Planificación de procesos: Algoritmos de planificación

Gunnar Wolf

Facultad de Ingeniería, UNAM
Instituto de Investigaciones Económicas, UNAM

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Índice

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Referencia para esta sección

Buena parte del material de esta unidad toma por referencia al capítulo 2 de *An operating systems vade mecum* (Raphael Finkel, 1988), disponible para su descarga en el sitio Web del autor.

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Principal decisión en un sistema multitareas

- ¿Qué proceso es el siguiente a ejecutar?
  - ¿Qué procesos han ido terminando?
- ¿Qué *eventos* ocurrieron que hacen que *cambien de estado*?
  - Solicitudes (y respuestas) de E/S
  - *Swap* de/a disco
- ¿Cual es el siguiente proceso al que le toca atención del CPU?
  - ¿Y por cuánto tiempo?

Vemos que hay tres tipos muy distintos de planificación.

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Planificador a largo plazo

- Cual es el siguiente proceso a ser iniciado
- Principalmente orientado a la operación *en lotes*
  - Principalmente a los sistemas con *spool*
  - También presente en la multiprogramación temprana
- Decide en base a los requisitos *pre-declarados* de los procesos, y a los recursos disponibles al ejecutarse
- Periodicidad: segundos a horas
- Hoy en día no se emplean
  - El usuario indica expresamente qué procesos iniciar
  - *Podría verse* a los programas como `cron`, `at`, o en Windows al *Planificador de procesos* como cubriendo este rol
    - Aunque son procesos *plenamente en espacio de usuario*

Planificador a largo plazo

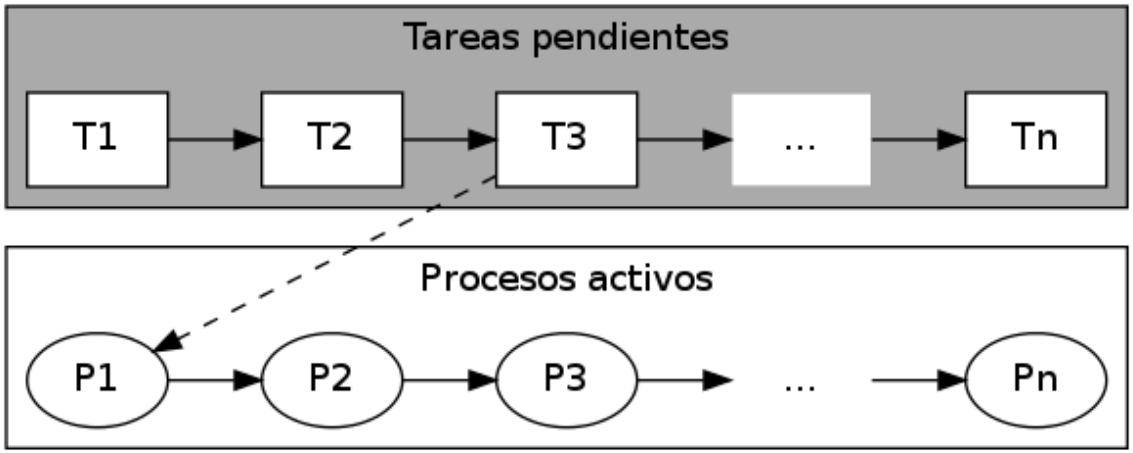

Figure: Planificador a largo plazo

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Planificador a mediano plazo

- Cuáles procesos hay que *bloquear*
  - Por escasez/saturación de algún recurso (p.ej. almacenamiento primario)
  - Por haber iniciado una operación que no puede satisfacerse aún
- Cuáles procesos hay que *desbloquear*
  - A la espera de algún dispositivo
  - Fueron enviados a *swap*, pero ya requieren o merecen ejecutarse
- Frecuentemente llamado *agendador* (*scheduler*)

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Planificador a mediano plazo

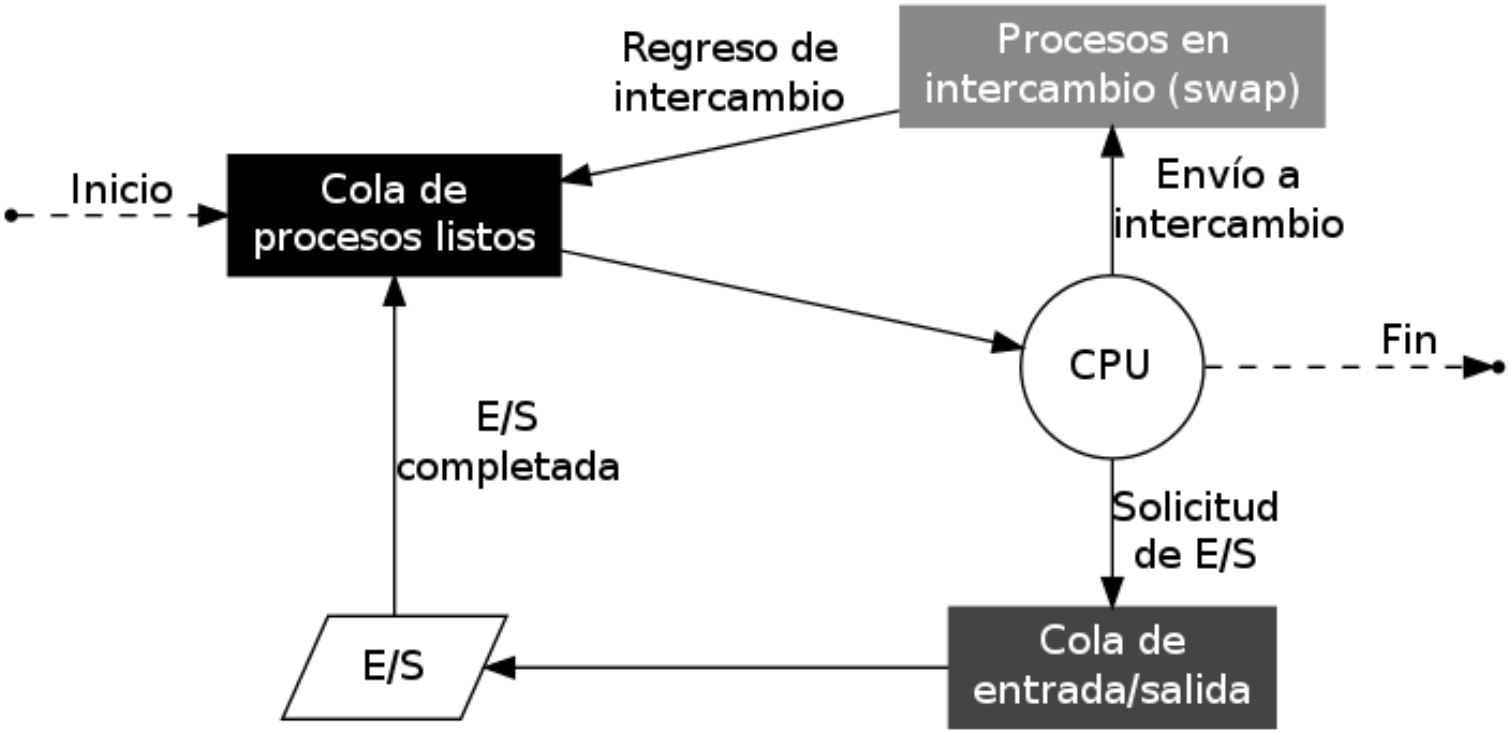

Figure: Planificador a mediano plazo, o *agendador*

Planificador a corto plazo

- Cómo compartir *momento a momento* al CPU entre todos los procesos
- Se efectúa decenas de veces por segundo
  - Debe ser simple, eficiente y rápido
- Se encarga de planificar los procesos *listos para ejecución*
  - Estados *listo* y *ejecutando*
- Frecuentemente llamado *despachador* (*dispatcher*)

