SERVIDOR DE IMPRESSÃO
PARA UM AMBIENTE COMPUTACIONAL
DISTRIBUÍDO

Rita Theresinha de Miranda Furquim

ORIENTADOR

Prof. Dr. Marcos José Santana

Dissertação apresentada ao Instituto de Ciências Matemáticas de São Carlos, da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre na Área de Ciências de Computação e Matemática Computacional.

São Carlos
1992

Dedico este trabalho aos
meus filhos Gustavo, Gabriel
e Anelisa.

AGRADECIMENTOS

Ao professor Marcos por ter acreditado em mim, agradeço a orientação conduzida com seriedade e segurança; agradeço ainda a paciência, a atenção, o incentivo que foram motivos para que esse trabalho se concretizasse.

Ao professor Onofre e à Ana Luisa pelas condições favoráveis que me proporcionaram.

Aos técnicos do LASD, Eduardo e Cabral, que nunca me negaram um pedido ou mediram esforços para me atender, quando necessitava da colaboração dos mesmos.

Às secretárias Laura e Beth pela eterna atenção desde o primeiro momento em que cheguei a este Instituto.

Às amigas Regina Sousa e Renata Cicilini pelas constantes palavras amigas e sinceras, que muito me incentivaram.

Às instituições FAPESP, CAPES e CNPQ pelo apoio que prestam ao desenvolvimento do grupo.

RESUMO

Este trabalho discute tópicos fundamentais relacionados com a conceituação básica sobre sistemas computacionais distribuídos, redes locais e servidores de impressão, destacando-se protocolos de comunicação e formas de implementação dos servidores de impressão.

Uma arquitetura para servidores de impressão operando em sistemas computacionais distribuídos baseados em rede local é apresentada. A organização proposta e implementada apresenta vantagens em relação a outras existentes, principalmente em termos de portabilidade do software, permitindo atendimento a uma comunidade de usuários diversificada.

A B S T R A C T

This work discusses fundamental concepts related to distributed computing systems based on local area networks and printer servers. Communication protocols and approaches for printer server implementation are discussed in great detail.

An architecture for LAN-based printer servers is presented. The system proposed and implemented has advantages over other existing systems, mainly in terms of software portability, allowing easy instalations in different environments and, therefore, extending the service to a greater user community.

ÍNDICE

capítulo I - INTRODUÇÃO.....	01
1.1 - Considerações gerais.....	01
1.2 - Organização da dissertação.....	03
capítulo II - REVISÃO BIBLIOGRÁFICA.....	05
2.1 - Considerações iniciais.....	06
2.2 - Sistemas Computacionais Distribuídos baseados em re- des locais de computadores.....	07
2.2.1 - Redes de computadores.....	07
2.2.2 - Sistemas Distribuídos.....	12
2.2.3 - Elementos básicos.....	15
2.2.4 - Sistemas existentes.....	18
2.3 - Protocolos de Comunicação.....	22
2.3.1 - Organização do Software.....	22
2.3.2 - Chamada de Procedimento Remoto.....	25
2.4 - Servidores de Impressão.....	28
2.4.1 - Definição.....	28
2.4.2 - Organização geral.....	29
2.4.3 - Servidores de Impressão existentes.....	33
2.5 - Considerações finais.....	35
capítulo III - ORGANIZAÇÃO GERAL DO SISTEMA.....	36
3.1 - Considerações iniciais.....	37
3.2 - Visão global do Servidor de Impressão.....	38
3.3 - Arquitetura do Servidor de Impressão.....	42
3.4 - Considerações finais.....	45
capítulo IV - O PROTOCOLO DO SERVIDOR DE IMPRESSÃO.....	46
4.1 - Considerações iniciais.....	47
4.2 - Características gerais.....	49

4.3 - As primitivas de comunicação.....	53
4.4 - O mecanismo de comunicação.....	55
4.5 - Considerações finais.....	57
capítulo V - O SERVIDOR DE IMPRESSÃO.....	58
5.1 - Considerações iniciais.....	59
5.2 - Estrutura modular do Servidor de Impressão.....	60
5.2.1 - Estruturas de dados.....	60
5.2.2 - Estrutura modular do software do Servidor de Impressão.....	66
5.3 - Considerações finais.....	68
capítulo VI - TESTES E VALIDAÇÃO.....	69
6.1 - Considerações iniciais.....	70
6.2 - Software de comunicação.....	70
6.3 - Chamada de Procedimento Remoto.....	72
6.4 - Os serviços disponíveis no Servidor de Impressão..	73
6.5 - Considerações finais.....	74
capítulo VII - CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS.....	75
7.1 - Sobre o estudo bibliográfico.....	76
7.2 - Sobre as diretrizes do trabalho.....	77
7.3 - A continuidade do trabalho.....	79
APÊNDICE A1.....	81
APÊNDICE A2.....	84
APÊNDICE A3.....	89
REFERÊNCIAS.....	93

CAPÍTULO I

INTRODUÇÃO

1.1. CONSIDERAÇÕES GERAIS

Os sistemas computacionais distribuídos têm passado por um processo de desenvolvimento contínuo, abrangendo considerações técnicas e econômicas, com o propósito de atendimento das necessidades de compartilhamento de recursos, alta disponibilidade de elementos do sistema, alta mobilidade dos usuários, entre outros fatores vantajosos provenientes do uso desses sistemas [MUL89]. Os sistemas distribuídos exploram a potência e flexibilidade de computadores, possibilitando o acesso a recursos de hardware e informações, de forma compartilhada, através de redes de comunicação.

As redes de comunicação empregadas são classificadas de acordo com as distâncias envolvidas. Em um extremo existem as redes de longa distância (WANs - Wide Area Network), que possibilitam a interconexão de máquinas dispersas fisicamente, atingindo distâncias de centenas de quilômetros. Por outro lado existem as redes locais (LANs - Local Area Network), que apesar de terem sido desenvolvidas a partir dos conceitos utilizados nas redes de longa distância, apresentam desempenho extremamente superior, principalmente devido ao envolvimento de interconexões de pequena distância: velocidades mais elevadas com baixa taxas de erros se tornaram possíveis. Mais recentemente surgiram as redes metropolitanas (MANs - Man Area Network) que se colocam entre as WANs e as LANs, explorando o avanço constante da tecnologia [COU88].

Para possibilitar a comunicação entre máquinas interconectadas em uma rede há necessidade de um software de controle, o que normalmente é organizado de forma hierárquica, através de vários níveis de protocolos. Os protocolos de comunicação para redes, propostos como padrão, não são considerados ideais para uso em sistemas baseados em redes locais, devido à complexidade e ao número de camadas necessárias, o que provoca atrasos inevitáveis [TAN85]. Assim, sistemas computacionais baseados em redes locais normalmente requerem a implementação de protocolos eficientes, compatíveis com as características dessa classe de sistemas.

O desenvolvimento de protocolos para sistemas computacionais distribuídos baseados em redes locais, segue normalmente o modelo geral de troca de mensagens para a comunicação entre proces-

sos. O mecanismo considerado ideal para essa inter-comunicação é o "Remote Procedure Call" (RPC), em virtude das facilidades de implementação e da segurança exigida para esses sistemas [BIR85, COU88, SCH89].

Diversas arquiteturas têm sido consideradas na implementação dos sistemas computacionais distribuídos baseados em redes locais. Entretanto, independente da organização adotada, normalmente os sistemas seguem o paradigma de cliente/servidor, onde um conjunto de máquinas desempenha a função de servidores, prestando serviços, enquanto outros elementos requisitam serviços.

Dentre os servidores presentes em um sistema computacional distribuído, o de arquivos (que permite o compartilhamento de dados) e o de impressão (através do qual os clientes podem utilizar os dispositivos de impressão disponíveis no sistema) são fundamentais [JAN83].

Este trabalho teve origem no interesse do Grupo de Sistemas Distribuídos e Programação Concorrente do Departamento de Ciências da Computação e Estatística do Instituto de Ciências Matemáticas de São Carlos - USP, em desenvolver um sistema computacional distribuído baseado em rede local, usando o esquema cliente/servidor, para atendimento dos diversos grupos de usuários existentes, incluindo a integração de vários sistemas. Em particular, atualmente o interesse maior está voltado para o desenvolvimento de diversos servidores destinados a ambientes heterogêneos onde existe um número elevado de usuários, que necessitam utilizar recursos sem alterar o seu próprio ambiente de trabalho.

Para que o objetivo de fácil portabilidade do servidor de impressão seja atingido, há necessidade da definição de um protocolo de comunicação o mais independente possível do sistema operacional, permitindo que o sistema seja altamente flexível e adaptável. Além disso, o servidor de impressão deve possuir um sistema de "spooling", por ser um mecanismo adequado de atendimento às solicitações dos clientes de um sistema computacional distribuído.

A importância do desenvolvimento deste trabalho está relacionada também de forma direta, com a possibilidade de futuros desenvolvimentos, pois existe a possibilidade de fazer o transporte do utilitário de acesso ao servidor de impressão para diferentes sistemas operacionais, de modo que faça o gerenciamento das chamadas recebidas dos sistemas operacionais autorizados a usá-lo, bem como de seus usuários. Também podem ser incluídas outras operações no servidor de impressão desenvolvido, caso seja necessário alterar a configuração do sistema e tendo de se compartilhar, por exemplo,

o mesmo equipamento de um servidor de arquivos. Além disso, no sistema pode também ser incluído um servidor de preparação ("*binding service*") e facilmente poderá ser registrado o endereço do servidor de impressão para a *importação* de seus serviços.

1.2. ORGANIZAÇÃO DA DISSERTAÇÃO

No capítulo II é apresentada a revisão bibliográfica efetuada para o desenvolvimento deste trabalho. Inicialmente são discutidos o conceito e as características dos sistemas computacionais distribuídos baseados em redes locais de computadores, visando melhor situar o sistema que suportará o servidor de impressão desenvolvido, além da apresentação das características básicas que devem ser atingidas para que um sistema possa ser classificado como distribuído. Em seguida são abordados tópicos referentes aos protocolos de comunicação, com uma conceituação geral e análise de protocolos, destacando um mecanismo de implementação considerado adequado para o desenvolvimento deste trabalho. Uma proposta de organização que reduz o custo de desenvolvimento, aumentando a velocidade e desempenho do sistema computacional distribuído é também discutida. Em continuação são apresentados o conceito, organização geral e exemplos de servidores de impressão.

No capítulo III é dada uma visão global do servidor de impressão desenvolvido, destacando o relacionamento existente entre esse servidor e os diversos clientes do sistema, assim como a relação entre o servidor e os sistemas de impressão, fornecendo linhas gerais da arquitetura do sistema implementado.

No capítulo IV é descrito o protocolo do servidor de impressão, apresentando suas características, primitivas e o mecanismo de comunicação utilizado na implementação do sistema. Destacam-se aspectos gerais não convenientes que propiciaram modificações na organização hierárquica das camadas necessárias para a troca de mensagens.

No capítulo V são apresentadas as características gerais do servidor de impressão desenvolvido, a sua arquitetura, considerações gerais sobre o hardware utilizado e a estrutura modular do software elaborado.

No capítulo VI são apresentados os resultados dos testes realizados com o protocolo, mecanismo de comunicação e o servidor de impressão, com exemplos de implementação e utilização.

No capítulo VII são apresentadas as conclusões finais, os aspectos relevantes e sugestões para continuidade deste trabalho. Em seguida são apresentados os apêndices que descrevem as funções do "Packet Driver" utilizadas no desenvolvimento do servidor de impressão (A1); a interface dos módulos do software executado no servidor de impressão, cujo objetivo principal é o atendimento de solicitações vindas dos diversos clientes do sistema (A2); a interface das funções que executam os diversos serviços disponíveis no referido servidor (A3).

CAPÍTULO II

REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta os conceitos básicos necessários para o entendimento do ambiente onde o projeto é desenvolvido. Diversas experiências registradas na literatura são revisadas, com o objetivo de se ter um melhor embasamento prático/teórico para a tomada de decisões ao longo do projeto e implementação do servidor de impressão discutido nesta dissertação.

2.1. CONSIDERAÇÕES INICIAIS

Neste capítulo é apresentado o estudo bibliográfico realizado para o desenvolvimento deste trabalho, introduzindo-se conceitos básicos sobre redes, sistemas distribuídos e mencionando-se os elementos fundamentais desses sistemas (que servem de base para situar o ambiente e as condições desejáveis para o atendimento dos clientes de um sistema distribuído).

Também é destacado o conceito de protocolo, que permite a comunicação entre os elementos de um sistema distribuído, assim como a sua forma de organização, com ênfase no mecanismo de implementação *RPC* (*Remote Procedure Call*) utilizado no desenvolvimento deste trabalho, fornecendo subsídios para o protocolo de comunicação apresentado no capítulo IV.

São explanados alguns pontos básicos do uso compartilhado de recursos de hardware, entre os clientes de um sistema distribuído, destacando-se a confiabilidade, tolerância às falhas e transparência. São também apresentados o conceito, formas de organização de um servidor de impressão e alguns exemplos, que fundamentam a definição da arquitetura e considerações gerais do servidor discutido no capítulo V.

2.2. SISTEMAS COMPUTACIONAIS DISTRIBUÍDOS BASEADOS EM REDES LOCAIS DE COMPUTADORES

Para melhor se conceituar, classificar e exemplificar sistemas computacionais distribuídos, deve-se antes introduzir conhecimentos básicos de redes de computadores. Assim, são apresentados inicialmente o conceito de rede de computadores e os aspectos que afetam o custo e desempenho. Posteriormente, o tema principal deste trabalho (*sistemas distribuídos*) é então detalhado.

2.2.1. REDES DE COMPUTADORES

Uma rede de computadores constitui um conjunto de equipamentos computacionais ligados entre si por canais de transmissão, permitindo o compartilhamento de recursos de hardware e informações do sistema, possibilitando assim um melhor aproveitamento da potência computacional instalada.

Dependendo do escopo geográfico envolvido têm-se três tipos de redes: de longa distância (WAN - Wide Area Network); local (LAN - Local Area Network) e *Internetwork* [COU88], sendo essa última conhecida através de uma nomenclatura mais recente: MAN (Metropolitan Area Network), que são as intermediárias.

As redes de longa distância (WAN) fazem as transmissões entre os nós com as operações de armazenamento e transmissão através dos PSEs (*Packet Switching Exchanges*), que se incumbem de armazenar temporariamente cada pacote e transmiti-lo através da rede. Podem ser usadas para conectar computador a computador, a qualquer distância, porém pode-se observar que o tempo gasto para a transmissão de dados é diretamente proporcional ao número de nós utilizados na rota (caminho); evidentemente tem-se em certos casos um baixo desempenho não sendo adequado aos sistemas computacionais distribuídos em geral e sim para grupos de LANs (*Internets*) [TAN85].

Dependendo do contexto as LANs são mais adotadas, pois não requerem o armazenamento dos pacotes para posterior retransmissão e apresentam baixa latência quando não existem sobrecargas (porque tem-se um circuito de comunicação dedicado, com extensão máxima de alguns quilômetros). Segundo Bal et al [BAL89] a diferença é que a comunicação nas WANs é mais lenta e menos confiável que nas LANs. No entanto, acredita-se que no futuro não haverá distinção entre os dois tipos de redes (LAN e WAN), devido à disponibilidade crescente de linhas de comunicação para grandes distâncias, com velocidades altas e baixa taxa de erros.

Os elementos que determinam o custo e o desempenho de uma LAN são: topologia, meio de transmissão e protocolo de controle de acesso ao meio [STA84].

TOPOLOGIA

Topologia é o arranjo geométrico dos nós (equipamentos) e as interligações constituindo a rede. No caso de redes locais as topologias relevantes são barramento e anel, apesar de existirem outras básicas como estrela e árvore.

Estrela [ADR88,COU88,LAG86,STA84]

A topologia estrela (figura 2.1.a) é caracterizada por possuir um elemento de comutação central, no qual todos os demais nós da rede estão conectados e através do qual é feita a comunicação. Fica evidente que se ocorrer algum problema com esse elemento central haverá ruptura da rede, causando a parada do sistema por completo. Por outro lado, a falha de outro elemento qualquer provocará efeitos mínimos no resto da rede. Nessa topologia existe uma ligação ponto-a-ponto para a comunicação entre os nós, pois os circuitos permitem apenas a conexão de somente dois equipamentos, sem se passar por outros intermediários.

Anel [COU88,LAG86,STA84]

Na topologia anel (figura 2.1.b) tem-se a formação de um elo, onde cada nó está ligado a um elemento de repetição. Os dados circulam no anel entre os repetidores, em forma de pacotes que contêm os endereços de origem e destino, bem como os dados. O nó destino copia os dados do pacote em um "*buffer*" local, quando o pacote contém como endereço destino o próprio nó. Nesse tipo de rede os nós são conectados por ligações ponto-a-ponto, formando um arranjo circular e sem quebra.

Barramento [COU88,LAG86,STA84]

Já na topologia barramento (figura 2.1.c), tem-se uma ligação multi-ponto, pois existe uma linha única compartilhada por diversos nós. Essa topologia permite que todos os elementos compartilhem um único meio de comunicação forçando que apenas um deles possa fazer a sua transmissão, de cada vez. Cada estação verifica o meio de transmissão, reconhece e faz uma cópia de pacotes que contenham seu endereço.

figura 2.1.a Estrela

figura 2.1.b Anel

figura 2.1.c Barramento

MEIOS DE TRANSMISSÃO

Para se fazer a comunicação entre os diversos elementos da rede, há necessidade do uso de um meio físico que permita a transmissão dos dados. Os meios utilizados atualmente para essa comunicação são: par trançado (fio telefônico), cabo coaxial e fibras óticas.

O par trançado é adequado para ligações ponto-a-ponto, de baixa (ou média) velocidade, sendo de baixo custo. O cabo coaxial apresenta um custo pouco superior ao par trançado, porém suporta velocidades mais altas, sendo relativamente imune a ruídos. No caso da fibra ótica o custo se torna bastante superior, mas possibilita alta faixa de passagem, com taxa de erro baixíssima, não permitindo interferência elétrica, eletromagnética ou de ruído.

PROTOCOLOS DE CONTROLE DE ACESSO AO MEIO

Os métodos de acesso ao meio de transmissão são técnicas que permitem a utilização de forma compartilhada do canal de comunicação, pelos nós que compõem a rede [GUI89,LAM80,LAM83,LUC78], podendo ser aleatória ou controlada.

Nos métodos de acesso aleatório tem-se: CSMA (Carrier Sense Multiple Access) e CSMA/CD (CSMA with Collision Detection) mais adequados à topologia em barramento; Inserção de Registro e "Slotted Ring" mais adequados à topologia anel. Nos métodos de acesso controlado tem-se o uso de "token" nas topologias barramento e anel, conhecidos por "token bus" e "token ring", respectivamente. Atualmente, os principais métodos utilizados são CSMA/CD e barramento com "token" ("token bus").

CSMA [LAG86,LUC78,STA84]

O método CSMA é também conhecido por "*Listen before talking*" ("*ouça antes de falar*"), que consiste em: uma estação que deseja transmitir, verifica antes se o meio está ocioso e então decide fazer ou não a transmissão. Em caso positivo, após algum tempo deverá receber uma mensagem de confirmação. Se a confirmação não ocorrer, a estação deverá retransmitir a mensagem. As colisões poderão ocorrer se duas ou mais estações transmitirem ao mesmo tempo. Os algoritmos utilizados no caso do meio de transmissão estar ocupado são: "*non-persistent*" (existe a espera de um tempo aleatório); "*1-persistent*" (continua a monitoração até que encontre a rede ociosa e então transmite); "*p-persistent*" (quando a rede estiver ociosa a estação pode ou não transmitir, de acordo com uma probabilidade p).

