UM SERVIDOR DE ARQUIVOS COM TRANSAÇÕES ATOMICAS E DIRETORIOS PARA REDES DE COMPUTADORES.

José Edgard Soares Júnior José Marcos Silva Nogueira

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Caixa Postal 702 - Fone (031) 443.4088
Belo Horizonte - 30.161 - Minas Gerais

RESUMO

Este artigo apresenta um servidor de arquivo para redes de computadores capaz de executar transações atômicas de forma transparente à aplicação. O servidor possui diretórios organizados hierarquicamente e é capaz de armazenar arquivos com tamanho máximo praticamente limitado apenas pela capacidade de armazenamento do dispositivo de memória secundária.

1. SERVIDOR DE ARQUIVOS.

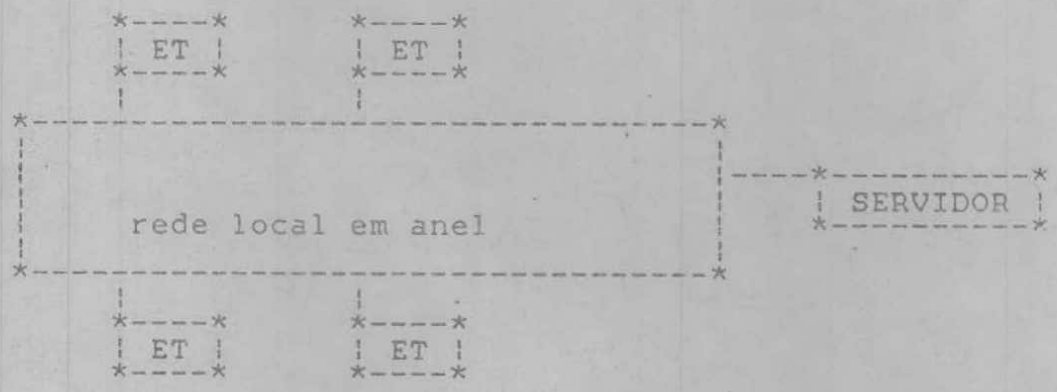
O objetivo deste servidor de arquivos é atender um ambiente de desenvolvimento de sistemas e edição de textos. Não é objetivo suportar aplicações características de bancos de dados distribuídos.

As aplicações que são executadas nas estações de trabalho conectadas à rede utilizam o serviço de arquivos remotos de forma transparente, ou seja, o acesso aos arquivos remotos são feitos da mesma forma que aos arquivos locais.

As idéias utilizadas na elaboração deste servidor foram inspiradas nos servidores de arquivos desenvolvidos na Universidade de Cambridge [Dion 80] e no Centro de Pesquisas da Xerox em Palo Alto [Sturgis et Mitchel 80]. Os algoritmos usados para a implementação da transação atômica são desenvolvimentos das idéias

apresentadas em [Guimarães e Toledo 87].

Para a implantação de um serviço de arquivos é necessário uma rede de computadores com várias estações de trabalho e um computador dedicado, conectado á rede, com grande volume de memória secundária. A figura 1, que se segue, apresenta um esquema de serviço de arquivos em uma rede local em anel, tal como a desenvolvida no Departamento de Ciência da Computação da Universidade Federal de Minas Gerais, descrita em [Silva Júnior 87].



ET - Estação de Trabalho

Figura: 1. Um Servidor de Arquivos em Rede de Computadores.

O servidor de arquivo foi projetado para utilizar os serviços da camada de transporte da rede de computadores, conforme definida em [ISO 84].

A implementação do servidor foi feita em linguagem C, utilizando um microcomputador tipo IBM-PC, com o sistema operacional MS-DOS [Microsoft 85].

2. O SISTEMA SERVIDOR DE ARQUIVOS.

O sistema servidor de arquivos é composto de três partes: um subsistema de armazenamento, um subsistema de interfaceamento com usuários e um subsistema de comunicação.

O subsistema de armazenamento inclui o dispositivo de armazenamento (disco) e o software de gerenciamento correspondente.

O subsistema de interfaceamento com usuários se compõe basicamente do software que se apresenta ao usuário como o prestador do serviço de arquivos, a nível de sistema operacional.

O subsistema de comunicação tem o objetivo de prover o transporte de informação entre os dois subsistema anteriores.

Do ponto de vista de arquitetura de redes de computadores, os subsistemas de armazenamento e de interfaceamento com os usuários configuram a camada prestadora do serviço de arquivos. Esta camada se utiliza dos serviços de transporte da rede.

