UM ALGORITMO DE ROTEAMENTO ADAPTATIVO EM ESTRUTURAS MESH N-DIMENSIONAIS

Celso A. S. Santos[1], Edward D.Moreno O[2], Martha X. Torres D.[3], Sérgio T. Kofuji[4]

Laboratório de Sistemas Integráveis

Departamento de Engenharia Eletrônica

Escola Politécnica da Universidade de São Paulo[5]

Resumo:

Este artigo apresenta um algoritmo de roteamento livre de travamento (**deadlock-free**), livre de ciclos vivos (**livelock-free**), tolerante a falhas, adaptativo e minimal, sempre que possível. As características básicas deste modelo são a quebra de ciclos de espera de recursos, potencialmente causadores de deadlock, e a utilização de canais virtuais na solução de conflitos entre mensagens. Simulações em estruturas mesh tridimensionais serão apresentadas e seus resultados comparados com outros algoritmos tradicionais de roteamento.

Abstract:

This paper presents a deadlock-free, livelock-free , fault tolerant, adaptive and minimal (if possible) routing algorithm. The major characteristics that model are the break of resources waiting cycles between messages. Simulations in 3d-mesh structures will be present and their results compared with others traditional routing algorithms.

[1] Pesquisador do LSI da USP. E-mail: saibel@lsi.usp.br

[2] Pesquisador do LSI da USP. E-mail: edmoreno@lsi.usp.br

[3] Pesquisador do LSI da USP. E-mail: mxtd@lsi.usp.br

[4] Professor e Pesquisador do LSI da USP. E-mail: kofuji@lsi.usp.br

[5] Av.Prof. Luciano Gualberto, trav.3, No. 158

CEP 05508-900 -São Paulo-Brasil

Tel/Fax: (011) 211-4574

1. Introdução:

Sistemas multiprocessadores surgiram com o objetivo de alcançar um maior poder de processamento. Redes diretas oferecem a possibilidade de construção de sistemas maciçamente paralelos [1,2,3], apresentando-se muito mais escaláveis que outras implementações com interconexão multiprocessadora. Sistemas baseados em redes diretas caracterizam-se pela execução assíncrona de tarefas em diferentes processadores (ou nós de processamento), cada um contendo seu próprio processador, memória local e algum dispositivo de suporte, tais como roteadores. A comunicação e o sincronismo são exclusivamente realizados pela troca de mensagens. Devido às dificuldades de construção, cada nó da rede conecta-se a alguns poucos nós, denominados vizinhos, de acordo com uma determinada topologia. Com isso, normalmente mensagens tem de ser roteadas através de nós intermediários para que alcancem seus destinos finais. Para manipular estes casos, um algoritmo de roteamento deve ser utilizado com o objetivo de que cada mensagem gaste o menor tempo possível (mínima latência) ao atravessar a rede. Este é um dos fatores críticos de limitação de desempenho em multicomputadores. Além disso, estes algoritmos devem possuir características que proporcionem um alto **throughput** para a rede de interconexão e facilidade de implementação em hardware. A inexistência de deadlocks e livelocks, tolerância a falhas, utilização de rotas minimais e o roteamento adaptativo das mensagens são essenciais para que um alto **throughput** e uma baixa latência sejam alcançados pela rede.

Técnicas fixas de roteamento, ou roteamentos determinísticos, limitam o **throughput** máximo sustentável da rede em cerca de 50% de sua capacidade [4] - **throughput** é definido como a quantidade de mensagens liberadas para rede num intervalo de tempo. Apresentar novos algoritmos adaptativos que melhorem este rendimento parece ser um caminho bastante viável para a obtenção de uma melhor desempenho em multicomputadores.

Na seção 2, serão dadas algumas definições relacionadas ao sistema em estudo e definições das notações necessárias. Na seção 3, será apresentado o conjunto de restrições utilizado pelo roteador. Nas seções seguintes serão descritos o algoritmo de roteamento adaptativo a ser analisado e o simulador SIM3D desenvolvido para o estudo de estruturas **mesh 3d.** Serão levantados alguns resultados para a comprovação da eficiência do algoritmo sob diversas situações.

