SERVIDORES DE ARQUIVOS DUPLICADOS EM REDES LOCAIS COM AMBIENTE UNIX

Afonso Jorge Ferreira Cardoso[1]
afonso@cpatu.embrapa.br
Ingrid Jansch Pôrto[2]
ingrid@inf.ufrgs.br

RESUMO: A utilização de técnicas de tolerância a falhas em uma rede local possibilita o aumento da confiabilidade e da disponibilidade dos sistemas computacionais. Considerando aplicações genéricas, as redes locais apresentam confiabilidade suficiente, a qual pode alternativamente ser controlada através do próprio software existente. Assim, a maioria dos problemas ainda é relacionado à indisponibilidade de longo prazo, e mudar essa realidade passou a ser a principal meta em uma rede local tolerante a falhas.

Como forma de aumentar a disponibilidade, este trabalho propõe uma solução de rede local cliente/servidor com servidor duplicado, tendo como sistema operacional básico o UNIX. O sistema trabalha com um único servidor (servidor primário), do ponto de vista dos usuários, que além de atender as requisições de serviços dos clientes, faz o "espelhamento" dos dados para uma unidade reserva (servidor secundário). Caso o primário falhe, o secundário assume suas funções na rede. O principal objetivo visado não foi o de propor uma solução inédita para o problema, mas o de estudar detalhadamente uma implementação operacional.

1. Introdução

Redes locais de computadores necessitam cada vez mais de características capazes de aumentar a sua disponibilidade, apesar das baixas taxas de erros (1 bit errado a cada 10^8 a 10^{11} bits transmitidos) e das altas taxas de transmissão (de 0,1 a 100 Mbps) [10]. A concentração da maior parte dos recursos computacionais nesse tipo de rede é um dos pontos chave desta necessidade; certamente outro é a dependência criada pelo uso intensivo de redes nas mais diversas aplicações *on-line*. No processamento comercial, por exemplo, os terminais interligados em rede passaram a substituir muitos papéis e cópias destes.

A disponibilidade está diretamente relacionada à capacidade de manter a rede em funcionamento, mesmo na ocorrência de falhas. Essa capacidade pode ser atingida através de mecanismos de tolerância a falhas, obtidos pelo uso de redundância em determinados componentes de um sistema, sejam eles de *software* ou *hardware*. Essa redundância atua como alternativa de funcionamento à parte do sistema afetado pela falha.

A implantação de um sistema tolerante a falhas requer um planejamento bem elaborado: os requisitos necessários serão dependentes do ambiente e objetivos funcionais da rede local, neste caso. A segurança de funcionamento desta, de forma global e em cada um dos seus nodos, estará associada à aplicação na qual ela se insere: provavelmente uma rede empregada em um ambiente acadêmico terá requisitos de funcionamento bastante diversos de outra que controle um ambiente industrial. Para que se possa justificadamente confiar no funcionamento desta segunda rede, na correção dos serviços prestados e na manutenção permanente destes, é preciso agregar mecanismos de tolerância a falhas.

As redes locais mais comuns, ou seja, aquelas encontradas na maioria das instituições, utilizam computadores pessoais como nodos da rede. Os computadores pessoais disponíveis comercialmente não possuem arquitetura tolerante a falhas em sua origem, exceto por algumas poucas técnicas de suporte. Se não é pos-

* 1 Mestre em Ciência da Computação, UFRGS; Analista de Sistemas da EMBRAPA - Amazônia Oriental; Professor Adjunto da UNAMA.
* 2 Doutora Ing. em Microeletrônica, INPG, FR, 1985; Profa. do Instituto de Informática da UFRGS.

sível assegurar informações no âmbito de cada máquina, pode-se reforçar a arquitetura de alguma poucas máquinas (servidores). Esta arquitetura robusta traduz-se em qualidade de funcionamento, permitindo a atribuição de responsabilidade necessária à execução de certas atividades tais como abrigar arquivos importantes (ou cópias destes) e realizar o controle de ações críticas.

Assim, a utilização de servidores duplicados em uma rede local de computadores torna-se uma técnica de uso simples, factível e de baixo custo para um grande número de instituições, possibilitando o aumento da disponibilidade e da confiabilidade dos sistemas computacionais.

Este trabalho visa apresentar e desenvolver uma solução de rede local cliente/servidor com servidor duplicado utilizando como base o sistema operacional UNIX. São tratados os seguintes aspectos: descrição do ambiente usado para a implementação; apresentação do modelo de falhas considerado; comentários sobre o sistema de arquivos utilizado; características de implementação; funcionamento do par de servidores, em funcionamento normal e sob falhas, e conclusões.

2. Ambiente usado para implementação

A parte experimental deste trabalho foi desenvolvida em uma rede local cliente/ servidor com sete computadores IBM-PC com processador Pentium, usando o sistema operacional Linux versão 2.0.0. Esses computadores foram organizados de forma a constituir uma rede com cinco computadores clientes e um servidor duplicado (dois computadores). No servidor, apenas um dos computadores é ativo em relação aos clientes, o outro atua como "reserva-quente".

Com relação aos servidores, o ambiente corresponde a duas redes *Ethernet*. A primeira constitui o barramento principal da rede, através do qual as estações-clientes comunicam-se com a unidade ativa (primário). O servidor secundário, ao se comunicar com o primário através do barramento principal, é visto por este como um cliente comum, apesar de não atuar como tal.

A segunda constitui o barramento especial (que utiliza cabo *ethernet* fino) através do qual estão interligados apenas as unidades primária e secundária. Esse barramento é utilizado exclusivamente para as ações relativas à duplicação, visando a mínima interferência destas no funcionamento normal da rede *Ethernet* base. Não existe qualquer dependência entre os barramentos principal e especial. A figura 1 mostra detalhes da interligação dos servidores primário e secundário.

3. Modelo de falhas utilizado

O modelo de falhas foi definido considerando as características de uma rede *Ethernet* típica modelo cliente/servidor, com poucos módulos clientes e com probalidade desprezível quanto à ocorrência de falhas "bizantinas"[1]. O modelo de falhas possui as seguintes definições:

1. Quaisquer servidores e clientes podem falhar a qualquer tempo.
2. Servidores falham por *crash* ou por desconexão completa, mas não é suposto o uso de máquinas especiais com características *fail-stop*.

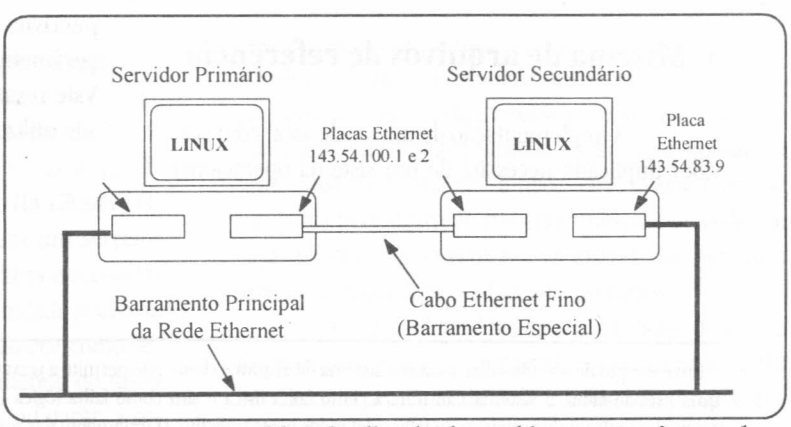

Figura 1 - O Servidor duplicado do ambiente experimental

[1] Modelo genérico de falhas, que supõe a possibilidade de ocorrência de qualquer conseqüência sobre o comportamento do sistema, como resultado de uma falha.

3. Não é considerada a possibilidade de particionamento na rede de estações.
4. A rede de comunicação pode perder pacotes mesmo na ausência de falhas de servidores e estações, mas este evento tem baixa probabilidade **em redes locais** devido aos protocolos usados.
5. Podem ocorrer falhas simultâneas em todas as estações, como conseqüência de falta de energia, por hipótese.
6. A religação lógica ou física de um cliente não é sinalizada por nenhum evento específico. Entretanto, na solução ora proposta, ocorre a sinalização quando se tratar dos servidores primário e secundário.
7. Podem ocorrer pseudo-falhas[2], ou seja, operações que não podem ser realizadas por ambos servidores devido a alguma condição anormal.
8. As conexões de rede e barramento especial podem falhar independentemente da ocorrência de falhas nos servidores. Esta definição implica a necessidade de independência entre a atividade nos barramentos principal e especial.

Este modelo de falhas foi escolhido com base no especificado para o sistema de arquivos RNFS [4], o qual foi modificado e integrado ao sistema de controle do servidor. As definições 4, 6 e 7 recém apresentadas foram adequadas de forma a contemplar os objetivos visados neste trabalho.

4. Sistema de arquivos de referência

A implementação de uma rede local com servidor duplicado necessita de um sistema operacional que tenha capacidade de manipulação de arquivos, de forma a atender requisitos como, por exemplo, a distribuição transparente de arquivos. Essa necessidade é justificada pela ação de "espelhamento" dos dados.