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Planificador a corto plazo

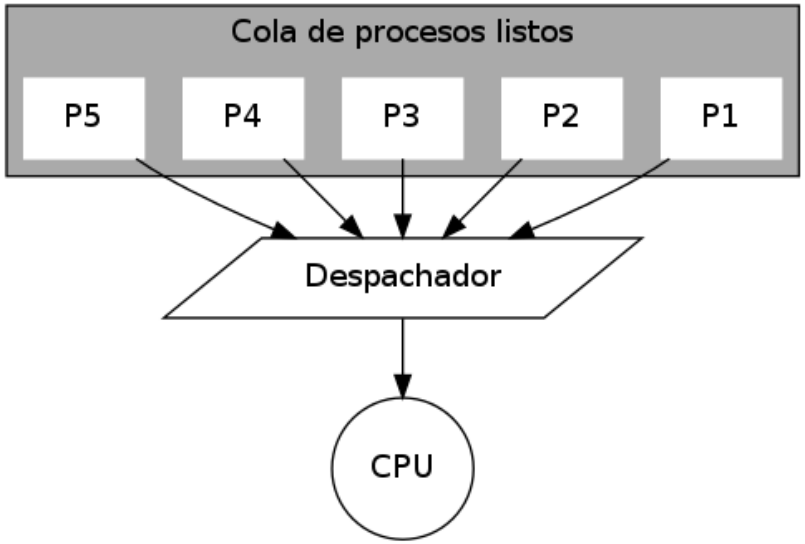

Figure: Planificador a corto plazo, o *despachador*

Tipo de planificador según transición

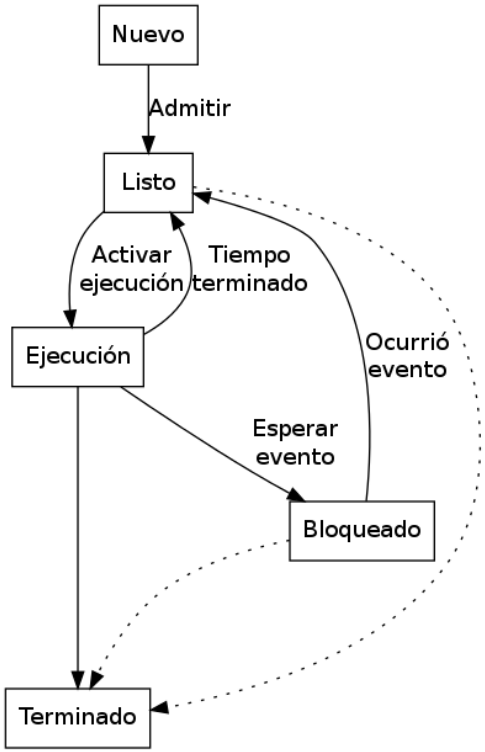

Figure: Diagrama de transición entre los estados de un proceso

- **Largo plazo**: Admitir
- **Mediano plazo**: Ocurrió evento, Esperar evento
- **Corto plazo**: Activar ejecución, Tiempo terminado

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

El enfoque de esta unidad

En esta unidad hablaremos particularmente del planificador *a corto plazo*

Cuando un proceso es *suspendido* (o *bloqueado*) y posteriormente reactivado, lo trataremos como *un proceso nuevo*.

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Tipos de proceso

- Diversos procesos tienen distintas características
- Alternan entre *ráfagas* (*bursts*)

  CPU Trabajando con datos ya existentes en el sistema

  E/S Mayormente esperando a eventos de E/S
- Un *programa* dado puede ser *mayormente* de un tipo u otro — Dicho programa está *limitado por CPU* o *limitado por E/S*
- Cuando termina una ráfaga de CPU y se suspende esperando E/S, deja de estar *listo* y sale de la vista del *despachador*
- Esto nos lleva a separar los procesos en...

Tipos de proceso

Ojo: ¡Un poco contraintuitivo!

Largos Han estado *listos* o *en ejecución* por mucho tiempo

- Esto es, están en una ráfaga de CPU

Cortos En este momento están en una ráfaga de E/S

- Requieren atención meramente ocasional del procesador
- Tienden a estar bloqueados, esperando a *eventos*
  - Como buena parte de los procesos interactivos

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Índice

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Unidades a manejar

Para hablar de planificación del procesador, *no* vamos a manejar tiempos *estándar* (s, ms, ns), sino que:

Tick Un tiempo mínimo dado durante el cual se puede realizar trabajo útil. Medida caprichosa y arbitraria.

- En Windows, un *tick* dura entre 10 y 15 ms.
- En Linux (2.6.8 en adelante), dura 1 ms.

Quantum Tiempo mínimo, expresado en *ticks*, que se permitirá a un proceso el uso del procesador.

- En Windows, 2 a 12 *ticks* (esto es, 20 a 180ms).
- En Linux, 10 a 200 *ticks* (10 a 200ms)

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

¿Qué es *mejor*?

- No hay un sólo criterio para definir qué es una *mejor* respuesta
- El patrón correcto varía según el propósito del sistema
- Un proceso interactivo *sufre* si el tiempo de respuesta incrementa, aunque pueda procesar por más tiempo corrido
  - En caso de sufrir demoras, debemos intentar que sean *consistentes*, aunque el *tiempo promedio* resulte deteriorado
  - Es mejor saber que el sistema *siempre* tardará 0.5s en responder a mis necesidades a que unas veces responda de inmediato y otras tarde 3s.
    - ¿O no?

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

¿Qué es *mejor*?

- No hay un sólo criterio para definir qué es una *mejor* respuesta
- El patrón correcto varía según el propósito del sistema
- Un proceso interactivo *sufre* si el tiempo de respuesta incrementa, aunque pueda procesar por más tiempo corrido
  - En caso de sufrir demoras, debemos intentar que sean *consistentes*, aunque el *tiempo promedio* resulte deteriorado
  - Es mejor saber que el sistema *siempre* tardará 0.5s en responder a mis necesidades a que unas veces responda de inmediato y otras tarde 3s.
    - ¿O no?

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

¿Qué métricas compararemos?

Para un proceso p que requiere ejecutarse por tiempo t,

Tiempo de respuesta (T) Tiempo total que toma el trabajo. Incluye el tiempo que pasó inactivo (pero listo).

Tiempo en espera (E) De T, cuánto tiempo está esperando ejecutar. (*Tiempo perdido*)

- $E = T - t$; Idealmente, para p, $E_p \to 0$

Proporción de penalización (P) Fracción del tiempo de respuesta durante la cual p estuvo en espera.

- $P = \frac{T}{t}$

Proporción de respuesta (R) Fracción del tiempo de respuesta durante la cual p pudo ejecutarse.

- $R = \frac{t}{T}$; $R = \frac{1}{P}$

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Además de los anteriores, para el sistema...

Tiempo núcleo o *kernel* Tiempo que pasa el sistema en espacio de núcleo

Tiempo desocupado (*idle*) Tiempo en que la cola de procesos listos está vacía y no puede realizarse ningún trabajo.

- El sistema operativo aprovecha este tiempo para realizar *tareas de mantenimiento*

Utilización del CPU Porcentaje del tiempo en que el CPU está realizando *trabajo útil*.

