UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FUNÇÕES DE GERENCIAMENTO PARA UM SISTEMA DISTRIBUÍDO APLICADO NA AUTOMAÇÃO DE USINAS E SUBESTAÇÕES

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA
PARA OBTENÇÃO DE GRAU DE MESTRE EM ENGENHARIA ELÉTRICA

ROGÉRIO SÉRGIO NEVES DE LEMOS

FLORIANÓPOLIS, ABRIL - 1988

FUNÇÕES DE GERENCIAMENTO PARA UM SISTEMA DISTRIBUÍDO APLICADO NA AUTOMAÇÃO DE USINAS E SUBESTAÇÕES

ROGÉRIO SÉRGIO NEVES DE LEMOS

ESTA DISSERTAÇÃO FOI JULGADA PARA A OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA - ESPECIALIDADE ENGENHARIA ELÉTRICA E APROVADA EM SUA FORMA FINAL PELO CURSO DE PÓS-GRADUAÇÃO.

Prof. Joni da Silva Fraga, Dr.Ing.
Orientador

Prof. Márcio Cherem Schneider, Dr.Sc.
Coordenador do Curso de Pós-graduação em Engenharia Elétrica

Banca Examinadora

Prof. Jean-Charles Valadier, Dr.

Prof. Jean-Marie Farines, Dr.Ing.

Eng. Homero Gonçalves de Andrade, M.Sc.

AGRADECIMENTOS

Ao professor Joni da Silva Fraga pela orientação e apoio na realização deste trabalho. Ao professor Jean-Marie Farines pela ajuda na modelização do algoritmo de diagnose por Redes de Petri. Ao Homero de Andrade pelo apoio fornecido durante as minhas visitas ao CEPEL.

Aos menbros da Banca Examinadora, pelos seus comentários e sugestões que enriqueceram o trabalho.

A todos os professores, colegas e funcionários, do Programa de Pós-Graduação em Engenharia Elétrica e do Laboratório de Controle e Microinformática (LCMI/DEEL), que de uma maneira de outra contribuiram para a realização deste trabalho.

A UFSC, ao CNPq e ao CEPEL pelo apoio financeiro que permitiu a realização deste trabalho.

SUMÁRIO

RESUMO

Este trabalho apresenta as funções de gerenciamento para um sistema digital de arquitetura distribuída, que está sendo desenvolvido pelo CEPEL com a finalidade de automatizar usinas e subestações. As especificações iniciais desenvolvidas estão baseadas numa arquitetura sem redundâncias que inicialmente objetiva a obtenção de um conhecimento efetivo do comportamento do sistema. Na primeira parte é apresentado um estudo introdutório na área de Segurança de Funcionamento, tendo como objetivos: primeiro, apresentar uma terminologia e conceituação básica, e segundo, apresentar um conjunto de técnicas de tolerância a faltas. Na segunda parte, o sistema digital de arquitetura distribuída é examinado em termos da arquitetura do sistema e sua funcionalidade. Na terceira parte, são discutidas as atividades de gerenciamento para o sistema em questão, juntamente com um modelo geral de gerenciamento. E por último, é abordada a diagnose de faltas no sistema distribuído, onde é descrito o algoritmo de diagnose sendo proposto.

ABSTRACT

This work presents the management functions for a distributed computer control system, which has been developed at CEPEL with the aim to automate Hydroeletric Power Plants and Extra-High Voltage Power Substations. The initial specifications developed are based on a architecture without redundancies which initially has the objective to get the effective knowledge of the system´s behaviour. In the first part is given an introduction of dependability, with two aims: to present the basic concepts and terminology, and, to present a set of techniques in the area of computing systems fault tolerance. In the second part of this dissertation, the distributed computer control system is introduced, in terms of its architecture and functionality. The third part introduces the management activities for the distributed system are introduced tgether with a general model. Finally, the fault diagnosis in the distributed system is discussed, where is presented a new diagnosis algorithm.

CAPÍTULO 1

INTRODUÇÃO

Nas últimas décadas, tem ocorrido um aumento substancial na aplicação de sistemas computacionais tolerantes a faltas nas mais diferentes áreas de aplicação, dentro das quais podemos destacar: automação industrial, controle de processos, aplicações aeroespaciais e centrais telefônicas. Estes sistemas se caracterizam por serem diferentes dos sistemas convencionais, pela alta disponibilidade e/ou confiabilidade que apresentam. Estas características são obtidas essencialmente pela adição de redundâncias, tanto a nível de hardware quanto de software, o que possibilita o fornecimento de um serviço próprio, mesmo na presença de faltas.

Os objetivos principais da aplicação de sistemas tolerantes a faltas em controle de processos e na automação industrial, são: prevenir danos ao processo, restaurar a operação o mais rápido possível, e melhorar a disponibilidade a longo prazo. O requisito de alta disponibilidade, está fundamentado na necessidade de prevenir a falha dos sistemas de controle, que podem causar perdas muitas vezes maiores que os custos do próprio sistema de controle envolvido.

A motivação inicial deste trabalho foi a realização de um estudo sobre a auto-diagnose e a viabilidade da sua implementação no Sistema Digital de arquitetura distribuída, sendo desenvolvido pelo CEPEL, com a finalidade de automatizar Usinas e Subestações (US/SE). A viabilidade da implementação de técnicas de tolerância a falta, levou à descrição de funções de gerenciamento para o referido sistema. É importante citar que o Sistema Digital está baseado numa arquitetura sem redundâncias, cujo o objetivo inicial é a obtenção de um conhecimento efetivo do comportamento deste sistema.

O Gerenciamento do Sistema Distribuído, classicamente tem como função o planejamento, organização, supervisão, contabilização e controle dos componentes do sistema em operação. Os componentes do sistema formam um conjunto de recursos para o processamento, armazenamento e comunicação da informação. O gerenciamento é um serviço essencial que tem como objetivo manter os sistemas operando efetivamente, para providenciar serviços aos usuários e possibilitar a evolução dos mesmos.

A diagnose de faltas se refere ao processo de determinar a localização de faltas num sistema. Uma diagnose correta de faltas é necessária, para possibilitar as ações de recuperação e/ou a reconfiguração do sistema. Essencialmente, a complexidade da diagnose está associada a dois fatores: a natureza do modelo de faltas assumido, e a extensão requirida para a diagnose. Num

ambiente distribuído, onde a diagnose de faltas do sistema tem que ser executada de forma descentralizada, é assumido que uma entidade com faltas é capaz de apresentar um comportamento arbitrário ou mesmo malicioso, e que as entidades livres de faltas têm que diagnosticar o estado operacional de todas as entidades do sistema.

A apresentação das atividades de gerenciamento é feita, primeiramente, em termos da descrição de um modelo geral para o gerenciamento, citando as várias funções que o compoêm, para posteriormente, abordar de forma mais detalhada, a diagnose de faltas no sistema distribuído. Neste último item é especificado um algoritmo descentralizado para a diagnose de faltas em ambientes distribuídos.

Este trabalho está dividido em seis capítulos. No segundo capítulo, é apresentado um estudo realizado sobre Segurança de Funcionamento, onde, além de se apresentar uma terminologia e uma conceituação básica, também foram apresentadas uma série de técnicas de tolerância a faltas aplicadas em sistemas informáticos.

No terceiro capítulo, é descrito o Sistema Digital de arquitetura distribuída, sendo desenvolvido pelo CEPEL, e que tem a finalidade de automatizar Usinas e Subestações. Esta descrição se restringe ao protótipo do sistema, atualmente em fase de instalação, em termos da sua arquitetura e funcionalidade.

No quarto capítulo, são abordadas as atividades de gerenciamento do Sistema Digital, que foram agrupadas, dentro do contexto global do sistema, numa função denominada de Função de Observação. Estas atividades são apresentadas em termos de um modelo geral para o Gerenciamento de Sistemas Distribuídos, com atributos de tempo real. Devido à complexidade envolvida, o Gerenciamento do Subsistema de Comunicação é visto mais detalhadamente, seguindo as padronizações internacionais atualmente emergentes, o Modelo de Referência OSI da ISO e o protocolo MAP.

No quinto capítulo, é abordada a diagnose de faltas em sistemas distribuídos. Inicialmente, é realizada uma apresentação de alguns modelos e algoritmos, normalmente referenciados na literatura, empregados na diagnose de faltas, para em seguida se realizar a descrição do algoritmo de diagnose sendo proposto, para o Sistema Digital. Esta descrição está baseada na modelagem da diagnose de faltas no sistema distribuído, e na formalização do algoritmo através de Redes de Petri.

No sexto capítulo, são apresentadas as conclusões finais deste trabalho, como também, recomendações para futuros trabalhos.

CAPÍTULO 2

SEGURANÇA DE FUNCIONAMENTO

2.1. Introdução

A crescente utilização de sistemas computacionais nos diferentes ramos da atividade humana tem implicado que cuidados sejam tomados para evitar que sistemas deixem de executar suas funções conforme especificado. Esta necessidade é essencial no sentido de evitar prejuízos materiais e humanos, segundo a aplicação sendo realizada.

A concepção de sistemas que apresentam qualificativos como 'seguros' ou 'confiáveis' exige disciplinas que determinam metodologias, técnicas de estruturação e incorporação de redundâncias. É neste sentido que se insere a segurança de funcionamento em sistemas informáticos.

Este capítulo se apresenta como um estudo introdutório da segurança de funcionamento, tendo como objetivos: primeiro, apresentação de uma terminologia e conceituação básica em segurança de funcionamento de sistemas computacionais, e segundo, apresentação de uma série de técnicas de tolerância a faltas que podem ser aplicadas a estes sistemas.

2.2. Terminologia e Conceituação Básica em Segurança de Funcionamento

A terminologia e a conceituação básica sobre segurança de funcionamento descritos nos itens subsequentes foi em grande parte influenciada por (Avizienis & Laprie, 1986). Cumpre salientar que além da referência citada, foram de fundamental importância para a composição de nomes e conceitos as publicações de (Anderson & Lee, 1981), (Avizienis, 1976), (Avizienis, 1978), (Laprie, 1985), (Randell et alii, 1978), e (Siewiorek, 1984). Neste trabalho de síntese também foram fatores importantes o uso corrente de determinados termos e a etimologia e características da língua portuguesa.

2.2.1. Conceito de Segurança de Funcionamento

O **serviço** fornecido por um sistema é uma abstração do seu comportamento, conforme percebido por outros sistemas (seus usuários) que interagem com o sistema em questão. A **especificação** do serviço é uma descrição bem definida do comportamento esperado do sistema. O **comportamento** é simplesmente que o sistema faz, enquanto a **estrutura** ou **organização** do sistema é que o habilita a ter um determinado comportamento.

A **segurança de funcionamento** está associada a 'qualidade' de um sistema informático de modo que se possa ter uma confiança justificável no serviço fornecido pelo sistema.

2.2.2. Noções de Falta, Erro e Falha: Definições e Propriedades

Faltas, erros e falhas são todas imperfeições e portanto circunstâncias indesejáveis ao comportamento do sistema:

- a **falha** é uma circunstância indesejável que afeta o serviço fornecido. É percebida pelo usuário e avaliada;

- o **erro** é uma circunstância indesejável interna ao sistema. Sendo parte de um estado do sistema (estado errôneo), o erro pode ser detectado;

- a **falta** no sentido fenomonológico, é a circunstância indesejável causadora do erro.

Um **erro** é uma manifestação de uma falta no sistema. Uma falha é o efeito de um erro no serviço do sistema. A criação e a manifestação das faltas, erros e falhas podem ser resumidos conforme segue:

- uma falta pode ser **dormente** ou **ativa**: uma falta dormente se tornará ativa a partir do momento em que passa a produzir um estado errôneo no sistema. Uma falta pode continuamente oscilar entre estes dois estados;

- um erro é parte de um estado do sistema (estado errôneo) que pode levar o sistema à falha. Erros podem ser **latentes** ou **detectados:** um erro é tido como latente quando se propaga pelo sistema, causando novos erros, até que seja detectado ou provoque uma saída errônea não especificada nos serviços do sistema (falha). Um erro pode ser originado pela ativação de uma falta dormente ou pela propagação de um erro no sistema.

Abaixo são apresentados exemplos que ilustram as relações de causa e efeito entre faltas, erros e falhas:

- Numa determinada posição de memória que tenha seu conteúdo inalterável, o efeito fenomenológico (distúrbio físico-químico) provoca uma falta dormente. Esta falta se torna ativa, por exemplo, quando de uma escrita que tenha como finalidade alterar o conteúdo da memória, produzindo assim, um estado errôneo no sistema. A cada leitura subsequente desta posição de memória, o erro latente se propaga pelo sistema até que este seja detectado ou cause uma saída não especificada, provocando uma falha.

- Um engano do programador na codificação de uma instrução cria uma falta dormente. Esta falta se torna ativa a partir do momento em que a instrução é executada, criando-se assim um estado errôneo no sistema. Este estado vai produzindo erros latentes até que estes sejam detectados ou produzam uma falha no sistema.

2.2.2.1. Classificação das Faltas

As faltas que ocorrem num sistema, são classicamente divididas em dois grupos (Avizienis, 1978): faltas físicas e faltas humanas.

1. **Faltas Físicas**

   São consequências de fenômenos tanto internos (desordens físicas e químicas) como externos (perturbações do ambiente) que afetam o sistema. As faltas físicas de um sistema podem ser identificadas pela duração, valor e extensão da falta:

   - **Duração:** propriedade relacionada com o tempo de ocorrência de uma falta.

     a. **Faltas Permanentes** - têm sua causa em mudanças irreversíveis e de duração ilimitada de fenômenos internos ao sistema.

     b. **Faltas Transientes** - têm sua causa em mudanças reversíveis e de duração limitada de interferências externas ao sistema.

     c. **Faltas Pseudo-Transientes** - têm sua causa em mudanças irreversíveis, e seus efeitos somente aparecem sobre certas condições em alguns

instantes. Estas faltas são difíceis de serem detectadas.

- **Valor:** propriedade que determina se a falta produz um comportamento errôneo de valor fixo ou variável no sistema.

  a. **Faltas Determinantes** - são aquelas cujos erros apresentam um valor inalterado no tempo.

  b. **Faltas Indeterminantes** - são aquelas que possibilitam às variáveis afetadas a alteração dos seus valores no tempo.

- **Extensão:** Determina a quantidade de variáveis lógicas que são simultaneamente afetadas por um única falta física.

  a. **Faltas Locais** - são aquelas que afetam somente variáveis singulares

  b. **Faltas Distribuídas** - são aquelas que afetam duas ou mais variáveis, um subsistema, ou um sistema inteiro.

2. **Faltas Humanas**

São consequência de ações humanas inadevertidas ou deliberadas. Estas faltas são agrupadas em:

a. **Faltas de Projeto** - são faltas cometidas durante as várias fases de concepção de um sistema.

b. **Faltas de Interação** - são violações (inadvertidas ou deliberadas) nos procedimentos de operação, utilização e manutenção do sistema.

As faltas físicas podem somente afetar diretamente os componentes físicos, enquanto que as faltas humanas podem afetar qualquer componente.

2.2.2.2. Classificação das Falhas

Dependendo do sistema, podem existir dois tipos de falha, que determinam dois estados antagônicos do ponto de vista do serviço fornecido:

- nas **falhas benignas**, as consequências são da mesma ordem de magnitude se comparadas com o serviço fornecido na ausência de falhas. Os erros que causam estas falhas podem ser tratados pelas técnicas clássicas de tolerância a faltas;

- nas **falhas malignas** ou **catastróficas**, as consequências não são comparáveis com as do serviço fornecido na ausência de falhas. Os erros envolvidos nestas falhas podem ser não recuperáveis (ex: um erro não detectável), e portanto, afetando a qualidade do serviço.

Na vida de um sistema existem dois estados do serviço fornecido com respeito ao serviço especificado:

- **serviço próprio:** o serviço é fornecido conforme especificado;

- **serviço impróprio:** o serviço fornecido é diferente do especificado.

Os eventos responsáveis pela a transição entre estes dois estados são a falha e a restauração do serviço, conforme representado na figura 2.1.

2.2.3. Meios para a Segurança de Funcionamento

O projeto e a realização de um sistema computacional seguro de funcionamento, passa pela utilização combinada de métodos e ferramentas que são agrupados em meios para obtenção e meios para validação da segurança de funcionamento.

2.2.3.1. Meios para Obtenção da Segurança de Funcionamento

Os meios para obtenção de um sistema seguro de funcionamento tratam com a necessidade de prover ao sistema, a habilidade de fornecer o serviço especificado. Isto é conseguido através de um conjunto de métodos classificados em:

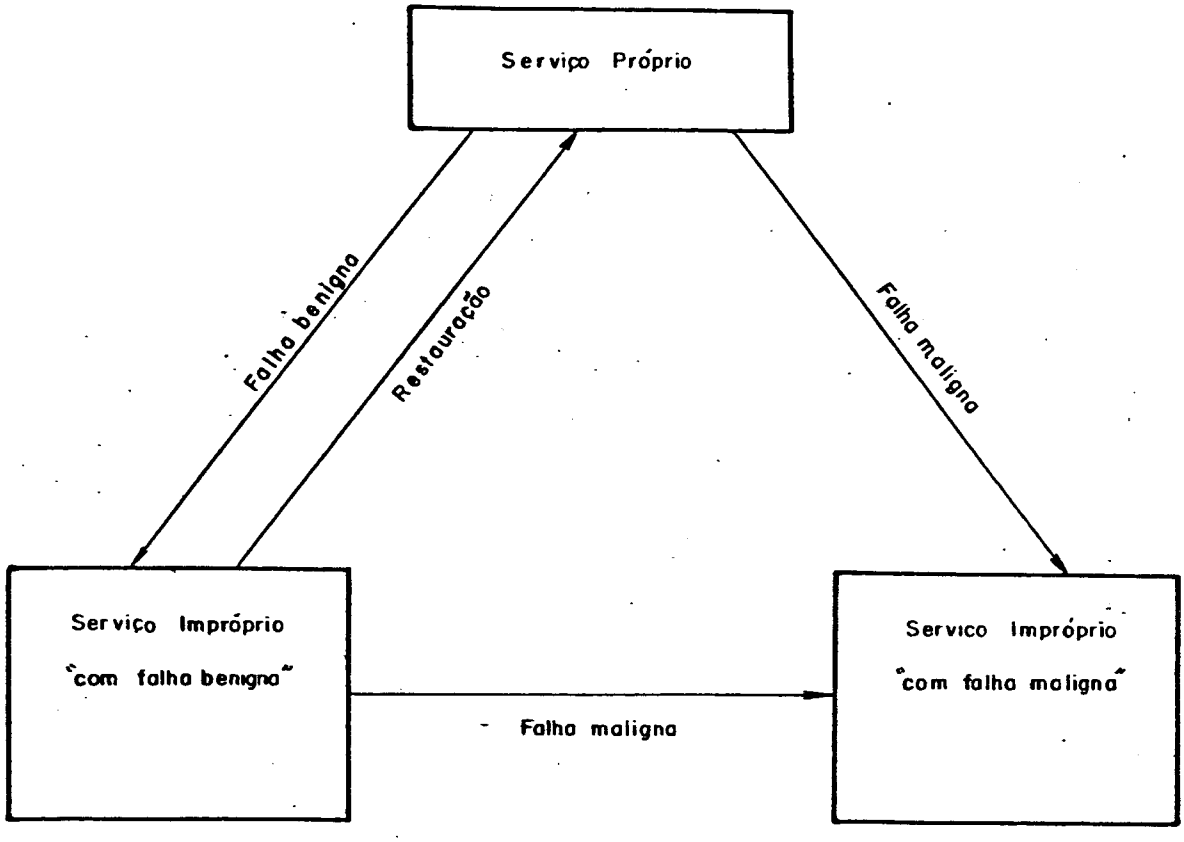


Figura 2.1. Estados do Serviço Fornecido com Respeito ao Serviço Especificado

- **prevenção a faltas:** está relacionado à prevenção, por construção, da ocorrência de faltas durante o projeto e implementação do sistema. Onde por construção se subentende a adoção de metodologias de projeto e de seleção de técnicas de implementação que objetivam a não ocorrência de faltas;

- **tolerância a faltas:** está relacionado ao fornecimento, pela redundância, de um serviço próprio apesar da presença de faltas. O grau de tolerância a faltas depende do sucesso com que estados errôneos são detectados e recuperados, e o sucesso com que as entidades com falta são identificadas e reparadas.

2.2.3.2. Meios para Validação da Segurança de Funcionamento

Os meios para a validação da segurança de funcionamento permitem que se tenha confiança na habilidade do sistema em fornecer o serviço especificado. Isto é conseguido através de duas classes de métodos:

- **remoção de faltas:** está relacionada com a minimização da presença de faltas, pelo o uso de métodos de verificação.

- **previsão de faltas:** está relacionada com a estimação da ocorrência e da consequência das faltas, usando métodos de avaliação.

A **cobertura** é medida da representatividade das situações nas quais o sistema é submetido durante a sua validação, em relação às situações reais nas quais será confrontado durante sua vida operacional.

É pela utilização conjunta dos quatro métodos mencionados acima: tolerância, prevenção, remoção e previsão de faltas, que se pode alcançar um sistema computacional seguro no funcionamento.

2.2.4. Medidas da Segurança de Funcionamento

Um conjunto de medidas é introduzido para quantificar a alternância entre os estados de **serviço próprio** e **serviço impróprio.** Como as circunstâncias que afetam a segurança de funcionamento não são deterministas, variáveis aleatórias são associadas a estes estados e estas medidas portanto, passam a ser probabilidades. Existem dois métodos para a quantificação destas medidas (Avizienis, 1976):

- **método analítico:** as medidas são obtidas através de modelos matemáticos do sistema;

- **método experimental:** erros são inseridas no modelo simulado do sistemas, e as medidas são estimadas a partir de dados estatísticos.

As medidas mais difundidas na literatura, no que concerne a segurança de funcionamento são:

a. **Confiabilidade** - é a probabilidade de um sistema executar continuamente um serviço próprio durante um período de tempo, de um instante inicial até a ocorrência de uma falha; a confiabilidade mede a permanência do sistema no estado de serviço próprio.

b. **Disponibilidade** - é a probabilidade de um sistema executar o serviço especificado quando requisitado. Esta medida leva em consideração a alternância entre os estados de serviço próprio e serviço impróprio.

c. **Taxa de Manutenção** - é a probabilidade, após uma falha, do sistema voltar a funcionar, ou equivalentemente, o tempo para a reabilitação do sistema.

d. **Segurança** - é a probabilidade de se evitar eventos catastróficos. Corresponde à medida de permanência no estado seguro-estado resultante do agrupamento dos estados de serviço próprio e serviço impróprio subsequente a falhas benignas.

Os conceitos de confiabilidade e segurança, apresentam uma certa analogia; enquanto a confiabilidade se preocupa com o tempo em que o sistema permanece no estado de serviço próprio, a segurança se preocupa somente com as situações de risco (falhas

malignas) e portanto, com o tempo em que o sistema permanece no estado de serviço seguro.

2.2.5. Síntese Conclusiva da Taxonomia

Uma representação gráfica (em árvore) da taxonomia da segurança de funcionamento é representada conforme figura 2.2, resumindo as definições apresentadas anteriormente, agrupando-as em três classes de atributos distintos:

a. as **imperfeições** da segurança de funcionamento, são as circunstâncias indesejáveis que fazem com que um sistema deixe de ser seguro no seu funcionamento;

b. os **meios** da segurança de funcionamento, são os métodos, ferramentas e técnicas que possibilitam:
   - prover ao sistema a habilidade de fornecer um serviço na qual confiança pode ser depositada,
   - ter confiança nesta habilidade;

c. as **medidas** da segurança de funcionamento permitem que a qualidade dos serviços seja avaliada na presença das imperfeições e dos meios da segurança de funcionamento, descritos acima.

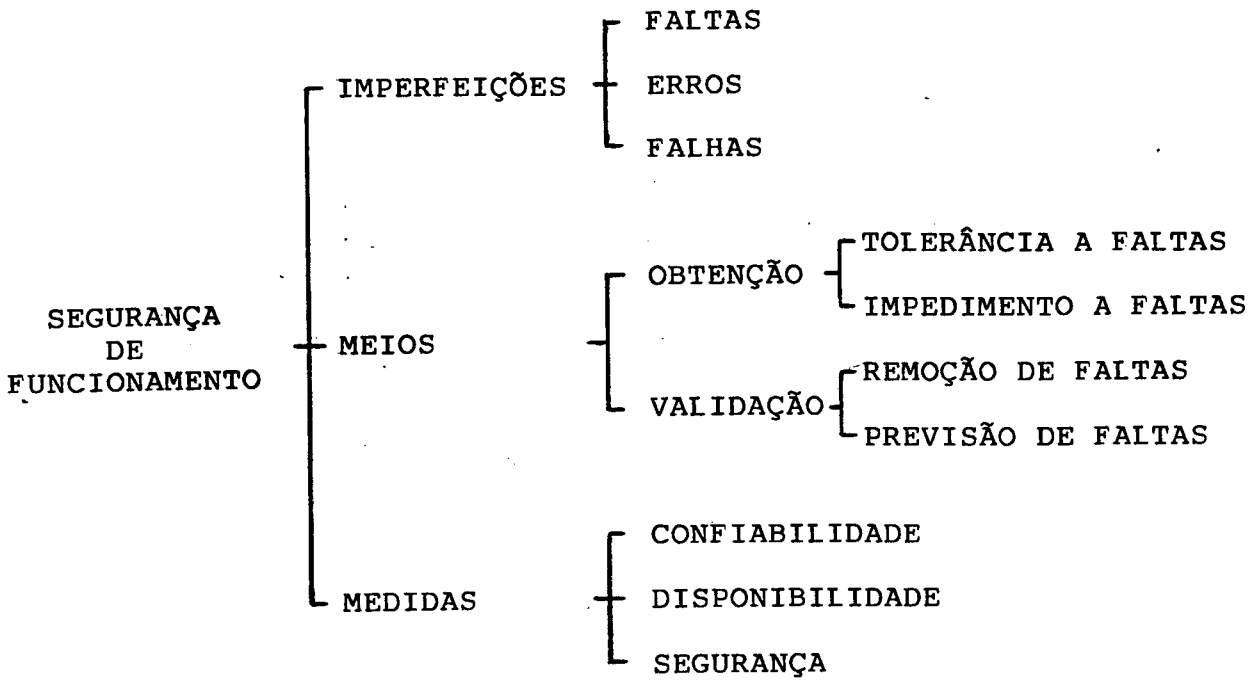


Figura 2.2. Taxonomia da Segurança de Funcionamento

2.3. Tolerância a Faltas

Esta parte visa apresentar de maneira sucinta as diferentes técnicas adotadas em sistemas computacionais no sentido de tolerar a presença faltas. Estas técnicas podem ser classificadas segundo duas estratégias:

- **detecção e recuperação de erros:** se baseia em técnicas de detecção de erro, localização e isolamento da unidade com erro. O objetivo, neste caso, é conduzir um sistema de um estado errôneo, para um estado não errôneo (ou válido);

- **compensação de erros:** se baseia no fornecimento do serviço especificado mesmo na presença de estados errôneos no sistema; os erros neste caso são ditos mascarados, ou seja, os efeitos da falta são automaticamente neutralizados sem notificação de sua ocorrência.

2.3.1. Formas de Redundância

O emprego das estratégias de tolerância a faltas num sistema computacional pressupoêm a existência de **redundâncias.** Por redundância se subentende a presença de recursos suplementares que não seriam necessários num sistema livre de faltas. Os recursos mencionados são: materiais, lógicos e de tempo. Para cada estratégia está associada um tipo de redundância:

a. a **redundância estática** está ligada às estratégias de compensação de erros. Neste caso existem várias unidades funcionalmente equivalentes (ou iguais), estruturadas de forma implícita e sistemática, tal que os erros produzidos por unidades faltosas são mascarados pela operação correta das restantes. Este tipo de redundância favorece o confinamento de erros.

b. a **redundância dinâmica** está associada às estratégias baseadas na detecção e recuperação de erros. Neste caso, a unidade funcional faltosa é substituída por outra equivalente. A redundância é realizada explicitamente por algoritmos especializados que são ativados depois de detectado um erro.

c. a **redundância híbrida** está associada à combinação de estratégias baseadas na compensação, e na detecção e recuperação de erros. Nesta redundância tanto existe o mascaramento de alguns tipos de erro como também a detecção e recuperação de erros.

2.3.2. Técnicas de Tolerância a Faltas

Neste item são apresentadas as diferentes técnicas adotadas na tolerância a faltas. Descrições mais detalhadas são encontradas em (Anderson & Lee, 1981), (Avizienis, 1978), (Johnson, 1984), (Kuhl & Reddy, 1986), (Miller, 1979), (Pilaud,

1982), (Randell, 1978), (Rennels, 1978), (Rennels, 1984) e (Tripathi & Wang, 1983).

Independente da forma de redudância utilizada, as técnicas básicas de tolerância a faltas podem ser divididas em quatro fases (Anderson & Lee, 1981) e (Siewiorek, 1984):

1. detecção de erros
2. confinamento e avaliação dos danos
3. recuperação de erros
4. tratamento de faltas e continuidade dos serviços

Estas quatro fases serão objeto dos itens subsequentes, juntamente com as técnicas asssociadas.

2.3.2.1. Detecção de Erros

Como uma falta não pode ser diretamente detectada pelo sistema, o ponto inicial para as técnicas de tolerância a faltas é a detecção de um estado errôneo (consequência da falta). Devido a isto, a detecção de erros tem que ser suficientemente efetiva, para evitar que um erro se propague gerando novos erros no sistema. Algumas das técnicas empregadas na detecção de erros são descritas em (Anderson & Lee, 1981), (Miller, 1979), (Randell, 1978):

Processamento por Comparação

Esta técnica é baseada na comparação de resultados obtidos de duas unidades de mesmo projeto (Emmerson & McGowan, 1984) e (De & Krakan, 1981), ou com projetos distintos. Caso seja detectada uma discordância nos resultados, dois procedimentos são possíveis: detectar qual das duas unidades é faltosa (através de observação interna), ou as duas unidade são trocadas por outro par de unidades de reserva em estado de espera (Emmerson, 1984).

A comparação dos resultados entre duas unidades de processamento, depende do nível de sincronismo existente entre as duas unidades:

- **microsincronizadas:** se o sincronismo é realizado dentro de um mesmo ciclo de relógio.

- **macrosincronizadas:** se o sincronismo é realizado a nível de segmento de programa ou de tarefa. Este tipo de sincronismo é utilizado em sistemas com programas de projetos diferentes.

Um outro tipo de processamento por comparação é o uso repetido do mesmo sistema para obter resultados a serem comparados (redundância de tempo). Esta técnica é viável somente na detecção de erros transientes.

Código de Detecção de Erros

Os códigos de detecção de erro são baseados na concatenação de redundância dentro da representação de um objeto (Anderson & Lee, 1981). Os dados redundantes apresentam uma relação fixa com a representação do objeto. Os erros causados pela corrupção num dos dois (representação do objeto ou redundância) irá causar uma discordância na relação, que por sua vez, pode ser detectada.

