



Escalonamento em Sistemas de Tempo Real

Walter Fetter Lages

w.fetter@ieee.org

Universidade Federal do Rio Grande do Sul

Escola de Engenharia

Departamento de Engenharia Elétrica

ENG04008 Sistemas de Tempo Real



Introdução

- Ordenação do uso dos recursos do sistema
- Previsibilidade do comportamento do sistema no pior caso
- Interleaving
 - 5 processos, não preemptivo, única CPU => 120 modos diferentes
- Preemptivo
- Não preemptivo
- Estático
- Dinâmico

Modelo Simplificado de Processo



- Conjunto de processos fixo
- Processos periódicos
 - Períodos conhecidos
- Processos independentes
- Overheads do sistema desprezados
- Deadline dos processo igual ao período
- Pior caso do tempo de execução dos processos fixo
- Instante crítico -> todos os processos liberados no mesmo instante



Notação

B tempo de processo bloqueado (pior caso)

C tempo de computação (pior caso)

D deadline

I tempo de interferência

J jitter de liberação

N número de processos

P prioridade do processo

R tempo de resposta do processo (pior caso)

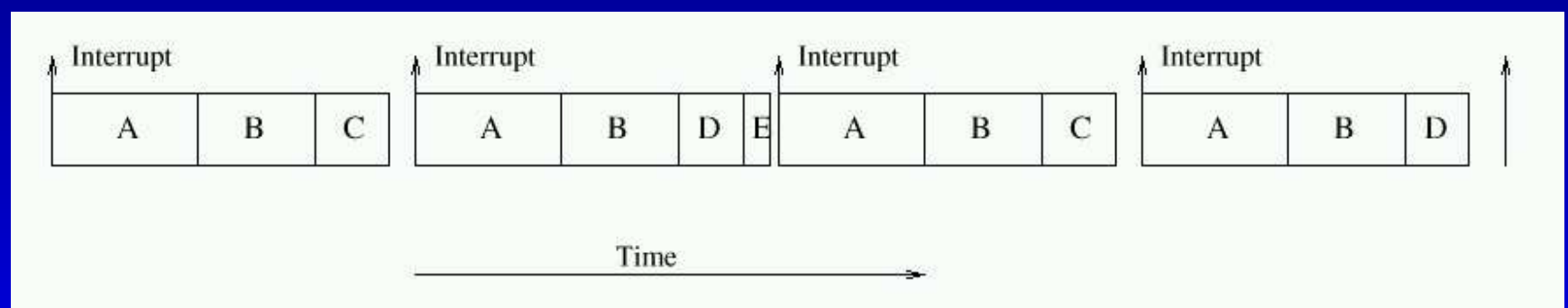
T tempo mínimo entre liberações (período)

U tempo de utilização do processo (C/T)

Execução Cíclica



Processo	Período	Tempo de Computação
A	25	10
B	25	8
C	50	5
D	50	4
E	100	2





Execução Cíclica

- Ciclos
 - Principal
 - Secundário
 - Seqüência de chamadas de subrotinas
- Não existe acesso concorrente à variáveis
- Período de "processo" = $n \times \text{ciclo secundário}$
 - Dificuldades para processos esporádicos
 - Problemas com processos com períodos longos
 - É difícil construir executivo cíclico
 - Processos com tempo de computação grandes terão que ser divididos em processo menores

Escalonamento de Processos



- Preemptivo
- Não preemptivo
- Preenpção retardada
- Despacho cooperativo
- Prioridade
 - Baseada nas propriedades temporais do processo



Rate Monotonic

- Quanto menor o período do processo, maior a prioridade
- Escalonamento ótimo para $D = T$
 - Se um conjunto de processos pode ser escalonado com um esquema de prioridades fixas, este conjunto de processos também pode ser escalonado com rate monotonic

Processo	Período	Prioridade
A	25	5
B	60	3
C	42	4
D	105	1
E	75	2



Teste de Utilização

- Prioridades atribuídas com rate monotonic
- Suficiente, mas não necessário
- Limite de utilização = 69.3%

$$\sum_{i=1}^N \left(\frac{C_i}{T_i} \right) < N \left(2^{1/N} - 1 \right)$$

N	Limite
1	100%
2	82.8%
3	78.0%
4	75.7%
5	74.3%
10	71.8%



Exemplo

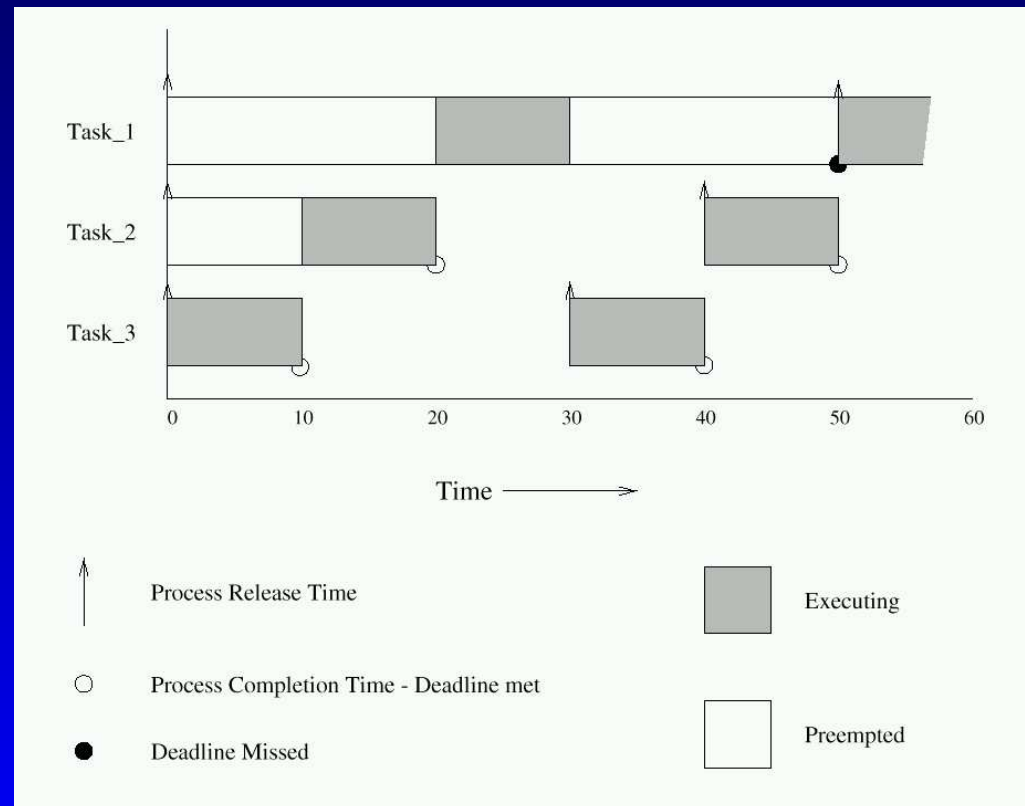
- Utilização total=82%
- Limite para três processos=0.78
- Falha no teste