2. Definições:

2.1- Flit

Um *flit* (*flow control digit*) é a menor unidade de informação que um canal pode aceitar ou recusar. Geralmente um *pacote*[1] consiste de muitos *flits*. A unidade de comunicação que é visível ao programador é a *mensagem*. Uma mensagem pode ser composta de um ou mais *pacotes*, cada um carregando informações de rota e sequência em seu **header** [7].

2.2- Wormhole routing

Wormhole routing [5] refere-se a um protocolo de controle de fluxo que avança cada flit de um pacote assim que ele chega a um nó (**pipelining**) e bloqueia pacotes que não podem prosseguir pois os recursos necessitados por eles estão indisponíveis. Esta técnica é bastante atraente porque reduz a latência da mensagem comparada à **store-and-foward** e requer somente poucos armazenadores (buffers) por nó. *Wormhole* difere basicamente da técnica **virtual cut-through** [6] por não exigir a retirada do pacote da rede (armazenando-o completamente), quando este é bloqueado.

2.3- Deadlock, Livelock, Fairness

Um *deadlock* ocorre quando um pacote espera por eventos que não podem acontecer. Sempre que existe um ciclo de espera de recursos, existe a possibilidade da ocorrência de deadlock [8]. Em redes que utilizam wormhole, um deadlock pode ocorrer quando pacotes esperam por outros ciclicamente como na Figura 1. Estes ciclos podem ser evitados pela definição de uma ordem parcial de utilização dos recursos (buffers e canais) [3,8,9]. *Fairness* é a garantia da não existência de repostagem infinita. Essa repostagem é similar ao *deadlock*, porém neste caso, um pacote espera indefinidamente numa situação de conflito de recursos, sem nunca poder obtê-lo. Isto ocorre quando um pacote espera por recursos que sempre estarão sendo disputados por outros pacotes simultaneamente. Uma situação de *Livelock* , ao contrário das anteriores, permite que o pacote continue se movimentando, entretanto ele é continuamente desviado, sem nunca alcançar seu destino.

2.4- Um *mesh* n-dimensional é uma estrutura com $k_0 \times k_1 \times ... \times k_{n-1}$ nós (ou processadores), k_i ao longo de cada dimensão, onde $k_i > 1$. Cada nó **p** é identificado por suas n coordenadas (Figura 2). Num *mesh-2D*, por exemplo, define-se **p** como um par (n_x, n_y), onde n_x corresponde à distância em relação a origem (0,0) na direção x e n_y na direção y. Os nós estão ligados a um número de vizinhos que varia entre n e 2n, dependendo de posições no *mesh*. Admite-se que a conexão entre

[1] Neste artigo estão sendo tratados mensagens e pacotes indistintamente.

os nós é feita através de dois canais unidirecionais, um em cada direção. Um canal é denotado por c , ou (f,d), onde f é o nó fonte do canal e d o destino.

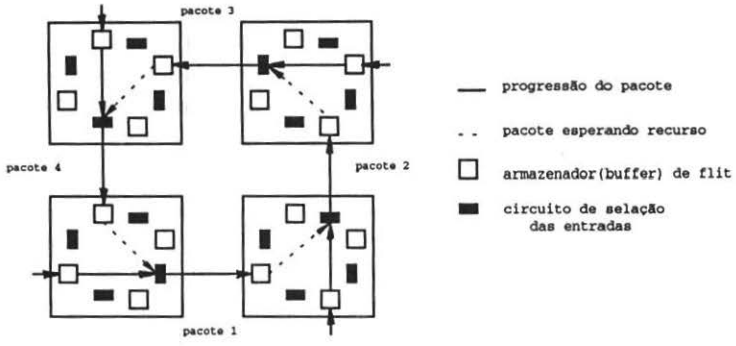

Figura 1 - Um deadlock usando wormhole envolvendo quatro roteadores e quatro pacotes.

2.5- Um canal $c = (f,d)$ é um canal na dimensão l se as coordenadas dos nós fonte e destino diferirem nesta dimensão. Assume-se que sendo n o número de um nó, $n(l)$ corresponde a coordenada na dimensão l. Assim, c está na dimensão l se $f(l) \neq d(l)$.

