Sistemas de Tempo-Real

Notas de curso realizado em Agosto de 2006 na
Universidade Federal do Rio Grande do Norte, Natal, Brasil

Francisco Vasques
Faculdade de Engenharia da
Universidade do Porto
http://www.fe.up.pt/~vasques

2. Escalonamento de Tempo-Real (parte 1)

Plano das Aulas

- **Introdução aos Sistemas de Tempo-Real**
- **Escalonamento de Tempo-Real**
  - Conceitos e Definições
  - Classificação de Algoritmos de Escalonamento
  - Algoritmos Clássicos de Escalonamento
  - Considerações suplementares
  - Exclusão Mútua no Acesso a Recursos Partilhados
  - Exemplo de Escalonamento em Computação Industrial
- **Protocolos de Comunicação de Tempo-Real**

Universidade do Porto
Faculdade de Engenharia

Plano das Aulas

- **Escalonamento de Tempo-Real**
  - Conceitos e Definições
  - Classificação de Algoritmos de Escalonamento
  - Algoritmos Clássicos de Escalonamento
  - Considerações suplementares
  - Exclusão Mútua no Acesso a Recursos Partilhados
  - Exemplo de Escalonamento em Computação Industrial

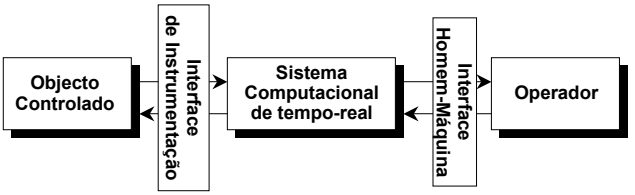
Universidade do Porto
Faculdade de Engenharia

Conceitos e Definições

- **Motivação para o estudo do escalonamento de TR**
  - A utilização de *metodologias e ferramentas convencionais*, tem sido considerada suficiente para o desenvolvimento de sistemas de tempo-real de baixa complexidade. No entanto, as aplicações desenvolvidas poderão ter um *comportamento temporalmente imprevisível*;
  - Para projectos com maior nível de exigência, é fundamental a utilização de *metodologias* de desenvolvimento que considerem de uma forma adequada os *requisitos temporais ao longo de todo o processo de desenvolvimento*;
  - Nomeadamente, para a obtenção de um *escalonamento* adequado da execução de tarefas num sistema de tempo-real, é fundamental a correcta consideração dos seus *requisitos temporais* (*metas temporais & atrasos temporais*).

Conceitos e Definições

- **Noção de Meta Temporal**
  - Meta temporal ("*deadline*") *vs*. Atraso ("*delay*")
    - » A imposição de um atraso ao sistema computacional de tempo-real indica que o sistema computacional deve retardar a sua execução para permitir que o ambiente (operador/objecto controlado) o consiga acompanhar;
    - » A imposição de uma meta temporal ao sistema computacional de tempo-real indica que o sistema computacional deve acelerar a sua execução por forma a adequar o seu tempo de execução à dinâmica do ambiente.



Conceitos e Definições

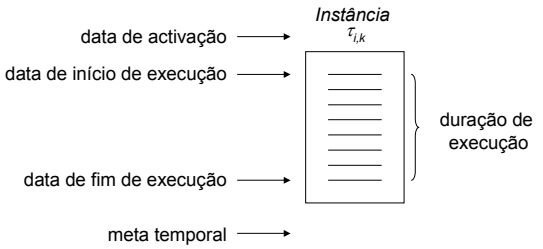
- **Noção de Meta Temporal**
  - Meta temporal ("*deadline*") *vs*. Atraso ("*delay*")
    - » A maior parte dos requisitos temporais podem ser expressos em termos de *atrasos* e de *metas temporais*;
    - » Exemplo:

```
loop
  do work by deadline
  delay until next release
end loop
```

    - » Um *atraso* pode ser facilmente implementado em qualquer RTOS. No entanto, garantir que uma *meta temporal* é sempre respeitada quando da execução do programa de tempo-real não é trivial;
  - A análise de escalonabilidade é o formalismo utilizado para verificação da garantia de respeito das metas temporais.

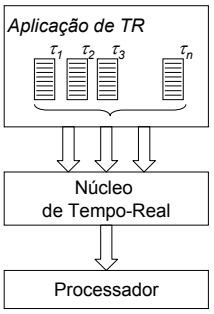
Conceitos e Definições

- **Programas em Sistemas de Tempo-Real**
  - … constituídos por *tarefas*.
- **Conceito de Tarefa:**
  - Tarefa (τ_i): Segmento de código de software que deverá ser executado múltiplas vezes com diferentes dados de entrada. Uma tarefa é caracterizada por ter uma execução com *características temporais próprias*;
  - Instância ("*job*") ($\tau_{i,k}$) é a execução *k* da tarefa τ_i;



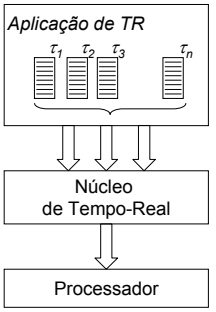
Conceitos e Definições

- **Núcleo ("*Kernel*") Multi-Tarefa de Tempo-Real**
  - Objectivo: *transparência* para a aplicação de tempo-real do processador e dos seus mecanismos de interface;
  - Permitir a execução de múltiplas tarefas num ambiente pseudo-paralelo, garantindo o respeito das metas temporais associadas a cada uma das tarefas.



Conceitos e Definições

- **Núcleo ("*Kernel*") Multi-Tarefa de Tempo-Real**
  - Pseudo-paralelismo: O processador executa *sucessivamente* múltiplas tarefas
    - » Se as diferentes tarefas são executadas sequencialmente: *escalonamento não preemptivo*;
    - » Se uma tarefa em curso de execução pode ser interrompida por uma outra tarefa: *escalonamento preemptivo*.



Conceitos e Definições

- **Núcleo ("*Kernel*") Multi-Tarefa de Tempo-Real**
  - Critérios para a selecção de um Núcleo de tempo-real
    - » Porquê utilizar um núcleo TR?
      - Desenvolvimento mais rápido das aplicações (*ganho de produtividade*), visto a partilha do processador ser transparente.
      - Utilização de espaço de memória suplementar, com *maior custo* em termos de tempo de execução.

    Critérios para a selecção de um Núcleo de tempo-real

    1. Hardware-alvo suportado;
    2. Linguagens de programação suportadas (ex. C, C++, Ada)
    3. Dimensão de memória ocupada ("*footprint*")
       - Núcleos modulares (dimensão de memória variável)
       - Problema: identificar o conteúdo correspondente às dimensões mínimas de memória indicadas pelos fabricantes.

Conceitos e Definições

- **Núcleo ("*Kernel*") Multi-Tarefa de Tempo-Real**
  - Critérios para a selecção de um Núcleo de tempo-real
    4. Serviços fornecidos pelo núcleo
       - Escalonamento: adaptado para o suporte de aplicações de tempo-real?
       - Sincronização entre tarefas: Que tipo de gestão de filas (FIFO, por prioridades)? Que mecanismos para gerir a inversão de prioridades?
       - Gestão estática ou dinâmica de memória? Gestão do tempo: activação de tarefas periódicas? Qualidade das temporizações (granularidade, precisão)?
       - Interruptibilidade do núcleo? Quais os serviços mínimos que têm que ser incluídos numa versão reduzida do núcleo?
    5. Componentes de software fornecidos para além do núcleo
       - Pilhas de protocolos, base de dados de tempo-real, serviços Web?

Conceitos e Definições

- **Núcleo ("*Kernel*") Multi-Tarefa de Tempo-Real**
  - Critérios para a selecção de um Núcleo de tempo-real
    6. Desempenho ("*performance*"):
       - A existência de análises efectuadas pelos fabricantes devem ser escrutinadas com base nos seguintes elementos: qual a plataforma para a qual foram efectuadas? em que condições de medida? tempos médios, máximos ou mínimos? presença de interrupções? testes em anel (tudo na cache!)? qual o estado do núcleo (n.º de objectos, fragmentação)?
    7. Existência de *drivers* para os periféricos a utilizar
    8. Qualidade do suporte técnico, para futura evolução para novas arquitecturas
    9. Natureza do produto (código fonte ou binário?)
    10. Custo (preço por licença?)
    11. Certificação para a área de aplicação pretendida?

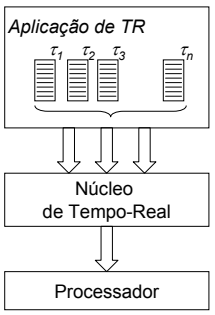
Conceitos e Definições

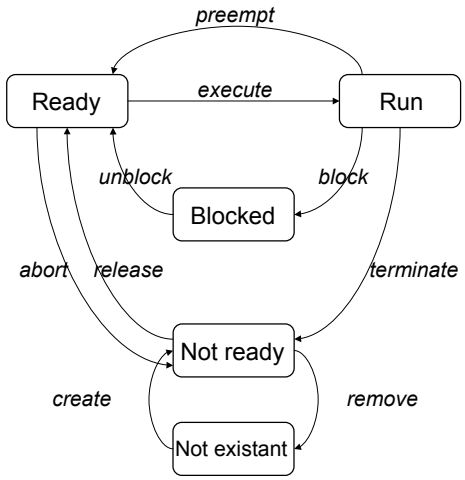
- **Objectivo de um Núcleo ("*Kernel*") Multi-Tarefa de Tempo-Real**
  - O elemento fundamental de um núcleo ("*Kernel*") de tempo-real é o *escalonador*, que tem como função permitir a execução de múltiplas tarefas num ambiente pseudo-paralelo, garantindo o respeito das sua metas temporais.

- **Escalonamento ("Scheduling") de Tempo-Real:**
  - O escalonamento de tempo-real pretende atribuir de uma forma óptima o processador às tarefas que os pretendem utilizar, garantindo o respeito das metas temporais ("*deadlines*") associadas à execução de cada uma das tarefas.

Conceitos e Definições

- **Estados de uma tarefa**
  - Durante a sua execução, uma tarefa de uma aplicação de tempo-real pode estar num, e num só, dos seguintes estados:





Conceitos e Definições

- **Um algoritmo de escalonamento de tempo-real deve:**
  - » Seleccionar entre as tarefas pendentes ("ready"), qual a que deve ser executada;
  - » Verificar se deve interromper a tarefa que está a ser executada ("*preemption*"), para permitir a execução de uma tarefa de maior prioridade;
  - » Verificar se as regras de execução impõem o bloqueio ("*blocking*") da tarefa que está a ser executada;

  por forma a optimizar a execução do conjunto de tarefas.

Conceitos e Definições

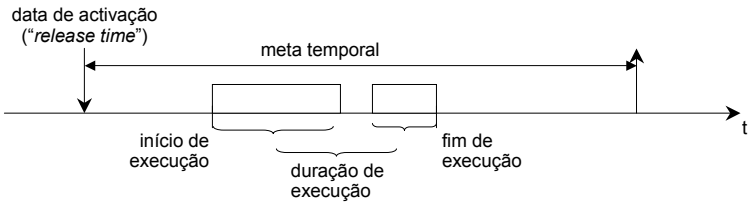
- **Escalonamento ("Scheduling") de Tempo-Real:**
  - – O escalonamento de tempo-real inclui a definição de:
    - » *Algoritmo de escalonamento* (ordem de sequência) do processamento de tarefas, combinado com as políticas de alocação de recursos;
      - Deve garantir a eficiente utilização dos recursos (processador e redes de comunicação)
    - » *Teste* para a análise da *escalonabilidade* do conjunto de tarefas, qualquer que seja a sua ordem de activação
      - Fórmula/algoritmo matemático que forneça uma prova de escalonabilidade;
      - Também permite efectuar uma previsão acerca do funcionamento do sistema, quando confrontado com o cenário de carga ou de falhas mais desfavorável.

Conceitos e Definições

- **Análise de escalonabilidade de um programa de tempo-real**
  - Elemento fundamental para determinar o comportamento temporal de um programa de tempo-real;
  - "*Guidelines*" para a análise de escalonabilidade de um programa de tempo-real:
    - » Definição de um *modelo* adequado para as tarefas, descrevendo de forma adequada as suas *propriedades temporais*;
    - » Definição de um *algoritmo de escalonamento* adequado para o sistema em análise;
    - » Utilização de um *teste* para a análise da *escalonabilidade* adequado para o algoritmo de escalonamento e o modelo de tarefas considerados.

