Escola de Computação'2000 - IME-USP

Sistemas de Tempo Real: Conceitos Básicos

Jean-Marie Farines
Joni da Silva Fraga
Rômulo Silva de Oliveira

LCMI - Laboratório de Controle e Microinformática
DAS - Departamento de Automação e Sistemas
UFSC - Universidade Federal de Santa Catarina

Caracterização

- Sistemas de Tempo Real
  - respostas devem ser válidas ("correctness") e dentro de prazos ("Timeliness") impostos pelo ambiente.
- Sistemas Não de Tempo Real
  - respostas válidas em prazos aceitáveis, não especificados.
- Uma Definição [Audsley90]:
  - Um Sistema de Tempo Real é um sistema que produz reações a estímulos oriundos do ambiente dentro de intervalos de tempos impostos pelo ambiente (é incluindo entre estes estímulos o passar do tempo físico).

Caracterização

- Forte acoplamento de um STR com o seu Ambiente:
  - Processamento ativado por estímulos do Ambiente
  - Tempos de Resposta delimitam Estímulos/Respostas
    - Processamentos devem terminar dentro de prazos (deadlines)
    - Se terminar fora de prazo sistema falha (falha temporal)
- Dados com prazos de validade:
  - Dados desatualizados ou não válidos (“outdated data”) podem conduzir a resultados ou respostas incorretas
- Fluxos de controle na execução dos processamentos é definida pelo ambiente:
  - Impossibilidade em muitas aplicações do STR exercer um controle ou limitação nos estímulos provenientes do Ambiente.

Conceitos Básicos

- **Tarefa** (task)
  - Segmento de código cuja execução possui atributo temporal próprio
  - Exemplo: método em OO, subrotina, trecho de um programa
- **Deadline**
  - Instante máximo desejado para a conclusão de uma tarefa
- Tempo real crítico (**hard real-time**)
  - Falha temporal pode resultar em consequências catastróficas
  - Necessário garantir requisitos temporais em projeto
  - Exemplo: usina nuclear, industria petroquímica, mísseis
- Tempo real não crítico (**soft real-time**)
  - Requisito temporal descreve apenas comportamento desejado
  - Exemplo: multimídia

Conceitos Básicos

- **Modelo de Tarefas** (task system)
  - Descrição das propriedades temporais das tarefas no sistema
  - Exemplo: periódicas ou não, com duração conhecida ou não, etc
- **Carga de Tarefas** (task load)
  - Descrição de quais tarefas deverão ser executadas
  - Estática: Limitada e conhecida em projeto
  - Dinâmica: Conhecida somente ao longo da execução
- **Previsibilidade** (predictability)
  - Capacidade de afirmar algo sobre o comportamento futuro do sistema
  - Determinista: Garante que todos os requisitos temporais serão cumpridos
  - Probabilista: Fornece uma probabilidade para o seu cumprimento

Previsibilidade

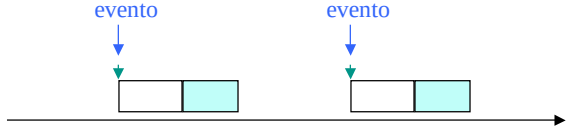
- Previsibilidade (“predictability”)
  - Está associada a capacidade de poder antecipar, em tempo de projeto, se os processamentos em um STR serão executados dentro de seus prazos especificados
- Associada a uma previsão determinista
  - todos os deadlines serão respeitados
- ou a uma antecipação probabilista
  - baseadas em estimativas, probabilidades são associadas a deadlines definindo as possibilidades dos mesmos serem respeitados
- A previsibilidade em STRs determina implicações em todos os níveis:
  - linguagens
  - sistemas operacionais
  - comunicação
  - arquitetura do computador
  - etc

Conceitos Básicos

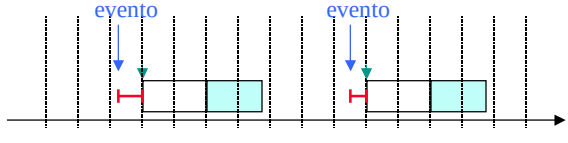
- **Escalonamento** (scheduling)
  - Identifica a forma como recursos são alocados às tarefas
- **Escalonador** (scheduler)
  - Componente do sistema responsável pela gerência dos recursos
- **Escala de Execução** (schedule)
  - Descreve quando cada tarefa ocupa cada recurso
- **Escalonamento Estático** (static scheduling)
  - Capaz de oferecer previsibilidade determinista
- **Escalonamento Dinâmico** (dynamic scheduling)
  - Capaz de oferecer apenas previsibilidade probabilista