Alguns métodos clássicos para a detecção de erros por código são: verificação de paridade, códigos de redundância cíclica (CRC), códigos aritméticos, verificação de somas, etc. Também existem códigos que além de detectar erros também são usados na recuperação. Este é o caso dos códigos de Hamming e do M-out-of-N, que tem a possibilidade de correção de 1 bit.

Observação Interna

A observação interna é uma classe de técnicas que suspende o funcionamento da unidade funcional, momentaneamente, e utiliza seus recursos para a verificação da presença de erros (Miller, 1979). A eficiência do método depende do funcionamento correto dos recursos utilizados, senão resultados falsos podem ser obtidos. Abaixo, são descritos alguns métodos normalmente empregados (Anderson & Lee, 1981), (Miller, 1979), (Morgan & Taylor, 1977):

a. Programas de Teste

Um programa que sistematicamente testa a execução de cada instrução do computador. Neste método não existe uma boa detecção de erros transientes.

b. Testes de Coerência ('Reasonabless Cheks')

Este teste verifica se o estado de um objeto está dentro de limites corretos estabelecidos pela especificação. Um teste especial é a sinalização de erro quando o computador tenta executar uma operação ilegal (ex: divisão por zero). Neste caso uma interrupção é acionada para a recuperação.

c. Testes Estruturais

Este teste é aplicado particularmente a estruturas de dados complexas, que consistem, por exemplo, de um conjunto de elementos interligados por apontadores. Existem duas formas de verificação: a **integridade semântica**, que consiste na verificação da consistência das informações contidas na estrutura de dados, e a **integridade estrutural**, que testa a consistência da estrutura.

d. Auto-Diagnose

O objetivo específico da auto-diagnose é verificar o comportamento dos componentes do sistema. Esta verificação consiste em excitar os componentes com um conjunto de entradas, para as quais as saídas corretas são conhecidas. No tempo decorrido entre dois testes, é assumido o funcionamento correto do componente.

e. **Testes Reversos** ('Reversal Checks')

Em sistemas com uma entrada e uma saída, os testes reversos podem ser facilmente aplicados: a partir das saídas do sistema, calculam-se as entradas correspondentes, para posteriormente se comparar com as entradas reais. Este tipo de verificação só pode ser aplicada a sistemas aonde a computação inversa é simples.

f. **Cão-de-Guarda** ('Watchdog-Timers')

Esta técnica se baseia na existência de pontos de verificação ou teste em um programa. Um temporizador é inicializado toda vez que estes pontos de verificação são atingidos no processamento do programa, e decrementado segundo um relógio quando o programa prossegue seu processamento, entre pontos de verificação. Nestas condições duas situações podem ocorrer:

- um ponto de verificação é atingido antes que o temporizador chegue a zero. Neste caso o temporizador é novamente inicializado e o programa deve prosseguir.

- devido a um procedimento anormal no programa, o temporizador alcança a contagem zero, antes de atingir o próximo ponto de verificação. Neste caso deve ser acionada uma ação de recuperação da anomalia.

Observação Externa

É uma classe de métodos na qual elementos logicamente distintos operando concorrentemente com a unidade funcional, observam a sua operação para determinar a existência de erros (Morgan & Taylor, 1977), (Liebowitz & Carson, 1985). Abaixo serão descritos dois métodos de observação externa:

a. Observação de Mensagens

Abrange uma faixa grande de técnicas que inclui: observação dos protocolos de comunicação, medição do desempenho do suporte de comunicação, e verificação dos limites, através de processadores exteriores (Ayache, et alii, 1982) e (Molva, et alii, 1985).

b. Detecção da Integridade Estrutural

Esta técnica é baseada na verificação dos desvios do fluxo de controle no processamento de um programa, através da comparação com uma seqüência de controle conhecida, tida como válida. Uma implementação para esta técnica é apresentada em (Namjoo, 1982) onde um processor 'watchdog', funcionando em paralelo com o processador principal, tem o objetivo de detectar erros no fluxo de controle e acessos indevidos à memória, na execução de programas, usando técnicas de verificação baseadas em assinaturas.

2.3.2.2. Confinamento e Avaliação de Danos

A existência de atrasos entre a ocorrência de um estado errôneo, causado por uma falta, e a detecção de um erro (períodos de latência), possibilita a propagação dos erros pelo sistema. Por isso que, antes da recuperação do erro, o sistema tem que adotar uma estratégia para estabelecer mais precisamente qual a extensão dos danos no sistema. As estratégias para o confinamento de erros dependem da estrutura do sistema (Randell, 1984).

Um projeto bem estruturado, usando conceitos de camadas e módulos, deve apresentar limites bem determinados na propagação dos erros. A utilização destes conceitos clássicos determinam a criação de pequenos domínios de proteção (Jones, 1978) que devem confinar os erros. A única possibilidade de propagação seria através das interfaces que devem estar sujeitas a uma cuidadosa definição e monitoração, para prevenir que os erros se propaguem. A definição de operações baseadas sobre noções como ações atômicas devem também restringir a propagação de erros em um sistema (Randell, 1978) e (Tripathi & Wang, 1983).

2.3.2.3 Recuperação de Erros

As técnicas para a recuperação de erro tem como objetivo transformar um estado errôneo do sistema em um estado bem definido e livre de erros, da qual a operação normal do sistema

pode prosseguir. As duas técnicas de recuperação são: recuperação para trás e recuperação para frente de erro.

Recuperação para Trás do Erro ('Backward Recovery')

Tem como objetivo colocar o sistema num estado consistente anterior, ou seja, um estado livre de erros. Para isto, é necessário a previsão de **pontos de recuperação** que são os meios pelo qual um estado, ou informação de estado de um processo (programa em execução) pode ser salvo, para depois, se necessário, ser restaurado. Na prática, mesmo para uma atividade sequencial pode haver a necessidade de se manter armazenado uma sequência de pontos de recuperação referentes a um conjunto de estados; isto depende da estrutura do programa (por exemplo, aninhamento de ações atômicas).

No referente a atividade concorrente - diversos processos se executando concorrentemente para a realização de uma atividade, cuidados especiais devem ser tomados, na escolha de pontos de recuperação por processo da atividade. Um problema crucial neste casos é o chamado efeito dominó, onde um retrocesso generalizado dos processos do sistema é desencadeado, devido a má escolha dos pontos de recuperação de um processo com erro; isto representa uma grande perda na atividade do sistema.

A procura de um conjunto apropriado de pontos de recuperação para processos que interagem entre si, é a procura de pontos no sistema que permitam um retorno ordenado para trás, não representando a perda muito grande de processamento. Estes

pontos formam o conjunto das chamadas **linhas de recuperação.** Uma linha de recuperação identifica um conjunto de pontos, cada um pertencendo a um dos diferentes processos da atividade concorrente, que obedece a critérios definidos por Randell et alii (1978). Se um processo com erro retrocede a um ponto de recuperação, todos os processos que interagem com este, a partir de um ponto de recuperação, devem também retroceder aos seus pontos pertencentes a mesma linha de recuperação, definida pelo ponto do primeiro processo. Duas abordagens são discutidas na literatura, para o estabelecimento destas linhas de recuperação:

- **linhas pré-estabelecidas:** são linhas definidas pelo programador;

- **linhas calculadas,** são linhas definidas, durante a recuperação dos erros: dentro de um conjunto de pontos e segundo critérios.

Em atividades concorrentes é necessário a preservação de vários estados (pontos de recuperação) por processo. Além de evitar o armazenamento excessivo de informações e/ou também para economizar recursos, são necessários estratégias de "comutação de linhas" (´committ´) onde com a troca de pontos de recuperação (descarta estado e salva estado), linhas de recuperação cessam de existir e linhas passam a existir.

O conceito de linha de recuperação embora não seja largamente difundido na literatura, é base para estruturações mais elaboradas de atividades concorrentes, como ações atômicas

ou ainda transações atômicas. A recuperação para trás é objeto de descrições exaustivas em publicações como (Anderson & Lee, 1981) e (Randell et alii, 1978), onde os conceitos citados aqui são examinados em detalhe. Em seguida são apresentados sucintamente algumas técnicas e mecanismos utilizados na recuperação para trás.

a. Pontos de Verificação ('Checkpointing')

Esta técnica se baseia na cópia completa do estado do processo quando um ponto de recuperação é atingido. Algumas variações destes pontos de recuperação surgirão, no sentido de minimizar a quantidade de informação armazenada; em (Anderson & Lee, 1981) é discutido o mecanismo de "pontos de salvamento incremental" ('incremental checkpointing'), baseado em memórias 'cache', onde as informações armazenadas não envolvem todos os recursos, mas somente aqueles que serão modificados num processamento, apartir de um ponto de recuperação, e em ações atômicas.

b. Troca Segura

Nesta técnica a modificação ou a atualização de recursos é precedida pela criação de uma cópia do objeto (ponto de recuperação). Na atualização, uma vez confirmada as modificações, a cópia é descartada. A técnica de múltiplas cópias (Tripathi & Wang, 1983) é uma variação da troca segura, onde na confirmação de uma atualização a versão corrente de um objeto é substituida por uma nova versão. Estas técnicas são muito utilizadas na construção

de transações atômicas, e na atualização de sistemas de arquivos.

c. Arquivos Diferenciais

Nesta técnica todas as atualizações são gravadas em um arquivo diferencial e o arquivo principal (o objeto) é deixado intocável. As atualizações uma vez confirmadas são passadas (periodicamente) para o arquivo principal e apagadas do arquivo diferencial. O arquivo principal é utilizado na recuperação, quando não são confirmadas as atualizações. Esta técnica é normalmente empregada quando se trata de objetos grandes.

d. Recuperação em Bloco ('Recovery Blocks')

A recuperação em bloco, corresponde a uma técnica de estruturação onde um programa é formado por um bloco principal, juntamente com um ou mais blocos alternativos e testes de aceitação. Inicialmente o bloco principal é executado e um teste de aceitação é aplicado. Se o teste for consistente, o bloco principal termina com êxito, caso contrário, o estado do processo deve ser restaurado ao ponto inicialmente anterior ao bloco executado, e o próximo bloco a ser executado é o alternativo.

e. Registro do Processamento ('Logs/Audit Trail')

Nesta técnica, quando estabelecido um ponto de recuperação, dados relativos ao estado do processo não são gravados, a exemplo da técnica de pontos de verificação. Ao invés, a atividade subsequente do processo é monitorada e

gravada, para possibilitar que todas as mudanças no estado do processo possam ser desfeitas a partir do estado atual, para a recuperação para trás.

Recuperação para a Frente do Erro ('Forward Recovery')

Este método manipula parte do estado encontrado com erro para produzir um novo estado livre de erros. Esta técnica depende da antecipação de faltas, ou no mínimo, de todas as suas consequências. Uma das técnicas de recuperação para frente é o uso de manuseadores de exceções ('Exception Handlers').

O manuseamento de exceções é a técnica mais comun na recuperação para a frente de erros. As exceções são condições de erro antecipadas no sistema, ou seja, são comportamentos esperados mas indesejáveis. Um manuseador de exceções é um programa que é chamado quando uma condição de exceção específica é sinalizada durante o processamento. O objetivo então é de colocar o sistema num estado consistente apartir desta sinalização, ou seja, mascarar a ocorrência de excessões fazendo com que o sistema possa fornecer um serviço normal.

Para a utilização destas técnicas é necessário um completo entendimento da aplicação para a qual o sistema é projetado; isto implica que estes mecanismos devem fazer parte do aplicativo (ao contrário daqueles utilizados na recuperação para trás, que são mantidos em separado dos algoritmos de aplicação). Uma solução para a construção de manuseadores de exceção que possam cooperar com eventos inesperados, é prever a anulação de

quaisquer efeitos colaterais produzidos no programa e que, sinalize uma falha na exceção ou invoque um programa alternativo numa tentativa de mascarar a exceção.

A recuperação para a frente não representa uma perda de atividade, o que dá certa vantagem em relação à recuperação para trás. Entretanto, a técnica de recuperação para frente não é apropriada na recuperação de erros provenientes de faltas de projeto (Melliar-Smith & Randell, 1977). Nestes casos a recuperação para trás é mais eficiente.

As duas técnicas de recuperação de erro, não são excludentes, pelo contrário, estas se complementam entre si na recuperação de sistemas. A recuperação para frente do erro possibilita um tratamento eficiente das condições errôneas previstas no projeto, enquanto que a recuperação para trás do erro fornece uma estratégia geral que recupera os erros não previstos.

2.3.2.4. Tratamento de Faltas e Continuidade de Serviços

Depois que o sistema volta a um estado livre de erros, é necessário que o sistema continue fornecendo os seus serviços conforme especificados. Por isso, existem técnicas que garantem a não reincidência de faltas, de cujos os efeitos o sistema se recuperou, pois estas podem se tornar novamente ativas, criando novos erros.

Com relação ao tratamento das faltas, as técnicas aqui adotadas podem ser divididas em duas etapas: localização das faltas e reparo do sistema.

Localização das Faltas

As técnicas que possibilitam a localização de faltas são as mesmas empregadas nas duas primeiras etapas do tratamento de erros em sistemas tolerantes a faltas: detecção de erros e confinamento e avaliação dos danos.

Reparo do Sistema

Entre as técnicas de reparo do sistema, a diferença fundamental está na reconfiguração do sistema, ou seja, se é baseada ou não, na intervenção do operador humano:

a. Reconfiguração controlada manualmente

Existe a necessidade da intervenção humana para o reparo do sistema. O sistema por si só não consegue eliminar as faltas.

b. Reconfiguração automática

Todas as ações de reparo são executadas pelo próprio sistema. Estas ações são normalmente referidas como técnicas de "auto-reparo".

Os sistemas tolerantes a faltas com reconfiguração automática, dependendo da disponibilidade do reparo externo podem ser classificados em:

- **Sistemas Fechados:** o reparo não é possível, e o sistema inevitavelmente falha depois que os recursos da redundância se esgotam;

- **Sistemas Reparáveis:** nestes sistemas os módulos com falha são automaticamente identificados e trocados.

Com relação ao estado do sistema após a reconfiguração, ou seja, a continuidade dos serviços, pode-se obter:

- **reconfiguração completa:** o sistema retorna a um conjunto de condições que existiam antes de um erro ter sido detectado;

- **reconfiguração degradada:** o sistema tem a sua capacidade computacional reduzida;

- **desligamento seguro:** é o caso limite de uma reconfiguração degradada, quando a capacidade computacional está abaixo da mínima aceitável.

2.3.2.5. Técnicas de Voto

Embora seja nas técnicas de redundância dinâmica que as quatro fases da tolerância a faltas se apresentem mais

explicitamente delimitadas, estas também se apresentam, de maneira mais subentendida, nas técnicas associadas à compensação de erros. Conforme comentado anteriormente, as técnicas de redundância estática estão baseadas em estruturas de múltiplas cópias, onde a presença de faltas em unidades individuais são mascaradas pelo funcionamento correto das restantes. As técnicas de redundância estática podem ser divididas em duas classes.

Técnicas por Voto Implícito

As unidades funcionando corretamente bloqueiam fisicamente a propagação dos erros através do sistema. As saídas das unidades não são comparadas. A quadruplicação é um exemplo de uma técnica que utiliza mecanismos de voto implícito:

- **lógica quadruplicada:** está associada aos sinais digitais. Independentemente dos erros existentes, as características do sinal de saída permanecem inalteradas desde que haja unidades funcionando corretamente, possibilitando assim, o mascaramento dos erros;

- **componentes quadruplicados:** está associada aos sinais analógicos. A presença de um erro pode alterar, ligeiramente, as características do sinal de saída.

O advento dos circuitos integrados tornou a utilização da quadriplicação uma técnica quase obsoleta.

Técnicas por Voto Explícito

Nesta técnica de compensação de erros, um elemento extra - **o votador** - é necessário para comparar as saídas das unidades equivalentes que formam a estrutura redundante. Existindo acordo na votação majoritária simples, a saída é considerada correta. A implementação do módulo votante pode ser tanto por 'hardware' como por 'software'.

A estrutura da 'Triple-Modular Redudancy' (TMR), é um exemplo da redundância estática com voto explícito. A estrutura TMR utiliza três módulos idênticos que estão conectados a um módulo votante. O caso genérico do TMR é a 'N-Modular Redundancy', que consiste em N unidades idênticas que toleram (N-1)/2 faltas (York, 1983). Uma variação do TMR é a utilização de módulos de projetos distintos, conforme discutido em (Avizienis & Kelly, 1984).

2.4. Conclusão

Este capítulo é um trabalho de síntese onde foram examinados aspectos como a terminologia, conceitos básicos e técnicas associadas à segurança de funcionamento. Se definiu alguns termos essenciais, com o intuito de se estabelecer um vocabulário básico, a ser adotado em todo o trabalho. O restante do capítulo envolveu a apresentação e discussão de várias técnicas de tolerância a faltas agrupadas nas quatro fases que proporcionam a implementação de sistemas tolerantes a faltas:

detecção do erro, confinamento de erros e avaliação dos danos, recuperação do erro, e por último, tratamento de faltas e continuidade dos serviços.

Algumas das técnicas descritas neste capítulo, serão novamente abordadas em capítulos posteriores, de forma mais detalhada e com o enfoque ligado à implementação em sistemas digitais de arquitetura distribuída para a automação de usinas e subestações.

No próximo capítulo é descrito o sistema digital, no qual este trabalho está baseado, em termos do seu funcionamento geral, e das unidades computacionais que o compoêm.

CAPÍTULO 3

SISTEMA DIGITAL PARA AUTOMAÇÃO DE USINAS E SUBESTAÇÕES (US/SE)

3.1. Introdução

O desenvolvimento marcante da tecnologia de semicondutores associada com o rápido progresso da engenharia de 'software', da comunicação digital e da engenharia baseada no conhecimento, tem provocado mudanças consideráveis na área de controle de processos e automação industrial.

Entre os processos que atualmente estão passando por uma reestruturação tecnológica nos níveis funcionais de controle e supervisão, podem ser citados os processos associados à geração e transformação de energia elétrica, que no âmbito deste trabalho, se limitarão às usinas hidroelétricas e às subestações de alta tensão.

Embora os vários fatores que levaram a estas mudanças já tenham sido enumerados na literatura (CIGRÉ, 1983), abaixo são relacionados alguns dos benefícios que podem ser obtidos com a digitalização de usinas e subestações:

a. a instalação do sistema global se torna mais econômica, em consequência da diminuição dos custos envolvendo cabiação e

equipamentos digitais; no caso destes últimos, deve-se à crescente produção de componentes;

b. informação e registros mais completos, e maior capacidade de processamento que pode levar a uma tomada mais rápida de decisões em todos os sentidos;

c. implementação de novas funções, sem um aumento demasiado nos custos;

d. menor incidência de leituras erradas nas grandezas sendo adquiridas, devido às tecnologias que podem ser adotadas, por exemplo, a comunicação de dados por fibra ótica;

e. maior confiabilidade em todas as funções através da implementação de auto-diagnósticos e verificação da integridade dos dados;

f. melhor desempenho devido ao aperfeiçoamento na monitoração da planta e dos equipamentos;

Juntamente com os benefícios apresentados acima, a instalação destes sistemas pode, inicialmente, levantar algumas questões que ainda não foram plenamente esclarecidas, tais como:

a. a falta de experiência com relação ao desempenho de sistemas computacionais em ambientes altamente desfavoráveis, do tipo subestações;

b. os custos de desenvolvimento do software, associado aos custos de projeto e manutenção do software;

c. o impacto da nova tecnologia, no pessoal técnico que está familiarizado com tecnologias antigas;

d. a difícil reposição de equipamentos, devido ao avanço tecnológico que pode rapidamente tornar tecnologias obsoletas.

Um exemplo de sistema aplicado na automação de usinas e subestações, é o sistema digital de arquitetura distribuída, no qual este trabalho está baseado, que está sendo desenvolvido pelo Centro de Pesquisas de Energia Elétrica (CEPEL). Atualmente este sistema está sendo instalado, a nível de protótipo, tanto em usinas hidroelétricas como em subestações.

Para a descrição de uma subestação, aonde está sendo implantado o protótipo, em termos de seu funcionamento geral, módulos de que é composta e possíveis configurações, referenciar-se a (Lemos, 1987), onde é apresentada uma visão introdutória da subestação de transformação de Palhoça, pertencente à ELETROSUL.

O capítulo será dedicado a uma descrição geral do protótipo do sistema digital de controle distribuído (SDCD), onde serão examinados aspectos da arquitetura do sistema e da sua funcionalidade. Este capítulo corresponde a uma síntese de um

conjunto de documentos produzidos pelo CEPEL e ELETROSUL, sobre o sistema a ser abordado.

3.2. Descrição do Sistema Digital

O Sistema Digital para Automação de Usinas e Subestações (US/SE) - Sistema Digital, será responsável pela realização das funções de supervisão, controle e commando, inerentes ao funcionamento de uma US/SE. Caso o processo seja uma subestação, estas funções devem incluir (ELETROSUL, 1986):

- **funções de supervisão:** aquisição e tratamento de grandezas elétricas, informações sobre o estado operacional dos equipamentos, informações sobre o comportamento anormal dos componentes do sistema elétrico, registro sequencial de eventos inicializados automaticamente, e emissão de relatórios diversos;

- **funções de comando e controle:** intertravamento dos principais equipamentos de manobra, comandos sequenciais e individuais para manobra de equipamentos (automaticamente, ou não), e verificação de sincronismo.

O Sistema Digital, por sua vez, pode ser fisicamente dividido em duas partes: a primeira, o Centro de Supervisão e Controle (CSC), constituído de todas as unidades computacionais situadas na sala de controle - os Operadores, e a segunda, o pátio do processo, constituído das unidades computacionais que

ficam localizadas junto ao processo (US/SE) - Terminais de Aquisição e Controle (TAC). Cada uma das partes está baseada numa rede local de aspectos construtivos distintos, para poder atender aos diferentes requisitos funcionais a que estão sujeitas.

Como o processo (US/SE) pode ser particionado em 'bays' no caso de subestações ou unidades geradoras no caso de usinas, foi alocado para cada uma destas partições, um Terminal de Aquisição e Controle. Devido ao fato que 'bays' ou unidades geradoras, apresentam funcionamento que pode ser considerado independente, a rede do pátio do processo foi particionada em subredes. Isto implica que cada subrede tenha um funcionamento independente das demais, ficando a cooperação entre as mesmas restrita à troca de alguns dados relativos ao processo (US/SE). As subredes do pátio do processo estão interligadas à subrede da sala de controle através de unidades de comunicação ('Gateways').

A arquitetura deste sistema, para uma configuração sem redundâncias das várias unidades físicas que o compoêm está representada na figura 3.1.

3.3. Suporte Básico do Sistema

A integração dos recursos deste sistema distribuído se deve a dois componentes básicos que estão presentes em unidades computacionais do sistema: o Núcleo do Sistema Operacional e a Via Geral de Interconexão.

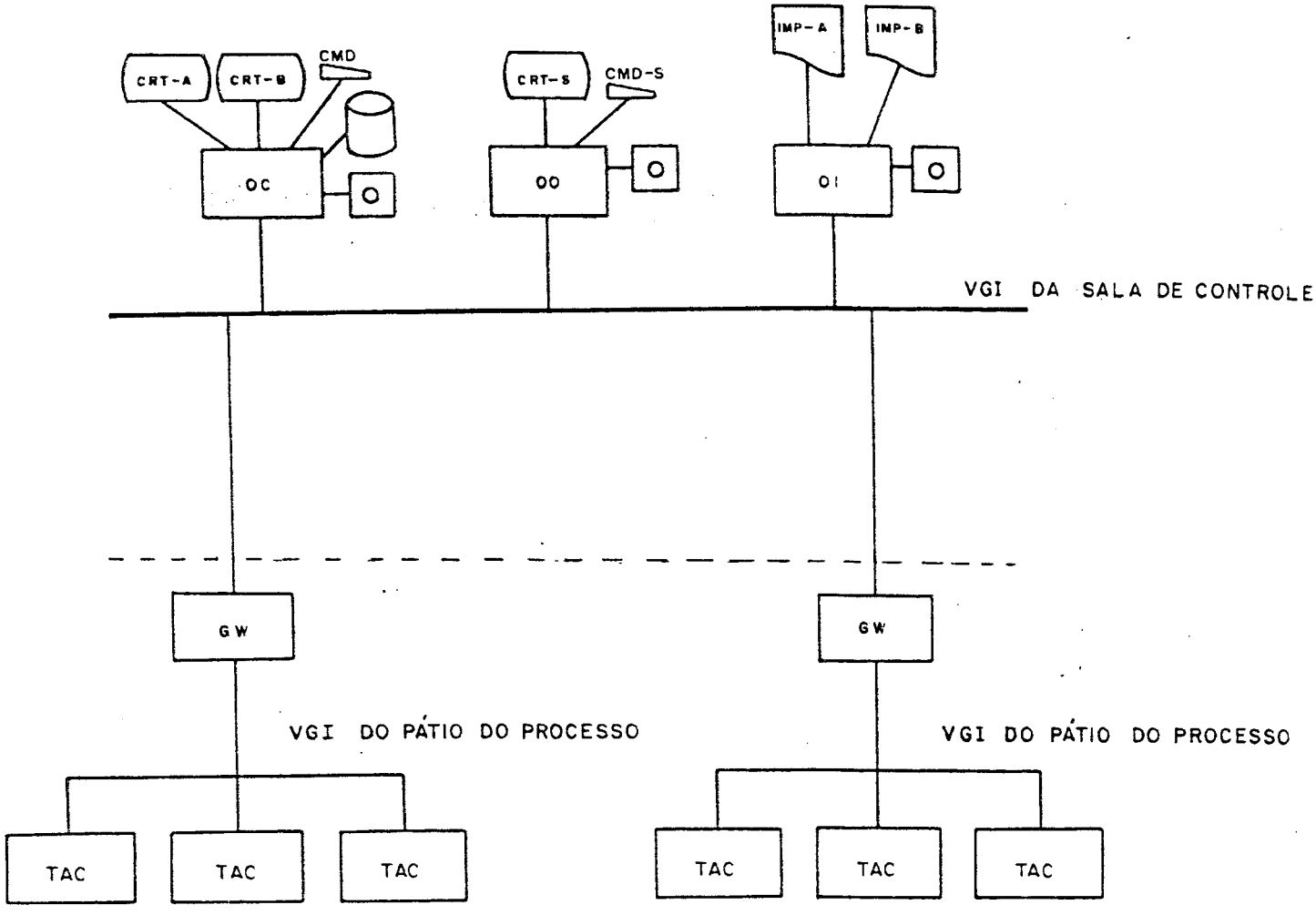


Figura 3.1. Sistema Digital para Automação de US/SE

3.3.1. Núcleo do Sistema Operacional (NSO)

O Núcleo do Sistema Operacional (NSO) é responsável pela implementação do ambiente multitarefas para a execução de aplicações distribuídas em tempo real. Para tanto, apresenta funções clássicas do tipo: comunicação e sincronismo entre tarefas, e gestão de recursos em tempo de execução (E/S, memória, interrupção...).

3.3.2. Via Geral de Interconexão (VGI)

A Via Geral de Interconexão (VGI) é a parte do sistema responsável pela transmissão de dados na rede de comunicação. Foi especificada com o objetivo de atender às necessidades do projeto e às especificações do Modelo de Referência para Interconexão de Sistemas Abertos (RM-OSI) da ISO. Com relação à padronização dos protocolos da VGI, foi seguido o padrão IEEE 802, que corresponde à camada física e à camada de enlace de dados do RM-OSI.

O controle de enlace da VGI proporciona, basicamente, serviços compatíveis com os propostos pelo o padrão IEEE 802.2 Classe I adicionados serviços de confirmação da recepção da mensagem transmitida.

No referente à camada física e o controle de acesso ao meio, foram basicamente adotados o proposto pelo o padrão IEEE 802.3, exceto com relação ao meio físico, onde foi adotada a fibra ótica, devido aos requisitos do processo (US/SE). No Centro de Supervisão e Controle foi adotado o cabo coaxial como meio físico.

3.4. Centro de Supervisão e Controle de Usinas/Subestações

O Centro de Supervisão e Controle é responsável, fundamentalmente, pelas atividades relacionadas com o interfaceamento homem-máquina de todo o sistema, tais como, a monitoração e controle do processo (US/SE). Estas atividades foram separadas em macro funções, conforme descritas abaixo (CEPEL, 1987).

a. **Função Acompanhamento:** apresentação contínua ao operador humano da evolução das variáveis do processo (US/SE) em supervisão, permitindo um perfeito conhecimento do estado presente de todo o processo (US/SE) e suas diferentes partes (estações).

b. **Função Comando:** coordenação na emissão de comandos para o processo (US/SE) a partir de comandos emitidos pelo operador, através da interface homen-máquina.

c. **Função Alarme:** detecção, análise, armazenamento e apresentação dos alarmes ocorridos na usina/subestação.

d. **Função Arquivamento:** registro e/ou armazenamento de informações de interesse sôbre a operação do sistema supervisionado.

e. **Função Consulta e Alteração:** obtenção, para fins de análise, de informações sem características tempo real dos pontos de supervisão do processo (US/SE), bem como a inserção de dados e a alteração de parâmetros destes pontos.

f. **Função Observação:** detecção de erros, recuperação e levantamento do estado operacional do sistema (levantamento da configuração do sistema, registro de eventos ocorridos, estatísticas diversas, etc.).

Para a execução destas funções o Centro de Supervisão e Controle é constituído de unidades de processamento com funções bem definidas no interfaceamento homem-máquina (Operadores), e das unidades de comunicação com a rede local do pátio do processo ('Gateways'). Todos os elementos da sala de controle estão interligados através de uma rede local que faz parte da Via Geral de Interconexão (VGI).

3.4.1. Operadores

Os Operadores são unidades computacionais responsáveis pelas funções da interface homem-máquina. Foram definidos os seguintes operadores para a sala de controle:

a. **Operador de Console (OC)**: responsável pela apresentação das variáveis supervisionadas do processo (US/SE); coordenação na emissão de comandos para o processo a partir de ações tomadas pelo operador humano; tratamento e registro de alarmes; e por último, armazenamento de informações sobre a operação do sistema supervisionado;

b. **Operador de Impressora (OI)**: responsável pela apresentação na impressora de relatórios periódicos e registro dos alarmes ocorridos;

c. **Operador de Observação (OO)**: responsável pela apresentação de informações relativas à supervisão do Sistema Digital, e pela execução de testes nas diversas unidades computacionais do sistema.

Um quarto operador, o Operador de Estação Mestre (OM), pode ser incluído se existir a necessidade de comunicação com o Centro de Operação do Sistema (COS).

As unidades computacionais que concentram as funções da interface homem-máquina dos operadores podem se apresentar replicadas ou não, na rede da sala de controle.

3.4.2. Unidades de Comunicação ('Gateways')

As unidades de comunicação ('gateways') são responsáveis pela comunicação entre as duas redes locais do sistema (da sala de controle e do pátio do processo). Estas unidades permitem o isolamento dos fluxos de informações de redes que apresentam características funcionais diferentes, aumentando desta forma a taxa de comunicação efetiva da rede global.

