Lidando com Sobrecarga no Escalonamento de Tarefas com Restrições Temporais : A Abordagem (p+i,k)-firm

[1]Carlos Montez
montez@lcmi.ufsc.br

[1]Joni Fraga
fraga@lcmi.ufsc.br

[2]Rômulo de Oliveira
romulo@inf.ufrgs.br

[1]LCMI - Depto de Automação Sistemas - Univ. Fed. de Santa Catarina
Caixa Postal 476 - 88040-900 - Florianópolis - SC - Brasil

[2]Inst. de Informática - Univ. Fed. do Rio Grande do Sul
Caixa Postal 15064 - 91501-970 - Porto Alegre - RS - Brazil

Resumo

Sistemas de tempo real se caracterizam pela necessidade de correção temporal em suas execuções. Usualmente, esses sistemas são implementados usando tecnologias e plataformas proprietárias que aumentam seus custos de desenvolvimento e manutenção. Recentemente, tem aumentado o interesse de executar sistemas distribuídos de tempo real, usando tecnologias abertas (tais como Internet e CORBA). Essa tendência tem demandado esforços complementares tanto no sentido de se obter suportes de execução previsíveis utilizando sistemas abertos, como também no sentido de se estabelecer modelos e abordagens de tempo real mais flexíveis e que considerem a possibilidade de ocorrência de sobrecargas. Este artigo apresenta a abordagem de escalonamento adaptativo (p+i,k)-firm para aplicações distribuídas de tempo real. A abordagem lida com condições de sobrecarga, objetivando uma degradação suave, através da combinação da técnica da computação imprecisa com a garantia (m,k)-firm.

Abstract

The main characteristic in real-time systems is the timeliness requirement. Usually, these systems are implemented by using proprietary technologies and platforms, which increase its development and maintenance costs. Recently, there is an interest in executing distributed real-time systems using open technologies (such as Internet and CORBA). This trend has demanded efforts both in getting predictable execution supports using open systems, and in establishing flexible real-time models and approaches to deal with overload. This article presents the (p+i,k)-firm adaptive scheduling approach for distributed real-time applications. The approach deals with overload conditions, aiming a graceful degradation, through the combination of the imprecise computation technique with the (m, k)-firm guarantee.

***Palavras-chave**: Sistemas de tempo real, Escalonamento adaptativo*

1. Introdução

Sistemas de tempo real são formados por tarefas que possuem restrições temporais. Usualmente, são compostos na sua maior parte por tarefas periódicas que possuem prazos (deadlines) para completarem suas computações. Exemplos de tais sistemas variam desde aplicações embarcadas de controle (*e.g.*, sistemas de controle de vôo e controle anti-bloqueio de freios) até aplicações de multimídia (*e.g.*, vídeo sob demanda).

Sistemas de tempo real vêm sendo implementados usando tecnologias e plataformas proprietárias, o que aumenta seus custos de desenvolvimento e manutenção. Recentemente, existe a tendência de se desenvolver sistemas distribuídos de tempo real utilizando tecnologias abertas (*e.g.*, especificações CORBA [15] e *Internet*), com objetivo de se obter uma maior portabilidade, interoperabilidade, flexibilidade e redução de custos.

O uso de tecnologias abertas em sistemas distribuídos de tempo real é uma área de pesquisa recente. Mesmo assim, essa tendência abrange um amplo conjunto de aplicações, envolvendo desde sistemas pequenos que interagem com ambientes deterministas, até sistemas de larga escala, caracterizados por uma carga computacional dinâmica. Muitos dos domínios dessas aplicações requerem garantias de tempo real das redes, sistemas operacionais e componentes de *middleware*, no sentido de atender suas restrições temporais. Isso tem demandado um esforço das organizações padronizadoras no sentido de estender padrões abertos para oferecimento de garantias às aplicações de tempo real. Um exemplo é o *middleware* RT CORBA [16] que está sendo alvo de padronização pelo consórcio OMG.

Complementar a esse esforço, também existe a necessidade de se estabelecer modelos e abordagens adaptativos e flexíveis para permitir que aplicações de tempo real não críticas possam executar em ambientes dinâmicos [3,6,7,9,11,13,17]. Aplicações de tempo real usualmente estão sujeitas a falhas temporais, devido ao não cumprimento das restrições temporais impostas sobre essas aplicações. Dessa forma, esses modelos e abordagens precisam lidar com situações de sobrecargas transientes (onde num determinado instante os recursos disponíveis são insuficientes para o atendimento dos deadlines das tarefas liberadas), tentando minimizar a ocorrência de falhas temporais no sistema.

A abordagem de escalonamento (p+i,k)-firm descrita neste artigo, é parte de uma pesquisa mais ampla que está sendo conduzida com o objetivo de desenvolver um modelo de programação para aplicações distribuídas de tempo real usando conceitos do RT CORBA [4,13,14]. O modelo é adequado para aplicações de tempo real formadas por tarefas periódicas não críticas, tais como aplicações de multimídia e algumas de controle [6,7], que toleram algumas perdas de deadlines, desde que essas não ultrapassem um determinado patamar especificado.