Existem duas entidades cooperantes entre si na camada prestadora do serviço: no subsistema de armazenamento existe a entidade chamada "servidor" e no subsistema de interfaceamento com usuários existe uma entidade chamada "cliente".

E responsabilidade do cliente mapear as solicitações feitas pela aplicação em operações primitivas do servidor. É responsabilidade do servidor executar as tarefas de armazenamento e recuperação de dados, além de outras funções.

A figura 2, que se segue, apresenta a arquitetura do sistema servidor de arquivos.

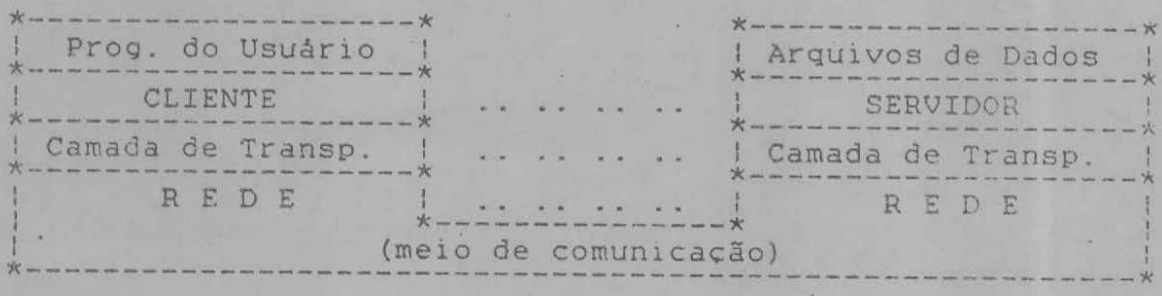
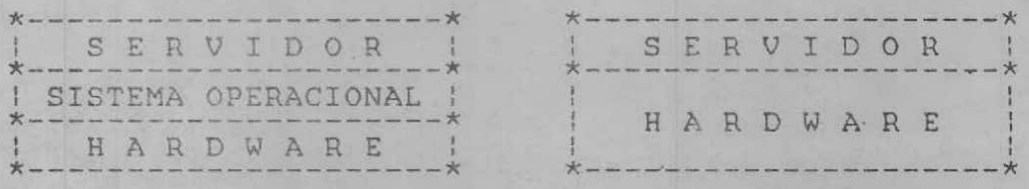


Figura 2. Arquitetura do Sistema Servidor de Arquivos.

O servidor foi implementado sobre o sistema operacional, apesar de se utilizar um computador dedicado. Esta opção gerou uma perda de desempenho, pois se construído diretamente sobre o "hardware" do equipamento, teria um desempenho melhor. A utilização do sistema operacional se deveu ao grande interesse com a portabilidade do sistema e à dificuldade de desenvolvimento que ocorreria

pela ausência das rotinas básicas do sistema operacional. A figura 3, que se segue, apresenta as duas opções de implementação do servidor.



maior portabilidade
menor complexidade de implementação

maior eficiência
maior complexidade de implementação

Figura 3. Opções de Implementação do Servidor.

O cliente foi projetado para ficar residente em memória primária, junto ao sistema operacional. Desta forma, a interface oferecida pelo cliente à aplicação pode ser a mesma que o sistema operacional local oferece, garantindo assim, a manutenção da transparência no tocante à localização remota ou não dos arquivos. A figura 4. apresenta a localização do cliente junto à estrutura de camadas das estações de trabalho.

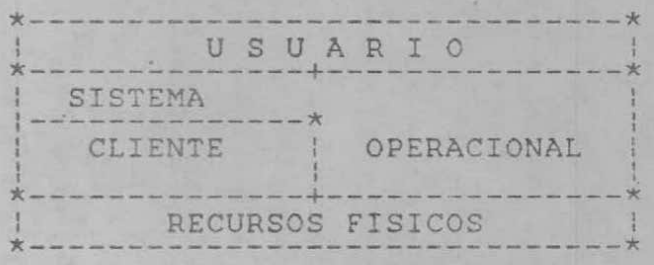


Figura 4. Implementação do Cliente nas Estações de Trabalho.

E interessante salientar que para implementar vários tipos de estações de trabalho diferentes, ou seja, conectar ao serviço equipamentos diferentes com sistemas operacionais também diferentes, é necessário que se tenha uma implementação de cliente para cada tipo estação de trabalho.

Em [Castro et alli 87] é apresentda a implementacaõ de um cliente de servidor de arquivos para o sistema operacional MS-DOS. O protocolo servidor-cliente utilizado é definido em [Soares Júnior 88].

