Sistemas Distribuídos

Aula 6 – Processos, Tarefas de Tempo Real e Escalonamento

Prof. Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

Processos e Tarefas de Tempo Real

Definições e Restrições Temporais

Processo (Job)

Unidade básica de trabalho. Ex.: transmissão de uma mensagem, leitura de um sensor, cálculo de uma função de controle.

Tarefa

Conjunto de Jobs relacionados. Cada ativação da tarefa gera um novo Job.

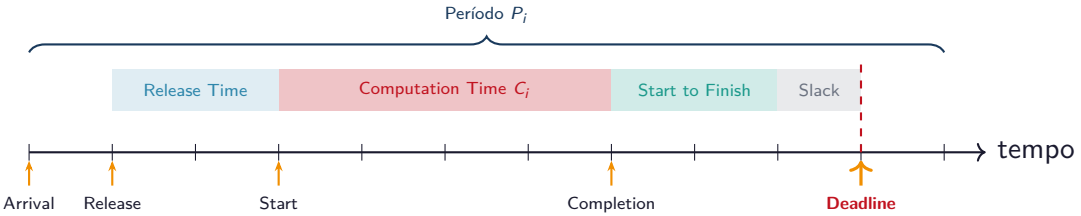
Sistemas de Tempo Real

Caracterizados por **restrições temporais** que devem ser cumpridas para que o sistema permaneça em um estado consistente. A violação de um **deadline** pode causar falhas catastróficas.

Restrições Temporais

Arrival Time	Instante em que o escalonador toma conhecimento da tarefa
Release Time	Instante em que o Job começa a poder executar
Computation Time C_i	Tempo necessário para execução completa
Start Time	Instante de início do processamento
Completion Time	Instante de conclusão da execução
Deadline	Limite até o qual deve terminar
Período P_i	Tempo entre execuções consecutivas

Linha do Tempo de uma Tarefa de Tempo Real



Dependência entre Processos

Precedência, Dados Compartilhados e Exclusão Mútua

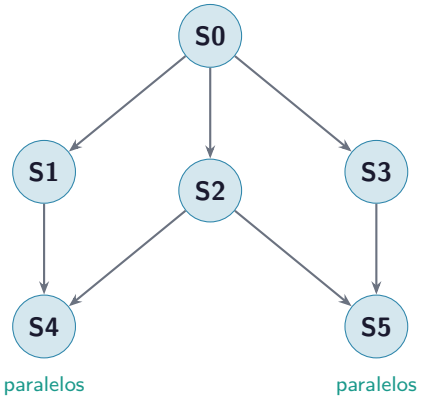
▪ Dependência Explícita

P_1 chama P_2, gerando um **grafo de precedências**.

Grafo dirigido que expressa relações imediatas de precedência:

- Nós: execuções de comandos
- Arcos: precedência imediata

Processos sem relação de precedência são **paralelos/concorrentes**.



▪ Dados Compartilhados

Acesso a **Regiões Críticas** via **Exclusão Mútua**:

```
P1.lock() || P2.lock()
```

Apenas um processo acessa o recurso por vez. O outro fica bloqueado até o lock ser liberado.

▪ Problema

A exclusão mútua pode causar **inversão de prioridade** em SDs de tempo real tarefa crítica bloqueada por uma tarefa de menor prioridade.

Escalonamento de Tarefas

Off-line, On-line, Preemptivo e Não Preemptivo

Tipos de Escalonamento

▪ Escalonamento Off-line

Todas as tarefas são conhecidas **a priori**. A validação é feita durante o projeto — o produto final é **correto por construção**.

Ex: sistemas embarcados, controladores industriais.

▪ Escalonamento On-line

Conjunto de tarefas **não é previamente conhecido**. Cada nova tarefa é verificada pelo **controle de admissão** antes de ser aceita.

Ex: servidor de vídeo *streaming*.

▪ Não Preemptivo

O processo libera a CPU somente ao terminar. Não pode ser interrompido. Simples, sem hardware extra.

Risco: monopolização da CPU.

Algoritmos: **FCFS** (FIFO) e **SJF**.

▪ Preemptivo

O escalonador pode **desalocar** um processo da CPU em qualquer instante. Maior custo, mas evita monopolização.

Algoritmos: por prioridade, Round-Robin, filas multi-nível, **Tempo Real**.

Tempo Real SEMPRE é preemptivo!

▪ Tarefas Críticas (Hard)

Terminar após o **Deadline** acarreta falhas catastróficas: danos irreversíveis, perdas de vidas ou de grandes somas.

