Escalonamento de Aplicações Tempo Real em uma Arquitetura Baseada em Multicomputadores

Luis Fernando Friedrich[+], Rômulo Silva Oliveira*, Thadeu Botteri Corso[+]

[+] INE - Univ. Fed. de Santa Catarina
Caixa Postal 476
Florianopolis-SC, 88070-900, Brasil
{lff,thadeu}@inf.ufsc.br

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

Resumo

Existem aplicações tempo real cuja complexidade exige arquiteturas com alto desempenho. Este artigo analisa a dificuldade associada com o escalonamento de aplicações tempo real em arquiteturas baseadas em multicomputadores. Inicialmente a arquitetura alvo é descrita. Em seguida são listados os modos de operação possíveis com respeito a processadores e canais de comunicação. Finalmente, a dificuldade para o escalonamento tempo real de 15 diferentes cenários é discutida. Foram apontadas algumas técnicas descritas na literatura de tempo real que podem ser empregadas no contexto específico dos cenários discutidos.

Abstract

Some real-time applications are so complex that they require high-performance architectures. This paper analyses how difficult is to schedule real-time applications on multicomputer-based architectures. First the target architecture is described. We then list several possible operation modes for processors and communication channels. Finally, problems associated with the real-time scheduling of 15 different cenarios are discussed. Some techniques described in the real-time literature are pointed as possible solutions for some of the cenarios discussed.

1. Introdução

Conforme evoluem, os sistemas de tempo real tem apresentado requisitos como desempenho e confiabilidade cada vez mais rígidos. Muitos problemas de tempo real não conseguem alcançar seus requisitos sem o uso de sistemas de computação altamente cooperativos, os quais são compostos por vários pontos de processamento e fornecem características como paralelismo, escalabilidade e tolerância a falhas. Várias aplicações nas áreas de robótica, simulações em tempo real ou baseadas em inteligência artificial apresentam simultaneamente requisitos temporais e demanda por enorme capacidade de processamento.

Os multicomputadores são máquinas que apresentam um grande potencial no que diz respeito a alto desempenho e confiabilidade, e por isso tem surgido como candidatos naturais para as aplicações de tempo real. Multicomputadores designam máquinas compostas por múltiplos elementos de processamento (nós) conectados através de uma rede de interconexão na qual é possível a troca de mensagens entre os nós de uma forma rápida, permitindo uma cooperação eficiente entre os mesmos. Cada nó é constituído minimamente de um processador, memória privativa e canais de comunicação. A presença de múltiplos canais de comunicação entre os nós, através das redes de interconexão, fazem dos multicomputadores máquinas robustas no que diz respeito a conectividade e falha dos nós, e também permitem suportar transmissões simultâneas em diferentes canais de comunicação. Os multicomputadores diferem entre si

principalmente pelas características de suas redes de interconexão que podem ser classificadas em estáticas e dinâmicas. Numa rede estática as ligações entre os nós são passivas e não podem ser reconfiguradas para conexão direta com outros nós, os canais são permanentes. Por outro lado, as ligações nas redes dinâmicas podem ser reconfiguradas através de circuitos de chaveamento eletrônico cuja manipulação permite a atribuição de canais físicos temporários.

Construir aplicações tempo real como redes de processos comunicantes é similar a usar multitarefas em uniprocessador ou multiprocessador mas sem memória (real ou virtual) compartilhada. Uma aplicaçao tempo real é escrita como um conjunto de componentes (processos, tarefas) que interagem para realizar a aplicação. Os processos cooperam apenas através da passagem de mensagens através de canais lógicos. Uma rede de processos comunicantes implementando uma aplicação tempo real pode ser expressa como um grafo onde vértices representam processos e arcos representam canais de comunicação [4]. Esta rede lógica pode exibir uma topologia estática durante toda a execução ou variações em função da criação de tarefas e canais lógicos.