	T	C	P	U
Task 1	50	12	1	0.24
Task 2	40	10	2	0.25
Task 3	30	10	3	0.33



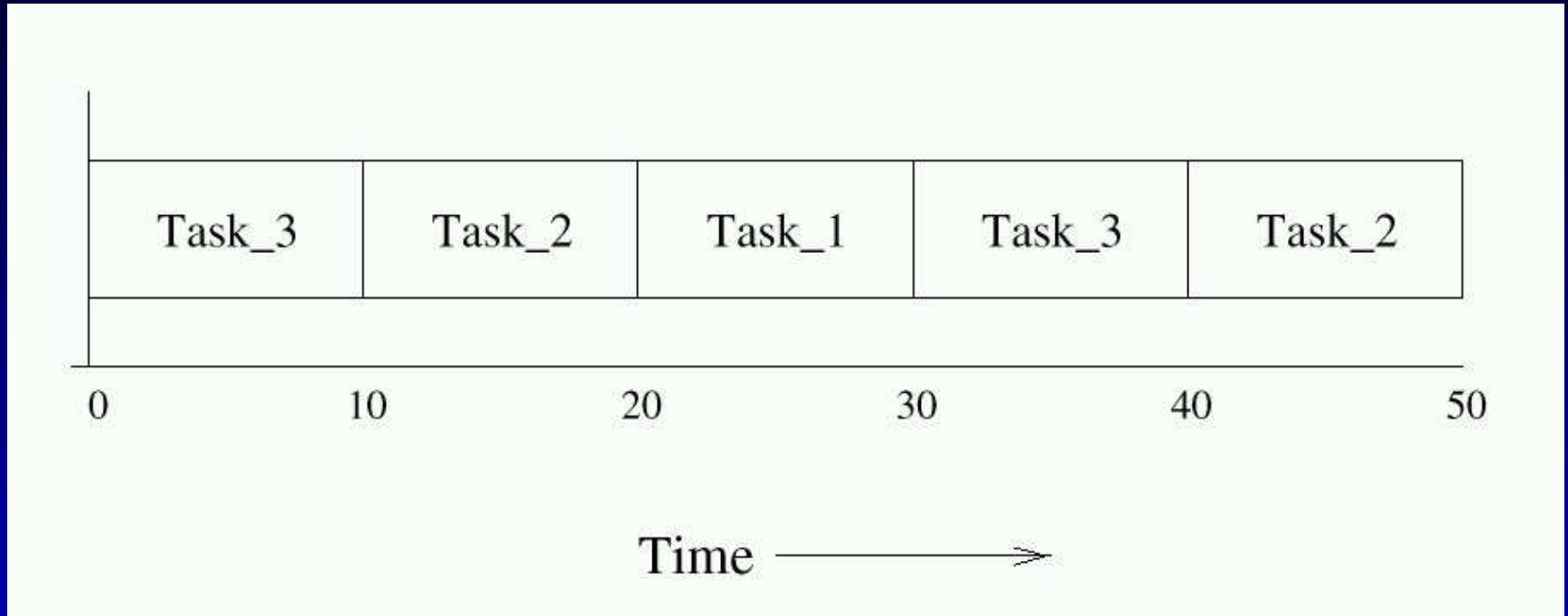
Time-line

- No instante 50, Task 1 executou apenas 10, quando necessitava executar 12
- perda de deadline





Gantt Chart





Teste Baseado em Time-line

- Time-line pode ser utilizada para determinar a escalonabilidade
- Quão longa precisa ser a time-line?
 - Se todos os processos tem o mesmo instante de liberação (possuem um instante crítico em comum), uma time-line igual ao período máximo é suficiente
 - Portanto, se todos os processo cumprirem sua primeira deadline, cumprirão todas as futuras