2.6- Um canal $c = (f,d)$ da dimensão l será positivo se $f(l) < d(l)$, caso contrário será negativo. Por exemplo, o canal c = [(0,1);(0,2)], conectando os nós (0,1) e (0,2), é um canal positivo na dimensão 1 (ou x).

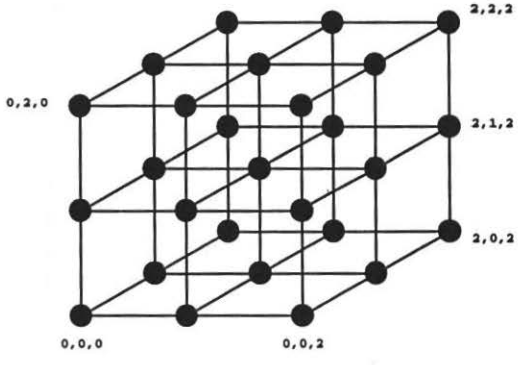

Figura 2 - Rede direta em estrutura Mesh tridimensional 3x3x3.

3. Restrições de Roteamento:

Suponha dois canais c_i (n_1 ,n_2) e c_j (n_2,n_3) . As restrições de roteamento são regras que determinam se os canais que chegam a um nó (n_2), como c_i (n_1 ,n_2), podem ou não repassar (**forward**) mensagens para outro canal saindo deste mesmo nó, c_j (n_2,n_3). Se uma mensagem de um canal c_i puder ser repassada para c_j e esse canal estiver ocupado por outro pacote, quatro políticas básicas podem ser utilizadas para solucionar a colisão: armazenagem, bloqueio, desvio e descarte [13]. A política de armazenar totalmente o pacote até que o canal seja desocupado é aplicada a redes que utilizam **store-and-foward** e **virtual cut-through**, enquanto que o bloqueio está presente em redes **wormhole**. O desvio é utilizado em roteamentos adaptativos e o descarte, essencialmente em redes com chaveamento de circuito (**circuit switching**).

No caso de roteadores determinísticos, o conjunto de restrições é extremamente rígido para que se assegure a inexistência de deadlocks. No algoritmo *e-cube* para hipercubos [3], por exemplo, um pacote é primeiro roteado numa dimensão mais baixa, *l*, até que $f(l) = d(l)$. Após isso, o pacote é repassado à próxima dimensão (mais alta), e assim sucessivamente. Essa ordenação das dimensões percorridas pelo pacote evita o **deadlock**, impedindo entretanto qualquer adaptatividade. Dally e Seitz extenderam este algoritmo para cubos-n k-ários através da utilização de canais virtuais [8]. Adicionar canais virtuais a uma rede corresponde a compartilhar canais físicos, com consequente redução de banda de passagem para cada via virtual, porém com um custo muito inferior ao simples aumento do número de ligações.

Algoritmos não-determinísticos são muito menos restritivos e proporcionam um alto grau de adaptatividade, além de uma maior tolerância a falhas. Define-se um algoritmo como minimal se ele permite rotas apenas através dos caminhos mínimos entre dois nós. Num algoritmo minimal parcialmente adaptativo, nem todos os caminhos mínimos existentes entre dois nós podem ser utilizados devido às restrições.

O conjunto de restrições definido abaixo é o utilizado pelo algoritmo proposto. Mensagens podem ser repassadas de um canal c_i(n_1 ,n_2) para outro c_j(n_1 ,n_2) através do nó n_2 , se e somente se uma das sentenças for verdadeira :

(1) c_i e c_j são canais negativos ;

(2) c_j é um canal positivo.

Ou seja, canais positivos não podem repassar mensagens para canais negativos. Estas condições são necessárias e suficientes para evitar o deadlock.

