Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento Acadêmico de Eletrônica (DAELN)

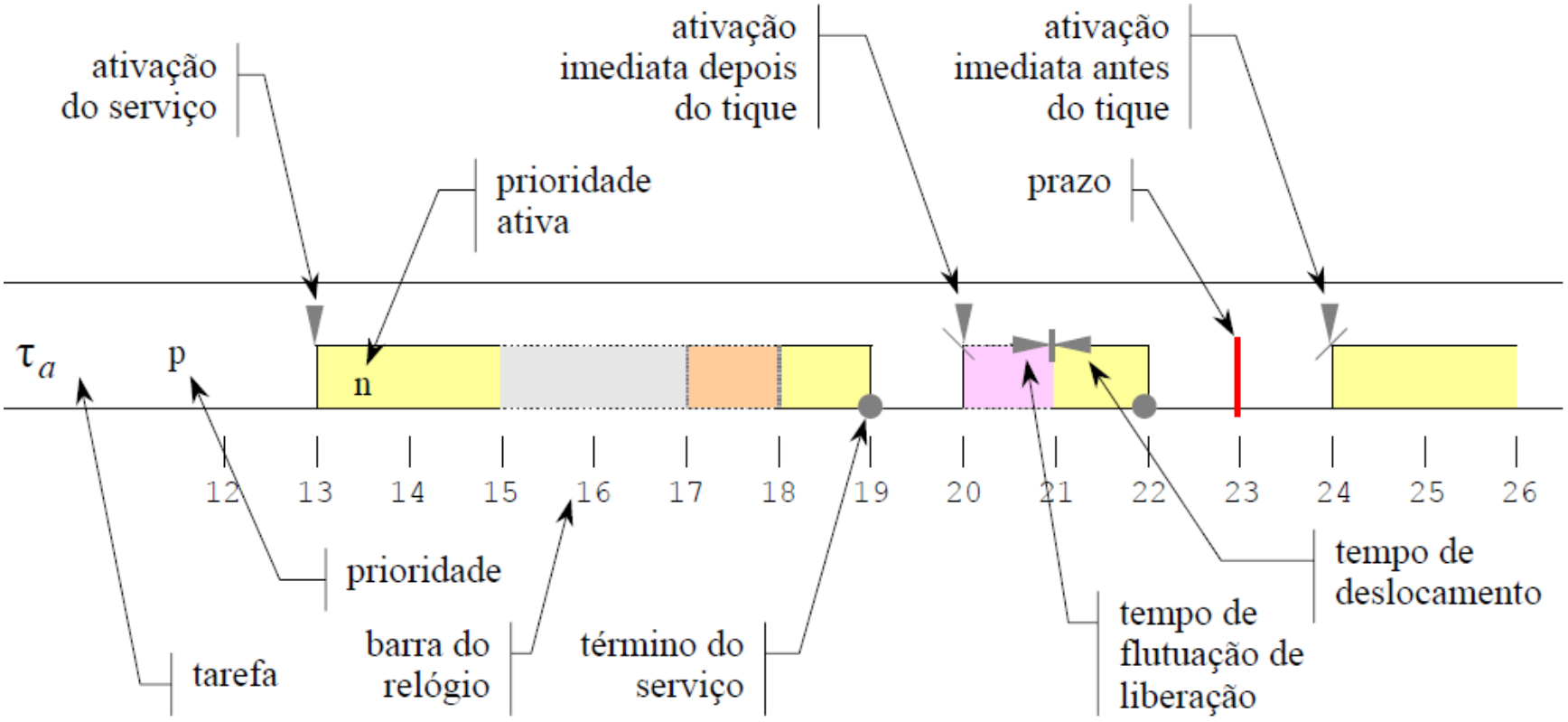
SISTEMAS EMBARCADOS

Escalonamento e Tempo Real

Prof. André Schneider de Oliveira

andreoliveira@utfpr.edu.br

Diagrama de Gantt



Escalonamento de processos

- O escalondor de processos é responsável por determinar qual tarefa deve ganhar o processador, possibilitando a execução de diferentes tarefas em sistemas *monocore*

- O gerenciamento das tarefas deve respeitar o compartilhamento de recursos e a comunicação entre tarefas

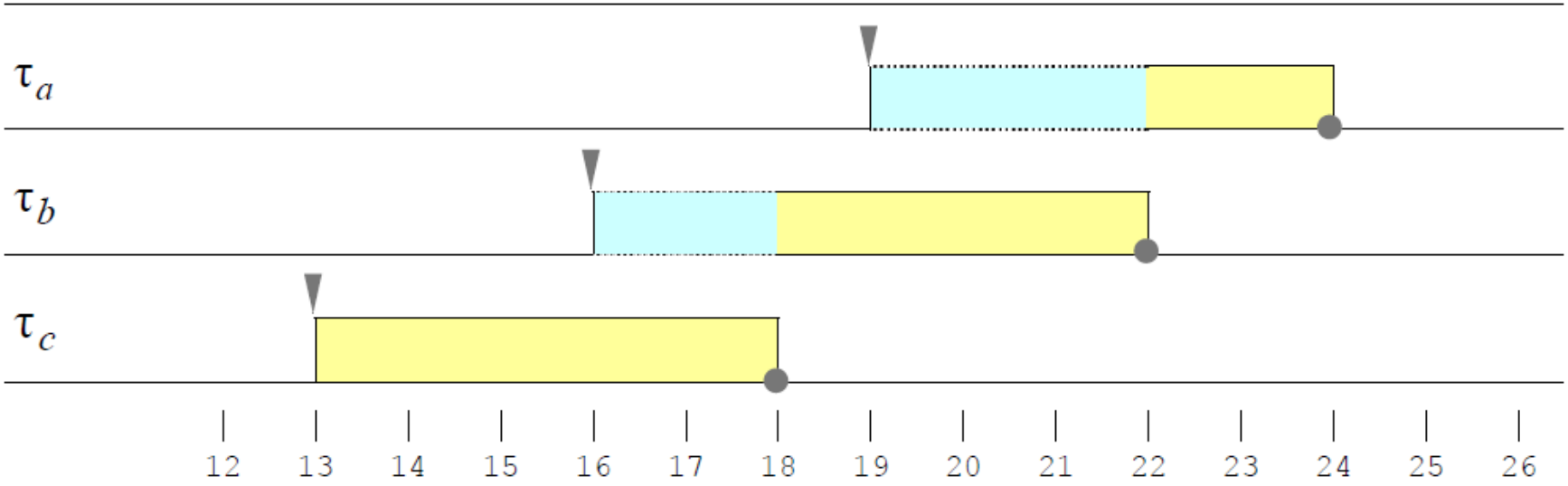
- O escalonador deve escolher a tarefa de acordo com critérios pré-estabelecidos (prioridade, deadline, ...), dentre os processos prontos (*ready*) e mudar o status da tarefa para execução (running)

- A preempção pode ser utilizada para retirar um processo em execução e colocar outro mais prioritário

Métodos de Escalonamento

- Preemptivo x Não-preemptivo
- Estático
  - Cooperativo, a cargo do programador
- Dinâmico
  - Prioridade fixa
    - Não-preemptivo (*ex: First Come First Served*)
    - Preemptivo (*ex: Round Robin*)
  - Prioridade variável

