Instituto de Ciências Matemáticas de São Carlos

ISSN - 0109-2569

Mecanismo de comunicação no LaSD-SPP

ANA LUISA PENA DE ANDRADE
MARCOS JOSE SANTANA
NIVALDI CALONEGO JR
ONOFRE TRINDADE JR
REGINA H. C. SANTANA

Nº 26

RELATÓRIOS TÉCNICOS DO ICMSC

SÃO CARLOS
Out. / 1994

SYSNO 872268
DATA / /
ICMC - SBAB

Mecanismo de Comunicação no LaSD-SPP

Ana Luísa Pena de Andrade
Marcos José Santana
Nivaldi Calônego Jr.
Onofre Trindade Jr.
Regina H. C. Santana

São Carlos
Setembro - 1994

Agradecimentos

À Capes, à Fapesp e ao CNPq, pelo apoio financeiro.

Índice

Lista das Figuras ii
Índice das Tabelas ii
Índice dos Gráficos iii
Glossário iv
Resumo vii
Abstract viii

1 - Introdução 01

2 - Revisão/Ampliação da Implementação Original do SPP 05

2.1 Organização do SPP 06
2.1.1 Hardware 06
2.1.2 Software 07
2.2 Decisões de Projeto 09
2.3 Projeto Lógico 11
2.4 Considerações Finais 15

3 - Utilização do Protocolo TCP/IP no Ambiente SPP 16

3.1 Disponibilidade/Flexibidade da Utilização do SPP em função de um protocolo mais difundido 16
3.2 O Programa *net.exe* 17
3.3 Projeto Lógico 18
3.3.1 Dificuldades Encontradas 20
3.3.2 Alterações Efetuadas no Programa *net* 20
3.3.3 Utilização dos Transputers 22
3.4 Considerações Finais 24

4 - Análise dos Resultados 25

4.1 Técnica Utilizada para o Teste 25
4.2 Testes Realizados - SPP 27
4.2.1 Primeiro Teste Realizado 27
4.2.2 Segundo Teste Realizado 29
4.3 Testes Realizados - Implementação TCP/IP 31
4.4 Considerações Finais 32

5 - Conclusão 33

5.1 Dificuldades Encontradas 33
5.2 Contribuições Deste Trabalho 34
5.3 Trabalhos Futuros 34
5.4 Comentários Finais 35

Referências Bibliográficas 36

Lista das Figuras

1.1 Rede Local com Segmentos Interligados por Pontes 03

2.1 Estrutura do SPP 07
2.2 Seqüência de Requisições 12
2.3 Estrutura em Camadas do Software 14

3.1 Ambiente de Desenvolvimento 17
3.2 Diagrama de Estrutura Modular - Programa *net* 19
3.3 Diagrama de Estrutura Modular - Programa *net* modificado 21
3.4 Diagrama de Estrutura Modular - Programa *net* (SPP - servidor) 21
3.5 Diagrama de Estrutura Modular - Programa *net* (SPP - estação) 22

Índice das Tabelas

3.1 Serviços Disponíveis no Sistema *net* 18
3.2 Organização Lógica da Implementação do Programa *net* 19

4.1 Dados Obtidos para o Primeiro Teste - Login 27
4.2 Dados Obtidos para o Primeiro Teste - Logout 28
4.3 Dados Obtidos para o Primeiro Teste - Lot 28
4.4 Dados Obtidos para o Primeiro Teste - Dlot 29
4.5 Dados Obtidos para o Segundo Teste - Sessão Mínima 30
4.6 Faixa de Tamanho de Arquivos para Teste 31

4.7 Resultados Comparativos com TCP/IP 31

Índice dos Gráficos

4.1 Análise das Primitivas para o Primeiro Teste 29
4.2 Análise de uma Sessão Mínima para as duas Semânticas 30
4.3 Análise das Três Versões Implementadas 32

Glossário

BT	Banco de Transputers. Constituído pelos processadores, controlador de banco, controladores de módulo, rede de chaveamento e componentes do software do sistema relacionados com a sua operação básica. Apresenta uma estrutura expansível, baseada em módulos.
CB	Controlador do Banco. Computador responsável pelo controle do banco de Transputers. Comunica-se com os controladores de módulo e com o processador de entrada.
CM	Controlador de Módulo. Computador responsável pelo controle de um módulo do banco de Transputers. Comunica-se com outros controladores de módulo e/ou com o controlador do banco.
CRC	"Cyclic Redundancy Code". Código binário utilizado para detecção de correção de erros, em particular, em pacotes de dados ethernet.
CSP	"Communicating Sequential Processes". Linguagem desenvolvida por Hoare para especificação de sistemas concorrentes.
DEM	Diagrama de Estrutura Modular. Diagrama utilizado em projeto estruturado para representar os diversos módulos de um sistema e a comunicação entre eles.
DFD	Diagrama de Fluxo de Dados. Diagrama utilizado em análise/projeto estruturados para representar ações sobre fluxos de dados em um sistema.
ET	Estação de Trabalho.
ISERVER	Módulo de software servidor do sistema de desenvolvimento TDS.
ISO	"International Standard Organization". Organização internacional voltada para o estabelecimento de normas técnicas da qual o Brasil figura como país membro.

LLC	"Logical Link Control". Subcamada de mais alto nível da camada de enlace de dados no modelo de referência ISO-OSI.
LT	Lote de Transputers. Conjunto de processadores Transputers alocados para um usuário do servidor de processamento paralelo.
MAC	"Medium Access Control". Subcamada de mais baixo nível da camada de enlace de dados no modelo de referência ISO-OSI.
NCSA	National Center for Supercomputer Applications.
OCCAM	Linguagem de alto nível baseada na linguagem CSP. Dispõe de mecanismos intrínsecos para a sincronização de processos concorrentes.
OSI	"Open Systems Interconnection" - Reference Model. Modelo de referência ISO para conexão de sistemas abertos.
PD	"Packet Driver" - NCSA . Mecanismo para acesso e multiplexação de pacotes de dados implementado na universidade de Clarkson.
PE	Processador de Entrada. Computador responsável pela comunicação do servidor de processamento paralelo com a rede local e pelo atendimento de parte dos serviços disponíveis no SPP.
PKT	Protocolo de comunicação utilizado pelo "Packet Driver" -NCSA.
RC	Rede de Chaveamento. Malha composta por chaves eletrônicas utilizada para o roteamento de dados entre os processadores.
REDE LOCAL	Componentes de hardware e software relacionados com a conexão de estações de trabalho e servidores.
RESET	Sinal de reiniciação aplicado em Transputers para o estabelecimento das condições iniciais de funcionamento.
RPC	"Remote Procedure Call". Mecanismo para chamada de procedimentos remotos. Utilizado neste trabalho para implementar a comunicação entre os pares servidor - cliente.
SP	"Server Protocol" - INMOS. Protocolo utilizado para comunicação entre o programa iserver e redes de Transputers.

SISTEMA SPP	Todos os itens de hardware e software relacionados com a operação do servidor de processamento paralelo.
SPP (1)	Servidor de Processamento Paralelo. Constituído pelo banco de Transputers, pelo processador de entrada, processadores, e componentes do software do sistema responsáveis pela operação básica desses.
SPP (2)	Protocolo de comunicação utilizado pelo mecanismo de chamadas a procedimentos remotos utilizado no servidor de processamento paralelo.
TCP/IP	"Transmission Control Protocol/Internet Protocol". Conjunto de protocolos de comunicação utilizados em redes de computadores locais e de grande distância.
TDS	"Transputer Development System". Sistema para o desenvolvimento de software em Transputers. Inclui editor de textos baseado em dobras ("folds"), compilador OCCAM, editor de ligação, depurador, carregadores e diversos outros programas utilitários.
TRANSPUTER	Microcomputador fabricado pela INMOS. Possui características que o tornam indicado como elemento básico na implementação de máquinas paralelas.
UC	Unidade de Controle. Subsistema de um computador responsável pela busca, interpretação e execução das instruções.
UP	Unidade de Processamento. Subsistema lógico de um computador responsável pela execução das instruções.

Resumo

Uma arquitetura para processamento paralelo baseada em um banco de transputers, foi desenvolvida e implementada com sucesso no LaSD (Laboratório de Sistemas Digitais) ICMSC-USP [TR91]. Essa arquitetura (SPP - Servidor de Processamento Paralelo) introduz recursos de processamento paralelo para os usuários de um sistema distribuído baseado em rede local de computadores.

Neste trabalho são revisados alguns pontos básicos que ficaram pendentes na implementação original do software do SPP. Uma nova semântica de comunicação é implementada ("*at-least-once*") e os resultados obtidos na comparação com a semântica original ("*at-most-once*") são analisados.

É apresentada uma nova implementação das primitivas de comunicação do SPP baseada na utilização do protocolo TCP/IP, visando maior abrangência no acesso remoto ao sistema.

O trabalho discute as diretrizes básicas seguidas no projeto, implementação e os resultados obtidos nos testes efetuados.

Abstract

An architecture for parallel processing based on a transputer bank was developed and successfully implemented at LaSD (Laboratório de Sistemas Digitais) ICMSC-USP [TR91]. This architecture (SPP- Servidor de Processamento Paralelo) provides parallel processing resources to users of a distributed LAN-based computing system.

This work presents a review of the original SPP software implementation. A new communication semantics ("at-least-once") is implemented and discussed and the results obtained from the comparison to the original semantics ("at-most-one") are also discussed.

A new implementation of the SPP communication primitives using the TCP/IP protocol is also introduced aiming at increasing the remote access facilities to the system.

This work also presents the basic approach used in the project, implementation and tests realized.

Capítulo 1

Introdução

O hardware e o software dos sistemas computacionais têm sofrido constante evolução. O hardware atravessou quatro gerações que foram caracterizadas pela tecnologia dos componentes utilizados na sua implementação. Inicialmente eletromecânicos, surgiram posteriormente as válvulas, transistores e finalmente os circuitos integrados. A evolução do hardware tem se caracterizado pelo aumento da velocidade de processamento e pela redução do espaço físico e consumo de energia. O software operacional evoluiu dos carregadores de código em linguagem de máquina, sistemas operacionais monousuários, sistemas operacionais multiusuários em tempo compartilhado até os sistemas operacionais distribuídos. A feliz combinação do hardware, mais acessível e de maior desempenho, com os sistemas operacionais distribuídos e as redes de computadores, introduz aos usuários recursos computacionais nunca antes proporcionados pelos sistemas até então existentes.

Sistemas Distribuídos têm sido projetados para possibilitar um compartilhamento de recursos, fazendo com que computadores separados (muitas vezes máquinas diferentes umas das outras), interligados por uma rede local utilizem periféricos, servidores de arquivos, de impressão e outros.

As redes de computadores estão presentes em vários ambientes de trabalho, conectando os elementos e possibilitando a troca de informações.

Os objetivos básicos de uma rede de computadores são:

- Compartilhamento de recursos.
- Confiabilidade provida pela multiplicidade de recursos.
- Economia proporcionada pela utilização de muitas estações de trabalho, ao invés de um computador centralizado, e pelo compartilhamento de recursos através de servidores especializados.
- Modularidade, ou seja, o aumento gradual na capacidade do sistema. (Vale observar que o acréscimo de uma estação de trabalho aumenta a capacidade computacional de uma rede de computadores, o que não ocorre com o acréscimo de um terminal a um sistema multiusuário por tempo compartilhado).
- Meio de comunicação entre usuários geograficamente separados.

