Escalonamento Baseado em Prioridades Fixas para Tarefas com Deadlines Nominal e Crítico

Romulo Silva Oliveira

I I - Univ. Fed. do Rio Grande do Sul
Caixa Postal 15064
Porto Alegre-RS, 91501-970, Brasil
romulo@inf.ufrgs.br

Joni da Silva Fraga

LCMI/DAS - Univ. Fed. de Santa Catarina
Caixa Postal 476
Florianopolis-SC, 88040-900, Brasil
fraga@lcmi.ufsc.br

Resumo

Requisitos de natureza temporal são tipicamente descritos na forma de deadlines. Alguns serviços podem ter simultaneamente um deadline nominal e um deadline crítico. O tempo máximo de resposta para um desempenho ótimo do sistema controlado dá origem ao deadline nominal. O deadline crítico é definido de forma a garantir a segurança do equipamento e das pessoas envolvidas no processo controlado. Este trabalho investiga o escalonamento baseado em prioridades fixas quando tarefas possuem os dois deadlines. São descritas heurísticas para a atribuição de prioridades neste modelo de tarefas. Também é mostrado que a técnica de prioridades duais pode ser usada para melhorar o comportamento do sistema. Os algoritmos propostos foram avaliados através de simulação.

Palavras-chave: Escalonamento tempo real, prioridades fixas, deadlines nominal e crítico.

1 Introdução

Sistemas computacionais de tempo real são identificados como aqueles sistemas submetidos a requisitos de natureza temporal. Nestes sistemas, os resultados devem estar corretos não somente do ponto de vista lógico, mas também devem ser gerados no momento correto.

Requisitos de natureza temporal são tipicamente descritos na forma de deadlines. As abordagens descritas na literatura para a construção de sistemas tempo real são normalmente classificadas em duas grandes categorias [14]. Sistemas tempo real do tipo *hard* são caracterizados por deadlines que devem ser sempre cumpridos. O não atendimento do requisito temporal representado por um deadline *hard* pode resultar em consequências catastróficas em termos financeiros, ambientais ou até mesmo vidas humanas. Em contrapartida, nos sistemas de tempo real *soft* os deadlines são apenas indicações de quando seria o melhor momento para concluir a tarefa. Nestes casos é aceito que o deadline não seja muitas vezes cumprido.

É possível encontrar vários textos descrevendo modelos de tarefas onde cada tarefa possui mais de um deadline. Um dos primeiros trabalhos seguindo esta abordagem pode ser encontrado em [6]. Os autores partem do conceito de que a conclusão de cada tarefa contribui para o sistema com um benefício e o valor deste benefício pode ser expresso em função do instante de conclusão da tarefa (*time-value function*). O conceito tradicional de deadline é uma simplificação desta visão mais ampla. Em [7] é apresentado como "exemplo realista" (*realistic example*) a situação

composta por um deadline nominal (*pre-deadline*) e um deadline crítico (*deadline*). O autor considera uma terminação como ótima quando a tarefa é concluída até o deadline nominal, como sub-ótima entre o deadline nominal e o deadline crítico e inaceitável após o deadline crítico.

Em [11] é feito um levantamento das soluções propostas na literatura para estimar as restrições temporais impostas pelo ambiente e também as respectivas degradações que o não cumprimento de tais restrições causam no comportamento do sistema. Fica claro que, em muitos sistemas, existe uma variação gradual do estado ideal até um estado catastrófico. A adoção de um deadline único para cada tarefa é uma simplificação arbitrária das reais restrições temporais.

Em [4] é reconhecido que alguns serviços podem ter simultaneamente um deadline *soft* e um deadline *hard*. O tempo máximo de resposta para uma reação ótima do sistema dá origem ao deadline *soft*. O deadline *hard* é definido de forma a garantir que o equipamento e as pessoas envolvidas no processo controlado não serão expostas a situações perigosas. O valor ou utilidade da resposta decresce na medida em que o tempo de resposta vai do deadline *soft* (valor máximo) até o deadline *hard* (valor mínimo aceitável).

Visão semelhante é apresentada em [12], cujo modelo proposto inclui, para cada tarefa, um deadline nominal e um deadline *hard*. Se um deadline nominal é perdido nenhuma consequência catastrófica ocorre, apenas a eficiência do controle como um todo diminui. Entretanto, o deadline *hard* está associado com um tempo de resposta no limite da operação segura do sistema. Pode-se entender que os deadlines nominal e *hard* correspondem, respectivamente, aos requisitos de desempenho (*performance*) e segurança (*safety*) do sistema controlado. Em [12] os autores destacam ainda que as margens de segurança, normalmente aplicadas em projetos de controle, reforçam a distinção entre tempo de resposta desejado e tempo de resposta aceitável. A diferença entre estes dois valores corresponde ao intervalo de tolerância do sistema (*grace time*).

Existem diversas abordagens possíveis para o escalonamento de tarefas tempo real. Uma delas é o escalonamento baseado em prioridades fixas. Um algoritmo deste tipo é o taxa monotônica (*rate monotonic*), descrito em [10] para um modelo de tarefas bastante limitado. Outro algoritmo clássico que emprega prioridades fixas é o deadline monotônico (*deadline monotonic*), introduzido em [9]. A partir destas primeiras publicações a teoria do escalonamento baseado em prioridades fixas evoluiu no sentido de suportar modelos de tarefas mais flexíveis, como pode ser visto em [1] e [13]. Por exemplo, em [5] é suposto que tarefas *hard* e *soft* compartilham o mesmo processador. É descrito um mecanismo onde cada tarefa *hard* possui duas prioridades (*dual priority scheduling*). O objetivo é minimizar o tempo médio de resposta das tarefas *soft*, enquanto é garantido o atendimento do deadline associado com cada tarefa *hard*.