CSMA/CD [COU88,LAG86,LUC78,STA84]

O método CSMA/CD conhecido por "*Listen while talking*" é uma extensão técnica do CSMA, que tenta solucionar os problemas causados pelas colisões. Assim, se ocorrer uma colisão o meio de transmissão é mantido inacessível por um certo tempo (o que pode reduzir a capacidade do sistema). Nesse método a estação "*ouve enquanto fala*" e se uma colisão for detectada, a mesma pára de transmitir imediatamente, sinalizando na rede o acontecimento, assegurando que todas as estações reconheçam a colisão. Após o sinal, todas as estações que estão transmitindo (e portanto *ouvindo*) cancelam o pacote corrente, tentando transmiti-lo após um certo tempo.

Para que não ocorra novamente outra colisão, toda estação envolvida no problema deve aguardar um tempo aleatório para iniciar a retransmissão do pacote.

INSERÇÃO DE REGISTRO [LAG86,STA84]

No método de inserção de registro, o pacote que deverá ser transmitido é colocado em um registrador de deslocamento, para ser inserido na rede e fazer parte do meio de comunicação. Esse registrador é inserido na rede quando esta estiver desocupada (entre dois pacotes, por exemplo) e retirado quando o registrador não contiver pacote útil, ou no retorno do pacote enviado à estação, ou ainda quando a rede estiver desocupada. Os dados são enviados por deslocamentos e a estação não poderá transmitir novos pacotes, até que o registrador seja retirado do anel. O tempo de transmissão dependerá do número e do tamanho dos pacotes, o que constitui uma desvantagem do método.

"SLOTTED RING" ou ANEL com "SLOTS" [LAG86,STA84]

No "*slotted ring*", outro método de acesso aleatório adequado à topologia anel, um conjunto de "*slots*" de tamanho fixo, circulam constantemente pela rede, possuindo cada um, um bit inicial indicativo de *vazio* ou *cheio*. Uma estação querendo transmitir aguarda por um "*slot*" *vazio*, assinala-o como *cheio* e insere o dado a ser transmitido; então aguarda pelo seu retorno marcando-o como *vazio*. Quando a mesma estação poderá transmitir novamente, depende do protocolo adotado, pois pode ser imediatamente em um novo "*slot*", ou pode esperar alguns ciclos, para dar chance às outras estações.

UTILIZAÇÃO DE "TOKEN" [LUC78,STA84]

Nos métodos de acesso controlado existe o uso de um "*token*", que é um pacote de controle responsável por regular o direito de acesso ao meio de transmissão. Caso uma estação possua o "*token*", terá o controle do meio por um tempo pré-determinado, durante o qual poderá transmitir um ou mais pacotes. Ao término, a estação passa o "*token*" para a próxima estação na sequência lógica, ou seja, caso a topologia seja anel o "*token*" circula na rede em um sentido pré-definido; no barramento as estações formam um anel lógico.

Quando uma estação deseja transmitir, deve aguardar o "token" disponível, modificá-lo para ocupado e transmitir o pacote imediatamente após o mesmo. As outras estações precisam aguardar a sua vez, sendo que o "token" só poderá ser colocado em disponibilidade pela estação que transmitir o pacote, após o término deste e após recebê-lo de volta. Nesse método pode ocorrer a perda do "token", que pode ser controlada por um mecanismo de "time-out"; o "token" pode também estar ocupado de forma persistente, o que pode ser evitado com a implementação de um "monitor-bit". O mecanismo de "time-out" usa um intervalo de tempo pré-definido pelo sistema, que determina o tempo de espera pela resposta de um pedido. No término desse tempo cessa a espera e pode ou não ocorrer a retransmissão do pedido. O "monitor-bit" é um gerenciador do "token", cuja função é verificar se o mesmo está disponível ao sistema, de tempos em tempos, de forma a regular e fiscalizar sua utilização.

2.2.2. SISTEMAS DISTRIBUÍDOS

O termo *sistema distribuído* tem uma conotação muito ampla, não possuindo uma definição consensual entre os especialistas da área [LAG86]. Pode-se dizer que não existe uma definição única, porém cada grupo de pesquisadores dita os requisitos para que um sistema seja considerado distribuído.

Enslow [ENS78] se baseia na multiplicidade de recursos lógicos e físicos, distribuição física dos componentes do sistema, transparência e autonomia cooperativa, que devem ser satisfeitas concomitantemente para que um sistema seja distribuído.

Tanenbaum e van Renesse [TAN85] definem como um sistema que aparenta ser centralizado, mas o sistema operacional na realidade é executado em várias unidades centrais de processamento (UCPs) independentes. O conceito chave nessa definição é a transparência, onde é afirmado que o uso de múltiplos processadores deve ser invisível ao usuário.

Coulouris e Dollimore [COU88] garantem que em um sistema distribuído o processamento das atividades pode estar localizado em mais de um computador e os equipamentos se comunicam através de uma rede para executarem juntos as tarefas.

Mullender [MUL89] fornece as características para que um sistema seja distribuído, incluindo: múltiplos elementos de processamento, hardware de interconexão, falha independente dos elementos e estado compartilhado.

Para cada definição encontram-se muitos sistemas que podem não satisfazê-la, mas que por sua natureza e utilização são chamados e classificados como distribuídos. Assim, sempre que se tenha um conjunto de computadores com capacidade de processamento autônomo, interconectados por uma rede, pode-se considerá-lo um sistema distribuído.

EVOLUÇÃO HISTÓRICA

Desde o surgimento dos primeiros computadores até a atualidade, podem ser consideradas quatro fases importantes [MUL89]. A primeira fase teve início com o surgimento dos primeiros computadores, caracterizados por um alto custo, utilização monousuária e dedicado a uma única tarefa de cada vez [MUL89]. Porém, as memórias tornaram-se suficientemente grandes para acomodar o sistema operacional e o programa de aplicação, surgindo uma nova fase, onde se obtinha uma utilização maximizada da máquina, na modalidade de processamento conhecida por "*batch*" (em lote) [MUL89].

Com a evolução tecnológica surgiu a terceira fase, onde havia a possibilidade de se compartilhar a máquina (modalidade "*time sharing*"), pois esta se tornou mais veloz e logicamente chegou-se a um preço menor por ciclo de máquina, possibilitando aos programadores se tornarem mais produtivos [MUL89, TAN87]. Com o surgimento do VLSI ("*Very Large Scale Integration*") chegou-se à computação pessoal, que com as LANs permitiram que as estações de trabalho se viabilizassem, superando o "*time-sharing*". As estações de trabalho surgiram em virtude das máquinas isoladas existentes não satisfazerem todas as necessidades dos sistemas atuais, como aplicações gráficas de alta interação com a tela, utilização de ambientes multiusuários, bem como máquinas pessoais em ambientes monousuários [MUL89, TAN87].

Pode-se ainda associar a cada uma dessas fases aproximadamente uma década, ocorrendo a primeira delas nos anos 50 e a última na década de 80 [MUL89]. Na década atual é previsto uma proliferação total das estações de trabalho, um baixo custo dos computadores de grande potência, com ampla integração nos ambientes computacionais. Os sistemas computacionais distribuídos, especialmente os baseados em redes locais, apresentam hoje um desenvolvimento crescente, com a experiência acumulada principalmente na última década. À medida em que a proliferação de estações de trabalho ocorre, o conceito e a aplicação de sistemas computacionais distribuídos também se difundem.

BENEFÍCIOS E LIMITAÇÕES

Um sistema distribuído pode crescer gradativamente, independente de fabricantes de hardware, pois a sua configuração pode ser expandida, reduzida ou alterada com relativa facilidade. O mesmo ocorre com novos software, que podem ser instalados, sem grandes problemas ou alterações. Os recursos de alto custo, como dispositivos físicos, dados e programas podem ser compartilhados, tornando a utilização relativamente barata, com maior disponibilidade aos usuários do sistema [MUL89].

Em virtude da replicação, pode-se ter um aumento na confiabilidade dos sistemas distribuídos, frente à necessidade de se ter diversas cópias de um mesmo arquivo armazenadas em locais distintos [MUL89]. Existe também uma maior disponibilidade dos recursos do sistema, pois a falha de um elemento não deve interromper a operação global do sistema. Porém, isto depende da confiabilidade, monitoração entre os componentes do sistema e procedimentos de detecção e recuperação de falhas [COU88,TAN85].

A mobilidade dos usuários pode também crescer, dependendo do modelo arquitetural utilizado, pois os usuários ficam livres para utilizar o sistema a partir de diversos pontos [COU88]. Por outro lado, dependendo do modelo usado, cada usuário pode precisar ter comportamento de um gerente do sistema distribuído, de forma que as atividades extras como alocação de tempo de processamento, manutenção do software, execução de "*back-up*" sejam realizadas [MUL89].

Com o objetivo de aumentar a flexibilidade do sistema, alguns usuários têm acesso à comunicação básica, enfraquecendo a segurança do mesmo. Apesar de existirem mecanismos para a melhoria da segurança, via de regra qualquer dado colocado na rede deixa de ser *segredo*. A complexidade de um sistema distribuído é bem maior que a de um sistema comum, pois os tipos de elementos, assim como o conjunto de componentes requeridos pelo sistema aumentam a dificuldade de implementação [MUL89].

Os elementos que estão presentes na maioria desses sistemas e que influenciam na sua complexidade, quando vistos separadamente não são tão complexos, porém ao se juntarem as peças é que começam a surgir os problemas [MUL89]. Já foi visto que um dos elementos básicos é uma rede local, com sua topologia, meio de transmissão e protocolo de controle de acesso ao meio, bem como os equipamentos. Além disso, devem ser destacados outros elementos, que são considerados básicos em um sistema distribuído baseado em rede local, que são os servidores.

2.2.3. ELEMENTOS BÁSICOS

Em um sistema computacional distribuído encontram-se elementos físicos e lógicos, assim como um relacionamento entre eles. Como elementos físicos tem-se uma rede de comunicação, interligando os vários computadores e demais elementos do sistema, de modo que ocorra cooperação. Como elementos lógicos tem-se os dados e o software que permite o uso desses recursos. A forma como são reunidos os elementos para compor o sistema, ou seja, a organização do mesmo, define o modelo arquitetural do sistema. Os modelos arquiteturais dos sistemas computacionais distribuídos devem ser definidos de modo a atender os requisitos básicos de transparência, confiabilidade e desempenho.

Tanenbaum e van Renesse [TAN85] definem três grandes categorias dos modelos de sistemas distribuídos: *minicomputador*, *estação de trabalho* e *banco de processadores*, garantindo que a maioria dos sistemas existentes se encaixa em um desses modelos. O modelo *minicomputador* é uma extensão do "time-sharing" central, pois o sistema consiste de poucos minicomputadores multiusuários, cada usuário em uma máquina específica, com acesso remoto às outras. No modelo *estação de trabalho*, cada usuário possui sua estação de trabalho pessoal, normalmente com potência de processamento, memória e algumas vezes com armazenamento de massa. A maior parte das atividades é realizada localmente, com compartilhamento de recursos e dados através da rede local. Uma grande quantidade de sistemas distribuídos adota o modelo *estação de trabalho* e o esquema *cliente/servidor*, formando uma organização que se refere mais aos elementos lógicos, conhecido por modelo *estação de trabalho/servidores*. O modelo "pool" (ou banco) de processadores se caracteriza pelos usuários possuírem terminais e não estações de trabalho, que são ligados a um banco de processadores, através da rede local. O usuário ao requisitar um ou mais processadores, faz com que seja aberta uma sessão de trabalho, liberando o(s) processador(es) no término do serviço. Os processadores usados nesse modelo são utilizados como um serviço, por serem gerenciados. Os computadores que fazem a interligação dos terminais à rede são chamados *PAD* (Packet Assembly Disassembly), considerados servidores mais simples, pois permitem a conexão dos terminais à rede.

Tem surgido também o modelo híbrido, segundo Coulouris e Dollimore [COU88], em virtude da inclusão de estações de trabalho na maioria dos sistemas distribuídos já existentes. Esse modelo é baseado no modelo *estação de trabalho*, incluindo-se um banco de processadores, que podem ser dinamicamente alocados para tarefas não adequadas às estações de trabalho, ou atividades que necessitam do uso de diversos equipamentos de forma concorrente.

Também é mencionado pelos pesquisadores da área o modelo integrado [COU88], que provê as vantagens dos sistemas distribuídos para redes heterogêneas que possuam computadores mono e multi-usuários. Sendo que o software de cada computador é similar a um sistema operacional para sistema multi-usuário centralizado, mais um software de rede.

SERVIDORES

No paradigma cliente/servidor o núcleo do sistema operacional deve possuir um mecanismo de comunicação para troca de mensagens, onde se utiliza com maior frequência o mecanismo *RPC*, a ser discutido posteriormente na seção 2.3.2 deste capítulo. Nesse modelo, para se requisitar algum serviço da rede local, o processo do cliente envia o pedido a um processo (normalmente) remoto, capaz de realizá-lo e de devolver uma resposta. O processo que nesse caso requisita algum serviço é chamado cliente, enquanto que o processo que executa a tarefa é dito servidor. Segundo Ravindran e Chanson [RAV89] os processos que gerenciam recursos são chamados servidores e os que acessam os recursos são chamados clientes. Os clientes e servidores, quando incluídos em um mesmo contexto, segundo Mitchell [MIT82], são chamados participantes.

Embora dê a impressão de que servidores e clientes estejam em máquinas diferentes, pode ocorrer casos em que eles compartilham a mesma máquina. Também pode ocorrer que um mesmo processo às vezes tenha comportamento de servidor, para alguns elementos do sistema e dependendo da situação, seja cliente de outro servidor [SAN89]. Como exemplo tem-se o servidor de impressão, que em algumas implementações é cliente do servidor de arquivos [SAN87] (vide exemplo na seção 2.4.3, deste capítulo) e ambos se comportam como servidores para os demais elementos do sistema. De modo geral, os servidores compõem uma classe importante e em certos casos fundamental de elementos de um sistema computacional distribuído moderno. É através dos servidores que se implementa o compartilhamento de hardware e de dados.

Exemplos de servidores de um sistema computacional distribuído incluem: o de arquivos, o de impressão, o de carga de programa ("*boot*"), o de relógio ("*time*"), o de nomes e o de autenticação, entre outros.

Para maior esclarecimento desses tópicos serão fornecidos os objetivos de cada servidor e em seguida apresentados exemplos de sistemas distribuídos com os seus respectivos servidores.

Servidor de arquivos

O servidor de arquivos tem como finalidade primordial fazer o armazenamento e gerenciamento do sistema de arquivos dos usuários. Normalmente está instalado em uma estação de trabalho com grande capacidade de armazenamento secundário, ou seja com mais de um disco rígido, para permitir a realização do "back-up" diário. Os servidores de arquivos tiveram uma evolução acentuada de forma a permitir que os usuários tivessem acesso aos dados a nível de arquivos, páginas, bytes ou registros [MIT82]. Birrel e Needham [BIR80] fornecem as características principais e discutem os problemas de implementação de um servidor de arquivos universal, que deve suportar acesso aleatório a pequenos campos, permitir o armazenamento e recuperação de arquivos inteiros, páginas ou byte e para proteção dos dados é importante suportar também transações atômicas. Um servidor de arquivos deve ser confiável, de forma que evite maiores problemas caso haja interrupção de operação por problema de hardware ou software do servidor ou do cliente [SV084]. Deve evitar que ocorra perda dos dados, bem como inconsistências, que precisam ser verificadas e detectadas para correção. Também necessita da implementação de técnicas de segurança, para evitar que usuários não autorizados tenham acesso a dados, caso o servidor de autenticação não esteja disponível, ou não realize essa função [LEV90].

Servidor de impressão

O servidor de impressão tem como função básica permitir que arquivos sejam impressos, para todos os clientes do sistema, isto é, os dispositivos de impressão disponíveis sejam compartilhados pelos diversos usuários. Um servidor de impressão pode também gerenciar filas de espera de arquivos a serem impressos, podendo dar suporte a vários tipos de impressoras. Janson et al [JAN83] afirmam que apesar de serem muitos os servidores discutidos na literatura, os básicos são o de arquivos e o de impressão, sendo esse principalmente um controlador de um recurso de hardware.

Servidor de nomes

O servidor de nomes deve gerenciar tabelas de nomes, endereços e funções para localização dos elementos e serviços do sistema. Esse servidor deve aceitar nomes em um certo domínio e mapeá-los em outro domínio [TAN85], registrando serviços, processos, etc, que devem ser publicamente conhecidos. Assim como o servidor de arquivos, um único servidor de nomes para sistemas distribuídos relativamente grandes é indesejável, pois pode ocorrer longas esperas

para se ter acesso às informações, tornando-se *gargalos* do sistema.

Servidor de carga de programa

O servidor de carga de programa, conhecido por servidor de "*boot*", faz o gerenciamento do carregamento de programas, especialmente do sistema operacional para as máquinas que não possuam disco. Por questões econômicas esse servidor pode estar instalado em uma máquina, junto a outros servidores [TAN85].

Servidor de relógio

O servidor de relógio ("*time*") gerencia o relógio interno do sistema, em tempo real, aceitando funções que permitam acertar e fornecer a hora correta [TAN85].

Servidor de autenticação

Encontra-se também o servidor de autenticação em alguns sistemas computacionais distribuídos, cujo objetivo básico é manter as relações dos serviços fornecidos pelo sistema e a classe de clientes que estão autorizados a acessá-los. Quando ocorre uma solicitação de certo cliente, faz a verificação da possibilidade de acesso ao serviço, autorizando ou não a execução do mesmo. Em alguns sistemas as tarefas deste servidor ficam diluídas no servidor de nomes.

2.2.4. SISTEMAS EXISTENTES

A seguir são descritos alguns dos sistemas existentes com o intuito de integrar os conceitos fornecidos, elucidando o objetivo a ser atingido nesse trabalho.

Sistema TRICE (Southampton University, U.K.)

O ambiente *TRICE* (*TR*ansportable *I*ntegrated *C*omputing *E*nvironment) [SAN89], é um sistema computacional distribuído que utiliza o anel de *Cambridge* ("*Cambridge Ring*"), sendo constituído de diversas redes interconectadas em estrela ("*star-ring*"). É caracterizado por um alto desempenho, relativa transparência, sendo

a organização baseada no modelo *estação de trabalho/servidores*, usando o mecanismo *RPC* para a comunicação e o paradigma cliente/servidor. Os principais elementos do sistema incluem a rede local, que interconecta as estações de trabalho aos servidores de arquivos, impressão, múltiplo, relógio, de acesso a um VAX e um "gateway".

A rede local tem a topologia *anel*, usando o par trançado como meio físico para conexão. Existe uma estação de monitoração do sistema, que deve iniciar a operação e manter uma sequência de "slots" circulando na rede. Os repetidores de sinal devem ficar no máximo a 100 (cem) metros de distância, onde devem ser acopladas as estações de trabalho e os servidores. As estações de trabalho podem ou não ter unidades de disco (flexível ou rígido) e ser de uso geral ou específico.

Cada um dos servidores de arquivos (o sistema possui vários) faz o armazenamento e gerenciamento do sistema de arquivos de um grupo de usuários, estando instalado em uma estação de trabalho avançada, normalmente com dois discos (um utilizado especialmente para a realização de "back-up"). O servidor de impressão além de imprimir os arquivos, faz o gerenciamento de filas de espera. Está instalado em uma estação de trabalho com capacidade média de armazenamento e diversas portas seriais e paralelas, para conexão de impressoras matriciais, de linha, laser, plotter, etc. O servidor múltiplo engloba os de nome, carga de programas e relógio, por questões econômicas. O servidor de acesso ao VAX permite conexão a uma ou mais máquinas VAX, através de uma estação de trabalho comum, como se fosse um terminal. O servidor "gateway" é uma porta de entrada e saída de informações, para interconexão com o mundo exterior.

