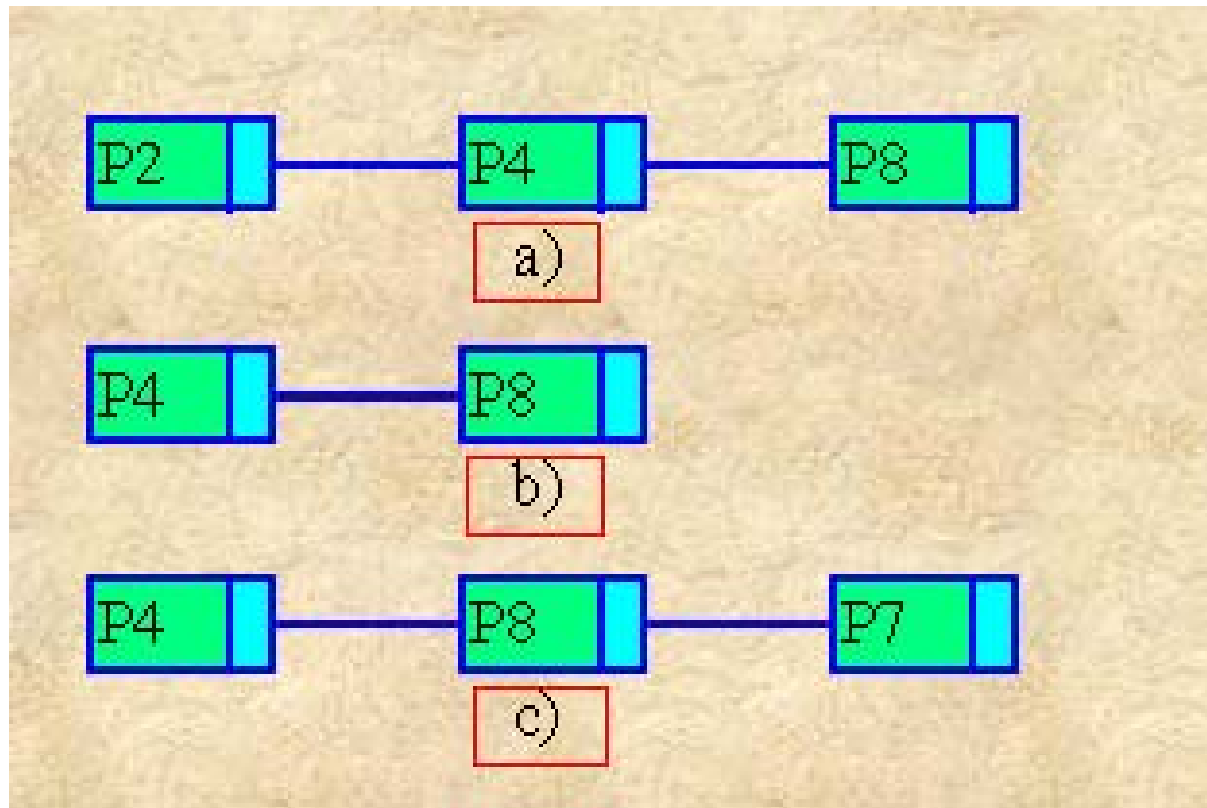


Planificación FIFO

- a) el proceso P7 ocupa el CPU, los procesos P2, P4 y P8 se mantienen en la lista de preparados.
- **b)** P7 se bloquea (ya sea al realizar una E/S, una operación WAIT u otra causa) y P2 pasa a ocupar el CPU.
- **c)** ocurre un evento (finalización de la operación de E/S, etc, ...) que desbloquea a P7, esto lo vuelve listo, pasando al final de la cola de procesos listos.

Planificación FIFO

-





Planificación FIFO

- Algunas de las características de este algoritmo es que es *apropiativo* y justo en el sentido formal, aunque injusto en el sentido de que: los trabajos largos hacen esperar a los cortos y los trabajos sin importancia hacen esperar a los importantes. Por otro lado es predecible pero no garantiza buenos tiempos de respuesta y por ello se emplea como esquema secundario.



