ARQUITETURAS DE REDES LOCAIS TOLERANTES A FALHAS

Afonso Jorge Ferreira Cardoso.*

RESUMO: O projeto de uma rede local de computadores, entre outras características, necessita estar inserido em classificações que permitam ao projetista delinear o seu campo de ação. Este trabalho apresenta uma proposta de classificação para redes locais tolerantes a falhas, considerando os diversos aspectos envolvidos nos projetos dessas redes e as soluções de hardware e software disponíveis nos meios comercial e acadêmico.

1. INTRODUÇÃO

A elaboração de um projeto de rede local de computadores pressupõe a coleta de diversas informações existentes no ambiente institucional como, por exemplo, recursos de hardware e software disponíveis e necessários, recursos humanos, volume de dados, qualidade de serviços, tolerância a falhas etc.

Entretanto, o projeto necessita, também, ser inserido em classificações que orientem as ações do projetista. No âmbito de redes de computadores, classificações como, por exemplo, quanto a forma de atendimento a serviços, são fundamentais para definir a forma de ação dos nodos servidores no atendimento das requisições de serviços dos usuários.

A classificação das redes por topologia também é um bom exemplo da importância dessas classificações para redes locais de computadores. Afinal, a topologia pode definir a tecnologia a ser empregada. A topologia baseada em barramento, por exemplo, pressupõe o uso da tecnologia Ethernet, enquanto a topologia baseada em anel, geralmente, o uso da tecnologia Token Ring.

Com redes locais tolerantes a falhas não é diferente. Há também a necessidade de classificá-las com o mesmo objetivo das classificações existentes. Entretanto, essa classificação não é encontrada nos meios comercial e acadêmico, por ser uma área de pesquisa relativamente nova. Este trabalho propõe uma classificação para arquiteturas de redes locais tolerantes a falhas, onde são considerados aspectos, isolados ou em grupos, que definem univocamente cada arquitetura.

*Mestre em Ciência da Computação, UFRGS; Analista de Sistemas da EMBRAPA - Amazônia Oriental; Professor Adjunto da UNAMA.

2. ARQUITETURA DE REDES LOCAIS TOLERANTES A FALHAS

A inclusão de tolerância a falhas em redes locais considerando que os nodos da mesma são compostos por microcomputadores e estações de trabalho, pode ser feita com base em arquiteturas que podem ser identificadas como: arquitetura baseada em *cluster*[1], arquitetura baseada em *hardware* tolerante a falhas, arquitetura baseada em replicação de dados e arquitetura baseada em servidor duplicado. A seguir, cada uma dessas arquiteturas é descrita de forma breve.

2.1 ARQUITETURA BASEADA EM *CLUSTER*

Uma rede com servidores organizados em *cluster* consiste em um conjunto de máquinas, construídas com a mesma tecnologia dos computadores pessoais, interconectadas através de um dispositivo de comunicação de alta velocidade como, por exemplo, uma conexão ATM (*Asynchronous Transfer Mode*) (1), (2). As estações de trabalho e os microcomputadores que formam a rede são conectados ao *cluster* através de uma rede local comum como, por exemplo, uma rede *Ethernet*. Apenas as máquinas que são servidoras estão ligadas através da conexão de alta velocidade formando o *cluster*. A figura 1 mostra uma rede local com topologia de barramento que utiliza servidores organizados em *cluster*.

A arquitetura baseada em *cluster* possui algumas características de funcionamento que, por serem muito importantes, são destacadas em seguida.

O endereço correspondente ao conjunto de servidores que formam o *cluster* é único, ou seja, os clientes consideram que a rede possui apenas um servidor. Quando o pacote de dados de um cliente é enviado com o prefixo que identifica o endereço do *cluster*, o protocolo IPv6 (*Internet Protocol version 6*), por exemplo, roteia esse pacote de dados para um dos servidores do *cluster*. Caso um outro cliente envie, por exemplo,um outro pacote de dados para o *cluster*, o IPv6 pode escolher um outro servidor do *cluster* para atender ao serviço, permitindo assim que dois servidores do *cluster* atendam requisições concorrentemente (03).