Sistema de CAMBRIDGE (Cambridge University, U.K.)

O sistema computacional distribuído de *Cambridge* tem a topologia *anel* e adota o método de acesso aleatório "slotted ring" como protocolo de acesso ao meio [TAN85]. Esse sistema foi inicialmente baseado em dois componentes básicos: banco de processadores e servidores remotos [SAN89]. Os processadores são alocados dinamicamente, para uma sessão inteira de trabalho [TAN85]. No topo do protocolo de pacote básico estão um protocolo simples de *RPC* (adotando o sistema datagrama) e outro de fluxo de byte (fazendo o controle total do fluxo e de erros) [BAC89]. Os serviços são alocados através do servidor de nomes, que aceita os dados do cliente em *ASCII* (American Standard Code for Information Interchange), pesquisa em uma tabela e retorna o número do serviço, o endereço do "porto" lógico e o protocolo esperado. Para verificar se o usuário está autorizado, existe um gerenciador de tabela de objetos ativos que -

mantém um controle dos usuários que estão com sessão aberta [BAC89]. Também apresenta o serviço de gerenciamento de recursos que manipula o principal recurso do sistema que é o banco de processadores. Os servidores presentes no anel de *Cambridge* são: arquivos, nome, gerenciador de recursos, tempo, impressão, "login", terminal e de erro.

O servidor de arquivos provê um serviço básico de armazenamento, com a idéia central de ser um servidor de arquivos universal, mas com um sistema muito primitivo de proteção e nomenclatura [TAN85]. Após algumas experiências foi criado um novo servidor, que ficou conhecido como "*filing machine*", servindo como apoio ao servidor de arquivos, melhorando o desempenho do mesmo, pois possui memória "*cache*" de grande capacidade. O servidor de arquivos universal suporta um tipo básico de arquivos não estruturados, com duas variações. A primeira constitui um arquivo especial, cuja característica é a gravação das informações ser feita de forma atômica. E a segunda é um arquivo chamado índice, que consiste de uma sequência de "*slots*", que armazena um identificador permanente do usuário. Para melhorar o sistema foram incluídos minicomputadores com discos locais e várias estações de trabalho [SAN89]. A versão mais recente do sistema distribuído de *Cambridge* utiliza uma nova topologia de rede local, conhecida por "*Port Cambridge Ring*", que possui uma velocidade de transmissão muito elevada. Essa nova velocidade da rede auxiliou potencialmente os sistemas de pesquisas desenvolvidos naquela universidade [BAC89].

Sistema AMOEBA (Vrije Universiteit, Amsterdam)

A arquitetura desse sistema, segundo Tanenbaum e van Renesse [TAN85], consiste de quatro componentes principais: estações de trabalho (uma por usuário, utilizadas para tarefas que requeiram respostas interativas), banco de processadores (alocados dinamicamente), servidores especializados (diretórios, arquivos, bloco, banco de dados, banco, carga de programa, etc.), "*gateways*" (ligam o sistema a locais diferentes). Santana [SAN89] considera essa arquitetura um modelo híbrido, considerando-se os componentes e a forma de organização dos elementos desse sistema. Para a comunicação cliente/servidor, o sistema utiliza um mínimo de *RPC*, com primitivas que detectam e retransmitem mensagens perdidas. Embora haja bloqueamento nessa comunicação, existe um mecanismo especial para manipular mensagens de emergência.

Os problemas de nomenclatura e proteção são solucionados através do mecanismo de "capacidade distribuída", gerenciadas pelo processo do usuário, criptografada, contendo o "*porto*" lógico do serviço, o número do objeto, operações permitidas e um número gerado aleatoriamente para autenticar a "capacidade". Essa "capacidade"

é utilizada também para o gerenciamento de recursos, que é feito de forma distribuída, facilitando a execução de tarefas que necessitam de múltiplos processadores. Em um nível superior o banco de processadores é gerenciado por um servidor de processos, que mantém a informação atualizada dos processadores que estão livres e os que estão ocupados. O servidor de banco gerencia o pagamento dos serviços executados pelo sistema e que devem ser cobrados.

A tolerância às falhas no sistema *Amoeba* leva em consideração que as falhas em máquinas não são freqüentes e que os usuários preferem não perder desempenho do sistema, para obtenção de transparência total nas falhas. Para isso existe um serviço de carga de programa, que detecta o servidor que não enviou uma resposta esperada, em um determinado intervalo de tempo, declarando-o com problema e requerendo uma nova cópia do servidor em um banco de processadores [ROS89,TAN90].

Sistema ATHENA (MIT, USA)

O objetivo básico do sistema *Athena* é a ligação de muitas estações de trabalho através de uma rede, usando o modelo cliente/servidor, sendo que o serviço mais importante é o armazenamento remoto de arquivos [SCH89a]. As estações de trabalho podem ser: públicas (disponíveis aos membros da comunidade, como um sistema de telefone público), dedicadas (utilizadas como servidores) e privadas (para grupos particulares como docentes e pessoal administrativo) [STE88]. O número de equipamentos e periféricos do sistema é grande, tendo aproximadamente uma impressora "laser" para cada doze (12) estações de trabalho, sendo as públicas disponíveis o dia inteiro, durante todo o ano. Algumas estações de trabalho possuem monitor de vídeo colorido, podendo ter características adicionais incorporadas, como: uso interativo, texto, gráficos e som [MUR89].

O software do sistema é, em grande parte, armazenado nos servidores de disco virtual remoto, considerado como um serviço de acesso de alto desempenho às "bibliotecas" que devem ser compartilhadas [SCH89a]. Além dos serviços citados anteriormente estão disponíveis no sistema os servidores de nome, administração, autenticação, gerenciamento de base de dados, correio eletrônico, gerenciamento dos serviços, atualização, impressão, relógio, janelas [SCH89a].

2.3. PROTOCOLOS DE COMUNICAÇÃO

Para que haja o aproveitamento de recursos por todos os elementos de um sistema computacional distribuído há necessidade de se definir padrões de comunicação, de forma bem organizada e estruturada.

2.3.1. ORGANIZAÇÃO DO SOFTWARE

Em todo sistema distribuído há necessidade de troca de informações entre os elementos de software, que são executados nos diversos equipamentos. Para que se tenha intercâmbio total, exige-se a definição de um conjunto de convenções e regras, que se denomina *PROTOCOLO*. Os protocolos de um sistema distribuído têm como finalidades básicas:

- . padronizar unidades de informação, para serem trocadas entre os elementos do sistema;
- . definir forma, tipo de representação dos dados, bem como controlar velocidade de transmissão;
- . estabelecer os caminhos das informações na rede, para que ocorra a comunicação [LAG86].

Normalmente, nas arquiteturas de sistemas distribuídos, elabora-se uma organização em camadas lógicas hierárquicas, que definem os vários protocolos necessários para a troca de informações. Entre as arquiteturas, tem-se a proposta em [WAT83] onde são definidos três aspectos fundamentais: o conjunto de camadas lógicas; os problemas comuns a todas elas; e a interação global entre as várias partes do sistema. O objetivo dessa divisão em camadas no projeto do sistema é facilitar o seu desenvolvimento, pois dessa forma os sistemas mais complexos podem ser subdivididos em partes mais simples, facilitando a implementação e testes, além de não tornarem visíveis as estruturas internas e algoritmos de uma camada, para outra. Os serviços executados por uma camada podem ser decompostos em módulos, que exigem a definição de interfaces entre as camadas adjacentes e entre os módulos e uma camada. Essa interface consiste de um conjunto de "*objetos*" abstratos, com operações e parâmetros, regras para controlar a sequência dessas operações e também codificação e formatação necessárias às operações e parâmetros [WAT83]. Nesse tipo de organização, a idéia essencial é que cada camada controle a imediatamente inferior e preste serviços à ime-

diatamente superior.

Apesar da divisão de um sistema em camadas e módulos ser um tanto arbitrária, houve uma padronização pela ISO (International Organization for Standardization), referenciado por modelo OSI (Open Systems Interconnection). Esse modelo é composto por sete camadas: Aplicação, Apresentação, Sessão, Transporte, Rede, Enlace e Física, cuja representação é dada na figura 2.2.

figura 2.2 - Modelo OSI

Um hospedeiro é todo computador que utiliza a rede para enviar e receber dados, através dos PSEs, que recebem os pacotes de dados dos hospedeiros, armazena-os para transmissão na rede e recepção dos que lhe são endereçados. Os PSEs (Packet Switching Exchange) foram inicialmente propostos para a comunicação de dados, pois os canais físicos não são específicos para uma comunicação *fim-a-fim*, mas permitem o compartilhamento através de conexões lógicas. Assim, as conexões entre usuários são lógicas e não físicas, dessa forma os PSE otimizam o uso de recursos da rede, assegurando que os canais físicos nunca estejam ociosos, exceto na ausência de tráfego. Esses pacotes roteadores, segundo Kessler [KES90] são adequados apenas para tráfegos de espera intensiva.

A seguir, será apresentado um resumo das funções básicas de cada camada, baseado em [ADR88,ASC86,COU88,KES90,LAG86,ZIM80].

A camada de *aplicação* é necessária à execução de tarefas de aplicação desejadas pelos usuários, que necessitam de comunicação na rede. A camada de *apresentação* faz a conversão do formato de dados usado no equipamento, para a representação padrão da rede. A camada de *sessão* é responsável pelo estabelecimento e manutenção de conexões virtuais entre dois processos de computadores diferentes, ligados pela rede. A camada de *transporte* especifica os protocolos e classes de serviços para comunicação livre de erros entre hospedeiros através da rede. Já a camada de *rede* transfere pacotes de dados em uma rede específica, podendo ser independentes (datagramas), ou então percorrer uma conexão pré-definida (circuito virtual). A camada de *enlace dos dados* é responsável pelo controle do fluxo, com mecanismos de recuperação e transmissão livre de erros entre meios adjacentes através de ligação física. A camada *física* consiste do percurso em hardware e seus circuitos da rede, transferindo bits através do meio físico de transmissão. Nesse nível são determinadas as características físicas do equipamento a ser usado.

Apesar da divisão em camadas ter uma série de vantagens, indicando ser um tipo de organização que será muito usado na concepção e desenvolvimento de sistemas distribuídos e redes de computadores, existem também desvantagens relevantes que propiciam o enfoque em um número menor de camadas, quando se trata de rede local. Assim, para redes locais são adotados modelos mais simples, por diversos motivos, entre os quais destacam-se: cada hospedeiro em uma rede local é conectado à mesma através de uma interface de hardware relativamente simples, que junto ao software da rede, pode estabelecer uma comunicação com alta velocidade e baixa taxa de erros. Logo, as três camadas de mais baixo nível: rede, enlace e física, do modelo proposto, podem ser implementadas como um protocolo único. Dessa forma, implementando-se um software relativamente simples em cada equipamento, para executar as tarefas de receber e transmitir mensagens, irá reduzir o custo de desenvolvimento, bem como aumentar a velocidade de transmissão e o desempenho do sistema global.

Comunicação entre processos de máquinas diferentes, conectadas por uma rede apresentam problemas interessantes e questões no projeto, que não surgem com comunicações normais. Para isso há necessidade de se estruturar esses sistemas de passagens de mensagens com técnicas para se evitar perdas de mensagens na rede, no servidor ou em ambos [TAN87].

2.3.2 CHAMADA DE PROCEDIMENTO REMOTO

Para a comunicação entre processos pode-se utilizar um modo bem simples de implementação, que não exige sincronismo. Ou seja, o cliente após enviar a mensagem fica liberado para iniciar outra atividade, enquanto o servidor pode armazenar o pedido. Nesse caso, a comunicação é implementada através do mecanismo *IPC* (Inter-Process Communication) [TAN85].

Porém, existe um caso especial de *IPC*, que exige sincronismo, onde há a passagem de parâmetros em uma direção, bloqueando o processo origem até o recebimento de uma resposta e liberando-o em seguida. Esse mecanismo é o *RPC* (*Remote Procedure Call*) que tem um procedimento semelhante ao de chamadas de rotinas locais, disponível na maioria das linguagens de programação. O *RPC* difere das chamadas locais, por não poder acessar variáveis, ou ponteiros para valores do ambiente do processo origem, pois será executado em um ambiente diferente.

O mecanismo *RPC* pode ser sintaticamente ou semanticamente transparente, caso tenha a mesma sintaxe ou semântica das chamadas locais [WIL87]. Schroeder e Burrows [SCH89] garantem que com *RPC* os programadores de aplicações distribuídas não precisam se preocupar com detalhes de gerenciamento e do canal de comunicação, sendo isto realizado como chamada de procedimento local, mas com semânticas diferentes.

Coulouris e Dollimore [COU88] consideram que um sistema computacional distribuído possa ser visto como um conjunto de componentes de software, executados em vários computadores de uma rede. A combinação de programas de aplicação e servidores pode ser considerada como um programa distribuído, que requer mecanismos de construção similares aos dos convencionais. Ou seja, há necessidade da decomposição de programas em procedimentos, sendo que os componentes padronizados são incluídos através de bibliotecas. O mecanismo *RPC* apesar de ser baseado em chamada de procedimento local, possibilita a um procedimento ser executado em processos diferentes e normalmente em computadores diversos.

Em redes locais de computadores as operações remotas devem ser realizadas de forma eficiente, evitando sobrecargas, diminuindo a latência e executando programas distribuídos de forma *paralela*. Spector [SPE82] descreve um modelo que executa certas funções eficientemente, como: transmissão e recepção de pacotes; acesso às tabelas de estado de informações; implementação de uma interface de comunicação adequada; processamento de operações simples. Esse modelo, chamado de *referência remota/operação remota*, pode ser imple-

mentado de forma bem mais eficiente que os mecanismos usados normalmente nas comunicações em redes locais.

Para cada camada de software (protocolo) existem questões a serem discutidas, que são: nomenclatura, controle de erro, gerenciamento de recursos e sincronização. A nomenclatura tem como finalidade identificar o servidor, para que a comunicação seja feita com o processo desejado. O controle de erro tem o objetivo de detectar possíveis falhas do sistema, pois estas podem ocorrer no servidor, no cliente, em ambos ou na rede. Para a detecção de falhas, nas chamadas remotas podem-se utilizar semânticas adequadas, com o objetivo de se evitar que processos sejam executados mais de uma vez, gerando inconsistências no sistema, ou que surjam execuções órfãs, no caso de falhas do cliente. As semânticas normalmente utilizadas em *RPC* são: *possivelmente recebida*; *pelo menos uma vez*; *no máximo uma vez* e *exatamente uma vez* [TAN85,SHR82].

A primeira, *possivelmente recebida*, não é muito utilizada nas implementações de *RPC*, pois não garante o recebimento e a execução da atividade, apesar de ter a implementação mais fácil. Mas, levando-se em consideração a alta taxa de confiabilidade das redes locais, dependendo da necessidade do sistema, ela pode ser utilizada.

A semântica *pelo menos uma vez* garante que, em caso de falha do servidor, a chamada será executada uma ou mais vezes. Nesse caso, o cliente envia a requisição e espera pela resposta; caso não a receba em um intervalo de tempo pré-definido ("*time-out*"), retransmite o pedido. Assim, para que as chamadas não sejam executadas mais de uma vez, as operações no servidor precisam ser "*idempotent*", isto é, execuções repetidas deverão ser equivalentes a uma única.

A semântica *no máximo uma vez* é bem aceitável, pois garante que uma chamada não seja executada mais de uma vez, porém não pode assegurar que em caso de falha durante a execução, a chamada seja executada por inteiro. Assim, pedidos duplicados são detectados e eliminados, exigindo maior complexidade em termos de implementação.

A semântica mais forte é a *exatamente uma vez*, pois o procedimento no servidor será executado uma única vez, garantidamente. As ações no mesmo são atômicas, ou seja, ou são realizadas por inteiro, ou não interferem nos dados. Segundo Svobodova [SVO84] aqui ocorre *OU TUDO OU NADA*, implicando em um processo nunca ser executado parcialmente.

Shrivastava e Panzieri [SHR82] indicam alguns problemas que precisam ser enfrentados com o uso desse mecanismo, que são: seleção de uma semântica adequada, projeto de uma interface de passagem de mensagem apropriada e tratamento de situações anormais como falhas de nós. O mecanismo *RPC* é implementado através de uma biblioteca conhecida por "*stub*", tendo um para o cliente e outro para o servidor, sendo que na máquina cliente é executado o programa do usuário e no servidor os programas de serviços. Os "*stubs*" do cliente e do servidor, considerados por Wilbur e Bacarisse [WIL87] procedimentos simulados ou fictícios, têm a finalidade de interceptar os dados e prepará-los em forma de mensagens, para transmissão na rede. O "*stub*" do servidor, além de executar essas atividades, precisa também identificar o processo para repassar a chamada. Logo, o objetivo dos "*stubs*" é a manipulação de dados e podem ser gerados automaticamente ou pelo programador. Quando for gerado automaticamente, se torna necessária uma linguagem de descrição de parâmetros para definir a interface cliente/servidor, que ao ser processada produz a geração dos "*stubs*".

Na preparação de uma chamada de procedimento remoto, o servidor precisa ter o seu endereço registrado em um arquivo, fixo no programa de aplicação, ou deve ser fornecido manualmente para que seus serviços sejam utilizados. Na prática existem boas implementações com endereço fixo, através do *serviço de preparação* ("*binding service*"). Sendo que o servidor que registra seus serviços no "*binding service*" faz uma *exportação* e os clientes que requisitam o endereço a este serviço fazem uma *importação* [WIL87].

A arquitetura dos sistemas distribuídos apresenta problema de segurança diferente dos sistemas tradicionais. Normalmente a criptografia é um método de segurança muito utilizado nos protocolos de comunicação, pois permite que apenas partes autorizadas possam criar, ter acesso, verificar e/ou modificar alguns ou todos os dados que estão disponíveis no sistema.

Birrel [BIR85] afirma que para estabelecer, manter e usar o canal de dados, de modo seguro, permitindo a criptografia, é necessário resolver uma série de problemas, entre os quais cita: identificação das partes autorizadas, autenticação dessas partes e transferência dos dados de modo não vulnerável aos perigos conhecidos. Mostra ainda que a comunicação pode ser feita de forma segura usando *RPC*, sendo apresentada aos programadores como uma facilidade, provando a praticidade e conveniência do seu uso.

2.4. SERVIDORES DE IMPRESSÃO

Nesta seção são explanados alguns tópicos específicos sobre servidores de impressão para situar com maior clareza o tema do trabalho em questão.

2.4.1. - DEFINIÇÃO

Um servidor de impressão é um software que permite o compartilhamento de um ou mais dispositivos de impressão pelos clientes de um sistema computacional distribuído, operando majoritariamente como controlador de um recurso de hardware compartilhado [JAN83].

Os motivos que ainda impedem um número elevado de dispositivos de impressão em um ambiente computacional são diversos, entre os quais destacam-se:

- . o nível de ruído produzido;
- . o calor dissipado;
- . o custo, pois impressoras (e plotters) de boa qualidade ainda apresentam custo relativamente alto face à utilização;
- . outro aspecto fundamental é que a permanência constante de impressoras ao lado dos usuários, faz com que as mesmas sejam utilizadas de forma demasiada, o que muitas vezes não é necessário.

