Um Sistema de Comunicação para Multicomputadores

Gustavo Peixoto de Azevedo

Núcleo de Computação Eletrônica
Universidade Federal do Rio de Janeiro
Caixa Postal 2324, Rio de Janeiro - RJ, CEP 20001
Telefone: (021) 598 3160

Resumo

Este artigo apresenta uma proposta de um sistema de comunicação para Multicomputadores de grande escala. O sistema proposto tem por objetivo principal realizar a comunicação entre as tarefas que compõem uma Aplicação Paralela Distribuída em execução em um Multicomputador.

A principal colaboração deste trabalho é a proposição um novo protocolo para a comunicação entre tarefas, que permite a migração das tarefas e é adequado a Multicomputadores de grande escala.

São apresentadas as propriedades fundamentais para um sistema de comunicação de um Multicomputador, as medidas de desempenho para a sua avaliação, a sua estrutura geral, soluções anteriores e finalmente as soluções adotadas neste trabalho, com ênfase especial para o novo protocolo proposto para a comunicação entre tarefas migrantes.

O novo protocolo para comunicação entre tarefas é definido, verificado quanto à sua correção e quanto à sua qualidade com relação à garantia destas propriedades fundamentais.

Abstract

This article presents a proposal of a communication system for large scale Multicomputers. The system's main objective is to provide the communication between the tasks that make up an Parallel Distributed Application running on a multicomputer.

The main contribuition of this work is the proposal of a new protocol that suports intertask communication between migrating tasks on a large scale multicomputer.

This article presents a multicomputer communicating system's fundamental properties, performance meters, basic structure, previous solutions and finally the solutions adoped in this work with special enfasis on the new protocol proposed for the communication between migrating tasks.

The new protocol for intertask communication is defined, its correction is verified and its quality is checked against the communication system's fundamental properties.

Gustavo Peixoto de Azevedo é M.Sc. pela COPPE/UFRJ no Programa de Engenharia de Sistemas e Computação em 1992, Bacharel em Informatica pela UFRJ em 1989 e Pesquisador do Núcleo de Computação Eletrônica da UFRJ. Suas áreas de interesse são: Sistemas Distribuídos Corporativos, Sistemas Operacionais, Arquiteturas Paralelas e Escalonamento Dinâmico Distribuído.

1 Apresentação

Daniel Reed [16] define o termo **multicomputador** (MC) para designar uma classe de sistemas distribuídos com um grande número de **nós computacionais**, interconectados em uma topologia regular determinada por **ligações ponto a ponto**, que cooperam assincronamente através da troca de mensagens para executar as tarefas de um programa paralelo. Cada nó computacional, fabricado com um pequeno número de circuitos VLSI, contém um **processador computacional**, um **módulo de memória local** e um **processador de comunicações** capaz de rotear as mensagens sem atrazar o processador computacional.

Uma aplicação para um MC, aqui denominada como **aplicação paralela distribuída** (APD), é um conjunto de unidades de execução seqüencial, denominadas **tarefas**, que executam paralelamente e se comunicam através da troca de mensagens, colaborando para a resolução de um mesmo problema.

Este artigo propõe um sistema de comunicação para os MC. Inicialmente (seções 2 e 3) são revistas as propriedades fundamentais requeridas e as medidas de desempenho dos sistemas de comunicação para MC. A seção 4 apresenta a estrutura geral dos sistemas de comunicação para MC, a partir de uma análise dos principais problemas relacionados ao seu projeto, com ênfase para o problema da realização da comunicação entre tarefas migrantes. A subseção 4.5 analisa as soluções anteriores para o problema da realização da comunicação entre tarefas migrantes. A seguir a seção 5, apresenta as soluções adotadas neste trabalho, com ênfase especial para o novo protocolo proposto para a comunicação entre tarefas migrantes. Nas subseções seguintes são apresentadas a definição deste protocolo (5.1), a verificação da sua correção (5.2) e a verificação da sua qualidade com relação a garantia das propriedades definidas na seção 2.

2 Propriedades Requeridas

Um sistema de comunicações para um MC deve implementar a troca de mensagens entre as tarefas que constituem uma APD, observando as seguintes propriedades:

1. **Invisibilidade da localização das tarefas.** A localização das tarefas, ou seja, os computadores em que elas estão executando, deve ser totalmente invisível à aplicação.
2. **Integridade das mensagens.** A transmissão deve ser realizada sem alterações nas mensagens.
3. **Unicidade das mensagens.** A transmissão deve ser realizada sem duplicação de mensagens.
4. **Ordenamento parcial das mensagens.** Para uma transmissão entre um par qualquer de tarefas, as mensagens devem ser recebidas na ordem em que foram enviadas.
5. **Eficácia.** O atraso para a entrega de uma mensagem tem que ser finito, mesmo que não seja determinístico. Isto significa que todas as mensagens tem que ser entregues.

A maioria dos algoritmos distribuídos, a serem implementados pelas APD, supõem algumas ou todas estas propriedades, sejam elas proporcionadas pelo sistema a nível de hardware ou de software de controle.

3 Medidas de Desempenho

Quanto ao desempenho, os objetivos de um sistema de comunicação são minimizar a latência e maximizar a vazão de mensagens entre as unidades de execução.

Latência é o tempo médio necessário a entrega de uma mensagem pelo sistema de comunicação. É medida pelo intervalo de tempo entre o pedido de transmissão da mensagem, a partir do computador fonte, até a recepção da mensagem no computador destino.

A **vazão** é a taxa máxima de mensagens entre tarefas que pode ser atendida pelo sistema de comunicação. Se a taxa de mensagens submetida ao sistema de comunicação for superior a sua vazão, o sistema de comunicação satura-se e a latência aumenta indefinidamente.

