GERENCIADOR DE DADOS REPLICADOS
PARA CENTROS DE SUPERVISÃO E CONTROLE BASEADOS EM UMA
ARQUITETURA DISTRIBUíDA

Antonio Joaquim Setubal de Rezende Silva

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE POS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSARIOS A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.) EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO

Aprovada por:

Prof. Valmir Carneiro Barbosa
(Presidente)

Prof. Amauri Marques da Cunha

Profª. Eliana Prado Lopes Aude

Engº. Maurício Moszkowicz

RIO DE JANEIRO, RJ - BRASIL
ABRIL DE 1988

SILVA, ANTONIO JOAQUIM SETUBAL DE REZENDE

Gerenciador de Dados Replicados para Centros de Supervisão e Controle baseados em uma Arquitetura Distribuída (Rio de Janeiro) 1988.

XII, 168 p. 29.7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1988)

Tese - Universidade Federal do Rio de Janeiro, COPPE.

1. Bancos de Dados I. COPPE/UFRJ II. Título (Série)

Agradeço à direção do Centro de Pesquisas de Energia Elétrica - CEPEL pela oportunidade de realização do presente trabalho.

Aos meus colegas e amigos do Departamento de Eletrônica do CEPEL, aos engenheiros das Centrais Elétricas do Sul do Brasil e aos engenheiros de Furnas Centrais Elétricas pela participação no projeto que originou este trabalho.

A Maurício Moszkowicz pela coordenação deste trabalho.

Ao colega George Gerber do Departamento de Eletrônica do CEPEL pela colaboração prestada.

Ao Prof. Gerhard Schwarz pela colaboração prestada no início deste trabalho.

Resumo da Tese Apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

GERENCIADOR DE DADOS REPLICADOS
PARA CENTROS DE SUPERVISÃO E CONTROLE BASEADOS EM UMA ARQUITETURA DISTRIBUíDA

Antonio Joaquim Setubal de Rezende Silva
Abril de 1988

Orientadores: Amauri Marques da Cunha
Núcleo de Computação Eletrônica - UFRJ
Valmir Carneiro Barbosa
Engenharia de Sistemas e Computação - COPPE/UFRJ

Este trabalho analisa a utilização de um Gerenciador de Banco de Dados Replicados em uma classe de sistemas em tempo real, baseada em uma arquitetura distribuída, aqui denominada Centro de Controle. A colocação do Gerenciador como um elemento do "software" básico do Centro de Controle é justificada como forma de reduzir os custos do desenvolvimento de "software" aplicativo tolerante a falhas, porquanto este Gerenciador representa o encapsulamento em uma solução padronizada de problemas como Controle de Acesso Concorrente e Recuperação de Falhas. A partir dos requisitos do Banco de Dados do Centro de Controle, são derivadas as principais características do Gerenciador e é proposto um algoritmo para controle de concorrência e recuperação de falhas parciais baseado em um protocolo de Difusão Confiável. Um detalhado modelo do protocolo é apresentado utilizando Redes de Petri.

Abstract of Thesis presented to COPPE/UFRJ in partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

REPLICATED DATA MANAGER
FOR SUPERVISORY AND CONTROL CENTRES BASED ON A
DISTRIBUTED ARCHITECTURE

Antonio Joaquim Setubal de Rezende Silva
April, 1988

Chairmen: Amauri Marques da Cunha
Nucleus of Eletronics Computing - UFRJ
Valmir Carneiro Barbosa
Computing and Systems Engineering - COPPE/UFRJ

This report focuses on a Manager for Replicated Data Bases applied to a class of real-time systems based on a distributed architecture, henceforth referred to as Control Centre. By including the Manager in the basic software of the Control Centre, the costs of developing fault-tolerant applications can be reduced, since it represents an encapsulation of standard solutions to problems like concurrent access control and error recovery. The main features of the Manager are derived from the requirements of the Control Centre data base, and an algorithm is proposed for concurrency control and recovery from partial failures. This algorithm builds upon a reliable broadcast protocol. A detailed model of the protocol is presented using Petri nets.

ÍNDICE

Capítulo I - Introdução 1

1. Introdução 2
   1.1 Definição e Localização de um Centro de Controle 2
   1.2 Distribuição e Modularidade 5
   1.3 Padronização e Redução do Custo do "Software" Aplicativo 6

2. Arquitetura do Centro de Controle 7
   2.1 Concepção 7
   2.2 "Hardware" 8
   2.3 "Software" Básico 9
   2.4 "Software" Aplicativo 10
   2.5 Subsistemas 11

3. Descrição dos Capítulos 12

Capítulo II - Conceitos Básicos e Alternativas Pesquisadas 16

1. Problemas Básicos em Bancos de Dados Distribuídos 17

2. Reavaliação das Funções de um Gerenciador de Banco de Dados Genérico segundo as Necessidades do Centro de Controle 18
   2.1 Funções 18
   2.2 Gerenciamento do Esquema Externo em um Centro de Controle 19
   2.3 Recuperação de Inconsistências 20
   2.4 Segurança e Integridade 20
3. Concorrência e Recuperação de Falhas 21
   3.1 O Conflito entre requisitos de Sistemas Comerciais e Sistemas em Tempo Real 21

3.2 Direcionamento da Pesquisa 22
3.3 Serialização e Controle de Acesso 23
3.3.1 Arquitetura de um Gerenciador de Banco de Dados Distribuídos 23
3.3.2 Classificação dos Algoritmos de Controle de Acesso 25
3.3.3 Serialização 25
3.3.3.1 Equivalência de Logs 26
3.3.3.2 Logs Serializáveis e Logs Corretos 27
3.3.3.3 Teorema da Serializabilidade 27
3.3.4 Estratégias de Escalonamento 28
3.3.4.1 Bloqueio em Duas Fases 29
3.3.4.2 Ordenação por Carimbo de Tempo 30
3.3.5 Distribuição do Gerenciador de Acesso 32
3.3.6 Replicação de Dados 33
3.3.6.1 Replicação Trivial: Lê Alguma e Atualiza Todas 33
3.3.6.2 Cópia Principal 34
3.3.6.3 Votação 35
3.4 Protocolos e Algoritmos Distribuídos 35
3.4.1 Algoritmos Distribuídos para Exclusão Mútua 40
3.4.1.1 Trabalhos Publicados 41
3.4.1.2 O Algoritmo de RICART e AGRAWALA - 84 42
3.4.1.3 Exclusão Mútua na presença de Falhas 43
3.4.2 Algoritmos para Difusão Confiável 44
3.4.2.1 Relação com o presente Trabalho 44

Capítulo III - Especificação do Problema: Banco de Dados e Transações em um Centro de Controle 47

1. Banco de Dados de um Centro de Controle 48
1.1 Banco de Dados do Processo 48
1.1.1. Características 49
1.1.1.1 Imagem do Processo Elétrico 52
1.1.1.2 Imagem de Processos Internos ao Centro de Controle 52
1.1.1.3 Variáveis Elaboradas 53
1.1.1.4 Alarmes 54

1.1.1.5 Eventos 55
1.1.1.6 Hora da Atualização 56
1.1.1.7 Parâmetros de Tratamento 57
1.1.2 Configuração 57
1.1.3. Interfaces 57
1.1.3.1 Interface com Outros Módulos 57
1.1.3.2 Interface com o Processo 58
1.1.4. Entidades e Relacionamentos 59
1.1.5 Transações e Organização Física do BDP 63
1.1.5.1 Atualização das Variáveis Primárias e Eventos 61
1.1.5.2 Atualização das Variáveis Elaboradas Globais 63
1.2 Banco de Dados Históricos 65
1.3 Banco de Dados Aplicativos 65
1.4 Banco de Dados Estáticos 66

2. Replicação do Banco de Dados 66
2.1 Replicação dos Dados do Processo 67
2.2 Replicação dos Dados Históricos 67
2.3 Replicação dos Dados Aplicativos 67
2.4 Replicação dos Dados Estáticos 67

3. Arquiteturas Típicas 68
3.1 Centro de Controle Local ao Processo 68
3.2 Centro de Controle Global a Vários Processos 70

4. Relação dos Requisitos para o Gerenciador de Dados Replicados 71

Capítulo IV - Proposta Básica 75

1. Discussão Preliminar 76
1.1 Suposições quanto aos Tipos de Faltas 76
1.2 Modelo do Banco de Dados Replicados e um Algoritmo Trivial 77
1.2.1 Modelo 77
1.2.2 Suposições Simplificadoras Sobre o Sistema 78

1.2.3 Um Algoritmo Trivial 79
1.3 Análise de Possíveis Otimizações 81
1.3.1. Registro Seqüencial 81
1.3.2 Análise da Etapa de Validação 82
1.3.3 Análise do Controle da Concorrência de Acesso a Itens do BDR 83

2. Estratégia Utilizada no Gerenciador 84

Capítulo V - Gerenciador de Banco de Dados Replicados 87

1. Arquitetura do Gerenciador 88

2. Gerenciador de Transações 92
2.1 Transações 92
2.1.1 Iniciar Transação 92
2.1.2 Abrir Arquivo 93
2.1.3 Bloquear Dado 95
2.1.4 Ler Dado 96
2.1.5 Escrever Dado 97
2.1.6 Finalizar Transação 98

3. Gerenciador do Acesso Concorrente 98
3.1 Solicitações ao GA 99
3.1.1 Iniciar Transação 100
3.1.2 Abrir Arquivo 100
3.1.3 Bloquear Dado 101
3.1.4 Abortar Transação 101
3.1.5 Finalizar Transação 102
3.1.6 Ler Dado Remoto 102
3.1.7 Desativar Estação 103
3.1.8 Fornecer Contexto 103

4. Falha e Recuperação de Estações 103

Capítulo VI - Protocolo de Difusão Confiável 105

1. Introdução 106

2. Seleção do Protocolo de Difusão Confiável 107

3. O Algoritmo de CHANG e MAXEMCHUK - 84 107
   3.1 Funcionamento da Fase Normal 110
   3.2 Funcionamento da Fase de Reforma 115

4. Vantagens e Problemas 118

5. Alterações Introduzidas 119

Capítulo VII - Conclusões e Perspectivas 120

1. Conclusões 121

2. Futuros Desenvolvimentos 121

BIBLIOGRAFIA 123

APÊNDICE : Modelo do Protocolo de Difusão Confiável 129
   A. Modelo 130
   A.1 Rede de Petri Interpretada 130
      A.1.1 Definição 130
      A.1.2 Funcionamento 131
   A.2 Descrição do Modelo do Protocolo
      de Difusão Confiável 131
      A.2.1 Formatos das Mensagens e Convenções
         Globais Auxiliares 132
      A.2.2 Estados do Protocolo 134
      A.2.3 M: Número de Seqüência das
         Mensagens a Receber 135
      A.2.4 PCT: Próximo Carimbo de Tempo 136

A.2.5 Ver: Versão corrente do Grupo de Difusão 136
A.2.6 O depósito de Dados Qb 136
A.2.7 A fila de Controle Qc 137
A.2.8 Primitivas do Protocolo de Difusão Confiável oferecidas aos Usuários 139
A.2.9 Primitivas Externas utilizadas pelo DC 139
A.2.10 Primitivas de Verificação das Filas Qc e Qb 140
A.2.11 Fases e Seções do Protocolo de Difusão Confiável 141
A.2.11.1 Modelo da Seção Mestre 142
A.2.11.2 Modelo da Seção Escravo 147
A.2.11.3 Modelo da Fase Normal 152

GLOSSARIO 165

FIGURAS

I-1	Centros de Controle	4
I-2	Descrição dos Capítulos	13
III-1	BDP e demais Módulos	51
III-2	Entidades e Relacionamentos do BDP	60
III-3	BDP para um Centro de Controle Local	69
III-4	BDP para um Centro de Controle Global	70
V-1	Arquitetura do GBDR	89
V-2	Três Estações e Dois BDRs	91
VI-1	Protocolo de Difusão Confiável	108
A-1	Protocolo de Difusão Confiável	142
A-2	Seção Mestre da Reforma	143
A-3	Seção Escravo da Reforma	148
A-4	Fase Normal	153

Capítulo I

Introdução

1. Introdução 2
   1.1 Definição e Localização de um Centro de Controle 2
   1.2 Distribuição e Modularidade 5
   1.3 Padronização e Redução do Custo do "Software" Aplicativo 6

2. Arquitetura do Centro de Controle 7
   2.1 Concepção 7
   2.2 "Hardware" 8
   2.3 "Software" Básico 9
   2.4 "Software" Aplicativo 10
   2.5 Subsistemas 11

3. Descrição dos Capítulos 12

1. Introdução

1.1. Definição e Localização de Um Centro de Controle

Um Centro de Supervisão e Controle, ou simplesmente Centro de Controle, é um sistema computacional constituído de equipamentos de interface homem-máquina, equipamentos de comunicação com outros Centros de Controle e comunicação com equipamentos que atuam diretamente sobre um determinado processo industrial.

O Centro de Controle é dedicado à realização das funções de: aquisição de dados; monitoração, controle e análise de processos; interface homem-máquina; e comunicação com outros Centros de Controle.

Neste trabalho, o Centro de Controle é apresentado como um elemento celular que aparece em vários níveis na hierarquia de sistemas computacionais de controle da produção e transmissão de energia elétrica. Estes níveis incluem: Centros diretamente ligados à operação das Usinas e Subestações; Centros Regionais e Centros de Despachos de Energia Elétrica das Empresas Concessionárias; e o Centro Nacional responsável pela coordenação e otimização do processo global.

Dessa forma, o Processo Supervisionado ou Processo Elétrico, mencionado no decorrer deste trabalho, refere-se, com variados graus de detalhamento (local, regional ou nacional), ao Processo de Geração e Transmissão de Energia Elétrica.

Um Centro de Controle possui, tipicamente, interfaces com três outros sistemas:

a - Rede de Terminais de Medição e Controle: equipamentos diretamente ligados ao Processo Elétrico que se comunica com o Centro de Controle através de comportas ("Gateways") e eventualmente modems e canais de microondas.

b - Centro Hierarquicamente Superior: outro Centro de Controle localizado em um nível hierárquico superior.

c - Equipe de Operação do Centro de Controle: operadores e despachantes responsáveis por realizar a supervisão e controle do Processo Elétrico.

Uma hierarquia de Centros de Controle é apresentada na Figura I-1. Os elementos dessa figura serão descritos ao longo deste Capítulo.

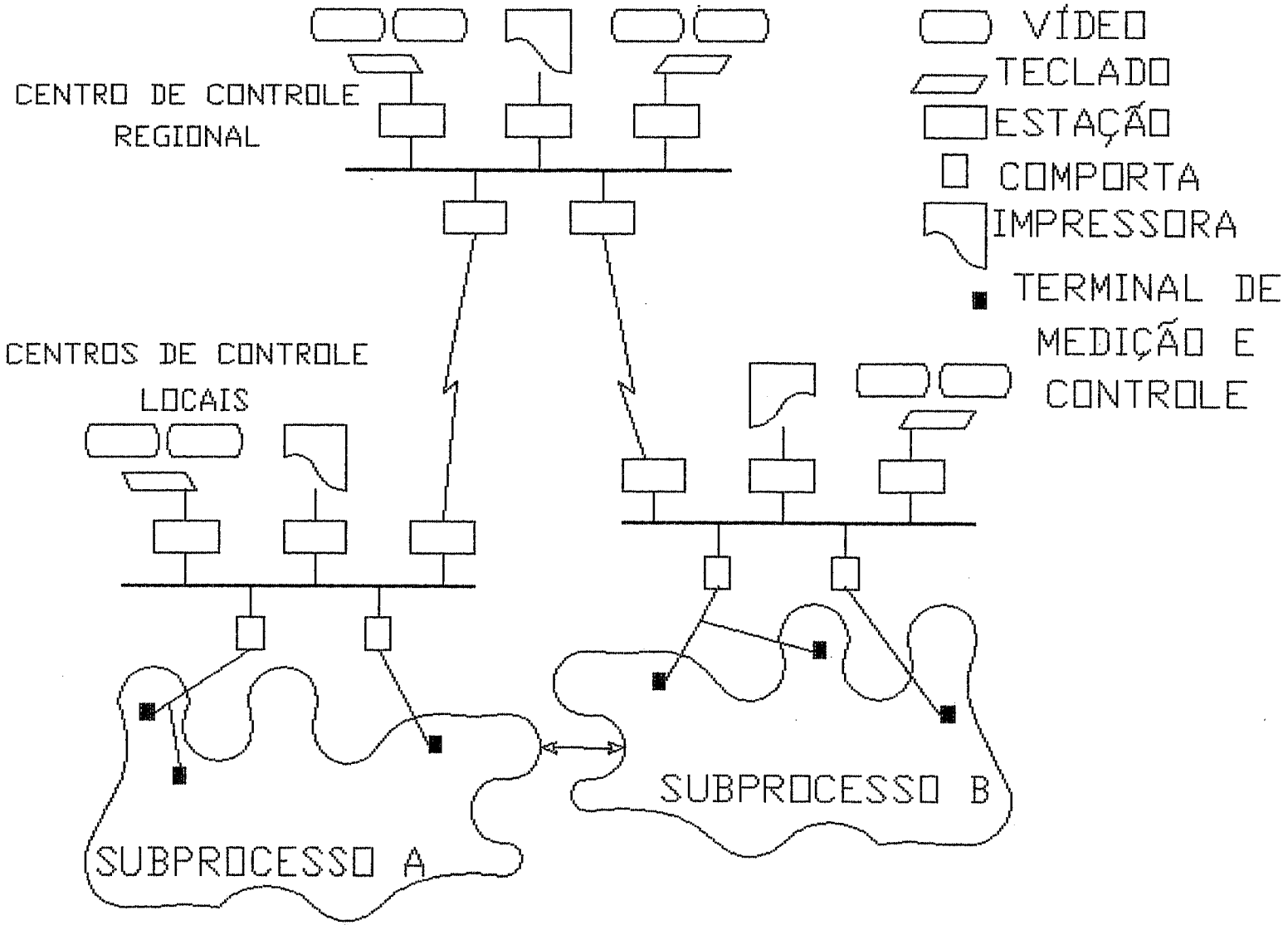


Figura I-1 : Centros de Controle

1.2 Distribuição e Modularidade

A flexibilidade de um sistema é uma medida da sua capacidade de adaptar-se a uma vasta gama de aplicações. Num sistema modular, a flexibilidade é obtida a partir de um processo simples de composição de um número variável de estruturas celulares ou módulos.

Um sistema computacional modular pode ser construído a partir de um conjunto de unidades de processamento interligadas por uma via de comunicação e a partir da distribuição de tarefas pelas unidades de processamento. Tal sistema propicia a flexibilidade sob dois aspectos: disponibilidade e capacidade de processamento.

O aumento da disponibilidade pode ser obtido através da replicação de unidades de processamento dedicadas a uma mesma tarefa. A replicação aliada a uma política adequada de manutenção das unidades defeituosas permite a obtenção do coeficiente de disponibilidade estabelecido para o sistema.

A capacidade de processamento necessária a determinada tarefa pode pode ser obtida através do seu desmembramento em subtarefas e repartição destas por várias unidades de processamento. Repartição e replicação de tarefas em um sistema computacional são técnicas utilizadas no desenvolvimento de "software" distribuído.

A distribuição, entretanto, traz uma série de problemas a nível de desenvolvimento de "software", como por exemplo: teste de algoritmos distribuídos e protocolos; controle de acesso concorrente sobre recursos distribuídos; e recuperação de falhas parciais. A solução desses problemas torna o "software" mais complexo e conseqüentemente aumenta o custo do seu desenvolvimento.

1.3 Padronização e Redução do Custo do "Software" Aplicativo

O objetivo deste trabalho é analisar a possibilidade de se utilizar um Gerenciador de Banco de Dados Replicados (GBDR), que é um caso particular de um Gerenciador de Banco de Dados Distribuídos (GBDD), para uma classe de Sistemas em Tempo Real, aqui identificada como Centro de Controle. A colocação do Gerenciador como um elemento do "software" básico do Centro de Controle é justificada como forma de reduzir os custos do desenvolvimento de "software" aplicativo tolerante a falhas, porquanto este Gerenciador representa o encapsulamento em uma solução padronizada de problemas como Controle de Acesso Concorrente e Recuperação de Falhas.

Deve-se notar que a alternativa à utilização de um GBDR é seguir uma abordagem casuística. Isto é, para a solução de problemas como controle da concorrência de acesso aos dados e recuperação de falhas parciais do "hardware", adotarem-se soluções fortemente dependentes de cada aplicação. Essa alternativa é extremamente custosa e perecível e, portanto, desaconselhável. Em contrapartida, a alternativa de se utilizar um GBDR pode comprometer os tempos de resposta do sistema em relação à alternativa casuística. Este trabalho busca, portanto, estabelecer sob que condições é viável a utilização de um gerenciador e como as funções de um GBDD genérico podem ser mapeadas nos problemas encontrados nos Centros de Controle.

A razão deste trabalho propor um GBDR, e não um GBDD englobando replicação e repartição, é que, como se verá mais adiante, as otimizações que viabilizam a replicação em tempo real não podem ser estendidas ao caso geral de distribuição. Ou seja, o GBDR é uma solução particular, mas que aborda importantes problemas encontrados em Centros de Controle baseados em arquiteturas distribuídas.

A reunião de um conjunto de problemas e a introdução de uma ferramenta que permita sua solução de uma maneira sistemática são esforços no sentido de possibilitar a abstração do ambiente distribuído e com isso reduzir o custo do desenvolvimento de

"software" aplicativo para o Centro de Controle. Uma ferramenta de "software" deve implementar conceitos simples e poderosos que inspirem o Analista durante a concepção de um sistema. No caso do GBDR, o conceito fundamental é a possibilidade de se criar no Centro de Controle repositórios de informações que sejam seguros (tolerantes a falhas parciais) e onipresentes. Esta última característica torna o Banco de Dados um elemento de difusão de informações. A ferramenta deve possibilitar que essa criação limite-se à declaração do tipo do repositório. Ou seja, a ferramenta permite a abstração dos detalhes de implementação do conceito.

Além da redução do custo do desenvolvimento de "software" aplicativo, através da utilização de uma ferramenta padrão, a introdução de um gerenciador é um esforço de síntese e formalização, no sentido de conceituar matematicamente os problemas e técnicas associadas à manipulação de dados replicados em um Centro de Controle. Essa formalização é a tendência seguida por todas as soluções de engenharia sempre que avanços tecnológicos permitem absorver a sobrecarga introduzida pela formalização, como é colocado por MELLOR et al [34]. Neste caso, o advento de microcomputadores de baixo custo (US$ 5,000.00), com capacidade de endereçamento de dezenas de "Megabytes" de memória real, processando até 4 Mips e armazenando centenas de "Megabytes" em memória secundária, constitui um avanço tecnológico que certamente assimila propostas como a que é feita neste trabalho. Tais dados são apresentados em [25].

2. Arquitetura do Centro de Controle

2.1 Concepção

A solução clássica adotada para Centros de Controle, descrita por FROST et alii [22] e por JERABEK et al [26], tem sido utilizar uma configuração dual de minicomputadores: um ativo e outro em reserva ativa, isto é, atualizando-se periodicamente com os dados vitais do sistema. O barramento é dividido em

barramento local e barramento compartilhado. O barramento local interliga os equipamentos periféricos exclusivos de uma CPU. O barramento compartilhado é chaveável, em caso de falha, entre as duas CPUs. A este barramento estão ligados os controladores de comunicação com terminais remotos de medição e controle, os controladores de comunicação com outros Centros e periféricos de interface homem-máquina.

Em Centros de Controle de níveis mais altos, outra configuração dual de minicomputadores, dedicada às funções de telemedição e comunicação com os terminais remotos, é conectada ao barramento dos minicomputadores principais como equipamento "front-end".

Neste trabalho, os Centros de Controle geograficamente próximos ao Processo são denominados Centros de Controle Locais. Os Centros que englobam vários processos, geograficamente distantes entre si, são denominados Centros de Controle Globais.

A filosofia adotada para a arquitetura do Centro de Controle, objeto deste trabalho, é substituir as funções realizadas pelos minicomputadores da configuração clássica por um conjunto de microcomputadores, cada qual dedicado a um pequeno conjunto de tarefas e à cooperação com os demais microcomputadores. Para tanto, os equipamentos periféricos (terminais de vídeo e impressoras) são repartidos entre os microcomputadores segundo suas funções.

Esta concepção é reforçada por um fator importante do ponto de vista político: a autonomia adquirida pela indústria nacional na fabricação de equipamentos de baixo custo e com enorme potencial de expansão da capacidade de processamento, armazenamento e comunicação homem-máquina.

2.2 "Hardware"

O "hardware" do Centro de Controle, a fim de apresentar um alto grau de modularidade, é baseado em uma rede local de

microcomputadores, aqui denominados Estações, que repartem entre si as tarefas de processamento e comunicação com a Rede de Terminais de Medição e Controle, comunicação com outros Centros e comunicação com os Operadores ou Despachantes. A comunicação homem-máquina é feita via teclados, vídeo, impressoras, painéis mímicos e registradores gráficos.

A rede local é um barramento serial utilizando uma taxa de transmissão de 2 Mbit/s em banda básica ("carrier band"). O controle de acesso utiliza o CSMA/CD por ser intrinsecamente descentralizado e facilmente implementável.

2.3 "Software" Básico

O "software" Básico do Centro de Controle é o conjunto de módulos de programa de uso geral, adequados a qualquer sistema computacional, baseado em uma arquitetura distribuída com funções de supervisão e controle.

O "software" básico das Estações é composto dos seguintes módulos:

a - Núcleo do Sistema Operacional Multitarefa para Tempo Real:

este módulo permite implementar um ambiente de multiprogramação, interno a cada Estação, e fornece funções primitivas de comunicação e sincronismo entre tarefas.

b - Sistema de Comunicação entre Estações:

este módulo permite a comunicação confiável entre tarefas localizadas em Estações distintas e estende às demais Estações o acesso remoto a dispositivos de armazenamento ou de Entrada e Saída.

c - Gerenciador dos Dispositivos de Memória de Massa:

este módulo permite a manipulação uniforme de dispositivos de memória de massa pertencentes à Estação como Unidades de Disco Magnético.

d - Gerenciador de Banco de Dados Replicado:

este é o módulo, objeto da presente análise, que deve permitir a implementação do conceito de repositórios de informações seguros e presentes em todas as Estações.

e - Sistema de E/S para Equipamentos Periféricos:

este módulo uniformiza a forma de entrada e/ou saída em equipamentos como Painéis de Teclas, Impressoras e Terminais de Vídeo.

f - Sistema de Comunicação Remota com outros Centros de Controle:

este módulo permite a comunicação confiável entre tarefas localizadas em Centros de Controle distintos.

g - Banco de Dados Dinâmicos do Processo Supervisionado:

este módulo implementa uma interface entre os módulos do "software" aplicativo e o Processo Supervisionado sob a forma de um Banco de Dados constituído pelas variáveis do processo.

Uma descrição mais detalhada desses módulos pode ser encontrada em [42].

2.4 "Software" Aplicativo

O "software" Aplicativo do Centro de Controle é o conjunto de módulos de programa extremamente dependente dos requisitos funcionais de cada usuário do sistema computacional.

O "software" Aplicativo de um Centro de Controle para o Processo Elétrico pode ser dividido em três grandes classes funcionais, relacionadas por MOONEY et al [35]:

a - Funções de Aquisição, Monitoração e Controle ("SCADA" e Controle Automático de Geração). Uma descrição das funções de Aquisição e Controle para Centros Regionais pode ser encontrada em [43]. Em particular, quatro destas funções

aplicativas serão mencionadas durante a descrição do Banco de Dados do Centro de Controle no Capítulo III:

.Função Diálogo - responsável pela recepção das solicitações dos operadores e despachantes do Centro de Controle, emissão de críticas e disparo das demais Funções de modo a atender as solicitações.

.Função Acompanhamento - responsável pela apresentação contínua da evolução do estado do processo através de telas de vídeo.

.Função Alarme - responsável pela detecção de mudanças alarmantes verificadas no processo e apresentação destas em vídeos e impressoras.

.Função Controle - responsável pela emissão de comandos para os equipamentos do processo a partir de solicitações dos operadores.

b - Funções Avançadas de Análise e Simulação (Análise de Falhas, Análise de Rede, Análise de Segurança e Programação da Geração).

c - Funções específicas e freqüentemente alteradas pelo usuário (relatórios de análises de tendências, relatórios estatísticos).

O detalhamento dos aspectos das Funções Aplicativas que influenciam o GBDR será feito no Capítulo III.

2.5 Subsistemas

O Centro de Controle pode ser, tipicamente, formado por três subsistemas em função dos seus usuários finais:

a - Subsistema de Supervisão e Controle Local: interage com a equipe de operadores e despachantes.

b - Subsistema de Tele-Supervisão e Controle: interage com outro Centro de Controle de um nível hieráquico superior.

c - Subsistema de Supervisão do Sistema Computacional: interage com a equipe de manutenção do próprio sistema computacional.

Estes subsistemas devem operar independentemente sob os seguintes aspectos:

a - Em caso de falha total de um subsistema, os demais devem poder continuar operando. Isto é, nenhum dos subsistemas é um elemento central.

b - Em caso de sobrecarga de processamento solicitada a um dos subsistemas, os demais não devem ser penalizados.

O Banco de Dados do Centro de Controle deve poder adaptar-se aos requisitos de cada um dos subsistemas e ainda assim conservar sua independência. A arquitetura desse Banco de Dados é apresentada no Capítulo III.

3. Descrição dos Capítulos

A seguir será feita uma breve descrição, esquematizada na Figura I-2, dos próximos Capítulos deste trabalho.

No Capítulo II é apresentada uma série de definições, conceitos e técnicas das áreas de Bancos de Dados, Algoritmos Distribuídos e Protocolos. Os conceitos e técnicas são analisados quanto a sua importância na composição de uma solução para o problema global descrito no Capítulo I: obtenção de repositórios de informações seguros e onipresentes através de um GBDR. Inicialmente, são relacionados os principais problemas associados a Bancos de Dados Distribuídos. O problema de controle de acesso concorrente é considerado o principal no contexto de Bancos de Dados Replicados para sistemas em Tempo Real. As principais funções de um Gerenciador de Banco de Dados genérico são também reavaliadas sob a ótica do Centro de Controle. A seguir, de modo a se classificar os diversos algoritmos para controle de acesso, são apresentados os

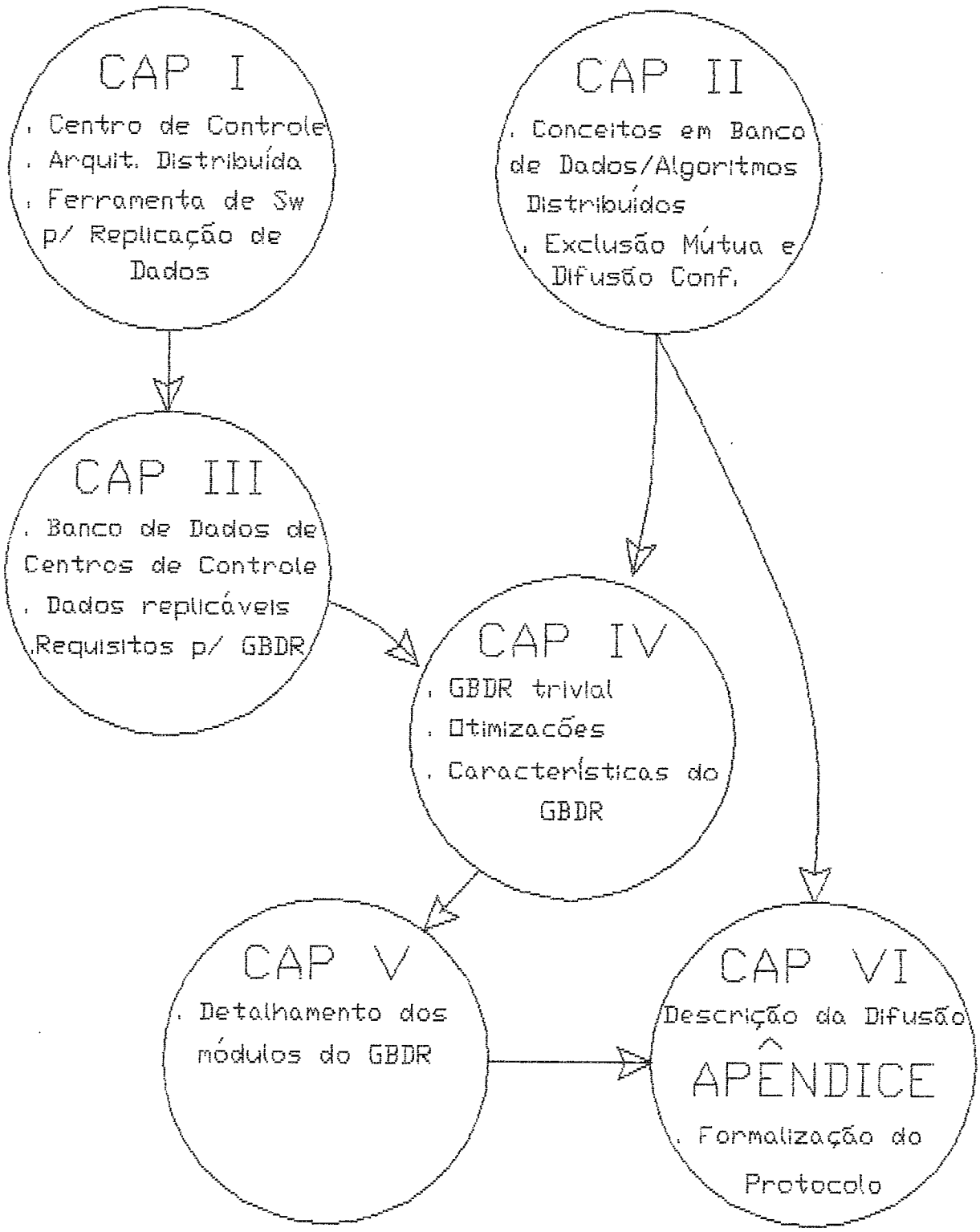


Figura I-2: Descrição dos Capítulos

conceitos e resultados básicos da Teoria da Serialização. A classificação dos algoritmos é feita segundo a estratégia de escalonamento, distribuição do gerenciador de acesso e forma de replicação. Com o propósito de examinar alternativas para assegurar a tolerância a falhas parciais, alguns algoritmos de Exclusão Mútua são discutidos. Neste caso, a Exclusão Mútua é apresentada como a forma de alguma Estação obter o privilégio de reorganizar o sistema após a falha de outra Estação. Finalmente, são analisadas algumas simplificações obtidas em um Gerenciador de Bancos de Dados Distribuído a partir do conceito da Difusão Confiável.

No Capítulo III, o Banco de Dados de um Centro de Controle típico é descrito de modo a se poder quantificar requisitos para o GBDR. O Banco de Dados é subdividido em quatro classes: Dados do Processo, Dados Históricos, Dados Aplicativos e Dados Estáticos. Sobre essa divisão, a replicação é analisada como meio de fornecer tolerância a falhas e difusão de um item de dado de determinada classe. A análise revela que são os Dados do Processo que influenciam mais pesadamente os requisitos para o GBDR. A seguir duas arquiteturas típicas para a replicação dos Dados do Processo são apresentadas. Finalmente, é apresentada uma relação de requisitos para o GBDR.

No Capítulo IV, os conceitos descritos no Capítulo II e os requisitos extraídos do Capítulo III são utilizados de modo a se obter uma proposta para os algoritmos de controle de acesso e recuperação de falhas do GBDR. O Capítulo se desenvolve a partir de uma proposta trivial que é discutida e otimizada. As otimizações propostas baseiam-se em: cada Estação utiliza sua própria réplica do Banco de Dados como rascunho durante uma Transação; implementação da Etapa de Validação de uma Transação baseada na Difusão Confiável; e processamanto replicado do algoritmo de controle de acesso também baseado na Difusão Confiável. Finalmente, são relacionados os princípios que compõe a estratégia utilizada no GBDR que incluí, entre outros, controle de acesso por Bloqueio em Duas Fases e prevenção de Bloqueio Perpétuo pela pré-ordenação do acesso aos itens de Dados.

No Capítulo V o GBDR é detalhado. São apresentados a sua divisão em módulos e o fluxo de informações entre seus módulos: Gerenciador de Transações, Gerenciador de Acesso e Iniciador/Recuperador. São descritas as ações desempenhadas por cada módulo e os parâmetros fornecidos.

No Capítulo VI, é justificada a seleção realizada em um conjunto de protocolos de Difusão Confiável examinados. A seguir, é apresentado um protocolo para Difusão Confiável que incorpora a Exclusão Mútua para recuperação de falhas como uma etapa de reforma do protocolo. Finalmente, algumas alterações são propostas no protocolo selecionado de modo a resolver alguns problemas apresentados.

No Apêndice, é feito o detalhamento do protocolo de Difusão Confiável utilizando Redes de Petri interpretadas.

