Escalonamento de Sistemas de Tempo Real

Sergio Cavalcante
Centro de Informática – UFPE

str-l@cin.ufpebr *svc@cin.ufpe.br*

Assunto: [str]

88350950 *34254714*

Criticidade

Sistema de tempo real crítico (Hard real-time system)

- Todas as tarefas têm Hard Deadline
  - Perda do deadline pode ter conseqüências catastróficas
- É necessário garantir requisitos temporais ainda durante o projeto
- Exemplo: usina nuclear, industria petroquímica, mísseis

Sistema de tempo real não crítico (Soft real-time system)

- O requisito temporal descreve apenas o comportamento desejado
- Perda do deadline não tem conseqüências catastróficas
- Existe interesse em terminar a tarefa mesmo com atraso
- Exemplo: início de gravação de vídeo-cassete

Criticidade

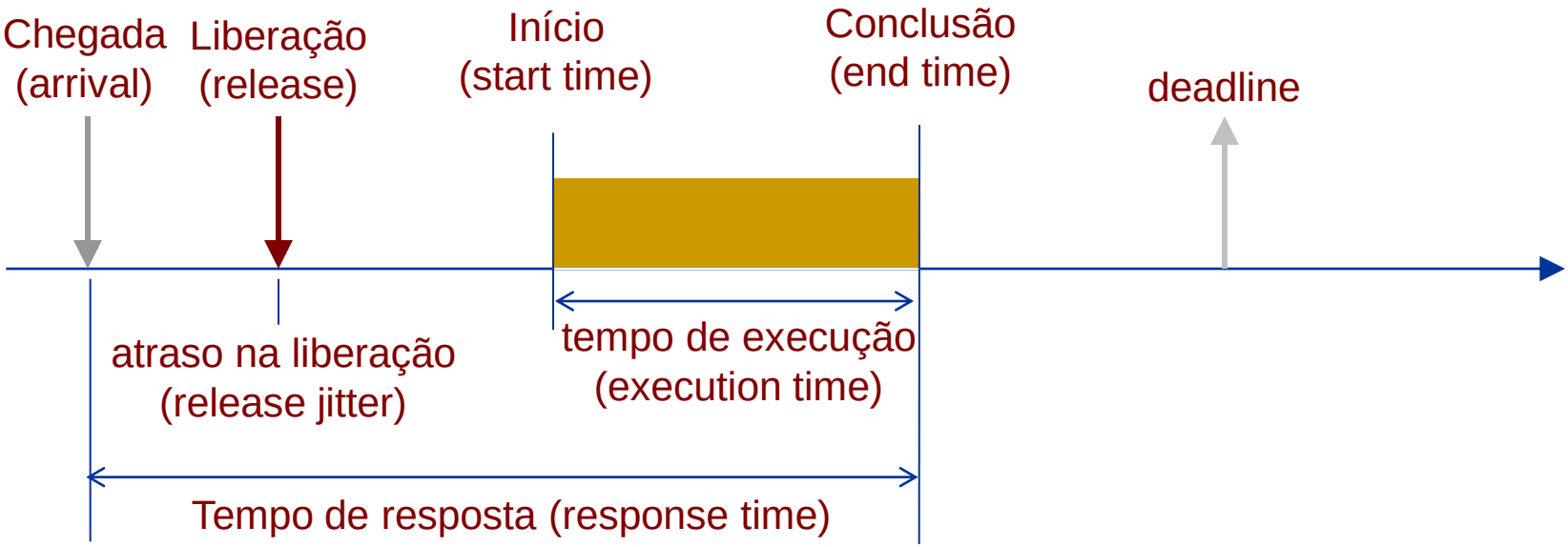
Deadline Firm

- Perda do deadline não tem conseqüências catastróficas
- Não existe valor em terminar a tarefa após o deadline
- Exemplo: ler o valor da temperatura

Modelagem das Tarefas

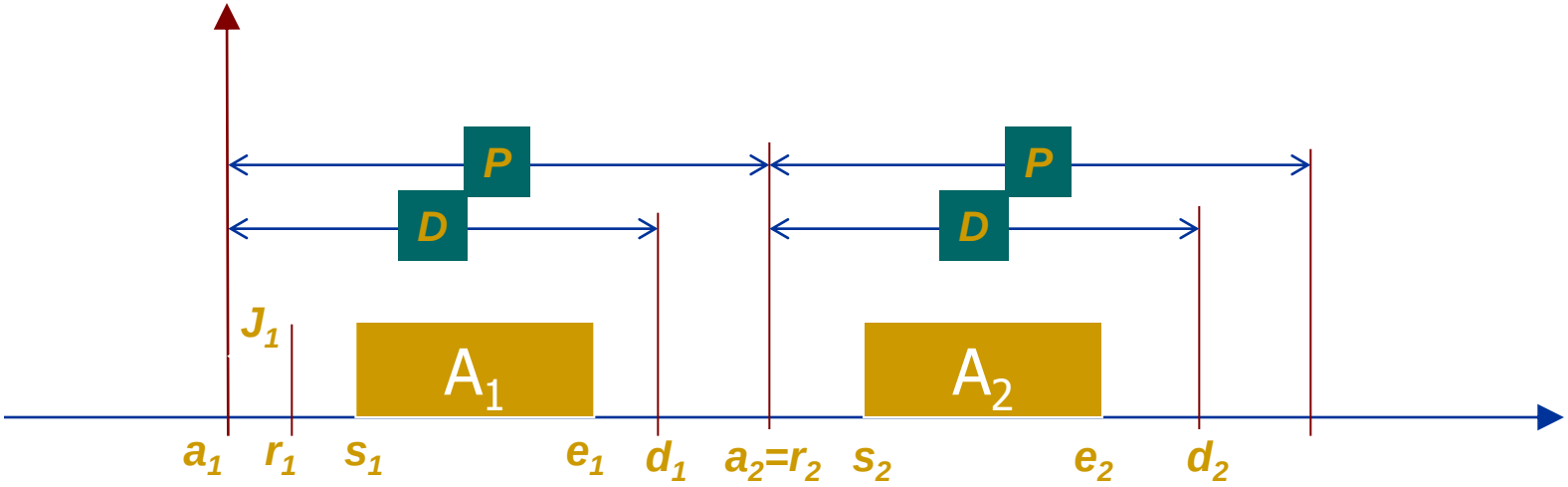
- Periodicidade
  - Tarefas Aperiódicas
    - São disparadas em intervalos imprevisíveis de tempo
    - Não garantem escalonabilidade
  - Tarefas Esporádicas
    - É conhecido o intervalo mínimo entre execuções (*inter-arrival time*)
  - Tarefas Periódicas
    - Devem ser executadas em intervalos regulares de tempo

Modelagem das Tarefas

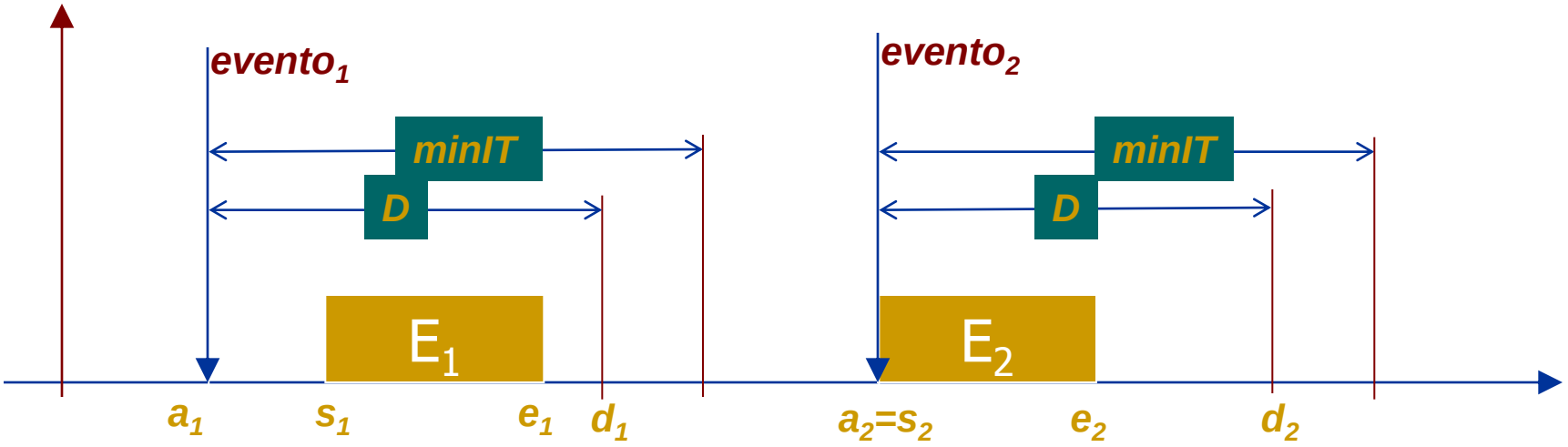


- Folga (*slack*) = deadline – liberação – tempo de execução
- Atraso = MÁX(0, conclusão – deadline)

Tarefas Periódicas



Tarefas Esporádicas



Escalonamento

- Define a ordem ou escala de execução das tarefas, a partir de um algoritmo ou política de escalonamento.

Escalonamento

- Preemptivo ou não preemptivo
  - Permite ou não a preempção de tarefas
- Estático
  - Quando a ordem de execução toma como base parâmetros definidos em tempo de projeto
- Dinâmico
  - Quando a ordem de execução toma como base parâmetros definidos em tempo de execução

Escalonamento

- Offline
  - Quando a escala é definida em tempo de projeto
- Online
  - Quando a escala é definida em tempo de execução

Obviamente é impossível ter um algoritmo que seja offline dinâmico. As demais combinações são possíveis.

Escalonamento

- Um algoritmo de escalonamento é dito ***ótimo*** quando consegue encontrar uma solução sempre que uma exista, ou ainda, se no caso de qualquer outro algoritmo encontrar uma solução, o primeiro também consegue encontrá-la.

