Sistemas de Comunicação Abertos em Redes Locais: Uma Implementação e Medidas de Desempenho em um Ambiente de Computadores Pessoais

Célio Vinicius Neves de Albuquerque [1]

Marcelo Luiz Drumond Lanza [2]

Otto Carlos Muniz Bandeira Duarte [3]

Universidade Federal do Rio de Janeiro
COPPE - Programa de Engenharia Elétrica
P.O. Box 68504 CEP 21945 - Rio de Janeiro - RJ - Brasil
FAX: +55 21 290.6626 Email: otto@coe.ufrj.br

Resumo

Este trabalho descreve uma arquitetura de implementação e analisa alguns resultados de desempenho de um sistema de comunicação conectado a uma rede local do tipo Ethernet. O sistema segue o Modelo de Interconexão de Sistemas Abertos e foi implementado pelo Grupo de Teleinformática e Automação da UFRJ. A implementação contém protocolos normalizados das sete camadas tendo sido concebida para rodar em ambientes de computadores pessoais do tipo IBM-PC, é programada em linguagem C e contém mais de 26.000 linhas de código que geram 134 koctetos de executável. A arquitetura de implementação usada se baseia em interfaces por filas e em estruturas de dados que evitam cópias, assim como, em módulos específicos e otimizados de gerenciamento de memória, de gerenciamento de temporizações e de escalonamento de tarefas. Os resultados obtidos comprovam um bom desempenho do sistema.

Abstract

This work presents an implementation architecture and analyses some performance results of a LAN communication system. The system is based on the Reference Model for Open System Interconection and was designed by the Grupo de Teleinformática e Automação of UFRJ. The system uses normalized protocols of the seven layers, runs on IBM Personal Computer environments and consists of more than 26.000 lines of C language code that provide 134 kbytes of execution code. The implementation architecture is based on interfaces and on data structures that avoid unnecessary copies, as well as, specialysed modules of memory and timer managers and task scheduling. The throughput results shows a good system performance.

[1] Aluno de mestrado da COPPE/UFRJ Email: celio@coe.ufrj.br

[2] Professor do Departamento de Eletrônica da EE/UFRJ Email: lanza@coe.ufrj.br

[3] Professor da COPPE/EE/UFRJ atualmente em pós-doutorado no Laboratório MASI
Laboratoire MASI - Equipe Réseau et Performance - Université Pierre et Marie Curie
4 Place Jussieu - 75252 Paris Cedex 05 - France - Fax: +33 1 44.27.62.86 Email: duarte@masi.ibp.fr

1 Introdução

Com o desenvolvimento das Redes de Computadores a partir do meio da década de setenta surgiu a necessidade da criação de padrões que permitissem a comunicação entre sistemas heterogêneos. Afim de suprir esta necessidade, a *International Organization for Standardization* (ISO) propôs o Modelo para Interconexão de Sistemas Abertos (RM-OSI). Este modelo define três níveis de abstrações (arquitetura, serviços e protocolos) que são utilizados para descrever um sistema aberto, suas relações e restrições. A técnica básica de estruturação da arquitetura RM-OSI consiste na divisão lógica do conjunto de funções relacionadas à comunicação entre processos de aplicação, em um número de camadas ordenadas e bem definidas. Um sistema aberto tem como arquitetura final sete camadas hierárquicas e independentes que interagem segundo o conceito de "mais valia", onde a camada inferior oferece um serviço à camada imediatamente superior.

O Grupo de Teleinformática e Automação (GTA) da UFRJ vem desenvolvendo uma série de trabalhos relacionados com o Modelo OSI. No que se refere a implementação de um sistema aberto, o GTA desenvolveu os protocolos de Correio Eletrônico (*Message Handling System - MHS*, de Transferência de Arquivos (*File Transfer Access and Management - FTAM*), Elemento de Serviço de Controle de Associação (ACSE), Elemento de Serviço de Transferência Confiável (RTSE), de Apresentação, de Sessão, de Transporte (Classes 0, 2 e 4), de Rede em modo não-conectado (CLNS), de Enlace (*Link Access Procedures on Channel D - LAPD, Logical Link Control - LLC, Medium Access Control - MAC*) e uma carta de comunicação serial. Todos estes trabalhos seguem uma arquitetura de implementação definda pelo GTA que visa priveligiar o desempenho sendo ao mesmo tempo flexível e modular.