O objetivo deste trabalho é investigar o emprego de algoritmos baseados em prioridades fixas para escalonar sistemas onde cada tarefa possui deadlines nominal e crítico. Nestes sistemas é absolutamente necessário garantir, em tempo de projeto, que todas as tarefas serão concluidas antes do seu deadline crítico. Simultaneamente, a conclusão de cada tarefa contribui com um valor para o sistema que é máximo quando a tarefa conclui até o deadline nominal. O valor da contribuição decresce linearmente até zero na medida que o instante de conclusão da tarefa aproxima-se do deadline crítico. A figura 1 mostra a função tempo-valor utilizada. Além da garantia em projeto para os deadlines críticos, dois outros objetivos são perseguidos: maximizar a média das contribuições das tarefas e minimizar a variância das contribuições das tarefas.

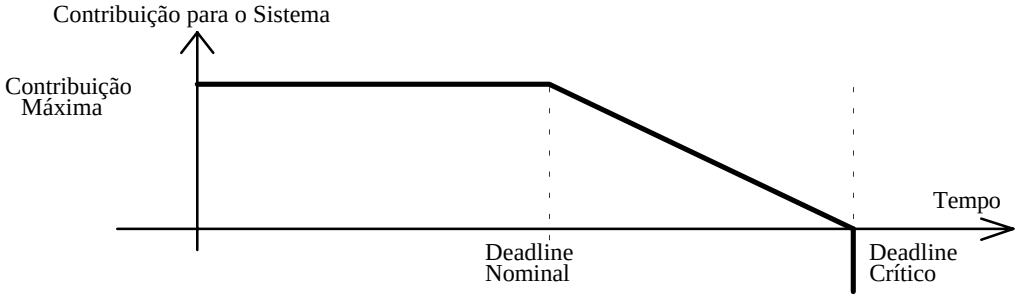

Figura 1 - Função tempo-valor adotada.

Este trabalho inicialmente investiga a atribuição de prioridades em sistemas deste tipo. É discutido o emprego de algoritmos clássicos quando tarefas possuem deadline nominal e crítico. Em seguida, são descritas heurísticas simples, com tempo de execução polinomial, para a atribuição de prioridades. Finalmente, Prioridade Dual (*dual priority scheduling*, [5]) é usada para melhorar o escalonamento no sentido de uma menor variância nas contribuições das tarefas.

É importante notar que a atribuição de prioridades fixas é feita antes de iniciar a execução do sistema (*off-line*). Isto significa que estes algoritmos não geram carga adicional (*overhead*) ao sistema tempo real. Ao mesmo tempo, a técnica de Prioridade Dual empregada demanda baixo uso de memória e processamento mínimo em tempo de execução. A filosofia deste trabalho é buscar soluções com baixo custo computacional em tempo de execução (*overhead*), para que possam ser utilizadas em uma ampla gama de aplicações, inclusive sistemas embutidos (*embedded systems*).

O restante do artigo está organizado da seguinte forma: a seção 2 descreve o modelo de tarefas adotado e o problema de escalonamento a ser resolvido. A seção 3 discute o emprego neste trabalho de algoritmos utilizados para sistemas de tarefas com um único deadline. A seção 4 descreve três heurísticas para a atribuição de prioridades no caso de deadlines nominal e crítico. Ela também descreve o mecanismo de prioridades duais e como ele pode ser usado no problema em questão. A seção 5 descreve as condições supostas nas simulações realizadas. Os resultados das experiências realizadas aparecem na seção 6. Na seção 7 são feitos os comentários finais.

2 Formulação do Problema

Neste trabalho é suposto que uma aplicação é formada por um conjunto de **n** tarefas executando em um único processador. Tarefas são executadas segundo uma política de prioridades. Qualquer tarefa pode ser *preemptada*, ou seja, ter sua execução suspensa e retomada mais tarde. As tarefas podem experimentar situações de bloqueio devido a relações de exclusão mútua entre elas. Cada tarefa T_i, onde $1 \leq i \leq n$, é caracterizada por *release jitter* máximo J_i, tempo de execução no pior caso C_i e período (tarefa periódica) ou intervalo mínimo entre liberações (tarefa esporádica) P_i.

A cada tarefa T_i estão associados dois deadlines. O deadline nominal DN_i define o final do intervalo de tempo dentro do qual a conclusão da tarefa gera uma contribuição máxima para o sistema. A partir deste instante o valor da contribuição da tarefa decresce linearmente até o seu deadline crítico DC_i, quando o valor chega a zero. Para todas as tarefas T_i, $1 \leq i \leq n$, temos que $DN_i \leq DC_i \leq P_i$. Uma tarefa T_i é dita viável se o seu tempo máximo de resposta R_i for menor ou igual ao seu deadline crítico, isto é, $R_i \leq DC_i$. Uma aplicação é dita viável quando todas as suas tarefas forem viáveis.

Supondo que o tempo de resposta da k-ésima ativação da tarefa T_i é R^k_i, a contribuição desta ativação para o sistema é denotada por V^k_i e dada por:

$V^k_i = 1$ quando $R^k_i < DN_i$,
$V^k_i = U^k_i$ quando $DN_i \le R^k_i \le DC_i$,
$V^k_i = -\infty$ quando $DC_i < R^k_i$,

onde $U_i^k = 1 - \frac{(R_i^k - DN_i)}{(DC_i - DN_i)}$ quando $DN_i \neq DC_i$ e

$U^k_i=1$ quando $DN_i=DC_i$ e, necessariamente, $DN_i=R^k_i=DC_i$.