Este artigo tem como tema principal a investigação do problema de escalonamento tempo real em um ambiente de execução baseado em multicomputador. A motivação fundamental parte da visão dessas máquinas como um ambiente natural para a execução de aplicações de tempo real que demandam grande poder computacional. Na seção 2 é descrita a arquitetura alvo deste trabalho, baseado em um multicomputador com rede de interconexão dinâmica. A seção 3 apresenta os diversos modos de operação possíveis nesta arquitetura. Na seção 4 são discutidas as necessidades de escalonamento tempo real que surgem como consequência dos modos de operação descritos na seção anterior. Finalmente, a seção 5 contém as considerações finais.

2. Arquitetura Alvo

O modelo geral da arquitetura alvo é um multicomputador composto por um conjunto de nós interconectados por meio de uma rede principal, conforme mostrado na figura 1. Esta seção descreve rapidamente a arquitetura. Maiores detalhes podem ser encontrados em [3] e [9].

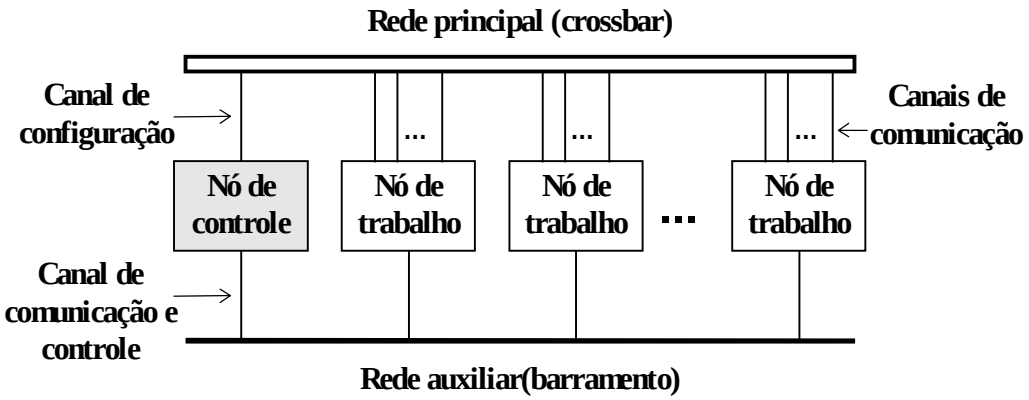

Figura 1 - Arquitetura do ambiente proposto.

O multicomputador é composto por um número expressivo de nós de trabalho, conectados através de uma rede de interconexão. Um nó de controle é responsável pela gerência da conexão/desconexão de canais físicos entre os nós de trabalho e também pela alocação/liberação dos nós de trabalho. Cada nó é constituído de um processador com memória privativa e um *hardware* para suporte à comunicação, o qual permite o estabelecimento das ligações com outros nós de trabalho e com o nó de controle, conforme a figura 2. Um número grande de canais de comunicação permite suportar transmissões simultâneas. O *hardware* de comunicação do nó de controle possui um canal para a configuração da rede principal. Os canais de comunicação são implementados a partir da utilização do adaptador de ligação IMS C011

(*link adaptor*) que fornece comunicação *full-duplex* com taxas de transmissão de 10 ou 20 Mbits por segundo. Os canais de comunicação funcionam com DMA.

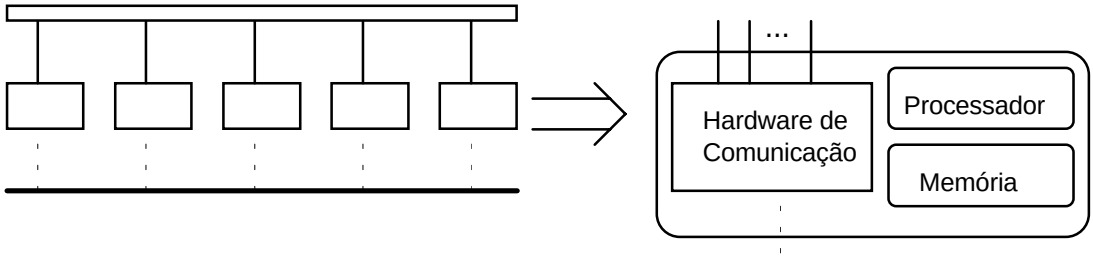

Figura 2 - Estrutura dos nós de trabalho.