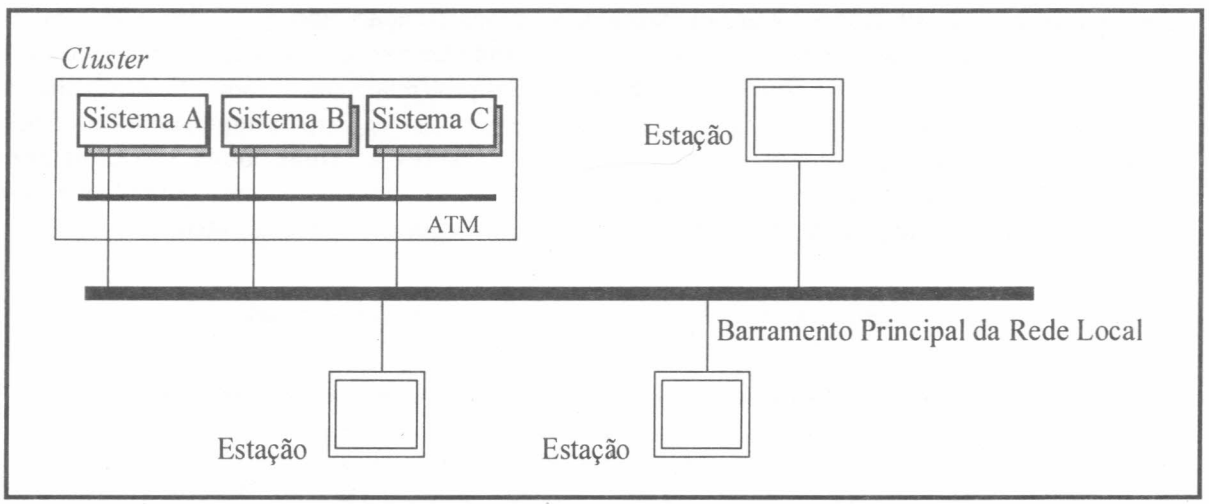

FIGURA 1 - Rede local com servidores organizados em cluster

[1] *Cluster* é uma nova geração de *hardware* que está surgindo para ser utilizada como suporte para aplicações cliente/servidor (02). Compreende um conjunto de computadores que possui um único endereço de rede, onde um pacote de dados recebido pode ser entregue para qualquer computador que integre o conjunto (03).

Segundo Birman (02), talvez a mais importante característica da arquitetura *cluster* seja o uso de cópias completas do sistema operacional sendo executadas em cada servidor. Essa característica evita que o sistema operacional trabalhe como se estivesse gerenciando um sistema com múltiplos processadores (em que apenas uma cópia do sistema operacional faz o gerenciamento do sistema), que necessitaria de *hardware* com configuração suficiente para suportar compartilhamento de memória. Ademais, distingue-o de sistemas operacionais do tipo "escravo" freqüentemente usado em sistemas de computações paralelas.

A arquitetura de rede com servidores organizados em *cluster* possui ainda as seguintes características (02):

- Componentes padrões: a arquitetura *cluster* busca utilizar, na sua composição, componentes que também são utilizados na construção de estações de trabalho e computadores pessoais, visando os benefícios proporcionados pela produção em larga escala, os baixos custos, assim como a total compatibilidade com os outros equipamentos da rede. Os servidores *cluster* não possuem monitores de vídeo e podem ser inseridos em um chassi não convencional.
- Interconexão de alta velocidade: o barramento de comunicação interna do *cluster* é tipicamente orientado a mensagens e pode ser baseado na tecnologia ATM ou em uma conexão CSMA (*Carrier Sense Multiple Access*) de alta velocidade.
- Infra-estrutura do *cluster*: além do chassi e do cabeamento que interliga os servidores, compõem também a infra-estrutura: o fornecimento de força independente da fornecida à rede e os adaptadores de comunicação.
- Subsistema de gerenciamento: as funções deste subsistema compreendem: manter a consistência dos dados e disponibilidade dos servidores, guardar informações sobre todos os servidores da rede e reagir a eventos como um defeito na máquina ou uma requisição de mudança de configuração. Estas funções compõem um processo monitor existente em cada servidor do conjunto que implementa o gerenciamento distribuído do HA (*highly Available*) *Cluster (01)*.
- *Cluster* API[1] (*Application Program Interface*): possui informações sobre o conjunto de nodos que formam o *cluster* e de como monitorar seus estados de funcionamento. Inclui também informações sobre como dar início a aplicações nos nodos, como realizar a sua monitoração durante a execução e como acessar as funções de gerenciamento do *cluster*.