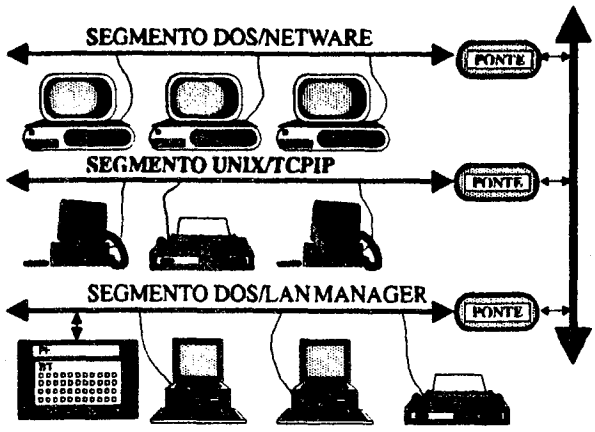
Para reger a comunicação nesses sistemas interligados por uma rede, existe um conjunto de regras formais denominado protocolo de comunicação. O primeiro passo dado no sentido da padronização desses protocolos utilizados, é representado pelo modelo de referência OSI ("Open Systems Interconnection"), patrocinado pela ISO ("International Standards Organization") [ZI80].

O modelo cliente-servidor representa uma alternativa natural para a implementação da comunicação em sistemas computacionais distribuídos, particularmente aqueles baseados em redes locais de computadores [TA85]. Nesse modelo, serviços são requisitados por processos cliente para processos servidores. Uma vez requisitado um serviço, o processo cliente aguarda uma resposta. Vários mecanismos podem ser adotados para a implementação desse modelo.

O mecanismo de Chamadas a Procedimentos Remotos, RPC, tem se destacado como um mecanismo simples e transparente para a implementação de sistemas distribuídos. O paradigma de chamadas a procedimentos locais, implementado na maioria das linguagens de programação, dá suporte teórico ao mecanismo RPC. A transparência ideal acontece quando os procedimentos remotos não podem ser distinguidos dos procedimentos locais. Alguns aspectos, como o acesso a parâmetros passados por referência, limitam o nível de transparência conseguido [WI87].

O Departamento de Ciências de Computação e Estatística do Instituto de Ciências Matemáticas de São Carlos, USP, conta hoje com um sistema computacional distribuído baseado em rede local (tecnologia Ethernet), composto de diversos servidores padrão (arquivos, impressão, correio eletrônico, etc), diversas estações de trabalho e um servidor especialmente desenvolvido para aplicações de computação concorrente/paralela, baseado na utilização de um banco de transputers (figura 1.1).

A utilização de computação concorrente (ou paralela) constitui uma área de atuação interdisciplinar, de grande interesse na atualidade. A utilização do microprocessador transputer da INMOS [IN90] é uma alternativa moderna e eficiente para a implementação de ambientes voltados à elaboração de sistemas concorrentes. A abordagem adotada no Laboratório de Sistemas Digitais (LaSD) do Departamento de Ciências de Computação e Estatística, ICMSC-USP, segue a filosofia de uso compartilhado de uma máquina paralela, baseada na utilização de um banco de transputers, que permite a alocação remota (via rede) e dinâmica de um certo número de transputers, para um usuário do sistema [TR91].



PE: Processador de Entrada do Banco. Responsável por estabelecer a comunicação entre estações e o banco, bem como gerenciar as atividades de controle dos usuários.
BT: Banco de Processamento Paralelo

Fig. 1.1 - Rede Local com Segmentos Interligados por Pontes

O desenvolvimento e implementação do sistema computacional distribuído, bem como o servidor de processamento paralelo (SPP), foram efetuados dentro de um projeto apoiado pela FAPESP, iniciado em agosto de 1990 e concluído em Julho de 1992. Esse projeto forneceu os subsídios iniciais para se criar uma plataforma básica para os trabalhos futuros. O produto final obtido foi um pequeno, mas expressivo, sistema computacional distribuído servindo os pesquisadores do LaSD e um protótipo do SPP, que no momento já está em funcionamento experimental.

Neste trabalho a implementação do sistema SPP (Servidor de Processamento Paralelo) existente no ICMSC-USP é detalhadamente estudada para levantamento de dados que permitam uma avaliação objetiva do desempenho do sistema.

Várias questões pendentes do projeto original são resolvidas, além de se propor uma nova implementação para a semântica de comunicação do núcleo de Chamadas Remotas.

É proposta uma implementação do sistema SPP utilizando-se o conjunto de protocolos TCP/IP. São mostrados os resultados obtidos nessa experiência.

Este trabalho está dividido em cinco capítulos que descrevem o desenvolvimento do projeto. É incluído um glossário com as principais abreviaturas e conceitos utilizados no trabalho.

No capítulo 2 são apresentadas as alterações efetuadas no sistema SPP - com

ênfase na implementação da semântica "at-least-once".

No capítulo 3 apresenta-se a utilização do protocolo TCP/IP no sistema SPP. São apresentadas as dificuldades encontradas e os resultados obtidos. O software utilizado foi o Ka9Q, que é descrito em detalhes.

A análise dos resultados obtidos com as alterações efetuadas, bem como na versão original é apresentada no capítulo 4.

No capítulo 5 são apresentadas as conclusões finais do trabalho, as contribuições e sugestões de trabalhos futuros.

Capítulo 2

Revisão/Ampliação da Implementação Original do SPP

O servidor de processamento paralelo desenvolvido no LaSD, conforme citado no capítulo 1, permite que diversos usuários compartilhem as unidades de processamento disponíveis no banco de transputers (BT). O banco permite, na versão implementada, que no máximo 4 usuários compartilhem os transputers disponíveis. A utilização dos processadores do banco é feita através do estabelecimento de uma sessão de trabalho, que na prática funciona como uma conexão lógica entre a estação de trabalho do usuário e o SPP. O usuário solicita um lote de transputers e efetua, dinamicamente, a configuração da arquitetura que deseja utilizar.

Os serviços de controle de acesso concorrente ao banco de transputers, bem como a comunicação com os usuários, são implementados no "processador de entrada" (PE) que implementa um "front-end" interligando a rede local com o BT.

Durante a utilização experimental do protótipo do SPP, constatou-se que, apesar de operando satisfatoriamente para uma primeira versão, o sistema ainda requer certos ajustes visando melhor desempenho no sistema de comunicação (basicamente o sistema de protocolos implementados) [TR91].

O protocolo de comunicação utilizado no SPP foi implementado segundo o paradigma cliente-servidor, utilizando o mecanismo de chamada a procedimentos remotos (Remote Procedure Call - RPC), conforme [WI87].

Existe um conjunto básico de primitivas que o cliente pode executar no BT. Para que a execução seja efetuada o cliente transmite um código de primitiva, juntamente com seus parâmetros, e aguarda resposta durante um intervalo de tempo. Caso a resposta não retorne no tempo estimado, o cliente retransmite a operação desejada até obter uma resposta ou esgotar um número máximo de tentativas. O servidor, caso receba um mesmo pedido mais de uma vez, identifica que se trata de uma mesma requisição e devolve a resposta processada anteriormente (semântica "at-most-once").

Na primeira versão não foi efetuado um estudo pormenorizado do tempo gasto para a execução de cada primitiva no servidor, ficando em aberto os níveis obtidos e o impacto gerado pelas decisões tomadas.

Este capítulo apresenta uma revisão geral do SPP incluindo estudo pormenorizado do protocolo implementado e sugestões de modificações visando melhorias no sistema.

2.1 Organização do SPP

Os itens a seguir apresentam o software e o hardware com a implementação original do sistema SPP.

2.1.1 Hardware

Em [PA93a] foi apresentada uma visão geral e detalhada do SPP. Esse item dará uma breve descrição física do sistema, visando a compreensão dos detalhes descritos nos demais itens.

O conjunto SPP é fórmado por vários elementos diferentes que se comunicam e cooperam para realizar a função a que se propõe: "fornecer recursos computacionais para programação paralela através de uma rede local de computadores".

Os elementos de hardware que fazem parte do SPP estão listados abaixo seguidos de uma breve descrição de sua função no sistema, e podem ser identificados na Figura 2.1.

- PE - Processador de Entrada
  Está ligado às Estações de Trabalho através da rede local e ao Controlador de Módulo. Na versão atual, sua principal função é fazer a interface entre as ET's e o Banco de Processadores.

- CM - Controlador de Módulo
  Faz o controle dos processadores do Banco (Módulo) além de gerenciar a comunicação. É formado por uma placa de Transputer - T800 e por uma interface que controla o barramento e os canais de comunicação com o banco. Também é responsável por configurar a rede de chaveamento (RC).

- RC - Rede de Chaveamento
  É configurada pelo CM através de um canal de configuração. Sua função é fazer a conexão entre "links" dos Transputers do banco com o objetivo de formar as redes de processadores. Da forma como está implementada permite a criação de qualquer topologia usando os processadores disponíveis.

- ET - Estação de Trabalho
  No sistema SPP, qualquer computador compatível com PC/AT/XT, ligado à rede local do sistema, pode ser transformado em uma ET, bastando para isso, a instalação

dos softwares necessários.

- P1-Pn - Elementos processadores a serem compartilhados

Cada "processador" do banco é um computador completo com memória RAM variando entre 1 e 4 MBytes. Possuem uma interface com o barramento do banco, através da qual são controlados pelo CM. A unidade central de processamento é o Transputer INMOS-T800.

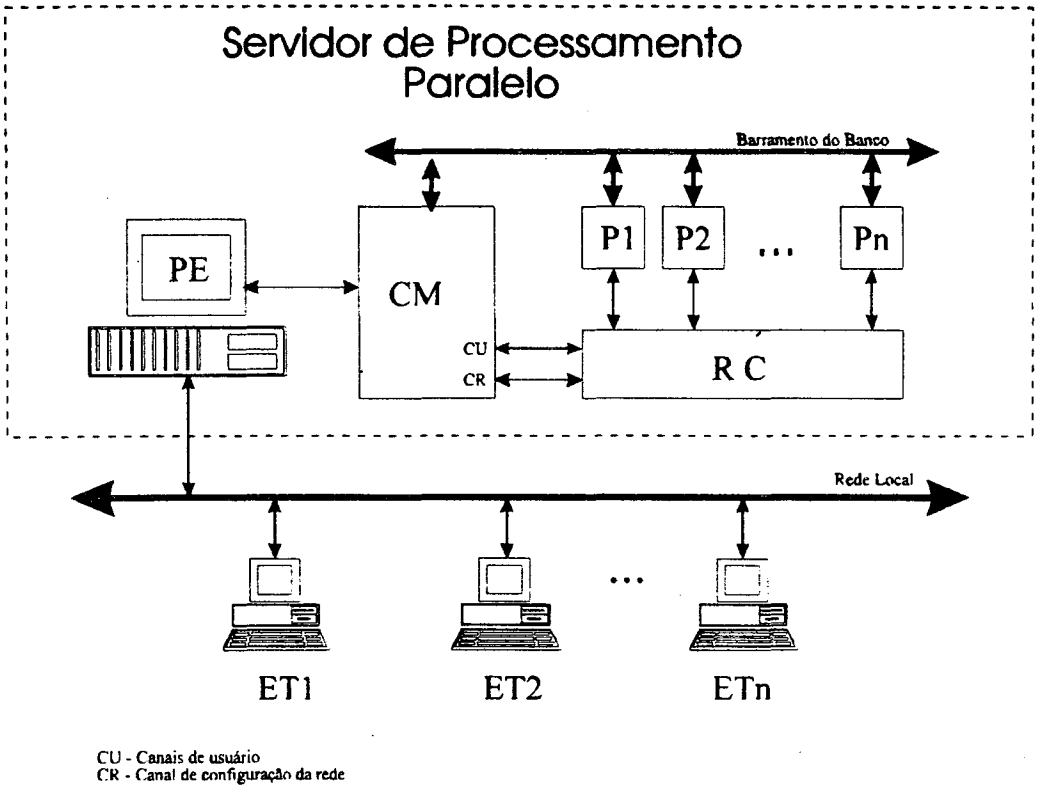


Fig. 2.1 - Estrutura do Hardware do SPP

2.1.2 Software

O software que compõem o SPP é dividido entre estação de trabalho e servidor. Nos próximos itens detalha-se cada um deles.