Entre os módulos que compõem o sistema operacional UNIX, o sistema de arquivos é o responsável pela manipulação de todos os arquivos existentes. Em uma rede com estações UNIX, o NFS (*Network File System*), desenvolvido pela *SUN Corporation*, é usado como sistema de arquivos padrão. Entretanto, ele não dá suporte para replicação de arquivos e diretórios [12], desestimulando seu uso neste trabalho. Deste modo, buscou-se uma alternativa compatível com o NFS, mas que introduzisse características de tolerância a falhas.

Foi escolhido o RNFS (*Reliable Network File System*) [4], que é um sistema de arquivos distribuído tolerante a falhas, projetado sobre a estrutura do sistema operacional UNIX e do sistema de arquivos NFS. Utiliza técnica para replicação de arquivos baseada no método de cópia primária e RPCs para a comunicação cliente-servidor e unidade primária - unidade secundária.

A implementação do RNFS envolveu a modificação do protocolo de RPC usado (para ação de replicação), do comportamento dos monitores que implementam os servidores e do comportamento dos *drivers* de acesso remoto a arquivos no núcleo do sistema operacional de todas as estações clientes.

O protocolo de RPC possui originalmente 18 tipos de procedimentos no NFS; foi necessário adicionar quatro procedimentos RPCs aos já existentes. A tabela 1 apresenta estes 4 procedimentos, com seus respectivos números de ordem, nomes, natureza, parâmetros de entrada, parâmetros de saída e funções. Vale ressaltar que apenas os procedimentos 18 e 22 são utilizados na proposta deste trabalho.

[2] Um exemplo de pseudo-falha seria um sistema de arquivos cheio: não permite a gravação momentânea, mas pode ser recuperado. Se ocorrer uma pseudo-falha, o sistema não tratará como falha física e sim como falha lógica (que deve ser tratada em outro nível), portanto não é acionado nenhum dos cenários de falhas definidos neste trabalho. O tratamento das pseudo-falhas, as quais devem ter sido detectadas por um sistema de gerência que providencie mensagens de alerta sobre elas, pode exigir a intervenção do administrador do sistema. Os cenários de falhas deste trabalho não englobam falhas lógicas.

N.	Nome	Natureza	Parâms. Entrada	Parâms. Saída
Função				
18	IMALIVE	Broadcasting	server_id	nenhum
Sinal periódico de sinalização de atividade				
19	REQ_MASTER	Broadcasting	volume_id	req_master_res
Requisição de novo servidor principal a partir dos clientes				
20	ELECTION	Unicasting	election_in	election_out
Pacote de inquirição usado no algoritmo de eleição				
22	REQ_RECUP	Unicasting	server_id	req_recup_res
Usado no algoritmo de recuperação de servidores				

Tabela 1 - Procedimentos adicionados ao protocolo NFS (Extraída de [4]).

5. Estrutura e funcionamento normal do sistema

O detalhamento de aspectos de implementação do sistema é apresentado a seguir, incluindo a função de cada módulo do sistema proposto e o funcionamento integrado destes na ausência de falhas. Assume-se que os aspectos de implementação na replicação de dados seguem o exposto no projeto RNFS [4], considerando a breve descrição e adaptações contidas anteriormente. Elementos como, por exemplo, a utilização de arquivos para registro de requisições de escrita, derivam de idéias contidas no sistema de arquivos HARP [6] e principalmente do HA *Cluster* [1].

O sistema é fundamentado em um servidor duplicado, composto de unidades primária e secundária. Para os clientes da rede, apenas a unidade primária é vista como ativa; a secundária permanece como reserva-quente, mantendo o mesmo conteúdo do disco primário. Se a unidade primária falhar ou parar, a secundária tornar-se-á automaticamente a nova unidade primária. Este processo é transparente para qualquer cliente que esteja usando o servidor mas o sistema está sujeito a um pequeno retardo.

Para a implementação do servidor, o sistema requer dois computadores dotados de configurações mínimas necessárias tais como: capacidade mínima de armazenamento em disco e memória RAM disponível em ambos e tipo de processador empregado. Essa necessidade de configuração mínima visa garantir que não haja queda substancial de desempenho e dificuldades no processamento como um todo. Na verdade, não há restrição de que uma unidade utilize, por exemplo, um processador 286 e outra utilize processador *Pentium*. Entretanto, neste caso, ambas teriam que operar na velocidade da mais lenta. É necessário também que as unidades pertençam ao mesmo ambiente de rede, ou seja, os clientes devem poder enviar pacotes para ambas, utilizando procedimento uniforme.

Cada unidade é ligada por dois tipos de canais de comunicação: um canal representado pelo barramento principal da rede e outro canal representado pelo barramento especial. O barramento principal é utilizado para comunicação com os clientes e para

que as unidades primária e secundária possam trocar informações de estado, entre si, com o objetivo de monitorar as condições de funcionamento do ambiente de rede.

O barramento especial é uma conexão ponto-a-ponto bidirecional usada pelas duas unidades para a sincronização mútua. Informações como requisições de clientes e confirmações de recebimento são passadas de uma a outra através do barramento especial. A sincronização ocorre em dois momentos: a primeira ocorre em freqüência pré-determinada, com o objetivo de sincronizar o relógio lógico das unidades; a segunda, no retorno de uma unidade à sua atividade normal, quando a unidade ativa transfere-lhe o conteúdo do arquivo de *logs*, e ela realiza todas as atualizações referentes ao tempo em que esteve parada.

Devido ao fato de que, em caso de falha de uma das unidades, o outra será mantida na condição de servidor independente (temporariamente), será empregada a nomenclatura de servidor primário e servidor secundário a cada uma das unidades.

5.1 Estrutura básica do sistema

Supondo uma configuração mínima de três máquinas, a organização funcional do sistema compreende os componentes mostrados na figura 2 e definidos no texto a seguir.

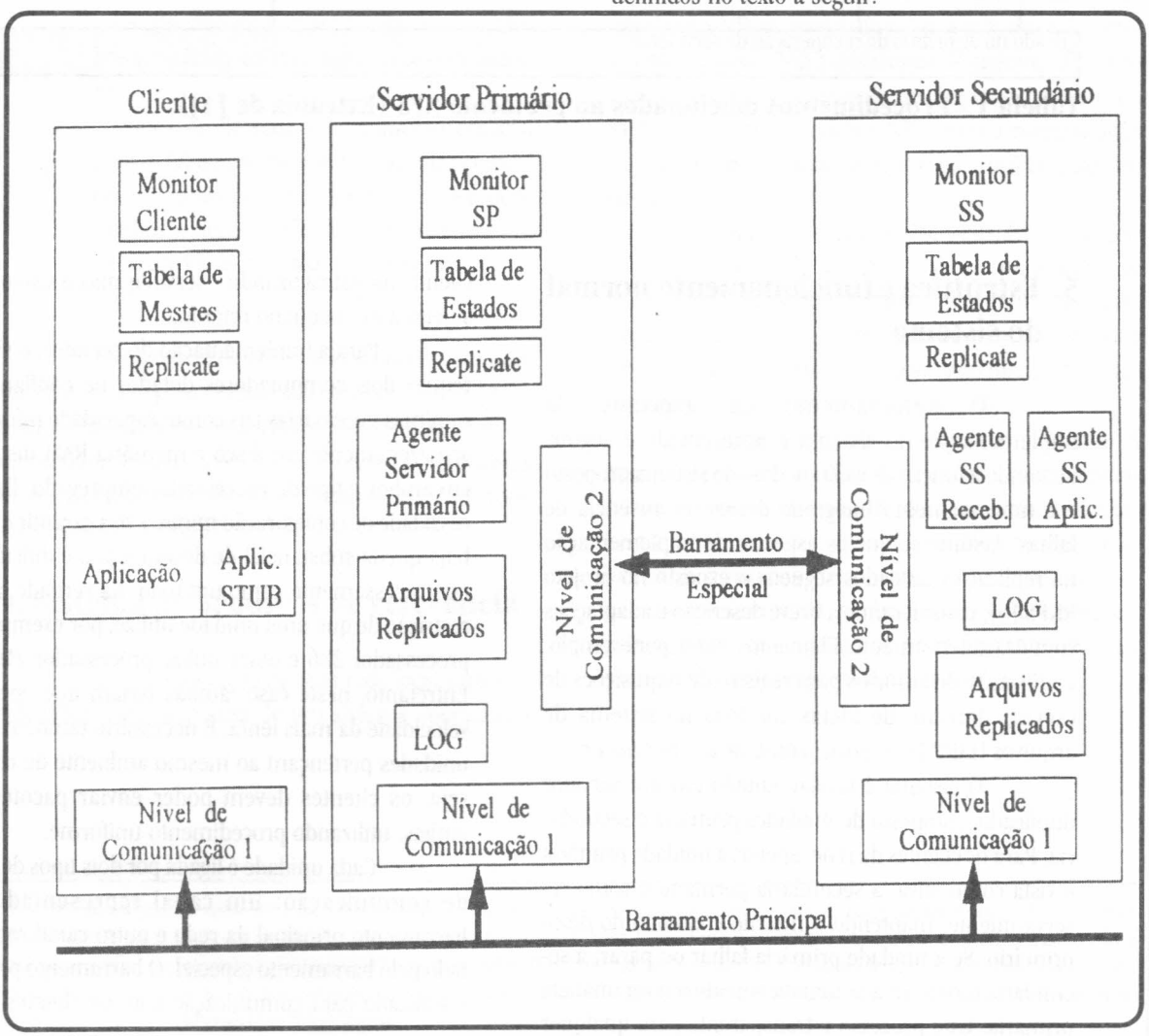

Figura 2 - Organização funcional do sistema

Tabela de estados (TE): existente em ambos servidores, contém o registro referente à situação de cada servidor, ou seja, indica a condição do servidor em um determinado momento: primário (0), secundário (1), falha de *hardware* (2), falha de *software* (3) e em recuperação (4).