A contribuição $V_i(t)$ da tarefa T_i até o instante **t** é dada por:

$$V_i(t) = \frac{\sum_{\forall k | T_i^k \in F_i(t)} V_i^k}{\#F_i(t)} ,$$

onde $F_i(t)$ representa o conjunto de todas as ativações da tarefa T_i concluidas até o instante **t** e $\#F_i(t)$ representa a cardinalidade do conjunto $F_i(t)$.

O valor V(t) da aplicação até o instante **t** é dado por:

$$V(t) = \frac{\sum_{i=1}^{n} V_i(t)}{n} .$$

A variância observada entre as contribuições das tarefas da aplicação será denotada por $VAR_V(t)$ e calculada por:

$$VAR_V(t) = \frac{\sum_{i=1}^{n} (V_i(t) - V(t))^2}{n} .$$

A solução de escalonamento proposta deve atender a dois objetivos:

- O objetivo primário é garantir em projeto que o tempo máximo de resposta de todas as tarefas será menor ou igual ao respectivo deadline crítico, isto é, $\forall T_i$, $R_i \le DC_i$;
- O objetivo secundário é maximizar o valor da aplicação V(t) ao mesmo tempo em que é minimizada a variância $VAR_V(t)$ entre as contribuições das tarefas.

3 Discussão do Problema

Em [1] é apresentado um método capaz de calcular o tempo máximo de resposta quando as tarefas são periódicas ou esporádicas, possuem prioridade fixa, *release jitter* máximo e situações de bloqueio. Este método independe da forma como prioridades fixas são atribuidas, desde que o deadline seja menor ou igual ao período. Em função da semelhança no modelo de tarefas, neste trabalho será usado o método descrito em [1] para calcular o tempo máximo de resposta das tarefas. Desta forma, o problema de escalonamento apresentado aqui pode ser dividido em duas etapas: investigar políticas para atribuição de prioridades fixas e investigar mecanismos para serem utilizados em tempo de execução.

Com respeito às políticas para atribuição de prioridades, é sabido que o deadline monotônico (PDM, *deadline monotonic*, [9]) é ótimo quando temos deadline único menor ou igual ao período, no sentido que: se a política PDM não conseguir satisfazer os deadlines, é impossível satisfazer com prioridades fixas. A aplicação de PDM ao problema em questão permite a definição de dois casos triviais. O primeiro caso trivial acontece quando é impossível satisfazer o objetivo primário, isto é, é impossível satisfazer os deadlines críticos das tarefas com prioridades fixas. Esta situação pode ser verificada da seguinte maneira:
- Considera o deadline crítico como único deadline, atribui prioridades seguindo o PDM;
- Calcula os tempos máximos de resposta seguindo o método em [1];
- Caso alguma tarefa possua tempo máximo de resposta maior que o seu deadline crítico, é impossível satisfazer o objetivo primário com prioridades fixas.

O segundo caso trivial acontece quando é possível garantir o deadline nominal de todas as tarefas, garantindo em projeto que o sistema apresentará sempre o seu valor máximo. Esta situação pode ser verificada da seguinte maneira:
- Considera o deadline nominal como único deadline, atribui prioridades seguindo o PDM;
- Calcula os tempos máximos de resposta seguindo o método em [1];
- Caso todas as tarefas possuam tempo máximo de resposta menor ou igual ao seu deadline nominal, o sistema possui valor máximo já garantido em projeto.

O problema mostra sua complexidade quando é possível usar PDM para garantir todos os deadlines críticos mas não é possível garantir com PDM todos os deadlines nominais. Neste caso é necessário escolher uma atribuição de prioridades que além de garantir os deadlines críticos ainda maximize o valor apresentado pelo sistema em tempo de execução.

Existe um total de **n!** diferentes atribuições de prioridades. Em [2] é descrito um algoritmo capaz de encontrar uma atribuição de prioridades que satisfaça o deadline único de cada tarefa, caso tal atribuição exista. Embora o algoritmo tenha sido criado para tarefas com um único deadline, ele foi usado como ponto de partida para a construção das heurísticas apresentadas nas seções 4.2 e 4.3 deste trabalho.

Em tempo de execução é possível empregar mecanismos para aumentar a contribuição das tarefas. Estes mecanismos baseiam-se no fato de, por definição, não existir ganho em terminar uma tarefa antes do seu deadline nominal. Existem na literatura várias propostas de servidores que poderiam, a princípio, serem utilizados neste sentido. Por exemplo, o *Deferrable Server* [8] e o *Priority Exchange Server* [8]. Entretanto, o uso de servidores implica em *overhead* adicional. O escalonamento baseado em Prioridade Dual (*dual priority scheduling*, [5]) serve ao mesmo propósito, mas apresenta um *overhead* menor. A seção 4 mostra como prioridades duais podem ser utilizadas para aumentar o valor de uma aplicação, como definido neste artigo.

4 Descrição dos Algoritmos

Nesta seção serão apresentadas heurísticas no sentido de atender os objetivos descritos antes. Inicialmente são descritas três políticas para a atribuição de prioridades. Em seguida é mostrado como o mecanismo de Prioridade Dual pode ser utilizado, em tempo de execução, no modelo de tarefas considerado. É importante observar que o mecanismo de Prioridade Dual pode ser usado em conjunto com qualquer uma das três políticas de atribuição de prioridades apresentadas.