♦ Estação de Trabalho

O software da Estação é composto por um conjunto de programas que devem ser devidamente instalados e executados para que o sistema possa funcionar de forma adequada.

O TDS ("Transputer Development System") [IN90] foi o primeiro aplicativo disponível no Banco de Transputers. Em [PA93b] descreve-se com mais detalhes a instalação do TDS ao SPP. Para se ter acesso ao TDS, de uma estação de trabalho, tendo o software devidamente instalado deve-se substituir o arquivo **iserver.exe** pelo **iserver** modificado **iserverb.exe**.

Utilitários de acesso às primitivas do servidor:

- **login.exe**
- **logout.exe**
- **lot.exe**
- **dlot.exe**

Arquivos usados pelo sistema, associados ao mecanismo de comunicação e que devem estar instalados e disponíveis na ET:

- **iniseq.exe**
- **termpd.exe**

O utilitário iniseq.exe é utilizado somente na versão original, com a semântica "at-most-once".

O mecanismo RPC implementado através do protocolo SPP faz acesso à rede com auxílio do "**packet driver**", portanto, o software do "packet" deve ser instalado na estação para que o sistema funcione. Isso é feito executando-se os programas **ne1000.exe** ou **ne2000.exe** com os respectivos parâmetros de configuração, dependendo da placa de rede instalada na ET. Caso a estação seja dedicada ao SPP, esses programas podem ser executados a partir do **autoexec.bat** da máquina.

Com todos os programas citados acima instalados, a estação está pronta para compartilhar os recursos do SPP.

♦ Servidor

Basicamente o software do servidor do SPP é constituído por 2 programas, executados no PE e no CM respectivamente.

- **serv.exe**
- **bt.t8**

O programa **serv.exe** é executado no PE, sendo responsável pela comunicação com às ET's através do mecanismo RPC implementado e com o CM. Foi desenvolvido em linguagem C.

O programa **bt.t8** gerado a partir da compilação do programa OCCAM **bt.tsr**, descrito no documento [PA93b], é executado pelo Transputer do Controlador de Módulo.

Para a inicialização do servidor existe uma seqüência que deve ser seguida:

1. Deve-se carregar e executar o programa **bt.t8** na placa do Controlador de Módulo. Isso é feito por um arquivo em lote, *bt.bat*, contendo os seguintes comandos:

```
set transputer=#150
set iboardsize=#400000
set iterm=c:\tds3\system\tds3.itm
set tdssearch=c:\tds3\system\
echo off

:again
c:\tds3\system\iserver /se /sb bt.t8

if errorlevel 4 if not errorlevel 5 echo ISERVER did not perform action
requested
if errorlevel 3 if not errorlevel 4 echo Transputer error flag observed
if errorlevel 2 if not errorlevel 3 echo USER BREAK
if errorlevel 1 if not errorlevel 2 echo EXIT - program failure
if errorlevel 0 if not errorlevel 1 echo EXIT - program success
```

Depois de carregado o programa **bt.t8** para o CM, deve-se abortar o programa **iserver** que está executando no PE, o que pode ser feito através do comando <CRTL>+<C>, digitando-se <x> quando solicitado.

2. Executar o programa **serv.exe**. Após esses passos, um usuário em uma ET, já pode ter acesso ao sistema.

2.2 Decisões de Projeto

Como é de interesse que o SPP se torne um sistema disponível para diferentes ambientes, a primeira decisão tomada foi a migração do software do SPP, desenvolvido originalmente no Microsoft C - 6.0 - ambiente PWB, para o ambiente Turbo C. O objetivo dessa migração foi principalmente o fato de se ter interesse em continuamente atualizar e completar o sistema SPP. E como descreve-se no capítulo 6, o software do SPP foi incorporado a uma implementação do conjunto de protocolos TCP/IP e os códigos-fonte disponíveis desta implementação estão escritos em Turbo-C, justificando assim essa migração.

A migração do sistema para o ambiente Turbo-C, à principio, não foi efetuada de maneira tão simples. O software do SPP é longo e complexo tendo apresentado vários problemas com as representações dos dados. Foi necessário um estudo do adequamento da linguagem C às duas versões de compiladores, sendo que algumas diretivas de compilação foram alteradas e outras inclusive retiradas.

Uma outra decisão tomada com o intuito de se melhorar o entendimento do software desenvolvido foi a de se reescrever em C as rotinas de comunicação com o "packet driver", escritas inicialmente em "assembly". Isso facilita bastante possíveis mudanças futuras que se façam necessárias.

A única rotina para a qual foi mantido o código original, em "assembly", é a rotina de recepção. Essa decisão ocorreu porque é a rotina que faz a interface direta com o "packet driver", devendo inevitalvelmente ser escrita em "assembly".

Alguns pontos pendentes na versão preliminar do sistema SPP foram concluídos nesta etapa do trabalho. Um exemplo é o tratamento de "login" ao sistema. Existia no servidor uma tabela fixa com o nome dos usuários que tinham permissão para utilização do sistema, assim a tabela de usuários era inicializada em tempo de execução, da seguinte maneira:

```
/* cadastro de usuarios provisorio */
strcpy ((char *)Tu[1].nome_usu, "ana");
strcpy ((char *)Tu[2].nome_usu, "onofre");
strcpy ((char *)Tu[3].nome_usu, "marcos");
strcpy ((char *)Tu[4].nome_usu, "regina");
strcpy ((char *)Tu[5].nome_usu, "nivaldi");
strcpy ((char *)Tu[6].nome_usu, "orlando");
```

A inclusão de um novo usuário no sistema, implicava na modificação do código fonte e na necessidade de se compilar o software novamente.

Foi desenvolvido um programa auxiliar que trata do cadastramento de clientes ao sistema. Esse cadastramento é feito em arquivo e quando o servidor é colocado no ar, a tabela do usuário, localizada no servidor, é carregada diretamente do arquivo. Esse sistema permite que se inclua usuários no sistema, verifica se um dado usuário faz parte ou não do sistema e carrega o arquivo de usuários para a tabela de usuários, se necessário. Assim para se incluir um novo usuário no sistema, basta executar o sistema *cadastra*, e efetuar o cadastramento. Dessa forma o código no servidor ficou da seguinte maneira:

```
/* carrega o arquivo de usuarios cadastrados */

nusu = LoadUsers();
```

sendo que na variável nusu fica armazenado o número de usuários que estão na tabela.

Para o desenvolvimento do núcleo RPC adotou-se a semântica "at-most-once", por oferecer uma solução de compromisso entre desempenho e confiabilidade, além de permitir implementação simplificada e que se alcance os objetivos desejados. Como a tecnologia de redes locais tem se mostrado cada vez com mais confiabilidade, com uma taxa de erros mínima, uma outra tentativa de se melhorar o desempenho foi a implementação da semântica "at-least-once", descrita no próximo item.

2.3 Projeto Lógico

Como visto no item 2.2, uma das modificações efetuadas no sistema original foi de se fazer uma nova implementação da semântica de comunicação do núcleo de RPCs. Na tentativa de se melhorar o desempenho da troca de mensagens, tentou-se alterar a semântica "at-most-once", originalmente implementada, para a semântica "at-least-once".

Essa modificação acarreta algumas alterações no projeto lógico do software do SPP. Na implementação da semântica "at-most-once", existe um mecanismo rígido para controle do número de seqüência das primitivas (para garantir que a primitiva seja executada apenas uma vez) que é efetuado conforme descrito a seguir.

Quando uma estação inicia suas operações, assume-se que não existem requisições pendentes no servidor. É acrescentado um campo de informação na mensagem transmitida, denominado "número de seqüência". O controle de seqüência, exemplificado na figura 2.2, é uma técnica de implementação da semântica "at most once" e nesse caso é deixado a cargo do usuário garantir a integridade dos dados, caso ocorra falha do servidor durante a execução de uma primitiva.

Para que seja efetuada a primeira requisição, o servidor assume que o cliente fará uma solicitação com número de controle de seqüência igual a zero. Ao receber a requisição o servidor compara o número da seqüência recebida com o valor esperado e constatando serem iguais, este executa a primitiva, armazena o resultado da execução, complementa o número de seqüência e transmite a resposta com o número de seqüencia original. O cliente ao receber o resultado da execução da primitiva, complementa o número de seqüência entrando novamente em sincronismo com o número esperado pelo servidor. Caso ocorra algum problema e o cliente efetue mais de uma vez uma mesma requisição, cabe ao servidor verificar o número de seqüência e retransmitir o valor do resultado previamente armazenado [AN93].