3. INTERFACE COM A REDE DE COMPUTADORES.

A camada de serviço de arquivos foi concebida para executar imediatamente sobre a camada de transporte da rede de computadores. Utiliza os serviços de conexão da camada de transporte conforme definido em [ISO 84].

A figura 5, a seguir, apresenta a interface da camada de transporte e a figura 6 apresenta um uso típico da rede pelo cliente.

FASE	SERVIÇO	PRIMITIVA	PARAMETROS
TC establishment	TC establishment	T-CONNECT request	Called Address, Calling Address, Expedited Data option, Quality of Service, TS User-Data
		T-CONNECT indication	Called Address, Calling Addresss, Expedited Data Option, Quality of Service, TS User-Data
		T-CONNECT response	Quality of Service, Responding Address, Expedited Data Option, TS User-Data
		T-CONNECT confirm	Quality of Service, Responding Address, Expedited Data Option, TS User-Data
Data transfer	Normal Data Transfer	T-Data request	TS User-Data
		T-Data indication	TS User-Data
TC Release	TC Release	T-DISCONNECT request	TS User-Data
		T-DISCONNECT indication	Disconnect Reason, TS indication User-Data

Figura 5. Primitivas da Camada de Transporte

CLIENTE		SERVIDOR
T-CONNECT request	---->	T-CONNECT indication
T_CONNECT confirm	<----	T-CONNECT response
T-DATA request	---->	T-DATA indication
T-DATA indication	<----	T-DATA request
T-DATA request	---->	T-DATA indication
T-DATA indication	<----	T-DATA request
T-DISCONNECT request	---->	T-DISCONNECT indication

Figura 6. Abertura e Fechamento de Conexão pelo Cliente - Processo Normal de Comunicação.

4. ESTRUTURAS DE DADOS.

O servidor de arquivos é capaz de realizar transações atômicas em um arquivo envolvendo atualização de diretórios sem a necessidade de duplicar todo o arquivo alterado. As estruturas de dados e os algoritmos tiveram de ser preparados para suportar estas tarefas sem perder o controle do espaço usado de memória secundária. Em [Guimarães et Toledo 87] são apresentados algoritmos e estruturas de dados que são adequados para tal tarefa. Porém estas não controlam atualizações em diretórios e as transações atômicas envolvem vários arquivos. Os algoritmos e estruturas de dados aqui apresentadas foram inspirados no trabalho acima citado.

4.1. Memória Secundária.

A estrutura de dados em memória secundária é toda construída sobre um vetor de registros. Estas registros são de tamanho fixo e igual a 512 octetos, doravante referenciados simplesmente por páginas. Este número foi escolhido por ser a unidade de acesso das controladoras dos discos Winchester e também o registro físico tratado pelo sistema operacional MS-DOS.

O vetor de páginas tem a seguinte definição:

```
constante MaxPoint = 2.147.483.647   ( ( 2**31 ) - 1 )
tipo SecondaryMemoryRecordType = vetor [0..511] de octetos;
tipo SecondaryMemoryType = vetor [0..MaxPoint] de
                                    SecondaryMemoryRecordType;
```

A área inicial do vetor será usada pelo servidor para trabalho, para o armazenamento dos mapas de espaço disponivel em memória secundária e para armazenamento de "flags" de controle de transações atômicas. A utilização do vetor de páginas é esquematizado pela figura 7.