A abordagem de escalonamento proposta possui característica adaptativa, pois algumas decisões de escalonamento são baseadas nas condições observadas do sistema através de monitoramentos de sobrecargas (perdas de deadline). Essa característica adaptativa é uma forma de responder às variações dinâmicas do ambiente não determinista, através do fornecimento de vários níveis de qualidade nos serviços oferecidos. A abordagem (p+i,k)-firm combina duas estratégias de escalonamento adaptativo — a computação imprecisa [11] (implementada através da técnica de invocação com polimorfismo temporal [17]) e a garantia (m,k)-firm [7].

Este artigo é dividido em 5 seções. A seção 2 discute técnicas adaptativas e relaciona alguns trabalhos nessa área. A seção 3 introduz a abordagem de escalonamento adaptativo (p+i,k)-firm. A seção 4 mostras alguns resultados dos experimentos efetuados para validar a abordagem. Finalmente, a seção 5 apresenta as conclusões e algumas observações finais.

2. Trabalhos relacionados

Uma das dificuldades encontradas em sistemas de tempo real é a necessidade de equilibrar requisitos de previsibilidade e flexibilidade. A necessidade de previsibilidade estrita em sistemas de tempo real críticos, implica geralmente em fortes restrições nos modelos empregados. No sentido de se adotar modelos mais flexíveis, abordagens de

escalonamento adaptativo (*adaptive scheduling*) têm sido propostas. Essas abordagens assumem que as condições do sistema são monitoradas, e as decisões do escalonamento são baseadas nas condições observadas [11].

Em sistemas onde podem ocorrer sobrecargas, Marucheck *et all* [12] definem o conceito de *previsibilidade na sobrecarga*, como a capacidade de garantir que durante uma sobrecarga, as tarefas terão suas restrições temporais atendidas seguindo uma ordem de importância. O grau de importância de cada tarefa pode ser determinado em tempo de projeto, ou pode assumir também uma semântica dinâmica. Por exemplo, algumas tarefas podem ter seus valores de importância alterados dinamicamente devido a um histórico recente de perdas de deadline [7,11]. Uma outra propriedade importante em sistemas de tempo real é a *configurabilidade* [12] que se refere à capacidade dos projetistas da aplicação indicarem a ordem de importância no conjunto das tarefas. As duas propriedades combinadas — *previsibilidade na sobrecarga* e *configurabilidade* — garantirão que em uma situação de sobrecarga, as tarefas menos importantes falharão antes das mais importantes.

Modelos e abordagens de escalonamento adaptativos são empregados com objetivo de lidar com a incerteza na carga do sistema e a obtenção de uma *degradação suave* [3,6,7,9,11,13,17]. Degradação suave em sistemas de tempo real, significa a manutenção das propriedades de previsibilidade na sobrecarga e configurabilidade.

Escalonamento adaptativo também pode ser usado em ambientes deterministas, onde a carga computacional é conhecida *a priori*. Recursos insuficientes em instantes críticos ou mesmo parâmetros temporais subestimados (por exemplo, os piores casos de tempo de execução) podem implicar em sobrecargas transientes e, nesse caso, escalonamento adaptativo também pode ser usado nesses ambientes como uma forma de tolerância a faltas [6].

Diversas técnicas adaptativas têm sido propostas recentemente. Por exemplo, uma técnica que permite adaptação é a baseada no ajuste das freqüências das tarefas periódicas em tempo de execução. Essa técnica pode ser usada através da mudança da freqüência de apresentação de quadros em uma aplicação de vídeo sob demanda. A adaptação das freqüências de tarefas também pode ser usada em alguns sistemas de controle realimentados [9]. Entretanto, essa técnica apresenta uma desvantagem, pois ao variar a freqüência de uma tarefa em um sistema distribuído, pode ser necessário mudar as freqüências das tarefas relacionadas no mesmo subsistema [7].

A computação imprecisa [11] — uma outra técnica que permite a combinação de garantia determinista com degradação suave — é usada para o tratamento de sobrecargas. Nessa técnica, cada tarefa é decomposta em duas partes: uma parte obrigatória e uma parte opcional. A parte obrigatória de uma tarefa deve ser completada antes do deadline da tarefa para produzir um resultado aproximado com uma qualidade aceitável. A parte opcional refina o resultado produzido pela parte obrigatória. Durante uma sobrecarga, um nível "mínimo" de operação do sistema pode ser garantido de forma determinista, através da execução apenas das partes obrigatórias. A invocação com polimorfismo temporal [17] é a técnica de computação imprecisa implementada em um ambiente distribuído, usando múltiplas versões.

Em [3], Chung *et al.* discutem o conceito de *erros cumulativos*, produzidos pelo uso da computação imprecisa. Esse tipo de erro é resultado de execuções *imprecisas*

consecutivas (somente a parte obrigatória) de uma tarefa**.** Uma heurística de escalonamento é apresentada que mantém o erro cumulativo de uma tarefa periódica abaixo de um certo valor limite. Esse objetivo é alcançado através da garantia que pelo menos uma execução *precisa* (parte obrigatória e parte opcional) da tarefa ocorra a cada janela de *k* ativações sucessivas da tarefa.

