Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica
Departamento de Sistemas de Energia Elétrica

Escalonamento em Tempo Real das Funções Avançadas de Análise de Rede Elétrica de um Moderno Centro de Controle

Eduardo Nicola Ferraz Zagari

Orientador: Prof. Dr. Ariovaldo Verandio Garcia

Co-Orientador: Prof. Dr. Alcir José Monticelli

Tese apresentada à Faculdade de Engenharia Elétrica, UNICAMP, como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.

Campinas, fevereiro de 1996.

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Z13e

Zagari, Eduardo Nicola Ferraz
Escalonamento em tempo real das funções avançadas de análise de rede elétrica de um moderno centro de controle / Eduardo Nicola Ferraz Zagari.--Campinas, SP: [s.n.], 1996.

Orientadores: Ariovaldo Verandio Garcia; Alcir José Monticelli.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica.

1. Redes elétricas - Análise. 2. Sistemas de energia elétrica - Controle. 3. Programação de tempo real. 4. Sistemas de tempo real. I. Garcia, Ariovaldo Verandio. II. Monticelli, Alcir José. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica. III. Título.

"Mesmo que eu tivesse o dom da profecia
e conhecesse todos os mistérios e toda a ciência,
mesmo que eu tivesse toda a fé,
a ponto de transportar montanhas,
se eu não tiver caridade, eu não sou nada."
I Coríntios 13, 2

Dedico este trabalho:

a Deus, sem o qual nada seria possível ...

ao meu pai, Mario Roberto, à minha mãe, Ceres, e aos meus irmãos Raquel, Cristina, Luísa e Kennedy, família maravilhosa e singular. Eles sempre me confortam, me apoiam e me incentivam, nunca se queixam e, além de tudo isto, ainda escrevem minhas dedicatórias...

Agradecimentos

- Ao Prof. Dr. Ariovaldo Verandio Garcia e ao Prof. Dr. Alcir José Monticelli, pela amizade, pelos conhecimentos transmitidos e pela inestimável orientação a mim dispensada ao longo destes anos,
- ao amigo João Paulo de S. Rocha, pelo apoio e companheirismo durante todo o desenvolvimento do trabalho,
- aos amigos Luciano Macedo Freire e B. Alencar de Melo Jr. pelas conversas sobre funções de análise de redes elétricas e sobre escalonadores em sistemas de tempo real, que tanto me ajudaram,
- aos amigos de Juiz de Fora, que aqui encontrei, pelas oportunidades que me proporcionaram de me sentir mais próximo de casa, e aos amigos que aqui fiz, em especial à Hermínia M. Dias Bruno, pela acolhida fraterna e generosa,
- aos velhos e novos amigos do Laboratório de Sistemas de Energia Elétrica (LSEE/DSEE), pela convivência tão proveitosa e enriquecedora que sempre mantivemos,
- ao Departamento de Sistemas de Energia Elétrica - UNICAMP, nas pessoas de seus professores, pela colaboração e pelas excelentes condições de pesquisa que proporcionam aos estudantes da pós-graduação,
- à Companhia Paulista de Força e Luz, pelo apoio,
- ao CNPq, pelo apoio financeiro, e
- àqueles a quem dedico este trabalho,

o meu muito obrigado!

Lista de Abreviaturas e Símbolos

$\lfloor x \rfloor$ - Maior inteiro menor ou igual a x

$\lceil x \rceil$ - Menor inteiro maior ou igual a x

$\{T_1/T_2\}$ - Equivale a $(T_2/T_1) - \lfloor T_2/T_1 \rfloor$, isto é, a parte fracionária de T_2/T_1

CAG - Controle Automático da Geração

DMSS - Algoritmo *Deadline Monotonic Sporadic Server*

DS - Algoritmo *Deferrable Server*

ECA - Erro de Controle de Área

E/S - Entrada/Saída

EMS - *Energy Management System* (Sistema de Gerenciamento de Sistemas de Potência)

FAREs - Funções de Análise de Rede Elétrica

FIFOs - Fluxos de dados unidirecionais *First In, First Out*

FPO - Fluxo de Potência Ótimo

FPORS - Fluxo de Potência Ótimo com Restrições de Segurança

IHM - Interface Homem-Máquina

IP - *Internet Protocol* (Protocolo Padrão Internet)

IPC - *Interprocess Communication*

LANs - *Local Area Networks* (Rede Local de Computadores)

MFLOPS - Milhões de instruções em pontos flutuantes por segundo

MIPS - Milhões de instruções por segundo

PE - Algoritmo *Priority Exchange*

PP - Algoritmo Próximo Prazo

RISC - *Reduced Instruction Set Computer* (Computador com um conjunto reduzido de instruções)

SCADA - *Supervisory Control And Data Acquisition* (Controle Supervisório e Aquisição de Dados)

