# TOLERÂNCIA A FALHAS EM BANCOS DE DADOS DISTRIBUÍDOS COM A UTILIZAÇÃO DA TÉCNICA DE COMUNICAÇÃO DE GRUPOS DE OBJETOS TOLERANTES A FALHAS

**Marcelo Hanel**
Graduando em Ciência da Computação - Pesquisador do Curso de Ciência da Computação Instituto de Ciências Exatas e Geociências – ICEG Universidade de Passo Fundo – UPF Km 171 BR285 Bairro São José Caixa Postal 611 CEP 99001-970 Passo Fundo – RS Fone (054) 3168100 da Universidade de Passo Fundo – UPF. Rio Grande do Sul. Brasil.
e-mail: 4570@lci.upf.tche.br

**Maykel Tres**
Graduando em Ciência da Computação - Pesquisador do Curso de Ciência da Computação Instituto de Ciências Exatas e Geociências – ICEG Universidade de Passo Fundo – UPF Km 171 BR285 Bairro São José Caixa Postal 611 CEP 99001-970 Passo Fundo – RS Fone (054) 3168100 da Universidade de Passo Fundo – UPF. Rio Grande do Sul. Brasil.
e-mail: 19434@lci.upf.tche.br

**Emerson Rogério de Oliveira Junior**
Doutorando da Universidade Federal do Rio Grande do Sul - UFRGS e Professor da Universidade de Passo Fundo – UPF. Áreas de Pesquisa: Arquitetura de Computadores, Processamento Paralelo e Distribuído, Tolerância a Falhas em Sistemas Distribuídos.
e-mail: emerson@inf.ufrgs.br

## RESUMO

Com o aumento do uso de redes de computadores, aplicações que se utilizam de bancos de dados distribuídos crescem na mesma proporção em grandes corporações. Com isso, falhas de comunicação são detectadas com maior freqüência, gerando assim prejuízos para tais empresas. Visando resolver esses problemas, surge a área de tolerância a falhas e dentro desta a técnica de comunicação de grupos. Tal técnica visa a detecção e correção erros, esses decorrentes de problemas de comunicação, processamento ou equipamento.

Com o objetivo de validar tal técnica, este estudo implementa uma biblioteca de classes que tem como função detectar erros gerados em ambientes de dados distribuídos. Uma vez detectados tais falhas, a aplicação também deve corrigi-las. A ferramenta utilizada para construir os módulos que compõem a aplicação, foi a linguagem Visual J++.

Palavras-Chave: Sistema Distribuídos, Tolerância a Falhas, Redes de Computadores, Banco de Dados e Comunicação de Grupos.

## ABSTRACT

With the increase of the use of nets of computers, applications that they are used of distributed databases they grow in the same proportion in great corporations. With that, communication fault are detected with larger frequency, generating like this damages for such companies. Seeking to solve those problems, the area of fault tolerance appears you it and inside of

this the groups communication technique. Such a technique seeks the detection and correction mistakes, that current of communication problems, processing or equipment.

With the objective of validating such technique, this study implements a library of classes that has as function to detect mistakes generated in environment of distributed data. Once detected such flaws, the application should also correct them. The tool used to build the modules that compose the application, the Visual J++ language was.

Word-key: Computers Network, Distributed System, Database, Fault Tolerance and Groups Communication.

# INTRODUÇÃO

Atualmente, os sistemas distribuídos têm sido amplamente utilizados nos sistemas de computação em geral. Isto se deve, principalmente, à grande difusão do uso de redes de computadores. Neste contexto surge a área de bancos de dados distribuídos, que é utilizada para garantir a flexibilidade e segurança dos dados e informações de grandes corporações.