Fig 2.2 - Seqüência de Requisições

É previsto nas variáveis necessárias, um campo para se armazenar essa resposta da execução da primitiva. No servidor as variáveis são as seguintes:

```
typedef struct                    /* cod_usu = 0 => sessao nao existente */
{
  BYTE cod_usu;                   /*codigo do usuario*/
  BYTE end_est[6];                /*endereco da estacao*/
  int  ide_proc;                  /*identificador do processo*/
  BYTE seq[NPRIM];                /*sequencia de requisicao da primitiva i*/
  int  tam_resp[NPRIM];           /*tamanho da ultima resposta da prim i*/
  BYTE *resp[NPRIM];              /*ultima resposta da primitiva i*/
  BYTE prbt;                      /*prim de comunicacao em execucao no BT*/
} TSES; /* Tabela de sessao associada a cada usuário */
```

localizadas no arquivo de definições globais. No arquivo `inics.h`, são declaradas e iniciadas variáveis relacionadas com a resposta de cada primitiva: (exemplo para duas primitivas)

```
  resp0  [NSES][40],  /* onde NSES e o numero maximo de sessoes */
  resp1  [NSES][20],

  for (i=0;i<NSES;i++)
  {
     /* Iniciar a tabela de respostas */
    resp0[i][0]=  '\0';
    resp1[i][0]=  '\0';
    Ts[i].resp[LOG_INN]= resp0[i];
    Ts[i].resp[LOG_OUT]= resp1[i];
  }
```

O módulo `AtenderProcedimentoRemoto`, do arquivo `sts.c`, que implementa o mecanismo de controle de seqüência, antes de enviar a resposta para o cliente, armazena a resposta na tabela de sessão e complementa a seqüência:

```
for (i=0; i<tam_bufaux; i++)
  Ts[sessao].resp[nro_prim][i]= bufaux[i];
Ts[sessao].tam_resp[nro_prim]= tam_bufaux;
MontarPacoteSPP (sessao, bufe[PRI_PAC_SPP], tam_bufaux, bufaux,
                 bufe, (BYTE) (0 | seqr));
Ts[sessao].seq[nro_prim]= (BYTE) (~Ts[sessao].seq[nro_prim]);
```

O módulo `ProcessarSPP`, que verifica a possibilidade de execução da primitiva solicitada, além de verificar, entre outras coisas, se a primitiva existe, se o usuário tem permissão para utilizá-la, se a sessão associada ao usuário existe, verifica também se o pacote é uma requisição nova ou se era um pedido repetido (com o mesmo número de seqüência), para poder assim decidir no módulo superior (o `AtenderProcedimentoRemoto`) se chama uma nova primitiva ou se simplesmente envia a resposta anterior.

No software da estação, a tabela de usuário tem um campo relacionado com o número da seqüência, de acordo com a estrutura abaixo:

```
typedef struct
{
  BYTE end_serv[END_ORG_PKT - END_DST_PKT];     /* endereco do servidor */
  BYTE ide_sessao[2];           /* identificador da sessao */
  BYTE seq[NPRIM];              /* sequencia de requisicao da primitiva i */
  long to[NPRIM];               /* time-out para resposta da primitiva i */
}TSERV;  /* tabela de usuario associada ao servidor */
```

Ao ser executada uma das primitivas, a seqüência de requisição é complementada da seguinte maneira:

```
/*complementa a sequencia a cada nova chamada*/
Ts[servidor].seq[LOG_OUT] = (BYTE) (~Ts[servidor].seq[LOG_OUT]);
```

onde `LOG_OUT` é o número associado à primitiva.

Na implementação da semântica "at-least-once", não é necessário que exista o mecanismo para controle do número de vezes que as primitivas são executadas. Conseqüentemente as estruturas de dados relacionadas com o mecanismo do número de seqüência foram eliminadas, ficando da seguinte maneira:

```
typedef struct                  /* cod_usu = 0 => sessao nao existente */
{
  BYTE cod_usu;                 /*codigo do usuario*/
  BYTE end_est[6];              /*endereco da estacao*/
  int  ide_proc;                /*identificador do processo*/
  BYTE prbt;                    /*prim de comunicacao em execucao no BT*/
} TSES; /* Tabela de sessao associada a cada usuário */
```

As variáveis relacionadas com o armazenamento das respostas, foram eliminadas do arquivo `inics.h`.

O módulo `AtenderProcedimentoRemoto` executa a primitiva solicitada e envia a resposta sem armazená-la previamente. O módulo `ProcessarSPP` verifica a possibilidade de execução da primitiva, só que não faz mais o teste do número de seqüência.

O programa `iniseq.exe`, responsável por iniciar todas as seqüências com 0 passou a não fazer sentido nessa nova implementação do protocolo.

No software da estação a estrutura de dados relacionada à associação usuário-servidor passou a ser:

```
typedef struct
{
  BYTE end_serv[END_ORG_PKT - END_DST_PKT];     /* endereco do servidor */
  BYTE ide_sessao[2];           /* identificador da sessao */
  long to[NPRIM];               /* time-out para resposta da primitiva i */
}TSERV;  /* tabela de usuario associada ao servidor */
```

tendo sido eliminadas as complementações de seqüência no final de cada primitiva executada.

No capítulo 4 são apresentados os resultados obtidos com as alterações acima descritas. É apresentado também uma análise estatística desses resultados.

O mecanismo RPC [AN91] foi implementado segundo uma estrutura hierárquica organizada em camadas, análoga ao modelo ISO-OSI [TA89], apesar desse modelo não considerar o mecanismo de RPC.

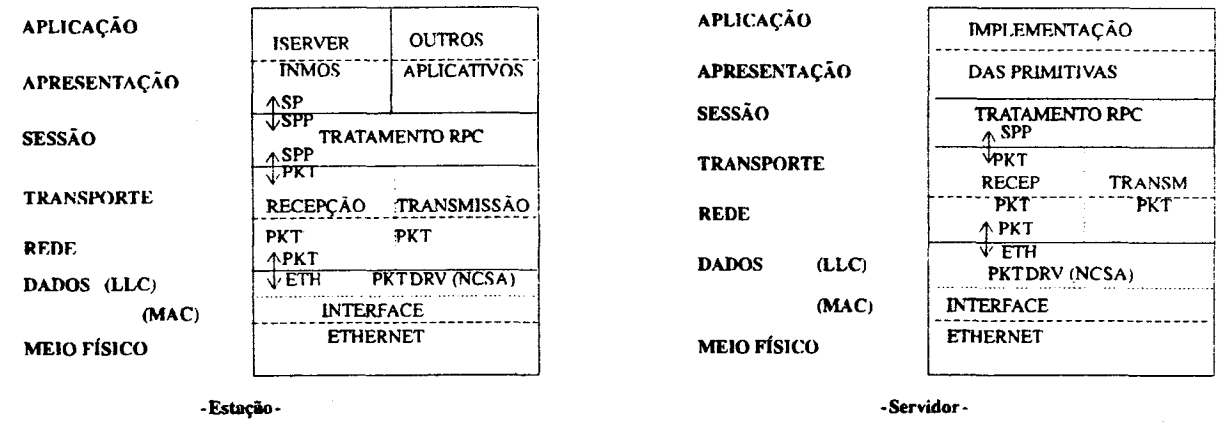


Fig. 2.3 - Estrutura em Camadas do Software
Correlação com o modelo ISO-OSI

A figura 2.3 mostra a correlação existente entre as camadas previstas no modelo ISO-OSI e as camadas implementadas neste sistema. A interface Ethernet está implementada até a subcamada MAC ("Medium Access Control") da camada de dados. A camada de rede e transporte não foram implementadas pois não houve necessidade das funções ali previstas. Em versões futuras do sistema, com a utilização de outras linguagens e aplicativos, pode surgir a necessidade da implementação da camada de transporte para possibilitar a transferência de pacotes de dados maiores que os permitidos pela camada de dados [TR91]. O mecanismo RPC foi implementado na camada de sessão, conforme sugere [TA89]. As primitivas e aplicativos foram implementados em camada mais elevada, englobando as camadas de apresentação e aplicação do modelo ISO-OSI.

Para efetuar a implementação da semântica "at-least-once" e das outras alterações descritas no item anterior foram modificadas praticamente todas as camadas do software.

Seguindo a figura 2.3 percebe-se que foram executadas as seguintes alterações:

- camada de aplicação e apresentação:
  - inclusão de mecanismo de cadastramento de usuários,
  - alteração das variáveis relacionadas com o armazenamento da resposta e com o controle de seqüência.

- camada de sessão:
  - modificação das rotinas que implementam o mecanismo de controle de seqüência, especialmente os módulos `AtenderProcedimentoRemoto` e `ProcessarSPP`.

- camada de dados:
  - alteração das rotinas escritas em assembly para Turbo-C.

2.4 Considerações Finais

O Servidor de Processamento Paralelo (SPP) é um sistema que continuará em constante evolução, requerendo sempre estudos pormenorizados para que as mudanças a serem efetuadas sejam bem fundamentadas.

Com a revisão do projeto do sistema SPP realizada neste capítulo teve-se o objetivo de se fechar alguns pontos que ficaram em aberto na implementação original do sistema.

Um outro ponto atingido com a revisão feita neste trabalho foi o aprimoramento da documentação do sistema SPP. Com as rotinas escritas em C, ao invés de "assembly", a documentação fica mais acessível para que qualquer pessoa possa entender mais facilmente o funcionamento do SPP facilitando futuras alterações.

Com o estudo e a revisão do SPP, realizados neste capítulo, obteve-se como produto a evolução do protocolo previamente desenvolvido. Como conseqüência uma discussão mais detalhada e documentada do funcionamento do protocolo SPP.

No próximo capítulo será discutida a utilização do protocolo do sistema SPP em um ambiente TCP/IP. Para isso é descrito o ambiente utilizado para esta implementação, bem como as alterações necessárias para que seja viável a utilização do protocolo TCP/IP para se ter acesso ao Banco de Transputers.

Capítulo 3

Utilização do Protocolo TCP/IP no Ambiente SPP

A decisão de se utilizar o protocolo TCP/IP para se ter acesso ao sistema SPP à partir desse protocolo foi motivada pela abrangência de uso do conjunto de protocolos TCP/IP.

A utilização do protocolo TCP/IP para esse fim (acesso ao sistema SPP) não é uma tarefa fácil de se realizar. A implementação desse conjunto de protocolos é complicada e aliado ao fato do software do SPP ser também de grande complexidade o desenvolvimento desta etapa do trabalho foi bem trabalhoso.

Neste capítulo são apresentadas algumas justificativas do porquê de se utilizar esse protocolo. É apresentado o sistema Ka9Q, que foi utilizado para a implementação do sistema e são apresentadas as dificuldades encontradas no decorrer do trabalho, detalhes das modificações e da implementação do sistema em estudo.

3.1 Disponibilidade/Flexibidade da Utilização do SPP em função de um protocolo mais difundido

O conjunto de protocolos TCP/IP se tornou, de fato, o padrão para a interconexão de sistemas abertos na indústria de computadores. A maioria dos sistemas de computação utilizam os protocolos TCP/IP para se comunicar, pois estes fornecem um alto grau de interoperabilidade, engloba grande parte dos sistemas vendidos e pode ser executado sobre várias tecnologias de rede [CO91]. Instituições de pesquisa e de educação têm utilizado os protocolos TCP/IP como sua plataforma primária de comunicação.

Além do uso convencional, os protocolos TCP/IP são utilizados também para se comunicar através da Internet, fazendo com que pesquisadores troquem informações e pesquisas mais rapidamente.

O pacote Internet Ka9Q é uma implementação da família de protocolos TCP/IP, sendo o resultado de vários anos de desenvolvimento. O grupo de usuários dos protocolos

TCP/IP cresceu e hoje em dia inclui membros de todo o mundo, sendo que muitos deles contribuiram com idéias e com trabalho para o desenvolvimento e implementação do pacote Ka9Q. O Ka9Q é um pacote de domínio público, disponível a partir da UUNET Technologies Inc, Falls Church, Virginia, via "ftp anonymous".

O programa *net* (versão executável do Ka9Q) implementa um sistema operacional multi-tarefas, que fornece suporte para programação concorrente, dando a ilusão, em máquinas com um único processador, de que vários programas possam ser executados simultaneamente.

O compilador utilizado para gerar a versão executável (o programa *net.exe)* é o Turbo-C v.2.0 e o TASM v.1.0. Para se ter um bom funcionamento e uma compilação bem sucedida é aconselhável que a máquina tenha 4MBytes de memória RAM.