Tabela de mestres (TM): contida nos clientes, onde é registrada a situação de cada servidor segundo o ponto de vista dos clientes: primário ou secundário. O cliente deve fazer acessos e enviar as suas requisições de serviço para o servidor classificado na condição de primário. Esse servidor também deve conter o código de primário em sua própria Tabela de Estados.

Arquivos replicados: formam a base de dados sobre a qual o sistema garante alta disponibilidade e proteção contra falhas simples (isoladas). Os arquivos são replicados no servidor secundário a partir do primário, e podem ser acessados a partir de qualquer estação da rede.

Arquivo de *logs*: registra as operações de escrita que chegam dos clientes; será utilizado para fins de recuperação, em caso de falha de servidores. Deve ser limitado a um tamanho que leve em consideração a demanda da rede e não prejudique o seu desempenho. O arquivo de *logs* está presente em ambos servidores, e a serialização das operações de escrita no secundário é assegurada pelo sistema de arquivos distribuído utilizado.

Arquivo *replicate*: utilizado para registrar todos os arquivos replicados existentes, com seus respectivos caminhos de acesso ("*paths*"), e a indicação dos servidores que os contêm (primário e secundário, quando não há falhas).

Aplicação *stub*: é o módulo do sistema que intercepta as requisições das aplicações e as envia para o servidor primário que fornecerá o serviço requisitado. O algoritmo de acesso ao serviço, no núcleo do cliente, é realizado da mesma forma que no NFS para os clientes que estão em atividade, conforme mostra a figura 3. O sistema operacional no cliente repete indefinidamente o acesso, até que este tenha sucesso ou que o processo gerador seja cancelado com um sinal *Kill*.

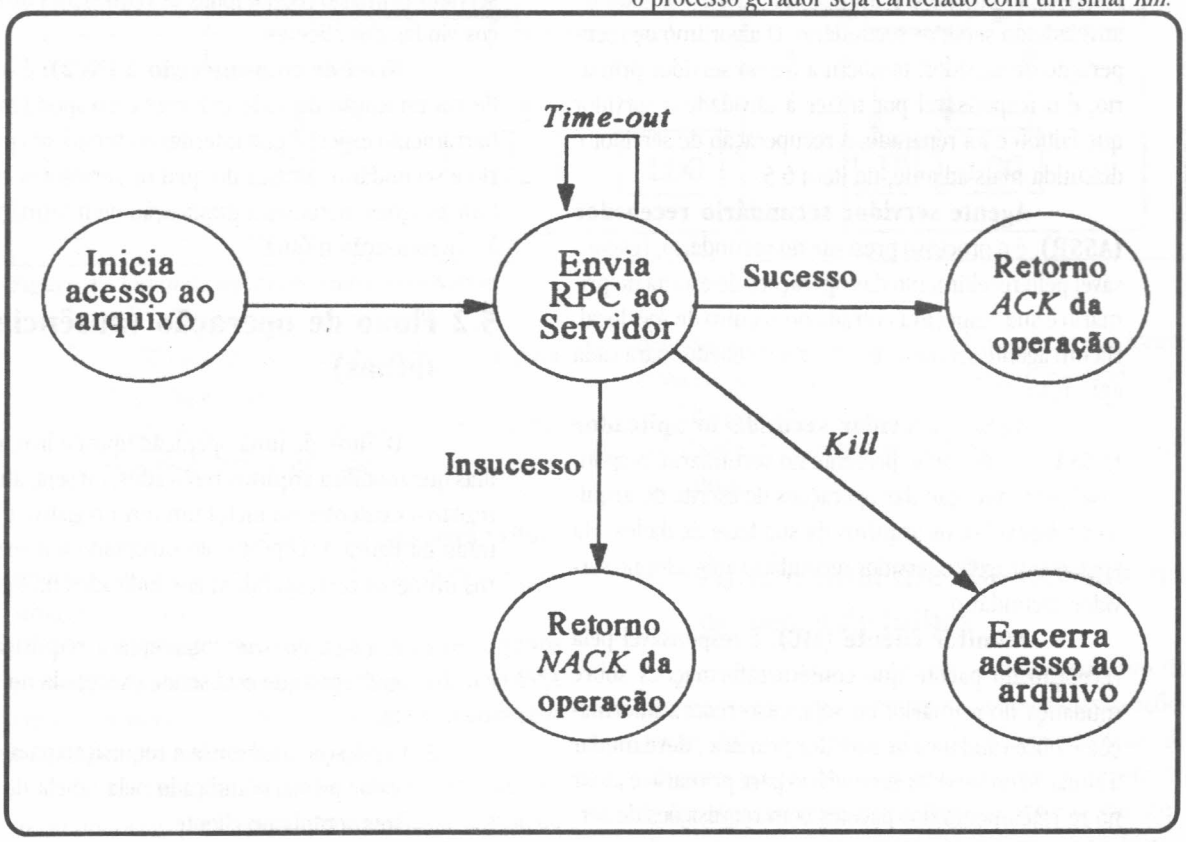

Figura 3 - Acesso NFS no núcleo dos clientes

Agente servidor primário (ASP): é o processo ativo no servidor primário responsável pela execução das requisições enviadas pela aplicação *stub* do cliente. Esse processo também realiza as entradas das operações de escrita no seu arquivo de *logs* e as envia para o servidor secundário. Há um agente para cada aplicação. Neste módulo estão presentes dois algoritmos cooperantes: o algoritmo de repetição de escrita e o de recuperação de servidor.

O algoritmo de repetição de escrita, ativo apenas quando o servidor está no estado de primário, é o responsável pelo espelhamento dos dados para o servidor secundário. Ao receber uma chamada RPC, testa o tipo de operação. Se for leitura, o ASP a executa imediatamente, mas se for uma operação de escrita, é incluída no arquivo de *logs* e feita a substituição dos parâmetros do RPC para o procedimento ser reencaminhado para o servidor secundário. Os valores de retorno de cada procedimento espelhado são anotados e usados para estabelecer pontos de recuperação no arquivo de *logs*, compor o valor final de retorno da resposta ao cliente e para manter na TE o estado de atividade do servidor secundário. O algoritmo de recuperação de servidor, também ativo no servidor primário, é o responsável por trazer à atividade o servidor que falhou e foi reparado. A recuperação de servidor é discutida mais adiante, no item 6.5.

Agente servidor secundário recebedor (ASSR): é o processo presente no secundário, responsável pelo recebimento das operações de escrita do primário e sua respectiva entrada no arquivo de *logs* local. Há um agente servidor secundário recebedor para cada aplicação.

Agente servidor secundário aplicador (ASSA): é o processo presente no secundário, responsável pela execução das operações de escrita do arquivo de *logs* sobre os arquivos da sua base de dados. Há apenas um agente servidor secundário aplicador no servidor secundário.

Monitor cliente (MC): é responsável pela recepção do pacote que contém informações sobre mudança do primário; ou seja, caso receba informações sobre mudança de servidor primário, deve mudar a situação do servidor secundário para primário e atuar no re-roteamento dos pacotes com requisições de serviços.

Monitor servidor primário (MSP): é responsável pela monitoração de toda comunicação vinda do secundário, com o objetivo de detectar possíveis falhas no mesmo e iniciar o processo de tratamento de falhas a partir do servidor primário.

Monitor servidor secundário (MSS): é responsável pela monitoração de toda comunicação vinda do primário, com o objetivo de detectar possíveis falhas nele, e iniciar o processo de tratamento de falhas a partir do servidor secundário. Neste módulo estão presentes os algoritmos cooperantes de sinalização e monitoração de atividade de servidores e de troca de servidor principal. O primeiro é responsável pela emissão do pacote periódico *I'm-alive* para o servidor principal, indicando que se encontra em operação, e o segundo é responsável pelo pacote de dados informando aos clientes sobre a troca de servidor e da necessidade de alterar suas TMs portanto.

Nível de comunicação 1 (NC1): é o nível de comunicação da rede *ethernet* que representa o barramento principal da rede, através do qual o agente servidor primário recebe todas as requisições de serviços vindas dos clientes.

Nível de comunicação 2 (NC2): é o nível de comunicação da rede *ethernet* correspondente ao barramento especial que interliga os servidores primário e secundário, através do qual os servidores executam as ações inerentes à duplicação de informações e à sincronização mútua.

5.2 Fluxo de operação (ausência de falhas)

O fluxo de uma operação típica e livre de falhas que modifica arquivos replicados, ou seja, atualiza registros existentes ou inclui um novo registro, é mostrado na figura 4. O protocolo executado é o seguinte (os números correspondem aos indicados na figura):

1. A aplicação *stub* intercepta a requisição da aplicação que está sendo executada no cliente.
2. A aplicação *stub* envia a requisição para o servidor primário indicado pela tabela de mestres contida no cliente.
3. O ASP recebe a requisição e grava no arquivo de *logs*.