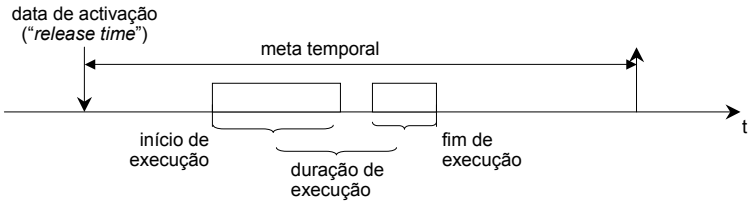
Conceitos e Definições

- **Modelo de tarefas:**
  - Parâmetros fundamentais de um modelo de tarefas, que permita descrever de uma forma adequada as *propriedades temporais* de um sistema de tempo-real:
    - valores relativos (letra maiúscula)
    - » *periodicidade de activação* de cada tarefa: T_i
    - » máxima *duração de execução* de cada tarefa (sem preempção): C_i
    - » *meta temporal* ("deadline") associada à execução de cada tarefa: D_i



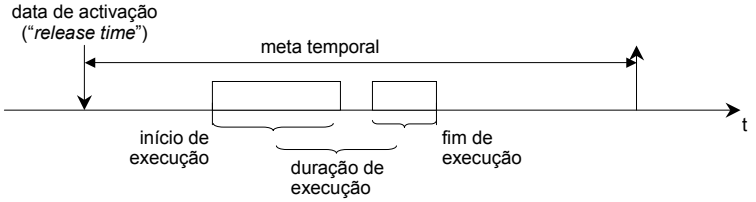
Conceitos e Definições

- **Modelo de tarefas:**
  - Parâmetros fundamentais de um modelo de tarefas:
    - valores absolutos (letra minúscula)
    - » *data de activação* da instância *k* da tarefa τ_i: $r_{i,k}$
    - » *data de início de execução* da instância *k* da tarefa τ_i: $s_{i,k}$
    - » *data de fim de execução* da instância *k* da tarefa τ_i: $f_{i,k}$



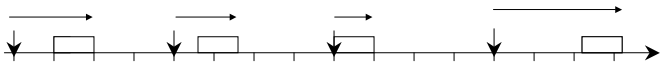
Conceitos e Definições

- **Outros parâmetros temporais:**
  - *Folga* (“laxity”): *L* = (meta temporal) - (conclusão de execução)
  - *Atraso* (“delay”) = MAX (0 , - Folga)
  - *Tempo de Resposta*: *R* = (conclusão de execução) - (data de activação)
  - *Factor de utilização* : $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$
  - Relações de precedência e de exclusão mútua entre tarefas;



Conceitos e Definições

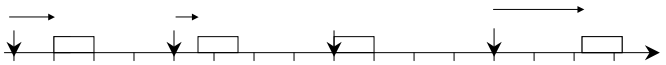
- **Outros parâmetros temporais:**
  - *"Jitter"*: variação temporal de um evento com activação periódica
  - "*Jitter*" no final de execução



» valor absoluto: $\max_k\left(f_{i,k}-r_{i,k}\right)-\min_k\left(f_{i,k}-r_{i,k}\right)$

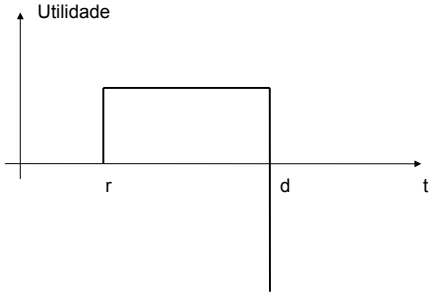
» valor relativo: $\max_k\left|\left(f_{i,k}-r_{i,k}\right)-\min_k\left(f_{i,k-1}-r_{i,k-1}\right)\right|$

  - "*Jitter*" no início de execução



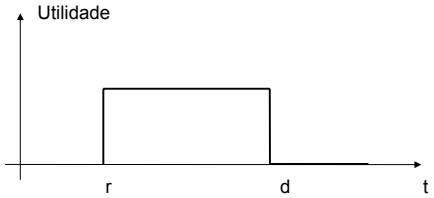
Conceitos e Definições

- **Modelo de tarefas**
  - *Tarefa crítica ("hard task")*, quando a ultrapassagem de uma meta temporal pode ter consequências desastrosas, ou seja, o valor do serviço prestado fora de prazo é muito menor que o valor negativo do prejuízo causado;



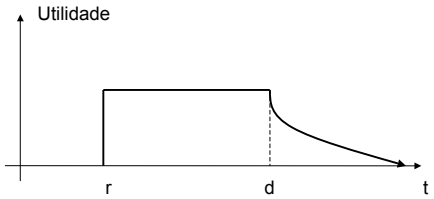
Conceitos e Definições

- **Modelo de tarefas**
  - *Tarefa firme ("firm task")*, quando a ultrapassagem de uma meta temporal não tem consequências desastrosas, mas o valor do serviço prestado pode ser considerado nulo;



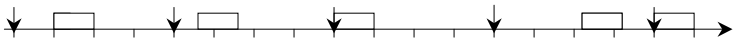
Conceitos e Definições

- **Modelo de tarefas**
  - *Tarefa não-crítica ("soft task")*, quando apesar da ultrapassagem de uma meta temporal, ainda existe algum valor associado ao serviço prestado fora de prazo;



Conceitos e Definições

- **Modelo de Tarefas:**
  - *Tarefa Periódica*:
    - » Tarefa com periodicidade de activação (e não de execução) constante;



  - *Tarefa Esporádica*:
    - » Tarefa cujo intervalo de tempo entre activações consecutivas tem um mínimo definido

Conceitos e Definições

- **Modelo de Tarefas:**
  - *Tarefa Aperiódica*:
    - » Tarefa cujo intervalo de tempo entre activações consecutivas não tem nenhum mínimo definido.
    - » Para poder ser considerada a utilização de tarefas aperiódicas em sistemas de tempo-real, torna-se necessário impor um limite superior à sua utilização de recursos computacionais.

Conceitos e Definições

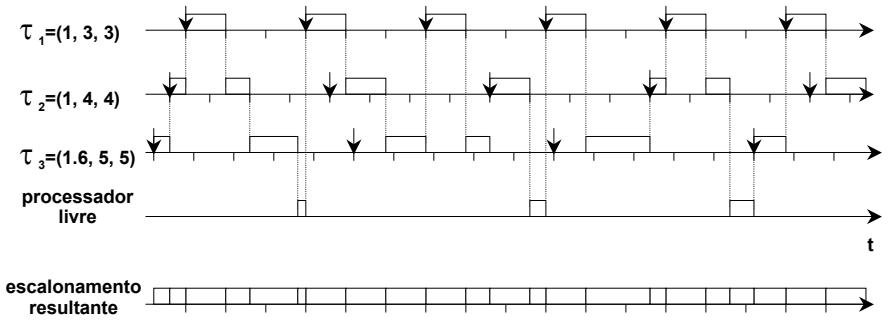
- **Preempção na Execução de Tarefas**
  - Num *escalonamento preemptivo*:
    - » uma tarefa de menor prioridade verá a sua *execução interrompida* sempre que uma tarefa de maior prioridade deseje ser executada (a menos que existam restrições que impeçam essa interrupção);
  - Num *escalonamento não-preemptivo*:
    - » cada tarefa será executada até ao final sem ser interrompida, sendo desta forma simplificado o processamento associado às mudanças de contexto;

Conceitos e Definições

- **Exemplos de escalonamento:**
  - Apresentam-se, para o mesmo conjunto de tarefas, 4 cenários de escalonamento para os quais se obtém sucessivamente:
    1. Escalonamento preemptivo, com 1os pedidos de execução arbitrariamente desfasados;
    2. Escalonamento preemptivo, com simultaneidade nos 1os pedidos de execução;
    3. Escalonamento não- preemptivo, com simultaneidade nos 1os pedidos de execução;
    4. Escalonamento não-preemptivo com activação de tarefa de menor prioridade imediatamente anterior à activação simultânea de todas as outras tarefas

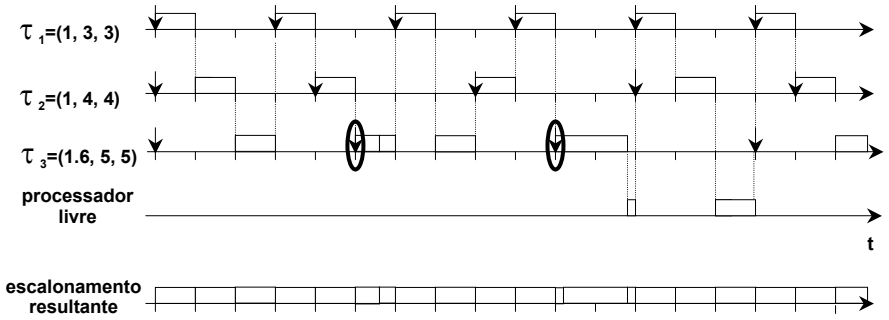
Conceitos e Definições

- **1º Exemplo:**
  - » Tarefas Periódicas: $\tau=\{C_i, T_i, d_i\}$; U=93,3%
  - » *Escalonamento preemptivo* por prioridades
  - » Atribuição de prioridades às tarefas (?)
  - » *1os pedidos de execução desfasados*



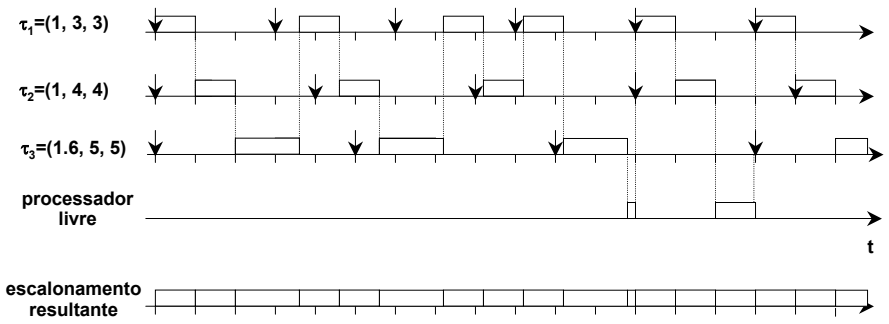
Conceitos e Definições

- **2º Exemplo:**
  - » Tarefas Periódicas: $\tau=\{C_i, T_i, d_i\}$; U=93,3%
  - » *Escalonamento preemptivo* por prioridades
  - » Atribuição de prioridades às tarefas (?)
  - » *1os pedidos de execução simultâneos*



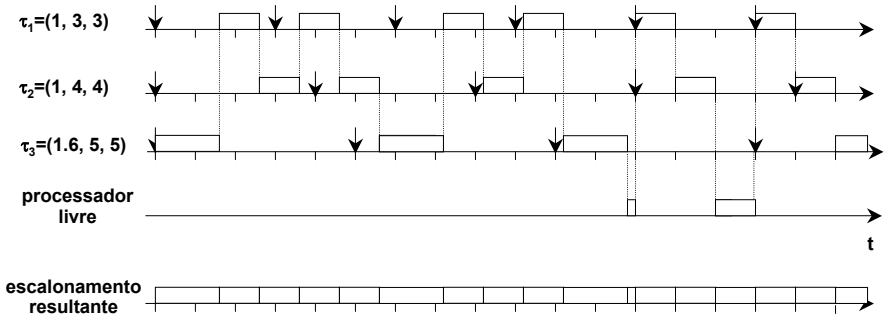
Conceitos e Definições

- **3º Exemplo:**
  - » Tarefas Periódicas: $\tau=\{C_i, T_i, d_i\}$; U=93,3%
  - » *Escalonamento não-preemptivo* por prioridades
  - » Atribuição de prioridades às tarefas (?)
  - » *1os pedidos de execução simultâneos*



Conceitos e Definições

- **4º Exemplo:**
  - » Tarefas Periódicas: $\tau=\{C_i, T_i, d_i\}$; U=93,3%
  - » *Escalonamento não-preemptivo* por prioridades
  - » *1º pedido de execução de uma tarefa de menor prioridade imediata-mente anterior à activação simultânea de todas as outras tarefas*



Conceitos e Definições

- **Instante Crítico:**
  - Num sistema com escalonamento preemptivo, a activação simultânea de todas as tarefas (*instante crítico*) tem como consequência:
    - » em termos de utilização, um *período de carga máxima* do processador;
    - » em termos de tempo de resposta, um intervalo de tempo (*intervalo crítico*) durante o qual o tempo de resposta é máximo para cada uma das activações das tarefas.
  - Num sistema com escalonamento não-preemptivo, o instante crítico está definido como a activação da tarefa de maior duração (efeito de bloqueio) imediatamente antes da activação simultânea de todas as outras tarefas [Liu and Layland, 73].