3.4.3. Banco de Dados Dinâmico (BDD)

O Banco de Dados Dinâmico (BDD) é parte do 'software' básico, existente em cada operador do Centro de Supervisão e Controle, responsável pela interface entre as funções de aplicação dos operadores e o processo supervisionado. O BDD está exclusivamente relacionado com as funções de supervisão, não participando da execução das funções de controle.

Todas as informações relativas às grandezas supervisionadas no processo são enviadas, em forma de mensagens, pelas controladoras de TAC, aos operadores do Centro de Supervisão e Controle, e recebidas pelo BDD de cada operador. A partir destas mensagens o BDD manterá uma imagem atualizada e um registro de eventos do processo.

As funções de aplicação dos operadores da sala de controle, consultarão o BDD sempre que necessitarem dados do processo, e serão sinalizados pelo BDD, automaticamente, quando da ocorrência de eventos no processo.

3.4.4. VGI da Sala de Controle

A VGI da sala de controle é responsável pela troca de informações entre os vários operadores, pela recepção dos valores correspondentes às variáveis do processo, e envio de comandos para o processo. Algumas das características desta rede são: topologia em barramento, o meio físico o cabo coaxial, e o método de acesso o CSMA/CD.

3.5. Pátio do Processo

Junto ao processo (US/SE) estão os Terminais de Aquisição e Controle (TAC) que estão interligados entre si através de uma rede local (pátio do processo), cujo o meio físico é a fibra-ótica. Esta rede é fisicamente constituída de um difusor ótico, localizado no pátio da US/SE, e que permite a conexão de três TACs e da unidade de comunicação ('gateway').

3.5.1. Terminais de Aquisição e Controle (TAC)

As TACs são as unidades computacionais que estão diretamente conectadas ao processo (US/SE), sendo responsáveis pelas seguintes funções:

- aquisição de dados do processo;
- acionamento de comandos para o processo;
- etiquetagem de tempo e transmissão imediata para a sala de controle de qualquer mudança no estado das variáveis discretas;
- transmissão periódica para a sala de controle de uma imagem atualizada de todas variáveis;
- intercomunicação entre TACs para a troca de variáveis.

Para a execução destas tarefas, os Terminais de Aquisição e Controle são constituidos de uma controladora de TAC responsável pelas funções de comunicação e coordenação do funcionamento da TAC, e pelos operadores do processo responsáveis pelas funções de aquisição de dados e de envio de comandos para o processo. A controladora de TAC e os operadores do processo estão interligados através da Via Singela de Interconexão (VSI), conforme representado na figura 3.2.

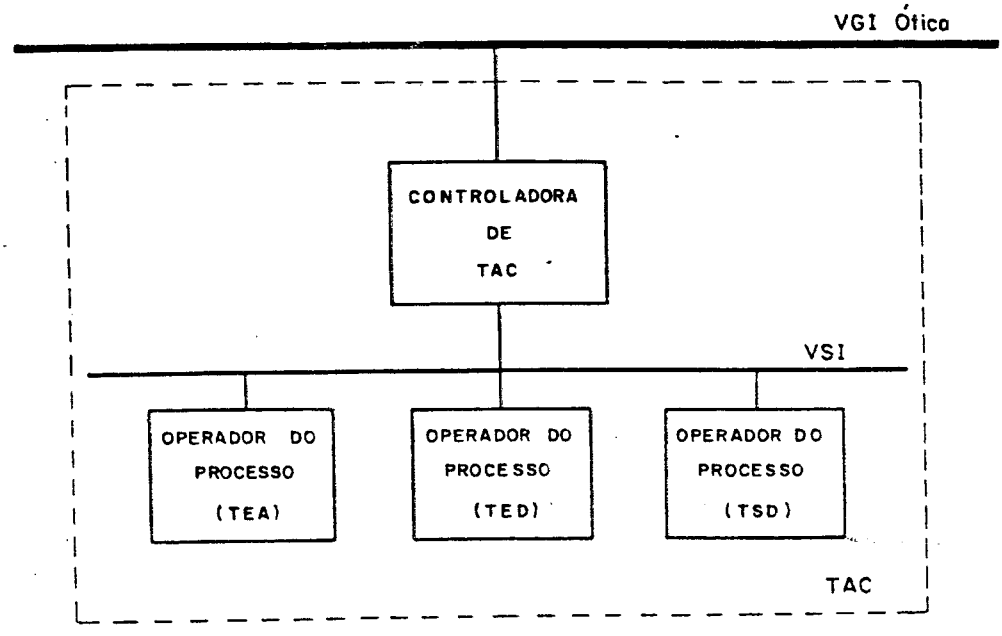


Figura 3.2. Terminal de Aquisição e Controle (TAC)

3.5.1.1. Controladora de TAC

Para a realização das funções de que é responsável, o software da controladora de TAC foi decomposto em vários blocos que podem ser agrupados em 'software' básico e 'software' de aplicação.

'Software' Básico

O 'software' básico da controladora de TAC é constituído do NSO, do 'software' da VGI, e do 'software' da VSI. A Via Singela de Interconexão (VSI) implementa a comunicação entre a controladora de TAC e os operadores do processo obedecendo a um protocolo mestre-escravo. A iniciativa de comunicação parte da controladora que envia uma mensagem a um ou mais operadores.

'Software' de Aplicação

a. **Tratamento das Entradas Analógicas (TEA)**: gerencia a aquisição dos dados analógicos, através da coordenação da transferência dos dados adquiridos por todos os operadores de entradas analógicas de um mesmo TAC. Processa estes dados e os envia à sala de controle. As atribuições do TEA são:
   - configurar os operadores de entradas analógicas;
   - sincronizar a aquisição dos dados analógicos entre TACs, que envolve a obtenção de grandezas de outros TACs, não disponíveis localmente;

- adquirir dos operadores analógicos as grandezas aquisicionadas;
- verificar a integridade e limites dos dados, escalonar grandezas, e atualizar o banco de dados analógicos da controladora;
- enviar periodicamente para a sala de controle os dados analógicos processados.

b. **Tratamento de Entradas Digitais (TED):** gerencia a aquisição dos dados digitais, através da coordenação da transferência dos dados adquiridos por todos os operadores de entradas digitais de um mesmo TAC. Processa estes dados, e os envia a sala de controle. As atribuições do TED são:
   - configurar os operadores de entrada digital;
   - sincronizar a aquisição num mesmo TAC;
   - requisitar periodicamente os dados adquiridos pelos operadores;
   - requisitar os dados que sofreram uma mudança de estado; a sinalização é feita pelos operadores, através de uma linha de interrupção;
   - rotular os dados digitais com data e hora, e atualizar o banco de dados;
   - enviar os dados digitais, periodicamente ou na ocorrência de uma mudança de estado, para a sala de controle e demais TACs que compoem a sub-rede.

c. **Tratamento das Saídas Digitais (TSD):** gerencia a execução dos comandos que são acionados pelo operador de saídas digitais. As atribuições do TSD são:
   - verificar na execução de um comando as condições do processo, por exemplo, o intertravamento e sincronismo no caso de subestações;
   - enviar uma ordem de comando ao operador de saídas digitais.

3.5.1.2. Operadores do Processo

Os operadores do processo são unidades computacionais que estão diretamente interconectadas com o processo. São responsáveis pela execução das funções de aquisição de dados e envio de comandos para o processo. Existem três tipos de operadores: Entradas Analógicas, Entradas Digitais e Saídas Digitais. Todos os operadores têm a mesma unidade central de processamento, as diferenças estão a nível de interfaces com o processo e no software de aplicação.

O funcionamento dos operadores do processo é comandado pela controladora de TAC, que tem condições de inicializá-los mesmo quando da necessidade de recuperação de um erro.

Os operadores são constituídos do 'software' de aplicação e do 'software' básico. Sendo este último por sua vez formado pelo NSO e do 'software' da VSI. Em seguida, são descritos o 'hardware' e o 'software' dos operadores do processo.

Operador de Entradas Analógicas

Tem como objetivo adquirir valores analógicos do processo e gerar grandezas derivadas destes valores. Permite a aquisição de 6 grandezas analógicas AC e 10 grandezas DC. É constituido por duas placas.

a. **Placa de aquisição:** trata da aquisição, conversão e processamento de grandezas analógicas;

b. **Placa de condicionamento:** trata do condicionamento dos sinais analógicos que envolve o escalonamento, o isolamento elétrico e a filtragem contra interferência eletromagnética.

O software do operador de Entradas Analógicas é subdividido nas funções descritas abaixo.

a. **Configuração:** fornece a configuração básica das entradas analógicas do operador do processo. Esta configuração indica: o número e a posição dos pontos DC, o número e o tipo de pontos AC, e o tipo de processamento que deve ser realizado com os pontos AC.

b. **Aquisição de valores analógicos DC (ADC):** tem como objetivo:
   - amostrar periodicamente os pontos DC;
   - verificar a existência de valores espúrios;

- atualizar o registro de dados do operador do processo.

c. **Aquisição de valores analógicos AC (AAC):** esta função, dependendo da configuração do operador, tem dois conjuntos de operações. As operações para qualquer tipo de configuração são:
   - amostragem dos valores AC;
   - cálculo e filtragem dos valores eficazes;
   - cálculo da frequência dos sinais;
   - filtragem dos valores espúrios.

As operações que dependem da configuração adotada para o operador são:
   - cálculo do ângulo de fase;
   - cálculo das potências ativa e reativa.

d. **Auto-diagnose (DIAG):** realiza testes nos vários circuitos do operador. Caso estejam funcionando corretamente o 'watch-dog' é inicializado.

Operador de Entradas Digitais

Tem como objetivo a aquisição das entradas de pontos digitais. Cada operador tem a capacidade de adquirir até 48 pontos digitais. É constituido por duas placas.

a. **Placa de aquisição:** trata da aquisição e filtragem contra 'bouncing' de contatos dos eventos digitais.

b. **Placa de condicionamento:** trata do isolamento elétrico entre o processo e o operador, filtra os eventos dos ruídos provocados por perturbações do sistema elétrico, alimenta as chaves do processo e executa testes de continuidade e funcionamento das entradas.

O software do operador de Entradas Digitais é subividido nas funções descritas abaixo.

a. **Aquisição:** trata da aquisição das entradas digitais que é realizada periodicamente. A cada mudança de estado detectada, é associada uma etiqueta de tempo contendo o intervalo de tempo decorrido entre a última mensagem de sincronismo (atualização do relógio tempo real) enviada pela controladora de TAC e a ocorrência do evento.

b. **Tratamento (TRAT):** os eventos e as respectivas etiquetas de tempo são enviados pela via de comunicação à controladora de TAC após a existência de uma mudança de estado nas entradas digitais. Em cada entrada digital é realizada uma filtragem para verificar se a frequência de mudanças de estado de cada ponto, num dado intervalo de tempo, não excede um limite pré-estabelecido. Caso exceda, a aquisição deste ponto será inibida temporariamente. A controladora de TAC tem também condições de inibir a aquisição de entradas digitais através de um comando.

c. **Auto-diagnose** (DIAG): tem como objetivo a realização de testes nos vários circuitos do operador. Caso estejam funcionando corretamente o 'watch-dog' é inicializado.

Operador de Saídas Digitais

Tem como objetivo o envio de comandos para o processo em função de mensagens provinientes da controladora de TAC que indica qual a saída que deve ser atuada. É composta por duas placas: Placa de Excitação e Placa de Condicionamento. Esta última, com a capacidade de 24 relés de contatos secos e sistema de proteção contra acionamento falso.

3.5.2. VGI do Pátrio do Processo

A VGI do pátio do processo é responsável pela troca de informações entre os Terminais de Aquisição e Controle, pelo envio dos dados adquiridos do processo para a sala de controle, e pela recepção de comandos da sala de controle enviados ao processo. Algumas das características desta rede são: topologia em barramento-estrela, o meio físico a fibra ótica, e o método de acesso o CSMA/CD.

3.6. Conclusão

Neste capítulo foi realizada a descrição do Sistema Digital de Controle Distribuído, sendo desenvolvido pelo c CEPEL, que

terá sua aplicação na automação de usinas hidroelétricas e subestações de alta-tensão. Inicialmente foram apresentadas algumas vantagens e desvantagens da instalação de SDCDs na automação de US/SE, sendo que, dentro das desvantagens outras duas poderiam ser enumeradas: a primeira relacionada à complexidade inerente do desenvolvimento e funcionamento de sistemas distribuídos com atributos de tempo real, e a segunda, relacionada aos requisitos de confiabilidade, disponibilidade e segurança que são essenciais nestes sistemas, devido aos prejuízos materiais e humanos que estes podem causar, caso deixem de executar as funções conforme especificadas. Nos capítulos seguintes, o enfoque a ser dado será neste sentido, inicializando-se pelas atividades de gerenciamento, que visam a obtenção de um conhecimento efetivo do comportamento do sistema.

CAPÍTULO 4

GERENCIAMENTO DO SISTEMA DIGITAL PARA AUTOMAÇÃO DE US/SE

4.1. Introdução

A interconexão de computadores através de redes de comunicação tem possibilitado o compartilhamento e o processamento de informações em ambientes distribuídos. Podendo estes computadores estar geograficamente próximos (redes locais) ou separados uns dos outros (redes longa distância). Entretanto, a criação destes sistemas fez surgir algunas novas características, tais como:

a. operação de redes direcionadas para aplicações diferentes, usando meios físicos e arquiteturas diferentes, com equipamentos, de vários fabricantes, implementados com tecnologias diferentes;

b. incorporação de novas funções, devido às mudanças de requisitos, ou utilização de novas tecnologias;

c. adoção de meios que facilitem o gerenciamento de recursos de sistemas distribuídos, de forma mais automatizada, mesmo em situações onde a abrangência geográfica e o número de usuários determinam a complexidade destes sistemas.

Portanto, para garantir a operacionalidade e a troca de informações, é necessário o desenvolvimento de um suporte que permita a administração destes sistemas durante todo o seu ciclo de vida, desde a definição do sistema até a sua operação. A Administração de Sistemas Distribuídos provê ferramentas que executam um conjunto atividades, que são: planejamento, pré-instalação, controle da instalação, controle operacional, e planejamento pós-instalação do sistema distribuído. Em particular, o controle operacional faz uso do Gerenciamento de Sistemas Distribuídos, para coletar informações, analizar e executar as ações de controle sobre os componentes do sistema.

O Gerenciamento de Sistemas Distribuídos está voltado para o planejamento, organização, supervisão, contabilização e controle dos componentes do sistema em operação. Os componentes do sistema formam um conjunto de recursos para o processamento, armazenamento, e comunicação de informação. Pode-se concluir então, que o gerenciamento está envolvido com todas as ações e reações que são necessárias para processar, armazenar e transferir a informação, sem contudo, tomar parte da atividade normal da aplicação. Nestas condições, portanto, o gerenciamento automatizado se torna crucial dentro do objetivo de manter estes sistemas operando efetivamente, para providenciar serviços aos usuários e possibilitar a sua evolução.

Neste capítulo é abordado o gerenciamento do Sistema Digital de Controle Distribuído para Automação de US/SE - Sistema Digital, apresentando no primeiro item um modelo geral

para o gerenciamento de sistemas distribuídos. Nos dois itens seguintes são descritas, respectivamente, as funções básicas e as de suporte para as atividades de gerenciamento de um sistema distribuído. Devido à complexidade envolvida é abordado em separado o Gerenciamento do Subsistema de Comunicação. Por último, são descritas as atividades de gerenciamento do Sistema Digital para Automação de US/SE dentro do contexto global do que se denominou de Função de Observação.

4.2. Modelo de Gerenciamento

A função de gerenciar um sistema é conduzida segundo políticas de gerenciamento, que especificam as regras básicas a serem adotadas, na escolha das decisões. Estas decisões são específicas da aplicação, e podem seguir critérios técnicos, comerciais, ou mesmo estratégicos.

O modelo adotado para o gerenciamento de sistemas distribuídos, está baseado no apresentado em (Sloman, 1987), um trabalho que teve como preocupação primordial, a especificação de serviços e protocolos de gerenciamento para ambientes distribuídos. Na figura 4.1 está representado um modelo, da relação existente entre a Administração e o Gerenciamento de Sistemas Distribuídos.

Dentro do enfoque do Gerenciamento de Sistemas Distribuídos, também é abordado, mais detalhadamente, o gerenciamento do subsistema de comunicação (gerenciamento de

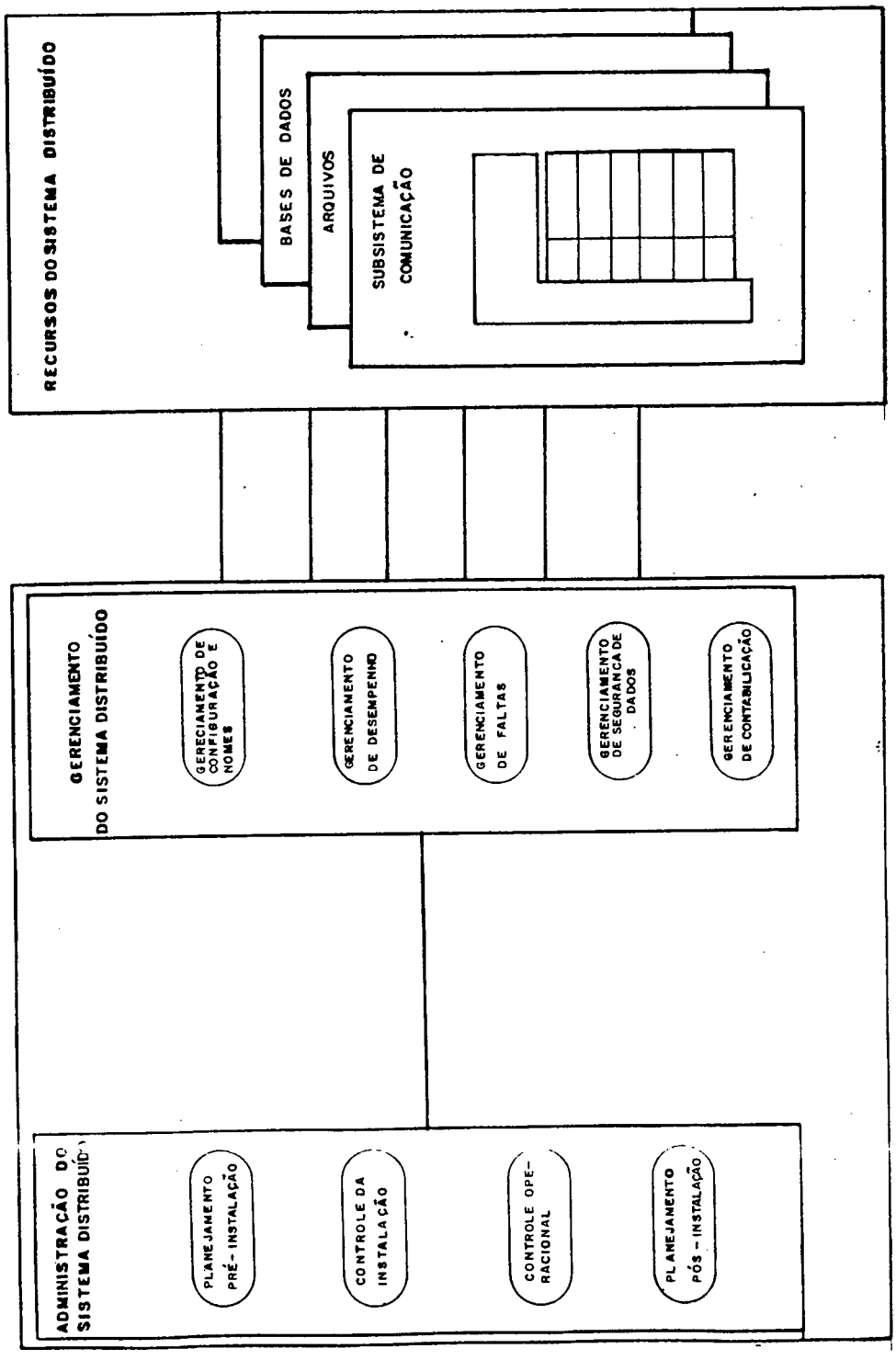


Figura 4.1. Administração e Gerenciamento de Sistemas Distribuídos

redes), que é identificado como um recurso crítico e altamente complexo, num ambiente distribuído. Devido a estas características, é essencial considerá-lo como um domínio separado no gerenciamento de sistemas distribuídos. A padronização de serviços e protocolos no gerenciamento de redes tem sido uma preocupação de vários organismos internacionais, e comitês de padronização (GM, 1987), (Langsford et alii, 1983), (Morris & Carter, 1986), (Wakid et alii, 1987). Entretanto, o Gerenciamento do Subsistema de Comunicação apresentado neste capítulo está baseado no relatório final sobre gerenciamento de redes locais, com atributos de tempo real, do programa europeu COST 11 BIS (Sloman, 1984), (Tschammer & Klessman, 1985). Neste relatório, são apresentados métodos de estruturação e classificação das atividades de gerenciamento, como também, das relações existente entre as entidades de gerenciamento e comunicação, tomando como modelo o RM-OSI da ISO.

É interessante assinalar que, embora o gerenciamento do subsistema de comunicação seja um subconjunto de atividades no gerenciamento de sistemas distribuídos, este último surgiu posteriormente. O modelo adotado, para o último, é uma generalização do gerenciamento do subsistema de comunicação, onde o suporte de comunicação é um dos recursos do sistema, e portanto, a ser gerenciado.

As atividades relacionadas ao Gerenciamento de Sistemas Distribuídos envolvem a **observação** da operação dos componentes (obtenção de informações de gerenciamento), a **tomada de decisões** baseadas nas observações e **execução de ações** para controlar ou

simplesmente influenciar o serviço sendo fornecido. A execução destas atividades está a cargo dos **agentes** do gerenciamento que são responsáveis pelos objetivos a atingir, determinados a partir de uma política de gerenciamento. O termo **administrador** está associado a um agente do gerenciamento humano. Os agentes se comunicam por meio de **protocolos de gerenciamento**, que controlam toda a troca de informações entre os vários agentes do gerenciamento. A execução de um protocolo de gerenciamento é também, uma **atividade** do gerenciamento.

4.3. Funções do Gerenciamento

As áreas funcionais no gerenciamento de sistema distribuídos, a serem abordadas em seguida e conforme representadas na figura 4.1, são as seguintes: configuração e nomes, faltas, desempenho, segurança de dados e contabilização. Além destas, existem as funções de suporte ao gerenciamento que são: gerenciamento de recursos, monitoração, Base de Informações do gerenciamento, e a interface homem-máquina. A Base de Informações é uma base de dados que contém informações relativas ao gerenciamento.

4.3.1. Gerenciamento da Configuração e Nomes

O gerenciamento da configuração num sistema distribuído é constituído das funções relacionadas com a instalação e interconexão dos componentes de hardware e software. Estas

funções, ou são executadas na inicialização dos serviços ou na reconfiguração do sistema, para prover a evolução, a otimização do desempenho, ou o suporte para o gerenciamento de faltas. A configuração tanto pode ser **estática** - o sistema deverá ser interrompido para a realização das mudanças, ou **dinâmica** - o serviço pode ser executado continuamente, sem a necessidade de ser interrompido, para a realização de mudanças de configuração.

Uma outra função da configuração é manter o estado operacional atualizado dos componentes de hardware e de software do sistema. As informações, abaixo relacionadas, são essenciais para o gerenciamento da configuração, e poderão ser mantidas na Base de Informações do gerenciamento.

Configuração física: a relação dos componentes do hardware instalados no sistema (por exemplo, ´gateways´, unidades computacionais e impressoras), o número da versão, e o seu estado operacional corrente.

Configuração lógica: a relação dos componentes do software alocados a cada entidade do sistema, incluindo o número da versão, o estado operacional corrente, e a relação com outros componentes. Este último, simplifica a localização dos componentes que serão afetados, quando da mudança de um dos componentes de software.

Abaixo são enumeradas algumas funções pertinentes ao gerenciamento da configuração que involvem a criação, conexão,

ativação, interrupção, desconexão e destruição de instâncias, além do carregamento e remoção do código, e a obtenção de informações sobre o estado das instâncias:

a. a instalação de componentes de hardware, que corresponde a mudança na configuração física; embora a instalação necessite da intervenção manual, uma vez realizada, é possível se mudar o estado operacional do componente físico por controle remoto. Algumas das mudanças de estado são: ativo para passivo, e normal para modo de manutenção;

b. a alocação dos componentes de software aos componentes de hardware; isto pode ser executado pelo administrador responsável pela configuração dos serviços, ou automaticamente, através de um gerente de recursos que realize o balanceamento de carga;

c. a identificação dos componentes de hardware ou de software, necessários a um serviço;

d. a alocação dos recursos de memória, programa ou dados, às instâncias. Estes recursos podem ser pré-alocados, de forma definitiva, na criação de uma instância, ou alocados e liberados dinamicamente conforme a necessidade;

Todas as mudanças ou eventos, na configuração, deverão ser registrados, pois poderão ser úteis na determinação das causas de erros subsequentes.

A atribuição de identificadores, sejam nomes lógicos ou endereços físicos, aos componentes, também é considerado parte do gerenciamento da configuração, se realizado durante o tempo de configuração. Usualmente, os identificadores têm que ser únicos dentro do contexto global do sistema e são compostos levando em consideração atributos de localização (Watson, 1981). Identificadores de grupo podem ser necessários se visado o endereçamento simultâneo de vários recursos ('multicast').

Para o levantamento da configuração do sistema e do seu estado operacional, o gerenciamento da configuração fará uso dos serviços da monitoração.

4.3.2. Gerenciamento do Desempenho

O gerenciamento do desempenho tem como objetivo a obtenção e a análise das medidas estatísticas relativas aos recursos do sistema distribuído. A análise serve para otimizar o desempenho do sistema através da modificação de parâmetros, alocação de recursos e troca de componentes.

Normalmente, no projeto de um sistema são estabelecidos os índices de desempenho que este deverá alcançar durante o seu funcionamento. Porém, é necessária uma estimativa preliminar, da viabilidade destes índices serem alcançados na implementação, antes de inicializado o projeto. Esta estimativa é obtida através de modelos analíticos, ou através de resultados de simulações. Como também é necessário monitorar, durante o

funciomanento, os índices de desempenho sendo atingidos pelo sistema, e por último, analisar se estes índices se mantêm coerentes com os inicialmente estabelecidos.

Este tipo de abordagem é usualmente aplicada aos recursos do sistema distribuído, que podem ser considerados críticos: subsistema de comunicação, unidades de processamento, etc. Dependem desta abordagem algumas medidas da segurança de funcionamento (confiabilidade, disponibilidade e taxa de manutenção).

O gerenciamento de desempenho, por sua vez, foi dividido na monitoração do desempenho e na análise do desempenho.

4.3.2.1. Monitoração do Desempenho

Os agentes do gerenciamento são responsáveis pela obtenção, através da monitoração, dos dados relacionados com a ocorrência de eventos no sistema, com o objetivo de acompanhar, com o sistema em funcionamento, os índices de desempenho. Estes eventos estão relacionados, com as atividades de diversas entidades do sistema como, por exemplo, o subsistema de comunicação.

4.3.2.2. Análise do Desempenho

Os levantamentos estatísticos realizados na monitoração do desempenho são analisados para se detectar pontos de estrangulamento nas atividades do sistema, e tendências negativas no seu funcionamento tal como, aumento da taxa de erros e diminuição na capacidade de processamento.

Como algumas destas atividades necessitam de uma visão global do sistema ou participação do operador humano, estas são normalmente alocadas numa entidade de supervisão do sistema.

4.3.3. Gerenciamento de Faltas

O gerenciamento de faltas tem como objetivo, resolver as situações anormais na operação do sistema. A existência de falhas (parciais) no sistema preconizam a existência de faltas nos componentes de hardware e/ou software do sistema. As faltas se manifestam através dos erros, que são detectados por outros componentes. Caso estes erros se manifestem nos serviços do sistema, então o sistema pode falhar como um todo. É este estado, nos serviços do sistema, que o gerenciamento de faltas tem de evitar que ocorra, através do emprego de técnicas de tolerância a faltas.

O gerenciamento de faltas é responsável somente pelos eventos causados da ocorrência de faltas no sistema, enquanto a redução no desempenho ficará a cargo do gerenciamento de

desempenho. Devido à similaridade no tratamento de eventos anormais no sistema, as atividades destes gerenciamentos têm que ser desenvolvidas de forma integrada.

Como algumas das técnicas e mecanismos, que podem ser empregados no gerenciamento de faltas de sistemas distribuídos já foram anteriormente discutidos, estes não serão novamente abordados.

4.3.4. Gerenciamento da Segurança de Dados ('Data Security')

O gerenciamento da segurança de dados tem como atribuições o controle da distribuição da informação aos vários serviços do sistema distribuído, para que estes possam implementar as políticas de segurança (Fraga & Lemos, 1987) . Os objetivos do gerenciamento da segurança de dados dentro de um sistema distribuído são:

- manter a autenticidade, a disponibilidade, a confidencialidade, e a integridade da informação mantida nos sistemas, ou em comunicação entre estes;

- controlar o acesso aos serviços e aos componentes, para garantir que cada usuário ou processo em execução só possa acessar os serviços para os quais está autorizado.

Os mecanismos usados para alcançar os objetivos acima mencionados incluem, o controle de acesso físico aos componentes

do sistema, o uso de técnicas criptográficas, listas de controle de acesso, etc.

4.3.5. Gerenciamento de Contabilização

O gerenciamento de contabilização possibilita ao administrador determinar e alocar custos e tarifas, pela utilização de um recurso do sistema distribuído. A contabilização está diretamente relacionada ao controle de acesso dos recursos do sistema, conforme os usuários são identificados. Usualmente esta atividade não é incorporada a sistemas distribuídos baseados em redes locais, por isso não será discutida em detalhe.

4.4. Serviços de Suporte ao Gerenciamento

4.4.1. Gerenciamento de Recursos

O Gerenciamento do Sistema Distribuído, está voltado para o controle dos recursos utilizados (subsistema de comunicação, entidades, 'gateway', base de dados, arquivos, etc.) para providenciar os serviços a serem oferecidos pelo sistema. Por conseguinte, um recurso do sistema distribuído pode ser visto como um único componente, num determinado nível, enquanto que, na realidade, pode consistir num "aninhamento" de recursos que estão fisicamente distribuídos pelo sistema.

4.4.1.1. Estados dos Recursos

Um recurso no sistema distribuído tem um estado que é indicado por uma variável de estado, que pode ser modificada através de operações efetuadas pelo gerente do recurso. Os estados principais de um recurso são descritos abaixo (Sloman, 1984). Na figura 4.2 é representada a relação entre os estados dos serviços e os estados do gerenciamento, para um recurso do sistema distribuído.

Desconhecido: um recurso não foi instalado no sistema; o recurso não é parte do sistema.