Este trabalho descreve a arquitetura de implementação usada e analisa os resultados de desempenho obtidos para a transferência de dados entre dois usuários conectados à uma rede local do tipo Ethernet. As medidas são realizadas camada por camada afim de permitir uma análise referente à cada protocolo.

2 O Sistema de Comunicação Implementado

No que concerne Redes Locais de Computadores a ISO adotou os padrões desenvolvidos no Projeto IEEE-802 que correspondem às camadas Enlace e Física do modelo RM-OSI (Figura 1). O sistema de comunicação implementado e que será objeto deste trabalho tem o perfil mostrado na Figura 2.

2.1 A Subcamada MAC

A subcamada de Controle de Acesso ao Meio (MAC), da arquitetura IEEE 802, fornece os serviços que permitem disciplinar o compartilhamento de um meio de transmissão comum

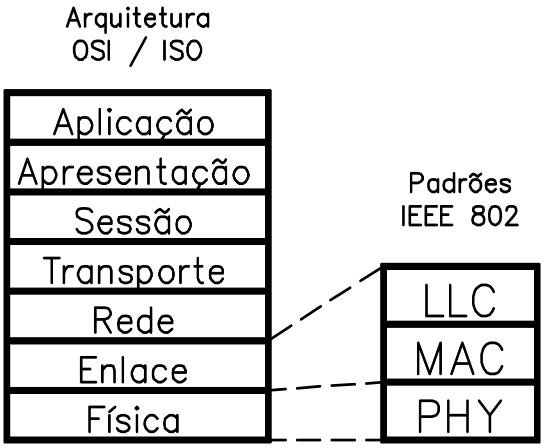

Figura 1: O Modelo RM-OSI.

aos sistemas usuários da rede. A existência da subcamada MAC na arquitetura IEEE 802 reflete uma característica própria das redes locais, que é a necessidade de gerenciar enlaces com origens e destinatários múltiplos compartilhando o mesmo meio físico de transmissão. A camada MAC implementada segue a norma IEEE 802-3 [1] para redes locais de acesso múltiplo com escuta de portadora e detecção de colisão (CSMA-CD), mais conhecida como rede Ethernet. Esta camada interage diretamente com as cartas controladoras encontradas no comércio. Afim de se obter uma maior independência em relação aos fabricantes destas cartas controladoras, a implementação realizada se serve de um conjunto de rotinas *freeware* conhecidas como *Packet Driver SPEC*. Este *software* permite o acesso a diferentes cartas controladoras Ethernet para computadores pessoais do tipo IBM-PC. Neste trabalho foram usadas as rotinas para as placas do tipo Ethernet NE1000.

2.2 A Subcamada LLC

O padrão IEEE 802.2 [2] especifica um protocolo de Controle de Enlace Lógico (*LLC - Logical Link Control* para uso com qualquer um dos outros padrões de controle de acesso ao meio. Existem dois tipos de procedimentos LLC. O LLC Tipo 2, é inspirado no protocolo de enlace HDLC (*High-level Data Link Control*) sendo necessário o estabelecimento de uma conexão ponto à ponto e possui funções de controle de fluxo e recuperação de erros. O LLC Tipo 1 oferece um serviço em modo não conectado permitindo além do endereçamento individual, a difusão parcial e a difusão total. Neste trabalho apenas o LLC Tipo 1 é considerado.

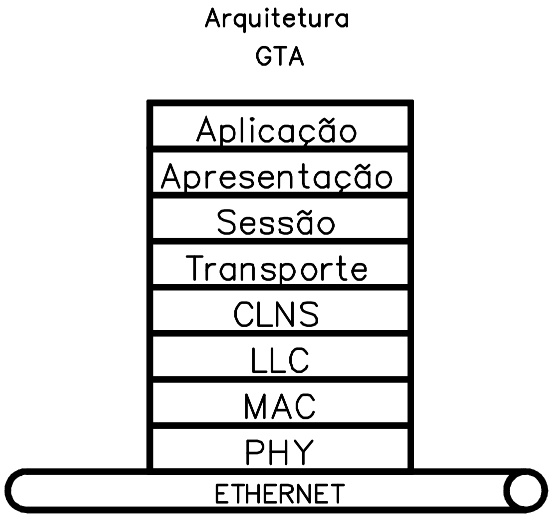

Figura 2: O Perfil de Protocolos do Sistema Implementado.