Assim, um número elevado de dispositivos de impressão em um dado sistema computacional foi considerado proibitivo ao longo da última década [CHU84], devendo manter essa tendência por mais alguns anos apesar da redução de custo registrada na década atual. Por outro lado, impressoras e outros dispositivos de impressão constituem recursos indispensáveis para várias aplicações. Em sistemas distribuídos esses recursos podem ser compartilhados (organizadamente e a um menor custo efetivo) através da adoção de servidores de impressão. Um servidor de impressão permite o acesso compartilhado a impressoras instaladas remotamente, sem necessidade do deslocamento do usuário até o equipamento ao qual está conectada.

2.4.2. - ORGANIZAÇÃO GERAL

Um servidor de impressão, em sua forma mais simples, pode ser implementado de modo que o cliente ao solicitar o seu serviço, faz uma alocação prévia de uma impressora, com o conseqüente bloqueio do dispositivo aos demais clientes do sistema [SAN87]. Ou seja, o cliente faz a solicitação; caso a impressora esteja livre, o servidor de impressão devolve uma mensagem autorizando o seu uso e faz o seu bloqueio para outros clientes. Se a impressora estiver em uso o servidor de impressão informa ao cliente, que deverá esperar até que a mesma seja liberada, fazendo a verificação da disponibilidade do recurso, de tempos em tempos. Nesse caso, o cliente não exerce efetivamente o seu papel, mas sim o papel de *escravo* do periférico.

Essa técnica de implementação, que é conhecida por *pré-alocação*, também pode levar um servidor a não atender às requisições de serviços na mesma ordem das solicitações.

Outra técnica utilizada (e bem eficiente) é o "*spooling*" (operações periféricas simultâneas), que consiste em enviar os pedidos de impressão ao servidor, que determina o armazenamento dos arquivos a serem impressos em um disco (ou mesmo em memória RAM), até que uma impressora adequada esteja disponível para imprimi-los. Esses arquivos são armazenados formando uma fila, que deverá ser gerenciada pelo servidor, enviando-os à impressora conforme a sua disponibilidade.

O pedido de impressão pode ser enviado ao:

- . servidor de impressão (como no caso da implementação adotada no sistema distribuído ITAUTEC [FER85]);
- . ao servidor de arquivos (como no sistema computacional distribuído de *Cambridge* [NEE82,WIL80]);
- . ou a uma caixa eletrônica "*spooler*", tratada como um diretório no sistema de arquivos (solução adotada no sistema distribuído da IBM em Zurich [JAN83]).

O servidor que armazena o arquivo pode ser:

- . o de impressão, se possuir capacidade de memória adequada;
- . o de arquivos, que se incumbe de avisar o servidor de impressão de tempos em tempos (como a implementação da

rede local GPLOC do SERPRO [SAN87]).

Isso quer dizer que o armazenamento do arquivo pode ser feito pelo servidor de arquivos, se o servidor de impressão não possuir capacidade de memória própria, mas nesse caso o servidor de impressão deve verificar em intervalos de tempos regulares se existe ou não arquivos na fila; ou então esse servidor pode assumir um papel *pacífico* e aguardar um aviso do servidor de arquivos informando a existência de arquivos a serem impressos.

Em qualquer caso o servidor de arquivos e o servidor de impressão podem ou não compartilhar o mesmo equipamento da rede. As impressoras não precisam estar conectadas ao mesmo equipamento do servidor de impressão. No sistema da IBM-Zurich [JAN83], as impressoras não estão conectadas ao servidor, mas diretamente à rede de comunicação através de sua própria interface, enquanto que na rede Amplus [AMP90] há necessidade de que as impressoras estejam conectadas ao servidor.

No caso do cliente enviar o pedido e o arquivo ao servidor de arquivos, o servidor de impressão deverá consultá-lo ciclicamente à procura de arquivos a serem impressos. Nesse caso o servidor de impressão opera como um cliente do servidor de arquivos, enquanto o "spooler" é um servidor do cliente e do periférico de impressão [TAN85].

Entre os diversos serviços que podem ser fornecidos por um servidor de impressão, destacam-se:

- . a impressão de um determinado número de cópias de um certo arquivo, que pode ser local ou remoto, podendo ter operações diferentes para cada caso, para não sobrecarregar o servidor de arquivos;
- . listagem dos arquivos de um determinado cliente, que estão na fila, podendo fornecer a posição do arquivo na fila, bem como nome, tamanho, etc.;
- . cancelamento da impressão de um arquivo que esteja na fila de *espera*;
- . troca de posições de dois arquivos da fila, desde que os pedidos dos mesmos sejam originários de um mesmo cliente.

Para a execução dessas atividades o servidor deverá:

- . colocar um arquivo na fila, mediante um pedido do cliente;
- . obter um arquivo da fila, para enviá-lo ao "buffer" de uma impressora disponível;
- . remover um arquivo que já foi impresso, isto é, retirá-lo da fila e apagá-lo da memória, caso já tenha terminado o serviço, ou caso o cliente tenha solicitado o seu cancelamento.

A fila de arquivos precisa ser gerenciada e para isso existem diversas técnicas disponíveis.

FIFO

O método mais simples de gerenciamento é a técnica *FIFO* (First In First Out), muito utilizada nos sistemas "batch", onde o primeiro a chegar será sempre o primeiro a ser atendido (no caso impresso).

Filas com disciplina de prioridade

Nesses modelos a ordem em que os membros da fila são selecionados para serem servidos se baseia nas prioridades que lhe são atribuídas. Assim, um cliente que está sendo atendido não pode ser devolvido à fila, caso um cliente de prioridade mais alta entre no sistema. As prioridades podem ser definidas pelo sistema, ou pelo cliente através de primitivas e os arquivos são impressos de acordo com a ordem de prioridade (e não de chegada). Uma fila de prioridades pode ser vista como uma fila única, que permite inserção em várias posições, como indicada na figura 2.3.

A1	A2	...	Ai	B1	B2	...	Bj	C1	C2	...	Ck
----	----	-----	----	----	----	-----	----	----	----	-----	----

figura 2.3 Fila de prioridades

Na figura 2.3 o arquivo A_{i+1} deve ser inserido entre o arquivo A_i e o B_1 ; o arquivo B_{j+1} entre o B_j e o C_1 ; o arquivo C_{k+1} após o C_k . Sendo que em uma fila sem prioridades, um arquivo só pode ser inserido no final da mesma e as remoções seriam apenas no seu início [TRE76]. Mas, a fila da figura 2.3 pode ser vista também como um conjunto de filas, como no esquema da figura 2.4.

PRIORIDADE 1 (maior)

A_1	A_2	...	A_i	...
-------	-------	-----	-------	-----

PRIORIDADE 2

B_1	B_2	...	B_j	...
-------	-------	-----	-------	-----

PRIORIDADE 3

C_1	C_2	...	C_k	...
-------	-------	-----	-------	-----

figura 2.4 Conjunto de filas

Sob esse enfoque os arquivos A_{i+1} , B_{j+1} , C_{k+1} seriam inseridos apenas no final. Assim, têm-se inserção só no fim da fila e remoção no início, como em filas comuns. A idéia básica é só processar os arquivos da fila de prioridade i , quando a de prioridade $i-1$ estiver vazia (com i variando de 2 até a prioridade mínima). Porém pode ocorrer que um cliente, de mais baixa prioridade, fique eternamente à espera de sua vez. No entanto, técnicas podem ser utilizadas para se evitar isso, entre elas têm-se:

- . processar um número fixo de arquivos para cada fila, podendo-se obviamente variar esse número de acordo com a prioridade da mesma. Como exemplo: imprimir no máximo quatro (4) arquivos da fila de prioridade 1; três (3) da fila de prioridade 2 e dois (2) arquivos da fila de prioridade 3;
- . pode-se também levar em consideração o tamanho dos arquivos e imprimir conforme a quantidade de bytes dos mesmos. É claro que deve ser impresso em uma proporção,

de forma que a fila de prioridade 1 tenha maior quantidade de bytes impressos;

- . pode-se também ser levado em consideração a ordem de chegada dos arquivos, além de sua prioridade.

2.4.3. SERVIDORES DE IMPRESSÃO EXISTENTES

Alguns exemplos são destacados nesta seção com o objetivo de enfatizar as funções implementadas nos diversos servidores de impressão discutidos na literatura.

SERVIDOR DE IMPRESSÃO DA REDE LOCAL ITAUTEC

O software servidor de impressão dessa rede, compartilha com o servidor de arquivos a mesma configuração, constituída de microcomputador, disco rígido, disco flexível e impressoras [FER85]. Os arquivos a serem impressos são gravados em disco, com nome diferente, transparente ao usuário, com a parte inicial fixa identificando ser do servidor de impressão; contém o número do nó que solicitou o pedido e a extensão indicando uma sequência numérica gerada pelo próprio servidor desde o início da sessão.

A impressão do arquivo só é feita após a gravação total do mesmo, sendo removido do disco no término da impressão. Os arquivos são colocados na fila e atendidos em ordem de chegada, a não ser que haja alteração da ordem da fila. As operações disponíveis são: listar, remover e trocar de ordem. Caso o disco esteja sem espaço suficiente o usuário recebe a mensagem *SI - SEM ESPAÇO DISPONÍVEL*. A transmissão do arquivo à impressora não prende os servidores, pois é utilizado um "buffer" intermediário, que transmite os dados ao "buffer" da impressora.

SERVIDOR DE IMPRESSÃO DE CAMBRIDGE

Provê a comunidade de usuários com facilidades de impressão e "spooling" dos arquivos, fornecendo certos tipos de formatação de arquivos [WIL80]. Pode receber documentos a serem impressos de vários clientes de forma intercalada, sendo responsável por escalonar suas últimas impressões, executando serviços de forma semelhante ao de um supervisor de um sistema "time-sharing". Foi implementado em um microcomputador com pequena capacidade de memó-

ria RAM, usando então o servidor de arquivos para armazenar os arquivos a serem impressos. A impressora não está conectada diretamente ao servidor, mas sim à rede através de seu próprio microprocessador. A comunicação entre a impressora e o servidor de impressão é feita por um protocolo de fluxo de bytes, usando um mini-pacote independente para cada byte transmitido, sendo a interface entre a impressora e o anel relativamente simples, tornando o esquema mais econômico.

Em 1982, o sistema possuía duas impressoras: uma de linha e outra "*margarida*", com atividades de reimpressão e abandono controladas por botões próprios [NEE82]. O cliente deveria interpretar o código indicativo de restauração, enviado pelo protocolo e repetir ou não o pedido de impressão, conforme suas necessidades.

SERVIDOR DE IMPRESSÃO DA REDE LOCAL GPLOC DO SERPRO

A topologia da rede é barramento, o meio pode ser par trançado ou cabo coaxial e os microcomputadores de 8 bits com no mínimo uma unidade de disco local rígido ou flexível. Na rede podem ser conectados servidores de arquivos, impressoras seriais e paralelas, fitas "*streammer*" para "*back-up*", plotter ou outros equipamentos de interface serial RS-232-C. O controle de comunicação nos níveis superiores é baseado no modelo ISO-OSI [SAN87], sendo que o servidor de impressão foi implementado com a técnica de "*spooling*", incluindo-se características para eliminação de certos problemas com o usuário. O servidor de impressão usa recursos do servidor de arquivos para o armazenamento de seus arquivos, que possuem um nome característico indicativo de pertencer ao servidor de impressão e com informações básicas para ser atendido o pedido (impresso). As impressoras podem estar ou não conectadas ao servidor de impressão, no entanto se for solicitado um pedido de impressão local o cliente deverá estar conectado a uma impressora.

SERVIDOR DE IMPRESSÃO IBM (IBM - ZURICH)

O sistema distribuído da IBM-Zurich é baseado em uma rede local do tipo "*token ring*" e o servidor de impressão controla uma impressora laser IBM 6670. Esse servidor possui dois subsistemas: um de controle da impressora e outro de "*spooling*" executados em máquinas diferentes. Segundo Janson et al [JAN83] a responsabilidade desse servidor é controlar e alimentar a impressora, pois o "*spooler*" além de ser remoto, é uma extensão do servidor de arquivos. Um comando de impressão é enviado ao servidor de arquivos e

resulta no arquivo ser copiado para um diretório especial do serviço de arquivos, denominado "*caixa postal spooler*". A função *colocar na fila* não exige ação particular do servidor de arquivos, porém a função *tirar da fila* apesar de residir no serviço de arquivos, torna o servidor de arquivos totalmente passivo, porque ele não causa a impressão dos arquivos da fila espontaneamente, mas apenas se o servidor de impressão alertá-lo para essa função. O protocolo de transferência do arquivo na rede é orientado a bloco e o de controle com a impressora orientado a caracter. O servidor de impressão não possui armazenamento secundário, porém faz o controle da velocidade de transferência e provê um "*buffer*" para manter a impressora sempre ocupada.

SERVIDOR DE IMPRESSÃO DA REDE AMPLUS

O servidor de impressão dessa rede executa o serviço de "*spooling*", além de gerenciar as filas de impressão [AMP90]. Permite a conexão de até 8 (oito) impressoras, que podem trabalhar no modo de pré-alocação ou de "*spooling*", que pode ser definido apenas pelo super-usuário. O usuário pode utilizar um diretório de rascunho, também definido pelo super-usuário, para manter seus arquivos a serem impressos, caso o servidor esteja *congestionado*. O arquivo é criado com nome diferente, contendo dados do cliente e outros definidos pelo servidor, sendo colocado em uma fila, gerenciada com a técnica *FIFO* para atendimento dos clientes. O super-usuário pode solicitar operações comuns do sistema, além das facilidades de iniciar a execução, encerrar atividades e mudar o modo de atendimento das impressoras. Ao usuário comum é permitido especificar o número de cópias, o tamanho de página, a impressão de identificação do usuário, o tipo de formulário para os serviços de impressão disponíveis ao sistema.

2.5. CONSIDERAÇÕES FINAIS

Neste capítulo tratou-se do embasamento teórico relativo a redes locais, sistemas distribuídos e protocolos de comunicação, além de experiências apresentadas na literatura e que foram obtidas a partir de implementações de sistemas distribuídos e servidores de impressão. No capítulo seguinte serão abordadas as características gerais que deve apresentar um servidor de impressão, objeto central deste trabalho.

CAPÍTULO III

ORGANIZAÇÃO GERAL DO SISTEMA

Neste capítulo são delineadas as características gerais do ambiente físico de execução do servidor de impressão e dos clientes desse sistema computacional. Os elementos responsáveis pelo controle de cada item (a nível de software ou de hardware) integrante do sistema são discutidos. A troca de informações entre os elementos, o projeto global do sistema desenvolvido e o destaque dos assuntos discutidos nos capítulos posteriores são aqui apresentados.

3.1 CONSIDERAÇÕES INICIAIS

O servidor de impressão desenvolvido no LASD permite o compartilhamento dos dispositivos de impressão disponíveis no sistema (periféricos de saída paralela ou serial) diminuindo o custo de utilização e aumentando a disponibilidade dos equipamentos. Esse servidor utiliza a técnica de "spooling", de forma que cada pedido de impressão provoca o armazenamento do arquivo em memória RAM e posteriormente em disco (flexível ou rígido).

Toda fila de impressão é gerenciada pelo próprio servidor de impressão, que da forma como foi implementado não se torna cliente de nenhum outro servidor disponível no sistema. Porém, facilmente poderá ser alterado para se tornar cliente de um servidor de arquivos, ou de algum servidor de nomes. O método utilizado para gerenciamento de cada fila de impressão é com disciplina de prioridade, sendo a prioridade definida pelo próprio sistema (com o apoio de um servidor de nomes, por exemplo) de acordo com o usuário solicitante e o tipo de atendimento do pedido.

O servidor de impressão pode ter diversas filas, uma para cada periférico, caso o sistema tenha vários equipamentos de saída disponíveis aos clientes. Existe uma correspondência biunívoca entre filas e periféricos, mas quando um periférico estiver desocupado poderá auxiliar um outro, desde que pertença à mesma categoria de atendimento.

Essas filas podem ser consideradas pseudo-filas, porque todas permitem a inserção em várias posições, dependendo da prioridade do cliente solicitante, porém a remoção é feita apenas no início. Para que um cliente de mais baixa prioridade não fique eternamente à espera, a posição de inserção de um certo arquivo é escolhida pelo sistema, que leva em consideração além da prioridade do cliente, a última posição de inserção.

Os arquivos devem ser locais, isto é, o usuário deve manter uma cópia do arquivo na estação para fazer o pedido (no entanto pode ser implementado futuramente a impressão de arquivos remotos, com a introdução de uma nova operação no servidor).

3.2 VISÃO GLOBAL DO SERVIDOR DE IMPRESSÃO

O trabalho desenvolvido tem como objetivo principal atender pedidos que venham de diversos clientes, sem que haja necessidade de se alterar os seus ambientes de execução. Considerando SO_i como sendo sistemas operacionais diferentes para valores de i entre 1 e n , a figura 3.1 mostra o tipo de atendimento que o servidor de impressão deve fazer.

figura 3.1 O servidor e os clientes do sistema

Cada cliente permanece em um ambiente que depende das necessidades do usuário e pode solicitar os serviços prestados pelo servidor de impressão disponível no sistema computacional distribuído. A comunicação entre os clientes e o servidor de impressão está definida em um protocolo de comunicação que utiliza o mecanismo de chamadas de procedimentos remotos. Esse protocolo e detalhes dos pacotes trocados entre os elementos do sistema computacional distribuído estão apresentados no capítulo IV.

No servidor de impressão existe um software cujas funções principais são:

- . receber pedidos dos diversos clientes;
- . atender as solicitações;
- . devolver uma resposta.

Além de realizar essas atividades, o servidor de impressão gerencia os k sistemas de impressão que estão diretamente conectados ao equipamento que executa esse software. A figura 3.2 mostra as conexões através de interfaces seriais ou paralelas, que ficam sob a responsabilidade desse servidor.

figura 3.2 O servidor e os sistemas de impressão

Da forma como o servidor de impressão foi implementado, a configuração mínima necessária para sua instalação é composta de:

- . um equipamento *PC* (*Personal Computer*), com capacidade de armazenamento médio, com no mínimo um disco flexível para fazer o armazenamento dos arquivos a serem impressos;
- . diversas portas seriais e/ou paralelas para conexão de impressoras matriciais, de linha, laser, "*plotter*", etc.

Os periféricos de saída paralela ou serial devem estar acoplados ao equipamento que está executando o servidor de impressão, pois não foi definido um protocolo que fizesse a comunicação entre o servidor e os periféricos (embora isto seja uma possibilidade).

Esse equipamento pode ser compartilhado com outros servidores, porém requer algumas alterações em alguns módulos do servidor de impressão evitando-se problemas de inconsistências. Como exemplo é citado o compartilhamento com o servidor de arquivos que implica em alterar o nome do arquivo a ser impresso, pois após a colocação na fila não é permitido atualizações em arquivos, além disso após a impressão o mesmo é apagado da memória, operação não obrigatória em um servidor de arquivos.

Esse servidor não depende da adoção de outros servidores, pois inicialmente é carregado um arquivo que contém todas as especificações necessárias para a sua execução. No entanto, caso o sistema disponha de um servidor de nomes, a prioridade de cada cliente pode ser registrado nesse servidor. No servidor de nomes pode ser adicionado também os direitos de acesso de cada cliente, pois isso é usado como técnica de segurança, impedindo que elementos não autorizados tenham acesso aos serviços e/ou periféricos disponíveis no sistema. Também pode ser acrescentado um servidor de "*binding*", que deverá registrar o endereço do servidor de impressão e os serviços disponíveis para que os clientes façam a *importação* dos mesmos.