- Conceptualmente, entre 0 y 100%
- En realidad, en un rango entre 40 y el 90%.

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Por ejemplo. . .

Los siguientes procesos forman la *cola de procesos listos*:

Proceso	Ticks	Llegada
A	7	0
B	3	2
C	12	6
D	4	20

Toma 1 *tick* realizar un cambio de contexto; cada *quantum* es de 5 *ticks*, y tenemos un ordenamiento *de ronda*[1]

[1] Que pronto describiremos

Precisiones sobre el ejemplo

Nuestro ejemplo *no es realista*

- ¡El cambio de contexto propuesto es desproporcionadamente largo! (sólo para ejemplificar)
- Consideraremos al tiempo núcleo como si fuera un proceso más
  - Midiendo *como si iniciara y terminara* junto con los demás
  - Normalmente el tiempo núcleo no se cuenta, es tomado por *burocracia*

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Graficando nuestro ejemplo

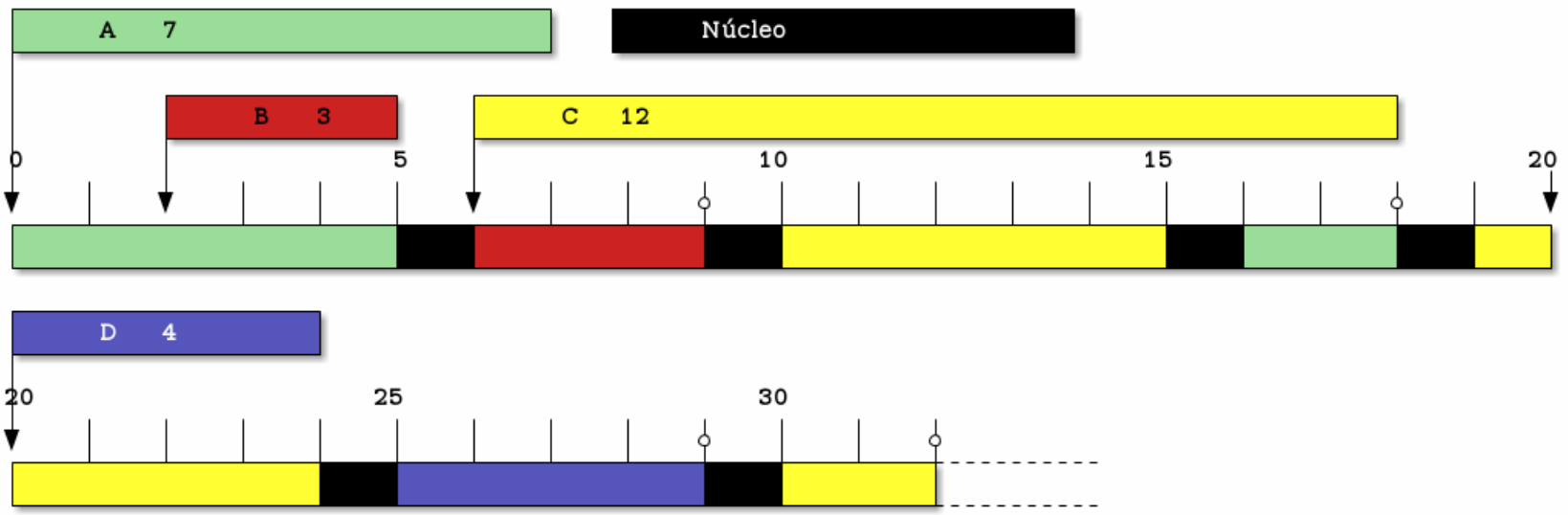

Figure: Ejecución de cuatro procesos con *quantums* de 5 *ticks* y cambios de contexto de 1 *tick*

Resolviendo nuestro ejemplo

Proceso	t	T	E	P	R
A	7				
B	3				
C	12				
D	4				
Promedio *útil*					
Núcleo	6				
Promedio total					

- Tiempo kernel
- Tiempo desocupado
- Utilización del CPU

Resultado de nuestro ejemplo

Proceso	t	T	E	P	R
A	7	18	11	2.57	0.389
B	3	7	4	2.33	0.429
C	12	26	14	2.17	0.462
D	4	9	5	2.25	0.444
Promedio *útil*	6.5	15	8.50	2.31	0.433
Núcleo	6	32	26	5.33	0.188
Promedio total	6.4	18.4	12.00	2.88	0.348

Tiempo kernel 14 *ticks*

Tiempo desocupado 0 *ticks*

Utilización del CPU 26 *ticks*

Frecuencias

Respecto al patrón de llegadas y salidas de procesos a la cola de procesos listos:

- α Frecuencia de llegada promedio
- β Tiempo de servicio requerido promedio
- ρ Valor de saturación, $\rho = \frac{\alpha}{\beta}$

Esto significa:

- $\rho = 0$ Nunca llegan procesos nuevos; el sistema estará desocupado
- $\rho = 1$ Los procesos salen al mismo ritmo al que entran
- $\rho > 1$ Los procesos llegan más rápido de lo que puede ser atendidos. La cola de procesos listos tiende a crecer. R disminuye para todos.

Índice

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

¿Cuándo se ejecuta el *despachador*?

Cuando un proceso:

1. Pasa de *ejecutando* a *en espera*
   - p.ej. por solicitar E/S, sincronización con otro proceso, *ceder el paso* (`yield`)
2. Pasa de *ejecutando* a *listo*
   - p.ej. al ocurrir una interrupción
3. Deja de estar *en espera* para estar *listo*
   - p.ej. cuando finaliza la operación E/S que solicitó
4. Pasa de *ejecutando* a *terminado*
   - Cuando finaliza su ejecución

Para la multitarea cooperativa, podrían ser sólo 1 y 4.

Nuestros procesos base

Para presentar los diferentes algoritmos, usarmos la siguiente tabla de procesos:

Proceso	Tiempo de llegada	Tiempo requerido (t)
A	0	3
B	1	5
C	3	2
D	9	5
E	12	5
Promedio		4

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Primero llegado, primero servido (*FCFS*)

- *First Come, First Serve*.
  - También referido como *FIFO* (*First In, First Out*)
- El esquema más simple de planificación
- Apto para multitarea cooperativa
- Cada proceso se ejecuta en órden de llegada
- Hasta que *suelta el control*

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Primero llegado, primero servido (*FCFS*)

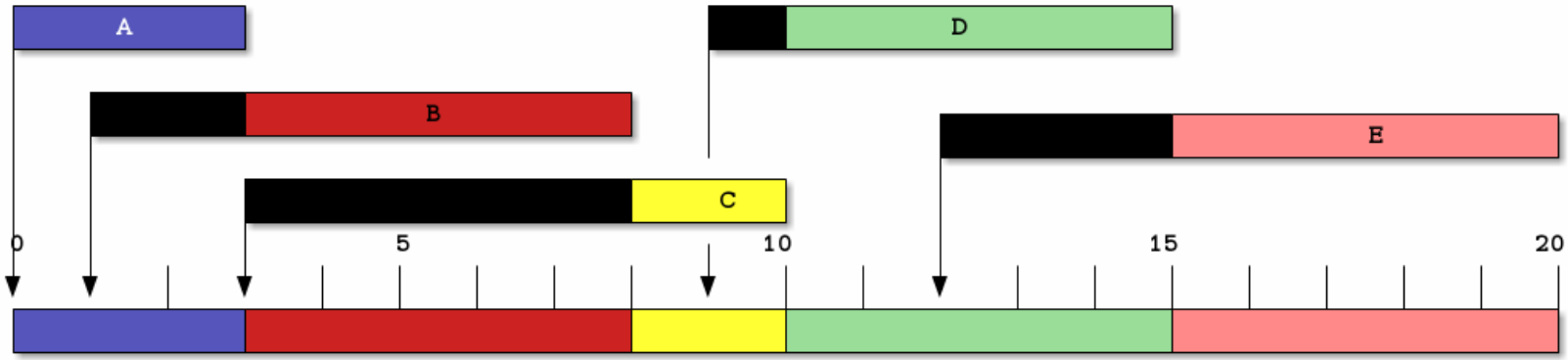

Figure: Primero llegado, primero servido (FCFS)