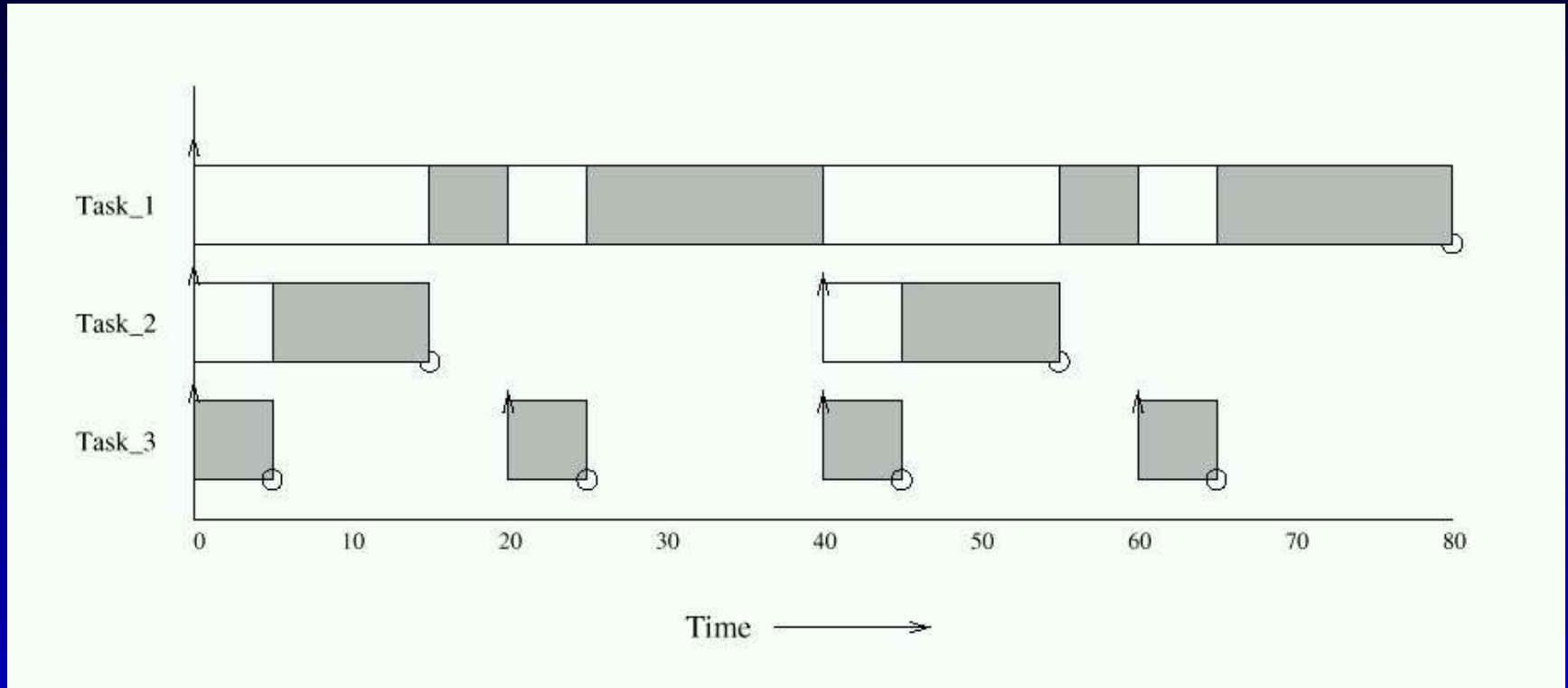
Contra-Exemplo

- Utilização=100%
- Obviamente falha no teste de utilização

	T	C	P	U
Task 1	80	40	1	0.50
Task 2	40	10	2	0.25
Task 3	20	5	3	0.25



Contra-Exemplo



Análise de Tempo de Resposta



- Teste de utilização não fornece indicação do tempo de resposta de cada processo
- Para o processo de maior prioridade, $R = C$
- Os demais, sofrerão interferência dos processos de maior prioridade

$$R_i = C_i + I_i$$

Interferência entre Processos



- Processo j com prioridade maior do que o processo i

$$\text{Número de Liberações} = \left\lceil \frac{R_i}{T_j} \right\rceil$$

$$\text{Interferência Máxima} = \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$I_i = \sum_{j \in \text{hp}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Cálculo do Tempo de Resposta



$$R_i = C_i + \sum_{j \in \text{hp}(i)} \left[\frac{R_i}{T_j} \right] C_j$$

$$\omega_i^{n+1} = C_i + \sum_{j \in \text{hp}(i)} \left[\frac{\omega_i^n}{T_j} \right] C_j$$



Exemplo

	Período	Tempo de Computação	Prioridade
Task 1	7	3	3
Task 2	12	3	2
Task 3	20	5	1

- $R_1 = 3$
- $\omega_2^0 = 3$
- $\omega_2^1 = 3 + \lceil 3/7 \rceil 3 = 6$
- $\omega_2^2 = 3 + \lceil 6/7 \rceil 3 = 6 = R_2$