Planificación FIFO

- Simulador FIFO
- http://wwdi.ujaen.es/~lina/TemasSO/PLANIFICACIONDEPROCESOS/PlanificadorProcesos/alg_planif_FIFO.html



Tamaño del quantum

- La determinación del tamaño del **cuanto** es vital para la operación efectiva de un sistema de cómputo. ¿Debe el **cuanto** ser pequeño o grande?, ¿fijo o variable?, ¿el mismo para todos los usuarios o debe determinarse por separado para cada uno?



Tamaño del quantum

- Si el **cuanto** de tiempo es muy grande, cada proceso tendrá el tiempo necesario para terminar, de manera que el esquema de planificación por turno rotatorio degenera en uno de primero-en-entrar-primero-en-salir. Si el **cuanto** es muy pequeño, el gasto extra por cambio de proceso se convierte en el factor dominante y el *rendimiento del sistema* se degradará hasta el punto en que la mayor parte del tiempo se invierte en la conmutación del procesador, con muy poco o ningún tiempo para ejecutar los programas de los usuarios.



Tamaño del quantum

- ¿Exactamente dónde, entre cero e infinito, debe fijarse el tamaño del **cuanto**? La respuesta es, lo bastante grande como para que la mayoría de las peticiones interactivas requieran menos tiempo que la duración del **cuanto**.



Tamaño del quantum

- Pongamos un ejemplo, supongamos que el cambio de proceso tarda 5 mseg., y la duración del **quantum** es de 20 mseg. Con estos parámetros, se utiliza un mínimo del 20% del tiempo del CPU en la ejecución del sistema operativo. Para incrementar la utilización del CPU por parte de los procesos de usuario podríamos establecer un **quantum** de 500 mseg., el tiempo desperdiciado con este parámetro sería del 1%.



Tamaño del quantum

- Pero consideremos lo que ocurriría si diez usuarios interactivos oprimieran la tecla *enter* casi al mismo tiempo. Diez procesos se colocarían en la lista de procesos listos. Si la CPU está inactiva, el primero de los procesos comenzaría de inmediato, el segundo comenzaría medio segundo después, etc. Partiendo de la hipótesis de que todos los procesos agoten su *quantum*, el último proceso deberá de esperar 4.5 seg. para poder ejecutarse. Esperar 4.5 seg. para la ejecución de una orden sencilla como *pwd* parece excesivo.



Tamaño del quantum

- En conclusión, un *quantum* pequeño disminuye el rendimiento de la CPU, mientras que un *quantum* muy largo empobrece los tiempos de respuesta y degenera en el algoritmo FIFO. Como solución al ejemplo es adoptar un término medio como 100 mseg.




Planificación por prioridad al más corto (SJF, *Short Job First*).

- Al igual que en el *algoritmo FIFO* las ráfagas se ejecutan sin **interrupción**, por tanto, sólo es útil para entornos batch o por lotes. Su característica es que cuando se activa el planificador, éste elige la ráfaga de menor duración. Es decir, introduce una noción de prioridad entre ráfagas. Hay que recordar que en los entornos *batch* se pueden hacer estimaciones del tiempo de ejecución de los procesos.




Planificación por prioridad al más corto (SJF, *Short Job First*).

- La ventaja que presenta este algoritmo sobre el algoritmo FIFO es que minimiza el tiempo de finalización promedio, como puede verse en el siguiente ejemplo:



Planificación por prioridad al más corto (SJF, *Short Job First*).

- Ejemplo: Supongamos que en un momento dado existen tres ráfagas listos R1, R2 y R3, sus tiempos de ejecución respectivos son 24, 3 y 3 ms. El proceso al que pertenece la ráfaga R1 es la que lleva más tiempo ejecutable, seguido del proceso al que pertenece R2 y del de R3. Veamos el tiempo medio de finalización (F) de las ráfagas aplicando FIFO y SJF:




Planificación por prioridad al más corto (SJF, *Short Job First*).

- $FIFO F = (24 + 27 + 30) / 3 = 27 \text{ ms.}$
- $SJF F = (3 + 6 + 30) / 3 = 13 \text{ ms.}$



Planificación por prioridad al más corto (SJF, *Short Job First*).

- No obstante, este algoritmo sólo es óptimo cuando se tienen simultáneamente todas las ráfagas. Como contraejemplo, considérense cinco ráfagas desde A hasta E, con tiempo de ejecución de 2, 4, 1, 1 y 1 respectivamente. Sus tiempos de llegada son 0, 0, 3, 3 y 3. Primero se dispone de A y B, puesto que las demás ráfagas no han llegado aún. Con el algoritmo SJF las ejecutaríamos en orden A, B, C, D, y E con un tiempo de finalización promedio de 6.4. Sin embargo, al ejecutarlas en orden B, A, C, D, E, se tiene un promedio de finalización de 6.8.