Recentemente, alguns trabalhos [1,2,7,8] estenderam o conceito convencional de *deadline firm*, permitindo que tarefas periódicas percam deadlines sem falhar. Nesses trabalhos é obrigatório para cada tarefa periódica do sistema, que se mantenha o número de deadlines perdidos abaixo de um determinado valor limite. De outra forma, uma falha temporal é assumida no sistema. Em [7] Hamdaoui *et al*. propuseram o conceito de *deadline (m,k)-firm*, definindo que uma tarefa periódica deve ter pelo menos *m* deadlines atendidos em cada janela de *k* ativações. O limite superior tolerado de perdas de deadlines é dado por *k-m.* Uma falha dinâmica é assumida no sistema quando esse limite é excedido. O DBP (*Distance Based Priority Assignment*) é a heurística de escalonamento usada com essa técnica no sentido de minimizar o número de falhas dinâmicas. Essa heurística de escalonamento atribui as mais altas prioridades para as tarefas que estão próximas de exceder o limite superior especificado para as perdas de deadlines.

Na próxima seção é introduzida nossa abordagem de escalonamento adaptativo, que compõe algumas das características da garantia (m,k)-firm com a computação imprecisa, possibilitando o desenvolvimento de aplicações adaptativas, que visam executar em ambientes dinâmicos. A abordagem proposta pode ser vista como uma combinação e uma generalização dos dois modelos citados acima. Primeiro, porque ela estende o conceito de deadline (m,k)-firm, permitindo que uma tarefa execute de forma imprecisa, obtendo melhores resultados do que a ocorrência de uma perda de deadline. Segundo, porque ela possibilita ao modelo de computação imprecisa [11] incorporar perdas de deadline, o que é natural que ocorra em sistemas com características de carga dinâmica.

3. A abordagem de escalonamento adaptativo (p+i,k)-firm

O escalonamento adaptativo proposto na abordagem *(p+i,k)-firm* usa um modelo de tarefas que permite execuções precisas e imprecisas de tarefas. Nessa técnica, em qualquer janela de *k* ativações consecutivas de uma tarefa periódica, as seguintes propriedades devem ser verificadas: (i) no máximo *k-(p+i)* deadlines podem ser perdidos; e (ii) pelo menos *p* execuções devem ser precisas dentre as ativações da tarefa que possuem seus deadlines atendidos. O limite inferior para o número de ativações cujos deadlines foram atendidos é dado por (*p+i*). O valor *i* representa o limite inferior para o número de execuções imprecisas que tiveram seus deadlines atendidos, quando apenas *p* execuções precisas são completadas em uma janela de *k* invocações.

O deadline (p+i,k)-firm é uma extensão do deadline (m,k)-firm [7] que inclui algumas propriedades da computação imprecisa descrita em [3] por Chung *et al.* Considerando uma tarefa concebida com o conceito de deadline (m,k)-firm, essa tarefa pode ser interpretada do ponto de vista do erro cumulativo. Isto é, quando a tarefa perde deadlines consecutivos, sua qualidade é degradada, acumulando erros de forma que, é necessário completar pelo menos *m* execuções em cada janela de *k* invocações para manter a taxa de erro dentro de valores aceitáveis. A extensão do deadline (m,k)-firm

com execuções precisas e imprecisas, permite uma tarefa executar na forma imprecisa, produzindo resultados aproximados e tendo um erro acumulado menor que quando um deadline é perdido. Além disso, se *p* execuções precisas em uma janela de *k* invocações garantem uma certa qualidade para a tarefa, então o escalonador terá um outro nível de flexibilidade para tratar sobrecargas. Uma execução imprecisa representa um tempo de computação menor, e o escalonador pode usá-la para controlar o número de perdas de deadlines de um conjunto de tarefas.

O conceito de falha dinâmica descrita em [7] é relacionado somente com a condição (i) apresentada acima. Em nosso modelo, esse conceito é estendido para também garantir a condição (ii): — uma falha dinâmica é assumida em uma tarefa quando o número de deadlines não atendidos supera a *k-(p+i)*, ou quando menos que *p* execuções precisas ocorrem em uma janela de *k* invocações.

A abordagem de escalonamento (p+i,k)-firm insere um nível a mais de flexibilidade do que o modelo da computação imprecisa e da garantia (m,k)-firm. É possível mostrar que os conceitos de deadline (m,k)-firm e de erro cumulativo na computação imprecisa (Chung) podem ser representados através do uso do deadline (p+i,k)-firm. A Tabela 1 mostra alguns exemplos de descrição de tarefas usando o (m,k)-firm e a técnica de Chung, e suas correspondências no deadline (p+i,k)-firm.