Prova: Num ciclo o número de canais positivos e negativos é o mesmo numa mesma dimensão. Se num ciclo com n nós, existirem $m \leq n$ nós numa mesma dimensão, por exemplo l, $m/2$ serão positivos e $m/2$ negativos. Para que seja completado um ciclo, quando a mensagem atravessar o

$m/2$-ésimo canal positivo, último positivo da dimensão l, deverá ocupar um canal também positivo, por (2), em outra dimensão e assim sucessivamente entre as dimensões. Com isso, a mensagem nunca poderá ser repassada para qualquer canal negativo, e o ciclo não existe, pois os $m/2$ canais negativos restantes na dimensão l não podem ser alcançados. Se as mensagens agora estão num dos $m/2$ canais negativos da dimensão l, em algum momento deverão ser repassadas para os $m/2$ canais positivos, a fim de que se estabeleça o ciclo. Neste caso, voltamos a situação inicialmente exposta, e o ciclo não pode ser completado. Assim, o algoritmo sob este conjunto de restrições é deadlock-free.

Um resultado bastante semelhante foi obtido por Qiang Li [10] para topologias hipercubo.

Seguindo as restrições descritas, o modelo do algoritmo torna-se exatamente idêntico ao obtido por Glass e Ni [11] utilizando o modelo das voltas (**turn model**), denominado roteamento negativo-primeiro (**negative-first routing**), no qual um pacote é roteado adaptativamente na direção negativa (se possível) e, só então, na positiva, também adaptativamente.

Vale ressaltar que o grau de adaptatividade do algoritmo varia de acordo com a posição relativa do par de nós fonte e destino. Sendo $\Delta x = dx - fx$ e $\Delta y = dy - fy$, num **mesh-2d**, tem-se que :

$$\text{Adaptatividade} = \begin{cases} 1 & \text{se} \quad \Delta x.\Delta y \leq 0 \\ \dfrac{(|\Delta x|+|\Delta y|)!}{|\Delta x|!|\Delta y|!} & \text{se} \quad \Delta x.\Delta y > 0 \end{cases}$$

Isto encorajaria um balanceamento de carga entre os processadores de forma que as comunicações explorassem as direções favoráveis para a adaptatividade conforme a Figura 3:

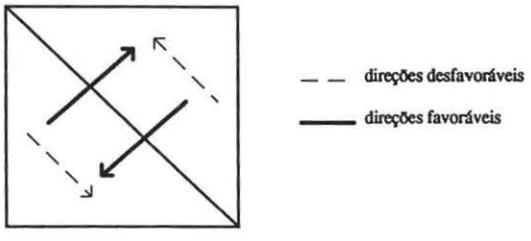

Figura 3 - Direções favoráveis e desfavoráveis para comunicação

A diferença básica entre a implementação anterior e a proposta neste artigo é que aumentam as possibilidades de utilização de caminhos mais curtos no caso de conflito de recursos. No algoritmo negativo-primeiro, as colisões são resolvidas através da seguinte sequência de ações: primeiro tenta-se enviar o pacote através de um dos canais pertencentes ao conjunto de caminhos mínimos permitidos pelas restrições entre os nós. Se todos estiverem ocupados, o pacote será desviado (**misrouting**), percorrendo um caminho não-minimal. Se isso não for possível, o pacote será finalmente bloqueado. Uma alternativa que será estudada é a utilização de canais virtuais compartilhando o canal físico em conflito, antes de tentar o desvio.

4. Descrição do Algoritmo:

Seja Nó_dest o número do nó destino com N coordenadas, uma em cada dimensão.
Seja Nó_atual o número do nó em que a mensagem se encontra num determinado instante.

Para cada mensagem que chega ao nó:
Dif = Nó_dest - Nó_atual { diferença em cada dimensão }

Sejam P1, P2, ..., Pn as n posições de Dif correspondentes às dimensões.
Comparar cada posição de Dif com 0
Se Dif (P_1) e Dif (P_2) e ... e Dif (P_n) = 0
então absorve a mensagem no nó
senão *Se* Dif (P_1) ou Dif (P_2) ou ... ou Dif (P_n) < 0
então escolher um canal negativo em uma das dimensões onde Dif (P_i) < 0
senão escolher qualquer canal positivo em uma das dimensões nas quais Dif (P_i) é diferente de 0.
fim se
fim se

Para a escolha do canal:
Para cada uma das posições Pi (Dimensões) não-nulas resultantes em Dif :
Obter o carregamento dos canais de saída (número de canais virtuais utilizado) para cada uma das i dimensões;
Determinar o canal menos ocupado e *enviar a mensagem* através dele;
Se todos os canais requisitados estiverem indisponíveis e se a mensagem tiver chegado por um canal negativo : *tentar desviar*.
Se todos os canais possíveis estiverem ocupados: *bloquear a mensagem*.