A rede de interconexão, na arquitetura proposta, é baseada em um mecanismo de comunicação com atribuição de canais físicos por demanda, composta por dois níveis básicos de comunicação: rede principal (*crossbar*) e rede auxiliar (barramento). A comunicação entre os nós de trabalho é realizada através de canais físicos diretos da rede principal, que utiliza uma rede de interconexão do tipo *crossbar*, com dimensão $N \times N$, permitindo que sejam estabelecidas N ligações simultâneas e exclusivas entre pares de nós. A rede principal é configurada pelo nó de controle. A rede auxiliar é utilizada para resolver de forma simples e eficaz o problema da conexão dos canais físicos da rede principal, transportando mensagens do suporte de execução, entre os nós de trabalho e o nó de controle, com pedidos de conexão/desconexão de canais físicos da rede principal. Assim, antes da comunicação entre dois nós de trabalho ser efetivada, é verificada a existência de um canal direto entre eles através da rede principal. No caso positivo, a comunicação pode ser iniciada; caso contrário, um pedido de conexão deve preceder a comunicação propriamente dita.

A implementação do mecanismo de comunicação com atribuição de canais físicos por demanda é realizada pelo suporte de execução, com a participação de um componente central executado no nó de controle e por componentes locais executados em cada nó de trabalho. O nó de controle, através do componente central, é responsável pela operação da rede principal, executando a conexão/desconexão de canais físicos em resposta a pedidos originados dos nós de trabalho.

3. Modos de Operação

Os dois principais recursos a serem escalonados no sistema são o processador em cada nó de trabalho e as conexões do *crossbar*. É suposto que as tarefas foram previamente alocadas aos processadores segundo algum algoritmo, como [2] ou [7]. Outros recursos possíveis são estruturas de dados compartilhadas por diferentes fluxos de execução em um mesmo nó e periféricos cujos controladores estão conectados aos nós através de outros barramentos que não o *crossbar*.

O escalonamento de tarefas na presença de estruturas de dados compartilhadas é objeto de intensa pesquisa. Existe farta literatura a este respeito, incluindo soluções baseadas em executivo cíclico [8], *priority ceiling* [6] e sincronização *lock-free* [1]. Recursos físicos, como periféricos acoplados a um processador, podem receber tratamento similar. Este artigo limita a discussão do escalonamento tempo real na máquina alvo aos recursos processador local e *crossbar*, por serem os elementos chave da arquitetura.

Nesta seção vamos listar os diferentes modos de operação possíveis da máquina, o que resulta em diferentes formas de utilização e compartilhamento dos processadores e do *crossbar*. Será utilizada a terminologia da área de tempo real. O termo tarefa designa um fluxo de execução que deve ser escalonado. Cada nó de trabalho executa uma ou mais tarefas, as quais pertencem a

uma única aplicação ou ao suporte de execução. Também é suposto que as tarefas de aplicação presentes em um mesmo nó de trabalho compartilham a memória.

Com respeito aos processadores, podemos identificar 3 modos principais de operação:

MONO - Cada processador executa uma única tarefa (fluxo de execução) da aplicação. Quando esta tarefa não está pronta para execução o processador não é utilizado pela aplicação.

MULT1 - Cada processador executa uma ou mais tarefas. Estas tarefas comunicam-se entre si através do acesso às estruturas de dados compartilhadas. Tarefas de um nó podem comunicar-se com tarefas de um outro nó através do *crossbar*. Para cada destino possível de comunicação inter-processador, somente uma tarefa é responsável pelo envio de mensagens.

MULT2 - Novamente, cada processador executa uma ou mais tarefas, que comunicam-se entre si através de estruturas de dados locais. Tarefas de um nó podem comunicar-se com tarefas de um outro nó através do *crossbar*. Entretanto, para cada destino possível de comunicação inter-processador, várias tarefas locais podem solicitar o envio de mensagens.