O principal objetivo de tais bancos de dados é fornecer o que normalmente se denomina transparência de localização. A transparência de localização significa que os usuários não tem a necessidade de saber onde se localizam os dados. Ao invés disso, todas essas informações de localização devem ser mantidas pelo sistema como parte de seu catálogo, e todas as solicitações de dados, por usuários, deveriam ser interpretadas pelo sistema de acordo com aquela informação. A solicitação pode operar das seguintes maneiras:

a) Os dados são levados até o objeto solicitante, para processamento local.
b) O processamento deve ocorrer direto na fonte do banco de dados e só o resultado do mesmo é levado ao objeto solicitante.
c) Uma combinação de “a” e “b”. Isso com o objetivo de garantir uma maior segurança na transação.

As vantagens de tal técnica são óbvias: simplifica a lógica dos programas de aplicação, e permite que os dados sejam deslocados de um nodo para outro à medida que os padrões de utilização mudarem, sem a necessidade de qualquer programação. Também vale lembrar que, este processo aumenta a segurança e o sigilo dos dados.

Bancos de dados distribuídos possuem vários tipos de técnicas de tolerância a falhas. O problema é que tais procedimentos só garantem a integridade na fonte dos dados, não se preocupando com problemas decorrentes de comunicação entre nodos, por exemplo. Com o objetivo de reduzir tais problemas, surge a área de comunicação de grupos, que é uma técnica de tolerância a falhas.

Visando observar o comportamento de aplicações que se utilizam de bancos de dados distribuídos, o presente estudo implementa ferramentas que auxiliam na recuperação e detecção de erros decorrente do processo de comunicação entre as bases de dados, estas separadas fisicamente.

# TOLERÂNCIA A FALHAS EM SISTEMAS DISTRIBUÍDOS

Existem alguns problemas em tornar um software tolerante a falhas. O estado da arte em tolerância a falhas em hardware diz que o hardware é muito confiável e sua confiabilidade continua aumentando com o passar do tempo. O software, por outro lado, não é confiável [JALOTE 94]. A principal causa de defeito em componentes de hardware tal como nodos e links é freqüentemente falha física.

Uma possibilidade para classificar as falhas em um sistema distribuído está baseada em como o componente falho se comporta em caso da falha. As falhas podem ser classificadas em 4 categorias: **crash**, **omissão**, **temporização** e falhas **bizantinas**, as quais estão representadas na figura 1 [SINGHAL 94].

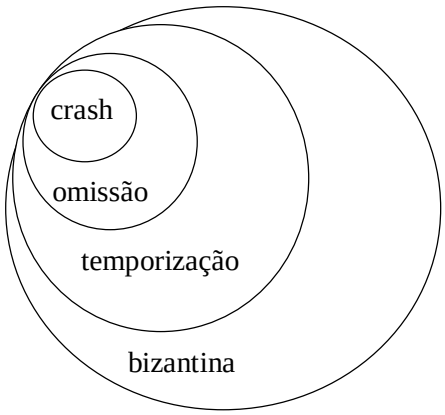


FIGURA 1: Classificação de Falhas [SINGHAL 94].

- Falha de Crash: faz com que o componente caia ou perca seu estado interno;
- Falha de Omissão: um componente não responde a algumas entradas;
- Falha de Temporização: um componente responde muito cedo ou muito tarde;
- Falha Bizantina: o componente comporta-se de uma forma arbitrária.

Estes quatro tipos de falhas formam uma hierarquia, com a falha de crash sendo a mais simples e a mais restritiva e a falha bizantina sendo menos abrangente.

# A TÉCNICA DE N-VERSÕES

Esta é uma técnica de tolerância a falhas que tem por objetivo garantir que a aplicação corrente continue sua execução normal, mesmo na presença de falhas.

É importante ressaltar também, que a técnica de N-Versões, não garante só a execução de uma aplicação na presença de falhas, mas também garante a confiabilidade dos resultados produzidos.

Este modelo consiste no uso de vários objetos que realizam a mesma tarefa, assim se um objeto sofrer falha existem outros que garantes a continuidade da aplicação.

