Sistemas de Tempo-Real

Aula 9

Outros aspectos do escalonamento de tempo-real

Escalonamento sem preempção
Questões de aplicação prática em sistemas reais

Adaptado dos slides desenvolvidos pelo Prof. Doutor Luís Almeida
para a disciplina “Sistemas de Tempo-Real”
Revisto em Nov/2009 por Paulo Pedreiras

Aula anterior (8)

- Execução conjunta de tarefas **periódicas e aperiódicas**
  - Execução de aperiódicas em *background*
- Utilização de **servidores de tarefas aperiódicas**
- Servidores de **prioridades fixas**
  - Polling Server - PS
  - Deferrable Server - DS
  - Sporadic Server - SS
- Servidores de **prioridades dinâmicas**
  - Total Bandwidth Server – TBS
  - Constant Bandwidth Server - CBS

Escalonamento sem preempção

O escalonamento **sem preempção** consiste em executar as tarefas completamente, **sem permitir a sua suspensão** para execução de outras tarefas de maior prioridade

Características principais (vantagens):

- É **muito simples de realizar** já que não é necessário salvaguardar o estado intermédio de execução das tarefas (estas executam sem interrupção).
- O tamanho total de ***stack* necessário para o sistema é mínimo** (igual aos requisitos de *stack* da tarefa com maiores requisitos)
- Não é necessário **nenhum protocolo** específico para controlo do **acesso a recursos partilhados** (as tarefas executam com exclusão mútua)

Escalonamento sem preempção

Características principais (desvantagens):

- Implica uma **penalização ao nível da escalonabilidade** do sistema, principalmente quando há tarefas com tempos de execução longos.
- Essa penalização torna-se excessiva quando simultaneamente se pretende executar tarefas com elevado ritmo de activação.

A penalização da escalonabilidade do sistema pode ser vista como um **bloqueio no acesso a um recurso partilhado, o CPU**. Isto permite utilizar os testes de escalonabilidade apresentados anteriormente para sistemas com preempção e acesso a recursos partilhados.

Neste caso,

$$B_i = \max_{k \in lp(i)}(C_k)$$

Escalonamento sem preempção

Para além de se considerar o termo de bloqueio correspondente, existem ainda algumas **adaptações** a efectuar nas análises de escalonabilidade que usam o cálculo do tempo de resposta.

Calculo de Rwc_i com **prioridades fixas**:

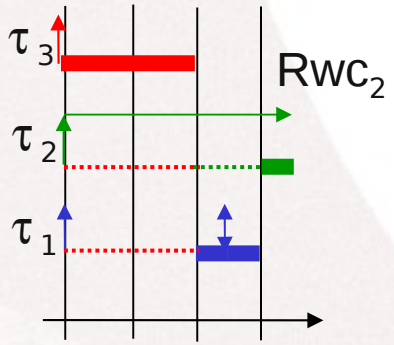
$$\forall i, R_{wc_i} = I_i + C_i$$

O processo iterativo é executado **apenas sobre I_i** uma vez que, após iniciar, a tarefa i executará até final, e deverá incluir activações no **último instante**

$$I_i = B_i + \sum_{k \in hp_i} \left(\left\lfloor \frac{I_i}{T_k} \right\rfloor + 1 \right) * C_k$$

