O Impacto do Protocolo TCP/IP na Computação Paralela Distribuída no Ambiente Windows95®

Paulo Sérgio Lopes de Souza - e-mail: pssouza@icmsc.sc.usp.br *
Marcos José Santana - e-mail: mjs@icmsc.sc.usp.br **
Regina Helena Carlucci Santana - e-mail: rcs@icmsc.sc.usp.br **
Luciano José Senger - e-mail: ljsenger@icmsc.sc.usp.br **
Robson Picinato - e-mail: picinato@icmsc.sc.usp.br **

* Universidade Estadual de Ponta Grossa (PR) - UEPG
Praça Santos Andrade, s/n - Caixa Postal: 992/993
84010-250 - Ponta Grossa - PR

** Instituto de Ciências Matemáticas de São Carlos - ICMSC/USP
Av. Dr. Carlos Botelho - Caixa Postal: 668
13560-970 - São Carlos - SP

Abstract

This paper describes the differences in terms of performance observed in both Windows95 and LINUX Operating Systems TCP/IP protocols implementations. The aim is to demonstrate the impact caused by these protocols on PVM-W95 (Parallel Virtual Machine for Windows95) and PVM (Parallel Virtual Machine) message passing environments. Some features of these environments are described to make clear how deeply the performance is affected by the communication protocol.

Resumo

Este artigo descreve a diferença de desempenho encontrada nas implementações do protocolo TCP/IP dos Sistemas Operacionais Windows95 e LINUX, com o objetivo de demonstrar o impacto gerado por esses protocolos nos ambientes de passagem de mensagens PVM-W95 (*Parallel Virtual Machine for Windows95*) e PVM (*Parallel Virtual Machine*). São descritas algumas características dos ambientes citados, com a intenção de demonstrar a importância da comunicação no desempenho de uma ferramenta de software, que tem por objetivo justamente melhorar o desempenho de aplicações seqüenciais.

1. Introdução

Uma das grandes vantagens da Computação Paralela Distribuída é permitir uma melhor relação custo/benefício para a computação paralela [ZAL91] [ALM94], oferecendo a potência computacional ideal às aplicações que não necessitam de uma máquina maciçamente paralela, porém necessitam de mais potência computacional do que uma máquina seqüencial pode oferecer [GEI94a] [GEI94b]. PVM (*Parallel Virtual Machine*) e MPI (*Message Passing Interface*) são dois exemplos de bibliotecas de passagem de mensagens, amplamente discutidos na literatura, que permitem a criação de máquinas paralelas virtuais em estações de trabalho, normalmente máquinas RISC com sistema operacional UNIX® [GEI94b] [MCB94].

Embora os computadores pessoais (PCs) e o Windows® sejam utilizados por um grande número de empresas, instituições de ensino, pesquisa e órgãos governamentais, apenas recentemente foram desenvolvidos trabalhos de pesquisa com o intuito de explorar a computação paralela distribuída em tal plataforma [SOU96] [SAN96] [FIS96] [ALV95].

O PVM-W95 (Parallel Virtual Machine for Windows95®) é um exemplo de ambiente de passagem de mensagens desenvolvido no Instituto de Ciências Matemáticas de São Carlos - ICMSC/USP, o qual atende às especificações do modelo PVM original [GEI94a] [GEI94b] [SUN94]. Este trabalho discute os aspectos de desempenho mais importantes do PVM-W95, observados no seu desenvolvimento e em estudos posteriores. Também são apresentadas algumas das principais características do PVM-W95, ressaltando a importância dos protocolos de comunicação em ambientes de passagem de mensagens.

Este artigo está organizado em seis seções. Na segunda seção são discutidas as principais características do PVM-W95, destacando seus objetivos, organização e protocolos de comunicação. A terceira seção aborda a avaliação de desempenho do PVM-W95 feita por Souza [SOU96], analisando-se os valores que indicam a presença de problemas com o desempenho do ambiente desenvolvido, quando comparado ao PVM original. A quarta seção expõe a avaliação feita no protocolo de

comunicação TCP/IP e os resultados obtidos. Finalizando o artigo, a quinta e a sexta seções apresentam, respectivamente, as conclusões e os agradecimentos.

2. PVM-W95 - Principais Características

O PVM-W95 fornece os recursos necessários para desenvolver programas concorrentes na linguagem C (ou C++), executados em Computadores Pessoais (PCs), com o Sistema Operacional Windows95® e conectados por uma rede de comunicação, de modo análogo ao PVM [SOU96] [SAN96].

A escolha do PVM deve-se a sua grande utilização, sendo considerado como um **padrão de fato** na computação paralela distribuída. Robustez, simplicidade e eficiência são outros fatores que também contribuíram para a escolha do PVM.

O PVM-W95 foi desenvolvido com base no código fonte do PVM. Todas as rotinas pertencentes ao PVM foram estudadas e, posteriormente, a grande maioria dessas foi adaptada para a nova plataforma. No total foram analisadas 33.258 linhas dispostas em 58 arquivos *.c* e *.h* . Essa estratégia possibilitou reduzir o tempo de desenvolvimento, permitiu portabilidade e heterogeneidade [SOU96] [SAN96].

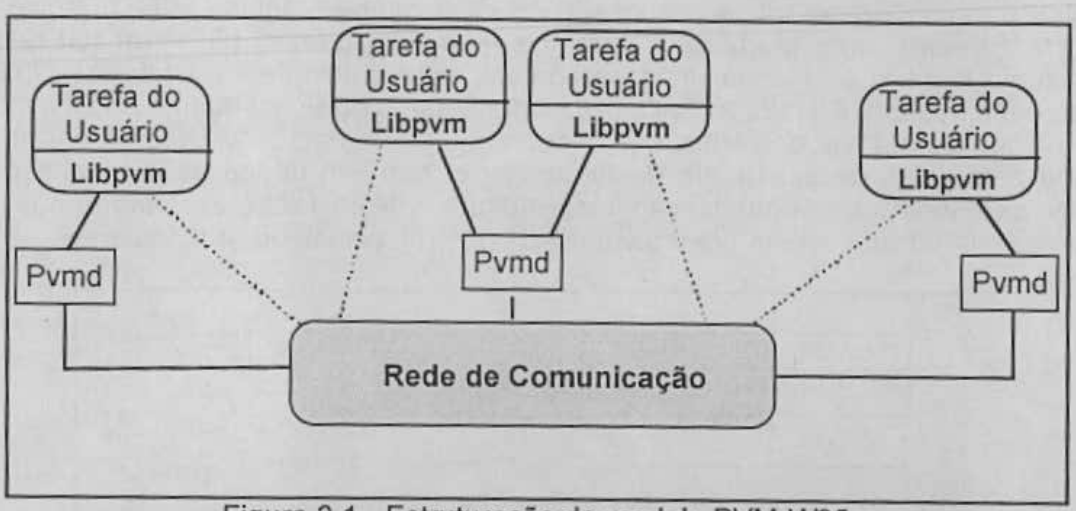


Figura 2.1 - Estruturação do modelo PVM-W95

O código do PVM-W95 foi estruturado utilizando-se os conceitos de orientação a objetos: abstração, encapsulamento, polimorfismo e herança [PIN91] [TAK90] [WIE91]. A inclusão da orientação a objetos forneceu ao código do PVM-W95 a organização necessária para que futuras manutenções sejam feitas com o menor custo possível.

