Sistemas Distribuídos

Aula 5 – Sistemas de Tempo Real: Conceitos, Tipos e Projeto

Prof. Ana Carolina Sokolonski

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

2026

Agenda

Conceitos Básicos

O que é Tempo Real?

Tipos de Tempo

▪ Tempo de Programação

Recurso gasto na **criação** (implementação) de um programa.

▪ Tempo de Execução

Recurso gasto na **execução** de um programa ou de um trecho deste.

▪ Tempo Lógico

Definido a base na **relação de precedência** entre eventos. Permite estabelecer ordens causais entre um conjunto de eventos.

▪ Tempo Físico

Tempo métrico — expressa **quantitativamente** o intervalo entre eventos.

▪ Tempo Denso

Medido de forma uniforme e contínua, pertencente ao conjunto dos **reais**.

▪ Tempo Discreto

Simplificação do tempo denso para o conjunto dos **naturais positivos**.

▪ Tempo Relativo

Tempo contado a partir de uma **referência local** (cada nó possui a sua própria).

▪ Tempo Absoluto

Tempo válido para **todo o sistema** como referência global única.

O que é Tempo Real?

▪ Tempo Real $\neq$ Rapidez!

Um Sistema de Tempo Real é especificado com base em **restrições temporais**. O que importa não é a **rapidez** e, sim, o **cumprimento** das restrições lógicas e temporais.

▪ Definição

Sistema computacional que interage com o ambiente via **sensores** (captam informações) e **atuadores** (interferem no ambiente). Deve produzir resultados válidos (*correctness* — correção lógica) em prazos determinados pelo ambiente (*timeliness* — correção temporal). [2]

▪ Exemplos comparativos

Caixa eletrônico — STR?
NÃO. Deseja-se rapidez para o usuário, mas não há restrição temporal rígida no terminal em si. O sistema bancário por trás é STR.

Piloto automático — STR?
SIM. Deve funcionar perfeitamente, respeitando as restrições lógicas e temporais, sem erros.

▪ STR vs. Sistema Convencional

Convencional: faz o que deve usando o tempo necessário.
Tempo Real: faz o que deve usando o **tempo disponível**. [1]

Aplicações de Sistemas de Tempo Real

▪ Militar

Sistemas de mísseis teleguiados, radar, sonar. Falha temporal pode ser fatal.

▪ Fábricas

Controle de processos industriais, robôs em linhas de montagem, CLP.

▪ Transportes

Sistema de freio eletrônico (ABS), aeronaves, controle ferroviário, metrô, carros autônomos.

▪ Saúde

Monitores cardíacos, bombas de insulina, sistemas de UTI, robôs cirúrgicos.

▪ Telecom e Lazer

Celular, videoconferência, jogos, vídeo sob demanda.

▪ Uso Doméstico

Eletrodomésticos inteligentes: geladeira, micro-ondas, máquina de lavar, termostato.

Componentes básicos: Sensores → Processador STR → Atuadores → Ambiente

Tipos de Sistemas de Tempo Real

Críticos, Firmes e Fracamente Críticos

Classificação dos Sistemas de Tempo Real

▪ Hard Real Time (Críticos)

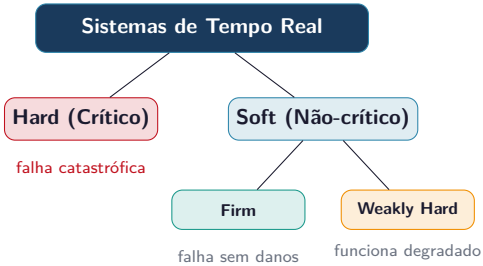
Falhas temporais levam a situações **catastróficas**: danos irreversíveis, perda de vidas ou grandes prejuízos. O deadline deve ser respeitado **sempre**.

Ex: piloto automático, sistemas de controle nuclear, freio ABS, marca-passo, sondas espaciais.

▪ Soft Real Time (Não Críticos)

Falhas temporais podem ser toleradas em certas situações — o processamento não é completamente perdido.

Ex: sistemas de exibição de vídeo, jogos online, apps de streaming.



Tarefas em Sistemas de Tempo Real

Deadlines, Release Time e Previsibilidade

Tarefas, Deadlines e Instante de Ativação

▪ Tarefa

Conjunto de ações com **restrições temporais próprias**.

Exemplo — Tarefa *Aterrissagem* (piloto automático):

1. Diminuir altitude e velocidade
2. Abaixar trem de pouso
3. Iniciar aproximação da pista
4. Tocar no solo (início da pista)
5. Acionar freios mecânicos
6. Parar totalmente

Deadline: máximo 1 minuto.

▪ Tipos de Deadline

Deadline Relativo: intervalo de tempo em que a tarefa deve executar e concluir.
Ex: $D_{rel} = 1$ min.

Deadline Absoluto: instante máximo para conclusão.
Ex: se iniciou em $t = 10$s, $D_{abs} = 70$s.

▪ Instante de Ativação (Release Time)

Instante de início da execução. Pode ter **delay** (dependência de outra tarefa) e **jitter** (variação máxima conhecida do atraso).

Tipos de Tarefas e Previsibilidade

▪ Tarefas Periódicas

Instante de ativação **conhecido**. Se repetem em intervalos regulares, com o período P_i. A cada início de período a tarefa é ativada automaticamente.

▪ Tarefas Esporádicas

Instante de ativação **desconhecido**, mas o **intervalo mínimo** entre duas ativações é conhecido. No pior dos casos, executa como um período.