$$I_i(0) = B_i + \sum_{k \in hp_i} C_k$$

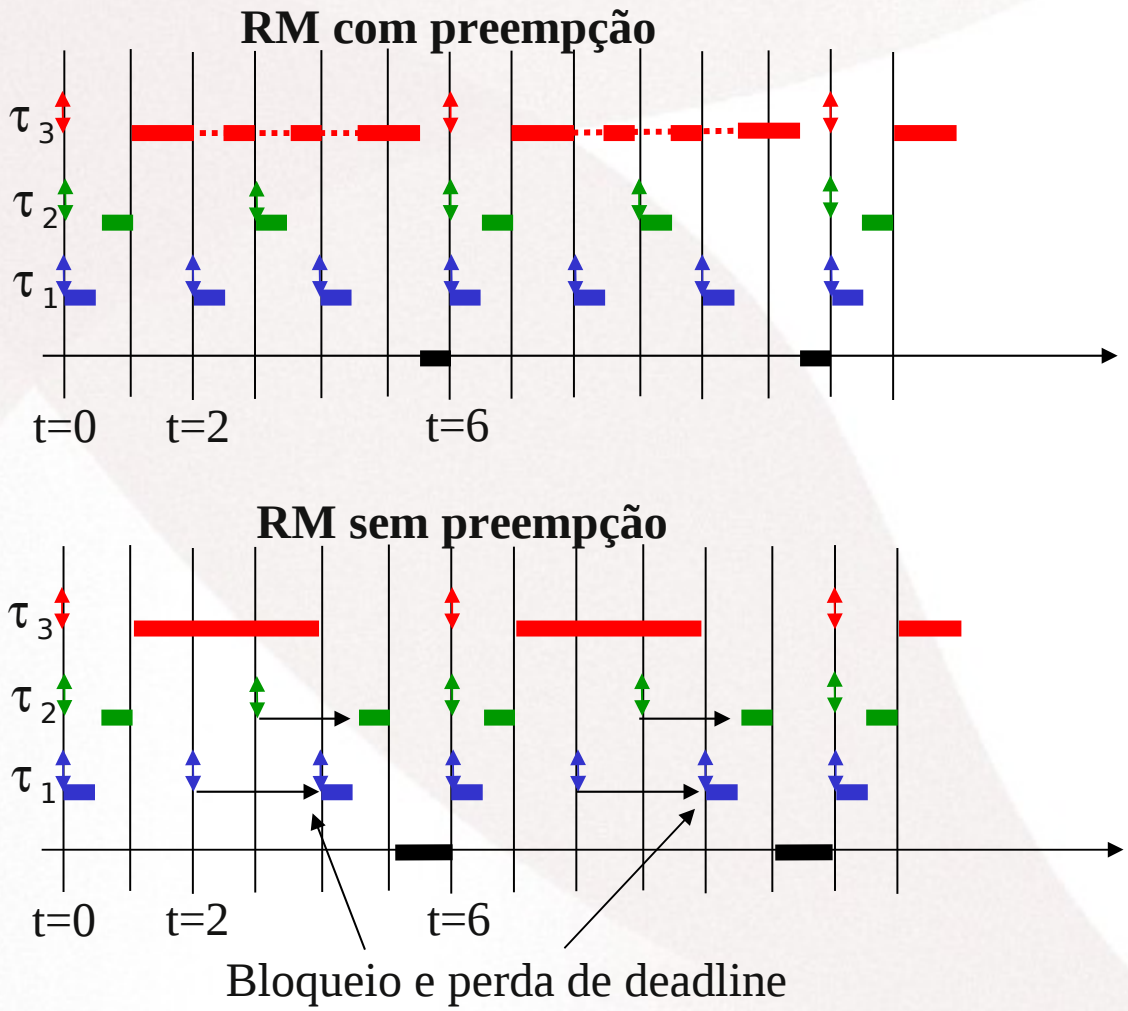
$$I_i(m+1) = B_i + \sum_{k \in hp_i} \left(\left\lfloor I_i \frac{(m)}{T_k} \right\rfloor + 1 \right) * C_k$$



