Instituto de Ciências Matemáticas de São Carlos

ISSN - 0109-2569

Servidor de processamento paralelo
Manual de utilização

ANA LUISA PENA DE ANDRADE

MARCOS JOSÉ SANTANA

NIVALDI CALÔNEGO JUNIOR

ONOFRE TRINDADE JUNIOR

REGINA H. C. SANTANA

Nº 24

RELATÓRIOS TÉCNICOS DO ICMSC

SÃO CARLOS
Out./ 1994

Servidor de Processamento Paralelo - Manual de Utilização

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

Índice das Figuras

Índice de Tabelas

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:

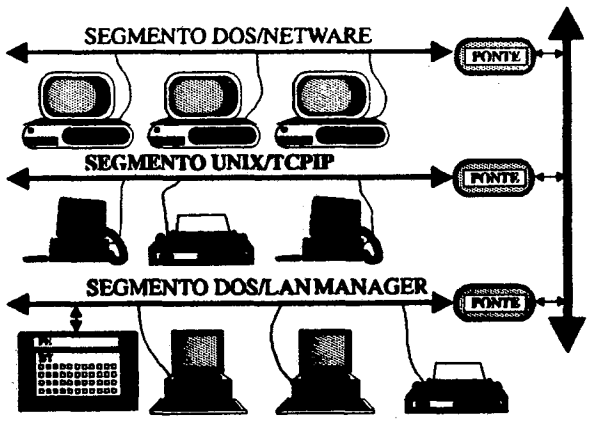
- Compartilhamento de recursos.
- Confiabilidade provida pela multiplicidade de recursos.
- Economia proporcionada pela utilização de muitas estações de trabalho, ao invés de um computador centralizado, e pelo compartilhamento de recursos através de servidores especializados.
- Modularidade, ou seja, 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).



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

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].

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.

Este documento fornece detalhes tanto para que usuários não experimentados com o ambiente do SPP o utilizem de maneira simplificada e transparente como para os usuários que necessitem interagir de forma mais efetiva com o SPP. Depois de uma abordagem geral do sistema, serão tratados todos os passos necessários para compartilhamento dos recursos computacionais da máquina.

Capítulo 2

Servidor de Processamento Paralelo

O servidor de processamento paralelo desenvolvido no LaSD 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.

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

Este capítulo apresenta uma revisão geral do SPP.

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 para utilização do sistema.

O conjunto SPP é formado 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.

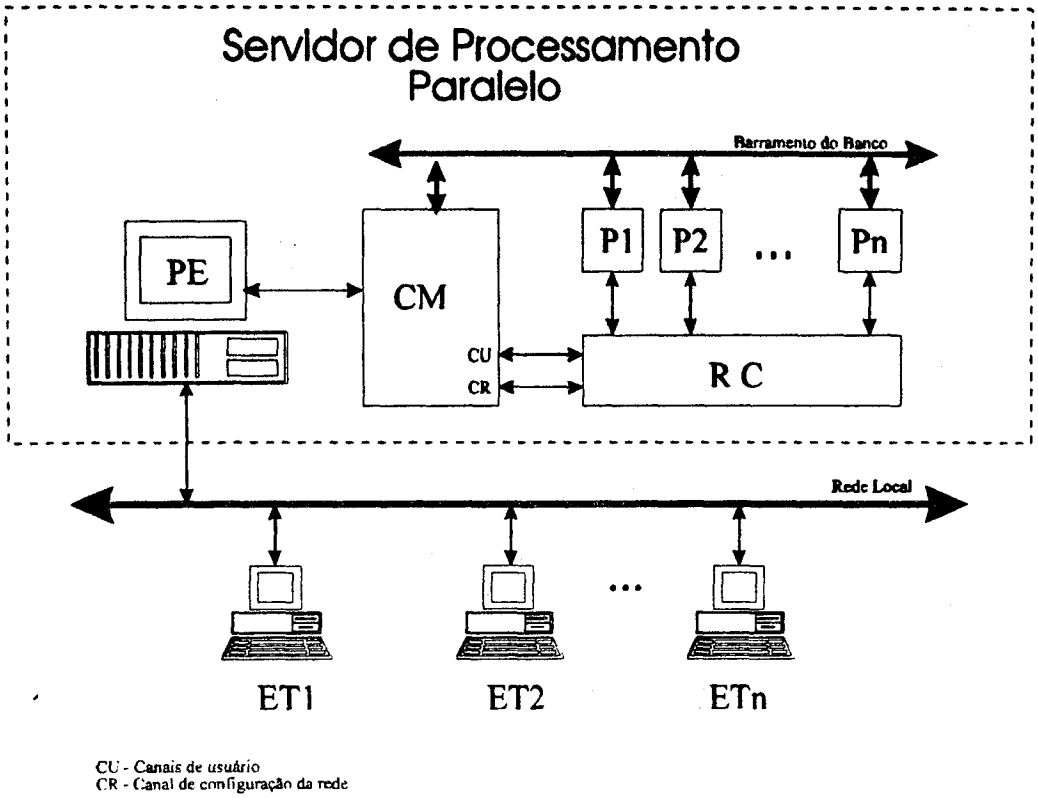


Fig. 2.1 - Estrutura do Hardware do SPP

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, conforme descrito ainda neste capítulo.