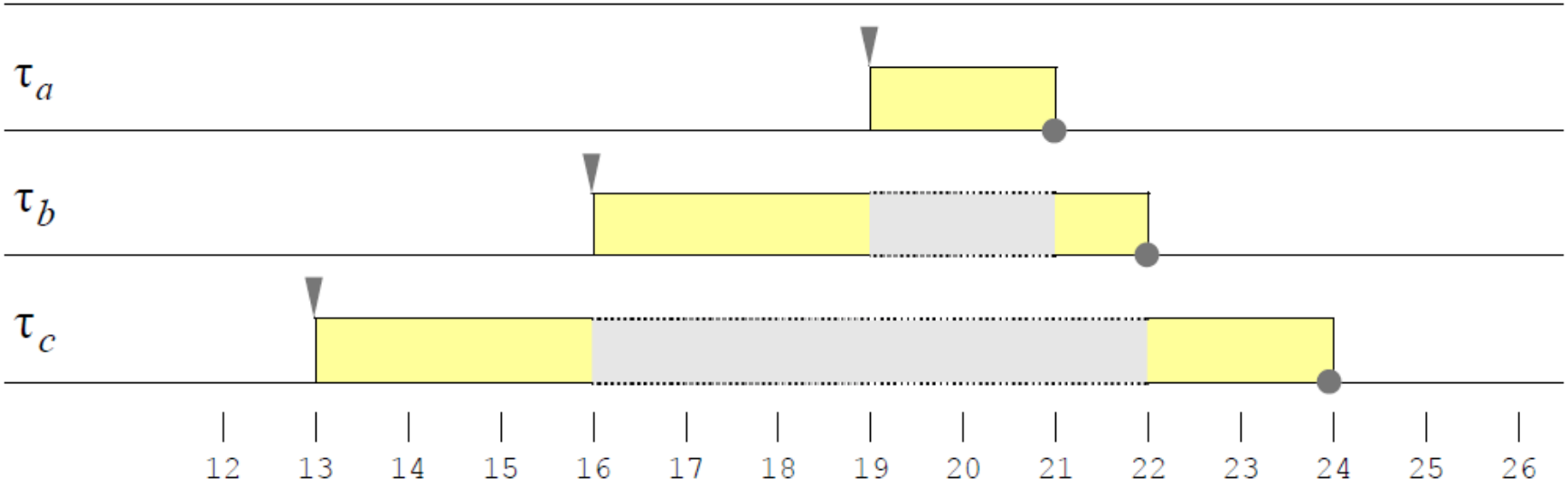
Escalonamento Não-preemptivo



Prioridades $\tau_a > \tau_b > \tau_c$

*** a execução não é interrompível**

Escalonamento Preemptivo



Prioridades $\tau_a > \tau_b > \tau_c$

*** a execução é interrompível**

First Come First Served

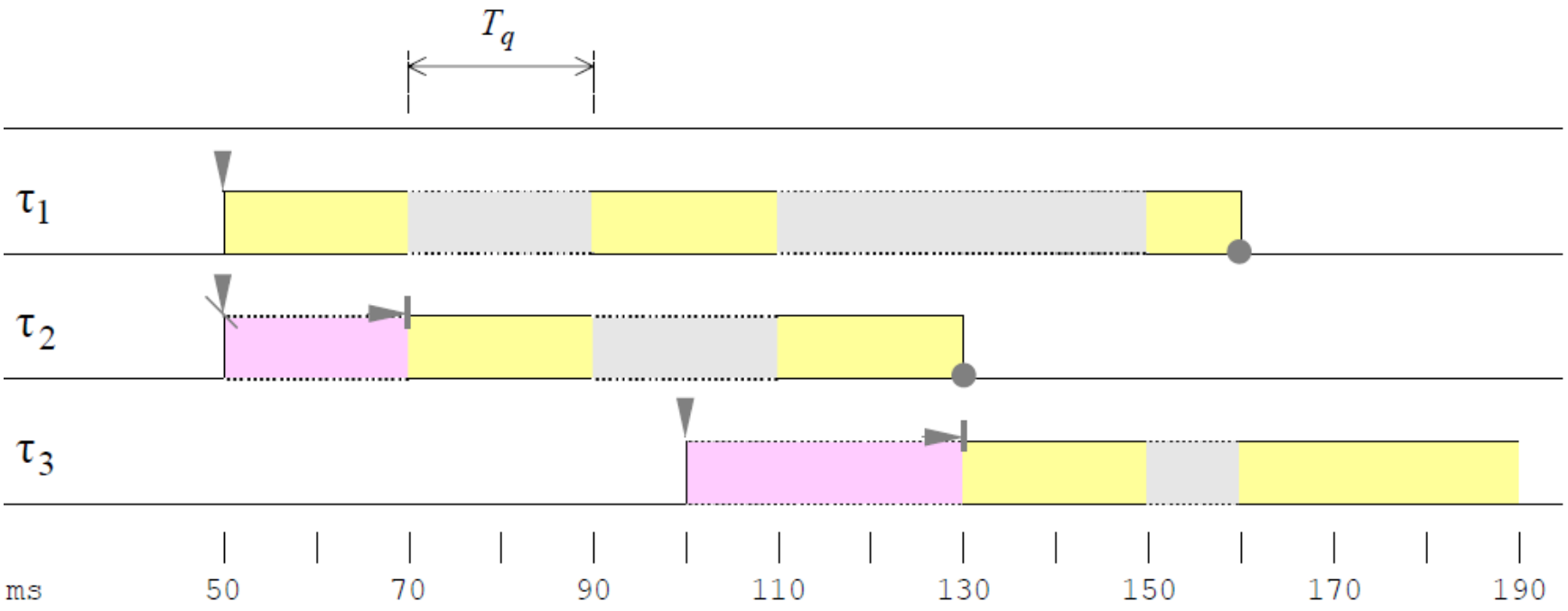
- Escalonamento dinâmico não-preemptivo em que a ordem de execução dos serviços é determinada pela **ordem em que entram no estado PRONTO**.

- Raramente utilizado em aplicações em tempo real, a não ser como meio de desempate entre serviços de uma mesma fila de prioridade em escalonadores mais complexos.

Round Robin

- Escalonamento dinâmico preemptivo semelhante ao FCFS no tratamento da ordem de execução dos serviços.
- No entanto, cada serviço recebe o controle do processador por um **intervalo de tempo máximo** Tq.
- Se o serviço **não terminar dentro do intervalo Tq, será preemptido** e terá que aguardar o próximo intervalo para continuar.
- Serviços podem ceder voluntariamente o controle do processador antes do final do seu intervalo de tempo.

Exemplo – Round Robin



Round Robin

- Desempenho fortemente influenciado por
  - Slice de tempo (Tq)
  - pelo tempo de execução das tarefas
  - pelo número de tarefas ativadas

- Resultado final situa-se entre comportamento semelhante ao
  - FCFS (Tq ≥ tempo de execução da tarefa mais demorada)
  - ineficiência total (Tq ≤ tempo de execução do mecanismo de preempção).

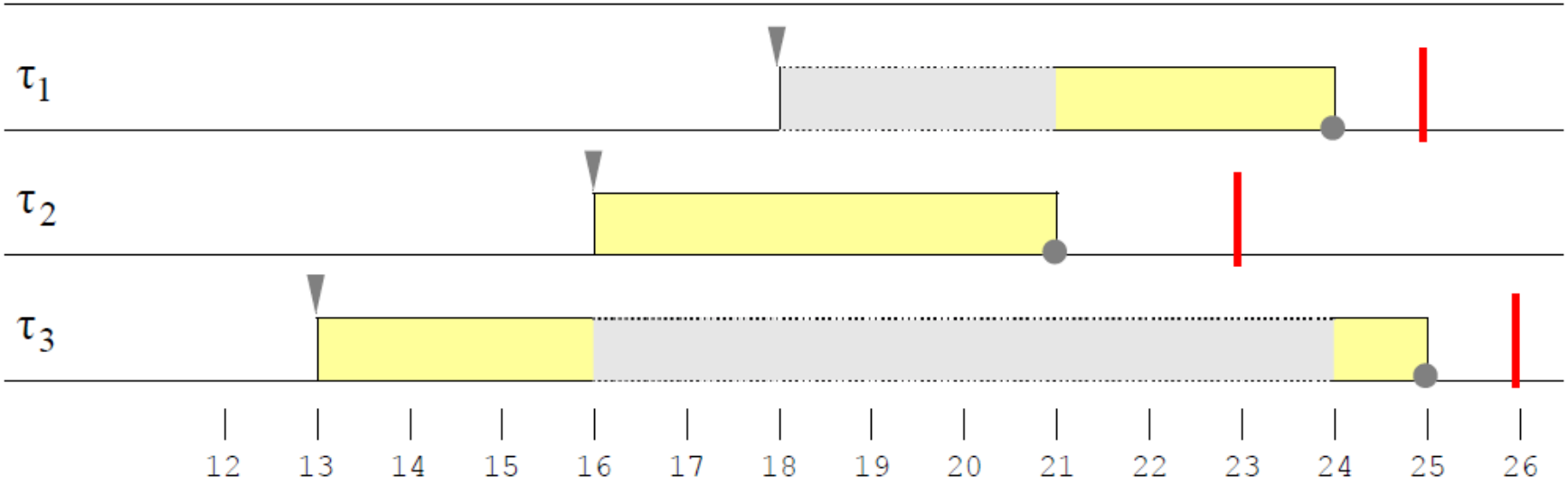
Earliest Deadline First

- Seleciona o serviço com prazo terminando mais cedo dentre o conjunto de serviços que estão no estado **PRONTO**.

- **Não-preemptivo:** o serviço selecionado executa até o final, até entrar no estado **SUSPENSO/ BLOQUEADO** ou até entrar voluntariamente no estado **PRONTO**.

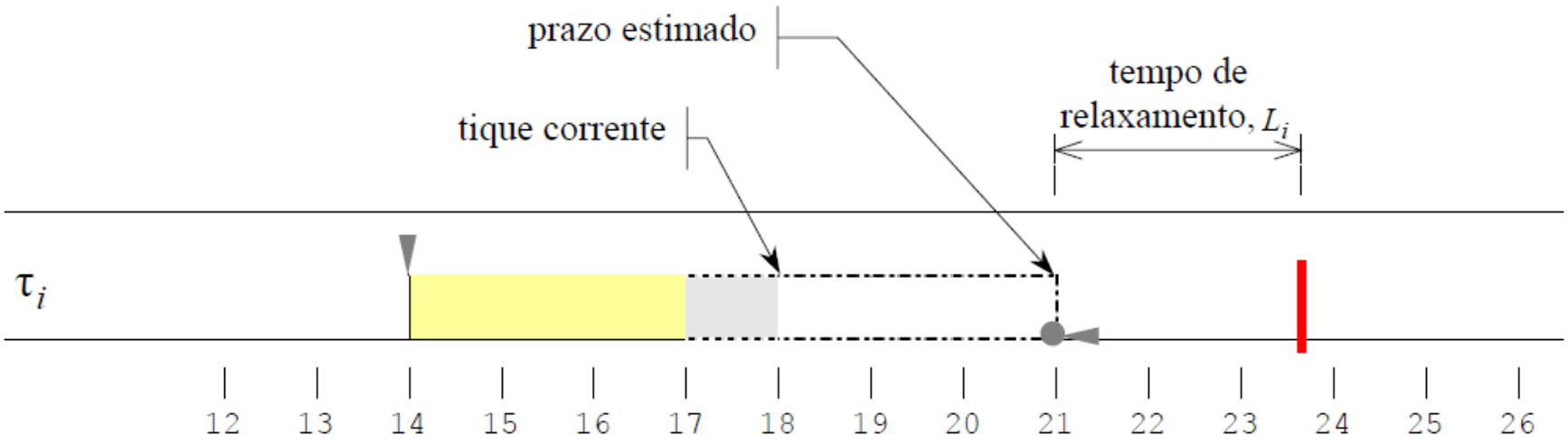
- **Preemptivo:** o serviço pode ser preemptido por outro serviço com prazo terminando mais cedo.

Exemplo – EDF Preemptivo



Least Laxity First

- **Tempo de relaxamento:** diferença de tempo entre o prazo efetivo e o prazo estimado do serviço.



Least Laxity First

- **Prioridade determinada dinamicamente**, de modo que o serviço com **menor tempo de relaxamento tenha a maior prioridade**.
- Enquanto o serviço está no estado EXECUTANDO, seu prazo permanece inalterado.
- No entanto, quando está no estado **PRONTO** ou **SUSPENSO/BLOQUEADO**, seu prazo estimado aumenta e portanto seu tempo de relaxamento diminui.
- Em algum momento no futuro será o serviço com maior prioridade.

Least Laxity First

- Quando duas ou mais tarefas possuem tempos de relaxamento muito próximos, acontece um efeito do tipo **“pingue-pongue”,** causando alto número de preempções que afetam negativamente o desempenho.