Primero llegado, primero servido (*FCFS*)

Proceso	Inicio	Fin	T	E	P
A	0	3	3	0	1
B	3	8	7	2	1.4
C	8	10	7	5	3.5
D	10	15	6	1	1.2
E	15	20	8	3	1.6
Promedio			6.2	2.2	1.74

Primero llegado, primero servido (*FCFS*)

- La *sobrecarga administrativa* es mínima
  - El algoritmo es extremadamente simple: una cola FIFO
  - Efectúa el mínimo posible de cambios de contexto
  - No requiere hardware de apoyo (temporizador / interrupciones)
  - $\rightarrow$ *Principio de histéresis* (Finkel): "Hay que resistirse al cambio"
- El rendimiento percibido por los últimos procesos disminuye
  - Los procesos cortos pueden esperar desproporcionadamente mucho tiempo
  - La demora aumenta fuertemente conforme crece ρ
- Tendencia a la inanición cuando $\rho \geq 1$

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ronda (*Round Robin*)

- Busca dar buena respuesta tanto a procesos cortos como largos
- Requiere multitarea preventiva
- Ejecutamos cada proceso por un *quantum*
  - Si no terminó su ejecución, se interrumpe y coloca de vuelta al final de la cola
  - Los procesos nuevos se *forman* también al final de esta misma cola

Ronda (*Round Robin*)

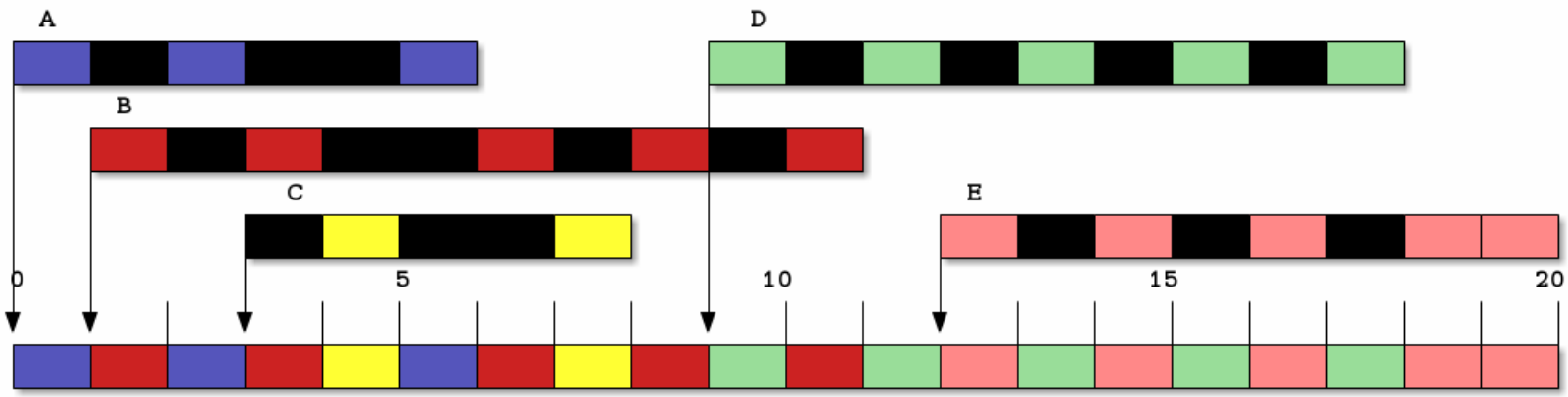

Figure: Ronda (*Round Robin*)

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ronda (*Round Robin*)

Proceso	Inicio	Fin	T	E	P
A	0	6	6	3	2.0
B	1	11	10	5	2.0
C	4	8	5	3	2.5
D	9	18	9	4	1.8
E	12	20	8	3	1.6
Promedio			7.6	3.6	1.98

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ronda (*Round Robin*)

- Alta frecuencia de cambios de contexto
  - A pesar de que el algoritmo es simple, la sobrecarga administrativa (*burocracia*) es alta
- Puede modificarse incrementando el *quantum*
  - Reduce la frecuencia de cambios de contexto
  - Para valores grandes de q, tiende a convertirse en FCFS

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ronda (*Round Robin*) con $q = 4$

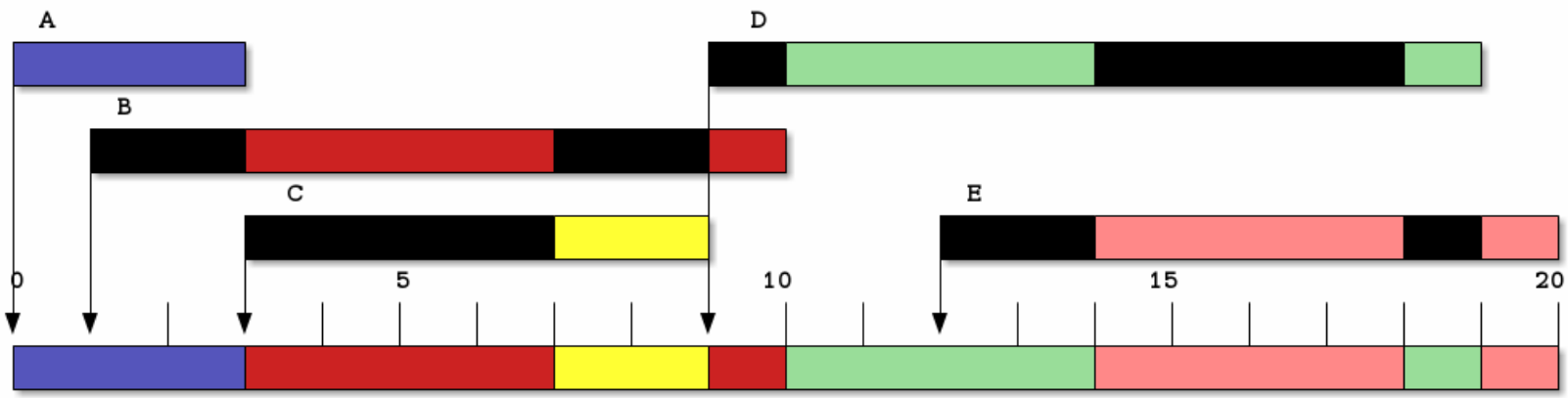

Figure: Ronda (*Round Robin*), con $q = 4$

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ronda (*Round Robin*) con $q = 4$

Proceso	Inicio	Fin	T	E	P
A	0	3	3	0	1.0
B	3	10	9	4	1.8
C	7	9	6	4	3.0
D	10	19	10	5	2.0
E	14	20	8	3	1.6
Promedio			7.2	3.2	1.88

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

El proceso más corto a continuación (SPN)