▪ Tarefas Não Críticas (Soft)

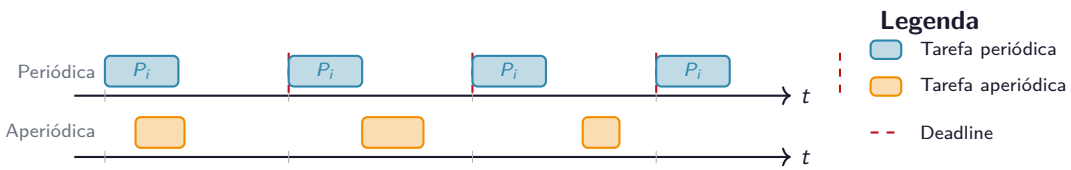
Perder o **Deadline** implica apenas em **diminuição de desempenho**. Falhas são benignas.

▪ Periodicidade

Periódicas: ativadas regularmente a cada período P_i.

Aperiódicas: ativadas por eventos aleatórios. Ex: recuperação de falhas

Esporádicas: são tarefas aperiódicas com intervalo mínimo entre ativações.



Escalonamento de Tarefas Periódicas

RM, DM e EDF

RM, DM e EDF — Visão Geral

Critério	**Rate Monotonic**	**Deadline Monotonic**	**Earliest Deadline First**
Tipo de prioridade	Fixa	Fixa	Dinâmica
Critério de prioridade	$\uparrow$ frequência ($\downarrow P_i$)	$\downarrow$ Deadline Relativo	$\downarrow$ Deadline Absoluto
Deadline relativo	$D_i = P_i$	$D_i \leq P_i$	$D_i \leq P_i$
On/Off-line	Online estático	Online estático	Online dinâmico
Ótimo na classe	Sim (prioridade fixa)	Sim (prioridade fixa)	Sim (prioridade dinâmica)

▪ Premissas comuns (RM/DM/EDF)

Tarefas periódicas e independentes; C_i conhecido e constante (*Worst Case Computation Time*); tempo de chaveamento nulo. (mundo ideal que não existe na prática)

▪ Todos são Preemptivos

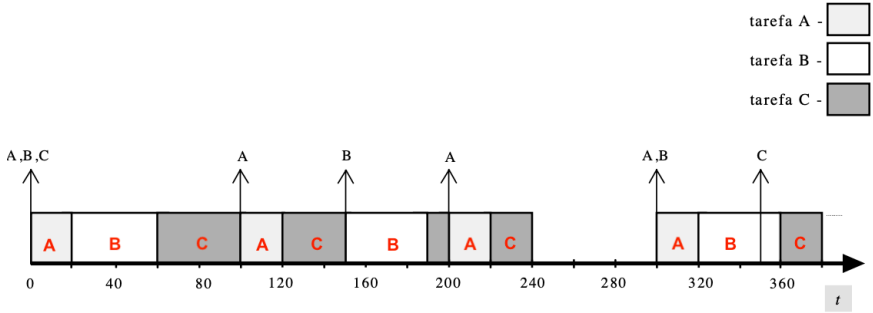
Tarefa com deadline mais urgente DEVE poder interromper tarefa com deadline menos urgente, garantindo a consistência do sistema.

Rate Monotonic — RM

Prioridade fixa baseada na **frequência**: quanto mais frequente ($\downarrow P_i$), maior a prioridade.

Restrição: $D_i = P_i$ (deadline coincide com o período).

Ótimo entre os algoritmos de prioridade fixa.



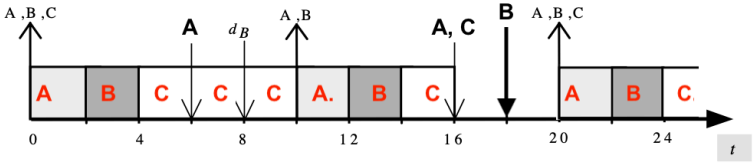
Deadline Monotonic — DM

Extensão do RM que relaxa a restrição de deadline: aceita $D_i \leq P_i$.

Prioridade fixa baseada no **Deadline Relativo inverso**: menor deadline ⇒ maior prioridade.

Mais flexível do que RM; também ótimo na classe de prioridade fixa.

tarefas periódicas	C_i	P_i	D_i	p_i
tarefa A	2	10	6	1
tarefa B	2	10	8	2
tarefa C	8	20	16	3



▪ Earliest Deadline First — EDF

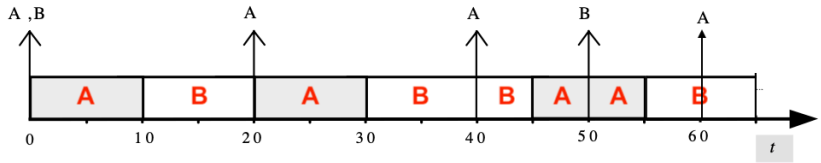
Prioridade **dinâmica**: a tarefa mais prioritária é sempre a de **Deadline Absoluto mais próximo**.