Para a instalação do ambiente TCP/IP é necessária a configuração adequada dos arquivos (descritos em [KA91]). Assim para se ter um ambiente de desenvolvimento adequado, foram considerados três microcomputadores, dedicados a este projeto, e o ambiente de desenvolvimento obtido é mostrado na figura 3.1.

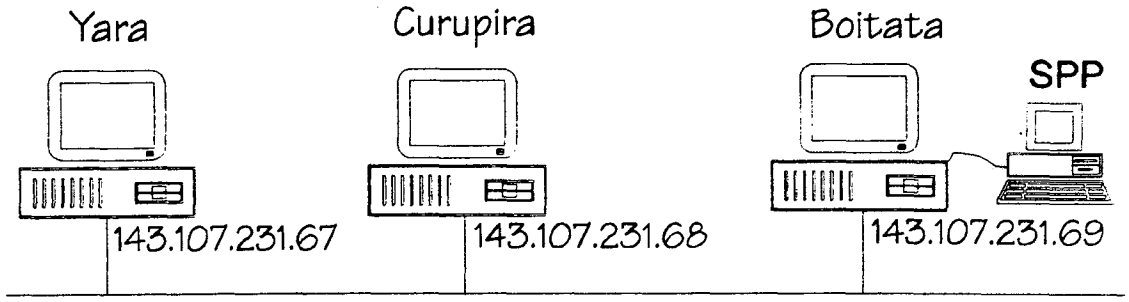


Fig. 3.1 - Ambiente de desenvolvimento

3.2 O Programa *net.exe*

Com a execução do programa net.exe (dentro do ambiente MS-DOS) é possível se ter acesso a recursos da Internet (TCP/IP, NET/ROM e AX25). O net.exe implementa um núcleo multi-tarefas, podendo se comportar como um cliente (por exemplo quando se executa o ftp em uma máquina remota), um servidor (quando se atende a uma solicitação de ftp gerada remotamente) ou ainda um roteador de pacotes (quando é recebido um pacote não TCP/IP, mas que o sistema conhece o destino).

A configuração do ambiente gerado pela execução do net.exe, pode ser definida tanto através dos arquivos de configuração, como através de intervenção via teclado.

Nesse caso o operador local tem controle sobre todo o ambiente, podendo alterar as interfaces acopladas ao sistema, os caminhos de roteamento, mudar o tamanho dos "buffers" de mensagens, etc.

Para utilização dos recursos do TCP/IP, NET/ROM e AX25, dentro do ambiente criado por net.exe, tem-se os serviços descritos na tabela 3.1.

• telnet:	O protocolo telnet, como implementado no software Ka9Q, permite que os usuários se comuniquem via uma conexão de teclado.
• mail:	O protocolo para transferência de mail (SMTP - Simple Mail Transfer Protocol) fornece serviços para enviar e receber mails.
• Tranferência de arquivos:	O protocolo para transferência de arquivos (FTP) permite que um usuário em um computador qualquer (cliente), copie ou envie arquivos para um outro computador (servidor). A segurança é controlada através do nome e da senha para se ter acesso ao outro computador.
• Serviços ax25:	São fornecidos também os serviços regulares AX.25.
• NET/ROM:	O sistema *net* permite também que o sistema sirva como um nó NET/ROM.

Tabela 3.1 - Serviços Disponíveis no Sistema *net*

Para acesso aos serviços listados na tabela 3.1, bem como outros serviços adicionais, também disponíveis através do net.exe, existem diversos comandos, descritos na documentação geral do pacote Ka9Q [KA91].

3.3 Projeto Lógico

A implementação do pacote Ka9Q é altamente modular e seguindo a estrutura mostrada na tabela 3.2, pode-se analisar as funções de cada módulo do sistema.

Clientes/Servidores	Esse módulo implementa os clientes e os servidores dos serviços descritos abaixo.
Internet	Nessa camada são implementados o protocolos TCP, UDP, IP,

ax25	Os serviços AX25 e net/rom são aqui implementados
net/rom	
Protocolo Ponto-a-Ponto	Funções relacionadas com a implentação do protocolo ponto-a-ponto.
net	Camada intermediária dos aplicativos, as funções aqui definidas são usadas para a implementação de novos aplicativos. Os mecanismos de comunição (*sockets*) são implementados nesta camada.
Interfaces Físicas	Implementa todo o suporte de hardware para as camadas superiores.

Tabela 3.2 - Organização Lógica da Implementação do Programa *net*

Para que os aplicativos tenham acesso ao protocolo de comunicação, o pacote Ka9Q implementa uma abstração chamada *"socket"*, que é uma generalização do mecanismo de entrada e saída do Unix. Os "sockets" têm opções para vários tipos de redes, entre eles TCP/IP [CO91].

Para representar a hierarquia de chamadas no programa *net*, a figura 3.2 mostra um diagrama de estrutura modular simplificado. Note que o único módulo detalhado é o que trata do serviço ftp, isso porque, conforme mostra o sub-item 3.3.2, as alterações são efetuadas a partir desse módulo.

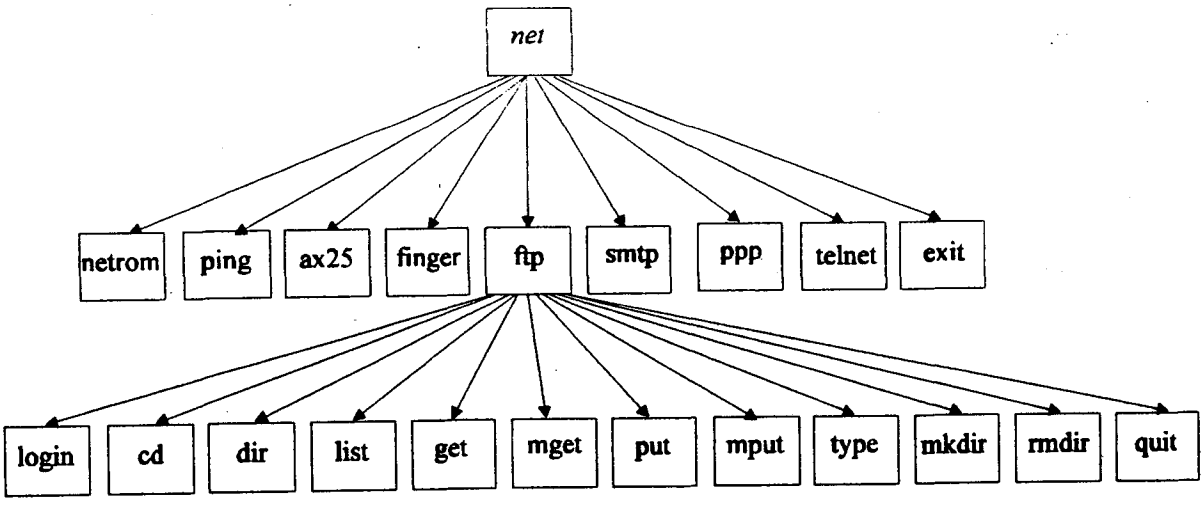


Fig. 3.2 - Diagrama de Estrutura Modular - Programa *net*

Na decisão de como se utilizar o programa *net* para se ter acesso ao sistema SPP foram encontrados alguns problemas. No próximo sub-item descrevem-se essas dificuldades e aponta-se qual a política adotada para a utilização do programa *net*.

3.3.1 Dificuldades Encontradas

A primeira abordagem dada para a utilização do protocolo TCP/IP foi a criação de uma biblioteca geral com todas as funções da implementação desse protocolo. A idéia inicial consistia em se utilizar apenas as rotinas do protocolo de comunicação. Essa abordagem foi abandonada, pois como o programa *net* implementa um sistema operacional multi-usuário, todas as funções das camadas superiores são necessárias para a criação do ambiente multi-tarefas.

Numa segunda abordagem resolveu-se implementar as rotinas de comunicação com o sistema SPP no mesmo nível dos serviços fornecidos pelo programa *net*. Essa abordagem foi abandonada pois chegou-se à conclusão de que todo o tratamento de sessão e de conexão deveria ser feito para cada rotina separadamente.

Resolveu-se então criar um serviço disponível à partir do programa *net*, que fornece as primitivas necessárias para a comunicação com o banco de transputers. Dessa maneira o tratamento de sessão e de conexão é gerenciado por esse serviço criado, chamado `spp` (descrito no próximo item, juntamente com as modificações necessárias no programa *net* para a implementação desse serviço).

3.3.2 Alterações Efetuadas no Programa *net*

Para que o protocolo SPP fique disponível no programa *net* decidiu-se deixar o sistema SPP como mais um serviço oferecido pelo programa. Como mostra a figura 3.3 criou-se o serviço `spp`, à partir do qual ficam disponíveis as rotinas de comunicação com o banco de transputers.

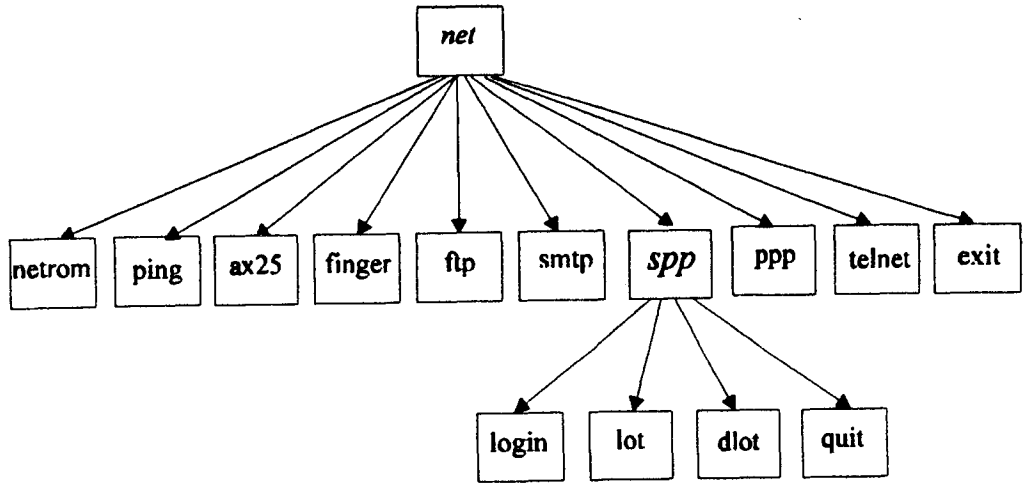


Fig. 3.3 - Diagrama de Estrutura Modular - Programa *net* modificado

Para ser ter acesso ao serviço spp, como a qualquer outro serviço fornecido pelo programa *net*, deve-se, na estação cliente, digitar o comando `spp <endereço do servidor>`, que no caso considerado é sempre o micro servidor, ligado ao Banco de Transputers. À partir desse comando será solicitado o nome do usuário e depois a sua senha. Os usuários (e suas respectivas senhas) devem estar cadastrados no arquivo "/FTPUSERS" da máquina servidora. As primitivas `lot` e `dlot` foram implementadas utilizando a abstração de "sockets" para se comunicarem com o servidor.

- **Servidor**

A figura 3.4 mostra um diagrama de estrutura modular do software do servidor.