Para desviar:
Determinar algum canal disponível em qualquer dimensão desde que o canal de entrada da mensagem no nó seja negativo (c_i e c_j devem ser negativos).

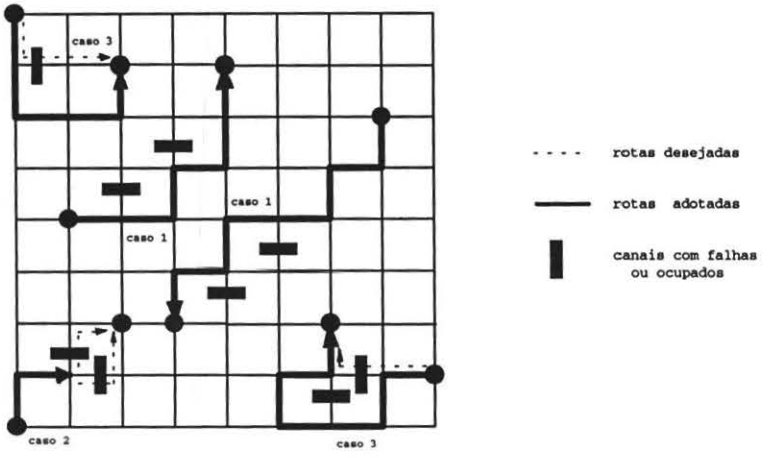

Figura 4 - Exemplos de roteamento seguindo o algoritmo proposto em mesh 2D. No caso 1, a mensagem é enviada por um dos caminhos permitidos; no caso 2 é bloqueada; no caso 3 é desviada (**misrouting**) de acordo com as restrições - caminho não-minimal .

5. Descrição do Simulador SIM3D :

Para estudar o comportamento do sistema sob diferentes formas de roteamento, foi desenvolvido um simulador denominado SIM3D. O programa é composto de aproximadamente 1000 linhas em C, sendo capaz de simular estruturas **mesh 3D** com um número máximo de 16x16x16 nós (16 em cada dimensão). Cada nó No(x,y,z) é constituído por um elemento de comunicação, COM, e por um elemento de processamento, PROC. O elemento de comunicação é responsável pelo roteamento das mensagens que chegam (ou saem) do nó através dos 7 canais de entrada e saída (6 correspondentes aos pares unidirecionais conectando os nós adjacentes em cada uma das dimensões e 1 à ligação com seu processador local). Assume-se que cada mensagem é composta por diversos flits, sendo cada flit composto por 5 bytes (1 byte de **header** e 4 bytes de dados). Para sinalizar o início e o término da tranmissão de uma mensagem, são utilizados dois flits , um de cabeçalho (**header**) e o outro de fim de comunicação (**tail**). O flit cabeçalho contém informações sobre o destino da mensagem, enquanto o de finalização carrega dados estatísticos que serão utilizados no levantamento das características do sistemas nas diferentes situações propostas. O simulador apresenta uma boa flexibilidade, permitindo variação do número de nós em cada dimensão, do tamanho dos flits (mensagens), do número de canais virtuais e principalmente, do algoritmo de roteamento para as mensagens.

6. Simulações:

Para comparar o algoritmo adaptativo proposto com determinístico ordenado por dimensão, foram consideradas as seguintes características para o sistema: todos os canais têm a mesma largura de banda. Os **buffers** dos canais de entrada têm largura de um flit (5 bytes). Os processadores geram mensagens de tamanho fixo, no caso 64 flits, segundo uma distribuição exponencial negativa, de acordo com os níveis de tráfego a serem estudados. Para cada simulação foram observadas as seguintes características: latência média de comunicação e intervalo de tempo entre a liberação de mensagens. Foi considerado um tráfego *uniforme* para a rede, no qual as mensagens podem ser enviadas para qualquer processador com a mesma probabilidade.