O sistema operacional que deve estar em execução no equipamento que vai hospedar o servidor de impressão é o MS/DOS, a figura 3.3 mostra as camadas de software que devem compor o sistema e como deve ser a configuração física desse ambiente em relação aos periféricos que atenderão as solicitações de impressão de arquivos.

Na primeira camada *SERVIMP.EXE* é feito o atendimento aos diversos pedidos que chegam (de forma intercalada) de todos os clientes do sistema; para isso faz a recepção de todos os pacotes, interpreta a operação solicitada e determina o atendimento de todos

os serviços requisitados.

O *PACKET DRIVER* é um software responsável por distanciar o programador de certos detalhes de hardware e do canal de comunicação, provendo uma interface comum e simples. Esse utilitário permite que aplicações múltiplas compartilhem uma mesma interface de rede a nível de enlace de dados, permitindo que implementações de protocolos sejam independentes de certas características do equipamento, onde está sendo executado [PAC89].

Servidor

Periféricos

figura 3.3 Ambiente físico do servidor de impressão

Pode-se notar que nesse ambiente podem estar conectados até 8 (oito) periféricos, limite definido no sistema e que pode ser facilmente incrementado. No entanto, o software servidor de impressão pode gerenciar o uso de um ou mais periféricos de saída, restringido o referido número a oito, o que é visto com maiores detalhes no capítulo V.

3.3 ARQUITETURA DO SERVIDOR DE IMPRESSÃO

Além de controlar os periféricos de saída paralela ou serial, o servidor de impressão desenvolvido faz o gerenciamento das filas de impressão utilizando a técnica "spooling", ocorrendo os seguintes passos quando um usuário solicita uma impressão:

- . o pedido é enviado diretamente ao servidor de impressão;
- . o servidor de impressão determina o armazenamento dos arquivos a serem impressos, inicialmente em RAM e após o recebimento de todos os pacotes que compõem esse arquivo, em um disco;
- . o servidor faz a escolha de uma fila adequada, de acordo com o tipo de atendimento da fila, os direitos do cliente e o arquivo enviado;
- . o servidor escolhe uma posição adequada para esse arquivo na fila selecionada, levando em consideração a prioridade do cliente e a última posição de inserção na fila.

A figura 3.4 mostra os três subsistemas do servidor de impressão, englobando o controle e alimentação dos sistemas de impressão disponíveis; o "spooler" e o controle dos pedidos de serviços.

figura 3.4 O Servidor de impressão e seus subsistemas

Este servidor foi desenvolvido para prestar diversos tipos de serviços para todos os clientes autorizados dos sistemas operacionais, sendo os pedidos solicitados por uma *interface* amigável dispondo dos seguintes serviços:

- . a impressão de um determinado número de cópias de um certo arquivo, sendo que esse número foi inicialmente limitado em 10 (dez), mas como a flexibilidade do sistema é grande, ele pode ser facilmente incrementado. O arquivo deve ser local, isto é, o usuário deve fazer uma cópia do mesmo para a sua estação, antes de solicitar o pedido;
- . listagem dos arquivos de um determinado cliente, que estão na(s) fila(s) de impressão, fornecendo o número da fila, a posição do arquivo na fila, o nome do arquivo e "status" indicando se está em impressão;
- . alteração do número de cópias de impressão;
- . troca da ordem (posição) de dois arquivos quaisquer, cujos pedidos de impressão tenham sido feitos pelo mesmo usuário;
- . cancelamento do pedido de impressão de um arquivo que esteja na fila de *espera*.

Para a execução dessas atividades o servidor realiza as seguintes tarefas:

- . coloca um arquivo na fila, mediante um pedido do cliente;
- . armazena o arquivo a ser impresso em um disco;
- . obtém um arquivo da fila, para enviar a um "buffer" intermediário e posteriormente ao "buffer" da impressora quando disponível;
- . remove um arquivo que já foi impresso (retira da fila e apaga da memória), caso já tenha terminado o serviço, ou se o cliente tiver pedido o seu cancelamento.

Como o sistema dispõe de vários periféricos de saída, esse é escolhido de forma eficiente, impedindo que qualquer um tenha tempo ocioso de aproveitamento, a não ser que pertença a uma classe especial de atendimento, como um *plotter* ou uma impressora *laser*. Esses detalhes ficam definidos ao se colocar em execução o servidor de impressão, que permite a configuração, início e término de exe-

cução desse sistema, também através de um *menú* de opções. Quando o servidor é colocado em execução pela primeira vez o sistema deverá ser configurado, antes de ser executado. No entanto, a opção *configurar* poderá ser escolhida sempre que for alterada as características físicas dos periféricos de saída e deverá ser informado ao sistema:

- . o *número de periféricos* limitado pelo sistema em 8 (oito), que não precisam estar todos conectados ao servidor, pois este se incumba de fazer a verificação antes de colocar cada periférico em disponibilidade aos usuários (isto quer dizer que podem ser definidas várias saídas, mas apenas algumas estarem disponíveis);
- . para cada periférico definido o sistema já cria uma fila correspondente que receberá um número inteiro, meio como será referenciada internamente;
- . o *nome da fila* que deve conter no máximo 8 (oito) caracteres alfanuméricos;
- . a *classe do periférico* que se refere ao tipo de atendimento que o periférico poderá fazer ao sistema, sendo definido como *geral* (caso possa atender clientes em geral) ou *específico* (caso atenda apenas clientes especiais). Na classe especial devem ser enquadrados apenas periféricos especiais como impressoras *laser* ou *plotters* de alta resolução, pois para que um arquivo seja enviado para a sua fila de impressão, o usuário solicitante deverá ter permissão e o arquivo ter uma extensão particular definida na instalação do sistema e que pode ser alterada caso ocorra problema de segurança;
- . a *extensão padrão* que será solicitada caso a classe de atendimento do periférico seja especial;
- . o *tipo de saída* do periférico que deve ser *serial* ou *paralela*, sendo verificada no início da execução do sistema a sua veracidade, para não prejudicar o atendimento aos clientes;
- . os *status* da fila e do periférico são atribuídos pelo sistema, sendo exibido no momento que o servidor é iniciado. Para a fila o *status* inicial pode ser: *vazia* (se estiver pronta para receber arquivos a serem impressos) ou *fechada* (não pode receber solicitações pois corresponde a um periférico não disponível ao sistema). Para o periférico o *status* pode assumir os valores: *ativo* ou *inativo* dependendo de estar ou não conectado ao servidor, ou estar ou não ligado.

A configuração definida na instalação é armazenada em um arquivo utilizado posteriormente na execução do servidor, devendo ser reconfigurado sempre que forem alterados os periféricos de saída, ou que se queira mudar nomes de fila e/ou extensões de arquivos, por precaução.

O servidor de impressão ao ser colocado em execução faz a leitura de um arquivo que atribui valores a todas as variáveis que simulam as atividades do servidor de nomes e outras inerentes ao próprio servidor de impressão. Em seguida procura no disco o arquivo de configuração do sistema, caso não encontre acusa erro e não prossegue as tarefas. Porém, se encontrar o referido arquivo faz uma exibição na tela do número, nome, classe, saída e status de cada fila e por medida de segurança aparece em extensão apenas 3 (três) asteriscos. Em *status do periférico* será exibido *NÃO SELECIONADO* quando o periférico não estiver conectado ao equipamento, ou caso esteja desligado; *SELECIONADO* se o periférico estiver em linha.

O software servidor de impressão para recebimento de pacotes e impressão de arquivos, entra em um "loop" que só termina ao se pressionar a tecla "@", determinando o retorno ao *menú* de opções que possibilita a determinação do término de execução desse software. A recepção de pacotes é feita através de uma interrupção de hardware e para a impressão dos arquivos é analisada o tipo de saída, pois se for serial é informado ao sistema como devem aparecer os dados enviados para essa interface. O *status* de um arquivo é armazenado pelo sistema para informar ao cliente sempre que esse solicitar, sendo assumido os valores: *em impressão*; *último a chegar*; *não especial*, onde apenas a primeira situação impede que o arquivo seja obtido para impressão por um outro periférico.

3.4. CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma visão global e a proposta de arquitetura do servidor de impressão desenvolvido, tendo sido destacado o ambiente, as funções, configuração do sistema, controle de periféricos e serviços prestados. No próximo capítulo serão discutidas as características do protocolo desse servidor, salientando-se as primitivas e o mecanismo de comunicação.

CAPÍTULO IV

O PROTOCOLO DO SERVIDOR DE IMPRESSÃO

Neste capítulo são identificadas as funções executadas pelo protocolo do servidor de impressão e as primitivas de comunicação definidas para esse intercâmbio. O mecanismo utilizado no protocolo do servidor para a comunicação a nível de pacotes é discutido e os motivos que levaram a escolha da semântica adotada no desenvolvimento são apresentados.

4.1 CONSIDERAÇÕES INICIAIS

Os protocolos de comunicação executam diversas funções incluindo transferência de dados, controle de fluxo e gerenciamento de conexão [LIN91]. Normalmente são definidos para gerenciar a troca de mensagens através de módulos de software de comunicação, tornando-se parte indispensável desses módulos [KAK91].

O *PSI* (Protocolo do Servidor de Impressão) do *LASD* (Laboratório de Sistemas Digitais) do *ICMSC - USP* foi desenvolvido com base no mecanismo de comunicação *RPC*, adotando-se a semântica *no máximo uma vez*. Essa semântica é considerada adequada pois medidas de desempenho indicam que *RPCs* implementados com essa semântica têm um custo menor (no máximo igual) do que aqueles que não garantem esse tipo de execução [LIS90].

A chamada de procedimento remoto constitui, na prática, em um software responsável pela comunicação a nível de pacote, com o objetivo de enviar um pacote a ser transmitido ao servidor, aguardar uma resposta e então recebê-la [BIR84]. Para isso, recebe um "*buffer*" de dados do "*stub*" do cliente, acrescenta informações de controle e passa para uma outra camada de nível inferior para se fazer a transmissão. Essa técnica leva a propriedades consideradas importantes no conceito de arquitetura em camadas: a remoção ou inclusão de camadas é facilitada e formas de implementações podem ser escolhidas [SCH88].

As informações de controle, conhecidas por cabeçalho do "*buffer*" de informação recebido da camada superior, deve ser usado pela camada de recebimento de mesmo nível na arquitetura [SCH88]. Assim como o *RPC* do cliente não vê as unidades de dados recebidas da camada superior, o *RPC* do servidor usará apenas o cabeçalho acrescentado ao "*buffer*" de informações, pela camada equivalente do cliente.

Para a comunicação entre cliente e servidor são definidas primitivas, tanto no cliente como no servidor. Apesar de existirem várias técnicas de implementação dessas primitivas [COU88], para o *PSI* descrito neste capítulo elas foram definidas em número de três: uma no cliente e duas no servidor. A primitiva de comunicação definida no cliente tem a função de determinar a transmissão do pacote/requisição n vezes, caso não ocorra problema de transmissão e o servidor não receba o pedido. Uma das primitivas do servidor entra em um laço para recebimento dos pacotes, saindo apenas quando for solicitado o término de execução do software, pois tem a função de receber solicitações enquanto estiver em execução, podendo esses pedidos chegarem de forma intercalada. A outra primitiva do

servidor passa o pacote/resposta *PSI* para a camada inferior, para que essa possa incluir as especificações de um pacote *PKT* e transmití-lo ao cliente.

4.2 CARACTERÍSTICAS GERAIS

O PSI tem a função básica de transferir dados entre os clientes de diversos sistemas computacionais distribuídos e o servidor de impressão desenvolvido, gerenciando esse intercâmbio de forma segura e precisa, adequando os referidos dados e estabelecendo uma comunicação livre de erros.

O sistema mantém uma tabela de usuários, capaz de armazenar a resposta da última solicitação feita por um cliente que será utilizada caso ocorra falha em um dos elementos (cliente, rede ou servidor). Com isso pretende-se que o sistema desenvolvido atenda a uma das exigências básicas de um sistema distribuído que é a segurança, liberando o usuário de certos detalhes de gerenciamento.

Pode-se salientar o esquema clientes e servidor através da figura 4.1 que fornece uma visão global desse contexto. Para atingir esses propósitos a mensagem enviada pelo cliente ao servidor contém:

- . tipo do pacote, que no caso é uma solicitação identificando o servidor em questão, pois o servidor está orientado a desprezar um pacote que não seja uma requisição, ou que não lhe pertença, independente de conter o seu endereço (observar que o servidor de impressão pode compartilhar o mesmo equipamento com outro servidor; um pacote pode ter seu endereço, mas poderá pertencer ao outro servidor);
- . a identificação da sessão solicitante, pois no cliente pode existir uma sessão para cada usuário (no caso de sistemas multi-usuários);
- . a identificação da operação a ser executada no servidor, pois esse dispõe de vários tipos de serviços a serem oferecidos aos clientes do sistema;
- . o número de seqüência de chamada, para controle de falhas que possam ocorrer no sistema;
- . o tamanho do pacote, isto é, tamanho dos parâmetros e/ou dados necessários para o atendimento do serviço solicitado;

- . parâmetros e/ou dados, que podem variar de 39 a 1493 bytes.

Por outro lado, a mensagem retornada pelo servidor ao cliente contém:

- . o tipo de pacote, que no caso constitui-se de uma resposta *PSI*, sendo o único aceito pelo cliente;
- . a identificação do processo, pois no servidor podem existir vários processos;
- . a operação executada pelo servidor, para controle de problemas em caso de falhas;
- . o estado da resposta, que pode apontar erro na execução da operação solicitada; não aceitação da requisição; não execução da operação pois o cliente não possui permissão para solicitá-la; a não existência da sessão; ou fim do "time-out" na execução da operação;
- . o tamanho do pacote, ou seja tamanho da resposta;
- . os dados/resposta que variam de 39 a 1493 bytes.

figura 4.1 Visão cliente/servidor

A figura 4.2 mostra as mensagens *PSI* trocadas entre um cliente de um sistema computacional distribuído e o servidor de impressão.

cliente para o servidor

1	2	1	1	2	39 ... 1493 "bytes"
ti po	ses são	op	se q.	tam	dados e/ou parametros

servidor para o cliente

1	2	1	1	2	39 ... 1493 "bytes"
ti po	proc	op	es ta do	tam	dados/erro ou codigo erro

figura 4.2 Mensagens PSI

Considerando que acima do protocolo de pacote básico, existe um protocolo de *RPC* que adota o sistema *datagrama*, no protocolo *PSI* são inseridos os endereços destino, origem e tipo de pacote que seguem os padrões do protocolo *Ethernet* [COU88], isto é seis bytes para cada endereço e dois para especificação do tipo de pacote, sendo que as quantidades mínima e máxima de bytes enviadas do cliente ao servidor e vice-versa, também ficam definidas por esse protocolo de comunicação.

O serviço de transporte do protocolo *PSI* está definido em uma camada de software, que tanto no cliente como no servidor faz o manuseio dos pacotes *PSI* e *PKT*, sendo que o primeiro tipo de pacote mantém parâmetros específicos do servidor de impressão e o outro possui especificações do pacote *Blue Book Ethernet* [PAC89]. A função básica de cada camada é receber e acrescentar (ou retirar) cabeçalhos de pacotes, conferir, alterar o tipo dos mesmos e transmitir para a camada imediatamente inferior (ou superior). A figura 4.3 mostra o que ocorre com os pacotes *PSI* e *PKT*, tanto na transmissão como na recepção, considerando que na transmissão é recebido um pacote *PSI*, enquanto que na recepção o pacote fornecido é do tipo *PKT*.

figura 4.3 Manuseio dos pacotes tipo PSI e PKT

Para a realização dessas atividades existem controles, pois um protocolo deve ser implementado formando camadas hierárquicas, de modo que cada camada controle a inferior e passe dados para a superior. A redução do número de camadas ocorre com o objetivo de se preservar o desempenho do sistema, levando-se em consideração a taxa mínima de erros existente em redes locais, além de facilitar a implementação, testes e adaptação a outros sistemas. Cada camada é subdividida em módulos com operações e/ou parâmetros para controle de execução, codificação e formatação.

A figura 4.4 faz uma analogia entre as camadas do modelo OSI e as camadas do protocolo PSI.

modelo OSI		protocolo PSI
APLICAÇÃO	←————→	SERVIÇOS DE IMPRESSÃO
APRESENTAÇÃO	←————→	RPC (pacote PSI)
SESSÃO		
TRANSPORTE	←————→	TRANSPORTE (pacote PKT)
REDE		
ENLACE	←————→	Packet Driver
FÍSICA		+ Interface (física)

figura 4.4 Correspondência entre as camadas do modelo OSI e do protocolo PSI

Na camada de *serviços de aplicação* tem-se a requisição do serviço (por parte do cliente) solicitada através de um menú de opções ou a execução da atividade (por parte do servidor). Na camada inferior (*RPC*) existe o manuseio de mensagens *PSI*, para controle de serviços e manipulação de arquivos (entre outras atividades dependendo do serviço requerido). A camada denominada *TRANSPORTE* faz o manuseio de mensagens *PKT*, onde são colocados os endereços para a comunicação entre dois computadores quaisquer da rede, além de ser responsável pelo controle de erros, fragmentação de mensagens e armazenamento. O nível físico contém as interfaces de comunicação, e um software denominado "*Packet Driver*", permitindo que aplicações múltiplas compartilhem a mesma interface de rede a nível de enlace de dados, tornando as implementações de protocolos independentes de certos detalhes de hardware. Nessa última camada não são tratados os pacotes duplicados (o que deve ser feito pela camada superior), porém descarta pacotes *danificados*.

Para que ocorra a troca de mensagens entre os elementos do sistema, além da padronização dos parâmetros definida nos protocolos, deve aparecer uma camada de transporte para conversão e controle dos pacotes que circulam pela rede e um outro software com a função de receber o pedido/resposta, verificando se o mesmo contém as especificações pré-determinadas pelo sistema. E em caso positivo o "*buffer*" de dados é retirado e o código de operação é solicitado, ambas informações são passadas para uma outra camada capaz de retirar parâmetros e determinar a execução/interpretação da operação. Em caso negativo o pedido é ignorado, ou então um estado indicativo de erro é enviado impedindo a passagem do pedido para outra camada, dinamizando. Essa camada de software forma o *RPC* do servidor, constituindo o mecanismo de comunicação do sistema.

4.3 AS PRIMITIVAS DE COMUNICAÇÃO

O servidor de impressão presta diferentes tipos de serviços aos clientes dos sistemas computacionais distribuídos, serviços esses solicitados através de operações distintas, selecionadas pelos usuários por meio de um menú de opções, definido em um módulo denominado *interface do cliente*, para tornar o sistema transparente ao usuário. No entanto, o mecanismo *RPC* para fazer a comunicação cliente/servidor necessita de primitivas de comunicação definidas no cliente e no servidor, que determinam o intercâmbio entre os elementos do sistema.

A comunicação é realizada de forma que o cliente faz a solicitação, fica bloqueado enquanto o servidor recebe o pedido, executa-o e devolve uma resposta. Fica assim definido para o cliente

um *tempo de espera* pela resposta (evitando que o cliente espere eternamente, impedindo a continuidade de outras execuções em caso de ocorrências de falhas na rede ou no servidor). Após o término desse *tempo de espera*, caso o cliente não tenha recebido a resposta do servidor, repete-se a solicitação até que o servidor devolva uma resposta, permanecendo nesse ciclo um número de vezes definido pelo sistema; ao fim desse tempo, informa ao cliente que o servidor não está disponível.