Tolerância a falhas na arquitetura baseada em *cluster* é provida, basicamente, através das características citadas. Porém, essas características dependem em grande parte da forma como o *cluster* é projetado. Os servidores podem trabalhar, por exemplo, com discos rígidos duplicados para aumentar a disponibilidade dos dados.

Birman afirma que servidores organizados em *cluster* projetados de forma apropriada, podem oferecer vantagens na implementação de sistemas considerando os aspectos de comunicação e gerência. Dentre estes aspectos, são citados alguns objetivos que os projetistas de aplicações almejariam obter: disponibilidade contínua do sistema, rápida detecção de erros, facilidade de gerenciamento incluindo soluções automáticas e visão do sistema como máquina única.

Vale ressaltar que outras características de tolerância a falhas como, por exemplo, replicação de volumes ou grupos de dados, ações atômicas, recuperação e manutenção de servidores, são introduzidas nesta arquitetura através das tecnologias usadas para desenvolver aplicações que utilizam o *cluster* como suporte.

2.2 ARQUITETURA BASEADA EM *HARDWARE* TOLERANTE A FALHAS

A arquitetura de uma rede pode incluir entre seus componentes servidores com *hardware* tolerante a falhas. Os servidores que

[1] Contém a definição completa de todas as funções do sistema operacional disponíveis a uma aplicação (as funções que a aplicação pode usar para executar tarefas como o gerenciamento de arquivos e apresentação de informações na tela do computador) e como a aplicação deve utilizar essas funções. Nos sistemas operacionais de redes, a API define um método padronizado que as aplicações podem utilizar para aproveitar todos os recursos da rede (08).

possuem essa característica são construídos com módulos de *hardware* redundantes, tais como: unidade central de processamento, memória, controladores de entrada e saída e controladores de comunicação.

A tolerância a falhas em hardware é construída de maneira que o sistema de controle permita que os seus módulos redundantes possam se auto-testar e conseqüentemente detectar uma falha que ocorra em um dos módulos (02). Os módulos devem ser definidos de forma a manter a independência de cada um para que a remoção e adição de módulos, visando substituição dos que falham, não afete o restante do sistema. O objetivo maior é que a dinâmica de renovação dos módulos não seja causa de interrupção da atividade do software que esteja sendo executado nos níveis mais altos de abstração (05).

O *hardware* tolerante a falhas oferece proteções contra erros que podem afetar a integridade dos dados e permite que os programas de aplicação executem de forma contínua mesmo na presença de falhas (02). Ele deve ser usado em sistemas que trabalham com aplicações críticas em tempo-real como, por exemplo, controle de reatores nucleares e monitoração dos batimentos cardíacos de um paciente(04).

Adicionalmente, a utilização de *hardware* tolerante a falhas possui um importante papel no controle de processo de sistemas onde uma parada tem implicações imediatas e de alto custo (02). A indústria de telecomunicações, por exemplo, necessita de *hardware* tolerante a falhas para manter suas aplicações de suporte as operações, aplicações em redes inteligentes e serviços através da *internet* [SUN96].

2.3 ARQUITETURA BASEADA EM REPLICAÇÃO DE DADOS

Informações que são guardadas em um único nodo de uma rede local não podem ser recuperadas caso este nodo falhe. Assim, para que uma operação seja realizada com sucesso, os dados a serem manipulados devem ser replicados para outros nodos da rede, de maneira que a ocorrência de falhas em um dos nodos não torne os dados inacessíveis às operações dos usuários (16).

Entretanto, o processo de replicação de dados introduz problemas de consistência dos dados e gerenciamento de réplicas. Manter a consistência dos dados significa garantir que todas as requisições sejam processadas por todas as cópias na mesma ordem. Esta ordenação denominada serialização como cópia única deve ser realizada pelos algoritmos de controle de réplicas, que devem ser capazes de mascarar falhas em sistemas que utilizam replicação de dados.

Os métodos de controle de réplicas podem ser otimistas ou pessimistas (06). No método otimista, se ocorre particionamento da rede, por exemplo, o processamento de requisições continua sendo realizado normalmente nas diversas partições. A serialização em cada grupo pode ser mantida, mas muitos processamentos podem não satisfazer o critério de serialização como cópia única, assim como, podem ocorrer inconsistências globais, onde cópias de diferentes partições podem não ser mutuamente consistentes (12).

No método pessimista, considerando o mesmo exemplo, particionamento da rede, cada partição assume a ocorrência de pior caso em relação as outras partições e passa a operar segundo essa hipótese. Segundo Jalote (12), as três abordagens pessimistas mais comuns para controle de réplicas são: cópia primária, replicação ativa e votação.