O TDS ("Transputer Development System") [IN90] foi o primeiro aplicativo disponível no Banco de Transputers. No item 2.3 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 ao 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 Projeto e Implementação do Software Original

Neste item são apresentadas as fases de projeto e implementação do sistema original. O projeto segue as diretrizes do projeto estruturado de sistemas [PA88], que tem como principal ferramenta o diagrama de estrutura modular. Para maior clareza nos diagramas, não estão representados os parâmetros de interface dos módulos.

2.2.1 Descrição dos Protocolos, Pacotes e Estruturas de Dados

O formato dos diversos tipos de pacotes de dados utilizados pode ser observado na figura 2.2. O pacote de mais baixo nível que circula pela rede local Ethernet eliminados os campos de preâmbulo e CRC ("Cyclic Redundancy Code"), resulta no pacote tipo PKT, empregado na interface com o "packet driver" NCSA.

O protocolo de comunicação do modelo cliente-servidor foi baseado em um mecanismo RPC que utiliza o pacote de dados SPP. Esse pacote é transmitido dentro do

pacote PKT, apresentando os seguintes campos:

→TP - Tipo do Protocolo. Esse campo permite a utilização de mais de um protocolo por tipo de pacote em futuras expansões do sistema.

→DT - Destino. Indica o destino do pacote.

→PR - Identificador da primitiva. Código da primitiva do sistema SPP. As seguintes faixas de código estão definidas:

0x00 - 0x3f primitivas atendidas pelo PE
0x40 - 0x7f primitivas atendidas pelo CM

→ET - Estado. Informações diversas sobre a requisição ou resposta da primitiva, codificadas bit a bit:

b7 → seqüência de chamada da primitiva
b6 → uso futuro
b5 → erro na execução da primitiva
b4 → requisição não válida
b3 → primitiva não existe ou permissão negada
b2 → sessão não existe
b1 → "time out" na execução da primitiva
b0 → uso futuro

→TM - Tamanho do pacote SPP.

O protocolo SPP dá suporte direto ao mecanismo RPC. Não são utilizadas chamadas aninhadas de procedimento, apesar de suportadas. Cada requisição e cada resposta ocupam um único pacote SPP, em ordem estrita, isso é, somente podem ser feitas novas requisições uma vez recebida a resposta da requisição anterior. A política adotada para o tratamento de falhas é "no máximo uma execução" [WI87]. O paradigma intrínseco do mecanismo RPC e as restrições acima simplificam a tarefa de validar o protocolo utilizado.

PACOTE PKT

ED- endereço ethernet destino
EO- endereço ethernet origem
TP- tipo do pacote (blue-book ethernet)

PACOTE SPP

TP- tipo do protocolo (b7->requisiçao/resposta)
DT- destino(processo/sessao)
PR- identificador da primitiva
ET- estado
TM- tamanho do pacote SPP

PACOTE SP*

B0<>0 B0+256*B1=tamanho da mensagem SP
B2=identificador de primitiva TDS ou resultado da execuçao
B0==0 B1=identificador da primitiva SPP
B2+256*B3=tamanho do pacote SP*

Fig. 2.2 - Definição dos Pacotes de Dados

O protocolo SP* é baseado no protocolo INMOS SP, utilizado pelo sistema de desenvolvimento TDS [IN90].

Diversas estruturas de dados foram utilizadas na implementação do software do sistema SPP. Apresenta-se a seguir aquelas relacionadas com a operação básica do sistema e que podem contribuir para o seu entendimento.