Conceitos Básicos

- Event-Triggered



- Time-Triggered



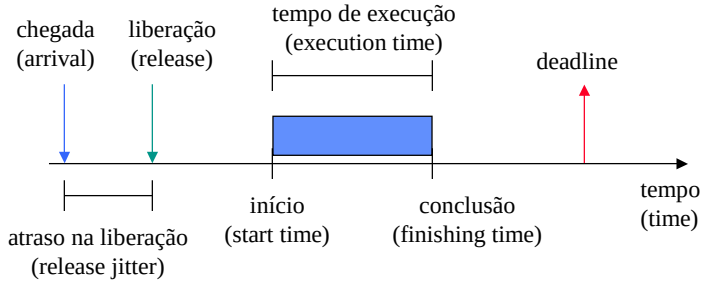
Event-Triggered x Time-Triggered

- Event-Triggered
  - Evento externo gera interrupção e dispara tarefa
  - Funciona bem com carga pequena
  - Pode falhar no caso de carga pesada
  - Ex: Cano estourado gera chuva de eventos
- Time-Triggered
  - Interrupção de relógio a cada T milisegundos (tick)
  - A cada tick alguns sensores e atuadores são acessados
  - Não existem interrupções além das do relógio
  - Problema é selecionar T (carga x atraso)
  - Ex: Relação causa-efeito em trem e cancela

Modelos de Tarefas

- Descrevem como são as tarefas do sistema
  - Quais as suas propriedades temporais
- Ponto de partida para:
  - Análise de escalonabilidade
  - Escolha do suporte
    - Linguagem de programação
    - Sistema operacional
    - Arquitetura do computador
- Varia muito de sistema para sistema

Propriedades Temporais das Tarefas



Folga = Deadline - Liberação - Tempo de execução

Atraso = MAX(0 , Conclusão - Deadline)

Tempo de resposta = Conclusão - Chegada

Tarefas Periódicas

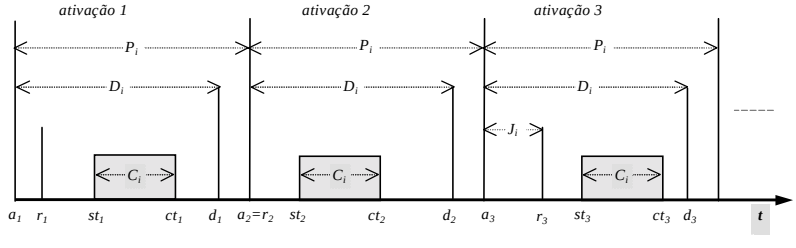


Figura 2.1: Ativações de uma tarefa periódica

Tarefas Esporádicas

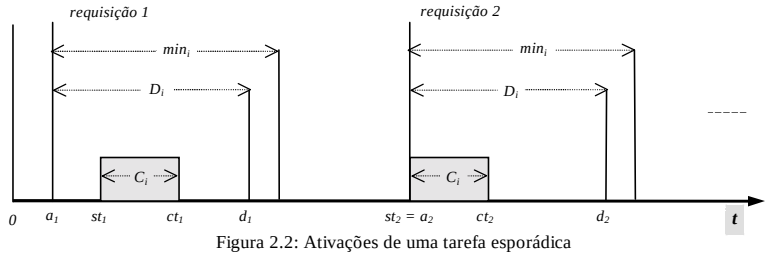


Figura 2.2: Ativações de uma tarefa esporádica

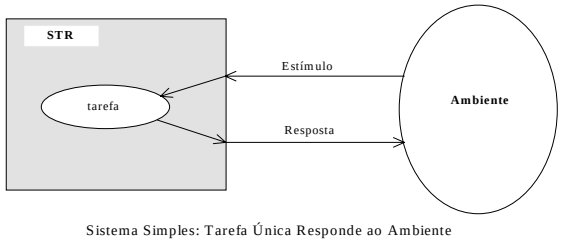
Propriedades Temporais das Tarefas

- **Deadline Hard**
  - Perda do deadline pode ter consequências catastróficas
  - Exemplo: abrir válvula em duto de alta pressão
- **Deadline Firm**
  - Perda do deadline NÃO tem consequências catastróficas
  - Não existe valor em terminar a tarefa após o deadline
  - Exemplo: amostrar periodicamente valor físico
- **Deadline Soft**
  - Perda do deadline NÃO tem consequências catastróficas
  - Existe valor em terminar a tarefa com atraso
  - Exemplo: movimento de objeto em vídeo game

