La calendarización y administración de procesos

Administrador del procesador

Roberto Gómez C.

Ciclos de ráfagas de CPU y E/S

cargar almacenar sumar almacenar leer archivo	*ráfaga de CPU*
esperar E/S	*ráfaga de E/S*
almacenar incremento indexar escribir archivo	*ráfaga de CPU*
esperar E/S	*ráfaga de E/S*
cargar almacenar sumar almacenar leer archivo	*ráfaga de CPU*
esperar E/S	*ráfaga de E/S*

Roberto Gómez C.

I/O bound vs CPU bound

- Un programa limitado por E/S (I/O bound) suele tener muchas ráfagas de CPU muy cortas
- Un programa limitado por CPU (CPU bound) suele tener unas cuantas ráfagas de CPU muy largas

Roberto Gómez C.

Planificador del CPU

- Siempre que CPU este ocioso, el S.O. debe escoger uno de los procesos que están en la cola de procesos listos para ejecutarlo
- El proceso de selección corre por cuenta del planificador a corto plazo o planificador de CPU
- Planificador escoge uno de los procesos que están en la memoria y listos para ejecutarse y se le asigna el CPU

Roberto Gómez C.

Situaciones cambio proceso

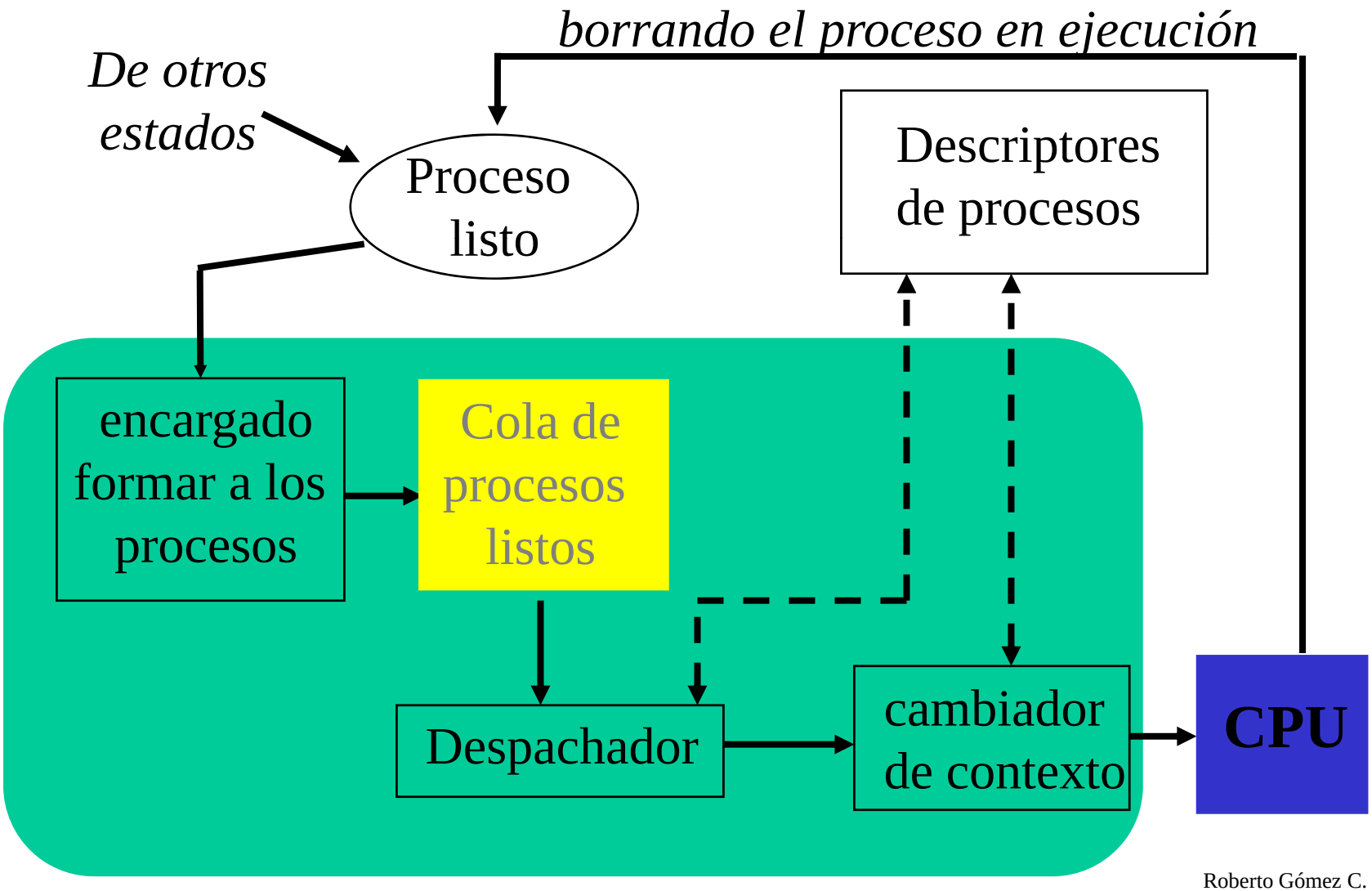
- Cuando un proceso pasa del estado en ejecución al estado bloqueado
- Cuando un proceso pasa del estado en ejecución al estado listo
- Cuando un proceso pasa del estado bloqueado al estado listo
- Cuando un proceso termina

Roberto Gómez C.

El despachador

- Es el modulo que cede el control del CPU al proceso seleccionado por el planificador a corto plazo
- Esta función implica
  - Cambiar de contexto
  - Cambiar a un modo de usuario
  - Saltar al punto apropiado del programa del usuario para reiniciar dicho programa

El calendarizador



Criterios de calendarización

- Uso de la CPU
  - mantener el CPU tan ocupado como se pueda
  - uso CPU varia entre 0 y 100%
  - en un sistema real, debe variar entre 40% (poca carga) y 90% (muy cargado)
- Rendimiento
  - si CPU esta ocupada ejecutando procesos, se esta ejecutando trabajos

- rendimiento: número de procesos que se completan por unidad de tiempo
- por ejemplo: un proceso por hora o 10 procesos por segundo

- Tiempo de regreso (servicio)
  - tiempo que se tarda en ejecutar un proceso
  - intervalo entre el momento de presentación de un proceso al planificador y el momento en que se terminó

- es la suma de los periodos durante los cuales el proceso espera entrar en memoria espera en cola de procesos listos, se ejecuta en el CPU y realiza E/S

- Tiempo de espera
  - no se toma en cuenta tiempo proceso pasa ejecutándose o realizando E/S
  - tan solo abarca el tiempo que un proceso pasa esperando en al cola de procesos listos

- suma periodos que pasa en la cola de procesos listos

- Tiempo de respuesta
  - tiempo que transcurre entre que se presenta una solicitud y se producen los primeros resultados
  - no incluye el tiempo que se toma en exhibir la respuesta
  - generalmente esta limitado por la velocidad del dispositivo de salida