Capítulo II

Conceitos Básicos e Alternativas Pesquisadas

1. Problemas Básicos em Bancos de Dados Distribuídos 17

2. Reavaliação das Funções de um Gerenciador de Banco de Dados Genérico segundo as Necessidades do Centro de Controle 18
   - 2.1 Funções 18
   - 2.2 Gerenciamento do Esquema Externo em um Centro de Controle 19
   - 2.3 Recuperação de Inconsistências 20
   - 2.4 Segurança e Integridade 20

3. Concorrência e Recuperação de Falhas 21
   - 3.1 O Conflito entre requisitos de Sistemas Comerciais e Sistemas em Tempo Real 21
   - 3.2 Direcionamento da Pesquisa 22
   - 3.3 Serialização e Controle de Acesso 23
     - 3.3.1 Arquitetura de um Gerenciador de Banco de Dados Distribuídos 23
     - 3.3.2 Classificação dos Algoritmos de Controle de Acesso 25
     - 3.3.3 Serialização 25
     - 3.3.4 Estratégias de Escalonamento 28
     - 3.3.5 Distribuição do Gerenciador de Acesso 32
     - 3.3.6 Replicação de Dados 33
   - 3.4 Protocolos e Algoritmos Distribuídos 35
     - 3.4.1 Algoritmos Distribuídos para Exclusão Mútua 40
     - 3.4.2 Algoritmos para Difusão Confiável 44

1. Problemas Básicos em Bancos de Dados Distribuídos

Existem três grandes dificuldades no projeto de Bancos de Dados Distribuídos, segundo COUCEIRO e BARRENECHA [16]: distribuição Otima dos Repositórios de dados pelas Estações; execução da Linguagem de Consulta e Atualização num ambiente distribuído; e Controle de Acesso Concorrente ao Banco de Dados. Para Centros de Controle baseados em uma arquitetura distribuída homogênea, geograficamente concentrada, a Distribuição dos Repositórios de dados pelas Estações é um problema que determina a própria arquitetura e é, neste caso, resolvido "heuristicamente" na própria concepção do sistema.

Deve-se notar ainda que, no caso do Banco de Dados estritamente Replicados, uma transação dispõe localmente de todos os itens de dados necessários à sua execução. Portanto, não cabe analisar, neste trabalho, a dificuldade de execução de Linguagens de Consulta quando os itens encontram-se dispersos pelas Estações.

Assim, o Controle de Acesso Concorrente é a principal dificuldade a ser equacionada em termos de um Banco de Dados em Tempo Real. A função deste controle é permitir implementação da característica da Atomicidade das atualizações do Banco de Dados.

Paralelamente, os requisitos de Disponibilidade acrescentam o aspecto da Recuperação de Falhas a partir das redundâncias do sistema distribuído.

Atomicidade e Disponibilidade são características interdependentes. Neste Capítulo procura-se relacionar alguns conceitos e, a partir destes, algoritmos que permitam a implementação destas características.

2. Reavaliação das funções de um Gerenciador de Banco de Dados Genérico segundo as Necessidades do Centro de Controle.

2.1 Funções

DATE [18] apresenta as seguintes funções de um Sistema Gerenciador de Banco de Dados:

a - Centralização do Controle de Dados, que resulta nas seguintes vantagens:
  - redução das redundâncias
  - evitar inconsistências
  - dados compartilhados
  - efetivação de padrões
  - restrições de segurança
  - manutenção da integridade
  - balanceamento dos requisitos conflitantes

b - Independência Lógica dos Dados, que resulta nas seguintes vantagens:
  - dados podem ser organizados do modo mais conveniente a cada usuário sem preocupação com os demais usuários.
  - modificações e acréscimos podem ser introduzidos no Banco de Dados sem afetar aplicações já existentes.
  - modificações podem ser introduzidas na forma física de armazenamento sem afetar os usuários do Banco de Dados.

A Independência Lógica é obtida pela introdução de dois níveis de gerenciamento dos dados, além do nível físico: Gerenciamento do Esquema Conceitual e Gerenciamento do Esquema Externo.

O Gerenciamento do Esquema Conceitual isola a forma física de armazenamento (Esquema Físico) da forma lógica segundo a qual o Banco de Dados é globalmente representado (Esquema Conceitual).

O Gerenciamento do Esquema Externo isola o Esquema Conceitual das Vistas definidas pelos usuários. Vista ou Esquema Externo é a forma mais adequada de apresentação, para cada usuário, das informações presentes no Banco de Dados.

Entretanto, um Centro de Controle é, antes de mais nada, um sistema de controle em Tempo Real e como tal deve apresentar um tempo de resposta compatível. Portanto, a introdução de qualquer sobrecarga de processamento não deve violar seus requisitos rígidos de tempo. Algumas das vantagens fornecidas por um SGBD devem ser reanalisadas sob essa ótica.

2.2 Gerenciamento do Esquema Externo em um Centro de Controle

Pode-se idenficar dois modos de operação de um Centro de Controle: "on-line" e "off-line". O primeiro é o modo normal de operação, quando o Centro de Controle está ligado ao Processo Elétrico e executando suas funções de supervisão e controle. O segundo é o modo de reconfiguração do sistema no qual o Centro de Controle é isolado do Processo para sofrer alterações. Esta reconfiguração é uma necessidade periódica em função de modificações que são efetuadas no Processo Elétrico. Alguns tipos de alteração que podem ser feitos no modo "on-line" são denominados Reconfiguração Dinâmica.

Em sistemas em Tempo Real, a estratégia típica de armazenamento, para que as aplicações passem a satisfazer seus requisitos de tempo, tem sido requerer que cada aplicação organize fisicamente os dados de modo a otimizar seu processamento. Isso não deve implicar necessariamente em manter "on-line" um gerenciamento de Esquemas Externos. A obtenção das Vistas necessárias a cada aplicação, a partir de um Banco de Dados sem redundância, é um processamento feito "off-line". Essa estratégia pode ser utilizada para qualquer atributo estático. Isto é, informações associadas às entidades definidas no Banco de Dados que não se modificam no decorrer do processamento "on-line".

Para atributos dinâmicos, o compartilhamento implica em dependência entre os usuários. Essa dependência pode ser diminuída se os requisitos de tempo o permitirem, como é o caso de alguns tipos de dados dinâmicos em Despachos (Centros de Controle Regionais de coordenação da produção de Energia Elétrica) mencionados por MOONEY et al [35]: características elétricas e as conexões entre Equipamentos do Sistema Elétrico. Isto é feito com a introdução de um nível de Gerenciamento de Esquema Externo bastante simplificado.

2.3 Recuperação de Inconsistências

Num sistema tolerante a falhas, as redundâncias de dados devem existir, pois constituem o princípio da recuperação de falhas.

Como as redundâncias são necessárias, as inconsistências podem surgir. É importante que no gerenciador existam mecanismos de detecção das inconsistências e recuperação.

2.4 Segurança e Integridade

Restrições de segurança de acesso e a verificação da integridade semântica do Banco de Dados são atribuições de um gerenciador que devem ser minimizadas num Sistema em Tempo Real, devido à sobrecarga de processamento que exigem.

Aqui é preciso recordar uma importante diferença, quanto a expectativas de erro por parte de tarefas usuárias, entre o Banco de Dados de um sistema de aplicação comercial e o Banco de Dados de um sistema de supervisão e controle.

No primeiro caso, trata-se de um sistema aberto, sujeito a todo tipo de intrusões e mau uso do Banco de Dados, em função da criatividade do usuário. Neste caso, são indispensáveis a segurança de acesso e a constante verificação da sua integridade semântica.

No segundo caso, a simplificação de tais mecanismos se justifica quando se considera que um sistema, quando é entregue a operação, é o resultado de uma rigorosa metodologia de desenvolvimento e teste. Isto é, as ferramentas de análise de integridade devem existir "off-line", embutidas na metodologia de desenvolvimento. "On-line", as tarefas usuárias não violam a integridade dos Dados, a não ser por falha ("hardware"). E, neste caso, um mecanismo de detecção e recuperação impede que a falha cause danos permanentes aos Dados. Além disso, as tarefas usuárias são pré-definidas e repetitivas, dispensando mecanismos de segurança de acesso.

3. Concorrência e Recuperação de Falhas

Neste item são apresentados alguns conceitos e algoritmos relacionados ao Controle de Acesso Concorrente e a Recuperação de Falhas em um Banco de Dados Replicados.

3.1 O Conflito entre requisitos de Sistemas Comerciais e Sistemas em Tempo Real

Considerando a totalidade dos sistemas computacionais existentes, a classe denominada comercial tem sido a principal estimuladora do surgimento de novas conceituações e ferramentas de "software" no sentido de otimizar seu desenvolvimento. Os sistemas de Banco de Dados são apenas mais um caso.

Freqüentemente, a transposição, para Sistemas em Tempo Real, de conceitos desenvolvidos para sistemas comerciais, leva algum tempo até que avanços tecnológicos permitam sua concretização. Um exemplo dessa transposição é a utilização de um Banco de Dados Relacional em um Centro de Despacho de Energia Elétrica apresentado por MOONEY et al [35]. Neste trabalho são apresentadas três características básicas que viabilizam sua realização em termos de desempenho:

a - Múltiplos Métodos de Acesso - cada usuário escolhe o método de acesso que forneça o nível de desempenho necessário à sua aplicação. Quanto mais eficiente o acesso, menor a independência lógica obtida.

b - Dados Residentes em Memória - qualquer estrutura de dados pode ser declarada residente em memória. Estas estruturas são carregadas para memória na iniciação do sistema. Esta característica fica transparente ao usuário em termos de método de acesso.

c - Armazenamento Puro - as estruturas de dados são armazenadas em memória, contiguamente e separadas das estruturas de controle impostas pelo Banco de Dados Relacional. Isto significa que os dados podem ser transferidos diretamente para a memória do usuário sem nenhuma transformação intermediária.

Estas características retratam bem a "desestruturação" que é necessário introduzir num sistema para que sua utilização em Tempo Real apresente um desempenho compatível.

3.2 Direcionamento da Pesquisa

Os princípios básicos que nortearam a pesquisa das técnicas para a solução dos problemas apresentados neste trabalho foram:

a - Selecionar soluções e algoritmos auto-contidos, isto é, que prescindissem de outros sistemas auxiliares mais completos do que os descritos no item sobre a arquitetura do sistema no Capítulo I. Isto porque não estariam disponíveis em um Centro de Controle baseado em microcomputadores de baixo custo. Normalmente, os algoritmos para Controle de Acesso Concorrente em sistemas de grande porte e Redes Abertas presumem uma série de facilidades como função de gerência da rede que informa, por exemplo, a configuração atual das Estações ativas.

b - Reexaminar vantagens apresentadas por algumas soluções, segundo a ótica de um Sistema em Tempo Real. Isto é, o processamento é composto por um conjunto de tarefas repetitivas pré-definidas com rígidos requisitos de tempo de execução.

3.3 Serialização e Controle de Acesso

Muitos artigos, apresentando algoritmos distribuídos para Controle de Acesso Concorrente, têm sido publicados ao longo dos anos. Em conseqüência, extensas relações foram publicadas por KOHLER [27] e por SON [44]. Neste item procura-se decompor o problema do Controle de Acesso Concorrente em alguns subproblemas e conceituá-los a partir de alguns resultados básicos da Teoria da Serialização. A base para este item é o trabalho de BERNSTEIN e GOODMAN [7].

3.3.1 Arquitetura de um Gerenciador de Banco de Dados Distribuídos

Para efeito de análise do problema do Controle de Acesso Concorrente, pode-se considerar que um Banco de Dados consiste de um conjunto de elementos de memória denominados Itens de Dados denotado por: {...,Im,...}, m= 1, 2,... . O usuário do Banco de Dados atua sobre um Item de Dado realizando operações de Leitura e Escrita sobre esse Item. Ler[Im] retorna o valor armazenado em Im e Escrever[Im] atualiza o valor armazenado em Im a partir de novo valor. Inicialmente, será considerado que cada Item de Dado está armazenado em uma única Estação. A replicação destes itens será discutida a seguir.

O usuário interage com o Banco de Dados executando programas denominados Transações e identificadas por Tn, n= 1, 2 ... . Uma Transação é caracterizada por produzir alterações que mantêm a consistência do Banco de Dados, desde que considerada sua ação isolada sobre o Banco de Dados. Um Banco de Dados está consistente, em determinado instante, se as informações que

armazena não violam as regras impostas pela realidade que ele representa.

O programa que controla o Banco de Dados Distribuídos está repartido por todas as Estações e é denominado Gerenciador de Banco de Dados Distribuídos (GBDD). Do ponto de vista do Controle de Acesso Concorrente, os GBDDs são constituídos por uma combinação dos seguintes blocos básicos: Gerenciador de Transações, Gerenciador de Acesso Concorrente e Gerenciador dos Dados.

Cada Transação envia suas solicitações para Iniciar, Ler, Escrever e Finalizar a um único Gerenciador de Transações. As operações de Iniciar e Finalizar delimitam uma Transação no Gerenciador de Transações. O Gerenciador de Transações repassa as solicitações ao(s) Gerenciador(es) de Acesso. O algoritmo de Controle de Acesso Concorrente utilizado determina a ordem segundo a qual o(s) Gerenciador(es) de Dados deverá(ão) receber as solicitações de Escrita e Leitura. De um modo geral, o Gerenciador de Dados está na mesma Estação onde fica o Item de Dado que se pretende acessar.

O Gerenciador de Acesso controla e estabelece a ordem efetiva das solicitações repassadas por ele aos Gerenciadores de Dados. Por razões que serão examinadas,um Gerenciador de Acesso, ao receber uma solicitação de acesso a um Item de Dado, pode repassá-la ao Gerenciador de Dados, pode atrasá-la de modo a alterar a ordem efetiva das solicitações, ou pode rejeitá-la como sendo uma solicitação imprópria. Neste último caso, a Transação que a emitiu deve ser abortada.

Abortar uma Transação significa que todas as Escritas por ela solicitadas até então são tornadas sem efeito e os valores anteriores dos Itens de Dados alterados são restaurados. Além disso, se, devido à concorrência entre Transações, alguma outra fez a Leitura de um valor tornado sem efeito, tal Transação deve também ser abortada. A fim de evitar o Aborto em Cascata decorrente dessa regra, os GBDDs, de um modo geral, não permitem que uma atualização produzida por uma Transação seja

consultada por outra até que a primeira Transação tenha sido concluída com sucesso. É uma solução que implica na redução do paralelismo.

O Gerenciador de Dados recebe as solicitações de Leitura e Escrita e atua efetivamente sobre os Itens de Dados sob seu controle.

3.3.2 Classificação dos Algoritmos de Controle de Acesso

O algoritmo de Controle de Acesso Concorrente num GBDD consiste de um conjunto de Gerenciadores de Acesso cooperando na execução de uma estratégia global de escalonamento das solicitações concorrentes. A classificação de tal algoritmo pode ser feita quanto aos seguintes aspectos:

a - tipo de estratégia de escalonamento;
b - localização do(s) Gerenciador(es) de Acesso, isto é, centralizada ou distribuída;
c - tratamento dos Itens Replicados, que são Itens com cópias em mais de uma Estação.

3.3.3 Serialização

A Teoria da Serialização é uma coleção de regras matemáticas que permitem estabelecer a correção de um algoritmo de Controle de Acesso Concorrente. Isto é feito observando-se as seqüências de solicitações, denominadas Logs, permitidas pelo algoritmo.Um algoritmo é correto se todos os Logs por ele permitidos são corretos. A correção de um Log é definida mais adiante.

O Log de uma Transação representa a ordem parcial entre as suas operações. Esta ordem é estabelecida pela Transação e passada ao Gerenciador de Acesso. Formalmente, o Log de uma Transação Tn é um conjunto parcialmente ordenado $Log(Tn)=(S_n,O_n)$, onde $S_n=\{Ler_n[Im],Escrever_n[Iy],Ler_n[Iz],...\}$ é o conjunto de Leituras e Escritas emitidas pela Transação Tn. Para se evitar ambigüidades, assume-se, neste modelo, que cada Transação, Tn,

realiza no máximo uma Escrita, $Escrever_n[Im]$, e/ou uma Leitura, $Ler_n[Im]$, de um determinado item Im. Além disso, O_n, que está contido em S_n X S_n, estabelece a ordem parcial segundo a qual as operações precisam ser executadas. Diz-se que $Ler_n[Im] < Escrever_n[Im]$, se e somente se $(Ler_n[Im], Escrever_n[Im])$ pertence a O_n.

O Log do GBDD representa a ordem parcial entre todas as solicitações feitas pelas Transações concorrentes. Esta é estabelecida pelos Gerenciadores de Acesso, a partir das ordens parciais especificadas por cada Transação. As solicitações aos Gerenciadores de Dados são emitidas nesta ordem.

Formalmente, o Log do GBDD é o conjunto parcialmente ordenado Log=(S,O), onde:

a - $Log(Tn) \square (S_n, O_n)$ são os Logs das transações usuárias do GBDD para n=1,...,z;

b - tem-se $S = U_{(n=1,...,z)} S_n$; e

c - tem-se que O contém ou é igual a $U_{(n=1...,z)} O_n$.

A segunda condição estabelece que o GBDD repasse aos Gerenciadores de Dados apenas as solicitações provenientes das Transações $T_1, T_2, ...T_z$.

A terceira condição garante que o GBDD não viola nenhuma das restrições de ordem impostas pelas Transações.

3.3.3.1 Equivalência de Logs

Intuitivamente, dois Logs são equivalentes se eles produzem o mesmo resultado final no Banco de Dados e as mesmas Leituras. Seja Log = (S,O) o Log de um GBDD obtido de um conjunto T de Transações e suponha que $Escrever_n[Im]$ e $Ler_j[Im]$ são solicitações em Log.

Diz-se que $Ler_j[Im]$ lê-o-valor-atualizado por $Escrever_n[Im]$, se, $Escrever_n[Im] < Ler_j[Im]$ em O.

Diz-se que $Escrever_n[Im]$ é escrita-final, se $Escrever_n[Im]$ é a maior operação segundo O no subconjunto de S composto por Escritas em Im.

Dois Logs são equivalentes se:

a - Qualquer $Ler_j[Im]$ le-o-valor-atualizado pelo mesmo $Escrever_n[Im]$ nos dois Logs, e,

b - O conjunto de escritas-finais é o mesmo nos dois Logs.

A primeira condição assegura que cada Transação lê os mesmos valores do Banco de Dados nos dois Logs.

A segunda condição assegura que os dois Logs produzem o mesmo estado final no Banco de Dados.

3.3.3.2 Logs Serializáveis e Logs Corretos

Um Log serial do GBDD é uma ordem total em S tal que para cada par de transações Tn e Tj, ou todas as operações de Tn precedem Tj ou vice-versa. Tal Log é correto, pois se cada Transação é executada por vez e se cada Transação deixa o Banco de Dados num estado consistente, o estado final do Banco de Dados é igualmente consistente.

De modo a reduzir o tempo total de execução do Log do GBDD, outros Logs apresentando Transações concorrentes (executadas paralelamente por vários Gerenciadores de Dados) podem ser considerados igualmente corretos. Todos os Logs, que forem equivalentes a um Log serial, serão por consequência considerados corretos. Neste caso, tais Logs são ditos serializáveis (SR).

3.3.3.3 Teorema da Serializabilidade

Seja um Log sobre $T=\{T_1,T_2,\ldots,T_n,\ldots\}$. Pode-se definir para Log um Grafo de Serialização GS(Log) = (T,A), direcionado, onde

T é o conjunto de nós constituído pelas Transações e A é o conjunto de arestas definido da seguinte forma:

$A=\{(Tn,Tj) \in TxT \mid$ (existe Im) ($Ler_n[Im] < Escrever_j[Im]$, ou,
$Escrever_n[Im] < Ler_j[Im]$, ou,
$Escrever_n[Im] < Escrever_j[Im]$)}.

As três condições da definição caracterizam conflito entre as transações Tn e Tj. Isto é, ambas acessam o mesmo Item Im e uma delas solicita uma Escrita. As duas primeiras condições constituem conflitos Escrita-Leitura (EL) e a terceira, Escrita-Escrita (EE).

Teorema da Serializabilidade:
Se GS(Log) é acíclico então Log é SR (serializável).

Para determinar que um algoritmo de Controle de Acesso Concorrente é correto, primeiramente se caracteriza o Log produzido pelos Gerenciadores de Acesso e então se demonstra que tal Log possui um GS acíclico.

Existe uma outra forma deste Teorema no caso de se tratar separadamente conflitos EL e EE. Neste caso dois tipos de GS são definidos. Essa alternativa pode ser encontrada no trabalho de BERNSTEIN e GOODMAN [7].

3.3.4 Estratégias de Escalonamento

Existem duas principais estratégias de escalonamento adotadas pelos Gerenciadores de Acesso para a produção de Logs SR: Bloqueio em Duas Fases e Carimbo de Tempo. Cada tipo de estratégia pode ser usado para escalonar conflitos EL, EE ou ambos. O escalonamento dos conflitos em separado permite algumas otimizações, como a "Thomas´ Write Rule" apresentada por THOMAS [47]. Neste trabalho apresentaremos apenas a versão voltada para o escalonamento de ambos os conflitos.

3.3.4.1 Bloqueio em Duas Fases

O Bloqueio em Duas Fases é definido por três regras:

a - Cada Transação, antes de solicitar qualquer $Ler_n[Im]$ ($Escrever_n[Im]$), deve obter um Bloqueio para novas Escritas, BE, (Bloqueio para novas Escritas ou Leituras, BEL) de Im. O bloqueio deve ser mantido pelo menos até o fim da operação.

b - Transações distintas não podem deter simultaneamente bloqueios conflitantes. Dois bloqueios são conflitantes se ambos referem-se ao mesmo Item e se pelo menos um deles é o BEL.

c - Após a liberação de um bloqueio, a transação não pode obter nenhum novo bloqueio.

Esta última regra caracteriza duas fases distintas que constituem as Transações: fase de alocação, na qual a Transação obtém mais e mais bloqueios sem realizar qualquer desbloqueio, e fase de liberação, iniciada com o primeiro desbloqueio.

Pode-se demonstrar que o Bloqueio em Duas Fases só produz Logs SR.

Devido à segunda regra acima, uma operação solicitada ao Gerenciador de Acesso pode ser atrasada porque outra detém o bloqueio do Item solicitado. Neste caso, a Transação não poderá evoluir até que ocorra o desbloqueio. Tais situações de paralisação podem levar ao Bloqueio Perpétuo. Por exemplo, uma determinada Transação está paralisada devido a um bloqueio obtido por uma segunda Transação sobre um item Im. Esta, por sua vez, está paralisada devido a um bloqueio obtido pela primeira sobre um outro item Iy.

O Bloqueio Perpétuo é caracterizado por determinar um ciclo no Grafo de Alocação de Recursos. Tal Grafo, direcionado, bipartite, é construído da seguinte forma: os nós são de duas categorias , a primeira correspondendo às Transações e a segunda aos Itens do Banco de Dados; as arestas são direcionadas de um Item para uma Transação se a Transação detém o bloqueio deste Item, e, direcionadas da Transação para o Item

se aquela está aguardando a obtenção do bloqueio do Item. Um modo clássico de conviver com a possibilidade de Bloqueios Perpétuos é manter o Grafo de Alocação de Recursos e periodicamente pesquisar a existência de ciclos. Ao ser detectado algum ciclo, uma das Transações (nó do grafo), participante do ciclo, é forçada a abortar de modo a eliminar o Bloqueio Perpétuo.

3.3.4.2 Ordenação por Carimbo de Tempo

Neste caso, o Gerenciador de Transações associa a cada Transação um Carimbo de Tempo (CT) usado para identificar univocamente a Transação. A obtenção de um CT global num sistema distribuído é descrita por LAMPORT [29].

A cada solicitação de Leitura ou Escrita contida em uma Transação é também associado o mesmo CT da Transação. A estratégia de ordenação por CT consiste de uma única regra: as solicitações conflitantes são ordenadas segundo seus CTs. Pode-se mostrar que esta estratégia só produz Logs SR.

Na Ordenação por Carimbo de Tempo o Gerenciador de Acesso repassa as solicitações ao Gerenciador de Dados na ordem em que são recebidas das Transações e rejeita as solicitações que, se passadas adiante, violariam a regra estabelecida. Isto resulta

no seguinte procedimento ao receber uma operação $Ler_n[Im]$:

Se CT de Tn < maior dos CTs de qualquer das Escritas efetuadas em Im

Então rejeita $Ler_n[Im]$ e aborta Tn

Senão aceita $Ler_n[Im]$ e repassa a solicitação ao Gerenciador de Dados;

Ao receber uma operação $Escrever_n[Im]$:

Se CT de Tn < maior dos CTs de qualquer Escrita ou Leitura efetuada em Im

Então rejeita $Escrever_n[Im]$ e aborta Tn

Senão aceita $Escrever_n[Im]$ e repassa a solicitação ao Gerenciador de Dados;

A Transação abortada deverá então ser reiniciada. Isto pode levar esta Transação a uma Condição de Esfomeamento ("starvation") quando o nível de concorrência é muito alto. Esta condição se caracteriza por determinadas Transações nunca concluírem e serem repetidamente abortadas devido a coincidências de conflitos com as demais Transações. Um modo de solucionar este problema é alterar a estratégia para o que é denomidado de modo Conservativo.

No modo Conservativo, o Gerenciador de Acesso, ao receber um início de Transação, emitido por determinado Gerenciador de Transações, não repassa qualquer das suas operações até que todos os demais Gerenciadores de Transações tenham igualmente iniciado Transações. Quando essa situação é obtida, o Gerenciador de Acesso seleciona a Transação com o menor CT e começa a repassar ao Gerenciador de Dados as suas operações. Isso garante que uma Transação jamais seja abortada. Entretanto, todas as operações sofrem um atraso que depende do volume de Transações no sistema.

Outras variações sobre a Ordenação por Carimbo de Tempo são apresentadas por KOHLER [27] e por THOMAS [47].

3.3.5 Distribuição do Gerenciador de Acesso

Uma alternativa a um Gerenciador de Acesso central é repartí-lo entre as Estações, de modo que, próximo (na mesma Estação) a cada Gerenciador de Dados, exista um Gerenciador de Acesso local responsável por escalonar o acesso aos dados da Estação. Tal organização impõe a necessidade da cooperação entre os vários Gerenciadores de Acesso de modo que as regras de escalonamento sejam globalmente cumpridas .

A distribuição da estratégia de Bloqueio em Duas Fases resulta no fato de as regras a (solicitação de bloqueio sobre o item) e b (rejeição de bloqueios conflitantes) poderem ser cumpridas por cada Gerenciador de Acesso localmente, isto é, independentemente dos demais. A regra c (delimitação das fases), entretanto, requer uma coordenação entre os Gerenciadores de Acesso: nenhum Gerenciador pode atender qualquer solicitação de Bloqueio de uma Transação depois que algum Gerenciador realizou a liberação de um Bloqueio da mesma Transação. Um modo de resolver este problema é permitir que o Gerenciador de Transação controle a delimitação das duas fases de uma Transação. Isto é, as solicitações de bloqueio só são emitidas depois que o Gerenciador de Transações obteve a confirmação de todos os bloqueios.

O principal problema com a utilização de Bloqueio em Duas Fases na sua forma distribuída é a possibilidade da ocorrência de Bloqueios Perpétuos Distribuídos. Isto é, Bloqueios, para a detecção dos quais, é necessário um Grafo Global de Alocação de Recursos. Como as informações para este grafo estão repartidas entre os vários Gerenciadores de Acesso, a sua manutenção envolveria uma intensa comunicação entre os Gerenciadores.

A distribuição do Gerenciador de Acesso baseada em Ordenação por Carimbo de Tempo é trivial. A única regra necessária a esta

estratégia (ordenação segundo os Carimbos de Tempo das operações conflitantes) pode ser preservada independentemente por cada Gerenciador de Acesso.

3.3.6 Replicação de Dados

Em Bancos de Dados Replicados cada Item de Dados lógico pode ter muitas réplicas físicas armazenadas nas diversas Estações. A réplica do m-ésimo Item de Dados, Im, armazenado na i-ésima Estacão, Ei, é denotado por Im_{Ei}. As Transações realizam Leituras e Escritas de Itens lógicos. Os Gerenciadores de Transações repassam as Leituras e Escritas para as réplicas físicas através dos respectivos Gerenciadores de Acesso e de Dados. Do ponto de vista de uma Transação, tudo se passa como se houvesse uma única cópia.

Há três modos de se tratar a Replicação de Itens de Dados, quanto à forma de repassar as Leituras e Escritas às réplicas físicas dos Itens de Dados: forma trivial, Cópia Principal e Votação.

3.3.6.1 Replicação Trivial: Lê Alguma e Atualiza Todas

Nessa forma, o Gerenciador de Transações transforma cada Leitura, $Ler_n[Im]$, em uma única solicitação ao Gerenciador de Acesso de alguma das Estações, Ei, que possua uma réplica de Im. Esta solicitação é denotada por $Ler_n[Im_{Ei}]$. Além disso, cada solicitação de Escrita, $Escrever_n[Im]$, é repetida para os Gerenciadores de Acesso: $Escrever_n[Im_{Ei}]$, onde Ei são Estações que possuam uma réplica de Im. Esta forma pode ser denominada Lê Alguma e Atualiza Todas.

Desta forma, desde que cada Gerenciador de Acesso produza Logs SR, tudo se passa como se existisse uma única réplica do Banco de Dados. Isto baseia-se no fato de qualquer cópia apresentar o valor atualizado, pois as Escritas são feitas em todas as réplicas.

3.3.6.2 Cópia Principal

Neste caso, para cada Item, Im, uma das Estações, Ep, detentora de uma réplica Im_{Ep}, é escolhida Cópia Principal de Im.

O Gerenciador de Transações, assim como no item anterior, transforma as solitações de Leitura de Im em uma solicitação para algum dos Gerenciadores de Acesso que possua uma réplica de Im.

No caso de uma solicitação de Escrita em Im, esta é repassada ao Gerenciador de Acesso da Cópia Principal: $Escrever_n[Im_{Ep}]$. O Gerenciador de Acesso então, no momento de repassar a solicitação ao seu Gerenciador de Dados, também repassa a solicitação de Escrita a todos os demais Gerenciadores de Acesso detentores de uma réplica de Im.

A principal característica desta alternativa é que a replicação fica transparente ao Gerenciador de Transações.

Essa alternativa é bastante vantajosa, em relação à alternativa trivial descrita anteriormente, no caso do escalonamento ser feito por Bloqueio em Duas Fases, pois evita a ocorrência de Bloqueios Perpétuos relacionados ao bloqueio das múltiplas réplicas. Isso é conseqüência de os Bloqueios emitidos pelo Gerenciador de Transações serem primeiramente submetidos ao Gerenciador de Acesso da Cópia Principal, garantindo um escalonamento inicial centralizado.

Deve-se notar que, no caso de uma Leitura, o BE emitido pelo Gerenciador de Transações deve ser enviado ao Gerenciador de Acesso da Cópia Principal, mesmo que a solicitação de Leitura propriamente seja repassada a um Gerenciador de Acesso distinto. Por exemplo, ao gerenciador da própria Estação onde se desenrola a Transação.

3.3.6.3 Votação

Este caso é idêntico ao caso trivial quanto ao repasse de solicitações de Leitura a um dos Gerenciadores de Acesso de uma das réplicas e quanto ao repasse de solicitações de Escrita a todos os Gerenciadores de Acesso detentores de alguma réplica.

Quando um Gerenciador de Acesso estiver pronto para repassar a solicitação de Escrita ao seu Gerenciador de Dados, êle devolverá ao Gerenciador de Transação original um voto positivo. Quando este receber o voto positivo da maioria dos Gerenciadores de Acesso, êle divulgará uma permissão a todos os Gerenciadores de Acesso para prosseguirem o repasse da solicitação de Escrita.

A correção deste processo deve-se ao seguinte fato: como os Gerenciadores de Acesso, que aprovam uma determinada solicitação, constituem a maioria, sempre haverá uma interseção com os Gerenciadores que aprovarão a próxima solicitação. Duas a duas, as solicitações sempre terão sido aprovadas por um Gerenciador comum. Se cada Gerenciador de Acesso estiver correto individualmente, o escalonamento global também estará. Maiores detalhes desse processo ("Majority Consensus") são descritos por THOMAS [47].

A principal vantagem da Votação é a tolerância a falhas. Este processo funciona corretamente enquanto a maioria das Estações estiver ativa. A forma trivial e a Cópia Principal apresentam restrições quando se considera a desativação e reativação de Estações.

3.4 Protocolos e Algoritmos Distribuídos.

Algoritmos, num contexto de programação, qualificados como paralelos, caracterizam-se pela coexistência de múltiplos fluxos de controle simultâneos. Inicialmente, paralelismo foi objeto de estudo exclusivo da área de Sistemas Operacionais: decomposição modular em atividades elementares de

processamento seqüencial, isto é, em Tarefas, e gerenciamento das interações. O surgimento das Redes de Computadores introduziu o conceito da comunicação, exclusivamente por troca de mensagens, como um fator básico na concepção de algoritmos paralelos distribuídos.

Como é caracterizado por RAYNAL [38], os algoritmos paralelos são denominados Protocolos na área de Redes. Eles realizam serviços de transferência de informações ou de controle de um conjunto de Tarefas localizadas em Estações distintas interligadas por vias de comunicação. Alguns desses protocolos foram ou estão sendo padronizados: X.25; MAP e TOP. Na área de Sistemas Distribuídos, os algoritmos paralelos são denominados Algoritmos Distribuídos. Eles realizam geralmente funções básicas, como Exclusão Mútua, por exemplo, e funções de controle inerentes à distribuição. A equação seguinte caracteriza tal classe de algoritmos:

Algoritmo Distribuído = Tarefas + Mensagens.

Deve-se notar aqui que as Estações hospedeiras das Tarefas são fracamente acopladas, isto é, não possuem memória em comum (memória compartilhada).

Há um elenco de características próprias aos Algoritmos Distribuídos que permitem avaliá-los e comparar os que desempenham as mesmas funções:

a - Independência em Relação às Tarefas

Um primeiro item de avaliação de um algoritmo é medir o quanto ele é independente. Isto é, o quanto o Algoritmo como um todo prescinde de alguma Tarefa em especial. Esta noção está ligada à simetria das atividades desempenhadas pelas várias Tarefas que constituem o algoritmo. RICART et al [39] apresenta o conceito de simetria. Pode-se ter diversos níveis de simetria:

- Assimetria : quando cada Tarefa executa atividades diferentes, isto é, algoritmos com textos diferentes; não é possível, neste caso, intercambiar duas Tarefas.

- Simetria de Texto : neste caso, o texto do algoritmo é o mesmo para todas as Tarefas; entretanto, o algoritmo faz referência a um identificador da Tarefa, que é único para cada Tarefa.
- Simetria Forte: as Tarefas executam todas o mesmo texto. Neste caso, apenas a chegada de mensagens diversas determina comportamentos distintos entre as Tarefas.
- Simetria Total: as Tarefas executam algoritmos idênticos, consomem as mesmas mensagens e comportam-se identicamente.

A Assimetria corresponde à mais fraca independência. A Simetria Total corresponde à mais forte.

b - Tolerância a Faltas

Até aqui, neste trabalho, o termo Falha têm sido empregado de forma intuitiva. A partir daqui será adotada a nomenclatura proposta por FRAGA et al [21] :

Falta ("Fault")- causa de um Erro existente no sistema. Por exemplo, uma variável não iniciada corretamente pelo programador (Falta "soft") ou a queima de um circuito eletrônico (Falta "hard");

Erro - um elemento do sistema potencialmente responsável por levá-lo a Falha. Isto é, um problema latente que pode ou não manifestar-se;

Falha ("Failure") - serviço oferecido por um sistema que contradiz sua especificação.

Deste modo, é possível falar-se em Tolerância a Faltas e Recuperação de Falhas.

É desejável que um algoritmo suporte faltas nas Estações e a desativação de todas as Tarefas aí localizadas. Este item está ligado ao anterior, pois quanto mais independente for o algoritmo, mais chance ele terá de prosseguir operando corretamente na presença de falha de alguma de suas partes. As

faltas, para análise de algoritmos distribuídos, podem ser classificadas da seguinte maneira:

Faltas por Omissão - faltas que levam um componente a nunca responder a determinado estímulo, contradizendo sua especificação. Aqui, por exemplo, estão incluídas as perdas de mensagens numa Rede de comunicações, seccionamento de uma Rede, o modelo biestável Estação-em-Falha/Estação-Ativa e um relógio que simplesmente pára;

Falta por Assincronismo - faltas que levam um componente a responder, ou adiantado, ou atrasado, a determinado estímulo. Aqui, por exemplo, estão incluídos o desvio inesperado entre relógios de um sistema que, devido à sobrecarga de processamento, atrase sua resposta além da especificação;

Falta Bizantina - falta que leva um componente a responder de um modo não especificado a um determinado estímulo. A corrupção de mensagens devido à interferência eletromagnética ou à sabotagem são exemplos de faltas bizantinas.

Deve-se notar que não é possível classificar uma Falta dissociada de um componente. Em particular, se um componente é formado por outros, uma Falta de um tipo em determinado componente pode levar a uma falta de outro tipo no primeiro componente. Por exemplo, um "chip" de memória que não responde a uma solicitação de escrita é causado por uma Falta por Omissão. Se, entretanto, essa escrita corrompe a confecção de uma mensagem que será transmitida, fica caracterizada uma Falta Bizantina da Estação como um todo.