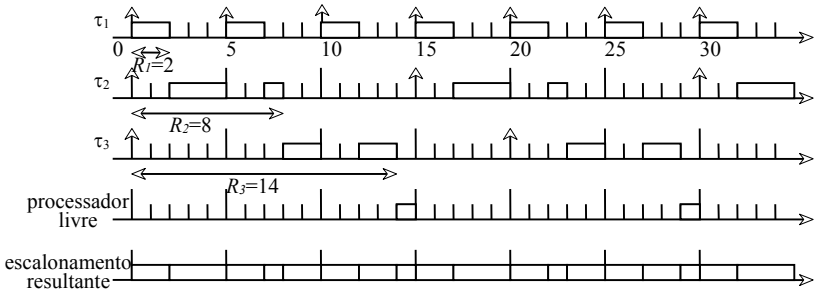
Conceitos e Definições

- **Exemplos de escalonamento:**
  - Apresentam-se, para o mesmo conjunto de tarefas, 3 cenários de escalonamento para os quais se obtém sucessivamente:
    1. Escalonamento preemptivo sempre realizável (intervalo crítico representado);
    2. Escalonamento não-preemptivo sobre o qual não pode ser retirada nenhuma conclusão acerca da sua escalonabilidade (intervalo crítico não representado);
    3. Escalonamento não-preemptivo não realizável (intervalo crítico representado).

Conceitos e Definições

- **1º Exemplo**
  - » tarefas periódicas: $\tau=\{C_i, T_i, d_i\}$; U=86,7%;
  - » data de activação das tarefas simultânea;
  - » *escalonamento sempre realizável*.

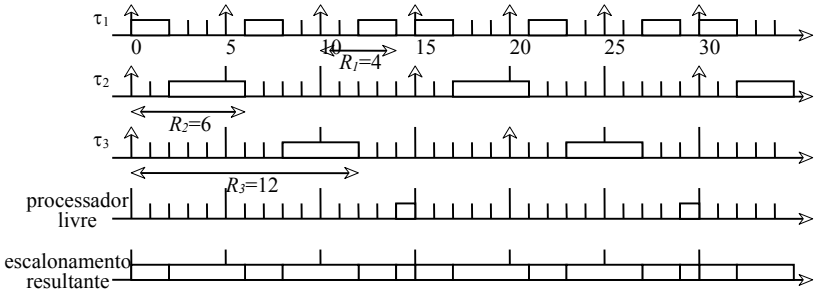
tarefa	*C*	*T*	*d*
τ_1	*2*	*5*	*5*
τ_2	*4*	*15*	*15*
τ_3	*4*	*20*	*20*



Conceitos e Definições

- **2º Exemplo**
  - » tarefas periódicas: $\tau=\{C_i, T_i, d_i\}$; U=86,7%;
  - » data de activação das tarefas simultânea;
  - » intervalo crítico não representado.

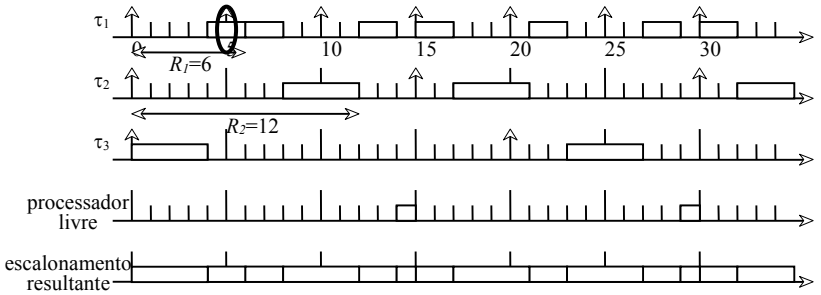
tarefa	*C*	*T*	*d*
τ_1	*2*	*5*	*5*
τ_2	*4*	*15*	*15*
τ_3	*4*	*20*	*20*



Conceitos e Definições

- **3º Exemplo**
  - » tarefas periódicas: $\tau=\{C_i, T_i, d_i\}$; U=86,7%;
  - » data de activação das tarefas não simultânea;
  - » *escalonamento não realizável*.

tarefa	*C*	*T*	*d*
τ_1	2	5	5
τ_2	4	15	15
τ_3	4	20	20



Conceitos e Definições

- **Desempenho do escalonamento preemptivo vs. não-preemptivo**
  - Em termos de tempo de resposta, um *escalonamento preemptivo* permite um *mais rápido início de execução das tarefas* de maior prioridade, logo é o tipo de escalonamento preferido
    - apesar de estar sujeito a uma sobrecarga, potencialmente elevada, associada às múltiplas mudanças de contexto;
  - Igualmente em termos de tempo de resposta, um *escalonamento não-preemptivo* será *interessante para sistemas com um elevado numero de tarefas de duração reduzida* (relativamente ao tempo de duração associado às mudanças de contexto).

Conceitos e Definições

- **Teste de Escalonabilidade**

  verificação "*off-line*" da escalonabilidade de um sistema

  – *Teste para determinar a exequibilidade do escalonamento* de um conjunto de tarefas, considerando um algoritmo determinado para a atribuição de prioridades às tarefas.

    » *Teste exacto* de escalonabilidade indica se um determinado conjunto de tarefas é ou não escalonável.

  – Este tipo de testes pode ser de grande complexidade, pelo que por vezes se opta pela utilização de testes não exactos, mas de maior simplicidade algorítmica:

    » *Teste necessário*: um teste necessário de escalonabilidade negativo, garante que o conjunto de tarefas é sempre não escalonável;

    » *Teste suficiente*: um teste suficiente de escalonabilidade positivo, garante que o conjunto de tarefas é sempre escalonável;

Conceitos e Definições

- **Teste de Escalonabilidade**

  verificação "*on-line*" da escalonabilidade de um sistema

  – *Teste para a aceitação/rejeição* da inclusão de uma nova tarefa num conjunto de tarefas previamente escalonável;

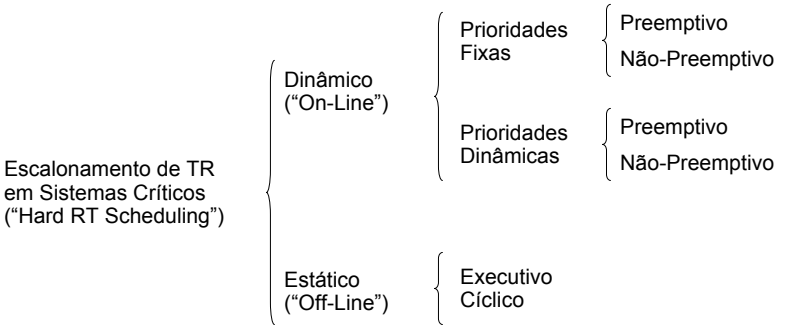
    » Este tipo de testes deve ser de reduzida complexidade, pelo que existe o risco de rejeição de tarefas potencialmente escalonáveis.

Plano das Aulas

- **Escalonamento de Tempo-Real**
  - Conceitos e Definições
  - Classificação de Algoritmos de Escalonamento
  - Algoritmos Clássicos de Escalonamento
  - Considerações suplementares
  - Exclusão Mútua no Acesso a Recursos Partilhados
  - Exemplo de Escalonamento em Computação Industrial

Classificação de Algoritmos de Escalonamento

- **Classificação de Algoritmos de Escalonamento (Stankovic et.al.)**

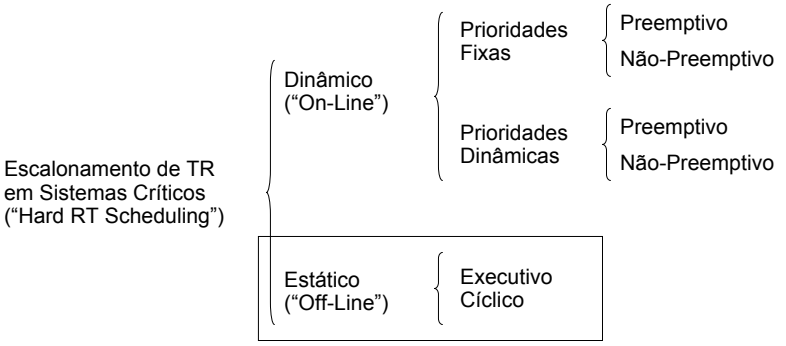


Classificação de Algoritmos de Escalonamento

- **Escalonamento em Sistemas Críticos vs. em Sistemas Não-Críticos**
  - O escalonamento em *sistemas críticos* é baseado na *adequabilidade de recursos*, ou seja, na garantia em fase de concepção de um tempo de resposta adequado para a execução de cada uma das tarefas;
    - » Os *testes de escalonabilidade* a efectuar previamente devem ser *determinísticos*, ou seja, devem verificar que, para os pressupostos de carga (e de falhas) assumidos, o tempo de resposta máximo associado a cada tarefa é inferior à sua meta temporal;
  - Para o caso de *sistemas não-críticos*, sabendo que a violação de metas temporais não é crítica, admite-se a utilização de métodos *probabilísticos* (por exemplo baseados em simulações) para a verificação da garantia de escalonabilidade.

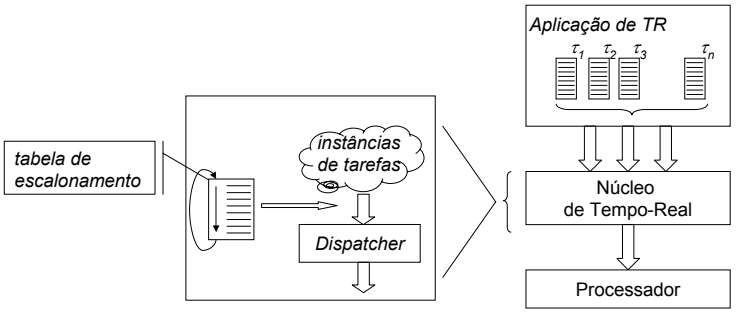
Classificação de Algoritmos de Escalonamento

- **Classificação de Algoritmos de Escalonamento (Stankovic *et.al.*)**



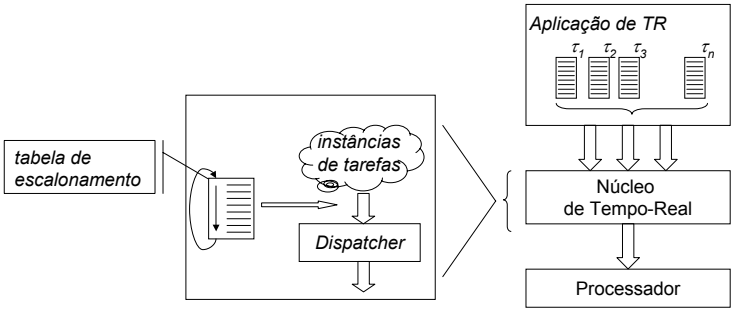
Classificação de Algoritmos de Escalonamento

- **Escalonamento Estático (Executivo Cíclico)**
  - Um *escalonador* é considerado *estático* se tomar previamente todas as decisões de escalonamento, gerando, em tempo de compilação, uma *tabela de escalonamento com a sequência de execução das tarefas*.
    - » Em tempo de execução, esta tabela será repetidamente executada pelo processo de "*dispatcher*", com a periodicidade adequada.



Classificação de Algoritmos de Escalonamento

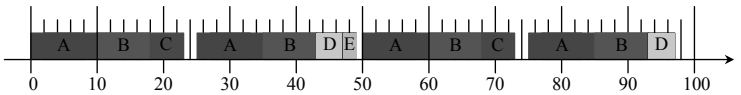
- **Escalonamento Estático (Executivo Cíclico)**
  - Num escalonador estático, qualquer variação no modelo de tarefas implicará a geração de uma nova tabela de escalonamento.
  - A *garantia de escalonabilidade é fornecida por simples inspecção da tabela* de escalonamento, por forma a ser verificada a não ultrapassagem de nenhuma meta temporal.



Classificação de Algoritmos de Escalonamento

- **Exemplo de Escalonamento Estático:**

tarefa	*C*	*T*	*d*	*U*
A	*10*	*25*	*25*	*0,4*
B	*8*	*25*	*25*	*0,32*
C	*5*	*50*	*50*	*0,1*
D	*4*	*50*	*50*	*0,08*
E	*2*	*100*	*100*	*0,02*
			U total:	*0,92*



– Tabela de escalonamento organizada em micro-ciclos com duração igual a m.d.c.{T}=25, sendo o comprimento total da tabela de escalonamento (macro-ciclo) igual ao m.m.c. {T}=100.