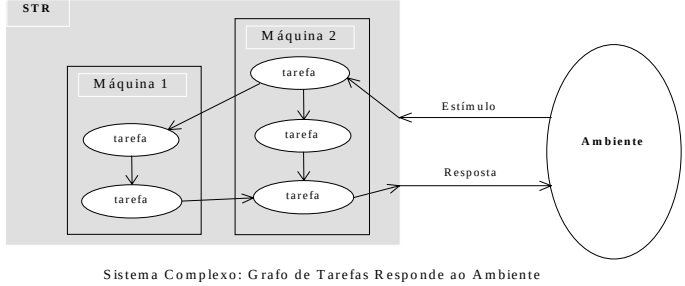
Propriedades Temporais das Tarefas

- **Tarefa Periódica**
  - Tarefa é ativada a cada P unidades de tempo
  - Instantes de chegada podem ser calculados a partir do inicial
  - Exemplo: controle de processo via laço de realimentação
- **Tarefas Esporádica**
  - Instantes de chegada não são conhecidos
  - Existe um intervalo mínimo de tempo entre chegadas
  - Exemplo: atendimento a botão de alarme
- **Tarefa Aperiódica**
  - Nada é sabido quanto as ativações da tarefa
  - Exemplo: aparecimento de objeto em tela de radar

Complexidade da Aplicação



Sistema Simples: Tarefa Única Responde ao Ambiente



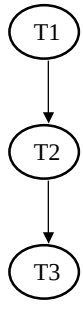
Sistema Complexo: Grafo de Tarefas Responde ao Ambiente

Propriedades Temporais das Tarefas

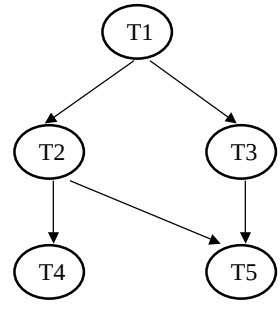
- Relações de **precedência** entre tarefas
  - Tarefa A é predecessora da tarefa B quando B somente pode iniciar depois que A estiver concluída
  - Neste caso, tarefa B é sucessora da tarefa A
  - Representado por A –> B
  - Exemplo: envio de mensagem de A para B
- **Atividade**
  - Conjunto de tarefas interligadas por relações de precedência
  - Representada por um grafo onde
    - Nodos são as tarefas
    - Flexas são as relações de precedência

Propriedades Temporais das Tarefas

Atividade com Relações de Precedência LINEARES



Atividade com Relações de Precedência ARBITRÁRIAS



Propriedades Temporais das Tarefas

- Relações de **exclusão mútua** entre tarefas
  - Tarefas A e B apresentam exclusão mútua quando NÃO podem executar simultaneamente
- Exemplos:
  - Estrutura de dados compartilhada
  - Arquivo
  - Controlador de periférico

Diferentes Abordagens

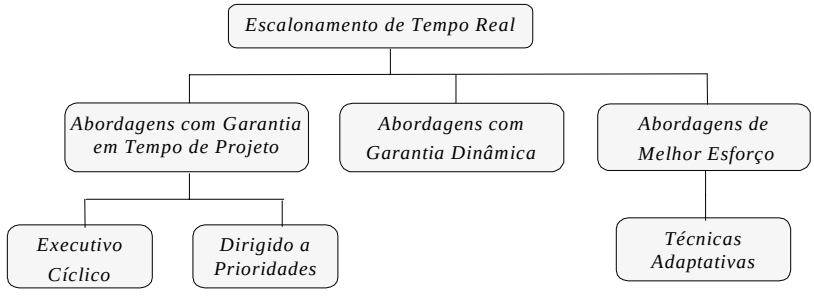


Figura 2.3: Abordagens de Escalonamento de Tempo Real

Abordagens com Garantia em Projeto

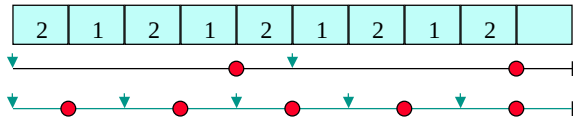
- Oferece previsibilidade determinista, com análise feita em projeto
  - Carga limitada e conhecida em projeto (Hipótese de Carga)
  - Suposto um limite para faltas (Hipótese de Faltas)
- Dividida em duas partes:
  - Análise da escalonabilidade
  - Construção da escala de execução
- Vantagens
  - Determina em projeto que todos os deadlines serão cumpridos
  - Necessário para aplicações críticas
  - Teoria serve de base para abordagens sem garantia
- 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

Executivo Cíclico

- Todo o trabalho de escalonamento é feito em projeto
- Resultado é uma **grade de execução** (time grid)
- Grade determina qual tarefa executa quando
- Garantia obtida através de uma simples inspeção da escala
- Durante a execução:
  - Pequeno programa lê a grade e dispara a tarefa aproprida
  - Quando a grade termina ela é novamente repetida
- Vantagem: Comportamento completamente conhecido
- Desvantagem: Escalonamento muito rígido, tamanho da grade
- Muito usado em aplicações embutidas (Embedded Systems)