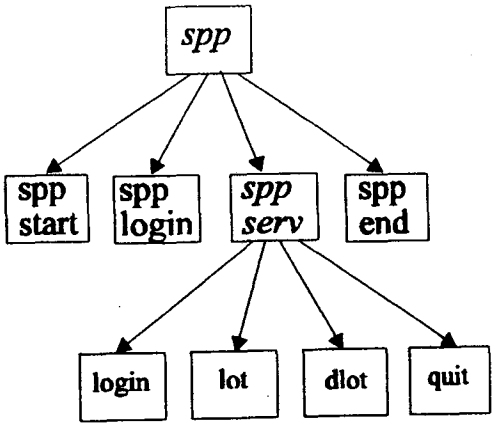


Fig. 3.4 - Diagrama de Estrutura Modular - Programa *net* (SPP - servidor)

No servidor é implementada uma rotina que inicia o servidor *spp*. Nela é criado o "socket" que fica aguardando a requisição de conexão de alguma estação. Assim que

alguma estação solicita uma requisição, é disparado um outro processo, `sppserv`, que trata das primitivas implementadas. Vale lembrar que o programa Ka9Q pode estar executando outros serviços e ainda assim responder às solicitações da estação cliente.

O programa bt.t8, descrito no capítulo 5, não foi alterado. Ele sempre aguarda um pacote no formato SPP, então juntamente com cada primitiva implementada foram associados os procedimentos `MontarPacoteSPP` e `DesmontarPacoteSPP`.

- **Estação**

Na estação é implementada uma rotina, `dospp`, que cria o "socket" para fazer a conexão com o servidor. Depois de feita a conexão as primitivas estão disponíveis para o usuário. A figura 3.5 ilustra o diagrama de estrutura modular do cliente.

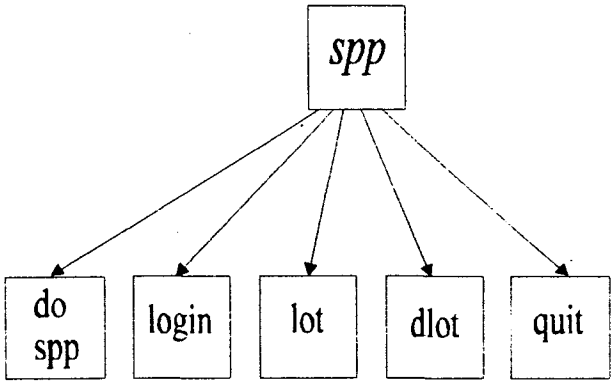


Fig. 3.5 - Diagrama de Estrutura Modular - Programa *net* (SPP - estação)

Tanto na estação quanto no servidor, o tratamento de sessão de trabalho, de manipulação dos pacotes e todos os detalhes relativos aos pacotes ethernet são tratados pelo programa *net*.

Todos os casos de teste: falha de comunicação com o Banco, não disponibilidade de lotes de transputers e outros, foram considerados na implementação.

3.3.3 Utilização dos Transputers

A idéia de se implementar o sistema SPP em ambiente TCP/IP é motivada pelo fato de que máquinas localizadas fora do escopo da rede do ICMSC-USP possam utilizar o Banco de Transputers. Nessa primeira implementação considera-se que o ambiente TDS não é viável para utilização a grandes distâncias e então o Banco de Transputers fica

disponível para aplicações que são desenvolvidas para executarem independentes do TDS, isto é, a partir do código objeto do programa gerado anteriormente.

Para a execução de um programa sobre uma rede de Transputers, independente do TDS, é necessário que esse programa seja carregado nas placas de Transputer. Uma das maneiras de se carregar o programa é com o auxílio de um computador hospedeiro conectado a rede.

O programa que permite que o computador hospedeiro carregue os programas no transputer e ofereça as facilidades de interface é o iserver. Nessa implementação o iserver considerado está localizado no servidor. Esse arquivo foi mais uma vez modificado para que as rotinas de comunicação com o Banco transmitam o pacote SPP conforme o Banco espera (item 3.3.2). As rotinas alteradas são as mesmas descritas em [AN93] com o objetivo agora de comunicação do servidor com o Banco de Transputers.

Um programa "*standalone*" que se comunica em tempo de execução com o iserver pode ser codificado de duas maneiras, descritas no documento [PA93b]. O código gerado em ambos os procedimentos é puramente binário, podendo ser migrado para qualquer hospedeiro que possua uma implementação do iserver.

Para se executar um programa "standalone" na versão do SPP para ambientes TCP/IP deve-se primeiro transferir o arquivo que será executado no Banco de transputers, via ftp, para a máquina servidora. Depois os passos são similares aos realizados para a execução do TDS.

- **Abrir uma sessão (login)**
- **Alocar os processadores (lot)**
- **Executar o programa**

Para se executar um programa *standalone* no SPP, foi desenvolvido um novo comando `iserv` que dispara o iserver (versão modificada) no servidor. Ao ser disparado o iserver é executado o seguinte arquivo em lote:

```
set transputer=#150
set iboardsize=#400000
set iterm=c:\tds3\system\tds3.itm
set tdssearch=c:\tds3\system\
echo off

:again
c:\tds3\system\iserver /se /sb %1.T8

if errorlevel 4 if not errorlevel 5 echo ISERVER did not perform action
requested
if errorlevel 3 if not errorlevel 4 echo Transputer error flag observed
if errorlevel 2 if not errorlevel 3 echo USER BREAK
if errorlevel 1 if not errorlevel 2 echo EXIT - program failure
if errorlevel 0 if not errorlevel 1 echo EXIT - program success
```

Depois de executado o programa, o usuário transfere, via ftp, o arquivo com a saída do programa.

A geração de um programa "standalone" difere da geração e execução de programas a partir do TDS. Algumas bibliotecas especiais, que dão suporte ao uso do protocolo SP devem ser usadas: **sphdr, splib, solib, sklib e spinterf** [IN90].

Maiores detalhes sobre os tipos de programa *standalone* podem ser obtidos em [IN90].

3.4 Considerações Finais

Como o conjunto de protocolos TCP/IP tem uma utilização altamente difundida a nível mundial, decidiu-se utilizá-lo para uma nova implementação do protocolo SPP, ficando como mais uma opção disponível (o sistema original não foi suprimido).

A maior parte das instituições de pesquisa utilizam o protocolo TCP/IP como plataforma de comunição, assim a disponibilidade do sistema SPP utilizando esse conjunto de protocolos faz com que o sistema fique mais acessível. Além disso o sistema fica mais flexível no sentido de ser incorporado a outros sistemas se for considerado o fato de que a maioria dos sistemas de computação utilizam os protocolos TCP/IP para se comunicar.

No próximo capítulo é feita uma análise dos resultados obtidos nas alterações do sistema original com a implementação da nova semântica de comunicação e dos resultados obtidos com a utilização dos protocolos TCP/IP.

Capítulo 4

Análise dos Resultados

Para se observar quantitativamente quais foram os efeitos das modificações descritas nos capítulos anteriores sobre a implementação original do SPP, neste capítulo são apresentados os resultados obtidos, bem como efetuada uma análise desses resultados.

O objetivo dos testes realizados é comparar o tempo de execução das primitivas remotas nas duas semânticas que foram implementadas para a comunicação com o Banco de Tranputers - "at-least-once" e "at-most-once". Além de se fazer uma comparação com a versão implementada utilizando-se o TCP/IP.

4.1 Técnica Utilizada para o Teste

A análise dos resultados obtidos nos testes realizados foi feita pelo software estatístico SAS (Statistical Analysis System) versão 6.04, realizado no LAE (Laboratório de Estatística Aplicada) do departamento de Estatística da Universidade Federal de São Carlos.

A técnica utilizada para a comparação do tempo de execução das primitivas nas duas semânticas, "at-least-once" e "at-most-once", é o Teste t.

Realiza-se o seguinte teste de hipótese:

$$H_0 : \mu_{ATL} = \mu_{ATM}$$
$$H_1 : \mu_{ATL} \neq \mu_{ATM}$$

onde: μ_{ATL} : tempo médio para a execução da variável em teste na semântica "at-least-once", e

μ_{ATM} : tempo médio para a execução da variável em teste na semântica "at-most-once".

A estatística de teste se as variâncias são iguais é:

$$t = \frac{(\bar{x}_{ATL} - \bar{x}_{ATM})}{\sqrt{s^2(\frac{1}{n_{ATL}} + \frac{1}{n_{ATM}})}}$$

onde: $\bar{x}_{ATL}$: média amostral para a semântica "at-least-once"

$\bar{x}_{ATM}$: média amostral para a semântica "at-most-once"

n_{ATL} : tamanho da amostra para a semântica "at-least-once"

n_{ATM} : tamanho da amostra para a semântica "at-most-once

$$s^2 = \frac{[(n_{ATL}-1)s^2_{ATL} + (n_{ATM}-1)s^2_{ATM}]}{n_{ATM}+n_{ATL}-2}$$

s^2_{ATL} e s^2_{ATM} : variância amostral para cada uma das semânticas.

Se as variâncias são diferentes a estatística de teste é:

$$t' = \frac{\bar{\bar{x}}_{ATL} - \bar{\bar{x}}_{ATM}}{\sqrt{\frac{\omega_{ATL}}{n_{ATL}} + \frac{\omega_{ATM}}{n_{ATM}}}}$$

onde: $\bar{x}_{ATL}$: média amostral para a semântica "at-least-once"

$\bar{x}_{ATM}$: média amostral para a semântica "at-most-once"

n_{ATL} : tamanho da amostra para a semântica "at-least-once"

n_{ATM} : tamanho da amostra para a semântica "at-most-once"

$$\omega_{ATL} = \frac{\sum (x_i - \bar{\bar{x}}_{ATL})^2}{n_{ATL}-1}$$

$$\omega_{ATM} = \frac{\sum (x_i - \bar{\bar{x}}_{ATM})^2}{n_{ATM}-1}$$

Verifica-se a igualdade das variâncias através da estatística:

$$F' = \frac{\max(s_i)}{\min(s_i)}$$

correspondente ao teste de hipótese:

$$H_0: \sigma_1 = \sigma_2$$
$$H_1: \sigma_1 \neq \sigma_2$$

F' tem distribuição F-Snedecor com (n_{ATL},n_{ATM}) graus de liberdade.

O valor *t* é comparado com o valor tabelado e especificado por *n* graus de liberdade (n = tamanho da amostra (ATL+ATM)) e pelo nível de significância α.O valor escolhido para α foi 0.01, isto é, permite-se errar em 1% das vezes.

Essa técnica estatística foi aplicada em dois casos de teste que serão apresentados no próximo item.

4.2 Testes Realizados - SPP

4.2.1 Primeiro Teste Realizado

O objetivo deste primeiro teste é analisar e comparar o tempo de transmissão relacionado a cada primitiva separadamente. Em todos os testes o tempo considerado é o tempo desde a chamada do procedimento remoto até a volta da requisição para a estação que fez a chamada.

Coleta dos Dados: Para coletar os dados para este teste, foi desenvolvido um programa na linguagem C, que dispara cada uma das primitivas 1000 vezes. O procedimento foi o mesmo para as duas semânticas em questão.