Universidade do Porto
Faculdade de Engenharia

Classificação de Algoritmos de Escalonamento

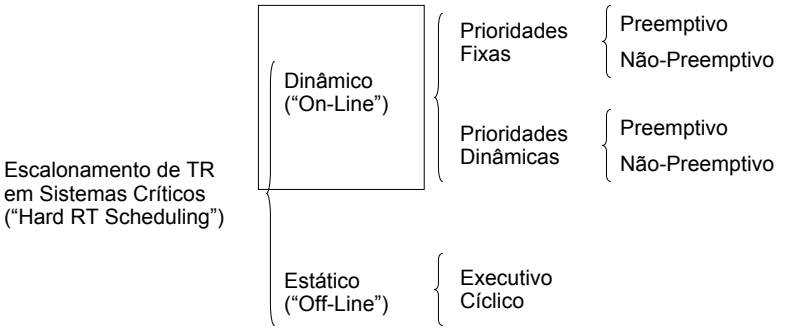
- **Escalonamento Estático:**
  - *Vantagens* de um escalonamento estático:
    - » Sendo geralmente baseados em algoritmos heurísticos, fornece um *suporte* eficaz à implementação de *relações de precedência* entre processos;
    - » Comportamento do sistema completamente *previsível*;
    - » *Sobrecarga mínima* em tempo de execução;
    - » Muito utilizado para *suporte de aplicações de elevada criticalidade* (devido à simplicidade (!) do processo de certificação).

Classificação de Algoritmos de Escalonamento

- **Escalonamento Estático:**
  - *Desvantagens* de um escalonamento estático:
    - » Escalonamento de *baixa flexibilidade*, dificultando tanto operações de modificação do conjunto de tarefas, como a execução de tarefas esporádicas e/ou aperiódicas;
  - *Dificuldades* relacionadas com a sua *implementação*
    - » Tarefas com duração elevada deverão ser repartidas entre múltiplos micro-ciclos, o que não é desejável em termos de engenharia de software.
    - » A periodicidade das tarefas deve ser simultaneamente múltipla do valor do micro-ciclo e divisível pelo valor do macro-ciclo, impondo regras apertadas para a escolha destes valores;

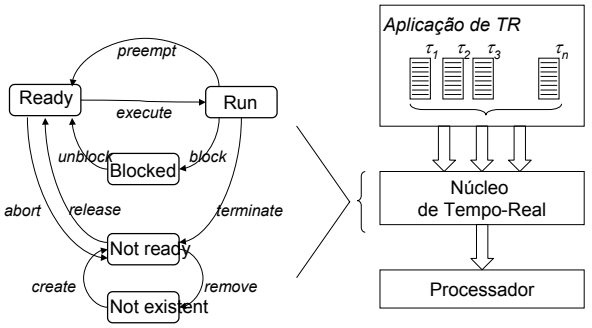
Classificação de Algoritmos de Escalonamento

- **Classificação de Algoritmos de Escalonamento (Stankovic *et.al.*)**



Classificação de Algoritmos de Escalonamento

- **Escalonamento Dinâmico (ou por prioridades)**
  - Um *escalonador dinâmico* toma as *decisões de escalonamento em tempo de execução* ("run time"), em função dos pedidos de execução pendentes;
    - » Atribuição do processador ("*ready*" → "*run*") à tarefa pendente mais prioritária;
    - » Também referido como *"escalonador por prioridades"*.



Classificação de Algoritmos de Escalonamento

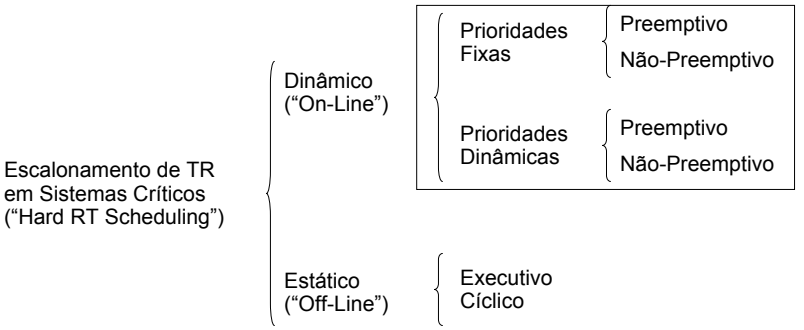
- **Escalonamento Dinâmico (ou por prioridades)**
  - *Vantagens*:
    - » adapta-se facilmente à presença de tarefas com requisitos variáveis no tempo, visto efectuar o escalonamento em tempo de execução;
    - » permite considerar a "importância relativa" das tarefas em função dos seus requisitos temporais.
  - *Aspectos relevantes a considerar*:
    - » A transformação "*requisitos temporais*" → "*prioridade das tarefas*" é da responsabilidade do utilizador;
    - » O escalonador não efectua a verificação do respeito das metas temporais associadas à execução das tarefas.

Classificação de Algoritmos de Escalonamento

- **Escalonamento Dinâmico (ou por prioridades)**
  - *Desvantagens*:
    - » processo escalonador mais complexo do que no caso estático, devido à necessidade de efectuar o escalonamento em tempo de execução;
    - » maior dificuldade de detecção de sobrecargas.

Classificação de Algoritmos de Escalonamento

- **Classificação de Algoritmos de Escalonamento (Stankovic *et.al.*)**



Classificação de Algoritmos de Escalonamento

- **Escalonamento com Prioridades Fixas vs. Prioridades Dinâmicas**
  - Num escalonamento *dinâmico com prioridades fixas*, a cada tarefa é atribuído um determinado nível de prioridade na fase de concepção. Enquanto o modo de funcionamento do sistema se mantiver, o *nível de prioridade* de cada tarefa permanece *inalterado* (a menos de mudanças impostas por mecanismos de sincronização no acesso a recursos partilhados);

Universidade do Porto
Faculdade de Engenharia

Classificação de Algoritmos de Escalonamento

- **Escalonamento com Prioridades Fixas vs. Prioridades Dinâmicas**
  - Num escalonamento *dinâmico com prioridades dinâmicas*, o *nível de prioridade evolui ao longo do tempo* em função da política de escalonamento seleccionada.
    - » *Exemplo*: algoritmo EDF (“Earliest Deadline First”), para o qual o nível de prioridade de uma tarefa será tanto maior quanto mais próxima estiver a sua meta temporal.

Plano das Aulas

- **Escalonamento de Tempo-Real**
  - Conceitos e Definições
  - Classificação de Algoritmos de Escalonamento
  - Algoritmos Clássicos de Escalonamento
    - » Algoritmo Rate Monotonic (RM)
    - » Cálculo do Tempo de Resposta
    - » Algoritmo Deadline Monotonic (DM)
    - » Algoritmo Earliest Deadline First (EDF)
  - Considerações suplementares
  - Exclusão Mútua no Acesso a Recursos Partilhados
  - Exemplo de Escalonamento em Computação Industrial

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Rate Monotonic" [Liu and Layland, 1973]**
  - Algoritmo definido para um modelo simplificado de tarefas:
    - » as tarefas são independentes entre si (não têm relações de precedência, nem necessidades de sincronização);
    - » todas as tarefas têm uma activação periódica, com uma meta temporal igual ao seu período (ou seja, qualquer execução deve ser finalizada antes da próxima data de activação da tarefa);
    - » todas as tarefas têm um tempo máximo de execução conhecido.

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Rate Monotonic" [Liu and Layland, 1973]**
  - Este algoritmo define a ordem de *atribuição de prioridades* a um conjunto de tarefas, *na ordem inversa da periodicidade* das tarefas: $T_i < T_j \Rightarrow P_i > P_j$
    - » *desde* a tarefa de menor período à qual é atribuída a maior prioridade, *até* à tarefa de maior periodicidade à qual é atribuída a menor prioridade;
    - » as situações de empate serão resolvidas arbitrariamente;
  - Trata-se de um *algoritmo óptimo* para sistemas mono-processador, no sentido que se um qualquer conjunto de tarefas (periódicas, independentes, $d_i = T_i$,…) pode ser escalonado por escalonador *dinâmico com prioridades fixas*, então também pode ser escalonado pelo algoritmo RM.

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Rate Monotonic" [Liu and Layland, 1973]**
  - Teste suficiente de escalonabilidade para um caso preemptivo:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n\left(\sqrt[n]{2} - 1\right)$$

    - » um teste suficiente (não necessário) significa que poderá haver conjuntos de tarefas escalonáveis apesar de não respeitarem o respectivo teste.
    - » para diferentes conjuntos de tarefas, $n = 2, \quad U = 0{,}83$
      $n = 3, \quad U = 0{,}78$
      $n = 4, \quad U = 0{,}76$
      $n \to +\infty, \; U \to 0{,}693$

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Rate Monotonic" [Liu and Layland, 1973]**
  - Vantagens do algoritmo RM
    - » Simplicidade;
    - » Adequado para utilização em sistemas operativos existentes;
    - » Pode ser utilizado para a atribuição de prioridades a níveis de interrupção.
  - Desvantagens do algoritmo RM
    - » Modelo de tarefas muito limitado;
    - » Não adequado quando se têm metas temporais inferiores ao período;
    - » Não suporta exclusão mútua no acesso a recursos partilhados.

Universidade do Porto
Faculdade de Engenharia

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Rate Monotonic" [Liu and Layland, 1973]**
  - Resultado fundamental:
    - » Em [Liu and Layland, 1973] foi também demostrado que, se a meta temporal da tarefa de menor prioridade for respeitada após um instante crítico, então o conjunto de tarefas é sempre escalonável.

Algoritmos Clássicos de Escalonamento

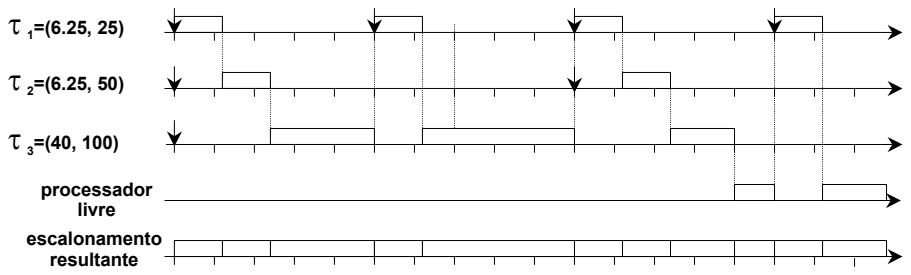
- **Exemplos de escalonamento utilizando o algoritmo RM :**
  - Apresentam-se 3 cenários de escalonamento utilizando o algoritmo RM (considerando conjuntos de tarefas com diferentes taxas de activação), para os quais se obtém sucessivamente:
    - » Conjunto de tarefas para o qual o teste de escalonabilidade é respeitado, logo:
      - o conjunto de tarefas é sempre escalonável.
    - » Conjunto de tarefas para o qual o teste de escalonabilidade não é respeitado. No entanto, devido ao facto de, após o instante crítico, a meta temporal da tarefa de menor prioridade ser respeitada, então:
      - o conjunto de tarefas é sempre escalonável.
    - » Conjunto de tarefas para o qual nem o teste de escalonabilidade, nem a meta temporal da tarefa de menor prioridade (após o instante crítico) são respeitados, logo:
      - o conjunto de tarefas não é escalonável

Algoritmos Clássicos de Escalonamento

- **1º Exemplo:**
  - » tarefas periódicas: $\tau=\{C_i, T_i\}$; $d_i=T_i$; U=77,5%;
  - » data de activação das tarefas simultânea;
  - » *escalonamento sempre realizável*:
    - teste *suficiente* de escalonabilidade *respeitado*:

tarefa	*C*	*T*	*U*
τ_1	6.25	25	0.25
τ_2	6.25	50	0.125
τ_3	40	100	0.4

$$U = 0{,}775 \leq 0{,}7798$$

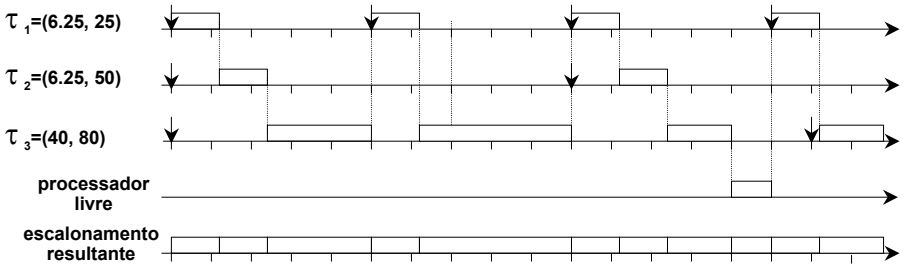


Algoritmos Clássicos de Escalonamento

- **2º Exemplo:**
  - » tarefas periódicas: t={Ci, Ti}; di=Ti ; U=87,5%;
  - » data de activação das tarefas simultânea;
  - » escalonamento sempre realizável:
    - teste suficiente de escalonabilidade não é respeitado:
    - no entanto, a meta temporal da tarefa t3 é respeitada após o instante crítico, logo o conjunto de tarefas é sempre escalonável.