El comando time de Linux

- Ejecuta programa especificado con sus argumentos
- Cuando termina, escribe mensaje a stderr proporcionando estadísticas de timpo de ejecución:
  - Tiempo transcurrido entre invocación y termino del programa
  - Tiempo CPU del usuario
  - Tiempo CPU sistema

Ejemplo uso time

```
$ more hola.c
int main( )
{
  printf("Hola mundo \n");
}
$ time gcc hola.c

real    0m0.228s
user    0m0.120s
sys     0m0.040s
$
```

Calendarizador óptimo

- Condición: no entran nuevos procesos en la lista de procesos listos, mientras que los que se encuentran en ella son atendidos
- Algoritmo óptimo calcula el número de tiempo-quantum (número de veces que el CPU será asignado a cada proceso) y entonces enumera todas las posibles opciones para calendarizar el CPU
- Estrategia óptima puede seleccionarse, basada en criterio de optimización, considerando cada opción (ordering)

Factores a considerar

- Procesos nuevos llegan mientras que se esta atendiendo a otros procesos, el calendarizador debe realizar cálculos cada vez que un proceso nuevo llega
- El tiempo de ejecución de un proceso se debe conocer antes de que el proceso se ejecute
- En algunas ocasiones el calendarizador usa más tiempo realizando cálculos dentro del algoritmo, que atendiendo procesos

Niveles de administración

- Administración de alto nivel
- Administración de nivel intermedio
- Administración de bajo nivel

Administración de alto nivel

- Planificación del trabajo
- Cuales trabajos(jobs) pueden competir por los recursos
- A veces llamada *administración de admisión*, ya que decide cuales trabajos deben de admitirse

Administración nivel intermedio

- Que procesos pueden competir por el CPU
- Suspención temporal y reanudación de procesos
- Intermediario entre la admisión de trabajos y la asignación del CPU a procesos

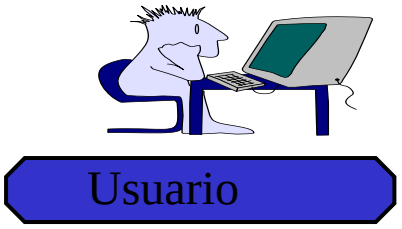
Roberto Gómez C.

Administración de bajo nivel

- A cual proceso *listo* se le asignará el CPU
- Se realiza mediante el despachador (dispatcher)
- Despachador siempre en memoria principal

Roberto Gómez C.

Esquema niveles administración



Usuario

Jobs esperando para entrar

Entrada de trabajos

Jobs en espera para iniciarse

Inicio de trabajos

Procesos suspendidos esperando por activación

Activación

Suspención

Procesos activos

Bloqueo o expiración de tiempo

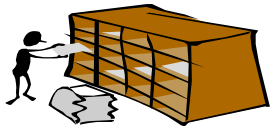
Despacho

Procesos en ejecución

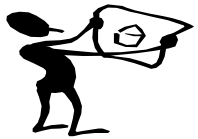
Terminación

Trabajo completado

Admon alto nivel



Admon. nivel intermedio



Admon bajo nivel

Objetivos

- Ser justa
- Elevar al máximo la producción o rendimiento
- Aumentar al máximo el número de usuarios interactivos
- Ser predecible
- Reducir al mínimo el gasto extra
- Equilibrar el aprovechamiento de recursos
- Lograr equilibrio entre la respuesta y el aprovechamiento

Más objetivos

- Evitar esperas infinitas
- Imponer prioridades
- Dar preferencia a procesos que ocupan recursos decisivos
- Dar mejor trato a procesos que muestren un comportamiento deseable.
- Degradarse paulatinamente de cargas pesadas

Admon. apropiada vs no apropiada

- No apropiada (non preemptive)
  - Una vez que el CPU se le asignó al proceso, ya no se le puede arrebatar
  - Util sistemas procesos alta prioridad requieren atención inmediata, (sistemas tiempo real)
  - Los trabajos largos atrasan a los cortos, pero el tratamiento es más justo
- Apropiada (preemptive)
  - Al proceso se le puede arrebatar el CPU
  - Util sistemas tiempo compartido: garantizar tiempos de respuesta aceptables
  - Costo: cambio de contexto implica gasto extra

Roberto Gómez C.

Prioridades

- Pueden ser asignadas automáticamente por el sistema, o se pueden asignar externamente
- Ganarse o comprarse
- Estáticas o dinámicas
- Asignación en forma racional o de manera arbitraria

Prioridades estáticas y dinámicas

- Prioridades estáticas
  - Prioridades estáticas no cambian
  - Son fáciles de llevar a la práctica
  - No responden a cambios en el ambiente
- Prioridades dinámicas
  - Responden a los cambios
  - Prioridad inicial tiene una duración corta, después de lo cual se ajusta a un valor más apropiado
  - Esquemas más complejos
  - Gasto extra justificado: aumento en la sensibilidad del sistema

Prioridades compradas

- Debe proporcionar un servicio competente y razonable a una gran comunidad de usuarios.
- Debe manejar las situaciones en las cuales un miembro de la comunidad necesite un trato especial
- Trabajo urgente esta dispuesto a pagar extra:
  - comprar prioridad (nivel más alto de servicio)
- Pago extra obligatorio puede ser necesario arrebatar recursos a otros usuarios que también pagan

Algoritmos de calendarización

- De plazo fijo
- Primeras Entradas Primeras Salidas o servicio por orden de llegada
- Por turno o Round Robin o por turno circular
- Trabajo más corto (SJF)
- Tiempo restante más corto (SRT)
- Tasa de respuesta más alta (HRN)
- De porción justa (FFS)

Algoritmos de calendarización

- Por prioridad
- Con colas de múltiples niveles
- Con colas de múltiples niveles con realimentación
- Garantizada
- Lotería
- En tiempo real

Roberto Gómez C.

Comparando los algoritmos

- Uso de diagrama de Gant para ver el desarrollo de los procesos a través del tiempo.
- Información proporcionada
  - Tiempo de llegada del procesos a la cola de procesos listos
  - Tiempo de duración del proceso
- Información calculada
  - Tiempo de termino de cada proceso
  - Tiempo promedio de termino

Administración plazo fijo

- Se programan ciertos trabajos para terminarse en un tiempo específico
- Gran valor si se entregan a tiempo
- Carecen de valor si se entregan después del plazo

Problemas admon. plazo fijo

- Administración compleja.
- Usuario debe informar por adelantado sus necesidades.
- Ejecución sin degradar el servicio a los otros
- Muchas tareas a plazo fijo:
  - planificación muy compleja.
- Posibilidad de producir gasto extra

Roberto Gómez C.