Planificación por prioridad al más corto (SJF, *Short Job First*).

- Simulador
- http://wwdi.ujaen.es/~lina/TemasSO/PLANIFICACIONDEPROCESOS/PlanificadorProcesos/alg_planif_SJF.html



Planificación por prioridad al tiempo restante más corto. SRTF (Short Remaining Time First)

- Es similar al anterior, con la diferencia de que si un nuevo proceso pasa a listo se activa el dispatcher para ver si es más corto que lo que queda por ejecutar del proceso en ejecución. Si es así el proceso en ejecución pasa a listo y su tiempo de estimación se decrementa con el tiempo que ha estado ejecutándose.



Planificación por prioridad al tiempo restante más corto. SRTF (Short Remaining Time First)

- Un punto débil de este algoritmo se evidencia cuando una ráfaga muy corta suspende a otra un poco más larga, siendo más largo la ejecución en este orden al ser preciso un cambio adicional de proceso y la ejecución del código del planificador.



Planificación a la tasa de respuesta más alta

- Brinch Hansen desarrolló la estrategia de prioridad a la tasa de respuesta más alta (HRN, highest-response-ratio-next) que corrige algunas deficiencias de *SJF*, particularmente el retraso excesivo de trabajos largos y el favoritismo excesivo para los trabajos cortos.



Planificación a la tasa de respuesta más alta

- HRN es un disciplina de planificación *no apropiativa* en la cual la prioridad de cada proceso no sólo se calcula en función del tiempo de servicio, sino también del tiempo que ha esperado para ser atendido. Cuando un trabajo obtiene el procesador, se ejecuta hasta terminar. Las prioridades dinámicas en HRN se calculan de acuerdo con la siguiente expresión:



Planificación a la tasa de respuesta más alta

- $\text{prioridad} = (\text{tiempo de espera} + \text{tiempo de servicio}) / \text{tiempo de servicio}$
- Como el tiempo de servicio aparece en el denominador, los procesos cortos tendrán preferencia. Pero como el tiempo de espera aparece en el numerador, los procesos largos que han esperado también tendrán un trato favorable.



Tiempo Real

- Un sistema de tiempo real es un sistema de procesamiento de información el cual tiene que responder a estímulos de entrada generados externamente en un período finito y específico.