O PVM-W95, assim como o PVM, é composto por duas partes principais: um *daemon*, também conhecido como Pvmd, e uma biblioteca de rotinas com a interface PVM, também chamada de Libpvm (Figura 2.1) [SOU96] [SAN96] [STE90] [SUN94]. O **Pvmd** é executado em cada *host* que compõe a máquina virtual atuando como "gerenciador" da máquina e roteador de mensagens. A **Libpvm** contém um conjunto de primitivas que atuam como elo de ligação entre uma tarefa e a máquina virtual (o Pvmd e as outras tarefas) [GEI94a] [GEI94b].

2.1. PVM *Daemon* (Pvmd)

O Pvmd é um processo servidor que atende às requisições feitas, ou pelas tarefas pertencentes àquele *host*, ou por outros Pvmds (no caso, atuando como processos clientes). O Pvmd também possui todos os métodos e estruturas de dados necessários à manipulação dos pacotes que compõem as mensagens. Sucintamente, o Pvmd é responsável por:

- configurar todo Pvmd que começa a ser executado, como por exemplo, fornecer o endereço IP, criar os mecanismos de comunicação entre os processos necessários e outras configurações necessárias;
- controlar todo o roteamento de mensagens feito pelo Pvmd com suas tarefas e com outros Pvmds;
- manter todas as filas de mensagens e pacotes recebidos e/ou enviados pelo Pvmd;
- criar e manter atualizadas a tabela de *hosts* e a tabela de tarefas (suas estruturas de dados mais importantes), permitindo assim que a configuração da máquina paralela virtual esteja sempre correta;
- estabelecer e executar o protocolo de comunicação Pvmd⇔Pvmd e Pvmd⇔Tarefa, para que as requisições e respostas sejam transmitidas e/ou recebidas corretamente;
- criar e manter os salvadores de contexto para permitir a execução multitarefa do Pvmd;

- criar novos Pvmds, permitindo assim modificar dinamicamente a máquina paralela virtual.

2.2. Biblioteca de Comunicação (Libpvm)

A biblioteca Libpvm é uma biblioteca de ligação dinâmica (*.dll*), responsável pela interface com a aplicação paralela. A Libpvm também possui todos os métodos e estruturas de dados necessários à comunicação e à manipulação dos pacotes que compõem as mensagens. Resumidamente, a Libpvm é responsável por:

- implementar os codificadores e decodificadores utilizados para plataformas heterogêneas;
- conectar e desconectar a tarefa no PVM-W95;
- criar os mecanismos de IPC (*Interprocess Communication*) necessários para comunicação com o Pvmd e/ou para comunicação diretamente entre as tarefas (esta indicada pela linha tracejada na Figura 2.1);
- manter os protocolos de comunicação Pvmd⇔Tarefa e Tarefa⇔Tarefa;
- implementar as rotinas para o gerenciamento de grupos de tarefas;
- restringir o acesso da aplicação paralela aos métodos e/ou estruturas de dados não permitidos.

2.3. Comunicação entre Processos no PVM-W95

A máquina paralela virtual emulada com o PVM-W95 é uma máquina MIMD (*Multiple Instruction Multiple Data Stream*), com memória distribuída [FLY72] [ALM94] [GEI94a] [GEI94b] [SOU96], tornando o meio físico e os mecanismos de comunicação entre processos fatores importantes na obtenção de desempenho. De fato, a principal finalidade, tanto do Pvmd quanto da Libpvm, é gerenciar e garantir a troca de mensagens entre os vários componentes do PVM-W95. A portabilidade, fator inerente aos ambientes de passagem de mensagens, também impõe sérias limitações quanto à utilização de protocolos de comunicação mais eficazes. Nem todas as plataformas (hardware + sistema operacional) dispõem uma grande variedade de protocolos aos seus usuários.

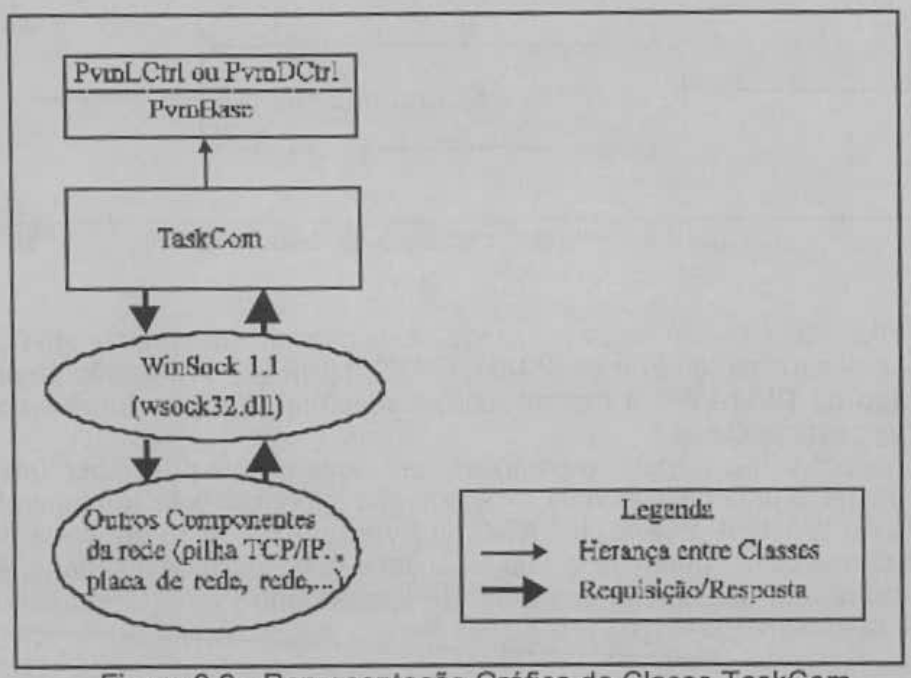


Figura 2.2 - Representação Gráfica da Classe TaskCom

Devido à importância da comunicação no PVM-W95 e para facilitar futuras atualizações, os mecanismos de comunicação entre processos (IPC) foram implementados separadamente do restante código fonte, através da orientação a objetos. Todos os métodos e estruturas de dados foram encapsulados em uma classe chamada *TaskCom* (Figura 2.2) [SOU96] [SAN96]. O encapsulamento torna o PVM-W95 independente do protocolo de comunicação utilizado (atualmente o TCP/IP). Visando facilitar manutenções posteriores foram criados 28 métodos, herdados por outras classes, os quais são responsáveis pela interface entre PVM-W95 e o protocolo de comunicação.