(m,k)-firm	*(p+i,k)-firm*
(1,2)	(1+0,2)
(2,3)	(2+0,3)
⋮	⋮
(m,k)	(m+0,k)

Chung	*(p+i,k)-firm*
Pelo menos 1 precisa a cada 2 execuções	(1+1,2)
Pelo menos 1 precisa a cada 3 execuções	(1+2,3)
⋮	⋮
Pelo menos 1 precisa a cada k execuções	(1+(k-1),k)

Tabela 1. Mapeando modelos de Chung e (m,k)-firm para o deadline (p+i,k)-firm.

3.1 Políticas de escalonamento na abordagem (p+i,k)-firm

No modelo de escalonamento proposto existe um grupo de tarefas competindo por serviço em uma CPU. Cada tarefa tem um deadline (p+i,k)-firm, e cada ativação (invocação) de tarefa possui um valor de deadline. O objetivo do escalonador do sistema é escalonar as tarefas evitando falhas (temporais) dinâmicas.

Conceitualmente, filas FIFO são definidas para a chegada de invocações de tarefas (Figura 1). A cada tarefa *j* do sistema é atribuída uma fila de chegada, assegurando que invocações de uma mesma tarefa serão servidas pelo escalonador na ordem de suas chegadas. Somente invocações de tarefas no topo de suas filas são candidatas a serem servidas, sendo inseridas em uma fila de prontos de acordo com uma *política de atribuição de prioridades*. Além dessa atribuição de prioridades, uma *política de aceitação de precisão* também é adotada. Essa última política selecionará qual versão (precisa ou imprecisa) da tarefa será executada. A política de aceitação de precisão pode também rejeitar completamente uma invocação de tarefa.

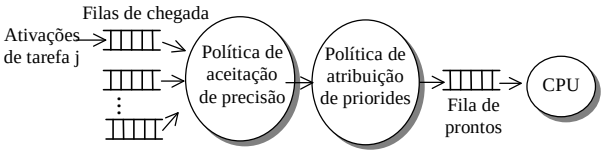

Figura 1. Modelo de escalonamento.

No modelo de escalonamento (p+i,k)-firm, as políticas de aceitação de precisão e de atribuição de prioridades trabalham de uma forma integrada, selecionando a invocação na versão precisa ou imprecisa, e atribuindo sua prioridade[1].

3.2 Tarefas com deadline (p+i,k)-firm

A abordagem de escalonamento baseia-se em um histórico de execuções precisas, imprecisa e de perdas de deadlines. Um *histórico de execução* de uma tarefa ou simplesmente *histórico* é uma k-tupla que armazena o estado das últimas *k* invocações da tarefa. Seja a seguinte representação para cada estado de invocação: *P* para execução precisa, *I* para execução imprecisa, e *X* para perda de deadline. Para cada novo estado, o histórico é deslocado (da direita para esquerda) e o novo estado adicionado. Por exemplo, em uma tarefa com deadline (2+0,4)-firm, um histórico *"PPPX"* indica que essa tarefa perdeu deadline em sua última invocação (o *X* na direita).

As seguintes variáveis também são definidas para cada tarefa *j* com deadline (p+i,k)-firm:

$p_j(k)$: número de execuções precisas nas últimas k invocações;
$i_j(k)$: número de execuções imprecisas nas últimas k invocações;
$x_j(k)$: número de deadlines perdidos nas últimas k invocações.

Onde $k = p_j(k) + i_j(k) + x_j(k)$

Por definição, uma execução normal (sem falha dinâmica) de uma tarefa ocorre quando:

$$\text{(i) } x_j(k) \leq k - (p+i) \wedge$$
$$\text{(ii) } p_j(k) \geq p$$

A condição (i) limita o número máximo de deadlines perdidos, e (ii) especifica o número mínimo aceitável de execuções precisas. Ambas as condições, (i) e (ii), não podem ser garantidas porque é assumido um ambiente não determinista. Entretanto, como no algoritmo DBP em [7], a política de atribuição de prioridades irá garantir maiores prioridades para tarefas próximas de violar a condição (i). Além disso, a política de aceitação de precisão nunca irá liberar menos que *p* execuções precisas em quaisquer janelas de *k* invocações.

Dois atributos importantes no modelo (p+i,k)-firm são *autonomia para perda de deadlines* e *autonomia para execução imprecisa*. O primeiro especifica o número de deadlines consecutivos que uma tarefa pode perder, em um dado momento, sem que ocorra uma falha dinâmica; e o segundo especifica o número de invocações imprecisas consecutivas que uma tarefa tolera. Tomemos como exemplo quatro tarefas: τ_1: (4+0,4)-firm, τ_2: (2+2,4)-firm, τ_3: (0+2,4)-firm e τ_4: (2+0,4)-firm. Considerando que em todas essas tarefas, nas últimas *k* invocações os deadlines foram atendidos na forma precisa, então:

---

[1] Na realidade, o modelo apresentado na Figura 1 é genérico. A política de atribuição de prioridades poderia ser usada, por exemplo, para implementar o escalonamento EDF (*Earliest Deadline First*) [10], onde as prioridades são atribuídas de acordo com a urgência dos deadlines das invocações. A política de aceitação de precisão poderia ser usada com a estratégia *Red Tasks Only* [8] que descarta invocações sempre que possível, uma vez que os descartes não excedam um determinado valor (um fator de *skip*).

