Estratégias de Comunicação em Multiprocessadores Fracamente Acoplados

João Paulo F. W. Kitajima (*)
Philippe O. A. Navaux (**)

Universidade Federal do Rio Grande do Sul
Pós-Graduação em Ciência da Computação
Caixa Postal 1.501
90.001 - Porto Alegre, RS, Brasil
Tel: (0512) 21-8499 - Fax: (0512) 24-4164

RESUMO

A comunicação entre processadores em um máquina paralela fracamente acoplada é um aspecto crucial que afeta o desempenho do sistema como um todo. Quatro estratégias são analisadas: **message switching** [KER_79], **cut-through** [KER_79], **wormhole** [DAL_87] e **rendez-vous** (caminho virtual). De acordo com os resultados dos modelos analíticos e de simulação, mensagens roteadas através de **cut-through** e de **wormhole** apresentaram tempos de comunicação (latência) menores em relação às demais estratégias.

ABSTRACT

The interprocessor communication in a loosely-coupled parallel machine affects decisively the system performance as a whole. Four strategies are analysed: **message switching** [KER_79], **cut-through** [KER_79], **wormhole** [DAL_87] and **rendez-vous** (virtual path). According to analytic and simulation models results, messages routed by **cut-through** and **wormhole** presented smaller communication time (latency) than those routed by **message switching** and **rendez-vous**.

(*) **Mestrando**
Universidade Federal do Rio Grande do Sul
Tecnólogo em Processamento de Dados pela
Universidade de Brasília
Áreas de Interesse: Processamento Paralelo
Avaliação de Desempenho

(**) Professor
Universidade Federal do Rio Grande do Sul
Doutor Engenheiro - Informática pelo
Instituto Nacional Politécnico de Grenoble, França
Áreas de Interesse: Arquitetura de Computadores
Processamento Paralelo
Avaliação de Desempenho

1. Paralelismo e Máquinas Fracamente Acopladas

Os problemas de caráter computacional exigem, cada vez mais, um maior poder de processamento. Estes problemas pertencem às diversas áreas do conhecimento e cobrem uma ampla gama de aplicações. A demanda por potência de computação pode ser satisfeita (a) através da estruturação adequada do algoritmo associado à solução ou (b) através de um computador mais eficiente. Um aspecto não está dissociado do outro e, de fato, somente através da combinação das duas alternativas acima, é possível conceber um aumento significativo do desempenho de aplicações.

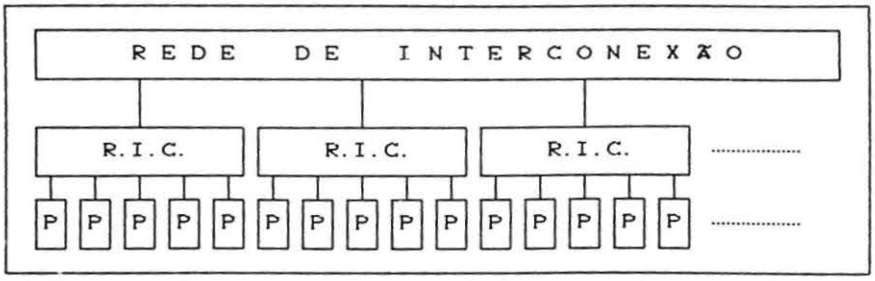
Dentro deste contexto, um novo modelo de computação surgiu a partir do questionamento sobre a possibilidade de um algoritmo ser executado por mais de um processador, assim como uma casa é construída por uma equipe de operários. O processamento paralelo é um termo que caracteriza uma gama variada de sistemas, desde redes de computadores até componentes de circuitos integrados. Mesmo arquiteturas uniprocessadoras convencionais apresentam certas formas de paralelismo.

Dentre as diversas arquiteturas paralelas idealizadas e existentes, os **multiprocessadores** constituem uma classe representativa. Caracterizam-se pela existência de múltiplas unidades de processamento, cada uma capaz de executar seu próprio programa. Além da memória privativa de cada processador, pode existir uma memória de comum acesso. Todos os recursos do multiprocessador são coordenados para um único objetivo: a execução do algoritmo corrente. Assim, o conjunto de processadores, controlados por um mecanismo central ou distribuído, forma uma entidade única [MAR_86]. O multiprocessador deve ser controlado por um único sistema operacional, com capacidade de prover, em diversos níveis, as interações entre processadores e programas [HWA_84].

Os multiprocessadores podem ser classificados [HWA_84] em **fracamente acoplados** ("loosely coupled") e **fortemente acoplados** ("tightly coupled"). Os multiprocessadores de fraco acoplamento possuem processadores com interface para dispositivos de entrada/saída e com uma memória local capaz de armazenar instruções e dados. Não há uma memória central. Os processadores estão interconectados por canais de comunicação, os quais formam uma rede de interconexão. Programas em diferentes processadores comunicam-se através de troca de mensagens que circulam pela rede de interconexão. As unidades de processamento podem ser agrupadas em conjuntos. Conjuntos podem ser agrupados em outros conjuntos: a estrutura do multiprocessador torna-se hierárquica (figura 1).