4. O ASP envia cópia da requisição para o servidor secundário e aguarda resposta do ASSR.
5. O ASSR recebe a requisição e grava no arquivo de *logs* local. Em seguida, envia resposta derecebimento correto (*ack*) para o ASP.
6. O ASP recebe a resposta do ASSR e executa a requisição sobre os arquivos replicados locais.

nem são enviadas para o secundário.

Em todas as operações são feitos acessos às tabelas de estado e mestre para que não passe despercebida qualquer alteração no estado de servidores; idem para o arquivo *replicate* onde está configurado o sistema de replicação.

Finalmente, é importante esclarecer que as mensagens recebidas pelo servidor secundário são or-

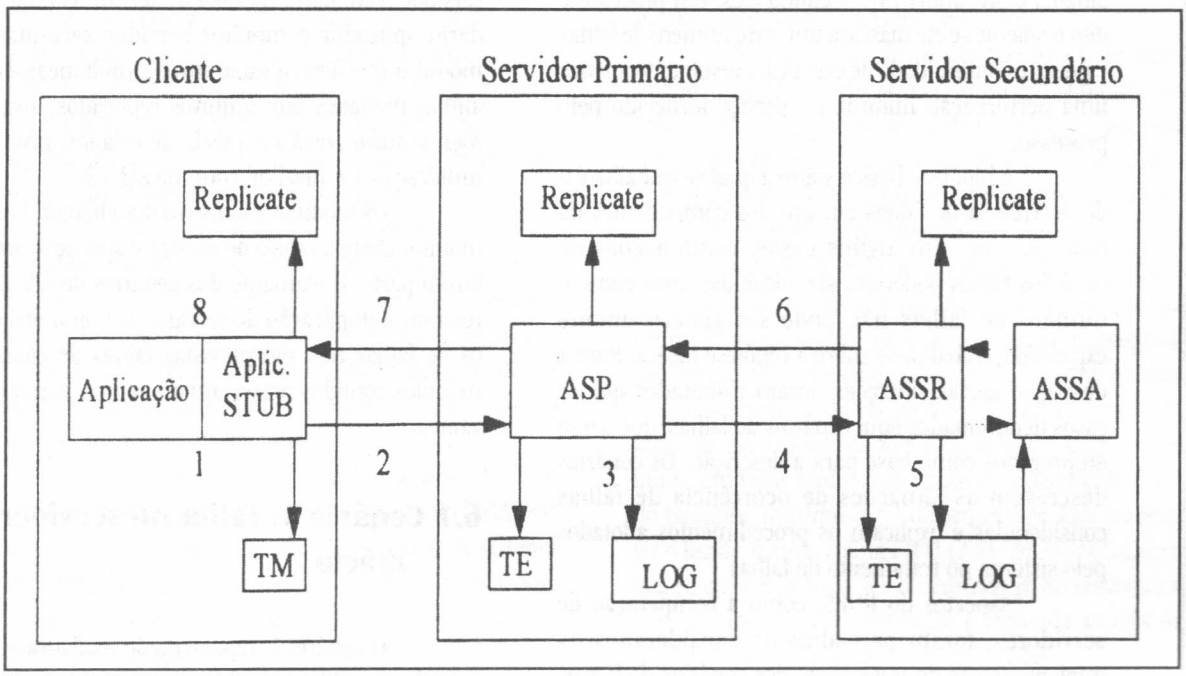

Figura 4 - Fluxo de operação típica do sistema

7. O ASP envia a resposta sobre a requisição para a aplicação *stub*.
8. A aplicação *stub*, de posse da resposta, retorna o controle para a aplicação.

Alguns aspectos adicionais referentes ao fluxo de operação recém apresentado são relatados nos próximos parágrafos.

A execução da requisição sobre os arquivos replicados do servidor secundário é realizada pelo ASSA em *background*, evitando que este procedimento contribua para um retardo da operação.

As operações de leitura não seguem o descrito no protocolo pois, como não modificam a base de dados, elas são executadas imediatamente pelo ASP. Assim, as leituras não são registradas no arquivo de *logs* denadas, ou seja, chegam na mesma ordem que foram enviadas pelo primário. Essa ordenação é realizada com facilidade por existir apenas uma cópia do servidor sincronizada e a forma de difusão de escritas ser centralizada.

6. Funcionamento do sistema na presença de falhas

Nesta seção é descrito o funcionamento do sistema na presença de falhas, sendo relatadas as ações dos módulos monitores, de acordo com hipóteses de falhas consideradas. As ações referentes à recuperação de servidor são abordadas ao final (item 6.5) e são complementares aos procedimentos descritos nos demais itens.

Considera-se que o funcionamento normal do sistema pode ser afetado por falhas no *hardware* das unidades servidoras propriamente ditas, nos adaptadores de LAN e no barramento especial. O ambiente deve ter capacidade de lidar com estas falhas de tal forma que elas não afetem qualquer aplicação do sistema. Os procedimentos devem ser discretos para manter transparência aos usuários. É aceito comportamento resiliente a falhas, na concepção de Singhal e Shivaratri [9]. Segundo eles, um processo é dito resiliente se ele mascara um certo número de falhas e garante continuidade de execução, resultando apenas uma perturbação mínima no serviço fornecido pelo processo.

A hipótese básica sobre a qual se trabalha é a de ocorrência de falhas em um dos componentes da rede por vez. Em alguns casos, combinações de módulos falhos poderiam ser admitidas, mas como o número de falhas não pode ser genericamente expandido, trabalha-se sobre a hipótese básica. Para a descrição destas situações, foram elaborados quatro casos denominados aqui cenários de falhas, que serão empregados como base para a descrição. Os cenários descrevem as situações de ocorrência de falhas consideradas e explicam os procedimentos adotados pelo sistema no tratamento de falhas.

Aspectos do RNFS, como a recuperação de servidores, foram parcialmente considerados na implementação do tratamento dos cenários de falhas. Os cenários de falhas tomados por base são os mesmos previstos no *NetWare SFT III* [7], contudo diferem daqueles nas decisões de implementação.

O funcionamento do sistema na presença de falhas consiste na detecção e tratamento de cada um dos cenários utilizando módulos adicionados ao ambiente de rede local Ethernet conforme descritos no item 5.1. Todos os módulos estão presentes nos servidores considerando que ambos podem, a seu tempo, assumir o papel de primário ou secundário, de acordo com o cenário de falha. Alguns módulos são excludentes em relação a outros, isto é, caso um esteja ativo, o seu par permanece em estado de latência. A ativação ou latência de um módulo é definida pelo estado do servidor na Tabela de Estados, ou seja, caso seu estado indique, por exemplo, que é um servidor primário, determinados módulos devem estar ativos e outros latentes.

Os módulos que devem estar ativos somente na unidade primária são: agente servidor primário e monitor servidor primário. Os módulos que devem estar ativos somente na unidade secundária são: agente servidor secundário recebedor, agente servidor secundário aplicador e monitor servidor secundário. Os módulos que devem estar ativos simultaneamente em ambas unidades são: arquivos replicados, arquivo de *logs*, arquivo *replicate*, tabela de estados, nível de comunicação 1 e nível de comunicação 2.

Os módulos exclusivos dos clientes, tais como monitor cliente, tabela de mestre e aplicação *stub*, não fazem parte diretamente dos cenários de falhas e são restritos à duplicação de servidores. Como nos cenários de falhas não são previstas falhas de clientes, os módulos contidos nestes atuam apenas como coadjuvantes.

6.1 Cenário 1: falha no servidor primário

O cenário 1, representado na figura 5, ocorre quando o servidor primário pára ou é desconectado da rede completamente. A detecção deste cenário é descrito, a seguir.

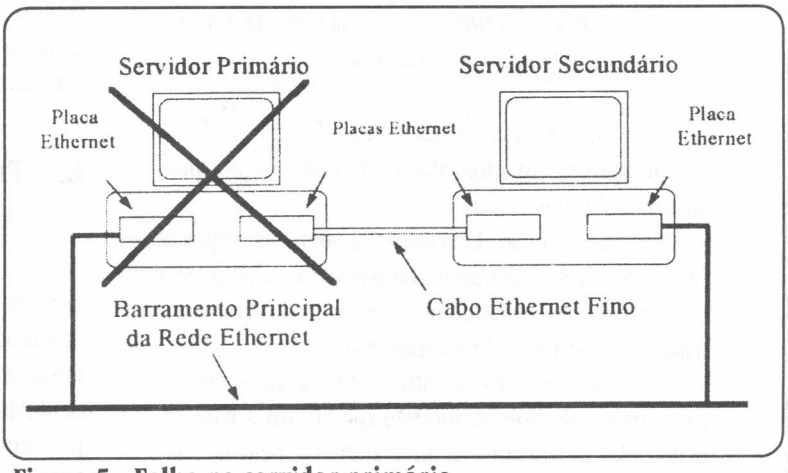

Figura 5 - Falha no servidor primário

Os servidores primário e secundário testam-se mutuamente com periodicidade pré-estabelecida de 2s. Os módulos responsáveis por esse teste são os monitores MSP e MSS. O teste é realizado através do procedimento denominado *I'm-alive*, adicionado ao protocolo NFS para compor o protocolo RNFS, com a função de sinalização de atividade. O *I'm-alive* é enviado através de ambos níveis de comunicação (NC1 e NC2).