tarefa	*C*	*T*	*U*
τ_1	6.25	25	0.25
τ_2	6.25	50	0.125
τ_3	40	80	0.5

$$U = 0{,}8750 \leq 0{,}7798$$

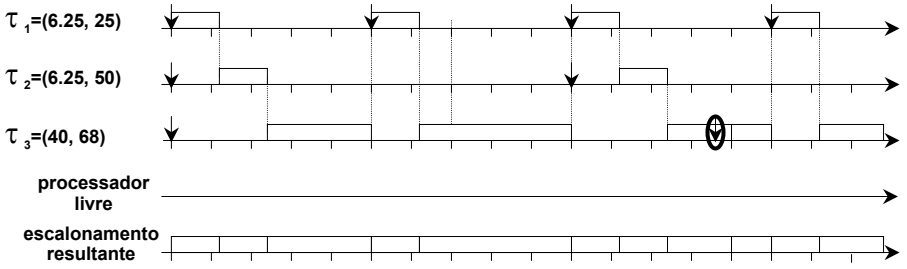


Algoritmos Clássicos de Escalonamento

- **3º Exemplo:**
  - » tarefas periódicas: t={Ci, Ti}; di=Ti ; U=96,3%;
  - » data de activação das tarefas simultânea;
  - » escalonamento não é realizável:
    - teste suficiente de escalonabilidade não é respeitado:
    - meta temporal da tarefa t3 não é respeitada após o instante crítico.

tarefa	*C*	*T*	*U*
τ_1	6.25	25	0.25
τ_2	6.25	50	0.125
τ_3	40	68	0.588

$$U = 0{,}963 \leq 0{,}7798$$



Algoritmos Clássicos de Escalonamento

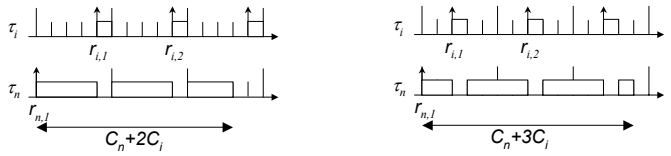
- **Prova de optimalidade do algoritmo RM**
  - Passos para a demonstração:
    1. Mostrar que o *instante crítico* para uma tarefa τ_i ocorre quando a data de activação desta tarefa é simultânea com a data de activação de todas as tarefas de prioridade superior;
    2. Mostrar que *após a ocorrência de um instante crítico*, se um conjunto de tarefas é escalonável com uma atribuição arbitrária de prioridades, então também é escalonável pelo algoritmo RM
       - Demonstração para o caso de 2 tarefas;
       - Extensão ao caso de *n* tarefas.

Instante crítico para uma tarefa τ_i: data de activação da tarefa τ_i que conduz ao mais longo intervalo de tempo para concluir a execução da tarefa menos prioritária (τ_n).

Universidade do Porto
Faculdade de Engenharia

Algoritmos Clássicos de Escalonamento

- **Prova de optimalidade do algoritmo RM**
  1. Mostrar que o *instante crítico* para uma tarefa τ_i ocorre quando a data de activação desta tarefa é simultânea com a data de activação de todas as tarefas de prioridade superior;
     - » Considere-se a tarefa τ_n (menor prioridade) e uma tarefa τ_i (priorid. intermédia)



  - Se se antecipar a data de activação ($r_{i,k}$) para a tarefa de prioridade intermédia, isso pode significar um incremento na data de fim de execução da tarefa menos prioritária. A data de fim de execução mais tardia para a tarefa τ_n será quando $r_{i,1}=r_{n,1}$;
  - Este resultado é transponível para qualquer par de tarefas, pelo que a demonstração pode ser facilmente alargada ao conjunto de *n* tarefas.

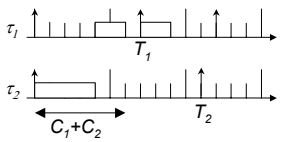
Algoritmos Clássicos de Escalonamento

- **Prova de optimalidade do algoritmo RM**
  - Mostrar que *após a ocorrência de um instante crítico*, se um conjunto de tarefas é escalonável com uma atribuição arbitrária de prioridades, então também é escalonável pelo algoritmo RM.
    - » Considerem-se duas tarefas τ_1 e τ_2;
    - » Metodologia de prova:
      - Atribuição de prioridades arbitrária (não RM)
      - Verificação da condição de escalonabilidade (E_1)
      - Atribuição de prioridades RM
      - Demonstra-se que se E_1 se verificar, então o sistema é igualmente escalonável pelo algoritmo RM

Universidade do Porto
Faculdade de Engenharia

Algoritmos Clássicos de Escalonamento

- **Prova de optimalidade do algoritmo RM**
  - Atribuição de prioridades arbitrária ($P_2 > P_1$)
    - » Considerando o instante crítico, o sistema é escalonável sse: $C_1+C_2<T_1$ (E_1)



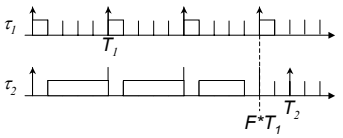
    - » Se E_1 se verificar, então o sistema é também escalonável segundo RM;
  - Atribuição de prioridades segundo RM
    - » Seja F o numero de períodos de τ_1 integralmente contidos em T_2 ($F \geq 1$)

$$F = \left\lfloor \frac{T_2}{T_1} \right\rfloor$$

Algoritmos Clássicos de Escalonamento

- **Prova de optimalidade do algoritmo RM**

  Caso A: C_1 tem uma duração suficientemente curta por forma que a instância de τ_1 activada no instante $F*T_1$ possa ser terminada antes da data T_2;



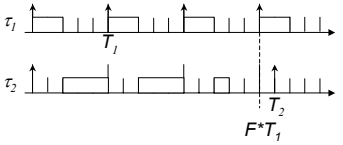
$$C_1 \le T_2 - FT_1 \qquad \text{(caso A)}$$

  - » O sistema é escalonável se $(F+1)C_1 + C_2 \le T_2$ (E_2)
  - » Demonstra-se que, se se verifica (E_1) (caso não RM), então (E_2) também se verifica (caso RM)
    - de acordo com (E1): $C_1+C_2<T_1$, multiplicando por F: $F*C_1+F*C_2 \le F*T_1$
    - Como $F \ge 1$, então $F*C_1+C_2 \le F*C_1+F*C_2 \le F*T_1$
    - Somando C_1 em ambos os termos: $(F+1)*C_1+C_2 \le F*T_1+C_1$
    - Como no caso A, $C_1 \le T_2-F*T_1$, então $(F+1)C_1+C_2 \le F*T_1+T_2-F*T_1$, ou seja, $(F+1)C_1+C_2 \le T_2$, o que verifica (E_2)

Algoritmos Clássicos de Escalonamento

- **Prova de optimalidade do algoritmo RM**

  Caso B: C_1 tem uma duração que não permite que a instância de τ_1 activada no instante $F*T_1$ possa ser terminada antes da data T_2;



$$C_1 \ge T_2 - FT_1 \qquad \text{(caso B)}$$

  - » O sistema é escalonável se $FC_1 + C_2 \le FT_1$ (E_3)
  - » Demonstra-se que, se se verifica (E_1) (caso não RM), então (E_3) também se verifica (caso RM)
    - de acordo com (E_1): $C_1+C_2<T_1$, multiplicando por F: $F*C_1+F*C_2 \le F*T_1$
    - Como $F \ge 1$, então $F*C_1+C_2 \le F*C_1+F*C_2 \le F*T_1$, o que verifica ($E_3$)
  - » Fica desta forma demonstrado que, se o sistema é escalonável com prioridades arbitrárias (E_1), então também o é com RM.

Plano das Aulas

- **Escalonamento de Tempo-Real**
  - Conceitos e Definições
  - Classificação de Algoritmos de Escalonamento
  - Algoritmos Clássicos de Escalonamento
    - » Algoritmo Rate Monotonic (RM)
    - » Cálculo do Tempo de Resposta
    - » Algoritmo Deadline Monotonic (DM)
    - » Algoritmo Earliest Deadline First (EDF)
  - Considerações suplementares
  - Exclusão Mútua no Acesso a Recursos Partilhados
  - Exemplo de Escalonamento em Computação Industrial

Algoritmos Clássicos de Escalonamento

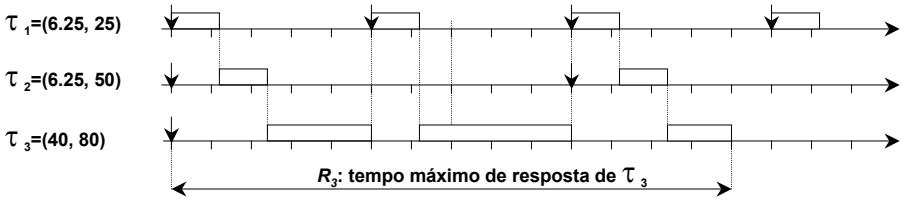
- **Análise de Escalonabilidade através do Tempo de Resposta:**
  - A análise de escalonabilidade de um conjunto de tarefas através do cálculo da Utilização apresenta grandes limitações, devido ao facto de não ser exacta e de apenas ser aplicável a modelos de tarefas muito simples.
  - Através do cálculo do Tempo de Resposta obtém-se um teste de escalonabilidade exacto:
    - » se o teste for positivo, então o conjunto de tarefas é sempre escalonável;
    - » se o teste for negativo, então algumas metas temporais serão ultrapassadas durante a execução
      - excepto se os tempos máximos de execução das tarefas forem muito pessimistas.

Algoritmos Clássicos de Escalonamento

- **Análise de Escalonabilidade através do Tempo de Resposta:**
  - A análise de escalonabilidade através do cálculo do máximo Tempo de Resposta a consideração de modelos de tarefas mais elaborados:
    - » Permite a consideração de relações de precedência e de exclusão;
    - » É válido para qualquer escalonamento dinâmico com prioridades estáticas, qualquer que seja a regra de atribuição de prioridades às tarefas.
  - Esta análise é baseada no cálculo da *máxima Interferência* que o escalonamento de uma determinada tarefa pode sofrer, devido ao escalonamento das tarefas de maior prioridade

Algoritmos Clássicos de Escalonamento

- **Cálculo do Tempo de Resposta:**
  - Tempo de Resposta da tarefa τ_i : $R_i = C_i + I_i$
  - I_i representa a interferência que o escalonamento da tarefa τ_i sofre devido ao escalonamento das tarefas de maior prioridade (I_1=0)
  - Interferência: $I_i = \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \times C_j$



Algoritmos Clássicos de Escalonamento

- **Cálculo do Tempo de Resposta:**
  - Tempo máximo de Resposta: $R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \times C_j$

    hp(i) é o conjunto de tarefas com prioridade superior à prioridade da tarefa τ_i
  - A equação é recursiva, pelo que deve ser calculada através de iterações sucessivas até que:
    - » ou o tempo de resposta da tarefa seja superior à sua meta temporal (logo a tarefa não será escalonável)
    - » ou o resultado convergir, ou seja o tempo de resposta na iteração x+1 seja igual ao tempo de resposta na iteração x.

Algoritmos Clássicos de Escalonamento

- **Cálculo do Tempo de Resposta:**
  - Cálculo do Máximo Tempo de Resposta:

    Através de uma equação recursiva, para a qual
    - » valor inicial: $w_i^0 = C_i$
    - » iterações posteriores: $w_i^{x+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^x}{T_j} \right\rceil \times C_j$
  - O conjunto de valores $w_i^0, w_i^1, w_i^2, \ldots w_i^n, \ldots$ é monotonamente não decrescente.
  - Quando $w_i^{n+1} = w_i^n$ a solução para a equação foi encontrada

Algoritmos Clássicos de Escalonamento

- **Cálculo do Tempo de Resposta:**

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \times C_j$$

» Tarefa t1: $\boxed{R_1 = C_1 = 6{,}25}$

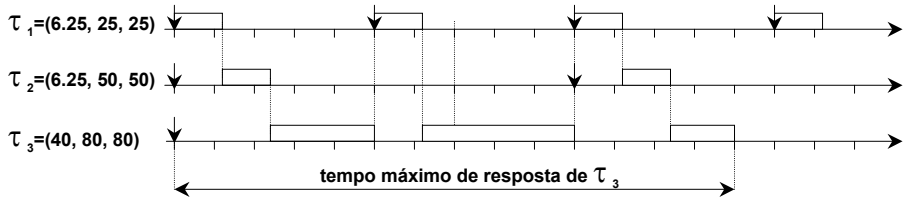
» Tarefa t2:

$$w_2^0 = 6{,}25$$

$$w_2^1 = 6{,}25 + \left\lceil \frac{6{,}25}{25} \right\rceil \times 6{,}25 = 12{,}5$$

$$w_2^2 = 6{,}25 + \left\lceil \frac{12{,}5}{25} \right\rceil \times 6{,}25 = 12{,}5$$

$$\boxed{R_2 = 12{,}5}$$



Algoritmos Clássicos de Escalonamento

- **Cálculo do Tempo de Resposta:**

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \times C_j$$

» Tarefa t3:

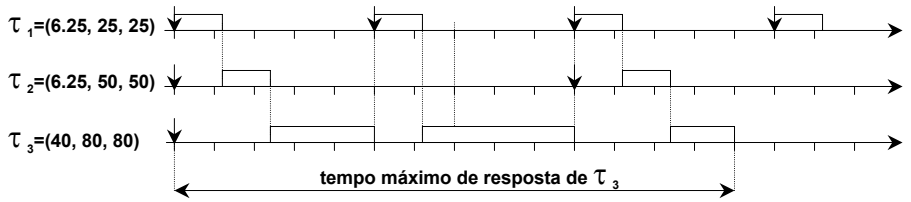
$$w_3^0 = 40$$

$$w_3^1 = 40 + \left\lceil \frac{40}{25} \right\rceil \times 6{,}25 + \left\lceil \frac{40}{50} \right\rceil \times 6{,}25 = 58{,}75$$

$$w_3^2 = 40 + \left\lceil \frac{58{,}75}{25} \right\rceil \times 6{,}25 + \left\lceil \frac{58{,}75}{50} \right\rceil \times 6{,}25 = 71{,}25$$

$$w_3^3 = 40 + \left\lceil \frac{71{,}25}{25} \right\rceil \times 6{,}25 + \left\lceil \frac{71{,}25}{50} \right\rceil \times 6{,}25 = 71{,}25$$

$$\boxed{R_3 = 71{,}25}$$



Algoritmos Clássicos de Escalonamento

- **Cálculo do Tempo de Resposta (conclusão):**
  - O conjunto de tarefas é escalonável (tempo de resposta de cada uma das tarefas inferior à sua meta temporal), apesar do teste suficiente de escalonabilidade (baseado na utilização) ser negativo.