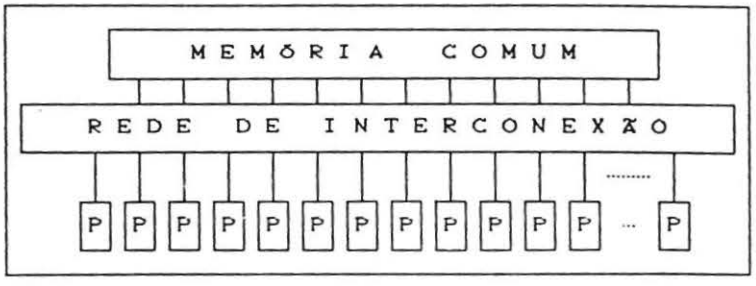
Por outro lado, multiprocessadores fortemente acoplados possuem uma memória de acesso comum a todos os processadores. O sistema de comunicação conecta os diversos processadores à memória comum e qualquer troca de dados é feita através desta memória. Os processadores possuem também

uma memória privativa para armazenamento de dados e de instruções freqüentemente utilizados (memória cache) (figura 2).



R.I.C. : rede de interconexão do conjunto
P : processador

Figura 1 - Multiprocessador Fracamente Acoplado



P : processador

Figura 2 - Multiprocessador Fortemente Acoplado

Os multiprocessadores representam uma evolução tecnológica em termos de arquitetura de computadores. Evidentemente, o advento do paralelismo e, conseqüentemente, dos multiprocessadores trouxe uma série de novos problemas, antes inexistentes no paradigma seqüencial. Como exemplos, pode-se citar a paralelização do algoritmo que implementa a solução do problema em questão, o mapeamento [BOK_81] dos módulos paralelos deste algoritmo em um conjunto arbitrário de processadores, o balanceamento da carga [EAG_86] do multiprocessador a fim de se obter uma utilização homogênea do mesmo e a determinação de um esquema adequado de suporte à comunicação entre os diversos processos paralelos.

Dentre as questões acima, a comunicação afeta decisivamente a execução do algoritmo paralelo. A fração do tempo em que os processadores estão envolvidos em trocas de mensagens deve ser o menor possível, a fim de não influenciar o tempo total de execução do algoritmo. Além disto, as outras questões associadas ao paralelismo

(mapeamento, paralelização, balanceamento de carga) dependem de um suporte adequado para a troca de mensagens.

2. Estratégias de Comunicação

A máquina paralela de interesse para este trabalho é o multiprocessador fracamente acoplado. Cada ligação entre dois processadores é composta de dois canais: um de saída e outro de entrada, que operam de maneira independente. Considera-se uma rede de interconexão com topologia arbitrária. A troca de dados entre os nós processadores é feita através de mensagens, compostas de duas partes: um cabeçalho e os dados propriamente ditos. O cabeçalho possui informações de controle como o endereço do nó destinatário. Todos os bits da mensagem são transmitidos sem interrupção e assincronamente. A mensagem , para chegar ao seu destino, pode atravessar mais de um processador.

A comunicação envolve uma série de atrasos [BER_89]:

- tempo de processamento: necessário para preparar a mensagem a ser transmitida, obter informações de roteamento e realizar procedimentos de controle de erros
- tempo de enfileiramento: tempo gasto na espera por liberação de recursos necessários à comunicação, como as linhas de transmissão
- tempo de transmissão: correspondente ao tempo gasto para a transmissão de todos os bits da mensagem
- tempo de propagação: intervalo de tempo entre o fim da transmissão do último bit da mensagem no processador origem e a recepção do último bit desta mensagem pelo processador vizinho.

O parâmetro de análise do grau de comunicação aqui considerado é o tempo médio de comunicação entre dois processos, localizados em nós distintos. Este intervalo de tempo abrange o momento em que o processador emissor envia a mensagem até a recepção completa da mesma pelo destinatário. Os únicos atrasos não desprezados são o de transmissão e o de enfileiramento (espera de liberação da linha de transmissão).

O modelo de comunicação supõe que a comunicação seja feita de forma totalmente confiável e que os algoritmos de roteamento sempre encontram um caminho entre o processador origem e o destino (o tempo gasto para determinar o caminho a ser seguido pela mensagem não é considerado - não há atraso de processamento). As estratégias de comunicação garantem a não ocorrência de bloqueios mortais (deadlock).

A comunicação entre dois processos situados em processadores fisicamente distintos pode ocorrer de diversas maneiras. Quatro estratégias foram escolhidas para análise: **message switching** [KER_79], **cut-through** [KER_79], **wormhole** [DAL_87] e **rendez-vous**. Estas estratégias não são as únicas, mas podem ser consideradas como representativas.