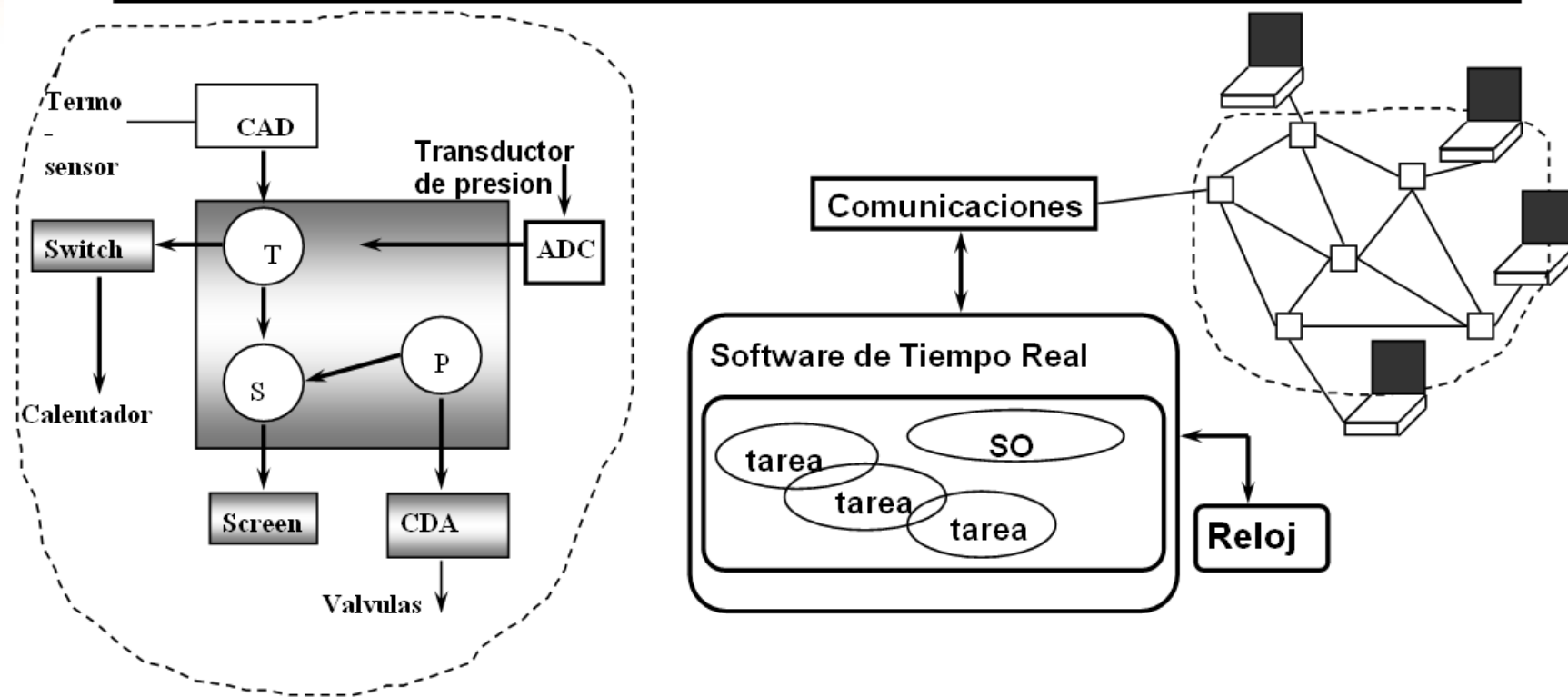


Tiempo Real

- * Las respuestas correctas dependen no solo de los resultado lógicos sino también del tiempo en que son entregadas.
- * Las fallas para responder a tiempo son tan malas como una mala respuesta

Tiempo Real

Elementos de un Sistema de Tiempo Real



Entender la Aplicacion

Caracterizar y Diseñar el Sistema

Controlar el Sistema

Monitorizar el Sistema



Características de un sistema de Tiempo Real

Tiempo

- Administración y control del tiempo
- Tareas que deben ser asignadas y terminadas antes de su plazo
- La ejecución correcta no solo considera la lógica sino también el tiempo en que se producen los resultados



Características de un sistema de Tiempo Real

Confiabilidad

- Predictibilidad
- Tolerancia a Fallos
- Seguridad



Características de un sistema de Tiempo Real

Ambiente

- Características dinámicas del ambiente

Un sistema en tiempo real es una combinación de computadoras, dispositivos de E/S, hardware y software de propósito específico donde:



Características de un sistema de Tiempo Real

- -Existe una interacción con el ambiente
- -El ambiente cambia con el tiempo
- -El sistema debe controlar y/o reaccionar a diferentes aspectos del ambiente.



Características de un sistema de Tiempo Real

- Como resultado:
 - Se imponen restricciones de tiempo al software
 - El software es naturalmente concurrente.
 - Se exige una alta confiabilidad.



Restricciones de una Sistema en Tiempo Real

- - **Restricciones de tiempo** (Cómputo, periodo, plazos)
- - **Restricciones de predictibilidad**
- - **Restricciones de recursos:** una tarea puede requerir acceso a ciertos recursos, además del procesador, como dispositivos de E/S, redes de comunicación, estructuras de datos, archivos, bases de datos, etc.



Restricciones de una Sistema en Tiempo Real

- - **Restricciones de precedencia:** una tarea puede requerir resultados de una u otra tarea antes de comenzar su ejecución.
- - **Restricciones de confiabilidad y desempeño:** una tarea podría tener que cumplir con ciertas restricciones de confiabilidad, disponibilidad o desempeño.



Predictibilidad

- Una característica distintiva de un sistema en tiempo real es la predictibilidad. La cual implica que debe ser posible demostrar o comprobar a priori que los requerimientos de tiempos se cumplen en cualquier circunstancia.



Predictibilidad

- Como consecuencia la predictibilidad implica:
 - - Una cuidadosa planificación de tareas y recursos.
 - - Cumplimiento predecible de requisitos temporales: determinismo.
 - - Anticipación a fallos, y sus requerimientos temporales.