Apesar do prazo previsto para o recebimento de um *I'm-alive* ser de 2s, o MSS toma por base um tempo maior para a suspeita de falha do servidor, pois pode haver problemas de retardo no servidor primário ou na linha de comunicação. Assim que o tempo de inatividade atinge o máximo de 5s no NC2, o MSS observa a atividade no NC1, considerando os mesmos 5s.

O MSS pode detectar duas situações no NC1: uma das situações é a detecção de atividade normal, ou seja, o *I'm-alive* está chegando dentro do tempo previsto, concluindo que o barramento especial falhou (esta situação será vista em detalhes mais adiante). A outra situação é detectar t = 0 (*time-out*), concluindo que o servidor primário falhou.

O controle de prazo máximo (5s) é implementado através do sinal denominado SIGALRM que dispara um relógio através da chamada de sistema denominada *alarm*. A chamada é referenciada por: *unsigned int alarm (unsigned int* ***sec)****;* onde ***sec*** representa o tempo máximo, a partir do qual é feita a contagem regressiva. A cada *I'm-alive* recebido é atribuído o valor 5 para *sec*. O *time-out* ocorre quando ***sec*** = 0.

Ao concluir que o servidor primário falhou, o MSS executa as ações, a seguir listadas, para o tratamento da falha.

• **Ação 1:** Avisa o administrador do sistema, através do terminal, sobre a falha do servidor primário e que ele, o servidor secundário, está assumindo o papel de primário. Esse aviso é feito através de uma função *printf*().

• **Ação 2:** Envia um pacote para os clientes informando que ele agora é o novo servidor primário e a rota de envio de requisições deve ser mudada. O pacote enviado contém o endereço IP do novo servidor primário. O MC recebe o pacote e muda o estado dos servidores na TM. O antigo primário passa para o estado 2 (servidor em falha física) e o secundário passa para o estado 0 (servidor principal).

O MC usa o endereço IP enviado pelo servidor secundário como argumento para mudar o estado dos servidores na TM. Ao receber o pacote, compara o endereço IP com os dois únicos endereços de servidores contidos na tabela: atribui o estado 0 ao que tiver o mesmo endereço IP; atribui ao outro, por exclusão, o estado 2. Assim, ao interceptar requisições para o primário e consultando a TM, a aplicação *stub* roteará os pacotes para o novo servidor primário.

As requisições do cliente que foram enviadas quando o servidor primário já estava em falha não receberam confirmação de recebimento; assim, são reenviadas agora para o novo servidor primário, pois o *software* cliente detecta a mudança de servidor primário na TM ao tentar a retransmissão.

• **Ação 3**: A partir do momento em que detectou a falha, bloqueia o arquivo de *logs* para que nenhum registro seja descartado, prevendo a recuperação futura do servidor que falhou. Em operação normal, o descarte de dados seria realizado em função do tamanho do arquivo de *logs* e da ação do ASSA. É atribuído um valor limite *n* de controle, equivalente a 2/3 do tamanho máximo (t_{max}) do arquivo de *logs*. Ao atingir o limite *n*, uma rotina FIFO (*First In First Out*) verifica se cada registro a ser descartado já foi gravado na base de dados. Essa verificação é efetuada através de marcas (*commit points*) feitas pelo ASSA.

O bloqueio ocorre fazendo $n = t_{max}$, considerando paradas curtas. Caso o número de registros se aproxime de *n*, a solução é abandonar o arquivo de *logs* e empregar a solução do número de versões. Na pior hipótese, de parada demasiadamente longa, a transferência de volumes deve ser utilizada.

A descrição do funcionamento deste cenário pode ser resumido no algoritmo descrito a seguir:

1. O MSS detecta inatividade excessiva (*time-out*) no NC2 e no NC1 (sempre nesta ordem), concluindo que o servidor primário falhou.
2. O MSS avisa ao administrador do sistema, através do terminal, que o primário falhou e que ele, o servidor secundário, está tomando o papel do primário.
3. O MSS envia um pacote para os clientes, infor-

mando que o estado dos servidores na TM deve ser invertido, assim todas as requisições que não receberam respostas com confirmação de recebimento pelo servidor primário e que são reenviadas, serão re-roteadas.

4. O MSS bloqueia o arquivo de *logs*, não permitindo o descarte de qualquer registro, a partir daquele momento.

Duas observações devem ser feitas em relação ao tempo previsto para a sinalização de atividade da rede e para a ocorrência de *time-out* sobre o cenário 1:

1. O tempo considerado é o tempo real, ou seja, não considera o tempo da CPU.
2. Ao testar o NC1, o MSS detectará um instante de tempo t, onde $0 \leq t \leq 5$.

O objetivo da primeira observação é esclarecer que o tempo considerado é o chamado "*wall clock time*" [11], ou seja, o tempo de um relógio independente do relógio da máquina onde o MSS está sendo executado; portanto os tempos de NC1 e NC2 podem ser diferentes.

A segunda observação serve como reforço à primeira e para indicar que *t* não necessariamente possui o mesmo valor em NC1 e NC2.

As observações feitas demonstram que, apesar do procedimento *I'm-alive* ser enviado pelo monitor simultaneamente para NC1 e NC2, não significa que os níveis de comunicação correspondentes no outro servidor vão recebê-lo no mesmo instante de tempo *t*. A explicação está no fato do *I'm-alive* enviado para NC1 ter de concorrer ao meio de transmissão com o tráfego normal da rede.

• A ação do servidor secundário de enviar um pacote de dados informando os clientes sobre a mudança do servidor primário através do MSS, é feita através de uma conexão do tipo *socket*, utilizando a opção *SO_BROADCAST* do tipo *internet SOCK_DGRAM* que habilita o sistema a enviar mensagens em *broadcast* [11]. Vale ressaltar que mensagens em *broadcast* só são possíveis em sistemas que utilizam pacotes de dados do tipo *datagram sockets* (serviço sem conexão) e em redes que suportam o conceito de mensagem em *broadcast* como, por exemplo, a rede *Ethernet* [2]. Uma mensagem em *broadcast* na rede *Ethernet* é especificada através da atribuição do valor 1 para todos os bits do campo que contém o endereço do destinatário [3].

6.2 Cenário 2: falha no servidor secundário

O cenário 2, representado na figura 6 e descrito a seguir, corresponde a falha no servidor secundário; configura-se quando o servidor primário deixa de receber confirmação de recebimento ou *I'm-alive* do servidor secundário.

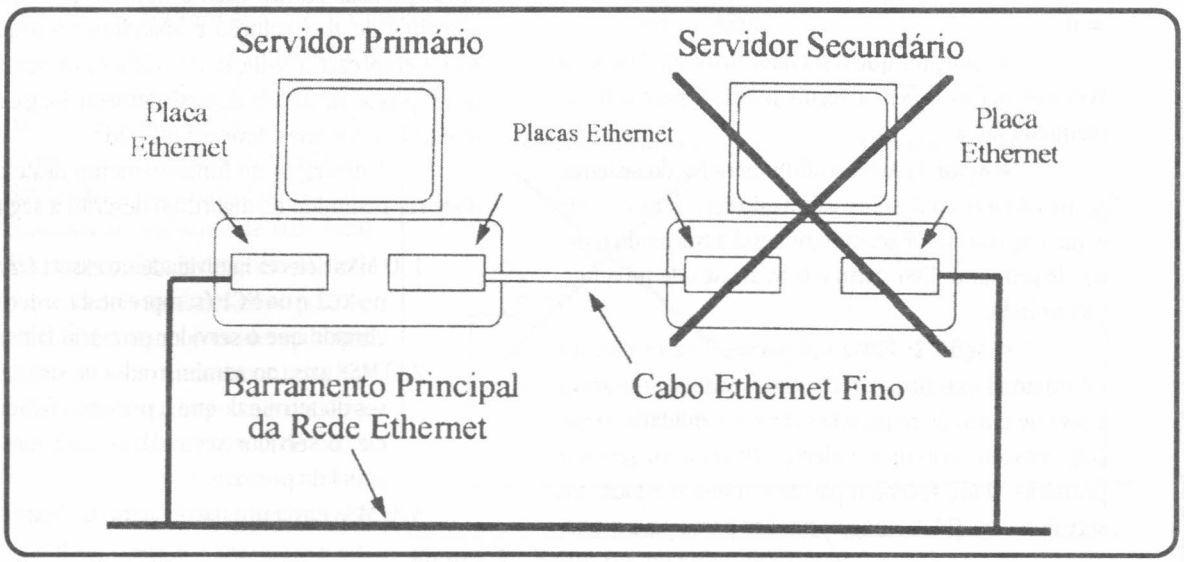

Figura 6 - Falha no servidor secundário

Durante o teste mútuo entre o MSP e o MSS, o MSP deixa de receber o *I'm-alive* do servidor secundário dentro do tempo máximo especificado de 5s. O MSP então conclui que o servidor secundário falhou.