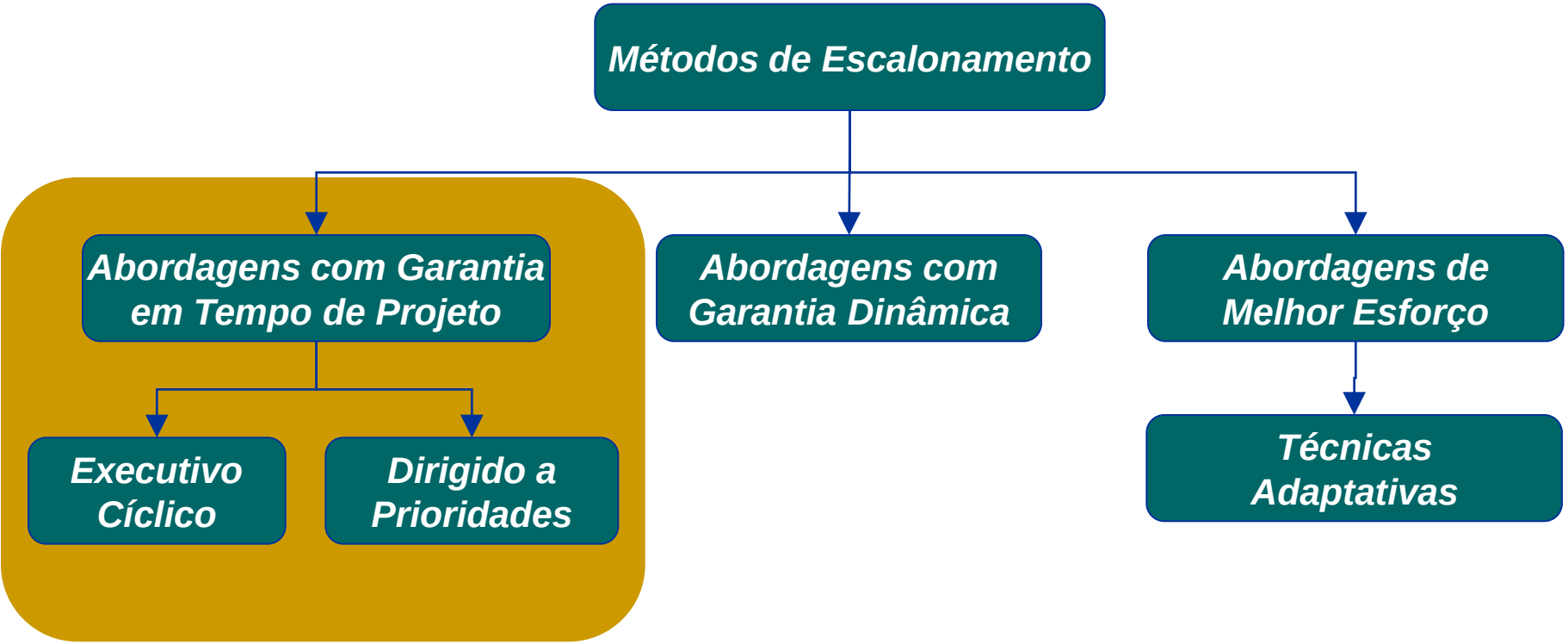
Teste de Escalonabilidade

- Permite dizer se um conjunto de tarefas conhecido é escalonável por um determinado algoritmo de escalonamento.

Voltando à Hipótese de Carga

- Carga Estática ou Limitada
  - Conhecemos a carga *a priori*
  - Permite obter garantia de escalonamento em tempo de projeto. O sistema é determinístico.
- Carga Dinâmica ou Ilimitada
  - Não permite garantias em tempo de projeto
  - Implica na existência de tarefas aperiódicas

Escalonamento



Garantia em Tempo de Projeto

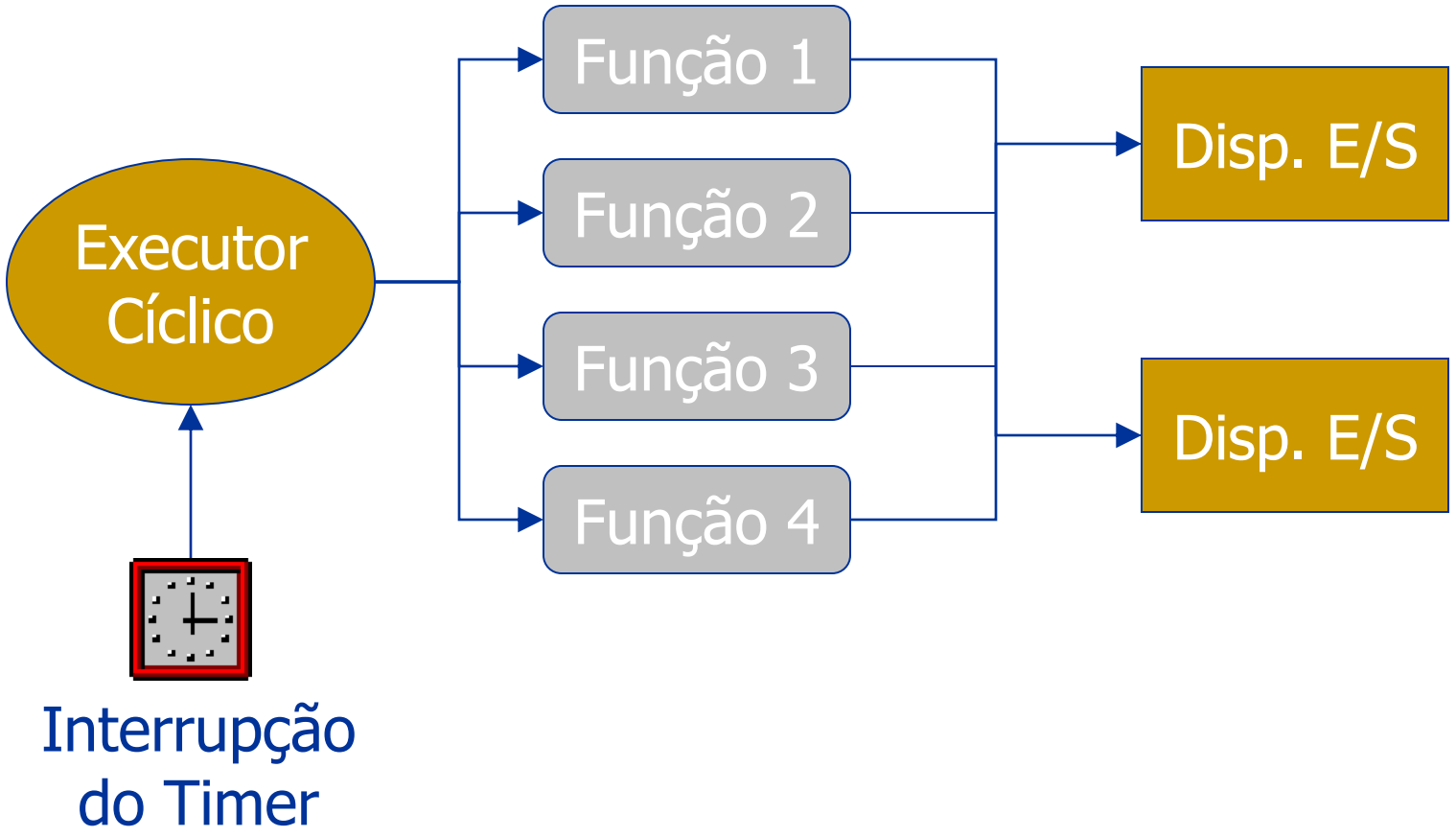
- Para obter garantias em tempo de projeto é necessário ser possível realizar um teste de escalonabilidade em tempo de projeto.
- Par isso é necessário:
  - Conhecimento completo das tarefas e do método de implementação em tempo de projeto.
  - Que a carga seja estática.

Abordagens com Garantia em Projeto

- Vantagens
  - Determina em projeto que todos os requisitos temporais serão cumpridos
  - Necessário para aplicações críticas
- Desvantagens
  - Necessário conhecer exatamente a carga
  - Necessário reservar recursos para o pior caso
  - Difícil determinar o pior caso em soluções off- the- shelf
  - Gera enorme subutilização de recursos

Time-Driven Systems: Executor Cíclico

- Arquitetura de Software



Modelos de Implementação

- Time-driven systems (Executivos Cíclicos)
  - O teste de escalonabilidade é a própria escala que é definida em tempo de projeto.
  - O executivo cíclico simplesmente põe as tarefas em execução segundo o tempo e a ordem definidas em tempo de projeto.
  - Não existe outra fonte de eventos além do timer interno.
  - É um escalonamento offline estático.

Time-Driven Systems

Executor Cíclico

- Tarefas são arranjadas numa lista que define a ordem e tempo de execução de cada uma.
- A busca por esta ordem é, em geral, realizada por um algoritmo de busca em árvore.
- São utilizadas heurísticas para limitar a busca, embora possa resultar na busca de todas as combinações possíveis (NP-completo).

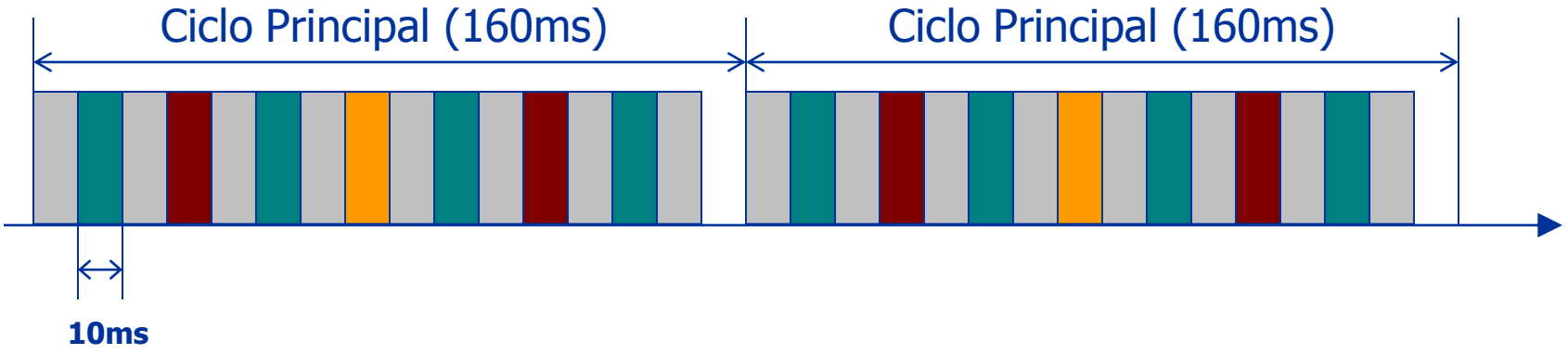
Time-Driven Systems

Executor Cíclico

- Cada tarefa é posta em execução nos momentos definidos na lista do escalonamento, segundo o temporizador (timer) do sistema.
- Apenas tarefas periódicas podem ser escalonadas.
- O ciclo completo é o Mínimo Múltiplo Comum dos períodos de todas as tarefas.
- A lista é executada repetidamente, caracterizando ciclos de execução.