Highest Priority First

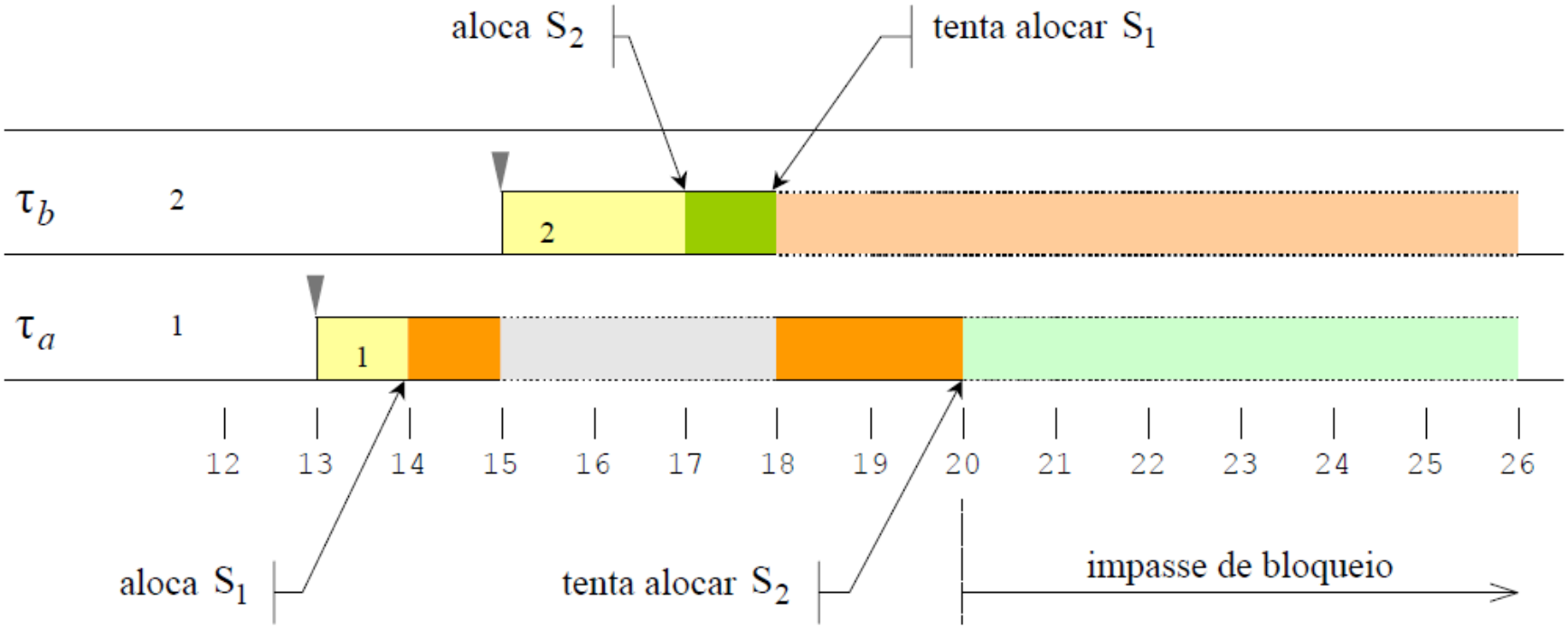
- Determina a ordem de execução das tarefas pela prioridade (pré-determinadas no projeto).

- Critérios para atribuição de prioridades:
  - Importância e periodicidade das tarefas
    - Prioridades estáticas pré-determinadas
  - Análise Monotônica de Taxa (RMA)
    - Prioridades maiores para serviços com **períodos** menores
  - Análise Monotônica de Prazo (DMA)
    - Prioridades maiores para serviços com **prazos** menores

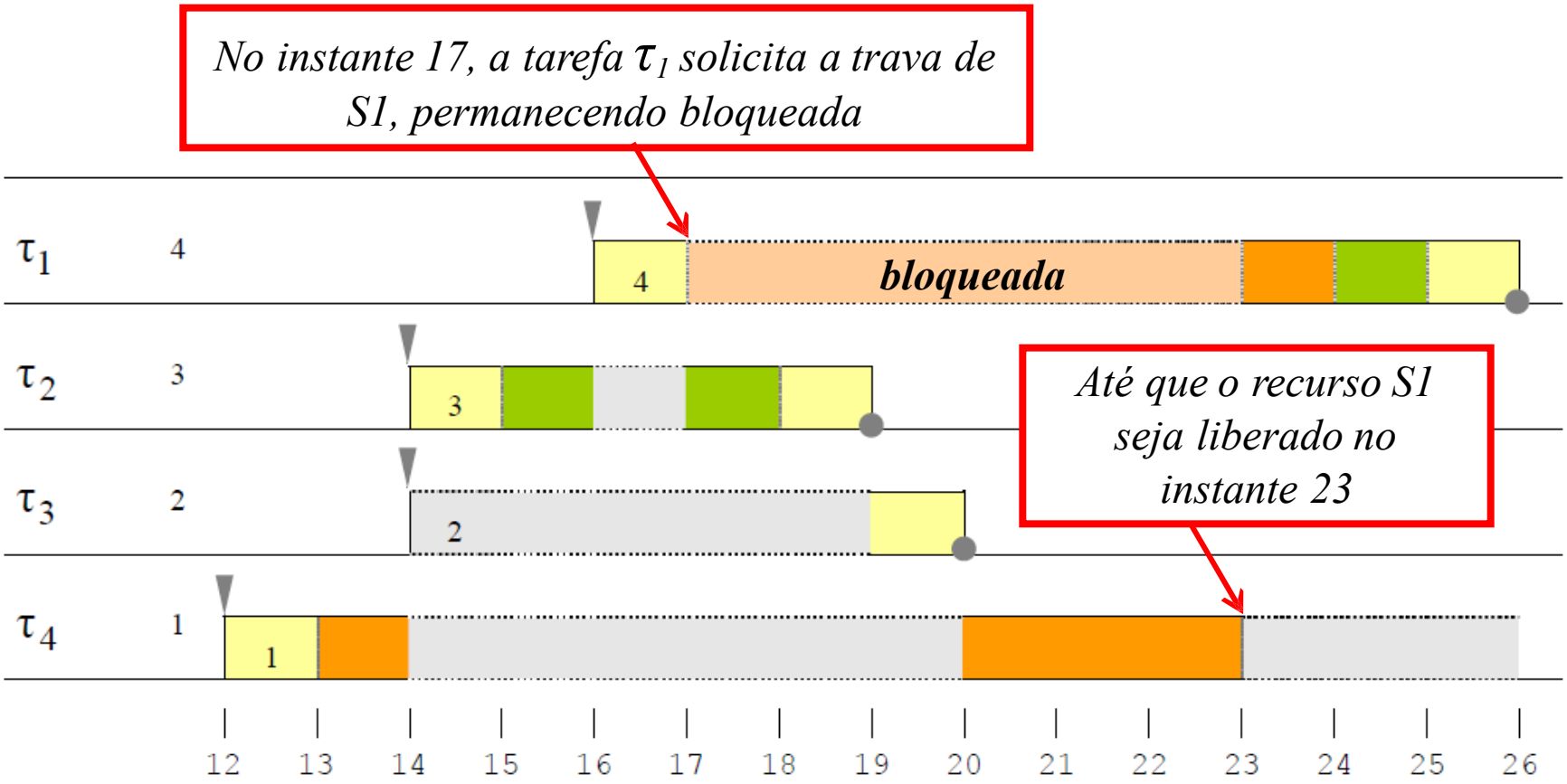
Escalonamento com Bloqueio

- Recursos compartilhados
  - Mutexes
  - Semáforos
  - Sincronização
- Problemas
  - Impasse de bloqueio (*deadlock*)
  - Inversão de prioridades
- Soluções
  - Protocolo de herança de prioridade
  - Protocolo de teto de prioridade

Impasse de Bloqueio



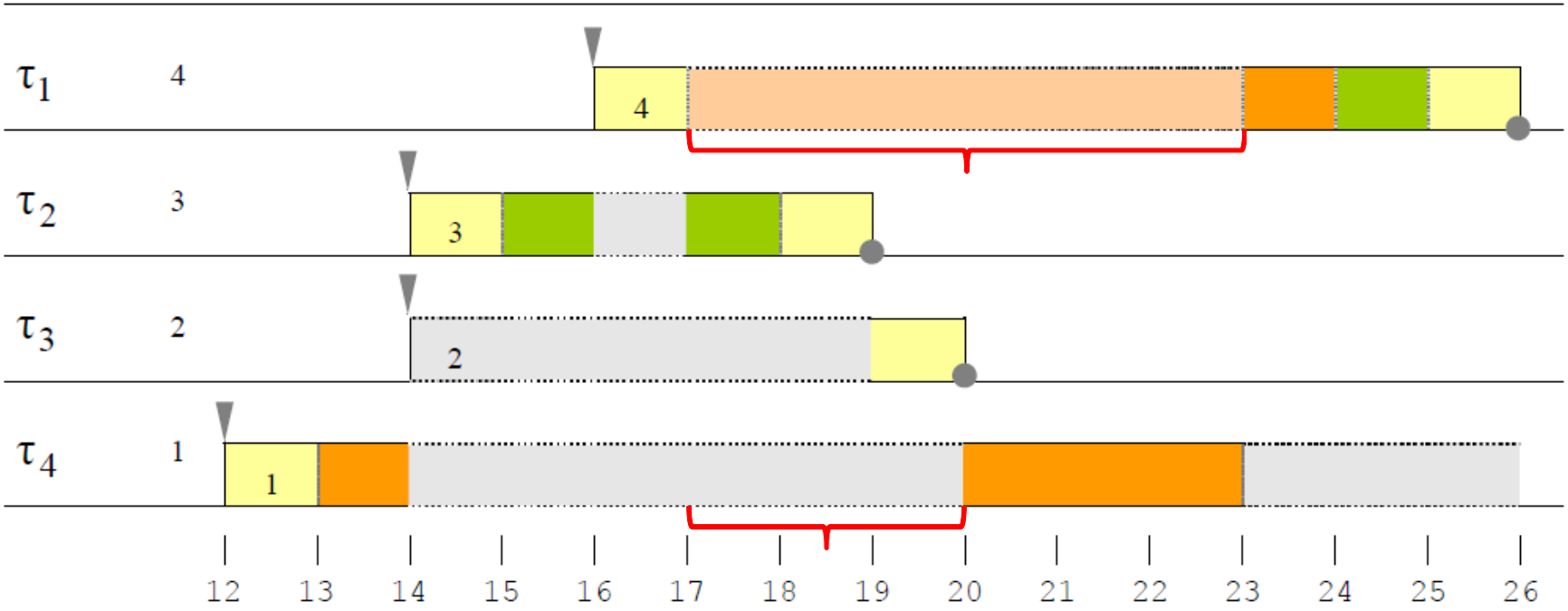
Inversão de Prioridades



Prioridades $\tau_1 > \tau_2 > \tau_3 > \tau_4$

Inversão de Prioridades

Prioridades $\tau_1 > \tau_2 > \tau_3 > \tau_4$



O tempo de bloqueio do recurso S1 é igual a 6, sendo fortemente influenciado pela interferência das tarefas τ2 e τ3 no tempo de resposta da tarefa τ4