2.3 A Camada Rede

A ISO estrutura a camada Rede em três sub-camadas [3]: Acesso da Sub-Rede, Convergência Dependente da Sub-Rede, Convergência Independente da Sub-Rede. O protocolo implementado é o protocolo para prover o Serviço de Rede em Modo não conectado (*Protocol for Providing the Connectionless-Mode Network Service - CLNS* [4]) que se trata de um Protocolo de Convergência Independente da Sub-rede (SNCIP). Além das funções de roteamento e encaminhamento este protocolo possui segmentação, roteamento pela origem, registro do caminho e prioridade. Nesta experiência estas opções não foram ativadas e o roteamento é trivial pela comunicação ser interna à rede local.

2.4 A Camada Transporte

A principal função da Camada Transporte é prover o transporte de dados de uma extremidade à outra da comunicação (fim-à-fim) de uma maneira confiável (com a qualidade de serviço exigida pelo usuário) e de forma otimizada (melhor relação custo/benifício). Ela tem a função de adaptar os serviços requisitados pelo seu usuário levando em consideração serviços oferecidos pela Camada Rede. Para executar esta função ela dispõe de 5 protocolos que garantem a integridade dos dados transferidos. Foram implementados os protocolos classes 0, 2 e 4. O protocolo Classe 0 é o mais simples. Ele possui a função de segmentação/remontagem e se desconecta ao detectar um erro. O Classe 2 inclui as funções de multiplexagem, numeração, concatenação/separação e controle de fluxo. O Classe 4 é o mais completo incluindo a detecção com recuperação de erros. Neste trabalho apenas o Protocolo de Transporte Classe 2 foi analisado.

2.5 A Camada Sessão

A Camada Sessão é a primeira das chamadas camadas altas e provê um serviço orientado ao usuário. Ela tem como objetivo fornecer os serviços para organizar e sincronizar a troca de dados entre duas entidades de Apresentação. Os principais conceitos envolvidos na oferta destes serviços são: *tokens*, gerenciamento de diálogo, sincronização e ressincronização, atividades, liberação ordenada e transferência dados. O Protocolo de Sessão implementado [5] tem a sua máquina de estados em forma de tabela, diferentemente de todas as outras máquinas de estados que usam a estrutura convencional do tipo *if then else*. Esta opção de implementação se mostrou conveniente pois apesar do enorme tamanho da máquina de estados do Protocolo de Sessão (80 eventos e 32 estados) a tabela se mostrou compacta, bem estruturada e com boa legibilidade.

2.6 As Camadas Apresentação e Aplicação

A Camada Apresentação provê uma representação comum a ser usada entre as entidades de Aplicação em sua comunicação. Para se obter este resultado, a camada pode efetuar transformações de sintaxe, de maneira a realizar uma sintaxe de tranferência comum entre os correspondentes. Só os problemas de sintaxe são tratados, sem referência à semântica que é função da Camada Aplicação. A Camada Apresentação oferece às entidades de Aplicação, além dos servicos específicos a esta camada, o acesso aos serviços da Camada Sessão. Neste trabalho, estão disponíveis todos os serviços relativos à Camada Sessão, mas não são oferecidos os serviços de alteração, definição e criação de contexto.

A Camada Aplicação tem uma arquitetura particular composta de Elementos de Serviços comuns (ACSE, RTSE, ROSE) e específicos (MHS, FTAM, etc.). As diferentes possibilidades de associação destes elementos entre si e com a Camada Apresentação torna a interface desta camada peculiar e sua implementação será explicada na seção que se segue.

3 A Arquitetura de Implementação

A arquitetura de implementação [6] usada define uma interface padrão entre as camadas visando normalizar o acesso aos serviços oferecidos pelas camadas adjacentes, possibilitar uma boa flexibilidade e modularidade, e além disto, conseguir um baixo grau de acoplamento entre as camadas (Figura 3). Estes objetivos foram considerados primordiais afim de facilitar a repartição do trabalho de implementação de cada camada. Esta interface consiste de um par filas FIFO (*first in first out*) por conexão e de um conjunto de rotinas de acesso às estruturas de dados.