Estruturas de dados na estação de trabalho:

Tabela de servidores - TSERV (indexada pelo ide_servidor)
end_serv[] - endereço do servidor
ide_sessão[] - identificador da sessão do usuário
seq[] - seqüência de requisição para cada primitiva
to[] - contador de "time out" para cada primitiva

Estruturas de dados no processador de entrada:

Tabela de sessões - TSES (indexada pelo ide_sessão)
cod_usu - código de identificação do usuário
end_est[] - endereço da estação
ide_proc - identificador do processo do usuário
seq[] - seqüência de requisição de cada primitiva
tam_resp[] - tamanho máximo da resposta para cada primitiva
*resp[] - ponteiros para as últimas repostas de cada primitiva

Tabela de Usuários - TU (indexada pelo cod_usu)
nome_usu[] - nome do usuário
senha[] - senha do usuário
perm[] - permissão de execução para cada primitiva
extensão - estatística e histórico de utilização (uso futuro)

Estruturas de dados no controlador de módulo:

Tabela de Processadores - TP (indexada pelo número do conector do módulo onde cada processador está fisicamente alojado)

Tabela de lotes - TLOT (indexada pelo ide_lote)
ide_sessão - identificador da sessão
conf_atual[] - configuração do lote, conforme o formato (para cada processador pertencente ao lote).

2.2.2 Estações de Trabalho

Os diagramas de estrutura modular correspondentes às tarefas executadas nas estações de trabalho aparecem na figura 2.3. Observa-se nessa figura a estrutura de chamada de uma primitiva genérica.

Cada primitiva tem associado um valor de tempo ("time out"), após o qual é feita uma nova requisição, conforme a política de tratamento de falhas implementada. Após um

certo número de falhas, especificado pela constante do sistema TIME_OUT_SPP, considera-se que o servidor não pode atender a requisição e as tentativas são interrompidas.

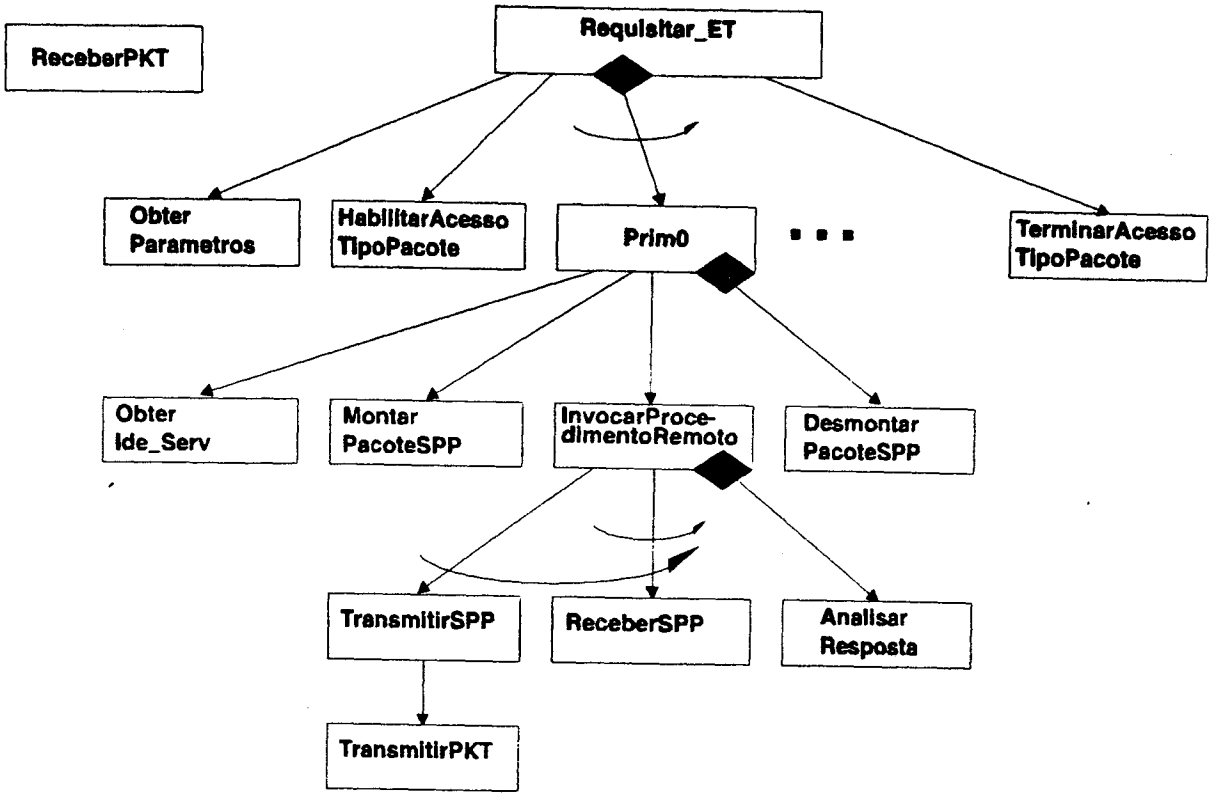


Fig. 2.3 - Diagrama de Estrutura Modular - Estação

2.2.3 Processador de Entrada

O processador de entrada está diretamente acoplado à rede local, comunicando-se através dela com as estações de trabalho. Também se comunica com o banco de Transputers. Sua função principal consiste no roteamento de pacotes entre as estações de trabalho e o banco.