- τ_1 e τ_2 não têm autonomia para perder deadline, mas τ_2 tem autonomia para executar as próximas 2 invocações de forma imprecisa;
- τ_2 e τ_3 têm autonomia para executar as próximas duas invocações na forma imprecisa, mas τ_2 não tem autonomia para perder deadline;
- τ_3 e τ_4 têm autonomia para perder os próximos 2 deadlines, mas τ_4 não tem autonomia para executar na forma imprecisa.

Enquanto a política de atribuição de prioridades é dirigida pelo conceito de *autonomia para perdas de deadline*, a política de aceitação de precisão é dirigida pelo conceito de *autonomia para execução imprecisa*. Em outras palavras: — "enquanto a política de atribuição de prioridades redefine prioridades tentando evitar perdas de deadlines das tarefas que têm menos folga para perder deadlines; a política de aceitação de precisão irá selecionar invocações de tarefas na forma imprecisa, em tarefas que têm mais folga para executar suas versões imprecisas".

3.3 Política de atribuição de prioridades

Seja $d_j(k)$ o valor da *autonomia de perdas de deadlines* de uma tarefa *j*, considerando suas últimas *k* ativações. A interpretação desse valor é apresentado abaixo:

$d_j(k)$	Interpretação
0	Estado de falha
1	Se perder próximo deadline então falha
⋮	
n	Se perder os próximos n deadlines então falha

É possível se obter $d_j(k)$ usando uma função similar à introduzida na heurística DBP [7]. Seja $pm_j(n,s)$ a função que denota para uma tarefa *j*, a posição (da direita para esquerda) da $n^{ésima}$ execução com deadline atendido na forma precisa ou imprecisa no histórico *s*. Se existirem menos que *n* deadlines atendidos nas últimas k ativações em *s*, então essa função retorna *k+1*. Por exemplo, $pm_j(1,\text{“XPP"}) = 1$, $pm_j(1,\text{“XPX"}) = 2$, $pm_j(2,\text{“IXP"}) = 3$, $pm_j(2,\text{“XXP"}) = 4$.

Usando essa função, o valor da autonomia de perdas de deadline pode ser calculado por:

$$d_j(k) := k - pm_j(p+i,s) + 1$$

Por exemplo, uma tarefa declarada com um deadline (2+0,4)-firm e um histórico *"PPXX"* teria: $d(k) = 4 - pm_j(2, \text{“PPXX"}) + 1 = 4 - 4 + 1 = 1$, indicando que se essa tarefa perder o próximo deadline irá falhar. Entretanto, se o histórico dessa tarefa fosse *"XPXP"*: $d_j(k) = 4 - pm_j(2, \text{“XPXP"}) + 1 = 4 - 3 + 1 = 2$, indicaria que essa tarefa poderia ainda perder o próximo deadline sem falhar.

O valor de $d_j(k)$ pode ser aplicado diretamente, através do seu mapeamento nas prioridades nativas do sistema operacional. Por exemplo, supondo que a prioridade mais alta é 0 (como ocorre em alguns sistemas operacionais), para qualquer ativação de uma tarefa *j* é suficiente fazer:

$$prioridade_j := d_j(k)$$

3.4 Política de aceitação de precisão

Seja $v_j(k)$ a função que representa a *autonomia de execução imprecisa* da tarefa *j*. A interpretação de $v_j(k)$ é apresentada na tabela abaixo:

$v_j(k)$	Interpretação
0	Estado de falha
1	Se executar uma invocação na forma imprecisa então falha
⋮	
n	Se executar n invocações na forma imprecisa então falha

O cálculo de $v_j(k)$ usa uma função $pp_j(n,s)$, que denota, para uma tarefa *j*, a posição do $n^{ésimo}$ deadline atendido com uma execução precisa no histórico *s*. Se existirem menos que *n* execuções precisas em *s*, então essa função retorna *k+1*. Por exemplo, $pp_j(1,\text{“XIP"}) = 1$, $pp_j(1,\text{”XPI"}) = 2$, $pp_j(2,\text{“PXP"}) = 3$, $pp_j(2,\text{“IXP"}) = 4$.

Usando essa função, o valor da autonomia de execução imprecisa pode ser calculado como:

$$v_j(k) := k - pp_j(p,s) + 1$$

Por exemplo, uma tarefa declarada com um deadline (2+2,4)-firm e com um histórico *"PPII"* teria: $v_j(k) = 4 - pp_j(2, \text{“PPII"}) + 1 = 4 - 4 + 1 = 1$, indicando que a próxima execução não pode ser na forma imprecisa. Entretanto, se o histórico dessa tarefa fosse *"IPIP"*: $v_j(k) = 4 - pp_j(2, \text{“IPIP"}) + 1 = 4 - 3 + 1 = 2$, indicando que a tarefa poderia ainda ter uma execução imprecisa.

A principal função da política de aceitação de precisão é decidir, em cada tarefa, a versão precisa ou imprecisa para a próxima invocação. Portanto, para cada tarefa do sistema, existe uma variável (nx_j) que armazena essa informação.