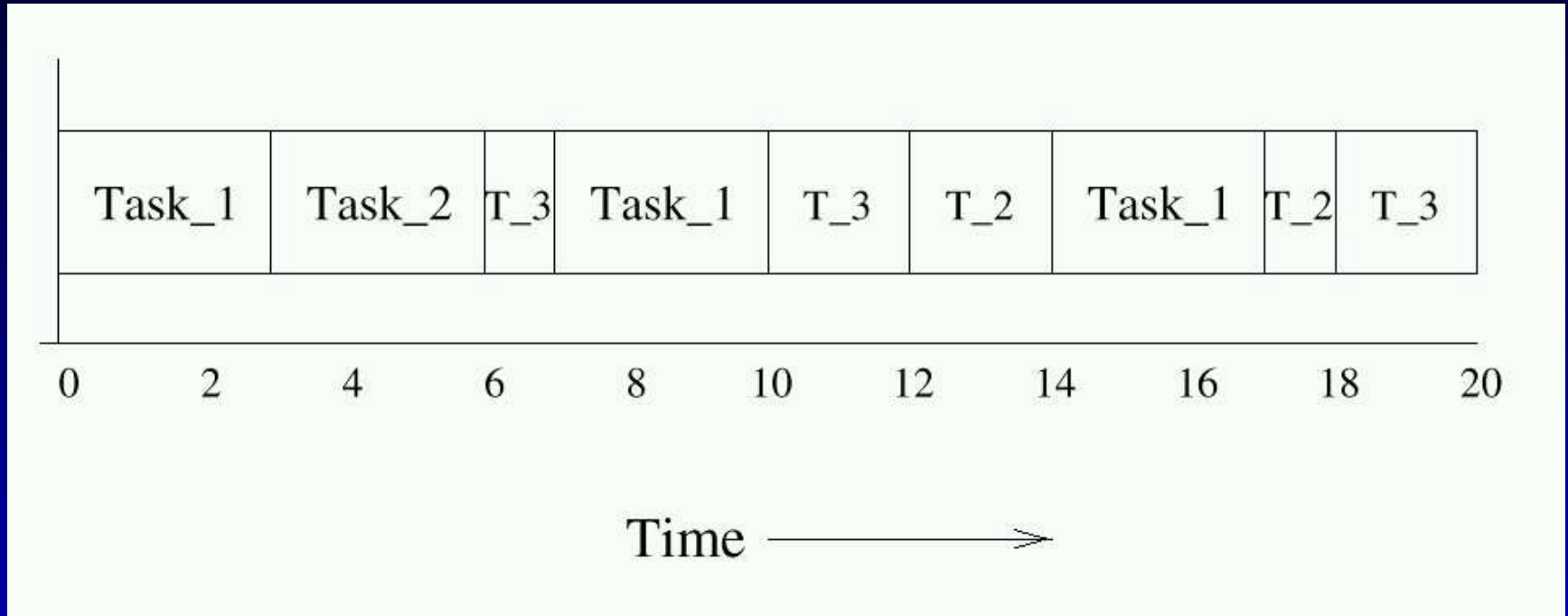


Exemplo

- $\omega_3^0 = 5$
- $\omega_3^1 = 5 + \lceil 5/7 \rceil 3 + \lceil 5/12 \rceil 3 = 11$
- $\omega_3^2 = 5 + \lceil 11/7 \rceil 3 + \lceil 11/12 \rceil 3 = 14$
- $\omega_3^3 = 5 + \lceil 14/7 \rceil 3 + \lceil 14/12 \rceil 3 = 17$
- $\omega_3^4 = 5 + \lceil 17/7 \rceil 3 + \lceil 17/12 \rceil 3 = 20$
- $\omega_3^5 = 5 + \lceil 20/7 \rceil 3 + \lceil 20/12 \rceil 3 = 20 = R_3$



Exemplo



Pior-caso do Tempo de Execução



- Medição
 - Dificuldade para garantir que o pior-caso foi observado
- Análise
 - Dificuldade para modelar o processador
 - Cache
 - Pipeline
 - Wait-states
 - Influência do compilador



Processos não Periódicos

- Processos esporádicos
 - T é interpretado como o período mínimo
 - $D < T$
 - Algoritmo de cálculo de R modificado para ter como critério de parada $\omega_i^{wn+1} > D_i$
- Processos aperiódicos
 - T é interpretado como período médio



Deadline Monotonic

- Quanto menor o deadline, maior a prioridade
- Escalonamento ótimo para $D < T$

	T	D	C	P	R
Task 1	20	5	3	4	3
Task 2	15	7	3	3	6
Task 3	10	10	4	2	10
Task 4	20	20	3	1	20

Prova de que DMPO é Ótimo



- DMPO é ótimo se para qualquer conjunto de processo Q , escalonável por um esquema de prioridades Ω , Q também é escalonável por DMPO
- Se DMPO é ótimo, será possível transformar as prioridades de Q segundo Ω , nas prioridades segundo DMPO, preservando a escalonabilidade



Prova de Otimalidade

- Sejam i e j dois processos com prioridades adjacentes, tais que utilizando Ω
 - $P_i > P_j$ e $D_i > D_j$
- Definindo-se Ω' tal que a ordem de i e j é invertida, tem-se
 - Processos com prioridade $> P_i$ não são afetados
 - Processos com prioridade $< P_j$ não são afetados
 - O processo j , agora possui maior prioridade e portanto é escalonável utilizando-se Ω'
 - É necessário provar que i ainda é escalonável



Escalonabilidade de i

- Utilizando-se Ω
 - $R_j \leq D_j, D_j < D_i, D_i \leq T_i$
 - O processo i interfere apenas uma vez durante a execução de j
- Utilizando-se Ω'
 - R_i passa a ser R_j utilizando-se Ω , pois $C_j + C_i$ continua sendo o mesmo
 - O processo j interfere apenas uma vez durante a execução de i
 - $R'_i = R_j \leq D_j < D_i$, portanto i é escalonável
- Por repetição, DMPO é escalonável e ótimo



Interferência entre Processos

- Em geral, processos não são independentes
 - Comunicação e sincronismo causam bloqueamento de processos
- Inversão de prioridade
 - Quando um processo é bloqueado esperando por um evento gerado por um processo de menor prioridade
 - Exemplo:
 - L_1 e L_4 compartilham uma seção crítica Q
 - L_3 e L_4 compartilham outra seção crítica V