Escalonamento sem preempção

Tabela de propriedades das tarefas

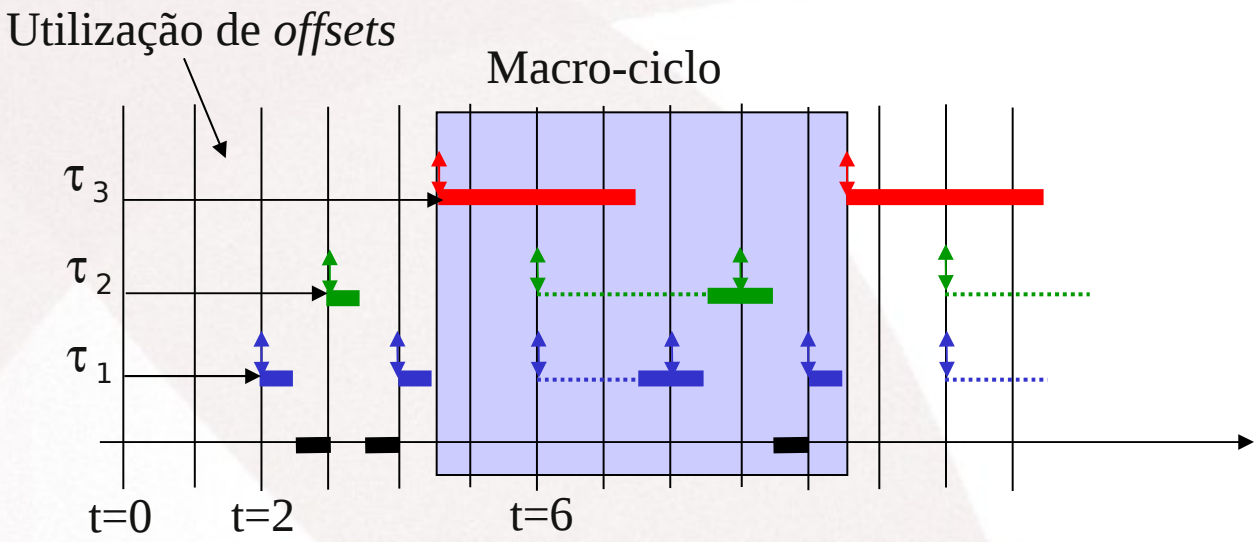
τ_i	T_i	C_i
1	2	0.5
2	3	0.5
3	6	3



Escalonamento sem preempção

Tabela de propriedades das tarefas

τ_i	T_i	C_i	O_i
1	2	0.5	**2**
2	3	0.5	**3**
3	6	3	**4.5**



A **utilização de *offsets*** pode ser particularmente eficaz no escalonamento sem preempção, permitindo por vezes **tornar escalonável** um sistema não-escalonável.

Aspectos de implementação prática

Na construção de **aplicações reais** existem alguns aspectos a ter em conta por influenciarem a exactidão dos testes de escalonabilidade das tarefas:

- O custo dos **mecanismos internos** do executivo (e.g. *tick handler*)
- O custo adicional das **mudanças de contexto**
- Os **tempos de execução** das tarefas
- Outras rotinas de atendimento de **interrupções**
- **Desvios nos instantes de activação** das tarefas

Aspectos de implementação prática

Determinação do custo adicional do atendimento do *tick*

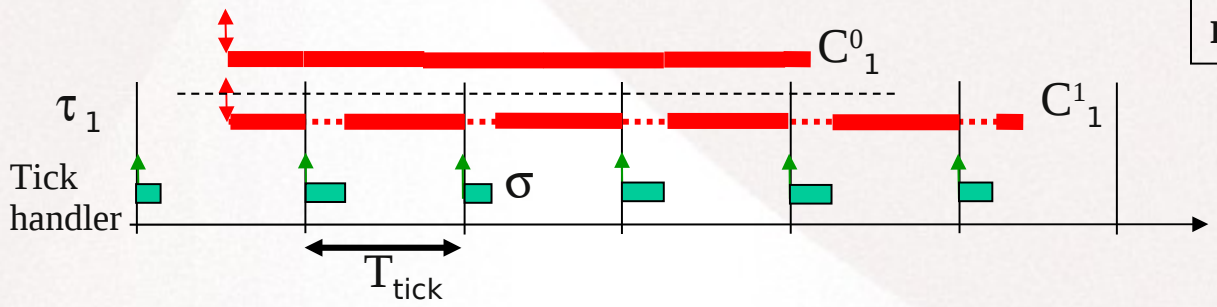
- O atendimento do **relógio do sistema** (*tick*) utiliza tempo de CPU para além da execução das tarefas (*overhead*).
- Trata-se da actividade de **maior prioridade** executada pelo sistema e pode ser modelada como uma **tarefa periódica**
- O respectico ***overhead*** (σ) tem um impacto substancial na eficiência do sistema já que se trata de uma parte da largura de banda que é subtraída à disponibilidade do CPU para execução das tarefas da aplicação.