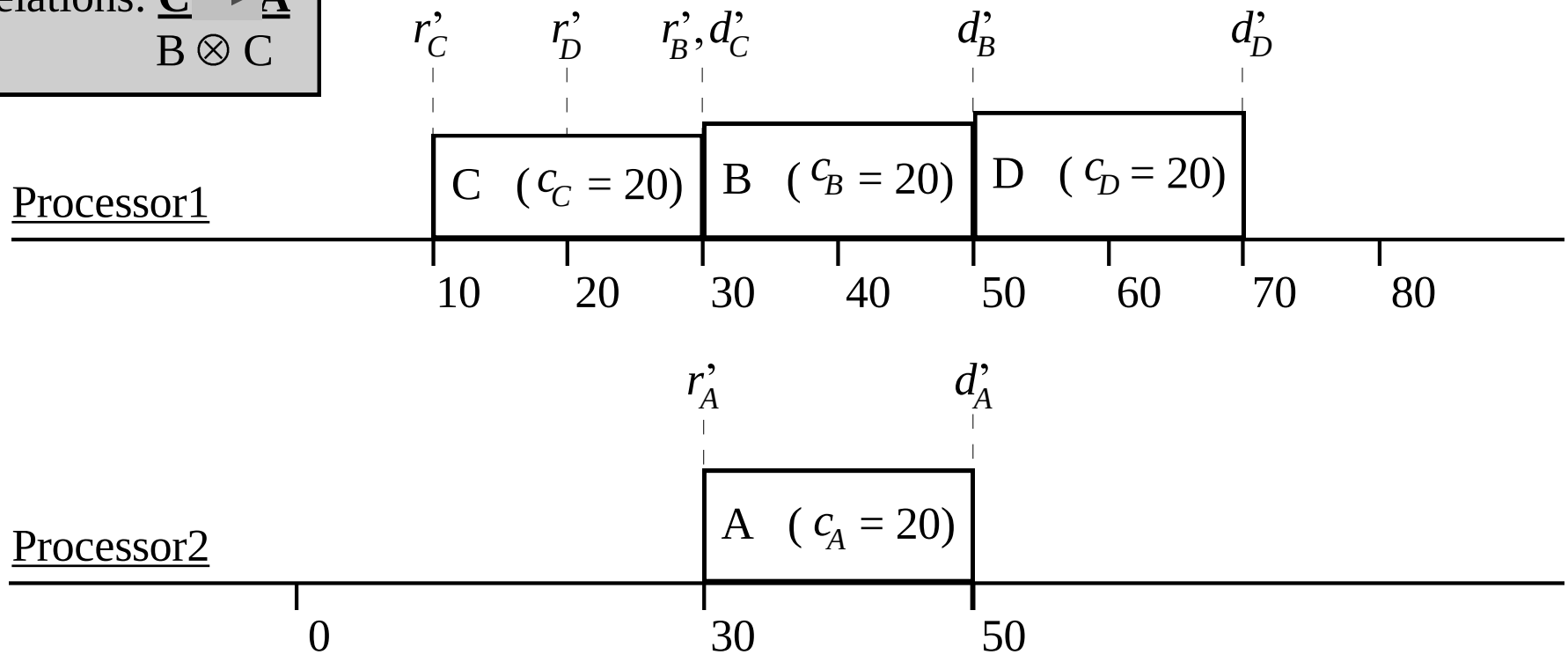
Time-Driven Systems: Exemplo 1

- Considere 4 funções com as seguintes características:
  - Função 1: ciclo de 50Hz (20ms)
  - Função 2: ciclo de 25 Hz (40ms)
  - Função 3: ciclo de 12,5 Hz (80ms)
  - Função 4: ciclo de 6,25Hz (160ms)



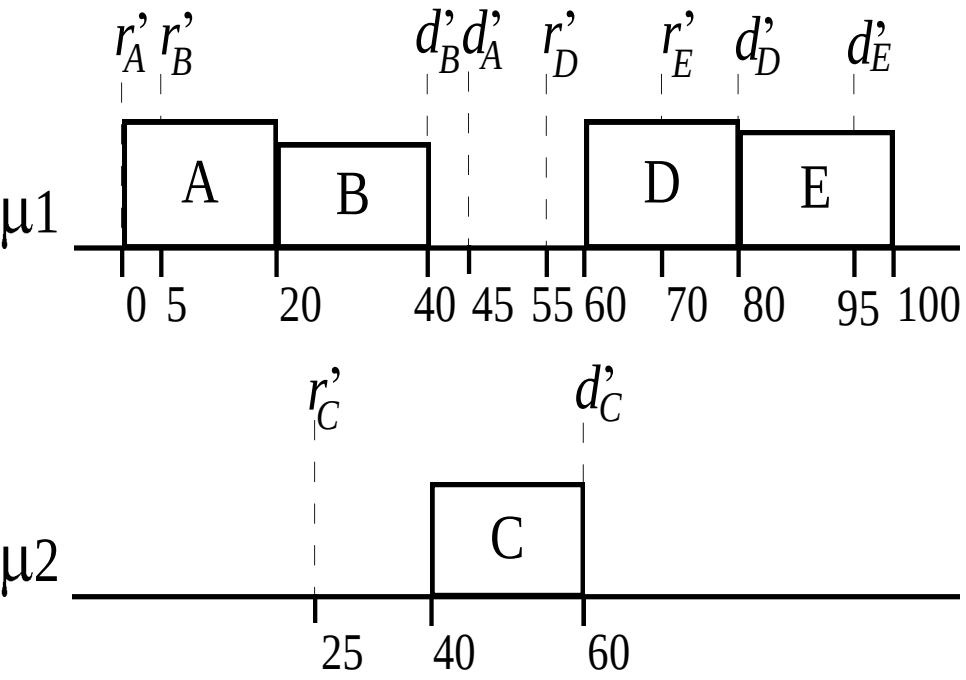
Time-Driven Systems: Exemplo 2

Relations: **C** → **A**
B ⊗ C

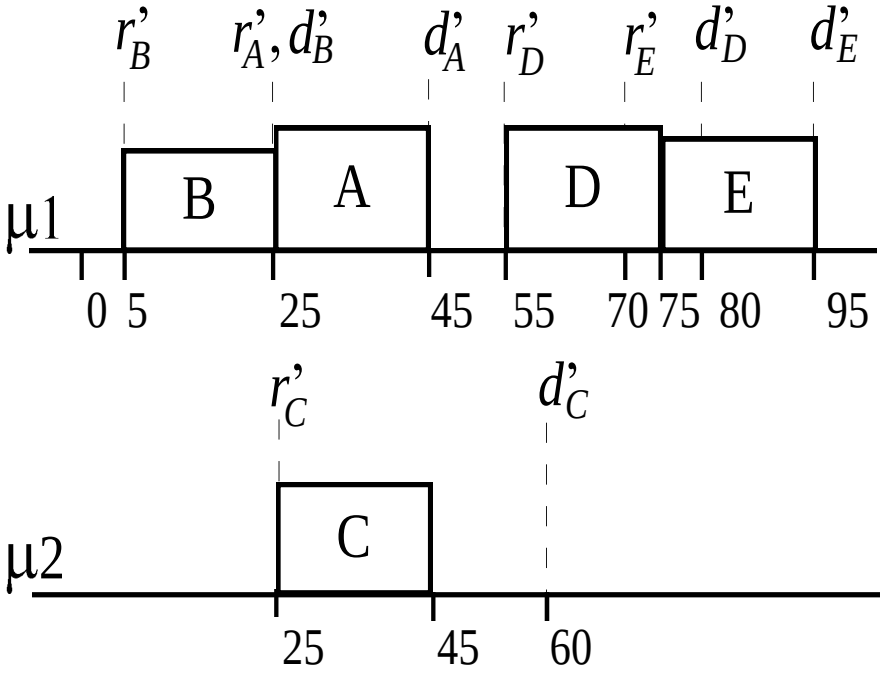


Time-Driven Systems: Exemplo 3

Relations: A ⊗ B,
B → C, C → D

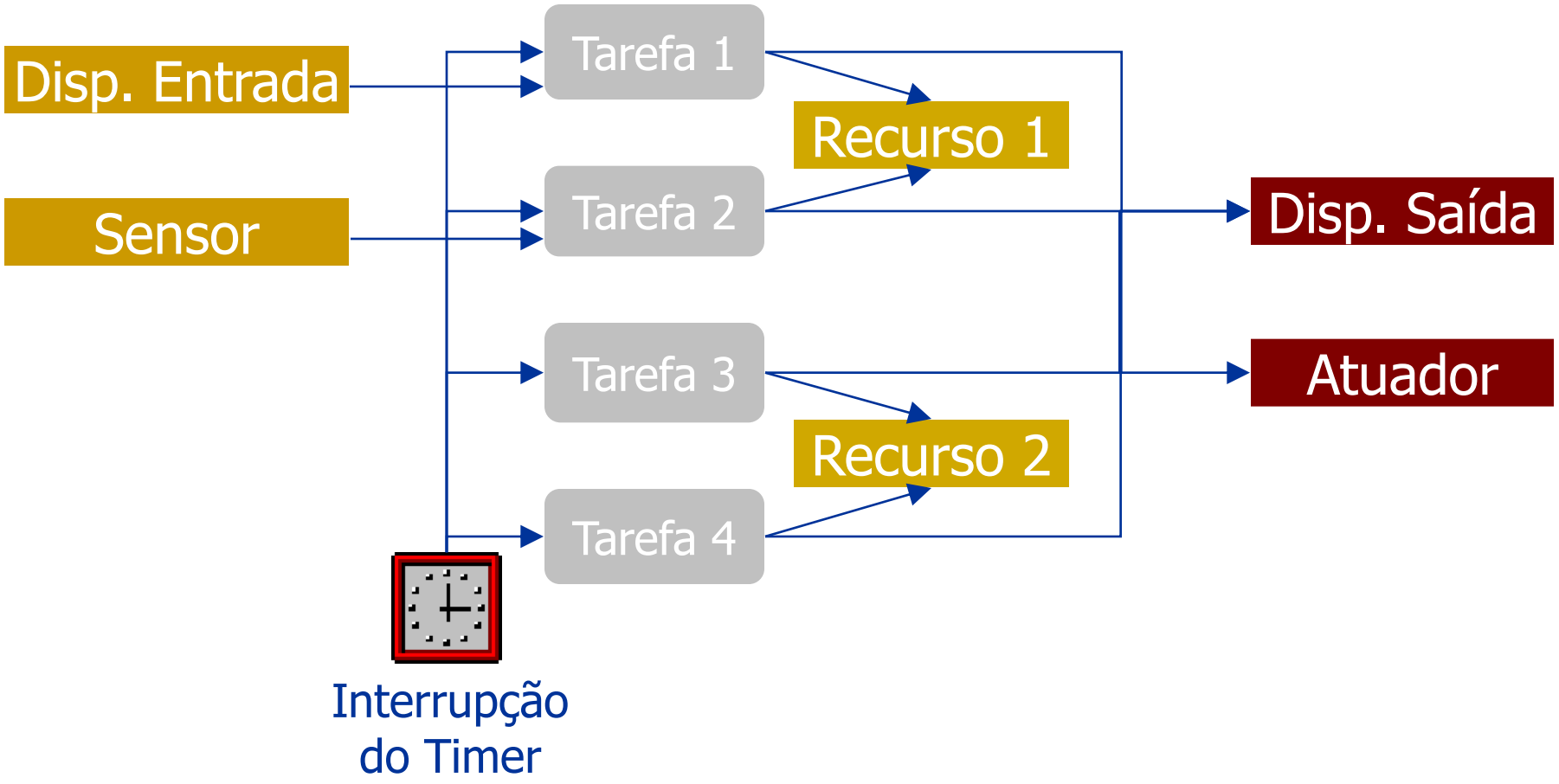


Relations: **B → A**,
B → C, C → D



Event-Driven Systems

Arquitetura de Software



Modelos de Implementação

- Event-driven systems
  - É feito um teste escalonabilidade em tempo de projeto, mas a escala propriamente dita é feita em tempo de execução.
  - A definição da escala obedece às prioridades das tarefas.
  - A ordem real de execução é regida por eventos (externos ou do timer) e pelas prioridades das tarefas.