Para adotar o resultado correto, são comparadas as respostas de todos os objetos e a resposta com o maior número de ocorrências pode ser a correta. A comparação destes resultados é de responsabilidade de um objeto específico, chamado de "votador".

Também é implementado por esta técnica o que chamamos de percentual de erro, ou seja, se o percentual estabelecido não for atingido, o objeto "votador" reenvia a tarefa para novo processamento.

Este tipo de técnica tem sido amplamente utilizada por diversos tipos de sistemas, como por exemplo, sistemas aéreos, hospitalares, militares, etc. Isso se dá em função de que a possibilidade de ocorrência de erro se torna praticamente inexistente. Como ilustrado na figura 2.

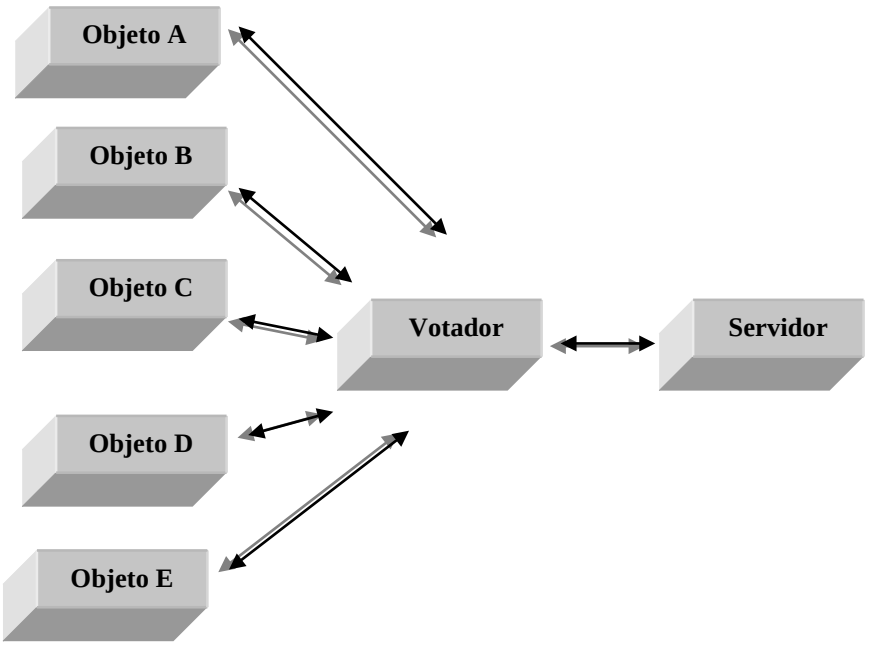


Figura 2: Técnica de N-Versões

## BANCO DE DADOS DISTRIBUÍDO

Um banco de dados distribuídos é aquele que não é armazenado em sua íntegra em uma única localização física, mas, ao contrário, está espalhado por uma rede de localizações geograficamente dispersas e ligadas por meio de links de comunicações. Naturalmente, esta definição é muito imprecisa, isto em função de que é difícil determinar com exatidão o que significa "geograficamente dispersas" e "uma única localização física". A definição seguinte é mais precisa, embora imediatamente menos compreensível: um banco de dados é distribuídos se puder ser dividido em tais partes distintas que, para um dado usuário ter acesso a algumas dessas peças é muito mais lento do que para outras. O exemplo seguinte baseia-se em outro dado por [CHAMPINE 77]. Um certo banco está localizado no estado da Califórnia, opera uma sistema em que os registros de contas para a área de Loa Angeles são ali mantidos num banco de dados e os registros de contas para a área de San Francisco são mantidos na própria San Francisco, e os dois bancos de dados são vinculados formando uma única base de dados "global" ou distribuído (Figura 3). As vantagens de tal organização são claras: combina a eficiência de processamento os dados são armazenados próximos do ponto onde são mais comumente utilizados com acessibilidade crescente (é possível

acessar uma contra de Los Angeles estando em San Francisco e vice-versa, através do link de comunicações).