Observa-se que a conclusão sobre a falha do servidor secundário é mais rápida em relação a situação inversa (detecção de falha do servidor primário pelo secundário), mas a explicação é simples. Quando o MSP detecta *time-out* no NC2, ele não necessita testar o NC1 por ser o servidor ativo, ou seja, se ele está atendendo às requisições dos clientes normalmente, significa que não existe nenhum problema com o NC1. A confirmação de chegada do *I'm-alive* no nível de comunicação NC1 seria desnecessário, por não haver urgência justificada de ação, em relação a um servidor reserva-quente, em detrimento ao desempenho no atendimento das requisições dos clientes.

Por outro lado, o perfeito funcionamento também do NC2 (o servidor primário consegue enviar informações normalmente) garante que o problema não é do barramento especial, confirmando assim a ocorrência do cenário 2. Após a conclusão sobre a falha do servidor secundário, o MSP executa as seguintes ações:

- Avisa o administrador do sistema sobre a falha do servidor secundário utilizando uma função *printf*().
- Bloqueia o arquivo de *logs*, não permitindo que registros sejam descartados. O limite *n*, a exemplo do cenário 1, também passa ser igual a t_{max} e as soluções de recuperação, caso o tamanho do arquivo de *logs* se aproxime de *n*, também devem ser consideradas.

Vale ressaltar que, pelo fato de não haver troca de servidor neste cenário, o início do tratamento da situação de falha não ocasiona nenhum retardo no desempenho da rede; portanto a ação é absolutamente transparente para a aplicação e para o usuário.

A descrição do funcionamento deste cenário pode ser resumido no algoritmo descrito a seguir:

1. O MSP detecta inatividade excessiva (*time-out*) no NC2, concluindo que o servidor secundário falhou.
2. O MSP avisa ao administrador do sistema, através do terminal, que o secundário falhou.
3. O MSP bloqueia o arquivo de *logs*, não permitindo o descarte de qualquer registro a partir daquele momento.

6.3 Cenário 3: falha do barramento especial

O cenário 3, representado na figura 7, corresponde a falha no barramento especial; sua ocorrência configura-se quando é detectada inatividade excessiva neste barramento. As formas de detecção e tratamento, bem como as peculiaridades do caso, são descritas a seguir.

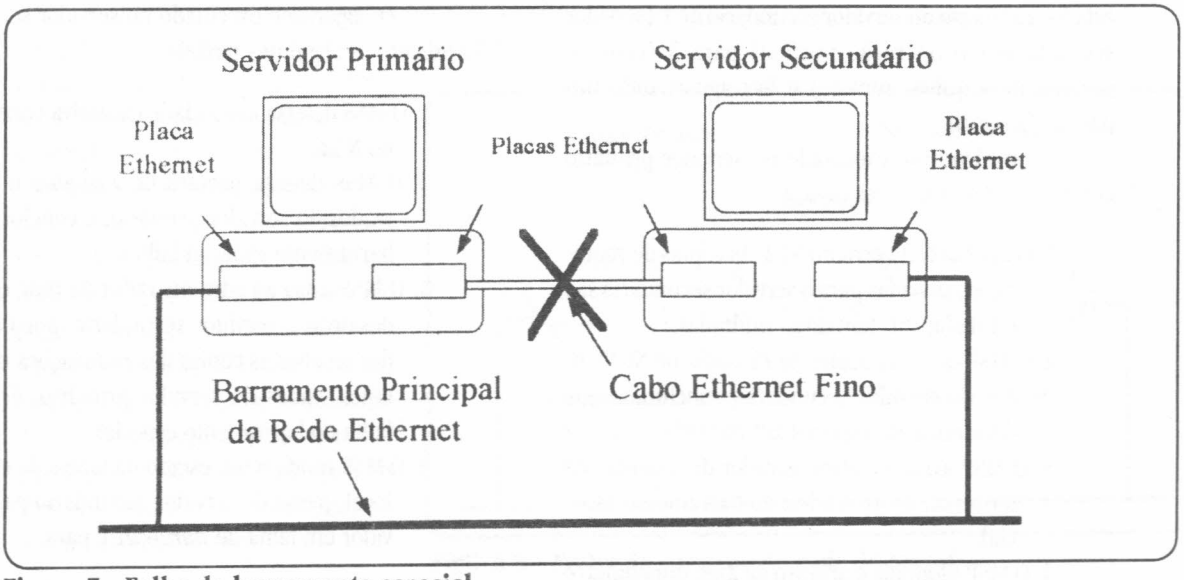

Figura 7 - Falha do barramento especial

A detecção deste cenário é diferenciada em relação aos outros, pois ocorre em ambos servidores primário e secundário. Esta dupla ocorrência, entretanto, não significa que a detecção ocorra no mesmo instante de tempo *t*. A seguir, são abordados os aspectos de implementação específica de cada um dos servidores.

Detecção no servidor primário

O MSP detecta problema no NC2 por receber sinalização de erro que pode vir diretamente do NC2, como resposta à tentativa de enviar o procedimento *I'm-alive*, ou
vir do ASP que, ao tentar enviar cópias das requisições para o servidor secundário, teve a ação refutada em repetidas tentativas.

Em seguida, o MSP testa o NC1 e detecta pacote de *I'm-alive* chegando do servidor secundário, concluindo que o barramento especial falhou. A partir deste ponto, o tratamento passa a ser idêntico ao usado no cenário 2, ou seja, o MSP avisa o administrador da rede que não pode mais fazer a replicação dos dados devido à falha e impede que, a partir daquele momento, registros sejam descartados do seu arquivo de *logs*. Em outras palavras, o servidor primário considera que está havendo problemas no sistema de arquivos remotos por não estar conseguindo fazer a replicação de dados. Assim, o servidor primário muda, na sua tabela de estado, a situação do servidor secundário de 1 (servidor secundário em operação normal) para 3 (erro no sistema de arquivos remoto) e fica aguardando um pacote *I'm-alive*.

O algoritmo executado no servidor primário compreende os seguintes passos:

1. O MSP detecta erro no NC2. As cópias de requisições enviadas para o servidor secundário são refutadas em tentativas múltiplas.
2. O MSP detecta pacotes de *I'm-alive* no NC1 vindos do servidor secundário, concluindo que o barramento especial falhou.
3. O MSP avisa ao administrador do sistema, via terminal, sobre a falha do barramento especial.
4. O MSP bloqueia o arquivo de *logs*, impedindo o descarte de qualquer registro do arquivo, a partir daquele momento.

Detecção no servidor secundário

O início da detecção no servidor secundário é idêntico ao do cenário 1, onde o MSS detecta inatividade excessiva (*time-out*) no NC2. A diferença é que ao testar o NC1, o MSS detecta procedimento *I'm-alive* chegando do servidor primário, concluindo assim que o barramento especial falhou.

Após a conclusão sobre a falha do barramento especial, o MSS age de maneira diferente do MSP. Ao avisar o administrador sobre a falha do NC2, informa-o também que está modificando o seu estado na TE local de 1 (servidor secundário em operação normal) para 2 (servidor em falha física), por não estar recebendo as cópias das requisições do servidor primário e portanto por não poder mais continuar sendo considerado como a duplicação do servidor primário.

A parada do secundário faz com que seja validada a ação realizada do servidor primário ao detectar a falha do barramento especial, ou seja, se o servidor secundário está em falha física e o primário considera que o sistema de arquivo remoto está com problemas, então a ação resultante é a mesma: o servidor primário continua agindo como tal (sem descartar registros do arquivo de *logs*) e o servidor secundário pára até que o barramento especial seja reparado.

O algoritmo executado no servidor secundário executa os seguintes passos:

1. O MSS detecta inatividade excessiva (*time-out*) no NC2.
2. O MSS detecta pacotes de *I'm-alive* no NC1, vindos do servidor primário, e conclui que o barramento especial falhou.
3. O MSS avisa ao administrador da rede que vai desativar o servidor secundário, por não poder receber as cópias das requisições dos clientes vindas do servidor primário, devido à falha do barramento especial.
4. O MSS muda o seu estado na tabela de estados local; passa de servidor secundário para servidor em falha de *hardware* e pára.

Observações complementares sobre o cenário 3

Cabe ressaltar que os algoritmos executados nos servidores primário e secundário são totalmente independentes entre si. Os servidores chegam ao mesmo diagnóstico (falha do barramento especial) por visões individuais da rede.

Outra observação importante é que a falha do barramento especial pode ser detectada primeiro pelo servidor primário ou pelo servidor secundário, sendo que a detecção pelo primário é mais vantajosa. A grande vantagem na detecção pelo primário é que, ao tomar a iniciativa de avisar o administrador e principalmente de bloquear o arquivo de *logs*, ele se mantém na condição de primário, evitando a troca de servidor e mantendo a transparência da operação de tratamento. Por outro lado, se o servidor secundário for o primeiro a detectar a falha, os diagnósticos dos dois servidores podem ser diferentes, deixando sob a responsabilidade do administrador do sistema a decisão sobre o cenário de falha real, o que é indesejável.