4.1 Deadline Monotônico Usando o Deadline Crítico (PDM/DC)

Considerando o modelo de tarefas apresentado, a forma mais simples de atribuir prioridades é aplicar a política deadline monotônico utilizando apenas os deadlines críticos. Como dito antes, esta solução vai atender o objetivo primário, garantir os deadlines críticos, sempre que isto for possível. Os deadlines nominais são ignorados. Em função de sua simplicidade, esta solução será usada como linha de base para comparar o resultado das simulações.

4.2 Deadline Monotônico Modificado (PDMM)

Para obter o valor máximo da aplicação é necessário que todas as tarefas cumpram o deadline nominal. Como o deadline nominal é sempre menor ou igual ao deadline crítico, ao cumprir o deadline nominal a tarefa estará também atendendo ao deadline crítico. Logo, uma solução atraente é aplicar a política deadline monotônico utilizando os deadlines nominais.

Ocorre que utilizar os deadlines nominais ao mesmo tempo em que os deadlines críticos são ignorados pode resultar em um sistema inviável. Por exemplo, considere a aplicação composta por duas tarefas, T_1 e T_2, com as seguintes características:

Tarefa T_1:	P_1=10	J_1=0	C_1=4	DN_1=4	DC_1=8
Tarefa T_2:	P_2=10	J_2=0	C_2=2	DN_2=5	DC_2=5

Caso PDM seja aplicado sobre os deadlines nominais, a tarefa T_1 terá prioridade superior. É fácil verificar que seu tempo máximo de resposta será 4, igual ao deadline nominal. Entretanto, a tarefa T_2 terá um tempo máximo de resposta igual a 4+2=6, podendo perder o seu deadline crítico. Obviamente neste exemplo é inviável aplicar PDM usando deadlines nominais. O sistema fica viável se PDM for aplicado aos deadlines críticos. Neste caso os tempos máximos de resposta das tarefas T_1 e T_2 serão 6 e 2, respectivamente.

A solução é criar uma política que inicia aplicando PDM aos deadlines nominais. Entretanto, sempre que o tempo máximo de resposta R_i de uma tarefa T_i ultrapassa o seu deadline crítico DC_i, a sua prioridade é elevada até que $R_i \leq DC_i$. O algoritmo que implementa esta política (Deadline Monotônico Modificado) é mostrado abaixo. O objetivo é criar uma ordenação de tarefas tal que a primeira tarefa possua a prioridade mais alta e a última tarefa possua a prioridade mais baixa. Sem perda de generalidade vamos denominar T_i a tarefa que ocupa em um dado momento a i-ésima posição deste ordenamento. O tempo máximo de resposta R_i desta tarefa é calculado supondo que ela recebe interferência das tarefas que a antecedem no ordenamento.

```
Ordena as tarefas de forma que i<j ⇒ DNi≤DNj
For i := n downto 1 do
  Calcula o tempo máximo de resposta Ri
  j := i − 1
  While Ri > DCi do
    If j = 0 then
      Exit "Impossível escalonar com prioridades fixas"
    EndIf
    Troca as posições das atuais tarefas Ti e Tj
    Calcula Ri da nova tarefa que ocupa a posição i
    j := j - 1
  EndWhile
EndFor
```

É possível mostrar que a atribuição de prioridades final resultará em uma aplicação viável sempre que for possível chegar a uma aplicação viável com prioridades fixas. Da mesma forma, o algoritmo é reduzido a uma aplicação de PDM aos deadlines nominais, sempre que esta solução resultar em uma aplicação viável.

4.3 Folga Nominal Monotônica Modificada (PFNMM)

No desenvolvimento deste trabalho várias heurísticas foram tentadas. Uma heurística que gerou bons resultados foi ordenar as tarefas conforme a sua folga (*laxity*) com relação ao deadline nominal. Cada tarefa T_i é dita possuir uma folga L_i dada por $L_i = DN_i - C_i - J_i$. O algoritmo é basicamente o mesmo da heurística anterior, apenas o tempo máximo de computação e o *jitter* máximo de liberação de cada tarefa são também considerados.

```
Ordena as tarefas de forma que i<j ⇒ DNi−Ci−Ji ≤ DNj−Cj−Jj
For i := n downto 1 do
  Calcula o tempo máximo de resposta Ri
  j := i − 1
  While Ri > DCi do
    If j = 0 then
      Exit "Impossível escalonar com prioridades fixas"
    EndIf
    Troca as posições das atuais tarefas Ti e Tj
    Calcula Ri da nova tarefa que ocupa a posição i
    j := j - 1
  EndWhile
Endfor
```

Como na heurística anterior, a atribuição de prioridades final resultará em uma aplicação viável sempre que for possível chegar a uma aplicação viável com prioridades fixas. Da mesma forma, o algoritmo é reduzido a uma aplicação da política "folga nominal monotônica" (*nominal laxity monotonic*) sempre que esta solução resultar em uma aplicação viável.