Com respeito a utilização do *crossbar*, são identificados 5 modos principais de operação:

ESTAT - O *crossbar* possui uma configuração estática durante a execução da aplicação. Esta configuração é calculada em tempo de projeto (*off-line*) de maneira que todas as necessidades de comunicação inter-processador sejam satisfeitas por conexão física direta. Em tempo de execução não ocorre chaveamentos no *crossbar* e o barramento auxiliar não é mais necessário. O nó de controle é utilizado somente durante a inicialização da aplicação para estabelecer a configuração estática calculada antes. É importante observar que nem sempre este modo de operação é viável, pois pode ser impossível encontrar uma configuração estática que satisfaça todas as necessidades de comunicação da aplicação.

ESTROT - Novamente o *crossbar* possui uma configuração estática, calculada antes da execução e estabelecida durante a inicialização da aplicação. Entretanto, algumas comunicações inter-processador são indiretas, ou seja, passam por uma tarefa executando em processador intermediário que atua como roteador. Este modo de operação permite uma configuração estática para o *crossbar* mesmo quando não é possível estabelecer uma conexão física direta entre todos os pares de processadores comunicantes. Assim, algumas conexões lógicas compartilham a mesma conexão física. Neste modo de operação é esperado que a grande maioria das conexões físicas sejam dedicadas e que as conexões indiretas passem por no máximo um nó intermediário.

DINDEM - Neste modo o *crossbar* possui uma configuração dinâmica. Sempre que uma tarefa necessita enviar uma mensagem para outro nó de trabalho ela antes envia uma mensagem via barramento de controle para o nó de controle. Após a conexão física estar estabelecida o nó de controle avisa a tarefa através do barramento de controle. Toda conexão lógica é mapeada diretamente em uma conexão física, isto é, não existe roteamento. Quando a tarefa termina de enviar a mensagem ela avisa ao nó de controle através do barramento de controle, liberando a conexão física. O tempo necessário para estabelecer tais conexões e os possíveis conflitos entre diferentes solicitações afetam o tempo de execução das tarefas da aplicação. Este é o modo de operação usual para aplicações sem requisitos de tempo real.

DINSINC - Este modo de operação também configura dinamicamente o *crossbar*. Entretanto, o nó de controle não atua em função das solicitações das tarefas via barramento de controle. É construida, em tempo de projeto, uma escala de configuração para o *crossbar*. Esta escala é cíclica e determina, a cada momento, quais conexões físicas serão estabelecidas. Desta forma, para uma tarefa enviar uma mensagem inter-processador, ela deve esperar até o próximo intervalo de tempo no qual a conexão física desejada está disponível. Este modo de operação exige sincronização de relógios entre todos os nós, a qual pode ser feita através do barramento de controle.

DINANUN - É possível eliminar a necessidade de sincronização de relógios entre os nós se o nó de controle anunciar, via o barramento de controle, quando uma conexão física é ligada e desligada. Como no modo de operação anterior, uma escala de configuração para o *crossbar* é calculada em projeto. Esta escala determina quando cada conexão física é ligada e desligada. Ela é executada pelo nó de controle, baseado em seu relógio local. A cada momento cada nó de trabalho sabe quais são as suas conexões físicas disponíveis através das mensagens difundidas no barramento de controle. As tarefas consultam estas informações para determinar o instante de enviar uma mensagem inter-processador.

4. Necessidades de Escalonamento

A seção anterior listou 3 diferentes formas de operação para os processadores e 5 formas diferentes de operação para o *crossbar*. Nos próximos parágrafos é discutida a dificuldade para analisar a escalonabilidade do sistema em cada um destes 15 cenários.

MONO + ESTAT

Neste cenário não existe a necessidade de escalonamento. Cada tarefa da aplicação tem a sua disposição um processador e todas as comunicações inter-processador sempre encontram uma conexão física já estabelecida e disponível.

MULTI1 + ESTAT

Como cada nó de trabalho executa várias tarefas, existe a necessidade de escalonamento do processador. O fato do *crossbar* possuir uma configuração estática e de cada conexão física possuir apenas uma tarefa usuária significa que a comunicação não possui escalonamento. Neste cenário a máquina tem o comportamento similar a um sistema distribuído onde o tempo máximo para comunicação entre dois nós é conhecido.