O PVM-W95 utiliza a versão 32 bits do *Windows Sockets*® (Winsock 1.1) [BON96] [DUM95] [HAL93], para permitir o acesso à pilha do protocolo TCP/IP. A escolha pelo WinSock 1.1 e pelo TCP/IP está relacionada à implementação do PVM para UNIX.

Como a comunicação no PVM-W95 é feita entre os Pvmds e entre as tarefas, há três conexões a considerar (assim como no PVM): entre Pvmds, entre um Pvmd e suas tarefas e entre tarefas.

A conexão entre Pvmds é feita através de UDP. Como o UDP não é um protocolo confiável, o PVM-W95 implementa serviços de confirmação e retransmissão das mensagens. O protocolo TCP não foi

utilizado por três motivos principais. O primeiro é a "escalabilidade". Em uma máquina com N *hosts*, cada Pvmd deve ter (N - 1) conexões, onde cada transmissão aberta com TCP consome um descritor de arquivos no Pvmd e alguns sistemas operacionais limitam o número de arquivos abertos. Um *socket* UDP aberto, entretanto, pode comunicar-se com qualquer outro *socket* UDP remoto. O segundo é o tempo extra (*overhead*) gasto para fazer as conexões necessárias com TCP. O terceiro é a tolerância a falhas. O sistema detecta que um Pvmd remoto está com problemas ou que a rede está paralisada através da utilização de *timeouts*. O TCP mantém opções que podem fazer esse trabalho, mas elas nem sempre possibilitam um controle adequado sobre os parâmetros [GEI94a] [GEI94b] [SOU96] [SUN94].

A comunicação entre um Pvmd e suas tarefas e entre tarefas é feita com TCP, devido principalmente ao fator confiabilidade. UDP necessita de confirmações extras e retransmissões, o que gera maiores *overheads*. Como as comunicações não têm o estilo das feitas entre Pvmds, uma vez que o tamanho das mensagens e a sua ocorrência dependem da aplicação desenvolvida, a melhor solução é a adoção do protocolo TCP.

3. Avaliação de Desempenho do PVM-W95

Para a avaliação de desempenho foi desenvolvida uma versão paralela do método de ordenação *quicksort*, a qual foi executada no PVM-W95 e no PVM original [SOU96]. Para coletar os tempos de execução no PVM-W95 foram utilizados 4 *hosts*, todos PCs conectados através de uma rede padrão *Ethernet*, utilizando o Sistema Operacional Windows95 e o compilador Borland C++ 4.5. Os equipamentos utilizados foram:

1 PC 486DX4-100, 12 Mbytes RAM;
1 PC 486DX2-66, 24 Mbytes RAM;
1 PC 486DX2-66, 20 Mbytes RAM;
1 PC 486DX2-66, 12 Mbytes RAM;

Para coletar os tempos de execução do PVM também foram utilizados 4 *hosts*, todos máquinas "SUN SPARCstations", conectadas por uma rede padrão *Ethernet*, utilizando o Sistema Operacional SunOS 4.1.3_U1 e o compilador C da SUN Microsystems, Inc. (cc - incluso na versão 4.1.3_U1). As estações utilizadas foram:

1 SPARCstation5, 70 MHz, 32Mbytes RAM;
1 SPARCstation2, 40 MHz, 16Mbytes RAM;
1 SPARCstation1, 20 MHz, 12Mbytes RAM;
1 SPARCstationSLC, 20 MHz, 8Mbytes RAM;

Analisando-se os *hosts* utilizados nas duas plataformas, observa-se que, além de possuírem arquiteturas completamente diferentes (RISC x CISC), os PCs apresentam uma maior potência computacional que as SUN SPARCstations.

Devido à característica multiusuário do UNIX, a qual afeta diretamente o desempenho do sistema, os tempos foram coletados em horários de pouca utilização dos laboratórios (finais de semana, à noite e feriados), minimizando, portanto, diferenças no tempo de execução com a existência de outros processos concorrentes nos equipamentos.

O *quicksort* foi executado variando-se duas características importantes: o número de elementos a ordenar e a granularidade.

Para aumentar o número de elementos, foram gerados vetores com 1.010, 10.150, 101.600 e 305.900 elementos inteiros. Considerando-se que cada fragmento de mensagem no PVM envia até 4064 bytes (valor default) e o tipo de dado inteiro ocupa 4 bytes, os tamanhos representam mensagens com 1 fragmento, 10 fragmentos, 100 fragmentos e 300 fragmentos respectivamente.

Para aumentar a granularidade de cada processo, sem aumentar, entretanto, o tamanho de cada mensagem, foi inserida uma estrutura de repetição. Esta estrutura força a ordenação dos elementos do vetor várias vezes, sem que sejam feitas novas trocas de mensagens. Os elementos foram ordenados 1, 50 e 100 vezes.