4 Estrutura Geral

De acordo com a seção 2 o sistema de comunicação de um MC proporciona à APD a abstração da troca de mensagens entre tarefas. No caso da comunicação entre duas tarefas localizadas em computadores distintos, a troca de mensagens entre as tarefas corresponde à transmissão de mensagens entre os computadores. Considerando-se que os MC normalmente não são completamente interconectados, dois computadores que se comunicam não estão necessariamente diretamente conectados. As mensagens entre os computadores de um MC têm então, em geral, que trafegar por vários canais de comunicação e eventualmente serem processadas por vários computadores intermediários até alcançar o seu destino final.

O projeto de um sistema de comunicação para um MC depende das decisões quanto à solução de uma série de problemas relacionados hierarquicamente. Há desde problemas que envolvem características físicas e construtivas de um MC até problemas lógicos. A definição da forma de funcionamento dos canais de comunicação e da topologia do MC são exemplos de problemas do primeiro grupo. Este trabalho aborda os seguintes problemas lógicos:

- **Comutação dos canais de comunicação.** Estabelece a forma de transmissão de mensagens por um canal de comunicação. Implementa a comunicação entre computadores vizinhos.
- **Roteamento.** Método para a determinação das possíveis seqüências de canais de comunicação entre cada par de computadores. Implementa a comunicação entre um par qualquer de computadores, transformando a rede de conexão em uma rede de comunicação.
- **Controle de fluxo.** Método para o controle do tráfego na rede de comunicação. Implementa uma política de utilização dos canais de comunicação, gerenciando os conflitos para o acesso aos canais de comunicação e garantindo o progresso das mensagens.
- **Comunicação entre tarefas migrantes.** Protocolo para o roteamento de mensagens entre tarefas migrantes. Implementa a comunicação entre tarefas que podem migrar a qualquer momento.

As subseções seguintes abordam cada um destes problemas.

4.1 Comutação dos Canais de Comunicação

Um dos problemas fundamentais do projeto de uma rede qualquer de computadores é a escolha de um método de comutação dos canais de comunicação. A escolha do método implica na definição da unidade de informação que pode ser enviada e a forma de transmissão desta unidade. Há três métodos básicos:

- **Comutação de circuitos.** A unidade de informação enviada é uma mensagem. Inicialmente é estabelecido um caminho completo que conecte o computador emissor ao computador destinatário. Este caminho pode ser composto por mais de um canal de comunicação e implica na alocação simultânea, exclusiva e estática de todos os recursos

da rede de comunicação necessários à transmissão da mensagem. Após o estabelecimento deste caminho, a mensagem é enviada diretamente ao computador destinatário. A seguir, o caminho é desfeito, com a liberação dos recursos alocados para a transmissão.

- **Comutação de mensagens.** A unidade de informação enviada também é uma mensagem. Este método, também denominado *"datagram"*, tem por objetivo tornar desnecessária a alocação simultânea de mais de um canal de comunicação para a transmissão de uma mensagem. A mensagem sofre transmissões sucessivas a computadores cada vez mais próximos do destinatário, até alcançá-lo. Se não há um canal de comunicação que conecte diretamente o computador emissor ao destinatário, um canal de comunicação até um computador mais próximo do computador destinatário é alocado dinamicamente e a mensagem inteira trafega por este canal. Este processo se repete sucessivamente até que a mensagem alcance o seu destino. Este método aumenta o tempo disponível dos canais de comunicação, no entanto também aumenta o tempo mínimo de latência de uma mensagem, devido a três fatores: (1) a sua transmissão pelo próximo canal de comunicação só se inicia após ela ter sido inteiramente recebida pelo computador intermediário, (2) a quantidade de dados de uma mensagem tipicamente é muito maior que o necessário ao estabelecimento de um circuito e (3) muito maior que os dados em transmissão por um canal a um dado instante.
- **Comutação de pacotes.** A motivação deste método é a diminuição do tempo de latência de uma mensagem. As mensagens são divididas em pacotes. Os pacotes têm endereçamento próprio e são transmitidos independentemente e da mesma forma que as mensagens no método de comutação de mensagens. Apesar de terem a mesma origem e destino, os caminhos percorridos pelos pacotes de uma mesma mensagem podem ser diferentes. A latência diminui porque em cada computador intermediário só é necessária a espera de um pacote, ao invés de toda a mensagem e porque os pacotes podem utilizar caminhos paralelos. Como contrapartida, os pacotes ao serem recebidos precisam ser reordenados e são necessárias mais decisões de roteamento, uma por pacote em cada computador intermediário.

Os MC normalmente utilizam a comutação de mensagens. Estas no entanto são divididas em pacotes ou em partes menores, que trafegam seqüencialmente pelo mesmo caminho.

4.2 Roteamento

Um método de roteamento define o próximo canal de comunicação a ser percorrido por uma mensagem a fim alcançar o seu destino. Os métodos de roteamento podem ser classificados como determinísticos, oblívios, ou adaptativos, de acordo com o tipo de informações utilizadas para a realização da decisão de roteamento.

Em um método de roteamento **determinístico** o caminho percorrido por um pacote depende apenas dos computadores de origem e de destino da mensagem. A utilização de um método determinístico pode apresentar como vantagem a ordenação dos canais de comunicação utilizados, que é útil para se evitar *"deadlocks"*.

Um método **oblívio** pode escolher um dentre vários caminhos diferentes para a realização de uma comunicação. Esta escolha, no entanto não se baseia no estado de utilização dos canais de comunicação.

Em um método de roteamento **adaptativo**, a escolha do caminho percorrido por um pacote ou mensagem é função das informações referentes ao estado corrente do sistema de comunicação. Este tipo de método apresenta duas vantagens:

- Se um caminho está sobrecarregado com o tráfego de mensagens, um outro caminho pode ser escolhido reduzindo a latência da mensagem.
- Se um caminho tem um canal defeituoso, outro caminho pode ser utilizado preservando a continuidade das comunicações.

4.3 Controle de Fluxo

A política de controle de fluxo resolve as disputas eventuais entre as mensagens em trânsito, quanto à utilização dos recursos do sistema de comunicação. O controle de fluxo é realizado por um método de gerenciamento dos recursos do sistema de comunicação, que aloca estes recursos aos dados a serem transmitidos, e tem por objetivo garantir o progresso das mensagens em direção ao seu destino.