2.1 Message Switching (Comutação de Mensagens)

O processador emissor envia uma mensagem a um nó determinado da rede. Esta mensagem poderá percorrer um ou mais canais de comunicação até chegar ao seu destino. Um processador intermediário, ao receber uma mensagem, analisa o cabeçalho da mesma e verifica o endereço de destino. Normalmente, uma acusação de recebimento (ACK) é enviada ao processador vizinho emissor da mensagem. A acusação de recebimento de mensagem é importante se há alguma exigência quanto à confiabilidade da comunicação, mas não é considerada neste estudo.

Analisado o cabeçalho, o processador verifica se ele próprio é o destino. Caso positivo, a comunicação está encerrada. Senão o processador toma uma decisão sobre para qual canal de saída despachar a mensagem. Esta decisão depende de informações de roteamento. Definido o canal de saída, a mensagem é colocada em uma fila e aguarda a liberação deste canal. Quando a linha estiver livre, a mensagem é transmitida, recebida pelo processador vizinho e novamente analisada. Esta rotina é repetida até a mensagem chegar ao processador destino (figura 3).

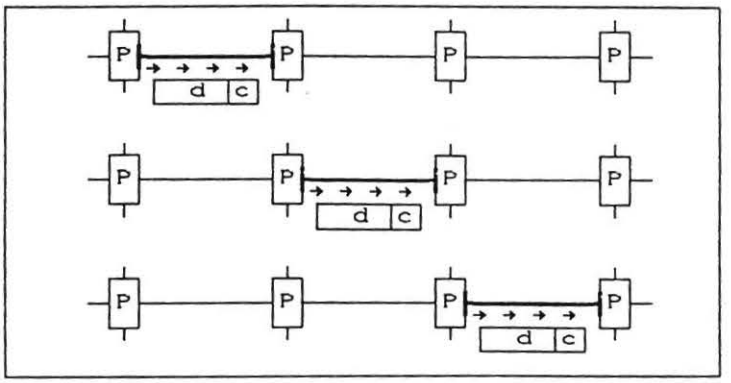
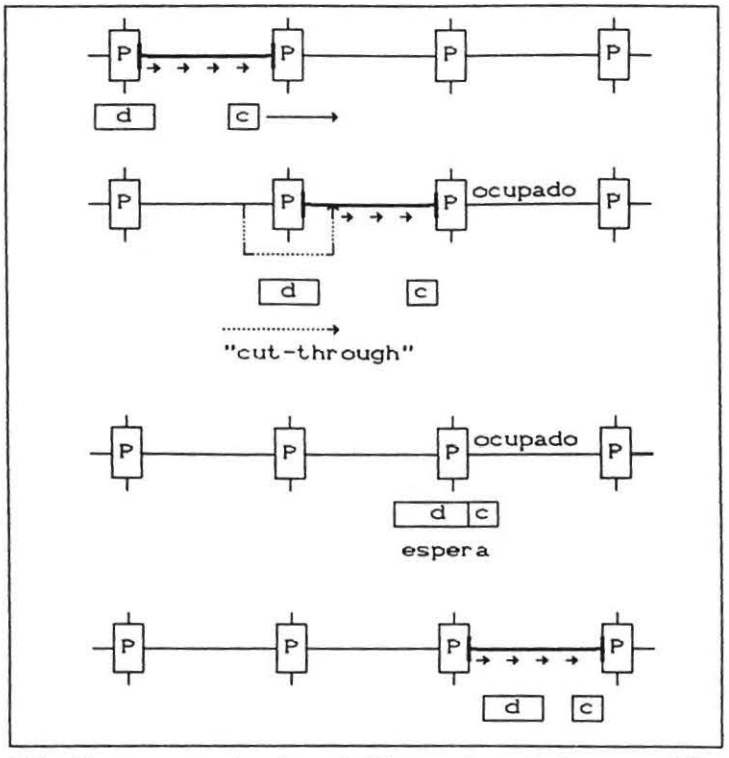


Figura 3 - **Message Switching**

2.2 Cut-through

O **cut-through** é muito semelhante ao **message switching**. A diferença está no momento em que a mensagem é recebida por um processador intermediário. O cabeçalho da mesma é analisado em tempo de recepção, antes mesmo da mensagem restante ser recebida. Para tanto, é necessário existir um dispositivo físico dedicado a esta análise. Decodificado o endereço do destino, o roteamento determina um canal de saída e se o mesmo estiver livre, a mensagem é automaticamente direcionada para esta linha de saída, não ocorrendo retransmissão da parte de dados. O armazenamento no nó intermediário com a colocação da mensagem na fila só

ocorre se o canal estiver ocupado (figura 4).