4.4 Emprego de Prioridade Dual

O objetivo do mecanismo de Prioridade Dual descrito em [5] é melhorar o tempo médio de resposta de tarefas com deadline *soft*, quando executadas juntamente com tarefas *hard* cujos deadlines foram garantidos em tempo de projeto (*off-line*). As prioridades são divididas em três bandas: alta, média e baixa. Tarefas com deadline *soft* executam com prioridades da banda média. As tarefas *hard* iniciam sua execução com prioridades da banda baixa. Após um determinado intervalo de tempo, o qual é calculado em projeto, a prioridade da tarefa *hard* é promovida para um valor da banda alta. O mecanismo não especifica como as prioridades são atribuidas. Apenas o intervalo de tempo até a promoção de cada tarefa *hard* é calculado. Este cálculo supõe que, no pior caso, nenhuma tarefa *hard* consegue executar com prioridade da banda baixa em função da interferência recebida das tarefas *soft*, executadas na banda média. Assim, o intervalo de tempo até a promoção de uma tarefa *hard* é calculado de forma que ela possa ser completamente executada após a promoção mas ainda antes do seu deadline.

É possível empregar o mecanismo de Prioridade Dual no problema discutido neste artigo. Embora não existam tarefas com deadline *soft*, existem tarefas cujo deadline nominal não pode ser garantido em tempo de projeto. Como não existe vantagem em terminar uma tarefa antes do seu

deadline nominal, a idéia é transferir temporariamente prioridade das tarefas com deadline nominal garantido para as tarefas cujo tempo máximo de resposta é maior que o respectivo deadline nominal. Isto é feito sem que as tarefas com deadline nominal garantido percam esta característica.

Inicialmente as tarefas recebem sua prioridade conforme qualquer uma das heurísticas apresentadas antes. Sem perda de generalidade é suposto que elas são numeradas na ordem decrescente das prioridades. Assim, se $i < j$ então T_i tem prioridade mais alta que T_j. Em seguida as tarefas são separadas em dois conjuntos. O conjunto α é formado pelas tarefas cujo deadline nominal foi garantido em projeto, isto é, $\alpha = \{ T_i \mid R_i \leq DN_i \}$. O conjunto β é formado pelas tarefas cujo deadline nominal não foi garantido, isto é, $\beta = \{ T_i \mid R_i > DN_i \}$.

As prioridades atribuidas originalmente às tarefas formam a banda alta. Como o modelo não inclui tarefas com deadline *soft*, a banda média de prioridades não será utilizada. O objetivo é aumentar a prioridade das tarefas do conjunto β, com relação ao conjunto α, sempre que possível. Tarefas do conjunto β sempre executam com sua prioridade original, isto é, na banda alta. Tarefas do conjunto α iniciam sua execução com prioridade da banda baixa. Passado um determinado intervalo de tempo após a sua liberação, a tarefa do conjunto α é promovida para sua prioridade original, na banda alta. A ordem das prioridades na banda baixa é qualquer.

O cálculo do intervalo de tempo até a promoção de cada tarefa do conjunto α é feito como descrito em [5]. Em [3] é mostrado que, em sistemas onde todas as tarefas com deadline *hard* são periódicas, é possível realizar um cálculo mais preciso e retardar a promoção sem perder o deadline. Como o modelo de tarefas adotado neste trabalho admite a possibilidade de tarefas esporádicas, será empregado o cálculo desenvolvido em [5]. O intervalo de tempo Y_i, entre a liberação da tarefa T_i e a sua promoção, é dado por $Y_i = DN_i - R_i$, onde R_i é o tempo máximo de resposta de T_i calculado sem levar em conta o mecanismo de Prioridade Dual. O fato das tarefas do conjunto α iniciarem sua execução com uma prioridade baixa vai na verdade aumentar o seu tempo máximo de resposta. Porém, o novo tempo máximo de resposta jamais passará do seu deadline nominal, em função de como o intervalo de tempo até a promoção Y_i é calculado. Em [5] também é mostrado como as promoções de prioridade podem ser atrasadas em função do fato das tarefas esporádicas não chegarem com frequência máxima e do fato das tarefas em geral não apresentarem o seu tempo máximo de computação.

5 Simulação

Os algoritmos propostos foram simulados com o objetivo de analisar o desempenho. Foram feitas experiências usando conjuntos de tarefas gerados aleatoriamente, com as caracteristicas:

- Aplicações com um número de tarefas igual a 20, 30, 40 ou 50;
- Tarefas periódicas, com períodos P_i entre 300 e 3000 (metade das tarefas) ou entre 3000 e 30000 (a outra metade das tarefas);
- Tempo máximo de computação C_i aleatório, mas de forma que a utilização total do processador fique entre 70% e 90%;
- Release jitter máximo J_i sorteado no intervalo $[0, 0.05 \times P_i]$, com distribuição uniforme, porém limitado a um valor máximo de 15 unidades de tempo;
- Deadline crítico DC_i sorteado no intervalo $[C_i, P_i]$, com distribuição uniforme;
- Deadline nominal DN_i sorteado no intervalo $[C_i, DC_i]$, com distribuição uniforme.

Os conjuntos de tarefas inviáveis com PDM/DC foram descartados. Cada aplicação foi simulada durante 10^6 unidades de tempo. Os valores V(t) e $VAR_V(t)$ considerados são aqueles observados ao final da simulação. Cada medida apresentada é a média de 20 aplicacões diferentes. O custo computacional (*overhead*) associado com atividades de escalonamento, tais como chaveamento de contexto e promoção de prioridades, foi suposto igual a zero.

6 Resultado das Experiências

Nesta seção aparecem os resultados das simulações realizadas. Foram simuladas 6 combinações: PDM/DC, PDMM, PFNMM, PDM/DC+dual, PDMM+dual e PFNMM+dual. Em todas as experiências realizadas o valor médio V(t) da aplicação variou muito pouco com respeito a política de atribuição de prioridades adotada e ao uso ou não de prioridades duais. Em cada cenário de carga simulado V(t) não variou mais que 1% entre as 6 soluções propostas. Por outro lado, a variância $VAR_V(t)$ depende bastante da solução de escalonamento adotada.