MULTI2 + ESTAT

Como no cenário anterior existe a necessidade de escalonar o processador. Agora várias tarefas disputam a mesma conexão física. Desta forma, também é necessário um algoritmo para decidir qual tarefa usa a conexão antes. O tempo para concluir uma tarefa depende simultaneamente dos algoritmos utilizados para escalonar o processador e os canais de comunicação.

MONO + ESTROT

Uma conexão indireta exige a presença de uma tarefa roteadora no nó de trabalho intermediário. Logo, o processador do nó intermediário é compartilhado entre sua tarefa local e a tarefa roteadora. Como um mesmo nó de trabalho pode, a princípio, servir como nó intermediário para várias conexões indiretas, diversas tarefas podem disputar o seu processador. Além disto, as suas conexões físicas são agora compartilhadas entre as suas próprias mensagens e as mensagens de outras tarefas que são roteadas localmente. Logo, várias tarefas podem disputar o uso de uma mesma conexão física: uma tarefa da aplicação e várias tarefas roteadores. O resultado é que ao usar o modo de operação ESTROT, o comportamento do processador passa de MONO para MULTI2.

MULTI1 + ESTROT

Como no cenário anterior, o processador de um nó de trabalho deve ser compartilhado entre as tarefas da aplicação e as tarefas roteadoras presentes no mesmo nó. Além disto, cada conexão física deve ser compartilhada entre a única tarefa local de aplicação usuária da conexão e as possivelmente várias tarefas roteadoras que usam este mesmo canal. Logo, esta combinação de modos de operação também resulta em comportamento semelhante a MULTI2 + ESTROT.

MULTI2 + ESTROT

Cada conexão indireta pode ser transformada em uma sequência de duas conexões diretas, acompanhadas de uma tarefa roteadora no nó intermediário. Uma vez feita esta

transformação, temos um cenário similar ao MULTI2+ESTAT, pois as conexões são estáticas, cada processador possui várias tarefas para executar e várias tarefas podem disputar a mesma conexão física. Para efeito de escalonamento não é possível distinguir entre o cenário de 3 tarefas da aplicação executando em processadores diferentes e enviando mensagens em sequência e o cenário de 2 tarefas da aplicação enviando mensagens através de uma tarefa roteadora.

MONO + DINDEM

Como não existem tarefas roteadoras, cada tarefa da aplicação possui um processador a sua disposição. Também não existe compartilhamento de canal físico estabelecido, pois existe apenas uma tarefa para enviar mensagens. O maior problema de escalonamento neste cenário é a configuração do *crossbar*. É possível que as tarefas da aplicação solicitem simultaneamente conexões físicas que vão além da capacidade do *crossbar*. Neste caso, o nó de controle deve empregar algum tipo de algoritmo para escolher quais conexões serão estabelecidas antes. O tempo de execução das tarefas é afetado pelo tempo de envio de mensagens que, por sua vez, é afetado pelo comportamento da aplicação como um todo.

MULTI1 + DINDEM

Este cenário possui, como o anterior, a necessidade de um algoritmo para o nó de controle decidir a ordem na qual as solicitações de conexão serão atendidas. O fato de cada nó de trabalho executar várias tarefas simultaneamente aumenta a chance de conflito no *crossbar*. O processador dos nós de trabalho deve ser escalonado entre as diversas tarefas locais.

MULTI2 + DINDEM

Semelhante ao cenário anterior, com o acréscimo da necessidade de escalonar o uso de uma mesma conexão física entre duas tarefas da aplicação. Agora quando uma tarefa deseja estabelecer uma dada conexão física ela deve antes verificar se esta conexão já não existe e está sendo utilizada por outra tarefa local. O algoritmo utilizado pelo nó de controle também fica mais complexo pois agora várias tarefas de um mesmo nó de trabalho podem solicitar o estabelecimento da mesma conexão física.