Os diagnósticos resultantes da detecção de falha no barramento especial pelo servidor secundário antes do primário são os seguintes:

1) Diagnóstico do servidor primário: falha do servidor secundário (cenário 2).
2) Diagnóstico do servidor secundário: falha do barramento especial (cenário 3).

O diagnóstico do primeiro a detectar a falha (servidor secundário) é o correto. O servidor primário é induzido ao erro da seguinte forma: ao detectar o erro no barramento especial, o servidor secundário avisa corretamente ao administrador sobre o erro e em seguida muda o seu estado na TE de 1 (servidor secundário em operação normal) para 2 (servidor em falha física), deixando assim de enviar pacotes *I'm-alive* pelo barramento principal. Quando o servidor primário detecta *time-out* no NC2, testa o NC1 em seguida, não recebendo pacotes *I'm-alive* vindos do secundário e conclui que este falhou.

Deste modo, o administrador terá duas mensagens diferentes nos terminais dos servidores primário e secundário. Para decidir corretamente, basta que o administrador identifique a mensagem que foi enviada primeiro. Essa identificação pode ser feita comparando-se os horários que as mensagens foram enviadas, daí uma necessidade de sincronização lógica periódica entre os servidores.

O servidor secundário deve ser desligado durante o período de inatividade para que, após o reparo do barramento especial, ele execute o algoritmo de recuperação na religação.

6.4 Cenário 4: falha de conexão de rede do servidor primário

O cenário 4, representado na figura 8 e descrito na seqüência, ocorre quando é detectada atividade anormal na conexão de rede NC1. Esta detecção pode manifestar-se através do excesso de pacotes descartados e pacotes recebidos com tamanho acima ou abaixo do normal.

O MSP detecta atividade anormal na atividade do NC1 através de um sinal de interrupção de um sistema de gerência. A atividade anormal pode ser detectada através de sistemas de gerência de diagnósticos, que usam multímetros ou TDR (*Time Domain Reflectrometry*) para detectar falhas no nível físico. As taxas consideradas como atividade normal dependem do sistema de gerência utilizado.

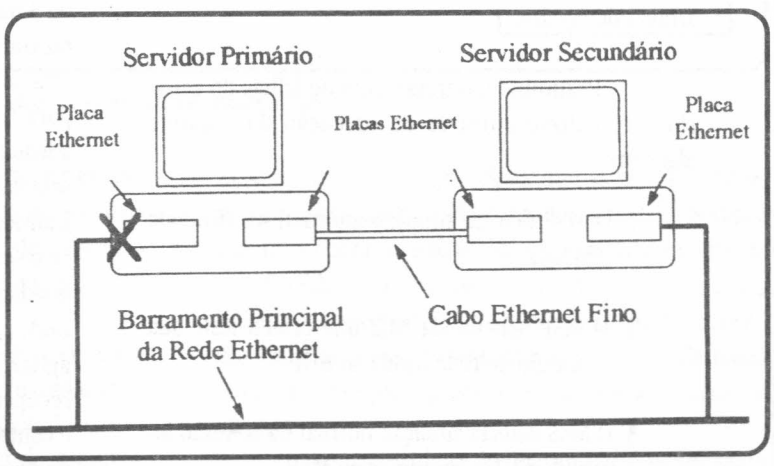

Figura 8 - Falha de conexão de rede do servidor

O subprojeto MAD (Módulo de Atualização de Diagnóstico) [8] integrante do projeto CINEMA, que está sendo desenvolvido pelo Grupo de Redes de Computadores da UFRGS, considera, por exemplo, que uma rede está com atividade anormal, quando a taxa de erros está acima de 2% e a taxa de colisão está acima de 1%, ou quando o percentual de pacotes descartados está acima de 1%.

Após a detecção de atividade anormal na conexão local de rede, o MSP solicita ao MSS que teste também a sua conexão local. Neste ponto, podem ocorrer duas situações:

- Se o MSS também detectar situação anormal na atividade da rede, ele avisa o MSP e ambos avisam o administrador que a rede está com problemas e que vão interromper o atendimento aos clientes. Neste caso, o servidor primário e secundário mudam seus estados na TE para o valor 3, correspondente à condição "erro no sistema de arquivos remoto". O administrador do sistema assume o controle da rede.

- Se o MSS não detectar situação anormal na atividade da rede, o MSP avisa o administrador que suas conexões e cabeamento devem estar com problemas e que por isso vai mudar seu estado na TE de 0 (servidor primário) para 3 (erro no sistema de arquivos remoto). Assim que ocorre a alteração do estado para o valor 3, o ASP deixa de receber requisições dos clientes e consequentemente não as envia para o servidor secundário. O MSS detecta inatividade excessiva no NC2 e então inicia o procedimento de execução descrito através do cenário 1.

Resumindo, o tratamento de falhas do cenário 4 é realizado através da execução do seguinte algoritmo:

1. O MSP detecta situação anormal na atividade do NC1.

2. O MSP solicita via NC2 que o MSS teste sua conexão de rede ligada ao NC1.

3. O MSS detecta situação normal na conexão local do NC1 e comunica ao MSP.

4. O MSP avisa ao administrador da rede que está com problemas na conexão de rede local e portanto vai parar.

5. O MSP modifica seu estado na tabela de estados local e pára.

O final da execução deste algoritmo força o sistema a executar o cenário 1, que trata de falha no servidor primário. É importante ressaltar que, caso o MSS detecte o mesmo problema na conexão de rede do servidor secundário, ambos servidores avisam o administrador do sistema e param sua atividade. Neste caso, a rede necessita de manutenção e o administrador passa a ser o responsável pela sua volta à atividade normal.

6.5 Recuperação de servidor

A recuperação de servidor tem por objetivo atualizar a unidade que falhou e que está retornando à atividade, para retomar a característica de servidor duplicado. Esta recuperação ocorre a partir do momento que o servidor em falha é reparado e religado. Deve ser feita incondicionalmente a partir do servidor que está na condição de primário, pois neste caso será a única unidade ativa da rede local.

A recuperação começa pela execução de um protocolo de inicialização executado entre o servidor principal e o secundário (a ser recuperado).

Ao ser religado, o servidor possui o estado 2 ou 3 na TE, ou seja está em estado de falha. A religação faz com que o servidor envie procedimento de *I'm-alive* através de NC1 e NC2. Ao receber o pacote, o MSP envia para o servidor emissor uma chamada REQ_RECUP, adicionada pelo RNFS ao NFS, que possui a função de solicitar a concordância do servidor que está voltando à atividade, para a entrada no algoritmo de recuperação. Se o servidor concordar (*ack*), a recuperação é iniciada imediatamente. Caso o servidor discorde (*nack* ou *time-out*), é mantido seu estado anterior na tabela e o MSP aguardará até o próximo *I'm-alive* para tentar a recuperação novamente. O algoritmo de concordância e controle das etapas na recuperação é esquematizado na figura 9, extraída de [4].