Tarefa	*C (duração)*	*T (período)* *d (deadline)*	*R (tempo de resposta)*	*U (utilização)*
$\tau 1$	6,25	25	6,25	0,25
$\tau 2$	6,25	50	12,5	0,125
$\tau 3$	40	80	71,25	0,5
				0,875 > 0,7798

Plano das Aulas

- **Escalonamento de Tempo-Real**
  - Conceitos e Definições
  - Classificação de Algoritmos de Escalonamento
  - Algoritmos Clássicos de Escalonamento
    - » Algoritmo Rate Monotonic (RM)
    - » Cálculo do Tempo de Resposta
    - » Algoritmo Deadline Monotonic (DM)
    - » Algoritmo Earliest Deadline First (EDF)
  - Considerações suplementares
  - Exclusão Mútua no Acesso a Recursos Partilhados
  - Exemplo de Escalonamento em Computação Industrial

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Rate Monotonic" [Liu and Layland, 1973]**
  - Limitação do algoritmo RM: segundo este algoritmo, a cada tarefa é atribuída uma prioridade proporcional à sua cadência de activação.
    - » No entanto, a importância de uma tarefa pode ser independente da sua cadência de activação (por ex. leitura de temperatura);
    - » Existem outros parâmetros temporais que podem ser considerados.

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Deadline Monotonic" [Leung and Whitehead, 1982]**
  - Algoritmo de atribuição de prioridades a um conjunto de tarefas periódicas, independentes e com metas temporais menores ou iguais ao respectivo período ($d_i \leq T_i$):
  - A atribuição de prioridades às tarefas é efectuada na ordem inversa do valor da sua meta temporal: $d_i < d_j \Rightarrow P_i > P_j$
    - » desde a tarefa com menor meta temporal à qual é atribuída a maior prioridade, até à tarefa de maior meta temporal à qual é atribuída a menor prioridade;
    - » as situações de empate serão resolvidas arbitrariamente;
  - Trata-se de um algoritmo óptimo para sistemas mono-processador.

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Deadline Monotonic" [Leung and Whitehead, 1982]**
  - Vantagens do algoritmo DM
    - » Simples e adequado para utilização em sistemas operativos existentes;
    - » Pode ser utilizado para a atribuição de prioridades a níveis de interrupção;
    - » Admite tarefas com metas temporais inferiores ao período.
  - Desvantagens do algoritmo DM
    - » Modelo de tarefas também muito limitado;
    - » Não suporta exclusão mútua no acesso a recursos partilhados.

Universidade do Porto
Faculdade de Engenharia

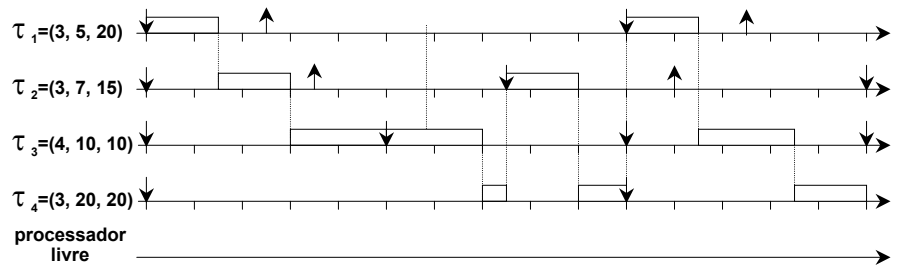
Algoritmos Clássicos de Escalonamento

- **Exemplos de escalonamento por prioridades fixas :**
  - Apresentam-se 2 cenários de escalonamento por prioridades fixas (idêntico conjunto de tarefas), considerando:
    - » prioridades atribuídas segundo o algoritmo DM (tarefas ordenadas por valor de meta temporal crescente):
      - calcula-se do tempo de resposta de cada uma das tarefas;
      - verifica-se que o conjunto de tarefas é sempre escalonável;
    - » prioridades atribuídas segundo o algoritmo RM (tarefas ordenadas por periodicidade crescente):
      - verifica-se que o conjunto de tarefas não é escalonável;

Algoritmos Clássicos de Escalonamento

- **1º Exemplo:**
  - » conjunto de tarefas ordenado por metas temporais crescentes

Tarefa	T (período)	C (duração)	d (m. temporal)	P (prioridade)	U (utilização)
τ1	20	3	5	1	0,15
τ2	15	3	7	2	0,20
τ3	10	4	10	3	0,40
τ4	20	3	20	4	0,15
					0,90



Algoritmos Clássicos de Escalonamento

- **1º Exemplo (cálculo do tempo de resposta):**

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \times C_j$$

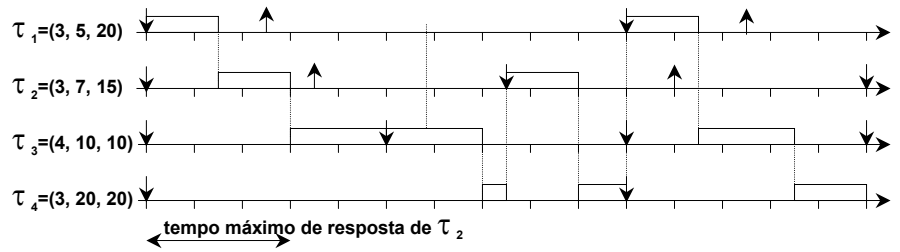
  - » Tarefa t1:
  - » Tarefa t2:

$$\boxed{R_1 = C_1 = 3}$$

$$w_2^0 = 3$$

$$w_2^1 = 3 + \left\lceil \frac{3}{20} \right\rceil \times 3 = 6$$

$$w_2^2 = 3 + \left\lceil \frac{6}{20} \right\rceil \times 3 = 6 \qquad \boxed{R_2 = 6}$$



Algoritmos Clássicos de Escalonamento

- **1º Exemplo (cálculo do tempo de resposta):**

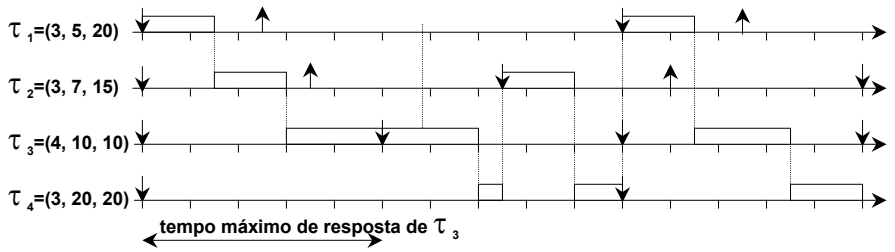
$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \times C_j$$

» Tarefa t3:

$$w_3^0 = 4$$

$$w_3^1 = 4 + \left\lceil \frac{4}{20} \right\rceil \times 3 + \left\lceil \frac{4}{15} \right\rceil \times 3 = 10$$

$$w_3^2 = 4 + \left\lceil \frac{10}{20} \right\rceil \times 3 + \left\lceil \frac{10}{15} \right\rceil \times 3 = 10 \qquad \boxed{R_3 = 10}$$



Algoritmos Clássicos de Escalonamento

- **1º Exemplo (cálculo do tempo de resposta):**

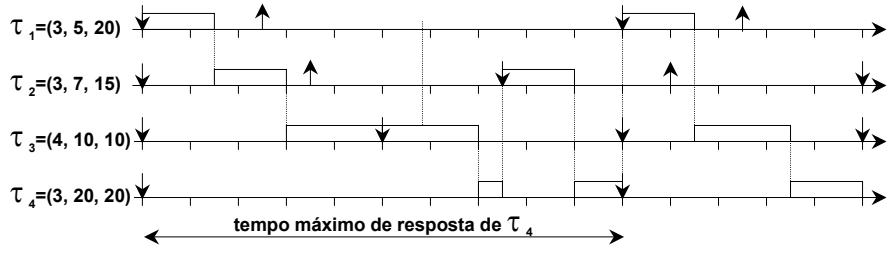
$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \times C_j$$

» Tarefa t4:

$$w_4^0 = 3 \qquad w_4^1 = 3 + \left\lceil \frac{3}{20} \right\rceil \times 3 + \left\lceil \frac{3}{15} \right\rceil \times 3 + \left\lceil \frac{3}{10} \right\rceil \times 4 = 13 \qquad w_4^3 = 3 + \left\lceil \frac{17}{20} \right\rceil \times 3 + \left\lceil \frac{17}{15} \right\rceil \times 3 + \left\lceil \frac{17}{10} \right\rceil \times 4 = 20$$

$$w_4^2 = 3 + \left\lceil \frac{13}{20} \right\rceil \times 3 + \left\lceil \frac{13}{15} \right\rceil \times 3 + \left\lceil \frac{13}{10} \right\rceil \times 4 = 17 \qquad w_4^4 = 3 + \left\lceil \frac{20}{20} \right\rceil \times 3 + \left\lceil \frac{20}{15} \right\rceil \times 3 + \left\lceil \frac{20}{10} \right\rceil \times 4 = 20$$

$\boxed{R_3 = 20}$



Algoritmos Clássicos de Escalonamento

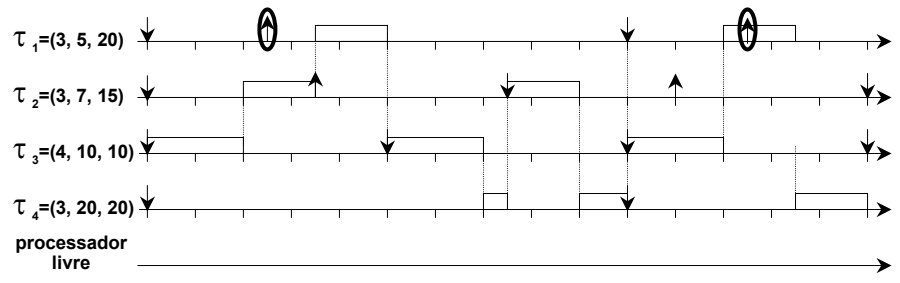
- **1º Exemplo (conclusão):**
  - O conjunto de tarefas é sempre escalonável (tempo de resposta de cada uma das tarefas inferior ao valor da sua meta temporal).

Tarefa	*T (período)*	*C (duração)*	*d (m. temporal)*	*R (tempo de resposta)*
τ1	20	3	5	3
τ2	15	3	7	6
τ3	10	4	10	10
τ4	20	3	20	20

Algoritmos Clássicos de Escalonamento

- **2º Exemplo:**
  - » conjunto de tarefas ordenado por períodos crescentes

Tarefa	*T (período)*	*C (duração)*	*d (m. temporal)*	*P (prioridade)*	*U (utilização)*
τ1	20	3	5	3	0,15
τ2	15	3	7	2	0,20
τ3	10	4	10	1	0,40
τ4	20	3	20	4	0,15
					0,90



Algoritmos Clássicos de Escalonamento

- **2º Exemplo (conclusão):**
  - Por simples inspecção da figura anterior, verifica-se que o conjunto de tarefas não é escalonável quando se utiliza o algoritmo RM;
  - O algoritmo RM é um algoritmo que não é óptimo quando se consideram conjuntos de tarefas com metas temporais inferiores aos períodos;
    - » Para este tipo de conjunto de tarefas (d<T), a atribuição dos níveis de prioridade deverá ser efectuada utilizando o algoritmo DM (algoritmo óptimo).