Deve-se notar também que as Faltas Bizantinas incluem as Faltas por Assincronismo que incluem as Faltas por Omissão.

c - Independência do Ambiente

A quantidade e severidade das hipóteses formuladas em relação à Rede de Comunicação e os Serviços externos imprescindíveis ao funcionamento do algoritmo estabelecem o seu grau de independência em relação ao Ambiente.

d - Tráfego Gerado

O desempenho de algoritmos distribuídos pode ser comparado em função do número de mensagens trocadas entre Tarefas, a carga de transmissão induzida na Rede de Comunicação e o tempo de espera imposto às Tarefas.

e - Utilização do Estado Global ou Local

Um modo trivial de se obter um algoritmo distribuído é através da replicação de um algoritmo centralizado. Ou seja, através da replicação do contexto do algoritmo em várias Estações e do estabelecimento de regras de atualização das variáveis (deve-se notar aqui que permitir a utilização dessa estratégia, de forma normalizada, e localizada no "software" aplicativo, é exatamente a proposta deste trabalho). Esta estratégia, entretanto, aplicada sem uma avaliação precisa, pode levar a algoritmos nos quais as Tarefas disponham de um Estado Global de manutenção custosa e do qual só utilizem uma pequena parcela. Portanto, um outro item de avaliação de algoritmos distribuídos é se eles se baseiam em Estados Locais (mais otimizados) ou Globais (solução força bruta).

Neste trabalho, exemplares das sequintes classes de Algoritmos Distribuídos são analisados:

- Exclusão Mútua
- Difusão Confiável

Nos itens que se seguem, alguns algoritmos dessas classes serão comentados e seu impacto neste trabalho será evidenciado. Daqui por diante, para evitar a ambigüidade na classificação das Faltas, assume-se que existem apenas dois componentes independentes nos sistemas distribuídos: Estações e canais de comunicação. As Faltas em subcomponentes destes não serão consideradas. Além disso, serão examinados algoritmos que suportam apenas Faltas por Omissão. Assim, as Estações assumem dois estados: ativas, quando estão operando normalmente, e em falha, quando simplesmente emudecem. As demais classes de faltas devem ser tratadas por algoritmos bem mais complexos, providos de redundância n-modular e Consenso Bizantino. Como

referência, tais algoritmos são relacionados por GARCIA-MOLINA et al [23] e por STRONG e DOLEV [45].

3.4.1 Algoritmos Distribuídos para Exclusão Mútua

Um problema, freqüentemente encontrado num conjunto de Tarefas que se comunicam e cooperam para a realização de um objetivo comum, é a atribuição de um privilégio, em determinado instante, a apenas uma das Tarefas participantes. Isto pode ser obtido através de um algoritmo de Exclusão Mútua de modo que o privilégio seja igualmente acessível por qualquer das Tarefas participantes. A obtenção da Exclusão Mútua significa a detenção do privilégio de realizar um processamento denominado Região Crítica.

No caso deste trabalho, o interesse por essa classe de algoritmos provém das seguintes considerações:
a - O GBDR é um algoritmo distribuído que deve suportar falhas e reativações esporádicas de Estações;

b - A concepção do GBDR pode ser simplificada se for possível estabelecer seu funcionamento em duas fases. A fase Normal na qual o conjunto de Estações está bem definido e o GBDR baseia-se no conhecimento desse conjunto e num Carimbo de Tempo para executar suas funções de sincronização entre Transações (Cf. item 3.3.6 sobre Replicação, no qual se pressupõe o conhecimento das Estações que possuem réplicas de um Item de Dados). E a fase de Reforma durante a qual procura-se chegar a um consenso, após a Falha ou reativação de Estações, sobre o conjunto de Estações ativas e o valor atual do Carimbo de Tempo;

c - Verifica-se que o importante para restabelecer a ordem da fase normal não é exatamente obter o conjunto real de Estações ativas, o que cairia num problema de conhecimento em sistemas distribuídos. Basta que todas as Estações concordem sobre um determinado conjunto;

d - A melhor maneira de se obter tal conjunto é atribuir a determinada Estação o privilégio de obtê-lo. O valor atual do Carimbo de Tempo será o maior dos Carimbos apresentados pelas Estações ativas. Daí a importância de um algoritmo de exclusão mútua que apresente Tolerância a Faltas e Independência do Ambiente em alto grau. Sobre um algoritmo com tais características podem ser construídas estruturas mais sofisticadas. Da mesma forma que, no âmbito dos Sistemas Operacionais Centralizados, a partir de primitivas básicas de Semáforos, é possível construir outros sofisticados mecanismos de sincronização entre Tarefas, tais como Caixas Postais e Monitores, descritos por TANENBAUM [46].

3.4.1.1 Trabalhos Publicados

O algoritmo de LAMPORT [29] utiliza 3(n-1) mensagens para obtenção da Exclusão Mútua em um conjunto de n Tarefas (ou Estações, no caso de uma Tarefa por Estação); (n-1) mensagens para difundir a solicitação de entrada na Região Crítica; (n-1) mensagens para responder à solicitação e (n-1) para liberação da Região Crítica. O algoritmo de RICART e AGRAWALA [39] reduziu para 2(n-1) o número de mensagens utilizadas, eliminando as mensagens de liberação da Região Crítica. A liberação da Região Crítica é feita, implicitamente, pela Tarefa que a detinha, pela resposta positiva a uma solicitação de Região Crítica. Esta otimização conservou a Simetria Forte do primeiro algoritmo e foi apresentada juntamente com as providências para o funcionamento na presença de Falhas. Este algoritmo será detalhado no próximo item.

Em seguida, as tentativas de melhorar o Tráfego Gerado do algoritmo levaram CARVALHO e ROUCAIROL [11] e novamente RICART e AGRAWALA, na mesma referência, a reduzirem para um número entre 0 e (n-1) as mensagens necessárias à obtenção da Região Crítica. A principal modificação em relação ao segundo algoritmo foi a seguinte: uma Tarefa, uma vez que obtenha permisão para entrar na Região Crítica, pode entrar e sair outras vezes, sem consultar as demais Tarefas, até que esse

direito seja cedido a outra tarefa. Entretanto, essa última forma do algoritmo já não é fortemente simétrica e a perda de mensagens e de Estações pode levar a situações de Bloqueio Perpétuo. Essa análise é feita por RAYNAL [38].

Um outro tipo de algoritmo de Exclusão Mútua, baseado em Votação e grupos de Tarefas "eleitoras", dois a dois não exclusivos, foi apresentado por MAEKAWA [32]. Tal algoritmo obtém a Exclusão Mútua com $3(n)^{1/2}$ mensagens. Entretanto, da forma como foi proposto, não apresenta qualquer simetria na presença de falhas. A idéia da Votação apresentada por THOMAS [47] é modificada aqui para exigir a aprovação, não mais da maioria, mas de um subconjunto das Tarefas ("Corum Consensus"). Daí a redução no número de mensagens. A correção é baseada no fato de que esses subconjuntos são intencionalmente escolhidos de modo a apresentarem, dois a dois, interseção não vazia. Esse algoritmo corresponde também a uma variação da técnica proposta por GIFFORD [24] e denominada Votação Ponderada.

Todos esses algoritmos, satisfeitas suas hipóteses sobre o ambiente, asseguram a propriedade de Exclusão Mútua em um tempo finito a qualquer Tarefa que a solicite.

3.4.1.2 O Algoritmo de RICART e AGRAWALA - 84

Este algoritmo provê Exclusão Mútua entre Estações que se comunicam exclusivamente via mensagens ponto-a-ponto. Presume-se um meio de comunicação livre de erros de transmissão e de perdas de mensagens (todas as mensagens são confirmadas). Admite-se eventuais reordenações das mensagens transmitidas.

Uma Estação que deseja entrar na Região Crítica envia (n-1) Solicitações, uma para cada Estação, e fica aguardando o Consentimento de todas, quando então poderá entrar na Região Critíca.

Cada Estação que recebe uma Solicitação pode: ou Consentir porque não se encontra na Região Crítica, ou esperar o fim da sua Região Crítica e só então Consentir.

Além disso, um Carimbo de Tempo, global às Estações, é mantido de modo a resolver os conflitos entre duas Solicitações. Esses conflitos ocorrem quando uma Estação que está na Região Crítica recebe mais de uma Solicitação. Neste caso o Consentimento é dado à Solicitação com o Carimbo de Tempo mais antigo. A outra Solicitação permanece pendente.

No caso de se levar em consideração falhas e reativações de Estações, o algoritmo deve ser modificado. Para evitar que uma falha na Estação, durante a Região Crítica, provoque um Bloqueio Perpétuo, um Temporizador é utilizado da seguinte forma: a Estação, ao emitir as Solicitações, dispara o temporizador; cada esgotamento do Temporizador provoca a transmissão de uma Indagação sobre o estado de cada Estação que ainda não Consentiu e o Temporizador é rearmado; se houver Estações que não responderam às Indagações, estas são consideradas em falha.

Deve-se observar aqui que, considerando a presença de Faltas por Omissão em Estações, este algoritmo já não garante 2(n-1) mensagens para a obtenção da Região Crítica.

3.4.1.3 Exclusão Mútua na presença de Falhas

Do exame dos algoritmos mencionados, pode-se concluir que, quando se considera Faltas por Omissão, como Falha nas Estações e Perdas de Mensagens na Rede, todo o esforço de otimização do Tráfego Gerado torna-se inútil, exceto nos casos em que o esquema de Votação é utilizado.

Portanto, é desejável que o esquema de Votação seja utilizado no algoritmo de Exclusão Mútua que será a base,na presença de Falhas, para a reorganização do grupo de Estações participantes do GBDR.

3.4.2 Algoritmos para Difusão Confiável

É muito comum em Redes Locais a existência de um serviço de transmissão de mensagens em difusão física, isto é, transmisão simultânea de uma mensagem para todas as Estações de um determinado grupo. Determinadas Redes permitem a configuração de diversos grupos, aqui denominados Grupos de Difusão. Cada Estação pode pertencer a vários Grupos de Difusão.

Embora este tipo de comunicação seja um modo bastante eficaz de divulgar a mesma informação a um grupo de Estações, algumas Estações podem perder mensagens devido a erros de transmissão ou inexistência de memória disponível.

Um protocolo de Difusão Confiável deve implementar, a partir da comunicação por difusão física (ou da comunicação ponto-a-ponto), uma difusão confiável caracterizada por três propriedades relacionadas por CRISTIAN et alii [17]:

a - Atomicidade da Difusão: todos as mensagens transmitidas por difusão ou chegam a todas as Estações do Grupo de Difusão, ou não chegam a nenhuma;

b - Ordem da Difusão: as mensagens chegam na mesma ordem a todas as Estações do Grupo;

c - Terminação da Difusão: as mensagens que chegam a alguma Estação, chegam em um tempo finito a todas as demais.

A referência a Estações nas propriedades acima subentende Estações ativas, isto é, livres de Falhas.

3.4.2.1 Relação com o presente Trabalho

Neste item são mencionadas as principais conseqüências da Difusão Confiável na solução de dificuldades apresentadas em Bancos de Dados Distribuídos. Para isso, são feitas referências aos conceitos apresentados no item 3.3 (Serialização e Controle de Acesso).

As propriedades de Atomicidade e da Ordem, descritas acima, têm um forte impacto nos algoritmos de Controle de Acesso Concorrente descritos anteriormente. Por exemplo, a desvantagem da forma de Replicação "Lê Alguma e Escreve em Todas", relacionada no item 3.3.6.2 como sendo a possibilidade de Bloqueio Perpétuo, desaparece desde que se considere que as mensagens chegam a todos Gerenciadores de Acesso na mesma ordem.

Além disso, essas propriedades também têm impacto sobre a obtenção da Atomicidade das Transações, que é imprescindível ao algoritmo de Recuperação de Falhas ou a qualquer aborto imposto às Transações em função da estratégia de Escalonamento.

A forma mais comum de se obter a Atomicidade das Transações é estabelecer uma etapa de Validação da Transação, na qual, ou todos os Gerenciadores de Acesso efetivam as atualizações, ou nenhum deles as efetiva. A decisão pela efetivação se dá quando não ocorre nenhuma falha ou Bloqueio Perpétuo e quando os Gerenciadores de Acesso confirmam a serializabilidade do Log. Essa etapa é normalmente efetuada por um protocolo de Comissionamento das Atualizações realizado em duas fases.

Tal protocolo envolve um Gerenciador de Transações que está conduzindo a Transação em curso e alguns (todos, no caso de Replicação) Gerenciadores de Acesso. Utilizando-se a Difusão Confiável, o protocolo poderia ser implementado da forma descrita por CHANG [14]:

Gerenciador de Transações:

Fase I: Transmite mensagens de atualização aos Gerenciadores de Acesso e recebe as confirmações correspondentes;

Fase II: Se todas as atualizações foram aceitas

Então Divulga (Difusão Confiável) a solicitação de Comissionamento;

Senão Divulga a solicitação de Aborto;

Gerenciador de Acesso:

Fase I: Aguarda mensagens de atualização dos dados, ou,
Modificação (Falha/Reativação)no Grupo de Difusão;
Se uma atualização dos Dados foi recebida
Então Responde com uma aceitação ou rejeição em função do escalonamento e/ou da detecção de Bloqueios Perpétuos;
Senão Aborta a Transação;

Fase II: Aguarda uma solicitação de Comissionamento, Aborto ou Modificação no Grupo de Difusão;
Se Comissionamento
Então Efetiva as atualizações;
Senão Aborta a Transação;

Outro subproduto do protocolo de Difusão Confiável, que transparece no protocolo de Comissionamento, é a monitoração das Estações ativas num Grupo de Difusão. É imprescindível à Difusão Confiável o controle das Estações ativas. Portanto, é natural que essa função seja oferecida como um serviço complementar.

Esta última consideração implica em localizar também no protocolo de Difusão Confiável a Etapa de Reforma, isto é, Exclusão Mútua para organização do Grupo de Difusão e manutenção de um Carimbo de Tempo, discutidos no item sobre algoritmos para Exclusão Mútua.

Todos esses serviços são oferecidos pelo protocolo que será descrito no Capítulo VI e que foi publicado por CHANG e MAXEMCHUK [13].

Capítulo III

Especificação do Problema: Banco de Dados e Transações em um Centro de controle

1. Banco de Dados de um Centro de Controle 48
   1.1 Banco de Dados do Processo 48
      1.1.1. Características 49
      1.1.1.1 Imagem do Processo Elétrico 52
      1.1.1.2 Imagem de Processos Internos ao Centro de Controle 52
      1.1.1.3 Variáveis Elaboradas 53
      1.1.1.4 Alarmes 54
      1.1.1.5 Eventos 55
      1.1.1.6 Hora da Atualização 56
      1.1.1.7 Parâmetros de Tratamento 57
      1.1.2 Configuração 57
      1.1.3. Interfaces 57
      1.1.4. Entidades e Relacionamentos 59
      1.1.5 Transações e Organização Física do BDP 63
   1.2 Banco de Dados Históricos 65
   1.3 Banco de Dados Aplicativos 65
   1.4 Banco de Dados Estáticos 66

2. Replicação do Banco de Dados 66
   2.1 Replicação dos Dados do Processo 67
   2.2 Replicação dos Dados Históricos 67
   2.3 Replicação dos Dados Aplicativos 67
   2.4 Replicação dos Dados Estáticos 67

3. Arquiteturas Típicas 68
   3.1 Centro de Controle Local ao Processo 68
   3.2 Centro de Controle Global a Vários Processos 70

4. Relação dos Requisitos para o Gerenciador de Dados Replicados 71

1. Banco de Dados de um Centro de Controle

O objetivo deste item é descrever o Banco de Dados de um Centro de Controle típico.

O Banco de Dados de um Centro de Controle pode, para efeito da análise que se seguirá, ser subdividido em 4 classes de dados: dados dinâmicos (em tempo real) obtidos pela aquisição e monitoração do processo; dados históricos obtidos da monitoração dos dados dinâmicos, da análise de tendências e de estatísticas realizadas sobre estes; dados aplicativos gerados especificamente por programas de análise e simulação; e dados estáticos em geral.

A classificação dos dados apresentada neste Capítulo têm sido utilizada em vários trabalhos como: ANDERSON et al [1]; DENZEL et alii [19]; e MOONEY et al [35].

1.1 Banco de Dados do Processo

Para que se possa compreender melhor o Banco de Dados do Processo, é preciso examinar primeiramente o comportamento dos processos elétricos de potência sob a ótica do Centro de Controle.

Do ponto de vista de um Centro de Controle, o processo elétrico é visto como um conjunto de componentes (linhas de transmissão, transformadores, geradores, chaves, etc...) formando circuitos. A interligação destes componentes é definida por um conjunto de Variáveis Binárias que informam sobre o estado das chaves e disjuntores. O estado do processo de geração e transmissão de energia elétrica é definido por um conjunto de Variáveis Analógicas que representam tipicamente as grandezas elétricas que fluem pelo circuito.

A variação das Variáveis Binárias se dá por uma manobra, realizada pelos Operadores, que visa modificar o circuito, ou por atuação de um sistema de proteção que detecta alguma falha no Sistema Elétrico e automaticamente isola alguns componentes.

A variação das Variáveis Analógicas se dá gradualmente, a partir da atuação dos Operadores ou do Controle Automático de Geração, ou, bruscamente, a partir de uma mudança na topologia do circuito por atuação do sistema de proteção.

De modo geral, as mudanças bruscas, devido a falhas, ocorrem em frações de segundos na forma de um surto. Para o Centro de Controle o que ocorreu durante o surto só poderá ser observado após a chegada das mensagens de Eventos enviadas pelos Terminais ligados ao processo. Um surto pode envolver a alteração de até algumas centenas de Variáveis Binárias. Durante o surto, os valores medidos das grandezas oscilam por valores espúrios e estabilizam-se imediatamente após o surto. A partir dessa descrição, podemos identificar dois comportamentos do sistema elétrico: grandes períodos de estabilidade e instantes de instabilidade. O Centro de Controle deve monitorar continuamente os períodos de estabilidade e, nas instabilidades, deve ser possível armazenar a descrição do que houve baseado na capacidade de discretização dos Terminais ligados ao processo. Após um surto, as Variáveis Analógicas devem, imediatamente, estar disponíveis no Centro de Controle. Deve-se notar que as manobras de rotina realizadas pelos Operadores do Sistema não provocam instabilidades no processo e produzem não mais que algumas dezenas de modificações em Variáveis Binárias ao longo de alguns segundos. Estas Variáveis devem estar disponíveis no Centro de Controle imediatamente após as manobras.

O Banco de Dados do Processo descrito a seguir é apresentado como um caso típico de banco de dados em tempo real e é certamente o elemento que impõe as mais severas restrições ao Gerenciador de Dados Replicados.

1.1.1. Características

O gerenciamento do Banco de Dados do Processo (BDP) é a parte do "software" básico do Centro de Controle, responsável pela

interface entre os módulos destinados às funções aplicativas e o processo supervisionado.

Todas as informações relativas às variáveis supervisionadas no processo são enviadas ao Centro de Controle, sob a forma de mensagens transmitidas pelos Terminais de Medição e Controle ou por Centros Locais, e são recebidas pelo BDP de cada subsistema. A partir dessas mensagens o BDP mantém:

a) uma imagem atualizada do processo.

b) uma lista contendo os alarmes detectados durante a atualização da imagem do processo.

c) Uma lista contendo os eventos detectados no processo ou durante a atualização da imagem do processo.

A definição mais precisa de alarmes, eventos e imagem do processo será feita mais adiante.

Os módulos das funções aplicativas consultam a imagem mantida pelo BDP sempre que necessitam dados do processo e são sinalizados pelo BDP quando da ocorrência de alarmes ou eventos no processo. Neste caso, devem consumi-los e esvaziar as listas.

Na Figura III-1 é mostrado o relacionamento entre o BDP e os demais módulos. A criação do BDP se justifica pelas seguintes razões:

a - A independência entre a Rede de Terminais de Medição e Controle (Capítulo I) e os subsistemas do Centro de Controle (comunicação com o nível hierárquico superior, comunicação homem-máquina e observação do sistema computacional). Isto é obtido através de um protocolo padrão para a transferência de dados da rede de medição para o Centro de Controle. As particularidades de aquisição e pré-tratamento de dados para cada subsistema

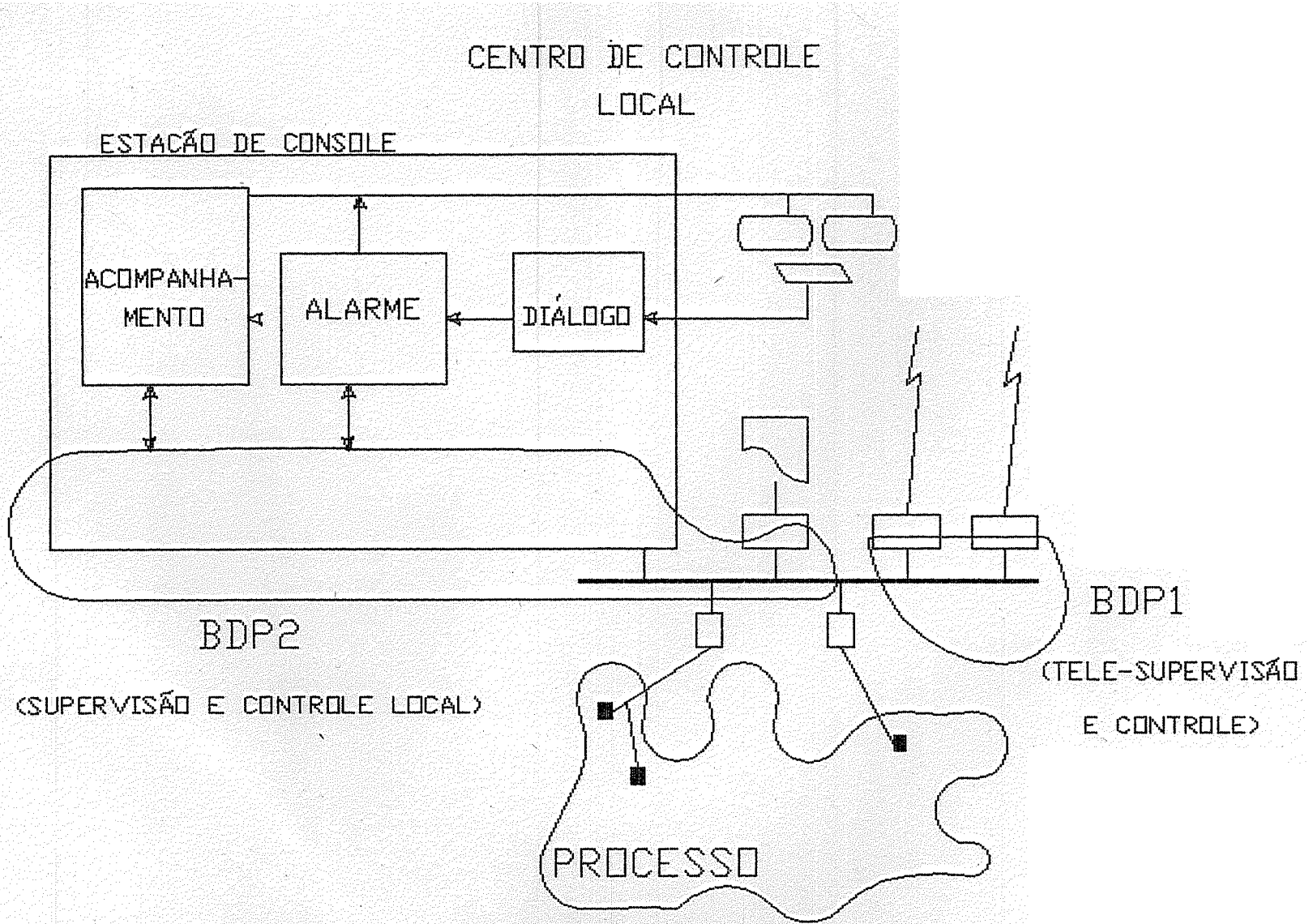


Figura III-1 : BDP e demais Módulos

ficam restritas à configuração do seu BDP. As mensagens de atualização recebidas da Rede de Medição e Controle disparam em cada subsistema uma Transação específica de atualização do seu BDP.

b - Independência entre as diversas funções aplicativas. Isto é obtido tornando o BDP o elemento de comunicação entre as diversas funções (Acompanhamento e Alarme), o que permite que um módulo desconheça a estrutura interna dos demais.

A seguir serão detalhadas as principais características do BDP.

1.1.1.1 Imagem do Processo Elétrico

O BDP mantém uma imagem do processo representada por um conjunto de variáveis binárias (assumem apenas 2 estados: aberto/fechado, verdadeiro/falso, alto/baixo, etc...) e um conjunto de variáveis contínuas digitalizadas, denominadas, respectivamente, Variáveis Binárias e Variáveis Analógicas. O fato de essas variáveis serem resultados de medidas diretas do processo permite classificá-las como Variáveis Primárias. Essa imagem é atualizada a partir das mensagens recebidas dos Terminais de Medição e Controle e fica disponível para ser acessada pelas funções aplicativas. Esses conjuntos estão organizados em duas tabelas - Variáveis Analógicas (VA) e Variáveis Binárias (VB).

Os atributos dessas Variáveis incluem parâmetros para o tratamento das grandezas recebidas do processo (filtragem, escalamento, etc.) e informações comuns a mais de um módulo das funções aplicativas (valor atual, hora da aquisição, limites de operação, condição de alarme, etc.).

1.1.1.2 Imagem de Processos Internos ao Centro de Controle

Além das Variáveis atualizadas pelos Terminais de Medição e Controle, pode-se definir no BDP Variáveis que representam o estado de processos internos ao sistema computacional do Centro de Controle.

Um primeiro exemplo de Variáveis associadas a processos internos ao Centro de Controle ocorre na Função Controle. Os procedimentos de controle são realizados pelos módulos de "software" da Função Controle. Para fins de supervisão, é necessário permitir o acompanhamento das ações de controle enviadas ao processo. Nesse sentido, o BDP permite que sejam configuradas Variáveis de supervisão (Binárias associadas a comandos emitidos pelo operador humano), as quais serão atualizadas por mensagens recebidas da Função Controle.

Outro exemplo ocorre na representação do estado do próprio sistema computacional no BDP através de um conjunto de Variáveis que são atualizadas a partir de mensagens enviadas pela Função Proteção e Recuperação.

Essa possibilidade de definir Variáveis no BDP, atualizadas por processos internos ao Centro de Controle, é um mecanismo bastante flexível e que só exige que exista no sistema uma entidade responsável pela atualização das Variáveis definidas. Essa entidade, para o BDP, deve comportar-se como um pseudo-Terminal de Medição e Controle.

1.1.1.3 Variáveis Elaboradas

O BDP permite que sejam definidas variáveis derivadas das Variáveis Primárias através de uma expressão algébrica. Estas variáveis são denominadas Variáveis Elaboradas.

Estas variáveis podem ser de 3 tipos:

. Binária - quando o resultado do cálculo for um valor binário (por exemplo, rotação < 45 RPM).

. Contínua - quando o resultado do cálculo representar uma variável contínua (por exemplo, somatório das vazões das comportas dos vertedores).

. Descontínua - quando o resultado do cálculo representar uma variável não contínua que não pode ser classificada como binária por apresentar mais de 2 estados (por exemplo, o estado de uma chave seccionadora: aberto, fechado, em trânsito, inconsistente, aberto-bloqueado, fechado-bloqueado).

As Variáveis Elaboradas Binárias são incluídas na tabela VB. As Variáveis Elaboradas Contínuas e Descontínuas são incluídas na tabela VA.

Uma Variável Elaborada é reavaliada a cada vez que alguma das variáveis utilizadas em sua expressão algébrica se modifica. Do ponto de vista de acesso às variáveis pelas funções aplicativas, não há distinção entre Variáveis Primárias e Elaboradas, pois ambas convivem nas tabelas VA e VB.

As Variáveis Elaboradas podem ainda ser classificadas como Locais ou Globais respectivamente, quando todas as variáveis utilizadas como parâmetros pertencem ou não a um mesmo Terminal de Medição e Controle.

1.1.1.4 Alarmes

Os valores assumidos pelas variáveis em VA e VB estão divididos em grupos caracterizados como estados normais e estados de alarme. A transição de uma variável de um estado normal para um estado de alarme é detectada pelo BDP que produz um item de dado denominado Alarme. Analogamente, uma Normalização é a transição contrária. Os Alarmes e as Normalizações são inseridos pelo BDP em uma lista denominada Lista de Alarmes. Cada item dessa lista informa a hora aproximada da transição, a identificação da Variável e o estado de alarme para o qual houve transição. Essa lista possui a função exclusiva de "buffer" e fica disponível para ser consumida pelas Funções aplicativas.

A definição dos estados normais e estados de alarme para cada um dos tipos de variável é fornecida a seguir:

. Tipo Binário - um de seus estados é considerado estado normal e o outro como alarme. Eventualmente, uma Variável Binária pode não ser considerada alarmante em nenhum dos dois estados.

. Tipo Analógico ou Contínuo - seu espectro de variação é dividido em 7 faixas (estouro positivo, alarme superior máximo, alarme superior normal, normal, alarme inferior normal, alarme inferior mínimo e estouro negativo). Eventualmente todo o espectro de variação de uma Variável Analógica pode ser considerado normal.

. Tipo Descontínuo - a detecção de alarme é feita de forma indireta, através da associação de uma Variável Elaborada Binária para cada Variável Descontínua com requisito de detecção de alarme. A expressão algébrica de cálculo da Variável Binária associada resume-se a atribuir um estado para os valores da Variável considerados como alarme e outro estado para os valores considerados normais.

1.1.1.5 Eventos

A detecção de eventos pode ser efetuada, no processo, pelos equipamentos da Rede de Medição, ou internamente no BDP. A ocorrência de eventos, detectados pelo processo, é caracterizada pela mudança de estado das Variáveis Binárias. A ocorrência de eventos, detectados internamente pelo BDP, é caracterizada pela mudança de estado ou valor de uma variável de VA ou VB, em função de seu tipo, conforme abaixo:

. Binária - a cada mudança de estado é associado um evento.

. Analógica ou Contínua - a cada mudança de faixa é associado um evento.

. Descontínua - a cada mudança de valor da variável é associado um evento.

O BDP mantém uma lista dos eventos ocorridos. Cada evento inserido na lista contém a informação de hora da ocorrência, identificador da Variável e seu valor.

A inserção na lista de eventos é efetuada por ordem de chegada, a partir de mensagens de eventos detectados no processo, bem como pela detecção de eventos feita internamente pelo BDP. Os eventos detectados no processo apresentam a resolução proporcionada pelos Terminais da Rede de Medição (tipicamente 1 milisegundo). A resolução obtida para os eventos detectados internamente pelo BDP varia em função da carga de processamento do Centro de Controle (aproximadamente 1 segundo). Adicionalmente, a resolução obtida para uma Variável Elaborada é inferior à de uma Primária, pois a execução de sua expressão algébrica somente é disparada ao final do ciclo de atualização do BDP, se foi constatada a variação de algum ponto utilizado como parâmetro no cálculo da expressão. Essa lista possui a função exclusiva de "buffer" e fica disponível para ser consumida pelas Funções aplicativas.

Deve-se notar que os alarmes são informações de alta prioridade que requerem a imediata ciência do operador humano e providências de normalização, e os eventos detectados pelo processo são informações precisamente sincronizadas e de baixa prioridade necessárias à análise pós-distúrbio. A detecção de eventos internamente pelo BDP visa suprir a impossibilidade do processo de detectar determinados eventos. Entretanto, tendo em vista a perda de resolução para esses eventos, deve-se, sempre que possível, utilizar a detecção de eventos pelo processo.

1.1.1.6 Hora da Atualização

As mensagens provenientes do processo são rotuladas com a hora da aquisição . O BDP mantém armazenada a hora de atualização de

cada Variável. O usuário do BDP pode verificar se uma determinada Variável está atualizada, através da comparação da sua hora de atualização com a hora do sistema.

1.1.1.7 Parâmetros de Tratamento

Associada a cada Variável de supervisão existe uma série de atributos utilizados pelo BDP no tratamento dos pontos. Por exemplo, equações de conversão para unidades de engenharia; "flags" de inibição de varredura, alarmes e eventos; valores pré-definidos; limites de operação. Esses parâmetros podem ser alterados pelos Operadores do Sistema através da Função Alteração.

1.1.2 Configuração

O BDP deverá ser configurado, "off-line", em função das características e requisitos de cada subsistema. Nesse sentido, deverão ser fornecidas ao BDP as informações de quantidade e tipo de Variáveis do processo, definição de quais pontos deverão ter a eles associados alarme e/ou seqüência de eventos, etc.

1.1.3. Interfaces

1.1.3.1 Interface com Outros Módulos

A interface com outros módulos é feita de dois modos:

- Acesso Direto (iniciativa dos módulos externos ao BDP):
  modo pelo qual a função aplicativa realiza uma Transação que consulta ou altera diretamente os valores contidos nas tabelas do BDP. Este modo de acesso é utilizado, por exemplo, pela Função Acompanhamento para a aquisição dos valores

dinâmicos da tela em apresentação ou pela Função Alarme para a atualização da condição de alarme das Variáveis.

- Sinalização (iniciativa do BDP):

  modo pelo qual uma Função aplicativa é informada sobre a ocorrência de um evento relevante para esta Função. A Função sinalizada poderá, então, consultar as estruturas de dados do BDP de modo a obter as informações sobre a ocorrência sinalizada. Esse modo é utilizado, por exemplo, para ativar a Função Alarme após a chegada de alarmes.

1.1.3.2 Interface com o Processo

A interface com o processo é feita por iniciativa dos Terminais de Medição e Controle, através de mensagens enviadas às Estações do Centro de Controle. Isto é, os equipamentos da Rede de Medição detectam a necessidade de enviar determinado tipo de dado ao Centro de Controle, montam as mensagens correspondentes e as enviam.

As informações enviadas a cada ciclo de tempo são denominadas informações periódicas e as informações enviadas assincronamente, a partir da ocorrência de eventos no processo, são denominadas informações aperiódicas.

As informações enviadas são ditas com confirmação, se as tarefas de transmissão dos equipamentos da Rede de Medição só evoluirem de estado após receberem a confirmação de recepção dos destinatários ou a evidência de que o destinatário está em falha. Caso contrário, são ditas sem confirmação.

Estão previstos os seguintes tipos de mensagens para o BDP:

- Informações periódicas sem confirmação - neste modo classificam-se as mensagens de Variáveis Analógicas e as mensagens de Variáveis Binárias para os casos em que não tenha havido mudança de estado.

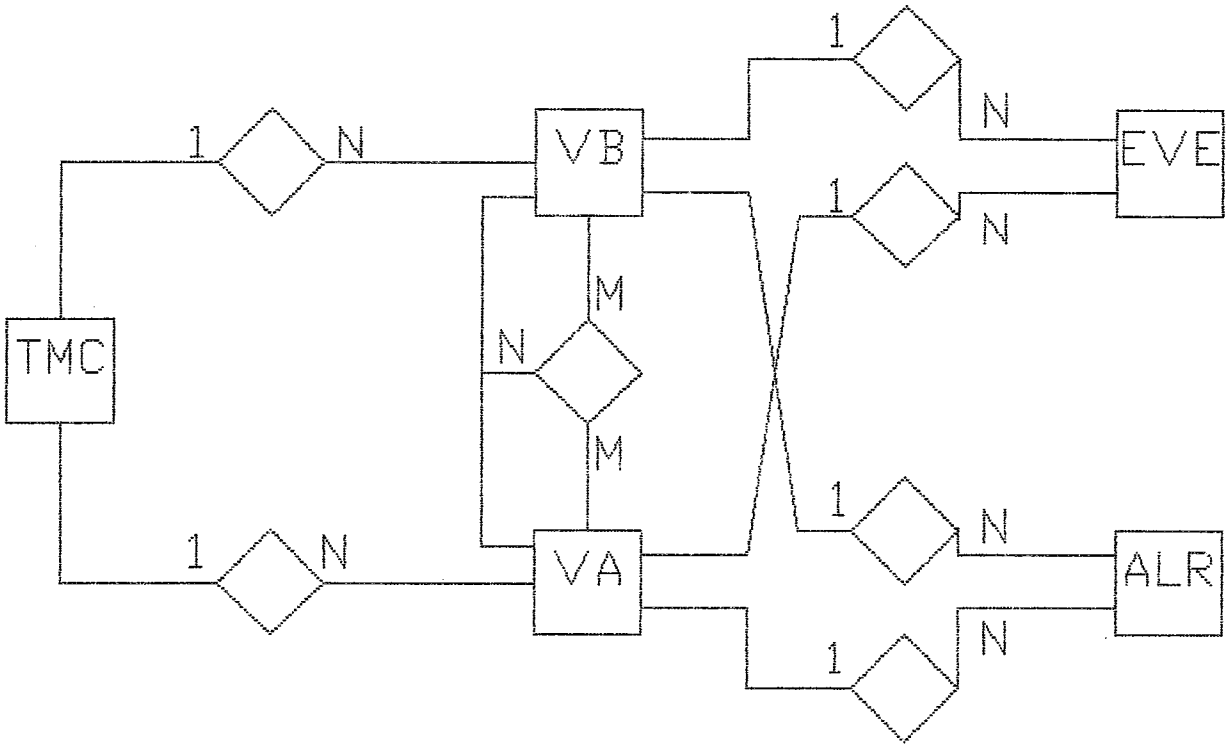
  As mensagens de Variáveis Analógicas destinam-se aos procedimentos de inicialização e atualização do BDP e têm período de 1 segundo para os Centros de Controle diretamente ligados ao processo e, tipicamente, 5 segundos para os Centros de Controle globais a vários processos. As mensagens de Variáveis Binárias sem confirmação destinam-se à inicialização e integridade do BDP e têm, tipicamente, período de 10 segundos.