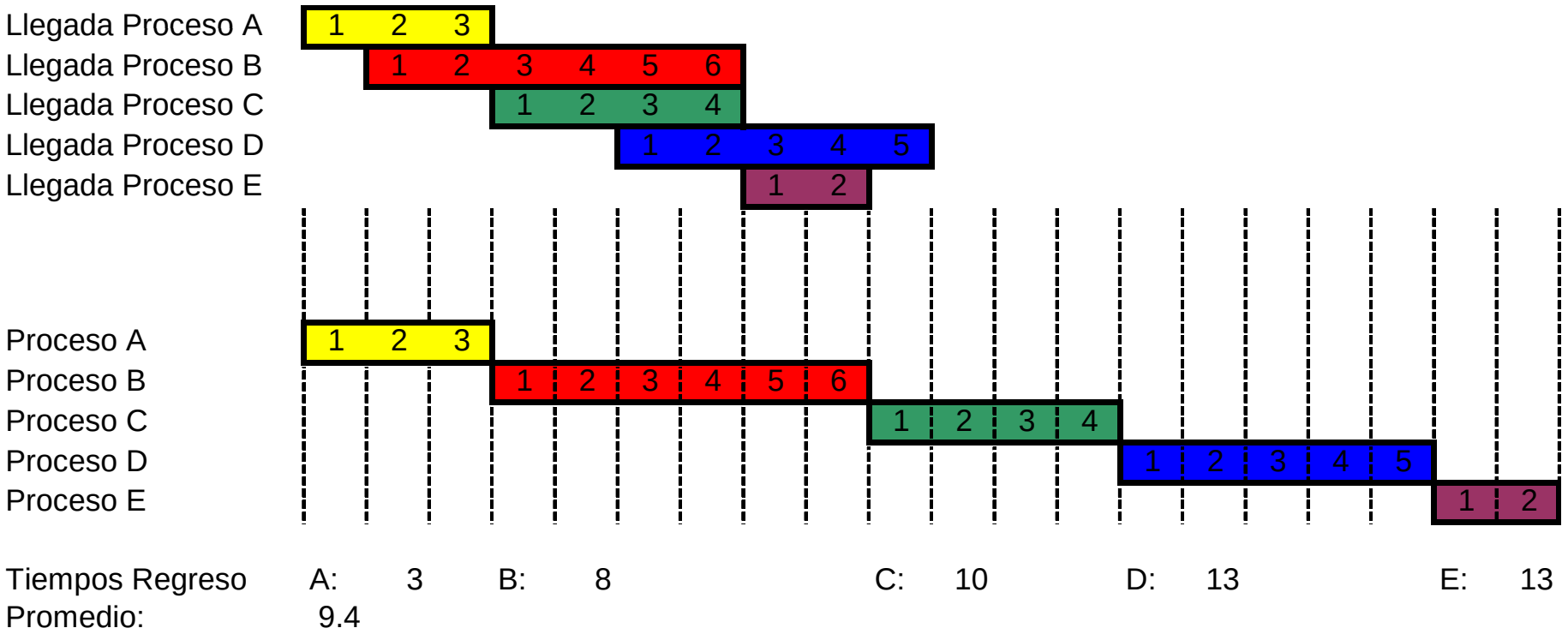
Primeras entradas primeras salidas

- Conocido como PEPS
- Disciplina más simple de planificación
- Procesos atendidos de acuerdo con su tiempo de llegada a la cola de procesos listos
- Cada vez que un proceso tiene la CPU, se ejecuta hasta terminar
- Trabajos largos hacen esperar a los pequeños
- Más predecible que otros esquemas
- Rara vez usado

Escenario

- Cinco procesos: A, B, C, D y E
- Tiempos llegada
  - A: 0
  - B: 1
  - C: 3
  - D: 5
  - E: 7
- Duración
  - A: 3
  - B: 6
  - C: 4
  - D: 5
  - E: 2
- Mismo escenario para el resto de los algoritmos

Ejemplo FIFO



Tiempos Regreso A: 3 B: 8 C: 10 D: 13 E: 13

Promedio: 9.4

Administración por turno

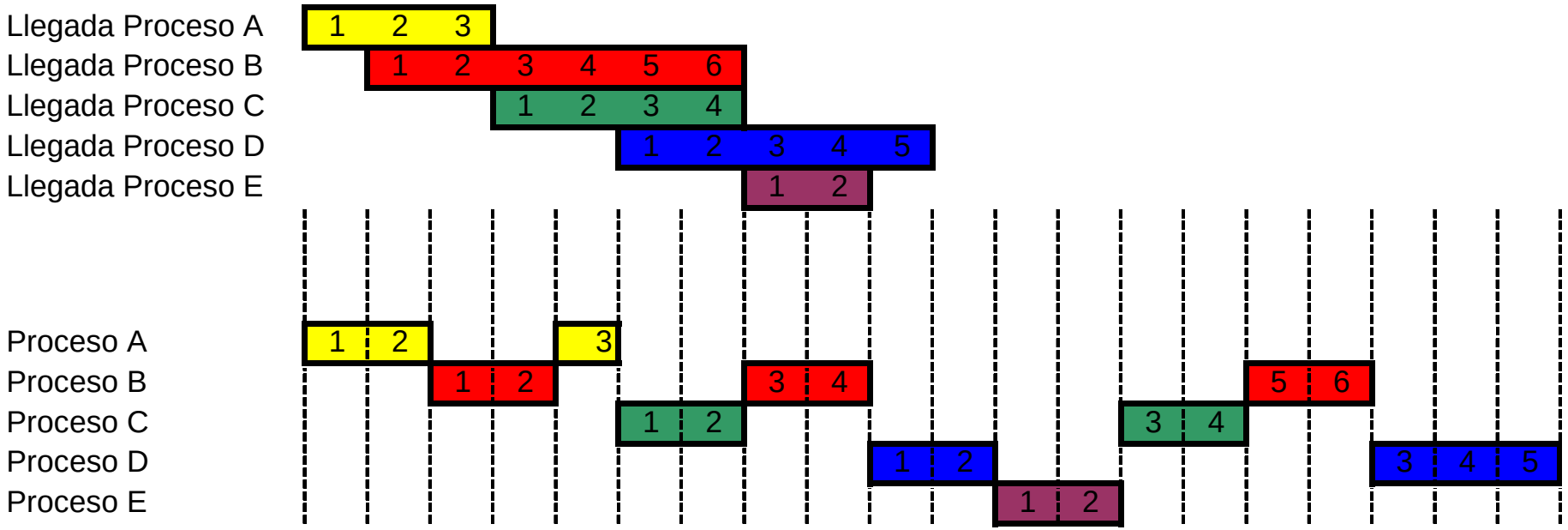
- Conocida como round-robin
- Los procesos atendidos en forma PEPS, se les asigna cantidad limitada de tiempo de CPU, (cuantum)
- Si proceso no termina en su tiempo se le quita el CPU se le asigna al siguiente proceso en espera
- Proceso desposeído se coloca al final de la lista

Por turno ...