Desligado: o recurso foi instalado mas não conectado, logicamente ou fisicamente, ao sistema, e com isto, não toma parte dos serviços oferecidos pelo sistema. Este estado pode corresponder a um estado intermediário no processo de instalação, ou em estado de manutenção; por exemplo, executando testes de auto-diagnóstico.

Ligado (Fora de Serviço): o recurso está fisicamente ou logicamente conectado ao sistema, mas não está sendo usado para fornecer um serviço. Este estado pode corresponder a uma entidade com falha. Neste caso duas situações são possíveis:

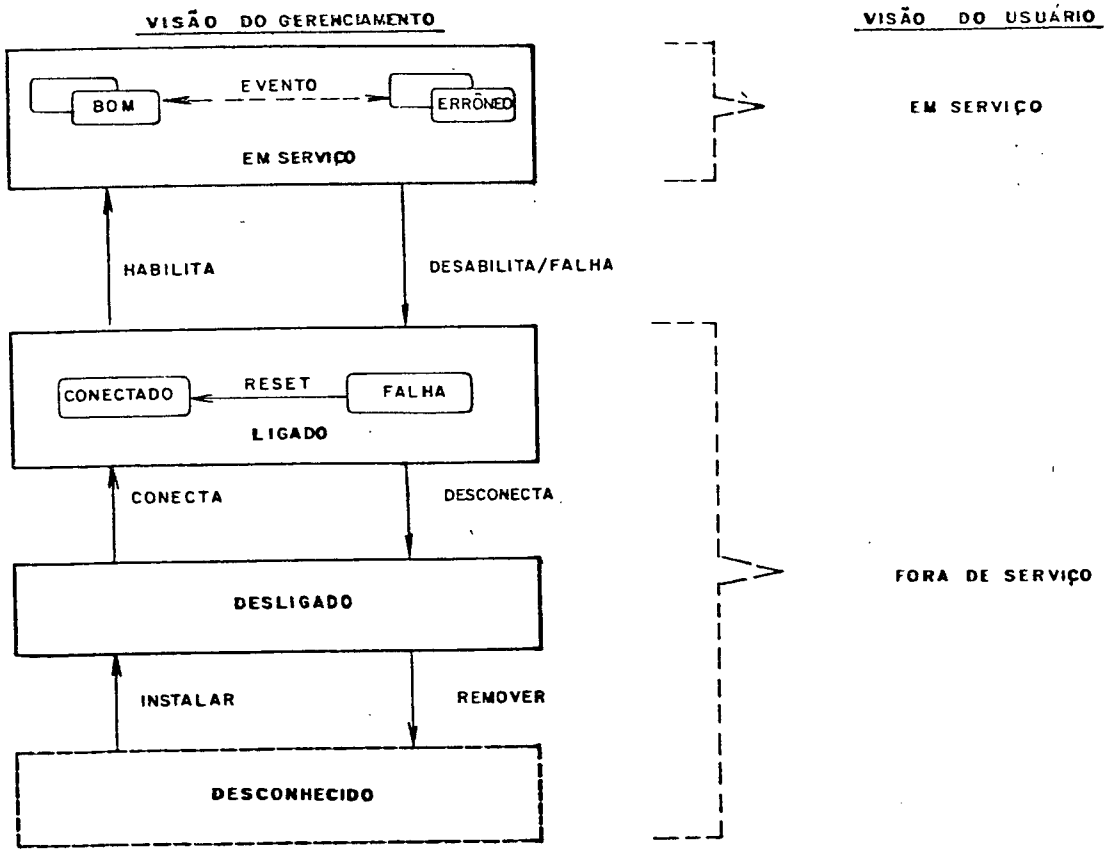


Figura 4.2. Diagrama de Transição de Estados nos Recursos

Conectado: corresponde a um recurso que está fisicamente ou logicamente conectado no sistema, mas não habilitado.

Falha: o recurso deixa de fornecer o serviço até que seja reparado ou re-inicializado. O estado errôneo não tem condições de ser recuperado.

Ligado (**Em Serviço**): o recurso está em funcionamento e contribuindo para o serviço sendo oferecido pelo sistema. Pode se caracterizar como um estado:

Bom: corresponde às diferentes qualidades no serviço sendo oferecido.

Livre: recurso disponível.

Ocupado: recurso temporariamente não disponível.

Uma visão, pelo usuário, do estado 'em serviço' para a condição de 'bom', está representado na figura 4.3.

Errôneo: um estado que está relacionado à presença de erros. Entretanto, a recuperação do erro pode colocar o recurso num estado livre de erros. Os serviços não chegam a ficar totalmente afetados.

Os estados, acima enumerados, não necessariamente são aplicados a todos os recursos do sistema.

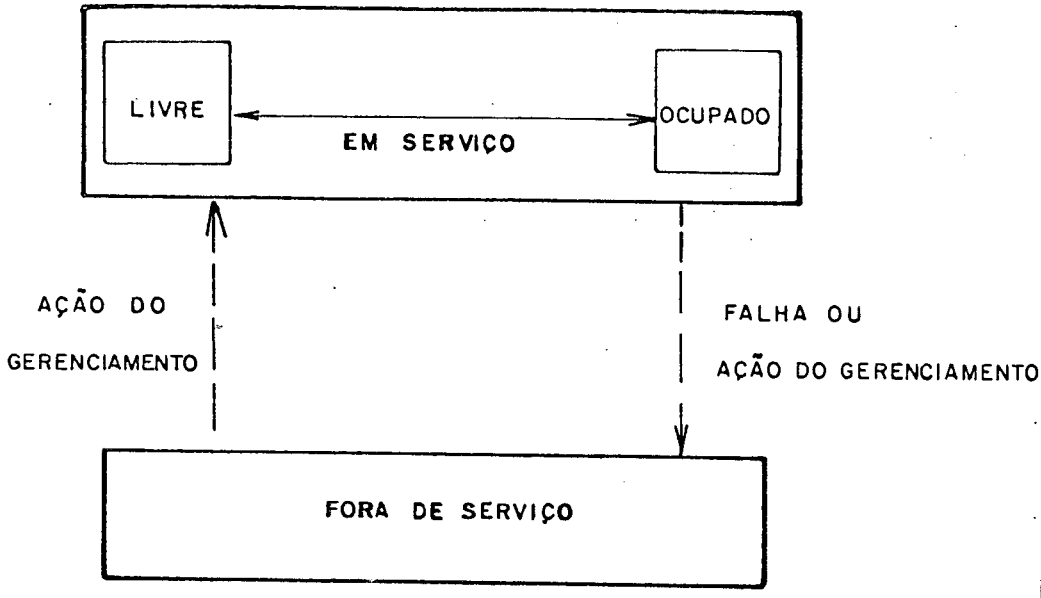


Figura 4.3. Estados do Serviço Vistos pelo Usuário

4.4.1.2. Operações no Gerenciamento de Recursos

Em geral o gerenciamento de um recurso envolve a monitoração do seu estado, a tomada de decisões baseadas na observação do estado, e a execução de ações de controle que podem modificar o estado do recurso.

Operações de Monitoração

São três as operações de monitoração: leitura de referências, leitura das variáveis de estado, e indicação das variáveis de estado.

Operações de Controle

São as seguintes as operações de controle: estabelecer referências, inicializar as variáveis de estado, instalar, remover, conectar, desconectar, habilitar, desabilitar e inicializar recursos.

4.4.2. Monitoração

A monitoração pode ser vista como um serviço de gerenciamento, pelo qual os agentes coletam informações do sistema distribuído, para que possam tomar decisões e executar

as ações de controle. Por exemplo, a monitoração do tráfego é necessária para a otimização do desempenho; a monitoração do estado corrente das entidades, mudanças de estado e relatório de erros é necessário para o tratamento de faltas; a monitoração das mudanças da configuração, eventos não usuais e mudanças de estado, podem ser importantes no propósito da segurança de dados. Os agentes responsáveis pela obtenção e síntese destas informações são normalmente referenciados como **agentes da monitoração.**

A dificuldade de se obter, num sistema distribuído, um tempo completamente sincronizado, torna difícil uma visão consistente do estado operacional do sistema global. Os dados da monitoração devem ser etiquetados na fonte, no sentido de estabelecer uma ordem cronológica global. As técnicas utilizadas nem sempre garantem um tempo global sincronizado. Alternativamente, os dados da monitoração poderiam ser etiquetados quando recebidos pela entidade supervisora, o que também poderia introduzir inconsistências, devido aos atrasos na comunicação.

As técnicas para a monitoração do sistema distribuído podem ser divididas em três categorias: a centralizada, a distribuída e a híbrida (Jacobson et alii, 1987).

A monitoração centralizada se apresenta como a solução mais natural, para sistemas distribuídos baseados em redes do tipo 'broadcast'. Uma entidade supervisora pode ser alocada para executar as funções agente de monitoração central, o que

restringe a ação da monitoração ao subsistema de comunicação. Para a monitoração do subsistema de comunicação existem duas categorias de agentes da monitoração:

- **agente de prova**, é considerado um monitor ativo pois, interfere diretamente com o funcionamento normal da rede, através da injeção de pacotes de dados que têm como objetivo o levantamento dos seguintes parâmetros: atraso na transmissão, número de colisões, e atraso no acesso ao meio de transmissão. A resolução da monitoração é dependente da frequência na qual pacotes são injetados na rede, e observados;

- **agente espião**, é considerado um monitor passivo pois, somente recebe e analisa todos os pacotes que trafegam na rede, não interferindo no seu funcionamento. Para uma monitoração completa, este monitor deve apresentar uma alta capacidade de processamento e grande espaço para armazenamento (Ayache et alii, 1982), (Molva et alii, 1985).

Embora os agentes da monitoração, de prova e espião, tendem a ser simples e ter um baixo custo de implementação, estes não apresentam condições de fornecer todo o tipo de informação necessária para o gerenciamento do sistema distribuído; pois as entidades restantes (recursos) do sistema, não têm condições de participar da monitoração. Algumas informações como, por exemplo, as relativas ao funcionamento de uma determinada

entidade e ao tempo de entrada de um pacote na rede podem ser impossíveis de se obter.

Na monitoração distribuída, as informações relativas ao tráfego da rede e ao funcionamento de cada entidade computacional, são facilmente disponíveis; cada entidade é responsável pela coleta e envio dos dados para a entidade supervisora. Esta última, somente realiza a análise dos dados, enviados pelos agentes da monitoração distribuídos.

Na monitoração distribuída, existem duas categorias de agentes da monitoração, aplicados, especificamente, ao subsistema de comunicação: os de hardware e os de software. Estes últimos apresentam algumas desvantagens no sentido que, introduzem uma maior carga no processamento, por entidade da rede.

De modo geral, a monitoração distribuída apresenta algumas desvantagens. Caso seja utilizado o mesmo canal na transmissão das informações de gerenciamento e das informações normais da rede, podem surgir dois problemas: o primeiro, devido a grande quantidade de informação transmitida para a entidade supervisora, o canal de transmissão poderá se tornar sobrecarregado, e o segundo, caso a rede falhe, irá interromper, também, o envio das informações relativas à monitoração. Para a execução das funções relativas à monitoração, é necessário a introdução de modificações, tanto no hardware como no software, a nível das entidades da rede. Para a monitoração distribuída é

fundamental a sincronização dos relógios tempo real, em cada interface da rede.

A monitoração híbrida consegue combinar as características principais das duas técnicas anteriores. Possibilita monitorações precisas e compreensivas, sem introduzir muitas modificações nas interfaces das entidadés, e nem sobrecarregar o subsistema de comunicação com informações da monitoração. As informações de gerenciamento na entidade supervisora, são obtidas pela monitoração direta dos recursos do sistema distribuído, mais os dados enviados pelas entidades (recursos).

4.4.3. Base das Informações do Gerenciamento

A Base de Informações tem como objetivo facilitar a interação entre as várias atividades relativas ao gerenciamento do sistema, através do acesso fácil e flexível a todas as informações de gerenciamento. Esta Base de Informações do gerenciamento, é constituída de bases de dados locais às entidades do sistema, e de uma base de dados central, na entidade supervisora, contendo informações de todo o sistema.

Na base de dados distribuída serão mantidas informações relativas às atividades de gerenciamento local em cada entidade: estado operacional dos componentes, levantamentos estatísticos do desempenho, versão dos componentes, estado operacional dos periféricos acoplados, informações sobre a configuração, etc. As

entidades manterão também na base de informações, os dados referentes ao gerenciamento do subsistema de comunicação.

A base de dados central, na entidade supervisora, mantém informações relativas ao gerenciamento do sistema distribuído: configuração e estado operacional do sistema, código para testes de diagnóstico, informações sobre mapeamento de nomes e endereços, e o armazenamento de informações relativas a eventos ocorridos no sistema.

4.4.4. Interface Homem-Máquina

A interface homem-máquina possibilita ao administrador do sistema, o acesso às atividades do gerenciamento, tais como, informação sobre serviços, tomada de decisões e ações de controle. Tem que ser projetada de tal modo que a informação fornecida seja facilmente entendida pelo operador humano, sempre observando os critérios que se aplicam às condições anormais de funcionamento, e que são representadas pela falha nos componentes do sistema e pela ultrapassagem de certos limites operacionais. Por isso, é necessário que esta se concentre em cima de situações críticas que possam surgir, para que as causas de uma condição anormal no funcionamento possam ser detectadas num curto espaço de tempo e sem ambiguidades, e se obtenha uma resposta rápida nas ações de correção (Wensley, 1983). Outro fator que deve ser previsto no projeto da interface homem-máquina é a facilidade na sua utilização.

Devido à quantidade de informação que pode ser gerada, é essencial uma filtragem e um processamento da informação para um formato mais adequado às necessidades do operador humano. Outrossim, este tem que ter condições de selecionar a informação que necessita, e especificar como deverá ser representada. A interface homem-máquina deverá ser orientada a menus, para limitar, ao mínimo, a intervenção do operador, e qualquer informação a ser inserida tem que ser sintaticamente corrigida. A intervenção do operador no sistema tem que ser limitada, o tanto quanto possível, devido a duas razões:

- reduzir a oportunidade de cometer ações impróprias (falhas humanas);

- eliminar a necessidade de monitorar constantemente as ações do operador.

Para facilitar, ao administrador, a tomada de decisões no gerenciamento, é necessário o emprego de ferramentas automáticas, como por exemplo, simuladores, para o tratamento de situações não usuais, ou mesmo, sistemas peritos.

4.5. Gerenciamento do Subsistema de Comunicação

O Gerenciamento do Subsistema de Comunicação, um subconjuntodo Gerenciamento do Sistema Distribuído, tem como objetivo, manter a operação do subsistema de comunicação, e permitir o seu desenvolvimento. Isto abrange, desde atividades

normais da comunicação, como o controle de acesso ao meio e o gerenciamento de conexão, até atividades de longo prazo, tais como, configuração do subsistema de comunicação, análise de desempenho, localização de faltas e manutenção do sistema.

A área de gerenciamento de redes foi abordada dentro do Modelo de Referência de Interconexão de Sistemas Abertos (RM-OSI), considerando fundamentalmente o gerenciamento de sistemas abertos autônomos e heterogêneos (Langsford et alii, 1983), (OSI,1981), (Wakid et alii, 1987). Contudo, as atividades de gerenciamento consideradas no RM-OSI não atendem satisfatoriamente aos requisitos de sistemas distribuídos com atributos de tempo real, voltados por exemplo, para a automação de usinas e subestações. Nestes sistemas os serviços são direcionados para uma aplicação em particular, com os componentes trabalhando de maneira integrada para execução dos serviços. Isto tem como resultado uma atividade de comunicação intensa, sujeita a requisitos de desempenho, confiabilidade e flexibilidade.

4.5.1. Níveis de Gerenciamento

A estrutura funcional adotada para o Gerenciamento do Subsistema de Comunicação, é a mesma do Gerenciamento de Sistemas Distribuídos. As atividades relacionadas ao gerenciamento dos recursos OSI, compreendem três níveis de gerenciamento (Wakid et alii, 1987): Gerenciamento de Protocolo, Gerenciamento de Camada e Gerenciamento de Sistema.

Gerenciamento de Protocolo

Consiste dos mecanismos internos ao protocolo, dentro de uma das sete camadas, necessários para controlar uma instância particular de comunicação. O gerenciamento do protocolo está descrito nos padrões de cada camada e serviços do protocolo de comunicação.

Gerenciamento de Camada

Consiste das atividades necessárias ao gerenciamento de todos os recursos OSI associados a uma camada particular do protocolo, podendo afetar várias instâncias de comunicação.

O gerenciamento de camada é realizado pelas Entidades de Gerenciamento de Camada ('Layer Management Entities' - LME). Uma LME está associada a cada camada do protocolo, sendo responsável pela:

- observação de informações específicas da camada, tais como: variáveis de estado, operações de protocolo, desempenho, e eventos (mudanças de estado, sinalização de excessões);

- parâmetros de protocolos e recursos a serem carregados, controlados ou estabelecidos;

- decisões específicas de camada, baseadas na observação do protocolo, realizadas tanto localmente como remotamente, por outras LMEs associadas a outras estações, da camada sendo gerenciada.

Gerenciamento de Sistema

Consiste daquelas atividades necessárias ao gerenciamento da totalidade dos recursos OSI, associados com alguma ou todas as camadas do protocolo, dentro de sistemas abertos.

O gerenciamento do sistema é sustentado pelas Entidades de Aplicação do Gerenciamento de Sistema ('System Management Application Entities' - SMAE), residentes na camada de aplicação. O SMAE pode ser considerado como um suporte aos serviços e protocolos para o gerenciamento da informação em sistemas abertos.

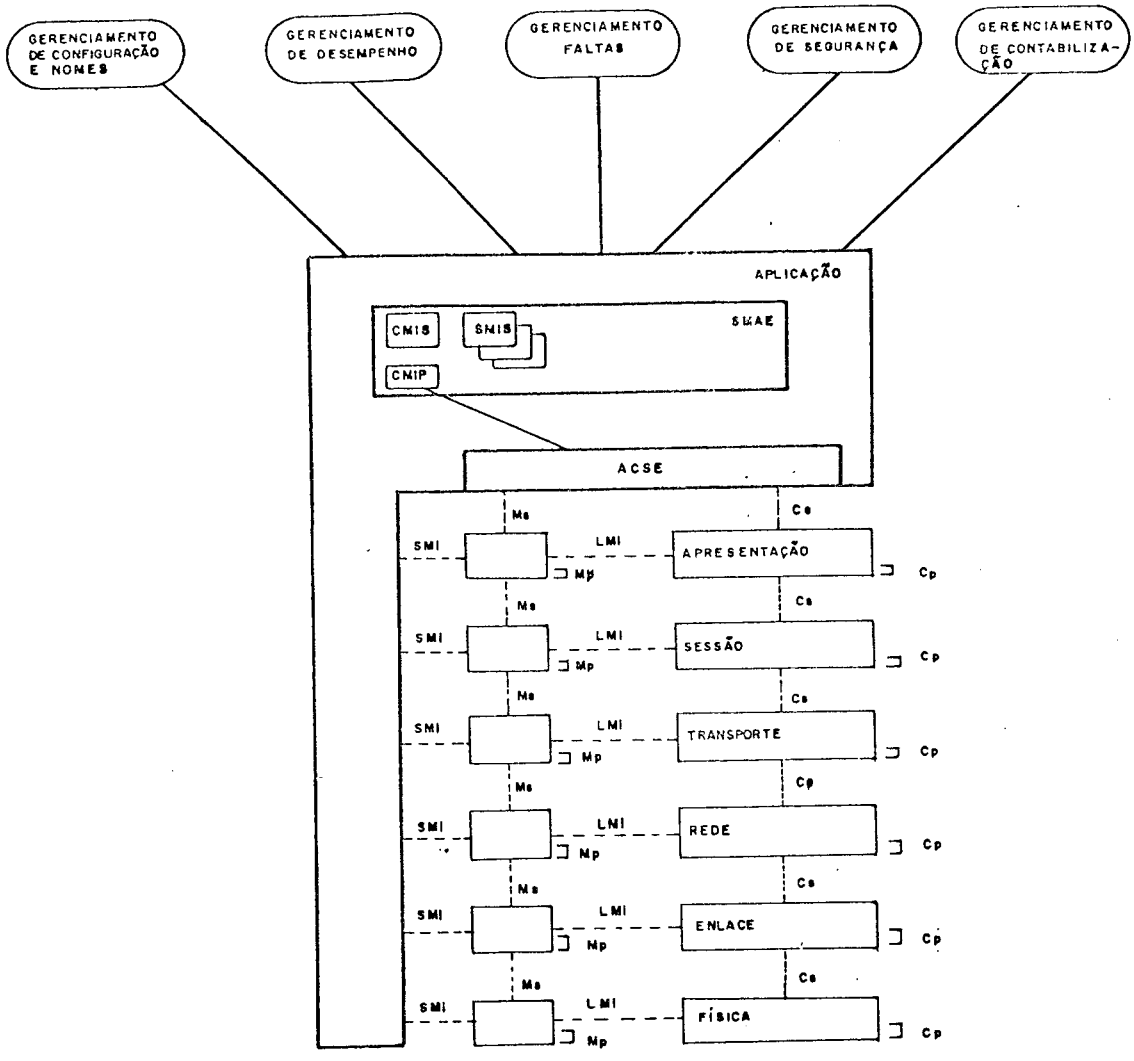
4.5.2. Modelo de Gerenciamento do Subsistema de Comunicação

A ordenação hierárquica das funções de gerenciamento é obtida pela introdução das camadas de gerenciamento (Sloman, 1984). As entidades dentro de uma camada de gerenciamento se comunicam com o objetivo de realizar um conjunto específico de serviços de gerenciamento. Cada camada de gerenciamento adiciona serviços aos já fornecidos pelas camadas mais baixas, tal que, a camada mais alta de gerenciamento tem todos os serviços

necessários para que possa gerenciar as aplicações distribuídas. Esta estrutura em camadas, das funções de gerenciamento, está representada na figura 4.4, onde cada camada de gerenciamento executa todas as funções específicas à camada.

A interação entre as entidades de comunicação e gerenciamento dentro de uma camada é chamada de Interação de Gerenciamento de Camada ('Layer Management Interaction' - LMi). As entidades de gerenciamento de camada, em diferentes estações, se comunicam pelos Protocolos de Gerenciamento da Camada-N ('N-Layer Management Protocols' - Mp) via os protocolos de gerenciamento do sistema na camada de aplicação, que, por sua vez, utilizam os protocolos de comunicação das sete camadas inferiores. Portanto, o serviço oferecido por uma camada à sua superior, conta com um componente de comunicação (Cs) e um componente de gerenciamento (Ms).

Embora o gerenciamento do sistema seja normalmente implementado na camada de aplicação, este necessita de um acesso direto aos serviços fornecidos tanto pelas entidades de comunicação como pelas de gerenciamento, das camada inferiores. Isto requer a adoção de uma "camada vertical", não hierarquizada, dentro da estrutura do RM-OSI. A Interação no Gerenciamento da Informação ('Systems Management Interaction' - SMi) representa a interação entre as entidades de gerenciamento do sistema e as entidades de gerenciamento de camada. Os SMAE se comunicam entre si, através do Protocolo Comum para o Gerenciamento do Sistema ('Common Management Information Protocol' - CMIP) no fornecimento do Serviço Comum para o



SMAE – ENTIDADES DE APLICAÇÃO DO GERENCIAMENTO DE SISTEMAS.

ACSE– ELEMENTO DO SERVIÇO COMUM DA APLICAÇÃO.

SMIS – SERVIÇO ESPECÍFICO PARA O GERENCIAMENTO DA INFORMAÇÃO - PASSANTE.

CMIS – SERVIÇO COMUM PARA O GERENCIAMENTO DA INFORMAÇÃO.

CMIP – PROTOCOLO COMUM PARA GERENCIAMENTO DA INFORMAÇÃO.

Cs – SERVIÇO DE COMUNICAÇÃO.

Cp – PROTOCOLO DE COMUNICAÇÃO.

Ms – SERVIÇO DE GERENCIAMENTO.

Mp – PROTOCOLO DO GERENCIAMENTO.

SMI – INTERAÇÃO DO GERENCIAMENTO DO SISTEMA.

LMI – INTERAÇÃO DO GERENCIAMENTO DE COMANDO.

Figura 4.4. Modelo de Gerenciamento do Subsistema de Comunicação

Gerenciamento da Informação ('Common Management Information Service' - CMIS), e do Serviço Específico para o Gerenciamento da Informação-Passante ('Specific Management Information-Passing Services' - SMIS).

Os serviços específicos no Gerenciamento do Subsistema de Comunicação, estão alocadas a um conjunto de processos de gerenciamento, centralizados ou distribuídos, que são aplicações residentes acima da camada de aplicação, e por conseguinte não fazendo parte do escopo da padronização OSI (ver figura 4.4). Estes processos da aplicação fazem parte, dentro do contexto geral do Gerenciamento do Sistema Distribuído, das funções de gerenciamento já definidas: configuração e nomes, desempenho, faltas, e segurança.

Estes processos recebem as informações obtidas tanto dos agentes locais, como dos SMAEs e LMEs locais e remotos. As decisões tomadas pelos processos de gerenciamento têm o seu efeito através de mecanismos locais aos recursos da comunicação do OSI, ou são comunicados aos recursos OSI remotos, através das respectivas SMAEs e LMEs.

O conjunto de informações, logicamente distribuídas, relativas aos recursos OSI, e que são usadas pelas funções de gerenciamento, é chamado de Base de Informações do Gerenciamento (MIB). Estes dados tanto podem fazer parte da camada de protocolo, como de bases de dados que suportam os processos da aplicação do gerenciamento do sistema.

4.6. Atividades de Gerenciamento do Sistema Digital

A partir do modelo genérico para o gerenciamento de sistemas distribuídos, serão descritas em seguida as principais atividades que poderão ser incorporadas no gerenciamento do Sistema Digital de Controle Distribuído para a automação de Usinas/Subestações.

4.6.1. Atividades do Gerenciamnto da Configuração

As atividades da configuração para o Sistema Digital ficarão restritas ao levantamento da configuração física e lógica do sistema, instalação de componentes de hardware, e o registro de mudanças e eventos que ocorram no sistema. Esta simplificação nas atividades está relacionada à arquitetura sem redundâncias e ao fato que os programas ficarão residentes nas unidades computacionais.

4.6.2. Atividades do Gerenciamento do Desempenho

As atividades do gerenciamento do desempenho são de fundamental importância para se acompanhar o comportamento do sistema, e compará-lo com o inicialmente pretendido. Para tanto, será necessário a incorporação de atividades relacionadas à monitoração e análise do desempenho.

Um exemplo das atividades de desempenho é o cálculo das medidas de segurança de funcionamento, relativas às unidades computacionais e ao subsistema de comunicação, baseando-se no estado operacional e registro de eventos do sistema.

4.6.3. Atividades do Gerenciamento de Faltas

Mesmo sendo um sistema sem redundâncias, e por conseguinte, com possibilidades limitadas de tolerância a faltas, é necessário que sejam fornecidos mecanismos de modo a evitar ações inadivertidas no processo (US/SE). Das quatro fases básicas que compõem a tolerância a faltas, será omitida, a seguir, a fase de confinamento de erros e avaliação de danos, que está relacionado à estrutura do sistema, e que por conseguinte, sai do escopo deste trabalho.

a. Detecção do Erro

A detecção de erros está associada à detecção de estados errôneos, tanto ao nível interno das unidades computacionais, como no subsistema de comunicação. Para tanto é necessário o emprego de técnicas de detecção de erro, que têm de ser suficientemente efetivas, sem contudo interferir de maneira significativa ou impor degradação, nos serviços a serem fornecidos pelo sistema.

A detecção de erros nas unidades computacionais tem como objetivo evitar que estes erros se propaguem pelo restante do sistema, tornando difícil, primeiro, a recuperação integral do sistema, e segundo, a localização de faltas para posterior tratamento. Nestas condições, a detecção de erros, sendo realizada a nível de componentes da unidade computacional, torna-se mais efetiva, permitindo que a causa (a falta) seja facilmente localizada.

Os testes, para efeitos de diagnose interna das unidades computacionais, devem ser efetuados periodicamente, em paralelo com o processamento normal, ou quando requisitados remotamente através de um comando externo emitido pela entidade supervisora. A cada inicialização de uma unidade computacional são realizados testes internos para verificação da sua operacionalidade. Este procedimento é importante, para evitar que a operação normal do sistema seja interrompida quando da conexão no sistema de uma unidade com falta.

Os resultados dos testes e levantamentos, são mantidos na Base de Informações contendo o estado corrente dos componentes que formam uma unidade computacional (incluindo as interfaces com o subsistema de comunicação). Qualquer mudança detectada no estado de um componente, implicará numa atualização da Base de Informações da unidade correspondente.

Também, no Sistema Digital, cada controladora de TAC mantém uma cópia da Base de Informações dos operadores de processo interligados, onde é mantido o estado operacional de cada

operador. As bases de informações são periodicamente requisitadas pela entidade supervisora, para que o estado operacional global do sistema seja mantido atualizado.

b. Recuperação do Erro

A recuperação do erro está normalmente associada à existência de redundâncias no sistema, sejam materiais, lógicas ou temporais. Como nesta etapa do projeto do Sistema Digital, não foram previstas redundâncias materiais nem lógicas, a nível dos componentes do sistema, somente será possível a recuperação de erros devido à ocorrência de faltas transientes, pois estes casos podem fazer uso, somente, de redundâncias temporais.

Embora todas as unidades computacionais do Sistema Digital se apresentem como um ponto crítico (ponto duro de falha) aos serviços fornecidos pelo sistema, isto devido à ausência de redundâncias, existe a possibilidade de se prever redundâncias em algumas unidades computacionais do sistema onde a ocorrência de uma falha poderia afetar os serviços de todo o sistema. Por exemplo, caso sejam utilizadas várias unidades computacionais na execução das funções de supervisão, e estas, não estejam respectivamente sobrecarregas pela execução das suas funções, então, podem ser implementadas reconfigurações entre as unidades da sala controle, para que a falha em uma das unidades não leve à falha o sistema global. Inicialmente, poderiam ser levadas em consideração, as atividades de supervisão do Sistema Digital, alocadas ao OO. Estas poderiam, caso este falhasse, passarem a

ser executadas pelo Operador de Console, mesmo de forma degradada.

c. Tratamento de Faltas e Continuidade dos Serviços

As atividades do tratamento de faltas, estão relacionadas à localização e isolação das faltas, para que estas não dêm, continuamente, origem a novos erros no sistema, criando, para o gerenciamento de faltas, uma condição difícil de ser contornada, e que poderia também provocar uma falha no sistema. A localização de uma falta num componente do sistema pode ser determinada pela execução de testes de diagnóstico, análise de registros de eventos, e relatórios de desempenho. Os testes de diagnóstico são utilizados para excitar um recurso, com o objetivo de reproduzir erros, e fornecer informações mais detalhadas sobre a condição do recurso. Na diagnose de faltas, também pode ser necessário a análise da informação relativa a eventos normais, antes de detectado o erro.