MONO + DINSINC

Neste cenário é necessário em cada nó de trabalho uma tarefa para executar o algoritmo de sincronização de relógio. Esta é tipicamente uma tarefa rápida, porém que exige alta prioridade. Desta forma, o tempo de execução da tarefa da aplicação é afetado pela interferência gerada pela tarefa de sincronização de relógio. Entretanto, como a sincronização de relógio é feita segundo um algoritmo bem definido, não é difícil computar tal interferência. Em projeto é necessário determinar a escala de configuração do *crossbar*. Esta escala deve levar em conta os pontos de comunicação das tarefas. Uma conexão física somente deve ser programada para ser estabelecida em momentos quando as tarefas envolvidas estão prontas para efetuar a comunicação. Caso contrário, uma conexão física inútil será estabelecida. O modo DINSINC de operar o *crossbar* é mais apropriado quando as tarefas da aplicação executam um código bastante repetitivo em termos de tempo de execução e pontos de comunicação.

MULTI1 + DINSINC

Novamente existe a necessidade de sincronizar os relógios dos nós, pois a disponibilidade das conexões físicas esta associada com intervalos de tempo previamente estabelecidos. Como diversas tarefas executam no mesmo nó de trabalho, é importante que a execução das tarefas seja sincronizada com a configuração do *crossbar*. Caso contrário, existe a possibilidade de uma conexão física não ser aproveitada porque a tarefa correspondente não estava pronta para enviar uma mensagem a tempo. Esta combinação de fatores sugere uma abordagem tipo executivo cíclico para este sistema [5][8].

MULTI2 + DINSINC

Este cenário é semelhante ao anterior, com a complexidade adicional devido a várias tarefas compartilharem a mesma conexão física durante os intervalos de tempo nos quais ela

permanece estabelecida. Entretanto, podemos considerar estes intervalos de tempo como compostos por vários sub-intervalos. Cada sub-intervalo é associado com uma das tarefas que utilizam a conexão física. Desta forma, o problema é reduzido ao cenário anterior.

MONO + DINANUN

Nesta combinação não é necessária uma tarefa específica para efetuar a sincronização de relógios. Entretanto, todo nó de trabalho deve observar as mensagens difundidas pelo barramento de controle para determinar os momentos nos quais suas conexões físicas são estabelecidas. Desta forma, a tarefa de sincronização de relógios é substituida por uma tarefa de monitoração do barramento de controle, a qual provavelmente representará um consumo maior de processador. Entretanto, com respeito ao escalonamento do processador, este cenário é semelhante ao MONO+DINSINC. O mesmo acontece com a configuração do *crossbar*, a qual deve ser definida antes da execução em função das propriedades das tarefas que são conhecidas a priori.

MULTI1 + DINANUN

Este cenário é semelhante ao anterior. Apenas possui mais tarefas para escalonar. Além das tarefas associadas com a monitoração do barramento de controle, a própria aplicação possui diversas tarefas. O maior problema deste cenário é garantir que a tarefa estará pronta para enviar uma mensagem quando a conexão física correspondente tornar-se disponível. Como não existe sincronização de relógios, não é possível prever quando isto vai acontecer. Este cenário parece mais apropriado para abordagens probabilistas e não deterministas.

MULTI2 + DINANUN

Como várias tarefas desejam utilizar a mesma conexão física, existe a necessidade de escalonamento. Quando uma mensagem no barramento de controle informa a disponibilidade da conexão, um escalonador local decide a ordem na qual as tarefas locais enviarão suas mensagens. O processador é escalonado entre a tarefa que monitora o barramento de controle e as diversas tarefas da aplicação.

A tabela abaixo resume as necessidades de escalonamento para os diversos cenários examinados nesta seção.

Escalonamento:	**Processador (Nó trabalho)**	**Conexão física**	**Crossbar (on-line)**	**Crossbar (off-line)**	**Sincron. relógios**
Operação:					
MONO + ESTAT	.	.	.	.	.
MULTI1 + ESTAT	X	.	.	.	.
MULTI2 + ESTAT	X	X	.	.	.
MONO + ESTROT	X	X	.	.	.
MULTI1 + ESTROT	X	X	.	.	.
MULTI2 + ESTROT	X	X	.	.	.
MONO + DINDEM	.	.	X	.	.
MULTI1 + DINDEM	X	.	X	.	.
MULTI2 + DINDEM	X	X	X	.	.
MONO + DINSINC	X	.	.	X	X
MULTI1 + DINSINC	X	.	.	X	X
MULTI2 + DINSINC	X	X	.	X	X
MONO + DINANUN	X	.	.	X	.
MULTI1 + DINANUN	X	.	.	X	.
MULTI2 + DINANUN	X	X	.	X	.