- *Shortest Process Next*
- Multitarea cooperativa
- Pero requerimos un algoritmo más *justo* que FCFS
- Sabemos cuánto tiempo va a requerir cada proceso
  - No por *magia*: Podemos estimar / predecir basados en su historia
    - Recuerden: Un proceso puede *entrar y salir* del ámbito del despachador
  - SPN puede mantener la contabilidad de los procesos incluso tras entregarlos de vuelta al agendador

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

El proceso más corto a continuación (SPN)

- *Shortest Process Next*
- Multitarea cooperativa
- Pero requerimos un algoritmo más *justo* que FCFS
- Sabemos cuánto tiempo va a requerir cada proceso
  - No por *magia*: Podemos estimar / predecir basados en su historia
    - Recuerden: Un proceso puede *entrar y salir* del ámbito del despachador
  - SPN puede mantener la contabilidad de los procesos incluso tras entregarlos de vuelta al agendador

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

SPN con tiempos *declarados*

Hace años, podía esperarse que los usuarios proporcionaran un estimado de sus tiempos de ejecución:

> *En un sistema que da alta prioridad a los procesos con estimación de tiempo corta, la política normal es terminar aquellos procesos que excedan sus límites estimados; de otro modo, los usuarios pronto arruinarían el esquema. En este caso, la mayoría de usuarios prefieren hacer predicciones conservadoras. Morris (1967) encuentra que los usuarios sobre-estimaron sus requisitos de almacenamiento por 50%, y dice que "las estimaciones en tiempo de procesamiento son mucho peores"*
>
> *–Per Brinch Hansen, 1973*

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Estimando para SPN: Promedio exponencial

Es común emplear un *promedio exponencial* para estimar la siguiente demanda de tiempo de p: Si en su última invocación empleó q quantums,

$$e'_p = fe_p + (1 - f)q$$

Donde $0 \leq f \leq 1$ es el *factor atenuante*, determinando qué tan *reactivo* será el promedio a cada cambio.
Es común que $f \approx 0.9$

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Estimando para SPN: Promedio exponencial

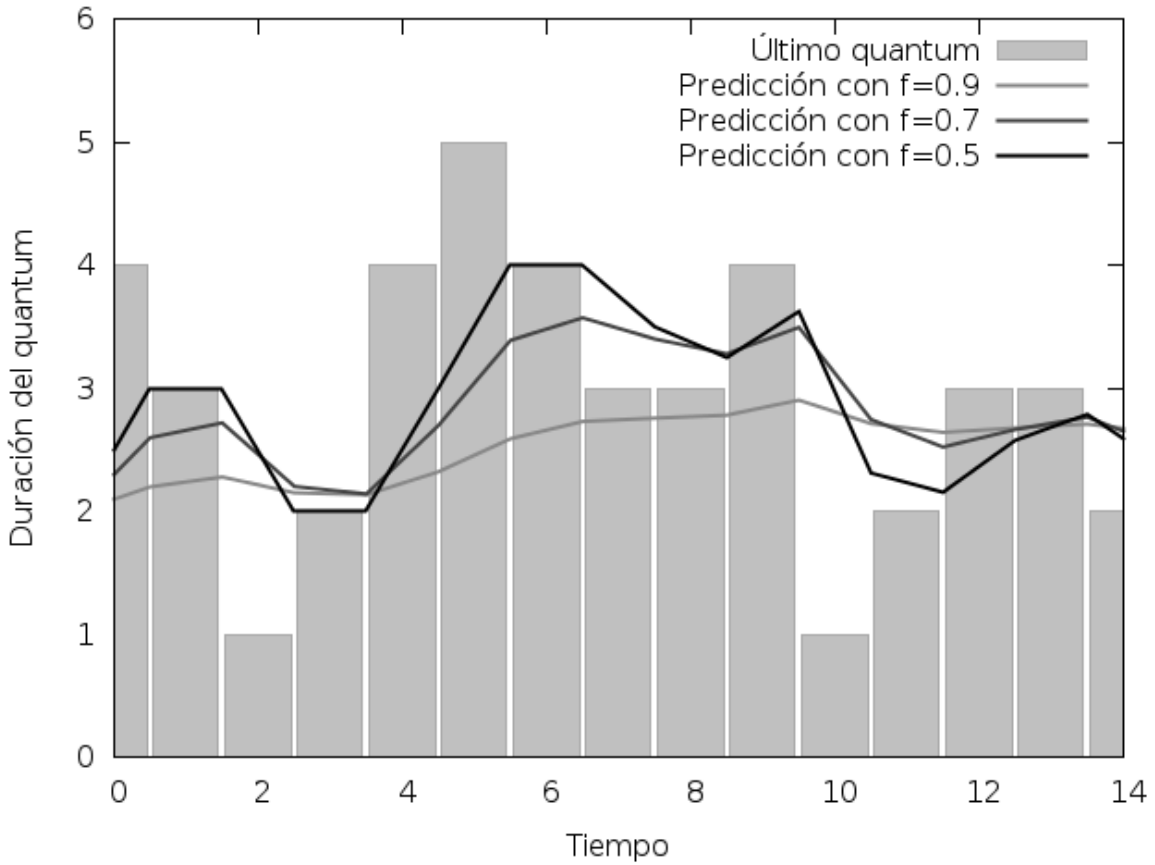

Figure: Predicción de próxima solicitud de tiempo de un proceso basado en su historia.

El proceso más corto a continuación (SPN)

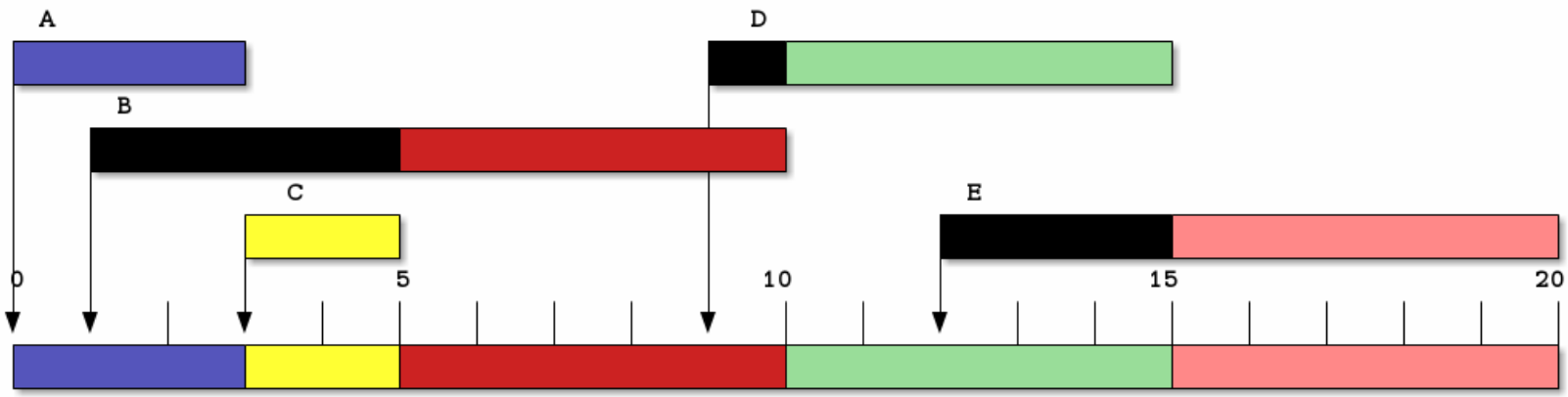

Figure: El proceso más corto a continuación (SPN)

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

El proceso más corto a continuación (SPN)

Proceso	Inicio	Fin	T	E	P
A	0	3	3	0	1.0
B	5	10	9	4	1.8
C	3	5	2	0	1.0
D	10	15	6	1	1.2
E	15	20	8	3	1.6
Promedio			5.6	1.6	1.32

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

El proceso más corto a continuación (SPN)

- Obviamente, SPN favorece a los procesos cortos
  - Un proceso largo puede esperar mucho tiempo antes de ser atendido
  - Con ρ alto, los procesos largos sufren inanición
- Con una cola de procesos listos chica, el resultado es similar a FCFS
  - Pero vimos que una sóla permutación entre el órden de B y C redujo fuertemtente los factores de penalización

Variaciones sobre SPN: SPN preventivo (PSPN)

- Emplea la estrategia de SPN, pero interrumpe cada *quantum*
- Finkel observa que la penalización a procesos largos no es mucho peor que la de la ronda
- Mantiene mejores promedios, porque los procesos cortos salen más temprano de la cola.