Inicialmente foram simuladas aplicações com um número variável de tarefas mas resultando sempre em 90% de utilização do processador. A figura 2 mostra a variância $VAR_V(t)$ observada ao final da simulação para as três políticas para atribuição de prioridades descritas na seção 4. O mecanismo de Prioridade Dual não foi usado. Neste caso, a política PFNMM apresentou o melhor resultado. A figura 3 repete a mesma experiência, com a única diferença que agora o mecanismo de Prioridade Dual é empregado. Neste caso o desempenho das três políticas fica equilibrado.

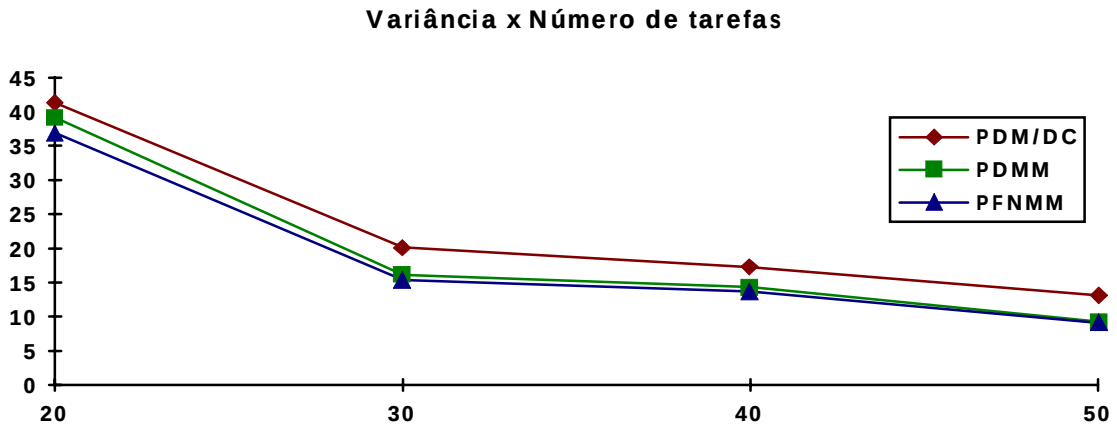

Figura 2 - 90% de utilização, número de tarefas variável, sem Prioridade Dual.

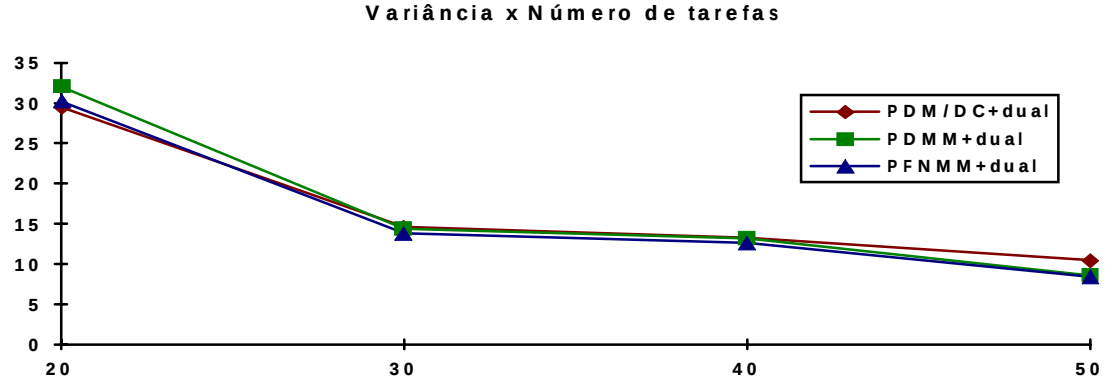

Figura 3 - 90% de utilização, número de tarefas variável, com Prioridade Dual.

A figura 4 procura identificar a importância do mecanismo de Prioridade Dual. Ela mostra a variância observada nas soluções PFNMM (melhor sem Prioridade Dual) e PFNMM+dual. A solução PDM/DC é mostrada junto para fins de comparação. Fica claro que PFNMM+dual é a melhor solução, no sentido de prover a menor variância nas contribuições das tarefas.

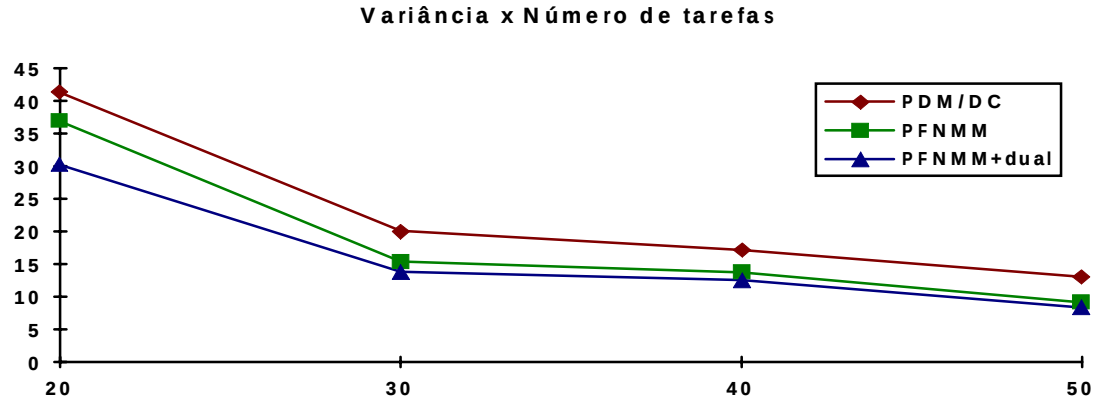

Figura 4 - 90% de utilização, número de tarefas variável.