SE (*SS*) - Algoritmo Servidor Esporádico (*Sporadic Server*)

STR - Sistema de Tempo Real

TCP - *Transmission Control Protocol* (Protocolo de Controle de Transmissão)

TM - Algoritmo Taxa Monotônica

TRC - Tubo de Raios Catódicos

U - Fator de Utilização

UCP - Unidade Central de Processamento

UTR - Unidade Terminal Remota

Resumo

Este trabalho apresenta dois modelos de implementação de escalonadores para as funções de análise de rede em tempo real em um Centro de Controle. Os escalonadores propostos gerenciam funções como o configurador de rede, o estimador de estado e o fluxo de potência *on-line*, tanto de forma periódica como por solicitação do operador. O primeiro modelo é um escalonador estático, baseado no algoritmo Taxa Monotônica com Servidor Esporádico, e o segundo, baseado no algoritmo Próximo Prazo, é um escalonador com atribuição dinâmica de prioridades. As características das funções avançadas de análise de rede foram analisadas quanto aos seus aspectos de restrições temporais e independência entre instâncias de varreduras do sistema de aquisição de dados distintas.

As implementações foram feitas para arquiteturas computacionais monoprocessadas, sob o sistema operacional UNIX. Foram realizados testes em um sistema elétrico da região de Campinas através de três ambientes distintos.

Palavras Chaves: *Funções de Análise de Redes Elétricas, Escalonamento em Tempo Real, Modernos Centros de Controle.*

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Ao longo dos últimos anos, tem se realizado grandes esforços para o desenvolvimento, implantação e consolidação de funções de análise de rede nos Centros de Controle. Esta tem sido a realidade tanto das empresas concessionárias de energia brasileiras, como também de empresas em todo o mundo.

As empresas que estão iniciando a implantação ou modernização de seus Centros de Controle têm à disposição uma nova concepção para o projeto destes Centros, baseada nos seguintes conceitos:

- integração dos processos produtivos e administrativos da empresa (rede corporativa),
- sistemas de informação integrados,
- arquitetura aberta e processamento distribuído em todos os níveis e funções.

A incorporação de novas funções de análise de rede elétrica, considerando aspectos de regime permanente e dinâmico, nos Centros de Controle, visa tornar estes Centros capazes de contornar os problemas surgidos na operação em tempo real, em razão da maior utilização do sistema elétrico.

Dentre as vantagens trazidas com a modernização dos Centros de Controle, destacam-se a melhoria da qualidade tanto dos processos referentes à operação em tempo real, como daqueles referentes à pré e à pós-operação, resultando assim em uma melhoria na qualidade do fornecimento de energia. A implantação das funções avançadas de análise de rede visa ainda o aumento da confiabilidade e da segurança operativa do sistema. Além disso, a partir do subsistema de análise de rede torna-se viável uma redução de investimentos e a operação econômica do sistema elétrico de potência.

Para a efetiva implantação das funções avançadas de um Centro de Controle em tempo real, é necessário o gerenciamento de suas execuções, o

que é realizado por um escalonador. O escalonador é responsável pela ativação periódica de todas as funções de análise de rede, pela alocação do processador àquela de maior prioridade em estado de pronto e pela suspensão de alguma função, sempre que necessário, para garantir o cumprimento do prazo das mesmas.

Além disso, muitas vezes o operador do sistema elétrico pode desejar realizar estudos de fluxo de carga em situações operativas diferentes ou analisar contingências, preparando-se para a operação do sistema. Assim, o escalonador deve gerenciar também a execução de funções de estudo *off-line*, que são solicitadas pelo operador do sistema esporadicamente.

O problema geral de escalonamento dos aplicativos de rede de um Centro de Controle é um problema complexo. Isto porque os Centros de Controle, devido às distintas demandas computacionais dos subsistemas que os compõem, apresentam arquiteturas computacionais muito heterogêneas. Uma alternativa de solução é a concepção de um ambiente computacional dedicado à execução do subsistema de análise de rede. As funções de análise de rede caracterizam-se por apresentarem processamento em ponto flutuante intensivo. Isto sugere que suas execuções devam ser realizadas em computadores que sejam eficientes no processamento em ponto flutuante, como o são as estações de trabalho com processadores *RISC* (*Reduced Instruction Set Computer*).