A abordagem denominada cópia primária tem por objetivo permitir continuidade de acesso aos dados do sistema, mesmo que ocorram falhas em alguns nodos da rede. Considerando que determinada informação está armazenada em n nodos da rede, o acesso a essa informação deve ser garantido mesmo na ocorrência de falhas em n-1 nodos.

Um dos nodos que detêm a informação é designado primário e os restantes secundários. Todas as requisições de operações sobre a informação são enviadas para o primário. Caso alguma requisição seja mandada para um secundário, este deve enviá-la para o primário.

As requisições de leitura são recebidas pelo primário, que as processa e envia seus

resultados para os clientes. As requisições de escrita recebidas pelo primário são enviadas para os nodos secundários antes de serem processadas. Quando todos os secundários confirmam o recebimento das requisições de escritas, então o primário as processa e envia os resultados para os clientes. O fato da requisição de escrita chegar primeiro para o primário e este enviar sua cópia sincronamente, assegura o processamento das requisições pelos nodos secundários na mesma ordem do primário, assumindo-se que o canal de comunicação é confiável e que preserva a ordenação. O funcionamento deste algoritmo está representado na figura 2, adaptada de Leboute (14).

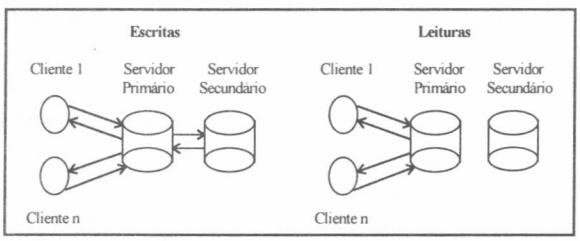

Figura 2 - Algoritmo de replicação com cópia principal

Se o grau de replicação for igual a n e se todos os n nodos que mantêm a informação falharem, nada pode ser feito. Entretanto, se o número total de falhas for igual a n-1, o sistema poderá lidar com essas falhas.

O sistema não sofre nenhum tipo de interrupção se falhas ocorrerem apenas em nodos secundários, pois o primário continuará atendendo às requisições dos clientes normalmente. Porém, se a falha ocorrer no nodo primário, é necessário que seja eleito um novo primário entre os nodos secundários.

Há várias formas de se eleger um primário. Uma discussão sobre os algoritmos de eleição pode ser encontrada em (09). A forma mais simples de eleger um novo primário é montar a rede de tal forma que os nodos estejam ordenados em uma cadeia linear, onde o de maior ordem sempre será o primário. Assim, caso o nodo primário falhe, o nodo de maior ordem na cadeia assume como o novo primário(12).

No entanto, para assumir como o novo primário, o nodo secundário eleito necessita estar atualizado, ou seja, ele deverá ter processado corretamente todas as requisições recebidas do primário, pois somente assim ele estará apto a atender as requisições dos clientes da mesma forma que o antigo primário.

O método de controle de réplicas por replicação ativa, também conhecido como abordagem estado-máquina (15), não possui um nodo centralizador das operações como na abordagem de cópia primária. Como o próprio nome sugere, todos os nodos servidores são ativos. Os clientes, em vez de mandarem suas requisições para um nodo pré-determinado, podem enviá-las para qualquer um dos nodos. Qualquer nodo pode receber, processar e responder requisições dos clientes.

Para que a consistência dos dados entre os nodos seja mantida, é necessário preservar o critério de serialização como cópia única. Para tanto, as propriedades denominadas acordo e ordenação devem ser satisfeitas.

Segundo Schneider (15), a propriedade acordo pode ser satisfeita utilizando um protocolo que permita designar um nodo, denominado transmissor, para disseminar as informações de tal forma que a veracidade destas informações possam ser confirmadas. A propriedade ordenação, pode ser alcançada utilizando-se um identificador para cada requisição, de maneira que todos os nodos possam ordenar as requisições antes de processá-las.

O método de controle de réplicas por votação pressupõe que qualquer operação a ser feita sobre dados replicados, deve ser decidida coletivamente pelas réplicas através de uma votação. Segundo Leboute , no controle de réplicas, votação significa o método em que os direitos de acesso são distribuídos e cada processo deve reunir um determinado número de "votos" até obter um quorum que lhe permita realizar uma determinada operação. O método de votação pode ser dividido em duas categorias: método estático e método dinâmico(12).