Event-Driven Systems

Escalonamento Rate-Monotonic (RMS)

- Escalonamento preemptivo de prioridade fixa.
- Quanto menor o período maior a prioridade de uma tarefa.
- Tarefas periódicas e independentes (sem relação de precedência, exclusão, etc).
- Os deadlines coincidem com os períodos.
- Tempo de mudança de contexto é considerado nulo.
- É ótimo, ou seja, nenhum outro método é melhor que este com estas condições.

Que tipo de escalonamento é este?

É um escalonamento online estático.

Event-Driven Systems

Análise de Escalonabilidade para o *Escalonamento Rate-Monotonic (RMS)*

- Este é um Teste suficiente mas não necessário.
- Notação: Para uma Tarefa T_i, temos que:

  P_i: Período

  Min_i: Intervalo Mínimo entre Requisições (tarefas esporádicas)

  C_i: Tempo de Computação

  D_i: Deadline

- U_i = Utilização do Processador por uma tarefa T_i
  - $U_i = C_i/P_i$, se T_i é periódica
  - $U_i = C_i/Min_i$, se T_i é esporádica

Event-Driven Systems

Análise de Escalonabilidade para o *Escalonamento Rate-Monotonic (RMS)*

- Um conjunto de *n* tarefas periódicas independentes escalonadas pelo RMS sempre obedecerá o seu deadline se:

$$U(n) = \frac{C_1}{P_1} + \ldots + \frac{C_n}{P_n} \leq n\left(2^{\frac{1}{n}} - 1\right)$$

ou

$$U(n) = \sum_{1}^{n} C_i / P_i \leq n\left(2^{\frac{1}{n}} - 1\right)$$

onde, U(n) é o limite de utilização para *n* processos.

Para $n \to \infty \Rightarrow U(n) \to 0{,}6993147\ldots \cong 69{,}93147\%$

Event-Driven Systems

Análise de Escalonabilidade para o *Escalonamento Rate-Monotonic (RMS)*

- Exemplo:

Tarefas	P_i	C_i	Pri_i	U_i
T_A	100	20	1	0,200
T_B	150	40	2	0,267
T_C	300	100	3	0,286

$$0{,}200+0{,}267+0{,}286=0{,}753\leq n\left(2^{\frac{1}{n}}-1\right)=3\left(2^{\frac{1}{3}}-1\right)=0{,}779$$

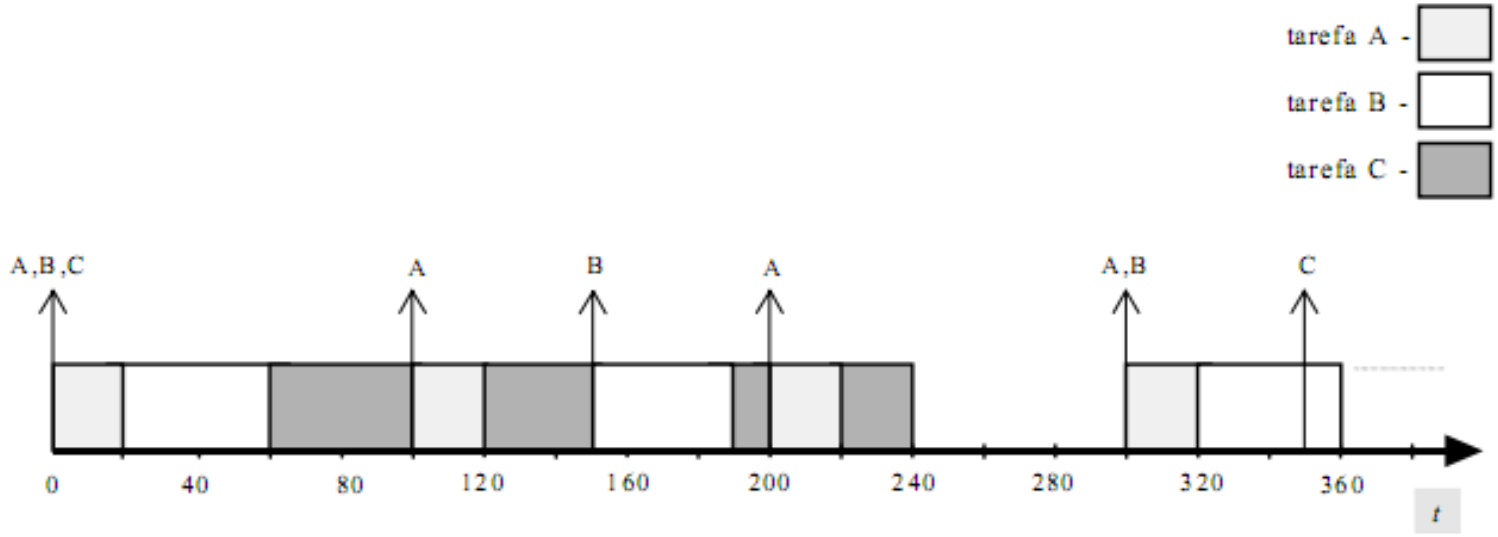
- **As tarefas são escalonáveis pelo RMS.**

Event-Driven Systems

Análise de Escalonabilidade para o *Escalonamento Rate-Monotonic (RMS)*

- Exemplo:

Tarefas	P_i	C_i	Pri_i	U_i
T_A	100	20	1	0,200
T_B	150	40	2	0,267
T_C	300	100	3	0,286



Event-Driven Systems

Análise de Escalonabilidade para o *Escalonamento Rate-Monotonic (RMS)*

- *U(n) = 1,0 se o conjunto de tarefas é harmônico*
  - Um conjunto de tarefas é harmônico se os períodos de todas as tarefas são múltiplos ou sub-múltiplos entre si

- *U(n) = 0,88 na média de tarefas randômicas*

Event-Driven Systems

Escalonamento Earliest-Deadline First (EDF)

- Escalonamento preemptivo de prioridade **dinâmica**.
- Quanto mais próximo o deadline maior a prioridade de uma tarefa.
- Tarefas periódicas e independentes (sem relação de precedência, exclusão, etc).
- Os deadlines coincidem com os períodos ($D_i = P_i$).
- Tempo de mudança de contexto é considerado nulo.
- É ótimo, ou seja, nenhum outro método é melhor que este com estas condições.

Que tipo de escalonamento é este?

É um escalonamento online dinâmico.

Event-Driven Systems

Análise de Escalonabilidade para o *EDF*

- Este é um Teste suficiente e necessário.

$$U(\tau) = \sum_{1}^{n} C_i / P_i \leq 1$$

Event-Driven Systems

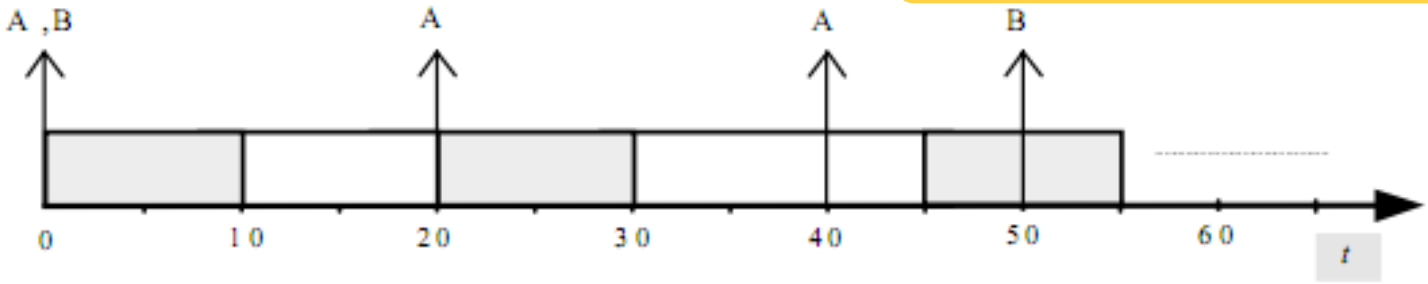
Análise de Escalonabilidade para o *EDF*

tarefas periódicas	C_i	P_i	D_i
tarefa A	10	20	20
tarefa B	25	50	50

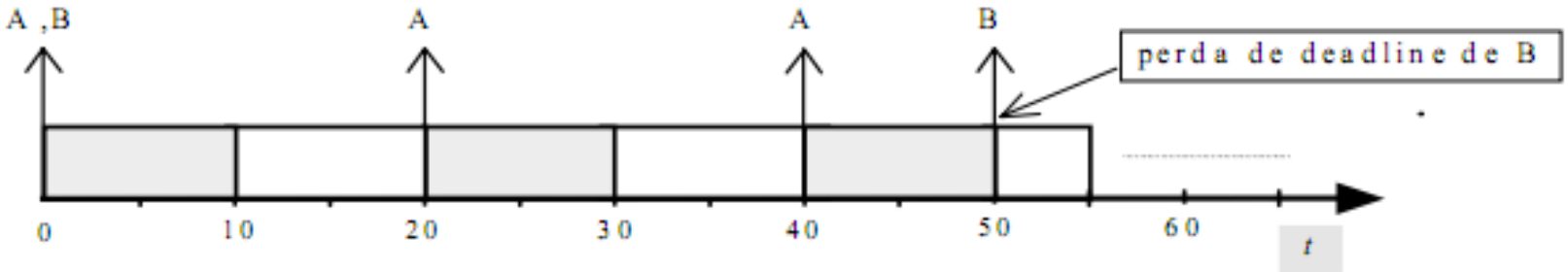
tarefa A -

tarefa B -

$$U = 10/20 + 25/50 = 1 \leq 1$$



(a) Escalonamento EDF



(b) Escalonamento RM

Event-Driven Systems

Problemas do *EDF*

1. Quando o sistema está com sobrecarga (*overloaded*) o conjunto de processos que perde seu deadline é altamente imprevisível: é uma função dos deadlines no momento em que ocorre a sobrecarga.
2. É muito difícil de implementar em hardware.
3. É difícil representar deadlines em poucos bytes para calcular deadlines relativos ao instante atual. Se for usada aritmética modular, as variáveis que armazenam os deadlines futuros devem acomodar um valor no mínimo do ((maior tempo esperado para o término} * 2) + "agora").
4. Assim, **EDF não é muito usado em sistemas industriais.**

http://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling

Event-Driven Systems

Escalonamento Deadline-Monotonic (DMS)

- Semelhante ao RMS, mas os deadlines podem ser menores ou iguais aos períodos
- Quanto menor o deadline da tarefa maior sua prioridade
- Os deadlines são fixos e relativos aos começos dos períodos.