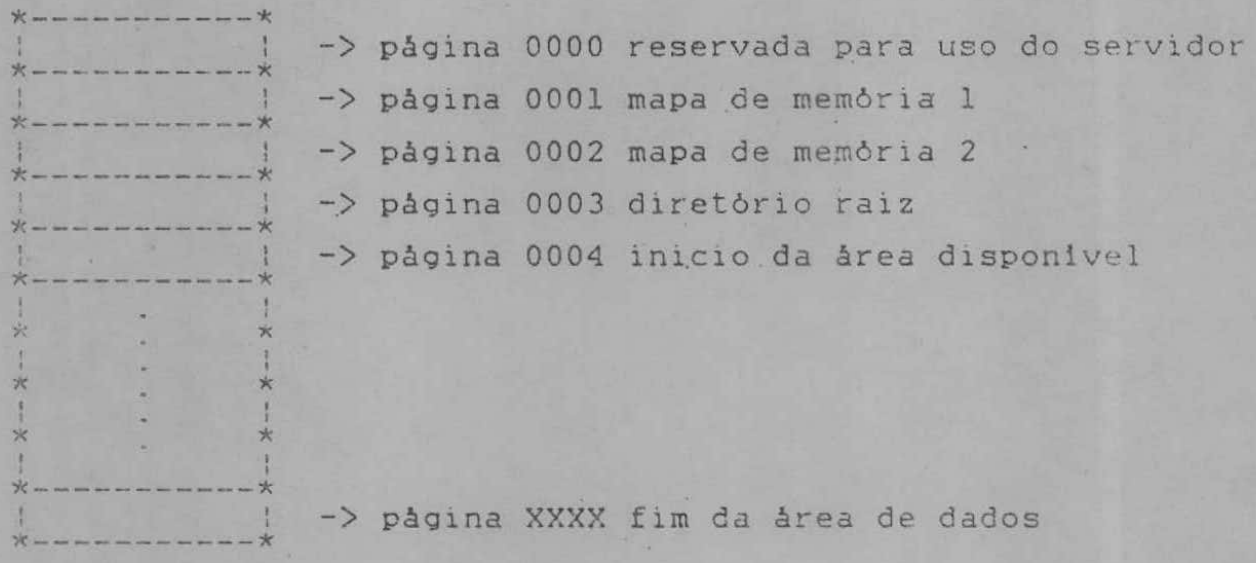


Figura 7. Distribuição da Estrutura de Dados em Memória Secundária.

Os tipos básicos para a definição das estruturas são os seguintes:

```
LongInteger    = 4 octetos;
Integer        = 2 octetos;
Byte           = 1 octeto;
TablePointType = Integer;
PagePointType  = LongInteger;
FileAttrType   = Integer;
DirAttribType  = Integer;
TimeDateType   = LongInteger;
```

```
FileNameType    =  12 octetos;
DirNameType     =  30 octetos;
PageType        =  512 octetos;
HandleType      =  Integer;
```

Cada entrada no diretório tem o seguinte formato:

```
tipo FileTableRecordType = registro
        FileName : FileNameType;
        FileRoot : PagePointType;
        Shadow   : PagePointType;
        LastChan : TimeDateType;
        FileAttr : FileAttribType;
        NPages   : LongInteger; { número de páginas }
        NTA      : Integer; { número da transação atômica }
        NLevels  : integer; { número de níveis da árvore }
        NLevelsShadow  : integer; {idem NLevels para Shadow }
        Type     : ( File, Dir );
    fim;
```

O mapa de memória, que possui os blocos de páginas livres do disco, tem o seguinte formato:

```
tipo MemoryMapType = registro
        Counter : 0..2; {verificar qual é o mais recente }
        MemoryList : vetor [ 1.. 63 ] de MemoryBlock;
        Next  : LongInteger; { Continuação do mapa }
    fim;
```

O servidor trabalha com duas cópias do mapa de memória. O campo Next aponta para a continuação do mapa e o Counter é usado para verificar qual dos dois mapas de memória do disco é o mais recente. Cada elemento de MemoryList é uma dupla de apontadores para o início e o fim de um bloco de páginas livres.

```
tipo MemoryBlock = registro
        BeginPoint : PagePointType;
        EndPoint   : PagePointType;
    fim;
```

Para a definição da estrutura interna dos mapas de alocação de memória foi estudada a estrutura de gerência de disco do sistema operacional PICK [Sequel 81]. Este gerencia uma área composta de um vetor de "frames" (páginas de 512 "bytes") exatamente como o servidor. Pela similaridade do problema, acreditou-se inicialmente que as boas qualidades inerentes a esta gerência de memória seriam herdadas. Mas, como a efetivação das transações atômicas se baseia em um processo de se trabalhar com dois mapas de memória, um atualizado com as transações pendentes e um na versão decidida, surgiu o problema de se colocar em duas listas encadeadas diferentes uma mesma página e, eventualmente, manter a sua área de dados intacta. Dada a impossibilidade de se implementar este processo, optou-se pelo uso dos mapas como apresentado em [Guimarães et Toledo 87], incluindo o contador módulo três para a descoberta do mapa mais recente.

4.2. Memória Primária.

A estrutura de dados básica que será mantida em memória primária é composta de:

Buffers de Páginas: conterá as páginas dos arquivos lidas ou a serem gravadas em disco.

Mapa de Memória Atual: Contém a lista de páginas livres do disco. É dele que todas as transações retiram páginas para uso.

Mapa de Memória Decidido: Contém a versão do mapa de disco após a conclusão das transações correntes. Só é atualizado após a passagem da transação pelo ponto de decisão.

Tabela de Páginas dos Buffers: Contém o mapeamento das páginas nos "buffers" de memória e a informação de página alterada ou não.