(Hipótese: canais 1 e 2 livres/canal 3 ocupado)

Figura 4 - **Cut-through**

2.3 Rendez-vous

O **rendez-vous** utiliza um procedimento semelhante à comutação de circuitos, adotado pela telefonia. O processador que deseja comunicar-se envia uma solicitação de comunicação ao processador destino. Esta solicitação é conduzida pela rede da mesma forma que uma mensagem é enviada pelo processo de comutação. A diferença é que, a cada linha percorrida, o canal é reservado, ou seja, nenhuma outra comunicação pode utilizá-lo. Se, durante o percurso, algum canal estiver alocado para outra comunicação, o pedido aguarda pela liberação do mesmo. No momento em que a solicitação de comunicação chega ao nó destino, o mesmo envia (pelos canais já reservados) um aceite de comunicação. O emissor, ao receber o aceite, envia a mensagem propriamente dita. A mensagem percorre o caminho reservado e libera o mesmo, à medida que atravessa a rede (figura 5).

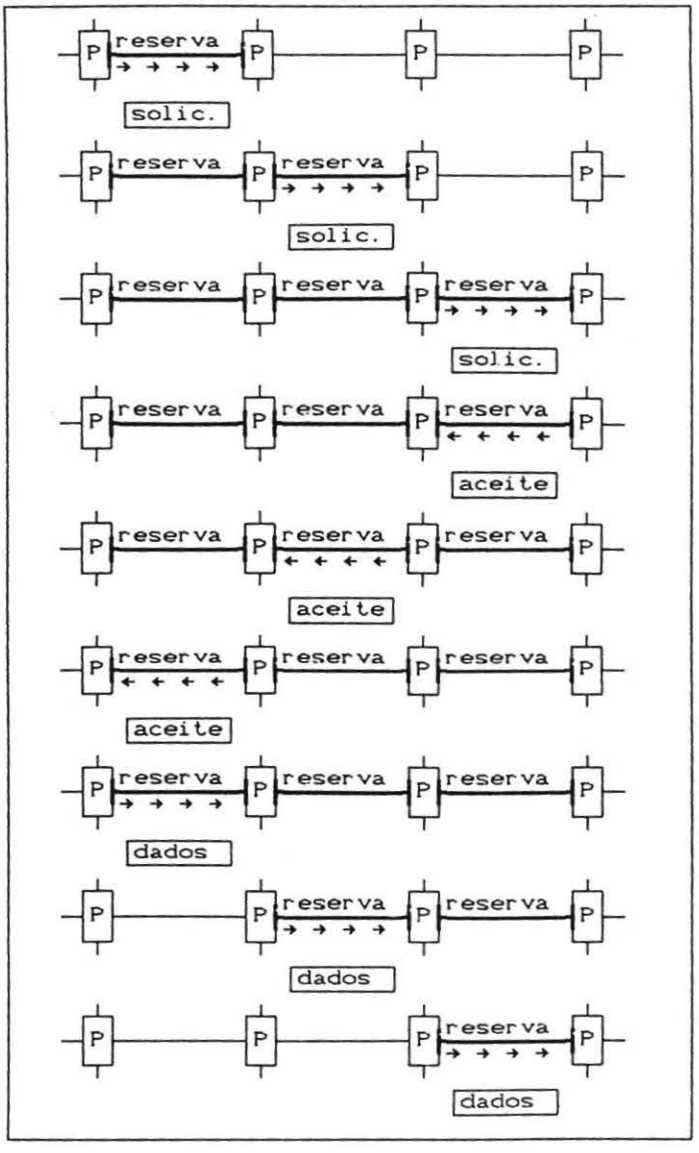
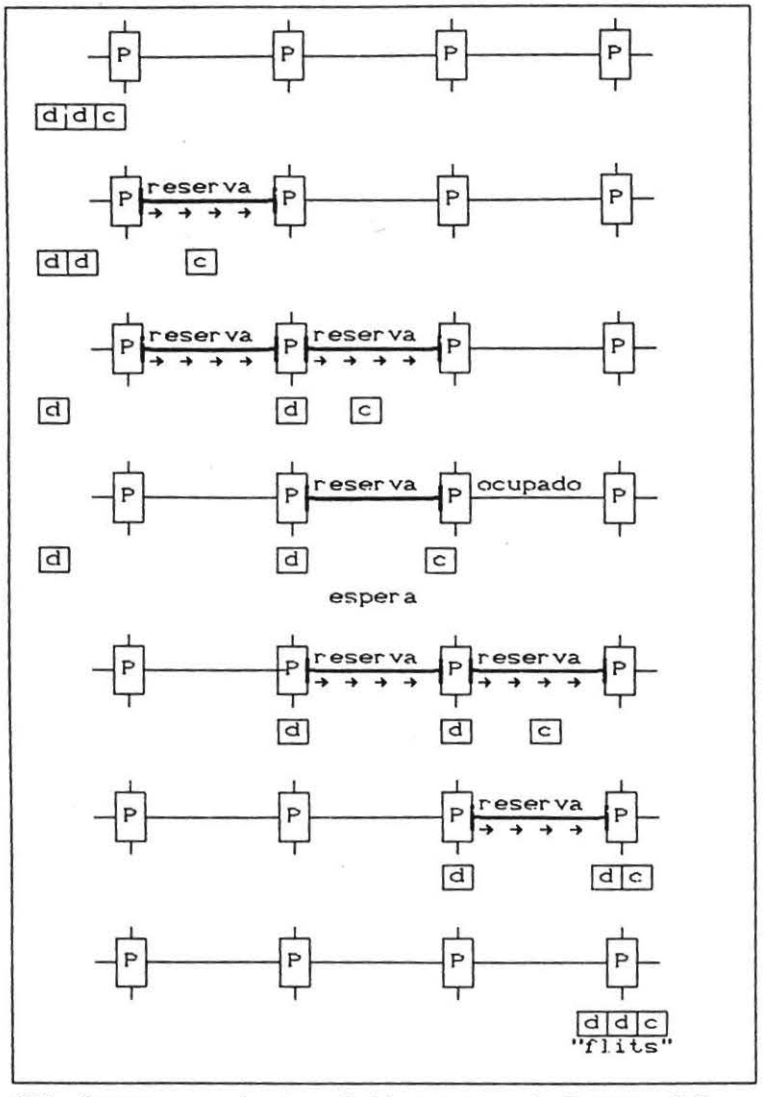