A política adotada para o controle de fluxo de um sistema de comunicação é fundamental para o seu desempenho geral, determinando a vazão e a latência de um sistema de comunicação. Esta subseção aborda os principais métodos para o controle de fluxo de um sistema de comunicações.

Quando há uma disputa pela utilização de um mesmo canal de comunicação por mais de uma mensagem em trânsito ocorrem três questões:

1. Alguma das mensagens consegue utilizar o canal ?
2. Em caso positivo, qual das mensagens ?
3. O que acontece com as outras mensagens ?

Quanto as duas primeiras questões a solução é simples. A mensagem que primeiro solicitar um canal de comunicação disponível o obtém e o utiliza até a conclusão da sua transmissão.

Com relação ao que acontece com cada mensagem que não consegue utilizar o canal de comunicação que precisa, há quatro estratégias básicas:

- **Armazenamento.** A mensagem pode ser armazenada no computador de origem do canal de comunicação solicitado, enquanto aguarda pelo direito à sua utilização. A vantagem é o aproveitamento integral da capacidade dos canais de comunicação. Em compensação, esta estratégia requer uma grande capacidade de armazenamento e cuidados especiais no armazenamento para se evitar eventuais *"deadlocks"* decorrentes da limitação da capacidade de armazenamento.
- **Bloqueio.** Ao invés de ser armazenada, a mensagem tem a sua recepção suspensa até que ela consiga o direito de utilizar o canal de comunicação que precisa. Deste modo, o canal de comunicação de onde ela provém é bloqueado, havendo uma perda no aproveitamento da capacidade do canal. Há ainda a possibilidade de *"deadlocks"* se mais de um canal está bloqueado simultaneamente. A vantagem é a limitação da capacidade de armazenamento necessária, possibilitando a construção de um controlador de comunicações mais simples e rápido.
- **Eliminação.** A mensagem pode ser simplesmente eliminada e o computador de origem da mensagem é informado. O resultado pode ser um grande desperdício da capacidade dos canais e a instabilidade do sistema. Outra desvantagem desta estratégia é a necessidade de confirmação do progresso da mensagem.
- **Roteamento errado.** A mensagem é roteada para um outro canal de comunicação que esteja disponível, assumindo-se que a mensagem retornará algum tempo depois, quando então novamente tentará utilizar o canal de comunicação originalmente solicitado, ou assumindo-se que a mensagem conseguirá alcançar o seu destino através de um outro

caminho alternativo. Esta estratégia desperdiça a capacidade dos canais ao afastar as mensagens do seu destino.

Estas estratégias estão relacionadas aos principais métodos utilizados em MC para o controle de fluxo. Há quatro métodos principais: *"roteamento desesperado"*, *"store and forward"*, *"virtual cut-through"* e *"wormhole"*.

O método conhecido por **"roteamento desesperado"** baseia-se na estratégia de roteamento errado. Este método pode resultar em *"livelocks"*, quando mensagens não conseguem progredir até o seu destino. Ele foi utilizado em alguns dos primeiros MC como o Denelcor HEP e a Connection Machine [2]. Este método não deve ser mais considerado para os MC atuais, já que não é eficiente nem eficaz, pois desperdiça a banda passante do sistema de comunicação e mesmo assim não garante o progresso das mensagens e portanto a entrega delas.

O método **"store and forward"** se baseia na estratégia de armazenamento e na comutação de pacotes ou de mensagens. Cada mensagem ou pacote é recebido integralmente e armazenado antes de ser transmitido pelo próximo canal. Este método pode apresentar problemas de *"deadlock"*.

O método denominado **"virtual cut-through"**, criado por Kermani e Kleinrock [7], tem por objetivo minimizar o tempo de latência das mensagens. Se o próximo canal a ser percorrido por uma mensagem estiver disponível a mensagem é enviada diretamente. Em caso contrário ela é recebida e armazenada, como no método *"store and forward"*.

Os métodos que utilizam o armazenamento das mensagens, como *"store and forward"* e *"virtual cut-through"*, apresentam como vantagem a possibilidade de aproveitamento total da banda passante dos canais de comunicação. Esta vantagem tem como custo a necessidade de armazenamento das mensagens que não podem prosseguir. Como a capacidade de armazenamento não é ilimitada é possível a ocorrência de *"deadlocks"* quando uma mensagem não puder prosseguir, o que liberaria os buffers onde está armazenada, porque não há buffers disponíveis no próximo computador, que também não são liberados pela mesma razão. Ocorrerá uma situação de *"deadlocks"* se computadores nesta situação formarem pelo menos um círculo de dependência.

A ocorrência de *"deadlocks"* nestes métodos pode ser evitada através da estruturação dos buffers de armazeamento em níveis. Nestas soluções, no entanto, a demanda por buffers tende a crescer com o número de computadores interligados, o que não é desejável para um MC escalável.

O método **"wormhole"**, apresentado por Seitz et al [17], está relacionado à estratégia de bloqueamento. Uma mensagem é decomposta em **flits** (*"flow control digits"*), que é a menor unidade de dados reconhecida pelo mecanismo de controle de fluxo. Normalmente, uma mensagem proveniente de um canal de entrada é transmitida a medida que é recebida, flit a flit, para o próximo computador, através de um canal de saída, e assim sucessivamente até o computador destinatário. Os flits de uma mensagem podem assim estar distribuídos por vários computadores ao longo da trajetória percorrida pela mensagem.

Os canais físicos de comunicação são compartilhados por canais virtuais de comunicação. Enquanto um canal virtual está bloqueado, um outro canal virtual, que compartilha do mesmo canal físico, pode estar ativo utilizando o canal físico comum. Assim, é possível evitar a perda da banda passante dos canais de comunicação. Um canal virtual mantém-se alocado a transmissão de uma mensagem até que todos os flits da mensagem tenham fluído por ele.