Para a execução do mecanismo *RPC* são definidas três primitivas de comunicação, que no lado do cliente tem a função de detectar e retransmitir mensagens possivelmente perdidas e no lado do servidor deve detectar mensagens repetidas para não executá-las novamente. Essas primitivas são:

ExecutarOperaçãoRemota(cabeçalho_requisição, pacote_PSI)

- . implementada no cliente com o cabeçalho contendo: a identificação de uma requisição, a operação solicitada, os parâmetros determinados para a execução da operação e os dados se necessários, preparados em um ou mais pacotes denominados *PSI*. Esta primitiva está definida para fazer 5 (cinco) tentativas e no caso de falhas emite uma mensagem ao usuário indicando que o *servidor não responde*;

Receber Pedido()

- . implementada no servidor com o objetivo de receber o pacote/pedido do cliente, verificar, aceitar ou não o pacote e quando aceito passar a uma outra camada que permite a execução do serviço solicitado. Caso haja erro ao analisar o pedido, a resposta é automaticamente enviada ao cliente;

EnviarResposta(cabeçalho_resposta, pacote)

- . implementada no servidor com a função de preparar o pacote resposta com as especificações definidas pelo sistema.

4.4 O MECANISMO DE COMUNICAÇÃO

Para o atendimento do pedido de um cliente solicitando um determinado serviço disponível no servidor de impressão, o usuário, através da *interface*, faz a opção de uma operação e fornece seus parâmetros. Esses dados são transferidos para o "*stub*" da estação, cujo objetivo principal é prepará-los em um "*buffer*", que será passado à camada de software implementando o *RPC* da estação (com a função de adequar esse *buffer* aos padrões do *PSI*). A camada de *RPC* da estação fica acima de uma camada de transporte, sendo a comunicação feita de forma sincronizada, ou seja, ocorrendo o bloqueio da estação para aguardar a resposta [BIR84]. A figura 4.5 esquematiza as camadas de software na estação e no servidor, que estão acima do transporte de mensagens, considerando que cada camada pode detectar erros e devolver os dados para a anterior, impedindo a continuidade do fluxo de dados que não sejam convenientes.

figura 4.5 Camadas de software

Esse mecanismo de comunicação na forma em que foi implementado é semanticamente transparente, isolando detalhes de gerenciamento de comunicação e permitindo que novos serviços sejam acrescentados no sistema. A semântica adotada foi a de *no máximo uma vez*, adequada para os serviços prestados por essa classe de servidores. Analisando-se os serviços prestados por um sistema computacional distribuído, pode-se notar que determinadas operações podem afetar a integridade do sistema, isto é, normalmente o cliente não recebendo a resposta do servidor em um determinado intervalo de tempo, retransmite o mesmo pedido, o que poderá

provocar a execução da mesma operação mais de uma vez. Porém, o não recebimento da resposta pode indicar que o servidor não executou o serviço, ou que houve falha no meio de transmissão durante o envio da resposta. Repetir a solicitação pode implicar em problemas de integridade para o sistema, ou desperdício de tempo de execução no servidor, por repetir a mesma atividade.

Assim, para se contornar esse possível problema, cada requisição possui um número de sequência de chamada, complementada no servidor a cada operação realizada e no cliente após o recebimento de cada resposta. Antes de atender cada requisição no servidor é feita a verificação dessa sequência, que no caso de não ser a esperada impedirá o servidor de executar a operação solicitada, devolvendo a resposta da última requisição enviada pelo cliente. As figuras 4.6.a, b e c mostram todas as falhas possíveis na estação, na rede e no servidor, respectivamente, com as ocorrências no cliente e/ou servidor, esclarecendo a forma de implementação desse mecanismo.

No próximo capítulo (V) serão detalhados os serviços prestados pelo servidor de impressão, esclarecendo que esta semântica é a mais adequada para este tipo de sistema, pois não se justifica a adoção de uma implementação mais sofisticada se os serviços não exigem tal sofisticação, o que ocorreria seria um prejuízo no desempenho do sistema.

figura 4.6.a Falha na estação

figura 4.6.b Falha na rede

figura 4.6.c Falha no servidor

É bom ressaltar que se o pedido for de impressão o arquivo pode ser apenas colocado na fila e não ser impresso, pois a falha pode ocorrer entre estes dois instantes.

4.5. CONSIDERAÇÕES FINAIS

Este capítulo tratou das funções essenciais do protocolo do servidor de impressão, evidenciando: as mensagens trocadas entre clientes e servidor, a correlação entre camadas do modelo OSI e do referido protocolo e forma e técnica de comunicação utilizadas. O capítulo seguinte detalha aspectos da implementação do software servidor de impressão.

CAPÍTULO V

O SERVIDOR DE IMPRESSÃO

Neste capítulo é apresentada a organização do servidor desenvolvido, mostrando as estruturas de dados utilizadas na implementação do sistema e a estrutura modular do software em questão.

5.1 CONSIDERAÇÕES INICIAIS

Este capítulo discute o servidor de impressão desenvolvido dando ênfase à organização interna adotada, bem como aos detalhes de interfaceamento com o módulo de comunicação.

Em linhas gerais, o desenvolvimento deste servidor foi baseado em uma estrutura hierárquica e modular, com o objetivo de se ter um desenvolvimento organizado, facilitando a fase de depuração e posterior manutenção. Em adição, a metodologia empregada favorece os aspectos de independência dos detalhes do meio físico e sistemas operacionais, facilitando o transporte do servidor a outros ambientes.

5.2 ESTRUTURA MODULAR DO SERVIDOR DE IMPRESSÃO

Nesta seção são apresentadas as estruturas de dados necessárias ao servidor de impressão.

5.2.1 ESTRUTURAS DE DADOS

Serão detalhadas inicialmente as estruturas de dados sobre as quais são executadas as atividades no servidor de impressão e a descrição dos pacotes utilizados na comunicação.

. TABELA DE USUÁRIO (*Tu*)

A tabela de usuário é mantida no servidor para controle do próprio sistema, sendo indexada pelo código do usuário e armazena:

- . *nome do usuário*: que poderá ser utilizado na identificação de formulários impressos, ou outra necessidade futura; atualmente é mantido para o controle total do sistema;
- . *senha*: fornecida pelo usuário como medida de segurança do sistema global;
- . *permissão*: que mantém os direitos de acesso do cliente aos serviços disponíveis no sistema, completando todo o esquema de segurança disponível aos diversos usuários;
- . *extensão*: contendo 1024 caracteres com o objetivo de manter o histórico do cliente, ou seja as últimas ocorrências do mesmo no acesso ao sistema.

TABELA DE SESSÃO (*Ts*)

Também é mantida no servidor uma tabela de todas as sessões que foram abertas, de acordo com os usuários registrados nesse sistema, indexadas pelo identificador da sessão, contendo as informações:

- . *código do usuário*: que faz a identificação do usuário, sendo zerado no início da execução do servidor, para evitar problemas, pois isso implica na não existência da sessão;

- . *endereço da estação*: onde o usuário está instalado para solicitar os serviços desse servidor, pois o método adotado é o *datagrama*, assim todo pacote deverá conter os endereços origem e destino;
- . *identificador do processo*: necessário para devolução de respostas ao cliente;
- . *número de seqüência*: identifica a requisição de cada operação para controle do sistema, implementando a seqüência definida no protocolo de comunicação;
- . *tamanho*: define o tamanho máximo da última resposta enviada para cada operação;
- . *resposta*: armazena a resposta da última solicitação de cada operação, sendo esta variável definida como um ponteiro, pois o objetivo é manter informações que serão copiadas para envio ao cliente, evitando novas execuções em caso de falhas.

TABELA PARA O PEDIDO DE IMPRESSÃO (Tb)

O cliente ao solicitar a impressão de um determinado arquivo local, informa o nome do arquivo e o número de cópias (dados que deverão ser transmitidos ao servidor). Dependendo do tamanho do arquivo, o mesmo deverá ser subdividido em pacotes (pois o número máximo de caracteres que a estação poderá enviar ao servidor e vice-versa é 1493 bytes). Esses pacotes serão enviados *um a um* e o servidor deverá ter o controle deles e só colocar o pedido em uma fila de impressão após o recebimento de todos os pacotes, para montar novamente o arquivo. Para gerenciar essa atividade é mantido nessa tabela, indexada pelo código do usuário, os seguintes dados:

- . *nome*: nome do arquivo a ser impresso;
- . *número de cópias*: define cópias de impressão solicitadas pelo usuário;

- . *tamanho*: quantidade de caracteres do arquivo que deverá ser transmitido (isso é fornecido pela estação do usuário, auxiliando no cálculo do número total de pacotes a serem recebidos);
- . *total de pacotes*: que deverão chegar ao servidor para completar o arquivo a ser colocado em uma fila de impressão;
- . *pacote*: se referencia ao número do último pacote recebido do arquivo em questão.

Para completar o controle de todos esses pacotes, existe uma troca de informações entre cliente e servidor segundo o esquema mostrado na figura 5.1. Após o usuário solicitar o pedido de impressão ele fornece ao sistema os seguintes parâmetros: nome do arquivo e número de cópias. Com esses dados o sistema verifica se o arquivo é local e em caso positivo obtém o seu tamanho, caso contrário rejeita o pedido. Os parâmetros e os dados do arquivo devem ser colocados no "buffer" da mensagem/pedido, pois o código da operação e o tamanho desse "buffer" estão incluídos no "header". Obviamente, precisa ser acrescentado um outro dado que corresponde ao tamanho dos parâmetros, pois é um valor variável de acordo com o tamanho do nome do arquivo.

figura 5.1 Cliente envia ao servidor solicitando uma impressão de arquivo.

Quando a solicitação de impressão do arquivo chega ao servidor, inicialmente verifica-se a existência ou não do referido arquivo naquele ambiente. Em caso positivo são verificados os pacotes que já foram recebidos (dado mantido na tabela provisória até que o arquivo esteja completo) isto é, para ser enviado à fila de impressão o total de pacotes esperado tem que ser igual ao número do último pacote recebido. No caso do servidor ter recebido todos os pacotes esperados (o que pode ocorrer se no cliente ou na rede houve falha durante a transferência de informações) será

enviado um código indicando *ARQUIVO JÁ ESTÁ NA FILA*. Se o pacote recebido é o último, o arquivo será colocado em uma fila de impressão e o cliente receberá um código indicando: *ARQUIVO COLOCADO NA FILA*.

Quando a mensagem for uma solicitação inicial, ou seja, o arquivo não existe no servidor, faz-se a verificação da:

- . disponibilidade de memória; em caso negativo envia ao cliente: *SEM MEMÓRIA DISPONÍVEL*;
- . disponibilidade na tabela do referido usuário, pois pode ocorrer problema da fila estar cheia, informando ao cliente: *SEM POSIÇÃO NA FILA*;
- . possibilidade de criar o arquivo reservando o tamanho fornecido como parâmetro, podendo devolver ao cliente um código que indique: *ARQUIVO NÃO PODE SER CRIADO*;

e em seguida:

- . registra os parâmetros na tabela provisória;
- . copia os dados para o arquivo e se ocorrer problema devolve código de erro na escrita ao cliente.

Os parâmetros de comunicação devolvidos pelo servidor ao cliente, no caso de solicitação da impressão de um arquivo, são mostrados na figura 5.2.a e em caso de devolução de código de erro ou término da operação é devolvido um pacote do tipo exibido na figura 5.2.b, sendo que nesse último caso o tamanho do pacote é sempre definido com o valor 2 (dois) e reconhecido de imediato pela estação.

figura 5.2.a Servidor envia resposta ao cliente

figura 5.2.b Servidor envia ao cliente um código

TABELA DA FILA DE IMPRESSÃO DE CADA PERIFÉRICO (F1)

Para cada periférico existe uma fila associada, obviamente indexada pelo número interno relacionado ao periférico, que recebe inserção em qualquer posição e remoção apenas no início, podendo ser vista como um conjunto de filas, subdividida de acordo com a prioridade. Ocorre que se vista dessa maneira o atendimento deveria ser controlado de outra forma e não como foi implementado, onde é levado em consideração a ordem de chegada além da prioridade. Em cada fila são mantidas as seguintes informações:

- . *status* da fila, podendo assumir os valores: *VAZIA* (à espera de arquivos); *N* número de arquivos a serem impressos, valor esse incrementado toda vez que chega um pedido à fila e decrementado quando o pedido é atendido, não podendo nunca passar do limite definido pelo sistema; *FECHADA* (o seu periférico correspondente não está disponível);
- . *código do usuário* que solicitou o pedido, mantido para controle e atendimento das outras atividades disponíveis;
- . *nome do arquivo* meio de se referenciar ao mesmo;
- . *número de cópias* de impressão do arquivo, sendo definida pelo usuário no ato do pedido;
- . *status do arquivo* indicando se o mesmo está em impressão ou pronto para ser impresso.

TABELA DE PERIFÉRICOS (*Perif*)

Ao se instalar o servidor de impressão são solicitadas informações à respeito de cada periférico que está ou estará disponível aos clientes desse servidor. Essas informações são gravadas em um arquivo de configuração do sistema, através de uma estrutura de dados que mantém:

- . *número* do periférico, forma de controle interno;
- . *nome* do periférico, com no máximo 8 (oito) caracteres;
- . *classe* de atendimento do dispositivo, que pode ser Geral ou Específico;
- . *extensão* que deve conter o arquivo a ser atendido por aquele periférico, caso possua classe de atendimento específico;
- . *interface* do dispositivo, que pode ser: Serial ou Paralela;

Por outro lado, a estação de trabalho deverá manter informações para que se complete a troca de mensagens e seja atendida de forma adequada às solicitações de serviços que serão executados pelo servidor de impressão.

TABELA DE SERVIDOR (*Ts*)

Essa estrutura de dados foi utilizada na implementação do software do sistema de impressão, estando diretamente relacionada à operação básica do sistema, indexada pelo identificador do servidor e mantendo:

- . *endereço* do servidor de impressão;
- . *identificador* da sessão do usuário;

- . *seqüência* de requisição para cada operação disponível no sistema;
- . *time-out* que define o tempo de espera para cada operação.

5.2.2 ESTRUTURA MODULAR DO SOFTWARE DO SERVIDOR DE IMPRESSÃO

O servidor de impressão deve realizar diversos tipos de tarefas que podem ser solicitadas, sendo que essas tarefas devem ficar em uma fila com atendimento baseado em uma prioridade pré-estabelecida pelo sistema. O software foi organizado em vários módulos, com funções distintas e bem definidas para facilitar qualquer tipo de alteração que possa ser necessária. As fases de projeto e implementação do software do sistema servidor de impressão foram baseadas no projeto estruturado de sistemas [WAR85], que tem como ferramenta o diagrama de estrutura modular.

No servidor são definidas várias camadas de software, assim como no cliente, sendo que cada camada possui funções equivalentes às funções da respectiva camada do cliente. Acima do *Sistema Operacional* surge o mecanismo de acesso ao meio físico da rede local que provê funções de transmissão de dados. Os sistemas que utilizam o mecanismo são normalmente incompatíveis entre si [TRI91], devendo ser implementada alguma técnica de multiplexação, ou utilização do endereço físico da estação e do servidor sendo o "*Packet Driver*" um mecanismo dessa natureza, que constitui um módulo do conjunto de protocolos *TCP/IP* [CLA89,FTP89]. Esse mecanismo possibilita uma interface padrão com as camadas superiores independente do meio físico, permite a execução de aplicações múltiplas de forma simultânea, além de descartar pacotes que não lhe são destinados.

Para controle na recepção de pacotes, o *Packet Driver* possui uma função que associa um identificador ao tipo de pacote e uma rotina para a atividade de recebimento específica para esse pacote. As funções disponíveis nesse mecanismo estão no apêndice A1, indicando que sempre que surgir um pacote validado pelo sistema, uma interrupção de hardware é gerada, através de uma rotina de recepção e tratamento de pacotes (nesse caso denominada *ReceberPKT* que poderá aceitar ou descartar o pacote de acordo com os dados obtidos na análise do referido pacote).

A interface de instalação do software servidor de impressão constitui um dos componentes desse sistema, tendo como objetivo definir a configuração do referido servidor, além de atender requisições oriundas das estações. Vale lembrar que foi utilizada a linguagem de programação C para o desenvolvimento dessa interface e das camadas inferiores que implementam o "stub" e o mecanismo de chamada de procedimento remoto. O *Packet Driver*, por sua vez está implementado em linguagem montadora da família de microprocessadores INTEL 8086/286/386 e para interfaceá-lo com os programas em C foi utilizada também linguagem montadora.

O atendimento às solicitações (ATENDER_SO) está esboçado em um diagrama de estrutura modular correspondentes às atividades ali executadas, que pode ser visto na figura 5.3 e os módulos que compõe esse diagrama estão especificados no apêndice A2.

figura 5.3 Estrutura modular do Servidor de Impressão

5.3. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentadas as estruturas de dados utilizadas para que o servidor execute as funções definidas anteriormente, detalhando-se as tabelas internas mantidas para controle do intercâmbio das mensagens, efetivação da comunicação servidor e clientes, de forma a manter o atendimento e dominar todos os periféricos presentes. O capítulo posterior apresenta os testes elaborados desde o início da implementação, com o objetivo de avaliar o resultado com relação às suas interfaces e às necessidades e expectativas dos clientes desse sistema.

CAPÍTULO VI

TESTES E VALIDAÇÃO

Neste capítulo são apresentados os testes elaborados para a validação do sistema de impressão implementado, verificando todas as possíveis respostas que poderiam ser dadas pelo servidor aos clientes do sistema distribuído.

6.1 CONSIDERAÇÕES INICIAIS

No capítulo anterior foram apresentadas as etapas de implementação do servidor de impressão desenvolvido no *LASD*. Este capítulo tem como objetivo validar o referido sistema através de testes realizados, constituindo uma das etapas finais de desenvolvimento do sistema.

Por ter sido desenvolvido de forma modular, cada módulo foi validado logo após a sua implementação. Os módulos de nível mais baixo (que estão logo acima do meio físico da rede local) se tornaram mais complexos para serem implementados.

6.2 SOFTWARE DE COMUNICAÇÃO

Realizou-se inicialmente o teste de comunicação através da rede local, com o intuito de se obter maiores detalhes do mecanismo de acesso ao meio físico (no caso o *Packet Driver*) e para validar a interface com esse mecanismo. Para isso foram utilizados dois equipamentos do tipo IBM/PC conectados por uma interface de rede local *Ethernet*, conforme pode ser visto na figura 6.1.

figura 6.1 Hardware utilizado para teste do software de comunicação

Para sanar os problemas encontrados, entre os quais cita-se a passagem de parâmetros quando ocorre uma interrupção de hardware para o recebimento de pacote (como especificado na figura 6.2), foram utilizadas técnicas de depuração envolvendo utilitários e códigos em linguagem montadora.

figura 6.2 Interrupção provocada para recebimento de pacote

O programa inicialmente elaborado para esses testes foi executado nos dois equipamentos e segue o algoritmo abaixo:

```

ENQUANTO (ch <> '@')
  SE (ch) for digitado
    envia (ch)
  SE (ch) recebido
    exhibe (ch)
  
```

Posteriormente o teste inicial foi modificado e em uma segunda etapa adotou-se um programa que permite a transmissão de pacotes de tamanhos diferentes, de acordo com um menú de opção, conforme descrito a seguir:

A	pacote de 125 bytes
B	pacote de 250 bytes
C	pacote de 500 bytes
D	pacote de 1000 bytes
E	pacote de 2000 bytes

6.3 CHAMADA DE PROCEDIMENTO REMOTO

A mesma configuração de hardware exibida na figura 6.1 foi utilizada para o teste do mecanismo de chamada de procedimentos remotos, porém um deles exerceu a função de servidor do sistema computacional distribuído baseado em rede local e o outro o papel de cliente do sistema.