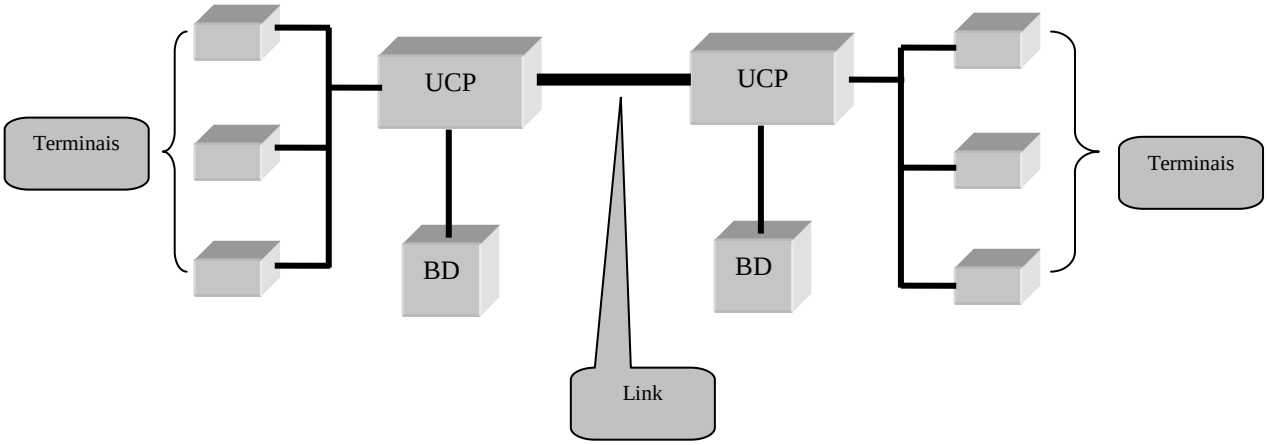


Figura 3: Banco de Dados Distribuídos

Vamos analisar este exemplo um pouco mais detalhadamente, porque ele ajudará a identificar alguns dos principais conceitos de sistemas distribuídos. Em geral, o sistema distribuído consiste em uma coleção de sites ou nodos, ligados dentro de uma rede de comunicações, (A rede do exemplo, naturalmente, é muito simples, e envolve apenas dois nodos). Cada nodo, por sua vez, constitui, um sistema de banco de dados com suas próprias regras: banco de dados, terminais, e processadores central próprios, processando seu próprio SGBD (completo com o gerente DC, o de Recuperação, o de Log, e assim por diante). Na verdade, cada nodo goza de uma grau muito alto de autonomia, e quase não depende de qualquer tipo de serviço ou controle centralizado. Assim, é mais útil, de muitas formas, pensar no sistema distribuído com uma parceria entre um conjunto de sistemas centralizados, independentes, mas cooperativos, e não como algo monolítico e indivisível, Pela mesma ótica o banco de dados distribuído pode ser considerado melhor como a união de um conjunto de bancos dados individuais centralizados.

## COMPONENTES DA APLICAÇÃO

A aplicação utilizada para a realização dos testes se compõe de quatro classes, cada uma das quais com as seguintes funções:

a) Comunica: Está classe é responsável pela comunicação entre o nodo solicitante e o banco de dados fonte. Por exemplo, estamos em San Francisco e precisamos de informações que estão em um banco de dados em Los Angeles, faz-se então um link até Los Angeles para a cópia dos referidos dados. A responsável por este link é a classe “comunica”.

b) Copia: Uma vez estabelecida a comunicação entre as duas bases de dados, entra em ação a classe “copia”, que tem por finalidade copiar os dados da base fonte para a base solicitante.

c) Votador: Esta classe tem por finalidade comparar os dados copiados pela classe acima e adotar como correto o com maior número de ocorrências.

d) Monta: Responsável pela montagem dos dados na base solicitante.