Tabela de Descritores de Transações: Contém os dados referentes a cada transação, como: cliente envolvido, arquivo utilizado, páginas alocadas, páginas liberadas, etc.

4.3. Formato das Mensagens.

O padrão das mensagens que transitam entre os clientes e o servidor é o descrito na figura 8 abaixo.

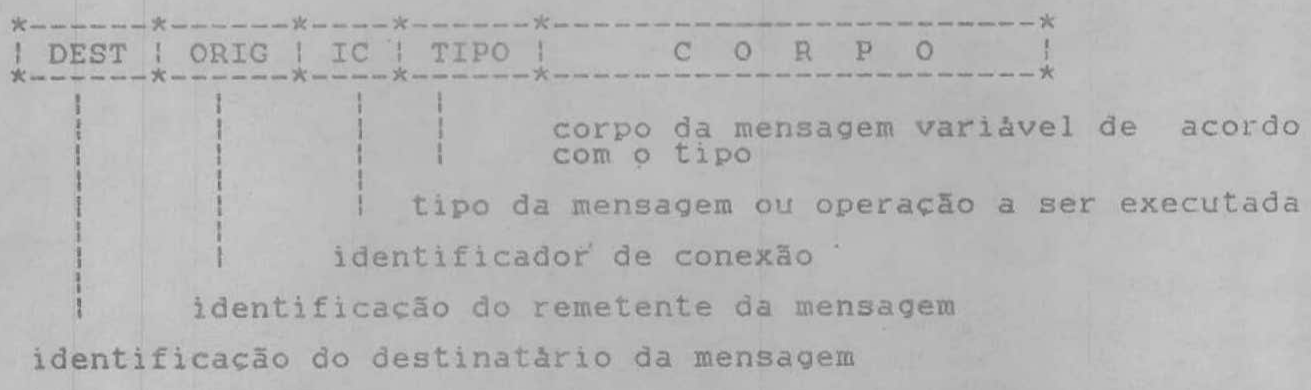


Figura 8. Formato Padrão das Mensagens.

O corpo da mensagem varia de acordo com o tipo da mensagem e com o remetente. Existe para cada primitiva do servidor um corpo de mensagem para o solicitante (cliente) enviar e um corpo para o solicitado (servidor) responder. Estes formatos seguem a este parágrafo de forma concisa. Os campos de destino e origem podem ser usados para uma análise estatística de desempenho e solicitações. Caso não se deseje este tipo de acompanhamento estes campos podem ser retirados da mensagem.

```
* Solicita Transação            -    Cliente -> Servidor
Corpo : ;     /* não é necessário nenhum dado */

* Solicita Transação (Resp)    -    Servidor -> Cliente
Corpo :   TransNo /* Número da Transação ( 2 octetos ) */
          Result /* 1 octeto */

* Término de Transação          -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */

* Término de Transação (Resp) -    Servidor -> Cliente
Corpo:    TransNO /* 2 octetos */
          Result /* 1 octeto */
```

```
* Aborto de Transação          -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
* Aborto de Transação (Resp)   -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */

* Criar Arquivo                -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          FileName /* 12 octetos */
          Pathname /* 30 octetos */
          FileAttr /* 2 octetos */
* Criar Arquivo (Resp)         -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */

* Excluir Arquivo              -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          FileName /* 12 octetos */
          Pathname /* 30 octetos */
* Excluir Arquivo (Resp)       -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */

* Trocar Nome                  -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          FileName /* 12 octetos */
          PathName /* 30 octetos */
          NewName /* 12 octetos */
* Trocar Nome (Resp)           -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */

* Criar Diretório              -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          PathName /* 30 octetos */
          DirName /* 12 octetos */
          DirAttr /* 2 octetos */
* Criar Diretório (Resp)       -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */
```

```
* Excluir Diretório              -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          Pathname /* 30 octetos */
          DirName /* 12 octetos */
* Excluir Diretório (Resp)       -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */

* Listar Diretório               -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          PathName /* 30 octetos só na primeira chamada */
          DirName /* 12 octetos só na primeira chamada */
* Listar Diretório (Resp)        -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */
          DirItem /* 18 octetos - Nome, Attr., TimeDate */

* Abrir Arquivo                  -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          FileName /* 12 octetos */
          PathName /* 30 octetos */
          Mode /* 1 octeto */
* Abrir Arquivo (Resp)           -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */
          Handle /* 2 octetos */

* Fechar Arquivo                 -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          Handle /* 2 octetos */
* Fechar Arquivo (Resp)          -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */

* Ler Arquivo                    -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          Handle /* 2 octetos */
          PageNo /* 4 octetos */
```

```
* Ler Arquivo (Resp)          -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Page /* 512 octetos */
          Result /* 1 octeto */

* Escrever Arquivo            -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          Handle /* 2 octetos */
          Page /* 512 octetos */
          PageNo /* 4 octetos */
* Escrever Arquivo (Resp)     -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */

* Trocar Atributo de Arquivo  -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          FileName /* 12 octetos */
          PathName /* 30 octetos */
          FileAttr /* 2 octetos */
* Trocar Atributo de Arquivo (Resp) -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */

* Trocar Atributo de Diretório  -    Cliente -> Servidor
Corpo:    TransNo /* 2 octetos */
          DirName /* 12 octetos */
          PathName /* 30 octetos */
          DirAttr /* 2 octetos */
* Trocar Atributo de Diretório (Resp) -    Servidor -> Cliente
Corpo:    TransNo /* 2 octetos */
          Result /* 1 octeto */
```