A interface entre as subcamadas de Enlace MAC e LLC contém um par de *buffers* circulares onde são copiados de uma forma contígua os dados que deverão ser enviados ao

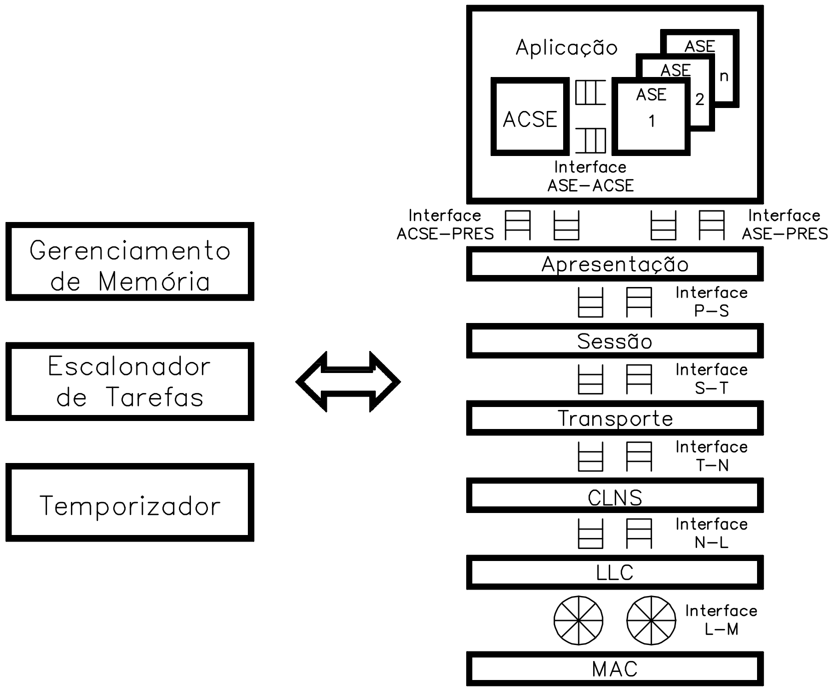

Figura 3: A Arquitetura de Implementação GTA.

controlador de comunicação. Este último é quem se encarrega da transmissão no meio físico.

A abertura de uma conexão corresponde a instanciar um par de filas na interface entre as camadas por onde veicularão todas as primitivas de serviço referentes a esta conexão. No entanto, a Camada Aplicação tem uma característica singular pois é composta de vários Elementos de Serviço que podem se associar de diversas formas entre eles e com a Camada Apresentação. A interface entre a Camada Aplicação e a Camada Apresentação se faz através do ACSE e de um outro Elemento de Serviço. Assim, foram criadas as filas ACSE-PRES e ASE(n)-PRES na interface Aplicação-Apresentação e filas ACSE-ASE(n) e ASE(n_1)-ASE(n_2) para a interface entre os Elementos de Serviço de Aplicação sendo o parâmetro n um identificador de um Elemento de Serviço.

Cada fila é formada por um elemento básico composto de estruturas de dados encadeadas, definidas de forma a evitar a cópia de dados que é um fator relevante de degradação de desempenho em um sistema de comunicação [7, 8, 9]. A Figura 4 mostra a estrutura de dados definida que se baseia em ponteiros e separa estruturas de tamanho fixo das estruturas de tamanho variável para possibilitar um gerênciamento de memória mais eficaz. O campo Num_Alocacoes é uma variável que é incrementada/decrementada toda a vez que uma camada aloca/desaloca esta estrutura sendo que os dados só são realmente apagados da memória quando esta variável atinge o valor zero. Este artifício foi fundamental para