A arquitetura alvo admite uma grande variedade de modos de operação, no que diz respeito aos processadores e canais de comunicação. O modo de operação mais apropriado vai depender dos requisitos temporais de cada aplicação em particular. A escolha do modo de operação vai determinar a melhor abordagem para o escalonamento tempo real do sistema.

5. Conclusões

Neste artigo foi analisada a dificuldade associada com o escalonamento tempo real em uma arquitetura baseada em multicomputadores. Inicialmente esta arquitetura foi descrita. Em seguida foram listados os diferentes modos de operação possíveis com respeito a processadores e canais de comunicação. Finalmente, a dificuldade para o escalonamento tempo real de cada um dos 15 cenários identificados foi discutida. Foram apontadas algumas técnicas descritas na literatura de tempo real que podem ser empregadas no contexto específico dos cenários.

Uma possibilidade interessante que ainda não foi analisada é a utilização de formas mistas de operação. Por exemplo, as conexões físicas no *crossbar* associadas com tarefas importantes da aplicação ou com tarefas sujeitas a fortes restrições temporais poderiam ser estáticas. As tarefas menos importantes ou sem restrições de tempo real teriam que requisitar dinamicamente suas conexões. Este cenário é interessante nas situações onde o *crossbar* não possui capacidade suficiente para estabelecer todas as conexões necessárias estaticamente, mas algumas tarefas não são capazes de suportar o tempo adicional necessário para estabeler conexões dinamicamente.

Um protótipo da máquina descrita neste artigo está em construção no INE-UFSC. Paralelamente, os autores esperam evoluir no sentido de construir um ambiente de programação e execução para esta máquina que contemple as necessidades de aplicações tempo real.

Agradecimentos

Este trabalho foi parcialmente financiado pela FAPERGS e pelo CNPq.

Referências

[1] J.H.Anderson, S.Ramamurthy. A Framework for Implementing Objects and Scheduling Tasks in Lock-Free Real-Time Systems. Proc. of the 17th IEEE Real-Time Systems Symposium, december 1996.

[2] A. M. Barroso, J.R.A.Torreão, J.C.B.Leite, O.G.Loques, J.S.Fraga. Metaheurísticas na Alocação de Tarefas em Sistemas Distribuídos de Tempo Real. Anais do VII Simpósio de Computadores Tolerantes a Falhas, Campina Grande-PB, junho 1997.

[3] T. B. Corso, J. S. Fraga , F. P. Freitas. Demand-Driven Multicomputer environment: Design and Evaluation. SCS Western Multi Conference 1998, San Diego, USA, January 1998.

[4] J. Kevin. The Real-Time Producer/Consumer Paradigm: A Paradigm for the Construction of Efficient, Predictable Real-Time Systems. Proceedings of the ACM/SIGAPP Symposium on Applied Computing, pp. 796-804, 1993.

[5] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabi, C. Senft, R. Zainlinger. Distributed Fault-Tolerant Real-Time Systems: The Mars Approach. IEEE Micro, pp. 25-40, february 1989.

[6] L. Sha, R. Rajkumar, J. P. Lehoczky. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. IEEE Transactions on Computers, Vol. 39, No. 9, pp. 1175-1185, september 1990.

[7] K. W. Tindell, A. Burns, A. J. Wellings. Allocating Hard Real-Time Tasks: An NP-Hard Problem Made Easy. Real-Time Systems, Vol. 4, No. 2, pp. 145-165, june 1992.

[8] J. Xu, D. L. Parnas. On Satisfying Timing Constraints in Hard-Real-Time Systems. IEEE Transactions on Software Engineering, Vol. 19, No. 1, pp. 70-84, January 1993.

[9] C. A. Zeferino. Projeto do Sistema de Comunicação de um Multicomputador. Dissertação de Mestrado, CPGCC-UFSC, Florianópolis, maio 1996.