5. TRANSAÇOES ATOMICAS.

Transações são conjuntos de operações primitivas delimitadas com uma marca de inicio e fim explicitas. Operações primitão conjuntos de operações primitivas delimitadas com uma marca de inicio e fim explicitas. Operações primitivas são, por exemplo, operações

de leitura em arquivos, gravação em arquivos, criação de arquivos, etc.

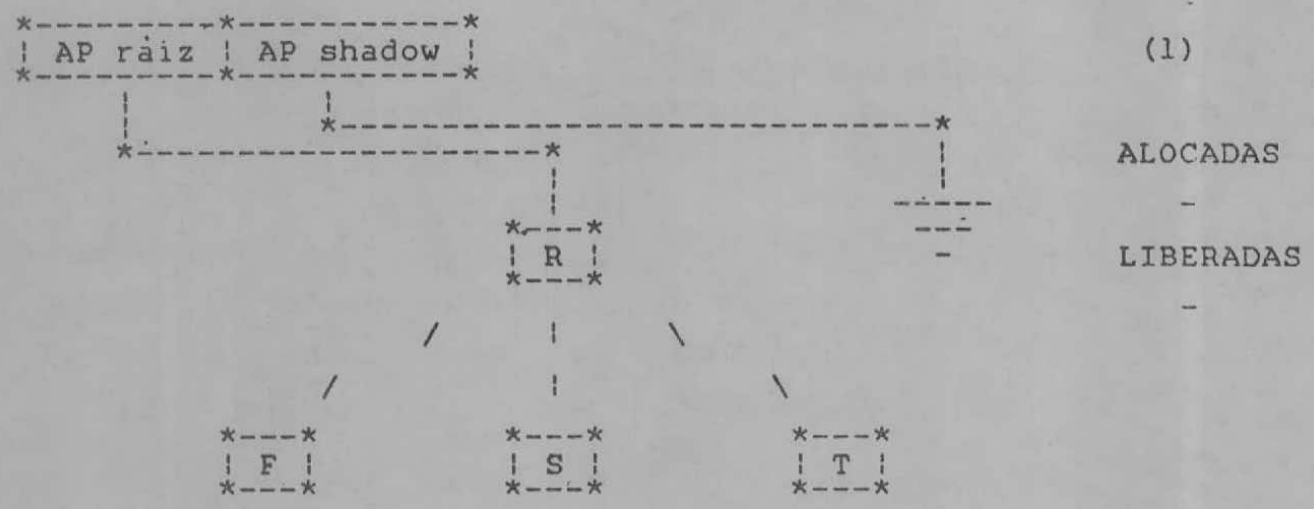
Transação atômica é a união do conceito de transação com o de átomo, ou seja, uma transação na qual se tenha a garantia do "tudo ou nada". Ou todas as operações que compõe a transação ocorreram com sucesso ou nenhuma delas ocorreu. Em outras palavras, o estado do objeto da transação é sempre constistente.

As transações atômicas são implementadas nos servidores de arquivos para garantir integridade aos arquivos do servidor. Neste servidor, as transações atômicas envolvem apenas atualização ou exclusão de um único arquivo, e, no caso de alteração, a transação dura da abertura ao fechamento do arquivo.

A idéia básica da implementação da transação atômica é a duplicação da área atualizada e a marcação de um "ponto de efetivação" da transação. Fazendo uma analogia, este ponto funciona como o alto de uma montanha na qual um automóvel sem freios está atravessando. Se a gasolina acabar antes de chegar no alto, o automóvel retornará todo o caminho percorrido; se acabar depois, ele terminará a travessia.