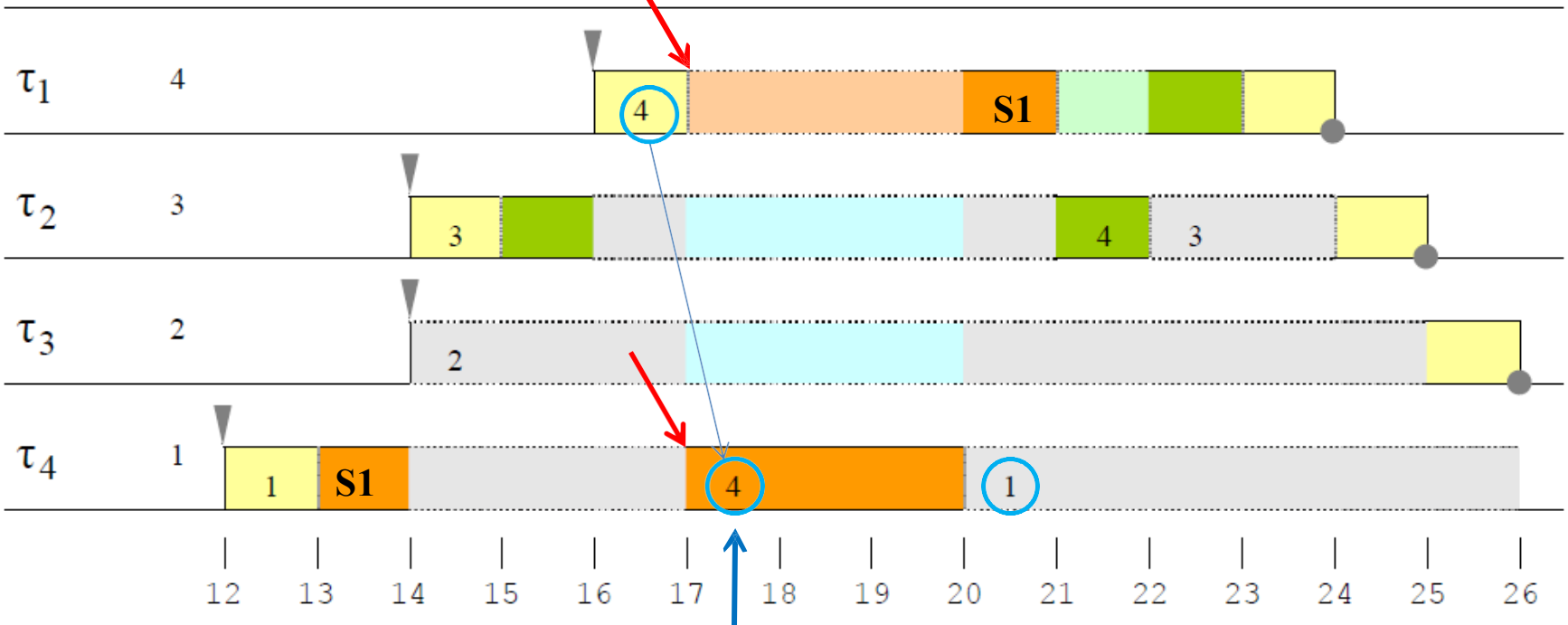
Priority Inheritance Protocol

- Sob o Protocolo de Herança de prioridade, uma tarefa possui **uma prioridade básica atribuída estaticamente e uma prioridade ativa.**

- A prioridade ativa de uma tarefa durante a seção crítica é igual ao máximo entre sua prioridade básica e a **maior prioridade ativa do conjunto de tarefas bloqueadas** que esperam pela liberação do recurso.

Priority Inheritance Protocol

No instante 17 a tarefa τ_1 requisita a trava do recurso S1, que está bloqueado pela tarefa τ_4



A tarefa τ_4 herda a prioridade da tarefa τ_1 e passa a ser executada, retomando sua prioridade original após a liberação do recurso em questão

Priority Ceiling Protocol

- O Protocolo de Herança de Prioridade **não previne impasses de bloqueio** e além disso pode levar à **formação de cadeias de bloqueio**.

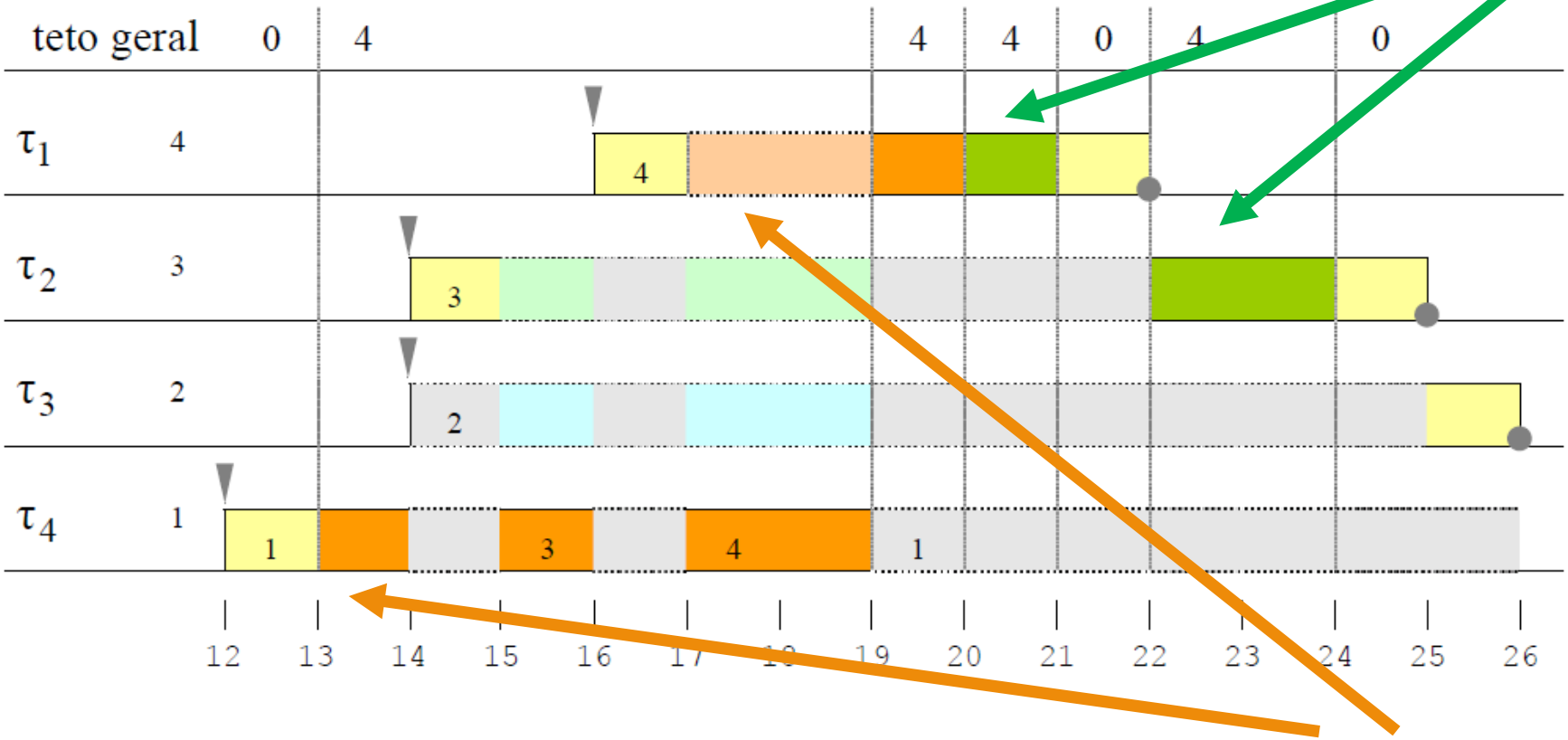
- Sob o Protocolo de Teto de Prioridade, o recurso tem seu teto de prioridade igual à **maior prioridade básica das tarefas em que é utilizado (teto geral).**

- A prioridade das tarefas é definida como no Protocolo de Herança de Prioridade.

- Uma tarefa obtém a trava de um recurso livre se **sua prioridade ativa for maior que o teto geral** ou **se detém a trava do recurso que fixou o teto geral**.