- Informações Aperiódicas com Confirmação - neste modo classificam-se as mensagens de Variáveis Binárias e Eventos, sempre que for detectada a ocorrência de pelo menos uma mudança de estado (resolução de 1 milisegundo). Estas informações ficam armazenadas em "buffers" da Rede de Medição, os quais somente são liberados após a confirmação da sua recepção por todos os Subsistemas do Centro de Controle. As mensagens de Eventos destinam-se à geração da lista de seqüência de eventos. As mensagens de Variáveis Binárias com confirmação destinam-se à atualização da imagem do processo.

A ocorrência de perda de uma mensagem do tipo sem confirmação não deverá causar problemas ao funcionamento do sistema devido à sua característica periódica, fazendo com que a atualização do BDP, momentaneamente prejudicada, seja recuperada no próximo ciclo.

1.1.4. Entidades e Relacionamentos

Conforme a Figura III-2, logicamente o BDP é constituído por 5



ENTIDADES

TMC - Terminal de Medição e Controle
VB - Variáveis Binárias
VA - Variáveis Analógicas
EVE - Eventos
ALR - Alarmes

Figura III-2 : Entidades e Relacionamentos do BDP

entidades, cada qual definida por um conjunto de atributos. O modelo utilizado é o de Entidades e Relacionamentos proposto por CHEN [15].

Os atributos destas entidades são classificados em 3 categorias:

a. Estáticos - são inicializados na configuração do BDP e mantidos inalterados durante o processamento "on-line".

b. Quase Estáticos- são inicializados na configuração do BDP e alteráveis "on-line" pelo Operador do Sistema.

c. Dinâmicos - são inicializados na configuração e alterados a partir das mensagens recebidas do processo.

As 5 entidades são:

. Terminais de Medição e Controle (TMC), que possuem como atributos:
  -Estático: Endereço do terminal;
  -Quase-Estáticos: "flags" de inibição de recepção de mensagens segundo seus tipos (mensagens de Variáveis Analógicas, Digitais e Eventos).

. Variáveis Binárias (VB), que possuem como atributos:
  -Estático: estado normal;
  -Quase-Estáticos: "flags" de inibição de varredura, alarmes e eventos ;
  -Dinâmicos: hora da última atualização, estado atual,condição de alarme.

. Variáveis Analógicas (VA), que possuem como atributos:
  -Estáticos: parâmetros de conversão e filtragem;

-Quase-Estáticos: faixas de operação, "flags" de inibição de varredura, alarmes e eventos.

-Dinâmicos: hora da última atualização, valor bruto, valor convertido, condição de operação, condição de alarme.

. Eventos (EVE), que possuem como atributos Dinâmicos:
tipo e identificador da Variável, condição de operação e hora do evento.

. Alarmes (ALR), que possuem como atributos Dinâmicos:
tipo e índice da Variável, condição de operação e hora do alarme.

Além dessas entidades, existem 7 relacionamentos:

. Variáveis Binárias de um Terminal, que é uma relação 1 para N de TMC para VB (relacionamento estático).

. Variáveis Analógicas de um Terminal, que é uma relação 1 para N de TMC para VA (relacionamento estático).

. Eventos de uma Variável Binária, que é uma relação 1 para N de VB para EVE (atualizado dinamicamente).

. Eventos de uma Variável Analógica, que é uma relação 1 para N de VA para EVE (atualizada dinamicamente).

. Alarmes de uma Variável Analógica, que é uma relação 1 para N de VA para ALR.

. Alarmes de uma Variável Binária, que é uma relação 1 para N de VB para ALR.

. Parâmetros de uma Variável Elaborada, que é uma relação N para M entre as Variáveis Elaboradas (VA ou VB) e as Variáveis Primárias que são seus parâmetros (VA ou VB), e,

que possuem como atributo a expressão algébrica de cálculo da Variável Elaborada.

1.1.5 Transações e Organização Física do BDP

1.1.5.1 Atualização das Variáveis Primárias e Eventos

A atualização das Variáveis Primárias é iniciada a partir da chegada de uma mensagem de atualização de algum Terminal. A mensagem comporta um lote de atualizações referentes a todo um Terminal. A Transação consiste em ler os atributos para tratamento, calcular o novo valor para cada variável e escrevê-lo. As Variáveis modificadas, que são parâmetros de Variáveis Elaboradas Locais, disparam a reavaliação deste. Finalmente, os eventos e alarmes detectados são inseridos nas respectivas listas ainda na mesma Transação.

Deve-se notar que é conveniente que a organização física das tabelas do BDP seja tal que as Variáveis de um mesmo terminal fiquem contíguas. Além disso, que não sejam entremeados atributos dinâmicos e quase-estáticos. Isso permite que tanto a leitura dos atributos para tratamento quanto a escrita dos valores atualizados sejam feitas como um lote.

A chegada de mensagens de Eventos detectados no processo dispara uma Transação que simplesmente os insere na Lista.

1.1.5.2 Atualização das Variáveis Elaboradas Globais

A transação de atualização das Variáveis Elaboradas Globais apresenta dois problemas principais:

a - Alta possibilidade de conflito com a atualização das Variáveis Primárias: cada Variável Elaborada Global utiliza como parâmetro Variáveis Primárias espalhadas por todo o Banco de Dados do Processo, sem nenhuma relação

com a ordenação por Terminal de Medição e Controle. Do ponto de vista do controle de concorrência,a atualização de uma única Variável Elaborada pode bloquear a atualização de todas as Variáveis Primárias, já que estas são atualizadas em lote, por Terminal de Medição e Controle.

b - Possibilidade de Resultados Espúrios: o resultado da avaliação de uma Variável Elaborada pode apresentar um valor irreal se as Variáveis Primárias utilizadas como parâmetros não apresentarem uma simultaneidade adequada. Ou seja, determinados parâmetros representam o valor apresentado pelo processo em instante t_1 e outros parâmetros refletem o processo num instante t_2. Se $|t_2-t_1|$ for demasiadamente grande, o resultado de uma variável elaborada que utiliza esses parâmetros pode não ter significado nenhum. Pode-se argumentar que, se a taxa de mudanças no processo não for muito grande, o resultado acabará convergindo para o valor correto. Entretanto, os valores espúrios podem significar eventos para o sistema e ficar armazenados no Banco de Dados Históricos representando uma informação incorreta.

Uma solução para o primeiro problema é separar em Transações distintas a atualização das Variáveis Primárias e das Variáveis Elaboradas. Isso implica em reduzir a simultaneidade do Banco de Dados do Processo. Mas desse modo é possível bloquear todas as Variáveis Primárias para leitura e atualizar as Variáveis Elaboradas como um lote.

No segundo problema, a solução é estabelecer para cada Variável Elaborada um atributo de limite de simultaneidade que, se ultrapassado, inibe a atualização da Variável Elaborada.

Nas Variáveis Elaboradas a partir de funções contínuas de Variáveis Primárias contínuas e com pequenas taxas de variação (freqüência da maior componente harmônica << 1/período de amostragem), esse limite pode ser da ordem do período de amostragem.

Nas Variáveis Elaboradas Descontínuas, deve-se utilizar o limite de simultaneidade igual a zero. Nesse caso, é preciso que as amostragens pelos Terminais de Medição e Controle estejam sincronizadas. Outra solução é inibir os registros históricos e suportar momentaneamente alguns valores espúrios na imagem do processo. Isso é aceitável se a variação ocorrer em surtos e se a informação apresentada é filtrada pelo "software" ou mesmo pelo operador humano. Durante o instante do surto de variações, o valor fica indisponível. Imediatamente após o surto, a Variável Elaborada estabiliza em algum valor que é apresentado.

1.2 Banco de Dados Históricos

Essa classe de dados engloba todas as informações de eventos relevantes ocorridos anteriormente no sistema . Tais informações serão, por exemplo, utilizadas em atividades de planejamento e estudo do processo e não devem ser descartadas até que os operadores do sistema autorizem. Tipicamente, o Banco de Dados Históricos é constituído por Listas de Eventos, Alarmes, Relatórios e Resultados de simulações que registrem alguma singularidade. Constituem ainda esta classe de dados as informações estatísticas e tendências extraídas da observação do sistema ao longo do tempo.

As Transações típicas nesse Banco de Dados são de inserção e retirada de elementos em Listas e ocorrem esporadicamente em função dos surtos no processo, ou por solicitação do operador.

1.3 Banco de Dados Aplicativos

De um modo geral, os programas que geram essa classe de dados são instalados em Centros de Controle Globais e de maior porte.

Os programas de Análise de Rede geram periodicamente configurações estimadas para o processo. Portanto, o volume de

dados produzidos a cada ciclo é muito grande, da ordem da quantidade de dados do BDP. A Transação típica é obter uma imagem do processo, analisá-la e produzir uma imagem consistente estimada, a partir das redundâncias existentes na imagem original e das características do circuito elétrico. A imagem estimada pode ser denotada por BDP*.

Os programas de Análise de Segurança, a partir da simulação de perdas e reentradas de determinados componentes sobre uma imagem estimada do processo, produzem listas de violações de sobrecargas nos elementos do processo e índices de severidade das ocorrências simuladas.

Os programas de análise de alarmes e diagnóstico de falhas examinam os alarmes e eventos e geram listas de possíveis causas.

1.4 Banco de Dados Estáticos

O volume de dados estáticos no Centro de Controle é muito grande (centenas de Kbytes). Estes dados são gerados "off-line" e incluem: formatos de telas e relatórios; mapeamentos entre a organização apresentada pelos dados dinâmicos e a organização definida para cada função aplicativa; descrição de topologia do processo elétrico e características elétricas de cada um dos seus componentes.

2. Replicação do Banco de Dados

Como foi observado anteriormente, o GBDR garante essencialmente duas propriedades principais dos seus repositórios de dados: a tolerância a faltas e a onipresença. O objetivo deste item é, com base nessa conceituação, justificar a seleção no Banco de Dados do Centro de Controle dos dados que devem ser replicados.

2.1 Replicação dos Dados do Processo

Os Dados do Processo devem ser replicados via o GBDR, principalmente para que sejam acessíveis por todas as Estações do Centro de Controle. Além disso, é importante que os "buffers" de Alarmes e Eventos e os parâmetros do tratamento mantidos pelo BDP sejam Tolerantes a Faltas. Outra vantagem de Replicação do BDP é que isso aumenta a capacidade de processamento, já que as Estações podem dinamicamente repartir entre si o tratamento das mensagens provenientes do processo e cada qual atualizar uma parte do BDP.

2.2 Replicação dos Dados Históricos

Os Dados Históricos representam sínteses e fatos relevantes ocorridos no Centro de Controle. Devem, portanto, ser replicados via GBDR de modo a apresentarem uma tolerância a faltas adequada.

2.3 Replicação dos Dados Aplicativos

O Banco de Dados Aplicativos é periodicamente atualizado. Algum resultado relevante pode produzir um item de Dados no Banco de Dados Históricos, mas, de um modo geral, em caso de falha, sua atualização pode ser interrompida e retomada após a recuperação. Quanto à divulgação da imagem estimada pelo Centro de Controle, BDP*, também pode ser feita via GBDR. Isto é, periodicamente é realizada uma Transação de atualização de todo o BDP*.

2.4 Replicação dos Dados Estáticos

A replicação de Dados Estáticos é trivial, pois pode ser feita "off-line" e é usada como forma de agilizar o processamento. Portanto, os Dados Estáticos não impõem restrições ao GBDR.

3. Arquiteturas Típicas

3.1 Centro de Controle Local ao Processo

Na Figura III-3, é apresentada uma possível arquitetura do BDP para um Centro de Controle Local a determinado Processo Elétrico. Por exemplo, em uma Usina de Geração de Energia Elétrica ou em uma Subestação de Transmissão.

Considerando a proximidade dos Terminais de Medição e Controle, não é necessária a interposição de nenhuma Estação dedicada à interface com os Terminais. Estes enviam os Dados diretamente às Transações que atualizam o BDP. As Transações podem estar repartidas entre as Estações de modo a aumentar a capacidade de processamento das atualizações. Além disso, são representados três subsistemas: Supervisão e Controle Local; Tele-Supervisão e Controle; e Supervisão do Sistema Computacional. Cada um destes subsistemas possui um BDP configurado segundo as suas necessidades.

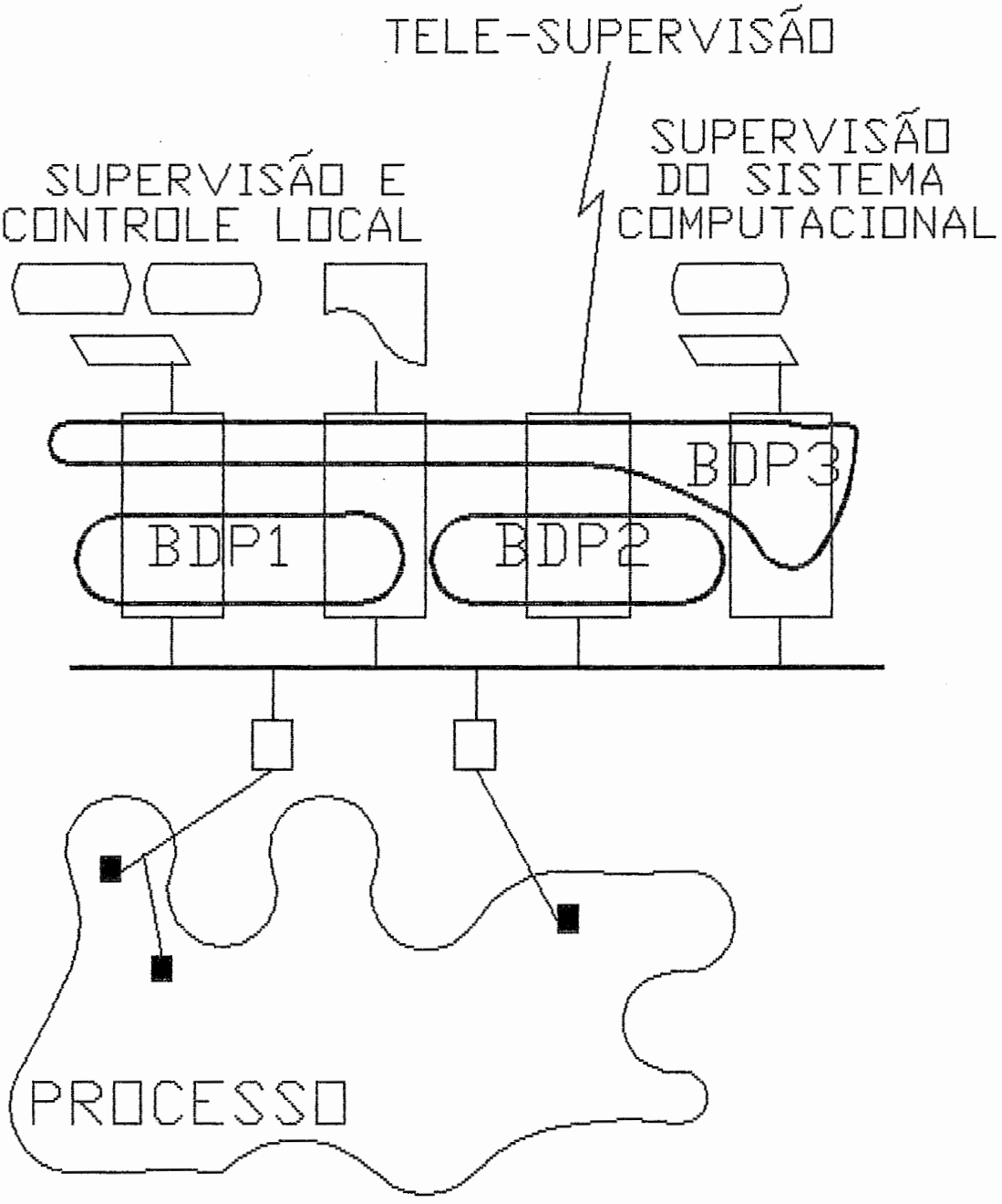


Figura III-3 : BDP para um Centro de Controle Local

3.2 Centro de Controle Global a Vários Processos

Na Figura III-4, é apresentada uma possível arquitetura do BDP para um Centro de Controle Global a dois outros Centros de Controle de um nível hierárquico inferior. Por exemplo, um Centro de Controle Regional coordenador de dois outros Centros.

Considerando o volume de informações a serem processadas e o gerenciamento dos canais de comunicação, é necessária a interposição de Estações ("Front-Ends") dedicadas à interface com os demais Centros e ao gerenciamento dos canais de comunicação. Estes enviam os Dados diretamente às Transações localizadas nos "Front-Ends" que atualizam o BDP. Neste caso, o BDP é repartido em função do número de "Front-Ends" existentes.

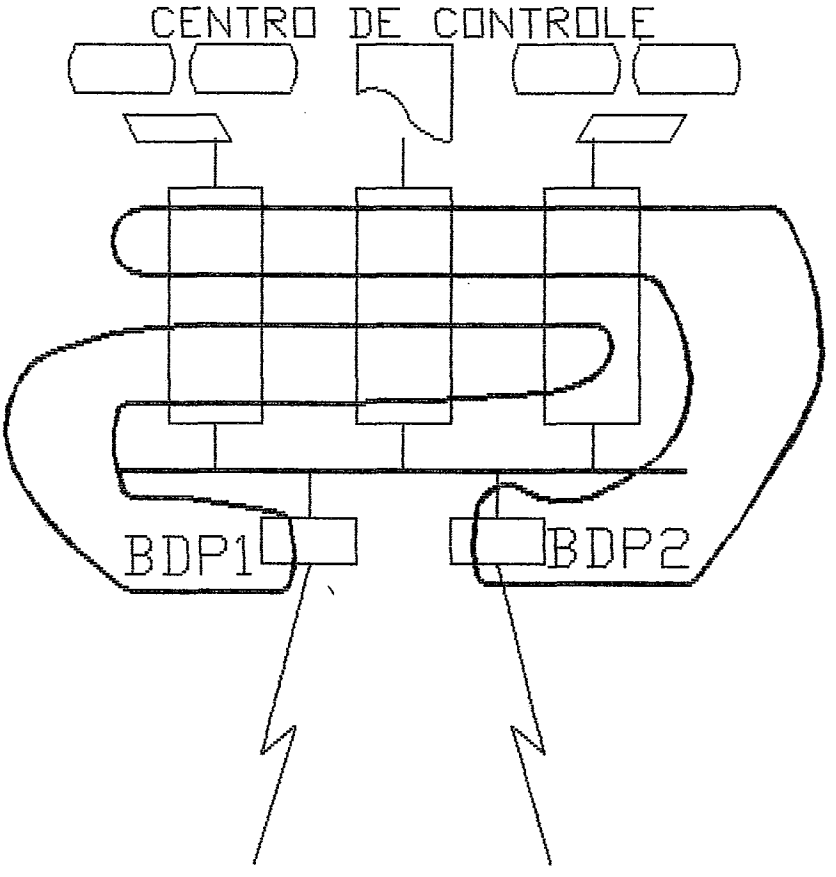


Figura III-4 : BDP para um Centro de Controle Global

4. Relação dos Requisitos para o Gerenciador de Dados Replicados

Em função de tudo o que foi apresentado, pode-se resumir os requisitos para o GBDR de um Centro de Controle nos seguintes itens:

a - O GBDR deve tipicamente suportar os seguintes tipos de Dados e Transações:

- Variáveis do Processo

  --> atualizadas a partir de chegada de informações do processo
  --> quantidade proporcional ao tamanho do processo (tipicamente 1000 Binárias e 500 Analógicas)
  --> tipicamente 10 bytes/Variável Analógica (hora+valor)
  --> tipicamente 5 bytes/Variável Binária (hora + valor)
  --> todas as Variáveis Analógicas são reescritas a cada segundo
  --> armazenadas na memória principal
  --> fisicamente organizadas por terminal de aquisição
  --> atualização por lote, explorando a contigüidade física

- Parâmetros de tratamento das Variáveis
  --> quantidade proporcional ao número de Variáveis do processo (tipicamente 1500)
  --> alteráveis pelo operador
  --> consultados a cada segundo para tratamento das Variáveis Analógicas
  --> armazenado na memória principal
  --> raramente alterados (algumas alterações por dia) mas o tempo de uma alteração deve ser inferior a um segundo.
  --> tipicamente 10 bytes por variável
  --> fisicamente organizados por Terminal de Aquisição
  --> leitura por lote, explorando a contigüidade física

- "Buffers" de Eventos e Alarmes

  --> registros de dados produzidos a partir do tratamento das Variáveis do processo ou detectados nos Terminais de Aquisição e Controle
  --> capacidade típica para 1000 itens
  --> 10 bytes/item
  --> os registros são raramente produzidas (algumas vezes por semana) em grandes lotes (200 itens) que devem ser inseridos em no máximo um segundo.
  --> o consumo dos registros é feito lentamente (até 1 segundo por item)
  --> armazenados em memória principal

- Dados Históricos

  --> armazenados em arquivo indexado (tipicamente 10.000 registros de 100 bytes)
  --> produzidos a partir das seqüências de Eventos e a partir de disparos períodicos de leitura dos Dados do Processo
  --> armazenados em memória secundária
  --> leitura e remoção dos dados comandados pelo operador

-Variáveis Estimadas

  --> atualizadas periódicamente pelo programa aplicativo de estimação
  --> tamanho igual ao das Variáveis do Processo.
  --> todas as Variáveis reescritas de uma só vez a cada 10 segundos.
  --> armazenadas em memória principal.

b - O GBDR deve implementar repositórios de informações replicados por um determinado conjunto de Estações pré-configurado para a rede. Os repositórios replicados são mapeáveis em memória primária ou dispositivos de memória secundária das Estações.

Por exemplo:

--> configuração da rede:

"O repositório /R0 está nas Estações 1, 2 e 3"

"O repositório /R1 está nas Estações 3 e 4"

--> configuração por Estação:

"Na Estação 1, o repositório /R0 é /user/abc"

"Na Estação 2, o repositório /R0 é /user/def"

"Na Estação 3, o repositório /R0 é /user/ghi, e, /R1 é /dev/ram"

"Na Estação 4, o repositório /R1 é /dev/ram"

b.1 - A justificativa para a implementação de um Gerenciador de vários repositórios replicados é que o repositório é a entidade à qual se podem associar características operacionais tais como:

- "disco residente em memória"-é um dispositivo implementado em memória principal que implementa a simulação de um disco físico com alto desempenho.

- "disco confiável"-é uma camada de "software" sobreposta ao disco físico que garante a atomicidade das escritas garantindo, assim, a integridade do disco mesmo quando são consideradas interrupções durante a escrita. Esta técnica é descrita por KOHLER [27].

- "disco com armazenamento contínuo" - é o "software" de acesso a disco que permite garantir a contigüidade física dos dados de um mesmo arquivo. Essa propriedade pode ser útil em situações tais como as apresentadas por MOONEY et al [35].

- "disco com duplo método de acesso" - é o "software" de acesso a discos em memória que permite, além da interface convencional via nome de arquivo, o acesso direto ao dispositivo de armazenamento, como é descrito por MOONEY [35].

c - O GBDR deve implementar a recuperação de falha parcial (entrada e saída de Estações) de modo de que "UNDOS" e "UNLOCKS" das Transações não concluídas fiquem transparentes à aplicação.

d - O GBDR deve fornecer um mecanismo que permita evitar Bloqueios Perpétuos de Transações.

e - O GBDR deve implementar de modo transparente à aplicação a iniciação dos repositórios replicados com o seguinte critério:

Ao ser ativada uma Estação:
Se há alguma outra Estação em serviço com este repositório
então copia todo o repositório da Estação em serviço para o repositório próprio
senão
Se o repositório próprio está consistente, isto é, não foi alterado desde a geração "off-line"
então assume o conteúdo gerado "off-line"
senão a Estação não pode ser iniciada

f - O GBDR deve suportar que:

- Vários repositórios sejam definidos numa mesma rede, cada qual replicado em um conjunto de Estações.
- Uma Estação contenha vários repositórios replicados.

É, finalmente, admissível a simplificação de uma Transação ficar sempre restrita a um único repositório. Isto é, todos os itens de dados por ela manipulados pertencerem a um único repositório.

Capítulo IV

Proposta Básica

1. Discussão Preliminar 76
   1.1 Suposições quanto aos Tipos de Faltas 76
   1.2 Modelo do Banco de Dados Replicados e um Algoritmo Trivial 77
      1.2.1 Modelo 77
      1.2.2 Suposições Simplificadoras Sobre o Sistema 78
      1.2.3 Um Algoritmo Trivial 79
   1.3 Análise de Possíveis Otimizações 81
      1.3.1. Registro Seqüencial 81
      1.3.2 Análise da Etapa de Validação 82
      1.3.3 Análise do Controle da Concorrência de Acesso a Itens do BDR 83

2. Estratégia Utilizada no Gerenciador 84

1. Discussão Preliminar

O objetivo deste Capítulo é discutir um algoritmo para o Controle de Acesso Concorrente e Recuperação de Falhas do GBDR.

Nenhum algoritmo pode funcionar corretamente submetido a todos os tipos de Faltas do sistema computacional. Portanto, é necessário estabelecer as condições mais plausíveis de Falta e adequar o algoritmo a essas condições. Essa delimitação será feita no item a seguir.

Mais adiante, a partir dos requisitos estabelecidos no Capítulo anterior, será estabelecido um modelo básico para o banco de dados e um algoritmo trivial ("força-bruta") será apresentado. A seguir, a partir dos algoritmos discutidos no Capítulo II, algumas otimizações serão exploradas.

1.1 Suposições quanto aos tipos de Faltas

Neste trabalho supõe-se que as Faltas nas Estações produzem Falhas estanques. Isto é, Falhas que não se propagam e cujo efeito é unicamente anular um componente do sistema: Estação ou canal de comunicação. Como foi visto no Capítulo II, as Faltas que ocasionam este tipo de Falha são denominadas Faltas por Omissão. Para a obtenção deste confinamento das Falhas, existe no Centro de Controle um subsistema de detecção e recuperação de Falhas: Sistema de Observação e Salvamento descrito por ANDRADE [2].

Assim, quando ocorre uma Falha em uma Estação, ela simplesmente é desativada. Quando a Falta é sanada, a Estação é reativada e inicia o processo de recuperação.

Aqui não são consideradas Faltas Bizantinas. Faltas por Omissão e Faltas Bizantinas são dois limites entre os quais encontram-se as Faltas de um sistema real. Supor Falhas estanques equivale a supor que a ocorrência de Falha é detectada antes que algum dano irrecuperável seja causado no Banco de Dados.

Supõe-se ainda que a Falha em determinada Estação é detectável pelas demais após um determinado número de repetições de envio de mensagens e esgotamento de temporizadores na espera da confirmação de mensagens. Essa suposição evita uma considerável sobrecarga de processamento e comunicação de um sistema dedicado à averiguação do estado de cada Estação.

Finalmente, supõe-se que a rede nunca sofre um seccionamento tal que todas as partições resultantes apresentem uma quantidade de Estações inferior à maioria. Ou seja, sempre existe um conjunto com a maioria das Estações podendo comunicar-se. Esta suposição é aceitável, considerando que as Estações estão interligadas por uma rede local sem elementos centrais ativos. Deve-se considerar nesse trabalho a possibilidade de duas Estações estarem ativas e ainda assim não conseguirem se comunicar na primeira tentativa devido a ruído no meio de comunicação ou transbordamento da fila de mensagens recebidas. Supõe-se, neste caso, que existe um número R de tentativas de comunicação, após o qual só não se obtém sucesso se a Estação estiver em Falha. Esse número R é função da taxa de ruído. O tempo entre tentativas depende da duração média do ruído e do tempo médio para esvaziar as filas de recepção de mensagens.

1.2 Modelo do Banco de Dados Replicados e um Algoritmo Trivial

O sistema pode possuir vários Bancos de Dados Replicados: BDR_1, BDR_2, etc... . Entretanto, uma transação dá-se dentro de um único Banco de Dados Replicados (BDR_j). Portanto, inicialmente, para efeito de modelagem, pode-se considerar um único BDR. O modelo descrito a seguir é uma extensão do modelo descrito no Capítulo II.

1.2.1 Modelo

Logicamente, um BDR é hierarquicamente constituído por um conjunto de arquivos ou Depósitos de Dados {D0,D1, ...} e cada

depósito de dados é constituído por uma seqüência de Itens de Dados {I0, I1, I2,...}. Cada Item representa uma porção logicamente contínua do Depósito de Dados. O gerenciador de Banco de Dados Replicados (GBDR) processa operações de Leitura e Escrita sobre esses Itens. A operação Ler (D0, I3) retorna o valor atual do Item I3. A operação Escrever (D1, I2) atribui um novo valor ao Item I2. Os programas usuários interagem com o GBDR através de seqüências de Leituras e Escritas denominadas Transações.

Fisicamente, um BDR possui várias réplicas. Cada uma está armazenada em uma Estação. As Estações do Sistema são identificadas por um número Ei. A réplica do BDR na Estação Ei é denotada por BD_{Ei}. A réplica do Depósito Dk na Estação Ei é denotada por Dk_{Ei}.

Logicamente, o GBDR deve executar as Transações concorrentes como se fossem realizadas e executadas serialmente, uma após outra, sobre o Banco de Dados e garantir que todas as suas réplicas permaneçam iguais. Deve garantir ainda que, na ativação de uma Estação, a sua réplica do BDR esteja igual às demais réplicas.

Uma Transação que opere sobre um determinado BDR deve residir em uma Estação que contenha uma réplica do BDR. Esta réplica é denominada Réplica Local à Transação. Analogamente, pode-se definir Réplica Local de um Depósito de Dados e de um Item.

1.2.2 Suposições Simplificadoras Sobre o Sistema

Falha total significa que todas as Estações hospedeiras de um mesmo BDR entraram em Falha.

Supõe-se que, dado o número de réplicas do Banco de Dados, a Falha total do sistema seja de probabilidade desprezível. Ou seja, uma Estação, quando é recuperada após uma Falha, encontra réplicas íntegras do Banco de Dados que permaneceram ativas durante a sua recuperação. Na hipótese remota de ocorrer a

Falha total, é admissível que o GBDR ignore o último estado assumido pelo Banco de Dados e reinicie de um estado pré-estabelecido.

Em caso de Falha, as Transações em curso naquela Estação são abortadas e a Réplica Local do Banco de Dados fica indisponível como efeito da Falha.

Supõe-se também que o conjunto Unidade de Processamento-Dispositivos Físicos de Armazenamento, que constitui uma Estação, comporta-se monoliticamente do ponto de vista de Falha. Isto é, se algum dispositivo falha, a Estação falha como um todo.

1.2.3 Um Algoritmo Trivial

Num sistema ideal, onde não houvesse restrições de capacidade de processamento e comunicação, um algoritmo simples, para Controle de Acesso Concorrente e Recuperação de Falhas em Banco de Dados Replicados, seria o seguinte:

a - O Controle de Acesso Concorrente é obtido através de Transações que utilizam o Bloqueio em Duas Fases. O GBDR controla a concorrência e é distribuído pelas Estações.

b - Quando uma Transação deseja ler um Item, ela bloqueia a sua Réplica Local para leitura, lê o item e o libera ao final da Transação.

c - Quando uma Transação deseja escrever um Item, ela bloqueia para escrita todas as réplicas deste Item no BDR, atualiza todas elas e as libera ao final da Transação. O valor anterior ao bloqueio é salvo em um Registro Seqüencial ("logging"), em cada Estação. No caso da Transação ser abortada, o Item pode ser restaurado a partir do valor salvo no Registro Seqüencial.

d - Para evitar o Bloqueio Perpétuo ("deadlock") de uma Transação, devido à concorrência de várias Transações no acesso

aos mesmos Itens, a estrátegia é a seguinte: todos os Itens do BDR são previamente ordenados: D0/I0, D0/I1, ..., D1/I0,D1/I1,... . Existem muitas possibilidades para se ordenar os Itens. Entretanto, é conveniente utilizar uma ordem associada a posição dos Itens, pois a posição já é uma informação necessária no acesso aos Itens. Para isso, deve-se formalizar a idéia de posição de um Item. Os Itens são subconjuntos dos Itens Elementares que constituem os Depósitos de Dados. Um Item Elementar é uma unidade mínima, indivisível, de armazenamento. Um Item elementar pode ser visto como um elemento de uma aplicação de um conjunto P de posições num conjunto C de conteúdos. De um modo mais intuitivo, pode-se assumir que a posição de um Item Elementar é a posição física do seu primeiro "byte" e é expressa em "bytes". Pode-se definir posição de um Item como a seqüência crescente das posições dos seus Itens Elementares. Os Itens, mesmo os que possuem alguma interseção, podem ser ordenados da seguinte maneira: dados Im e Is em Dk, Im < Is sempre que a posição de Im for lexicograficamente menor que a posição de Is. Em qualquer Transação, o bloqueio dos Itens segue esta mesma ordem. A tentativa de bloqueio, em qualquer outra ordem, leva o GBDR a abortar a Transação.

e - Em caso de Falha de uma Estação, as Transações em curso residentes nessa Estação são abortadas. Nas demais Estações, os Itens bloqueados pelas Transações abortadas são automaticamente liberados pelo GBDR.

f - Em caso de recuperação, após Falha, de uma Estação, nenhuma nova Transação é autorizada pelo GBDR. As Transações em curso prosseguem até sua conclusão. Quando a atividade de atualização do Banco de Dados cessar, a Estação recuperada iniciará a cópia de todo o Banco de Dados de alguma outra Estação para a sua Réplica Local.

A simplicidade desse algoritmo provém de duas suposições que devem ser explicitadas:

a - A Possibilidade de Difusão Atômica de Informações entre Estações.
Isto está implícito em vários passos: no item c (Escrita de um Item), a solicitação de bloqueio e o valor atualizado do Item de Dado precisam ser divulgados pelas Estações atomicamente, isto é, todas as Estações ativas precisam receber a mesma informação e na mesma ordem em relação às demais difusões; nos itens e (Falha de uma Estação) e f (Recuperação de uma Estação), a Falha e recuperação são eventos que também precisam ser divulgados atomicamente.

b - Restrições ao Paralelismo.
Os itens a (Bloqueio em Duas Fases) e d (Solicitação de Bloqueio em Ordem pré-fixada) combinados podem, dependendo das Transações típicas, tornar puramente serial a atividade de atualização do Banco de Dados. Ainda o item f (Recuperação a partir da paralisação das atualizações) presume que a capacidade de processamento e comunicação seja de tal ordem que interromper a atualizaçao para cópia do Banco de Dados não comprometa o desempenho do sistema.

Nem todas as simplificações utilizadas podem ser suportadas pelos requisitos relacionados para o BDR de um Centro de Controle. É necessário otimizar determinados pontos que serão discutidos a seguir.

1.3 Análise de Possíveis Otimizações

1.3.1. Registro Seqüencial

É observado por KOHLER [27] que a recuperação de Falhas no sistema necessita que as Transações sejam implementadas com as seguintes características:

a - Os Itens de Dados atualizados não devem ser liberados até que toda a Transação se complete. Uma vez que um Item atualizado seja liberado, a recuperação não poderá mais ser garantida, pois uma nova Transação poderá utilizá-lo e ele se

tornará dependente dessa nova Transação. Sua recuperação implicaria em abortar a nova Transação. No limite, ocorreria o Aborto em Cascata.

b - Os estados iniciais de todos os itens modificados por uma transação devem poder ser restaurados. Isso normalmente envolve a manutenção de um Registro Seqüencial ("logging"), em dispositivo de armazenamento não volátil, contendo todas as atualizações realizadas.

Num sistema de supervisão e controle distribuído, no qual só sejam admissíveis Falhas parciais, um Registro Seqüencial em memória não volátil das ações que constituem uma Transação é dispensável. No caso de Falha total, a recuperação a partir de um estado padrão, ou a recuperação a partir do último estado anterior à Falha, é indiferente.

Segundo esta consideração, é importante elaborar um mecanismo eficaz de recuperação às Falhas parciais. E, no caso de um Banco de Dados Replicados, a recuperação de uma Estação pode ser feita a partir de outra que tenha permanecido ativa durante a Falha.

No caso da interrupção de uma Transação no meio, deve-se considerar que as atualizações são feitas sobre a Réplica do BDR Local. Ao final da Transação, na etapa de Validação, o resultado da transação é divulgado a todas as demais réplicas do Banco de Dados. Se a transação for concluída com êxito, todas as réplicas serão atualizadas, senão a Réplica Local será recuperada a partir das demais réplicas. Deve-se notar que uma Transação só é abortada devido a uma Falha, pois o controle de acesso descrito previne Bloqueios Perpétuos desde que as Transações estejam corretas.

1.3.2 Análise da Etapa de Validação

A agregação de várias etapas de um protocolo em poucas mensagens é uma técnica largamente utilizada para otimizar

algoritmos distribuídos. No algoritmo de BIRMAN et al [9] todas as mensagens de atualização são reunidas às mensagens de controle de concorrência. O princípio básico é que, em muitos sistemas distribuídos, o número de mensagens enviadas, e não o seu tamanho, é o principal fator de sobrecarga.