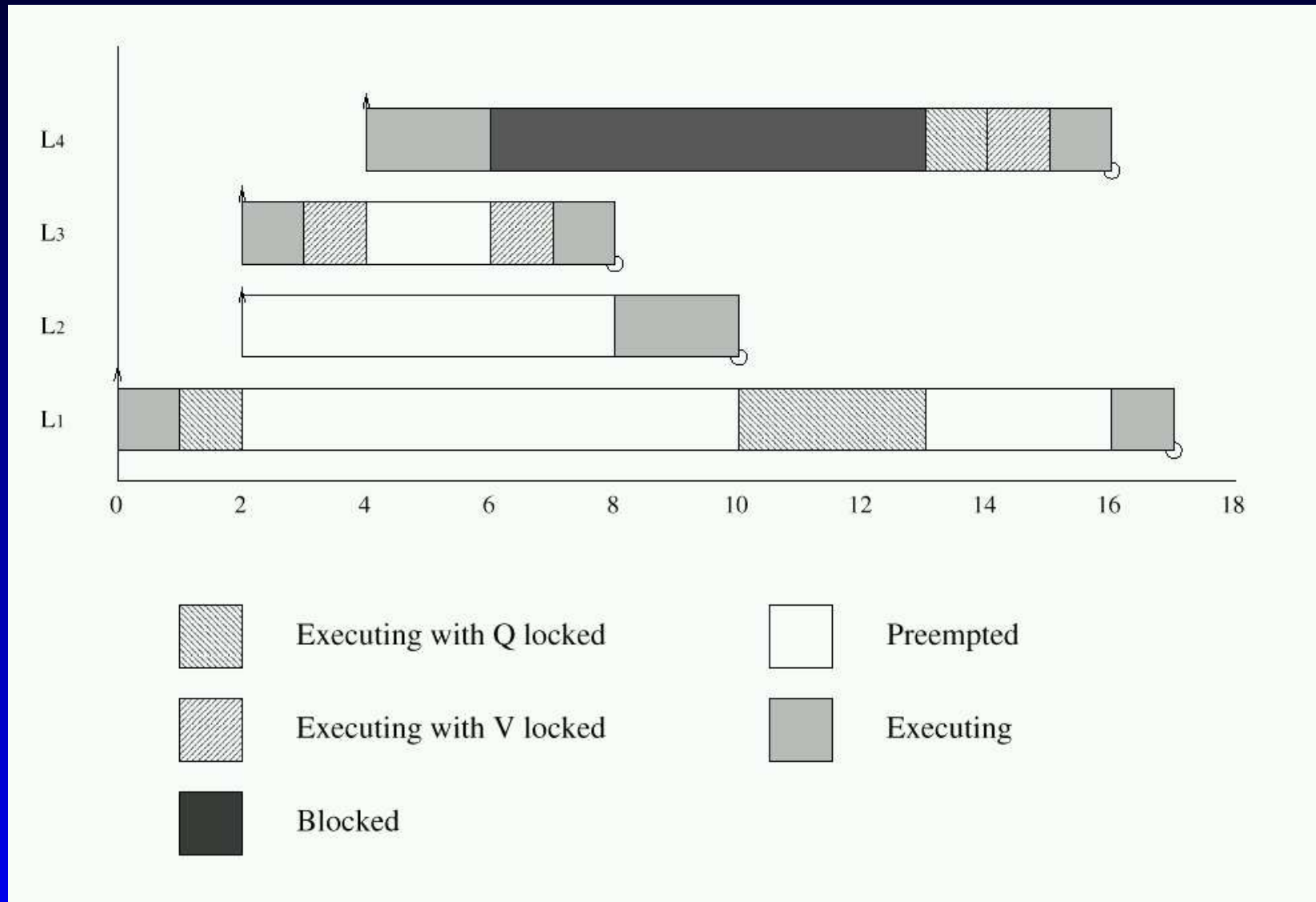


Exemplo

Processo	P	Seqüência de Execução	Liberação
L_4	4	EEQVE	4
L_3	3	EVVE	2
L_2	2	EE	2
L_1	1	EQQQQE	0