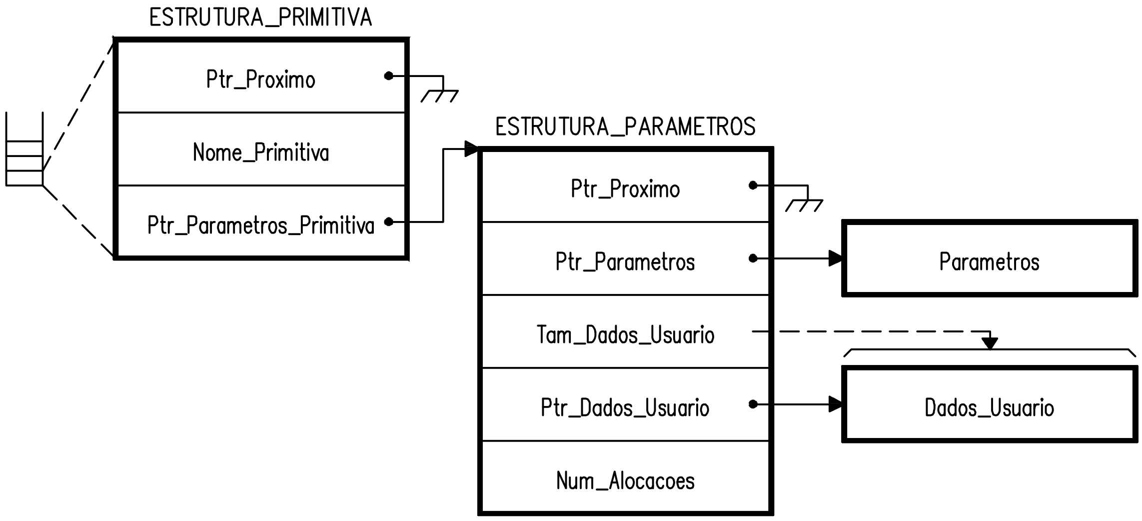

Figura 4: Estrutura de Dados do Sistema.

se evitar cópias quando na utilização de protocolos que contém recuperação de erros por retransmissão (protocolos de Enlace, Transporte, Sessão e RTSE).

A inserção/retirada de PCIs (informações de controle de protocolo) à/de cada camada (ou subcamada) implica necessáriamente em um gerenciamento de memória eficiente para se obter um bom desempenho. É importante também que se consiga segmentar/remontar dados sem a necessidade de realizar cópia, fazendo apenas manipulações de ponteiros. Estas razões levaram à implementação de um Gerenciador de Memória específico e otimizado. Ele é dividido em uma região de memória para a parte transmissora e outra para a parte receptora. Afim de evitar o excesso de fragmentação, estas regiões são ainda subdivididas de forma a permitir alocações/dealocações de memória de tamanho fixo e variável e ainda, utilisa-se algorítimos otimizados que levam em conta o comportamento específico de um sistema OSI.

A primeira versão deste sistema continha um programa principal que testava as filas das interfaces afim de verificar se existiam elementos (primitivas de serviço) para serem tratados. O desempenho deste procedimento se mostrou bastante degradado. Um grande salto de desempenho foi obtido com a criação do módulo "Escalonador de Tarefas". Este módulo é constituido por apenas uma fila contendo a sequência de funções que deverão ser executadas. Para entender o funcionamento do sistema é necessário lembrar que um protocolo nada mais é que uma Máquina de Estados onde a partir de um evento (recebimento de uma primitiva de serviço) e do "estado atual" executa-se (ou não) algumas funções e muda-se (ou não) para um novo estado criando-se (ou não) um ou mais novos eventos (envio de primitivas às

Tabela 1: Número de Linhas do Programa e Tamanho do Executável.

Camada	**Código Fonte (linhas)**	**Código Executável (koct.)**
Escalonador de Tarefas	155	
Gerenciador de Memória	965	
Gerenciador de Temporizações	350	
MAC	1.120	46
LLC	1.240	49
CLNS	2.290	56
Transporte	2.550	65
Sessão	15.130	129
Apresentação	2.930	134

camadas adjacentes). Sendo assim, na arquitetura descrita a cada primitiva de serviço que é colocada em uma fila das interfaces (criação de um evento) a função de tratamento desta primitiva é colocada na fila do Escalonador de Tarefas. A atomização exigida na execução da máquina de estados fica também facilmente solucionada.

A implementação de protocolos inclui procedimentos baseados em temporizações utilizados nas funções de recuperação de erros, controle de tempo máximo de estabelecimento e liberação de conexão etc.. A maneira na qual foi implementado o Gerenciador de Temporizações foi baseda na proposta de Varghese e Lauck [10], onde uma analogia a um relógio real é feita através de um contador em módulo. Esta implementação se mostrou bastante eficiente evitando testes lógicos desnecessários a cada interrupção de *hardware* viabilizando a criação de um grande número de temporizadores em tempo de execução. O estouro de um temporizador é tratado na arquitetura implementada da mesma forma que uma primitiva de serviço, ou seja, a indicação do estouro (evento) é colocada na fila da interface e a função que deve ser executada na fila do Escalonador de Tarefas.