Figura 5 - Rendez-vous

2.4 Wormhole

O **wormhole** apresenta características de reserva de canais. A mensagem a ser enviada é particionada no emissor em segmentos indivisíveis denominados **flits** (**flow control digits** - dígitos de controle de fluxo). Uma mensagem pode

conter um ou mais flits de cabeçalho e um ou mais flits de dados. O processador emissor (sempre de acordo com regras



(Hipótese: canais 1 e 2 livres/canal 3 ocupado)

Figura 6 - Wormhole

ditadas pelo roteamento) determina um canal de saída. Com o canal livre, os flits da mensagem são enviados. Ao chegar em um nó intermediário, estes segmentos são analisados por dispositivos dedicados ao roteamento. Se a linha escolhida estiver livre, os flits são automaticamente comutados. Caso

contrário, são armazenados no nó a espera da liberação do canal. Os demais ficam distribuídos nos nós já percorridos. Os canais, entre o nó que contém o cabeçalho e o que contém o último **flit** da mensagem, estão todos reservados, pois os segmentos de dados não contém informações de roteamento. É possível que o primeiro **flit** da mensagem chegue no destino antes do último ser enviado pelo processador emissor. Quando o primeiro **flit** chega no destino, todos os demais não esperarão por liberação de canais, pois o caminho, do primeiro ao último **flit**, já está todo reservado (figura 6).

3. Modelagem

O **message switching** e o **cut-through** foram modelados analiticamente [KER_79]. Estas estratégias são do tipo **store-and-forward** (armazena-e-transmite) e, como tal, não provocam reserva nem bloqueio simultâneo de recursos. A teoria de filas se adequa bem a este tipo de estratégia. O **wormhole** e o **rendez-vous**, por outro lado, apresentam características de reserva de canais. Este aspecto é de difícil modelagem. São propostos modelos analíticos aproximados. Mas a simulação é também utilizada, tanto para obtenção do tempo de comunicação como também para validação do modelo analítico proposto.

3.1 Modelo Analítico : Message Switching

Este modelo, desenvolvido em [KER_79] considera as seguintes hipóteses: a distribuição do tempo entre chegadas de mensagens no sistema é exponencial, o comprimento da mensagem segue uma distribuição exponencial, a capacidade do nó para armazenar mensagens é infinita, o roteamento é determinístico, os componentes da rede são independentes entre si, a rede está balanceada (utilização homogênea dos recursos) e as ligações são confiáveis (não há necessidade de ACKs). Os parâmetros do modelo são a taxa de chegadas de mensagens (λ), o comprimento médio das mensagens (L), a capacidade média do canal (taxa de transmissão, representado por C) e o número médio de **hops**, ou seja, o número médio de canais a serem percorridos pelas mensagens (nh). Cada canal pode ser considerado como um servidor, ou seja, ele realiza uma tarefa que consiste em transmitir mensagens de um processador a outro. As mensagens são os elementos que impõem uma demanda sobre o servidor. Quando o canal está ocupado, as mensagens esperam. De acordo com este comportamento e com as premissas descritas acima, o modelo de um canal pode ser representado por um sistema de filas M/M/1 (chegada Poisson, serviço exponencial e um servidor). Em cada processador intermediário, a mensagem espera e é transmitida. Este processo ocorre nh vezes:

(1)

$$T = (\text{serviço} + \text{espera})\, n_h = \left(\frac{L}{C} + \frac{\lambda (L / C)^2}{1 - \lambda (L / C)} \right) n_h$$