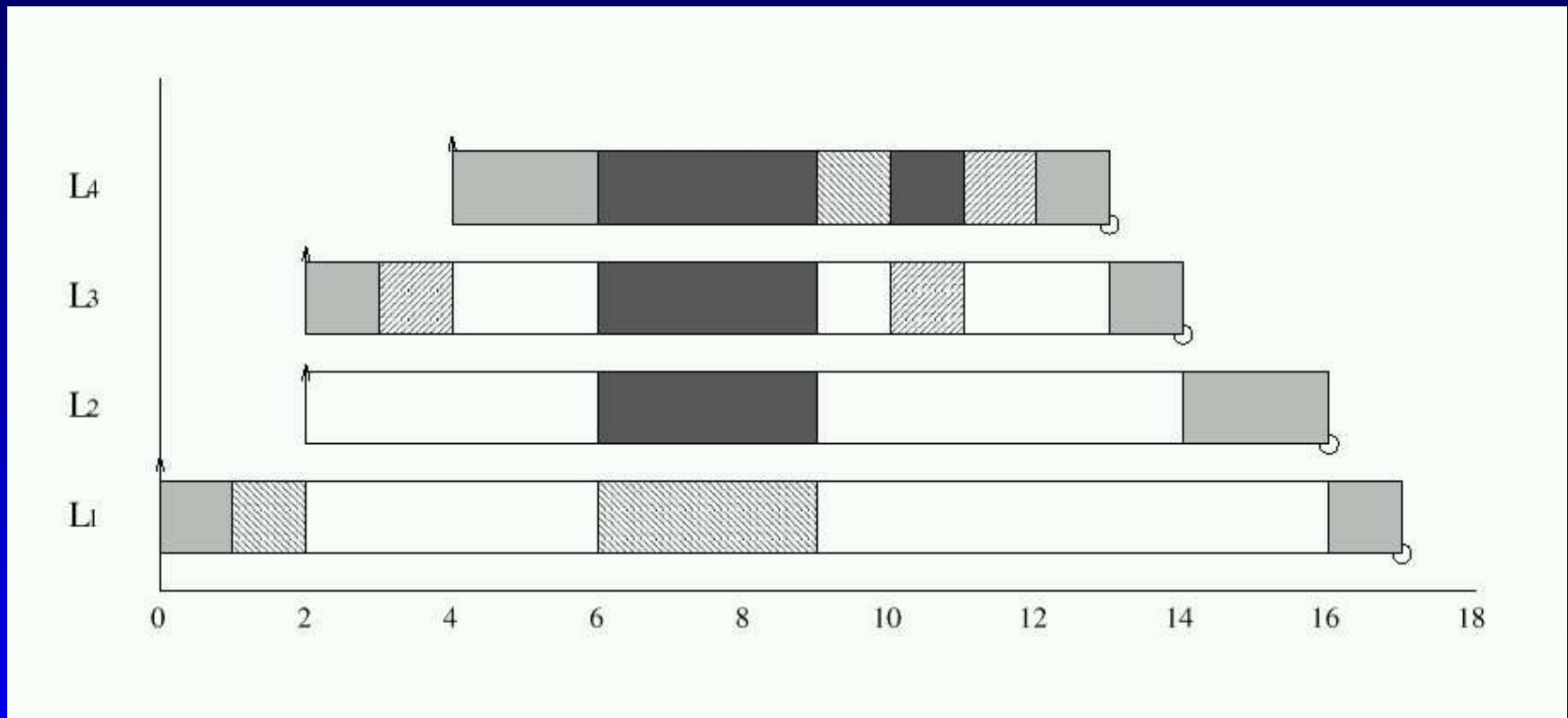


Exemplo



Herança de Prioridade

- Se um processo p é suspenso esperando por um processo q , de menor prioridade, \Rightarrow prioridade de $q =$ prioridade de p



Cálculo do Tempo de Bloqueio



- $usage(k, i) = 1$ se o recurso k é usado por pelo menos um processo de prioridade $< i$ e pelo menos um processo de prioridade $\geq i$ e 0 caso contrário
- $CS(k)$ = tempo de execução da seção crítica

$$B_i = \sum_{k=1}^K usage(k, i)CS(k)$$

Tempo de Resposta com Bloqueio



$$R_i = C_i + B_i + \sum_{j \in \text{hp}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Protocolos de Priority Ceiling



- Herança de prioridade
 - Cálculo de pior-caso pessimista
 - Bloqueamento transiente
 - Possibilidade de deadlock
- Priority ceiling
 - Um processo de alta prioridade pode ser bloqueado uma única vez por um processo de menor prioridade
 - Bloqueamento transiente e deadlocks são prevenidos
 - É garantida a exclusão mútua no acesso aos recursos

Original Ceiling Priority Protocol



- Cada processo tem uma prioridade estática
- Cada recurso tem um valor de ceiling
 - $\text{Máx}\{\text{prioridade dos processos que o acessam}\}$
- Cada processo tem uma prioridade dinâmica
 - $\text{Máx}\{\text{prioridade estática, prioridade herdada}\}$
- Um processo pode reservar um recurso somente se a sua prioridade dinâmica é maior do que o ceiling de qualquer recurso já reservado por outros processos



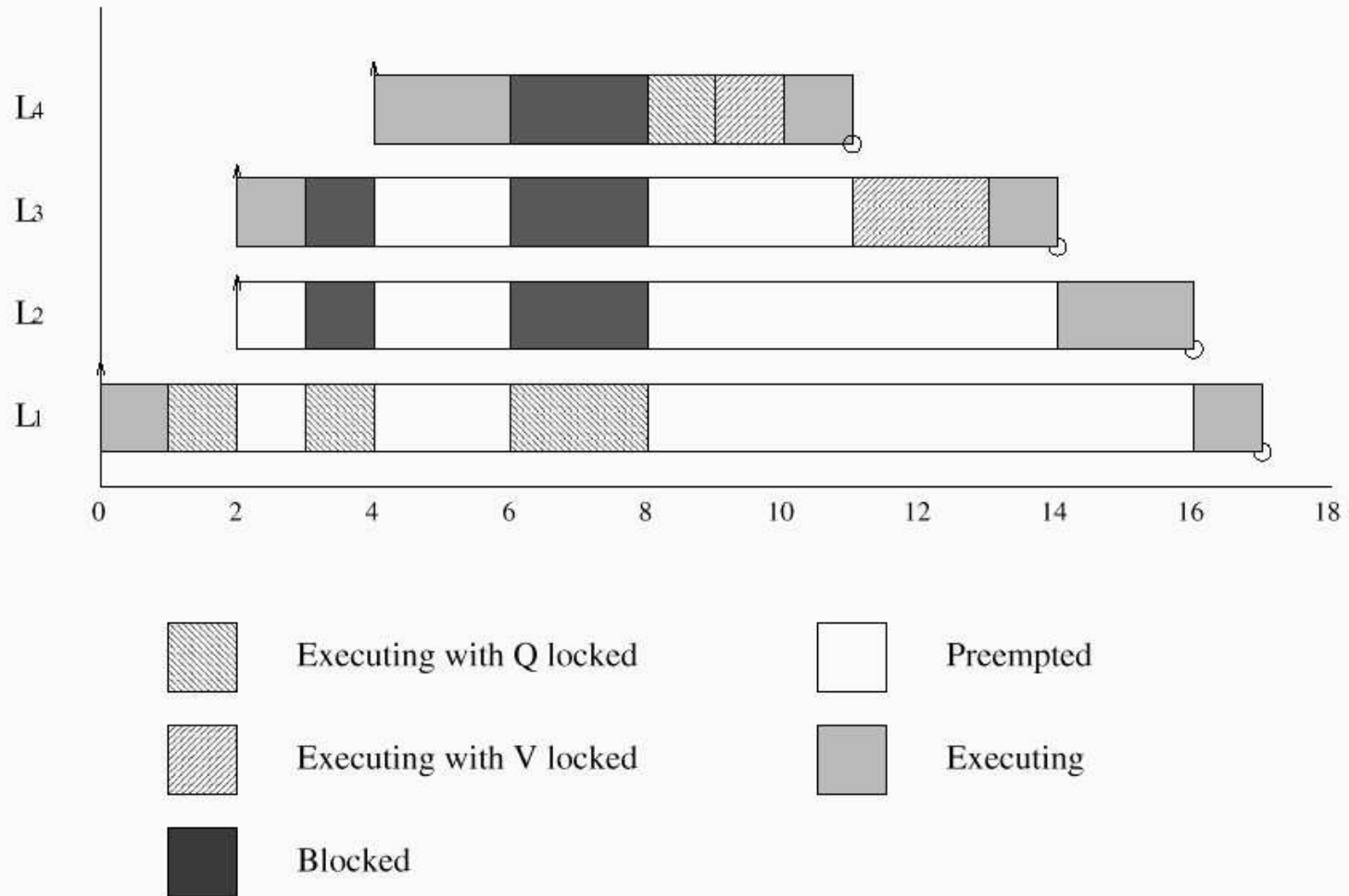
OCPP

- O primeiro recurso sempre será alocado
- Um segundo recurso só será reservado se não existir um processo de prioridade maior que use o mesmo recurso
- Processos de alta prioridade só serão bloqueados uma única vez por processo de baixa prioridade