Que tipo de escalonamento é este?

É um escalonamento online estático.

Event-Driven Systems

Análise de Escalonabilidade para o *Escalonamento Deadline-Monotonic (DMS)*

- Este é um Teste Exato.
- Notação: Para uma Tarefa T_i, temos que:
  - P_i: Período
  - C_i: Tempo de Computação
  - D_i: Deadline
  - Pri_i: Prioridade
  - R_i: Tempo de Resposta
  - I_i: Interferência
- *Calcula o tempo de resposta no pior caso e compara ao deadline*
- *Se $R_i < D_i \; \forall \; T_i$, o teste é válido.*

Event-Driven Systems

Análise de Escalonabilidade para o *Escalonamento Deadline-Monotonic (DMS)*

- *Para a tarefa **T_1**, a mais prioritária:*

  $R_1 = C_1$

- *As demais sofrem **interferência** das que tem prioridade maior*

  Neste caso, $R_i = C_i + I_i$

- *Interferência é máxima a partir do **Instante Crítico***
  - *Instante onde todas as tarefas são liberadas simultaneamente*

Event-Driven Systems

Análise de Escalonabilidade para o *Escalonamento Deadline-Monotonic (DMS)*

Interferência entre tarefas

Seja $\mathbf{T_j}$ uma tarefa com prioridade maior que $\mathbf{T_i}$

- Quantas vezes $\mathbf{T_j}$ pode acontecer durante a execução de $\mathbf{T_i}$?

  $\lceil R_i/T_j \rceil$

- Qual a interferência total de $\mathbf{T_j}$ sobre $\mathbf{T_i}$?

  $\lceil R_i/T_j \rceil \times C_j$

- Qual a interferência total sobre $\mathbf{T_i}$?

  $\mathbf{I_i} = \sum \lceil R_i/T_j \rceil \times C_j$, onde $Pri_j > Pri_i$

Event-Driven Systems

Análise de Escalonabilidade para o *Escalonamento Deadline-Monotonic (DMS)*

O tempo máximo de resposta de $\mathbf{T_i}$ é $\mathbf{R_i = C_i + I_i}$

$$R_i = C_i + \sum \lceil R_i/T_j \rceil \times C_j$$

O cálculo é feito recursivamente em iterações sucessivas, até:

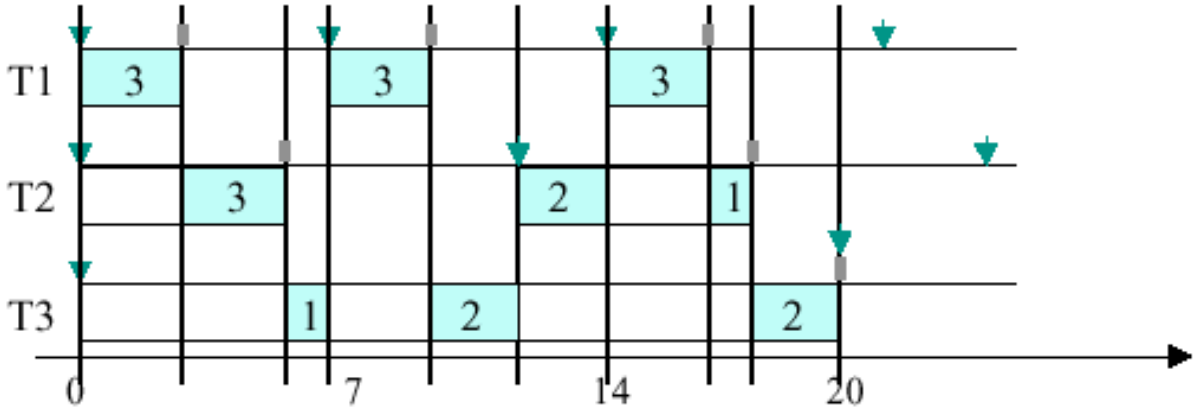
- Tempo de resposta passar do deadline (falha)
- Resultado convergir, ou seja, iteração ***x+1*** igual a iteração ***x***

$$W_i^{x+1} = C_i + \sum \lceil W_i^x/T_j \rceil \times C_j, \text{ onde } W_i^0 = C_i$$

Event-Driven Systems

Exemplo

T_1	T_2	T_3
$P_1=7$	$P_2=12$	$P_3=20$
$C_1=3$	$C_2=3$	$C_3=5$
$Pri_1=1$	$Pri_2=2$	$Pri_3=3$



Resposta:

$R_1 = 3$ $R_2 = 6$ $R_3 = 20$

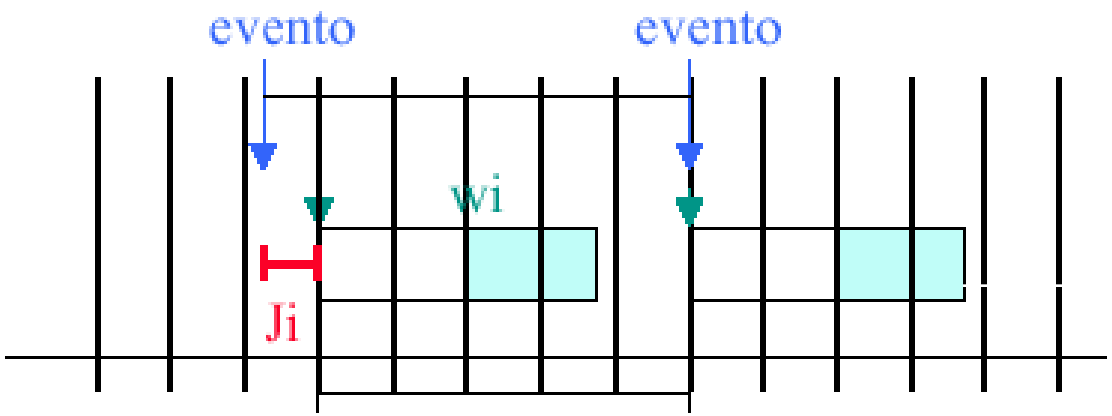
Event-Driven Systems

Atrasos na Liberação das Tarefas:

Suponha uma tarefa liberada por evento externo

- Eventos podem ser amostrados periodicamente
- Sinalização do evento pode ter atraso variável
- Release Jitter J_i: Atraso máximo na liberação da tarefa
- A fórmula do tempo de resposta se torna:

$$R_i = J_i + W_i, \quad \text{onde } W_i = C_i + \sum \lceil (W_i + J_j)/T_j \rceil \times C_j$$



Event-Driven Systems

Deadlines Arbitrários: deadline pode ser maior que o período

- Podem ocorrer *Interferências Internas:*
  Uma ativação anterior da tarefa interfere nela mesma
- Dada uma ativação de uma tarefa, se ela pode sofrer interferência de *q* ativações anteriores dela mesma, a fórmula se torna:

$$W_i(q) = (q+1)C_i + \sum_j \left\lceil \frac{W_i(q)}{P_j} \right\rceil . C_j$$

Onde **$q.C_i$** é a interferência interna que a instância atual sofre no tempo **$W_i(q)$** e o tempo de resposta desta será:

$$R_i(q) = W_i(q) - q.Pi$$

Event-Driven Systems

Deadlines Arbitrários: deadline pode ser maior que o período

- Para se obter o tempo de resposta máximo R_i de T_i, deve-se inspecionar todos as instâncias (q = 0, 1, 2, ...) até que:

$$W_i(q) \leq (q+1)P_i, \quad e \quad R_i = \max_{q=0,1,2,\ldots} R_i(q)$$

- Considerando a ocorrência de Jitter, fica:

$$W_i(q) = (q+1)C_i + \sum_{j \in hp(i)} \left\lceil \frac{W_i(q) + J_j}{P_j} \right\rceil C_j$$

$$e \qquad R_i = \max_{q=0,1,2,\ldots} (J_i + W_i(q) - qP_i).$$

- O sucesso do teste de escalonabilidade continua sendo:

$$Di \leq Ri, \quad \forall i$$

Event-Driven Systems

Deadlines Arbitrários. Exemplo:

tarefas periódicas	J_i	C_i	P_i	D_i
tarefa T_1	1	10	40	40
tarefa T_2	3	10	80	25
tarefa T_3	-	5	20	40

Event-Driven Systems

Deadlines Arbitrários. Exemplo:

$$W_3(q) = (q+1).5 + \left\lceil \frac{W_3(q)+1}{40} \right\rceil .10 + \left\lceil \frac{W_3(q)+3}{80} \right\rceil .10$$

- Para $q = 0$, fica:

$$W_3^{0}(0) = C_3 = 5$$

$$W_3^{1}(0) = 5 + \left\lceil \frac{5+1}{40} \right\rceil .10 + \left\lceil \frac{5+3}{80} \right\rceil .10 = 25$$

$$W_3^{2}(0) = 5 + \left\lceil \frac{25+1}{40} \right\rceil .10 + \left\lceil \frac{25+3}{80} \right\rceil .10 = 25$$

e o $R_3(0) = 25$. Como R3(0) = 25 > P3 = 20, temos que continuar o teste,

Event-Driven Systems

Deadlines Arbitrários. Exemplo:

Para $q = 1$, fica:

$$W_3^0(1) = C_3 = 5$$

$$W_3^1(1) = 10 + \left\lceil \frac{5+1}{40} \right\rceil .10 + \left\lceil \frac{5+3}{80} \right\rceil .10 = 30$$

$$W_3^2(1) = 10 + \left\lceil \frac{30+1}{40} \right\rceil .10 + \left\lceil \frac{30+3}{80} \right\rceil .10 = 30$$

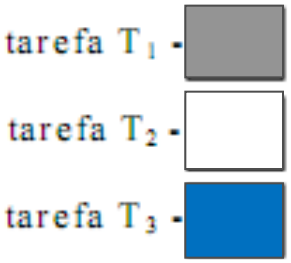
e $R_3(1) = W_3(1) - P_3 = 10$

Como $W_3(1) = 30 < 2.P_3 = 40$, podemos parar o teste.

e Ri = max(25, 10) = 25 < Di = 40, o conjunto é escalonável!