- Efectiva en tiempo compartido con necesidad de tiempo de respuesta razonables
- Variante: turno egoísta:
  - dos colas
  - al principio el proceso espera en una cola de espera
  - cuando su prioridad alcanza un cierto nivel se le pasa a la cola de procesos activos.

Ejemplo Round Robin

Quantum: 2



Tiempos Regreso A: 5 B: 16 C: 12 D: 15 E: 6
Promedio: 10.8

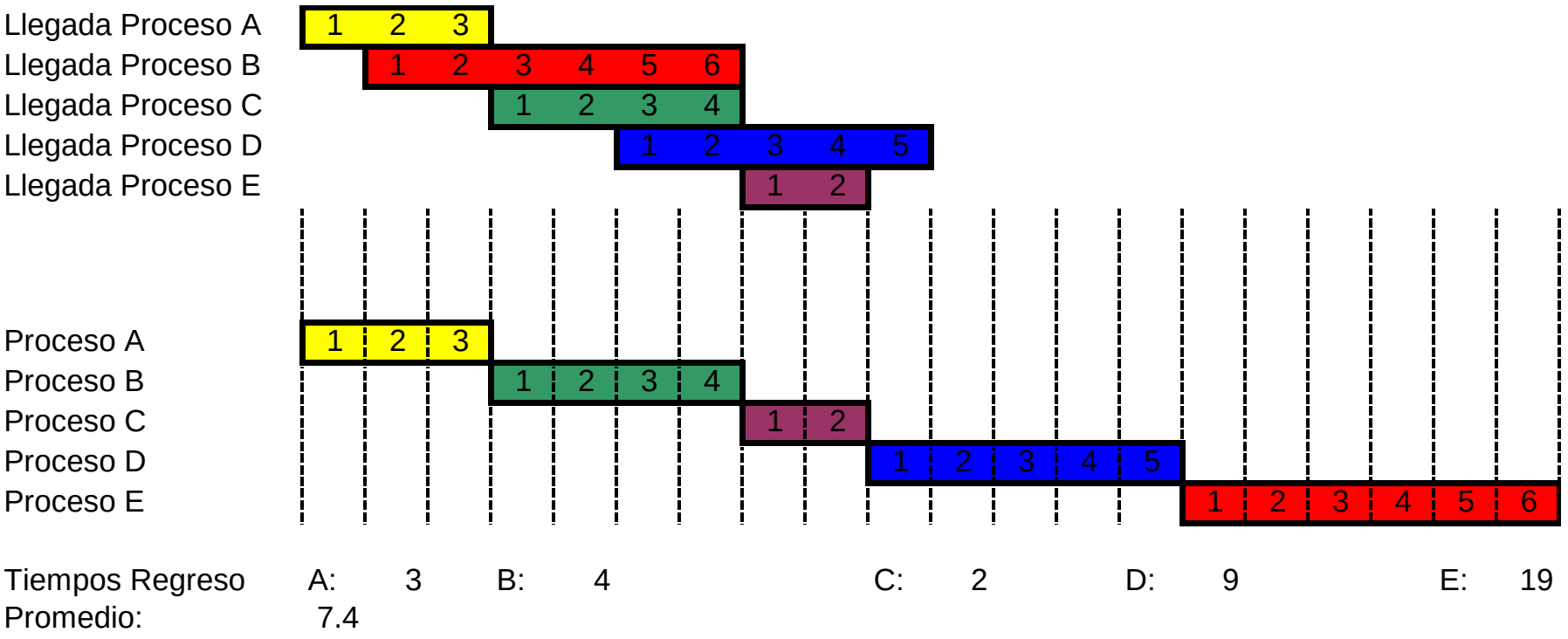
Trabajo más corto (SJF: Shortest Job First)

- Disciplina no apropiada
- Se ejecuta primero el proceso en espera que tiene el menor tiempo estimado de ejecución hasta terminar
- Reduce el tiempo de espera promedio de PEPS
- Favorece procesos cortos a expensas de los largos

SJF ...

- Selecciona procesos asegurándose que el siguiente proceso se completará y saldrá del sistema lo antes posible
- Conocimiento tiempo de ejecución de un proceso: se basa en los tiempos de ejecución estimados por el usuario (PELIGRO!!!!!)
- No resulta útil en sistemas de tiempo compartido

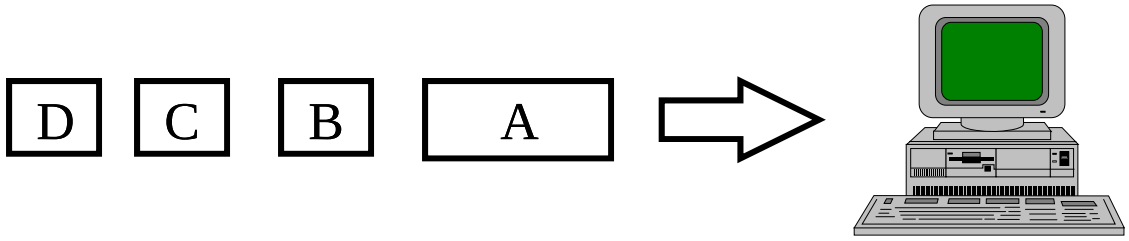
Ejemplo SJT



Tiempos Regreso A: 3 B: 4 C: 2 D: 9 E: 19
Promedio: 7.4

Segundo ejemplo SJF

- Cuatro tareas:
  - A: tiempo ejecución 8 minutos
  - B: tiempo ejecución 4 minutos
  - C: tiempo ejecución 4 minutos
  - D: tiempo ejecución 4 minutos
- En FIFO:



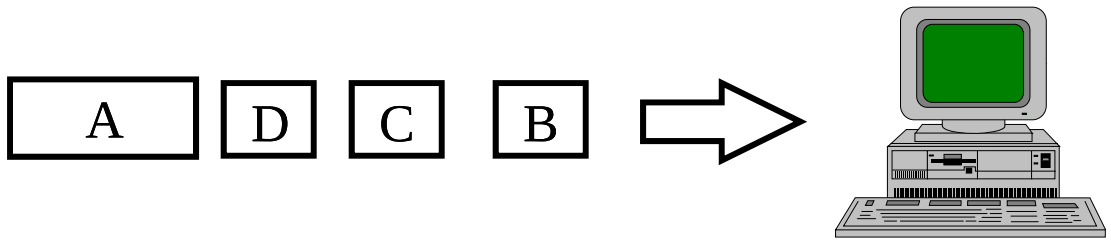
Tiempos regreso, ejecución en ese orden:

A: 8 mins, B: 12 mins, C: 16 mins, D: 20 mins

Promedio: 14 mins

Continuación ejemplo

- Cambiando el orden, (más corto primero):



Tiempos regreso:

B: 4 mins, C: 8 mins, D: 12 mins, A: 20 mins

Promedio: 11 mins

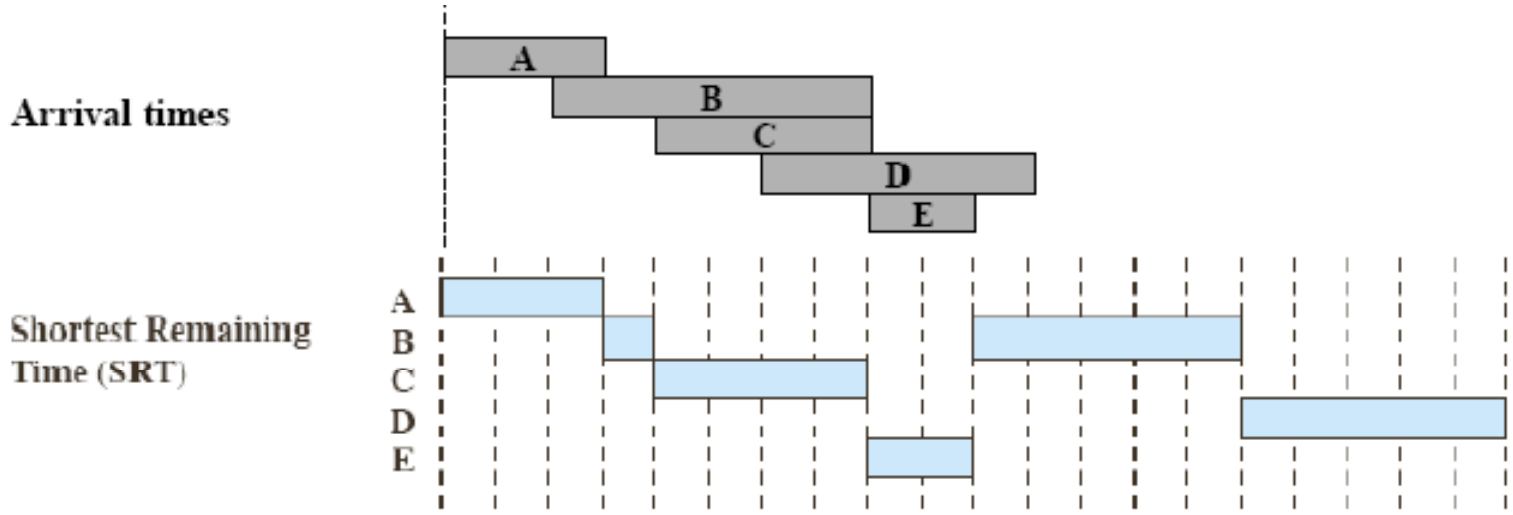
Admon por el tiempo restante más corto

- En inglés SRT: Shortest Remaining Time Scheduling
- La contraparte *apropiada* (preemptive) del anterior
- Util en tiempo compartido
- Como en SJF existe el peligro de starvation
- El proceso con el menor tiempo estimado de ejecución para terminar es el primero en ejecutarse, (incluyendo procesos nuevos)

Características SRT

- Un proceso en ejecución puede ser desposeído por uno nuevo con menor tiempo de ejecución estimado.
- También requiere estimaciones
  - Implica un mayor gasto extra:
  - pendiente del trabajo en ejecución
- Manejar apropiaciones ocasionales
- Procesos pequeños se ejecutan casi de inmediato
- Procesos grandes cuentan con un tiempo promedio de espera más grande, y tiempos de espera más variados que en SJF

Ejemplo SRT



SRT	Finish Time	A	B	C	D	E	Mean
	Turnaround Time (T_r)	3	13	4	14	2	7.20
	T_r/T_s	1.00	2.17	1.00	2.80	1.00	1.59

Admon. Tasa respuesta más alta

- HRN: Highest Response ratio Next
- Propuesto por Brinch Hansen (Br71)
- Corrige algunas deficiencias del SJF:
  - retraso excesivo de trabajos largos
  - favoritismo excesivo de trabajos cortos
- Prioridad trabajo en función de:
  - tiempo de servicio
  - tiempo de espera del trabajo

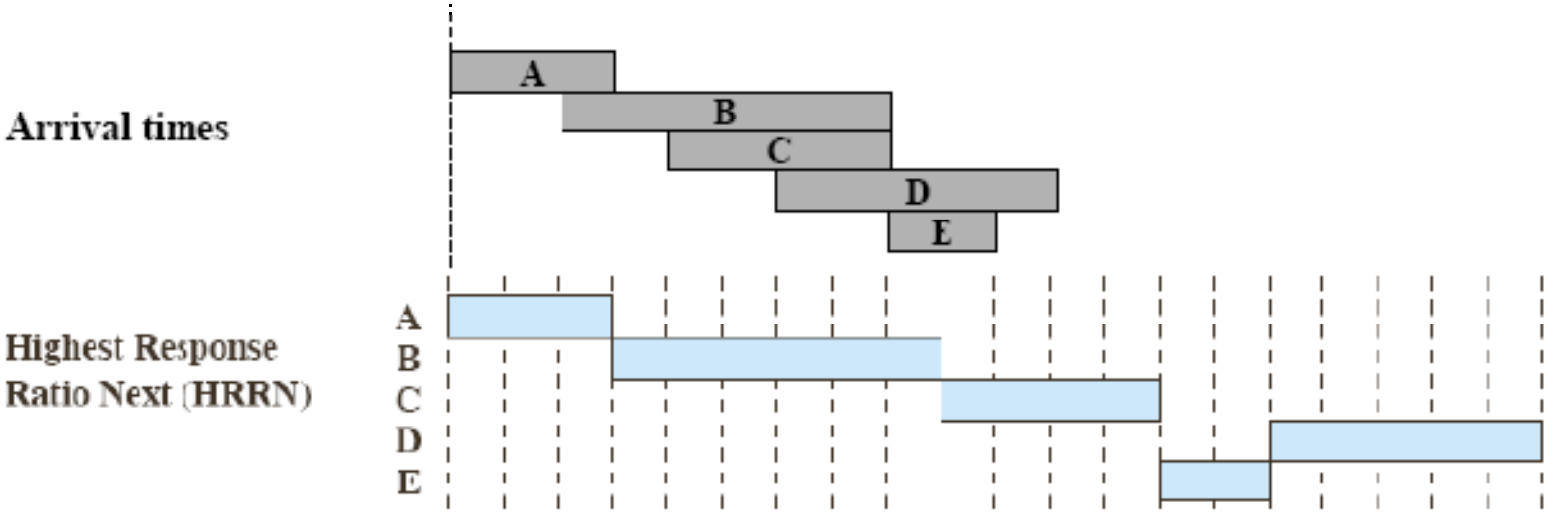
Cálculo de la prioridad en HRN

- Calculada a partir de:

$$\text{prioridad} = \frac{(\text{tiempo espera}) + (\text{tiempo servicio})}{(\text{tiempo servicio})}$$

- Observación:
  - *(tiempo espera)* + *(tiempo servicio)* es el tiempo de respuesta del sistema para el trabajo, si éste se inicia de inmediato

Ejemplo HRN



HRRN											
	Finish Time	**A**	3	**B**	9	**C**	13	**D**	20	**E**	15	**Mean**
	Turnaround Time (T_r)		3		7		9		14		7	8.00
	T_r/T_s		1.00		1.17		2.25		2.80		3.5	2.14

Fuente imagen: Stallings Operating Systems (2004, 5ª edición)

Calendarización por prioridad

- Se asigna al proceso que tiene la prioridad más alta
- Generalmente se habla en términos de prioridad alta y baja.
- Generalmente se utiliza en general un esquema expropiador ya que si un proceso con mayor prioridad que el que esta ejecutando llega a la lista de procesos listos, será asignado al procesador
- Algunos sistemas emplean número bajos para prioridades baja y en otros indica una prioridad alta

Roberto Gómez C.