O referido mecanismo foi testado utilizando-se um conjunto de programas que implementam o "stub" e o RPC no servidor e no cliente, com duas primitivas iniciais que fazem a manipulação de cadeia de caracteres, sendo feita uma opção dos serviços através do menu abaixo:

- 1 - IntroduzirString
- 2 - InverterString
- 3 - EstatísticaString
- 4 - ComplementarSeqüência
- 5 - Terminar

Esse teste demonstrou a execução remota das operações para tratamento de cadeias de caracteres e foi possível se fazer alterações nos programas de testes e validar alguns aspectos do mecanismo de chamada de procedimento remoto. A semântica adotada "*no máximo uma vez*" foi validada quando se introduziu a opção Complementar Seqüência (4), cuja função era fazer o complemento da seqüência de chamada do pedido, sem ter feito um pedido. Isto foi verificado no servidor (número de seqüência esperado diferente do número recebido), provocando o retorno da resposta anterior. Também verificou-se que a não execução pelo término do contador de "*time-out*", após várias tentativas de solicitação sem resposta, ocorria como era esperado: "servidor não responde".

Com relação aos parâmetros padrões do pacote específico do protocolo PSI, também foi possível fazer-se a verificação chegando-se à validade dos campos:

. tipo do protocolo;

. pacote pedido/pacote resposta;

6.4. OS SERVIÇOS DISPONÍVEIS NO SERVIDOR DE IMPRESSÃO

Para a inclusão de novas operações no sistema desenvolvido basta seguir as seguintes instruções:

NA ESTAÇÃO DE TRABALHO

- . implementar a referida operação no "*stub*" da estação, lembrando que para transmitir os dados e/ou parâmetros necessários para a execução no servidor tem-se um pacote de 1500 bytes;
- . definir o protótipo da referida função no arquivo de definições dos tipos e estruturas de dados utilizados na estação (DEFEST.H);
- . alterar o SIMP.C para incluir essa função no menu de opções e obter os parâmetros (se necessários) para a execução da operação.

NO SERVIDOR

- . implementar a operação no SERVIMP.C, através de uma função que recebe sempre três parâmetros (código do usuário, tamanho do "*buffer*" de dados e o "*buffer*");
- . iniciar o ponteiro para a referida função e alocar espaço para manter a última resposta devolvida à estação, no arquivo que simula as funções de outros servidores (INISERV.C);
- . incluir a definição e o protótipo da nova operação no arquivo DEFSERV.H.

Com base nesse roteiro foram retiradas as operações colocadas para teste do mecanismo de RPC e incluídas as operações que o servidor de impressão deve realizar:

- . IMPRESSÃO DE ARQUIVOS
- . DIRETÓRIO DO USUÁRIO
- . ALTERAÇÃO DO NÚMERO DE CÓPIAS
- . TROCA DE POSIÇÃO DE DOIS ARQUIVOS
- . CANCELAMENTO DO PEDIDO DE IMPRESSÃO

Inicialmente testou-se todas as respostas que poderiam retornar caso não tivesse arquivo na fila, ficando obviamente a operação de impressão de arquivos em último lugar para ser testada.

Esses testes iniciais foram realizados através da simulação de filas de impressão, pois na realidade o servidor automaticamente encerra o atendimento aos clientes quando certifica que não existe periférico disponível. Para validação de todos os testes foram incluídos na configuração da figura 6.1 novos equipamentos, obtendo-se o esquema mostrado na figura 6.3.

figura 6.3 Configuração do sistema para testes

Todos os equipamentos utilizados foram IBM/PC conectados por uma interface de rede local ETHERNET, sendo que utilizou-se o sistema *Packet Driver* acima do sistema operacional e nas estações de trabalho o software desenvolvido para testes, enquanto que no servidor o sistema para atendimento dos serviços relacionados aos periféricos de saída. O software que comunica o servidor aos periféricos de saída fica residente permitindo um atendimento de forma mais eficiente e liberando o servidor para recebimento de pedidos enquanto o periférico executa a sua atividade com um determinado arquivo.

6.5. CONSIDERAÇÕES FINAIS

O presente capítulo mostrou os testes elaborados, passo a passo, para avaliação do servidor de impressão produzido, tendo sido validados: a comunicação através da rede local, a transmissão de pacotes, o mecanismo de comunicação e, finalmente os serviços disponíveis no servidor. O próximo capítulo discute o contexto geral deste trabalho, com um resumo das contribuições e uma lista de sugestões para trabalhos futuros.

CAPÍTULO VII

CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Neste capítulo são apresentadas as conclusões gerais atingidas no desenvolvimento do sistema servidor de impressão, baseadas na experiência obtida durante o período. Também são apresentadas sugestões para a continuidade deste trabalho, ou que possam decorrer do mesmo em virtude de se extrapolar a proposta feita inicialmente.

7.1 SOBRE O ESTUDO BIBLIOGRÁFICO

Este trabalho procurou, inicialmente, dar uma visão geral (com um certo grau de profundidade) que permitisse a absorção de conhecimentos ligados à área de informática, especificamente a sistemas computacionais distribuídos baseados em redes locais de computadores. Assim, apresentou-se tópicos relativos a redes de computadores, arquiteturas de sistemas distribuídos, elementos envolvidos nessa classe de sistema, protocolos de comunicação, padronizações adotadas para uniformização da troca de dados, métodos de comunicação, servidores e exemplos específicos de aplicações no domínio desse contexto. Procurou-se abordar tópicos que permitissem revelar maior experimentação da área de sistemas distribuídos e de servidores de impressão, mesmo em detrimento de assuntos importantes, como outros paradigmas adotados na implementação de sistemas distribuídos, protocolos de níveis superiores, sistemas operacionais distribuídos e metodologias para análise de desempenho dessa categoria de sistemas.

Em seguida forneceu, sob o ponto de vista geral, aspectos envolvidos na organização do servidor de impressão desenvolvido, situando o referido servidor em um ambiente de instalação e especificando elementos de hardware e de software envolvidos nesse meio. Foram apresentadas as funções principais do software e seus respectivos fins, além da correlação entre os itens abrangidos nessa estrutura, para que se mantivesse uma organização da mesma.

Deste quadro geral destacaram-se as características do protocolo de comunicação adotado para permitir a troca de parâmetros e dados entre o servidor e os clientes do sistema computacional distribuído. Todos os tipos de mensagens trocadas entre os participantes do sistema foram detalhadas, além do método de gerenciamento escolhido na tomada de decisões em caso de falhas em um dos elementos integrantes desse sistema. Como todas as opções foram fundamentadas em tópicos práticos e teóricos apresentados na literatura, fez-se um paralelo entre os métodos utilizados e o adotado para implementação dos elementos de software.

Para caracterizar e esclarecer detalhes do software servidor de impressão forneceu-se as estruturas de dados utilizadas para produzir o objeto deste trabalho, além da estrutura modular destacando cada unidade e a forma de planejamento adotada de modo a ajustar cada uma delas às outras análogas, formando um todo homogêneo e funcional. Isto é, apresentou-se a arquitetura do projeto do sistema sob o aspecto da incorporação da funcionalidade e portabilidade inicialmente propostas.

O trabalho descreveu a motivação, projeto, implementação, facilidade de comunicação que combina os elementos de um sistema

computacional distribuído. Adotou-se uma abordagem comum para comunicação, tirando proveito desse fato sempre que possível, gerenciando o controle dos dados de forma simples, tornando essa comunicação de fácil utilização e compreensão.

7.2 SOBRE AS DIRETRIZES DO TRABALHO

Os sistemas distribuídos têm sido utilizados para atender a uma gama variada de interesses em diversas áreas. A tecnologia contribui para o atendimento das necessidades dos usuários, estimulando e viabilizando o crescimento em volume, extensão e grandeza dessa classe de sistemas. A diretriz principal deste trabalho foi o desenvolvimento de um servidor de impressão que, além de atender a pedidos que possam ser executados por qualquer periférico de saída, fosse implementado de forma a permitir o atendimento de uma variedade de sistemas operacionais. Em outras palavras o software de teste desenvolvido para cada estação de trabalho deve ser facilmente adaptável a outros sistemas operacionais. Parte da motivação para a elaboração deste trabalho está relacionada diretamente com a portabilidade exigida do software para essa variedade de ambientes operacionais.

Com o objetivo de proteger ao máximo os usuários da escolha da topologia da rede, configuração do sistema, novas versões de ferramentas é que se torna necessária a implementação de sistemas que possam ser utilizados por vários ambientes computacionais.

O servidor de impressão tem disponível um conjunto de operações que visam fornecer aos clientes do sistema facilidades de acesso e impressão, entre outras atividades. Procurou-se elaborar as interfaces desses serviços de forma simples, ou seja, constituindo-as de poucos elementos, e, portanto, de fácil utilização, o que não ocorre na maioria dos servidores disponíveis no mercado. Com o objetivo de fornecer uma gama de operações e pensando em todas as ocorrências possíveis desses serviços é que os servidores são desenvolvidos, esquecendo-se sempre da praticidade de utilização do mesmo. Esse foi um fator considerado como básico para a implementação deste servidor, que possui uma interface voltada ao cliente permitindo a realização de consultas e solicitações rotineiras, de forma natural, concisa e prática.

Percebeu-se com facilidade, em virtude da larga experiência proporcionada, que a utilização de pacotes permite o compartilhamento de variáveis de uma forma organizada e explicitamente declarada. Sendo que já era de grande conhecimento prático que a comunicação entre unidades de programas deve ser feita através de chamadas de procedimento, com passagem de parâmetros ou pelo com-

partilhamento de variáveis visíveis a ambas as unidades, conhecidas por variáveis globais. Assim, a utilização do mecanismo de comunicação escolhido (RPC) ficou bem semelhante ao método já experimentado e de conhecimento bem elevado, que relaciona unidades de programas dentro de um mesmo ambiente. Percebeu-se que um mecanismo de RPC, da forma como foi utilizado, pode ser visto como a composição de um protocolo de transação único recebido em todas as camadas acima do nível de apresentação do modelo OSI, junto a um ambiente no qual clientes e servidores existem. O protocolo é uma parte vital do sistema de gerenciamento dos dados e tecnologia para esse tipo de gerenciamento é considerada crítica pela literatura.

A maioria das pesquisas em sistemas operacionais distribuídos aponta o mecanismo de chamada de procedimento remoto como primitiva básica de comunicação. Além de indicar formas de organização das características que se apresentam distintas dos sistemas operacionais centralizados, tais como nomenclatura, proteção e gerenciamento de recursos, tolerância às falhas do meio ou dos equipamentos e como os serviços podem estar disponíveis aos clientes do sistema. Alguns tipos de servidores podem ser incorporados no sistema distribuído tirando funções dos sistemas operacionais tradicionais que estão fora do *kernel* e no espaço do usuário, como sistema de arquivos, gerenciamento de processos e de terminais.

O projeto apresentado neste trabalho provê uma abstração de chamada de procedimento remoto de forma poderosa, porém encontraram-se problemas bem complexos na integração de elementos que compõe esse mecanismo. O melhor da habilidade foi checar todas as situações normais e anormais de forma clara, passo a passo, não permitindo a inclusão de outra atividade enquanto não se verificava completamente cada item envolvido. Claramente, os exercícios e experimentações de uma validação formal com a implementação completa pode expor alguma inadequação no projeto.

7.3 A CONTINUIDADE DO TRABALHO

Foi definido um conjunto de objetivos principais para sistema distribuídos, alguns deles relacionados diretamente ao desempenho do sistema. Outros porém, tais como interface do usuário e extensibilidade, são relativos mais à qualidade de como o sistema é percebido pelo usuário. Com o objetivo de atingir um sistema adaptável e confiável deixou-se de lado a avaliação a nível de desempenho, usando mais referências bibliográficas na escolha de formas de implementação. Até o ponto de testes onde foram utilizadas três estações de trabalho, o protocolo definido permitia um bom desempenho do sistema, mas podem ser feitas novas avaliações incluindo mais estações e definindo um protocolo que transfira

qualquer arquivo em um único pacote, no lugar de subdividi-lo em vários pacotes para transmissão. Baseando-se nesses dois fatores podem ser elaborados testes mais precisos de desempenho.

Considerações para o projeto de sistemas distribuídos são apresentadas, plano de testes são discutidos e examinadas inspeções de projeto, além de diretrizes para o processo de implementação. Porém, projeto é uma atividade criativa que envolve arte e tecnologia, logo não se consegue exaurir todas as formas de implementação possíveis para se escolher a melhor delas. Fica em aberto uma série bem distinta de especificações, entre elas pode-se citar o excesso de zelo para evitar que pacotes de um mesmo arquivo cheguem ao servidor fora de ordem. Isso é controlado por dois fatores: um na semântica adotada no mecanismo de comunicação e outro no parâmetro definido no buffer de transmissão. Pode ser verificado se isso se torna um problema em termos de perda em desempenho do sistema, caso seja definida uma outra semântica para o mecanismo de chamada de procedimento remoto. Dependendo da confiabilidade da rede a semântica *possivelmente recebida* pode ser utilizada em virtude das operações executadas nesse servidor não exigirem um rigor maior em verificações, visto que deixa de provocar inconsistências no sistema caso haja falha na estação, servidor ou na rede.

Para o transporte do sistema para outros ambientes operacionais deve-se levar em conta a adaptação das atividades a serem executadas nos ambientes de testes, para a execução no servidor (especificadas no capítulo VI) com relação à inclusão de operações nas estações de trabalho.

Em continuidade à este trabalho popõe-se, portanto:

- . avaliação do sistema a nível de desempenho;
- . inclusão de mais estações de trabalho para análise do desempenho, com participação a nível de clientes do sistema;
- . definição de um outro protocolo que permita a transmissão de qualquer arquivo, independente do seu tamanho, em um único pacote, conhecido na literatura como "*Single Shot Protocol*";
- . alterar algumas especificações do projeto, como meio de controle dos pacotes de um arquivo enviados e recebidos pelos participantes;
- . adoção de outras semânticas para o mecanismo de comunicação, para análise a nível de desempenho;
- . inclusão de novas operações no sistema inicialmente proposto, para que o mesmo possa conviver com outros servidores no

mesmo ambiente;

. transporte do software de teste da estação de trabalho que participa como cliente do sistema distribuído para outros ambientes operacionais.

APÊNDICE A1

FUNÇÕES DISPONÍVEIS NO "PACKET DRIVER"

A descrição detalhada dos módulos abaixo apresentados pode ser encontrada em [CLA89]; aqui o objetivo é apenas apresentá-las para um entendimento melhor do trabalho desenvolvido.

```
. driver_info(handle)      AH == 1 AL == FF (255)
    int handle;            BX
```

Cuja função é retornar informações sobre a interface com o meio físico da rede local. Caso ocorra erro o seu código será especificado em *DH* e o *flag carry* posicionado, sendo *BAD_HANDLE* o erro possível. Se o *flag carry* estiver zerado indicará um retorno sem erro informando os seguintes dados, nos respectivos registradores:

```
version          BX
class            CH
type             DX
number          CL
name             DS:SI
basic/extendend  AL ( 1 == básica
                    2 == extendida
                    5 == básica e alto desempenho
                    6 == básica, alto desempenho e
                      extendida
                    FF == não disponível)
```

```
. int acess_type(if_class, if_type, if_number, type,
                 typelen, receiver) AH == 2
    int if_class;      AL
    int if_type;       BX
    int if_number;     DL
```

```

char far *type;           DS:SI
unsigned typelen;         CX
int (far *receiver) ();   ES:DI

```

Essa interface da função inicia o acesso a um tipo específico de pacote, que quando for recebido provoca a chamada da rotina de recepção duas vezes. A primeira vez para solicitar o *buffer* (AX == 0) cujo retorno será um ponteiro para o *buffer* (em ES:DI) ou 0:0 caso não tenha *buffer* disponível. A segunda chamada (AX == 1) é necessária para indicar o término da cópia dos dados no *buffer*. Os erros possíveis são:

```

NO_CLASS;
NO_TYPE;
NO_NUMBER;
BAD_TYPE;
NO_SPACE ou
TYPE_ INUSE,

```

cujo código será retornado em *DH* e o *flag carry* posicionado. No retorno sem erro o *flag carry* estará zerado e o *handle* devolvido em *AX*, sendo a rotina de recepção definida da seguinte forma:

```

. (*receiver)(handle, flag, len [, buffer])
  int handle;           BX
  int flag;             AX
  unsigned len;         CX
  if AX == 1,
    char far *buffer;    DS:SI

```

Para terminar o acesso aos pacotes associados com o *handle* está definida a rotina:

```

. int release_type(handle)  AH == 3
  int handle;              BX

```

Ocorrendo os mesmos tipos de retornos que as anteriores, ou seja, *flag carry* posicionado indica erro, cujo código será devolvido em *DH* e sendo *BAD_HANDLE* o provável erro, enquanto que zerado indica retorno sem erro.


```

. int send_pkt(buffer,length) AH == 4
  char far *buffer;          DS:SI
  unsigned length;           CX

```

Cuja finalidade é transmitir uma quantidade *length* de bytes de dados, apontados por *buffer*, incluindo todo o pacote inclusive *headers* de rede local. O *flag carry* zerado indica retorno sem erro e posicionado acusa um erro, cujo código estará em *DH*, sendo *CANT_SEND* o provável.

```

. terminate(handle)          AH == 5
  int handle;                 BX

```

Nessa função é encerrado o *driver* associado com o *handle*, sendo os possíveis erros *BAD_HANDLE* e *CANT_TERMINATE* devolvidos em *DH*, quando o *flag carry* estiver posicionado. Caso contrário indica um retorno sem problemas.

```

. get_address(handle,buf,len) AH == 6
  int handle;                 BX
  char far *buf;              ES:DI
  int len;                     CX

```

Faz a cópia para *buf* do endereço da interface de rede local que foi associado ao *handle*. Em caso do *flag carry* estar zerado, indicando retorno sem erro, devolve o comprimento no registrador *CX*. Já no retorno com erro (*flag carry* posicionado) o código do mesmo é encontrado em *DH*, sendo *BAD_HANDLE* e *NO_SPACE* os erros prováveis.

```

. reset_interface(handle)     AH == 7
  int handle;                 BX

```

Responsável pelo retorno da interface associado ao *handle*, para um estado padrão, abortando alguma transmissão no processo e reiniciando o recebimento. Da mesma forma que as outras funções o *flag carry* zerado indica retorno sem erro; enquanto que posicionado devolve em *DH* o código de erro *BAD_HANDLE* ou *CANT_RESET*.

APÊNDICE A2

INTERFACE DOS MÓDULOS - ATENDER_SO

São apresentadas as especificações de interface dos módulos correspondentes ao software executado no servidor de impressão, com o objetivo de esclarecer tópicos delineados no conteúdo deste trabalho.

```
. ObterParametros(int intno, int handle, int *version,  
                  int *class, int *type, int *number,  
                  int *basic)
```

Com a função de retornar informações sobre a interface com a rede local, a partir do número da interrupção e do identificador do processo de recepção. O possível erro devolvido é: *handle inválido* (1) e como entrada são fornecidos:

- o número da interrupção (*intno*);
- o identificador do processo de recepção (*handle*).

E na saída são devolvidos:

- a versão do *Packet Driver* (*version*);
- a classe da interface (*class*);
- o tipo da interface (*type*);
- o número da interface (*number*);
- o tipo de funções disponíveis (*basic*).

```
. HabilitarAcessoTipoPacote(int intno,int *handle,int if_  
                             class,int if_type,int if_number,  
                             BYTE _far *type, unsigned short  
                             typelen,void (_far *receiver)())
```

Com a função de iniciar o acesso a um certo tipo de pacote, devolvendo os possíveis erros:

- . classe de interface não válida (2);
- . tipo inválido de interface de rede (3);
- . número de interface inválido (4);
- . tipo não válido de pacote (5);
- . sem espaço disponível para armazenamento do pacote (9);
- . tipo em uso (10).