É um algoritmo **online e dinâmico**. Ótimo entre os algoritmos de prioridade dinâmica — maximiza a utilização do processador. [1]

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

tarefa A -

tarefa B -



Escalonamento EDF

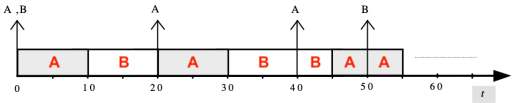
EDF vs RM — diferença-chave

RM: prioridades fixas, deterministas, simples de implementar

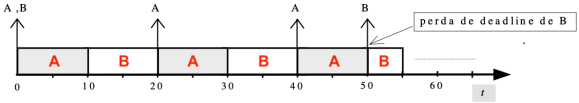
EDF: prioridades mudam a cada instante, mais complexo, mas permite utilização de até 100% do processador

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

tarefa A - A
tarefa B - B



Escalonamento EDF



Escalonamento RM

Escalonamento de Tarefas Aperiódicas

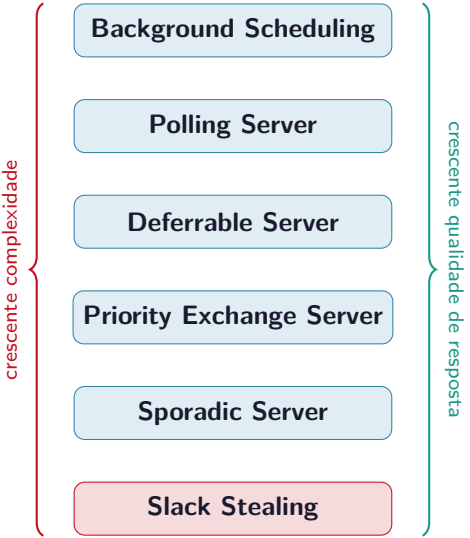
BS, PS, DS, PES, SS e Slack Stealing

O problema

Como escalonar tarefas aperiódicas e esporádicas **sem causar perda de deadlines** das tarefas periódicas?

Dois objetivos em conflito:

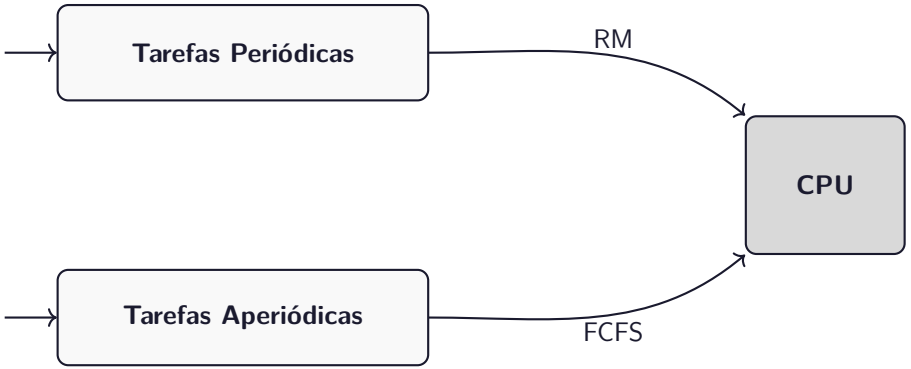
1. Aceitar/escalonar tarefas esporádicas sem prejudicar as periódicas
2. Completar tarefas aperiódicas **o mais cedo possível** sem prejudicar periódicas e esporádicas [2]



▪ Background Scheduling (BS)

Executa tarefas aperiódicas apenas quando **não há periódicas ou esporádicas** para executar.

Vantagem: simplesíssimo de implementar.
Problema: em carga alta, tempo de resposta das aperiódicas pode ser inaceitável.



▪ Polling Server (PS)

Cria uma **tarefa periódica servidora** (P_{PS}, C_{PS}) escalonada pelo RM. A cada ativação, atende requisições aperiódicas pendentes até esgotar C_{PS}.

Se não há aperiódicas na ativação: servidor é suspenso e o tempo é dado às periódicas.