$$B_i = \max_{k=1}^K \text{usage}(k, i) \text{CS}(k)$$



Exemplo





Immediate Ceiling Priority Protocol

- Cada processo tem uma prioridade estática
- Cada recurso tem um valor de ceiling
 - $\text{Máx}\{\text{prioridade dos processos que o acessam}\}$
- Cada processo tem uma prioridade dinâmica
 - $\text{Máx}\{\text{prioridade estática, ceiling dos recursos reservados}\}$
- Os processo serão bloqueados apenas no início de sua execução
 - Quando o processo realmente começar a executar todos os recursos estarão disponíveis

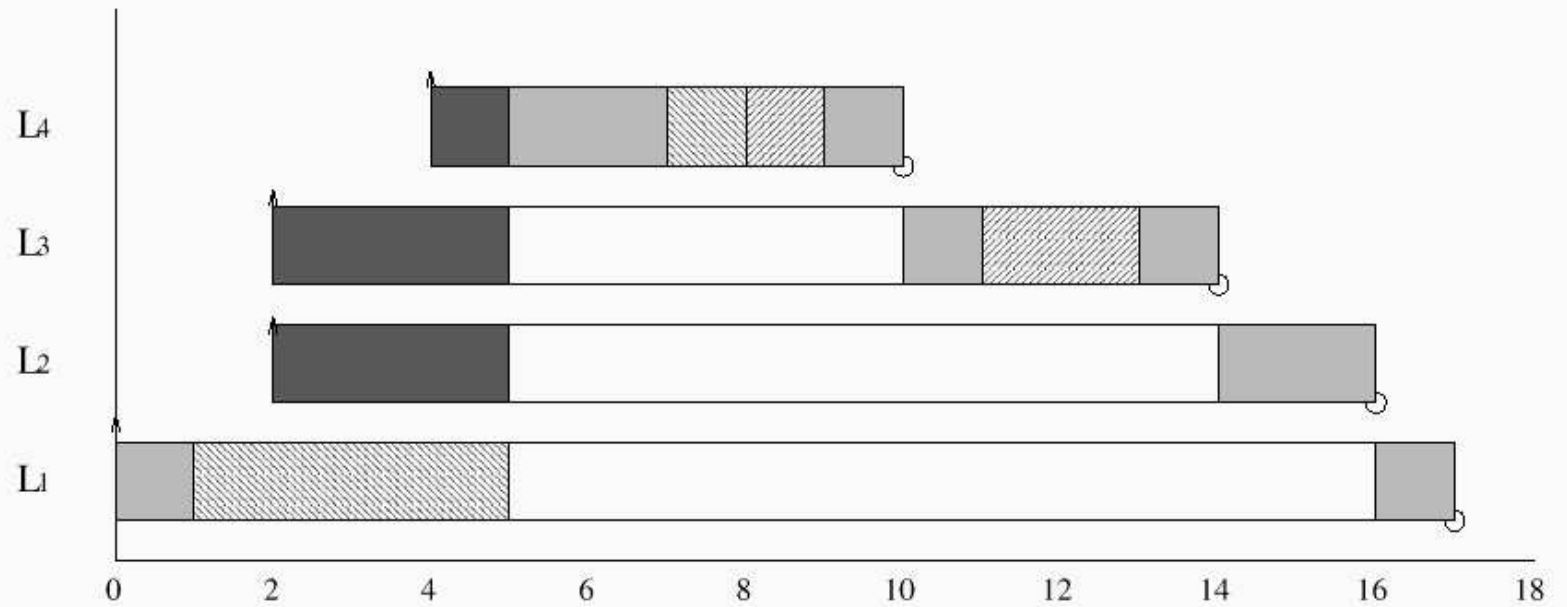


ICPP

- Pior caso do tempo de bloqueio=OCP
- É mais fácil de implementar do que OCP
 - Não necessita monitorar as relações de bloqueio
- Provoca menos trocas de contexto
- Causa mais trocas de prioridade
 - Trocas a cada reserva de recurso
 - OCP apenas causa troca de prioridade se um bloqueio é provocado
- ICPP é denominado Priority Protected Protocol no POSIX
 - Utilizado por variáveis mutex



Exemplo





Ceiling, Exclusão Mútua e Deadlocks

- Os protocolos OCPP e ICPP implementam a exclusão mútua por sí só, sem necessidade de outras primitivas para garantir a exclusão mútua
- Previnem Deadlocks
 - Não existe a possibilidade de espera circular

Escalonamento Cooperativo



- $B_{\text{máx}}$ = máximo tempo de bloqueio
- Cada processo é dividido em blocks não-preemptíveis, limitados por $B_{\text{máx}}$
 - Ao final de cada bloco é chamada uma função do kernel para oferecer a chance de "desescalonamento"
 - O kernel pode monitorar o tempo e enviar um sinal ou abortar o processo se um bloco não preemptível for maior do que $B_{\text{máx}}$
- Exclusão mútua garantida dentro de cada bloco não-preemptível