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Variaciones sobre SPN: El más penalizado a continuación (HPRN)

- *Highest Penalty Ratio Next*
- Multitarea cooperativa
  - Las alternativas (FCFS y SPN) parecen injustas para muchos proesos
  - Busca otorgar un mejor balance
- Todos los procesos incian con un valor de penalización $P = 1$
- Cada vez que un proceso es obligado a esperar un tiempo w por otro, $P = \frac{w+t}{t}$ (acumulando w)
- Se elige el proceso cuyo valor de P sea mayor

El más penalizado a continuación (HPRN)

- Mientras $\rho < 1$, HPRN evita inanición incluso en procesos largos
- Finkel apunta que, ante la experimentación, HPRN se ubica siempre entre FCFS y SPN
- Principal desventaja: Es un algoritmo *caro*
  - Cuando hay muchos procesos en la cola, P tiene que calcularse para todos ellos a cada invocación del despachador

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Mecanismos con múltiples colas

- Hasta ahora, se evalúa cómo ordenar los procesos en la *cola única* de procesos listos
- Dar trato diferenciado a procesos con perfiles distintos es complicado
- . . . ¿Y si montamos *distintas colas* de procesos listos?
  - Asignando determinado *patrón de comportamiento* a la migración de una cola a otra
  - Dando un *trato diferenciado* a los procesos de distintas colas

Mecanismos con múltiples colas

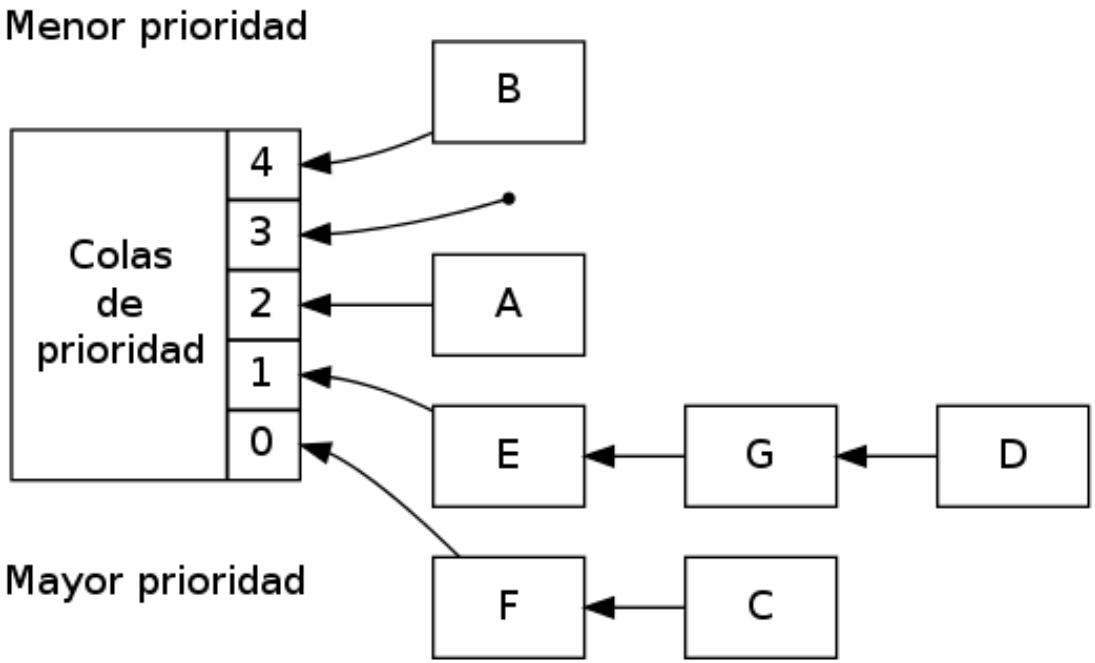

Figure: Representación de un sistema con cinco colas de prioridad y siete procesos listos

Retroalimentación multinivel (FB)

- *Multilevel Feedback*
- Multitarea preventiva
- Se crea no una, sino varias colas de procesos listos
  - Cada cola con un distinto nivel de prioridad, C_P
- El despachador toma el proceso al frente de la cola de más prioridad
- Tras n ejecuciones, el proceso es *degradado* a C_{P+1}
- Favorece a los procesos cortos
  - Terminan su trabajo sin ser marcados como de prioridad inferior
- El algoritmo es *barato*
  - Sólo hay que actualizar a un proceso a cada ejecución, y evaluar un número limitado de colas

Retroalimentación multinivel (FB)

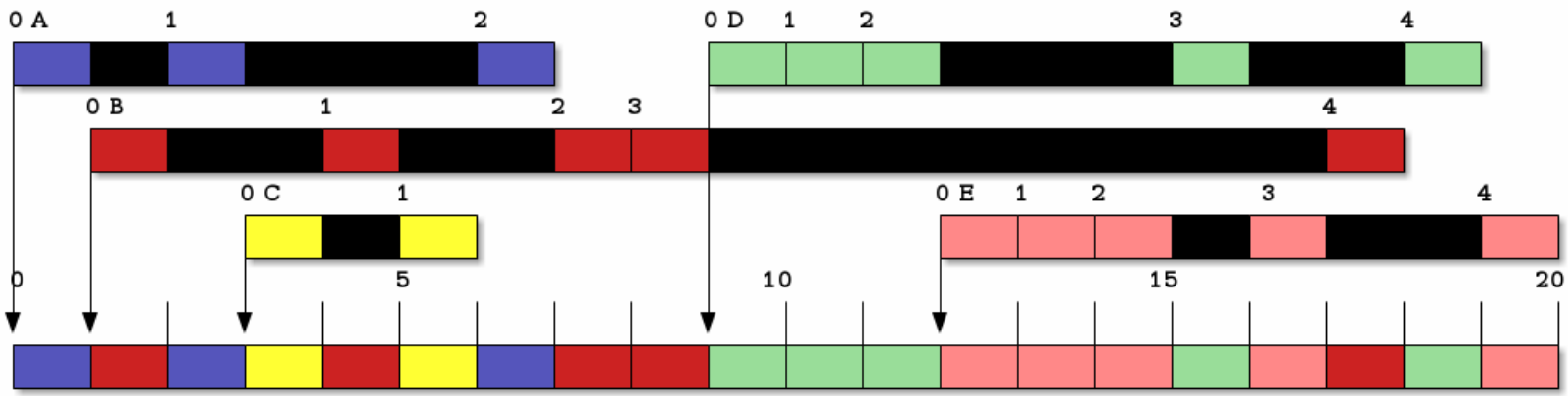

Figure: Retroalimentación multinivel (FB) básica. En la línea superior al proceso se muestra la cola antes del *quantum* en que se ejecuta.