A implementação dos canais virtuais é bem simples. A um canal físico é acrescido um circuito de comutação e para cada canal virtual implementado, é acrescido um buffer capaz de armazenar apenas o próximo flit a ser enviado, ou o flit recebido mais recentemente. Assim

a demanda por capacidade de armazenamento em cada computador é proporcional ao seu número de canais de comunicação, e não ao número de computadores em todo o sistema.

A ocorrência de *"deadlocks"* na utilização do método de *"wormhole"* pode ser prevenida. Basta restringir as possíveis combinações entre os canais virtuais de entrada e de saída, de modo a estabelecer uma ordem entre os canais virtuais e assim garantir a inexistência de ciclos com os canais virtuais. Esta solução também restringe os caminhos alternativos entre os computadores, o que diminui a adaptabilidade do sistema de comunicação. Há alternativas para a implementação de *"wormhole"* adaptativo [8].

Como descrito na seção 3 o desempenho de um sistema de comunicação é dado pela sua latência e pela sua vazão. O método de controle de fluxo determina estas características quanto a comunicação entre computadores. Assim, a escolha do método de controle de fluxo a adotar deve ser baseada na análise destas medidas relativas a estes métodos.

Uma análise completa da latência de um método de controle de fluxo é muito complexa pois envolve muitas variáveis e pouco precisa pois os valores atribuídos a estas variáveis podem ser mal avaliados ou pouco representativos. Este trabalho apresenta uma análise mais simples, mas que já é bastante ilustrativa.

A análise da latência destes métodos pode ser realizada a partir da definição de alguns parâmetros independentes. Seja L o número de bits de uma mensagem; T_C o tempo de um ciclo do canal de comunicação, ou seja, o tempo necessário a transmissão de um flit; W o número de bits de um flit; e N o número de canais que uma mensagem deve percorrer. Como simplificação, assuma-se que os canais estão sempre disponíveis.

No método *"store and forward"* uma mensagem deve ser recebida integralmente antes de ser enviada pelo próximo canal. Assim para este método, a latência é dada pelo produto do tempo necessário para que a mensagem percorra integralmente um canal, pelo número de canais a serem percorridos, ou seja:

$$T_{SF} = \left(\frac{L}{W} \times T_C\right) \times N \quad ou \quad T_{SF} = T_C \times \left(\frac{L}{W} \times N\right)$$

Assumindo-se que os canais permanecem disponíveis, a latência do método *"virtual cut-through"* é igual a do método *"wormhole"*. Nestes métodos uma mensagem é enviada integralmente, sob a forma de uma seqüência de flits e cada flit recebido da mensagem já pode ser enviado pelo próximo canal, sem aguardar pela recepção dos posteriores. Para estes métodos, que funcionam como um *"pipeline"*, a latência é dada pela soma do tempo necessário a que um flit percorra todos os canais e alcance o seu destino, mais o tempo necessário a que todos os flits iniciem sua transmissão, ou seja:

$$T_{WH} = N \times T_C + \frac{L}{W} \times T_C \quad ou \quad T_{WH} = T_C \times \left(\frac{L}{W} + N\right)$$

Comparando-se as expressões constata-se que os métodos *"wormhole"* e *"virtual cut-trough"* são muito superiores ao método *"store and forward"* quanto ao tempo de latência. Quanto a vazão, os métodos se equivalem, pois em todos eles é possível o total aproveitamento da banda passante dos canais de comunicação.

4.4 Comunicação entre Tarefas Migrantes

O objetivo primordial de um sistema de comunicação é propiciar a comunicação entre as diversas entidades do sistema. Nos MC onde ocorre a criação dinâmica de tarefas, é fundamental a comunicação entre as tarefas e estas podem migrar. Assim é necessária a obediência a um protocolo que garanta a troca de mensagens, não apenas entre computadores distintos, mas entre tarefas migrantes.

Os métodos para controle de fluxo têm por objetivo realizar a comunicação entre cada par de computadores de um MC. A capacidade de migração das tarefas distingue o problema da comunicação entre computadores de um MC, do problema da comunicação entre tarefas migrantes, em razão das seguintes diferenças:

- *Comunicação entre computadores de um* MC. Os computadores e os canais de comunicação de um MC são definidos estaticamente. Assim os caminhos entre cada par de computadores permanecem inalterados durante toda a execução da APD.
- *Comunicação entre as tarefas de uma* APD. As tarefas são criadas dinamicamente de acordo com a evolução da execução da APD e podem migrar a qualquer instante. Conseqüentemente, os caminhos para comunicação entre tarefas se alteram durante a execução da APD e, portanto, precisam ser redefinidos dinamicamente.

Um dos muitos problemas a ser resolvido é como garantir as propriedades definidas na seção 2, quando as tarefas comunicantes estão em migração. Neste caso, as mensagens são endereçadas a alvos em movimento. Um sistema de comunicação deve prover um esquema de migração que realize o roteamento das mensagens com a recuperação e o rerroteamento das mensagens perdidas.

Este trabalho propõe na subseção 5.1 um novo protocolo para a comunicação entre tarefas migrantes em um MC.

4.5 Trabalhos Anteriores

Esta subseção apresenta os principais trabalhos anteriores quanto a protocolos para a comunicação entre tarefas migrantes.

Um protocolo para a migração de processos e o encaminhamento de mensagens foi implementado pela primeira vez em um sistema distribuído no sistema operacional DEMOS/MP[14]. Esta implementação não previa a execução concorrente de várias instâncias do protocolo no mesmo computador. Isto significa que um computador somente poderia realizar uma migração por vez.

Fowler [4][5] propôs o protocolo *"forwarding address"* para localizar objetos que se moviam em um sistema distribuído. Este protocolo assume que cada computador tem uma tabela completa com um endereço recente ou não, de cada tarefa no sistema. Nos protocolos do tipo *"forwarding address"* as mensagens são enviadas em direção ao provável endereço da tarefa destinatária. Se quando a mensagem chegar, a tarefa já houver migrado, a mensagem é enviada a um endereço mais atual, e assim sucessivamente até alcançar a tarefa destinatária. Os endereços das tarefas nos computadores são atualizados passivamente a medida que a mensagem trafega. Este protocolo não prevê uma atualização ativa dos endereços das tarefas que migram.