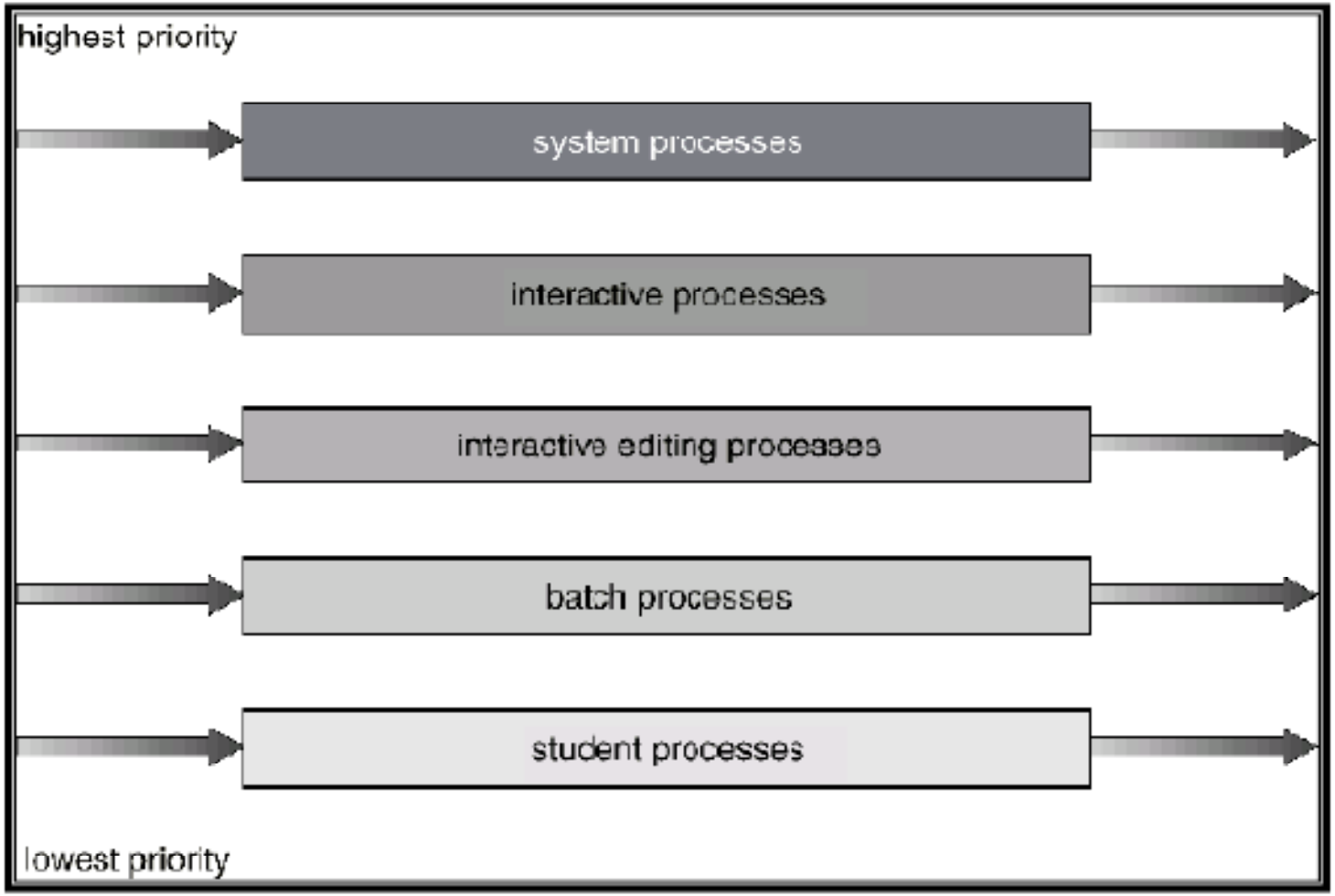
Calendarización con colas de múltiples niveles

- Se clasifican a los procesos en diferentes grupos, de acuerdo a sus cualidades.
- Suele hacerse una división entre los que se ejecutan en primer plano (interactivos) y los que lo hacen en segundo plano (por lotes)
- En colas de múltiples niveles divide la cola de procesos listos en varias colas distintas

Calendarización con colas de múltiples niveles

- Los procesos se asignan permanentemente a una cola, en base a alguna propiedad del proceso
- Cada cola tiene su propio algoritmo planificación
- Se debe contar con una estrategia de planificación entre las diferentes colas
  - no se ejecuta un proceso de una cola si no están vacías las colas de menor prioridad
  - dividir el tiempo entre colas, cada cola obtiene cierta porción del tiempo de CPU

Ejemplo colas múltiples niveles



Colas múltiples con realimentación

- En el algoritmos de colas múltiples, los procesos no se mueven de una cola a otra.
- Se deben categorizar los procesos según el uso de CPU que tengan
  - La cola de mayor prioridad será la de los procesos con gran actividad de E/S (I/O bound) y la de menor procesos con alto uso de CPU (CPU bound)
  - De esta forma, se garantiza que los procesos de poco uso de procesador tengan mayor prioridad, y los que consumen mucho procesador tendrán baja prioridad.