Aspectos de implementação prática

Determinação do custo adicional do atendimento do *tick* *(cont.)*

- Pode ser medida quer directamente quer recorrendo a uma função longa, executada sem e com interrupções do *tick* (período T_{tick}) e medindo a diferença dos tempos de execução (C^0_1 e C^1_1 respectivamente).
  Neste caso,

$$\sigma = \frac{C_1^1 - C_1^0}{\left\lceil \frac{C_1^1}{T_{Tick}} \right\rceil}$$

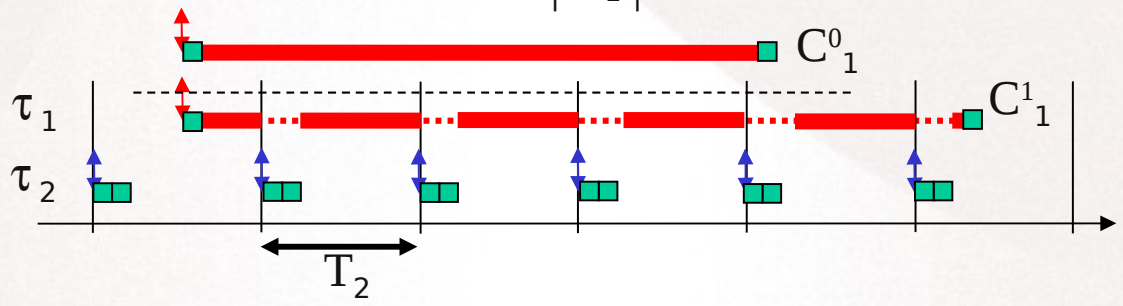


Esta técnica fornece um valor médio de σ !

Aspectos de implementação prática

Determinação do custo adicional das mudanças de contexto

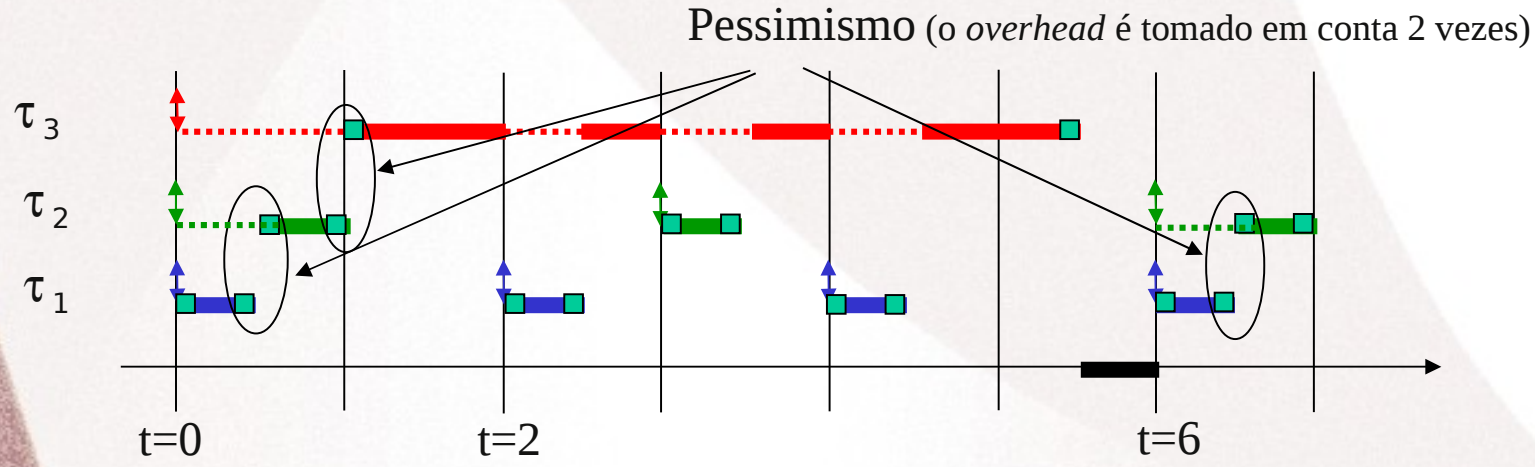
- As **mudanças de contexto** também requerem tempo de CPU para além da execução do código das tarefas (*overhead*).
- Uma forma simples de medir este *overhead* (δ) consiste em utilizar um par de tarefas, uma longa e a outra de maior prioridade, rápida (T_2) e vazia (sem código). Basta medir o tempo de execução da primeira, sozinha (C^0_1) e juntamente com a segunda (C^1_1).
- Neste caso, $\delta = \frac{C_1^1 - C_1^0}{\left\lceil \frac{C_1^1}{T_2} \right\rceil}$