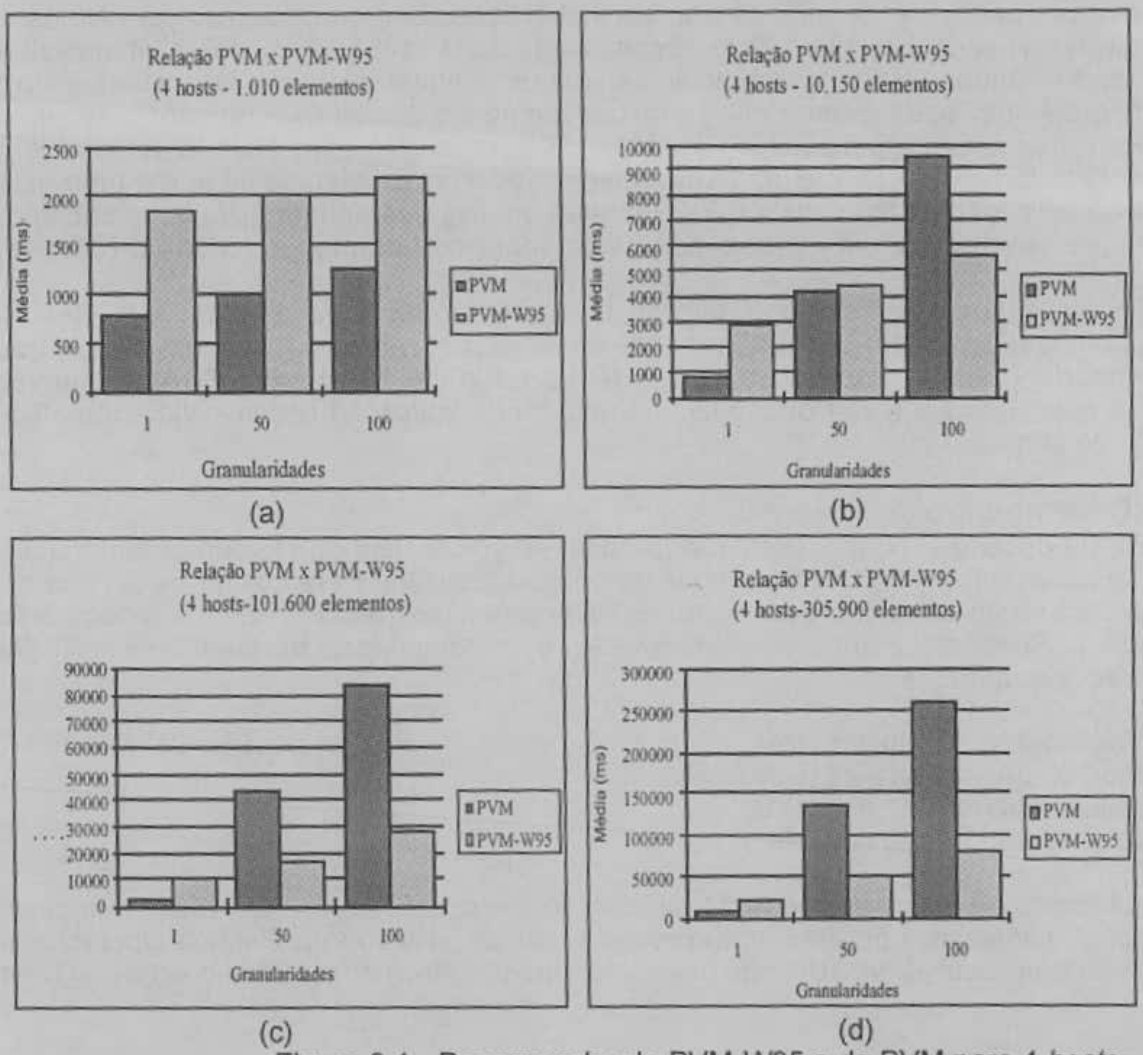


Figura 3.1 - Desempenho do PVM-W95 e do PVM para 4 *hosts*.

Os tempos obtidos são **tempos de usuário**, que consistem nos tempos gastos em ciclos de CPU e enquanto o processo está esperando por determinados eventos do sistema operacional (por exemplo operações de I/O e troca de contexto em sistemas de tempo compartilhado) [DIL95] [TAN96]. Os resultados finais representam a média obtida com 100 execuções.

A Figura 3.1 demonstra os resultados obtidos com a avaliação de desempenho feita por Souza [SOU96]. Analisando-se os valores obtidos com a **granularidade** 1 para qualquer número de elementos utilizados (Figura 3.1 a,b,c,d) verifica-se que o PVM obteve um desempenho melhor que o PVM-W95, o mesmo ocorrendo para ordenar **1.010 elementos com as 3 granularidades** (Figura 3.1 a). A característica em comum nesses dois casos é que a relação da granularidade das tarefas x comunicação é ruim, ou seja, o tempo utilizado é determinado mais pela comunicação do que pelo processamento em si.

Nos casos que possuem um maior número de elementos (**10.150, 101.600 e 305.900**), com maior **granularidade (50 e 100)** (Figura 3.1 b, c, d), onde os tempos de processamento de cada tarefa são maiores, o PVM-W95 obteve melhores resultados. Considerando que os equipamentos utilizados para executar o PVM-W95 possuem uma potência computacional maior que os utilizados pelo PVM, esse desempenho superior pode estar afetado por um processamento mais rápido durante a ordenação de cada subvetor.

Comparando o PVM-W95 com o PVM, observam-se duas situações distintas:

- o desempenho do PVM-W95 é superior ao PVM nos casos onde a granularidade é mais alta, ou seja, quando o tempo total de execução é influenciado mais pelo tempo de processamento do que pelo tempo de comunicação;
- o desempenho do PVM-W95 é inferior ao PVM nos casos onde a granularidade é mais baixa, ou seja, quando o tempo total de execução é influenciado mais pelo tempo de comunicação do que pelo tempo de processamento.

Considerando a maior potência computacional dos equipamentos utilizados pelo PVM-W95, há indícios de que a comunicação empregada no PVM-W95 seja menos eficiente que a comunicação empregada no PVM.

Devido às diferenças nas arquiteturas utilizadas (RISC x CISC), acredita-se que os dados obtidos não podem ser considerados conclusivos, necessitando, portanto, de novas avaliações [SOU96]. A seguir são apresentados os estudos feitos a partir da avaliação descrita nesta seção.

4. Avaliação de Desempenho do TCP/IP

Com base nos trabalhos descritos na seção anterior, o protocolo de comunicação TCP/IP foi analisado com os seguintes critérios:

- minimizar o número de características diferentes (heterogeneidade) entre as duas plataformas computacionais. Para tanto foram utilizados, nas duas plataformas, os mesmos hardware e algoritmo para *benchmark*. A heterogeneidade é encontrada nas implementações do protocolo TCP/IP (alvo desta avaliação), nos Sistemas Operacionais e nos compiladores, sendo que estes últimos também podem afetar os tempos obtidos;
- avaliar apenas o protocolo TCP/IP, sem utilizar nenhum recurso do PVM-W95 ou do PVM. A intenção é verificar qual é o impacto gerado com a utilização de duas implementações diferentes do mesmo protocolo na computação paralela distribuída em ambientes distintos (UNIX x Windows95);
- não escolher um "aplicativo" para a avaliação, como adotado por Souza [SOU96]. Para os testes foram escolhidos os algoritmos descritos por Dillon e Nevin [DIL95] [NEV96]. Os algoritmos utilizados como *benchmark* caracterizam-se por sua objetividade e simplicidade, fatores que contribuem para uma melhor avaliação "apenas" da transmissão de mensagens entre processos.

Seguindo estes critérios, a avaliação do TCP/IP foi realizada utilizando-se como hardware base dois PCs, conectados através de uma rede padrão *Ethernet* com os Sistemas Operacionais Windows95 e LINUX - versão 1.3.20 e com os compiladores Borland C++ 4.5 (no Windows95) e o GNU Compiler 2.6 (no LINUX). Os dois equipamentos utilizados foram:

1 PC 486DX2-66, 24 Mbytes RAM;
1 PC 486DX2-66, 20 Mbytes RAM;

A versão do TCP/IP no Windows95 foi a mesma utilizada pelo PVM-W95, ou seja, a versão 32 bits do *Windows Sockets*® (Winsock 1.1). No LINUX foi utilizado o TCP/IP que acompanha a versão 1.3.20 do Sistema Operacional.