Para a construção da referida biblioteca de classes foi utilizada a ferramenta Visual J++ da Microsoft. A Base de dados, por sua vez, foi hospedada no banco de dados Access 97 também da Microsoft.

## CONSTITUIÇÃO DA REDE

O ambiente de rede utilizado para a realização dos testes constitui-se de quinze máquinas PCs, com processador Intel 300 MHZ e 128 Mb de RAM, que se utilizam de uma banda de 100 Mbits para a comunicação entre si na rede local. Para a comunicação externa existe um link dedicado de 2 Mbits. Observa-se também, que a tecnologia de rede utilizada foi a Ethernet.

## OS TESTES

Buscando obter o melhor resultado possível, os testes foram realizados em diferentes datas e horários. Isso em função de que a rede utilizada não era de domínio exclusivo da aplicação, ou seja, a mesma era ocupada por outras atividades.

Também vale lembrar que, foram realizados sete testes. O resultado, por sua vez, foi obtido pela média de tais testes.

Para a realização dos testes, foram instanciados 13 objetos da classe “copia” em máquinas distintas. Também foi instanciado um objeto da classe “votador”, este por sua vez, era responsável por reter os resultados e compará-los. Os objetos “monta” e “comunica” foram instanciados em uma máquina própria.

A aplicação utilizada nos testes tinha como objetivo buscar informações em uma base de dados remota através de um link, este procedimento transcorreu da seguinte maneira:

a) A base de dados fonte estava situada em três locais distintos, São Paulo, Brasília e Porto Alegre.

b) A base de dados solicitante estava localizada em Passo Fundo.

c) Todos os objetos responsáveis pela cópia dos dados nas bases fontes, foram instanciados em máquinas localizadas em Passo Fundo.

d) O processo de testes durou 7 dias e foram copiados 2.415.600 registros, gerando com isso, um banco de dados com 50 Mbytes.

e) A classe “comunica” gerava os links entre as bases de dados, isso para que as cópias fossem possíveis.

f) Para o processo de cópia foi criado um grupo com 13 PCs, onde cada máquina realizava a mesma tarefa e enviava o resultado para o objeto “votador”.

g) O objeto “votador”, por sua vez, comparava os dados recebidos e enviava o correto para a classe “monta”.

h) A “votador” adotava o resultado correto através da comparação, ou seja, a resultado com o maior percentual de ocorrências era tido como correto.

i) Forma detectados ao longo desse processo 35.509 registros com falha ou seja, 1,47% do total resultou em falhas decorrentes do processo de comunicação.

Vale lembrar que, na prática os erros detectados não tiveram influência alguma na integridade do banco de dados, isto em função da utilização da técnica de comunicação de grupos.

## CONCLUSÕES

Após a análise dos resultados produzidos pelos testes, chega-se a algumas conclusões.

Primeiramente, observa-se uma deficiência na comunicação entre ambientes remotos de redes de computadores, ou seja, o percentual de erro detectado é significativo para uma organização, pois o mesmo pode influenciar diretamente na base de dados.

Também vale lembrar que o uso de aplicações distribuídas torna o trabalho muito mais seguro, em contra partida, mais lento. Isto deve-se ao fato de que a comunicação de grupos gera um grande tráfego de rede, já que vários objetos realizam o mesmo trabalho.

Assim sendo, concluímos que a técnica de comunicação de grupos pode garantir que, uma aplicação que se utilize de um banco de dados distribuídos seja confiável, ou seja, os resultados obtidos, são na maioria da vezes, os mesmo que os esperados.

## BIBLIOGRAFIA


[ALSBERG 76] ALSBERG, P. A. and DAY, J.D. **A Principle for Resilient Sharing of Distributed Resources**. Proc. of the Second Intern. Conf. on Software Engineering. San Francisco,CA. p.562-570, 1976.