Event-Driven Systems

Deadlines Arbitrários. Exemplo:



tarefas periódicas	J_i	C_i	P_i	D_i
tarefa T_1	1	10	40	40
tarefa T_2	3	10	80	25
tarefa T_3	-	5	20	40

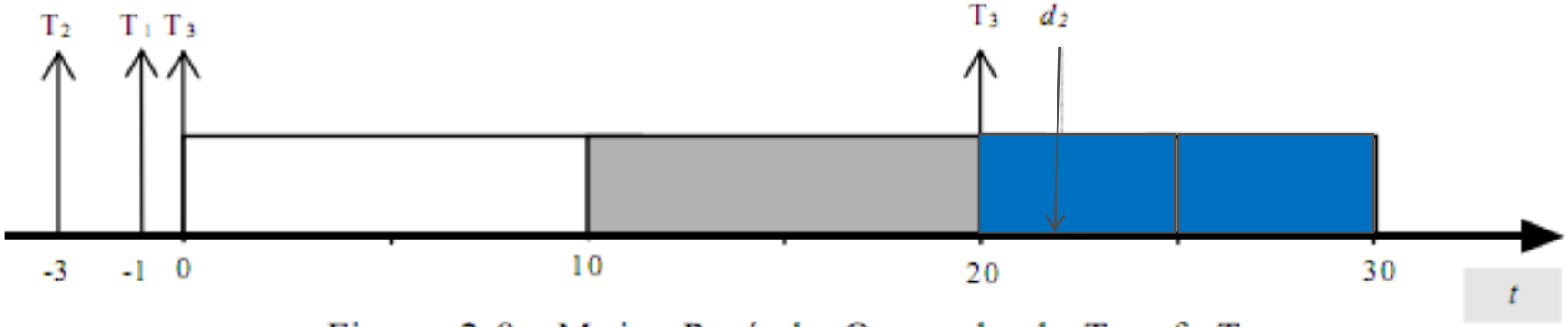


Figura 2.8: Maior Período Ocupado da Tarefa T_3

Event-Driven Systems

Compartilhamento de Recursos: Bloqueios

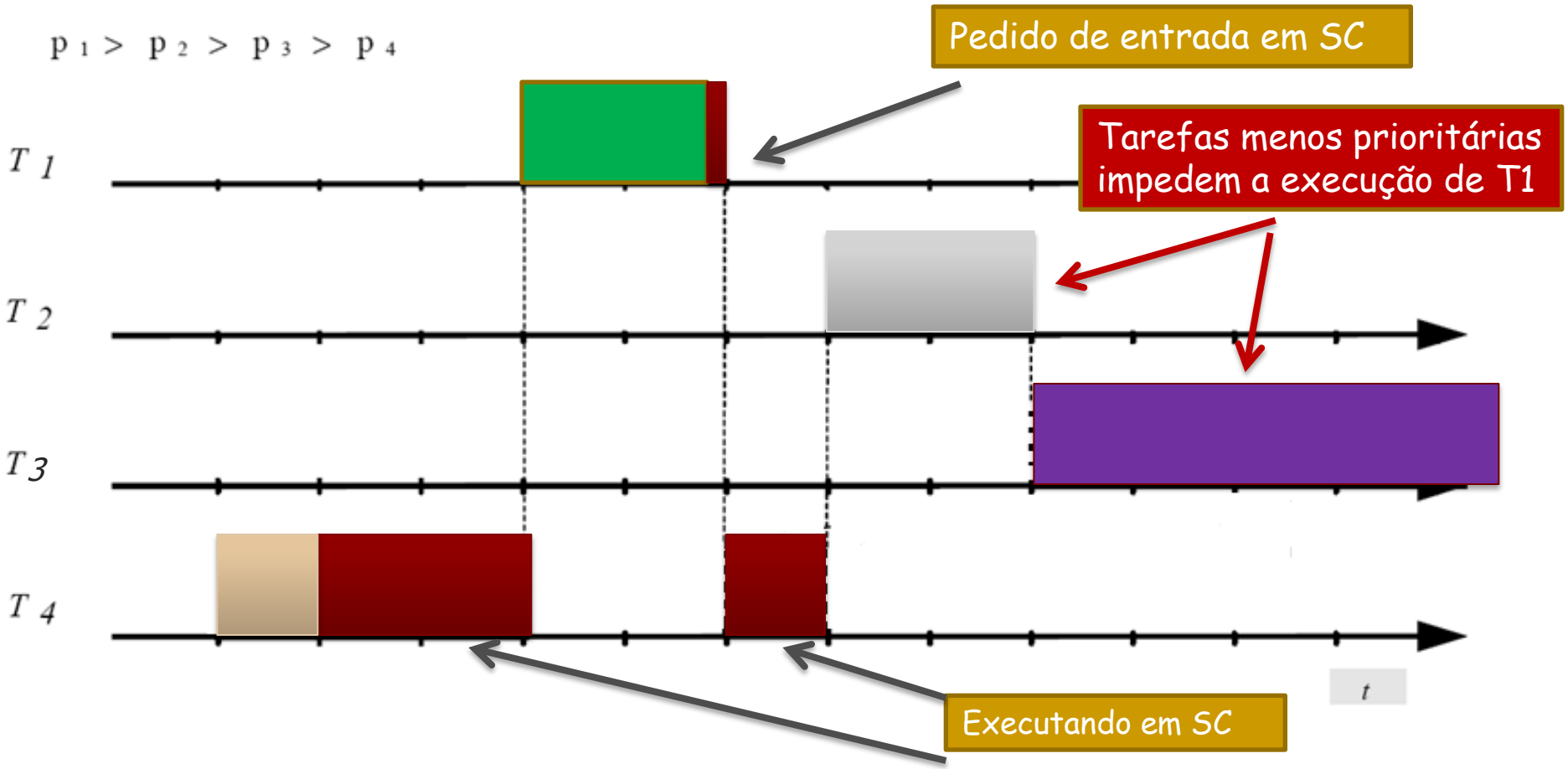
- Ocorrem devido às relações de exclusão mútua
- Suponha T_1 e T_2, sendo T_1 com maior prioridade
  - Se T_2 fica bloqueada esperando por T_1
    Ok, T_1 tem mesmo prioridade superior
  - Se T_1 fica bloqueada, esperando por T_2
    Cálculo do tempo de resposta deve incluir a espera máxima B_i

$$R_i = J_i + W_i$$

$$W_i = C_i + B_i + \sum \lceil (W_i + J_j)/T_j \rceil \times C_j$$

Compartilhamento de Recursos

- Seções Críticas e Inversão de Prioridades



Compartilhamento de Recursos

- Este comportamento é denominado de *inversão de prioridade.*
  - Tarefas menos prioritárias bloqueiam as mais prioritárias por estarem utilizando um recurso compartilhado.

Compartilhamento de Recursos

- Este comportamento é denominado de *inversão de prioridade.*
  - Tarefas menos prioritárias bloqueiam as mais prioritárias por estarem utilizando um recurso compartilhado.
  - Problemas
    - Tarefas mais prioritárias podem ficar um longo período de tempo bloqueadas.
      - Tarefas intermediárias vão provocar sucessivas preempções na tarefa em sessão crítica.
    - Como resolver isso?

Compartilhamento de Recursos

- Algoritmos mais comuns:
  - Protocolo Herança de Prioridade
  - Protocolo de Prioridade Teto

Protocolo Herança de Prioridade (PHP)

Tarefas possuem duas prioridades:

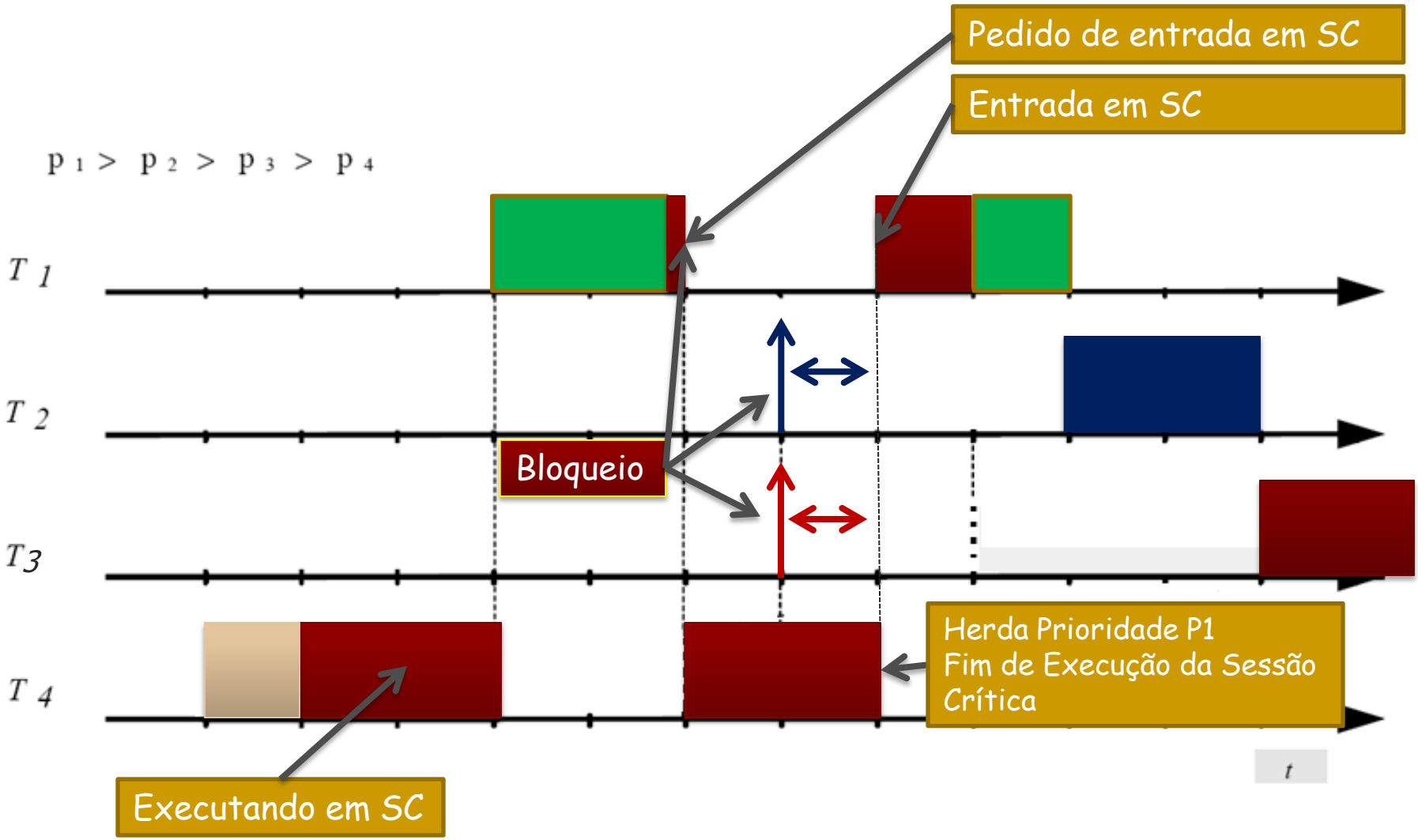
- Prioridade nominal ou estática
  - Definidas no RMS, DMS, etc.
- Prioridade dinâmica ou ativa
  - Derivadas das ações de bloqueio que ocorrem no sistema.