Plano das Aulas

- **Escalonamento de Tempo-Real**
  - Conceitos e Definições
  - Classificação de Algoritmos de Escalonamento
  - Algoritmos Clássicos de Escalonamento
    - » Algoritmo Rate Monotonic (RM)
    - » Cálculo do Tempo de Resposta
    - » Algoritmo Deadline Monotonic (DM)
    - » Algoritmo Earliest Deadline First (EDF)
  - Considerações suplementares
  - Exclusão Mútua no Acesso a Recursos Partilhados
  - Exemplo de Escalonamento em Computação Industrial

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Earliest Deadline First" [Liu and Layland, 1973]**
  - Algoritmo de atribuição dinâmica de prioridades a um conjunto de tarefas periódicas, independentes (sem restrições de precedência) e com metas temporais iguais ao respectivo período ($d_i = T_i$);
  - Trata-se de um algoritmo óptimo para sistemas mono-processador, no sentido que se existir um algoritmo capaz de escalonar um conjunto de tarefas periódicas, independentes e com metas temporais iguais ao respectivo período, então o algoritmo EDF também é capaz de o escalonar.

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Earliest Deadline First" [Liu and Layland, 1973]**
  - A atribuição dinâmica de prioridades às tarefas é efectuada na ordem inversa da distância, em cada momento, à meta temporal :
    - » no momento de activação de uma tarefa, ser-lhe-à atribuída uma prioridade tanto maior quanto menor a sua distância à meta temporal (relativamente ao estado de todas as tarefas pendentes no momento);
    - » sempre que uma nova tarefa é activada, a fila de tarefas pendentes deverá ser reordenada em função da prioridade da tarefa activada.

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Earliest Deadline First" [Liu and Layland, 1973]**
  - Teste necessário e suficiente de escalonabilidade para o caso preemptivo:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

    - » o que significa que qualquer conjunto de tarefas será escalonável pelo algoritmo EDF, desde que a utilização do processador não exceda 100%.

Algoritmos Clássicos de Escalonamento

- **Algoritmo "Earliest Deadline First" [Liu and Layland, 1973]**
  - Vantagens:
    - » algoritmo óptimo, capaz de escalonar conjuntos de tarefas com utilizações até 100%;
  - Desvantagens:
    - » maior complexidade associada à sua implementação, consequência do caracter dinâmico da atribuição de prioridades;
    - » perda de metas temporais difícil de prever para o caso de sobrecargas transitórias.

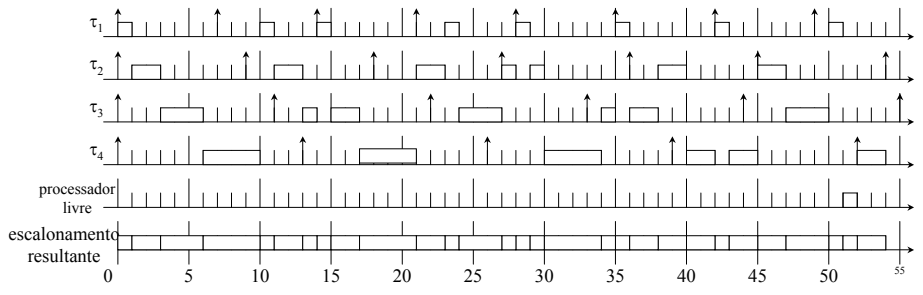
Algoritmos Clássicos de Escalonamento

- **Exemplos de escalonamento EDF vs. RM:**
  - Apresentam-se 2 cenários de escalonamento (idêntico conjunto de tarefas), considerando:
    - » prioridades dinâmicas atribuídas segundo o algoritmo EDF:
      - tarefas ordenadas (em tempo de execução) por distância à meta temporal crescente;
      - verifica-se que o conjunto de tarefas é sempre escalonável;
    - » prioridades fixas atribuídas segundo o algoritmo RM:
      - tarefas pré-ordenadas (em fase de concepção) por valor de meta temporal crescente:
      - verifica-se que o conjunto de tarefas não é escalonável;

Algoritmos Clássicos de Escalonamento

- **1º Exemplo (algoritmo EDF):**
  - activações simultâneas;
  - teste de escalonabilidade respeitado, logo o conjunto de tarefas é sempre escalonável;
  - a prioridade das tarefas varia ao longo do tempo, o que torna o sistema de difícil previsibilidade no caso de sobrecargas transitórias.

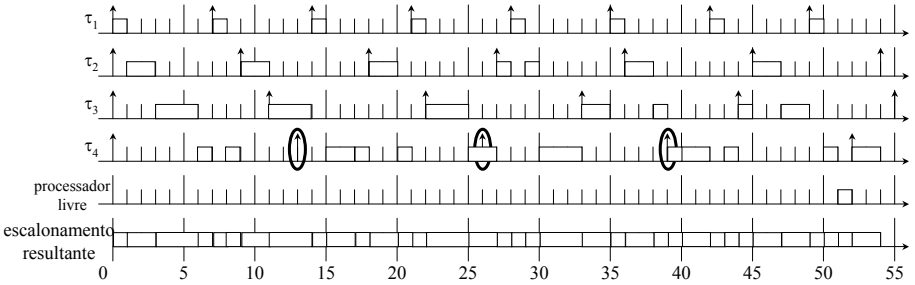
tarefa	*C*	*T*	*d*	*U*
τ_1	*1*	*7*	*7*	*0,1429*
τ_2	*2*	*9*	*9*	*0,2222*
τ_3	*3*	*11*	*11*	*0,2727*
τ_4	*4*	*13*	*13*	*0,3077*
			U total:	*0,9455*



Algoritmos Clássicos de Escalonamento

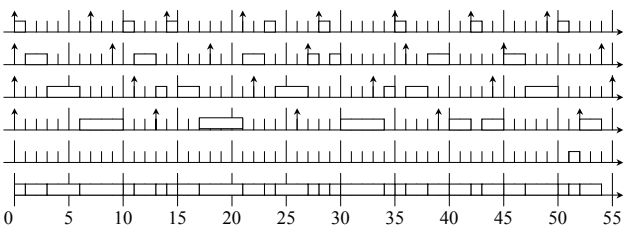
- **2º Exemplo (algoritmo RM):**
  - activações simultâneas;
  - A meta temporal da tarefa de menor prioridade após o instante crítico não é respeitada, logo o conjunto de tarefas não é escalonável;
  - no caso de sobrecargas transitórias, as tarefas que perderão as suas metas temporais serão as tarefas de menor prioridade (sistema previsível).

tarefa	*C*	*T*	*d*	*U*
τ_1	*1*	*7*	*7*	*0,1429*
τ_2	*2*	*9*	*9*	*0,2222*
τ_3	*3*	*11*	*11*	*0,2727*
τ_4	*4*	*13*	*13*	*0,3077*
			U_total:	*0,9455*

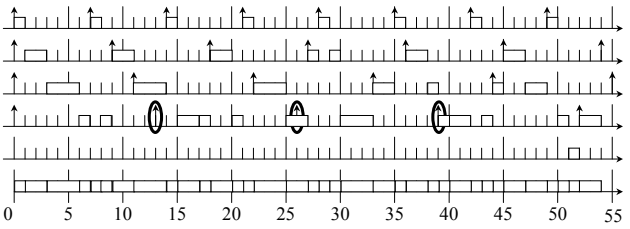


Algoritmos Clássicos de Escalonamento

» EDF



» RM



Plano das Aulas

- **Escalonamento de Tempo-Real**
  - Conceitos e Definições
  - Classificação de Algoritmos de Escalonamento
  - Algoritmos Clássicos de Escalonamento
  - Considerações suplementares
    - » Tempo Máximo de Execução de uma tarefa ("WCET")
    - » Escalonamento de Tarefas Esporádicas/Aperiódicas
    - » Utilização de Servidores
  - Exclusão Mútua no Acesso a Recursos Partilhados
  - Exemplo de Escalonamento em Computação Industrial

Considerações suplementares

- **Considerações Suplementares: Tempo Máximo de Execução ("WCET"):**
  - Pode ser obtido através da medição do tempo de execução ou através de análise do código:
    - » Caso sejam utilizadas técnicas de medição, é difícil garantir que foi observado o "tempo máximo";
    - » Caso se pretendam utilizar técnicas de análise de código, é necessário dispor de um modelo do funcionamento do processador adequado (que inclua o funcionamento de "caches", "pipelines", "wait states", etc.

Considerações suplementares

- **Tempo Máximo de Execução de uma tarefa ("WCET")**
  - A utilização de técnicas de análise de código envolve tradicionalmente duas actividades:
    1. A partir da estrutura da tarefa, decompor o seu código de alto nível num grafo orientado de blocos básicos
       - bloco básico: código linear sem ramificações;
    2. Para cada bloco básico, analisar o seu código máquina e utilizar o modelo de funcionamento do processador para determinar o seu tempo máximo de execução.
    - » Quando forem conhecidos os tempos máximos de execução para cada bloco básico, o grafo poderá ser colapsado.

Considerações suplementares

- **Tempo Máximo de Execução de uma tarefa ("WCET")**
  - Necessidade de conhecimento de informação suplementar sobre a semântica dos programas
  - Exemplo:

```
for I in 1.. 10 loop
  if Cond then
    -- basic block of cost 100
  else
    -- basic block of cost 10
  end if;
end loop;
```

    - » Tempo máximo de execução: 10*100 (+overhead) = 1005
    - » Caso `Cond` seja verdadeira só em 3 ocasiões, então o tempo máximo de execução será 375.
  - Existe uma variedade extensa de literatura sobre este assunto, nomeadamente sobre a construção adequada de compiladores e a modelação do funcionamento de processadores.

Considerações suplementares

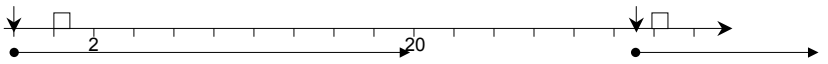
- **Tempo Máximo de Execução de uma tarefa ("WCET")**
  - A utilização de técnicas de medição deve ser baseada numa cobertura de casos adequada; A ocorrência de *casos raros* não pode ser descartada.
  - A utilização de técnicas estatísticas baseadas em Valores Extremos têm sido objecto de estudos recentes.

Plano das Aulas

- **Escalonamento de Tempo-Real**
  - Conceitos e Definições
  - Classificação de Algoritmos de Escalonamento
  - Algoritmos Clássicos de Escalonamento
  - Considerações suplementares
    - » Tempo Máximo de Execução de uma tarefa ("WCET")
    - » Escalonamento de Tarefas Esporádicas/Aperiódicas
    - » Utilização de Servidores
  - Exclusão Mútua no Acesso a Recursos Partilhados
  - Exemplo de Escalonamento em Computação Industrial

Considerações suplementares

- **Escalonamento de Tarefas Esporádicas**
  - » Tarefa Esporádica: Tarefa com um intervalo mínimo de tempo (T_i) entre 2 activações consecutivas definido;
  - » Uma tarefa esporádica (ex.: tarefa de processamento de alarmes) caracterizada por {T_i=20ms; d_i=2ms} será activada, no máximo, uma única vez num intervalo de 20ms, devendo o seu processamento ser terminado, no máximo, em 2ms;



Considerações suplementares

- **Escalonamento de Tarefas Esporádicas**
  - – Por definição, uma tarefa esporádica pode estar um longo período de tempo sem ser activada. No entanto, após a sua activação, esta deverá ser atempadamente executada. Ou seja, $d_i<T_i$.
    - » Em consequência, o algoritmo DM é particularmente adequado para o escalonamento de conjuntos de tarefas com tarefas esporádicas.
  - – De uma forma genérica, para um conjunto de tarefas com pelo menos uma tarefa esporádica, o teste de escalonabilidade através da análise do tempo de resposta será um teste suficiente.

Considerações suplementares

- **Escalonamento de Tarefas Esporádicas**
  - Procedimento para análise de escalonabilidade de conjuntos de tarefas esporádicas:
    1. todas as tarefas com metas temporais críticas deverão ser escalonáveis para os seus tempos máximos de execução e a suas taxas máximas de activação (pior caso: *abordagem resposta garantida*);
    2. todas as tarefas (críticas e não críticas) deverão ser escalonáveis quando são considerados os seus tempos médios de execução e a suas taxas médias de activação (*abordagem melhor esforço*).
       - A 1ª regra garante o respeito das metas temporais para todas as tarefas críticas;
       - Uma consequência da 2ª regra é que possivelmente nem todas as metas temporais das tarefas não críticas serão cumpridas no caso de uma sobrecarga transitória.