Todos os tempos foram coletados minimizando-se o número de processos em execução nos equipamentos e em horários de pouca utilização da rede de comunicação, e os resultados finais representam a média de 100 execuções.

4.1. Medidas Utilizadas para a Avaliação

O tempo de envio de uma mensagem com n bytes em uma comunicação ponto-a-ponto ($t_{a\rightarrow b}(n)$) pode ser calculado pelo tempo utilizado para o empacotamento da mensagem ($t_{delay\_sendA}(n)$), pelo envio da mensagem de uma interface até a outra interface ($t_{net}(n)$) e pela liberação da mensagem para o processo destino após o recebimento na outra máquina ($t_{delay\_recvB}(n)$) [NEV96] [DIL95]. O tempo descrito pode ser representado como:

$$t_{a\rightarrow b}(n) = t_{delay\_sendA}(n) + t_{net}(n) + t_{delay\_recvB}(n) \qquad (1)$$

No entanto, um dos grandes problemas da avaliação de desempenho na comunicação ponto-a-ponto é a falta de sincronismo entre os *clocks* das máquinas envolvidas na transmissão das mensagens [DIL95] [NEV96]. Devido a esse problema, foi utilizado o algoritmo *ping-pong*, o qual permite que sejam coletados os tempos *round-trip* (*round-trip time*) [TAN96] [NEV96] [DIL95]. O *round-trip time* representa o tempo utilizado por uma mensagem para chegar no seu destino e retornar ao remetente. O tempo utilizado é medido somente em uma máquina, evitando assim os problemas de sincronismo. A equação abaixo fornece uma definição mais apropriada:

$$t_{a\leftrightarrow b}(n) = t_{delay\_sendA}(n) + t_{net}(n) + t_{delay\_recvB}(n) + t_{delay\_sendB}(n) + t_{net}(n) + t_{delay\_recvA}(n) \qquad (2)$$

Assumindo que o tempo de envio de uma mensagem de um processo A para um processo B seja idêntico ao tempo de envio de uma mensagem do processo B para o processo A, pode-se simplificar a equação acima para:

$$t_{a\leftrightarrow b}(n) = 2 . (t_{delay\_sendA}(n) + t_{net}(n) + t_{delay\_recvB}(n))$$

ou

$$t_{a\leftrightarrow b}(n) = 2 . t_{a\rightarrow b}(n) \quad (3)$$

Assim, pode-se assumir que o tempo utilizado para a troca de uma mensagem entre dois *hosts* idênticos é:

$$t_{a\rightarrow b}(n) = \frac{t_{a\leftrightarrow b}(n)}{2} \quad (4)$$

O *throughput - p(n)* [TAN96] [NEV96] [DIL95] é expresso em Mbits/s ou Kbits/s neste trabalho e é representado por:

$$p(n) = \frac{8 . n}{t_{a\rightarrow b}(n) . 10^6} \quad (5)$$

ou, utilizando-se o *round-trip time*:

$$p(n) = \frac{2 . 8 . n}{t_{a\leftrightarrow b}(n) . 10^6} \quad (6)$$

4.2. Resultados Obtidos

As Figuras 4.1 a 4.4 demonstram os resultados obtidos com a avaliação de desempenho realizada neste trabalho. A análise dos protocolos TCP e UDP foram realizadas sob dois aspectos: a transmissão local e a remota, ou seja, entre processos diferentes executados na mesma máquina e entre processos executados em máquinas distintas.

Analisando-se os valores obtidos, percebe-se que o TCP/IP no Windows95 apresentou um resultado pior, em comparação com os resultados do TCP/IP no LINUX.

O TCP no Windows95 apresentou uma perda média de desempenho da ordem (obteve-se a média da divisão de todos os *throughputs* do TCP/IP no LINUX pelos *throughputs* do TCP/IP no Windows95) de 2,32 para transmissões remotas (utilizadas no PVM-W95 e PVM para as transmissões diretas entre as tarefas dos usuários). Para transmissões locais com o TCP (utilizadas no PVM-W95 e PVM entre as tarefas dos usuários e o Pvmd), a perda foi ainda maior, da ordem de 3,73.

O UDP no Windows95 apresentou menos eficiência ainda, visto que a perda de desempenho é da ordem de 4,50 para transmissões remotas e de 14,08 para transmissões locais. A transmissão remota do UDP é muito utilizada, visto que é empregada para a comunicação entre os Pvmds e a transmissão local do UDP é utilizada para iniciar todos os novos Pvmds, através do *Pvmd Shadow* ou Pvmd'.

5. Conclusões

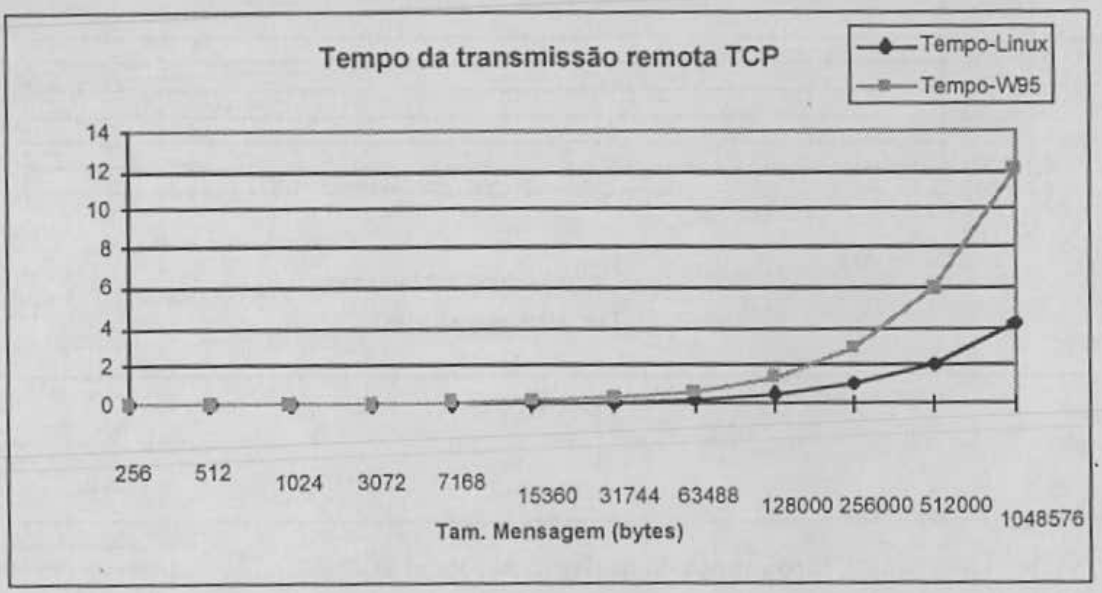
Este trabalho apresentou a relação existente entre o desempenho do TCP/IP no LINUX - versão 1.3.20 e no Windows95 (Windows Sockets 1.1 - v.32 bits). O objetivo é verificar qual o impacto gerado pelo protocolo de comunicação na computação paralela distribuída com o PVM-W95. Para isso, foram apresentadas algumas das características de implementação do PVM-W95, destacando-se principalmente os aspectos relacionados com a comunicação e seus protocolos.