Num Banco de Dados Distribuído, a divulgação e aprovação final das atualizações devem ser feitas numa etapa de Validação. Esta aprovação envolve a divulgação das atualizações e a recepção da confirmação de todas as Estações.

Por outro lado, a implementação de um protocolo de Difusão Confiável apresenta as seguintes propriedades:

a - Todas as estações ativas recebem todas as mensagens divulgadas.

b - Todas as mensagens são recebidas na mesma ordem pelas estações.

A implementação destes requisitos sobre uma rede não confiável requer um protocolo específico.

A implementação da Validação, baseada nas propriedades do protocolo de Difusão Confiável, possibilita uma otimização considerável.

1.3.3 Análise do Controle da Concorrência de Acesso a Itens do BDR

A replicação é um caso particular da distribuição de dados. Entretanto, considerar um banco de dados replicados como um banco de dados distribuídos, no qual uma das restrições semânticas seja que as suas várias réplicas mantenham-se iguais, mascara uma grande possibilidade de otimização. No sentido de ressaltar essa possibilidade, BERNSTEIN et al [8] faz uma distinção entre controle da replicação e controle do acesso concorrente.

A possibilidade de otimização decorre do fato de que, num banco de dados replicados, qualquer Transação atua sempre sobre todas as réplicas, enquanto num banco de dados distribuídos há a possibilidade de uma Transação atuar sobre apenas uma das réplicas. Na primeira hipótese, para se ganhar o controle sobre um determinado Item, basta que uma das Estações aprove o pedido de bloqueio para sabermos que as demais estarão na mesma situação. Na segunda hipótese , como um Item pode ser bloqueado individualmente, é necessário que todas as Estações validem uma solicitação de bloqueio. A otimização implícita no primeiro caso resulta da necessidade de se aguardar apenas uma confirmação para prosseguir uma Transação.

2. Estratégia Utilizada no Gerenciador

A partir das otimizações discutidas no item anterior, a estratégia utilizada no Gerenciador baseia-se nos seguintes pontos:

a - Difusão Confiável para manter a consistência das Estações na questão do Controle do Acesso Concorrente e divulgação das atualizações. Esta difusão deverá ser estendida ao protocolo de comunicação de modo a otimizar o desempenho (utilização da difusão Física).

b - Escalonamento das Transações por Bloqueio ("Locking") Incremental e desbloqueio total ao final da Transação. Isto é, durante a Transação Bloqueios são solicitados sem nenhuma solicitação de desbloqueio. Ao final da Transação todos os Itens são desbloqueados simultaneamente. Este é um caso particular do Bloqueio em Duas Fases. A título de ilustração, pode-se indicar a forma de demonstração de que o escalonamento produzido pelo Bloqueio em Duas Fases é correto (ver [7]).

Demonstração:
A partir dos conceitos desenvolvidos no Capítulo II, uma estratégia de escalonamento de um GBDR é correta se só produz

Logs SR, isto é, serializáveis. Ou seja, se os grafos de serialização associados não possuem ciclos. Seja um Log produzido pelo Bloqueio em Duas Fases e GS(Log) o grafo de serialização associado. Se um arco (Ti,Tj) pertence ao GS(Log), então, significa que Ti realizou um desbloqueio de um Item de Dados antes de Tj bloquear o mesmo Item. Se existisse um ciclo em GS(Log), isto é, um caminho de Ti a Ti, então, por transitividade, Ti teria desbloqueado um Item de Dados antes que a própria Ti tivesse bloqueado um outro Item. Mas, pela definição de Bloqueio em Duas Fases, depois do primeiro desbloqueio, nenhum outro Bloqueio é solicitado por Ti. Portanto, o Bloqueio em Duas Fases só produz GSs acíclicos, ou seja, Logs SR.

c - Utilização de uma espécie de Replicação por Cópia Principal replicada (Capítulo II), na qual todas as réplicas atuam identicamente mas só uma, a Local à Transação, é responsável pela confirmação das solicitações de bloqueios.

d - Utilização da Réplica Local como rascunho para as ações de escrita realizadas durante a Transação e divulgação das escritas, atomicamente, ao fim da Transação. No caso da transação ser abortada, a Réplica Local será recuperada a partir das demais réplicas.

e - A iniciação de uma Estação é feita bloqueando e copiando, um a um, todos os Depósitos de Dados do BDR. Durante a cópia, as Transações da Estação em recuperação ficam inibidas. As atualizações de outras Transações das demais Estações são recebidas e efetuadas. Só ao final da cópia, as Transações Locais são ativadas.

f - Evitar Bloqueios Perpétuos através do pré-estabelecimento de Transações não conflitantes, a partir do bloqueio ordenado dos Itens de Dados. A título de ilustração, pode-se indicar a forma de demonstração de que essa estratégia evita o surgimento de Bloqueios Perpétuos.

Demonstração:

A partir dos conceitos desenvolvidos no Capítulo II, um Bloqueio Perpétuo representa um ciclo no Grafo de Alocação de Recursos.
Seja {(I0,T0),(T0,I1),(I1,T2),...,(Im,Tj),(Tj,Is),...,(Ti,I0)} um ciclo em tal grafo. A aresta (Im,Tj) significa que Tj já bloqueou para si o Item Im, e a aresta (Tj,Is) significa que Tj está tentando bloquear Is. Como todas as Transações seguem a mesma ordem de bloqueio, pode-se depreender do ciclo que: I0<I1<...<Im<Is<...I0. Isto é absurdo pois I0 é o maior e o menor elemento de um conjunto estritamente crescente. Portanto, não pode haver ciclos no Grafo de Alocação de Recursos se todas os bloqueios seguem uma mesma ordem.

g - Tornar o BDR L-resiliente, isto é, tolerante a Faltas em até L Estações. Ou seja, uma solicitação produzida por uma Transação só é tratada quando o protocolo garantir que pelo menos L+1 Estações a receberam. Neste caso, mesmo que L Estações falhem, haverá pelo menos uma que poderá processar e repetir esta solicitação às Estações que eventualmente não a tenham recebido.

h - Em caso de conflito (tentativa de uma nova Transação bloquear um Item que já está bloqueado por outra Transação), a nova Transação aguarda em uma fila, por ordem de chegada, que a Transação anterior encerre e libere o Item.

Capítulo V

Gerenciador de Banco de Dados Replicados

1. Arquitetura do Gerenciador 88

2. Gerenciador de Transações 92
   2.1 Transações 92
      2.1.1 Iniciar Transação 92
      2.1.2 Abrir Arquivo 93
      2.1.3 Bloquear Dado 95
      2.1.4 Ler Dado 96
      2.1.5 Escrever Dado 97
      2.1.6 Finalizar Transação 98

3. Gerenciador do Acesso Concorrente 98
   3.1 Solicitações ao GA 99
      3.1.1 Iniciar Transação 100
      3.1.2 Abrir Arquivo 100
      3.1.3 Bloquear Dado 101
      3.1.4 Abortar Transação 101
      3.1.5 Finalizar Transação 102
      3.1.6 Ler Dado Remoto 102
      3.1.7 Desativar Estação 103
      3.1.8 Fornecer Contexto 103

4. Falha e Recuperação de Estações 103

1. Arquitetura do Gerenciador

A arquitetura proposta para o GBDR, esquematizada na Figura V-1, assemelha-se a divisão em blocos estabelecida no Capítulo II para descrever algoritmos de controle de acesso. Isto porque, como foi então observado, as principais funções do GBDR são: Controle de Acesso Concorrente e Recuperação de Falhas. Entretanto, deve-se observar que um GBDD genérico pode apresentar uma arquitetura bem mais complexa (ver [16]).

A Figura V-1 mostra também as interações entre os módulos do GBDR que serão descritas no decorrer deste Capítulo.

A partir da divisão em módulos estabelecida no Capítulo II, pode-se estabelecer que o GBDR é constituído por:

a - Gerenciador de Transações (GT)
Existe um GT_{Ei} em cada Estação Ei que contenha um BDR. No caso de uma Estação possuir mais de um BDR, o mesmo GT_{Ei} pode conduzir Transações nos vários BDRs. A função do GT é controlar as Transações da sua Estação. Este controle implica em criar e finalizar Transações, conhecer as ações que constituem a Transação, distinguir as ações realizadas localmente das ações globalmente divulgadas, armazenar a descrição de cada ação durante uma Transação de modo a realizar a sua finalização, comunicar-se com os Gerenciadores de Dados e de Acesso (descritos a seguir) para realização das ações e comunicar-se com o Iniciador/Recuperador (descrito adiante) para restaurar os dados alterados por Transações abortadas por falha.

b - Gerenciador de Acesso aos Dados (GA)
Existe um GA_{EiBDRj} replicado em cada Estação Ei, para cada BDRj que a Estação contenha. As funções do GA são controlar globalmente as solicitações de acesso concorrente aos Dados provenientes dos GTs e comunicar-se com o Gerenciador de Dados para efetuação da finalização de cada Transação. Este controle é baseado em Bloqueio e adota uma política "o primeiro a

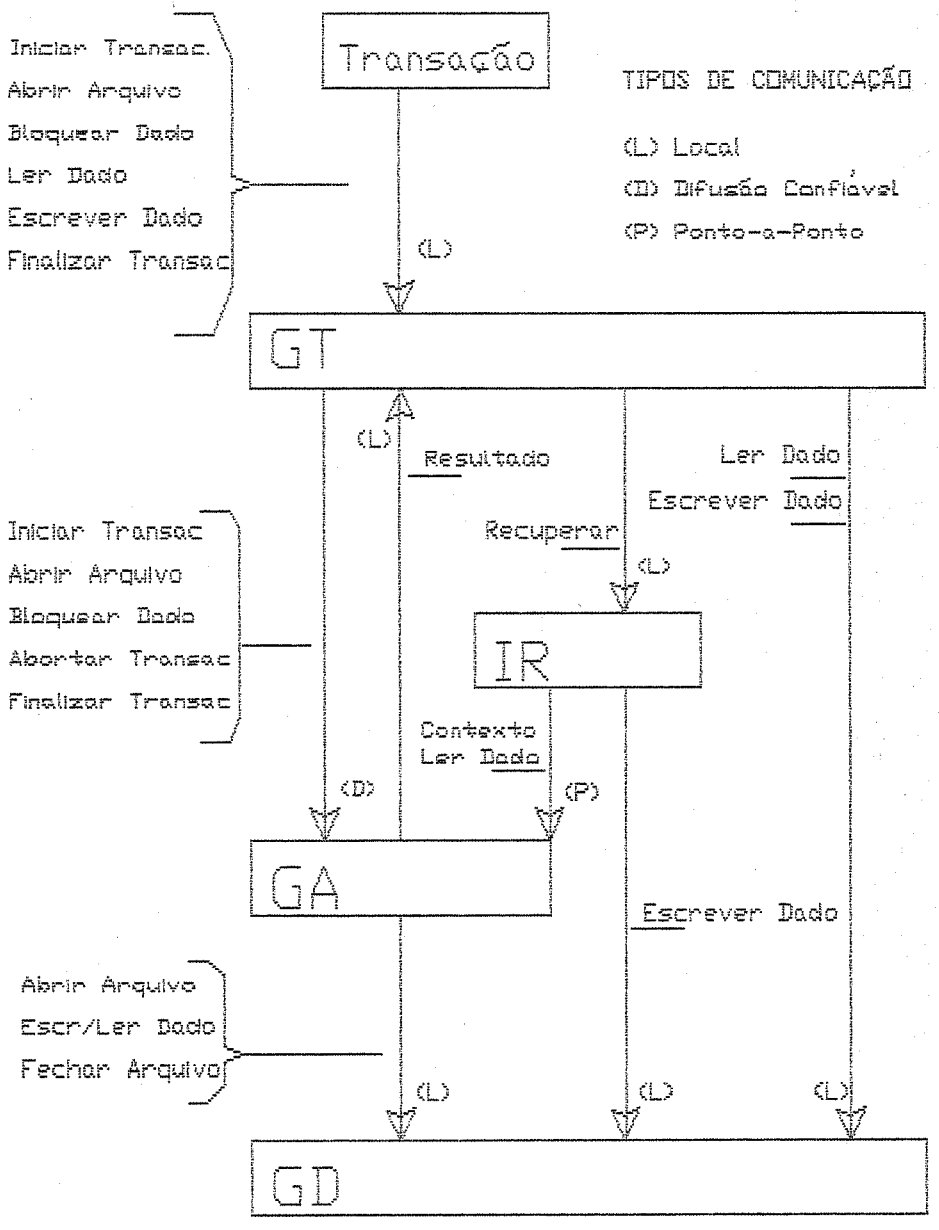


Figura V-1 : Arquitetura do GBDR

solicitar é o primeiro atendido". Para efetuar esse controle, os GAs conhecem, para cada Transação em curso, todos os Bloqueios por ela solicitados. Apenas a Réplica Local à Transação é responsável por confirmar o sucesso das solicitações de bloqueio dos GTs. Todas as réplicas dos GAs processam cada solicitação validada proveniente dos GTs. A validação das solicitações se dá quando é garantido, pelo protocolo de Difusão Confiável, que pelo menos L réplicas do GA receberam as solicitações.

c- Gerenciador de Dados (GD). Existe um GD_{Ei} em cada Estação do sistema. Os GDs, assim como os GTs, independem do número de BDRs por Estação. Deve-se notar que os GDs são módulos que existem nas Estações independentemente da existência do GBDR. As funções do GD são controlar localmente o acesso concorrente, estabelecer o mapeamento "nome lógico X armazenamento físico" dos Depósitos de Dados e efetuar as consultas e atualizações dos dados. Deve-se aqui caracterizar dois tipos de GD:
. Gerenciador de Arquivos ("File Manager") - Neste caso, os Depósitos de Dados são organizados como arquivos que possuem nomes lógicos associados. O mapeamento "nome lógico X armazenamento físico" é fornecido pelo Gerenciador através da manutenção de um Diretório. Esta é a forma clássica de organizar a memória secundária de um sistema. Os arquivos são vetores de caracteres e podem ser criados, aumentados, alterados e consultados pelas Transações.

. Gerenciador de Tabelas em Memória - Este conceito deve ser introduzido quando se pode prescindir da independência lógica "on-line", fornecida pelo conceito de arquivo, e se tem necessidade de alto desempenho. Neste caso, os Depósitos de Dados estão organizados em Tabelas, em memória real ou virtual, em posições definidas em tempo de compilação (ou "locate"). Este modelo é o mais adequado para sistemas em Tempo Real. A independência lógica existe apenas em tempo de compilação. As tabelas são pré-definidas com um tamanho máximo.

d - Iniciador/Recuperador (IR). Existe um IR_{EiBRDj} em cada Estação, Ei, e para cada BDRj do sistema. Esse módulo é

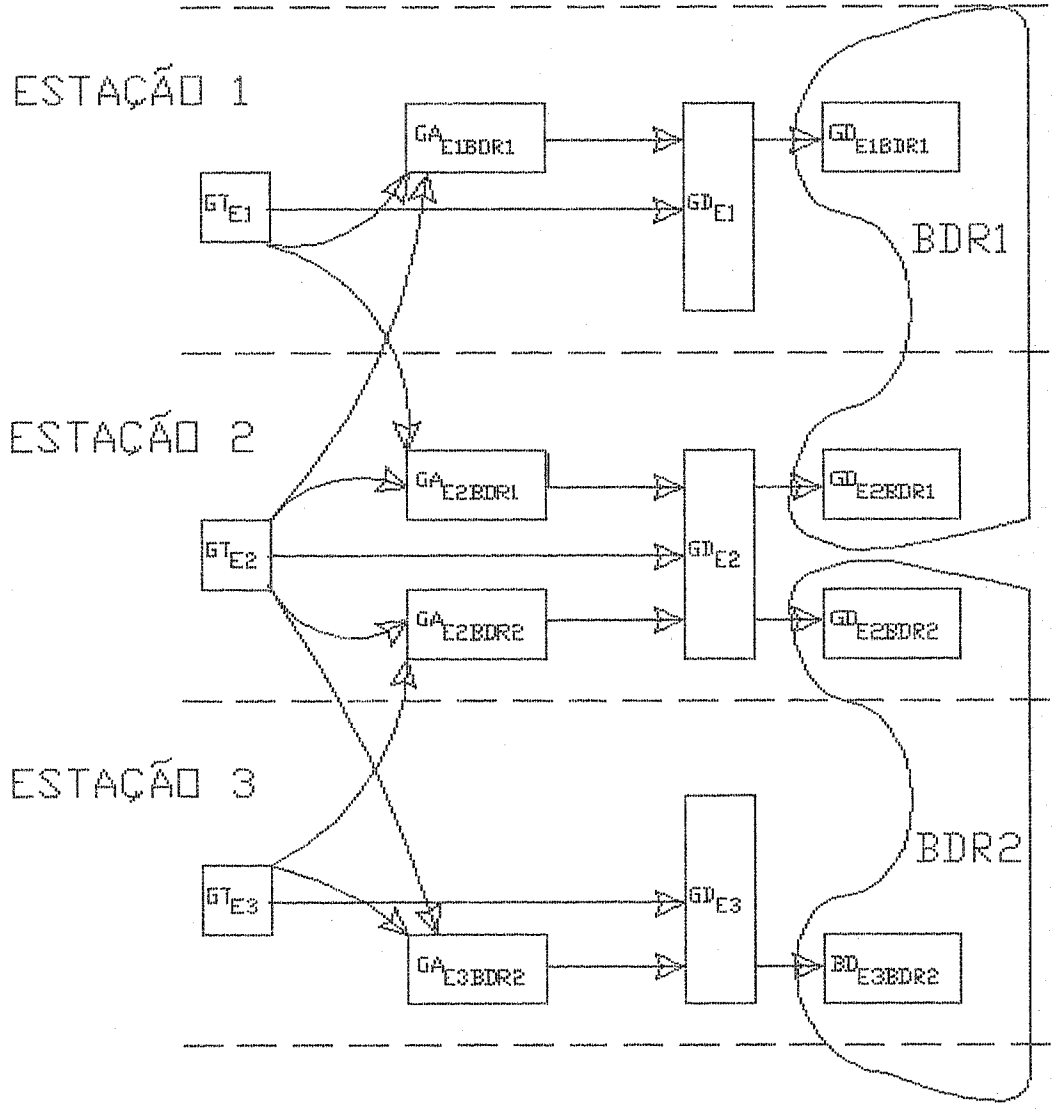


Figura V-2 : Três Estações e Dois BDRs

responsável pela iniciação de uma réplica do BDRj ou pela recuperação de uma réplica que ficou inconsistente devido a uma Transação não concluída.

Um exemplo dessa arquitetura é mostrado na Figura V-2 para 3 Estações e 2 BDRs.

2. Gerenciador de Transações

Cada Transação envia suas solicitações ao GT existente na sua Estação hospedeira.

2.1. Transações

Cada Transação é uma combinação das seguintes ações:

Iniciar Transação (i)
Abrir Arquivo (a)
Bloquear Dado (b)
Ler/Escrever Dado (l/e)
Finalizar Transação (f)

A sintaxe de uma Transação, em termos das ações que a contituem, é a seguinte: i(a(b|l|e)*)*f

2.1.1. Iniciar Transação

Essa ação identifica para o GT o início de uma nova Transação.

O parâmetro fornecido ao GT nessa ação é o nome do BDR sobre o qual se processará a transação. Se houver qualquer erro da Transação, será retornado um código de erro e a Transação deverá ser interrompida. Erros típicos são: BDR inexistente, excedido o limite de Transações simultâneas. Deve-se notar que erros como "BDR inexistente" e muitos outros, que serão mencionados no decorrer deste Capítulo, são decorrentes de um

erro na implementação de uma Transação e só devem ocorrer efetivamente durante a fase de depuração do sistema.

O processamento no GT consiste em achar o número do BDR a partir do seu nome e atribuir um identificador único à transação. Esse identificador é uma concatenação do número da Estação, Ei, com o número do BDR, BDRj, e com um número de seqüência da Transação, Tn, atribuído pelo próprio GT e incrementado a cada Transação: <Ei,BDRj,Tn>. Em caso de falha e recuperação da Estação Ei, o número de seqüência inicial do GT_{Ei} é TO. Este número apresenta a seguinte propriedade: TO atual é maior do que qualquer Tn utilizado por Ei anterior à sua falha. Um número com essa característica pode ser obtido do Carimbo de Tempo global mantido pelo Protocolo de Difusão descrito no próximo Capítulo.

Uma vez obtido o identificador da Transação, o GT divulga-o a todos os GAs com a solicitação de Início de Transação. Apenas o GA Local à Transação retornará ao GT o resultado da solicitação: Resultado. Em caso de insucesso, como por exemplo, excesso de transações simultâneas no GA, a Transação é informada e o identificador <Ei,BDRj,Tn> é abandonado.

Em caso de sucesso, o identificador <Ei,BDRj,Tn> é retornado à Transação e será usado por ela como parâmetro de todas as ações subseqüentes.

2.1.2 Abrir Arquivo

Essa ação retorna à Transação o identificador de um arquivo a partir do seu nome lógico. Isto é, realiza o mapeamento "nome lógico X armazenamento físico". Adicionalmente, essa ação permite que a Transação bloqueie o arquivo.

Os parâmetros fornecidos ao GT nessa ação são o identificador da Transação, <Ei,BDRj,Tn>, o nome do arquivo a ser aberto, Dk, e o tipo de bloqueio. O tipo de bloqueio identifica se o arquivo deve ser bloqueado para Escrita, BE, para Escrita e

Leitura, BEL, ou se não deve ser bloqueado, BN. Se houver qualquer erro da Transação na abertura do arquivo, será retornada à Transação uma indicação de que ela deverá ser interrompida. Erros típicos são: Arquivo Inexistente, Bloqueio em ordem inválida, Erro no Dispositivo Físico e Arquivo já aberto por essa mesma Transação.

O processamento do GT consiste em divulgar a todos os GAs a solicitação de abertura do arquivo: Abrir Arquivo. Apenas o GA Local à Transação retornará ao GT o resultado da solicitação: Resultado. Em caso de insucesso, isto é, ocorrência de uma falha, o GT deverá recuperar Réplica Local e então divulgar uma solicitação de interrupção da Transação a todos os GAs: Abortar Transac. Para o GT, abortar a Transação implica em restaurar o valor anterior dos Itens alterados pela Transação na Réplica Local. Como as Escritas só são divulgadas na etapa de finalização, a interrupção de uma Transação no meio não introduz nenhuma inconsistência nas demais réplicas do BDR. A identificação dos Itens alterados fica numa lista, mantida pelo GT, denominada Lista de Finalização da Transação, L_{Tn}. A lista dos Itens alterados é enviada ao módulo Iniciador/Restaurador para sua recuperação.

Em caso de sucesso, o resultado da solicitação conterá um identificador local (interno à Estação e conhecido pelo GD local) do arquivo, Dk_L, e um identificador global Dk_G. O identificador global permite referenciar o arquivo para os demais GAs para ações de Bloqueio e Escrita de Itens desse arquivo. O identificador local permite referenciar o arquivo para o GD para ações de leitura e escrita (feitas localmente). Deve-se notar que o GA local só retornará a indicação de sucesso quando o arquivo tiver sido liberado para o acesso solicitado. Enquanto este não estiver liberado, o GA não retornará resultado algum e o GT deverá manter a Transação esperando.

Há ainda o caso em que algum GA não obtem sucesso na tentativa de abertura do arquivo discrepando dos demais. Este caso será analisado no item relativo aos GAs.

2.1.3 Bloquear Dado

Essa ação permite que a Transação obtenha o Bloqueio de um Item de Dado de um Arquivo previamente aberto sem Bloqueio (BN).

Os parâmetros fornecidos ao GT nessa ação são o identificador da Transação, <Ei,BDRj,Tn>, o identificador global do arquivo, Dkα , e o Item de Dado a bloquear, Im. Se houver qualquer erro da Transação no bloqueio do Dado, será retornado um código do erro e a Transação deverá ser interrompida. Os erros típicos são: Transação inexistente, Bloqueio em ordem inválida e Dado já bloqueado por essa Transação.

As interações entre as ações Bloquear Dado e Abrir Arquivo são apresentadas na tabela seguinte. A tabela mostra os estados assumidos por um arquivo ou Item de Dado (linhas), as solicitações de bloqueio possíveis (colunas) provenientes de Transações concorrentes e o próximo estado assumido pelo arquivo ou Item de Dado (posições da tabela):

ação--> / estado V	Abrir s/ Bloq (BN)	Abrir c/ Bloq Escrita (BE)	Abrir c/ Bloq Esc/Lei (BEL)	Bloq Dado 1	Bloq Dado 2
Arquivo não Aberto	Arq não Bloq	Arq Bloq p/ Escrita	Arq Bloq p/ Esc/Lei	-	-
Arquivo não Bloq	Arq não Bloq	C	C	Dado 1 Bloq	Dado 2 Bloq
Arquivo Bloq p/ Escrita	C	Arq Bloq p/ Escrita	C	-	-
Arquivo Bloq p/ Esc/Lei	C	C	C	-	-
Dado 1 Bloq	Dado 1 Bloq	C	C	C	Dados 1 e 2 Bloq

"C": conflito

"-": não se aplica

Ação de Bloquear Dado só pode ser usada em arquivos abertos s/ bloqueio.

O processamento do GT consiste em divulgar a todos os GAs a solicitação de bloqueio: Bloquear Dado. Apenas o GA local à Transação retornará ao GT o resultado da solicitação: Resultado. Em caso de insucesso, isto é, ocorrência de uma falha, o GT deverá divulgar uma solicitação de interrupção da Transação a todos os GAs: Abortar Transac. Para o GT, abortar a Transação implica em restaurar o valor anterior dos Itens alterados pela Transação na Réplica Local.

Deve-se notar que o GA local só retorna a indicação de sucesso quando o Item Im houver sido liberado pela Transação anterior e bloqueado para Tn. Enquanto este não estiver liberado, o GA não retornará resultado algum e o GT deverá manter a Transação, Tn, esperando.

2.1.4 Ler Dado

Essa ação permite que a Transação consulte um Item de Dado.

Os parâmetros fornecidos ao GT nessa ação são o identificador da Transação, <Ei,BDRj,Tn>, o identificador global do arquivo, Dk_G , e o Item de Dado a ser lido, Im. Se houver qualquer erro da Transação na leitura do Dado, será retornado um código do erro e a Transação deverá ser interrompida. Os erros típicos são: Transação inexistente, Arquivo inexistente e Erro no Dispositivo Físico.

O processamento do GT consiste em solicitar ao GD local a leitura do Item Im: Ler Dado. Isto é feito mediante a utilização do identificador local do arquivo Dk_L. Em caso de insucesso, isto é, ocorrência de uma falha, o GT deverá divulgar uma solicitação de interrupção da Transação a todos os GAs: Abortar Transac. Para o GT, abortar a Transação implicará em restaurar o valor anterior dos Itens alterados pela Transação na Réplica Local.

Em caso de sucesso, o GT retorna à Transação o Item Im solicitado.

2.1.5 Escrever Dado

Essa ação permite que a Transação atualize um Item de Dado.

Os parâmetros fornecidos ao GT nessa ação são o identificador da Transação, <Ei,BDRj,Tn>, o identificador global do arquivo, Dk_G , e o Item de Dado a ser escrito, Im. Se houver qualquer erro da Transação na escrita do Dado, será retornado um código do erro e a Transação deverá ser interrompida. Os erros típicos são: Transação inexistente, Arquivo inexistente e Erro no Dispositivo Físico.

O processamento do GT consiste em solicitar ao GD local a escrita do Item Im: Escrever Dado. Isto é feito mediante a utilização do identificador local do arquivo Dk_L. Em caso de insucesso, isto é, ocorrência de uma falha, o GT deverá divulgar uma solicitação de interrupção da Transação a todos os GAs: Abortar Transac. Para o GT, abortar a Transação implica em restaurar o valor anterior dos Itens alterados pela Transação na Réplica Local.

Em caso de sucesso, o GT cria uma célula de informação que é inserida na Lista de Finalização da Transação, L_{Tn}. Nessa célula é colocada a informação de que o Item Im, do arquivo Dk_G, foi atualizado localmente e portanto deve ser divulgado na finalização da Transação. Verifica-se que um requisito para o protocolo de Difusão é que esta possa ser feita em pacotes. Assim, mesmo Itens que constituem todo um arquivo podem ser divulgados em pacotes.

2.1.6 Finalizar Transação

Essa ação informa ao GT a conclusão da Transação e a necessidade da divulgação e Comissionamento das atualizações por todos os GAs.

O parâmetro fornecido ao GT nessa ação é o identificador da Transação, <Ei,BDRj,Tn>. Se houver qualquer erro da Transação na finalização, será retornado um código do erro e a Transação deverá ser interrompida. Um erro típico é: Transação inexistente.

O processamento do GT consiste em processar a Lista de Finalização da Transação, L_{Tn}. Esse processamento gera uma única mensagem, constituída por vários pacotes e que deve ser divulgada atomicamente a todos os GAs. São processadas as células produzidas pelas Escritas na ordem em que foram realizadas. Esse processamento implica em ler o Item atualizado e, para cada célula, divulgar um pacote que contenha as mesmas informações da célula e o conteúdo do Item. No caso de mais de uma escrita sobre o mesmo Item, o GT otimiza e naturalmente divulga apenas o pacote correspondente à última atualização. Ao ser confirmada a Divulgação da mensagem, o GT retorna um código de fim de Transação.

Em caso de falha durante a formação dos pacotes (Erro no Dispositivo Físico), o GT deverá divulgar uma solicitação de interrupção da Transação a todos os GAs: Abortar Transac. Para o GT, abortar a Transação implica em restaurar o valor anterior dos Itens alterados pela Transação na Réplica Local.

3. Gerenciador do Acesso Concorrente

Cada GT envia solicitações por Difusão aos GAs replicados nas Estações. Todos os GAs processam as solicitações, mas apenas o GA local à Transação retorna o Resultado. Cada solicitação recebida por Difusão é rotulada pelo Protocolo de Difusão com um número de ordem, CT (Carimbo de Tempo). Uma solicitação

produz alterações nas variáveis que constituem o contexto do GA. Entretanto, como cada GA recebe as solicitações, provenientes do GT, na mesma ordem, seus contextos se mantêm consistentes. Além disso, é possível associar às versões dos contextos o número de ordem, CT, das solicitações que o produziram. Tal associação é utilizada na iniciação de um GA.

Um tipo de solicitação enviado pelos IRs é a Leitura Remota para recuperação dos itens de dados. Esta é uma comunicação ponto-a-ponto entre um IR e algum GA remoto. Além disso, o GA deve ser informado pelo Protocolo de Difusão Confiável quando qualquer Estação tornar-se inativa. Finalmente, quando uma Estação é reativada, o IR dessa Estação deve enviar a algum GA remoto uma solicitação do seu contexto através de uma comunicação ponto-a-ponto.

3.1 Solicitações ao GA

As solicitações ao GA são:

Iniciar Transação (i) , proveniente do GT;
Abrir Arquivo (a) , proveniente do GT;
Bloquear Dado (b) , proveniente do GT;
Abortar Transação (x) , proveniente do GT;
Finalizar Transação (f) , proveniente do GT;
Ler Dado Remoto (l) , proveniente do IR;
Desativar Estação , proveniente do protocolo de Difusão;
Fornecer Contexto , proveniente do IR.

A ordem das solicitações ao GA, originadas no GT, devido a uma mesma Transação, é a seguinte: i(a(l|b)*)*(f|x). O controle dessa ordem é efetuado globalmente pelo GA. Isto é, a tentativa de manipular um item de dado fora de uma Transação e um Item de um arquivo que não esteja aberto são erros detectados pelo GA.

Desativações de Estações são solicitações que podem ocorrer em qualquer instante, informam que determinadas Estações foram desativadas e provocam o aborto das Transações que estavam em curso nas Estações desativadas.

A solicitação do contexto de um GA também pode ocorrer em qualquer instante em função da reativação de uma Estação.

3.1.1 Iniciar Transação

Essa solicitação identifica para o GA o início de uma nova Transação em uma determinada Estação.

O parâmetro fornecido ao GA nessa solicitação é o identificador da Transação <Ei,BDRj,Tn>. O GA local ao GT solicitante retorna uma confirmação da aceitação de uma nova Transação. Em caso de erro, o GA retorna uma indicação de insucesso ao GT. Um erro típico é: esgotado o limite máximo de Transações simultâneas no GA.

O processamento no GA consiste apenas em armazenar o identificador da Transação com a condição de Transação iniciada.

3.1.2 Abrir Arquivo

Os parâmetros fornecidos ao GA nessa solicitação são o identificador da Transação, <Ei,BDRj,Tn>, o nome do arquivo a ser aberto, Dk, e o tipo de bloqueio. É retornado ao GT o resultado da solicitação.

O processamento no GA consiste em verificar se a Transação existe e se o tipo de acesso solicitado não produz conflito com o estado atual do arquivo. Os tipos de conflito foram relacionados na descrição do GT. Em caso de conflito, isto é, o Arquivo está bloqueado por outra Transação, esta Transação é colocada numa fila de espera do Arquivo e só será reativada quando o Arquivo for liberado. O GA verifica também se está correta a ordem de abertura do Arquivo, considerando os demais Arquivos abertos pela Transação e a ordenação pré-estabelecida de Arquivos no Banco de Dados. No caso de não ocorrer conflito,

GA deve então repassar a solicitação de abertura ao GD que retorna um identificador local do arquivo, Dk_L. Em caso de falha, o GD pode também retornar indicações como: Arquivo Inexistente e Falha no Disposivo de Armazenamento. Se, por alguma falha no dispositivo físico de armazenamento, não for possível realizar a abertura do arquivo, a Estação que contém esta réplica do BDR será desativada.

Finalmente, o GA obtém um identificador global para o arquivo, Dk_G. Os identificadores são retornados ao GT.

3.1.3 Bloquear Dado

Os parâmetros fornecidos ao GA nessa solicitação são o identificador da Transação, <Ei,BDRj,Tn>, o identificador global do arquivo a ser aberto, Dk_G, e o Item de Dado a bloquear, Im. É retornado ao GT o resultado da solicitação.

O processamento no GA consiste em verificar se o bloqueio solicitado não produz conflito com o estado atual do Item de Dado. Os tipos de conflito foram relacionados na descrição do GT. O GA verifica também se está correta a ordem do bloqueio do Item, considerando os demais Itens bloqueados pela Transação e a ordenação pré-estabelecida de Itens no Banco de Dados. Em caso de conflito, isto é, o Item de Dado está bloqueado por outra Transação, esta Transação é colocada numa fila de espera do Arquivo e só será reativada quando o Arquivo for liberado.

3.1.4 Abortar Transação

Esta solicitação permite ao GT abortar uma Transação em curso nos GAs.
O parâmetro fornecido nesta solicitação é o identificador da Transação, <Ei, BDRj, Tn>. O processamento no GA consiste em liberar todos os bloqueios e solicitar ao GD o fechamento de todos os arquivos abertos pela Transação. O resultado é retornado ao GT pelo GA local.

3.1.5 Finalizar Transação

Esta solicitação permite ao GT enviar a todos os GAs a lista de atualizações produzidas pela Transação e encerrá-la. Os parâmetros fornecidos ao GA são o identificador da Transação, <Ei,BDRj,Tn>, e a lista de Itens alterados com seus novos valores, L_{Tn}.

O processamento nos GAs consiste em verificar se a Transação existe e a partir daí processar cada uma das células que constituem a L_{Tn}. Esse processamento consiste em verificar se a atualização solicitada não produz nenhum conflito. Em seguida, é obtido o identificador local do arquivo, Dk_L, e a atualização é solicitada ao GD. Se, por alguma falha no dispositivo físico de armazenamento, não for possível realizar a atualização, a Estação que contém esta réplica do BDR será desativada.

3.1.6 Ler Dado Remoto

Essa solicitação permite que o IR de alguma Estação obtenha o valor anterior de um Item de Dado, modificado por uma Transação em curso, com o fim de restaurá-lo. Os parâmetros fornecidos ao GA nessa solicitação são o identificador da Transação, <Ei,BDRj,Tn>, o identificador global do arquivo, Dk_G, e o Item a ser lido, Im.

O processamento efetuado pelo GA é verificar se a Transação existe, se o arquivo está aberto e se o acesso a este Item não produz nenhum conflito. A seguir, obter o identificador local do arquivo, Dk_L, e solicitar ao GD a Leitura do Item. O Item lido e o resultado da solicitação são retornados ao IR.

3.1.7 Desativar Estação

Inicialmente, deve-se notar que o conjunto de Estações que contém as réplicas de um determinado BDR constitui também, para o protocolo de Difusão Confiável, um Grupo de Difusão.

Esta solicitação é produzida pelo próprio protocolo de Difusão Confiável cada vez que alguma Estação de um Grupo de Difusão é desativada. Os GAs de todos os BDRs baseados em tal Grupo de Difusão recebem a solicitação.

O processamento nos GAs consiste em abortar todas as Transações em curso na Estação desativada e liberar todos os Itens por ela bloqueados.

3.1.8 Fornecer Contexto

Essa solicitação permite que o IR obtenha o contexto atual dos GAs e inicie o seu GA local.