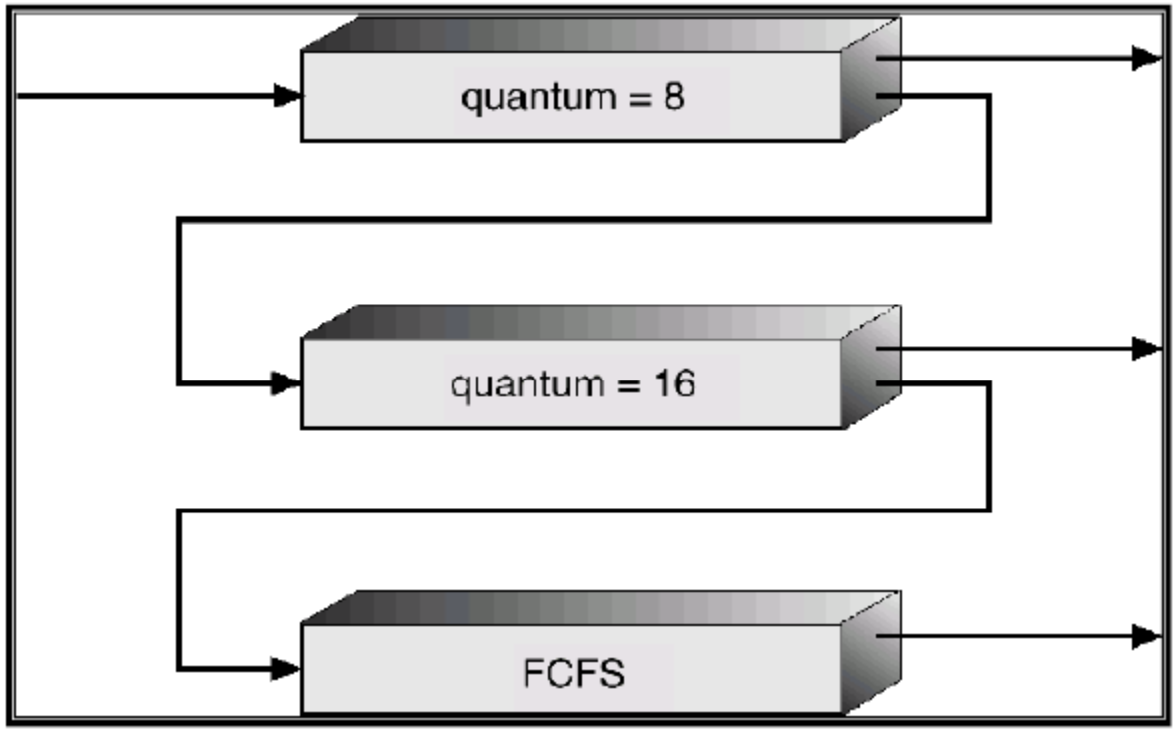
Colas múltiples con realimentación

- La realimentación permite a un proceso pasar de una cola a la otra
  - separar procesos con diferentes características en cuanto uso de CPU
  - según el consumo de CPU que hagan, los procesos serán promovidos a una cola de mayor prioridad o rebajados a una de menor prioridad.
  - si un proceso gasta demasiado tiempo de CPU se le pasará a una cola con menor prioridad

Planificador colas múltiples con realimentación

- Un planificador de cola múltiples con realimentación
  - El número de colas.
  - El algoritmo de planificación para cada cola.
  - El método utilizado para promover a un proceso a una cola de mayor prioridad.
  - El método utilizado para bajar a un proceso a una cola de menor prioridad.
  - El método utilizado para determinar a que cola será asignado un proceso cuando este pronto.

Ejemplo colas múltiples con retroalimentación



Calendarización garantizada

- Realizar una promesa al usuario acerca del desempeño y después dejarle el resto a él
- Un ejemplo es:
  - si existen n usuarios conectados mientras uno esta trabajando, a este se le otorgará el 1/n de la potencia del CPU
  - en sistema de un solo usuario, si hay n procesos, cada uno tendrá 1/n de los ciclos del CPU

- Para hacer lo anterior, el sistema debe llevar un historial de cuanto CPU cada proceso ha tenido desde su creación
- Se calcula rango uso:
  - rango 0.5 un proceso solo ha tenido la mitad de lo que le corresponde
  - rango 2.0 un proceso ha tenido el doble de lo que ha tenido
  - se escoge al proceso con el menor rango

Planificación lotería

- Algo garantizada es una buena idea, pero difícil de implementar
- Proporcionar a un proceso boletos para varios recursos, como tiempo CPU
- Cuando se tiene que elegir un proceso se toma un boleto al azar, y el proceso que tenga dicho boleto es el escogido
- Procesos más importantes se le pueden dar boletos extras

- Propiedades importantes:
  - a cada proceso nuevo se le asigna un número boletos, el cual se incrementará, por cada nueva lotería tendrá más oportunidad de ganar conforme tenga más boletos
  - procesos cooperantes pueden intercambiar boletos entre ellos
  - puede usarse para casos especiales
    - servidor vídeo, en el cual varios procesos están pasando streams videos a sus clientes pero con velocidades diferentes

Tiempo real

- Dos tipos de sistemas con respecto al tiempo de respuesta
  - Duros
  - Suaves
- *Sistemas tiempo real duros*
  - se debe completar una tarea crítica dentro de un lapso determinado
  - procesos se presentan junto con el tiempo de CPU que necesita
  - planificador admitirá el proceso, garantizando que terminará a tiempo o bien rechazará la solicitud por ser imposible

- *Sistemas tiempo real blando*
  - requiere que los procesos críticos tengan mayor prioridad que otros menos afortunados
  - la adición de este tipo de sistemas a un sistema de tiempo compartido puede dar lugar a una asignación no equitativa
  - sistema puede apoyar multimedia, gráficos interactivos de alta velocidad y otros