A principal vantagem da abordagem de Fowler é que as entradas de roteamento não são atualizadas nos computadores não relacionados às trajetórias percorridas pelas mensagens. Uma desvantagem é que não há um mecanismo que garanta a atualização dos endereços das tarefas que migraram. Estes endereços podem ficar muito desatualizados, prolongando os

caminhos percorridos pelas mensagens. Outra desvantagem, é a necessidade do conhecimento prévio do endereço inicial de todas as tarefas por todos os computadores.

No sistema Emerald [6] foram implementadas tanto a mobilidade de tarefas como de dados. Para possibilitar a localização dos objetos pelos computadores, o sistema incorpora o protoloco *"forwarding address"* de Fowler. Quando não se conhece o endereço de um objeto, o sistema Emerald difunde uma mensagem por todo o sistema solicitando o endereço do objeto. A principal desvantagem deste esquema é a necessidade da difusão de mensagens por todo sistema, o que o torna não eficiente.

Nos sistemas operacionais Locus [13] e Sprite [3][12] as tarefas interagem através de chamadas ao sistema. A cada tarefa é atribuído um computador especial, para o qual são enviadas todas as chamadas remotas relativas à tarefa. Este computador é encarregado de enviar as chamadas remotas para as tarefas a partir de uma estimativa para a localização atual da tarefa. Esta estratégia apresenta vários problemas:

- Há uma dependência da tarefa em relação ao seu computador especial, mesmo após ter migrado.
- Em MC com um grande número de computadores o comprimento adicional do caminho a ser percorrido pelas mensagens, necessário à passagem pelo seu computador especial, pode ser muito maior do que a distância efetiva entre as tarefas envolvidas na chamada remota.
- A passagem obrigatória por um computador especial, estaticamente associado a uma tarefa, de todas as mensagens endereçadas a esta tarefa, torna a migração de tarefas ineficaz para reduzir as distâncias de comunicação.

O protocolo de roteamento de mensagens do sistema V [18] utiliza um mecanismo de cache para a localização de tarefas e o protocolo de *"forwarding address"* para o envio das mensagens. Se a localização correta de uma tarefa não está disponível no cache de um computador, uma mensagem é difundida por todo o sistema requerendo a localização atual da tarefa. A grande desvantagem deste esquema é a realização de um número indeterminável de difusões de mensagens por todo o sistema.

O mecanismo de migração de tarefas proposto por Lu, Chen e Liu [9] assume que cada tarefa se comunica apenas com um número limitado de outras tarefas, conhecidas a priori, denominadas tarefas adjacentes. Cada computador tem uma tabela com a localização das tarefas adjacentes às tarefas localizadas no computador. A migração de uma tarefa é precedida pelo bloqueio da transmissão de mensagens endereçadas a esta tarefa; após a migração, o novo endereço da tarefa é difundido para os computadores com tarefas adjacentes e as transmissões para a tarefa são desbloqueadas. As desvantagens deste esquema são a impossibilidade do envio de mensagens para as tarefas em migração, e a restrição ao grupo de tarefas com que uma tarefa pode se comunicar.

Ravi [15] apresenta uma especificação formal de um protocolo para a comunicação e migração de tarefas, em sistemas distribuídos ponto a ponto. A correção do protocolo também é apresentada. Este protocolo, no entanto, não garante uma das propriedades definidas na seção 2. O protocolo não garante que a ordem de recepção das mensagens entre duas tarefas será a mesma ocorrida na transmissão.

Não há assim uma solução perfeita para o problema. Na seção seguinte é proposta mais uma solução, que não impõe restrições a comunicação entre as tarefas, e que não requer a difusão de mensagens por todo o sistema, o que é especialmente importante para um melhor desempenho em sistemas com muitos computadores.

5 Sistema de Comunicação Proposto

Esta seção explica o funcionamento do sistema de comunicação proposto para um MC. São descritos e explicados as estruturas de dados utilizadas e os algoritmos executados para o roteamento das mensagens pelo sistema.

Um objetivo fundamental do sistema de comunicação de um MC é propiciar a comunicação entre tarefas de forma eficaz e eficiente. Como uma tarefa pode migrar a qualquer momento, inclusive quando há mensagens já transmitidas em sua direção, e como o conhecimento da ocorrência de uma migração não é imediato em todos os computadores de um sistema distribuído, então o roteamento de mensagens entre tarefas tem que se basear em informações sobre a localização das tarefas que podem estar desatualizadas.

Para ser eficiente, um protocolo para a comunicação entre tarefas migrantes deve ser capaz de redefinir a trajetória de uma mensagem ao longo da sua transmissão, sempre que se conheça uma trajetória melhor que a prevista. O método de controle de fluxo a ser adotado deve permitir a redefinição da trajetória de uma mensagem mesmo antes que ela alcance o computador ao qual ela inicialmente se destinava.

A livre redefinição da trajetória de uma mensagem em trânsito não é possível no método de controle de fluxo *"wormhole"*. A não ocorrência de *"deadlocks"* neste método depende do ordenamento dos canais da trajetória completa da mensagem. A trajetória estabelecida não pode ser alterada sob o risco de desobedecer a este ordenamento. Assim, este método não é adequado para a realização da comunicação entre tarefas migrantes.

O método de controle de fluxo a ser adotado é o *"virtual cut-through"*. Este método independe da trajetória efetivamente percorrida pela mensagem e apresenta as melhores características de desempenho, que são equivalentes as do método *"wormhole"*. O método de comutação dos canais de comunicação é o *"message switch"*, que é adequado a comunicação intermitente que se espera das tarefas de uma APD e ao método *"virtual cut-through"*. O método de roteamento a ser utilizado decorre da topologia do sistema MC utilizado, mas é adaptativo.

Toda comunicação entre tarefas é realizada por processadores de comunicação (PC). Quando uma tarefa decide enviar uma mensagem a outra tarefa, a tarefa emissora requer este serviço ao PC do computador no qual ela está localizada.