- A análise que envolve grande quantidade de informação, referente a todo o sistema, necessita ser centralizada, embora a obtenção dos dados, na qual está baseada esta análise, seja distribuída. Sistemas peritos e bases de conhecimento podem ser usadas para ajudar a diagnose de faltas.

O tratamento de faltas do sistema é realizado tanto a nível do sistema global como das unidades computacionais. A nível do sistema, são aplicadas técnicas de auto-diagnose distribuída,

para a localização das unidades com falta. Este assunto, será tratado, com mais detalhe, no capítulo seguinte, onde é apresentado o modelo de diagnose do sistema, juntamente com o algoritmo de auto-diagnose. A nível de unidades computacionais, serão empregadas técnicas de detecçao de erros.

Na etapa atual, de protótipo, do Sistema Digital, devido à inexistência de redundâncias no enlace físico do subsistema de comunicação e à característica da interconexão das unidades computacionais, foi assumida uma simplificação no modelo de faltas do sistema, desconsiderando a presença de faltas nos enlaces de comunicação. Com isto, caso seja localizada uma unidade com falta, a princípio, não existem meios de confirmar se o enlace de comunicação ou a unidade, que está com falta, devido essencialmente, à existência de caminhos únicos entre estas. Em etapas futuras, prevendo-se redundâncias a nível do sistema de comunicação, existirá a necessidade de se modelar estas redundâncias, para que, caso o enlace de comunicação esteja com falta, isto não invalide o funcionamento da unidade computacional.

A localização de faltas no sistema de comunicação que, por exemplo, afetam mais de uma unidade, necessita de técnicas de análise, a nível de entidade supervisora, das condições de erro no sistema. Outra alternativa que poderia ser implementada, para este exemplo em particular, seria a adoção de mecanismos, associados às tecnologias da camada física, para a localização de faltas, por exemplo, ruptura nos enlaces de comunicação.

A isolação das unidades com falta, a nível de sistema, pode ser tanto pelo desligamento físico, como pelo desligamento lógico, com as unidades livres de faltas ignorando a existência das unidades identificadas com falta. O desligamento lógico é preferível diante o desligamento físico, pois este último necessita de uma unidade central do tipo 'hard-core' (livre de faltas), para realizar automaticamente os desligamentos na via de comunicação.

A continuidade dos serviços está relacionada à reconfiguração do sistema. Considerando-se que o sistema sendo tratado, é um sistema sem redundâncias, na eventual falha de um dos seus componentes, é necessária uma avaliação, no sentido de verificar a importâmcia dos serviços fornecidos por este componente diante dos serviços fornecidos pelo sistema. Caso seja considerado um componente crítico, então existirá a necessidade de se efetuar um desligamento seguro do sistema, para que sejam evitadas ações indevidas, no processo (US/SE), por parte do Sistema Digital.

Caso seja possível ao sistema, continuar executando suas funções, com a capacidade computacional reduzida, então se poderia propor uma reconfiguração degradada, através da eliminação de serviços que não seriam considerados essenciais. Esta abordagem poderia ser aplicada, por exemplo, a nível das unidades da Sala de Controle, onde as redundâncias lógicas entre as unidades de supervisão podem ser facilmente implementáveis, devido às arquiteturas não especializadas destas unidades; na eventual falha do OI, todas as mensagens a ele encaminhadas,

poderiam ser redirecionadas para o OC, e gravadas temporariamente em disco. Quando o OI retomasse as suas atividades, o conteúdo do arquivo seria transferido para o OI, para a impressão das mensagens.

d. Relatório de Faltas

Depois de detectado um erro e localizada uma falta, as ocorrências devem ser registradas, para que o histórico de faltas do sistema possa ser analizado, e possíveis tendências possam ser previstas. Estes registros são mantidos na entidade supervisora do sistema.

Estes relatórios deverão prover meios para notificar a ocorrência dos eventos anormais e imperfeições, tanto aos usuários (através da entidade supervisora), como também aos agentes do gerenciamento; particularmente aqueles responsáveis pela configuração do sistema.

4.6.4. Atividades do Gerenciamento da Segurança de Dados

Na etapa atual, de protótipo, do Sistema Digital para Automação de US/SE, as funções do gerenciamento da segurança poderão ser plenamente omitidas sem prejuízo algum para o sistema, diante dos objetivos inicialmente estabelecidos. Entretanto, com a perspectiva de instalação definitiva destes sistemas, como também, com a possibilidade de interligação

destes sistemas com sistemas supervisores, existirá a necessidade de serem incorporadas, às atividades de gerenciamento do sistema, as funções de gerenciamento de segurança de dados.

4.6.5. Atividades do Gerenciamento da Contabilização

Na especificação das funções do gerenciamento, é omitida a função de contabilização, pois esta, não se caracteriza por ser uma função importante para o gerenciamento do Sistema Digital.

4.6.6. Estrutura da Monitoração no Sistema Digital

O tipo de monitoração mais adequada para a etapa atual do Sistema Digital, devido a este estar particionado em subredes, e estas conterem um número reduzido de unidades, seria a monitoração distribuída, onde cada unidade computacional do sistema além de manter uma monitoração própria da sua configuração interna, estado operacional, e ocorrência de eventos não usuais, também mantém uma monitoração do subsistema de comunicação. Na figura 4.5 está representada a configuração do sistema, para uma monitoração distribuída.

Nesta configuração, os agentes da monitoração em cada entidade, serão responsáveis pela coleta de dados e o envio destes para a entidades supervisoras locais e de todo o sistema. Entretanto, uma análise em termos da monitoração centralizada,

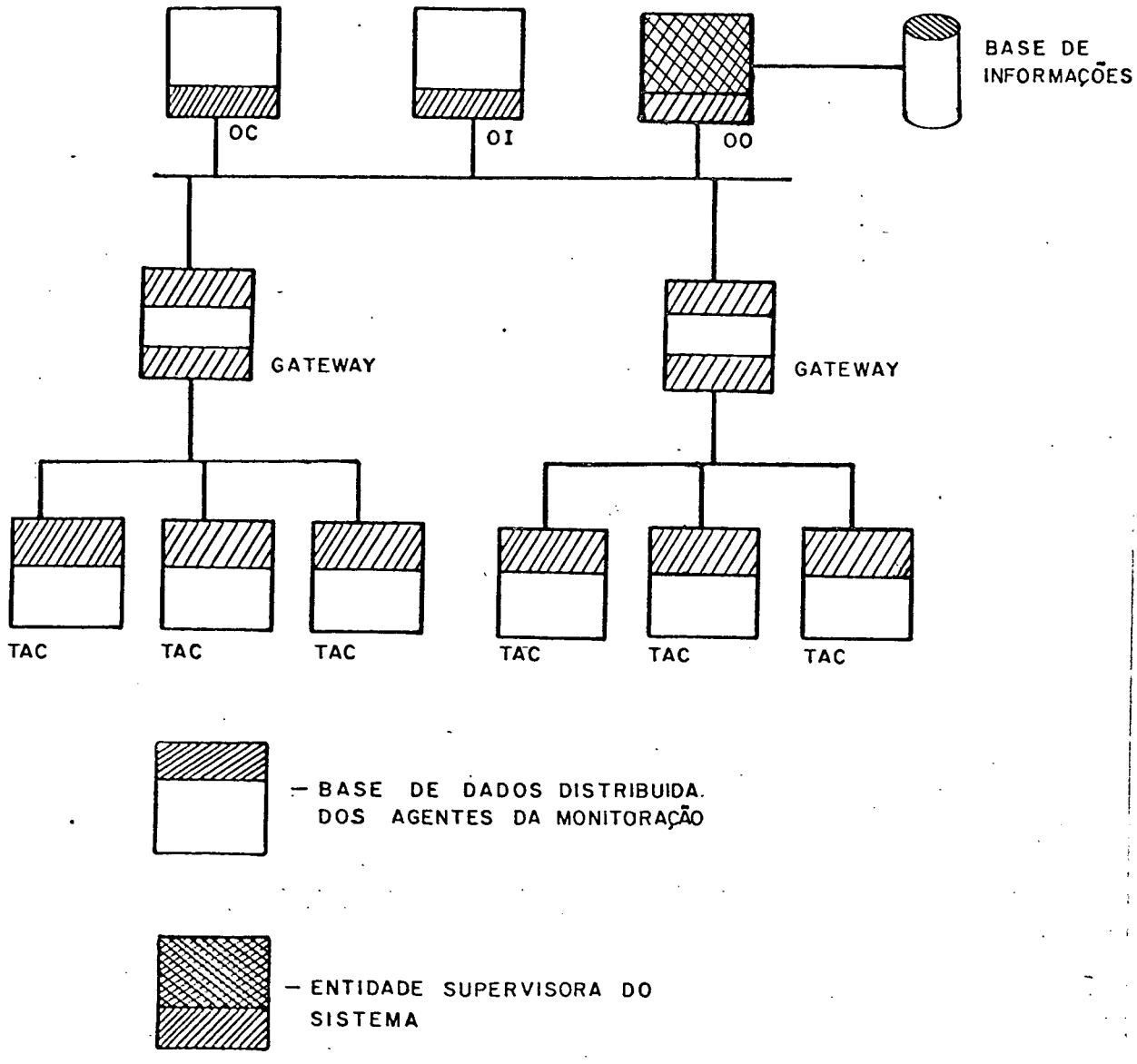


Figura 4.5. Monitoração Distribuída

indicaria duas possibilidades de implementação, ambas com os seus inconvenientes: a primeira, seria alocar uma única entidade de monitoração para todo o sistema, o que acarretaria num aumento de tráfego no sistema de comunicação, principalmente entre subredes, e a segunda, seria a alocação de uma entidade de monitoração para cada subrede, que além de ser uma solução inviável devido ao número reduzido de unidades, também implica em limitações de projeto.

Para etapas futuras do Sistema Digital, em que provavelmente o número de unidades por subrede será maior, a implementação híbrida da monitoração se torna mais adequada, pois, além de diminuir o tráfego de informação entre subredes, também diminui o processamento a ser realizado em cada entidade, relativo à monitoração. Para esta configuração do Sistema Digital, uma entidade de supervisão seria alocado para cada subrede do sistema, e uma entidade de supervisão central, seria alocada na Sala de Controle; a entidade supervisora de subrede é responsável pela coleta e análise das informações relativas a uma subrede, e o envio das informações, já analisadas, à entidade supervisora do sistema. A atividade de monitoração nas entidades restantes se restringe à coleta de informações locais à entidade. Na figura 4.6 está representada a configuração do sistema, para uma monitoração híbrida.

Este tipo de arquitetura é encontrada em outros sistemas distribuídos. A arquitetura MAP, voltada para a automação industrial apresenta este tipo de abordagem: para cada célula

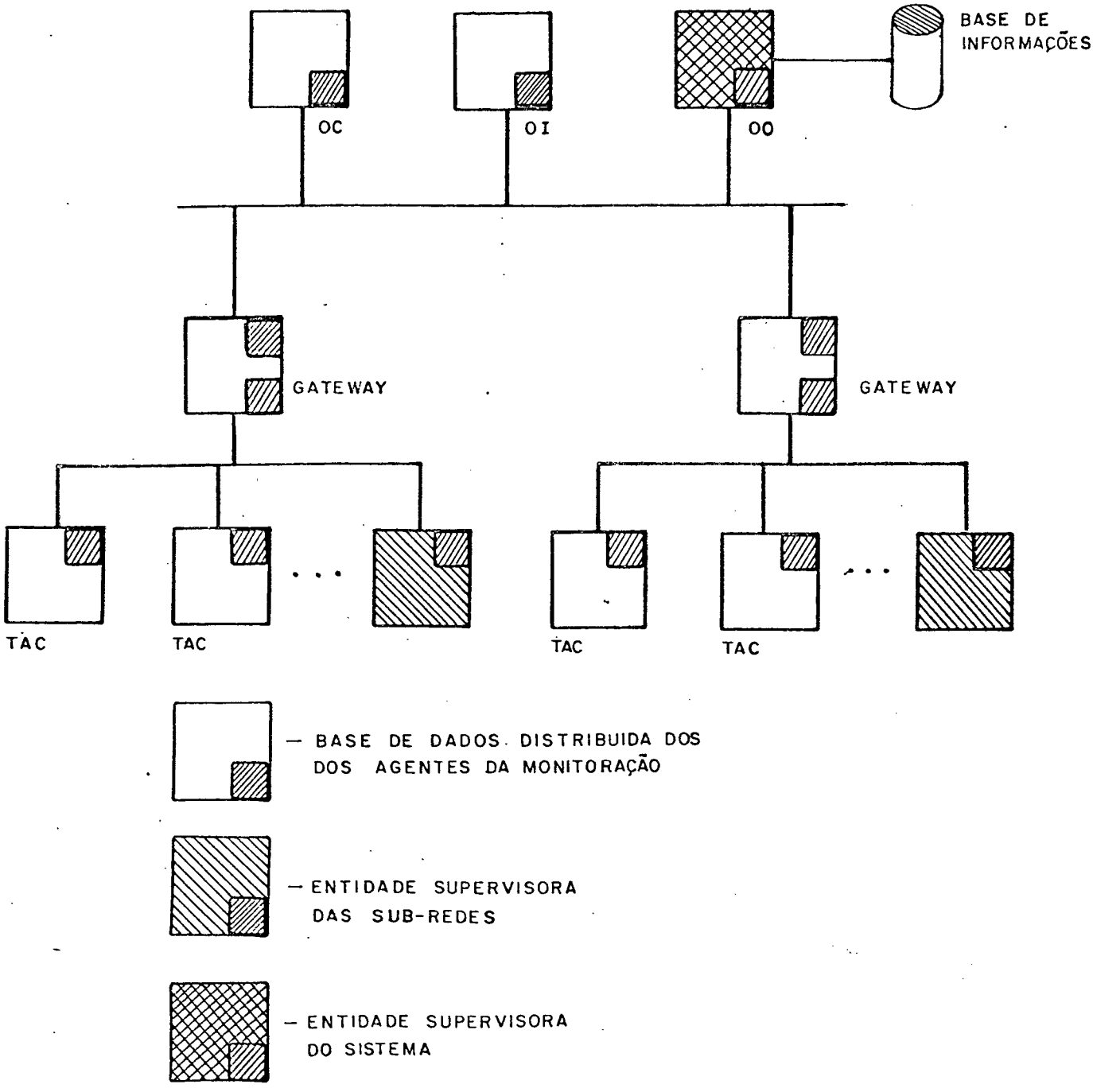


Figura 4.6. Monitoração Híbrida

flexível de manufatura é alocado um 'computador de célula', cujo o objetivo é monitorar as atividades locais à célula.

4.6.7. Operador de Observação

Dentro da Função Observação, o Operador de Observação (OO) é a entidade responsável pela interface homem-máquina e pelas atividades relacionadas à entidade supervisora do gerenciamento de sistema distribuído. O Operador de Observação, tem como objetivo colocar ao alcance do operador humano todas as informações sobre o estado operacional do Sistema Digital, que correspondem tanto à sinalização local em cada unidade computacional, como também, à visão global do sistema através da entidade supervisora.

A nível do sistema global, a interface homem-máquina compreende a execução das seguintes atividades.

a. apresentar as informações relativas ao gerenciamento da configuração:

   - o estado operacional do sistema global e por unidade (no caso das controladoras de TAC fornecer também as informações correspondentes aos seus respectivos operadores de processo),

   - um histórico da operacionalidade do sistema a partir do registro de eventos;

b. apresentar a análise de desempenho do sistema, a partir dos levantamentos estatísticos;

c. armazenar todas as informações relativas à ocorrência de eventos no sistema;

d. facilitar a manutenção preventiva, através do comando remoto de testes de diagnose, nas unidades do sistema, e por último;

e. notificar através de alarmes, as condições anormais no funcionamento do sistema, sejam, falhas nas unidades computacionais ou ultrapassagem de limites operacionais do sistema.

A sinalização local, em cada unidade computacional, é necessária para que o operador humano, responsável pela manutenção, identifique facilmente um circuito danificado, tornando o processo de reparo mais rápido e menos sujeito a faltas.

4.7. Conclusão

Neste capítulo foi apresentado uma estrutura funcional para o gerenciamento do Sistema Digital para a Automação de US/SE, onde foram definidas as funções básicas e de suporte do gerenciamento, referenciando-se sua aplicabilidade, quando

possível, ao Sistema Digital. Sendo que, as atividades de gerenciamento estão envolvidas com as ações de monitoração, análise e controle, necessárias para permitir e causar o processamento, o armazenamento ou a transferência de informação, sem contudo, tomar parte da atividade normal da aplicação. Por último, é abordado o gerenciamento de recursos no sistema distribuído, se atendo com mais detalhe ao Gerenciamento do Subsistema de Comunicação.

Um dos aspectos que normalmente não é comentado na literatura, é o da segurança ('safety'), que trata com os eventos catastróficos que os serviços do sistema podem causar, no momento da ocorrência de uma falha maligna (Levenson, 1985). No Sistema Digital para Automação de US/SE, tratado neste trabalho, existirá a necessidade de se preverem requisitos de segurança para o sistema, devido às consequências catastróficas, em termos de prejuízos materiais e humanos, que uma falha desta natureza pode causar no sistema.

CAPÍTULO 5

DIAGNOSE DE FALTAS NO SISTEMA DISTRIBUÍDO

5.1. Introdução

Um sistema depois que retorna a um estado livre de erros, deve continuar fornecendo os serviços conforme especificados. Para isso, existem técnicas que garantem a não reincidência de faltas, de cujos os efeitos o sistema se recuperou, pois estas podem se tornar novamente ativas, criando novos erros.

O tratamento de faltas tem como objetivo, primeiro, localizar as faltas presentes no sistema e segundo, reparar o sistema, para que este continue fornecendo os serviços conforme especificados.

Uma das técnicas empregadas para a localização de faltas num sistema é a técnica da diagnose, que por sua vez, está voltada especificamente para a verificação do comportamento dos componentes de um sistema, ao invés de verificar o comportamento deste sistema. Esta verificação do comportamento dos componentes, está baseada na interpretação dos resultados de

testes, como também, do levantamento de eventos, efetuados durante o funcionamento normal do sistema, com o objetivo de localizar e identificar, de forma precisa, as faltas presentes no sistema. A diagnose a ser realizada num sistema será mais eficiente, quando a sua execução for mais próxima das condições normais de processamento.

Mesmo considerando sistemas não tolerantes a faltas, a diagnose de faltas se caracteriza por ser um aspecto importante na estratégia de manutenção. Esta consideração vem no sentido de que, a manutenção é efetuada de forma pessoal, e normalmente não automatizada, tornando-se assim, dentro das fases do ciclo de vida de um sistema, uma etapa de alto custo. Daí vem o fato que, um projeto visando uma limitação nos custos de manutenção, deve possibilitar boas condições de teste, como também, ter ferramentas de diagnóstico eficientes.

A diagnose tanto pode ser realizada, toda a vez que é detectado um erro no sistema, com o objetivo de localizar a falta para a sua posterior remoção, como também, pode ser realizada periodicamente, com o objetivo de localizar as faltas que por ventura não tenham provocado um estado errôneo no sistema, passível de ser detectado.

Em sistemas distribuídos, onde as funções podem ser executadas por vários processadores, existindo assim uma redundância intrínseca, a diagnose de faltas tem como objetivo a

localização destas, para que o sistema possa ser reconfigurado e com isto, continuar oferecendo os serviços, conforme especificados.

Este capítulo apresenta no primeiro item, uma comparação entre as técnicas para a diagnose de faltas em sistemas distribuídos. No segundo item, é apresentada uma introdução ao tema diagnose de faltas em sistemas distribuídos, através da discussão de alguns modelos e algoritmos empregados na diagnose de faltas; isto é necessário para o entendimento do algoritmo de diagnose a ser proposto. No terceiro item são inicialmente fornecidas as razões que levaram à especificação deste algoritmo, juntamente com o modelo de diagnose de faltas adotado. Além da descrição do algoritmo, é realizada a sua formalização por Redes de Petri. E por último, são fornecidos alguns aspectos práticos, relacionados à implementação deste algoritmo, no sistema distribuído em questão.

5.2. Técnicas de Diagnose de Faltas em Sistemas Distribuídos

Com relação às técnicas de diagnose de faltas para sistemas distribuídos, existem duas possibilidades de implementação: a centralizada e a descentralizada. A técnica centralizada, necessita de uma entidade, normalmente denominada de "observador central", responsável pela realização da análise dos resultados dos testes. Dentro do modelo de diagnose, o "observador central"

é assumido como sendo uma entidade não participante do modelo e não sujeita a faltas (entidade 'hardcore'). Nos casos em que existem requisitos severos com relação à disponibilidade do sistema, esta configuração se torna difícil de ser implementada, considerando as limitações em técnicas de projeto e de validação para sistemas complexos.

Mesmo assim, para implementações que adotam este tipo de configuração, algumas considerações têm que ser feitas, no sentido de possibilitar ao "observador central" resolver os conflitos criados, por exemplo, por falhas nas entidades do sistema, ou mesmo, nele próprio. Alguns conflitos e soluções para falhas que ocorram nas entidades do sistema e na comunicação, são descritas em (Liebowitz & Carson, 1985), e serão tratados neste capítulo sob a ótica de modelos de diagnose de faltas em ambientes distribuídos. Um conflito, particular a sistemas que adotam a diagnose centralizada, é a falha do "observador central"; neste sentido, duas situações são possíveis:

a. uma falta que afete parcialmente o "observador central" pode ser perigosa pois este, em situações de mau funcionamento, pode provocar ações destrutivas no sistema;

b. a utilização de uma entidade sobressalente no modelo "observador central" não impede e existência de conflitos potenciais. Para minimizar eventuais problemas são

necessários testes e estratégias que permitam a troca de entidades, envolvidas com a função do observador central.

A outra técnica de diagnose, a descentralizada, executa nas várias entidades de um sistema, um algoritmo de diagnose de faltas; tanto os testes, como a análise dos resultados dos testes serão executados de forma distribuída. Com isto, é assumido a existência de um "observador distribuído" que não tem uma visão global e consistente de todo o sistema, o que consequentemente, adiciona uma certa complexidade ao processo de diagnose.

Como este capítulo está voltado para a abordagem de algoritmos de diagnose de faltas para sistemas distribuídos, será considerado como ambiente de diagnose distribuída ou auto-diagnose de sistemas distribuídos, um sistema no qual as funções relativas à diagnose de faltas, são executadas de forma descentralizada, por várias entidades interligadas por enlaces de comunicação, como por exemplo, uma rede local. É através destes enlaces que serão executados os testes, tanto entre as entidades como no subsistema de comunicação. Os resultados dos testes serão disponíveis às entidades envolvidas com a diagnose, de modo que cada entidade livre de faltas possa ter condições de determinar, através de um algoritmo distribuído, o estado operacional de todas as entidades do sistema. Conforme comentado anteriormente, este tipo de diagnose é mais complicada pois, nem todos os resultados dos testes, em certas situações, podem estar

disponíveis numa determinada entidade executando a diagnose, como também, nenhuma entidade é confiável o suficiente, para executar a diagnose sem estar sujeita a erros ou mesmo, a faltas maliciosas.

Para um ambiente distribuído, se adotando algoritmos descentralizados, é assumido que cada resultado de teste é gerado numa entidade denominada **origem do teste**, e comunicado para outras entidades denominadas **analizadores**, responsáveis pela obtenção e análise dos resultados dos testes. Estes analizadores têm como objetivo determinar as entidades com falta no sistema. Os resultados do diagnóstico são comunicados aos **controladores**, entidades responsáveis pelas ações de reparo no sistema (Holt & Smith, 1985).

5.3. Diagnose de Faltas em Sistemas Distribuídos: Modelos e Algoritmos

A localização de faltas pela diagnose, através da formulação de um modelo bem definido, começou em meados de 1960. De início, devido ao hardware dos sistemas ser basicamente composto de componentes discretos e de integração a pequena escala, os modelos se preocupavam com as faltas nos circuitos combinacionais. No passo seguinte, a diagnose a nível de subsistema tornou-se importante, no sentido de que os sistemas começaram a ser subdivididos em unidades que podiam ser

facilmente trocadas. Esforços foram realizados para tornar possível o projeto, sistemático, de procedimentos para diagnose de sistemas, e analizar, também sistematicamente, dados relativos a testes. Motivando inclusive, alguns trabalhos teóricos na formulação de modelos para a diagnose de sistemas (Preparata et alii, 1967) e (Kime, 1970). Outro fator determinante para esta linha de trabalhos, foi a crescente penetração de sistemas tolerantes a faltas em diferentes áreas de aplicação, implicando na necessidade de adoção de redundâncias em níveis funcionais cada vez mais altos (Kime, 1985).

Como introdução à proposta de um algoritmo de diagnose para sistemas distribuídos, a ser tratado ainda neste capítulo, serão apresentados neste item alguns modelos e algoritmos de diagnose, normalmente tratados na literatura. Inicialmente, será abordado o modelo de Preparata, Metze e Chien (1967), ou modelo PMC, que embora não tenha sido concebido para a modelização de sistemas distribuídos, foi o primeiro modelo que adotou uma abordagem mais rigorosa na relação de testes e faltas através de grafos orientados (dígrafos). A abordagem adotada deu origem ao surgimento de novos modelos de diagnose, que tentaram complementar, pela generalização, o modelo PMC. Alguns destes modelos serão tratados em seguida.

5.3.1. Modelo PMC

O modelo PMC (Preparata et alii, 1967), pode ser resumido nas seguintes considerações:

a. um sistema é particionado em entidades, aonde cada uma pode testar individualmente outra entidade;

b. os resultados dos testes classificam uma entidade do sistema como estando livre de faltas ou com falta;

c. o resultado do teste é sempre correto se a entidade testadora é livre de faltas, de outro modo, o resultado é imprevisível.

A última consideração, corresponde ao que usualmente se chama de invalidação **simétrica** (Smith, 1979). Na invalidação **assimétrica**, se a entidade realizando o teste falha, então, indepentemente do estado da entidade sendo testada, o resultado do teste vai sempre indicar a presença de faltas.

Neste modelo, cada teste envolve a aplicação controlada de um estímulo e a observação dos resultados correspondentes. Ao conjunto de resultados dos testes realizados no sistema, se dá o nome de **síndrome**; através da codificação da síndrome são identificadas as entidades com falta do sistema.

Em (Preparata et alii, 1967) são definidas duas medidas para a diagnose de faltas em sistemas, descritas em seguida.

a. Sistemas t-Faltas Diagnosticáveis num Passo

Um sistema de n entidades é **t-faltas diagnosticáveis num passo**, se todas as entidades com falta dentro do sistema podem ser identificadas, sem troca, desde que o número de entidades com falta não exceda a t.

b. Sistemas t-Faltas Diagnosticáveis Sequencialmente

Um sistema de n entidades é **sequencialmente t-faltas diagnosticáveis**, se pelo menos uma entidade com falta pode ser identificada sem a necessidade de troca de entidades, desde que o número de entidades com falta não exceda a t.

Na literatura, também são normalmente referenciadas, respectivamente, como, sistemas t-faltas diagnosticáveis sem reparo e sistemas t-faltas diagnosticáveis com reparo (Friedman & Simoncini, 1980).

Neste modelo, para um sistema ser t-faltas diagnosticável num passo, é demonstrado que o número n de entidades tem que ser pelo menos 2t+1, e que uma entidade tem que ser testada por pelo menos t outras entidades. A partir destas condições de necessidade, um sistema ótimo t-faltas diagnosticável num passo

pode ser definido, com n=2t+1, e tendo cada entidade testada por exatamente t outras entidades. Dentro dos vários projetos ótimos, foi considerada uma classe de projetos na qual a conexão de testes em cada entidade é idêntica. Um sistema então, é dito pertencer a um projeto $D_{\delta t}$ quando um enlace de testes de u_i para u_j existe, se e somente se $j-i=\delta m$ (módulo n), onde m assume os valores 1,2,...,t, e δ corresponde ao passo entre os enlaces. Um sistema é t-faltas diagnosticável num passo se emprega o projeto ótimo $D_{\delta t}$, com $(\delta,n)=1$ (i.e. δ e n são primos entre si). Vide figura 5.1, para uma conexão D_{1t} dos enlaces de testes.

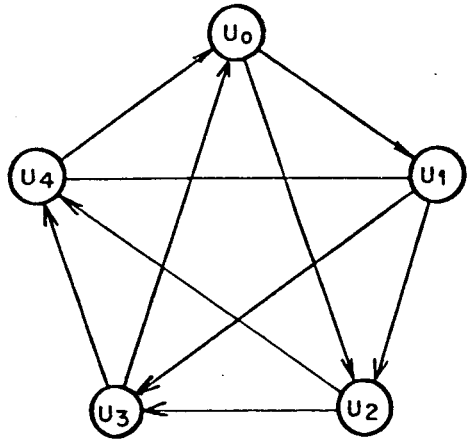


Figura 5.1. Conexão D_{1t}

A identificação de todas as entidades com falta, num sistema t-faltas diagnosticáveis sequencialmente, pode envolver um procedimento de várias etapas. Na primeira iteração, pelo menos uma entidade com falta é identificada; depois da troca desta entidade, os testes são novamente executados para a identificação de outras entidades com falta. Este processo é

repetido até que todas as entidades com falta sejam identificadas e trocadas. O estudo de sistemas sequencialmente diagnosticáveis é motivado pelo fato que menos enlaces de testes são necessários, se comparando com os sistemas t-faltas diagnosticáveis num passo.

O estudo do modelo de Preparata, Metze e Chien se desenvolveu ao longo de duas linhas principais: primeira, a derivação analítica das condições de necessidade e suficiência para um sistema ser tanto t-faltas diagnosticáveis num passo ou sequencialmente, e a segunda, síntese de sistemas t-faltas diagnosticáveis sequencialmemte (Friedman & Simoncini, 1980). Embora o modelo PMC não possa ser diretamente aplicado a sistemas reais, este lançou a base para trabalhos subsequentes, que generalizaram o modelo ao adicionar outras restrições, associadas a sistemas reais. Em (Friedman & Simoncini, 1980) são descritos alguns destes modelos, que surgiram posteriormente ao trabalho de Preparata, Metze e Chien, com o objetivo de aperfeiçoá-lo.

5.3.2. Algoritmos de Diagnose

Um algoritmo de diagnose, independentemente do modelo adotado e da resolução disponível, provê um meio efetivo de determinar as faltas presentes num sistema, através da análise dos resultados de testes.

São apresentados, em seguida, como termo de comparação, quatro algoritmos de diagnose para sistemas distribuídos normalmente referenciados na literatura. Os dois primeiros algoritmos, estão baseados no modelo PMC, o terceiro algoritmo, introduz uma modificação na representação do modelo pela utilização de dígrafos bipartidos, e no último, o modelo é representado por um grafo acíclico.

a. Algoritmo de Meyer e Masson

Este algoritmo (Meyer & Masson, 1978), baseado no modelo PMC, trata de sistemas t-faltas diagnosticáveis num passo, para projetos de interconexão do tipo $D_{\delta t}$, com invalidação simétrica. Este algoritmo assume que existe n entidades no sistema ($n > 2t+1$), cada uma testada por no mínimo t entidades, e com no máximo t entidades com falta.