No método estático, o número de votos de uma réplica e o quorum exigido nunca mudam.

A primeira abordagem sobre um método estático de votação foi proposta por Thomas (18) e estendida por Guifford (10) passando a ser conhecida como votação ponderada.

Na votação ponderada, cada réplica possui um determinado número de votos. São estabelecidos quoruns de leitura e escrita com valores não necessariamente iguais. Assim, caso um nodo queira realizar uma operação de leitura ou escrita, deve solicitar votos dos outros nodos do sistema até atingir o quorum exigido para realizar a operação. Detalhes deste método são discutidos em (10), (12).

Os valores dos quoruns de escrita e leitura aumentam de acordo com o número de réplicas, contribuindo para um indesejável aumento no número de mensagens trocadas pelos nodos. Para evitar o aumento no valor do quorum de forma linear, foi proposta por Kuma uma abordagem denominada votação hierárquica (13), em que é definida uma ordem em forma de árvore, onde um quorum é associado a cada nível desta. Para a obtenção do quorum, os nodos são testados de acordo com a ramificação e o nível em que se encontram na árvore, de modo que o total de tentativas seja reduzido.

No método dinâmico, o número de votos de uma réplica e o valor dos quoruns pode variar. Este método é voltado para o tratamento de falhas prolongadas e principalmente particionamento de rede, como pode ser observado nos estudos de (11) apud (12) e (07). No caso de particionamento da rede, por exemplo, o objetivo é permitir que os nodos participantes da partição majoritária (única em que os nodos podem continuar realizando operações de escrita) possam reconhecer a diminuição do número de nodos e ao mesmo tempo informar aos nodos as mudanças nos valores dos quoruns, ambas causadas pelo particionamento da rede.

2.4 ARQUITETURA BASEADA EM SERVIDOR DUPLICADO

Através do sistema operacional, pode ser introduzida em uma rede a arquitetura baseada em servidor duplicado. Essa arquitetura se baseia no "espelhamento" de dados entre dois servidores: primário e secundário, que possuem funções bem definidas (14). O servidor primário é o único servidor ativo em relação aos clientes, enquanto o servidor secundário comporta-se como reserva-quente, ou seja, o primário responde a todas as requisições de serviços dos clientes e as repassa para o servidor secundário também processá-las como forma de manter os dados íntegros.

Um dos mecanismos usados para a replicação é o "espelhamento" dos dados. A existência de um servidor ativo (primário), que representa um elemento centralizador de operações, e um servidor cópia (secundário) virtual para os clientes, favorece o uso do método de cópia primária para a realização do "espelhamento". O método de cópia primária, entretanto, normalmente pressupõe apenas a replicação dos dados considerados críticos. Nas arquiteturas baseadas em servidor duplicado, todos os dados são replicados através do "espelhamento".

A tolerância a falhas introduzida com a duplicação do servidor, permite que quando ocorra uma falha no servidor primário, o servidor secundário assuma as suas funções em relação aos clientes de forma automática, contribuindo assim para o aumento da disponibilidade da rede.

3 CRITÉRIOS UTILIZADOS NA CLASSIFICAÇÃO

A classificação das arquiteturas de redes locais tolerantes a falhas foi elaborada a partir de revisão bibliográfica que incluiu livros, artigos de periódicos e material publicitário referente a soluções comerciais, contidos na bibliografia deste trabalho, considerando os seguintes aspectos:

a) Técnica de difusão de escrita usada;

b) Arquitetura dos computadores utilizados como servidores na rede;

c) Forma de interligação dos servidores;

d) Forma de endereçamento dos servidores;

e) Multiplicidade de servidores envolvidos na replicação dos dados;

f) Redundância de dados.

A classificação foi realizada com o objetivo de identificar os diversos enfoques de arquiteturas de redes tolerantes a falhas que existem, assim como identificar a(s) característica(s) que identificasse(m) univocamente cada arquitetura, chegando-se a seguinte conclusão:

a) Arquitetura baseada em *cluster*: é identificada univocamente pela combinação de quatro características: multiplicidade de servidores, servidores ativos com endereço único, interligação exclusiva dos servidores por barramento especial e redundância parcial de dados.

b) Arquitetura baseada em *hardware* tolerante a falhas: é identificada univocamente pela arquitetura dos computadores utilizados como servidores. Esses servidores possuem módulos duplicados como, por exemplo, unidade central de processamento, memória e controladores de entrada/saída. Apesar de não ter sido usado como critério, esta arquitetura pode ser também identificada pelo fato de não utilizar meios externos de comunicação de dados.

c) Arquitetura baseada em replicação de dados: é identificada univocamente por duas características combinadas: multiplicidade de servidores envolvidos na replicação de dados e redundância parcial de dados.

d) Arquitetura baseada em servidor duplicado: quantidade de servidores envolvidos na duplicação de dados e redundância total dos dados.