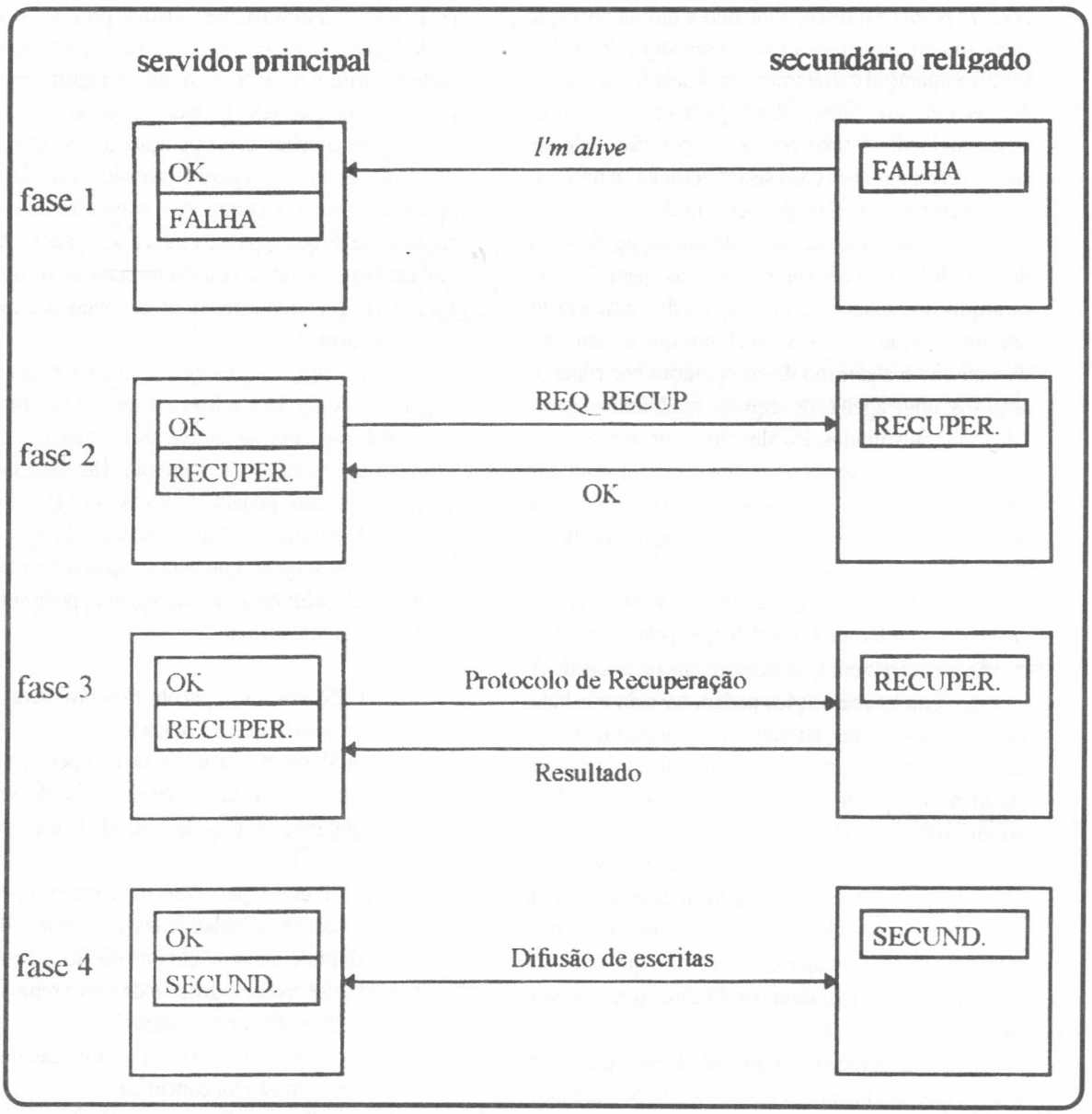

Figura 9 - Concordância e controle das etapas na recuperação

Após o algoritmo de concordância ser executado, o ASP envia o conteúdo do seu arquivo de *logs* para o ASSR. O servidor muda seu estado de 2 ou 3 na TE para 4 (servidor está tendo seu conteúdo recuperado).

Considerando que este trabalho leva em consideração um ambiente de rede local com paradas curtas, o método de **retenção de *logs*** é o ideal. Este método foi proposto no projeto de sistema de arquivos HARP [5], tendo por idéia básica a alocação de espaço no disco do servidor principal, que sirva como *cache* de operações (organizada como fila circular). Sempre que uma operação não puder ser realizada em virtude de uma falha, ela será registrada em *cache* (com todos os seus parâmetros); após a falha ser corrigida, o servidor principal envia à unidade a ser recuperada todas essas informações gravadas para que a mesma se atualize.

Na necessidade de determinar o ponto exato de falha do servidor secundário, o sistema mantém

commit points em disco, anotando a última operação corretamente transmitida (para o servidor principal) e a última operação corretamente aplicada (para o servidor secundário). Estes *commit points* necessitam de armazenamento estável pois a recuperação pode ser sincronizada por eles. Caso sejam perdidos, é necessário utilizar outro método de recuperação.

O tamanho da *cache* de operações deve ser definido de forma a não comprometer as operações com os arquivos regulares. Caso o espaço da *cache* esgote, entram em ação os outros algoritmos que devem estar disponíveis: o algoritmo de recuperação por cópia de arquivos, onde apenas os arquivos modificados após a falha são transmitidos, e o algoritmo por transferência de volumes, onde todos os arquivos são transmitidos. Vale ressaltar que este último algoritmo deve ser usado apenas em recuperações muito extensas, incomuns em redes locais.

Considerando que a falha no servidor primário ocorreu no tempo t_0 e a detecção pelo servidor secundário ocorreu em t_1, conclui-se que no intervalo $\Delta t = t_1 - t_0$ algumas requisições podem ter sido recebidas pelo primário e não enviadas ao secundário. Assim, antes que ocorra a atualização do arquivo de *logs* do secundário, deve ser realizado um retorno dentro deste arquivo para um estado consistente.

O retorno para um estado consistente é realizado com base em um contador incluído no arquivo de *logs*, que tem por objetivo registrar o número de mensagens incluídas no arquivo de *logs*, mas que não receberam o *ack* do secundário, significando que o mesmo não as recebeu.

O ASSR recebe o arquivo de *logs* e repassa para o ASSA, que não inicia de imediato a recuperação baseada nas informações deste arquivo. Ele inicialmente acessa o arquivo de *logs* local para verificar o ponto, indicado pelo contador, que representa a última mensagem enviada com sucesso para o servidor secundário. O número de seqüência desta mensagem é comparado com o número de seqüência do arquivo de *logs* recebido após a recuperação. A operação estará correta se o número de seqüência do arquivo de *logs* recebido for igual ao número de seqüência do último registro válido do arquivo de *logs* local acrescido de uma unidade. Em seguida, ele processa todo o arquivo de *logs* local até o ponto indicado pelo contador (os registros posteriores são descartados). Depois processa o arquivo de *logs* vindo do servidor primário, a partir do ponto onde foi armazenado como o último registro recebido e gravado corretamente na base de dados.

Ao concluir a recuperação, o ASSA modifica o estado do servidor de 4 para 1 (servidor secundário em operação normal) e comunica a conclusão da recuperação ao ASSR que repassa a informação para o ASP. O ASP também modifica o estado do servidor recuperado na sua TE, voltando a repassar as cópias das requisições dos clientes.

Por fim, vale ressaltar que a operação de recuperação de servidor é feita através de um processo especial de recuperação (através da chamada *exec*), que também monitora sua terminação. Detalhes do processo recuperador podem ser vistos em [4].

A descrição do funcionamento da ação de recuperação do servidor que falhou, após ser reparado e religado pelo administrador do sistema, pode ser resumida da seguinte forma:

1. O MSS envia um pacote *I'm-alive* para o MSP indicando que foi religado.
2. O MSP inicia o processo de recuperação com a execução do protocolo de aceite do servidor que está retornando a atividade e a sincronização dos mesmos.
3. O ASP envia, para o servidor em recuperação, todas as entradas feitas no arquivo de *logs* durante o tempo em que ele ficou parado.
4. O ASSR recebe o arquivo de *logs* e repassa para o ASSA efetuar a atualização.
5. O ASSR recebe do ASSA a informação de que a recuperação foi concluída.
6. O ASSR comunica a conclusão da recuperação para o ASP e modifica a sua tabela de estados, na qual passa da condição de servidor em recuperação para servidor secundário.
7. O ASP efetua também a modificação da sua tabela de estados, reconhecendo a volta à atividade do servidor secundário.

7. Conclusões

A implementação da proposta deste trabalho partiu do pressuposto que a replicação de dados já es-

taria operacional, através do RNFS, sendo necessária apenas a realização das adaptações devidas. Portanto, centrou-se na introdução de processos monitores capazes de detectar falhas em rede local com servidor duplicado e tratá-las de acordo com cada cenário diagnosticado.

Os processos monitores deveriam ser desenvolvidos e incluídos no núcleo do sistema operacional Linux. Mas considerando o crescimento significativo da complexidade de implementação, optou-se por executá-los como processos simulados no nível do usuário. Esta opção permite identificar o funcionamento dos módulos, avaliando suas características funcionais no cumprimento das especificações. Entretanto, perde-se em desempenho, cuja repercussão real sobre sistema não pode ser avaliada imediatamente a partir desta implementação. Como prosseguimento, a opção de implementação deverá ser modificada a fim de reduzir o impacto causado sobre o desempenho do sistema.

Bibliografia Consultada:

AZAGURY, A. et al. Highly Available Cluster: A Case Study. In: Annual International Symposium on Fault-Tolerant Computing, 24., Austin: Digest of Papers; IEEE Computer Society Press, 1994. p.404-413.

BLOOMER, J. *Power Programming with RPC.* Sebastopol, CA: O'Reilly & Associates, 1992. 486p.

FREEMAN, R. *Practical Data Communications.* New York: John Wiley & Sons, 1995. 632p.

LEBOUTE, M. *RNFS - Um sistema de arquivos distribuído tolerante a falhas para o UNIX.* Porto Alegre: CPGCC/UFRGS, 1995. (Dissertação)

LISKOV, B. et al. *A Replicated Unix File System: Operating Systems Review.* New York, v.25, n.1, p.60-64, Jan. 1991.

LISKOV, Barbara et al. *Replication in the Harp File System.* Operating Systems Review, New York, v. 25, n. 5, p. 226-238, 1991.

NOVELL. *Netware System Fault Tolerance Level III.* An in-Depth Look. Provo, Utah - USA: Novell, 1993. 19p. v3

NUNES, C. *Um discriminador inteligente de eventos de rede para o ambiente CINEMA.* Porto Alegre: CPGCC/UFRGS, 1997. (Dissertação)

SINGHAL, M.; SHIVARATRI, N. *Advanced Concepts in Operating Systems:* Distributed, Database, and Multiprocessor Operating Systems. New York: McGraw-Hill, 1994. 522p.

SOARES, L.F.; LEMOS, G.; COLCHER, S. *Redes de Computadores. Das LANs, MANs e WANs às redes ATM.* Rio de Janeiro: Campus, 1995. 576p.

STEVENS, R. W. *UNIX Networking Programming.* Englewood Clifs, NJ.: Prentice-Hall, 1990. 772p.

TANENBAUM, A. *Modern Operating Systems.* Englewood Cliffs, NJ.: Prentice-Hall, 1992. 728p.