Cada entidade u_i contém uma tabela com resultados de testes B_i (n x n), que representa a visão de u_i com respeito ao estado das restantes entidades do sistema. Para a montagem desta tabela, cada entidade, independente do seu estado, se considera livre de faltas, e preenche parte da tabela com os resultados dos testes realizados por ela própria, e com os enviados por outras entidades. A partir destes resultados cada entidade preenche o restante das posições, através do processamento da

síndrome (Meyer & Masson, 1978), com isto, é garantido a existência de n-t tabelas idênticas no sistema. A localização das entidades com falta é obtida através de uma votação que é realizada em cada entidade em cima dos resultados dos testes da sua respectiva tabela.

b. Algoritmo de Kuhl e Reddy

Neste artigo (Kuhl & Reddy, 1980a), o modelo de diagnose abordado, além de representar o sistema através de dígrafos, a exemplo do modelo PMC, também define um grafo do sistema (não orientado), para representar os enlaces físicos entre as entidades. É assumido que os testes são executados através destes enlaces de comunicação, e que os arcos existentes no grafo de testes só existem entre as entidades que estão diretamente conectados no grafo do sistema. Dentro o modelo adotado os testes são invalidados somente pelas entidades executando os testes, ou seja, quando estas estiverem com falta.

Kuhl e Reddy definiram um sistema distribuído como sendo **t-faltas auto-diagnosticáveis**, se cada entidade livre de faltas pode corretamente identificar todas as entidades com e sem faltas de um sistema, desde que não mais que t entidades estejam com falta. Deste modo, todas as entidades são consideradas como analizadores, obtendo assim a sua própria visão do estado

operacional do sistema. Isto é possível pela execução do algoritmo SELF (Kuhl & Reddy, 1980a) em cada entidade, o que permite a montagem de um vetor diagnose, correspondente ao estado operacional do sistema.

O algoritmo SELF, para diagnose em sistemas distribuídos, pode ser resumido nos seguintes passos. No primeiro passo, são executados todos os testes conforme o grafo de testes. No segundo passo, independentemente do seu estado interno, a entidade se considera livre de faltas, e preenche o vetor de diagnose com os resultados dos testes realizados pela própria entidade. No terceiro passo, a entidade prepara uma mensagem com o conteúdo do seu vetor diagnose, e a envia para todas as entidades que a testam. No último passo, a entidade recebendo uma mensagem verifica se dentro do seu vetor diagnose, a entidade de origem da mensagem é considerada com falta ou sem falta; caso seja considerada com falta, o conteúdo da mensagem é desconsiderado, caso seja livre de falta, os resultados dos testes contidos na mensagem são considerados. Este processo é repetido, até que o vetor de diagnose, que representa o estado operacional das entidades do sistema, fique completo.

Na figura 5.2 é representado o funcionamento do algoritmo de diagnose, para um sistema de n entidades; as linhas contínuas representam os testes realizados por uma determinada entidade, enquanto a linha tracejada representa os resultados dos testes que esta entidade tem condições de considerar como corretos.

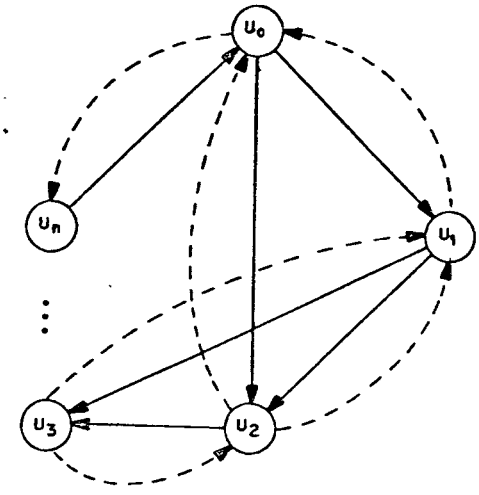


Figura 5.2. Fluxo de Informações no Algoritmo Kuhl & Reddy

c. Generalizações no Algoritmo de Kuhl e Reddy

Em seguida, são apresentadas algumas modificações introduzidas no algoritmo SELF (Kuhl & Reddy, 1980a), com o objetivo de generalizá-lo, associando-lhe outras restrições.

A primeira limitação está relacionada à ocorrência de faltas durante uma rodada de testes (aplicação de todos os testes representados pelo grafo de testes do sistema) (Kuhl & Reddy, 1980b). Esta limitação foi tratada considerando o tempo durante uma rodada de testes através da associação de pesos aos arcos do grafo de testes, correspondentes ao instante de tempo em que os testes são realizados entre as entidades. Embora exista uma semelhança com o estudo sobre faltas transientes

realizado em (Mallela & Masson, 1978), as faltas são consideradas permanentes.

Uma segunda limitação está relacionada à presença de faltas nos enlaces físicos da comunicação. Embora em (Kuhl & Reddy, 1980b) esta restrição já tivesse tido uma abordagem teórica, em (Kuhl & Reddy, 1981) é apresentado um algoritmo distribuído que possibilita a todas as entidades do sistema, diagnosticar corretamente e de maneira independente, o estado operacional das entidades e dos enlaces físicos da comunicação. Este algoritmo tem a característica de impor considerações menos restritas na forma pela qual a informação de diagnóstico se propaga pelo sistema; nos modelos anteriores, a informação de diagnóstico estava relacionada com o grau de conectividade t entre as entidades do grafo de testes do sistema.

Uma outra restrição que foi contornada, diz respeito ao reparo dinâmico, através da introdução de novos componentes nos serviços do sistema (Hosseini et alii, 1984).

Uma última restrição está relacionada ao fato que os modelos de diagnose apresentados até agora estão, voltados para sistemas constituídos somente de unidades testadoras (sistemas homogéneos). Em (Hosseini et alii, 1985) foi apresentado um algoritmo que considerava a diagnose de faltas em sistemas não homogéneos, ou seja, sistemas constituídos de entidades que

apresentam vários graus na capacidade de realizar testes (testadoras, semi-testadoras e não testadoras).

d. Comparação dos Algoritmos de Meyer & Masson e Kuhl & Reddy

Realizando-se uma comparação, em termos do espaço de memória e do tempo de processamento para os dois algoritmos anteriormente apresentados, pode-se chegar às seguintes conclusões.

O algoritmo de Kuhl & Reddy apresenta uma vantagem no sentido que trabalha com vetores de dimensão n, enquanto que o algoritmo de Meyer & Masson trabalha com matrizes de n x n, representando para este último, a necessidade de uma grande capacidade de memória, quando se trata de sistemas com um número razoável de entidades.

Em termos do tempo de processamento, já não existe uma comparação direta entre os dois algoritmos, pois, ambos adotam processos diferentes para diagnosticar as entidades com falta. No algoritmo de Meyer & Masson, depois de disseminados os resultados dos testes, existe a necessidade de cada entidade preencher o restante da matriz síndrome (n x n - n x t), para em seguida, baseando-se no conteúdo de cada matriz, proceder a votação. No caso de Kuhl & Reddy, o tempo com o processo de diagnose do sistema, está relacionado com o fato que, os

resultados dos testes têm que percorrer sequencialmente, de uma forma pré-determinada, todas as entidades do grafo de testes.

A diferença fundamental nos algoritmos de Meyer & Masson e Kuhl & Reddy, embora ambos estejam baseados no modelo PMC, reside no fato que, no primeiro, para que seja feita a diagnose do sistema, é necessário um processamento da síndrome, de forma independente, por cada entidade do sistema. Enquanto que no segundo, nâo existe um processamento da síndrome, mas sim, o processo de diagnose se restringe a verificar se a entidade da qual está recebendo o vetor de diagnose está livre de faltas, e repassar este vetor, para as entidades que a testam. Portanto, uma implementação mais otimizada para o algoritmo de Meyer & Masson, seria se adotar um "observador central" para o processamento da síndrome do sistema, ao invés de distribuir este processamento por todas as entidades.

Um ponto que é comun nos dois algoritmos, é o tratamento quando da localização de t entidades com falta; neste caso, mesmo que o processo de diagnose não tenha finalizado, o restante das entidades serão consideradas livre de faltas até à remoção das entidades com falta, e o reinício do processo de diagnose.

5.3.3. Modelo de Arcos Bipartidos

Em Holt e Smith (1985) foi considerado o problema da auto-diagnose em sistemas distribuídos, adotando-se um modelo mais geral e de maior utilidade na visualização dos conceitos de diagnose de faltas, embora ainda estivesse relacionado aos modelos teóricos mais tradicionais, do tipo PMC.

Enquanto que no modelo PMC, cada teste é executado e considerado completo para uma única entidade, no modelo de arcos bipartidos, é assumido que um teste pode ser completo para mais de uma entidade, e que mais de uma entidade pode invalidar um teste, conforme representado na figura 5.3. Nesta figura, as entidades f_1 e f_2 são responsáveis pela execução do teste t_1 nas entidades f_3 e f_4. A vantagem deste modelo com relação aos outros, está no fato que o subsistema de comunicação pode ser explicitamente representado, tanto no grafo do sistema como no grafo de testes.

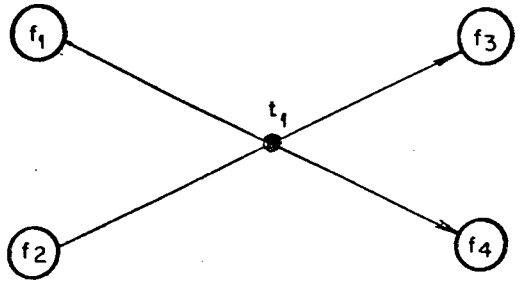


Figura 5.3. Modelo de Arcos Bipartidos

O modelo de arcos bipartidos foi utilizado no estudo de dois problemas: reconfiguração completa e reconfiguração degradada. Para a reconfiguração completa do sistema, este é periodicamente diagnosticado de forma tal as entidades com falta podem ser reparadas, mantendo assim o sistema operando em toda a sua capacidade. A técnica consiste em diagnosticar corretamente, pelo menos uma entidade com falta, e enviar esta informação a um controlador, para que a entidade seja trocada. Na reconfiguração degradada, é considerado que o sistema, inicialmente, tem um excesso na capacidade computacional, e conforme as entidades vão falhando, o sistema vai alocando a execução das funções pelas restantes entidades sem falta. Neste caso, a técnica consiste em identificar, através de analisadores livres de faltas, o máximo de entidades sem falta.

5.3.4. Modelo de Diagnose "Deslizante" (´Roving Diagnosis´)

Este modelo de diagnose de faltas não está baseado no modelo PMC, onde se parte do princípio que a operação normal do sistema é interrompida no momento de inicialização do processo de diagnose. Ao invés disto, o sistema é dividido em duas partes: uma envolvida com as atividades operacionais do sistema, e a outra envolvida com as atividades de diagnóstico. Esta última, por sua vez, é subdividida nas entidades realizando os testes e as sendo testadas.

Neste modelo, parte do sistema mais recentemente diagnosticada e considerada livre de faltas, começa testando outras partes do sistema. Esta estratégia está baseada num grafo deslizante - um subgrafo do grafo de testes, criando-se assim um sequenciamento na diagnose de todas as entidades do sistema. Com isto, se obtêm a aparência que um grafo, formado por um conjunto de entidades diagnosticando e outro sendo diagnosticado, "desliza" por todo o sistema, procurando entidades com falta.

A diagnose "deslizante" depende da diagnose das entidades que iniciam os testes. Os testes iniciais são executados por entidades 'hardcore', ou através de um processo de cooperação na diagnose inicial, estabelecida com a finalidade de se considerar uma unidade inicial livre de faltas; este segundo tipo de testes está ligado ao conceito de diagnose 'bootstrap'. A diagnose 'bootstrap' está baseada num grafo de diagnóstico acíclico, onde cada entidade é testada antes que seja utilizada para testar outras entidades; inicialmente são realizados os testes que não estão sujeitos a arcos que os invalidem, ou seja, entidades 'hardcore' (Kime,1985).

Os algoritmos até aqui descritos estão baseados no conceito da diagnose 'bootstrap', ou seja, as condições operacionais de uma entidade são inicialmente diagnosticadas, antes que esta seja utilizada para a comunicação ou origem de resultados de testes.

5.3.5. Modelos para Faltas Transientes

Todos os trabalhos, até agora abordados, sobre diagnose de sistemas, assumiram somente a existência de faltas permanentes. Considerações sobre faltas transientes são geralmente mais difíceis, pois, além de ser necessário um modelamento do comportamento destas faltas, também, são necessárias estratégias mais elaboradas nos testes para detectar esta classe de faltas. Embora na literatura de diagnose de faltas, o termo utilizado para referenciar circunstâncias revervíseis e de duração limitada seja de faltas intermitentes, o termo aqui adotado, para manter uma coerência com a nomenclatura anteriormente assumida, será de faltas transientes.

Em (Mallela & Masson, 1978) foram considerados os efeitos de faltas transientes em sistemas diagnosticáveis num passo. A existência de faltas permanentes e transientes num sistema, por exemplo, afetam os resultados dos testes, que por sua vez podem gerar uma diagnose incompleta das faltas existentes, pois, nem todas as entidades com falta no sistema podem ser localizadas. Com isto se apresenta um problema maior - se uma diagnose incompleta não pode ser evitada, uma diagnose incorreta de uma entidade livre de faltas tem que ser evitada no caso de faltas transientes. Neste sentido, um sistema é definido como t_i-faltas **diagnosticáveis** se não mais que t_i entidades apresentam faltas

transientes, tal que, uma entidade sem faltas nunca será classificada com faltas e a diagnose será no máximo incompleta, mas nunca incorreta.

Uma condição necessária para que um sistema seja t_i-faltas diagnosticáveis é que, cada entidade seja testada por pelo menos outras t_i+1 entidades. Num paralelo com sistemas t_p-faltas diagnosticáveis num passo (o subescrito p está relacionado com permanência), Mallela e Masson forneceram os limites para t_i como função de t_p. Em geral, o fato de que um sistema é t_p-faltas diagnosticáveis, não necessariamente implica que seja também t_i-faltas diagnosticáveis com $t_p=t_i$. É demonstrado que, em sistemas na qual não existam entidades se testando entre si t_i é menor que t_p, enquanto que t_i pode ser igual a t_p somente em sistemas na qual existem enlaces de testes bidirecionais. Um procedimento geral para o cálculo de t_i, para qualquer tipo de sistema, também é fornecido em (Mallela & Masson, 1978).

A diagnose de faltas híbridas (transientes e permanentes) num sistema distribuído, é tratada por Dahbura e Masson (1983a), através do método da 'greedy diagnosis' - diagnose voraz. Este método tem como objetivo diagnosticar uma entidade, independentemente dos estados potenciais mas não conclusivos, das outras entidades do sistema. Ou seja, um processamento diferenciado do processamento de síndromes pf-consistentes. Numa síndrome pf-consistente, ou síndrome consistente com faltas permanentes, nenhuma conclusão é obtida sobre as condições das

entidades do sistema, até que a síndrome seja relacionada a um conjunto de entidades com faltas permanentes. A diagnose de faltas, aplicado o processamento de 'greedy diagnosis', será no máximo incompleta mas nunca incorreta. A diagnose incompleta é devido, essencialmente, aos testes não aplicados e à característica de intermitência das faltas.

Dentro do modelo adotado, a síndrome do sistema é obtida a partir de um conjunto de testes que são realizados entre as entidades do sistema e representados por um dígrafo, conforme o modelo PMC. Para a situação de faltas híbridas, o processamento da síndrome na diagnose de uma entidade, consiste na identificação de um conjunto possível de entidades com falta, que não contenha a entidade sendo diagnosticada, e que esteja contido dentro de um conjunto maior formado a partir de todos os conjuntos possíveis de entidades com falta. Se nâo existe tal conjunto, então a entidade é considerada com falta. Para garantir uma diagnose correta, as entidades do sistema são consideradas, em princípio, livres de faltas, até que se prove o contrário.

5.3.6. Modelos por Comparação

Um novo modelo de diagnose de faltas para sistemas t-faltas diagnosticáveis, foi introduzido por Chwa e Hakimi (1981), para contornar algumas limitações encontradas no modelo PMC. Neste

último é assumido que cada teste é suficientemente completo, para que o resultado do testes seja inteiramente confiável; isto, quando a entidade executando o teste for livre de faltas. Entretanto, é de conhecimento prático que, a existência de um teste completo é questionável.

O modelo proposto por Chwa e Hakimi ao invés de utilizar os resultados dos testes executados entre as entidades do sistema, realiza a comparação dos resultados de tarefas idênticas executadas por pares de entidades. Estas comparações podem ser consideradas como testes completos pois, é difícil um par de entidades com falta, e vizinhas falharem de maneira idêntica.

A vantagem neste tipo de diagnose está no fato que, a melhor forma de se testar o comportamento dos componentes de um sistema, é exercitá-los pela execução das tarefas de que são responsáveis, ao invés de se utilizar de outros meios, tais como, testes de auto-diagnose. Estes últimos, além de suspender a execução das funções normais do sistema, também possibilitam a manifestação de faltas entre períodos de teste, e não oferecem meios de testar completamente todas as condições de funcionamento dos componentes do sistema.

No modelo por comparação de resultados de tarefas, é assumido existir um conjunto de tarefas $J = (J_1, J_2, \ldots)$ a serem executadas, e um conjunto de entidades idênticas $U = (u_1, u_2, \ldots, u_n)$ disponíveis para executá-las. Cada tarefa é

alocada a um par de entidades, e os dois resultados são comparados. Os resultados destas comparações são essenciais para a identificação das entidades com falta. O conjunto de pares de entidades na qual as tarefas são atribuídas, é referenciado como sendo a **estrutura da comparação**, e denotado por $C = (...,c_{ij},...)$, onde $c_{ij} = (u_i,u_j)$ é um par de entidades.

A estrutura da comparação corresponde a um grafo não direcionado G(U,C), onde cada $u_i \in U$ é um nó, e cada $c_{ij} \in C$ é um arco. Para cada par de entidades c_{ij}, quando for atribuída e completada uma tarefa, é formada a **síndrome da comparação**, associando a cada arco c_{ij} um peso $a_{ij} = 0(1)$, tal que o resultado da comparação indica um acordo (desacordo). Um arco que tenha associado um peso "1" ("0"), será referenciado como enlace_1 (enlace_0).

Tarefas são atribuídas e executadas por um par de entidades durante o que será referido como **período de tempo**. Durante estes períodos de tempo são executados no máximo $\lfloor n/2 \rfloor$ tarefas. Com isto, serão necessários k períodos de tempo para a execução de toda a estrutura da comparação. Os pares de entidades aos quais foram atribuídas tarefas durante o período i, i = 1, 2,...k, serão denotados por P(i). Depois do período k de tempo, toda a indicação da comparação C, estará completa P(1) U P(2)..,U P(k) = C.

Uma nova estratégia na diagnose de faltas foi introduzida por Dahbura e Masson (1983b), adaptando o princípio do método 'greedy diagnosis', ao processamento de síndromes da comparação. A cada periodo de tempo, depois de atualizada a síndrome da comparação, é aplicado o método da 'greedy diagnosis' para a localização de entidades com falta. Depois de reparar ou trocar as entidades com falta, é necessário se refazer a síndrome, para que o procedimento de diagnose prossiga no período de tempo seguinte. Esta estratégia além de remover a dependência das síndromes relacionadas a resultados de testes pf-consistentes, também elimina a dependência do limite máximo de faltas presentes t com relação ao número de conexões de testes existentes em cada entidade, isto usualmente encontrado em abordagens clássicas. Entretanto é necessário que se estipule um valor para t, e neste caso o número de faltas presentes no sistema nunca deve ultrapassá-lo.

Seguindo o mesmo princípio da comparação de resultados de tarefas Dahbura et alii (1985), adicionaram uma nova generalização: atribuiram probabilidades ao fato que o resultado de uma tarefa, executada por uma entidade com falta, pode ser tanto correto como incorreto, baseando-se que uma entidade mesmo com falta pode executar corretamente algumas das suas tarefas. A estratégia adotada para a identificação de faltas, é de continuamente selecionar e remover, de uma síndrome de comparação atualizada, uma entidade na qual exista a maior incidência de comparações com resultados em desacordo

(enlaces_1); isto depois de atualizada a síndrome da comparação. Este procedimento é repetido até que deixem de existir enlaces_1 na síndrome. Embora com probabilidade mínima de ocorrer, esta estratégia não garante, para todos os casos, a identificação completa das entidades com falta.

Na diagnose de faltas em sistemas distribuídos Dal Cin e Floriam (1985) eliminaram a necessidade de cada entidade ter que produzir de forma independente, a diagnose das restantes entidades do sistema. Para tanto, o sistema deve ser particionado em domínios, onde as entidades são responsáveis, somente, pela diagnose de algumas entidades vizinhas. As entidades são testadas através da comparação de resultados relativos a tarefas idênticas, executadas entre pares de entidades. Nas entidades executando a diagnose, consideradas livres de faltas (pelo menos houve acordo numa comparação de resultados com uma entidade vizinha), caso exista discrepância na comparação com outras entidades vizinhas, é considerado que estas últimas estão com falta.

A diagnose, se valendo do particionamento, assume um sentido de distribuída, por não ser necessário um conhecimento global da topologia e do estado do sistema; a informação normal do sistema é somente transmitida através das entidades vizinhas livres de faltas. O grau de diagnose t está associado ao conjunto de entidades com o qual, uma entidade executando a diagnose, mantém comparações de resultados. Para este algoritmo,

as faltas permanentes são tratadas como transientes; isto é possível se for estabelecido um número limite de execuções do procedimento de diagnose por uma entidade. Caso exceda um determinado limite e o estado continua indefinido, a entidade é considerada com falta permanente.

Também é assumido que comparações entre entidades com falta, sempre irão produzir discrepâncias; o quanto isto é verdadeiro, irá depender das rotinas de teste, dos procedimentos de comparação e da classe de faltas esperadas.

5.3.7. Estratégias Gerais da Diagnose Distribuída

Os modelos para a diagnose de faltas apresentados anteriormente, podem ser agrupados em dois tipos de estratégias, conforme comentado por Kuhl e Reddy (1981): os procedimentos por processamento de síndrome, e os procedimentos por auto-diagnóstico.

Esta divisão é normalmente realizada pois, estes primeiros estão associados ao processamento de informação que, embora possa ser executada de forma descentralizada, tem um melhor desempenho, se executada de forma centralizada. Ainda com relação ao processamento de síndromes, é necessário uma política de consenso, na qual um acordo e uma validação é implicitamente alcançada na diagnose de faltas, pelo conhecimento anterior que

todas as entidades livres de faltas terão um comportamento consistente. Dentro deste grupo estariam incluidos, por exemplo, os algoritmos de Meyer e Masson (1978) e de Dahbura e Masson (1983b), e no outro grupo, estariam os algoritmos de Kuhl e Reddy (1980) e de Dal Cin e Florian (1985).

Os procedimentos por auto-diagnóstico pareçem, a primeira vista, ser mais facilmente implementáveis, entretanto, alguns problemas podem ser encontrados: a necessidade de se definir a codificação da informação de diagnóstico a ser recebida por uma entidade, a forma de acesso que uma entidade pode ter a esta informação, e por último, quais as decisões a tomar baseando-se na informação de diagnóstico.

Embora existam outros modelos na literatura, que abordem a diagnose de faltas em sistemas distribuídos, os modelos e os algoritmos apresentados acima, são suficientes para o entendimento da linha de raciocíno adotada no algoritmo de diagnose de faltas a ser apresentado em seguida.

Finalizada a introdução à diagnose de faltas em sistemas distribuídos, em seguida será descrito o algoritmo de auto-diagnose proposto para o sistema distribuído sendo tratado. Nesta abordagem, além de ser descrito o algoritmo, será realizada a sua modelização por redes de Petri e posterior análise deste.

5.4. Algoritmo de Auto-Diagnose Distribuída

Os modelos e algoritmos de diagnose de faltas, abordados anteriormente, não são adequados para ambientes distribuídos com atributos tempo real, devido às generalizações normalmente adotadas, que tendem aumentar o grau de complexidade na diagnose de faltas. Dentro destas generalizações podemos citar: ocorrência de faltas dentro do processo de testes, grau de conectatividade do sistema, não homogenidade no sistema, requisitos de testes completos, faltas no meio físico, faltas transientes, etc.

Para o algoritmo de diagnose de faltas, a ser apresentado em seguida, será considerado um sistema distribuído constituído de várias subredes conectadas entre si através de entidades de interconexão ("pontes"). Nestas subredes é assumido a inexistência de redundâncias entre as entidades, e a interconexão direta entre estas através de um barramento. Cada uma das subredes é considerada um **domínio de diagnose**, onde é executado o algoritmo de diagnose distribuído. O confinamento da diagnose nas subredes em que um sistema pode ser particionado, reduz o número de mensagens manipuladas por cada entidade. Um outro motivo para a adoção de um sistema particionado, está relacionado ao baixo grau de conectatividade que um sistema global pode obter, devido à existência de caminhos únicos na

interconexão de subredes, limitando a diagnosticabilidade do sistema.

Nestes domínios de diagnose cada entidade é responsável, somente, pela diagnose das entidades vizinhas, ou seja, com aquelas com que mantém uma conexão física. Portanto, dentro do ambiente distribuído proposto, o algoritmo de diagnose tem por objetivo, a localização de entidades com falta, depois de detectado um erro no sistema.

Com isto, embora a comparação de resultados de tarefas idênticas alocadas a pares de entidades seja mais eficiente na localização de faltas, que a realização de testes completos entre entidades, deve-se tomar em conta que, em muitos casos torna-se difícil uma comparação de resultados de tarefas idênticas, pois cada entidade tem funções bem específicas e diferentes entre si, dentro do contexto global do sistema. A otimização dos recursos, num sistema tempo real sem redundâncias, impede a alocação de uma determinada tarefa numa entidade especializada ou envolvida com outras funções. Pela inexistência de funções comuns, a simples duplicação de tarefas visando atender as necessidades para a diagnose seria também um inconveniente, pois não atingiria os objetivos do modelo de comparação (Chwa & Hakimi, 1981), já descrito anteriormente.

Portanto, cada entidade é responsável pela execução de testes internos em seus componentes, vizando a realização da

auto-diagnose, e pela disseminação destes resultados entre entidades restantes do sistema.

O desempenho deste algoritmo está fundamentalmente baseado na disseminação confiável, através do subsistema de comunicação, dos resultados da auto-diagnose, realizada por cada entidade, entre as restantes entidades do domínio da diagnose. A disseminação confiável está baseada nas seguintes permissas (Drummond, 1987):

a. **Concordância** - todos as entidades corretas aceitam o mesmo valor;

b. **Validade** - se a entidade transmissora está correta, então todas as entidades corretas aceitam o valor disseminado pela transmissora.

Outras considerações, além das já assumidas, são:

a. o algoritmo terá que ser t-faltas diagnosticável num passo;

b. cada uma das entidades assumirá as funções de testadora, analizadora e controladora;

c. tanto as faltas permanentes como as transientes podem afetar uma entidade;

d. o susbsistema de comunicação não está sujeito a faltas;

e. uma entidade mesmo com falta, pode continuar a fornecer alguns dos seus serviços, e duas entidades com falta, têm uma probabilidade desprezível de produzirem serviços incorretos e idênticos.

Em seguida, é descrita a formalização, por grafos, a ser adotada na diagnose de faltas de um sistema distribuído, para posteriormente se descrever, informalmente, o algoritmo de auto-diagnose.

5.4.1. Modelagem do Sistema

Um sistema consistindo de n unidades, denotadas por um conjunto $U = (u_1, \ldots, u_n)$, é modelado por três grafos: o Grafo do Sistema (S), o Grafo de Testes (T), e o Grafo de Comparação (G), constituídos por um conjunto de arcos E e um conjunto de nós V:

a. Grafo do Sistema (S): é caracterizado por ser um grafo não orientado, representado por S=(E(S),V(S)).

V(S) = conjunto de nós correspondentes às entidades do sistema

$E(S) = ((u_i, u_j) \mid$ existe uma interconexão direta entre u_i e $u_j)$

b. Grafo de Testes (T): caracterizado por ser um grafo não orientado, representado por T=(E(T),V(T)).

$V(T) = V(S)$

$E(T) = ((u_i, u_j) \mid$ existe uma troca dos resultados da auto-diagnose entre u_i e $u_j)$

A cada arco $(u_i, u_j) \in E(T)$ é associado um peso $b_{ij}=(0,1)$, correspondente ao resultado da auto-diagnose realizada pela entidade u_j.

$b_{ij} = 0$ u_j considerada livre de faltas;

$b_{ij} = 1$ u_j considerada com falta.

Portanto, o conjunto de pesos associados (b_{ij}), aos resultados da auto-diagnose nas entidades, é o vetor diagnose.

c. Grafo de Comparação (G): é caracterizado por ser um grafo não orientado, representado por G=(V(G),E(G)).

V(G) = V(S)

E(G) = $((u_i,u_j) \mid (u_i,u_j)$ existe uma comparação entre os vetores de diagnose de u_i e $u_j)$

C = (..., C_{ij},...) onde $C_{ij} = (u_i,u_j)$ representa o par de entidades que executam a comparação de seus vetores de diagnose.

O resultado da comparação dos vetores de diagnose associados a um par de entidades $(u_i,u_j) \in E(S)$ é $a_{ij} = (0,1)$, onde:

$a_{ij} = 0$ resultados da comparação concordam;

$a_{ij} = 1$ resultados da comparação discordam.

5.4.2. Descrição do Algoritmo

Neste item, será descrito, informalmente, um algoritmo de diagnose de faltas para ambientes distribuídos, que é uma derivação do apresentado em (Dal Cin & Florian, 1985).

Cada entidade participante do domínio da diagnose deve montar o seu vetor diagnose. Este vetor representa a visão que cada entidade mantém do estado operacional das restantes entidades do seu domínio.

Com isto, deixa de existir a necessidade de uma entidade executar t outras comparações com entidades vizinhas, pois esta já mantém no vetor diagnose o estado operacional das restantes entidades, e que precisa ser confirmado somente por pelo outra entidade. Este último passo é também executado por Dal Cin e Florian (1985).

A diagnose de entidades com falta no sistema distribuído, é realizada temporariamente ou quando detectado um erro. Para o primeiro caso, existe a necessidade de suspender o serviço das entidades (e do sistema), para que possam ser realizados os testes relativos à auto-diagnose das entidades do sistema.

Para o caso em que seja detectado um erro, é suficiente se atualizar o vetor diagnose, ou seja, existe uma nova disseminação dos estados operacionais por parte das entidades do domínio de diagnose. Com isto, uma entidade não necessita de suspender seus serviços para a realização dos testes internos. Quando o vetor diagnose estiver atualizado, seja "definido" ou "indefinido", o anterior pode ser substituído, possibilitando ao processo de diagnose de faltas ser executado concorrentemente com o serviço normal da entidade.

Com relação à presença de faltas transientes, inicialmente, todas as faltas que ocorrem no sistema são consideradas transientes; se depois de um determinado período a condição de falta ainda persistir, então é considerada como falta permanente.