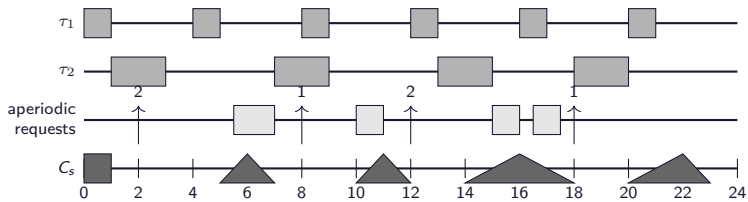
Problema: aperiódica que chega após suspensão espera o próximo período.

	C_i	T_i
τ_1	1	4
τ_2	2	6

Server

$C_s = 2$

$T_s = 5$



Deferrable Server (DS)

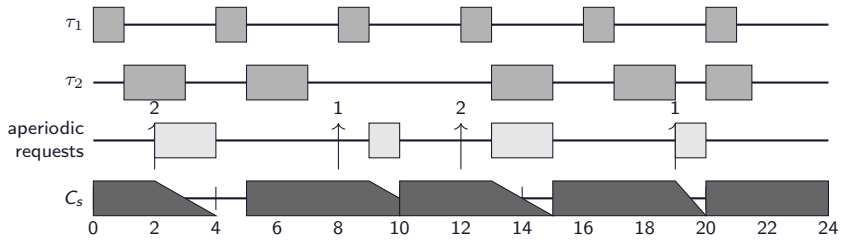
Melhoria do PS: **conserva sua capacidade** quando não há pedidos pendentes, mantendo-a até o fim do período. Se uma aperiódica chega no meio do período, o DS ainda tem capacidade disponível para atendê-la imediatamente. Melhor tempo de resposta do que PS.

	C_i	T_i
τ_1	1	4
τ_2	2	6

Server

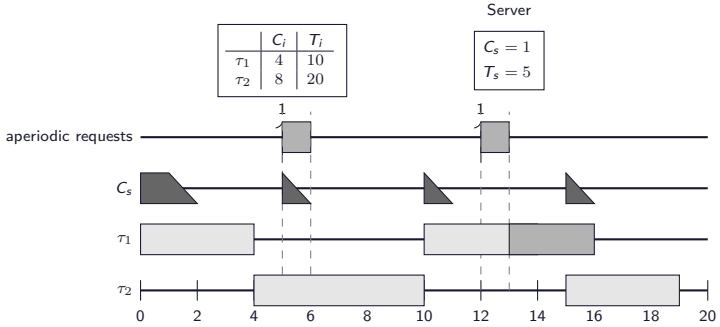
$C_s = 2$

$T_s = 5$



▪ Priority Exchange Server (PES)

Preserva capacidade trocando-a por tempo de execução de tarefas periódicas de menor prioridade. Quando sem aperiódicas: troca prioridade com a periódica ativa de maior prioridade — a periódica executa no nível do servidor, enquanto o servidor acumula capacidade no nível dela. Desempenho ligeiramente pior que DS para aperiódicas, mas melhor para periódicas.



Sporadic Server (SS)

Cria tarefa de alta prioridade que **preserva capacidade** como o DS, mas a renova de forma diferente.**DS**: renova capacidade plena a cada período. **SS**: renova capacidade apenas **após ela ter sido consumida** (após execução de aperiódica). Resultado: melhor utilização sem degradar periódicas. [2]

Slack Stealing

Não cria tarefa servidora. Cria uma **tarefa passiva (Slack Stealer)** que se apropria de todo tempo livre das periódicas sem causar perda de deadlines. Fórmula de apropriação:

$$SLACK_i(t) = D_i - t - C_i(t)$$

Princípio: *nenhum benefício existe para periódicas terminarem antes do prazo*. Melhor tempo de resposta entre todos os métodos.

Comparação dos Algoritmos para Tarefas Aperiódicas

Algoritmo	Servidor?	Preserva cap.?	Renovação	Complexidade
BS	Não	—	—	Mínima
PS	Sim	Não	A cada período	Baixa
DS	Sim	Sim (até fim do período)	A cada período	Média
PES	Sim	Troca prioridade	A cada período	Alta
SS	Sim	Sim	Após consumo	Alta
Slack	Passiva	Todo slack	Dinâmica	Máxima

Referências I

- TANENBAUM, A. S.; VAN STEEN, M. *Distributed Systems: Principles and Paradigms*. 2. ed. Pearson, 2007.
- BURNS, A.; WELLINGS, A. *Real-Time Systems and Programming Languages*. 3. ed. Addison-Wesley, 1997.
- COULOURIS, G. et al. *Distributed Systems: Concepts and Design*. 5. ed. Addison-Wesley, 2013.