3.2 Modelo Analítico : Cut-through

As mesmas premissas do **message switching** valem também para o **cut-through**. O parâmetro extra a ser considerado é a razão (α) entre o tempo de transmissão do cabeçalho e o tempo total de transmissão da mensagem. Para cada **cut-through** realizado, um período de transmissão de mensagem é economizado, dado que o tempo de espera foi nulo. Mas como o **cut-through** corresponde a uma transmissão do cabeçalho e não dos dados, efetivamente o ganho corresponde ao número de **cut-throughs** multiplicado pelo tempo de transmissão da parte de dados. O número de **cut-throughs** é dado por:

(2)

$$n_c = (n_h - 1)(1 - \lambda(L / C))$$

Logo a diferença entre o tempo de comunicação no **message switching** e o tempo de comunicação com **cut-through** é dado por:

(3)

$$T_m - T_c = (n_h - 1)(1 - \lambda(L / C))(1 - \alpha)t_m$$

Que se reduz a:

(4)

$$T_c = T_m - (n_h - 1)(1 - \lambda(L / C))(1 - \alpha)t_m$$

3.3 Modelo Analítico : Rendez-vous

A reserva de um caminho entre o processador emissor e o receptor e a espera de outras solicitações pela liberação deste percurso tornam a modelagem desta estratégia um tanto complexa. O tempo médio de comunicação entre dois processadores constitui uma soma de atrasos de transmissão e de espera: tempo de transmissão da solicitação (tsol), tempo médio de espera da solicitação em cada processador pela liberação do canal (W), tempo de transmissão do aceite de comunicação (tack) e tempo de transmissão da mensagem (tmsg). Primeiro, a solicitação percorre a rede e gasta um tempo de espera em cada nó (nh * (tsol + W)). O aceite é então enviado (nh * tack). Finalmente, a mensagem percorre a rede (nh * tmsg).

(5)

$$T = (t_{sol} * n_h) + (W * n_h) + (t_{ack} * n_h) + (t_{msg} * n_h)$$

Os atrasos de transmissão podem ser calculados a partir do tamanho das mensagens trocadas entre os processadores e a

capacidade das linhas de transmissão. Todo o problema está em determinar W, ou seja, o tempo de espera enfrentado pela solicitação a fim de obter um canal de comunicação. Este tempo de espera é calculado segundo o tempo de espera de um sistema M/M/1. O tempo entre chegadas de mensagens é o parâmetro de variação e o tempo de serviço corresponde ao **holding time** (tempo de alocação). O **holding time** é definido como a média aritmética do tempo de alocação das várias comunicações que utilizam aquele canal. A distribuição do tempo entre chegada de mensagens é aproximada por uma exponencial. Assim, a relação entre o **holding time** e o tempo de espera é representado por um sistema de duas variáveis. O W é calculado e substituído em (5).

(6)

$$\begin{cases} TMS = \dfrac{(n_h+1)}{2} t_{sol} + \dfrac{(n_h-1)}{2} W + n_h t_{ack} + \dfrac{(n_h+1)}{2} t_{msg} \\ W = \dfrac{\lambda * TMS^2}{1 - \lambda * TMS} \end{cases}$$

3.4 Modelo Analítico : Wormhole

O **wormhole** é também uma estratégia difícil de ser modelada, pois apresenta situações de bloqueio e de reserva. O tempo de comunicação (T) entre dois processadores é dado pela equação abaixo.

(7)

$$T = n_h(T_f+W) + (n_f-1)T_f$$

Como no **rendez-vous**, o problema também é determinar o tempo médio de espera (W) enfrentado pelos **flits** em cada processador intermediário. Este tempo médio de espera é determinado a partir de um tempo de serviço virtual do canal, que corresponde a um tempo de alocação de cada canal ou **holding time**. Este tempo é considerado como a média aritmética do tempo de alocação das comunicações que reservam aquela ligação (comunicações externas ou oriundas do processador emissor). O modelo consiste em um sistema de equações (nas variáveis Th e W). A hipótese exponencial para a geração de mensagens internamente em um nó é mantida. As expressões do **holding time** e do tempo de espera são dadas pelo sistema abaixo. O W é calculado e substituído na equação acima.

(8)

$$\begin{cases} T_h = \dfrac{(n_h - 1)}{2} W + (n_{flit} * t_{flit}) \\ W = \dfrac{\lambda T_h^2}{1 - \lambda T_h} \end{cases}$$

3.5 Modelos de Simulação : Rendez-vous e Wormhole

As estratégias **rendez-vous** e **wormhole** foram simuladas em GPSS (General Purpose System Simulator) [DON_79]. Considerou-se uma seqüência linear de processadores que geram mensagens e seguem o mesmo comportamento descrito na seção 2. O vetor de processadores é uma topologia adequada, mas apresenta alguns problemas nas extremidades: a utilização dos canais não é uniforme no sistema. O modelo teórico considera uma topologia qualquer, independente do número de processadores da rede. Assim, os resultados obtidos são genéricos, mas passíveis de uma análise preliminar das estratégias.