Como a abordagem de escalonamento não é *overload-aware* [12] (*i.e.*, não pode prever perdas de deadline), a política de aceitação de precisão é dirigida pelas ocorrências de sobrecarga. A cada indicação de uma perda de deadline no conjunto de tarefas do sistema, uma invocação de tarefa é selecionada para executar sua versão imprecisa (reduzindo sua qualidade), tomando como base a sua autonomia de execução imprecisa.

A política de aceitação de precisão seleciona a invocação de tarefa, no topo de uma das filas de chegada, que possui o maior valor de autonomia de execução imprecisa ($v_j(k)$) e cujo nx_j especifica versão precisa (*i.e.*, $\max(v_j(k)) \wedge nx_j$ = “precisa”). A versão imprecisa dessa tarefa é então selecionada para executar na próxima e nas invocações seguintes (nx_j := “imprecisa”), até que a situação crítica indicada pela autonomia de execução imprecisa é alcançada ($v_j(k)=1$; isto é, a tarefa não pode executar sua versão imprecisa na próxima ativação, de outra forma uma falha dinâmica ocorrerá). Quando essa última situação é alcançada, a tarefa é assinalada para executar sua versão precisa novamente (nx_j := “precisa”).

4. Avaliação da abordagem de escalonamento

A abordagem (p+i,k)-firm está sendo avaliada através de simulações que visam estabelecer o seu comportamento em situações de sobrecarga. Duas características importantes aferidas nas simulações são a de redução da probabilidade de falhas (temporais) dinâmicas, e de redução gradual nas qualidades (benefícios) obtidas de cada tarefa. A primeira característica implica que, em situações de sobrecarga, as perdas de

deadline devem ser distribuídas entre as tarefas de forma a minimizar a quantidade de falhas dinâmicas. A segunda característica implica que, dentro do possível, uma tarefa não deve ser penalizada de forma desproporcional com relação a outras. Assim, soluções tais como o descarte completo de uma tarefa (ou de quase todas suas ativações) que poderiam reduzir muito o número de falhas dinâmicas em situações de cargas altas, são consideradas ruins, pois violam essa última característica.

O simulador, desenvolvido em linguagem C, permite a configuração do número de tarefas e de suas restrições temporais. Cada tarefa recebe requisições, cujos tempos de chegadas são distribuídos exponencialmente, com uma taxa média igual a valores predeterminados. Nos experimentos, cujos resultados são apresentados nas Figuras 2 e 3, são utilizadas cinco tarefas com as mesmas restrições temporais (*i.e.*, mesmos tempos de computação e deadlines). Os deadlines das tarefas são especificados como sendo cinco vezes os tempos máximos de computação. Para a simulação da abordagem (p+i,k)-firm, os tempos de computação das versões imprecisas são calculados como sendo aproximadamente 20% dos tempos da versão precisa. Os experimentos executados consideram que as ativações de tarefas, mesmo perdendo seus deadlines, são servidas até o final.

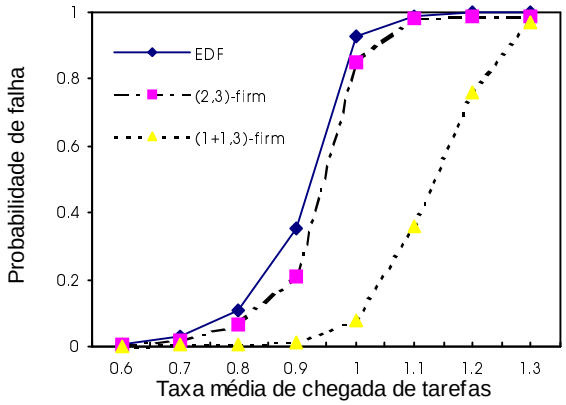

Figura 2. Comparação entre abordagens de escalonamento.

A abordagem de escalonamento (p+i,k)-firm é comparada com a DBP do (m,k)-firm, usando, respectivamente, tarefas com restrições (1+1,3)-firm e (2,3)-firm. Portanto, em ambas é tolerada a perda de um deadline a cada três ativações de tarefas. Tanto na abordagem (p+i,k)-firm quanto a do (m,k)-firm usam os valores de deadlines absolutos para desempatar requisições que possuem a mesma prioridade. A título de comparação, as duas abordagens também são confrontadas com a tradicional EDF (*Earliest Deadline First*) [10].

Os resultados mostram que a abordagem (p+i,k)-firm consegue um redução substancial na quantidade de falhas dinâmicas com relação as outras duas. O motivo é que em situações de sobrecarga, a abordagem em (p+i,k)-firm pode executar transitoriamente algumas ativações de tarefas em suas versões imprecisas, reduzindo a carga média, e conseqüentemente a quantidade de perdas de deadlines. Ou seja, a possibilidade de execuções imprecisas é mais um grau de flexibilidade que apresenta o (p+i,k)-firm e que é determinante para o melhor desempenho em relação as outras abordagens.