O sistema implementado foi programado em liguagem C e roda em ambientes de computadores pessoais do tipo IBM-PC com sistema operacional MS-DOS. A Tabela 1 mostra o número de linhas de código de cada camada e o tamanho do código executável (sem informações de depuração) computado acumulativamente à cada camada.

4 Medidas de Desempenho

Para avaliar o desempenho dos protocolos foram feitas medidas de vazão, que correspondem à relação entre o volume de informação e o tempo necessário para transferí-lo. Entende-se por volume de informação o produto do número total de quadros de informação enviados

Tabela 2: Tempo Total de Transferência (segundos) do Volume de Informações.

IBM PC-AT 286 12 MHz

Volume de informação
10.000 mensagens

Camada	**Tamanho (oct.)**			
	128	**256**	**512**	**1024**
MAC	20.3	22.3	27.4	38.2
LLC	29.6	32.3	37.5	48.1
CLNS	47.3	49.7	52.7	62.5
Transporte	92.0	94.5	100.1	110.4
Sessão	107.6	111.6	114.5	124.7
Apresentação	114.0	117.3	120.3	130.2

por seus respectivos tamanhos. Estas medidas de vazão foram feitas para cada camada do sistema implementado e a nível de usuário do serviço da camada. Portanto, elas incluem o *overhead* introduzido pela criação/destruição das mensagens, envio/retirada das mensagens nas/das filas, alocação e dealocação de memória, etc.. Para cada mensagem enviada são introduzidos *overheads* devido aos cabeçalhos das camadas inferiores. Este *overhead* chega à 33 octetos para o usuário da camada Apresentação. Na metodologia empregada, considera-se que o tempo total de transferência do volume de informações é o tempo decorrido entre a transmissão da primeira mensagem e o reconhecimento da última mensagem. O tempo medido é obtido com o auxilio de funções baseadas no relógio de um IBM-PC, o que limita sua precisão a 55 ms. Sendo assim, para tornar o erro de medida desprezível, transfere-se um volume de informações composto de 10.000 quadros, o que torna o tempo de transferência muitas vezes superior ao período usado como unidade básica de tempo.

As medidas de desempenho foram realizadas entre dois IBM-PC AT 286 12 MHz. Vale salientar que as medidas foram efetuadas durante uma condição de baixo tráfego da rede local, ou seja, praticamente o tráfego era constituido pelos quadros envolvidos no procedimento de medidas.

A Tabela 2 indica os tempos de transferência obtidos para cada camada. A Tabela 3 mostra os valores covertidos em vazão. A análise destas tabelas revela alguns pontos importantes.

A Camada MAC é responsável pela passagem dos parâmetros e dados que serão transferidos (por cópia ou acesso direto à memória) da memória do microcomputador para a memória da carta Ethernet e vice-versa. Esta é normalmente a limitação inicial do sistema pois normalmente o microcomputador não consegue acompanhar a velocidade da carta controladora Ethernet e muitas cartas controladoras não conseguem transmitir/receber continuamente na taxa máxima teórica da norma Ethernet (6 Mbps para quadros de tamanho mínimo e 9.8 Mbps para quadros de tamanho máximo). Dos resultados obtidos pode-se verificar que o

Tabela 3: Vazão do Sistema (kbps) oferecida ao Usuário de cada Camada.

	Camada	Tamanho (oct.)			
IBM PC-AT 286 12 MHz		**128**	**256**	**512**	**1024**
	MAC	502	916	1492	2141
Volume de informação	**LLC**	345	633	1089	1702
10.000 mensagens	**CLNS**	216	411	776	1309
	Transporte	139	270	511	926
	Sessão	118	229	446	821
	Apresentação	112	218	425	786

Tabela 4: Comparação do Desempenho da Camada MAC em Microcomputadores Diferentes.