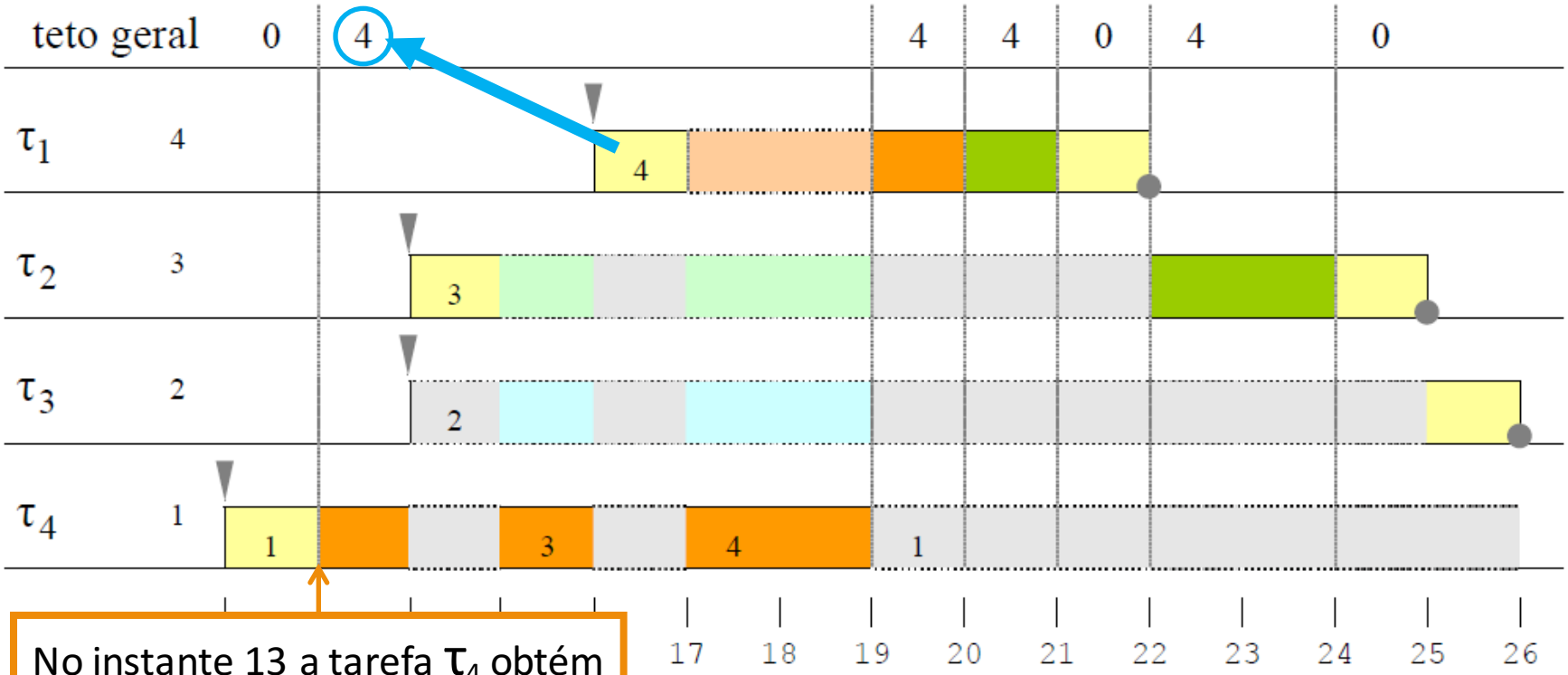
Priority Ceiling Protocol

O recurso S2 (verde) também tem teto de prioridade igual a 4 (utilizado nas tarefas τ_1 e τ_2)



O recurso S1 (laranja) tem teto de prioridade igual a 4 (utilizado nas tarefas τ_4 e τ_1)

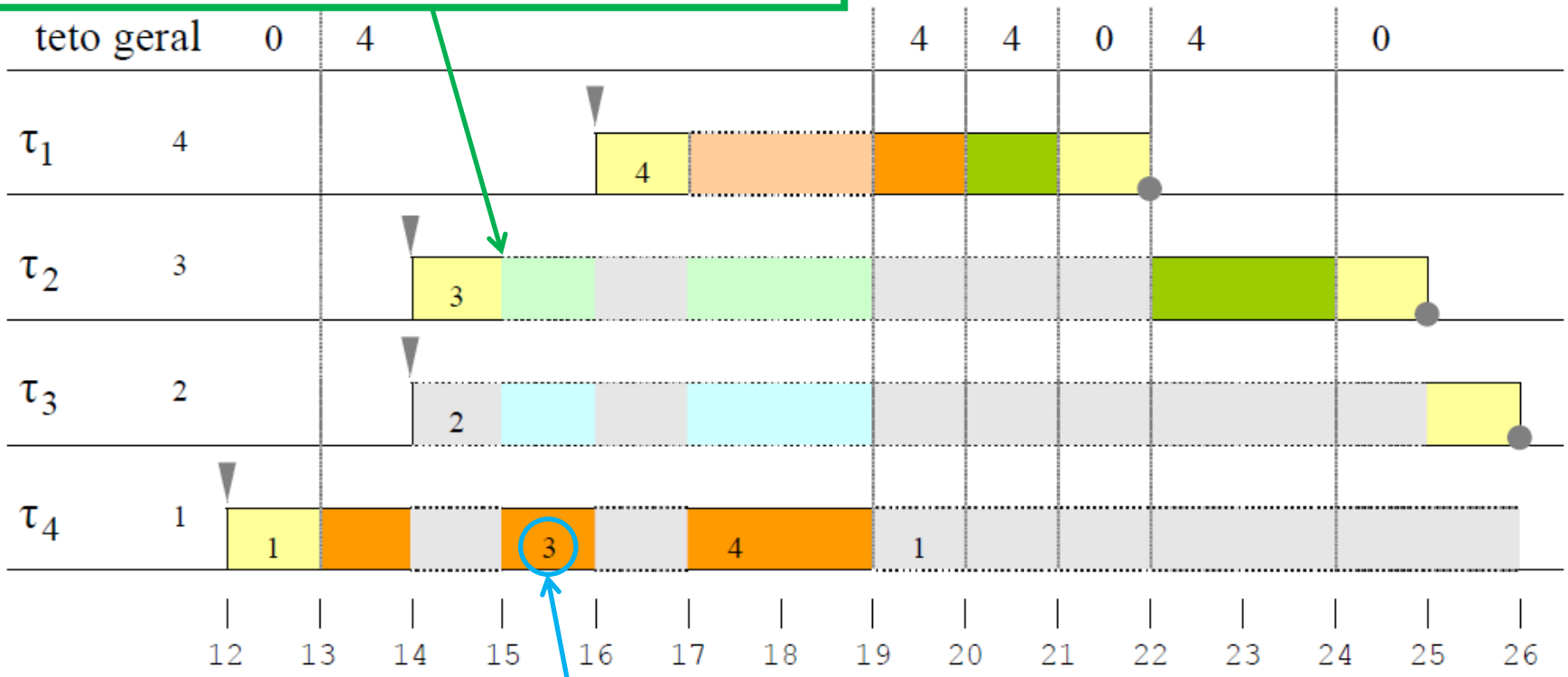
Priority Ceiling Protocol



No instante 13 a tarefa τ_4 obtém a trava do recurso **S1** e o teto geral passa para o teto de prioridade desse recurso

Priority Ceiling Protocol

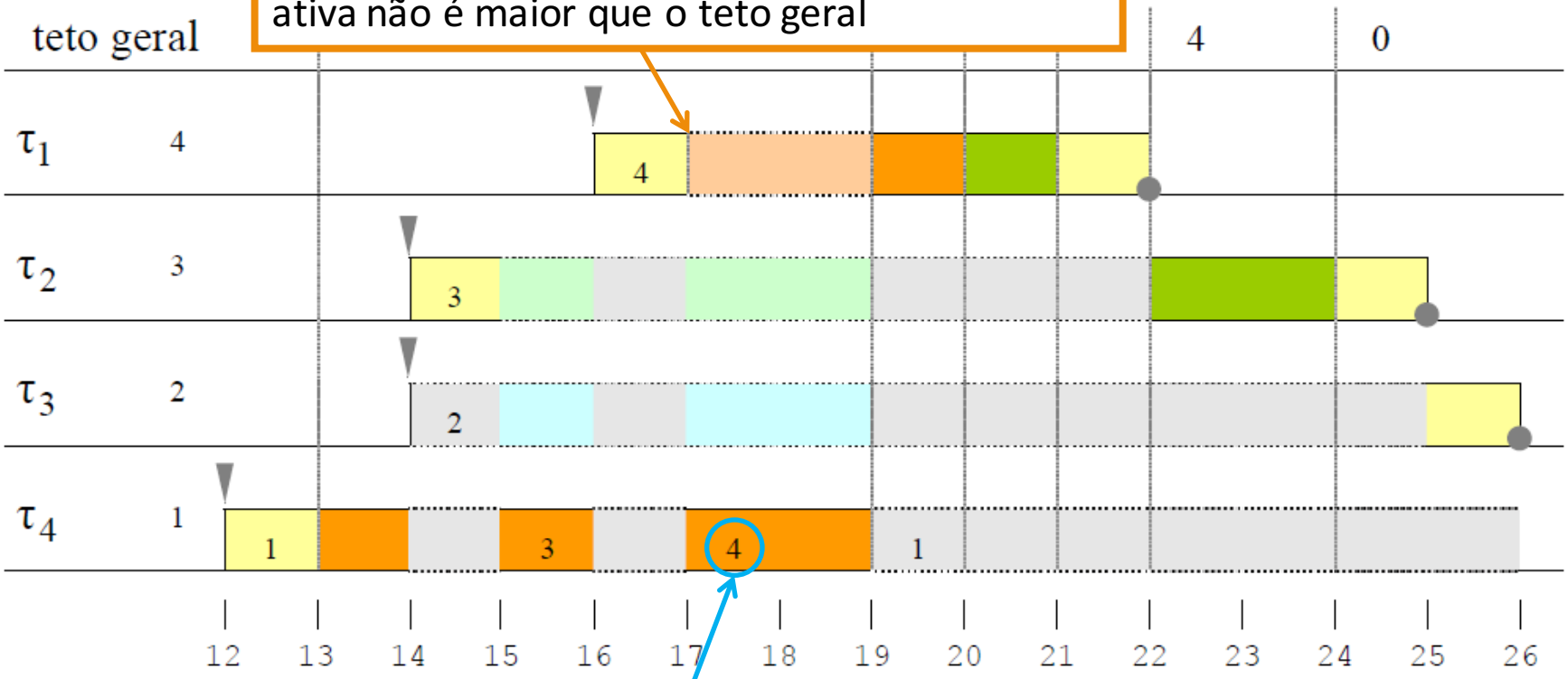
No instante 15 a tarefa τ_2 requisita a trava do recurso **S2** e não consegue, pois sua prioridade ativa não é maior que o teto geral



a prioridade ativa da tarefa τ_4 passa para o máximo das prioridades ativas das tarefas τ_2 e τ_4, envolvidas com o recurso em questão