Em seguida, serão descritos os vários passos do algoritmo de diagnose:

Primeiro Passo - Execução de Testes Internos

- Cada entidade realiza um conjunto de testes internos para a auto-diagnose. A auto-diagnose de uma entidade é acionada por uma das três maneiras: periodicamente, quando requisitada pela entidade supervisora, ou após o retorno do estado com falta.

Segundo Passo - Disseminação dos Resultados

- Para o estado operacional (EO) de cada entidade é realizada uma disseminação confiável entre as entidades restantes do domínio de diagnose.

- Cada entidade que receba uma mensagem contendo o EO de uma entidade vizinha, dissemina o seu EO através de uma mensagem, caso já não o tenha feito.

Terceiro Passo - Montagem do Vetor Diagnose

- O vetor diagnose (VD) corresponde ao estado operacional de todas as entidades do domínio de diagnose. Este vetor é montado em cada entidade, a partir dos resultados da auto-diagnose das outras entidades e dos mecanismos de comunicação que sinalizam excessões na transmissão das mensagens contendo os EO;

- A montagem do vetor de diagnose está associada a uma temporização. Uma vez ultrapassado o 'time-out' estabelecido para a montagem do VD, o restante das entidades são consideradas com falta;

- Depois de montado, o VD de cada entidade é disseminado pelas restantes entidades do domínio de diagnose.

Quarto Passo - Comparação dos Resultados

- Depois de enviado o VD, cada entidade fica aguardando a recepção dos VDs enviados pelas outras entidades;

- A cada VD recebido das entidades vizinhas, é realizada uma comparação com o VD da própria entidade. Com isto, são definidos dois estados para o VD: estado definido e estado indefinido;

- O processo de comparação de um VD de uma entidade com os das entidades vizinhas está associado a um 'time-out'. Expirado este 'time-out', e a entidade esteja livre de faltas, esta terá conhecimento do EO das restantes entidades do sistema, através do conteúdo de VD. Este 'time-out', elimina a necessidade de se esperar a recepção de (n-1) mensagens relativas aos VDs das restantes entidades.

Quinto Passo - Estado Definido

- Um VD de uma entidade assume o estado definido quando existe uma concordância com pelo menos um VD de uma entidade vizinha. Com isto, a entidade realizando a diagnose é considerada livre de faltas;

- Se existir uma concordância entre todos os VDs, então todas as entidades são consideradas livres de faltas;

- Caso não exista uma concordância entre os VDs de uma ou mais entidades, então será enviada a estas, uma mensagem de parada, sinalizando que estão com falta.

Sexto Passo - Estado Indefinido

- Um VD de uma entidade assume o estado indefinido quando não existir uma concordância com os VDs das entidades vizinhas. Um VD no estado indefinido fica aguardando um dos eventos para finalizar a diagnose:

  a. caso chegue uma mensagem de parada de uma entidade vizinha, então a entidade é considerada com falta, e é induzida a suspender os seus serviços;

  b. caso estoure o 'time-out' associado à diagnose do sistema, e a entidade não tenha recebido nenhum mensagem de parada, então esta se considera livre de faltas e finaliza a diagnose, sem enviar nenhuma mensagem de parada às entidades vizinhas.

É interessante ressaltar que, neste algoritmo de diagnose de faltas, o resultado da diagnose obtido por cada entidade não tem a necessidade de ser disseminado pelo restante do sistema. Quando finalizada a diagnose, ao final do 'time-out', somente serão notificadas as entidades que não obtiveram uma concordância dos seus VDs com outras entidades do sistema.

Uma entidade saindo do estado com falta, para voltar a participar dos serviços do sistema, terá que executar a sua

auto-diagnose e disseminar o seu EO pelas restantes entidades do sistema, dando início a um novo processo de diagnose do sistema.

Com relação à distinção entre a ocorrência de faltas transientes e permanentes nas entidades do sistema, conforme já comentado, todas as faltas serão inicialmente consideradas como sendo faltas transientes. Cada vez que for localizada uma entidade com falta transiente no sistema, é enviada uma mensagem de parada para forçar que os seus serviços sejam interrompidos. Se depois de um determinado número de tentativas em acessá-la, a entidade ainda permanece com falta, então esta, será considerada com falta permanente. Caso seja localizada uma falta permanente numa entidade, é necessário que sejam efetuadas as devidas ações de reparo no sistema.

De maneira diferente da abordagem realizada por Dal Cin e Florian (1985), o algoritmo aqui proposto não trata o problema das faltas transientes/permanentes, dentro do próprio algoritmo, e sim, no número de vezes em que a diagnose é executada. Com isto, deixa de existir a necessidade, durante a execução do algoritmo, que uma entidade realizando diagnose fique repetindo continuamente a mesma tarefa, para determinar se a entidade vizinha está com falta transiente ou permanente, conforme apresentado em Dal Cin e Florian (1985).

Em seguida, o algoritmo de diagnose será modelado por redes de Petri, o quê permitirá a verificação da sua exatidão na localização de entidades com falta, num ambiente distribuído.

5.4.3. Validação/Verificação do Algoritmo Proposto

Para garantir uma implementação correta do algoritmo de diagnose de faltas para ambientes distribuídos, descrito anteriormente, é necessário que seja verificada a sua conformidade com os requisitos funcionais previstos. Esta verificação pode ser realizada, ou através de uma simulação do algoritmo que permite avaliar o seu desempenho dinâmico, ou através da análise de um modelo que represente as operações deste algoritmo. Neste trabalho, foi adotada a segunda forma. Foi utilizado a rede de Petri Predicado-Ação (ou etiquetada) e com Temporização, cuja a descrição é dada no Anexo I, como modelo para representar o algoritmo.

A validação do modelo foi realizada através da análise, verificando-se as propriedades gerais da rede (limitada, viva, binária e reinicializável), com auxílio de dois analisadores automáticos, o ARP - Analisador de Redes de Petri, desenvolvido no Laboratório de Controle e Microinformática do Departamento de Engenharia Elétrica da UFSC, e o PAREDE - Programa de Análise de Redes de Petri com Temporização, desenvolvido no Departamento de Engenharia Elétrica da PUC/RJ. A conformidade do algoritmo com

os requisitos, foi realizada a partir da verificação da adequação entre o modelo do algoritmo e o modelo do serviço a ser fornecido.

5.4.3.1. Modelagem

A modelagem do algoritmo de diagnose está dividida em duas partes. Na primeira, foi realizada a modelagem dos estados de uma entidade do sistema, e na segunda parte, foi feita a modelagem do algoritmo de diagnose de faltas, conforme especificado anteriormente. Estas duas modelagens são apresentadas em seguida, juntamente com os respectivos grafos de marcações.

a. Modelo da Entidade

Na modelagem dos estados da entidade na diagnose de faltas, foi assumido a existência de quatro estados: sem falta, com falta, realizando diagnose do sistema, e realizando auto-diagnose da entidade. Uma entidade só entra no estado de auto-diagnose, quando este for requisitado pela estação supervisora, temporariamente conforme programado, ou toda a vez que sair do estado com falta.

A rede de Petri e o grafo de marcações correspondente, estão representados, respectivamente, nas figuras 5.4 e 5.5. O

algoritmo de diagnose de faltas no sistema está representado pelas as transições t3, t4 e t5, e pelo o lugar DSF.

Lugares

SF - entidade sem falta;

CF - entidade com falta;

DSI - diagnose do sistema iniciada;

DSF - diagnose do sistema finalizada;

AD - entidade em auto-diagnose;

Transições

t0 - (!MEO/?MPD), início da diagnose do sistema solicitada por mensagem (MPD), e envio de mensagem contendo o EO (MEO);

t1 - (/?MAD), início da auto-diagnose da entidade, solicitada pela entidade supervisora através de uma mensagem (MAD);

t2 - ultrapassado o 'time-out' (TOO)associado à auto-diagnose da entidade;

t3 - diagnose do sistema;

t4 - fim da diagnose do sistema, entidade sem falta;

t5 - fim da diagnose do sistema, entidade com falta;

t6 - (/?ACREPA), ação de reparo executada;

t7 - fim da auto-diagnose da entidade, com falta;

t8 - fim da auto-diagnose da entidade, sem falta.

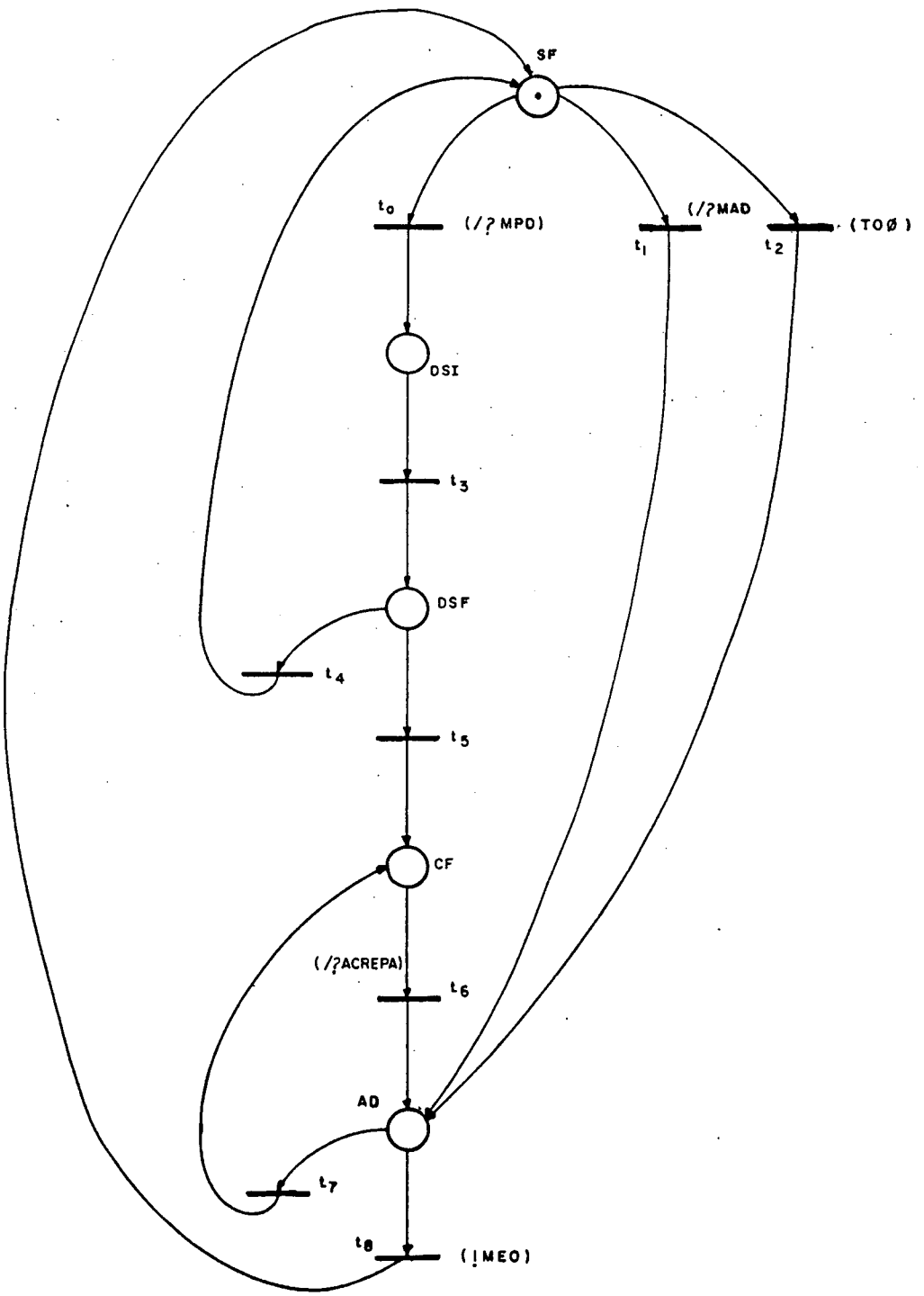


Figura 5.4. Modelo da Entidade - Rede de Petri

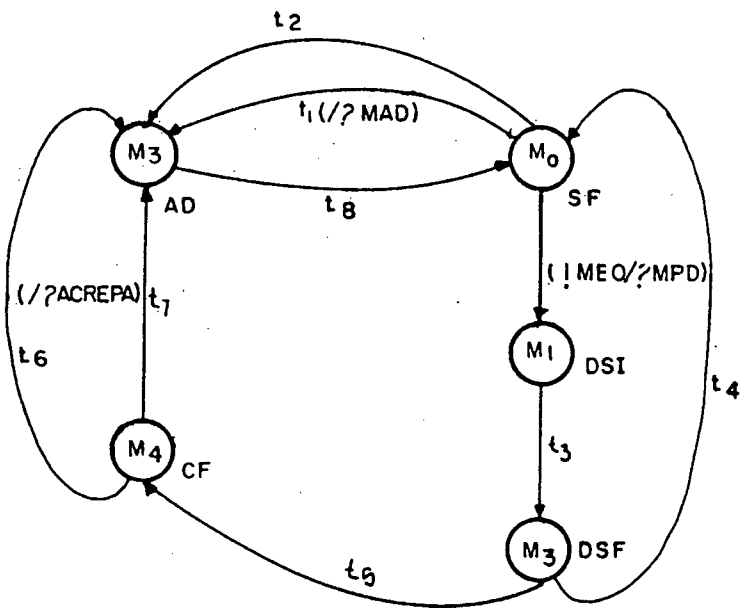


Figura 5.5. Modelo da Entidade - Grafo de Marcações

b. Modelagem do Algoritmo de Diagnose de Faltas

Foi modelado o algoritmo de diagnose de faltas para um sistema constituído de apenas três entidades (a entidade de suporte do algoritmo será chamada de i, as duas outras de j e k). A generalização para n entidades, consiste em repetir a parte da estrutura da rede correspondente à comparação entre VDs da entidade suporte e de uma entidade vizinha, de (n-1) vezes.

Conforme representado pelo modelo, existem duas possibilidades de terminar o algoritmo da entidade i: estado fim diagnose com falta localizada e estado fim diagnose sem falta localizada.

O primeiro estado é possível quando da chegada de uma mensagem de parada (MSM), ao mesmo tempo em que não existe concordância entre o VD da entidade i e os VDs das outras entidades (j, k).

No que concerne à obtenção de um estado fim de diagnose sem falta, 3 casos são possíveis. No caso de discordância de VDs entre entidades, expirar o 'time-out' e não tenha sido recebida uma mensagem de parada (MSM), então a entidade i se considera livre de faltas. Os outros dois casos para a entidade i se considerar livre de faltas, é quando existe pelo menos uma concordância entre o VD da entidade i e um VD de uma entidade

vizinha. Para as entidades com os quais existe discordância, é enviada uma mensagem de parada. A rede de Petri e grafo de marcação estão representados, respectivamente, nas figuras 5.6 e 5.7.

Lugares

EeIND - VD da entidade "e" com estado indefinido (p/ e=i,j,k);

EeDEF - VD da entidade "e" com estado definido (p/ e=i,j,k);

IMVD - iniciada a montagem do VD;

FMVD - finalizada a montagem do VD;

AGVDe - aguarda a recepção dos VDs das outras entidades (p/ e=j,k);

COMPe - os VDs da entidade i e "e" em comparação (p/ e=j,k);

TOle - 'time-out' relativo ao processo de diagnose expirado (p/ e=j,k);

FeDEF - aguarda final de diagnose com VDe no estado definido (p/ e=j,k);

FeIND - aguarda final de diagnose com VDe no estado indefinido (p/ e=j,k);

INDeTl - entidade "e" com estado indefindo, após expirado 'time-out' (p/ e=j,k);

FeDIAG - final da diagnose das entidades "e" (p/ e=j,k);

Transições

t0 - início da diagnose do sistema;

t2 - (/?MVDC), montagem do VD completo, recebida a mensagem do EO de todas as entidades;

t3 - (/?MVDI), montagem do VD incompleto com 'time-out' TO0 associado, não foi recebida a mensagem com o EO de todas as entidades;

t4 - (!MVDI), envia mensagem contendo o VD montado pela entidade (MVDI);

t7,t10 - fim da comparação, VDs das entidades i e "e" diferentes (p/ e=j,k);

t8,t9 - fim da comparação, VDs das entidades i e "e" iguais (p/ e=j,k);

t11,t12 - VD da entidade "e" passa para o estado definido;

t14 - fim de 'time-out', VD da entidade i com estado indefinido;

t15 - VD da entidade i definido, entidade i sem falta;

t17 - (/?MSMe), final da diagnose, aguarda recepção de mensagem de parada (MSM), entidade i com falta (p/ e=j,k);

t19,t24 - (!MSMeL), envia mensagem de parada (MSMeL) para as entidades "e" (p/ e=j,k);

t20,t23 - final da diagnose, entidade "e" sem falta (p/ e=j,k);

t21,t22 - não chegou mensagem de parada, entidade i sem falta;

t26 - final da diagnose, entidade i sem falta;

t29 - 'time-out' TO1 associado ao processo de diagnose;

t30 - (/?MVDe), recebe mensagens contendo o VD das entidades "e" (MVDe), (p/ e=j,k);

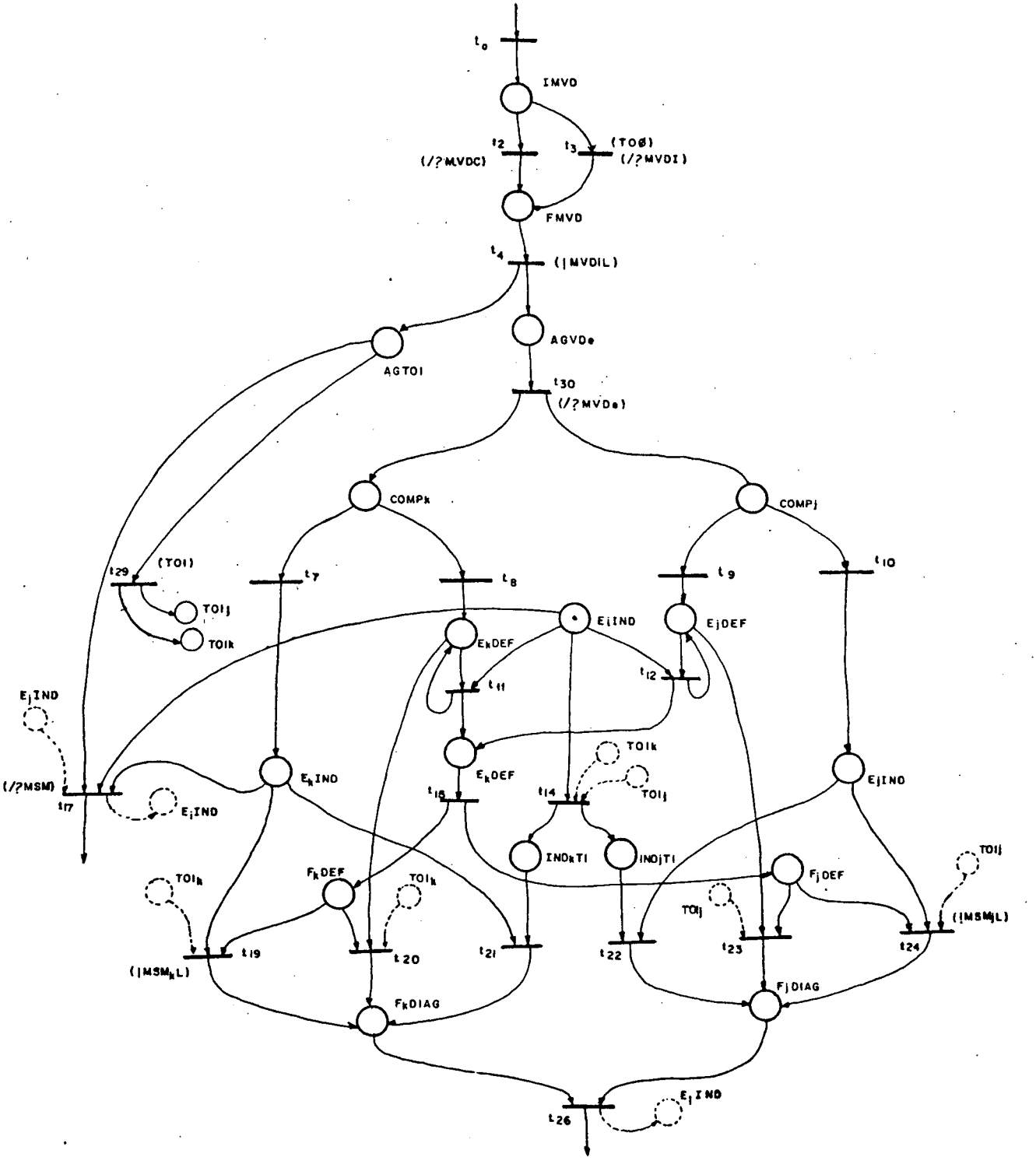


Figura 5.6. Modelo do Algoritmo de Diagnose - Rede de Petri

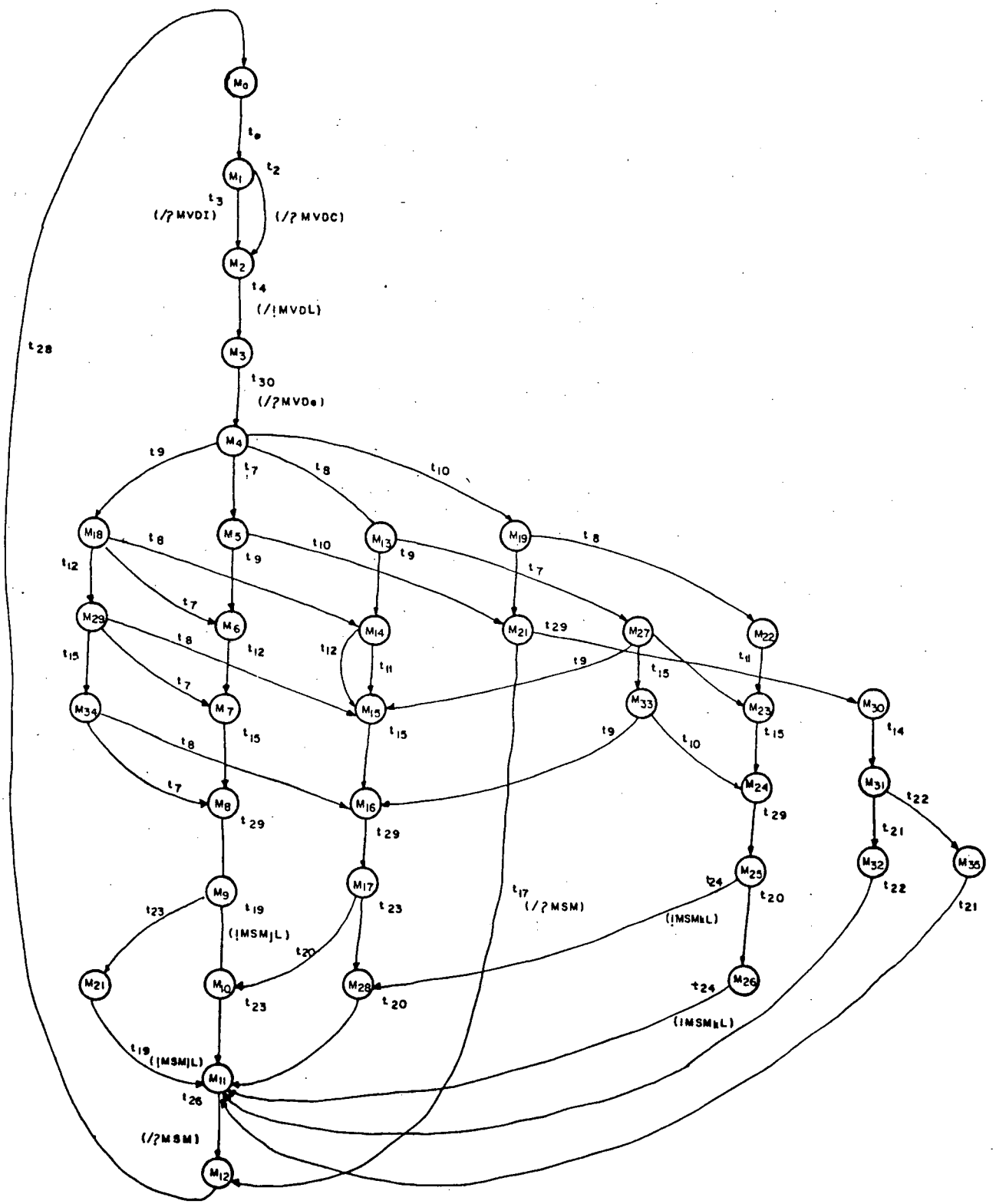


Figura 5.7. Modelo do Algoritmo de Diagnose-
Grafo de Marcações

5.4.3.2. Validação/Verificação do Algoritmo

i. Validação do Modelo

As redes representadas nas figuras 5.4 e 5.5 possuem as propriedades de limitação e vivacidade; estas propriedades foram encontradas através do uso das ferramentas de análise já citadas. Para isto, estas redes tiveram que ser sensivelmente alteradas para a realização desta análise. A rede completa, composição destas duas, representando a entidade i com o seu algoritmo de localização de faltas num sistema de 3 entidades, possue também as boas propriedades de uma rede.

ii. Verificação da Conformidade do Algoritmo aos Requisitos

A verificação da conformidade do algoritmo aos requisitos, foi realizada através de uma comparação entre o grafo de marcações reduzido por projeção, obtido a partir da modelagem completa do algoritmo, e o grafo de marçações obtido a partir da modelagem do serviço desejado a ser fornecido pelo algoritmo.

A rede de Petri representando o serviço desejado e grafo de marcações correspondente estão representados, respectivamente, nas figuras 5.8 e 5.9.

Lugares

AGMVD - entidade aguarda montagem do VD;

REDIAG - executando a diagnose do sistema;

F_DIAG - final de diagnose;

Transições

t0 - (/?MVDI,?MVDC), recebe o EO das outras entidades;

t1 - (!MVDiL/?MVDe), entidade i envia VD através de mensagem (MVDiL), e recebe VDs das outras entidades (MVDe);

t2 - entidade i sem falta;

t4 - (!MSMeL), entidade i sem falta, envia mensagem de parada (MSMeL) para entidades "e" com falta (p/ e=j,k);

t5 - (/?MSM), entidade i com falta, recebe mensagem de parada (MSM);

t7 - diagnose finalizada.

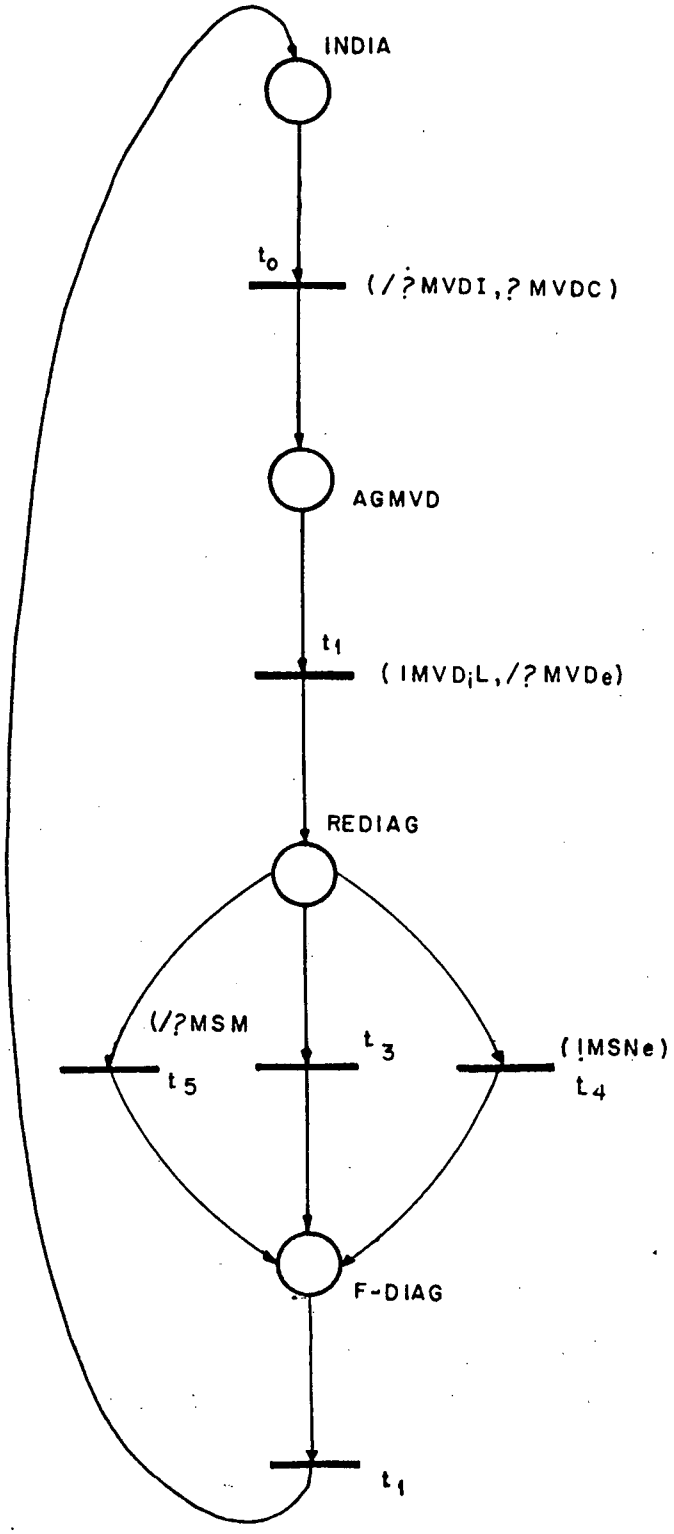


Figura 5.8. Modelo do Serviço do Algoritmo - Rede de Petri

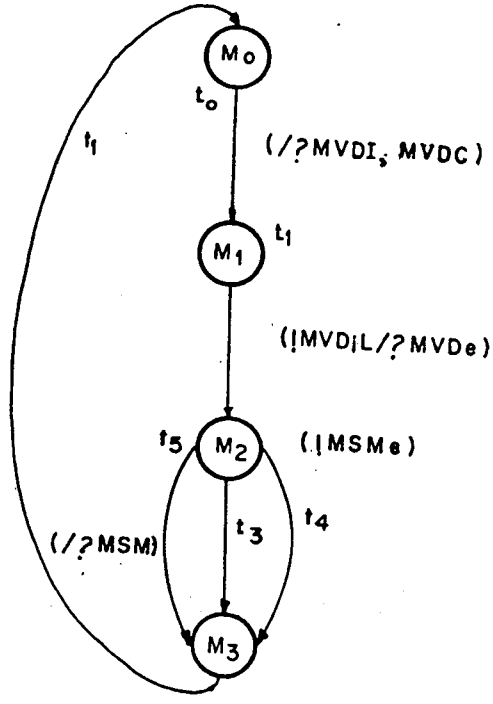


Figura 5.9. Modelo do Serviço do Algoritmo - Grafo de Marcações

O método de verificação através da projeção (Peterson, 1981) e (Ayache et alii, 1985), tem os seguintes passos:

1 - separação das transições do grafo em
    * transições internas (invisíveis), chamadas de λ-transição.
    * transições visíveis (com etiquetas);

2 - eliminação das λ-transições e determinação dos estados equivalentes;

3 - comparação do modelo do serviço desejado com o modelo global do algoritmo reduzido por projeção; esta comparação pode ser direta nos automatos obtidos ou sobre as linguagens regulares geradas pelos automatos.