O diagrama de estrutura modular correspondente às tarefas executadas no PE é apresentado na figura 2.4.

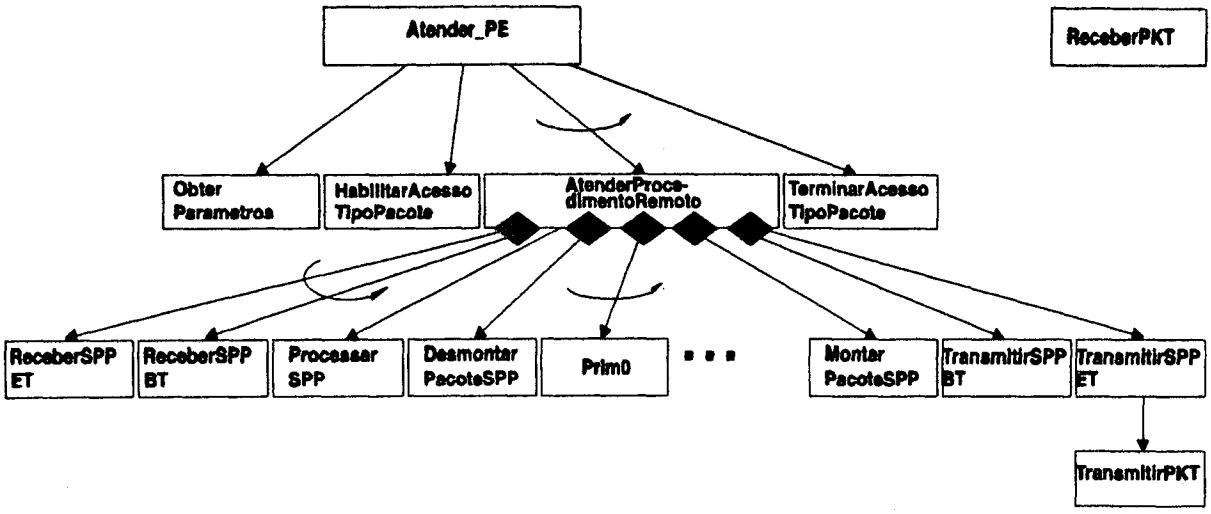


Fig. 2.4 - Diagrama de Estrutura Modular - Servidor

Algumas primitivas do mecanismo RPC são atendidas pelo processador de entrada. Entre essas estão as primitivas de controle de acesso ao sistema:

→ Login (nome_usuário, senha, *ide_sessão) - Inicia uma sessão no servidor.

→ Logout (ide_sessão) - Encerra uma sessão no servidor.

2.3 Instalação do Sistema de Desenvolvimento TDS

A instalação do sistema de desenvolvimento TDS introduz no sistema SPP ferramentas para o desenvolvimento de programas na linguagem Occam. Isso possibilita um mínimo de funcionalidade para o sistema.

2.3.1 Organização do Sistema de Desenvolvimento TDS

O TDS pode ser migrado para diferentes máquinas e utilizado com vários sistemas baseados em transputers (T400, T800). Essa flexibilidade é possível porque a INMOS fornece o código fonte do aplicativo "iserver", permitindo que sejam feitas as alterações necessárias para que se consiga a configuração desejada do ambiente TDS.

O sistema de desenvolvimento TDS consiste em um módulo executado no computador hospedeiro, o ISERVER[IN90], e um módulo principal, o TDS, executado no processador raiz de uma rede de Transputers. O ISERVER possibilita o carregamento de programas objeto no processador raiz da rede e pode permanecer ativo, atendendo

requisições de serviços relativos a arquivos, vídeo e teclado provenientes do TDS ou de um programa do usuário.

A organização geral do software do sistema, bem como a interrelação entre os diversos módulos que o compõem são apresentadas na figura 2.5. De acordo com a organização do iserver, todas os procedimentos que interagem com a placa do transputer estão localizadas no módulo b004link.c e aparecem tracejadas na figura 2.5.