Essa solicitação não possui parâmetros. O processamento no GA consiste em produzir uma cópia de todas as suas variáveis de controle que constituem seu contexto, anexar o Carimbo de Tempo, CT, da última solicitação processada, e enviar esse conjunto ao IR solicitante. O Carimbo de Tempo é um número fornecido pelo próprio protocolo de Difusão Confiável juntamente com as solicitações.

4. Falha e Recuperação de Estações

Como já foi comentado, quando uma Estação Falha, o protocolo de Difusão Confiável detecta e avisa aos GAs. Estes, então, abortam todas as Transações em curso naquela Estação. Quando uma Estação, Ei, se recupera de uma Falha e é reativada, quem comanda a reativação do seu GBDR é o módulo IR. Esta reativação é feita independentemente para cada BDR. Existe um módulo IR_{EiBDRj} para cada BDRj do sistema.

O algoritmo de iniciação conduzido pelo IR é o seguinte:
a - O IR obtém do protocolo de Difusão as Estações ativas no Grupo de Difusão correspondente ao BDR.

b - O IR escolhe o GA de alguma das Estações ativas, solicita seu contexto e inicia o contexto do seu GA local. A partir daí, o GA local ao IR já pode realizar normalmente as solicitações recebidas por difusão, exceto a de Ler Dados. Se a Estação do IR recém-ativado é a única Estação ativa e se o repositório associado ao BDR está marcado como consistente desde a geração "off-line", então a recuperação é considerada terminada.

c - O IR então obtém, do repositório de Dados associado ao BDR, o nome de todos os arquivos que constituem o BDR. Deve-se notar que nesta proposta considera-se que são pré-determinados, inicializados "off-line", os arquivos que constituem um BDR.

d - O IR então inicia Transações, nas quais ele bloqueia, um por um, todos os arquivos do BDR, e solicita, a algum dos GAs, o conteúdo desses arquivos através da solicitação Ler Dado Remoto.

e - Quando todo o BDR estiver recuperado o GT local será habilitado e o GA local será liberado inclusive para a solicitação de Ler Dados.

A estrutura de dados básica do IR é um conjunto de informações sobre o BDR, incluindo o identificador do repositório de informações associado.

Capítulo VI

Protocolo de Difusão Confiável

1. Introdução 106

2. Seleção do Protocolo de Difusão Confiável 107

3. O Algoritmo de CHANG e MAXEMCHUK - 84 107
   3.1 Funcionamento da Fase Normal 110
   3.2 Funcionamento da Fase de Reforma 115

4. Vantagens e Problemas 118

5. Alterações Introduzidas 119

1. Introdução

As propriedades que caracterizam a Difusão Confiável, descritas no Capítulo II, são resultados tão fortes que tornam o Protocolo um elemento fundamental no algoritmo de Controle de Acesso Concorrente e Recuperação de Falhas do Gerenciador de Banco de Dados Replicados descrito no Capítulo V. Isto implica em uma cuidadosa seleção e modelagem do Protocolo a ser utilizado pelo GBDR. Este capítulo apresenta ponderações feitas na pesquisa de um Protocolo adequado, uma descrição do Protocolo selecionado e soluções para os problemas existentes no Protocolo selecionado. No Apêndice, é apresentado um modelo do protocolo baseado em Redes de Petri.

A partir da descrição do GBDR realizada no Capítulo V, pode-se relacionar os seguintes requisitos para o Protocolo de Difusão Confiável:

a - Deve implementar a Difusão Confiável para um determinado Grupo de Difusão. Isto é, deve garantir a atomicidade das difusões, baseando-se numa Rede que ofereça apenas o serviço de difusão física com possibilidade de Faltas por Omissão.

b - Deve apresentar Independência do Ambiente. Isto é, o protocolo deve prescindir de quaisquer serviços ou sistemas auxiliares que não o descrito no item a.

c - Deve fornecer o serviço de detecção de modificações na configuração de Estações ativas no Grupo de Difusão.

d - Deve fornecer o serviço de detecção de seccionamento da Rede a um nível que não permita a formação de um Grupo de Difusão.

e - Deve fornecer o serviço de consulta a um Carimbo de Tempo global ao Grupo de Difusão.

2. Seleção do Protocolo de Difusão Confiável

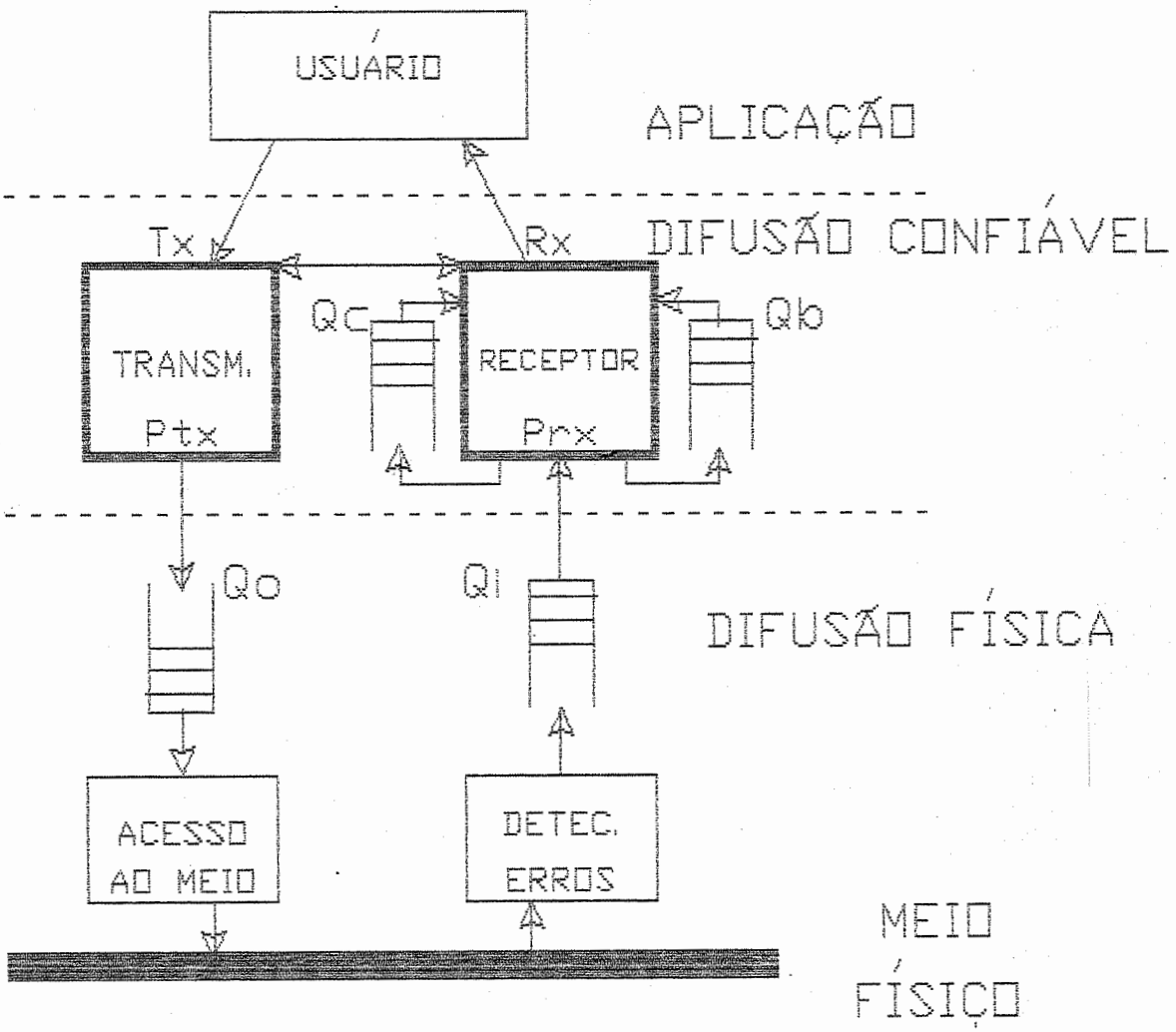
Dentre os Protocolos de Difusão Confiável examinados (CHANG et al [13], SEGALL et al [41], BABAOGLU et alii [6], AWERBUCH et al [5], CRISTIAN et alii [17], SCHNEIDER et allii [40]), o Protocolo desenvolvido por CHANG e MAXEMCHUK em 1984 revelou características bastante interessantes em vista dos requisitos apresentados neste trabalho. As vantagens e problemas apresentados por essa alternativa serão relacionados num próximo item. Nossa alternativa foi, a partir do trabalho de CHANG e MAXEMCHUK, elaborar algumas alterações e propor uma variação daquele Protocolo.

3. O Algoritmo de CHANG e MAXEMCHUK - 84.

O objetivo desse item é explicar o funcionamento desse Protocolo através da nomenclatura utilizada na modelagem que será feita no Apêndice. O protocolo apresentado aqui é a versão original proposta [13]. Os problemas encontrados nessa versão bem como as soluções adotadas serão discutidos no próximo item.

O Protocolo de Difusão Confiável (DC) deve operar entre uma camada de programas usuários do protocolo, no presente trabalho um GBDR, e uma camada de programas de recepção e transmissão em difusão física. Isto é representado na Figura VI-1.

Em qualquer Estação do Grupo de Difusão, identificada por um endereço único na Rede, o usuário transmite mensagens via DC através da interface Tx. Todas as Estações do Grupo as recebem. Os usuários recebem as mensagens através da interface Rx. Um determinado usuário pode confirmar a terminação de sua difusão através da recepção da mensagem que ele mesmo transmitiu. A ordem das recepções é aleatória em relação à ordem das difusões. Entretanto, o DC garante que, qualquer que seja a ordem, ela será a mesma em todas as Estações.



Tx, Rx - Interface c/ a Difusão Confiável

Ptx, Prx - Interface c/ a Difusão Física

Qo - Fila de Transmissão

Qi - Fila de Recepção

Qb - Depósito de Mensagens de Dados

Qc - Fila de Controle das Confirmações

Figura VI-1 : Protocolo de Difusão Confiável

Mensagens a serem transmitidas são colocadas na fila Q_o para difusão física. O programa receptor das difusões físicas descarta as mensagens com erro de transmissão. Uma mensagem também pode ser descartada por falta de espaço em Q_i. É responsabilidade do DC retransmitir as mensagens descartadas e reordená-las na recepção.

A justificativa para a estratégia utilizada no DC pode ser resumida como se segue. Quando se trata de um único Transmissor num sistema de comunicação, a ordenação das mensagens nos Receptores é trivial: o Transmissor associa um número de seqüência às transmissões. É desnecessário que os Receptores confirmem explicitamente a recepção das mensagens, pois a chegada de um número de seqüência maior que o esperado denuncia a perda de mensagens e provoca a solicitação de retransmissão. Fatalmente os Receptores obtêm todas as mensagens na ordem correta. Um sistema com múltiplos Transmissores pode se assemelhar a um sistema com um único Transmissor pela "passagem" de todas as mensagens por um Receptor "principal" (Cf. replicação via Cópia Principal descrita no item 3.3.6 do Capítulo II). Apenas tal Receptor é responsável por confirmar, também por difusão, a recepção da mensagem através da associação de um número de seqüência, ou Carimbo de Tempo, à mensagem. Assim fica resolvido o problema da confirmação num sistema com múltiplos Transmissores. Entretanto, existem ainda algumas deficiências:

- Não há como saber quando todos os Receptores obtiveram todas as mensagens. Portanto, as mensagens já transmitidas teriam que ser conservadas para sempre de modo a atender a uma eventual solicitação de retransmissão.
- O trabalho de confirmação e retransmissão sobrecarrega uma única Estação ao invés de ser distribuído.

Essas deficiências podem ser eliminadas se se fizer circular entre as estações do Grupo de Difusão o privilégio, materializado numa Ficha, de ser o Receptor "principal". O

Receptor "principal" é denominado Estação com Ficha. Além disso, é necessário que antes de aceitar a Ficha, uma Estação esteja com sua fila de mensagens recebidas completa segundo os números de seqüência. E, ainda, que qualquer Receptor só transfira uma mensagem recebida para o usuário quando verificar que pelo menos L+1 Estações a receberam também. Isto torna o DC L-resiliente, ou seja, mesmo na presença de falhas em L Estações, o DC conserva suas propriedades pois persevera ainda uma Estação com sua fila de mensagens completa. O funcionamento do protocolo se dá em duas fases que serão detalhadas a seguir: Fase Normal e Fase de Reforma.

3.1 Funcionamento da Fase Normal

O DC, na sua fase normal, pode ser decomposto em três partes: Transmissor, Receptor da Estação sem Ficha e Receptor da Estação com Ficha.

O DC assume que o Grupo de Difusão possui um número fixo n, pré-determinado de Estações, cada qual com um identificador Ei único. Estas Estações podem falhar e ser reativadas ao longo do tempo.

Cada Estação, Ei, mantém as seguintes informações:

Ver : É a versão atual do Grupo de Difusão. Toda vez que é detectada a falha ou reativação de uma Estação, o DC entra na Fase de Reforma do Grupo de Difusão para a redefinição das Estações ativas. O Grupo redefinido recebe um número de versão. Durante a Fase Normal, a Estação assume a versão do Grupo de Difusão fornecido pela última Reforma. A todas as mensagens é acrescentada a informação de versão. Se, por alguma Falta, for recebida uma mensagem com versão discrepante, uma nova Reforma é iniciada.

M[] : É um vetor que contém, para cada Estação, o próximo número de ordem das mensagens transmitidas por difusão. Cada Transmissor mantém o seu número de ordem, M[Ei], que é apenas um número de seqüência, interno a cada Estação pertencente ao

Grupo de Difusão. A cada mensagem transmitida esse número é agregado à mensagem. Os Receptores, a partir das mensagens recebidas, vão incrementando posições do vetor M e podem detectar perda de mensagens e mensagens repetidas. O vetor M possui n posições. As posições relativas às Estações em Falha não são utilizadas.

PCT : É o próximo Carimbo de Tempo a ser utilizado pela Estação com Ficha na confirmação de uma nova mensagem. A mensagem de confirmação associa PCT, Ei e M[Ei]. Com isso, toda Estação, ao receber a confirmação, pode verificar se perdeu alguma mensagem ou confirmação anterior.

Quando a Fase Normal é iniciada, todas as Estações possuem os mesmos valores em Ver, M[] e PCT.

A difusão de uma mensagem passa por três etapas: Transmissão, Confirmação e Repasse para o Usuário.

a - Transmissão: o Transmissor do DC, ao receber uma mensagem de Dados de um usuário, via Tx, agrega as informações Ver, Ei e M[Ei], e retransmite periodicamente (este período é determinado por um temporizador denominado Temp3) a mensagem, via Ptx, até que o Receptor da Estação com Ficha confirme sua recepção. Ou então, que um número máximo de R tentativas se esgote. Neste caso é assumido que a Estação com Ficha falhou e uma nova Reforma é provocada.
Deve-se notar que Ei e M[Ei] constituem um identificador único para a mensagem, pois M[Ei] só será incrementado pelo Receptor quando for recebida a confirmação da mensagem.

b - Confirmação: o Receptor da Estação com Ficha, ao receber, via Prx, uma nova mensagem de Dados com o número de ordem igual a M[Ei], assume que esta ainda não foi confirmada. Então inicia o processo de confirmação e armazena a nova mensagem numa fila de mensagens não confirmadas denominada Qb. O Receptor da Estação sem Ficha, ao receber uma mensagem com número de ordem igual a M[Ei], também armazena-a em Qb. No caso de o número de ordem, existente na mensagem, ser inferior a M[Ei], a mensagem

será considerada repetida e o Receptor da Estação com Ficha assumirá que o Transmissor perdeu a confirmação e, portanto, repetirá a confirmação.

O processo de confirmação consiste na difusão de uma mensagem, Ack-Dados, com as seguintes informações: PCT, Ei e M[Ei]. Esta mensagem tem o propósito triplo de: confirmar ao Transmissor e aos demais Receptores a recepção da mensagem de Dados; confirmar à Estação com Ficha anterior que a presente Estação recebeu a Ficha; e, finalmente, passar a Ficha à próxima Estação. Este último propósito leva a Estação com Ficha a repetir periodicamente (Temp2) o Ack-Dados até que a próxima Estação com Ficha confirme a recepção da mesma. Se, após R tentativas, a passagem da Ficha não for confirmada, será presumido que a próxima Estação estará em falha e uma nova Reforma será iniciada .

A ordem da passagem de Fichas, ou seja, o anel lógico formado pelas Estações ativas do Grupo de Difusão, é um dado fornecido pela Reforma. Assim sendo, passar a Ficha significa que cada Estação deve, ao receber um Ack-Dados, verificar se é a próxima no anel lógico. Se for, deverá verificar se lhe falta alguma mensagem de Dados ou Ack-Dados até o valor do PCT contido no Ack-Dados mais recentemente recebido. Tal verificação será feita a partir da fila de controle Qc, que armazena todas as confirmações recebidas bem como a indicação da ausência das mensagens de Dados correspondentes. Se faltar, deverá realizar o processo de recuperação de tais mensagens e, finalmente, assumir a condição de Estação com Ficha. A confirmação para a Estação com Ficha anterior da passagem da mesma pode dar-se de três modos: um Ack-Dados da próxima Estação com Ficha, como já foi mencionado; um Ack-Nulo; e uma simples Confirmação. A ocorrência desses dois últimos modos será explicada mais adiante.

Em seguida, todos os Receptores recebem a mensagem de confirmação, Ack-Dados. Se o Carimbo de Tempo em Ack-Dados for igual ao PCT e a mensagem de Dados correspondente estiver em

Qb, esta será extraída de Qb, e será inserida juntamente com o Ack-Dados, marcado com o estado de Pronto-com-Mensagem em uma fila de controle, Qc, ordenada pelo PCT contido no Ack-Dados. Além disso, M[Ei] e PCT são incrementados. Finalmente, é disparado o processo de Repasse aos Usuários das mensagens prontas e ordenadas.

Se o Carimbo de Tempo em Ack-Dados for igual ao PCT corrente da Estação, mas a mensagem correspondente, com os mesmos Ei e M[Ei], não se encontrar em Qb, Ack-Dados será colocada em Qc marcada como Falta-Msg e será disparado o processo de recuperação da mensagem de Dados.

Se o Carimbo de Tempo em Ack-Dados for menor que o PCT corrente da Estação, significará que esta é uma confirmação velha que deverá ser descartada. Se o Carimbo de Tempo for maior que PCT, significará que outras confirmações foram perdidas. Neste caso, o Ack-Dados será inserido em Qc marcado como Falta-Msg e será disparado o processo de recuperação das mensagens de confirmação.

c - Repasse para o Usuário: só podem ser repassadas ao usuário as mensagens na ordem determinada pelas confirmações e que não violem a L-resiliência do DC. Como Qc é ordenada por PCT, está sempre no topo de Qc a mensagem candidata a ser repassada ao usuário. A mensagem poderá ser retirada de Qc e repassada ao Usuário sempre que, na fila Qc de qualquer Estação Ei, houver uma diferença maior ou igual a L unidades entre o topo de Qc (par mensagem de Dados - Ack-Dados) e o Ack-Dados mais recentemente recebido. Esse repasse de uma mensagem aos usuários é denominado Comissionamento da mensagem. Isto confirma que pelo menos (L + 1) Estações também receberam a mensagem de Dados do topo da Lista. Isto porque, cada vez que um Ack-Dados é gerado, também a Ficha é passada à próxima Estação. Como uma Estação só pode receber a Ficha quando está com a fila Qc completa, pelo menos L Estações receberam o atual topo de Qc.

O tríplice uso do Ack-Dados só é possível em situações em que haja um tráfego intenso e constante de modo que a confirmação da próxima mensagem de Dados seja usada também para a passagem da Ficha. Entretanto, se a próxima mensagem de Dados demorar mais do que um período Temp4, desde a recepção da Ficha, e existir, no topo da fila Qc, uma mensagem de Dados aguardando as L passagens de Ficha para que tal mensagem possa ser repassada ao usuário, então ocorrerá o seguinte: um tipo de mensagem, denominado Ack-nulo, é divulgado pela atual Estação com Ficha contendo apenas o PCT da atual Estação com Ficha. Esta mensagem tem o propósito duplo de: confirmar à Estação com Ficha anterior que a presente Estação recebeu a Ficha; e de passar a Ficha à próxima Estação. Este último propósito leva a Estação com Ficha a repetir periodicamente (Temp2) o Ack-Nulo até que a próxima Estação com Ficha confirme a recepção da mesma. Se, após R tentativas, a passagem da Ficha não for confirmada, será presumido que a próxima Estação está em falha e uma nova Reforma será iniciada .

Deve-se notar que os Ack-nulos são inseridos em Qc pelos Receptores da mesma forma que os Ack-Dados, só que marcados como Prontos-sem-Mensagem e, ao atingirem o topo de Qc, são removidos sem repassarem qualquer mensagem aos usuários.

Se não chegar qualquer mensagem de Dados e também não houver mais mensagens prontas em Qc, isto é, só restarem Ack-nulos em Qc, um terceiro tipo de mensagem, a Confirmação, será divulgada, pela atual Estação com Ficha, com o propósito de confirmar à Estação com Ficha anterior que a mesma foi recebida. A Ficha permanece na Estação atual até a chegada de uma nova mensagem de Dados.

Deve-se observar que o processo de solicitação de retransmissão de Dados (Ack-Dados), uma vez disparado, divulga periodicamente, a cada Temp1, a mensagem Req-Dados (Req-Ack), contendo apenas o PCT correspondente, a qual deve ser atendida pela atual Estação com Ficha e/ou pela Estação que está tentando passar a Ficha. Isto é, até que uma Estação com Ficha receba a confirmação da passagem da mesma, é responsável por

responder às solicitações de retransmissão. Portanto, em determinados instantes, duas Estações poderão responder a estas solicitações.

3.2 Funcionamento da Fase de Reforma

Uma falha de comunicação ocorre quando uma Estação do Grupo de Difusão não obtém confirmação, após R retransmissões, de que outra Estação recebeu a mensagem enviada. O parâmetro R é feito grande o suficiente para tornar quase certo que se ocorreu uma falha de comunicação, a outra Estação está em Falha. No DC, como a Ficha é circulada entre as Estações ativas do Grupo de Difusão, em n passagens de Ficha, no máximo, serão detectadas as Estações em Falha. Sempre que uma falha ou reativação de Estação é detectada, o DC entra em Fase de Reforma para redefinir as Estações ativas no Grupo de Difusão.

Inicialmente, deve-se notar que a Reforma é essencialmente um algoritmo de Exclusão Mútua, baseado em Votação e Carimbo de Tempo, para permitir a seleção de uma Estação que irá reorganizar o DC em função de uma mudança no conjunto de Estações ativas.

Para preservar a correção do esquema de passagem de Ficha do DC, a Reforma deve garantir que:
a - a antiga configuração (Estações ativas) do Grupo de Difusão, identificada pela versão Ver, não poderá ser utilizada para o Comissionamento de nenhuma nova mensagem;
b - apenas uma nova configuração do Grupo de Difusão deve surgir como produto da Reforma.

Para satisfazer a primeira condição, a nova versão do Grupo de Difusão deve conter, como Estação ativa, pelo menos, uma das L Estações seguintes à última Estação com Ficha da versão anterior. Como o sistema não pode Comissionar mensagens sem que a Ficha seja transferida entre as L Estações seguintes à Estação com Ficha, qualquer Estação que por falta (seccionamento da Rede) não esteja a par do processo de Reforma

não poderá Comissionar novas mensagens pela versão anterior. A verificação desta condição no DC é denominada Teste de Resiliência.

Para atender à segunda condição, uma Estação só pode aderir a uma única nova versão, isto é, ser considerada Estação ativa de uma única nova versão do Grupo de Difusão. E tal versão só poderá ser aprovada pelo Teste de Maioria, se contiver a maioria das Estações do Grupo. Assim, a nova versão é única. O modo de garantir que uma versão é univocamente identificada pelo seu número de versão, Ver, é formar este número pela concatenação de um número de seqüência das versões, mantido pelas próprias Estações, com o identificador da Estação: <Seq//Ei>. Uma Estação só pode aderir a uma versão maior que a atual.

A Reforma é um protocolo em três fases iniciadas pela(s) Estação(ões) que descobriu(ram) a falha de comunicação ou por uma Estação recém-reativada. Estas Estações disputaram o privilégio de serem a Estação Mestre da Reforma. No caso de seccionamento da Rede, em cada partição haverá um Mestre. Mas apenas a partição que detiver a maioria das Estações obterá êxito na Reforma.

Na fase I, o Mestre convida as demais Estações, denominadas Escravos, a aderirem à nova versão do Grupo de Difusão. A mensagem enviada, Convite, contém o número da versão proposta, Ver, e é repetida, periodicamente (Temp5), até que todos os Escravos respondam ou até que se esgotem R tentativas. Uma nova versão do Grupo é formada pelos Escravos que aderiram. Um Escravo só adere se a nova versão proposta, Ver, passar no seu Teste de Seqüência e se o Escravo já não tiver aderido a uma outra versão em formação. A mensagem de adesão, Ack-Convite, informa ao Mestre o PCT e o M[Ei] do Escravo. Se algum Escravo não aderir, uma mensagem Rej-Convite será enviada ao Mestre.

Se o Mestre verificar que a nova versão satisfaz os Testes de Maioria e Resiliência e se não for recebida nenhuma rejeição, será iniciada a fase II da Reforma com a divulgação da nova

configuração de Estações no Grupo de Difusão, M, o Carimbo de Tempo inicial, PCT, e a determinação da nova Estação com Ficha. Tudo isso é reunido na mensagem Novo-Grupo. Senão, uma mensagem de interrupção da Reforma é divulgada aos Escravos que aderiram, Abou. Esta mensagem é repetida R vezes. O Mestre fica então inativo por um período aleatório, incrementa seu número de seqüência de versão, Seq, de modo a ser maior do que todas as versões anteriores e reinicia nova Reforma.

A Estação com Ficha, escolhida pelo Mestre, para a nova versão, é aquela que apresentar o maior PCT. Os Escravos, ao receberem o Novo-Grupo, iniciam a recuperação das mensagens que lhes faltam, baseados nos novos PCT e M. A fila Qb é esvaziada. Da fila Qc são extraídas as mensagens não comissionadas e as mensagens que faltam são recuperadas a partir da nova Estação com Ficha, que necessariamente está completa. Ao final da recuperação, uma mensagem, Ack-Novo-Grupo, é enviada ao Mestre. Nesse ponto, para todos os Escravos, exceto para a Estação-com-Ficha, a reforma é considerada terminada.

O Mestre, ao receber os Ack-Novo-Grupo de todos os Escravos, inicia a fase III da Reforma. Se algum Escravo deixar de Responder, dentro de um período Temp6, uma mensagem de interrupção da Reforma é divulgada aos Escravos que aderiram, Abou. O Mestre fica então inativo por um período aleatório, incrementa seu número de seqüência de versão, Seq, e reinicia nova Reforma.

Na fase III da Reforma, o Mestre envia uma mensagem, Habilita-Novo-Grupo, que autoriza a Estação com Ficha a iniciar a Fase Normal. A partir daí, a Reforma é considerada encerrada para o Mestre.

Como o Mestre pode falhar durante a Reforma, é permitido aos Escravos abandonar a adesão a determinada versão e aderir a uma versão mais alta. Em particular, isso ocorre se o Mestre demorar mais que um determinado período, Temp7(Temp8), para enviar a mensagem Novo-Grupo (Habilita-Novo-Grupo).

4. Vantagens e Problemas

As vantagens observadas no Protocolo de CHANG e MAXEMCHUK, considerados os requisitos do presente trabalho, são:
a - A grande eficiência apresentada pelo protocolo quando se dispõe de: Difusão física; uma baixa taxa de ruído no meio de transmissão; e um tráfego constante e relativamente alto de mensagens de dados (alto aqui significa que se a confirmação de uma difusão levar duas ou três vezes o tempo médio entre mensagens de dados, isto será perfeitamente aceitável). Nestas condições a eficiência atinge a duas mensagens reais por Difusão Confiável, isto é, torna a Difusão comparável à comunicação ponto-a-ponto com confirmação. A análise de desempenho deste protocolo foi apresentada por MAXEMCHUK et al [33].

b - O Carimbo de Tempo e a configuração de Estações ativas no Grupo de Difusão são informações globais mantidas pelo Protocolo e que podem ser úteis em outros algoritmos distribuídos localizados em níveis mais altos que o do protocolo.

Os principais problemas observados no Protocolo foram:
a - A imposição de um número n pré-definido de Estações no Grupo de Difusão de modo a viabilizar o Teste de Maioria. Isto é um obstáculo à sua utilização em sistemas ininterruptos, ou seja, sistemas que devem evoluir sem nunca serem totalmente desligados. Na forma proposta por CHANG e MAXEMCHUK, esta evolução em termos de número de Estações estaria limitada entre (n/2 + 1) e n. Para modificar estes limites seria necessário desligar e reconfigurar o sistema.

b - A falta de ponderações sobre o tamanho finito de Qc e Qb. Essa limitação de tamanho impõe que a partir de um momento, as mensagens mais antigas em Qc e Qb sejam descartadas. Assim, se uma Estação permanecer ativa mas isolada das outras por um tempo suficientemente longo, ao ser reconectada ao Grupo, poderá verificar que as mensagens por ela perdidas, já terão

sido descartadas das filas de mensagens das demais Estações. Neste caso, esta Estação terá perdido definitivamente as mensagens.

c - A inexistência, no domínio público, de uma especificação formal do Protocolo em linguagens como Estelle, Lotos ou Redes de Petri, e a ausência de qualquer demonstração de correção do Protocolo proposto deixam vários pontos sujeitos à interpretação subjetiva.

5. Alterações Introduzidas

As principais alterações introduzidas no Protocolo foram:

a - Relaxamento do Teste de Maioria de modo a não se pré-fixar o número máximo de participantes do Grupo de Difusão (o número mínimo é L, que constitui o limite imposto pela Resiliência desejada do Sistema). Tal relaxamento consistiu em considerar o n, para efeito de Teste de Maioria, como sendo o número de Estações ativas na última versão do Grupo de Difusão formado. Além disso, no Teste de Maioria, só contam as Estações que participaram do Ultimo Grupo de Difusão. Isto é, as Estações recém-ativadas podem originar uma Reforma, mas não contam no Teste de Maioria, pois não possuem o contexto das Estações que participaram do Grupo anterior.

b - A fim de remediar o problema da perda de mensagens, mencionado anteriormente, em caso de seccionamento da rede, estabeleceu-se que: quando uma Estação, que foi isolada do Grupo de Difusão mas não foi desativada, consegue retornar ao Grupo de Difusão, ela deve reiniciar-se. Para isto, o próprio Protocolo deve sinalizar esta ocorrência. Na reiniciação, todo o contexto da Estação será obtido de uma das Estações ativas.

Capítulo VII

Conclusões e Perspectivas

1. Conclusões 121

2. Futuros Desenvolvimentos 121

1. Conclusões

O Centro de Pesquisas de Energia Elétrica - CEPEL desenvolve atualmente um projeto de Digitalização de Usinas e Subestações que consiste na substituição dos equipamentos convencionais de supervisão e controle por equipamentos baseados em microprocessadores interconectados em Rede. O presente trabalho procurou identificar e indicar a solução para um problema estratégico dentro desse projeto: o encapsulamento de uma solução padrão para replicação de dados no Centro de Controle. A solução apresentada foi fruto de uma combinação harmoniosa de alguns algoritmos publicados nos últimos anos na área de Controle de Acesso Concorrente, Recuperação de Falhas e Protocolos de Difusão. A principal dificuldade na escolha dos algoritmos foi não comprometer os tempos de resposta de um sistema que deve operar em Tempo Real. Para tanto, algumas simplificações foram feitas na consideração sobre os tipos de faltas toleráveis pelo sistema.

Atualmente, dentro do projeto mencionado, está sendo utilizada a concepção de Banco de Dados do Processo descrita no presente trabalho. Numa próxima etapa, deverá ser realizada a replicação desse Banco de Dados tomando por base o GBDR apresentado neste trabalho.

2. Futuros Desenvolvimentos

No transcorrer deste trabalho, três pontos se revelaram passíveis de investigação futura com aparente proveito para o projeto em questão:

a - Investigação de uma alternativa para evitar ou remediar Bloqueios Perpétuos que aumentasse o paralelismo em relação à solução aqui apresentada.

b - A definição de um ambiente para teste e simulação de algoritmos paralelos e distribuídos, como os que foram apresentados neste trabalho, o qual determinaria, inclusive, a forma de modelagem.

c - A ampliação do conceito de replicação de dados para replicação de todo o contexto das tarefas, levando a definição de um sistema operacional distribuído adequado a sistemas em Tempo Real.

BIBLIOGRAFIA

[1] ANDERSON, M. D., SMITH, R. A.,
"Data Bases for Energy Management Systems - A Real-time Operational Approach", Power Industry Computer Applications Conference, 1977,
páginas: 74 a 82.

[2] ANDRADE, H. G.,
"Avaliação da Disponibilidade de um Centro de Supervisão Baseado em uma Arquitetura Distribuída", Tese de Mestrado, COPPE/UFRJ, 1983.

[3] APPEL, O., SILVA, A. J.,
"Arquitetura de Software para Centros de Supervisão e Controle de Usinas e Subestações", Anais do IX SNPTEE, Belo Horizonte, outubro /1987.

[4] ATTAR, R., BERNSTEIN, P. A., GOODMAN, N.,
"Site Initialization, Recovery and Backup in a Distributed Database System", IEEE Transactions on Software Engineering, Vol. SE-10, N. 6, Novembro/1984,
páginas: 645 a 649.

[5] AWERBUCH, B., EVEN, S.,
"Efficient and Reliable Broadcast is Achievable in an Eventually Connected Network", Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing, Agosto/1984,
páginas: 278 a 281.

[6] BABAOGLU, O., et alii,
"The Impact of Communication Network Properties on Reliable Broadcast Protocols", IEEE, 16th Simposium on FTCS, Digest of Papers, 1986,
páginas: 212 a 217.

[7] BERNSTEIN, P. A., GOODMAN, N.,
"A Sophisticate's Introduction to Distributed Database Concurrency Control", Proceedings of the Eighth International Conference on Very Large Databases, Cidade do México, Setembro/1982,
páginas: 62 a 76.

[8] BERNSTEIN, P. A. - GOODMAN, N. -
"An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases", ACM Transactions on Database Systems, Vol. 9, N. 4, Dezembro/1984, páginas: 596-615.

[9] BIRMAN, K. P., JOSEPH, T. A.,
"Low Cost Management of Replicated Data in Fault-Tolerant, Distributed Systems", Department of Computer Science, Cornell University, New York, Technical Report TR84-644, Outubro/1984.

[10] BORGES, M.R.S.,
"A Flexible Mechanism for Concurrency Control in Database Systems", Anais do VII Congresso da SBC, Salvador, julho/1987,
páginas: 142-151.

[11] CARVALHO, O., ROUCAIROL,G.,
"On Mutual Exclusion in Computer Networks", Communications of the ACM, Vol. 26, N. 2, fevereiro/1983,
páginas: 146-148.

[12] CHANG, J., MAXEMCHUK, N., F.,
"A Broadcast Protocol for Broadcast Networks", Proceedings of GLOBCOM, Dezembro/1983,
páginas: 649-653.

[13] CHANG, J., MAXEMCHUK, N. F.,
"Reliable Broadcast Protocols", ACM Transactions on Computer Systems, Vol. 2, N. 3, Agosto/1984,
páginas: 251-273.

[14] CHANG, J. M.,
"Simplifying Distributed Database Systems Design by Using a Broadccast Network",
Proceedings of ACM SIGMOD, Junho/1984,
páginas: 223-233.

[15] CHEN, P. P.,
"The Entity-Relationship Model - Toward a Unified View of Data",
ACM Transactions on Database Systems, V.1, N.1, Março/1976,
páginas: 9-36.

[16] COUCEIRO, L., BARRENECHA, H.,
"Sistemas de Gerência de Bancos de Dados Distribuídos",
LTC - Livros Técnicos e Científicos, 77 páginas, 1984.

[17] CRISTIAN, F. et alii,
"Atomic Broadcast: From Simple Message Diffusion to Byzantine Agreement", Research Report RJ 4540 (48668), IBM Research Lab., San Jose, Califórnia, 31 páginas, Dezembro/1984.

[18] DATE, C.J.
"An Introduction to Database Systems", Addison-Wesley, 3a. Edição, Fevereiro/1982.

[19] DENZEL, D., MAUTYNEN, R., CAPRIO, U. D.,
"Some Aspects of Data Acquisiton for Alarm Display and Recording of Disturbances", CIGRÉ, International Conference on Large High Voltage Electric Systems, 1984 Session, Study Commitee 39-12, Paris,
páginas: 1 a 9.

[20] EAGER, D.L., SEVCIK, K.C.,
"Achieving Robustness in Distributed Database Systems", ACM Transactions on Database Systems, Vol.8, N. 3, setembro/1983,
páginas: 354 a 381.

[21] FRAGA, J., LEMOS, R.,
"Conceituação e Terminologia em Sistemas Informáticos Tolerantes a Faltas e Intrusões", Anais do 2º Simpósio em Sistemas de Computação Tolerantes a Falhas, Campinas, Agosto/1987,
páginas: 177 a 188.