4 CONSIDERAÇÕES FINAIS

A classificação proposta sintetiza o estudo realizado sobre redes locais e os aspectos de tolerância a falhas que podem ser incluídos em projetos desta natureza. Sistematiza, de forma objetiva, os critérios, que isolados ou combinados, definem univocamente os tipos de arquiteturas de redes locais tolerantes a falhas existentes.

Considerando as inúmeras soluções de *hardware* e *software* que surgem a cada dia, esta proposta não pretende esgotar a discussão em torno da classificação de arquiteturas para redes locais tolerante a falhas e sim fomentar a discussão em torno do assunto.

BIBLIOGRAFIA CONSULTADA

01. AZAGURY, Alain et al. Highly Available Cluster: A Case Study. In: Annual International Symposiumon Fault-Tolerant Computing, 24., 1994, Austin. **Digest of Papers...** Washington: IEEE Compu-Computer Society Press, 1994. p.404-413.

02. BIRMAN, Kenneth P. **Building Secure and Reliable Network Applications**. New York: Department of Computer Science, Cornell Univerty, 1995. 503p.

03. COMER, Douglas E. **Computer Networks and Internets**. Upper Saddle River, NJ.: Prentice-Hall, 1997. 506p.

04. COULORIS, George; DOLLIMORE, Jenn; KINDBERG, Tim. **Distributed Systems - Concepts and Design**. Eokinghan, England: Addison-Wesley, 1994. 644p.

05. CRISTIAN, F. **Understanding Fault-Tolerant Distributed Systems**. New York: Communications of ACM, v.34, n.2, feb. 1991. p.56-70.

06. DAVDISON, Susan B.; GARCIA-MOLINA, Hector; SKEEN, Dale. **Consistency in Partioned Networks**. New York: ACM Computing Surveys, v. 17, n.3, sept. 1985. p.341-370.

07. DAVCEV, Danco. **A Dynamic Voting Scheme in Distributed Systems**. New York: IEEE Transactions on Software Engineerig, v.15, n.1, jan. 1989. p.93-97.

08. DYSON, Peter. **Novell Dicionário de Redes**. Rio de Janeiro: Campus, 1995.

09. GARCIA-MOLINA, Hector. **Elections in a Distributed Computing System**. Washigton: IEEE Transactions on Computers, 31,n.1, Jan. 1982. p.48-59.

10. GIFFORD, D.K.Weighted Voting for Replicated Data. In: Symposium on Operating Systems Principles, 7., 1979, New York. **Proceedings...** New York: ACM Press, 1979.p.150-162.

11. JAJODIA, S.; MUTCHLER, D. Dynamic Voting. In: ACM Sigmod International Conference on Mamagement of data , San Francisco: 1987. **Proceedings...** New York: ACM Press, 1987. p.227-238.

12. JALOTE, Pankaj. **Fault Tolerance in Distributed Systems**. Englewood Clifs, NJ.: Prentice-Hall, 1994. 432p.

13. KUMAR, Akhil. **Hierarchical Quorum Consensus**: A New Algorithm for Managing Replicated Data. New York: IEEE Transactions on Computers, v. 40, n.9, sept. 1991. p.996-1004.

14. LEBOUTE, Mário M. **RNFS - Um Sistetema de Arquivos Distribuído Tolerante a Falhas para o UNIX**. Porto Alegre: CPGCC da UFRGS, 1995. (Dissertação de mestrado).

15. SCHNEIDER, Fred B. **Implementing Fault-Tolerant Services Using The State Machine Approach**: A Tutorial. New York: ACM Computing Surveys, v.22, n.4, jan. 1989. p.299-319.

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

17. SUN MICROSYSTEMS. **SUN Announces FT-SPARC:** An Open, Fault-Tolerant Platform for the Telecommunications Industry. set. 1996.Disponível por http://ww.sun.com

18. THOMAS, Robert H. **A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases**. New York: ACM Transactions on Database Systems, v.4, n.2, june 1979. p.180-209.