A partir dos dados obtidos pelas simulações, pode ser observado que a diferença de comportamento do sistema é muito mais acentuada no que diz respeito a latência média da mensagens para tráfegos mais pesados (maior número de mensagens entrando na rede). A utilização de rotas não fixas (algorimo não-determinístico) proporcionou um aumento do throughput sustentável da rede, mantendo uma latência razoável, como mostra a Figura 5. Outro ponto a ser observado é que quando o carregamento da rede diminui (maior tempo entre liberação das mensagens), as latências para o caso de roteamento fixo e adaptativo tendem a se igualar. Maiores estudos sobre os efeitos do uso de canais virtuais em redes diretas deverão ser feitos posteriormente, a fim de encontrar outros pontos que possam ser influenciados por eles no sistema.

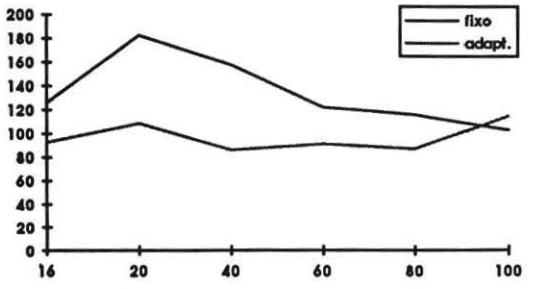

Figura 5 - Comportamento da latência com a variação do intervalo entre mensagens geradas por cada processador- o eixo vertical corresponde à latência média e o horizontal ao intervalo entre as mensagens.

BIBLIOGRAFIA:

[1] D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta, and J. Hennessy, "The directory-based cache coherence protocol for the DASH multiprocessor", Proc. of the 17th International Symposium on Computer Architecture, pp. 148-159, Mai 1990.

[2] Seitz, C. L., "The cosmic cube", Communications of the ACM 28, 1, pp 22-23, Jan 1985.

[3] Sullivan, H. e Brashkow, T. R., "A large scale homogeneous machine", Proc. of the 4th Annual Symposium on Computer Architecture, pp. 105-124, 1977.

[4] Dally, W. J., "Desempenho analysis of k-ary n-cube interconection networks", IEEE Transactions on Computers, vol. C-39, pp. 775-785, Jun 1990.

[5] Dally, W. J. e Seitz, C. L., "The torus routing chip", Journal of Distributed Computing, vol. 1, no. 3, pp187-196, 1986.

[6] Kermani, P. e Kleinrock, L., "Virtual cut through: A new computer communication switching technique", Computer Networks, vol. 3, pp. 267-286, 1979.

[7] Dally, W. J., "A VLSI Architecture for concurrent data structures", Ph.D. thesis, California Institute of Technology, 1986.

[8] Dally, W. J. e Seitz, C. L., "Deadlock-free message routing in multiprocessor interconection networks", IEEE Trans. on Computers, vol C-36, pp 547-553, Mai 1987.

[9] Gelernter, D., "A DAG-based algorithm for prevention of store-and-foward deadlock in packet networks", IEEE Trans. on Communication, vol. COM-29, pp 512-524, Abr 1981.

[10] Li, Q., "Minimum Deadlock-free Message Routing Restrictions in Binary Hypercubes", Journal of Parallel and Distributed Computing 15, pp 153-159, 1992.

[11] Glass, C. J. e Ni, L. M., "The Turn Model for Adaptative Routing", Proc. of the 19th Annual Symposium on Computer Architecture, pp 278-287, Mai 1992.

[12] Reed, D. A. e Fujimoto, R. M., "Multicomputer Networks : Message-Based Parallel Processing", MIT Press, 1987.

[13] Vários, "VLSI and Parallel Computation", Editado por Robert Suay e Grahan, Morgan Kaufmann Publishers, INC 1990.

[14] Ni, L. e McKinley, P. K., "A Survey of Wormhole Routing Techniques in Direct Networks", Computer, vol 26, no. 2, pp 62-76, Fev 1993.

[15] Grunwald, D. C., "Circuit Switched Multicomputers and Heuristic Load Placement", PhD Thesis, Universidade de Illinois - Urbana, Set. 1989.