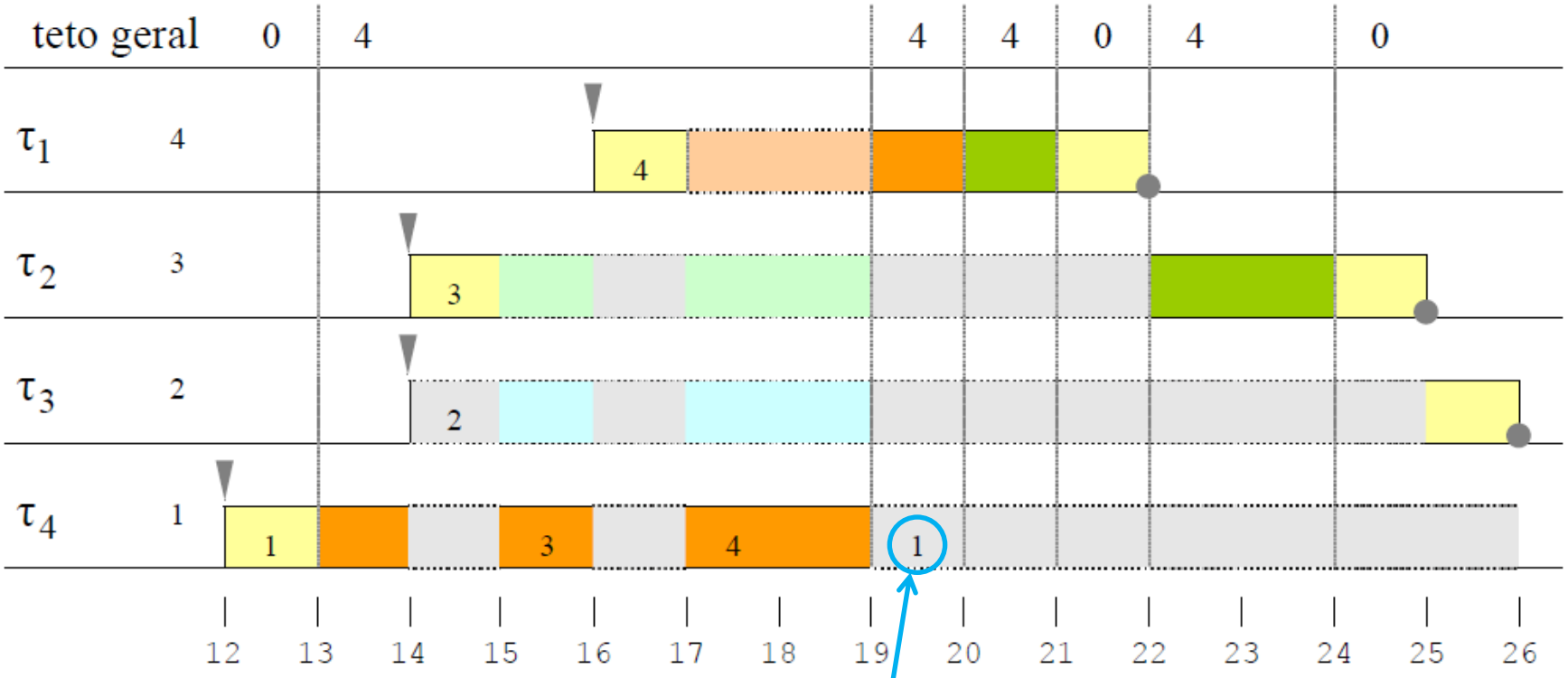
Priority Ceiling Protocol

No instante 17 a tarefa τ_1 requisita a trava do recurso **S1** e não consegue, pois sua prioridade ativa não é maior que o teto geral



a prioridade ativa da tarefa τ_4 passa para o máximo das prioridades ativas das tarefas τ_1 e τ_4, envolvidas com o recurso em questão

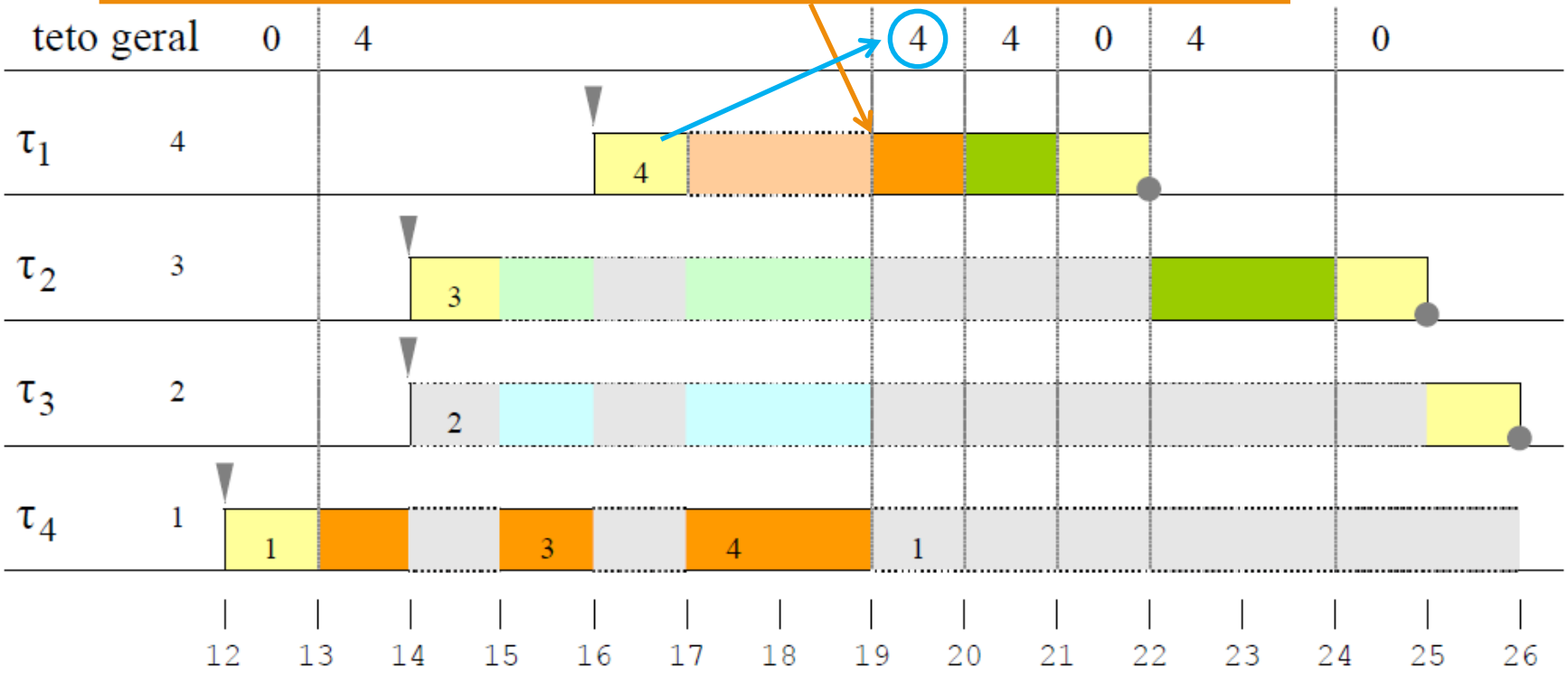
Priority Ceiling Protocol



No instante 19 a tarefa τ_4 libera o recurso **S1** e sua prioridade ativa volta para a sua prioridade básica

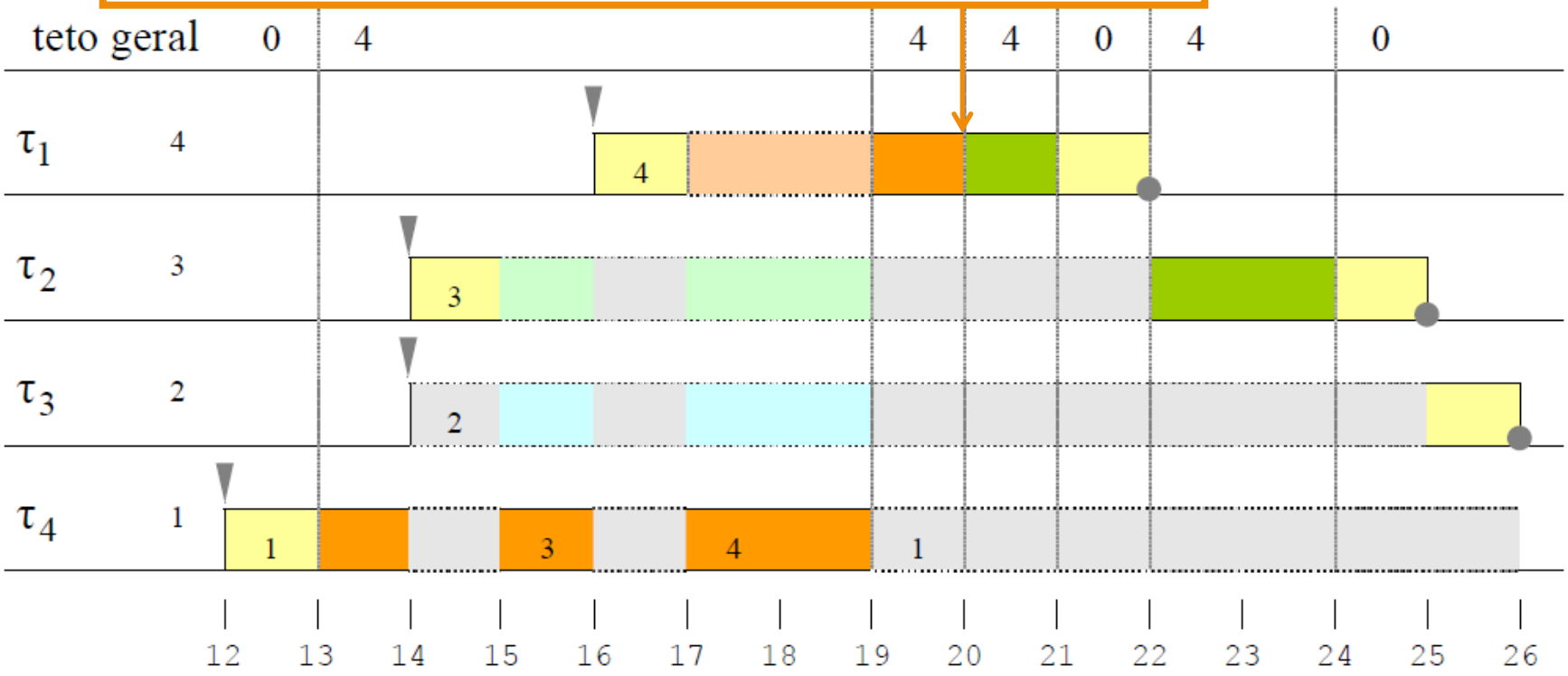
Priority Ceiling Protocol

Ainda no instante 19 a tarefa τ_1 obtém a trava do recurso **S1** e o teto geral passa para o teto de prioridade do recurso em questão