[ARNOLD 97] ARNOLD, K. **The Java Programming Language**. Massachussets, EUA. Addison-Wesley. 1997. 442p.

[BAL 89] BAL, H. E. , STEINER, J. G. and TANENBAUM, A. S. **Programming Languages for Distributed Computing Systems**. New York. ACM Computing Surveys, v.21, n.3, p.261-320, Sept. 1989.

[BARTLETT 81] BARTLETT, J. F. **A NonStop Kernel**. Proceedings of the Eighth Symposium on Operating Systems Principles. In ACM Operating Systems Review. V.15, n.5, 1981.

[BHIDE 91] BHIDE, A. ELNOZAHY, E. N. , MORGAN, S.P. **A Highly Avaiable Network File Server**. Proc. of the USENIX. P.199-205, 1991.

[BUDHIRAJA 92] BUDHIRAJA, N. and MARZULLO, K. **Tradeoffs in Implementating Primary-Backup Protocols**. Department of Computer Science, Cornell University. Tech. Report TR 92-1307, Ithaca, Ny, 1992.

[BUDHIRAJA 93] BUDHIRAJA, N. et al. **The Primary-Backup Approach**. In: Distributed Systems. ACM Press. New York, p.199-216, 1993.

[ECKEL 98] ECKEL, B. **Thinking in Java**. N.J., EUA. PrenticeHall PTR. 1998. 1098p.

[GEIST 94] GEIST, A.et al. **PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Networked Parallel Computing**. Cambridge, Massachussets: The MIT Press, 1994.

[GHOSH 97] GHOSH, S. et al. Fault-Tolerance Through Scheduling of Aperiodic Tasks in Hard Real-Time Multiprocessos Systems. **IEEE Trans. on Parallel and Distributed Systems**. New York, v. 8, n. 3, p.272-284, mar. 1997.

[GUERRAOUI 97] GUERRAOUI, R. and SCHIPER, A. **Software-Based Replication for Fault Tolerance**. In: IEEE Computer. New York, v.30, n.4, p.68-74, Apr.1997.

[HANEL 99] HANEL, M.; TRES, M.; OLIVEIRA JR. E. R. Comunicação entre Objetos Distribuídos Replicados Tolerantes a Falhas. (Artigo publicado no CACIC99 – Congreso Argentino de Ciencias de la Computación, Tandil, Buenos Aires, outubro 1999).

[JALOTE 94] JALOTE, P. **Fault Tolerance in Distributed Systems**. New Jersey: PTR Prentice Hall, Englewood Cliffs, 1994.

[KORTH 95] KORTH, H. F. and SILBERSCHATZ, A. **Sistemas de Banco de Dados.** Makron Books, São paulo, 1995.

[MULLENDER 95] MULLENDER, S. **Distributed Systems**. Addison Wesley Publishing Company. ACM Press. New York. 1995.

[OLIVEIRA 98] OLIVEIRA JUNIOR, E. R. Replicação de Objetos em um Sistema Distribuído. In: SBAC-PAD, 10., 1998, Buzios, BRJ. **Anais...** Rio de Janeiro: COPPE / UFRJ, 1998. P.157-160.

[OLIVEIRA 99] OLIVEIRA JR, E.R. **Replicação de Objetos Distribuídos no DPC++.** Porto Alegre: PPGC/UFRGS,1999. 95p. Dissertação de Mestrado.

[SHAMPINE 77] SHAMPINE G. A. **Six Approaches to Distribuited Databases.** Long Beach: IEEE Computer Society, 1977

[SINGHAL 94] SINGHAL, M. et al. **Advanced Concepts in Operating Systems: Distributed, Database and Multiprocessor Operating Systems**. New York: McGraw-Hill, 1994.

[TANENBAUM 81] TANENBAUM, S. **Computer Network,** Prentice-Hall, Englewood Clifs, NJ, 1981.