5.1 Comunicação entre Tarefas Migrantes

Esta subseção apresenta o protocolo desenvolvido para realizar a comunicação entre tarefas migrantes em um sistema MC.

As mensagens podem ser classificadas em mensagens do escalonamento global, mensagens do mecanismo de roteamento e mensagens da aplicação. Esta seção aborda as mensagens do mecanismo de roteamento e as mensagens da aplicação.

As mensagens entre os PC são:

- INTER: transmite uma mensagem entre tarefas.
- LOCAL: transmite a atual localização de uma tarefa.

Estas mensagens contêm um bloco de controle incluindo a identificação, localização e grau de atualidade da localização, tanto da tarefa emissora e como da destinatária. O grau de atualidade de uma localização é igual a um mais o número de migrações já ocorridas com a tarefa, quando ela apresentava esta localização.

O mecanismo de roteamento de mensagens entre tarefas utiliza o conhecimento aproximado da localização da tarefa destinatária. O PC da tarefa emissora envia a mensagem em direção ao computador que segundo o seu conhecimento é a localização mais recente da tarefa destinatária.

A primeira aproximação para a localização de uma tarefa é definida estaticamente, de acordo com a identificação da tarefa. Há um conjunto de computadores, distribuídos uniformemente pelo MC, associado a cada tarefa. A escolha dos membros deste conjunto depende da topologia particular do muticomputador disponível. Esta escolha, no entanto, é definida de tal forma que a partir da identidade de uma tarefa seja possível, a qualquer PC, determinar o computador mais próximo associado a tarefa, que será conhecido como a localização da tarefa com grau de atualidade igual a zero.

Quando uma tarefa é criada o conjunto de computadores associado a ela é informado da sua localização através de uma mensagem LOCAL. Os PC armazenam as mensagens endereçadas às tarefas associadas ao seu computador até a recepção de uma mensagem LOCAL, informando a criação e a localização de uma destas tarefas, quando então, as mensagens relativas a tarefa cuja localização foi comunicada são enviadas.

A atualização do conhecimento da localização de uma tarefa só pode ocorrer de duas formas:

- quando um computador for o destinatário, ou pertencer ao caminho percorrido por uma mensagem que contenha uma localização mais atual da tarefa que a disponível neste computador e
- quando um computador participa da migração de uma tarefa, seja como transmissor ou como receptor.

Uma das informações de estado da tarefa é o número de migrações em que já participou. Quando ocorre a migração de uma tarefa, este número é incrementado. Um computador ao receber uma tarefa passa a conhecer o grau de atualidade desta localização da tarefa a partir deste número de migrações. O grau de atualidade é igual ao número de migrações da tarefa, mais um.

As mensagens são inicialmente enviadas em direção a localização conhecida da tarefa destinatária pelo PC da tarefa emissora. Se ao longo da trajetória de uma mensagem um dos PC intermediários conhece uma localização mais atual da tarefa destinatária, então a localização e o seu grau de atualidade presentes na mensagem são corrigidos, a mensagem é redirecionada e é indicado na mensagem que o PC da tarefa emissora não conhece a localização mais recente da tarefa destinatária.

Quando uma mensagem é recebida pelo PC do computador onde a tarefa destinatária se encontra ela é colocada na fila de mensagens da tarefa (veja a frente). Se há a indicação de que o PC da tarefa emissora não conhece a localização atual da tarefa destinatária então, é enviada uma mensagem LOCAL direcionada a tarefa emissora.

```
Envia uma mensagem de atualização da localização para
os computadores definidos como a primeira aproximação
da sua localização.
```

Figura 1: Primeira Execução de uma Tarefa

O PC participa do processo de roteamento de uma mensagem quando ocorre um dos seguintes eventos:

- Primeira execução de uma tarefa.
- Recepção de uma mensagem de atualização da localização de uma tarefa.

```
Se a tarefa destinatária estiver presente
    Coloca a mensagem na fila da tarefa destinatária.
    Fim.
Envia a mensagem em direção ao computador onde
a tarefa destinatária deve se localizar.
```

Figura 2: Pedido de Transmissão de Mensagens

- Pedido de transmissão de mensagens por parte de uma tarefa em seu computador.
- Recepção de uma mensagem entre tarefas proveniente de outro PC

As figuras 1, 2, 3 e 4 explicitam as ações realizadas em cada caso.

```
Atualiza a localização da(s) tarefa(s) especificadas.
    Se há mensagens acumuladas endereçadas à(s) tarefa(s)
    cuja localização foi atualizada.
        Envia as mensagens em direção ao computador onde
        a(s) tarefa(s) deve(m) estar.
Se a mensagem for endereçada a outro computador.
    Envia a mensagem em direção ao outro computador.
```

Figura 3: Recepção de Mensagens de Localização

Cada PC gerencia as seguintes informações:

- **Tabela das tarefas**: indica a localização das tarefas. Para cada tarefa há duas informações: endereço mais recente e atualidade deste endereço.
- **Tarefas presentes**: lista das tarefas em um computador. Para cada tarefa presente estão associadas as seguintes informações: *fila de mensagens não entregues* e as *listas de correspondentes*.

A ordem parcial das mensagens trocadas entre as tarefas é garantida por duas estruturas de dados por tarefa, *as listas de correspondentes* e a *fila das mensagens não entregues*. Há uma **lista de correspondentes** com as tarefas que receberam mensagens desta tarefa e outra com as tarefas que enviaram mensagens a esta tarefa. Cada elemento destas listas contém a identificação de uma tarefa que já se correspondeu com esta e o número de mensagens já transmitidas ou recebidas. A **fila de mensagens** contém as mensagens que ainda não foram entregues às tarefas destinatárias, seja porque as tarefas ainda não as solicitaram, ou porque mensagens anteriores ainda não chegaram.