A seguir foram realizadas simulações de aplicações sempre com 20 tarefas, porém com utilização de 70%, 80% e 90% do tempo total do processador. A figura 5 mostra a variância $VAR_V(t)$ observada ao final da simulação das três políticas para atribuição de prioridades descritas na seção 4. O mecanismo de Prioridade Dual não foi usado. Novamente, como na figura 2, a política PFNMM apresentou o melhor resultado quando Prioridade Dual não é empregada. A figura 6 repete a experiência para quando o mecanismo de Prioridade Dual está presente. Neste caso, o desempenho das três políticas fica equilibrado, como acontecera antes na figura 3.

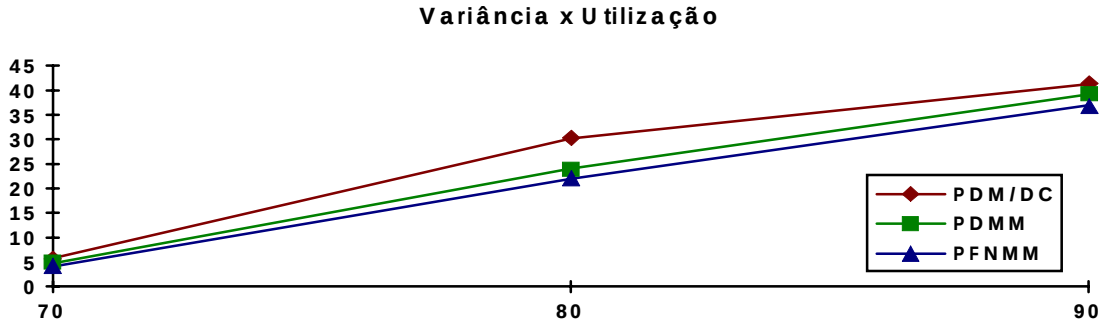

Figura 5 - 20 tarefas por aplicação, utilização variável, sem Prioridade Dual.

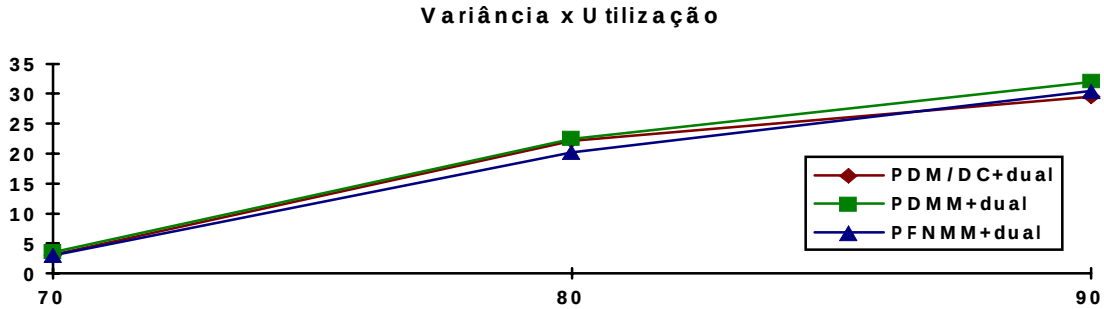

Figura 6 - 20 tarefas por aplicação, utilização variável, com Prioridade Dual.

A experiência representada pela figura 7 compara a variância observada nas soluções PFNMM (melhor sem Prioridade Dual), PFNMM+dual e a solução PDM/DC. Como fora observado antes na figura 4, a solução PFNMM+dual resulta na menor variância.

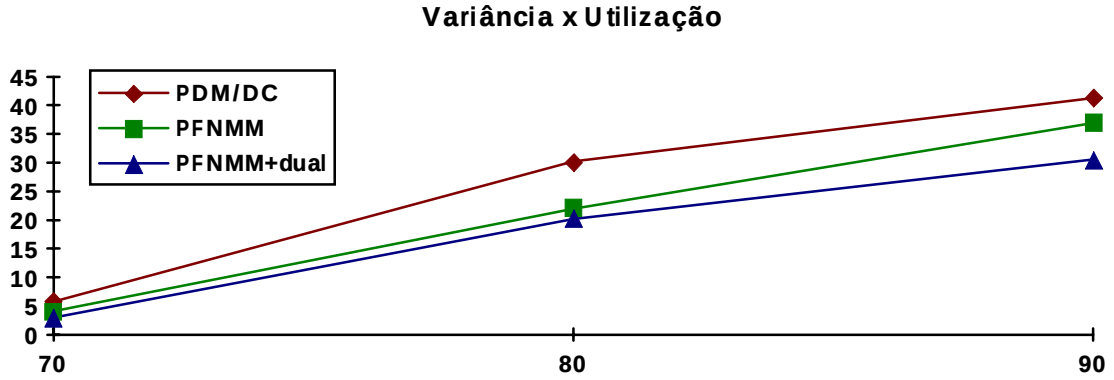

Figura 7 - 20 tarefas por aplicação, utilização variável.