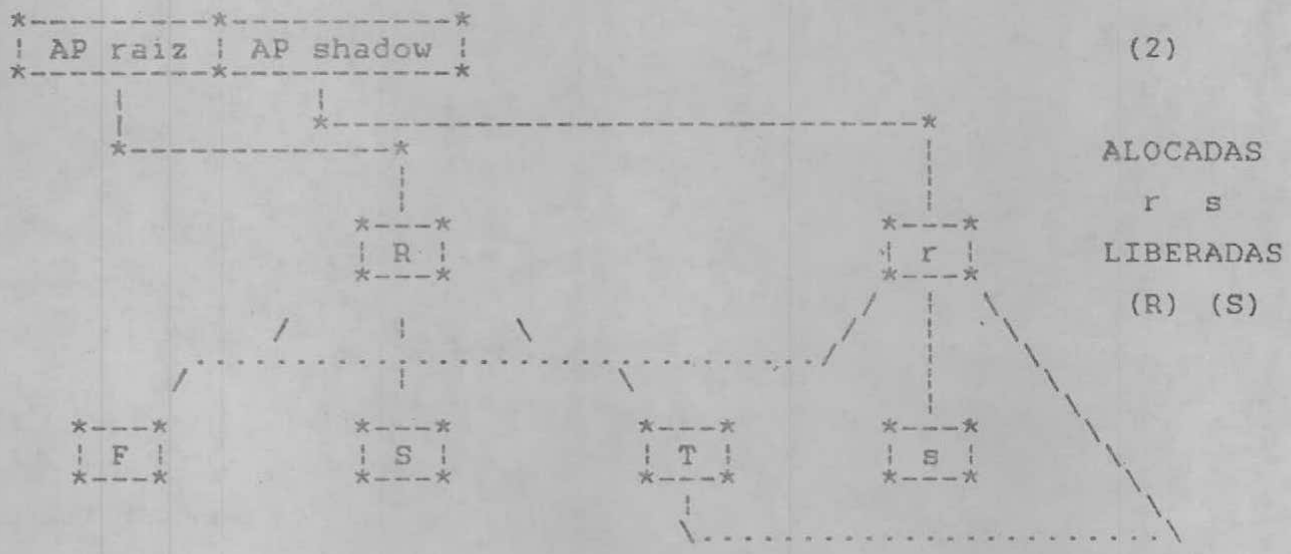
Como os arquivos são implementados como árvores, a duplicação de dados é feita sobre novas páginas chamadas páginas "shadows". O esquema é apresentado nas figuras 9, 10 e 11.

Os algoritmos aqui apresentados são adaptações dos publicados em [Guimarães et Toledo 87].



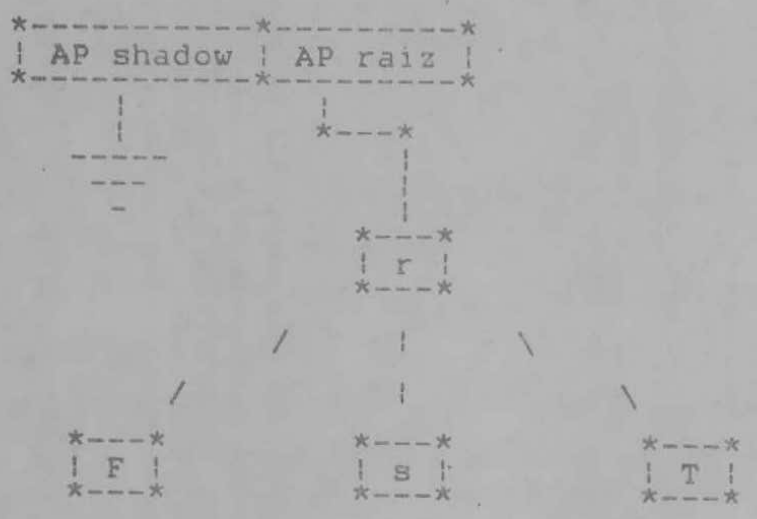
Supondo o arquivo acima com 3 nodos de dados (F, S e T). O objetivo é de alterar o conteúdo do nodo S.

Figura 9. Transação Atômica - Estado Inicial.



O primeiro passo é alocar 2 nodos (r e s) e colocar o apontador da versão "shadow" contido na tabela de arquivos para o nodo r. Copiar o conteúdo de R em r e fazer o apontador de S apontar para s. Gravar ás alterações em s e marcar a transação como pronta para decidir. Isto é feito gravando o identificador da transação em um espaço especial do disco. As páginas R e S são colocadas em uma lista de páginas prontas para a liberação da transação e as páginas r e s são colocadas em uma lista de prontas para alocação da transação. Estas listas ficam em memória primária.