Foram relacionados os valores obtidos com a avaliação de desempenho preliminar feita por Souza [SOU96]. Essa avaliação indica prováveis problemas quanto ao desempenho do ambiente de passagem de mensagens desenvolvido, os quais motivaram a realização deste trabalho.

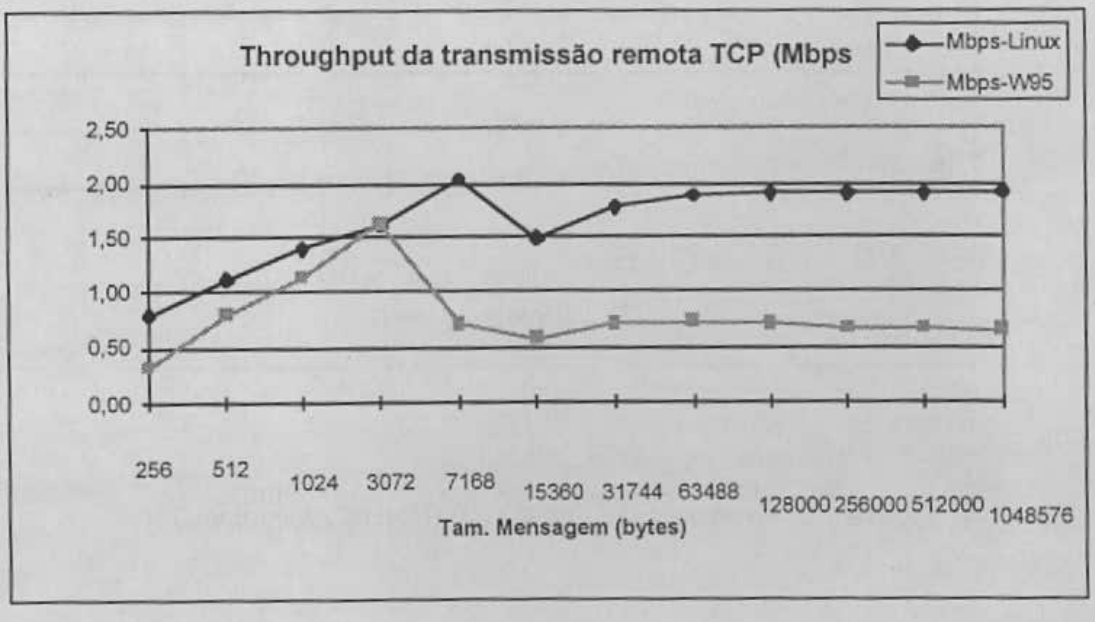
Os resultados obtidos demonstram claramente que o TCP/IP no Windows95 impõe sérias limitações ao desempenho do PVM-W95. Com isso, o PVM-W95 deve apresentar desempenhos inferiores, quando comparado ao PVM original na mesma plataforma de hardware.

Outro fator a ser considerado é que, embora os valores obtidos indiquem um desempenho inferior do PVM-W95 em relação ao PVM no LINUX devido ao protocolo, o PVM-W95 permite a perfeita construção da máquina paralela virtual no ambiente Windows®, com comportamento estável e desempenho satisfatório para várias aplicações paralelas.

Atualmente encontram-se em desenvolvimento pesquisas que procuram solucionar os problemas apontados por este trabalho. As pesquisas concentram-se na utilização de novos protocolos pelo PVM-W95, bem como na redução da troca de mensagens feita atualmente pelos métodos do PVM-W95.

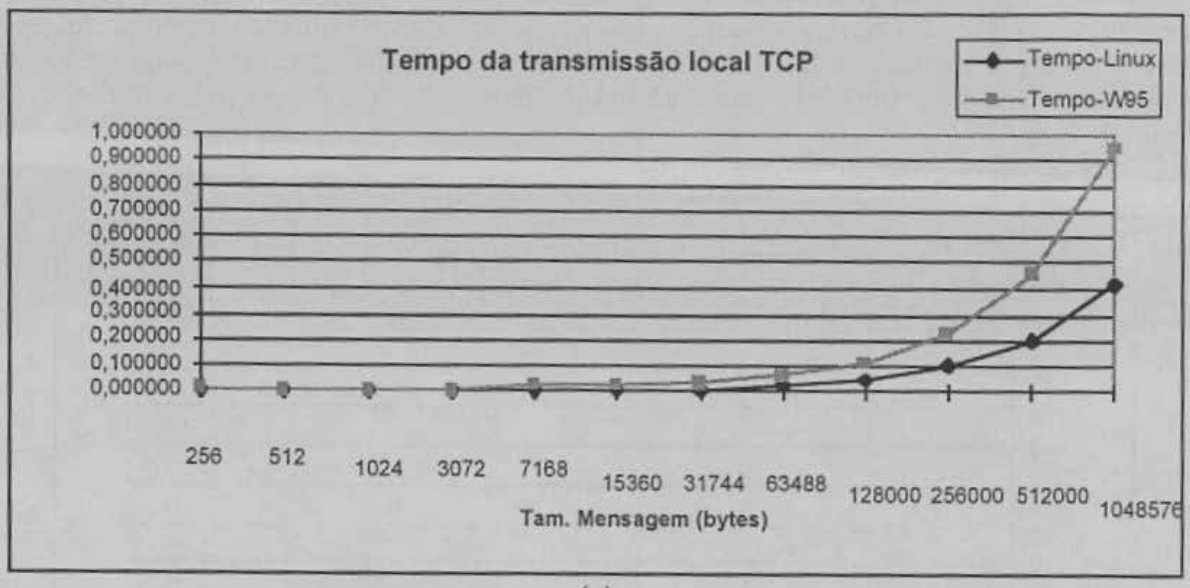


(a)