[22] FROST, R., HUYNEN, M., STAHL, U.,
"Becos 30 Software Systems for Large Dispatching Centers", Brown Bovery Review, 3:788, Vol. 66, março/1979.

[23] GARCIA-MOLINA, H., ABBOTT, R.K.,
"Reliable Distributed Database Management", Proceedings of the IEEE, Vol. 75, N. 5, maio/1987,
páginas: 601-620.

[24] GIFFORD, D. K.,
"Weighted Voting for Replicated Data",
Proceedings of the 7th Symposium on Operating Systems Principles, ACM, Nova Iorque, Dezembro/1979, páginas: 150 a 162.

[25] Intel Corporation,
"Introduction to 80386",
Abril/1986.

[26] JERABEK, A., RISCHEL, H.,
"Becos 20 - A Software System for Regional Dispatching Centers", Brown Bovery Review, 3:181, Vol. 66, março/1979.

[27] KOHLER, W. H.,
"A Survey of Techniques for Syncronization and Recovery in Decentralized Computer Systems", ACM Computing Surveys, Vol. 13, N. 2, Junho/1981, páginas: 149 a 183.

[28] LAMPSON, B. W., et alii,
"Distributed Systems - Architecture and Implementation. An Advanced Course", Lecture Notes in Computer Science 105, Springer Verlag, 1982.

[29] LAMPORT, L.,
"Time, Clocks, and the Ordering of Events in a Distributed System", Communications of the ACM, Vol.21, N. 7, Julho/1978, páginas: 558 a 565.

[30] LANN, G.,
"A Distributed System for Real-Time Transaction Processing",Computer, IEEE, V. 14, N. 2, Fevereiro/1981, páginas: 43-48.

[31] LOQUES, O. G.,
"Uma Técnica para a Construção de Software Distribuído Tolerante a Falhas", Anais do IV Simpósio sobre o Desenvolvimento de Software Básico da SBC, S. J. dos Campos, outubro/1984, páginas: 166-172.

[32] MAEKAWA, M.,
"A sqrt(N) Algorithm for Mutual Exclusion in Decentralized Systems", ACM Transactions on Computer Systems, Vol. 3, N. 2, Maio/1985,
páginas: 145-159.

[33] MAXEMCHUK, N. F., CHANG, J.,
"Analysis of the Messages Transmitted in a Broadcast Protocol",Proceedings of the International Conference on Communications, Amsterdan, 1984.

[34] MELLOR, S. J., WARD, P. T.,
"Structured Development for Real-Time Systems", Yourdon Press, Nova York, 1986.

[35] MOONEY, H. P., EVANS, J. W.,
"A Complete Relational DBMS for an EMS Product", Power Industry Computer Applications Conference, 1987,
páginas: 289 a 293.

[36] PARKER, D.S. et alii,
"Detection of Mutual Inconsistency in Distributed Systems", Proceedings of the Fifth Berkeley Workshop on Distributed Data Management & Computer Networks, Emeryville, CA, fevereiro/1981.

[37] POPEK, G. et alii,
"LOCUS: A Network Transparent, High Reliability Distributed System", Proceedings of the Eighth Symposium on Operating Systems Principles, 1981.

[38] RAYNAL, M.,
"Algorithmes Distribués et Protocoles",
Édition Eyrolles, 141 páginas, 1985.

[39] RICART, G., AGRAWALA, A.K.,
"An Optimal Algorithm for Mutual Exclusion in Computer Networks", Communications of the ACM, Vol.24, N. 1, janeiro/1981,
páginas: 9-17.

[40] SCHNEIDER, F. B. et alii,
"Fault-Tolerant Broadcasts",
Science of Computer Programming 4, North Holland,
páginas: 1-15.

[41] SEGALL, A.; AWERBUCH, B.,
"A Reliable Broadcast Protocol", IEEE Transactions on Communications, Vol. COM-31, n.7, Julho/1983, páginas: 895-901.

[42] SILVA, A.J. et alii,
"Descrição do Software Básico do Centro de Operação Regional", Anais do IV Simpósio sobre o Desenvolvimento de Software Básico da SBC, S. J. dos Campos, outubro/1984,
páginas: 34-60.

[43] SILVA, A.J. et alii,
"Projeto do Software Aplicativo e de Suporte para o Centro de Operação Regional de Supervisão e Controle de FURNAS", Anais do VIII SNPTEE, SP/GPC/17, São Paulo, maio/1986.

[44] SON, S. H.,
"Syncronization of Replicated Data in Distributed Systems", Information Systems, V.12, N.2,
páginas 191-220, 1987.

[45] STRONG, H., DOLEV, D.,
"Byzantine Agreement", Proceedings of COMPCOM Spring 83, São Francisco, 1983,
páginas: 77 a 81.

[46] TANENBAUM, A. S.,
"Operating Systems - Design and Implementation",
Prentice Hall, 720 páginas, 1987.

[47] THOMAS, R.H.,
"A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases", ACM Transactions on Database Systems, Vol. 4, N. 2, junho/1979,
páginas: 180-209.

[48] VOSS, K.,
"Nets in Data Bases",
Advanced Course on Petri Nets, GMD, Bad Honnef, Alemanha, 8 a 19/setembro/1986.

[49] BRAMS, G. W.,
"Réseaux de Petri: Théorie et Pratique",
Masson, V.2, 1983, 160 páginas.

Apêndice

Modelo do Protocolo de Difusão Confiável

A. Modelo 130
A.1 Rede de Petri Interpretada 130
A.1.1 Definição 130
A.1.2 Funcionamento 131
A.2 Descrição do Modelo do Protocolo de Difusão Confiável 131
A.2.1 Formatos das Mensagens e Convenções Globais Auxiliares 132
A.2.2 Estados do Protocolo 134
A.2.3 M: Número de Seqüência das Mensagens a Receber 135
A.2.4 PCT: Próximo Carimbo de Tempo 136
A.2.5 Ver: Versão corrente do Grupo de Difusão 136
A.2.6 O depósito de Dados Qb 136
A.2.7 A fila de Controle Qc 137
A.2.8 Primitivas do Protocolo de Difusão Confiável oferecidas aos Usuários 139
A.2.9 Primitivas Externas utilizadas pelo DC 139
A.2.10 Primitivas de Verificação das Filas Qc e Qb 140
A.2.11 Fases e Seções do Protocolo de Difusão Confiável 141
A.2.11.1 Modelo da Seção Mestre 142
A.2.11.2 Modelo da Seção Escravo 147
A.2.11.3 Modelo da Fase Normal 152

A. Modelo

Devido à complexidade que este protocolo apresenta, decidiu-se obter um modelo detalhado que permitisse principalmente a familiarização com todas as suas sutilezas. Devido à capacidade de percepção gráfica humana, na identificação visual de simetrias e outros tipos de recursão, decidiu-se utilizar Redes de Petri na modelagem. Com o intuito de introduzir no modelo, além do fluxo de controle, as pricipais estruturas de dados envolvidas, decidiu-se utilizar uma variação denominada Rede de Petri Interpretada.

A.1. Rede de Petri Interpretada

Redes de Petri Interpretadas são definidas por BRAMS [49]. As Redes de Petri Interpretadas são uma forma de descrição de sistemas paralelos, na qual é explicitada a separação entre o fluxo de controle e as manipulações sobre um conjunto de dados.

O fluxo de controle é descrito por uma rede de Petri. Os dados constituem o contexto do Sistema. As manipulações são descritas como operações sobre o contexto.

A.1.1 Definição

Uma Rede de Petri Interpretada consiste de uma Rede de Petri R = (Lg,Tr,Pré,Pós); de um contexto e operações sobre este, <DD,OP,PR>; e de uma função F que liga a Rede ao Contexto.

Lg é um conjunto de Lugares.

Tr é um conjunto de Transições.

Pré é uma função de Lg em Tr que indica os arcos que ligam os Lugares às Transições.

Pós é uma função de Tr em Lg que indica os arcos que ligam as Transições aos Lugares.

DD é o conjunto de estados do contexto.

OP = { op_1, op_2,...,op_s} é um conjunto de operadores;

op_i: DD -> DD.

PR = {pr_1, pr_2,...,pr_s} é um conjunto de predicados sobre DD; pr_i: DD -> {verdadeiro, falso}.

F: Tr -> PR x OP, é uma função que associa a cada transição da Rede uma dupla (pr_i, op_i).

A.1.2 Funcionamento

A descrição do funcionamento de uma Rede de Petri simples pode ser encontrada em [49] e é considerada conhecida aqui.

Uma Rede Interpretada representa um sistema no qual o estado é uma dupla (d, m) onde d ∈ DD e m é a marcação (distribuição de Fichas pelos Lugares) da Rede associada. O estado, d, do contexto pode modificar-se pela execução de uma operação op_i; a aplicação de op_i a d resulta em um estado d´ = op_i(d).

Em relação a uma Rede de Petri simples, as condições de disparo das Transições são modificadas para levar em consideração o estado do contexto. Uma transição t é disparável se os lugares que lhe são incidentes contêm um número suficiente de Fichas e o predicado da dupla F(t) resulta verdadeiro pelo estado atual do contexto.

O disparo de uma Transição, t, consiste em aplicar a mesma regra de disparo das Redes de Petri simples, no que diz respeito ao consumo e produção de novas Fichas, e, por outro lado, efetuar uma transformação no estado do contexto através da aplicação do operador da dupla F(t).

A.2 Descrição do Modelo do Protocolo de Difusão Confiável

Consideram-se nesta descrição apenas os dados, as estruturas e os algoritmos necessários ao bom funcionamento da lógica de controle do protocolo de Difusão Confiável (DC). Em particular, não interessam os dados do usuário (parte útil das mensagens), que, por isso, não aparecem na descrição. Tampouco é examinada

a colocação de mais de um Grupo de Difusão nas Estações da rede.

A descrição a seguir deve ser confrontada com a descrição informal do DC feita no Capítulo VI.

Optou-se, ao longo desta descrição, por apresentar os formatos através de um conjunto de declarações num estilo semelhante ao da linguagem MODULA-2.

A.2.1. Formato das mensagens e convenções globais auxiliares

As reticências "..." assinalam itens cujo detalhamento está fora do âmbito desta descrição.

```
TIPO
  Num_Seq_Ver=...
    (*Número de seqüência das versões dos Grupos de Difusão.*)

  Ident_Estação=...
    (*Identificação de estação.*)

  Ident_Versão=
    ESTRUTURA
     Num_Ver:Num_Seq_Ver;
     Est:Ident_Estação
    FIM ESTRUTURA;
    (*Identificação que permite distinguir as diversas
    versões de um grupo, em função da ativação e desativação de
    estações.*)

  Num_Seq=...
    (*Número de seqüência de mensagem de dados da estação.*)
```

```
Ident_Dados=
  ESTRUTURA
    Est:Ident_Estação;
    MEi:Num_Seq
  FIM ESTRUTURA;
  (*Serve para identificar as mensagens de dados.*)

Carimbo_Tempo=...
  (*"Carimbo de tempo" usado como critério de ordenação
  das mensagens de dados do grupo.*)

Tipo_M=...
  (*Permite armazenar, para cada estação do grupo de
  difusão, o número de seqüência da próxima mensagem de
  dados a ser recebida. Tipo_M também define, de forma
  implicita, uma ordem cíclica das estações do grupo.*)

Formato_Mensagem=
  (Convite, Ack_Convite, Rej_Convite, Abou,
  Novo_Grupo, Ack_Novo_Grupo, Habilita_Novo_Grupo, Dados,
  Ack_Dados, Ack_Nulo, Confirmação, Req_Ack, Req_Dados);
```

```
Mensagem=
  ESTRUTURA
    Ver:Ident_Versão;
    CASO Fmt:Formato_Mensagem
      DE Convite: Est:Ident_Estação
        (*Para distinguir estação que não pertence ainda
        ao grupo.*)
      DE Ack_Convite: Est:Ident_Estação;
        CT:Carimbo_Tempo; M:Tipo_M
      DE Rej_Convite: Est:Ident_Estação
      DE Abou:
      DE Novo_Grupo: CT:Carimbo_Tempo; M:Tipo_M;
        Est:Ident_Estação (*Detentor da ficha*)
      DE Ack_Novo_Grupo: Est:Ident_Estação
      DE Habilita_Novo_Grupo:
      DE Dados: Id:Ident_Dados; (*D:Tipo_Dados*)
      DE Ack_Dados: Id:Ident_Dados; CT:Carimbo_Tempo
      DE Ack_Nulo: CT:Carimbo_Tempo
      DE Confirmação: CT:Carimbo_Tempo
      DE Req_Ack: CT:Carimbo_Tempo
      DE Req_Dados: CT:Carimbo_Tempo
    FIM CASO
  FIM ESTRUTURA;
```

A.2.2. Estados do protocolo

A Rede de Petri que modela o DC pode ser seccionada em sub-redes que são Máquinas de Estados. Os principais estados dessas sub-redes são relacionados a seguir.

```
VAR

  Estado:
    CASO Principal:(Normal, Reforma)
      DE Normal:
        Secundário_Recepção:(Com_Ficha_Recuperando_Mensagens,
          Com_Ficha, Passando_Ficha, Sem_Ficha,
          Sem_Ficha_Recuperando_Mensagem)
        Secundário_Transmissão:(Inativa, Transmitindo_Mensagem)
      DE Reforma:
        Secundário_Mestre:(Inativo, Convite_a_Grupo,
          Esperando_Aceitação_de_Grupo);
        Secundário_Escravo:(Inativo, Esperando_Grupo,
          Esperando_Ficha,
          Atualizando_Mensagens_após_Reforma)
        Secundário_Transmissão:(Livre, Mensagem_Pendente)
    FIM CASO;
```

A.2.3. M: Número de seqüência das mensagens a receber

Primitivas de acesso a M:

- M.Val(Est:Ident_Estação):Num_Seq;
  (*Retorna o número de seqüência da próxima mensagem a ser recebida da estação Est.*)

- M.Inc(Est:Ident_Estação);
  (*Incrementa o número de seqüência da próxima mensagem a ser recebida da estação Est.*)

- M.Est_com_Ficha(ENT CT:Carimbo_Tempo):Ident_Estação;
  (*Retorna a Estação responsável por emitir um Ack com o Carimbo de Tempo fornecido.*)

A.2.4. PCT: Próximo carimbo de tempo

Primitivas de acesso a PCT:

- PCT.Val():Carimbo_Tempo;
  (* Retorna o valor do próximo Carimbo de Tempo *)
- PCT.Inc;
  (* Incrementa o Carimbo de Tempo *)

A.2.5. Ver: Versão corrente do Grupo de Difusão

Primitivas de acesso a Ver:

- Ver.Val():Ident_Versão;
  (* Retorna o valor atual da Versão *)
- Ver.Inc;
  (* Incrementa a Versão*)

A.2.6. O depósito de dados Qb

O depósito de dados Qb armazena as mensagens recebidas enquanto não puderem ser aproveitadas, isto é, ordenadas, em Qc, de acordo com o Carimbo de Tempo.

Primitivas de acesso a Qb:

- Qb.Inserir(ENT Id:Ident_Dados);
  (* Insere uma mensagem de Dados em Qb*)

- Qb.Existe(ENT Id:Ident_Dados):BOOLEANO;
  (* Verifica se existe uma determinada mensagem de Dados em Qb *)
- Qb.Existe_Algum():BOOLEANO;
  (* Verifica se Qb não está vazia *)

```
- Qb.Qualquer():Ident_Dados;
        (* Retorna o identificador de alguma mensagem de
             Dados em Qb *)
- Qb.Remover(ENT Id:Ident_Dados);
        (* Remove uma determinada mensagem de Dados de Qb *)
```

A.2.7. A fila de controle Qc

Em Qc são inseridas, ordenadas pelo Carimbo de Tempo, as mensagens de confirmação: Ack-Dados e Ack-Null. O funcionamento de Qc permite que seja associada a uma determinada confirmação, Ack-Dados, a mensagem de Dados correspondente. A fila Qc pode ser pesquisada a partir do identificador de uma mensagem de Dados ou do Carimbo de Tempo de uma confirmação. Essa pesquisa informa os estados das mensagens de confirmação. Isto é, se existem e se já foram associadas.

Os estados das mensagens de confirmação são:

```
TIPO

  Status_Msg=(Falta_Msg, Falta_Ack, Pronto_com_Msg);

  Status_Ack=
    (Falta_Msg, Falta_Ack, Pronto_com_Msg, Pronto_sem_Msg);
```

Primitivas de acesso a Qc:

```
- Qc.Inserir(ENT CT:Carimbo_Tempo; ENT Id:Ident_Dados);
        (* Insere uma mensagem de confirmação. No caso de um
        Ack-Dados, este é inserido no estado Falta_msg. No
        caso de Ack-Null, este é inserido no estado
        Pronto_sem_Msg *)
```

```
- Qc.Status_Msg(ENT Id:Ident_Dados):Status_Msg;
        (* Retorna o estado de uma confirmação em uma
        pesquisa feita a partir do identificador de uma
        mensagem de Dados:
        Falta_Ack significa que não existe uma confirmação
              associada a esta mensagem de Dados em Qc;
        Falta_Msg significa que a confirmação existe mas não
              foi feita a associação ainda;
        Pronto_com_Msg significa que a confirmação existe e a
             associação também.*)

- Qc.Status_Ack(ENT CT:Carimbo_Tempo):Status_Ack;
        (* Retorna o estado de uma confirmação em uma
        pesquisa feita a partir do Carimbo de Tempo de uma
        mensagem de confirmação:
        Falta_Ack significa que não existe esta confirmação
              em Qc;
        Falta_Msg significa que a confirmação existe mas não
              foi feita a associação com a mensagem de Dados
              ainda;
        Pronto_com_Msg significa que a confirmação existe e a
             associação também.
        Pronto_sem_Msg significa que a confirmação existe e é
        do tipo Ack_Null que não é associada a nenhuma
        mensagem de Dados.*)

- Qc.Mensagem(ENT CT:Carimbo_Tempo):Ident_Dados;
        (* Retorna o identificador da mensagem de Dados a
        partir do Carimbo de Tempo da confirmação associada*)

- Qc.Associar(ENT Id:Ident_Dados);
        (* Associa a mensagem de Dados a uma confirmação
        existente em Qc e altera seu estado de Falta_Msg
        para Pronto_com_Msg*)

- Qc.Existe_Algum():BOOLEANO;
        (* Verifica se existe alguma mensagem L-resiliente em
        Qc *)
```

- Qc.Remover():Ident_Dados;
  (*Retira a primeira mensagem L-resiliente.*)

A.2.8 Primitivas do Protocolo de Difusão Confiável Oferecidas aos Usuários

O DC fornece cinco primitivas ao usuário:

- Tx(ENT D:...);
  (* Transmite dados do usuário por Difusão Confiável*)

- Rx():...;
  (* Recebe dados via DC*)

- DC.Config():Tipo_M;
  (* Retorna a configuração de Estações ativas no Grupo de Difusão*)

- DC.PCT.Val():Carimbo_Tempo;
  (* Retorna o valor do Próximo Carimbo de Tempo*)

- DC.Cria_Grupo():Signal;
  (* O usuário pode, através dessa primitiva, determinar que uma Estação seja a originadora de um novo Grupo*)

A.2.9 Primitivas Externas utilizadas pelo DC

O DC chama duas primitivas de difusão física, uma de reiniciação da Estação e uma de sinalização de Seccionamento da Rede:

- Ptx(ENT Msg:Mensagem);
  (* Transmite uma mensagem para um determinado Grupo de Difusão*)
- Prx():Mensagem;
  (* Recebe uma mensagem de um determinado Grupo de Difusão*)

- Rede_Seccionada():Signal;

(* Sinaliza a ocorrência de um seccionamento da Rede tal que não permite a formação de um Grupo de Difusão. A partir dessa sinalização, o usuário pode desativar a Estação ou forçar a criação de um novo Grupo através da primitiva DC.Cria_Grupo*)

- Reativa_Estação():Signal;

(* Sinaliza que o DC detectou que foram passadas mensagens discrepantes ao usuário dessa Estação e portanto todo o contexto do usuário deve ser reiniciado*)

A.2.10 Primitivas de Verificação das Filas Qc e Qb

Como os Acks em Qc ficam ordenados por seu Carimbo de Tempo, pode-se estabelecer uma correspondência entre o valor de PCT e posições em Qc.

A fila Qc, quando completa, tem no seu topo um Ack_Dados ou Ack_Nulo com Carimbo de Tempo igual a (PCT-1). Além disso, todos os Acks estão marcados como Prontos. Entretanto, como há perda de mensagens, Qc pode apresentar "buracos". Buracos são Ack_Dados marcados com Falta_Msg ou Acks posicionados além de PCT.

Existem duas primitivas para a verificação de Buracos em Qc.

TIPO

Status_Buraco=(Não_tem_Buracos, Tem_Buracos, Tapa_Buracos, Cria_Buracos)

- Q.Verifica_Buraco(ENT Msg:Mensagem):Status_Buraco;

(* Esta primitiva verifica se a inserção de um Ack em Qc cria buracos em Qc que estava completa, tapa todos os buracos de Qc que estava incompleta ou não altera o estado de Qc, isto é, se Qc tinha buracos ou estava completa, a inserção de Msg não altera essa condição. No caso da inserção de um Ack_Dados, a primitiva verifica se a mensagem de Dados correspondente está

disponível em Qb. Se não, significa que a inserção do Ack_Dados cria um Buraco. No caso de uma mensagem de Dados, a primitiva verifica se é a mensagem que faltava associar a um Ack_Dados de Qc para tapar os Buracos. Deve-se notar que essa primitiva apenas consulta Qb e Qc sem fazer qualquer inserção e retorna a condição na qual Qc ficaria na hipótese de uma inserção efetiva*)

- Q.Falta(SAI CT:Carimbo_Tempo):Formato_Mensagem;
(* Esta primitiva consulta Qc e Qb e obtém o Carimbo de Tempo de uma mensagem que tapa algum Buraco em Qc. Além disso, a primitiva retorna o tipo de mensagem de recuperação a ser utilizado: Req_Dados ou Req_Ack *)

A.2.11 Fases e Seções do Protocolo de Difusão Confiável

A Figura A-1 apresenta uma Rede de Petri simplificada que representa a interligação entre as três sub-redes que constituem o modelo: Seção Mestre da Fase de Reforma (Mestre), Seção Escravo da Fase de Reforma (Escravo) e Fase Normal (Normal).

As Transições serão descritas em função dos predicados que as disparam e das correspondentes operações realizadas sobre o contexto. Deve-se observar que, por uma questão de simplificação, as Transições ("loops"), que consomem a Ficha de um Lugar e a retorna a este mesmo Lugar, não foram representadas nas Figuras. A descrição das operações, neste caso, é feita na descrição do Lugar. Também por simplificação, assume-se que, em qualquer estado de espera de mensagens, as mensagens não especificadas são simplesmente desprezadas ao chegarem.

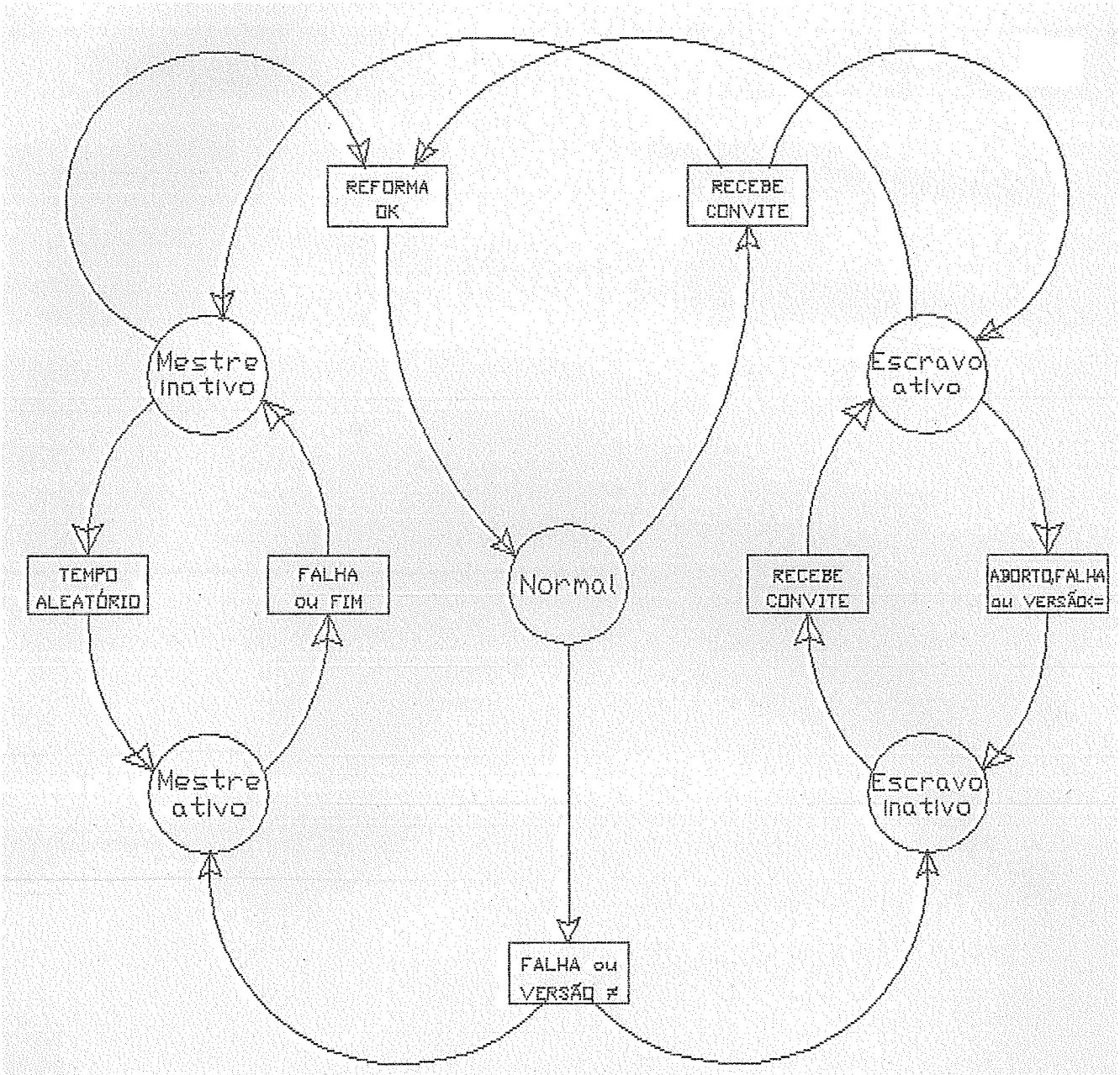


Figura A-1 : Protocolo de Difusão Confiável

A.2.11.1 Modelo da Seção Mestre

A Seção Mestre é apresentada na Figura A-2. A seguir são descritos os oito Lugares e quatorze Transições da Seção Mestre.

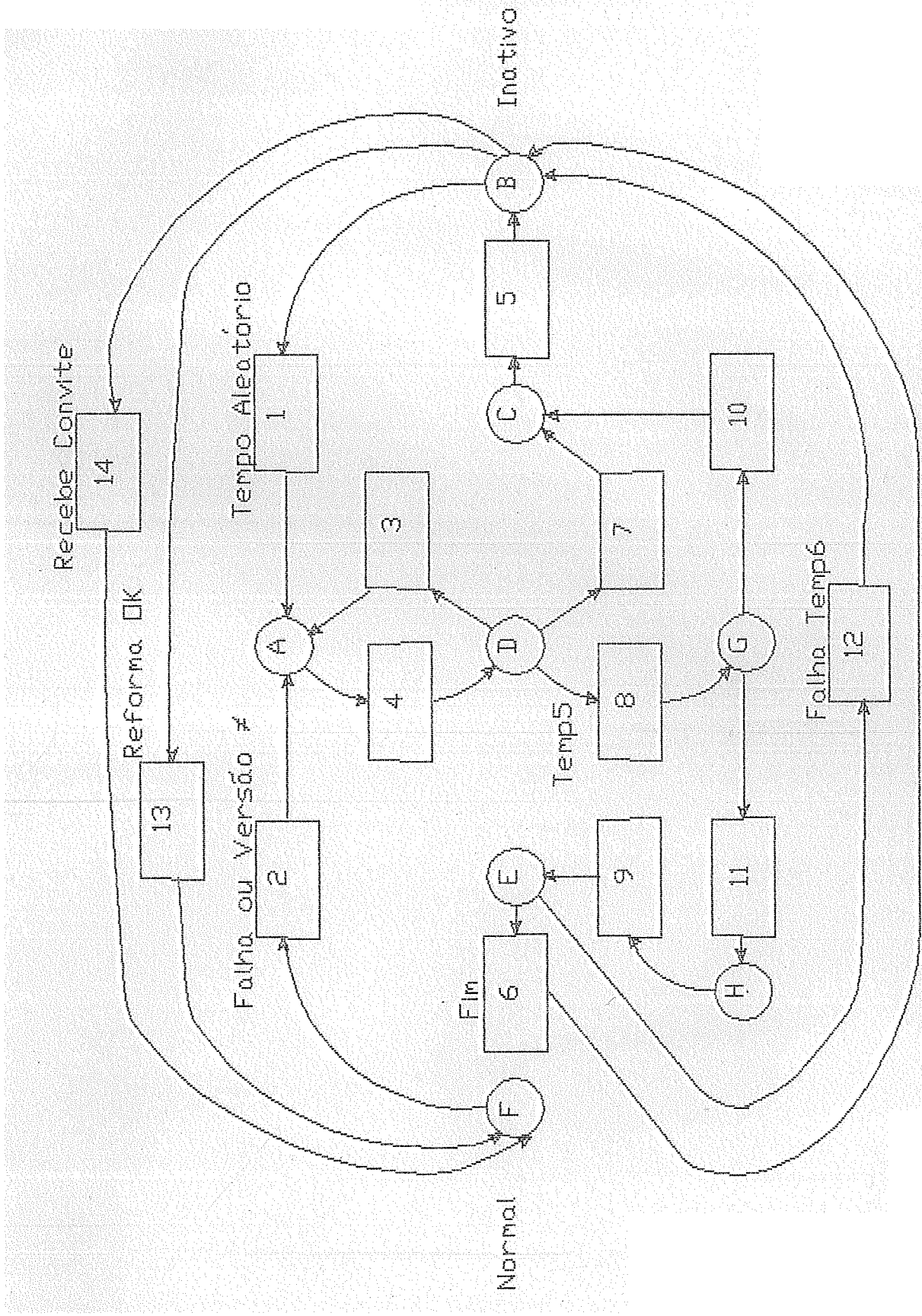


Figura A-2 : Seção Mestre da Reforma

a - Lugares

(A) Envio do Convite (fase I)

Este Lugar representa que a Seção Mestre iniciou uma Reforma e vai enviar um Convite ao Grupo. Neste estado, se for verificado que foi recebida uma mensagem, diversa de um Convite, com versão maior que a atual, será presumido que a Estação esteve algum tempo isolada numa partição da Rede e que não acompanhou alguma Reforma do Grupo. Ou seja, há a possibilidade de discrepância de mensagens e portanto é melhor reiniciar a Estação:

```
Se (Msg.Fmt<>Convite e Msg.Ver>Ver.Val())
Então Reativa_Estação();
```

(B) Inativo

Este Lugar representa que a Seção Mestre está inativa, aguardando um tempo randômico para iniciar uma Reforma. Este é o estado inicial após a ativação de uma Estação.

(C) Abortando a Reforma

Este Lugar representa a Seção Mestre abortando uma Reforma. A operação executada aqui é:

```
Se não é a R-ésima vez
     Então Msg.Fmt:=Abou;Msg.Ver:=Ver.Val();Ptx(Msg);
```

(D) Convite a Grupo

Este Lugar representa a Seção Mestre aguardando as respostas do Convite para uma nova versão Grupo. A operação executada aqui é:

```
Se (Msg:=Prx()) e (Msg.Fmt=Ack_Convite e
                        (Msg.Est não completa M[])
     Então marca a adesão em M[Msg.Est];
```

(E) Esperando Aceitação de Grupo

Este Lugar representa a Seção Mestre aguardando as confirmações de que os escravos aceitaram a nova versão e terminaram a recuperação de mensagens. Enquanto aguarda, a Seção Mestre repete, periodicamente (Temp6), o Novo_Grupo. As operações são:

```
Se (Esgotou Temp6) e (Não é a R-ésima vez)
     Então Arma Temp6; Msg.fmt:=Novo_Grupo; Msg.Ver:=Ver.Val();
          Ptx(Msg);
Se (Msg:=Prx()) e (Msg.Fmt:=Ack_Novo_Grupo) e
                       (Msg.Est não completa M[])
     Então marca a adesão em M[Msg.Est];
```

(F) Normal

Este Lugar representa toda a Fase Normal do protocolo.

(G) Verifica Grupo

Este Lugar representa a realização dos teste de Maioria e Resiliência. O teste de Maioria verifica se a maioria das Estações pertencentes à última versão do Grupo de Difusão aderiu ao Convite. A operação executada aqui é:

```
Se (as Estações em M[] não são a Maioria em relação ao M[]
      anterior) e (não há Estações do M[] anterior ativas)
     Então Rede_Seccionada();
```

(H) Divulga novo Grupo (fase II)

Este Lugar representa a divulgação do Novo Grupo.

b - Transições

(1) Tempo Aleatório

Se Esgotou um Tempo Aleatório inativo
Então Ver.Inc;

(2) Detecção de Falha

```
Se (Esgotou pela R-ésima vez Temp1, Temp2 ou Temp3) ou
      ((Msg:=Prx()) e Msg.Ver<>Ver.Val())
Então Ver.Inc;
```

(3) Prepara Repetição do Convite

Se (Esgotou Temp5) e (Não é a R-ésima vez)

```
(4) Envia Convite
Se Verdadeiro
Então Msg.Fmt:=Convite;Msg.Est:=Esta_Estação;
      Msg.Ver:=Ver.Val();Ptx(Msg);Arma Temp5;

(5) Envia Aborte
Se Verdadeiro
Então Msg.Fmt:=Abou;Msg.Ver:=Ver.Val();Ptx(Msg);

(6) Fim
Se (Msg:=Prx()) e (Msg.Fmt:=Ack_Novo_Grupo) e
                        Msg.Est completa M[])
Então Desarma Temp6;Msg.Fmt:=Habilita_Novo_Grupo;
      Msg.Ver:=Ver.Val();Ptx(Msg);

(7) Recebe Veto
Se (Msg:=Prx()) e (Msg.Fmt:=Rej_Convite)
Então Desarma Temp5;

(8) Termina Convites
Se ((Esgotou Temp5) e (É a R-ésima vez)) ou
      ((Msg:=Prx()) e (Msg.Fmt:=Ack_Convite) e
                        (Msg.Est completa M[]))
      Então marca a adesão em M[Msg.Est];Desarma Temp5;

(9) Transmite Novo Grupo
Se Verdadeiro
      Então Arma Temp6; Msg.fmt:=Novo_Grupo; Msg.Ver:=Ver.Val();
            Ptx(Msg);

(10) Grupo não passa nos Testes
Se ((as  Estações em  M[] não  são a  Maioria em relação ao M[]
       anterior) e (há Estações do M[] anterior ativas))
      ou
      (não aderiram  a M[] nenhuma das L Estações que sucediam a
      Estação com Ficha no M[] anterior))
Então;
```

```
(11) Grupo passa nos Testes
Se ((as Estações em M[] constituem uma Maioria em relação ao
     M[] anterior) e
     (aderiu a M[] pelo menos uma das L Estações que sucediam
      a Estação com Ficha no M[] anterior))
     ou
     (DC.Cria())
Então;

(12) Falha durante a Reforma
Se (Esgotou Temp6) e (É a R-ésima vez)
     Então Msg.fmt:=Abou; Msg.Ver:=Ver.Val();Ptx(Msg);

(13) Reforma OK
Se disparem as transições (12) ou (14) da Seção Escravo
Então;

(14) Recebe Convite
Se (Msg:=Prx()) e (Msg.Fmt=Convite)
Então;
```

A.2.11.2 Modelo da Seção Escravo

A Seção Escravo é apresentada na Figura A-3. A seguir são descritos os sete Lugares e as quinze Transições da Seção Escravo.