Todas as mensagens têm um número de ordem, que é incluído pelos PC, a partir da lista de receptores da tarefa emissora. As mensagens são ordenadas por este número e permanecem na fila até que as mensagens anteriores sejam entregues. A lista de emissoras de uma tarefa têm o número de ordem da última mensagem recebida pela tarefa, proveniente de cada tarefa que já lhe enviou mensagens.

A migração de uma tarefa envolve a migração do seu estado de execução, incluindo o número de migrações que já sofreu, a sua fila de mensagens e as listas de correspondentes.

5.2 Verificação da Correção do Protocolo

O protocolo proposto é correto se não produz *"deadlocks"* e se as mensagens que são roteadas de acordo com o protocolo são em algum momento entregues as tarefas destinatárias.

```
Se a mensagem contém uma localização da tarefa emissora
ou destinatária mais atual que este processador de
comunicações.

    Atualiza a localização da(s) tarefa(s) no processador
    de comunicações.

        Se há mensagens acumuladas endereçadas à(s) tarefa(s)
        cuja localização foi atualizada.

            Envia as mensagens em direção ao computador onde
            a(s) tarefa(s) deve(m) estar.

Se a mensagem contém uma localização da tarefa destinatária
menos atual que este processador de comunicações.

    Atualiza a localização da tarefa na mensagem.

    Indica na mensagem que o processador de comunicações
    do computador de origem da mensagem não conhece
    a localização atual da tarefa destinatária.

Se a tarefa destinatária estiver presente

    Coloca a mensagem na fila da tarefa destinatária.

    Se a mensagem não estava destinada a este computador
    ou está indicado que o processador de comunicações
    do computador da tarefa emissora não conhece a
    localização atual da tarefa destinatária.

        Envia uma mensagem de atualização da
        localização da tarefa.

    Fim.

Envia a mensagem em direção ao computador onde
a tarefa destinatária deve se localizar.
```

Figura 4: Recepção de Mensagens entre Tarefas

A ausência de *"deadlocks"* decorre da inexistência de bloqueios no protocolo em si e da ausência de *"deadlocks"* no método de controle de fluxo utilizado.

A prova de que as mensagens roteadas segundo o protocolo proposto alcançam as tarefas destinatárias se baseia nas seguintes premissas definidas no próprio protocolo:

1. Para cada tarefa há um conjunto de computadores que conhecem ou conhecerão uma localização desta tarefa. Todos os computadores reconhecem um dos membros deste conjunto como a localização da tarefa com o grau de atualidade igual a zero.
2. A criação de uma tarefa faz com que a localização e o seu grau de atualidade no computador da criação sejam alterados para respectivamente a identidade deste computador e um.
3. A criação de qualquer tarefa envolve o envio de uma mensagem com a sua localização de grau de atualidade um para todos os computadores reconhecidos como a localização de grau de atualidade zero da tarefa criada.
4. O PC de um computador armazena as mensagens recebidas, que forem dirigidas a tarefas cuja localização prevista na mensagem tenha um grau de atualidade igual a zero e que seja o próprio computador. Quando este PC passar a conhecer uma localização de uma destas tarefas com o grau de atualidade superior a zero as mensagens armazenadas endereçadas a esta tarefa continuaram o seu trajeto.

5. A migração de uma tarefa faz com que a localização e o seu grau de atualidade nos dois computadores envolvidos sejam atualizados respectivamente para o computador receptor e para um mais o número de migrações (presente no estado da tarefa).
6. As informações em um computador sobre a localização e o seu grau de atualidade de uma tarefa só são alterados pela recepção ou passagem de uma mensagem com uma localização da tarefa com um grau maior de atualidade, ou pela recepção da própria tarefa.
7. A localização de uma tarefa e o seu grau de atualidade presentes em uma mensagem sempre se originam de informações idênticas do computador de origem da mensagem ou de algum computador pelo qual a mensagem passou, que apresentava uma localização com um grau de atualidade maior.
8. As mensagens sempre são enviadas em direção a localização mais atual da sua tarefa destinatária.

As premissas 1, 2, 3 e 4 têm como conseqüência o seguinte corolário:

Qualquer mensagem enviada para um computador que é a localização com grau de atualidade zero da sua tarefa destinatária, será enviada para o computador onde esta tarefa destinatária foi criada.

Para mostrar a correção do protocolo proposto é necessário provar o seguinte teorema:

Mesmo que a informação em um computador sobre a localização de uma tarefa não esteja necessariamente atualizada, ela sempre indica uma localização correta da tarefa no passado ou no presente.

A premissa 1 garante a correção da primeira informação quanto a localização de tarefas em todos os computadores. Resta verificar se estas informações podem se tornar incorretas após alguma atualização.

Segundo a premissa 6 as informações em um computador sobre a localização de uma tarefa são atualizadas pela recepção de uma mensagem com uma informação mais atual ou pela participação em uma migração. Elas estarão erradas se em algum destes casos o computador receber uma informação errada quanto a localização de uma tarefa.

De acordo com a premissa 5 a participação de um computador na migração de uma tarefa garante a este computador uma informação correta sobre a localização da tarefa. Assim esta descartada a possibilidade de um computador receber uma informação errada, quanto a localização de uma tarefa, através da participação em uma migração.

De acordo com a premissa 7, as informações sobre a localização de uma tarefa em uma mensagem se originam de um dos computadores, portanto as mensagens não introduzem informações erradas no sistema e se todas as informações anteriores estavam corretas, as mensagens não as tornarão erradas. Assim esta descartada a possibilidade de um computador receber uma informação errada, quanto a localização de uma tarefa, através de uma mensagem.

A partir do corolário, deste teorema e das premissas definidas no protocolo proposto pode-se provar a correção do protocolo proposto.

O corolário e a premissa 5 garantem que a trajetória mais longa percorrida por uma mensagem será constituída pelas seguintes trajetórias:

1. do computador de origem da mensagem até um computador que conhece a localização da tarefa com grau zero,
2. deste computador até o computador de criação da tarefa destinatária,
3. a trajetória percorrida pela tarefa em suas migrações.