A redução de falhas dinâmicas no modelo (p+i,k)-firm (da mesma forma que ocorre no (m,k)-firm) também é obtida através da redução da ocorrência de perdas consecutivas de

deadline de ativações de uma mesma tarefa. A Tabela 2 apresenta o resultado de um experimento para uma taxa média de chegada de tarefas de 70%. Nessa tabela, é possível se observar que enquanto na abordagem em (p+i,k)-firm não ocorreu em nenhuma tarefa cinco perdas de deadlines consecutivas, no EDF as perdas de deadline raramente são isoladas e ocorrem geralmente em seqüência (na forma de rajadas).

	Perdas consecutivas de deadlines										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**> 10**
EDF	416	162	75	45	25	14	8	6	4	3	21
(2,3)-firm	650	209	81	31	13	5	2	0	0	0	0
(1+1,3)-firm	482	56	11	3	0	0	0	0	0	0	0

Tabela 2. Perdas consecutivas de deadline com uma taxa média de chegada de 0.70.

Um outro levantamento feito, foi com relação à qualidade média obtida das tarefas pelas abordagens de escalonamento. Nos experimentos efetuados e apresentados na Figura 3, considerou-se que as qualidades obtidas pelas execuções das tarefas são proporcionais aos seus tempos de execução. Em outras palavras, uma tarefa ao perder seu deadline oferece qualidade 0 (não oferece nenhum benefício); ao ter seu deadline atendido em uma execução precisa, oferece qualidade 1; e quando tem seu deadline atendido em uma execução imprecisa, a qualidade obtida é 0.2 (já que os tempos de computação das versões imprecisas são 20% dos tempos de computação das versões precisas).

A Figura 3 mostra que a abordagem (p+i,k)-firm consegue manter quase sempre a qualidade das tarefas em valores superiores as duas outras abordagens comparadas.

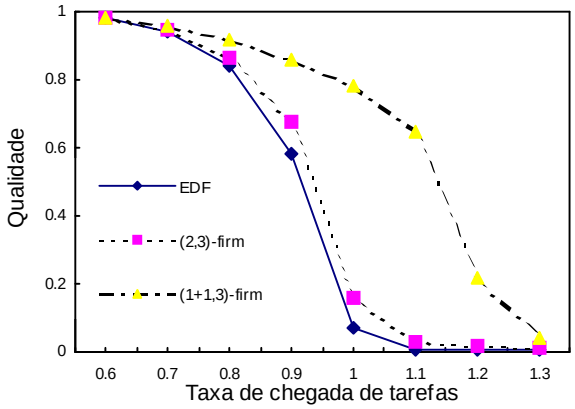

Figura 3. Comparação entre "qualidades" obtidas.

Os resultados das simulações mostram que a abordagem (p+i,k)-firm consegue simultaneamente, reduzir a quantidade de falhas dinâmicas, e elevar o nível de qualidade obtida das tarefas. Os valores levantados são altamente favoráveis a abordagem (p+i,k)-firm quando comparada com a do (m,k)-firm e a EDF. Os melhores resultados são obtidos quando a carga está próxima ou ligeiramente superior a 100%.

5. Comentários finais

Usualmente, sistemas de tempo real são implementados usando tecnologias e plataformas proprietárias que aumentam seus custos de desenvolvimento e manutenção. Recentemente, tecnologias abertas têm sido adotadas em muitas aplicações de tempo real, objetivando estender as propriedades de portabilidade, interoperabilidade, flexibilidade e redução de custos aos sistemas de tempo real contemporâneos. Essa

tendência tem demandado pesquisas em modelos e abordagens flexíveis e adaptativas que lidem com carga dinâmica e ocorrência de sobrecargas.

Em alguns trabalhos prévios envolvendo sistemas abertos, nosso grupo investigou como aplicar algumas técnicas adaptativas em aplicações de tempo real distribuídas [5,13,14], e como obter serviços de tempo global em sistemas de larga escala [4]. Atualmente, o principal esforço em nossas pesquisas é explorar a adequação das abstrações RT CORBA em aplicações não críticas de tempo real envolvendo ambientes dinâmicos.

Esse artigo apresentou a abordagem (p+i,k)-firm que pode ser vista como uma combinação e generalização da computação imprecisa e do deadline (m,k)-firm. Essa abordagem estende o conceito (m,k)-firm permitindo que uma tarefa possa executar de forma imprecisa, obtendo melhores resultados que uma simples perda de deadline. O (p+i,k)-firm, olhado sob o ponto de vista da computação imprecisa, pode ser considerado como uma extensão dessa última, onde é permitido perdas de deadline. Isso é desejável se considerarmos a aplicação da computação imprecisa em sistemas dinâmicos.

Atualmente um conjunto de testes vem sendo desenvolvido para avaliar a abordagem e sua capacidade de reduzir a probabilidade de falhas dinâmicas e obter uma degradação suave em situações de sobrecarga. Dessa forma, algumas simulações estão sendo feitas para verificar o desempenho de sistemas com deadline (p+i,k)-firm, fazendo a comparação com outras técnicas adaptativas citadas nesse artigo. Partes dos resultados das simulações foram apresentados neste texto.