Figura 10. Transação Atômica - Estado Intermediário.



(3)

ALOCADAS
r s
LIBERADAS
(já efetuado)
R S

Após passar pela fase intermediária, a transação termina com sucesso, mesmo se houver falha no servidor. Basta atualizar os apontadores de raiz na tabela de arquivos, liberar os nodos da lista de prontos para liberação e alocar os nodos da lista de prontos para alocação. Se houver falha no servidor, quando este retornar, deve verificar se existe alguma transação por decidir, e em caso positivo, seguir o apontador para a versão "shadow", alocando os novos nodos, e seguir o apontador para a versão antiga, liberando os nodos antigos.

Figura 11. Transação Atômica Estado Final.

Os pontos principais para implementar a "passagem pelo topo da montanha" são:

- Salvar o apontador para a versão "shadow" em memória não volátil;
- No momento da decisão, salvar uma identificação do arquivo em um local especial da memória não volátil.

O algoritmo para decisão da transação atômica é:

- Gravar as páginas atualizadas pela transação em disco;
- Gravar um identificação do arquivo em um ponto determinado do disco;
- Atualizar do mapa de memória na memória e no disco;
- Atualizar o diretório no disco;
- Retirar a identificação do arquivo do disco.

O algoritmo de recuperação em caso de falha do servidor é:

```
se ( Existe arquivo marcado no ponto especial do disco ) então
   se ( arquivo marcado para exclusão ) então
      percorrer o arquivo colocando as páginas na lista de
      páginas liberadas da transação;
      Executar o algoritmo de decisão de transação;
   senão
      Percorrer a versão antiga do arquivo colocando as páginas
      na lista de páginas liberadas da transação;
      Percorrer a versão "shadow" colocando as páginas na lista
      de páginas alocadas da transação;
      Executar o algoritmo de decisão de transação;
   fim se;
fim se;
Percorrer todos os diretórios desmarcando os arquivos excluídos e
apagando os apontadores "shadows";
```

Este algoritmo possui a propriedade de poder ser recomeçado várias vezes sem causar danos á estrutura de dados, isto é, ele não é destrutivo.

BIBLIOGRAFIA.

[Castro et alli 87] IMPLEMENTAÇAO DE UM CLIENTE DE SERVIDOR DE ARQUIVOS PARA A REDE LOCAL DCC/NET. Castro, M. A. C., Nogueira, J. M. S., Soares Júnior J. E. & Ribeiro. B. A. N. Belo Horizonte, DCC - ICEx - UFMG, 1987.

[Dion 80] THE CAMBRIDGE FILE SERVER. Dion, J, Computer Laboratory University of Cambridge. ACM SIGOPS Open System Review 14,4 (October), pp 26-35, 1980.

[Guimarães & Toledo 87] IMPLEMENTAÇAO DE UM SERVIDOR DE ARQUIVOS COM TRANSAÇOES ATÔMICAS. Guimarães, C. C. & Toledo, M. B. F., In: Anais do VII Congresso da Sociedade Brasileira de Computação - SBC, Salvador, julho/1987

[ISO 84] ISO/TC 97 - INFORMATION PROCESSING SYSTEM - OPEN SYSTEM INTERCONECTION - TRANSPORT SERVICE DEFINITION, International Organization for Standardization, 1984.

[Microdata 81] SEQUEL ASSEMBLY LANGUAGE. Programming Manual, Microdata Corporation, 1981.

[Microsoft 85] MICRO SOFT - DISK OPERATION SYSTEM. VERSION 3.00 - TECHINICAL REFERENCE, 1985

[Silva Júnior 87] UM CONTROLADOR DE COMUNICAÇOES PARA UMA REDE LOCAL EM ANEL, Silva Júnior, D. C., Tese de Mestrado em Ciência da Computação, Belo Horizonte, DCC - ICEx - UFMG, 1987.

[Sturgis & Mitchell 80] ISSUES IN DE DESIGN AND USE OF A DISTRIBUTED FILE SYSTEM. Sturgis, H., Mitchell, J. & Israel, J., ACM SIGOPS OpenSystem Review 14,3 (july), pp 55-69, 1980.

[Soares Júnior 88] UM SERVIDOR DE ARQUIVOS COM TRANSAÇOES ATOMICAS E DIRETORIOS PARA REDES DE COMPUTADORES. Soares Júnior, J. E., Tese de Mestrado em Ciência da Computação, Belo Horizonte, DCC - ICEx - UFMG, 1988.