Como de acordo com o teorema todas as informações quanto a localização e o seu grau de atualidade de uma tarefa estão sempre corretas e como pelas premissas 7 e 8 a continuação da trajetória de uma mensagem é sempre em direção ao computador conhecido pelos computadores por onde a mensagem passou como a localização mais atual da tarefa naquele momento, então qualquer outra trajetória percorrida pela mensagem será mais curta.

No pior caso então, a trajetória percorrida por uma mensagem é maior que a trajetória da sua tarefa destinatária por um número constante de computadores intermediários (trajetória 1 e 2 acima). Supondo-se que uma mensagem trafega com uma velocidade maior do que a velocidade de migração de uma tarefa, então todas as mensagens alcancaçam suas tarefas destinatárias.

Esta suposição é razoável pois normalmente as mensagens entre tarefas são muito menores que o somatório de mensagens para realizar uma migração. Qualquer que seja o protocolo para a comunicação entre tarefas migrantes, se as migrações tiverem uma maior velocidade que as transferências de mensagens, então com o passar do tempo, as mensagens poderão estar cada vez mais distantes das suas tarefas destinatárias. Assim, sem a suposição acima não se pode garantir que as mensagens alcancaçaram as suas tarefas destinatárias qualquer que seja o protocolo.

5.3 Qualidade do Protocolo Proposto

A verificação da qualidade do protocolo proposto pode ser realizada verificando-se se o protocolo proposto garante as propriedades definidas na seção 2.

A propriedade 1, relativa à invisibilidade do processo de migração para as tarefas, é garantida pelo uso de um identificador distinto para cada tarefa, que independe da localização da tarefa, e cujo reconhecimento em todo o sistema é realizado totalmente pelos PC.

As propriedades 2, relativa à integridade das mensagens, e 3, relativa a unicidade das mensagens, são garantidas pela concepção do protocolo. As mensagens não são nem particionadas, nem duplicadas.

A propriedade 4, relativa ao ordenamento parcial das mensagens, é garantida pela gerência da fila de mensagens recebidas e das listas de tarefas correspondentes de cada tarefa, que é realizada pelo PC do computador onde a tarefa se encontra.

A propriedade 5, relativa à eficácia, é decorrência da correção do protocolo, que foi provada na subseção 5.2.

6 Estágio Atual e Trabalhos Futuros

O sistema de comunicações apresentado foi utilizado com sucesso em um simulador hibrido, cujo objetivo era a avaliação de algoritmos para escalonamento dinâmico distribuído em multicomputadores.

A implementação do sistema proposto por software neste simulador foi utilizada para a emulação da execução de centenas de aplicações paralelas distribuídas. Cada uma destas aplicações apresentava centenas de tarefas comunicantes, que podiam migrar de acordo com as decisões do algoritmo de escalonamento global em ação.

Como trabalho futuro de pesquisa pode-se realizar uma avaliação do desempenho do sistema proposto e uma comparação com outras abordagens para o mesmo problema.

Referências

[1] Azevedo G. P. de; **"Escalonamento Dinâmico Distribuído em Multicomputadores de Alto Desempenho"**; *Dissertação de tese de Mestrado, Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ, maio de 1992.*

[2] Dally W.; **Network and processor architecture for message driven computers**; *VLSI and Parallel Computation, capítulo 3, 1989.*

[3] Douglis, F. e J. K. Ousterhout; **Process Migration in the Sprite Operating System**; *Seventh International Conference on Distributed Computing, pp. 18-25, Sept. 1987.*

[4] Fowler, R. J.; **Decentralized Object Finding Using Forwarding Addresses**; *PhD thesis, University of Washington, Seatle, Washington, December 1985.*

[5] Fowler, R. J.; **The Complexity of Using Forwarding Addresses for Decentralized Object Finding**; *Proceedings of the Fifth ACM Symposium on the Principles of Distributed Computation, Calgary, Canada, August 1986.*

[6] Jul, E., H. Levy, N. Hutchinson e A. Black; **Fine-Grain Mobility in the Emerald System**; *ACM Transactions on Computer Systems, Vol. 6, No. 1, February 1988.*

[7] Kermani P. e Kleinrock L.; **Virtual cut-through: a nnew computer communication switchig technnique**; *North-Holland, Computer Networks, Vol 3, No 4, pp. 267-286, September 1979.*

[8] Linder D. H e Harden J C.; **An adaptative and fault tolerant wormhole routing strategy for k-ary n-cubes**; *IEEE Transactions on Computers, Vol 40, No 1, pp. 2-12, January 1991.*

[9] Lu C., Chen A. e Liu J.; **Protocols for reliable process migration**; *In Proceedings of INFOCOM'87, San Francisco, California, March 1987.*

[10] Macharia G. M.; **CLD: A novel approach to dynamic load balancing**; *Microprocessing and Microprogramming 28, pp. 43-48, 1989.*

[11] Ni L. M., Xu C.-W. e Gendreau T. B.; **A distributed drafting algorithm for load balancing**; *IEEE Transactions on Software Engineering SE-11(10), pp. 1153-1161, October 1985.*

[12] Ousterhout, J. K. et al.; **The Sprite Network Operating System**; *Computer, February 1988.*

[13] Popek, G. J. e B. J. Walker; **The LOCUS Distributed System Architecture**; *Computer Systems Serires, The MIT Press, 1985.*

[14] Powell, M. L. e B. P. Miller; **Process Migration in DEMOS/MP**; *Proceedings of the Ninth Symposium on Operating Systems Principles, October 1983.*

[15] Ravi T. M. e Jefferson D.; **A basic protocol for routing messages to migrating processes**; *In proceedings of the International Conference on Parallel Processeing, Vol. II, Software, August 1988.*

[16] Reed D. A. e Fujimoto R. M.; **Multicomputer networks: message based parallel processing**; *MIT Press series in scientific computation, 1987.*

[17] Seitz C. et al; **Wormhole chip project report**; *Inverno 1985.*

[18] Theimer, M.; **Preemptable remote execution facilities for loosely-coupled distributed systems**; *Stanford University Technical Report STAN-CS-86-1128, June 1986.*