Aspectos de implementação prática

Utilização do custo adicional das mudanças de contexto (cont.)

- Uma forma **simples** (mas pessimista) de ter em conta o *overhead* das mudanças de contexto (δ) consiste em **adicionar** esse termo aos **tempos de execução** das tarefas. Dessa forma, não apenas se toma em linha de conta a mudança de contexto relativa à própria tarefa como também as relativas a todas a preempções que possam ocorrer.



Aspectos de implementação prática

Determinação dos tempos de execução das tarefas

- Normalmente efectuada com recurso à **análise do código fonte**, para determinar o **trajecto de execução mais longo**, de acordo com os dados de entrada.
- Depois é analisado o **código objecto** para contar os ciclos de relógio do CPU necessários para executar as instruções do referido trajecto mais longo.

- Notar que o **tempo de execução** de uma tarefa **poderá variar** de instância para instância de acordo com os dados de entrada ou com condições do sistema, por via da utilização de **condicionais** e **ciclos**.

Aspectos de implementação prática

Determinação dos tempos de execução das tarefas (cont.)

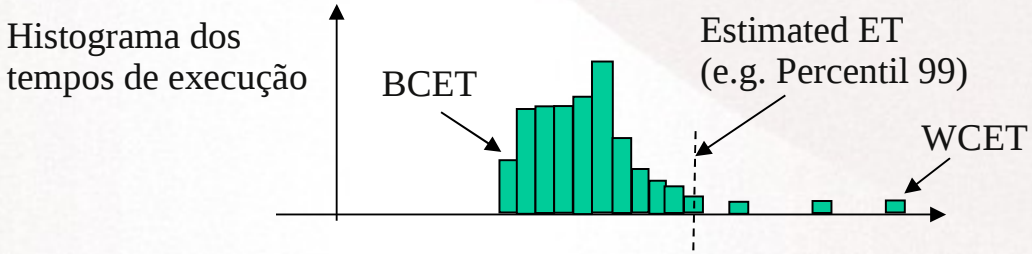
- Também é possível **executar a tarefa isoladamente** e de forma controlada, fornecendo-lhe os **dados de entrada adequados** e medindo o tempo de execução na plataforma destino.
  - Este método experimental requer algum cuidado para garantir que se obtém um majorante do tempo de execução!

- Os **processadores complexos** actuais usam *pipelines* e *caches* (de dados e/ou de instruções) que melhoram substancialmente o tempo médio de execução de um programa mas apresentam um pior caso muito penalizante.
- Para este casos, usam-se análises específicas para reduzir o pessimismo do pior caso (máximas ocorrências de *cache misses* e *pipeline flushes* de acordo com a efectiva sequência de instruções).

Aspectos de implementação prática

Determinação dos tempos de execução das tarefas (cont.)