As experiências permitiram concluir que, no modelo de tarefas considerado, a contribuição total das tarefas não é afetada substancialmente pela escolha da política para atribuição de prioridades nem pelo uso ou não de Prioridade Dual. Por outro lado, a escolha da política de escalonamento afeta a variância observada nas contribuições das tarefas. Quando Prioridade Dual não é usada, a política PFNMM resultou no melhor comportamento. O emprego de Prioridade Dual diminui ainda mais a variância e também a torna relativamente insensível à política para atribuição de prioridades utilizada. Os resultados permaneceram consistentes quando a utilização variou entre 70% e 90% e quando o número de tarefas por aplicação variou entre 20 e 50.

7 Conclusões

Este artigo inicialmente descreveu um modelo de tarefas onde cada tarefa possui dois deadlines. O deadline nominal é do tipo *soft*, definido em função do desempenho do sistema controlado. O deadline crítico, sempre maior ou igual ao deadline nominal, é do tipo *hard*, sendo definido em função da criticalidade do sistema controlado. É suposto no trabalho o escalonamento baseado em prioridades fixas onde é associado um valor a cada tarefa concluida; esse valor é máximo quando a mesma é concluida antes do deadline nominal. Este valor decresce linearmente até chegar a zero quando o instante de conclusão coincide com o deadline crítico.

O objetivo deste trabalho foi investigar políticas para atribuição de prioridades fixas que garantam os deadlines críticos ainda em projeto, maximizem a contribuição das tarefas e minimizem a variância destas contribuições. Foram analisadas três políticas: Deadline Monotônico Usando o Deadline Crítico (PDM/DC), Deadline Monotônico Modificado (PDMM) e Folga Nominal Monotônica Modificada (PFNMM). Além disto, foi considerado o uso do mecanismo de Prioridade Dual descrito em [5] para melhorar o escalonamento.

Foram realizadas simulações para avaliar o desempenho das soluções descritas. As experiências mostraram que, no modelo de tarefas considerado, o valor do sistema não é afetado pela escolha da política para atribuição de prioridades nem pelo uso ou não de Prioridade Dual. Para um cenário específico de carga qualquer, uma mudança na solução de escalonamento utilizada gera

alteração não maior que 1% em relação as demais soluções. Entretanto, a escolha da política para atribuição de prioridades e o uso ou não de Prioridade Dual afeta a variância entre as contribuições das tarefas. Quando Prioridade Dual não é usada, a política PFNMM resultou no melhor comportamento. O emprego de Prioridade Dual diminui ainda mais a variância e a torna insensível à política para atribuição de prioridades utilizada.

Existem na literatura de tempo real muitas indicações de que o modelo de tarefas com deadline nominal e crítico possui grande importância prática. Critérios de desempenho e criticalidade, tantas vezes presentes em aplicações reais, não podem ser corretamente expressos quando cada tarefa possui um único deadline. O estudo apresentado neste artigo aumenta o conhecimento sobre o comportamento de tais sistemas quando escalonamento com prioridades fixas é empregado.

Agradecimentos

Este trabalho foi parcialmente financiado pela FAPERGS e pelo CNPq (Protem-CC).

Referências

[1] N. C. Audsley, A. Burns, M. F. Richardson, K. Tindell, A. J. Wellings. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. Software Engineering Journal, Vol. 8, No. 5, pp.284-292, 1993.

[2] N. Audsley, K. Tindell, A. Burns. The End of the Line for Static Cyclic Scheduling? Proceedings of the Fifth Euromicro Workshop on Real-Time Systems, pp. 36-41, june 1993.

[3] G. Bernat, A. Burns. Combining (n,m)-Hard Deadlines and Dual Priority Scheduling. Proceedings of the IEEE Real-Time Systems Symposium, pp.46-57, december 1997.

[4] A. Burns, A. Wellings. Real-Time Systems and Programming Languages. Addison-Wesley, 2nd edition, 1997.

[5] R. Davis, A. Wellings. Dual Priority Scheduling. Proceedings of the IEEE Real-Time Systems Symposium, pp.100-109, december 1995.

[6] E. D. Jensen, C. D. Locke, H. Tokuda. A Time-Driven Scheduling Model for Real-Time Operating Systems. Proc. IEEE Real-Time Systems Symposium, pp.112-122, dec. 1985.

[7] E. D. Jensen. A Timeliness Model for Asynchronous Decentralized Computer Systems. Proceedings of the IEEE Real-Time Systems Symposium, pp.173-182, december 1993.

[8] J. P. Lehoczky, L. Sha, J. Strosnider. Enhancing Aperiodic Responsiveness in Hard Real-Time Environment. Proc. IEEE Real-Time Systems Symposium, pp.110-123, dec. 1987.

[9] J. Y. T. Leung, J. Whitehead. On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks. Performance Evaluation, 2 (4), pp. 237-250, december 1982.

[10] C. L. Liu, J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the ACM, Vol. 20, No. 1, pp. 46-61, january 1973.

[11] A. P. Magalhães. A Survey on Estimating the Timing Constraints of Hard Real-Time Systems. Design Automation for Embedded Systems, 1, pp. 213-230, 1996.

[12] A. P. Magalhães, M. Z. Rela, J. G. Silva. On the Nature of Deadlines. Microprocessors and Microsystems, 20, pp. 79-88, 1996.

[13] L. Sha, R. Rajkumar, S. S. Sathaye. Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems. Proceedings of the IEEE, Vol. 82, No. 1, pp.68-82, january 1994.

[14] J. Stankovic, K. Ramamritham. Advances in Real-Time Systems. IEEE Computer Society Press, 1993.