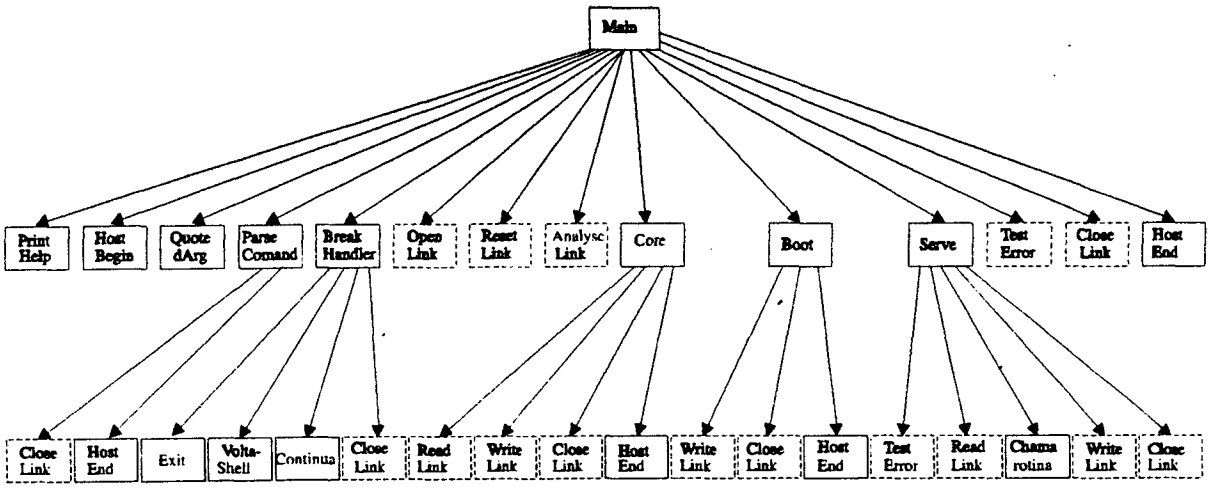


Fig. 2.5 - Diagrama de Estrutura Modular - ISERVER

A função original dos módulos do ISERVER foi na maioria dos casos preservada, conforme descrição em [AN93].

O sistema RPC desenvolvido foi incorporado ao servidor a partir do módulo b004link.c, conforme ilustra a figura 2.6.

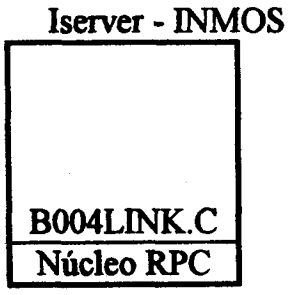


Fig. 2.6 - Incorporação do Núcleo RPC ao Servidor INMOS

Depois de se apresentar uma visão geral do sistema SPP, nos próximos capítulos serão mostrados os passos necessários para a utilização do sistema.

Capítulo 3

Instalação e Utilização do Sistema SPP - Versão Original

Neste capítulo são descritos os passos necessários para a instalação e a utilização do sistema SPP na versão original.

3.1 Sistema SPP - Versão Original

O Sistema de Desenvolvimento TDS é constituído de duas partes, uma delas executada no computador hospedeiro, o "iserver", e a principal, o TDS, executada no processador raiz de uma rede de Transputers. O "iserver" possibilita o carregamento de programas objetos no processador raiz da rede, permanecendo ativo, atendendo requisições de serviços relativos a arquivos, vídeo e teclado, provenientes do TDS ou de um programa do usuário.

O "iserver" teve a sua estrutura ligeiramente alterada para a operação com o SPP [AN91,AN93]. Uma vez que o sistema opera através de uma rede local, toda comunicação entre o TDS ou programa do usuário com a Estação de Trabalho (ET) é feita através da rede. As alterações realizadas no "iserver" concentraram-se nos módulos de comunicação, os quais passaram a receber e enviar os pacotes SP através da rede para o servidor.

Os procedimentos necessários para a configuração do ambiente de utilização do Servidor de Processamento Paralelo, com o Sistema de Desenvolvimento TDS, foram descritos no capítulo 2. Qualquer usuário com acesso a uma ET do sistema pode executar o TDS com o auxílio dos recursos computacionais do Banco de Transputers, desde que esses recursos estejam disponíveis. A seguir são descritas as primitivas disponíveis no sistema:

- **Abrir uma sessão**

Abrir uma sessão no sistema consiste em se identificar perante o sistema, fornecendo um nome identificador. Na atual implementação, não é verificado nenhum tipo de senha de proteção, de forma que qualquer pessoa pode ter acesso ao sistema desde que use um nome cadastrado no arquivo de login.

Para se abrir uma sessão, na ET devidamente configurada e preparada para trabalhar como estação do SPP, execute o utilitário **login**, digitando

```
login <user>
```

na linha de comando do sistema operacional.