Universidade do Porto
Faculdade de Engenharia

Considerações suplementares

- **Escalonamento de Tarefas Esporádicas**
  - Considerando que, na maior parte dos casos, os valores de *T* (intervalo mínimo de tempo entre activações consecutivas) para as tarefas esporádicas são muito menores que os intervalos de tempo *reais* entre activações consecutivas:
    - » O cálculo de testes de escalonabilidade baseado nos valores de *T* será muito pessimista;
    - » A utilização total admissível para o sistema será muito reduzida;
    - » A utilização de *servidores* é aconselhável para efectuar o escalonamento de tarefas esporádicas.

Considerações suplementares

- **Escalonamento de Tarefas Aperiódicas**
  - » Tarefas cujo intervalo de tempo entre activações consecutivas *não tem mínimo definido*.
  - » 1ª solução: atribuir os menores valores de prioridade às tarefas aperiódicas; Consequência: difícil garantir o respeito das metas temporais
  - » Para poder ser considerada a utilização de tarefas aperiódicas em sistemas de tempo-real, torna-se necessário impor um limite superior à sua utilização de recursos computacionais através da utilização de servidores.

Plano das Aulas

- **Escalonamento de Tempo-Real**
  - Conceitos e Definições
  - Classificação de Algoritmos de Escalonamento
  - Algoritmos Clássicos de Escalonamento
  - Considerações suplementares
    - » Tempo Máximo de Execução de uma tarefa (“WCET”)
    - » Escalonamento de Tarefas Esporádicas/Aperiódicas
    - » Utilização de Servidores
  - Exclusão Mútua no Acesso a Recursos Partilhados
  - Exemplo de Escalonamento em Computação Industrial

Considerações suplementares

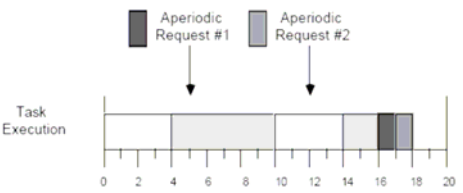
- **Utilização de Servidores para o Escalonamento de tarefas Esporádicas/Aperiódicas**
  - Técnicas básicas
    - » "*Background Service*"
    - » "*Polling*"
  - Prioridades fixas
    - » "*Deferrable Server*" [Str 95]
    - » "*Priority Exchange*" [Str 95]
    - » "*Sporadic Server*" [Spr 89]

Considerações suplementares

- **Técnicas Básicas:**

  "Background Service"
  - As tarefas aperiódicas são executadas unicamente quando o processador está ocioso.

tarefa	*C*	*T*	*d*	*U*
τ_1	4	10	10	0,4
τ_2	8	20	20	0,4
				U_total: 0,8



Fonte:
http://www.eg.bucknell.edu/~bsprunt/publications/phd_thesis/aperiodic_task_scheduling_thesis.pdf

Considerações suplementares

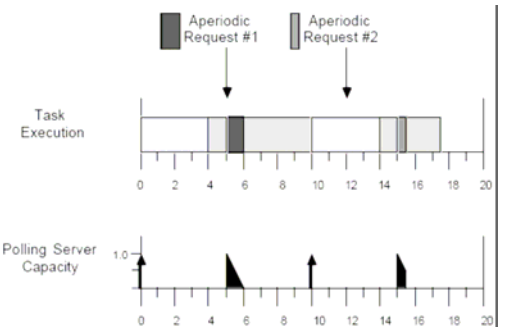
- **Técnicas Básicas: "Background Service"**
  - Vantagens
    - » Tem um impacto nulo sobre o escalonamento das tarefas periódicas (condições de escalonabilidade mantém-se);
    - » Simplicidade de implementação.
  - Desvantagens
    - » A capacidade de processamento atribuída às tarefas aperiódicas depende da carga imposta pelas tarefas periódicas;
    - » O tempo de resposta a pedidos de activação de tarefas aperiódicas pode ser muito longo.

Considerações suplementares

- **Técnicas Básicas:**

  "*Polling Service*"
  - Uma tarefa periódica extra: o servidor de "polling", é adicionada ao conjunto de tarefas a escalonar;
    - » Tarefa com capacidade C_s e periodicidade T_s
  - Durante o tempo de execução do servidor de "polling" (C_s), as tarefas aperiódicas serão executadas

tarefa	*C*	*T*	*d*	*U*
$\tau_{Polling}$	*1*	*5*	*5*	*0,2*
τ_1	*4*	*10*	*10*	*0,4*
τ_2	*8*	*20*	*20*	*0,4*
				U_total: 1



Fonte:
http://www.eg.bucknell.edu/~bsprunt/publications/phd_thesis/aperiodic_task_scheduling_thesis.pdf

Considerações suplementares

- **Técnicas Básicas: "*Polling Service*"**
  - Teste de escalonabilidade para as tarefas periódicas:
    - » Teste suficiente de escalonabilidade (RM)

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} + \frac{C_s}{T_s} \leq (n+1)\left(\sqrt[n+1]{2} - 1\right)$$

    - » Extensão a *m* servidores com prioridades diferentes para o escalonamento de tarefas aperiódicas com relevâncias diferentes

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} + \sum_{j=1}^{m} \frac{C_j}{T_j} \leq (n+m)\left(\sqrt[n+m]{2} - 1\right)$$

Universidade do Porto
Faculdade de Engenharia

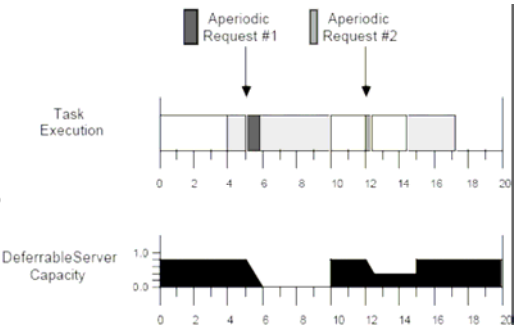
Considerações suplementares

- **Técnicas Básicas: "*Polling Service*"**
  - Vantagens:
    - » Simplicidade, quando se considera o teste de escalonabilidade para as tarefas periódicas;
    - » Fornece um melhor serviço às tarefas aperiódicas, quando comparado com o "background service".
  - Desvantagens:
    - » A capacidade do servidor é perdida, caso não existam tarefas aperiódicas com pedidos de execução activados;
    - » Não é capaz de fornecer uma resposta imediata às tarefas aperiódicas.

Considerações suplementares

- **"Deferrable Server"**
  - O algoritmo DS implementa uma tarefa de alta prioridade para servir pedidos aperiódicos.
  - O algoritmo DS mantém a sua capacidade de execução ao longo do período, enquanto a sua capacidade não se esgotar; *i.e.*, os pedidos de execução aperiódicos poderão ser executados de imediato.
  - No início do período, a capacidade do servidor é recolocada no seu valor máximo.

tarefa	*C*	*T*	*d*	*U*
τ_{DS}	0,8	5	5	0,16
τ_1	4	10	10	0,4
τ_2	8	20	20	0,4
				U_total: 0,96



Fonte:
http://www.eg.bucknell.edu/~bsprunt/publications/phd_thesis/aperiodic_task_scheduling_thesis.pdf

Considerações suplementares

- **"Deferrable Server"**
  - Teste de escalonabilidade para as tarefas periódicas:
    - » Considerando $U_s=(C_s/T_s)$, o sistema é escalonável se a condição seguinte for verificada (teste suficiente):

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n\left(\left(\frac{U_s+2}{2U_s+1}\right)^{\frac{1}{n}}-1\right)$$

Considerações suplementares

- **"Sporadic Server"**
  - O algoritmo SS implementa uma tarefa periódica (servidor) de alta prioridade para servir pedidos aperiódicos, que mantém a sua capacidade de execução até que um pedido de activação de uma tarefa aperiódica ocorra.
  - É equivalente ao algoritmo DS, excepto no que respeita os instantes em que a capacidade do servidor é recolocada no seu valor máximo.

Considerações suplementares

- **"Sporadic Server"**
  - Definições:
    - » P_s: nível de prioridade da tarefa em curso de execução;
    - » P_i: nível de prioridade de tarefa; P_1 é a maior prioridade do sistema;
    - » RT_i: Instante de recarregamento para o nível de prioridade P_i;
    - » Activo: O nível de prioridade P_i está activo se a prioridade actual do sistema, P_s, é maior ou igual que a prioridade P_i : $P_s \geq P_i$;
    - » Ocioso: O nível de prioridade P_i está ocioso ("*idle*") se $P_i < P_s$.

Considerações suplementares

- **"Sporadic Server"**
  - Para um servidor esporádico que execute a um nível de prioridade P_i,
    - » Se o servidor ainda tiver tempo de execução disponível, o seu instante de recarregamento RT_i é definido no instante t em que o nível de prioridade P_i fica activo; se a sua capacidade estiver já esgotada, o seu instante de recarregamento RT_i será definido quando a capacidade do servidor for de novo não nula e P_i estiver activo. Em ambos os casos: $RT_i = t + T_i$;
    - » O recarregamento da capacidade do servidor será igual ao tempo de execução consumido desde a ultima vez que P_i mudou de ocioso para activo.

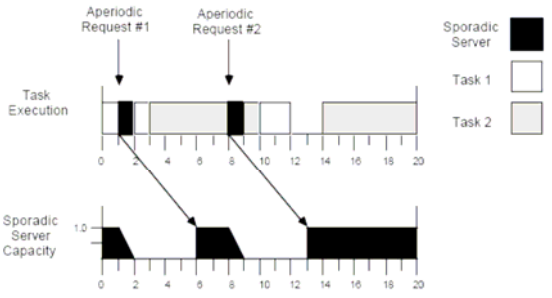
http://www.eg.bucknell.edu/~bsprunt/publications/phd_thesis/aperiodic_task_scheduling_thesis.pdf

Considerações suplementares

- **"Sporadic Server"**

  SS com o nível de prioridade mais elevado: P_1.

  - Como o SS é a única tarefa com o nível de prioridade mais elevado, P_1 só fica activo quando existe um pedido de execução de uma tarefa aperiódica. Logo, RT_1 só é definido neste instante.
  - Logo, a capacidade do servidor é reposta um período após o pedido de execução de uma tarefa aperiódica.



Task	Exec Time	Period	Utilization
SS	1	5	20.0%
τ_1	2	10	20.0%
τ_2	6	14	42.9%

Fonte:
http://www.eg.bucknell.edu/~bsprunt/publications/phd_thesis/aperiodic_task_scheduling_thesis.pdf

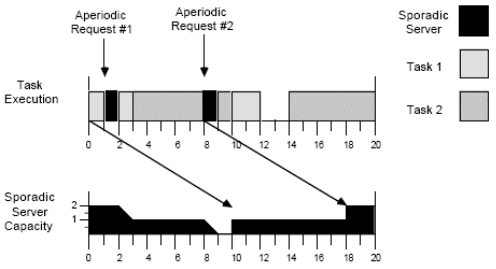
Considerações suplementares

- **"Sporadic Server"**

  Outra tarefa com o mesmo nível de prioridade do servidor (P_1).

  – No instante t = 0, τ_1 inicia a sua execução, P_1 fica activo e RT_1 é definido. No instante t = 1, o primeiro pedido aperiódico é servido.

  – No instante t = 3, τ_1 termina a sua execução, P_1 passa a ocioso, e o recarregamento no instante t = 10 é definido: 1 unidade de tempo.



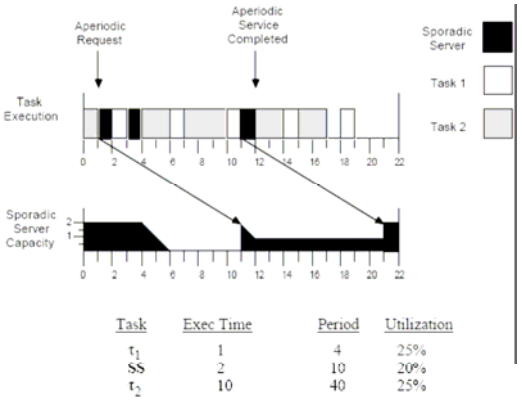
Task	Exec Time	Period	Utilization
SS	2	10	20.0%
τ_1	2	10	20.0%
τ_2	6	14	42.9%

Fonte:
http://www.eg.bucknell.edu/~bsprunt/publications/phd_thesis/aperiodic_task_scheduling_thesis.pdf

Considerações suplementares

- **"Sporadic Server"**

  – Efeito da exaustão da capacidade do servidor, devido a pedido aperiódico para um tempo de execução (3) superior à capacidade do servidor (2).



Task	Exec Time	Period	Utilization
τ_1	1	4	25%
SS	2	10	20%
τ_2	10	40	25%

Fonte:
http://www.eg.bucknell.edu/~bsprunt/publications/phd_thesis/aperiodic_task_scheduling_thesis.pdf