4. Resultados

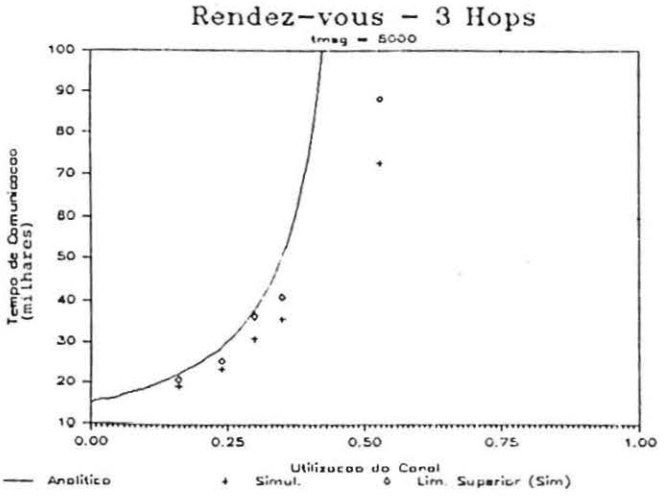
A análise das estratégias foi realizada em duas etapas: compararam-se inicialmente os modelos analíticos do **rendez-vous** e do **wormhole** com os respectivos modelos de simulação. Em seguida, utilizaram-se os quatro modelos analíticos para comparação entre as estratégias.

O **Rendez-vous** e **wormhole** foram simulados (figura 7). Trabalhou-se na faixa de 2 a 3 **hops**, com mensagens de 1.000, 5.000 e 10.000 bits. Para o **rendez-vous**, o tamanho da solicitação é de 5 bits e o tamanho do aceite (ack) é de 1 bit. Para o **wormhole**, o tamanho do **flit** é de 5 bits. Em ambas as estratégias, considerou-se uma linha com capacidade de transmissão de 1 bit/unidade de tempo. Os resultados da simulação mostram que o modelo analítico se apresentou pessimista.

Dados os modelos analíticos descritos anteriormente, as estratégias de comunicação foram comparadas, sob as mesmas condições de carga (figura 8).

5. Análise dos Resultados e Conclusões

As estratégias de comunicação analisadas podem ser classificadas em dois grupos: as bloqueantes (**rendez-vous** e **wormhole**) e as não-bloqueantes (**message switching** e **cut-through**). As estratégias bloqueantes apresentam uma série de desvantagens, tais como sub-utilização de canais e