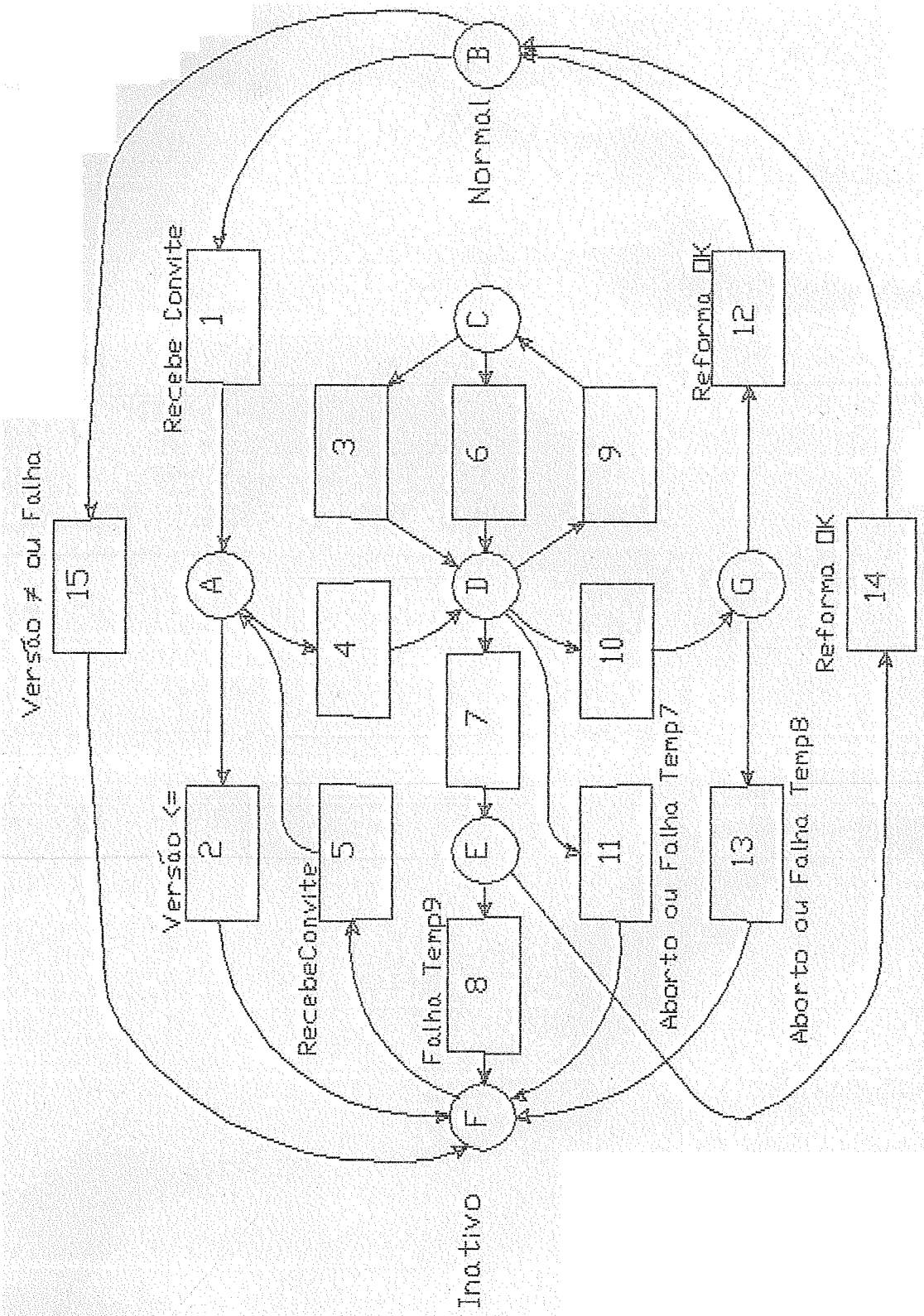


Figura A-3 : Seção Escravo da Reforma

a - Lugares

(A) Verifica Convite
Este é um estado de espera da comparação da versão da mensagem Convite recebida com a versão atual do Grupo de Difusão.

(B) Normal
Este Lugar representa toda a Fase Normal do protocolo.

(C) Verifica Convite Repetido
Este lugar representa a espera do teste de Versão de uma nova mensagem Convite que chegue enquanto era esperada a mensagem Novo_Grupo.

(D) Esperando Grupo
Este lugar representa a espera da mensagem Novo_Grupo.

(E) Recuperando Mensagens após Reforma
Este lugar representa a recuperação das mensagens que faltam em Qc após a reforma. As seguintes operações são executadas:

```
Se (falta em Qc algum Ack_Dados ou Ack_Null) e (não está
     recuperando nenhuma mensagem)
     Então Arma Temp9;Msg.Fmt:=Req_Ack;
          Msg.CT:=Carimbo de Tempo;Msg.Ver:=Ver.Val();
          Ptx(Msg);
Se (algum Ack_Dados em Qc está no estado Falta_Msg) e
 (não está recuperando nenhuma mensagem)
     Então      Arma Temp9; Msg.Fmt:= Req_Dados;
                Msg.CT := Carimbo de Tempo; Msg.Ver:=Ver.Val();
                Ptx(Msg);
Se (Msg:=Prx()) e (Msg.Fmt = Ack_Dados ou Msg.Fmt = Ack_Null) e
     (Msg.CT = Carimbo de Tempo)
     Então Desarma Temp9; Qc.Inserir(Msg.CT,Msg.Id);
Se (Msg:=Prx()) e (Msg.Fmt = Dados ) e
     (Msg.CT = Carimbo de Tempo)
     Então Desarma Temp9; Qc.Associar(Msg.Id);
```

(F) Inativo

Este estado representa que a Seção Escravo está inativa. Isto é, não aderiu a nenhuma nova versão. Este é o estado inicial após a ativação da Estação.

(G) Esperando Ficha

Este estado representa, na Estação com Ficha, a espera da mensagem Habilita_Novo_Grupo que encerra a fase III da reforma. Enquanto isso, recusa novos convites e responde aos pedidos de recuperação de mensagens. As operações são as seguintes:

```
Se (Msg:=Prx()) e Msg.Ver>Ver.Val() e Msg.Fmt=Convite
     Então Msg.Fmt:=Rej_Convite; Msg.Ver:=Ver.Val();Ptx(Msg);
Se (Msg:=Prx()) e Msg.Fmt=Req_Ack
     Então Msg.Id:=Qc.Mensagem(Msg.CT); Msg.Fmt:=Ack_Dados;
           Msg.Ver:=Ver.Val();Ptx(Msg);
Se (Msg:=Prx()) e Msg.Fmt=Req_Dados
     Então Status:=Qc.Status_Msg(Msg.Id); Msg.Fmt:=Dados;
           Msg.Ver:=Ver.Val();Ptx(Msg);
```

b - Transições

```
(1) (5) (9) Recepção de Convite
     Se (Msg:=Prx()) e (Msg.Fmt = Convite)

(2) Versão de Convite Inválida
     Se Msg.Ver <= Ver.Val()
     Então Msg.Fmt := Rej_Convite; Msg.Ver:=Ver.Val();Ptx(Msg);

(3) Convite Repetido Válido
     Se Msg.Ver = Ver.Val()
     Então Msg.Fmt := Ack_Convite; Msg.Ver:=Ver.Val();Ptx(Msg);

(4) Convite Válido
     Se Msg.Ver > Ver.Val()
     Então Ver.Atualiza(Msg.Ver);Msg.Fmt:= Ack_Convite;
           Msg.Ver:=Ver.Val();Ptx(Msg);Arma Temp7;
```

```
(6) Convite Repetido Inválido
     Se Msg.Ver = Ver.Val()
     Então Msg.Fmt := Rej_Convite;Msg.Ver:=Ver.Val();Ptx(Msg);

(7) Recepção do Novo Grupo
     Se (Msg:=Prx()) e Msg.Fmt=Novo_Grupo e
           Msg.Est<>Esta_Estação
     Então Desarma Temp7;

(8) Falha na Recuperação de Mensagens
     Se Esgotou Temp9 pela R-ésima vez
     Então;
(10) Recepcão novo Grupo e Ficha
     Se (Msg:=Prx()) e Msg.Fmt=Novo_Grupo e
           Msg.Est=Esta_Estação
     Então Msg.Fmt := Ack_Novo_Grupo; Msg.Ver:=Ver.Val();
          Ptx(Msg);

(11) Aborto da Reforma
     Se ((Msg:=Prx()) e Msg.Ver=Ver.Val() e Msg.Fmt=Abou) ou
          (Esgotou Temp7)
     Então;

(12)  Reforma OK para Estação com Ficha
     Se (Msg:=Prx()) e Msg.Fmt ◘ Habilita_Novo_Grupo
     Então Desarma Temp8;

(13) Aborto da Reforma
     Se ((Msg:=Prx())  e Msg.Ver=Ver.Val()  e Msg.Fmt=Abou)  ou
          (Esgotou Temp8)
     Então;

(14) Reforma OK para Estação sem Ficha
     Se Qc está completo
     Então Desarma Temp9;
           Enquanto Qb.Existe_Algum()
               Qb.Remover(Qb.Qualquer());
               Msg.Fmt:=Ack_Novo_Grupo;
               Msg.Ver:=Ver.Val();Ptx(Msg);
```

(15) Falha ou Versão Errada
Se (Esgotou Temp1 ou Temp2 ou Temp3) e (É a R-ésima vez)
    ou
    ((Msg:=Prx()) e (Msg.Ver<>Ver.Val())
Então;

A.2.11.3 Modelo da Fase Normal

É apresentado na Figura A-4. A seguir são descritos os nove Lugares e as cinqüenta e duas Transições da Rede de Petri da Fase Normal. Embora constituam um elevado número, estas Transições resultam da combinação de um pequeno número de Predicados e Operações que são descritos a seguir. Para evitar repetições de Predicados e Operações utilizam-se reticências (...) em seguida ao nome dos mesmos.

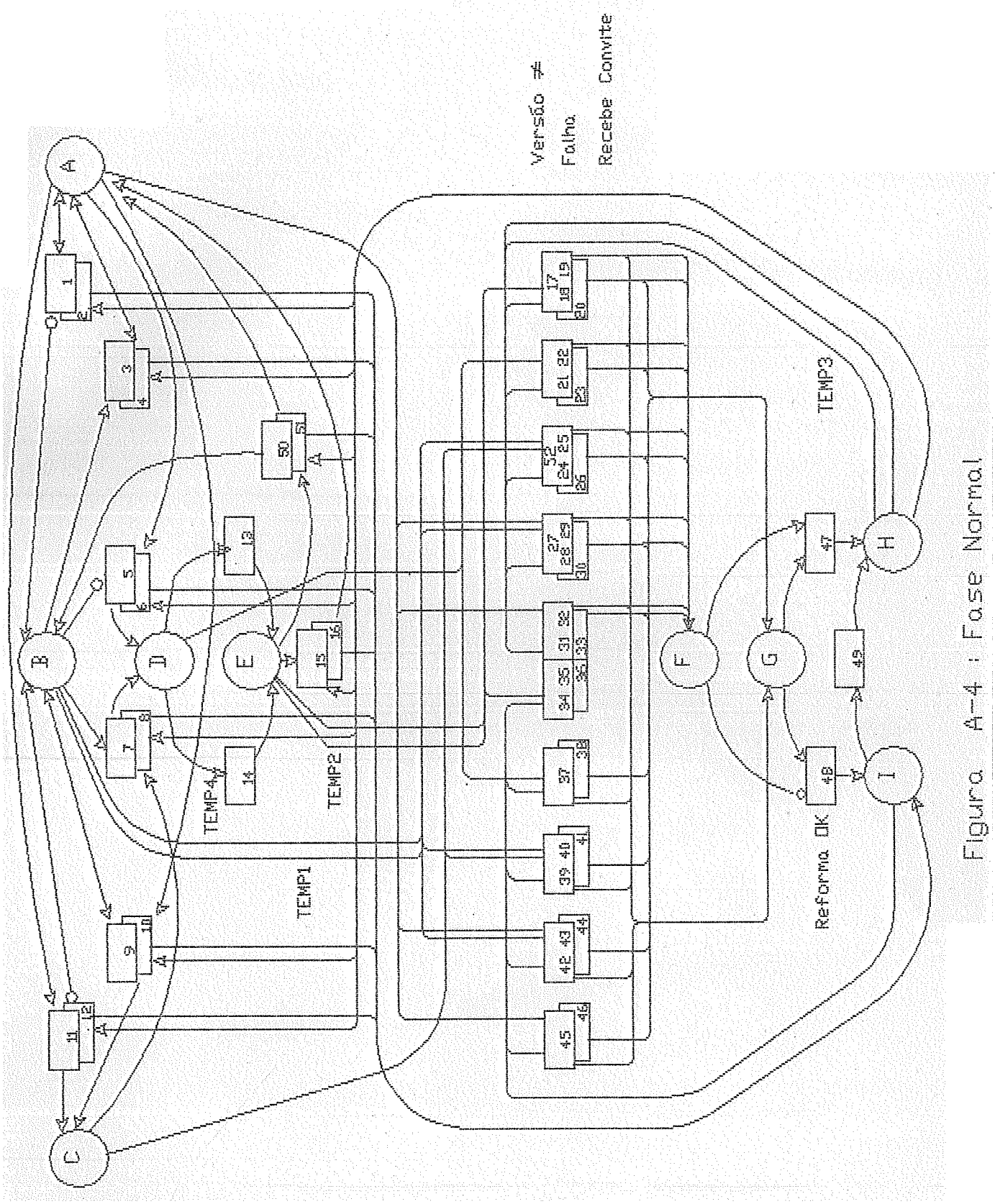


Figura 4-4 : Fase Normal

a - Lugares

(A) Sem Ficha.

Este Lugar representa a recepção de mensagens por uma Estação sem Ficha. As operações são:

```
(*Recebe Ack novo que não cria Buracos*)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e
     (Msg.Fmt=Ack_Dados ou Msg.Fmt=Ack_Nulo) e
     (Esta_Estação<>M.Est_com_Ficha(PCT.Val()) e
     (Msg.CT>=PCT.Val()) e (Qc.Status_Ack(Msg.CT)=Falta_Ack) e
     (Não_tem_Buracos=Q.Verifica_Buraco(Msg)) e
     ((Msg.Id.Est <> Esta_Estação) ou
 ((Msg.Id.Est=Esta_Estação)e(Msg.Id.MEi<>M.Val(Esta_Estação))))
     Então (*Insere o Ack*)
            Qc.Inserir(Msg.CT,Msg.Id);
           (*Trata Qc*)
           Enquanto (Qc.Status_Ack(PCT.Val())<>Falta_Ack)
                e (Qc.Status_Ack(PCT.Val())<>Falta_Msg ou
                Qb.Existe(Qc.Mensagem(PCT.Val())))
                Faça Caso Qc.Status_Ack(PCT.Val())
                       Pronto_sem_Msg:
                       Pronto_com_Msg:
                       Falta_Msg:
                          Qb.Remover(Qc.Mensagem(PCT.Val()));
                          Qc.Associar(Qc.Mensagem(PCT.Val()));
                          M.Inc(Qc.Mensagem(PCT.Val()).Est)
                     PCT.Inc();
```

```
(*Recebe Ack novo que não tapa todos os Buracos enquanto
 recupera mensagens*)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e
     (Msg.Fmt=Ack_Dados ou Msg.Fmt=Ack_Null) e
     (Esta_Estação<>M.Est_com_Ficha(PCT.Val()) e
     (Msg.CT>=PCT.Val()) e (Qc.Status_Ack(Msg.CT)=Falta_Ack) e
     (Tem_Buracos=Q.Verifica_Buraco(Msg)) e
     ((Msg.Id.Est <> Esta_Estação) ou
 ((Msg.Id.Est=Esta_Estação)e(Msg.Id.MEi<>M.Val(Esta_Estação))))
     Então (*Insere o Ack*) ...
          (*Trata Qc*) ...
          (*Requisita retransmissão de outra mensagem*)
          Desarma Temp1;
          Msg.Fmt=Q.Falta(Msg.CT);Msg.Ver:=Ver.Val();Ptx(Msg);
          Arma Temp1;

(*Recebe Dados novos que não tapam todos os Buracos*)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e (Msg.Fmt = Dados) e
     (não Qb.Existe(Msg.Id) e
     (Qc.Status_Msg(Msg.Id)<>Pronto_com_Msg) e
     (Tapa_Buracos<>Q.Verifica_Buraco(Msg))
     Então (*Trata Dados*)
          Caso Qc.Status_Msg(Msg.Id)
               Pronto_com_Msg:
               Falta_Ack :Qb.Inserir(Msg.Id)
               Falta_Msg :Qc.Associar(Msg.Id);
                         (*Trata Qc*) ...

(*Recebe Dados  novos que  não tapam  todos os Buracos enquanto
 recupera mensagens*)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e (Msg.Fmt = Dados) e
     (não Qb.Existe(Msg.Id) e
     (Qc.Status_Msg(Msg.Id)<>Pronto_com_Msg) e
     (Tapa_Buracos<>Q.Verifica_Buraco(Msg))
     Então (*Trata Dados*) ...
          (*Requisita retransmissão de outra mensagem*) ...
```

```
(*Usuário ativa Rx*)
Se Rx e Qc.Existe_Algum()
     Então retorna Rx com Qc.Remover();
```

(B) Recuperando Mensagens

Este Lugar é temporizado por Temp1 para aguardar a recuperação das Mensagens de Dados ou Acks. Se Temp1 se esgotar pela R-ésima vez, será iniciada a Reforma. As operações são:

```
Se (Esgotou Temp1) e (não foi a R-ésima vez)
     Então (*Requisita retransmissão de mensagem*)
          Msg.Fmt=Q.Falta(Msg.CT);Msg.Ver:=Ver.Val();Ptx(Msg);
          Arma Temp1;
```

(C) Com Ficha Recuperando Mensagens.

```
(*Recebe Ack novo que não tapa todos os Buracos enquanto
recupera mensagens*) ...

(*Recebe Dados novos que não tapam todos os Buracos enquanto
recupera mensagens*) ...

(*Usuário ativa Rx*) ...
```

(D) Com Ficha

Este Lugar é temporizado por Temp4 para aguardar a chegada de uma Mensagem de Dados que permita a passagem da Ficha. Se Temp4 se esgotar e houver mensagens de Dados para comissionamento, a Ficha será passada por um Ack_Nulo. Se não houver mais mensagens de Dados para comissionamento a posse da Ficha será confirmada por uma mensagem de Confirmação e a Marca continuará neste Lugar sem armar o temporizador Temp4.

```
(*Usuário ativa Rx*) ...
```

```
(*Recebe Dados velhos*)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e (Msg.Fmt=Dados) e
     (Pronto_com_Msg=Qc.Status_Msg(Msg.Id))
     Então (* Repete o Ack anterior*)
          Caso  Qc.Status_Ack(PCT.Val()-1)
               Falta_Msg:Msg.Fmt:=Ack_Dados;
                    Msg.Id:=Qc.Mensagem(PCT.Val()-1)
               Pronto_sem_Msg:Msg.Fmt:=Ack_Nulo
               Pronto_com_Msg:Msg.Fmt:=Ack_Dados;
                    Msg.Id:=Qc.Mensagem(PCT.Val()-1)
          Msg.CT:=PCT.Val()-1;Msg.Ver:=Ver.Val();Ptx(Msg)

Se (esgotou Temp4) e
     (Qc.Status_Ack(PCT.Val()-n)=Pronto_sem_Msg e ...)n=1...L
     Então (*Envia Confirmação*)
          Msg.Fmt:=Confirmação;Msg.CT:=PCT.Val();
          Msg.Ver:=Ver.Val();Ptx(Msg)

(* Recebe Ack velho e já transmitiu a confirmação*)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e
     (Msg.Fmt=Ack_Dados ou Msg.Fmt=Ack_Null) e
     (Msg.CT<PCT.Val()) e
     "já transmitiu a Confirmação"
     Então (* Envia a Confirmação*)...

(* Recebe Ack velho e não transmitiu a Confirmação*)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e
     (Msg.Fmt=Ack_Dados ou Msg.Fmt=Ack_Null) e
     (Msg.CT<PCT.Val()-1)
     "não transmitiu a Confirmação"
     Então (* Repete o Ack anterior*)
          Caso  Qc.Status_Ack(PCT.Val()-1)
               Falta_Msg:Msg.Fmt:=Ack_Dados;
                    Msg.Id:=Qc.Mensagem(PCT.Val()-1)
               Pronto_sem_Msg:Msg.Fmt:=Ack_Nulo
               Pronto_com_Msg:Msg.Fmt:=Ack_Dados;
                    Msg.Id:=Qc.Mensagem(PCT.Val()-1)
           Msg.CT:=PCT.Val()-1;Msg.Ver:=Ver.Val();Ptx(Msg)
```

```
(* Recebe Requisição de Ack*)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e (Msg.Fmt=Req_Ack)
     Então Caso Qc.Status_Ack(Msg.CT)
                Pronto_sem_Msg:Msg.Fmt:=Ack_Nulo
                Pronto_com_Msg:Msg.Fmt:=Ack_Dados;
                     Msg.Id:=Qc.Mensagem(Msg.CT)
           Msg.CT:=Msg.CT;Msg.Ver:=Ver.Val();Ptx(Msg)

(*Recebe Requisição de Dados*)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e (Msg.Fmt=Req_Dados)
     Então Msg.Fmt:=Dados;Msg.Id:=Qc.Mensagem(Msg.CT)
          Msg.Ver:=Ver.Val();Ptx(Msg)
```

(E) Passando Ficha

Este lugar é temporizado por Temp2 para aguardar a confirmação da passagem da Ficha. Se Temp2 se esgotar, será iniciada a Reforma.

```
(*Recebe Ack velho*)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e
     (Msg.Fmt=Ack_Dados ou Msg.Fmt=Ack_Null) e
     (Msg.CT<PCT.Val())
     Então (* Repete o próprio Ack*)
          Caso  Qc.Status_Ack(PCT.Val())
               Falta_Msg:Msg.Fmt:=Ack_Dados;
                    Msg.Id:=Qc.Mensagem(PCT.Val())
               Pronto_sem_Msg:Msg.Fmt:=Ack_Nulo
               Pronto_com_Msg:Msg.Fmt:=Ack_Dados;
                    Msg.Id:=Qc.Mensagem(PCT.Val())
           Msg.CT:=PCT.Val();Msg.Ver:=Ver.Val();Ptx(Msg)

(*Recebe o próprio Ack *)
Se (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e
     (Msg.Fmt=Ack_Dados ou Msg.Fmt=Ack_Null) e
     (Msg.CT=PCT.Val()) e
     (Qc.Status_Ack(PCT.Val())=Falta_Msg)
     Então Qb.Remover(Qc.Mensagem(PCT.Val()));
          Qc.Associar(Qc.Mensagem(PCT.Val()));
          M.Inc(Qc.Mensagem(PCT.Val()).Est)
```

```
(*Recebe Dados novos que não tapam todos os Buracos*) ...

(*Recebe Dados velhos*) ...

(*O usuário ativa Rx*) ...

(*Recebe Requisição de Ack*) ...

(*Recebe Requisição de Dados*) ...

Se (Esgotou Temp2) e (não foi a R-ésima vez)
     Então (*Repete o próprio Ack*) ...
          Arma Temp2;
```

(F) Mensagem_Pendente

Este Lugar serve para marcar se a Reforma foi iniciada durante a transmissão de uma mensagem de Dados que permanece sem confirmação.

(G) Reforma

Este Lugar representa toda a Fase de Reforma do Protocolo.

(H) Transmissão_Ativa

Este Lugar é temporizado por Temp3 para aguardar um Ack que confirme a transmissão realizada. Se Temp3 se esgotar pela R-ésima vez, será iniciada a Reforma.

```
Se (Esgotou Temp3) e (não foi a R-ésima vez)
     Então (*Transmite a mensagem do Usuário*)
          Msg.Fmt=Dados;Msg.Id.Est:=Esta_Estação;
          Msg.Id.MEi:=M.Val(Esta_Estação);Msg.Ver:=Ver.Val();
          Ptx(Msg);Arma Temp3;
```

(I) Transmissão Inativa

Este Lugar indica que a última transmissão ativada pelo usuário foi concluída e que o processo de transmissão está inativo.

```
b - Transições
(1) Recebe Ack novo que cria Buracos
Se   (*Recebe Ack novo*)
     (Msg:=Prx()) e(Msg.Ver=Ver.Val()) e
     (Msg.Fmt=Ack_Dados ou Msg.Fmt=Ack_Null) e
     ((Msg.CT>=PCT.Val()) e(Qc.Status_Ack(Msg.CT)=Falta_Ack)) e
     (*Não transfere Ficha para esta Estação*)
     (Esta_Estação<>M.Est_com_Ficha(PCT.Val())) e
     (*Cria_Buracos*)
     (Cria_Buracos=Q.Verifica_Buraco(Msg)) e
     (*Confirma Dados de outra Estação*)
     ((Msg.Id.Est <> Esta_Estação) ou
 ((Msg.Id.Est=Esta_Estação)e(Msg.Id.MEi<>M.Val(Esta_Estação))))
     Então (*Insere o Ack*) ...
          (*Trata Qc*) ...
          (*Requisita retransmissão de mensagem*) ...
```

(2) (4) (6) (8) (10) (12) (16) (51) são Transições semelhantes respectivamente a (1) (3) (5) (7) (9) (11) (15) (50). A diferença entre ambas é que o Predicado (*Confirma Dados de outra Estação*) existente nas primeiras é alterado nas últimas para: (*Confirma Dados desta Estação*)
(Msg.Id.Est=Esta_Estação)e(Msg.Id.MEi=M.Val(Esta_Estação)).
Além disso, é acrescida uma operação: Desarma Temp3.

```
(3) Recebe Dados ou Ack novos que tapam Buracos
Se ((*Recebe Ack novo*)... e
     (*Não transfere Ficha para esta Estação*)... e
     (*Confirma Dados de outra Estação*) ...)
     ou
     ((*Recebe Dados novos*)
      (Msg:=Prx())e(Msg.Ver=Ver.Val())e(Msg.Fmt=Dados)e
      (não Qb.Existe(Msg.Id))e
      (Qc.Status_Msg(Msg.Id)<>Pronto_com_Msg))
     e (*Tapa Buracos*)
     (Tapa_Buracos=Q.Verifica_Buraco(Msg))
```

```
     Então Caso Msg.FmT
               Ack_Nulo ou Ack_Dados:   (*Insere o Ack*) ...
                                        (*Trata Qc*) ...
               Dados: (*Trata Dados*) ...
          Desarma Temp1;
```

(5) Recebe Ack novo que mantém sem Buracos e transfere a Ficha para esta Estação.

```
Se   (*Recebe Ack novo*) ...e
     (*Transfere Ficha para esta Estação*)
     (Esta_Estação=M.Est_com_Ficha(PCT.Val())) e
     (*Não Tem Buracos*)
     (Não_tem_Buracos=Q.Verifica_Buraco(Msg)) e
     (*Confirma Dados de outra Estação*) ....e
     Então (*Insere o Ack*) ...
          (*Trata Qc*) ...
          Arma Temp4;
```

(7) Recebe Dados ou Ack novos que tapam Buracos da Estação com Ficha

```
Se ((*Recebe Ack novo*) ... e
     (*Não transfere Ficha para esta Estação*) ... e
     (*Confirma Dados de outra Estação*) ...)
     ou
     ((*Recebe Dados novos*) ...)
     e (*Tapa Buracos*) ...
     Então Caso Msg.FmT
               Ack_Nulo ou Ack_Dados:   (*Insere o Ack*) ...
                                        (*Trata Qc*) ...
               Dados: (*Trata Dados*) ...
          Desarma Temp1;
          Arma Temp4;
```

(9) Recebe Ack novo que mantém Buracos e transfere a Ficha para esta Estação.

```
Se   (*Recebe Ack novo*) ...e
     (*Transfere Ficha para esta Estação*) ...e
     (*Tem Buracos*)
     (Tem_Buracos=Q.Verifica_Buraco(Msg)) e
     (*Confirma Dados de outra Estação*) ....e
     Então (*Insere o Ack*) ...
          (*Trata Qc*) ...
          (*Requisita retransmissão de mensagem*) ...
```

(11) Recebe Ack novo que cria Buracos e transfere a Ficha para esta Estação.

```
Se   (*Recebe Ack novo*) ...e
     (*Transfere Ficha para esta Estação*) ...
     (*Cria Buracos*) ...
     (*Confirma Dados de outra Estação*) ....e
     Então (*Insere o Ack*) ...
          (*Trata Qc*) ...
          (*Requisita retransmissão de mensagem*) ...
```

(13) Recebe Dados novos Transfere Ficha com Ack_Dados

```
Se (*Recebe Dados novos*)... ou
   (Qb.Existe_Algum())
     Então Se Qb.Existe_Algum()
           Então Msg.Id:=Qb.Qualquer()
           Senão Qb.Inserir(Msg.Id)
           Msg.Fmt:=Ack_Dados;Msg.CT:=PCT.Val();
           Msg.Ver:=Ver.Val();Ptx(Msg);
           (*Insere o Ack*) ...
           Arma Temp2;
```

(14) Transfere Ficha com Ack_Nulo

```
Se (Esgotou Temp4) e
     (Qc.Status_Ack(PCT.Val()-n)=Pronto_com_Msg ou ...)n=1...L
     Então Msg.Fmt:=Ack_Nulo;Msg.CT:=PCT.Val();
           Msg.Ver:=Ver.Val();Ptx(Msg);
           (*Insere o Ack*) ...
           Arma Temp2;
```

```
(15) Recebe Confirmação ou Ack que confirma a passagem da Ficha
 sem criar Buracos.
Se   ((*Recebe Ack novo*) ...e
     (*Não Tem Buracos*) ...e
     (*Confirma Dados de outra Estação*) ...)
     ou
     ((*Recebe Confirmação*)
     (Msg:=Prx()) e (Msg.Ver=Ver.Val()) e
     (Msg.Fmt=Confirmação) e
     (Msg.CT>PCT.Val())
      Então Caso Msg.Fmt
                Ack_Null ou Ack_Dados:   (*Insere o Ack*) ...
                                         (*Trata Qc*) ...
                Confirmação: PCT.Inc();
           Desarma Temp2;

(17)(21)(24)(27)(31)(34)(37)(39)(42) e (45) Versão Errada.
Corresponde à  Transição  (2) da Seção Mestre do Protocolo e à
Transição (15) da Seção Escravo.
Se (Msg:=Prx()) e (Msg.Ver<>Ver.Val()) e (Msg.Fmt<>Convite)
     Então;

(18)(22)(25)(28) e (32) Falha na Transmissão.
Corresponde à  Transição  (2) da Seção Mestre do Protocolo e à
Transição (15) da Seção Escravo.
Se Esgotou Temp3 pela R_ésima vez
     Então;

(19) e (35) Falha na Passagem da Ficha.
Corresponde à  Transição  (2) da Seção Mestre do Protocolo e à
Transição (15) da Seção Escravo.
Se Esgotou Temp2 pela R_ésima vez
     Então;

(20)(23)(26)(30)(33)(36)(38)(41)(44) e (46) Recebe Convite.
Corresponde à  Transição (14)  da Seção Mestre do Protocolo e à
Transição (1) da Seção Escravo do Protocolo.
```

```
(29)(40)(43) e (52) Falha na Requisição de Retransmissão.
Corresponde à  Transição  (2) da Seção Mestre do Protocolo e à
Transição (15) da Seção Escravo.
Se Esgotou Temp1 pela R_ésima vez
     Então;

(48) e (47) Reforma OK.
Corresponde às Transições (12) e (14) da Seção Escravo.

(49) Usuário ativa Tx
Se (Tx)
     Então (*Transmite a mensagem do Usuário*) ...

(50) Recebe  Ack que  confirma  a  passagem  da  Ficha  criando
     Buracos.
Se   (*Recebe Ack novo*) ...e
     (*Cria Buracos*) ...e
     (*Confirma Dados de outra Estação*) ...
      Então (*Insere o Ack*) ...
            (*Trata Qc*) ...
          Desarma Temp2;
```

GLOSSARIO DE NOTAÇÕES, SIGLAS E VARIAVEIS

$1\_<>\_N$ Relacionamento "1 para N" em um Modelo de Entidades e Relacionamentos de um Banco de Dados.

A Conjunto de arestas de um Grafo de Serialização.

Abou Mensagem que aborta Reforma do Protocolo de Difusão Confiável.

Ack_Dados Mensagem que confirma a recepção de uma mensagem de Dados do Protocolo de Difusão Confiável.

Ack_Convite Mensagem que confirma a recepção de uma mensagem de Convite do Protocolo de Difusão Confiável.

Ack_Nulo Mensagem que confirma a recepção da Ficha do Protocolo de Difusão Confiável.

Ack_Novo_Grupo Mensagem que confirma a recepção de uma nova versão para o Grupo de Difusão do Protocolo de Difusão Confiável.

ALR Entidade "Alarme" do Banco de Dados do Processo.

BD Réplica do Banco de Dados.

BD_{EiBDRj} Réplica do j-ésimo BDR na Estação i.

BDP Banco de Dados do Processo.

BDP* Banco de Dados do Processo estimados.

BDRj j-ésimo Banco de Dados Replicados.

BE Bloqueio para Escrita.

BEL Bloqueio para Escrita e Leitura.

BN Sem Bloqueio.

Confirmação Mensagem que confirma a recepção da Ficha do Protocolo de Difusão Confiável.

Convite Mensagem de Convite para iniciar uma reforma do Protocolo de Difusão Confiável.

CT Número de ordem global (Carimbo de Tempo) das mensagens de dados transmitidas por Difusão.

CSMA/CD Controle de Acesso ao Meio Físico do tipo "Carrier Sense Multiple Access with Collision Detection".

Dados Mensagem de Dados do Protocolo de Difusão Confiável.

DC Protocolo de Difusão Confiável.

DD É o conjunto de estados do contexto de uma Rede de Petri Interpretada.

Dk k-ésimo Depósito de Informações.

Ei i-ésima Estação.

Escrever[] Operação de escrita de um Item do Banco de Dados.

Est Identificador de uma Estação.

F : Tr -> PR x OP, é uma função que associa a cada transição da Rede uma dupla (pr_i, op_i).

$GA_{E_iBDR_j}$ Gerenciador de Acesso Concorrente do j-ésimo BDR na i-ésima Estação.

GBDR Gerenciador de Banco de Dados Replicados.

GBDD Genrenciador de Banco de Dados Distribuídos.

GD_{E_i} Gerenciador dos Dados da i-ésima Estação.

GS(Log) Grafo de Serialização de um Log.

GT_{E_i} Gerenciador de Transações da Estação i.

Habilita_Novo_Grupo Mensagem para encerrar a Reforma do Protocolo de Difusão Confiável.

$IR_{E_iBDR_j}$ Iniciador/Recuperador da Réplica do j-ésimo BDR na i-ésima Estação.

Im m-ésimo Item de Dado.

L Grau de Resiliência de um Grupo de Difusão.

Ler[] Operação de leitura de um Item do Banco de Dados.

Lg Conjunto de Lugares de uma Rede de Petri.

Log Conjunto parcialmente ordenado das Leituras e Escritas escalonadas por um Gerenciador de Acesso.

L_{T_n} Lista de Finalização da n-ésima Transação.

MEi Número de ordem local (interno a uma Estação) das mensagens transmitidas por Difusão.

M[] Relação das Estações ativas em um determinado Grupo de Difusão com os respectivos números de ordem local de cada Estação.

n Número máximo de Estações num Grupo de Difusão.

$N_{-<>-M}$ Relacionamento "N para M".

Novo_Grupo Mensagem de Novo Grupo obtido na Reforma do Protocolo de Difusão Confiável.

O Ordem parcial em S.

On Ordem parcial em Sn.

OP = { op_1, op_2,...,op_s} é um conjunto de operadores;

op_i: DD -> DD.

PCT Próximo CT esperado por uma Estação em determinado instante.

Pós É uma função de Tr em Lg que indica os arcos que ligam as Transições aos Lugares.

PR = {pr_1, pr_2,...,pr_s} é um conjunto de predicados sobre DD; pr_1: DD -> {verdadeiro, falso}.

Pré É uma função de Lg em Tr que indica os arcos que ligam os Lugares às Transições.

Prx Interface entre o DC e a recepção de difusões.

Ptx Interface entre o DC e a transmissão por difusão física.

Qb Fila de mensagens de dados não confirmadas.

Qc Fila de controle das confirmações recebidas.

Q_1 Fila para Prx.

Q_0 Fila para Ptx.

Rej_Convite Mensagem de rejeição do convite para Reforma do Protocolo de Difusão Confiável.

Req_Ack Mensagem de pedido de retransmissão de um Ack_Dados do Protocolo de Difusão Confiável.

Req_Dados Mensagem de pedido de retransmissão de Dados do Protocolo de Difusão Confiável.

Rx Interface entre o usuário e a recepção do DC.

S Conjunto de escritas e leituras de todas as Transações.

Seq Número de seqüência das versões do Grupo de Difusão.

Sn Conjunto de leituras e escritas da n-ésima Transação.

SR Propriedade de um Log que é serializável.

Temp1 Temporizador para recuperação de mensagens de dados.

Temp2 Temporizador para recepção da confirmação da passagem da Ficha.

Temp3 Temporizador para recepção da confirmação transmissão de uma mensagens de dados.

Temp4 Temporizador para aguardar uma mensagen de dados que permita a passagem da Ficha.

Temp5 Temporizador do Mestre para aguardar a adesão dos Escravos durante a Reforma.

Temp6 Temporizador para o Mestre aguardar o início da fase II da Reforma.

Temp7 Temporizador para o Escravo aguardar o início da fase II da Reforma.

Temp8 Temporizador para a Estação com Ficha aguardar o início da fase III da Reforma.

Temp9	Temporizador para o Escravo aguardar a recuperação de mensagens.
TMC	Terminal de Medição e Controle.
EVE	Entidade "Evento" do Banco de Dados do Processo.
T	Conjunto de Transações.
Tn	Identificador da Transação para o GT.
Tr	Conjunto de Transições de uma Rede de Petri.
Tx	Interface entre o usuário e a transmissão do DC.
VA	Tabela de Variáveis Analógicas.
VB	Tabela de Variáveis Binárias.
Ver	Versão de um Grupo de Difusão.