		Processador	Tamanho (oct.)			
IBM PC-AT			**128**	**256**	**512**	**1024**
Volume de informação 10.000 mensagens	**Vazão (kbps)**	**386 SX 20MHz**	832	1.422	2.212	3.055
		286 12MHz	502	916	1.492	2.141
	Tempo (seg.)	**386 SX 20MHz**	12.3	14.4	18.5	26. 8
		286 12MHz	20.3	22.3	27.4	38.2

microcomputador (processamento dos protocolos) é a limitação preponderante.

Existe um processamento que é comum à todas as camadas e consiste em receber uma primitiva de serviço, decodificá-la e tratá-la. Quando o tratamento implica na geração de uma (ou mais) primitiva(s) deve-se acrescentar o PCI, montar uma primitiva e colocá-la em uma fila da interface. Isto exige um tempo de processamento devido ao gerenciamento da interface (testes de filas e inserções/retiradas de elementos), gerenciamento de memória (alocações/dealocações) e execução da máquina de estados que provoca uma perda de desempenho a cada camada. Particular a camada MAC existe a tranferência dos dados da memória do micro para a carta controladora (e vice-versa) e por isto os resultados são dependentes do tamanho do quadro. Obviamente, o resultado do desempenho é dependente do microcomputador (tipo e velocidade do processador, memória *cache* etc.). A Tabela 4 mostra o resultado do desempenho da Camada MAC para um microcomputador mais veloz.

Verifica-se que a Camada LLC é responsável por uma queda de desempenho que depende do tamanho do quadro. Isto se justifica pelo fato desta camada ser responsável pela cópia dos dados encadeados para uma região contígua de memória (*buffer* de transmissão da interface

Tabela 5: Vazão (kbps) com a Camada Transporte segmentando em 100 octs.

IBM PC-AT 286 12 MHz

Volume de informação
10.000 mensagens

Camada	**Tamanho (oct.)**		
	256	**512**	**1024**
Transporte	105	106	114
Sessão	103	104	111
Apresentação	89	96	107

LLC-MAC). A cópia de dados é um fator importante de degradação de desempenho, porém todos os controladores de comunicação exigem que os dados a serem transferidos estejam em uma área contígua de memória. Esta cópia pode ser evitada se nas camadas superiores se "reservasse" um espaço contíguo em memória para os "dados do usuário" e as PCIs que deverão ser inseridas por cada camada. No entanto, este procedimento pode levar à um grande desperdício de memória pois, obrigatoriamente, o comprimento máximo de quadro teria que ser reservado. Um outro argumento contrário a utilização dete procedimento é a necessidade das camadas superiores conhecerem o tamanho máximo do quadro que pode ser transmitido, pois este fato vai de contra ao princípio de independência das camadas defendido no modelo RM-OSI.

A Camada Transporte é responsável pela maior queda de desempenho. Isto já era esperado pois, na fase de transferência de dados e para o perfil de protocolos utilizado, é a camada que executa o maior número de funções. Este protocolo exige a criação de Unidades de Dados de Protocolo (PDUs) de reconhecimento para as PDUs de dados enviadas e isto onera sobremaneira o processamento do sistema.

Da Tabela 2, pode-se observar que a perda de desempenho devido a Camada Rede e a Camada Sessão é maior que a provocada pela Camada Apresentação. Em princípio poderia se pensar que esta última executa as mesmas funções mas a Sessão e a Rede possuem facilidades (segmentação/remontagem) que mesmo não usadas consomem tempo em testes. Além disso, a Camada Sessão concatena/separa a PDU de dados com/da PDU de *token*.

A Tabela 5 mostra os resultados de vazão quando os dados são segmentados/remontados na Camada Transporte. Devido ao grande *overhead* introduzido pelo tratamento das PDUs geradas pela segmentação, a vazão fica praticamente constante independentemente do tamanho da mensagem a ser segmentada.

Os resultados obtidos podem ser considerados bastante satisfatórios com valores de vazão que atendem a maioria das aplicações comerciais existentes. Se por um lado o Modelo OSI pode ser criticado pelo *overhead* introduzido para se transferir dados entre dois usuários, por outro lado, há que se levar em conta o valor de uma interconectividade devido à utilização de protocolos normalizados, e ainda, à quantidade de facilidades oferecidas por este sistema que até hoje foram muito pouco exploradas.