As primitivas consideradas no teste são `login`, `lot`, `dlot` e `logout`.

Na tabela 4.1 encontram-se os valores obtidos neste primeiro teste realizado, considerando a primitiva `login`.

Primitiva Login

Semântica	N	Média	Desvio Padrão
ATL	1000	0.02423075	0.00086312
ATM	1000	0.02538459	0.00086668

F'=1.01 gl(graus de liberdade)= (999,999) Prob>F'=0.8966

Variâncias	t	gl	Prob.>t
≠	-0.9433	999	0.3457
=	-09433	1998	0.3456

Tabela 4.1 - Dados Obtidos para o Primeiro Teste - Login

Para a primitiva `login` as variâncias são iguais e como a (prob>t)>0.01 (nível de significância escolhido) aceita-se a hipótese H_0, isto é, a média de execução para a

primitiva `login` nas duas semânticas é igual e não existe uma diferença estatisticamente significante entre as duas semânticas.

Primitiva Logout

Semântica	N	Média	Desvio Padrão
ATL	1000	0.00093407	0.00710635
ATM	1000	0.00252747	0.01151597

F'=2.63 gl(graus de liberdade)= (999,999) Prob>F'=0.0000

Variâncias	t	gl	Prob.>t
≠	-3.7236	999	0.0002
=	-3.7236	1998	0.0002

Tabela 4.2 - Dados Obtidos para o Primeiro Teste - Logout

Considerando a primitiva `logout`, conforme mostra a tabela 4.2, as variâncias são diferentes (prob>F')<0.01 e como a (prob>t)<0.01 (nível de significância escolhido) rejeita-se a hipótese H_0, isto é, a média de execução para a primitiva `logout` nas duas semânticas é diferente e existe uma diferença estatisticamente significante entre as duas semânticas.

As tabelas 4.3 e 4.4 representam os dados para as primitivas `lot` e `dlot`. Para essas primitivas as variâncias são diferentes (prob>F')<0.01 e como a (prob>t)<0.01 rejeita-se a hipótese H_0, isto é, a média de execução para as duas primitivas nas duas semânticas é diferente e existe um diferença estatisticamente significante entre as duas semânticas.

Primitiva Lot

Semântica	N	Média	Desvio Padrão
ATL	1000	0.00203297	0.01037670
ATM	1000	0.09170320	0.03350342

F'=10.42 gl(graus de liberdade)= (999,999) Prob>F'=0.0000

Variâncias	t	gl	Prob.>t
≠	-80.8478	999	0.0000
=	-808478	1998	0.0000

Tabela 4.3 - Dados Obtidos para o Primeiro Teste - Lot

Primitiva Dlot

Semântica	N	Média	Desvio Padrão
ATL	1000	0.00170330	0.00952770
ATM	1000	0.09186804	0.03564389

F'=14.00 gl(graus de liberdade)= (999,999) Prob>F'=0.0000

Variâncias	t	gl	Prob.>t
≠	-77.2797	999	0.0000
=	-77.2797	1998	0.0000

Tabela 4.4 - Dados Obtidos para o Primeiro Teste - Dlot

Com os dados do teste descrito acima obtém-se o gráfico 4.1 que demonstra que no geral a semântica "at-least-once" é mais eficiente do que a semântica "at-most-once". A primitiva `login` gasta um tempo praticamente igual nas duas semânticas.

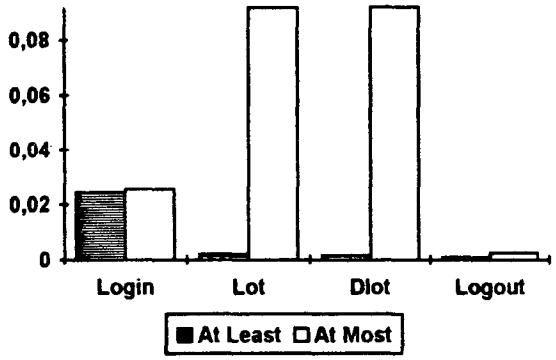


Gráfico 4.1 - Análise das Primitivas para o Primeiro Teste.

4.2.2 Segundo Teste Realizado

O objetivo deste teste é analisar e comparar o tempo gasto para se executar uma sessão mínima no banco de transputers. Isso significa o tempo de se entrar no sistema, alocar um lot, desalocá-lo e depois finalizar a sessão. O tempo considerado é o tempo desde a chamada do procedimento remoto até a volta da requisição para a estação que fez a chamada.

Coleta dos Dados: Para coletar os dados para este teste, foi desenvolvido um programa na linguagem C, que dispara as primitivas `login`, `lot`, `dlot` e `logout` 1000 vezes, respectivamente. O procedimento foi o mesmo para as duas semânticas em comparação.

A variável considerada neste teste é a soma do tempo de cada primitiva: `login`, `lot`, `dlot` e `logout`.

Sessão Mínima: Tempo de login+lot+dlot+logout			
Semântica	**N**	**Média**	**Desvio Padrão**
ATL	1000	0.02890107	0.03067268
ATM	1000	0.21148331	0.05972045

F'=3.79 gl(graus de liberdade)= (999,999) Prob>F'=0.0000

Variâncias	t	gl	Prob.>t
≠	-85.9999	999	0.0000
=	-85.9999	1998	0.0000

Tabela 4.5 - Dados Obtidos para o Segundo Teste - Sessão Mínima

Na tabela 4.5 encontram-se os valores obtidos neste teste realizado, considerando a sessão mínima.

Para a variável sessão mínima as variâncias são diferentes (prob>F')<0.01 e como a (prob>t)<0.01 rejeita-se a hipótese H_0, isto é, a média de execução de uma sessão mínima nas duas semânticas é diferente e existe um diferença estatisticamente significante entre as duas semânticas.

O gráfico 4.2 mostra que o tempo médio de se executar uma sessão mínima na semântica "at-least-once" chega ser 0,18 segundos mais rápido do que na semântica ät-most-once".

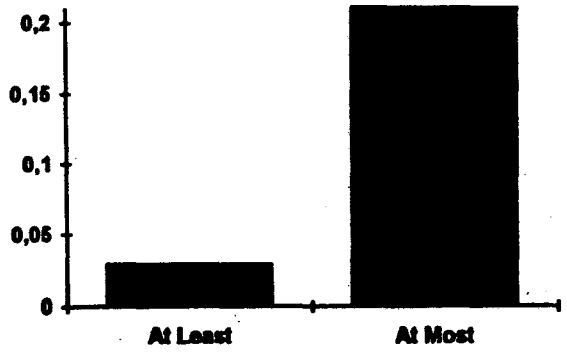


Gráfico 4.2 - Análise de uma Sessão Mínima para as duas Semânticas

4.3 Testes Realizados - Implementação TCP/IP

O acesso ao SPP através dos protocolos TCP/IP é altamente interativo, dificultando a realização de testes quantitativos similares aos realizados nas versões discutidas anteriormente. Além disso, a implementação das primitivas para acesso ao Banco de Transputers na versão TCP/IP utiliza uma comunicação diferente da versão original do SPP. Na versão TCP/IP somente é transmitido o número da primitiva e o argumento para a primitiva, caso necessário, assim o método de comparação utilizado para as primitivas, discutido nos itens 4.1 e 4.2 não é adequado.

Assim para se fazer uma análise comparativa dos protocolos desenvolvidos com a implementação TCP/IP, criou-se uma primitiva, para efeito de teste, que na estação transfere arquivos para o servidor e no servidor simplesmente devolve um pacote de reconhecimento da recepção. A mesma implementação da primitiva foi utilizada para as três versões em comparação. Medindo-se o tempo de transferência dos arquivos, pode-se ter uma análise quantitativa da diferença de desempenho entre as três versões implementadas.

Para a execução do teste foram consideradas três faixas de tamanho de arquivos, conforme a tabela 4.6:

Pequenos	0 < Tamanho < 100KB
Médios	100KB < Tamanho < 1MB
Grandes	1MB < Tamanho < máximo

Tabela 4.6 - Faixa de Tamanho de Arquivos para Teste

A tabela 4.7 apresenta os resultados obtidos no teste para cada tipo de arquivo:

Tamanho / Protocolo	64143 Bytes	526055 Bytes	1050564 Bytes
At-Most-Once	0.4945	4.8175	9.7802
At-Least-Once	0.3846	3.9714	7.6923
TCP/IP	3.2967	28.768	57.637

Tabela 4.7 - Resultados Comparativos com TCP/IP

Como pode-se observar pela tabela 4.7 e pelo gráfico 4.3, a versão "at-least-once" é a implementação mais eficiente.

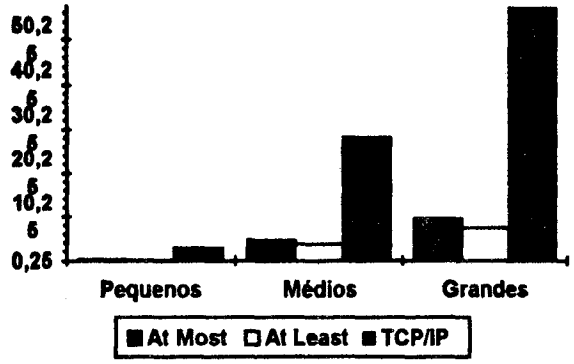


Gráfico 4.3 - Análise das Três Versões Implementadas

4.4 Considerações Finais

Os testes realizados na versão original do sistema SPP, com as duas semânticas implementadas, mostram que a semântica "at-least-once" é mais eficiente do que a semântica "at-most-once". Esse fato é justificado pela grande parte de código, responsável pelo controle de seqüência da semântica "at-most-once", que foi eliminada.

O usuário final deve decidir qual versão é mais adequada para o seu ambiente de desenvolvimento. Se ele pode garantir que a sua rede local possui alta confiabilidade de transmissão, e portanto baixa taxa de erros, a versão adequada a ser utilizada é a versão "at-least-once". Isso porque não se pode deixar de considerar o fato de que a versão "at-least-once" não garante uma única execução da primitiva e a repetição de uma mesma requisição pode causar o comprometimento do sistema, pois as primitivas do SPP não são idempotentes [WI87], isto é, se elas forem executadas mais de uma vez, o resultado está comprometido.

Os resultados obtidos com a implementação baseada no conjunto de protocolos TCP/IP sugerem que esta implementação deve ser utilizada em casos de acesso remoto ao SPP (isto é fora do ambiente LaSD) e que apesar da desvantagem observada em termos de desempenho, a utilização do Ka9Q deve ser, sempre que possível, incentivada para acesso remoto a ambientes computacionais.

No próximo capítulo são discutidas as dificuldades e as contribuições deste trabalho. Também são apresentadas sugestões para trabalhos futuros e as considerações finais.

Capítulo 5

Conclusão

Nos capítulos anteriores foram apresentados os diversos aspectos relacionados com a revisão e ampliação do sistema SPP [TR91]. Foram discutidas duas propostas: a realização de uma nova semântica sobre o protocolo originalmente desenvolvido para o sistema e uma nova abordagem, baseada na utilização do conjunto de protocolos TCP/IP. Análises estatísticas visando quantificar o desempenho dos protocolos implementados foram realizadas e os resultados obtidos foram discutidos.