O sistema irá responder com um pedido de identificação, caso não tenha sido passado na linha de comando.

O utilitário **login** executado na ET, nada mais é do que um comando remoto executado no Processador de Entrada (PE) através de uma chamada RPC ao servidor. No servidor (PE), fica reservado um identificador de sessão que será usado para todas as próximas operações do usuário.

- **Alocar os processadores**

Os recursos computacionais paralelos (placas de Transputers) são compartilhados entre até quatro usuários, segundo a implementação atual. Portanto, cada usuário deve reservar os recursos necessários para a sua sessão de trabalho. Para isso, deve executar o utilitário **lot** na linha de comando da estação:

```
lot
```

O sistema irá solicitar o número de processadores que serão alocados ao usuário:

```
Nro.de procs:
```

Esse utilitário faz uma chamada remota ao SPP, sendo que a alocação é feita pelo Controlador de Módulo, uma vez que, quando a chamada chega ao PE, ela é roteada para o Controlador de Módulo do Banco.

Caso não haja um número de processadores suficientes, uma mensagem de **"Erro na alocação dos procs."** aparece. Outras mensagens são geradas quando ocorre algum problema de comunicação com o servidor.

Como resultado da alocação do lote, é apresentado na estação a seguinte mensagem:

```
Lote Alocado            : <lote alocado>.
Processador Raiz        : <nro do proc raiz>.
Canal do Proc. Raiz     : <canal do proc raiz>
Processadores Alocados  : <lista de procs.>
```

Um exemplo dessa mensagem para um usuário que solicite 3 processadores seria:

```
Lote Alocado            : 01
Processador Raiz        : 05
Canal do Proc. Raiz     : 3
Processadores Alocados  : 05 06 08
```

Após a alocação dos processadores por parte de um usuário, o sistema tem a configuração física apresentada na Figura 3.1.

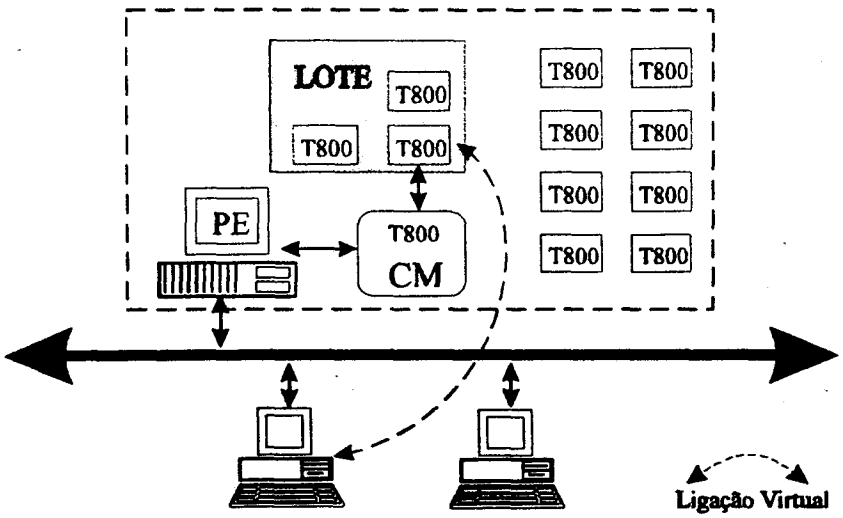


Figura 3.1 - SPP após a alocação de 3 Procs.

Pode ser observado na Figura 3.1 que um dos processadores do lote (processador raiz) é conectado virtualmente à ET e não há ligação nenhuma entre os Processadores alocados.

Os processadores alocados são reservados ao usuário até que ele execute o utilitário **dlot**, para liberar os processadores.

- **Executar o programa TDS**

A execução do TDS normalmente é inicializada através do comando **TDS3** digitado na linha de comando do computador hospedeiro. **TDS3** é um arquivo em lote, contendo comandos que têm acesso ao programa "iserver", encarregado de controlar a comunicação hospedeiro/Transputer. O "iserver" carrega o TDS no processador raiz e passa a controlar disco, teclado e vídeo na estação.

No caso do SPP, o programa "iserver" foi modificado e passou a chamar **iserverb**. Com isso, o arquivo **TDS3.BAT** também foi alterado, passando a denominar-se **TDSB.BAT**, portanto, para a execução do TDS em uma ET, após os procedimentos de abertura de sessão e alocação de processadores, basta o usuário digitar:

```
tdsb
```

Daí para frente tudo passa a ser transparente para o usuário, isto é, aparentemente o usuário tem uma placa de Transputer ligada diretamente à sua ET, como na Figura 3.2.