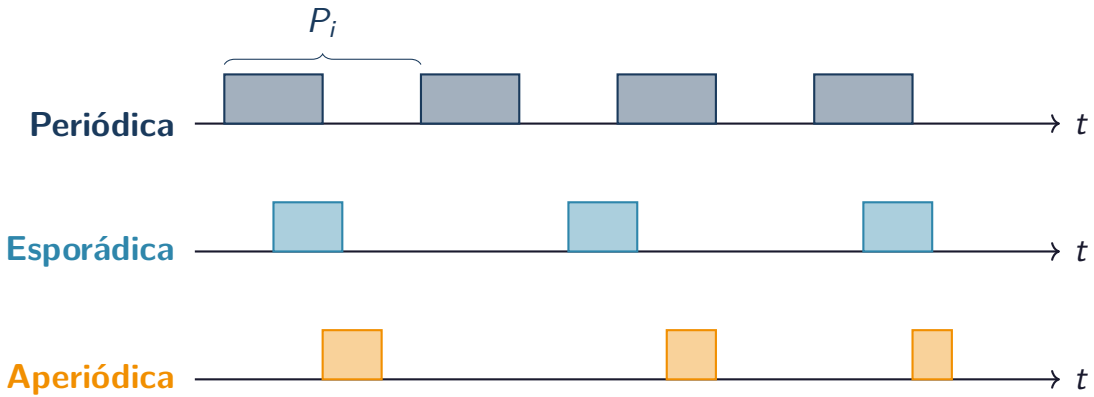
▪ Tarefas Aperiódicas

Instante de ativação completamente **desconhecido**. Nada se sabe antes da ativação — chegam de forma imprevisível.

▪ Sistema Previsível

Um STR é **previsível** quando, independentemente de variações de hardware, de carga ou de falhas, o comportamento pode ser **antecipado em tempo de projeto**. Todos os deadlines poderão ser atendidos.

Tipos de Tarefas e Previsibilidade



Projeto de Sistemas de Tempo Real

Hardware, Software e WCET

▪ Aspectos críticos do projeto

- **Seleção de HW/SW**: custo × benefício, paralelismo, sincronização, distribuição
- **Especificação**: representação correta do comportamento funcional e temporal [1]
- **Linguagem de programação**: abstrações temporais, representação em código de máquina
- **Tolerância a falhas**: prevenção e recuperação

▪ WCET — Worst Case Execution Time

O **maior problema do projeto de hardware** para STR.

Sem o conhecimento exato do WCET, não é possível garantir que o sistema executará corretamente — uma tarefa pode consumir mais tempo do que declarado, causando perda de deadlines de outras tarefas.

Hardwares tradicionais são **imprevisíveis**: interrupções, cache, pipeline, branch prediction — todos introduzem variabilidade no tempo de execução. [1]

Hardware para Sistemas de Tempo Real

CISC, RISC e Previsibilidade

Hardware para STR — CISC vs. RISC

▪ CISC — Complex Instruction Set

Instruções **complexas** que exigem vários ciclos de relógio. Suporta formatos variados de instrução.

Vantagem: programas menores (uma instrução faz mais).
Desvantagem: tempo de execução variável — WCET difícil de determinar.

Ex: Intel x86, AMD64.

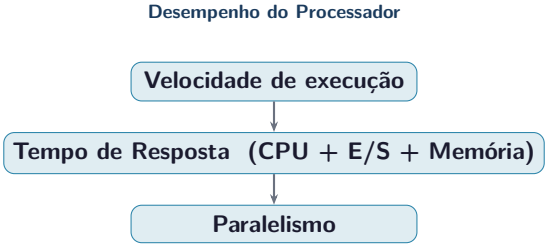
▪ RISC — Reduced Instruction Set

Instruções **simples** executadas em **um ciclo de relógio**. Formatos fixos.

Vantagem: WCET previsível, pipeline eficiente.
Desvantagem: programas maiores.

Ex: ARM (embarcados STR), SPARC, MIPS, RISC-V.

Hardware para STR — CISC vs. RISC



Critério	CISC	RISC
Complexidade instrução	Alta	Baixa
Ciclos por instrução	Variável	1
Previsibilidade WCET	Baixa	Alta
Uso em STR crítico	Raro	Comum

Conclusão e Resumo

O que aprendemos

- STR $\neq$ rapidez — o que importa é **cumprir deadlines**
- Hard (catastrófico) vs. Soft (tolerável) vs. Firm/Weakly Hard
- **Tarefas:** periódicas, esporádicas e aperiódicas
- **Previsibilidade:** base do projeto de STR
- **WCET:** maior desafio do hardware para STR
- **RISC** prefere STR crítico; **CISC** mais flexível

Custo $\times$ Benefício

O aumento da imprevisibilidade implica **redução de custo**. Não se deve superdimensionar sistemas para STR Não Críticos.

STR Crítico $\rightarrow$ RISC, hardware dedicado, alta confiabilidade
STR Soft $\rightarrow$ hardware convencional com alguma imprevisibilidade aceitável

Conexão com SDs

STRs distribuídos herdam todos os problemas de Sistemas Distribuídos: sincronização de relógios, falhas parciais, latência de comunicação e inconsistência de dados.

Referências I

- 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.
- TANENBAUM, A. S.; VAN STEEN, M. *Distributed Systems: Principles and Paradigms*. 3. ed. Pearson, 2017.