5 Conclusão

O desempenho de um sistema de comunicação é extremamente dependente da implementação. Portanto, uma escolha criterioza da arquitetura de implementação é fundamental para a obtenção de altas velocidades de transferência. Este trabalho descreveu a arquitetura de implementação usada no desenvolvimento de um sistema aberto.

A padronização da interface e o seu modelo baseado em filas permitiu uma alta flexibilidade e uma grande modularidade. Consequentemente, foi possível uma melhor repartição do trabalho e a reutilização de alguns programas de testes em várias camadas diferentes ajudando a depuração do sistema.

Um ganho significativo em desempenho deve ser creditado à estrutura de dados usada que eliminou a necessidade de cópias e permitiu a implementação de um gerenciador de memória específico e otimizado. O Gerenciador de Temporizações se mostrou bastante eficaz.

Conseguiu-se um grande salto de desempenho quando se introduziu na arquitetura o escalonador de tarefas. A idéia básica de se colocar sequencialmente em uma fila única os eventos de todo o sistema evitou os testes repetitivos nas filas das interfaces. Esta opção parece ser fundamental em sistemas monotarefas e, segundo alguns testes já realizados no GTA em ambiente UNIX, seu emprego em sistemas multitarefas evita perdas em desempenho devido a mudança de contexto.

A partir dos resultados obtidos pôde-se constatar que o gerenciamento de memória (inserções e retiradas de PCIs), gerenciamento das interfaces (testes, inserções e retiradas de elementos das filas) são procedimentos altamente custosos em tempo. São estes pontos que devem ser investigados afim de se conseguir desempenhos significativos em qualquer sistema de comunicação que siga a arquitetura em camadas independentes.

As vazões obtidas são bastante satisfatórias e demonstram que com estações de baixo custo (microcomputadores pessoais compatíveis IBM-PC) pode-se implementar aplicações distribuídas de bom desempenho.

6 Agradecimentos

Os autores agradecem a todas as pessoas do Grupo de Teleinformática e Automação e em especial aos responsáveis e implementadores de alguns dos protocolos tais como: Prof. João Amaro Baptista Pereira, Luis Felipe Baginski, José Ferreira de Rezende, Rainer Schatzmayr, Rogério Leone Teixeira da Cunha, Fernando Mascarenhas Cavalcanti de Barros, Frederico dos Santos Liporace, Marcelo Dias Nunes e Marcelo Macedo Achá.

Referências

[1] I. P. 802, *Draft IEEE standard 802.3: CSMA/CD access method and physical layers specifications.* IEEE, Revision D, 1982.

[2] I. P. 802, *Draft IEEE standard 802.2: LLC Logical Link Control specification.* IEEE, Revision D, 1982.

[3] DIS8648, *Internal Organization of the Network Layer.* International Standard Organization, Genebra, 1985.

[4] DIS8473, *Protocol for Providing the Connectionless-Mode Network Service.* International Standard Organization, Genebra, 1985.

[5] J. F. de Rezende e O. C. Duarte, "Implementação e análise de desempenho do protocolo de sessão", in *X Simpósio Brasileiro de Redes de Computadores*, Recife, pp. 289–302, abril 1992.

[6] L. F. Baginski e O. C. Duarte, "Um modelo de implementação de alto desempenho para sistemas abertos", in *IX Congresso da Sociedade Brasileira de Telecomunicações*, São Paulo, pp. 19.3.1–19.3.5, setembro 1991.

[7] L. F. Baginski, F. M. C. de Barros, R. Schatzmayr e O. C. Duarte, "Implementação e análise de desempenho de um sistema de transferência de dados", in *X Congresso da Sociedade Brasileira de Computação*, Vitória, pp. 157–169, julho 1991.

[8] C. M. Woodside e J. R. Montealegre, "The effect of buffering strategies on protocol execution performance", *IEEE Transactions on Communications*, vol. COM-37, no. 6, pp. 545–553, junho 1989.

[9] L. Svobodova, "Implementing OSI system", *IEEE Journal on Selected Areas in Communications*, vol. JSAC-7, no. 9, pp. 1115–1130, setembro 1989.

[10] G. Varghese e T. Lauck, "Hashed and hierarchical timing wheels: Data structures for the efficient implementation of a timer facility", in *Proccedings of the 11th ACM Symposium on Operating Systems Principles, ACM Operating Systems Review*, Austin TX, pp. 25–38, novembro 1987.