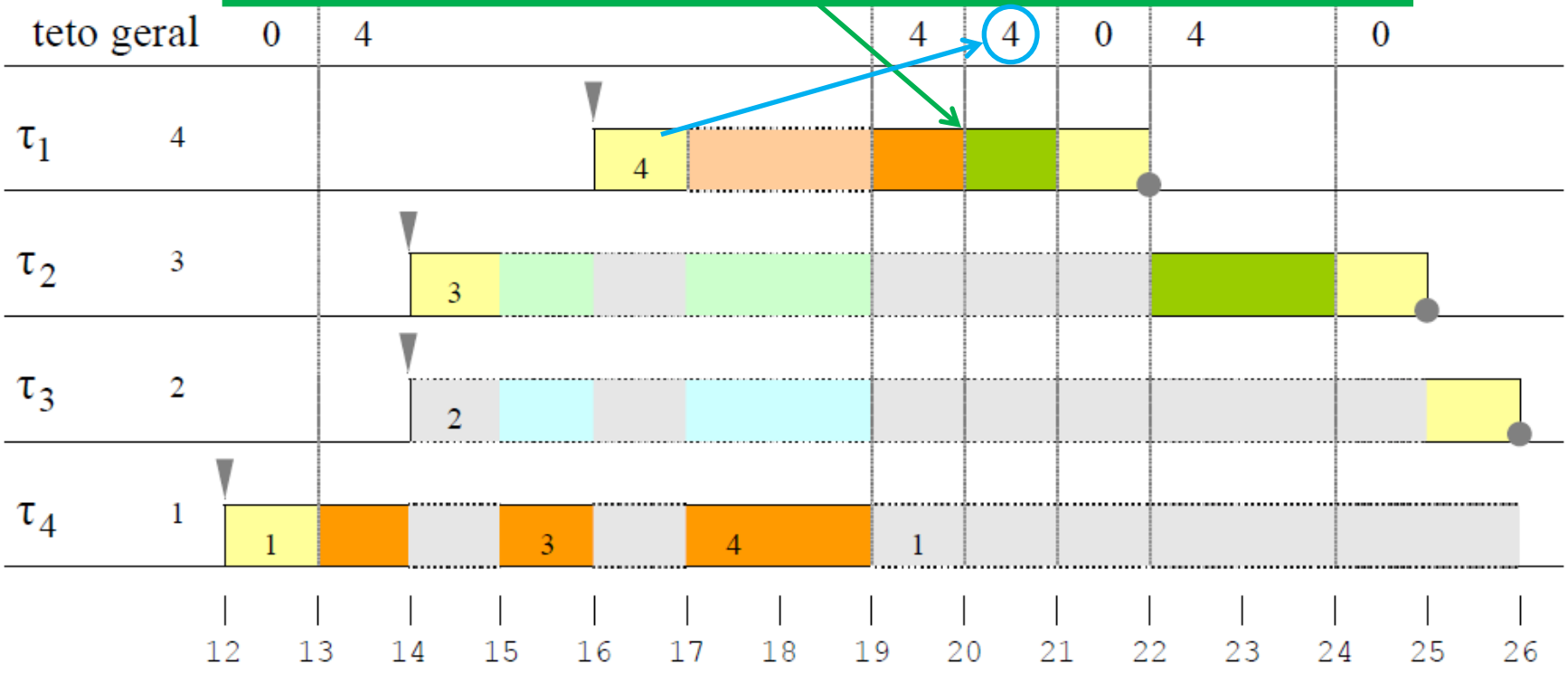
Priority Ceiling Protocol

No instante 20 a tarefa τ_1 libera o recurso **S1** e sua prioridade ativa volta para a sua prioridade básica (a mesma)