Executivo Cíclico

- Restrições devem ser observadas na construção da grade
  - Período, tempo máximo de computação
  - Precedências, exclusões, etc
- Escalonamento pode ser:
  - Preemptivo - Tarefa pode ser suspensa e depois retomada
  - Não Preemptivo - Depois que inicia tarefa vai até o fim
- Exemplo:
  - T1: P1=5 C1=2 D1=4
  - T2: P2=2 C2=1 D2=1

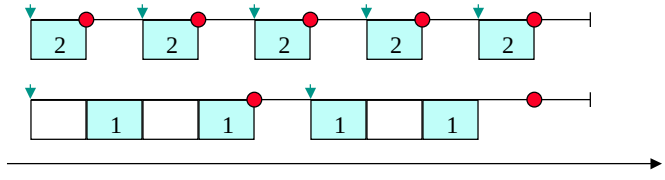


Prioridades + Teste de Escalonabilidade

- Cada tarefa recebe uma prioridade
- Prioridades podem ser fixas ou variáveis
- Escalonamento em geral é preemptivo
- Teste executado em projeto determina escalonabilidade
  - Teste considera forma como prioridades são atribuidas
  - Complexidade depende do modelo de tarefas
- Na execução:
  - Escalonador dispara as tarefas conforme as prioridades
- Vantagem: Suporta tarefas esporádicas, não tem grade
- Desvantagem: Testes limitados a alguns modelos de tarefas
- Usado em aplicações que exigem garantia mas são muito complexas para construir uma grade

Prioridades + Teste de Escalonabilidade

- Políticas mais utilizadas:
  - Earliest Deadline First - EDF
  - Rate Monotonic - RM
  - Deadline Monotonic - DM
- Exemplo usando RM:
  - T1: P1=5 C1=2 D1=4 Prioridade menor
  - T2: P2=2 C2=1 D2=1 Prioridade maior



Abordagens com Melhor Esforço ou Garantia Dinâmica

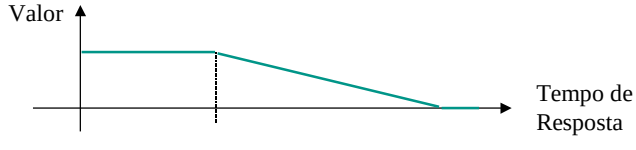
- Não existe garantia que os deadlines serão cumpridos
- O “Melhor Esforço” será feito neste sentido
- Capaz de fornecer análise probabilista
  - Simulação, teoria das filas de tempo real, etc
- Algumas abordagens oferecem Garantia Dinâmica
  - Garante o deadline (ou não) no início da ativação
- Existe a possibilidade de Sobrecarga (overload)
  - Não é possível cumprir todos os deadlines
  - Situação natural uma vez que não existe garantia off-line
- Vantagens
  - Não é necessário conhecer o pior caso
  - Sistemas mais baratos, projetados para o caso médio
  - Não é necessário conhecer a carga exatamente
- Desvantagens
  - A princípio qualquer deadline poderá ser perdido

Descarte de Tarefas na Sobrecarga

- Em sobrecarga NÃO EXECUTA algumas tarefas
- As tarefas executadas cumprem o deadline
  - Mais apropriado para tarefas com deadline firm
- Pode cancelar:
  - Ativações individuais
  - Tarefas completas
- Objetivo é maximizar o número de tarefas executadas
- Tarefas podem ter importância (peso) diferentes
  - Maximiza o somatório dos pesos das tarefas executadas
- Algumas soluções utilizam um parâmetro de descarte **s**
  - Distância temporal entre ativações descartadas deve ser **s**
- Abordagem semelhante: aumenta o período das tarefas

Perda de Deadlines na Sobrecarga

- Em sobrecarga ATRASA algumas tarefas
- Baseado em função tempo-valor (time-value function)
- Cada tarefa possui uma função que
  - Indica o valor da tarefa para o sistema em função do seu instante de conclusão
- Objetivo é maximizar o valor total do sistema
  - Valor do sistema é o somatório do valor das tarefas executadas
  - Tarefas podem possuir importância (peso) diferentes



Redução da Precisão na Sobrecarga

- Em sobrecarga DIMINUI a precisão de algumas tarefas
- Objetivo é “fazer o trabalho possível dentro do tempo disponível”
- Exemplos:
  - Ignorar bits menos significativos de cada pixel
  - Trabalhar com amostras de áudio menos precisas
  - Alterar resolução e tamanho de imagem na tela
  - Simplificar animações
  - Usar algoritmos de controle mais simples
  - Interromper pesquisa em algoritmos de inteligência artificial
- Aparece associada com a técnica de
  Computação Imprecisa (Imprecise Computation)