Retroalimentación multinivel (FB)

Fenómenos observados:

Al *tick* 8, 10, 11, 13, 14, el despachador interrumpe al proceso activo *y lo vuelve a programar*

- En una implementación ingenua, esto causa un cambio de contexto
  - Burocracia innecesaria
- ¿Puede prevenirse esta interrupción?

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Retroalimentación multinivel (FB)

Proceso	Inicio	Fin	T	E	P
A	0	7	7	4	1.7
B	1	18	17	12	3.4
C	3	6	3	1	1.5
D	9	19	10	5	2.0
E	12	20	8	3	1.6
Promedio			9	5	2.04

Retroalimentación multinivel (FB)

- ¡Pero todos los números apuntan a que es una peor estrategia que las anteriores!
- Los únicos beneficiados son los recién llegados
  - Entran a la cola de mayor prioridad
  - Un proceso largo, a mayor ρ, enfrenta inanición
- El rendimiento del algoritmo puede ajustarse con dos variables básicas:
  - n Cuántas ejecuciones para ser *degradado* a C_{P+1}
  - Q Duración del *quantum* de las siguientes colas
- Veamos cómo se comporta cuando:
  - Mantenemos $n = 1$
  - $Q = 2^{nq}$ (donde q es la duración del *quantum* base)

Retroalimentación multinivel (FB)

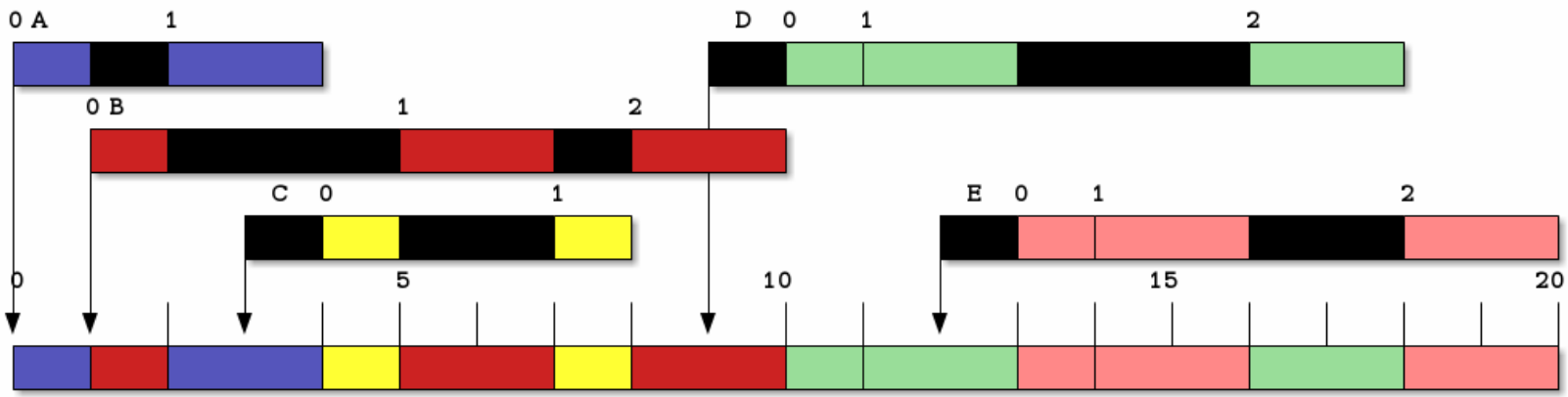

Figure: Retroalimentación multinivel (FB) con Q exponencial

Retroalimentación multinivel (FB)

Fenómenos observados:

- Aunque FB favorece a los procesos recién llegados, al *tick* 3, 9 y 10 los procesos que llegan son puestos en espera
  - Llegaron a la mitad del *quantum* largo de otro proceso

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Retroalimentación multinivel (FB)

Proceso	Inicio	Fin	T	E	P
A	0	4	4	1	1.3
B	1	10	9	4	1.8
C	4	8	5	3	2.5
D	10	18	9	4	1.8
E	13	20	8	3	1.6
Promedio			7	3	1.8

Retroalimentación multinivel (FB)

- Con Q exponencial, los promedios resultan incluso mejores que *ronda*
  - Típicamente los incrementos son más suaves — $Q = nq$ o incluso $q = q\log(n)$
  - Un proceso largo con Q exponencial puede causar inanición por largo tiempo
- Para evitar la inanición ante un ρ alto, puede considerarse la retroalimentación en sentido inverso
  - Si un proceso largo es degradado a C_P y pasa demasiado tiempo sin ejecutarse, promoverlo de vuelta a C_{P-1}

Retroalimentación multinivel (FB)

- El mecanismo es muy flexible, y permite muchas mejorías simples
- Hoy en día es empleado por muchos de los principales sistemas operativos
  - FreeBSD, Linux (pre-2.6), MacOS X, NetBSD, Solaris, Windows (2000 en adelante) (ref: Wikipedia “Scheduling algorithm”)
  - Con diferentes parámetros y prioridades

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ronda egoísta (SRR)

- *Selfish Round Robin*
- Multitarea preventiva
- Favorece a los proesos que *ya llevan tiempo ejecutando* sobre los recién llegados
  - Un proeso nuevo se forma en la cola de *procesos nuevos*, el despachador avanza sólo sobre los *procesos aceptados*
- Parámetros ajustables:
  - a Ritmo de incremento de prioridad de procesos *aceptados*
  - b Ritmo de incremento de prioridad de procesos *nuevos*
- Cuando la prioridad de un proceso nuevo *alcanza* a la de uno aceptado, éste se acepta.

Ronda egoísta (SRR)

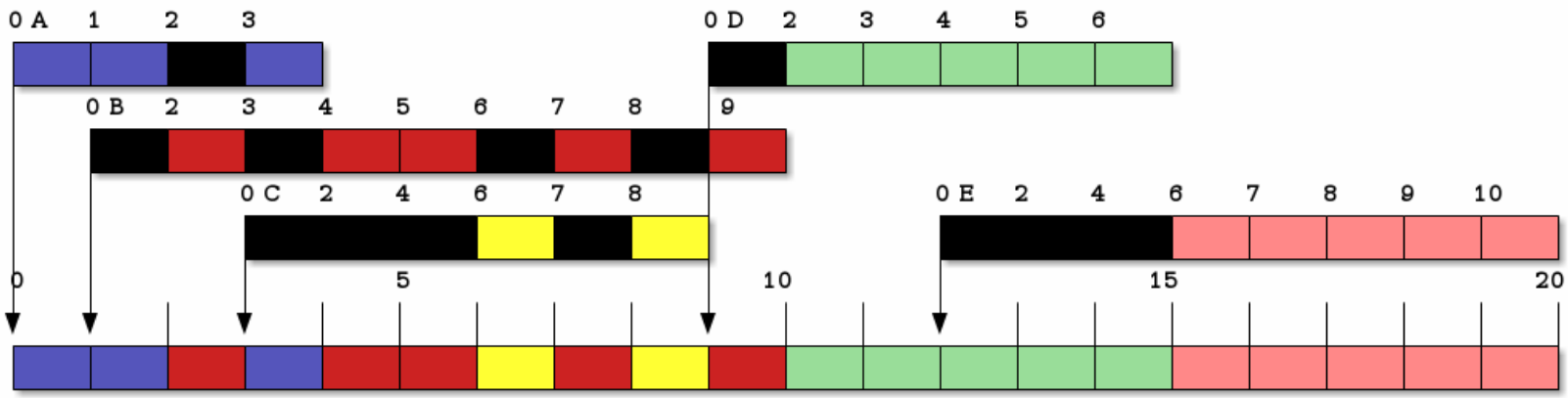

Figure: Ronda egoísta (SRR) con $a = 2$ y $b = 1$

Ronda egoísta (SRR)