Escalonamento Cooperativo



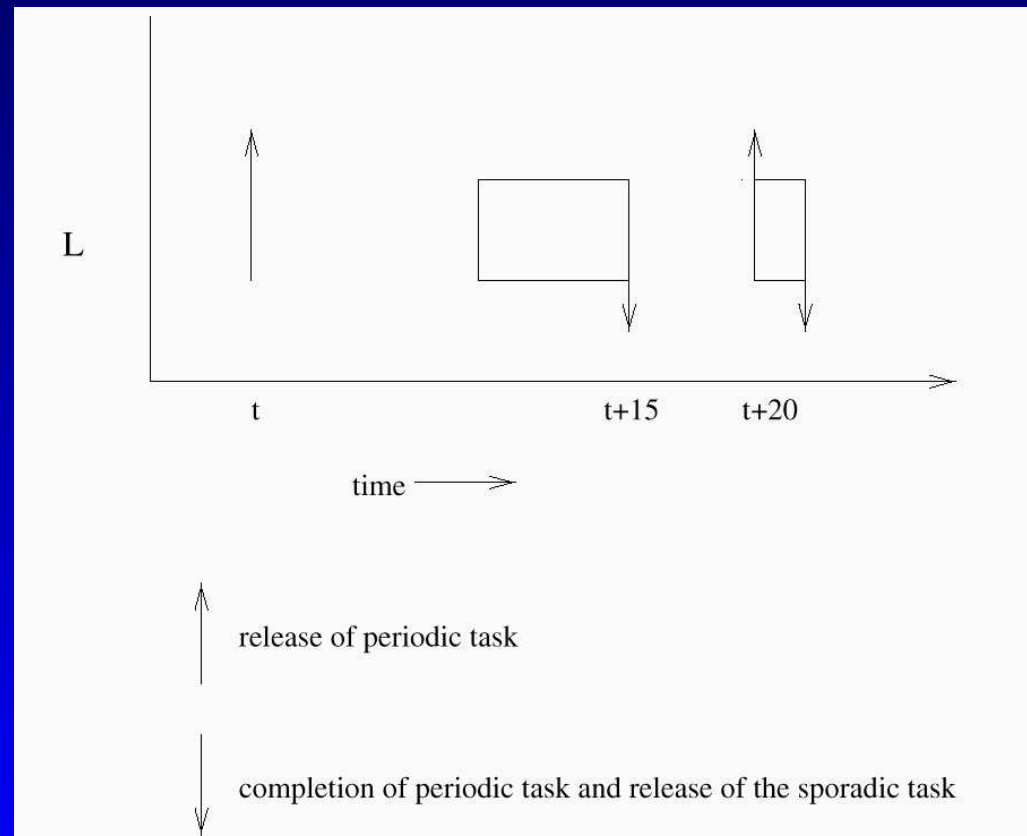
- Aumenta a escalonabilidade do sistema
- Tende a causar menores valores de tempo de computação
- Não ocorre interferência no último bloco
- F_i =tempo de execução do último bloco

$$\omega_i^{n+1} = B_i + C_i - F_i + \sum_{j \in \text{hp}(i)} \left\lceil \frac{\omega_i^n}{T_j} \right\rceil C_j$$

$$R_i = \omega_i^n + F_i$$

Jitter de Liberação

- Processo periódico (L) liberando um processo esporádico (S)
- $T_L = T_S$?





Jitter de Liberação

- Execução do processo periódico pode ser qualquer valor entre R_L e $C_L = 0$
- $T_L - R_L < T_S < T_L$
- $J =$ jitter de liberação
- O processo i vai sofrer:
 - Uma interferência de S se $0 \leq R_i < T - J$
 - Duas interferências de S se $T - J \leq R_i < 2T - J$
 - Três interferências de S se $2T - J \leq R_i < 3T - J$

Tempo de Resposta com Jitter



$$R_i = B_i + C_i + \sum_{j \in \text{hp}(i)} \left\lceil \frac{R_i + J_j}{T_j} \right\rceil C_j$$

$$R_i^{\text{periódico}} = R_i + J_i$$



Deadline Arbitrário

- $D > T$
- É necessário considerar a liberação de diversas instâncias do processo
- $\omega(q)$ = janela com a superposição de q instâncias

$$\omega_i^{n+1}(q) = B_i + (q + 1)C_i + \sum_{j \in \text{hp}(i)} \left\lceil \frac{\omega_i^n(q)}{T_j} \right\rceil C_j$$

Resposta com Deadline Arbitrário



$$R_i(q) = \omega_i^n(q) - qT_i$$

$$R_i(q) \leq T_i$$

$$R_i = \max_{q=0,1,2,\dots} R_i(q)$$



Atribuição de Prioridade

- Para deadlines genéricos, nenhum algoritmos simples (rate ou deadline monotonic) é ótimo
- Teorema:
 - Se um processo T possui a prioridade mais baixa e é factível, então
 - Se um esquema de prioridades factível existe para todo o conjunto de processos, então
 - Existe um esquema de prioridades tal que T possui a menor prioridade
- O teorema pode ser aplicado sucessivamente
- Se o teste de factibilidade for exato (necessário e suficiente) o esquema obtido é ótimo



Outros Esquemas

- Esquemas sem prioridade
 - Round-robin
 - É alocado um quantum de tempo para cada processo
 - First-in First-out
 - Pode-se ter um esquema de prioridade com vários processos com a mesma prioridade
 - Escalonamento RR ou FIFO em cada fila de prioridade
 - Utilizado no POSIX
- Shortest Job First



Outros Esquemas

- Escalonamento dinâmico
 - Earliest deadline
 - Least slack time
 - $\text{slack} = \text{deadline} - \text{tempo de execução}$
 - Algoritmos ótimos
 - Garantem escalonabilidade com utilização de 100%
 - Adequados para processos aperiódicos
 - Deadlines são perdidos durante sobrecargas transitórias
 - Pouco previsibilidade quando é introduzido bloqueamento