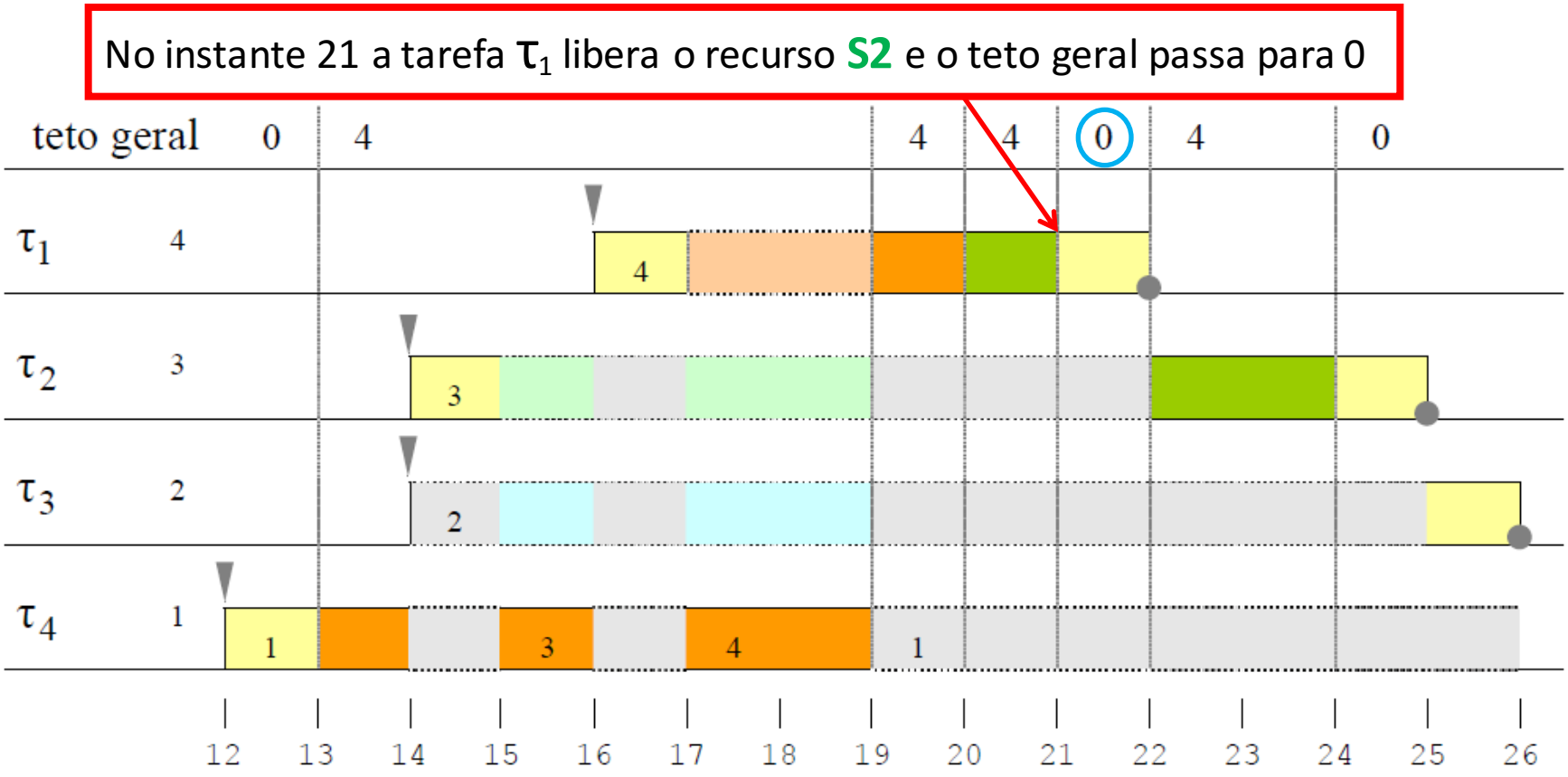


Priority Ceiling Protocol

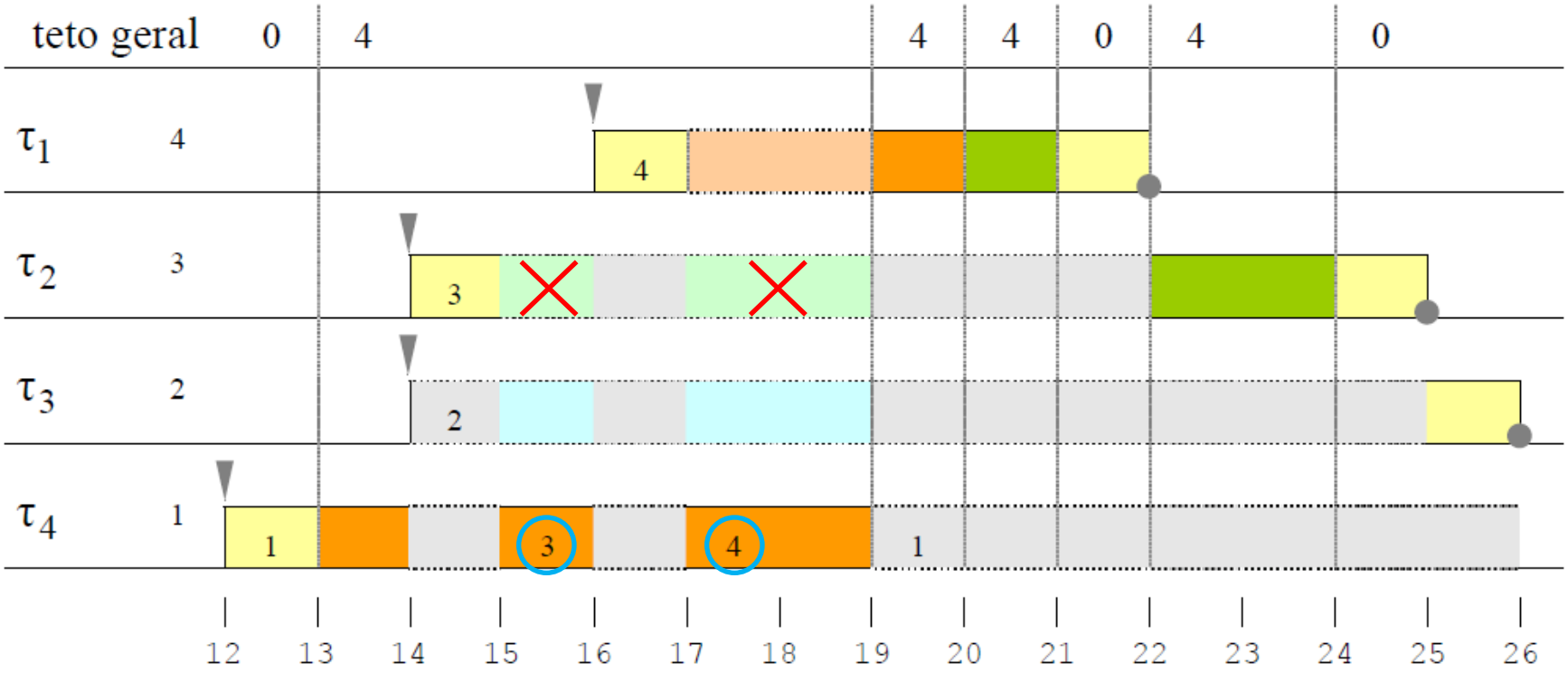
Ainda no instante 20 a tarefa τ_1 obtém a trava do recurso **S2** e o teto geral passa para o teto de prioridade do recurso em questão



Priority Ceiling Protocol



Priority Ceiling Protocol



O que acontece nos instantes 15 e 17 previne impasses de bloqueio, simplesmente não permitindo que outros recursos com teto de prioridade menor ou igual ao teto geral (recurso S2 nesse exemplo) sejam travados.

Configurações do RTOS para escalonamento

- Arquivo **RTX_Conf_CM.c**

Configurações de Escalonamento

•**OS_ROBIN** = 1 habilita o escalonamento por round-robin

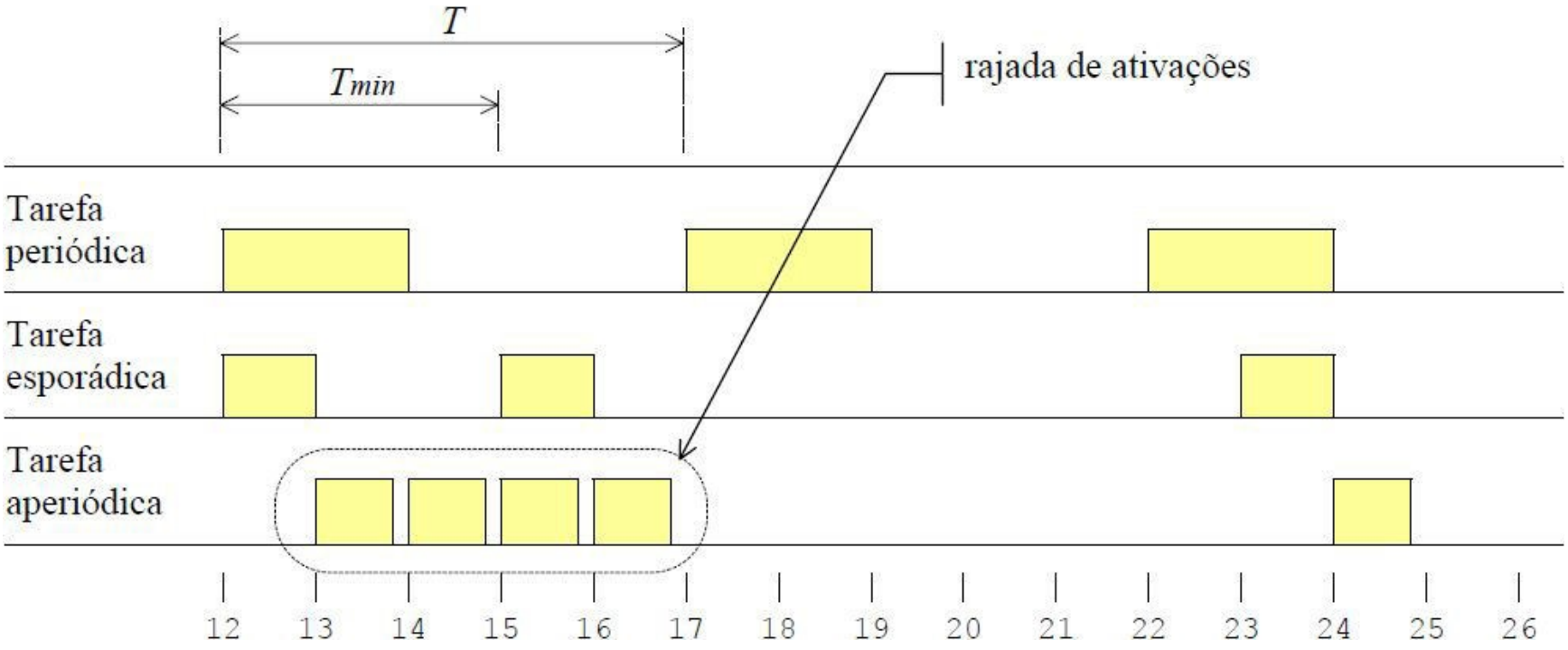
•**OS_ROBIN** = 0 desabilita o escalonamento por round-robin e emprega o escalonamento por timer/evento

•**OS_ROBINTOUT** = define o intervalo para o escalonamento round-robin [em ticks] <padrão 5>

Sistemas em Tempo Real

- Executam de acordo com restrições temporais
- Os processos serão definidos com duas especificações temporais
  - **Liberação (release) -** tempo (ticks) em que o processo fica pronto para execução (ready)
  - **Prazo (deadline)** - tempo (ticks) que o processo deve finalizar
- As restrições (deadline) podem repetir-se em
  - **Periódicas** - intervalos de tempo fixos
  - **Aperiódicas** - aleatoriamente

Sistemas em Tempo Real



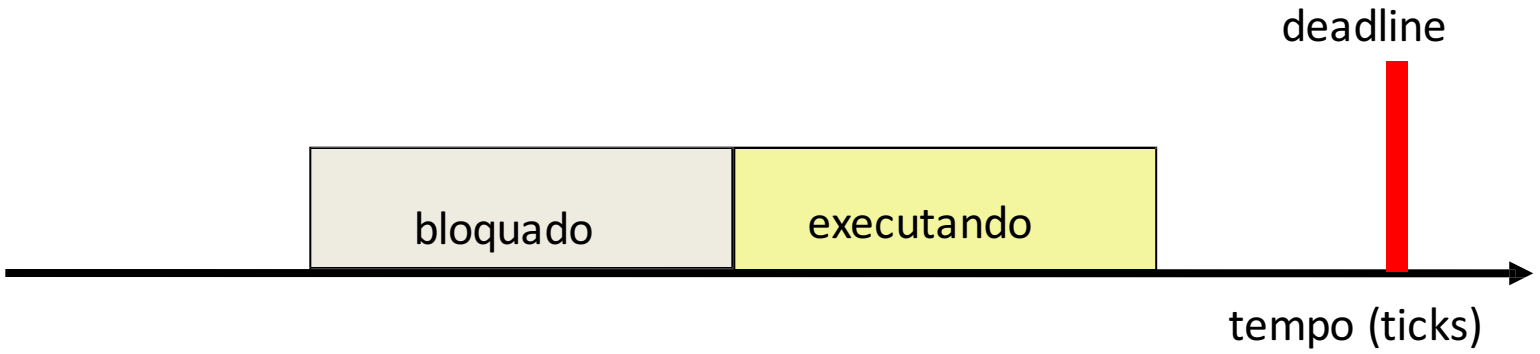
Sistemas em Tempo Real

- O não cumprimento de um deadline de execução de uma determinada tarefa gera penalizações

- A não execução de uma tarefa dentro do prazo determinado (deadline) acarreta
  - **Hard** - na falha geral do sistema
  - **Soft** - na degradação da resposta do sistema
  - **Firm** - na degradação da resposta do sistema, mas algumas respostas atrasadas podem ser toleradas

Sistemas em Tempo Real

- O processo deve ser finalizado antes do **deadline** para não sofrer penalizações



Sistemas em Tempo Real

- O que acontece se o processo não terminar antes do seu deadline (violação de tempo)?
  - **Hard deadline** - ocorre uma falha geral
  - **Soft deadline** - o usuário nota uma falha, mas não ocorre uma falha geral no sistema

Referências

- Luguesi, J. L., **Ambiente de Apoio ao Ensino e Aprendizado do Escalonamento em Sistemas em Tempo Real**. Dissertação de Mestrado, UTFPR, 2006
  - Capítulos 2 e 3, especialmente

- Liu, C. L.; Layland, J. W., **Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment**. Journal of the Association for Computing Machinery 20(1), pp. 46-61, 1973

- Sha, L.; Rajkumar, R.; Lehoczky, J. P., **Priority Inheritance Protocols: An Approach to Real-Time Synchronization**. In IEEE Transactions on Computers 39(9), pp. 1175-1185, 1990