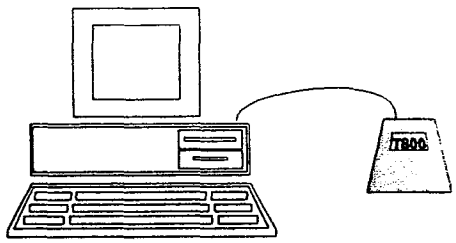


Figura 3.2 - ET ligada diretamente ao processador Transputer

- **Liberar os processadores**

Quando uma sessão de trabalho é terminada, o usuário deve liberar os processadores para que fiquem disponíveis a outros usuários. Isso é feito de forma simples através do utilitário **dlot**.

O comando `dlot` libera o lote de transputers reservado ao usuário e desfaz a ligação virtual entre a estação e o processador raiz, voltando o sistema à configuração original.

- **Liberar a sessão**

O processo de liberar o lote de Transputers, apenas libera os processadores para que possam ser usados por um usuário em outra ET. A sessão iniciada é mantida. Isso significa que um usuário pode abrir uma sessão, alocar um lote com 1 processador para a execução do TDS, liberar esse lote e alocar outro com vários processadores para executar um programa em uma rede, tudo em uma mesma sessão de trabalho. Como o sistema possui um limite máximo de 4 sessões, caso as quatro estejam sendo utilizadas, nenhum novo usuário poderá ter acesso ao sistema. Portanto, quando uma sessão de trabalho é terminada e o usuário não mais usará os recursos do SPP, a sessão obtida com o utilitário "login" deve ser liberada. Nesse caso deve ser usado o utilitário "logout" como abaixo:

```
logout
```

A execução desse utilitário libera a sessão, pemitindo que um novo usuário possa ter acesso ao sistema.

Capítulo 4

Instalação e Utilização do Sistema SPP - Versão TCP/IP

Neste capítulo são descritos os passos necessários para a instalação e a utilização do sistema SPP na versão TCP/IP.

4.1 Arquivos de Configuração do Programa *net*

O pacote Internet Ka9Q, utilizado para esta etapa do trabalho, 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 ambiente gerado pelo programa net pode ser configurado através de parâmetros fornecidos em diversos arquivos.

• O Arquivo AUTOEXEC.NET

O arquivo AUTOEXEC.NET (às vezes chamado de STARTUP.NET, por exemplo no sistema Unix) funciona como o arquivo AUTOEXEC.BAT no DOS (por isso a semelhança do nome). Quando o programa *net* é executado, ele lê o arquivo AUTOEXEC.NET e executa todos os comandos existentes, como se eles tivessem sido passados para o programa via teclado. Esse é um mecanismo fácil para se

definir a configuração inicial do sistema, incluindo o nome do *host*, os parâmetros AX.25, interfaces usadas, servidores a serem disparados e variáveis do protocolo em geral.

O arquivo do sistema em estudo tem a seguinte configuração:

```
hostname yara
ip address [143.107.231.67]
smtp timer 1200
tcp mss 512
log net.log
tcp window 432
start telnet
start ftp
start spp
start echo
start finger

attach packet 0x60 le0 8 1500

ifconfig le0 ipaddress 143.107.231.67 broadcast 143.107.231.67 netmask
0xffffffc0

route drop 143.107.231.67 le0
route add [143.107.231.64]/26 le0
route add [143.107.231.128]/26 le0 143.107.231.65 1
route add default le0 143.107.231.66 1
```

• O Arquivo FTPUSERS

Como o sistema DOS foi projetado para ser um sistema operacional mono-usuário, ele não fornece controle de acesso aos arquivos e eles podem ser eliminados, copiados ou modificados pelo usuário local. Não é uma boa política permitir que usuários remotos em um sistema baseado em rede tenham todos esses privilégios.

O arquivo "/FTPUSERS" (localizado na máquina servidora) é usado para controlar o acesso remoto de FTP. Se por um acaso a máquina servidora não possuir esse arquivo, o servidor FTP não poderá ser utilizado. Um usuário remoto deve primeiro executar um login ao sistema, dando um nome e uma senha (que devem estar nesse arquivo), para que possa fazer qualquer transferência de arquivos. O arquivo "/FTPUSERS" também é utilizado para o controle de acesso ao SPP.

Cada entrada do arquivo "/FTPUSERS" consiste em uma linha com o seguinte formato:

username password path1 permissions1 path2 permissions2 ...

devendo existir exatamente um espaço entre um campo e outro e as linhas de comentários começam obrigatoriamente com um "#" na coluna 1.