Sendo que na entrada devem ser fornecidos os seguintes parâmetros de controle:

- número da interrupção (*intno*);
- classe da interface de rede (*if_class*);
- tipo de referida interface (*if_type*);
- comprimento do pacote (*typelen*);
- rotina de recepção (*receiver*).

E devolvido:

- identificador do processo de recepção (*handle*).

. AtenderProcedimentoRemoto()

Tem a função de atender a chamada de um procedimento remoto, devolvendo uma resposta, sendo executadas diversas funções de forma iterativa. Os possíveis erros que podem ser devolvidos são:

- . erro na execução da operação (0x12);
- . requisição não aceita (0x10);
- . usuário sem permissão para solicitar o pedido (0x08);
- . sessão não existe (0x04);
- . fim do contador de *time-out* para a execução da operação (0x01).

. TerminarAcessoTipoPacote(int intno, int handle)

Esta função termina o acesso a um pacote associado ao *handle*, podendo devolver o código 1 indicando que o *handle* é inválido. Para isso necessita dos seguintes parâmetros que são fornecidos na entrada:

- . número da interrupção (*intno*);
- . identificador do processo de recepção (*handle*).

Para o atendimento das requisições que chegam ao servidor de impressão diversas funções são realizadas, sendo descritas a seguir a interface desse módulo.

. ReceberPSI(BYTE *buf)

Tendo a função de receber um pacote/pedido da estação, retirar o *header* e após verificação passar ao outro módulo, dando abertura para a realização da operação solicitada pelo cliente. No término é fornecido um *buffer* com o conteúdo idêntico ao do pacote tipo PSI, ou um código indicando que não existem pacotes a serem retirados (-1).

. AnalisarPedido(int sessao, BYTE *buf_ent,int *usuario)

Com a finalidade de fazer a análise do pacote que chegou ao servidor, para verificar se satisfaz as especificações de um pacote de solicitação de serviços. Analisando se a sessão já foi aberta, se o cliente tem permissão e se o pacote é do tipo esperado. Como entrada são dados:

- identificador da sessão (*sessao*);
- pacote tipo PSI recebido (*buf_ent*);
- código do usuário solicitante (*usuario*).

Como saída será passado ao outro módulo:

- estado após a análise (*buf_ent*).

. DesmontarPacotePedido(BYTE *buf_ent, int *tam_sai, BYTE buf_sai)

Com o objetivo de receber um pacote do tipo PSI, retirar o seu cabeçalho retornando um campo de dados e seu respectivo tamanho, para realização da operação solicitada. Como parâmetro de entrada é fornecido:

- o pacote PSI recebido da estação (*buf_ent*).

E na saída fornecido os dados:

- tamanho do *buffer* de saída (*tam_sai*);
- dados para execução da operação (*buf_sai*).

Em seguida é realizada a operação correspondente solicitada, de forma alternativa estando no sistema disponíveis cinco (5) delas, que estão especificadas no apêndice A3, gerando um pacote resposta ao cliente.

. MontarPacoteResposta(int sessao, BYTE oper, int tam_dad, BYTE *dados, BYTE pac_psi, BYTE estado)

Cuja função é montar um pacote com especificações do PSI, sendo que para essa finalidade são passados os seguintes parâmetros:

- identificador da sessão (*sessao*);
- identificador da operação solicitada (*oper*);
- quantidade de bytes que estão sendo enviados (*tam_dad*);
- dados e/ou códigos de mensagens que estão sendo transmitidos à estação como resposta à solicitação (*dados*);
- estado resultante da execução do serviço (*estado*).

Como saída tem-se:

- o pacote com especificação de resposta do servidor de impressão (*pac_psi*).

. EnviarResposta(int sessao, BYTE oper, int tam_dad, BYTE *buf_ent, BYTE *pac_psi, BYTE estado)

Tendo a função de devolver ao cliente o pacote que corresponde à resposta e está sendo aguardado, para isso inclui as especificações de um pacote PKT. Os seguintes dados devem ser passados a essa interface desse módulo:

- o identificador da sessão (*sessao*);
- o identificador da operação executada (*oper*);
- quantidade de bytes a serem enviadas (*tam_dad*);
- dados e/ou parâmetros a serem transmitidos (*buf_ent*);
- estado resultante da execução (*estado*).

Como resultado pode ser obtido um erro de transmissão, o qual será informado, ou então a devolução à estação do pacote resposta.

. TransmitirPSI(int sessao, BYTE *buffer)

Tendo a função específica de montar e transmitir um pacote tipo resposta PKT a um processo devidamente identificado. Como entrada tem-se:

- identificador da sessão (*sessao*);
- pacote com especificação resposta PSI (*buffer*).

Sendo devolvido:

- pacote com especificação resposta PKT (*buffer*).

APÊNDICE A3

INTERFACE DOS SERVIÇOS DISPONÍVEIS AOS USUÁRIOS DO SISTEMA

Conforme já foi explanado o servidor de impressão tem disponível os serviços básicos que devem ser executados para atendimento aos clientes dos vários sistemas. Aqui são apresentadas as interfaces das funções que realizam essas operações.

. ImprimirArquivo(int usuario, int *tam_dad, BYTE *dados)

Cuja função é atender o pedido da operação solicitada pelo cliente para imprimir um determinado arquivo, cujo nome, tamanho, número de cópias e conteúdo desse arquivo são fornecidos em *dados*. Na entrada devem ser passados os seguintes parâmetros:

- identificador do usuário (*usuario*);
- comprimento em bytes dos dados a serem lidos (*tam_dad*);
- *buffer* de entrada contendo parâmetros para a execução da operação e os dados (*dados*).

Como saída são retornados:

- comprimento em bytes dos dados resposta (*tam_dad*);
- resposta da operação (*dados*).

As possíveis mensagens resultantes da solicitação desse serviço são:

- . Sem memória disponível (901);
- . Sem posição na fila (902);
- . Arquivo não pode ser criado (903);
- . Problema de escrita no arquivo (904);

- . *Arquivo já está na fila* (905);
- . *Arquivo colocado na fila* (906).

. `RelacionarArquivos(int usuario,int *tam_dad, BYTE *dados)`

Tem a função de relacionar os arquivos do cliente que fez a solicitação, os quais estão na fila de impressão, em impressão, ou em um registro provisório aguardando a liberação de uma posição na fila. Na entrada deve ser fornecido:

- identificador do usuário (*usuario*).

E na saída são retornados:

- tamanho em bytes da resposta (*tam_dad*);
- relação dos arquivos, com respectiva fila, número de cópias e *status* (em impressão ou aguardando) (*dados*).

No caso desse serviço o usuário poderá receber a seguinte mensagem:

- . *Sem arquivos na fila* (911);

. `AlterarCopias(int usuario,int *tam_dad, BYTE *dados)`

Com a função de alterar o número de cópias de um arquivo que está na fila de impressão. Para esse tipo de atendimento os seguintes dados devem ser fornecidos:

- identificador do usuário (*usuario*);
- tamanho em bytes dos parâmetros para a realização desse tipo de serviço (*tam_dad*);
- o nome do arquivo e o novo número de cópias (*dados*).

Devendo ser retornado:

- a quantidade em bytes da resposta (*tam_dad*);
- resposta da execução (*dados*).

Os erros ou informações que podem ocorrer com a solicitação desse serviço são:

- . *Número de cópias alterado* (921);
- . *Arquivo não está na fila* (922);
- . *Arquivo não pertence ao usuário* (923).

. *TrocarPosicao*(int usuario,int *tam_dad,BYTE *dados)

Faz a troca de posição e/ou fila de dois arquivos que estejam aguardando um periférico de impressão livre. Como entrada são passados:

- identificador do usuário (*usuario*);
- quantidade em bytes dos parâmetros (*tam_dad*);
- nome dos dois arquivos que serão trocados de posição e/ou fila (*dados*).

Como retorno desse serviço tem-se:

- tamanho em bytes da resposta (*tam_dad*);
- código indicativo do resultado da execução (*dados*).

As mensagens prováveis são:

- . *Os arquivos não estavam na fila* (931);
- . *Primeiro arquivo não encontrado* (932);
- . *Segundo arquivo não encontrado* (933);
- . *Posições trocadas* (934).

. *CancelarImpressao*(int usuario,int *tam_dad,BYTE *dados)

Determina o cancelamento do pedido de impressão de um arquivo que esteja na fila de impressão. Na entrada são fornecidos os parâmetros:

- identificador do usuário (*usuario*);
- quantidade em bytes que devem ser lidos (*tam_dad*);

- nome do arquivo a ser interrompida a impressão (*dados*).

Na saída tem-se:

- quantidade em bytes da mensagem resultante da operação (*tam_dad*);
- código resposta ao usuário (*dados*).

Sendo possíveis as seguintes mensagens:

- . *Impressão cancelada* (941);
- . *Arquivo não encontrado* (942).

Para qualquer um dos serviços solicitados, caso não seja diagnosticado o problema ocorrido, após o cancelamento das execuções parcialmente realizadas será devolvida a seguinte mensagem:

- . *Recomece a operação* (999).

Vale a pena observar que existe uma uniformidade nos parâmetros de entrada e de saída, que o módulo independente de utilizá-lo faz o seu manuseio de forma clássica, possibilitando uma facilidade na adaptação de outros serviços nesse sistema.

REFERÊNCIAS

- [ADR88] ADRIAN, MERV. *The workstation Data Link*. Mc Graw Hill, 1988.
- [AMP90] MANUAL TECNICO DA REDE AMPLUS.
- [ASC86] ASCHENBRENNER, J. R. *Open Systems Interconnection*. IBM Systems Journal, vol. 25, nos 3/4, pag. 369 a 379, 1986.
- [BAC89] BACON, J. M.; LESLIE, I. M.; NEEDHAM, R. M. *Distributed Computing with a Processor Bank*. Technical Report no 168, University of Cambridge Computer Laboratory, abril 1989.
- [BAL89] BAL, HENRI E.; STEINER, JENNIFER G.; TANENBAUM, ANDREW S. *Programming languages for Distributed Computing systems*. ACM COMPUTING SURVEYS, vol. 3, pag. 261 a 322, setembro 1989.
- [BIR80] BIRREL, A. D.; NEEDHAM, R. M. *A universal file server*. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING no 6, pag. 450 a 453, setembro 1980.
- [BIR84] BIRREL, ANDREW D.; NELSON, B. J. *Implementing Remote Procedure Calls*. ACM TRANSACTIONS ON COMPUTER SYSTEMS, vol. 2, n. 1, pag. 39 a 59, fevereiro 1984.
- [BIR85] BIRREL, ANDREW D. *Secure communication using Remote Procedure Calls*. ACM TRANSACTIONS ON COMPUTER SYSTEMS, vol. 3, no 1, pag. 1 a 14, fevereiro 1985.
- [CHU84] CHURCHILL, B. N. *Plan 4000 - An overview of a token passing local area network for the IBM PC*. PC TECH. JOURNAL, fevereiro 1984.
- [CLA89] USER DOCUMENTATION FOR THE PACKET DRIVER COLLECTION. Version 1.09, US. Clarkson University, 1989.
- [COU88] COULOURIS, F.; DOLLIMORE, J. *Distributed Systems - concepts and design*. Addison Wesley, 1988.
- [ENS78] ENSLOW JR., PHILIP H. *What is a distributed data processing system?*. COMPUTER, janeiro 1978.

- [FER85] FERRO, VANIA MARIA. *O servidor de impressão da Rede local ITAUTEC*. 3o Simpósio Brasileiro sobre Rede de Computadores. Rio de Janeiro, 1985.
- [FTP89] FTP SOFTWARE INC. *PC/TCP Packet Driver Specification. Version 1.09*. US, FTP Software Inc., 1989.
- [GUI89] GUIBALY, FAYEZ EL. *Design and analysis of Arbitration Protocols*. IEEE TRANSACTIONS ON COMPUTERS, vol. 38, no 2, pag. 161 a 171, fevereiro 1989.
- [JAN83] JANSON, P.; SVOBODOVA, L. MAEHLE, E. *Filing and Printing Services on a Local Area Network*. Proc. 8th. Symp. on a Data Communications ACM, pag. 211 a 220, 1983.
- [KAK91] KAKUDA, YOSHIAKI; SAITO, HIRONORI. *An Integrated Approach to Design of Protocol Specifications Using Protocol Validation and Synthesis*. IEEE TRANSACTION ON COMPUTERS, vol. 40, n. 4, pag 459 a 467, abril 1991.
- [KES90] KESSLER, GARY C. *ISDN Concepts, Facilities and Services*. Mc Graw - Hill Communications Series, 1990.
- [LAG86] LAGES, NEWTON A. C.; NOGUEIRA, JOSE M. S. *Introdução aos Sistemas Distribuídos*. Editora da UNICAMP - Papirus, 1986.
- [LAM80] LAM, SIMON S. *A carrier Sense Multiple Access Protocol for Local Networks*. COMPUTER NETWORKS, no 4, pag. 21 a 32, 1980.
- [LAM83] LAM, SIMON S. *Multiple Access Protocols*. COMPUTER COMMUNICATIONS, vol. 1, 1983.
- [LEV90] LEVY, ELIEZER; SILBERSCHATZ, ABRAHAM. *Distributed File Systems: concepts and examples*. ACM Computing Surveys, vol. 22, no 4, dezembro 1990.
- [LIN91] LIN, HUAI - AN. *Constructing Protocols with Alternative Functions*. IEEE TRANSACTIONS ON COMPUTERS, vol. 40, n. 4, pag. 376 a 386, abril 1991.
- [LIS90] LISKOV, BARBARA; SHRIRA, LIUBA; WROCLAWSKI, JOHN. *Efficient At-Most-Once Messages based on Synchronized Clocks*. Computer Communications Review, vol. 20, n. 4, pag. 41 a 49, setembro 1990.
- [LUC78] LUCZAK, EDWARD C. *Global Bus Computer Communications Techniques*. Proc. Computer Networking Symposium, dezembro 1978.

- [MIT82] MITCHELL, JAMES G. *File Servers for Local Networks*. Local Area Network. Lecture Notes. Canterbury University of Kent IRIA, pag. 83 a 114, 29 março - 2 abril/1982.
- [MUL89] MULLENDER, S. J. *Systems Distributed*. ACM Press, 1989.
- [MUR89] MURMAN, EARLL M. *Introduction the Athena Project: Goals, Philosophy, Status and Experiences*. Technical Notes. MIT Massachusetts Institute of Technology. Cambridge, 1989.
- [NEE82] NEEDHAM, R. M. et al. *The Cambridge Distributed Computing System*. INTERNATIONAL COMPUTER SCIENCE SERIES, 1982.
- [PAC89] THE PACKET DRIVER SPECIFICATION, version 1.09.
- [RAV89] RAVINDRAN, K.; CHANSON, SAMUEL T. *Failure Transparency in Remote Procedures Calls*. IEEE TRANSACTIONS ON COMPUTERS, vol. 38, no 5, agosto 1989.
- [ROS89] ROSSUM, GUIDO VAN. *AIL - A Class-Oriented RPC stub Generator for Amoeba*. Lecture Notes in Computer Science Edited by G. Goos and J. Hartmanis, no 433. Progress in Distributed Operating Systems and Distributed Systems Management. European Workshop, Berlin, FRG. Proceedings Springer-Verlag, abril 1989.
- [SAN87] SANTOS, LUIS G. L. *Servidor de Impressão da Rede Local GPLOC do SERPRO*. SUCESU Informática XX Congresso Nacional de Informática - 31/8 a 6/9/1987.
- [SAN89] SANTANA, MARCOS J. *An advanced Filestore Architecture for a Multiple-LAN Distributed Computing System*. Department of Eletronics and Computer Science. University of Southampton U.K., outubro 1989.
- [SCH88] SCHWARTZ, MISCHA. *Telecommunication Network: Protocols, Modeling and Analysis*. Addison Wesley Publishing Company 1988.
- [SCH89] SCHROEDER, MICHAEL D. e BURROWS, MICHAEL. *Performance of Firefly RPC*. In Proceedings of the 12th. ACM Symposium on Operating Systems Principles, dezembro 1989.
- [SCH89A] SCHMIDT, NAOMI B. *An overview of Project Athena*. Technical Notes. Massachusetts Institute of Technology, setembro 1989.

- [SHR82] SHRISVASTAVA, S. K. e PANZIERI, F. *The design of a reliable Remote Procedure Call Mechanism*. IEEE TRANSACTIONS ON COMPUTERS, vol. C-31, no 7, pag. 692 a 697, julho 1982.
- [SPE82] SPECTOR, ALFRED Z. *Performing Remote Operations Efficiently on a Local Computer Network*. COMMUNICATIONS OF THE ACM, vol. 25, no 4, pag. 246 a 260, abril 1982.
- [STA84] STALLINGS, WILLIAM. *Local Networks*. COMPUTING SURVEYS, vol.16, no 1, pag. 3 a 39, março 1984.
- [STE88] STEINER, JENNIFER G. e GEER JR., DANIEL E. *Network Services in the Athena Environment*. Technical Notes. Massachusetts Institute of Technology, julho 1988.
- [SVO84] SVOBODOVA, LIBA. *File Servers for Network-Based Distributed Systems*. COMPUTER SURVEYS, vol. 16, no 4, pag. 353 a 398, dezembro 1984.
- [TAN85] TANENBAUM, ANDREW S.; VAN RENESSE, ROBERT. *Distributed Operating Systems*, COMPUTING SURVEYS, vol. 17, no 4, pag. 419 a 471, dezembro 1985.
- [TAN87] TANENBAUM, ANDREW S. *Operating Systems: Design and Implementation*. Prentice-Hall Inc. Englewood Cliffs, New Jersey 1987.
- [TAN90] TANENBAUM, ANDREW S.; VAN RENESSE, ROBERT; HANS VAN SHARP; MULLENDER, GREGORY J., JANSEN, SAPE J.; ROSSUM, JACK; VAN, GUIDO. *Amoeba*. Communications of the ACM, vol. 33, no 12, dezembro 1990.
- [TRE76] TREMBLAY, J. P. e SORENSON, P. G. *An introduction to Data Structures with Applications*. MC GRAW Hill Computer Science Series, Estados Unidos 1976.
- [TRI91] TRINDADE Jr., ONOFRE. *Servidor de Processamento Paralelo baseado em banco de Transputers*. Tese de Doutorado - IFQSC - 1991.
- [WAR85] WARD, PAUL T. e MELLOR, STEPHEN J. *Introduction & tools Structured Development for Real-Time Systems*. Yourdon Press Computing Series, vol 1, 1985.
- [WAT83] WATSON, R. W. *Distributed System Architecture Model*. In: *Distributed Systems Architecture and Implementation*. Ed. Lampson, Paul & Slegert Spring-Verlag, Berlin Heidelberg New York 1983.

- [WIL80] WILKES, MAURICE V. e NEEDHAM, R. M. *The Cambridge Model Distributed System*. ACM OPERATING SYSTEMS REVIEW, vol. 14, no 1, janeiro 1980.
- [WIL87] WILBUR, STEVE e BACARISSE, BEN. *Building Distributed Systems with Remote Procedure Call*. SOFTWARE ENGINEERING JOURNAL, pag. 148 a 159, setembro 1987.
- [ZIM80] ZIMMERMAN, H. *OSI Reference Model: the ISO Model of Architecture for Open Systems Interconnection*. IEEE TRANSACTIONS ON COMMUNICATION, vol. 28, no 4, pag. 425 a 432, abril 1980.