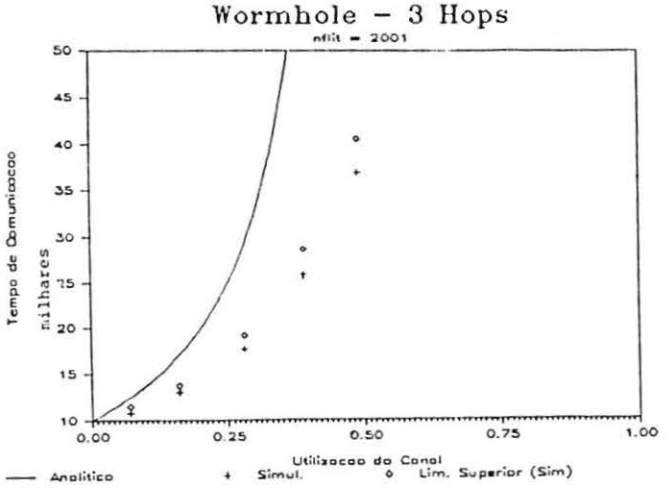


Figura 7 - Simulação x Modelo Analítico

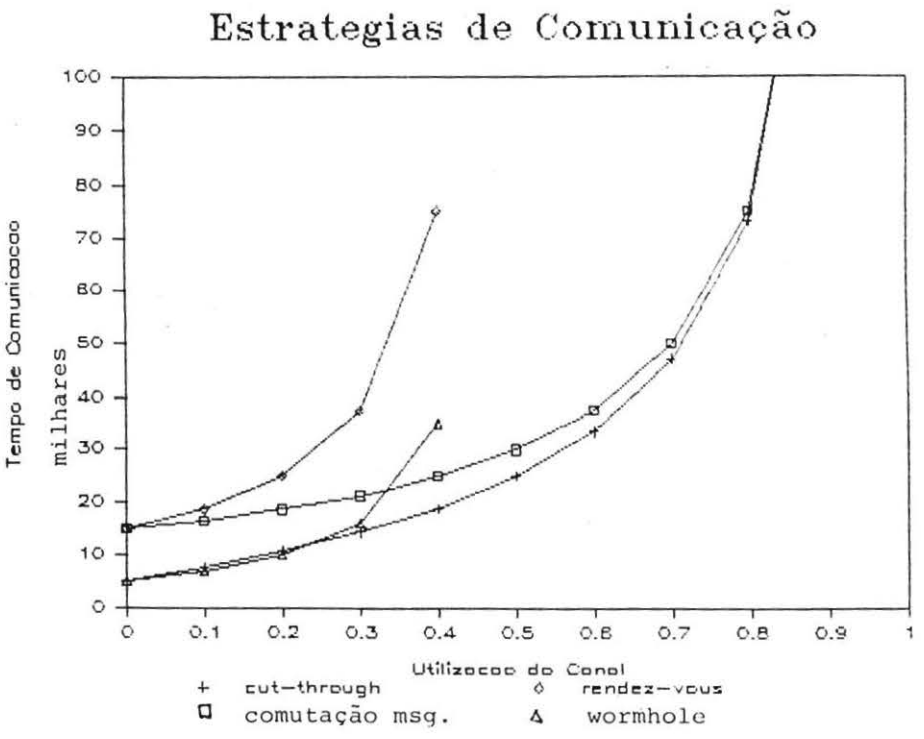


Figura 8 - Estratégias de Comunicação

tendência à ocorrência de **deadlocks**. O **rendez-vous** é a pior estratégia, pois o grau de utilização efetiva dos recursos é pequena: os canais passam a maior parte do tempo alocados, sem nada transmitir. O controle de utilização de canais e de transmissão é complexo e o tempo de processamento pode ser considerável. O **wormhole**, por apresentar um certo paralelismo de comunicação, apresenta uma utilização mais eficiente dos recursos. Em casos de baixo grau de troca de mensagens, o desempenho desta estratégia pode ser melhor do que o das estratégias não-bloqueantes. A utilização de hardware específico também torna a comunicação mais eficiente. Em ambas as estratégias acima, o espaço necessário para armazenamento de mensagens em um nó processador não é consideravelmente grande, pois no **rendez-vous** há armazenamento de uma solicitação por comunicação e no **wormhole**, há armazenamento de um **flit** por comunicação. O tratamento de **deadlocks** deve ser adequado. Uma maneira de tornar o **rendez-vous** mais eficiente é utilizar hardware específico para comutação, o que poupa transmissões desnecessárias de aceites e de mensagens: um circuito único seria fechado entre origem e destino.

As estratégias não bloqueantes não apresentam bloqueios causados pela espera mútua de canais. O **cut-through**, por considerar que há chances da mensagem comutar sem atrasos, apresenta melhor desempenho, se o grau de utilização do sistema é baixo. Para utilizações maiores, esta estratégia se iguala ao **message switching** pois o número de **cut-throughs** diminui para zero no limite (sistema saturado - utilização de 100%). A desvantagem das estratégias não-bloqueantes recai na memória necessária para armazenamento de mensagens que esperam por canais de comunicação. Este espaço deve prever o enfileiramento de uma mensagem inteira por comunicação. O **message switching** depende ainda de software para realizar a comutação, tornando esta estratégia ineficiente se forem considerados os atrasos de processamento. O **cut-through** realiza comutações de forma mais eficiente através de hardware específico. O interesse crescente por estratégias do tipo **wormhole** e **cut-through** reside no fato de que estas estratégias são, de certa maneira, independentes do número de **hops**. Em sistemas de baixa utilização, isto se traduz em tempos de latência significativamente menores.

Conclue-se que as estratégias não bloqueantes são as mais adequadas para a comunicação entre processos em um multiprocessador fracamente acoplado. Das bloqueantes, apenas a **wormhole** apresenta melhor desempenho sob condições de baixa utilização de canais.

Bibliografia

[BER_89] BERTSEKAS, Dimitri P. & TSITSIKLIS, John N. Parallel and distributed computation: numerical methods. Englewood Cliffs, Prentice-Hall, 1989.

[BOK_84] BOKHARI, Shahid H. On the mapping problem. IEEE Transactions on Computers, New York, 30(3):207-14, Mar. 1981.

[DAL_87] DALLY, William J. & SEITZ, Charles L. Deadlock-free message routing in multiprocessor interconnection networks. IEEE Transactions on Computers, New York, 36(5):547-53, May 1987.

[DON_79] O'DONOVAN, Thomas M. GPSS : simulation made simple. Chichester, Wiley, 1979.

[EAG_86] EAGER, Derek L.; LAZOWSKA, Edward D. & ZAHORJAN, John. A comparison of receiver-initiated and sender-initiated adaptive load sharing. Performance Evaluation, Amsterdam, 6(1):53-67, Mar. 1986.

[HWA_84] HWANG, Kai & BRIGGS, Fayé A. Computer architecture and parallel processing. New York, McGraw-Hill, 1984.

[MAR_86] MARSAN, M. Ajmone; BALBO, G. & CONTE, G. Performance models of multiprocessor systems. Cambridge, The MIT Press, 1986.