Proceso	Inicio	Fin	T	E	P
A	0	4	4	1	1.3
B	2	10	9	4	1.8
C	6	9	6	4	3.0
D	10	15	6	1	1.2
E	15	20	8	3	1.6
Promedio			6.6	2.6	1.79

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ronda egoísta (SRR)

- Mientras $\frac{b}{a} < 1$:
  - Los procesos nuevos serán aceptados eventualmente
  - Si el control va alternando entre dos procesos, su prioridad se mantendrá igual, y serán despachados por ronda simple
- Si $\frac{b}{a} \geq 1$, el proceso en ejecución terminará antes de que se acepte el nuevo $\rightarrow$ Tiende a FCFS
- Si $\frac{b}{a} = 0$ (esto es, si $b = 0$)
  - Los procesos recién llegados son aceptados inmediatamente $\rightarrow$ Tiende a ronda
- Si $0 < \frac{b}{a} < 1$, la ronda es *relativamente egoísta*
  - Se da entrada a procesos nuevos
  - Incluso si hay procesos muy largos ejecutando

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Clasificando a los distintos esquemas

Los siete algoritmos presentados pueden caracterizarse sobre dos descriptores primarios

Tipo de multitarea si el esquema está planteado para operar bajo multitarea *preventiva* o *cooperativa*

Emplea información *intrínseca* Si, para tomar cada decsión de planificación, emplean información propia (intrínseca) a los procesos evaluados, o no — Esto es, si el historial de ejecución de un proceso tiene impacto en cómo será planificado a futuro.

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Clasificando a los distintos esquemas

Table: Caracterización de los mecanismos de planificación a corto plazo

	No considera intrínseca	**Considera intrínseca**
Cooperativa	Primero llegado primero servido (FCFS)	Proceso más corto (SPN), Proceso más penalizado (HPRN)
Preventiva	Ronda (RR)	Proceso más corto preventivo (PSPN), Retroalimentación (FB), Ronda egoísta (SRR)

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Índice

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Esquemas híbridos

- Los esquemas de planificación empleados normalmente usan *mezclas* de los algoritmos presentados
- Permite emplear el algoritmo que más ventajas presente *ante una situación dada*
  - Y evitar algunas de sus deficiencias

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Esquemas híbridos: Algoritmo por cola en FB

- Manejamos varias colas en un esquema FB
- Cada cola usa internamente un algoritmo distinto para elegir el proceso que está *a la cabeza*. Algunas ideas como ejemplo:
  - Una cola bajo PSPN: *Empuja* a los procesos más largos hacia colas que sean interrumpidas con menor frecuencia
  - Emplear SRR para las colas de menor prioridad
    - Sus procesos ya esperaron mucho para tener respuesta; cuando obtienen el procesador, avanzan lo más ágilmente posible
    - Pero no obstaculizan a los procesos cortos que van llegando

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Esquemas híbridos: Dependientes del estado del sistema

- Podemos considerar también información *extrínseca* para despachar
  - Información *externa* al estado y ejecución de cada uno de los procesos
  - Información dependiente del estado del sistema, del tipo de usuario, etc.
- A continuación, algunos ejemplos

Preventiva o cooperativa, dependiendo de ρ

- Si los procesos son *en promedio* cortos y $\rho < 1$
  - Métodos con la mínima sobrecarga administrativa (FCFS o SPN)
  - O un RR con *quantum* muy largo (evitando los problemas de la multitarea cooperativa)
- Si los procesos tienden a ser más largos o si sube ρ
  - Cambiamos a RR con un *quantum* más bajo o a PSPN

Ronda con *quantum* dependiente de procesos pendientes

- Esquema simple de ronda
- La duración de un *quantum* es ajustada periódicamente
- Cada *quantum* depende de la cantidad de procesos en el total de procesos listos, siguiendo $Q = \frac{q}{n}$
- Pocos procesos esperando
  - Mayor $Q \rightarrow$ Menos cambios de contexto
- Muchos procesos esperando
  - Menor Q
  - Nunca más allá de un minimo, para evitar sobrecarga burocrática

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ronda + Prioridad externa

- Usamos un esquema simple de ronda, con una sola cola
- La duración del *quantum* dependerá de la prioridad *externa*
  - Fijada por el usuario o por el sistema por factores *ajenos* al despachador
- Un proceso de mayor prioridad ejecutará por mayor tiempo

Peor servicio a continuacion (WSN)

- Generalización sobre HPRN
- No sólo se considera penalización el tiempo esperado en la cola de procesos listos
  - Veces que ha sido interrumpido por el temporizador
  - Prioridad externa
  - Espera por E/S u otros recursos
- El proceso que ha *sufrido peor servicio* es seleccionado para su ejecución
- Desventaja: Considerar demasiados factores (con distintos pesos) impacta en el tiempo de ejecución del algoritmo
  - Puede llamarse a WSN periódicamente para formar colas
  - Proceder con esquemas más simples
  - . . . Aunque esto reduce la velocidad de reacción

Lindura (*niceness*)

- Empleado por varios Unixes históricos
- El usuario inicia (`nice`) o modifica (`renice`) la prioridad de su proceso
  - Típicamente sólo *hacia arriba* — Se porta *más lindo*.
- Esta prioridad *externa* y el tiempo consumido recientemente por el proceso constituyen una prioridad *interna*
- La prioridad interna aumenta cuando el proceso espera
  - Por el despachador, por E/S, o cualquier otra causa
- La prioridad interna es matizada por el tamaño de la cola de procesos listos
  - Entre más procesos pendientes, mayor el peso que *modifique* a la prioridad

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

. . . ¡Hora de otra tarea!

- Implementar y comparar los algoritmos *más sencillos*
  - En su lenguaje favorito
  - FCFS, RR (¿duración de quantum?), SPN
  - ¿Quieren divertirse? FB, SRR, alguno más
- Bajo *algunas* cargas, no sólo sobre una carga ejemplo
  - Generadas de forma aleatoria
- Presentar los resultados de varias ejecuciones
  - Unos cinco resultados, para poder comparar un poco las tendencias
  - ¡Verifiquen manualmente algunos de los resultados!
- (Y claro: Presentar el código)
- Entrega vía plataforma: *una semana desde hoy*

Gunnar Wolf • sistop@gwolf.org • http://sistop.gwolf.org
Sistemas Operativos (1554) • Facultad de Ingeniería UNAM

Ejemplo de resultados

```
$ compara_planif
- Primera ronda:
  A: 0, t=3; B: 1, t=5; C: 3, t=2; D: 9, t=5; E: 12, t=5
       (tot:20)
  FCFS: T=6.2, E=2.2, P=1.74
  RR1:  T=7.6, E=3.6, P=1.98
  RR4:  T=7.2, E=3.2, P=1.88
  SPN:  T=5.6, E=1.6, P=1.32
- Segunda ronda
  A: 0, t=5; B: 3, t=3; C: 3, t=7; D: 7, t=4; E:8, t=4 (tot:23)
  (...)
```