Predictibilidad

- - Consideraciones de sobrecargas: degradación controlada.
- - Consideraciones de elementos no predecibles.
- - Dotar al sistema con capacidades de monitorización y control de tiempos (hardware, software, sistema operativo, lenguaje, líneas y protocolos de comunicación.)



Planificación en sistemas de Tiempo Real

- En las aplicaciones de tiempo real, el funcionamiento correcto NO sólo depende de los resultados de los cálculos, sino que también depende del instante en el que éstos son obtenidos. El sistema operativo, y concretamente la política de planificación debe garantizar que todas las tareas críticas sean capaces de cumplir sus plazos de ejecución (obtener los resultados antes del plazo máximo especificado).



Tareas de tiempo real

- Podemos identificar 3 tipos de tareas:
- Tareas periódicas
 - **Se ejecutan de forma repetitiva cada cierto tiempo**
- Tareas esporádicas o de plazo fijo
 - **Se ejecutan una vez en un instante determinado**
- Tareas aperiódicas
 - **Se ejecutan de forma repetitiva sin un tiempo establecido**



Tareas de tiempo real

- Si atendemos a las características de las tareas, podemos dividir las en dos grupos:
- **Críticas:** El fallo de una de estas tareas puede ser catastrófico.
- **Opcionales** (no críticas): Se pueden utilizar para refinar el resultado dado por una tarea crítica, o para monitorizar el estado del sistema, etc



Tareas de tiempo real

- Los estudios de planificación suponen que todas las tareas críticas son periódicas o esporádicas; y las tareas opcionales se tratan como aperiódicas, bien con plazo o sin plazo de finalización asignado.



Características de las políticas de planificación

- Los objetivos que persigue toda política de planificación de tiempo real son:
- -Garantizar la correcta ejecución de todas las tareas críticas.
- -Ofrecer un buen tiempo de respuesta a las tareas aperiódicas sin plazo.
- -Administrar el uso de recursos compartidos.



Características de las políticas de planificación

- -Posibilidad de recuperación ante fallos de software o hardware.
- -Soportar cambios de modo. Cambiar en tiempo de ejecución el conjunto de tareas. Por ejemplo: un cohete espacial tiene que realizar acciones muy distintas durante el lanzamiento, estancia en órbita y regreso; en cada fase, el conjunto de tareas que se tengan que ejecutar ha de ser distinto.



Clasificación de las políticas de planificación

- Los primeros planificadores se diseñaban »a mano«, esto es, se construía durante la fase de diseño del sistema un plan con todas las acciones que tenía que llevar a cabo el planificador durante la ejecución. Durante la ejecución, el planificador tan sólo tenía que consultar la tabla (plan). Las ordenes de planificación que estaban en la tabla se repetían constantemente. A estos planificadores se les llama CÍCLICOS.



Clasificación de las políticas de planificación

- El principal problema que presentan es la poca flexibilidad a la hora de modificar alguno de los parámetros de las tareas, pues ello conlleva la re-elaboración de todo el plan.
- Aún hoy este tipo de planificación se utiliza en la industria.



Clasificación de las políticas de planificación

- El otro gran grupo de planificadores son los basados en prioridades. Estos a su vez se dividen en prioridades estáticas y prioridades dinámicas.
- En los planificadores estáticos, durante la fase de diseño a cada tarea se le asigna una prioridad. Después, durante la ejecución, el algoritmo de planificación simplemente tiene que ordenar la ejecución de las tareas en función de la prioridad asignada.



Cambios de Contexto y Test de planificabilidad

- Hasta ahora hemos supuesto que los cambios de contexto consumían tiempo.
- En los actuales procesadores RISC, CISC, Intel, etc., que cuentan con un gran número de registros, el tiempo de salvar todos estos registros en memoria principal y recargar los registros de otra tarea para continuar con su ejecución no es despreciable.



Cambios de Contexto y Test de planificabilidad

- Si estudiamos en detalle cómo se producen los cambios de contexto veremos que podremos acotar, el tiempo que cualquier tarea pierde en esta labor.