Além disso, nossa abordagem de escalonamento está sendo avaliada através da programação de experimentos envolvendo a transmissão de quadros de imagens. Para isso, está se usando um protótipo de nosso modelo de programação fundamentado em conceitos RT CORBA. Todos os experimentos usam codificação JPEG para comprimir quadros de imagens. O padrão JPEG oferece taxas de compressão excelentes em imagens contínuas e simples, através da eliminação de redundâncias espaciais. O principal motivo de usar JPEG em nossos experimentos é o fato que seus decodificadores podem negociar velocidade (uso de CPU) contra qualidade de imagem, através de uso de algoritmos de aproximações rápidas, porém inexatas. Em todos os experimentos, um servidor envia quadros JPEG previamente armazenados para um cliente remoto que implementa o decodificador em duas versões — uma oferecendo melhor qualidade e usando mais CPU, e outra que oferece uma qualidade “pobre” mas tendo um tempo de execução menor. Em cada experimento, estamos modelando o sistema usando diferentes valores de deadline nas mensagens (quadros de imagens) dos servidores, e diferentes deadlines (p+i,k)-firm na implementação da decodificação no cliente. Nesses experimentos, também estamos comparando nosso modelo com o (m,k)-firm [7].

Agradecimentos

Este trabalho foi parcialmente suportado pela CAPES e CNPq. Gostaríamos de agradecer a Clovis Petry pela sua ajuda na implementação de nosso protótipo em CORBA.

6. Referências

[1] G. Bernat, A. Burns, "Combining (n m)-Hard Deadlines and Dual Priority Scheduling", *In Proc. of the 18th IEEE RTSS*, Dec. 1997.

[2] M. Caccamo, G. Butazzo, "Exploiting Skips in Periodic Tasks for Enhancing Aperiodic Responsiveness", *In Proc. of the 18th IEEE RTSS*, Dec. 1997.

[3] J. -Y. Chung, J. W. S. Liu, K. -J. Lin, "Scheduling Periodic Jobs that Allow Imprecise Results", *IEEE Transactions on Computer*, 39(9), 1990, pp.1156-1174.

[4] J. Fraga, J-M. Farines, C. Montez, "Um Serviço de Tempo Global para Sistemas Distribuídos de Larga Escala", SBRC'98, Rio de Janeiro, Brazil, May 1998.

[5] O. Furtado, F. Siqueira, J. Fraga, J-M. Farines, "A Reflective Model for Real-Time Applications in Open Distributed Systems", *Proc. IFIP/IFAC WRTP´96*, Brazil, Nov. 1996.

[6] M. Gergeleit, E. Nett, M. Mock, "Supporting Adaptive Real-Time Behavior in CORBA", *Proc. of 1st IEEE WMDRTSS*, San Francisco, CA, Dec. 1997.

[7] M. Hamdaoui, P. Ramanathan, "A Dynamic Priority Assignment Technique for Streams with Deadline (m,k)-firm", *In IEEE Trans. on Computer*, Apr. 1995.

[8] G. Koren, D. Shasha, "Skip-Over: Algorithms and Complexity for Overloaded Systems that Allow Skips", *In Proc. of the 16th IEEE RTSS*, Pisa, Italy, Dec. 1995.

[9] T. Kuo, A. K. Mok, "Incremental Reconfiguration and Load Adjustment in Adaptive Real-Time Systems", *IEEE Trans. on Computers*, Vol. 46, No. 12, Dec. 1997.

[10] C. L. Liu, J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", Journal of the ACM, 20(1), Jan. 1973.

[11] J. W. S. Liu, W. Shih, K. -J. Lin, R. Bettati, J. -Y. Chung, "Imprecise Computations", *Proceedings of IEEE*, Vol. 82, No. 1, Jan. 1994, pp. 83-94.

[12] M. J. Marucheck, J. K. Strosnider, "Some Insight into the Fault Recovery Properties of Priority-Driven Schedulers", *The Real-time Systems Journal*, Oct. 1995.

[13] C. Montez, J. Fraga, R. S. Oliveira, J-M. Farines, "An Adaptive Scheduling Approach in Real-Time CORBA", 2nd IEEE International Symposium on Object-oriented Real-time Distributed Computing - ISORC'99, Saint-Malo, France, May 1999.

[14] C. Montez, R. S. Oliveira, J. Fraga, "An Adaptive Model for Programming Distributed Real-Time Applications in CORBA", *Proc. of XVIII International Conference of Chilean Computer Science Society, SCCC'98*, Antofagasta, Chile, Oct. 1998.

[15] OMG, "The Common Object Request Broker: Architecture e Specification - Revision 2.2", Object Management Group (OMG), Feb. 1998.

[16] OMG, "Realtime CORBA - Joint Revised Submission", Object Management Group (OMG), Document orbos/98-10-05, Oct. 1998.

[17] K. Takashio, M. Tokoro, "Time Polymorphic Invocation: A Real-Time Communication Model for Distributed Systems", *In Proc. of the IEEE WPDRTS'93*, 1993.