"username"	É o nome de login do usuário.
"password"	É a senha associada ao nome do usuário. Se for um "*" (um asterisco) qualquer senha é aceita.
"/pathN"	Caminho a partir do qual se tem permissão.
"permissionsN"	É um número decimal especificando qual a permissão para acesso aos arquivos. Se o bit menos significatico (0x1) for colocado em 1, o usuário tem permissão de leitura.. Se o próximo bit (0x2) for colocado em 1, a permissão é de se criar um novo arquivo, desde que não sobre-escreva um que já exista. Se o terceiro bit (0x3) for colocado em 1, o usuário pode escrever, mesmo que já exista um arquivo com o mesmo nome e ainda pode apagar arquivos.

Tabela 4.1 - Linha de Entrada do Arquivo "/FTPUSERS"

Segue abaixo um exemplo de configuração para o arquivo "/FTPUSERS".

```
# arquivo de usuarios e senhas
# o usuario user com senha user, tem qualquer tipo de acesso ao servidor
# a partir do diretorio \
user user \ 7
```

• O Arquivo HOSTS.NET

O Arquivo HOSTS.NET fornece um mapeamento entre os endereços Internet e os nomes simbólicos dos *hosts*. É usado pelo programa *net* para descobrir qual endereço IP usar a partir do nome do *host*.

A seguir é apresentado um exemplo desse arquivo para o ambiente de desenvolvimento utilizado neste trabalho:

```
# ambiente de desenvolvimento (LaSD)
143.107.231.67 yara
143.107.231.68 curupira
143.107.231.69 boitata
```

```
# micros do ICMSC
143.107.231.1     xavante
143.107.231.2     ianomami
143.107.231.3     caiapo
143.107.231.4     ioruba
143.107.231.5     caiua
143.107.231.6     maia
143.107.231.12    banto
143.107.231.13    xapacura
143.107.231.17    tapuia

# Roteadores (Rede-Ensino e Rede-Pesquisa)
143.107.231.66    xapa1
143.107.231.30    federacao
```

4.2 Sistema SPP - Versão TCP/IP

Para um bom funcionamento do sistema *net*, modificado para *spp.exe*, os arquivos de configuração, descritos no item anterior, devem estar instalados no diretório raiz, para evitar que se especifique o caminho na linha de comando. Depois copia-se o programa *spp.exe* no diretório desejado para a utilização.

Para utilizar o sistema deve-se digitar na linha de comando

```
spp
```

e a partir desse ponto já estará sob execução na *shell* do ambiente TCP/IP.

O sistema SPP foi implementado como um serviço fornecido pelo net original. Assim para abrir uma sessão para comunicação com o Banco de Transputers deve-se digitar, na *shell* sistema do net, o comando:

```
net> spp boitata
```

ou também

```
net> spp 143.107.231.69
```

que é o endereço IP da máquina servidora. Assim o sistema já solicitará o nome do usuário seguido da sua senha, que devem estar cadastrados no arquivos FTPUSERS da máquina servidor, corforme descrito no item anterior.

- **Alocar os processadores**

  Para se alocar um lote de transputers no banco deve-se digitar o comando:

```
spp>lot <nro_processadores>
```

Caso não haja um número de processadores suficientes, uma mensagem de **"Erro na alocação dos procs."** aparece. Outras mensagens são geradas quando ocorre algum problema de comunicação com o servidor.

Como resultado da alocação do lote, é apresentado na estação uma mensagem informando quantos transputers foram alocados e a seguinte mensagem aparece na tela do servidor:

```
Lote Alocado            : <lote alocado>.
Processador Raiz        : <nro do proc raiz>.
Canal do Proc. Raiz     : <canal do proc raiz>
Processadores Alocados  : <lista de procs.>
```

- **Executar um programa "standalone" no Banco de Transputers**

Para se executar um programa (previamente transferido para a máquina servidora) "standalone" no banco de transputers deve-se digitar o comando **iserv** que executa o programa e grava o resultado em um arquivo.

```
spp>iserv
```

- **Liberar os processadores**

Para se liberar um lote de transputers no banco deve-se digitar o comando:

```
spp>dlot
```

que informa se o lote foi liberado com sucesso.

- **Liberar a sessão**

Para se liberar a sessão deve-se digitar o seguinte comando:

```
spp> quit
```

Com esse comando volta-se para a *shell* do sistema net e para voltar ao DOS, deve-se digitar:

```
net> exit
```

Finalizando assim a operação do sistema.

Capítulo 5

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.

Os recursos do SPP estarão sendo constantemente incrementados e aperfeiçoados, exigindo que sua documentação seja atualizada e divulgada aos usuários.

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.