O grafo de marcações do modelo global do algoritmo uma vez reduzido pelo método da projeção se apresenta na forma da figura 5.10. A composição em termos dos automatos da figura 5.9 e 5.10, nos permite verificar que o algoritmo de diagnose de faltas está conforme o serviço previsto.

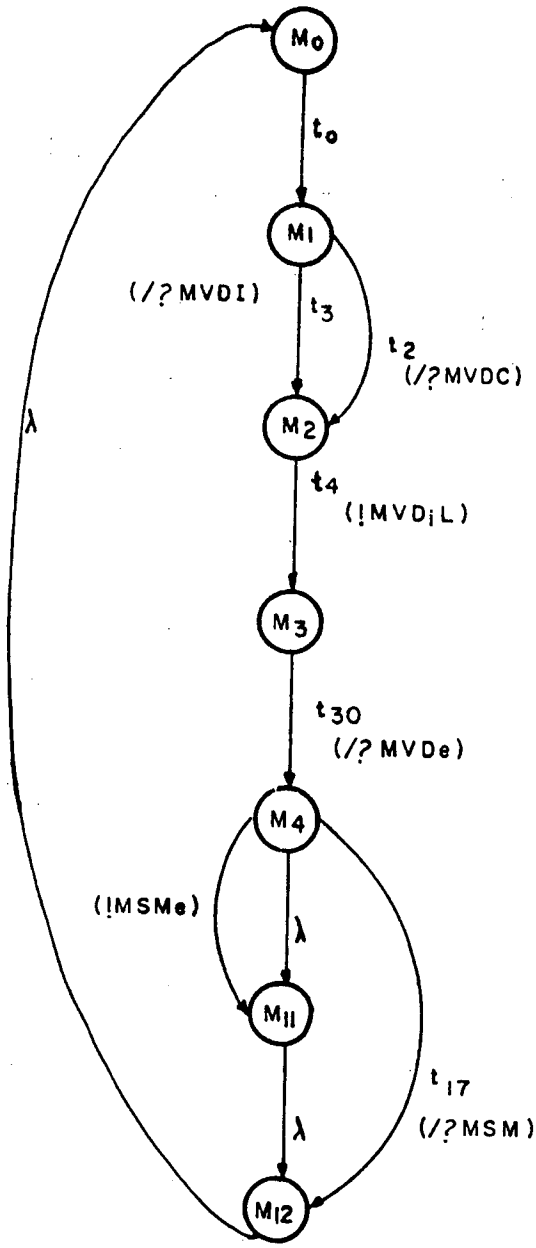


Figura 5.10. Grafo de Marcações Reduzido do Modelo do Algoritmo de Diagnose

iii. Simulação

O estudo do comportamento do algoritmo, através de simulação do seu modelo global (representado na figura 5.6) foi também realizado. Os resultados foram os esperados conforme o mostram os seguintes estudos de caso.

Caso 1: VDi ≠ VDj
VDi ≠ VDk

Sem a chegada de nenhuma mensagem de parada, a entidade i se considera livre de faltas, mas não envia nenhuma mensagem de parada para as entidades vizinhas.

Recebendo uma mensagem de parada, a entidade i se considera como estando com falta.

Caso 2: VDi = VDj
VDi ≠ VDk

As entidade i e j se consideram livres de faltas; é enviada uma mensagem de parada para a entidade k, considerada como estando com falta.

Caso 3: VDi ≠ VDj
VDi = VDk

As entidade i e k se consideram livres de faltas; é enviada uma mensagem de parada para a entidade j, considerada como estando com falta.

Caso 4: VDi = VDj
VDi = VDk

Todas as entidades do sistema são consideradas livres de faltas.

5.5. Implementação do Algoritmo

Um algoritmo de diagnose de faltas, para o tipo de sistema distribuído sendo tratado, terá que ter, fundamentalmente, um requisito: interferir ao mínimo nos serviços do sistema.

Para a implementação do algoritmo de auto-diagnose distribuída no Sistema Digital, este último foi particionado em subredes (domínios de diagnose), com o objetivo de simplificar a modelagem, e consequentemente, otimizar a sua implementação. Caso fosse realizada a diagnose do sistema global, a diagnosticabilidade do sistema ficaria seriamente limitada,

devido à distribuição das entidades do sistema e da existência de caminhos únicos ('Gateways').

Nas figuras 5.11 e 5.12, são representados, respectivamente, o Grafo do Sistema e o Grafo de Testes da modelização da diagnose de faltas do Sistema Digital. O algoritmo de diagnose faltas será implementado nas três subredes ('domínios de diagnose') em que o Sistema Digital foi particionado, sendo que cada um dos 'gateways', devido à função assumida, irá pertencer a duas subredes do Grafo de Testes.

Para a disseminação do estado operacional de cada entidade, é proposto o uso da técnica da disseminação seletiva, uma característica já incorporada no subsistema de comunicação do Sistma Digital. Nesta técnica, depois da disseminação, para cada entidade que não tenha enviado um reconhecimento de recepção, uma nova mensagem é enviada através de um enlace ponto-a-ponto, contendo o estado operacional da entidade emissora.

Uma simplificação adotada na modelagem do sistema, foi a desconsideração das faltas no subsistema de comunicação. Com isto, para o modelo de diagnose adotado, é assumido que estão sujeitas a faltas, somente as unidades do sistema. As faltas ao longo dos enlaces de comunicação podem destruir o contexto de uma mensagem, sem contudo alterá-lo de maneira maliciosa.

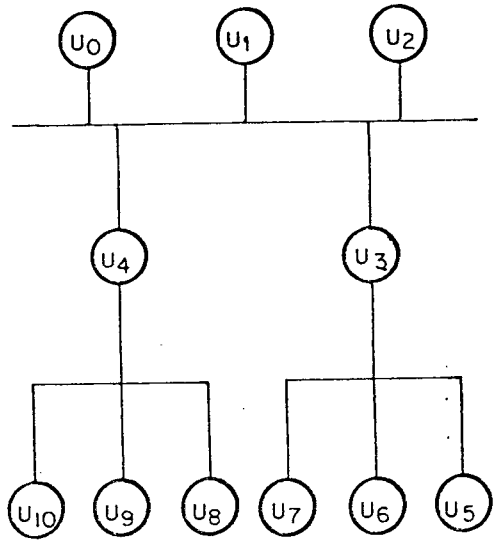


Figura 5.11. Grafo do Sistema

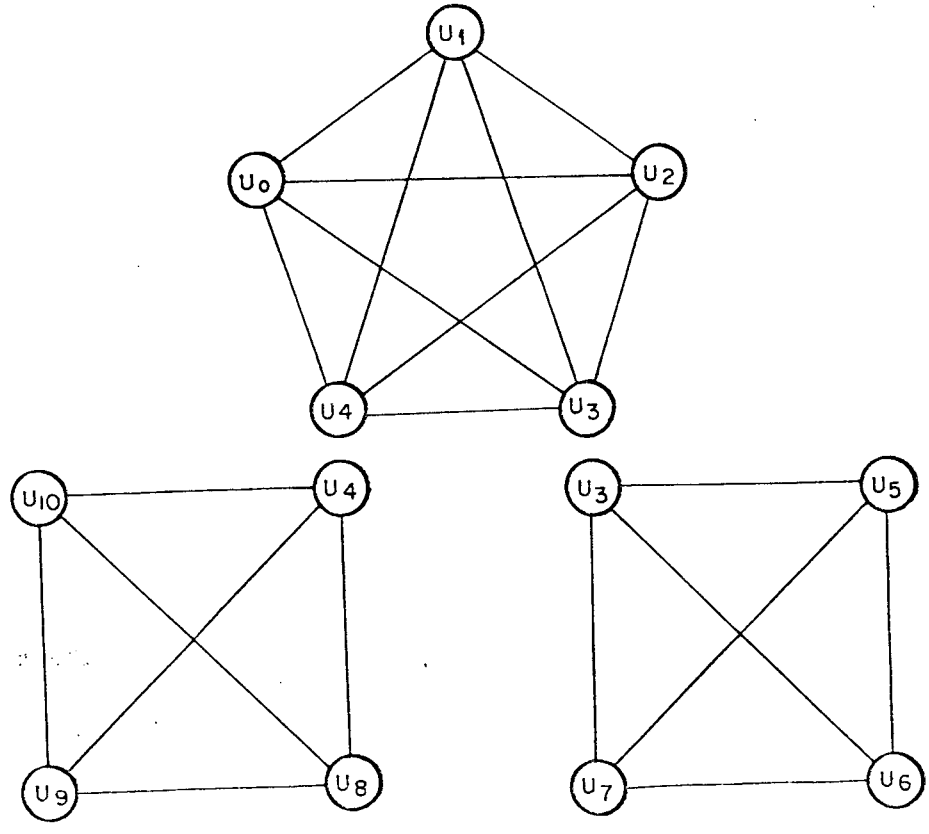


Figura 5.12. Grafo de Testes

Como o sistema distribuído, nesta etapa de protótipo, é um sistema sem redundâncias, não estão previstas ações de reparo imediato nas entidades consideradas com falta. Nestas condições, existe a necessidade de se considerar a diagnose de faltas como sendo t-faltas diagnosticáveis num passo, ou seja, todas as entidades com falta no sistema terão que ser identificadas sem ter a necessidade de trocá-las, isto dentro de limites, previamente estabelecidos.

As ações de reparo podem se constituir de desligamento de todo o sistema, de desligamento remoto de entidades (logicamente ou fisicamente), ou mesmo por auto desligamento. Devido à inexistência de redundâncias no sistema, dependendo da entidade com falta, será necessário se realizar o desligamento de todo o sistema, para evitar que ações indevidas sejam executadas no processo (US/SE). O desligamento lógico remoto, é mais adequado para os casos em que as entidades consideradas livres de faltas, identificando entidades como estando com falta permanente, executem ações de reparo. E por último, a ação de auto desligamento é necessária quando uma entidade, depois de realizar uma auto-diagnose, se considera com faltas.

5.5. Conclusão

O tratamento de faltas em sistemas distribuídos, tem se tornado um requisito importante, devido à crescente utilização destes sistemas em diferentes áreas de aplicação. Técnicas clássicas na tolerância a faltas, do tipo redundância massiva com votação majoritária, são incapazes de oferecer níveis altos de confiabilidade, sem envolverem custos altos de implementação e/ou baixo desempenho. Como consequência, alguns modelos teóricos foram propostos, como fundamento a estratégias mais eficazes de diagnose de faltas.

Um destes modelos, é o modelo clássico de diagnose de faltas introduzido por Preparata, Metze e Chien (1967), baseado em considerações rígidas sobre a natureza das faltas e dos testes, e com isto, acarretando alguns problemas fundamentais na sua aplicação: o primeiro, os resultados dos testes dependem, essencialmente, da existência de um conjunto de testes (provavelmente não realistas) que as entidades do sistema realizam entre si, o segundo, a análise dos resultados dos testes (síndrome) só é possível de ser executada quando estes são consistentes com uma situação de faltas permanentes, e por último, é assumido que os testes são executados de forma alternada com o processamento normal da entidade, possibilitando a manifestação de faltas entre períodos de teste. Estas considerações impedem que o modelo PMC, se torne uma estratégica prática para a diagnose de faltas num sistema.

Qualquer tentativa para contornar as limitações acima mencionadas, sempre se recai em modelos e algoritmos com graus de complexidade maiores, e não muito adequados para sistemas distribuídos com atributos de tempo real. Com o objetivo de contorná-las, surgiram outros modelos e algoritmos. Um destes modelos, é a comparação de resultados, relativos à execução de tarefas idênticas em duas entidades do sistema. Um outro modelo, uma generalização deste último, é a votação entre várias entidades dos resultados de tarefas executadas por estas.

Para o algoritmo de diagnose de faltas, proposto neste capítulo, verificou-se que tanto o algoritmo como a entidade que o contém, representados por modelos de rede de Petri, estão conforme os requisitos inicialmente previstos, e realizaram corretamente as funções para as quais foram concebidas.

Uma representação do sistema global de n entidades cada uma com algoritmo de diagnose próprio, interligado por um meio, também representado por rede de Petri, permitiria a verificação do comportamento do algoritmo proposto.

CAPÍTULO 6

CONCLUSÃO

O estudo apresentado neste trabalho aborda a segurança de funcionamento em sistemas distribuídos, tratando mais especificamete dos aspectos ligados a auto-diagnose. Num sistema informático, é notório que, quanto maiores são as exigências de altos graus de confiabilidade e/ou disponibilidade, mais os recursos e as funções do sistema são descentralizados. A descentralização torna difícil o gerenciamento de recursos e também, determina a complexidade na detecção de eventos indesejáveis e na localização de faltas nestes sistemas. Estes aspectos foram pontos de partida para as reflexões que conduziram este estudo.

Neste sentido, acreditamos que as principais contribuições deste trabalho são essencialmente:

- estudo e proposição de uma terminologia, em português, na área de segurança de funcionamento;

- estruturação e descrição das atividades de gerenciamento para um sistema distribuído;

- especificação de um algoritmo descentralizado, para a diagnose de faltas em ambientes distribuídos.

Com relação à terminologia na área de segurança de funcionamento, basta referenciar-se a (SCTF, 1987) e (Paula Jr et alii, 1987), para se constatar a existência de terminologias conflitantes, assumidas entre os diversos autores dos trabalhos. O que demonstra a necessidade de se adotar uma terminologia consistente, para o português, a exemplo do que vem sendo realizado a nível de organismos internacionais (Laprie, 1987). Neste trabalho, foi feita uma tentativa de se estabelecer uma terminologia e uma conceituação básica, na área de segurança de funcionamento.

Conforme especificações iniciais, a Função de Observação, tinha como objetivo acompanhar o comportamento do Sistema Digital, fundamentalmente, em termos do desempenho e do estado operacional. Devido à diversidade das funções a serem executadas, e à complexidade do sistema, existiu a necessidade de se estruturar todas as atividades relacionadas à Função de Observação em termos de um modelo de Gerenciamento de Sistemas Distribuídos. Além das funções básicas, previamente especificadas, foram incluídas outras funções necessárias para o gerenciamento de sistemas distribuídos.

Por último, devido a inexistência de redundâncias no sistema, que possam recuperar os serviços de uma entidade com falta, é necessário que, tão logo seja detectado um erro no sistema, a entidade com falta seja imediatamente localizada e isolada. Para tanto, foi realizado um estudo sobre diagnose de faltas em sistemas distribuídos, e proposto um algoritmo que tem

como objetivo a identificação das entidades com falta, no Sistema Digital. Este algoritmo pode ser, igualmente, implementado num sistema distribuído com redundâncias.

As perspectivas de continuidade deste trabalho, podem seguir duas linhas distintas: a primeira relacionada às funcões de gerenciamento para o sistema distribuído, onde seria necessário o detalhamento das funções de gerenciamento, já apresentadas, em termos do Sistema Digital, e a segunda, no estudo da diagnose de faltas em sistemas distribuídos, relacionada à implementação do algoritmo de diagnose de faltas.

Em contrapartida às vantagens múltiplas que determinam a utilização crescente dos sistemas informáticos distribuídos, uma série de dúvidas e temores são inspirados na confiabilidade destes sistemas. Esperamos ter mostrado, nesta dissertação, que a segurança de funcionamento pode tirar proveito das características inerentes dos sistemas informáticos distribuídos, contribuindo deste forma para a continuidade dos serviços.

BIBLIGRAFIA

Anderson, W. & Lee, P. (1981). "Fault-Tolerance: Principles and Practice". Prentice Hall Int´l. 1981.

Avizienis, A. (1976). "Fault Tolerant Systems". IEEE Trans. on Computers, VOL C-25 No 12. DEC 1976. pg 1304-1312.

Avizienis, A. (1978). "Fault-Tolerance: The Survival Attribute of Digital Systems". Proceedings of the IEEE, VOL 66 No 10. OUT 1978. pg 1109-1125.

Avizienis, A. & Kelly, J. (1984). "Fault-Tolerance by Design Diversity and Experiments". IEEE Computer, VOL 17 No 8. AGO 1984. pg 67-80.

Avizienis, A. & Laprie, J. C. (1986). "Dependable Computing: From Concepts to Design Diversity". Proceedings of the IEEE, Vol 74 No 5. MAI 1986. pg 629-638.

Ayache, J.M.; Courtiat, J.P. & Diaz, M. (1982). "Self-Checking Software in Distributed Systems". IEEE 3rd Int. Conf. on Distributed Computer Systems. 1982. pg 163-170.

Ayache, J.M. et alii (1985). "Utilisation des Reseaus de Petri pour le Modelisation et la Validation de Protocoles". TSI-Technique et Science Informatique Vol.4 No 1. pg 51-71. 1985.

Bologna, S. & Leveson, N. G. (1986). "Reliability and Safety in Real-Time Systems". IEEE Trans. on Software Engineering, Vol SE-12 No 8. SET 1986. pg 877-878.

Bruso, Sally. (1985). "A Failure Detection and Notification Protocol for Distributed Computing Systems". Proc. 5th Int. Conf. Distributed Computing Systems. pg 116-123. Denver, Colorado. MAI 1985.

Campbell, R. & Randell, B. (1986). "Error Recovery in Asynchronous Systems". IEEE Trans. on Software Engineering, Vol SE-12(8). AGO 1986. pg 811-826.

CEPEL (1987). "Especificação Funcional do Software do Centro de Controle dos Protótipos de Usinas e Subestações". CEPEL. 1987.

Chwa, K.-Y. & Hakimi, S. (1981). "Schemes for Fault-Tolerant Computing: A Comparision of Modularly Redundant and t-Diagnosable Systems". Information and Control Vol 49. pg 212-238. 1981.

CIGRÉ (1983). "Interim Report on Computer Based Protection and Digital Techniques in Substations". WG 34.02, CIGRÉ SC 34. Tokyo. NOV 1983.

Dahbura, A. & Masson, G. (1983a). "Greedy Diagnosis of Hybrid Fault Situations". IEEE Trans. on Computers Vol C-32 No 8. pg 777-782. AGO 1983.

Dahbura, A. & Masson, G. (1983b). "Greedy Diagnosis as the Basis of an Intermittent-Fault/Transient-Upset Tolerant System Design". IEEE Trans. on Computers Vol C-32 No 10. pg 953-9562. OUT 1983.

Dahbura, A. et alii. (1985). "The Comparision Approach to Multiprocessor Fault Diagnosis". Proc. 15th Symp. Fault Tolerant Computing. pg 260-265. 1985.

Dal Cin, M. & Florian, F.-H. (1985). "Analysis of a Fault-Tolerant Distributed Diagnosis Algorithm". Proc. 15th Symp. Fault Tolerant Computing. pg 260-265. 1985.

De, B. B. & Krakan, H. B. (1981). "Fault-Tolerance in a Microprocessor Digital Switching System". IEEE Trans. on Reliability, Vol R-30 No 3. AGO 1981. pg 246-252.

Diaz, M. (1982). "Modeling and Analysis of Communication and Cooperation Protocols Using Petri Net Based Models". Protocol Specification, Testing and Verification. Ed. C. Sunshine. North-Holland Publishing Company. 1982. pg 465-510.

Drummond, R. (1987). "Sistemas Distribuídos". Introdução a Tolerância a Falhas. II SCTF. Campinas. Agosto 1987. pg 147-171.

ELETROSUL (1986). "Projeto DIPROS: Requisitos Funcionais". DES/ELETROSUL. 1986.

Emmerson, R. & McGowan, M. (1984). "Fault Tolerance Achieved in VLSI". IEEE Micro, Vol 4 No 4. DEC 1984. pg 34-43.

Fraga, J. & Lemos, R. (1987). "Conceituação e Terminologia em Sistemas Informáticos Tolerantes a Faltas e Intrusões". II SCTF. Campinas. Agosto 1987. pg 177-188.

Friedman, A. & Simoncini, L. (1980). "System-Level Fault Diagnosis". Computer. pg 47-53. MAR 1980.

GM (1988). "Network Manegement Requirements Specifications". MAP_Chapter 11. General Motors Manufacturing Automation Protocol. Julho 1987.

Holt, G. & Smith, J. (1985). "Self-Diagnosis in Distributed Systems". IEEE Trans. on Computers Vol C-34 No 1. pg 19-32. JAN 1985.

Hosseini, S.; Kuhl, J. & Reddy, M. (1984). "A Diagnosis Algorithm for Distributed Computing Systems with Dynamic Failure and Repair". IEEE Trans. on Computers Vol C-33 No 3. Março 1984. pg 223-233.

Hosseini, S.; Kuhl, J. & Reddy, M. (1985). "On Self Fault-Diagnosis of the Distributed Systems". Proc. 15th Int. Symp. Faul-Tolerant Computing. pg 30-35. 1985.

ISO, (1981). "Data Processing - Open Systems Interconection - Basic Refence Model". ISO/TC97/SC16. Computer Networks 5. 1981. pg 81-118.

Jacobson, D. et alii (1987). "A Master/Slave Monitor Meaurement Technique for an Operating Ethernet Network". IEEE Network Vol 1 No 3. Julho 1987. pg 40-48.

Johnson, B. W. (1984). "Fault-Tolerant Microprocessor Based Systems". IEEE Micro. DEC 1984. pg 6-21.

Jones, Anita K. (1978). "The Object Model: A Conceptual Tool for Structuring Software". Lectures Notes in Computer Science, No 60. Springer-Verlag. New York. 1978.

Kime, C. (1970). "An Analysis Model for Digital System Diagnosis". IEEE Trans. on Computers Vol C-19 No 11. pg 1063-1073. NOV 1970.

Kuhl, J. & Reddy, S. (1980a). "Distributed Fault-Tolerance for Large Multi-Processor Systems". Proc. 7 Int. Symp. Computer Architectures. pg 23-30. 1980.

Kuhl, J. & Reddy, S. (1980b). "Some Extensions to Theory of Systems Level Fault Diagnosis". Proc. 10 Symp. Fault-Tolerant Computing. pg 291-296. Kyoto, Japan. Out. 1980.

Kuhl, J. & Reddy, S. (1981). "Fault-Diagnosis in Fully Distributed Systems". Proc. 11th Symp. Fault-Tolerant Computing. pg 100-105. 1981.

Kuhl, J. & Reddy, S. M. (1986). "Fault-Tolerance Considerations in Large Multiple-Processor Systems". IEEE Computer. MAR 1986. pg 56-67.

Langsford, A.; Naemura, K. & Speth, R. (1983). "OSI Management and Job Transfer Services". Proc. of the IEEE Vol 71 No 12. Dezembro 1983.

Laprie, J. C. (1984). "Dependable Computing and Fault-Tolerance: Concepts and Terminology". LAAS Research Report 84.035. Toulouse. JUN 1984.

Laprie, J. C. (1985). "Dependable Computing and Fault-Tolerance: Concepts and Terminology". Proc. 15th FTCS. Ann Arbour, Michigan. JUN 1985. pg 2-11.

Laprie, J. C. (1987). "Dependable Computing and Fault-Tolerance: Concepts and Terminology". II SCTF. Campinas. Agosto 1987. pg 189-200.

Lemos, R. (1987). "Estudo Introdutório da Subestação de Palhoça". Nota Técnica LCMI 87/7. JAN 1987.

Levenson, N. (1985). "Software Safety". Resilient Computing Systems. ed. T. Anderson. Collins Professional and Technical Books. London. 1985.

Liebowitz, B. H. & Carson, J. H. (1985). "Multiple Processor Systems for Real-Time Applications". Prentice-Hall Inc. 1985.

Mallela, S. & Masson, G. (1978). "Diagnosable Systems for Intermittent Faults". IEEE Trans. on Computers Vol C-27 No 6. pg 560-566. JUN 1978.

Melliar-Smith, P. M. & Randell, B. (1977). "Software Reliability: The Role of Programmed Exception Handling". ACM SIGPLAN Notices, Vol 12 No 3. MAR 1977. pg 95-100.

Meyer, G. & Masson, M. (1978). "An Efficient Fault Diagnosis Algorithm for Symmetric Multiple Processor Architectures". IEEE Trans. on Computers, Vol C-27 No 11. pg 1059-1063. NOV 1978.

Miller, D. H. (1979). "A Taxonomy of Fault-Tolerance Techniques". Technical Note No 175 Computer Systems Lab, Stanford University. JUN 1979.

Molva, R.; Diaz, M. & Ayache, J. M. (1985). "Observer: A Run-Time Checking Tool for Local Area Network" Proc. 5th Int. Workshop on Protocol Specification, Verification and Testing, Toulouse França. JUN 1985. pg 11.1-11.12.

Morgan, D. E. & Taylor, D. J. (1977). "A Survey of Methods of Achieving Reliable Software". IEEE Computer, Vol 10 No 2. FEV 1977. pg 44-53.

Morris, G. & Carter, R. (1986). "MAP Management and Administration". Control Engineering. Outobro 1986. pg 23-25.

Namjoo, M. (1982). "CERBERUS-16: An Architecture for a General Purpose Watchdog Processor" CRC Technical Report No 8-19, Stanford University. DEC 1982.

Paula Jr, A. et alii (1987). "Introdução a Tolerância a Falhas". II SCTF. Campinas. Agosto 1987. pg 147-171.

Peterson, J.L. (1981). "Petri Net Theory and the Modeling of Systems". Prentice-Hall. 1981.

Pilaud, E. (1982). "Conception et Validation de Systemes Informatiques aHaute Surete de Functionnement". Tese de Doutorado, L.A.A.S. NOV 1982.

Preparata, F.; Metze, G. & Chien, R. (1967). "On the Connection Assignment Problem of Diagnosable Systems". IEEE Trans. on Eletronic Computers, Vol EC-16 No 6. DEC 1967.

Randell, B. (1975). "Systems Structure for Software Fault-Tolerance". IEEE Trans. on Software Engineerig, Vol SE-1 No 2. JUN 1975. pg 220-232.

Randell, B.; Lee, P. A. & Treleaven, P. C. (1978). "Reliability Issues in Computing Systems Design". Computing Surveys, VOL 10 No 2. JUN 1978. pg 123-165.

Randell, B.; Lee, P. A. & Treleaven, P. C. (1978). "Reliable Computing Systems". Lectures Notes in Computer Science, No 60. Springer-Verlag, New York. 1978.

Randell, B. (1984). "Fault-Tolerant and Systems Structuring". 1984.

Rennels, D. A. (1978). "Architecture of Fault-Tolerant Spacecraft Computers". Proceedings of the IEEE, Vol 66 No 10. OUT 1978. pg 1255-1268.

Rennels, D. A. (1984). "Fault-Tolerant Computing - Concepts and Examples". IEEE Trans. on Computers, Vol C-30 No 12. DEC 1984. pg 1116-1129.

Roux, J.-L. Juanole, G. (1987). "Functional and Perfomance Analysis Using Extended Time Petri Nets". Int. Workshop on Petri Nets and Perfomance Models. AGO 1987. pg 14-23.

Russel, J. & Kime, C. (1975). "System Fault Diagnosis: Closure and Diagnosability with Repair". IEEE Trans. on Computers Vol C-24 No 11. pg 1078-1088. NOV 1978.

SCTF, (1987). II Simpósio em Sistemas de Computadores Tolerantes a Falhas. Campinas. Agosto 1987.

Siewiorek, D. (1984) . "Architecture of Fault Tolerant Computers". IEEE Computer, VOL 17 No 8. AGO 1984. pp 9-18.

Sloman, M. (1984). "Management of Local Area Networks". COST 11 bis Local Area Network Project, Part 2. Ed. M. Sloman. Imperial College, Dept. of Computing. 1984.

Sloman, M. (1987). "Distributed Systems Management". Imperial College Research Report DOC 87/6. Ed. M. Sloman. Abril 1987.

Smith, J. (1978). "Universal System Diagnosis Algorithms". IEEE Trans. on Computers Vol C-28 No 5. pg 374-378. MAI 1978.

Tschammer, V. & Klessmann, H. (1985). "Network Management in Distributed Control Systems". IFAC Distributed Computer Control Systems 1985. pg 115-122. 1985.

Tripathi, A. R. & Wang, P.-S. (1983). "An Object-Oriented Design Model for Reliable Distributed Systems". 1983.

Wakid, S.; Brusil, P. & LaBarre, L. (1987). "Coming to OSI: Network Resource Manegement and Global Reachability". Data Communications. Dezembro 1987. pg 137-150.

York, G. (1983). "Software Voting in NMR Computing Structure". Msc Report-DEE Carnigie-Mellon University. FEV 1983.

ANEXO I

Redes de Petri Predicado-Ação com Temporização

Neste Anexo, é fornecida uma visão introdutória sobre redes de Petri com Temporização e redes de Petri Predicado-Ação (Diaz, 1982), (Roux & Juanole, 1987) e (Ayache et alii, 1985), utilizadas na modelização do algoritmo de diagnose de faltas para ambientes distribuídos.

1. Redes de Petri com Temporização

As redes de Petri (RdP) com Temporização são uma derivação das RdP ordinárias, onde para cada transição são associados dois tempos especificando o intervalo de disparo (tmin,tmax). O primeiro valor (tmin) está relacionado ao tempo mínimo que uma transição, depois de sensibilizada, tem que esperar antes que possa ser disparada. E o segundo valor (tmax), está relacionado ao tempo máximo na qual uma transição seja disparada após a sua sensibilização.

Formalmente uma RdP com Temporização é definida com uma sextupla:

(P, T, I, O, Mo, Io)

onde,

P = (p1,p2,...,pm), é um conjunto finito de lugares;

T = (t1,t2,...,tn), é um conjunto finito de transições;

I: T x P -> N, é a função do lugar precedente;

O: P x T -> N, é a função do lugar seguinte;

Mo: P -> N, é a marcação inicial;

Io: T -> (Q+ U 0) x (Q+ U ∞)

ti -> Ii = (ai,bi) com $0 \leqslant ai \leqslant bi$

Vi, $1 \leqslant i \leqslant n$

(**N** é o conjunto dos números Naturais)

(Q+ é o conjunto dos números Racionais não negativos)

2. Redes de Petri Predicado-Ação

As redes Predicado-Ação permitem representar o estado interno do sistema e a sus interação com o mundo externo.

Uma RdP Predicado-Ação é constituída de:

a) uma RdP;

b) uma expressão associada a cada transição, do tipo

se Pt(x) faça x' <- Ft(x)

onde, Pt é o predicado, Ft é a ação, ambas sobre variáveis do sistema.

Para existir um tiro numa transição, é necessário que esta esteja sensibilizada e o predicado correspondente seja verdadeiro. Quando existe o tiro, a ação associada é executada de forma indivisível.

Para a representação da mensagem, foi adotada uma notação específica:

(/?m) - representa que uma mensagem foi recebida;

(!m) - representa que uma mensagem foi transmitida;