Cambios de Contexto y Test de planificabilidad

- Cada vez que una tarea de más prioridad se activa (suponemos que en esos momentos está en ejecución otra de menor prioridad) ha de expulsar a la menos prioritaria (lo que implica suspender la menos prioritaria y activar la nueva tarea);



Cambios de Contexto y Test de planificabilidad

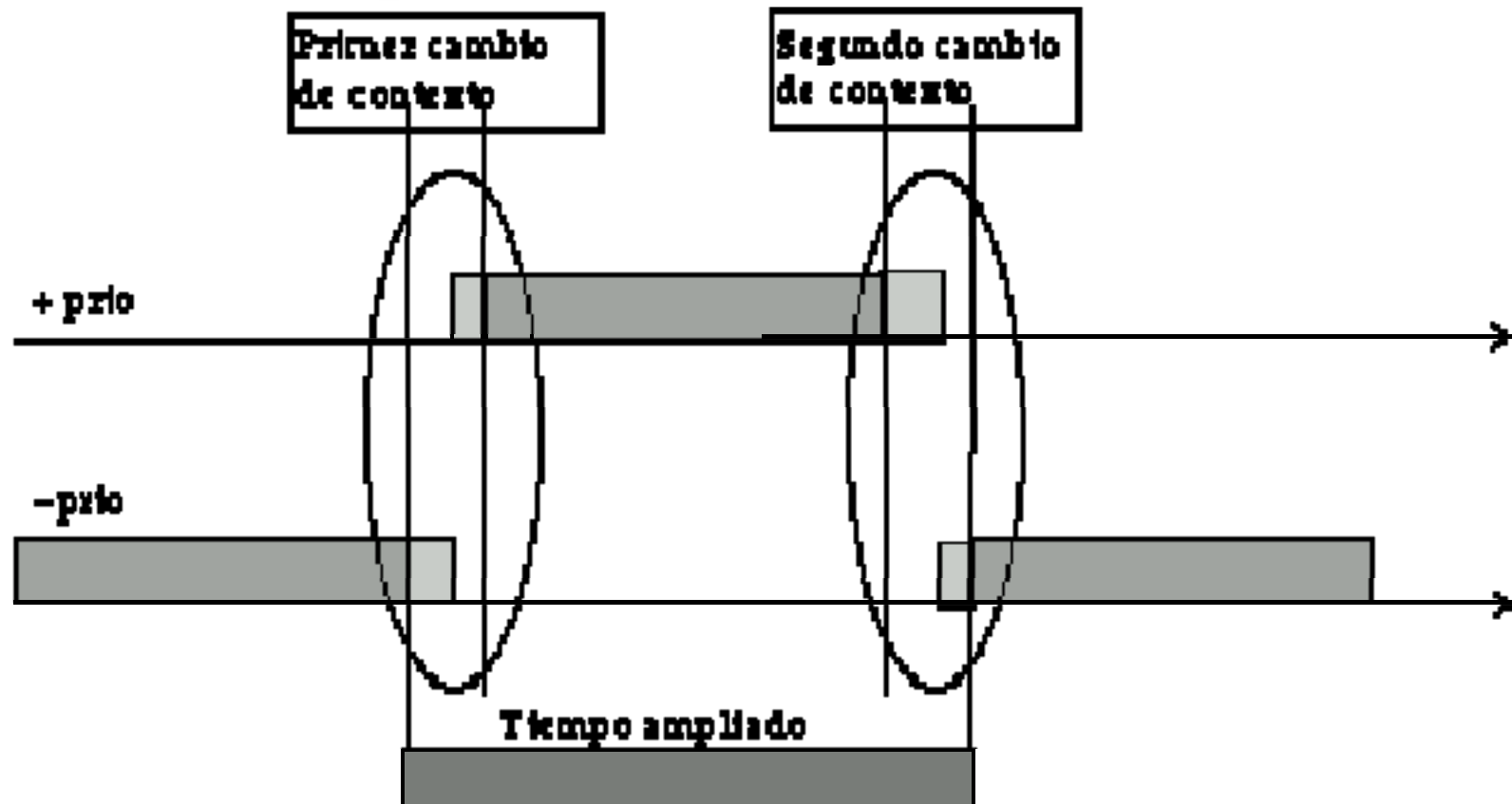
- y al finalizar la ejecución hay que reponer la ejecución de la tarea interrumpida (eliminando la más prioritaria y reponiendo la tarea anterior). Es importante notar que no se producen más cambios de contexto (si suponemos que no hay más activaciones).