- Hoje em dia existe um interesse crescente na **análise estocástica** dos tempos de execução e respectivo impacto em termos de interferência.
- A ideia consiste em determinar a distribuição de probabilidade dos tempos de execução e utilizar uma **estimativa para o tempo máximo** que cubra uma percentagem elevada (e.g. 99%) das ocorrências.
- Em muitos casos (principalmente quando o pior caso é raro e muito maior do que o caso médio) esta técnica permite reduzir drasticamente pessimismo do WCET e aumentar a taxa de utilização do CPU (**maior eficiência**)



Aspectos de implementação prática

Impacto de outras rotinas de atendimento de interrupções

- Regra geral as rotinas de **atendimento de interrupções** executam com um **nível de prioridade superior** ao de todas as tarefas no sistema.
- Assim, num sistema de **prioridades fixas**, o respectivo impacto pode ser directamente tido em conta, incluindo estas **rotinas como tarefas** nas análises de escalonabilidade.
- Em sistemas de **prioridades dinâmicas** a situação é mais complexa (e.g. como atribuir deadlines?). Considera-se, neste caso, que as janelas de tempo em que as rotinas executam não estão disponíveis para a execução de tarefas. Isto pode ser tido em conta na análise de carga imposta ao CPU.

Aspectos de implementação prática

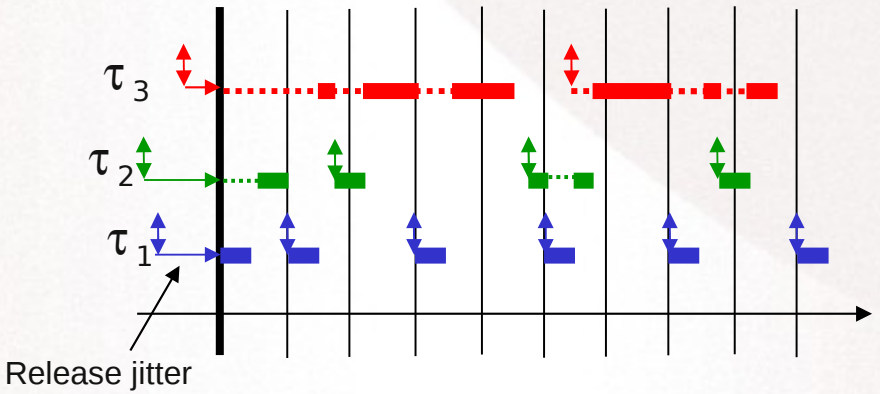
Impacto de desvios nos instantes de activação de tarefas

- Uma tarefa pode sofrer **desvios** nos respectivos instantes de activação, e.g. quando a tarefa é activada pela terminação de outra, ou por uma interrupção externa, ou por uma mensagem recebida por um porto de comunicação, a activação pode variar relativamente ao instante previsto – chama-se a este efeito **variação de disparo** (***release jitter***)
- A existência de *release jitter* tem de ser tida em conta na análise de escalonabilidade pois a tarefa poderá executar fora dos instantes previstos.

Aspectos de implementação prática

Impacto de desvios nos instantes de activação de tarefas

- Uma tarefa pode sofrer **desvios** nos respectivos instantes de activação, e.g. quando a tarefa é activada pela terminação de outra, ou por uma interrupção externa, ou por uma mensagem recebida por um porto de comunicação, a activação pode variar relativamente ao instante previsto – chama-se a este efeito **variação de disparo** (***release jitter***)
- A existência de *release jitter* tem de ser tida em conta na análise de escalonabilidade pois a tarefa poderá executar fora dos instantes previstos.



Aspectos de implementação prática

Impacto de desvios nos instantes de activação de tarefas

- A existência de *release jitter* pode ser tida em conta como uma antecipação dos instantes de activação das instâncias seguintes.

Cálculo de Rwc_i usando preempção e prioridades fixas

$$\forall i, Rwc_i = I_i + C_i, com\ I_i = \sum_{k \in hp(i)} \left\lceil \frac{Rwc_i + J_k}{T_k} \right\rceil * C_k$$

$$Rwc_i(0) = \sum_{k \in hp(i)} C_k + C_i$$

$$Rwc_i(m+1) = \sum_{k \in hp(i)} \left(\left\lceil \frac{Rwc_i(m) + J_k}{T_k} \right\rceil * C_k \right) + C_i$$

Resumo da Aula 9

Outro aspectos do escalonamento de tempo-real

- Escalonamento **sem preempção**
- Questões de **aplicação prática** em sistemas reais