Planificación por reparto equitativo

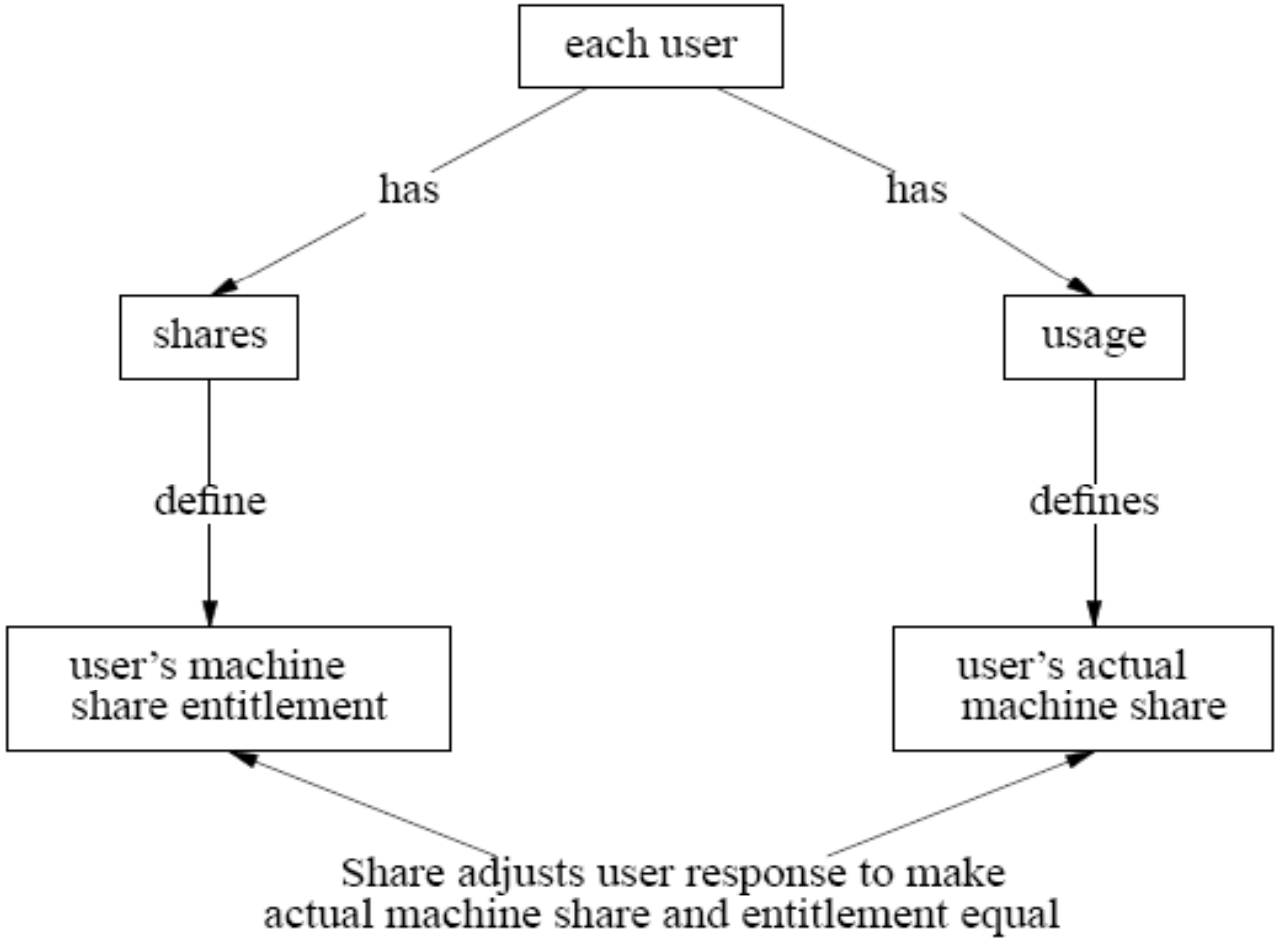
- Interés no en como se comporta un proceso en particular, sino en cómo se comparta el conjunto de procesos de un usuario que constituyen una aplicación.
- Tomar decisiones de planificación en función de estos grupos de procesos.
- Concepto se puede ampliar a grupos de usuarios, incluso si el usuario esta representado por un solo proceso
  - Por ejemplo en tiempo compartido considerar a todos los usuarios de producción, desarrollo y calidad
  - Degradación tiempo de respuesta afecta principalmente a los miembros de un departamento en lugar de afectar a los usuarios de otros departamentos.

Roberto Gómez C.

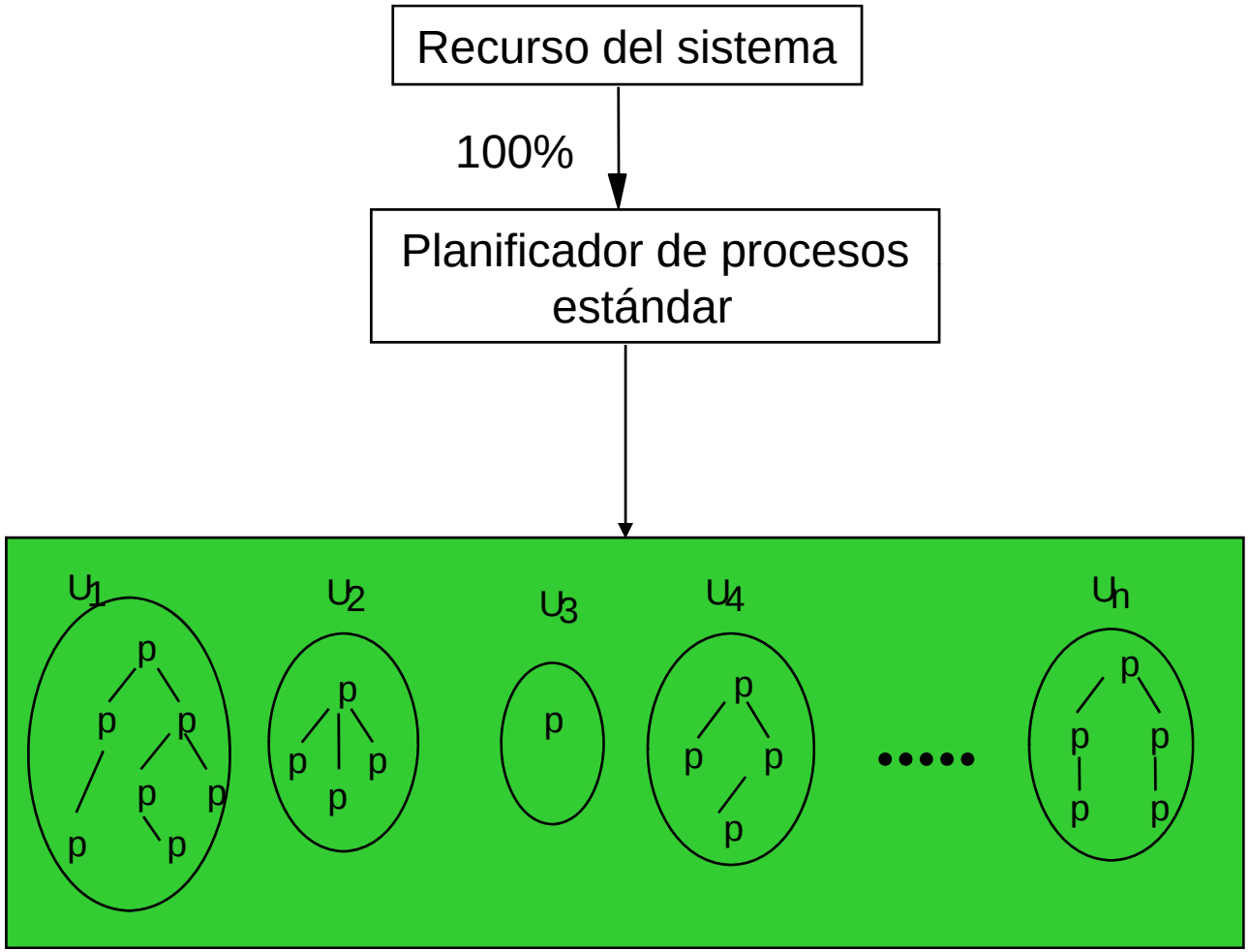
¿Reparto equitativo?

- Cada usuarios dispone de una parte del procesador
- Esquema funciona de una forma más o menor lineal.
  - Si usuario A tiene un peso dos veces mayor que el del usuario B, entonces, a la larga, el usuario A debe poder hacer el doble de trabajo que B
  - Objetivo de este esquema: repartir menos recursos a los usuarios que han consumido más de lo que les corresponde y más recursos a los que han consumido menos de lo que le corresponde.
- Ejemplo de este tipo de algoritmo, FSS de Unix
  - FSS: Fair-Share Scheduling

El reparto equitativo



Esquema admon. proceso estándar



Esquema admon justa