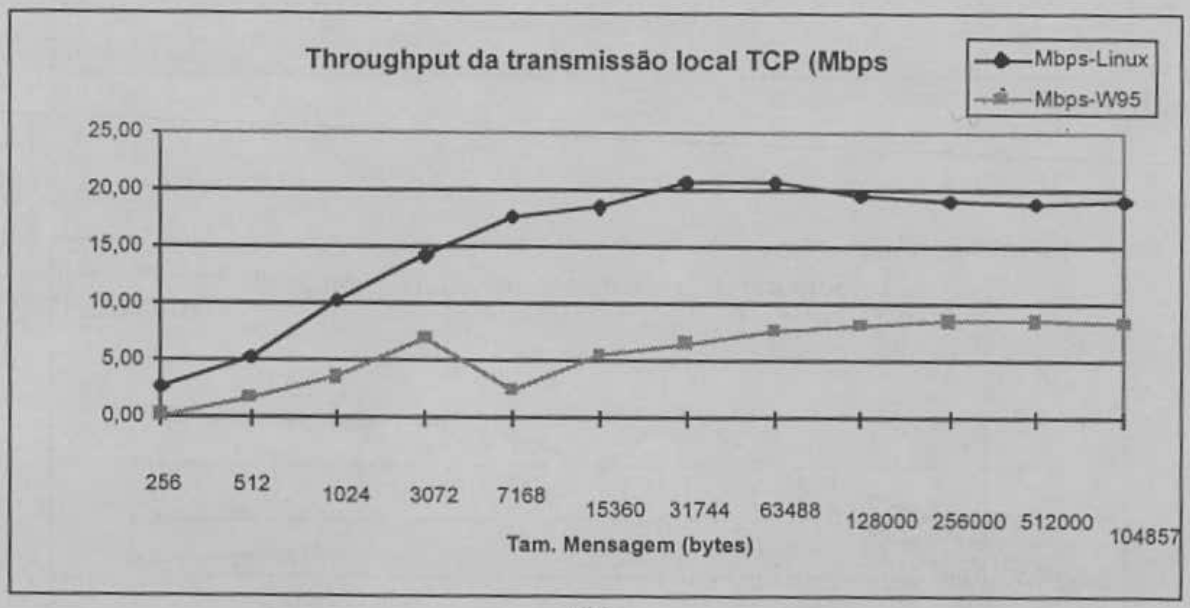


(b)

Figura 4.1 - Transmissão Remota TCP (LINUX x Windows95)

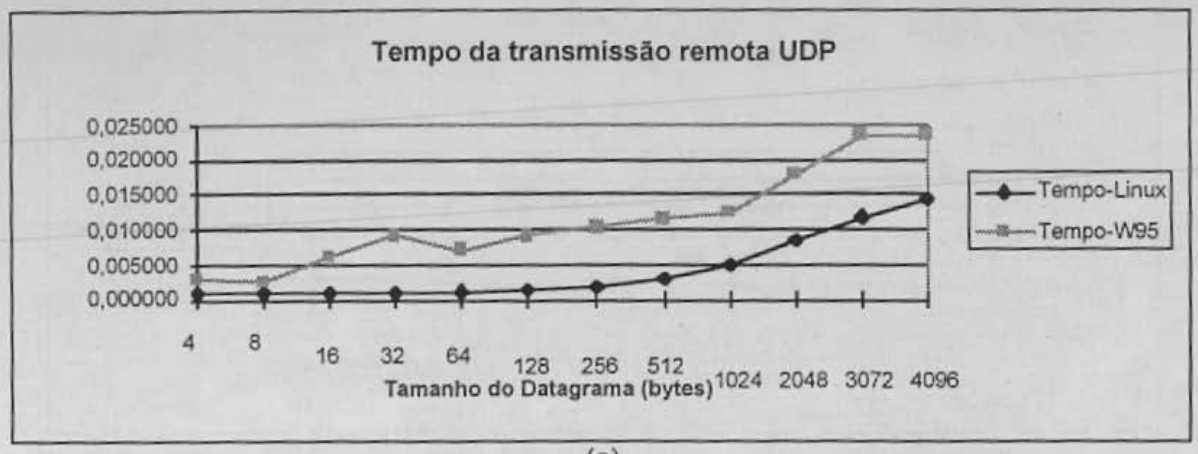


(a)

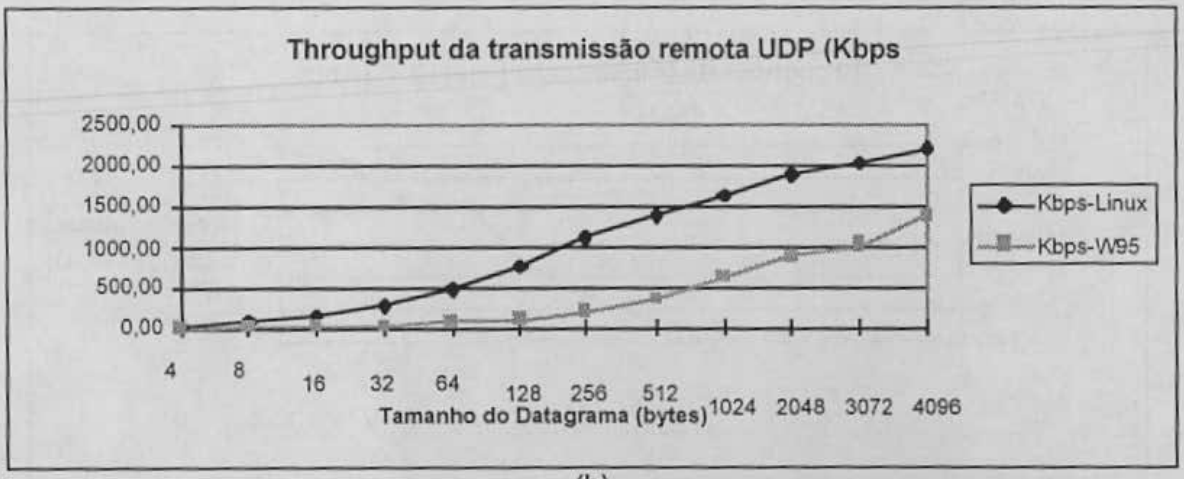


(b)

Figura 4.2 - Transmissão Local TCP (LINUX x Windows95)

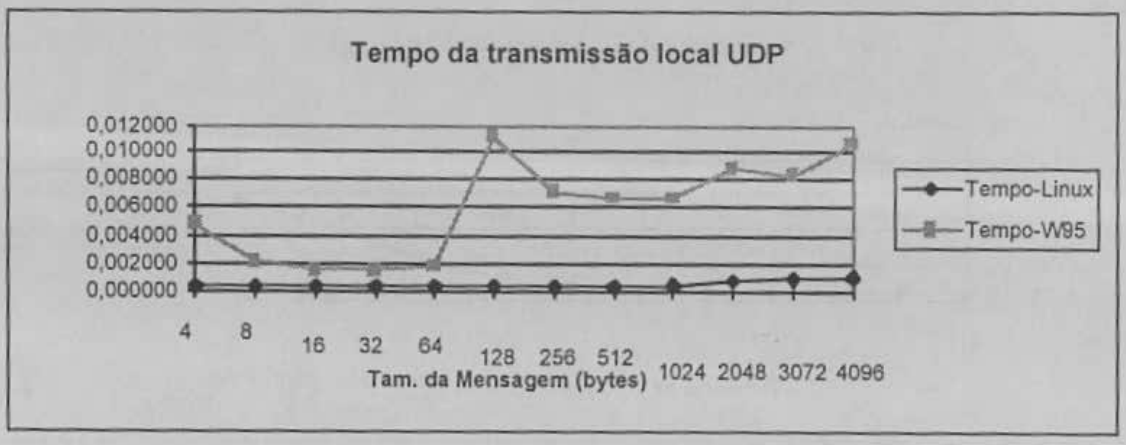


(a)