Cambios de Contexto y Test de planificabilidad

- Los dos cambios de contexto han sido forzados por la tarea más prioritaria, por lo que podemos considerar este tiempo como tiempo de cómputo de ésta. Así, la tarea menos prioritaria puede no tener en cuenta los tiempos de cambio de contexto producidos por tareas más prioritarias.

Cambios de Contexto y Test de planificabilidad





Cambios de Contexto y Test de planificabilidad

- El test de planificabilidad permite garantizar:
“Un conjunto de »n« tareas será planificable bajo cualquier asignación de prioridades si y sólo si: Cada tarea cumple su plazo de ejecución en el peor caso”.



Cambios de Contexto y Test de planificabilidad

- Este test de planificabilidad sigue un conjunto de expresiones matemáticas que permiten conocer de manera aproximada y en el peor de los casos los tiempos de respuesta en base a prioridades de los procesos que se ejecutan en sistemas de tiempo real.



Cambios de Contexto y Test de planificabilidad

- El test de planificabilidad no se ve alterado por los cambios de contexto si modificamos los tiempos de cómputo de todas las tareas con dos veces el tiempo de cambio de contexto.



Cambios de Contexto y Test de planificabilidad

- A la tarea menos prioritaria si bien no puede provocar expulsiones a otras tareas, también hay que añadirle los tiempos de cambio de contexto pues en realidad si expulsa a otra tarea: la tarea idle. Ésta es una tarea comodín utilizada en casi todas las implementaciones de sistemas operativos para simplificar el código del planificador.



Tareas aperiódicas

- El objetivo que se pretende es el conseguir el **menor tiempo de respuesta posible**, evidentemente sin que por ello se pierda ningún plazo de ejecución de ninguna tarea crítica (periódica).



Tareas aperiódicas

- Podemos hacer la siguiente clasificación de algoritmos de planificación:
- -Servidor en background (segundo plano)
- -Bandwidth preserving (conservación del tiempo de servicio)
- -Pooling (por consulta)
- -Priority exchange (intercambio de prioridades)
- -Deferrable server (servidor aplazable)
- -Sporadic server (servidor esporádico)
- -Slack Stealing (extracción de holgura)
- -Estático
- -Dinámico



Background

- Es el más sencillo de implementar. Cuando llega una petición aperiódica no es atendida inmediatamente, sino que se deja suspendida. Cuando no hay activaciones de tareas periódicas pendientes de ejecutar, entonces se atiende a todas las peticiones de trabajo aperiódico que hay esperando. Básicamente consiste en sustituir el proceso idle del sistema por servicio aperiódico.



Background

- Otra forma de implementarlo consiste en añadir al sistema una tarea con periodo, plazo de ejecución y tiempo de cómputo infinitos y la prioridad más baja.
- Este método si bien es muy sencillo de implementar, ofrece un mal tiempo de respuesta, concretamente ofrece el peor tiempo de respuesta posible de entre los planificadores que no dejan el procesador ocioso cuando hay trabajo que servir.



Pooling

- Se añade una nueva tarea periódica a la tareas críticas originales. Cada vez que se activa el servidor, consulta si hay trabajo aperiódico pendiente y lo sirve. Estará sirviendo trabajo aperiódico mientras le quede tiempo de cómputo o haya trabajo que servir. Cuando acaba de servir, se suspende hasta que nuevamente se active en el siguiente periodo.



Pooling

- La tarea que realiza el pooling tiene asignados un periodo, un tiempo de cómputo máximo y una prioridad. Estos tres valores son arbitrarios, siempre y cuando el conjunto de todas las tareas sea planificable. La prioridad no se obtiene a partir del periodo, sino que se puede asignar la prioridad que más convenga.



Deferrable server

- -El servidor diferido se comporta como una tarea periódica de máxima prioridad:
- -Mientras le quede tiempo de cómputo disponible, el servidor atenderá las peticiones aperiodicas inmediatamente (pues tiene la máxima prioridad).
- -La capacidad del servidor se repone completamente al inicio de cada periodo.



Deferrable server

- Es importante hacer notar que a diferencia de el servidor por pooling, que sólo podía servir trabajo al inicio de cada periodo, el deferrable server puede ejecutar tareas aperiódicas en cualquier instante (siempre y cuando disponga de tiempo).