Protocolo Herança de Prioridade (PHP)

Funcionamento:

- Tarefas são escalonadas pela sua prioridade estática enquanto não existir recurso bloqueado.
- Quando existe recurso bloqueado, a tarefa em sessão crítica **herda a maior prioridade** entre as tarefas mais prioritárias bloqueadas na seção.

Protocolo Herança de Prioridade (PHP)

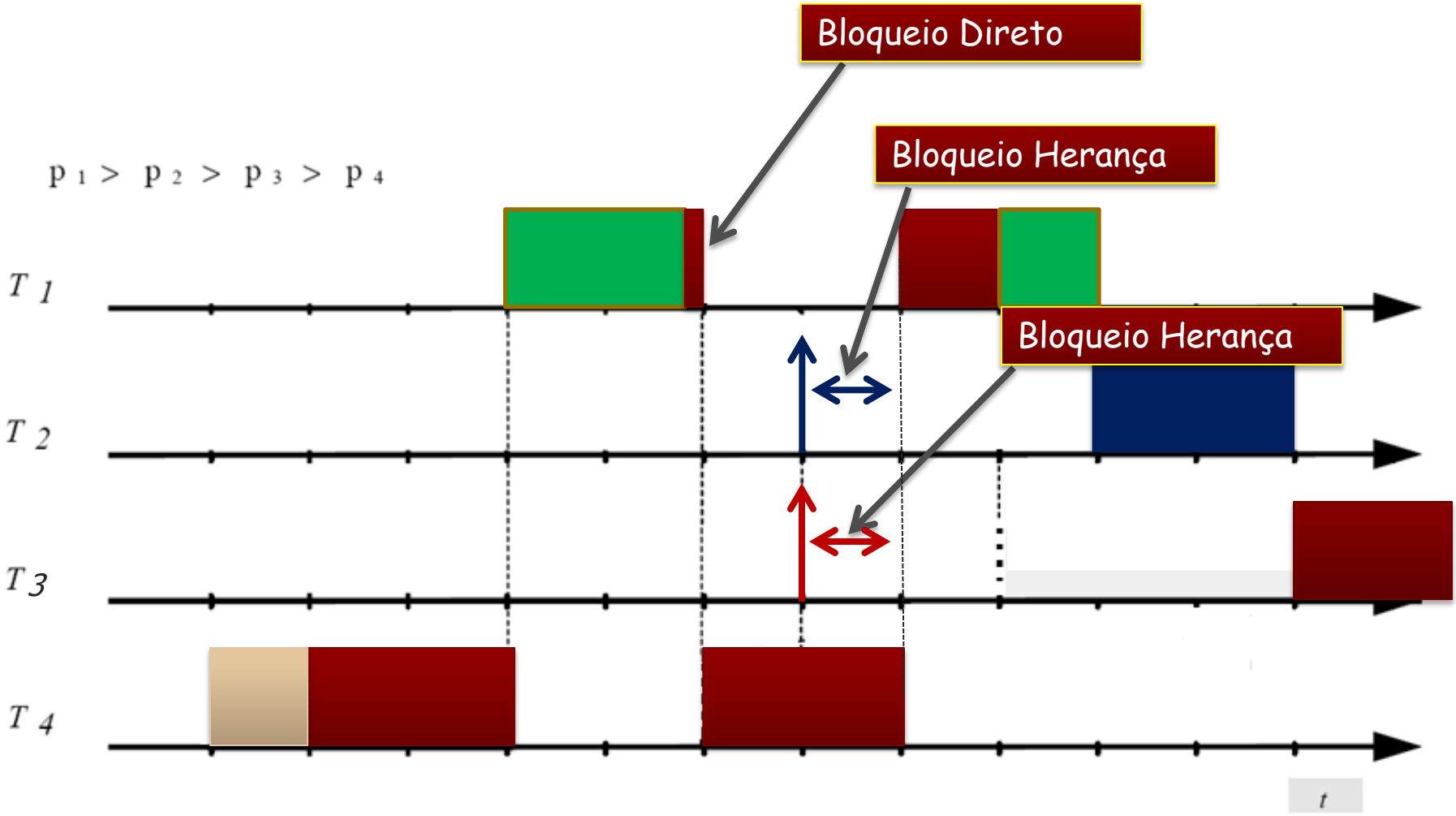


Protocolo Herança de Prioridade (PHP)

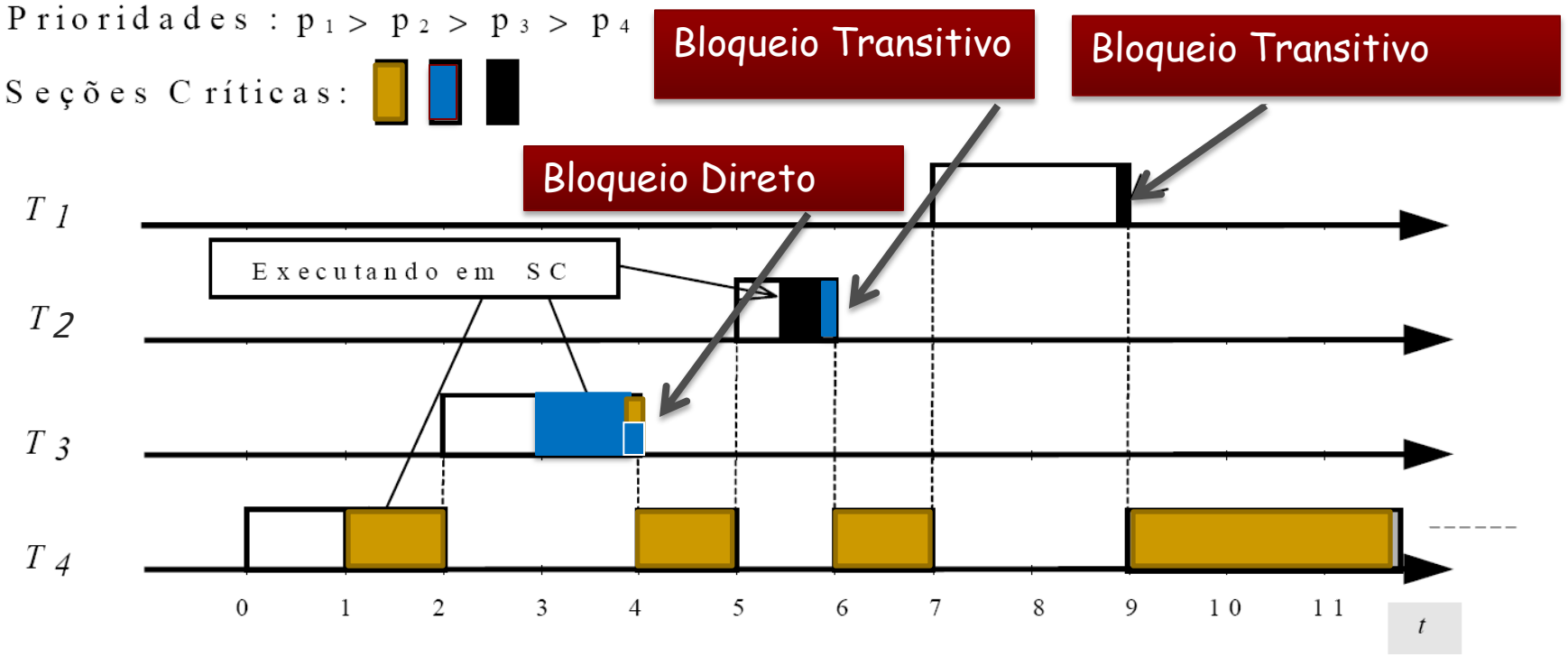
No PHP um tarefa pode sofrer três tipos de bloqueios

- Bloqueio direto
  - (Pri1 > Pri2) e compartilham recursos.
  - T2 bloqueia T1.
- Bloqueio por herança
  - (Pri1 > Pri2 > Pri3) e compartilham recursos.
  - T3 bloqueia T1 e conseqüentemente T3 bloqueia T2.
- Bloqueio transitivo
  - T1 bloqueia T2. T2 bloqueia T3. T1 bloqueia T3.

Identifique os Bloqueios



Identifique os Bloqueios



Protocolo Herança de Prioridade (PHP)

Teste de Escalonabilidade.

- Similar aos anteriores, mas leva em consideração o bloqueio máximo (B_i) de cada tarefa.
- B_i = maior sessão crítica que bloqueia T_i.

$$\left(\sum_{j=1}^{i} \frac{C_j}{P_j} \right) + \frac{B_i}{P_i} \leq i \, (2^{1/i} - 1), \quad \forall \, i$$

Protocolo Herança de Prioridade (PHP)

Verificar Escalonabilidade:

- Usar algoritmo Rate Monotonic.
- Bi = Blocking Time

Tar. Periódicas	Período	T. de Computação	B_i	Prioridade RM
Tarefa A	18	6	2	1
Tarefa B	20	4	4	2
Tarefa C	50	10	0	3

- Use $\left(\sum_{j=1}^{i} \frac{C_j}{P_j} \right) + \frac{B_i}{P_i} \leq i\,(2^{1/i} - 1), \quad \forall i$

Protocolo Herança de Prioridade (PHP)

Tar. Periódicas	Período	T. de Computação	B_i	Prioridade RM
Tarefa A	18	6	2	1
Tarefa B	20	4	4	2
Tarefa C	50	10	0	3

$$\frac{C_1}{P_1} + \frac{B_1}{P_1} \leq 1$$

$$\frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{B_2}{P_2} \leq 0,82$$

$$\frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} \leq 0,78$$

Protocolo Herança de Prioridade (PHP)

- Ou ainda poderíamos utilizar:

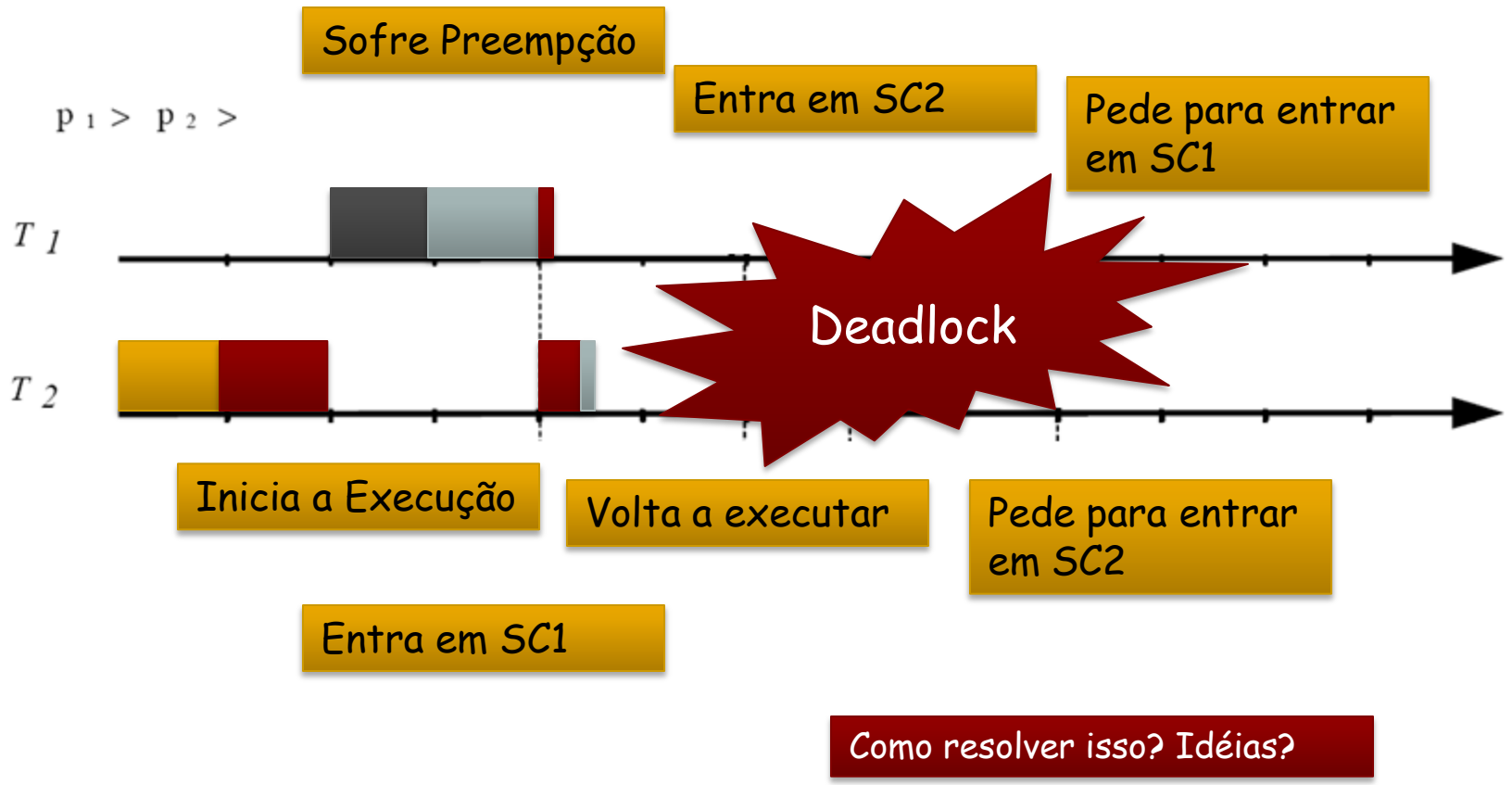
$$U_i(t) = \frac{\sum_{j=1}^{i} \left\lceil \frac{t}{P_j} \right\rceil C_j + B_i}{t}, \quad \forall i, \ min_{0<t\leq P_i} U_i(t) \leq 1.$$

- Faça o teste usando a equação acima.

Protocolo Herança de Prioridade (PHP)

- Qual o problema do PHP?
  - PHP é sujeito a *deadlock*.
  - Alguém consegue dar um exemplo?

Protocolo Herança de Prioridade (PHP)



Protocolo de Prioridade Teto

- Limita o número de bloqueios ou inversões de prioridade para evitar deadlocks.
- Dirigido para escalonamento de prioridade fixa.
- Similar ao PHP, porém corrige suas falhas.
  - Também trabalha com herança de prioridades.

Protocolo de Prioridade Teto

Tarefas possuem duas prioridades:

- Prioridade nominal ou estática
  - Definidas no RMS, DMS, etc.
- Prioridade dinâmica ou ativa
  - Igual ao PHP.

Recursos têm uma Prioridade Teto

- Prioridade da tarefa mais prioritária que acessa o recurso

Protocolo de Prioridade Teto

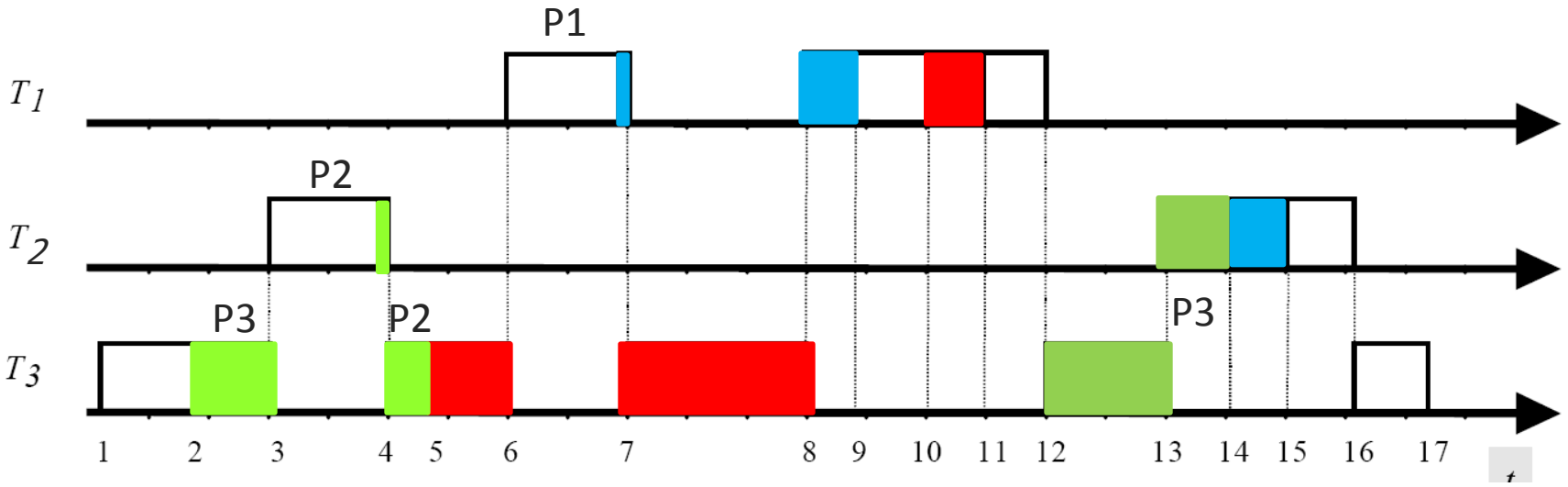
Funcionamento:

- Cada recurso possui uma prioridade teto (prioridade igual a da tarefa mais prioritária que pode alocar o recurso).
- Se nenhum recurso compartilhado está bloqueado, quem requisita é atendido.
- Se alguma tarefa bloqueia outra mais prioritária, a menos prioritária herda sua prioridade.
- No caso de haver recurso em uso:
  - Uma tarefa só acessa um recurso se sua prioridade ativa for maior que que a prioridade teto de qualquer recurso já bloqueado.

Protocolo de Prioridade Teto

Seja P1 > P2 > P3

Legenda para os recursos
- RC1 (Prioridade Teto = P1)
- RC2 (Prioridade Teto = P1)
- RC3 (Prioridade Teto = P2)



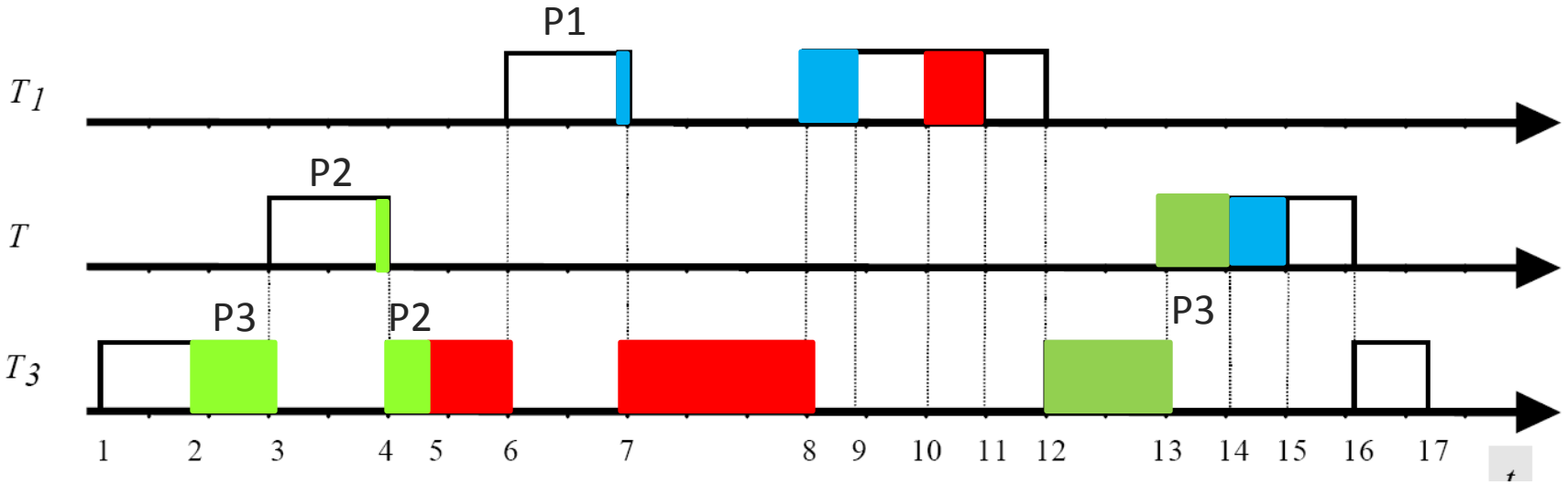
Teste de Escalonabilidade no PCP.

- Mesmas fórmulas do PHP
- Só muda o conceito de bloqueio máximo (B_i).
  - Duração da maior sessão crítica que pode bloquear pelo algoritmo de Teto.

Teste de Escalonabilidade no PCP

$$B_1 = max\ (1,4) = 4$$
$$B_2 = max\ (8) = 8$$
$$B_3 = 0$$

tarefas	S_1	S_2	S_3
tarefa T_1	1	1	0
tarefa T_2	1	0	1
tarefa T_3	0	4	8



Sistemas Operacionais de Tempo Real

- Escalonamento preemptivo
- Escalona processos com base em prioridades (não é justo, pode provocar *starvation*)
- Todas as chamadas ao S.O. tem tempo máximo de execução definido e otimizado
- A mudança de contexto tem tempo limitado, conhecido (*fixed overhead*) e otimizado
- Tratamento de inversão de prioridade

Referências

- **Livro de Sistemas de Tempo Real**
  Jean- Marie Farines, Joni da Silva Fraga, Rômulo Silva de Oliveira. Escola de Computação'2000 - IME- USP
  http:// www. lcmi. ufsc. br/ gtr/ livro/ principal. Htm
- **IEEE Computer Society, Technical Committee on Real- Time Systems (IEEE- CS TC- RTS)**
  http:// www. cs. bu. edu/ pub/ ieee- rts
- The Concise Handbook Of Real-Time Systems. TimeSys Corporation, Versão 1.1, 2000.
  http://www.timesys.com