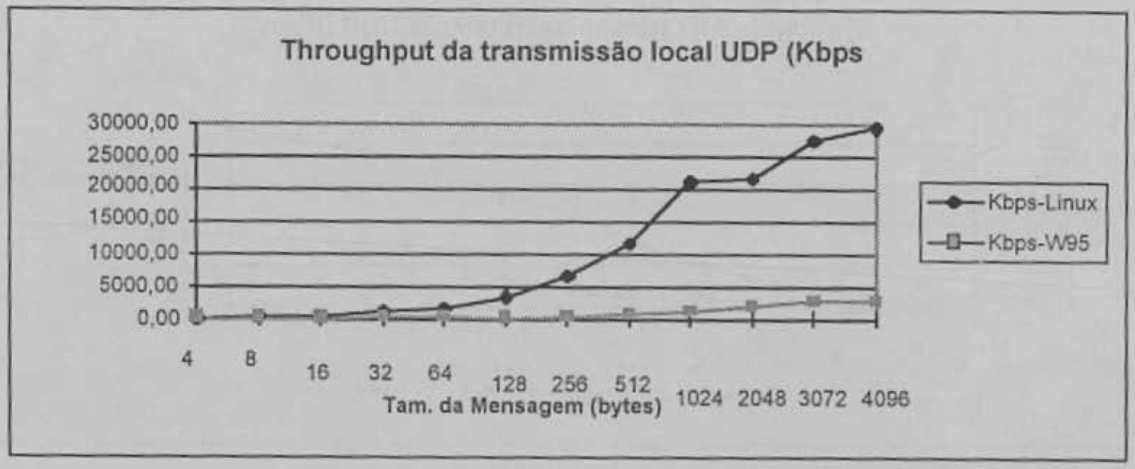


(b)

Figura 4.3 - Transmissão Remota UDP (LINUX x Windows95)



(a)



(b)

Figura 4.4 - Transmissão Local UDP (LINUX x Windows95)

6. Agradecimentos

Os autores agradecem à CAPES, FAPESP e CNPq pelo apoio dado ao Grupo de Pesquisa em Sistemas Distribuídos e Programação Concorrente do ICMSC/USP.

Referências

[ALM94] Almasi, G. S., Gottlieb A., *Highly Parallel Computing*, 2ª ed., The Benjamin Cummings Publishing Company, Inc., 1994

[ALV95] Alves A., Silva, L., Carreira, J., Silva, J., G.: "WPVM: Parallel Computing for the People. Procedings of HPCN'95", High Performance Computing and Networking Conference, in Springer Verlag Lecture Notes in Computer Science, Milan, Itália, May, 1995, pp. 582-587.

[BON96] Bonner, P.: *Network Programming with Windows Sockets*, Prentice-Hall, 1996.

[DIL95] Dillion, E., Santos, C. G. D., Guyard, J., "Homogeneous and Heterogeneous Networks of Workstations: Message Passing Overhead", INRIA/CRIN, Junho, 1995.

[DUM95] Dumas, A.: *Programming WinSock*, Sams Publishing, 1995.

[FIS96] Fischer, M., Dongarra, J., "Another Architecture: PVM on Windows95/NT", 3° EuroPVM'96, Third Euro PVM User´s Group Meeting, Munich, Germany, 7 -9 de Outubro,1996.

[FLY72] FLYNN, M. J., "Some Computer Organizations and Their Effectiveness", IEEE Transactions on Computers, v. C-21, pp.948-960, 1972.

[GEI94a] Geist, A., Beguelin, A., Dongarra J., Jiang, W. Manchek, R., Sunderam V.: *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*, The MIT Press, 1994.

[GEI94b] Geist, A., Beguelin, A., Dongarra J., Jiang, W., Manchek, R., Sunderam V.: "PVM3 User's Guide and Reference Manual", Oak National Laboratory, September, 1994.

[HAL93] Hall, M., Towfiq, M., Geoff, A., Treadwell, D., Henry, S.: *Windows Sockets. An Open Interface for Networtk Programming under Microsoft Windows, version 1.1*, 1993.

[MCB94] McBryan, O. A.: "An overview of message passing environments", Parallel Computing, v. 20, (1994) 417-444.

[NEV96] Nevin, N., "The Performance of LAM 6.0 and MPICH 1.0.12 on a Workstation Cluster", Ohio Supercomputer Center Technical Report, Março, 1996.

[PIN91] Pinson, L. J.,Wiener, R. S.: *Programação Orientada para Objeto e C++*, São Paulo, McGraw-Hill, 1991.

[SAN96] Santana, M.J., Souza, P.S.L., Santana, R.H.C., Souza, S.R.S., "Parallel Virtual Machine for Windows95", Anais do 3° EuroPVM'96, Third Euro PVM User´s Group Meeting, in Springer Verlag Lecture Notes in Computer Science, Munich, Germany, 7-9 de Outubro 1996.

[SOU96] Souza, P. S. L.: "Máquina Paralela Virtual em Ambiente Windows", Dissertação de Mestrado, Instituto de Ciências Matemáticas de São Carlos (ICMSC), Universidade de São Paulo (USP), Maio, 1996.

[STE90] Stevens, W. R.: *UNIX Network Programming*, Prentice Hall International Inc.,1990.

[SUN94] Sunderam, V. S., Geist, A., Dongarra, J., Manchek, R.: "The PVM concurrent computing system: evolution, experiences and trends, Parallel Computing", v. 20, 1994, 531-545.

[TAK90] Takahashi, T., Liesenberg H. K. E., Xavier, D. T.: *Programação orientada a objetos: uma visão integrada do paradigma de objetos*, São Paulo, IME-USP, 1990.

[TAN96] Tanenbaum, A. S., *Computer Networks*, 3ª. ed., Prentice Hall International Inc., 1996.

[WIE91] Wiener, R. S., Pinson, L. J.: *C++ Programação Orientada para Objeto. Manual Prático e Profissional*, São Paulo, McGraw-Hill, 1991.

[ZAL91] Zaluska E. J.: "Research lines in distributed computing systems and concurrent computation", Anais do Workshop em Programação Concorrente, Sistemas Distribuídos e Engenharia de Software, (1991) 132-155, Instituto de Ciências Matemáticas de São Carlos (ICMSC), Universidade de São Paulo (USP), São Carlos, SP, Brasil.