Neste capítulo apresentam-se as conclusões obtidas após a análise das atividades e resultados de cada uma das etapas do desenvolvimento deste trabalho. As principais dificuldades encontradas, as contribuições apresentadas por esta dissertação e sugestões para trabalhos futuros são discutidas. Nos comentários finais são avaliados os resultados obtidos, de acordo com os objetivos e a motivação inicialmente estabelecidos.

5.1 Dificuldades Encontradas

No decorrer do desenvolvimento deste trabalho foram encontradas diversas dificuldades de ordem prática que merecem ser destacadas.

Ao se migrar o software original do sistema SPP para o compilador Turbo-C versão 2.0, o principal problema encontrado foi a grande diferença entre esse compilador e o Microsoft C versão 6.0. Parte do tempo do projeto foi dedicado a esta etapa, visto que o sistema é relativamente grande e complexo.

Uma outra dificuldade encontrada foi na utilização de software de domínio público. A documentação disponível é normalmente precária e muitas vezes consiste unicamente no código fonte. Isso faz com que se leve um tempo excessivamente longo no entendimento do software, principalmente nos aspectos relativos à sua implementação. Não se pode, porém, deixar de salientar a importância dessa classe de software, uma vez que sua utilização permite reduzir em muito o esforço dispendido na implementação de novos sistemas.

As restrições de memória impostas pelo sistema MS-DOS foi outra dificuldade

encontrada no desenvolvimento do projeto. Alguma modificação foi necessária para diminuir os requisitos de memória dos sistemas originais, possibilitando sua integração e execução sobre esse sistema operacional.

5.2 Contribuições Deste Trabalho

Algumas contribuições deste trabalho merecem destaque especial e são relacionadas abaixo:

- Validação do software e algoritmos utilizados nos protocolos do sistema SPP original.
- Quantificação do desempenho do sistema de comunicação original, não realizada em [TR91].
- Implementação de uma semântica alternativa mais simples e eficiente ("at-least-once") para o núcleo RPC.
- Introdução do conjunto de protocolos TCP/IP, oferecendo maior versatilidade no acesso remoto ao sistema SPP.
- Extensão do acesso a transputers via "iserver", utilizando-se LANs.
- Domínio da tecnologia utilizada em sistemas baseados em processadores transputer.

5.3 Trabalhos Futuros

A atividade de pesquisa relacionada com as áreas de conhecimento associadas ao desenvolvimento do servidor de processamento paralelo estão em constante atualização. Outros trabalhos deverão ser realizados para que o SPP e sua documentação atendam às características desejáveis de fácil utilização e manutenção. São apresentadas a seguir algumas sugestões para trabalhos futuros.

- Adaptação do ambiente SPP para a execução de outros utilitários e linguagens de programação (hoje só está disponível a linguagem OCCAM 2).
- Implementação de uma nova interface de programação que facilite o desenvolvimento de aplicações paralelas.

- Desenvolvimento de ferramentas para a depuração dos programas paralelos.
- Instalação de software/hardware para monitoração, permitindo a otimização do desempenho do servidor e das aplicações sendo executadas no banco de transputers.

5.4 Comentários Finais

O sistema de processamento paralelo desenvolvido no LaSD foi revisado e ampliado seguindo as especificações iniciais do projeto [TR91], no sentido de que as idéias originais de se ter um recurso compartilhado para processamento paralelo, disponível eficientemente através de um sistema baseado em rede local, estão mantidas.

Manipular, alterar e expandir um sistema do porte do SPP não é uma tarefa trivial. Este trabalho serviu para exemplificar as dificuldades existentes e para incentivar cada vez mais a produção de documentação de bom nível, bem como a utilização de recursos que facilitem a fase de manutenção dos sistemas.

Os resultados obtidos com as novas implementações realizadas mostram que a versão original do SPP, embora não tenha sofrido uma bateria de testes exaustiva na ocasião da sua implementação, foi altamente bem sucedida.

As soluções alternativas propostas neste trabalho demonstram a possibilidade de se adotar semânticas mais rápidas para o núcleo RPC do sistema. Cabe ao usuário adequar a versão do protocolo ao ambiente disponível para a utilização do sistema. Os resultados e o desempenho adicional obtidos com a nova semântica serão de grande valia em implementações futuras e mais complexas no sistema SPP.

Referências Bibliográficas

AN91 ANDRADE, A.L.P. Um Protocolo Dedicado para Acesso Remoto a Transputers via TDS - LASD - ICMSC- USP- Agosto, 1991

AN93 ANDRADE, A.L.P. **Um Protocolo Dedicado para Acesso Remoto a Transputers via TDS**. In: Congresso da SBC, 13., XX Semish, Florianópolis - SC, Agosto, 1993

BA87 BACON, J.M.; HAMILTON, K.G. **Distributed Computing with RPC: The Cambridge Approach**, Technical Report n. 117 - University of Cambridge, Outubro, 1987.

BA89 BAL, H.E.; STEINER, J.G.; TANENBAUM, A. S.; Programming Languages for Distributed Computing Systems; **ACM Computing Surveys**, Vol. 21, n. 3, Setembro, 1989.

BE89 BERSHAD, B.N.; ANDERSON, T.E.; LAZOWSKA, E.D.; LEVY, H.M. Lightweight Remote Procedure Call, **Operating Systems Review**, Special Issue, Vol. 23, n. 5, Dezembro, 1989.

BI84 BIRREL, A.; NELSON, B.J. Implementing Remote Procedure Calls. **ACM Transactions on Computer Systems**, Vol.2, n.1, Fevereiro, 1984.

CH90 CHANDRAS, R.G. Distributed Message Passing Operating Systems, **Operating Systems Review**, Vol. 24, n. 1, Janeiro, 1990.

CL89 Clarkson University, THE PACKET DRIVER SPECIFICATION, **User Documentation for the Packet Driver Collection**. Version 1.09, US., 1989.

CO91 COMER, D. E.; STEVENS, D.L. **Internetworking with TCP/IP Design, Implementation and Internal**, Prentice Hall, Vol II, 1991.

CO91a COMER, D. E.; **Internetworking with TCP/IP Principles, Protocols and Architecture**, Prentice Hall, Vol I, 1991.

CO88 COULOURIS, F. G. and DOLLIMORE, J. **Distributed Systems - Concepts and Design**, Addison-Wesley Publishing Company; 1988.

HO80 HOPPER, A.; **The Cambridge Ring - A Local Network**, in Harina, K.A. (ed), *Advanced Techniques for Microprocessor Systems*, Peter Peregrinus, 1980.

IE85a *IEEE Standards for Local Area Networks*: Carrier Sense Multiple Access with Collision Detection (CSMA-CD) Access Method and Physical Layer Specification.

IE85b *IEEE Standards for Local Area Networks*: Token-Passing Bus Access Method and Physical Layer Specification.

IE85c *IEEE Standards for Local Area Networks*: Token Ring Access Method and Physical Layer Specification.

IN90 INMOS Limited - **Transputer Development System** - Second Edition, Prentice Hall, Reino Unido, 1990

KA91 The Ka9Q Internet Software Package, 1991

KI89 KIRNER, C. **Sistemas Operacionais para Ambientes Paralelos.** In: Congresso SBC, 9., Uberlândia - 16 a 21 de Julho, 1989.

LE81 LELANN, G. **Motivations, Objectives and Characterization of Distributed Systems** in Lampson et al, 1981.

MU89 MULLENDER, S. **Distributed Systems** ACM Press Addison - Wesley Publishing Company; 1989.

PA93a PALMA Jr., O.; Manual de Utilização do Servidor de Processamento Paralelo (SPP). São Carlos - SP, 1993. (Relatório Fapesp)

PA93b PALMA Jr., O.; Documentação do Software do CM/SPP - bt.tsr. São Carlos - SP, 1993. (Relatório Fapesp)

PO81 POPEK, G.; WALKER, B.; CHOW, J.; EDWARDS, D.; KLINE, C.; RUDISON, G.; THIEL, G. **LOCUS: a Network Transparent, High Reliability Distributed System.** Proceedings of the Eighth SOSP, Pacific Grove, California.

RO85 ROBERTS, W.T. **A Comparison of Two Remote Procedure Call Mechanisms** Computer Science Report, Queen Mary College, London, 1985

SA90 SANTANA, R.H.C; SANTANA, M.J.; ZALUSKA, E. **TRICE - a Transparent Multi-LAN Distributed Computing System** Anais da XVI Conferência Latino Americana de Informática, 10 a 14 de Setembro, 1990, Paraguai.

SC82 SCHROEDER, M.; BIRREL, A.D.; LEVIN, R.; NEEDHAM, R.M. Grapevine: An Exercise in Distributed Computing, **Communications of the ACM**, Vol. 25, Abril, 1982.

SC84 SCHOROEDER, M.; BIRREL, A.D.; LEVIN, R.; NEEDHAM, R.M. Experience with Grapevine: the Growth of a Distributed System, ACM Transactions on Computer Systems, Vol 2, no.1, Fevereiro, 1984.

SI92 SINHA, A. Client-Server Computing, **Communications of the ACM**, Vol. 35, n. 7, Julho, 1992.

SP82 SPECTOR, A.Z. Performing Remote Operations Efficiently on a Local Computer Network, **Communications of the ACM**, Vol.25, n. 4, Abril,1982.

ST84 STALLINGS, W. Local Networks **Computing Surveys**, Vol.16, n.1, Março 1984.

TA85 TANENBAUM, A. S.; van RENESSE, R. Distributed Operation Systems **Computing Surveys**, Vol. 17, n. 4, 1985.

TA86 TAROUCO, L. M. R. **Redes de Computadores: Locais e de Longa Distância**, McGraw-Hill, São Paulo, 1986.

TA89 TANENBAUM, A. S. **Computer Networks** Prentice-Hall, NJ, 1989.

TA90 TANENBAUM, A. S.; van RENESSE, R.; van STAVEREN, H.; SHARP, G.J.; MULLENDER, S.J.; van ROSSUM, G. **Experiences with the AMOEBA Distributed Operating System** Communications of the ACM, Vol. 33, n. 12, Dezembro, 1990.

TA92 TANENBAUM, A. S. **Modern Operating Systems** Prentice-Hall, 1992.

TR91 TRINDADE Jr, O.; SANTANA, M.J. **Um servidor de Processamento Paralelo Baseado em Transputers** tese apresentada junto ao Instituto de Física e Química da USP de São Carlos, em Dezembro de 1991.

UL93 ULLMAN; E. **Client / Server Frees Data - Byte, Junho, 1993.**

WE89 WEXLER, J. **Concurrent Programming in Occam2** Ellis Horwood, 1989.

WI80 WILKES, M. V.; NEEDHAM, R. M. The Cambridge Model Distributed System, **ACM Operating Systems Review**, Vol. 14, n. 1, Janeiro, 1980.

WI87 WILBUR, S. Building Distributed Systems with Remote Procedure Call, **Software Engineering Journal**, Setembro, 1987.

ZI80 ZIMMERMANN, H. OSI Reference Model - The ISO Model of Architecture for Open System Interconnection, **IEEE Transactions on Communications**, Vol.28, n.4, Abril, 1980.