O objeto deste trabalho é a pesquisa e desenvolvimento de um escalonador para as funções de análise de rede em tempo real, utilizando recursos do sistema operacional UNIX[1], amplamente utilizado em estações de trabalho. O desenvolvimento do escalonador foi realizado para um ambiente monoprocessador dedicado à execução das funções em tempo real. Desta forma, é possível se conceber um subsistema de análise de rede em um Centro de Controle com apenas a integração do ambiente proposto à arquitetura computacional já empregada no Centro. A Figura 1.1 ilustra a arquitetura computacional sugerida, na qual se baseou todo o desenvolvimento deste trabalho. As informações adquiridas do sistema elétrico através das Unidades Terminais Remotas (UTRs) são transmitidas ao Centro de Controle e exteriorizadas aos operadores através da Interface Homem-Máquina (IHM). A informação pertinente é enviada, através de um *gateway*, à estação de análise de rede, que executa os aplicativos de rede, devolvendo seus resultados para que possam também ser exteriorizados ao operador do sistema.

Com este intuito, no Capítulo 2, após um breve histórico, é apresentado um modelo conceitual para os Centros de Controle e são comentados seus componentes. Cada função que compõe o subsistema de análise de rede é discutida e tem suas características analisadas.

No capítulo seguinte, são introduzidos os conceitos básicos de sistemas de tempo real. Após serem discutidas as características dos escalonadores em sistemas de tempo real, são apresentados alguns algoritmos de escalonamen-

[1] UNIX é uma marca registrada da AT&T.

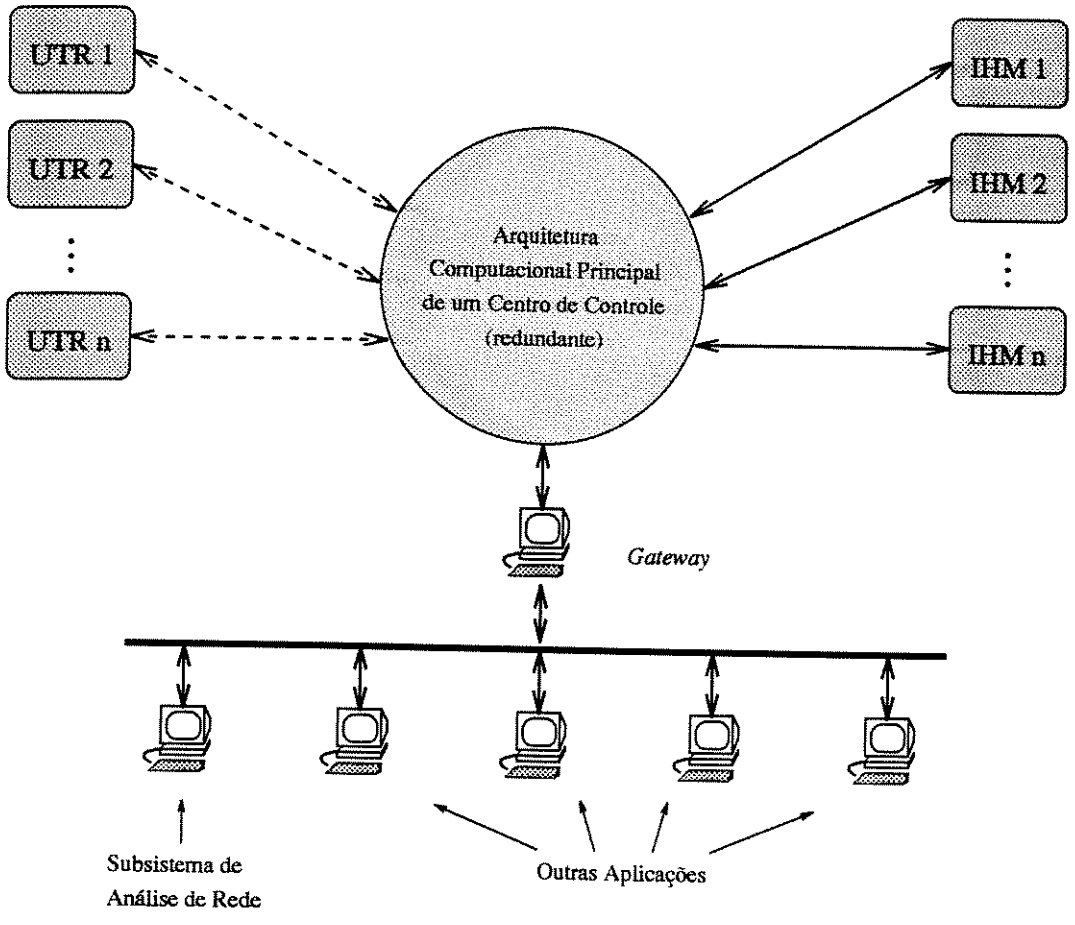

Figura 1.1: Arquitetura de um Centro de Controle de Energia

to, bem como algumas de suas extensões. A análise de aplicação de cada uma destas políticas de escalonamento para o caso dos subsistemas de análise de rede é realizada ao final do capítulo.

No Capítulo 4 são descritos e detalhados dois modelos de implementação de um escalonador para as funções de análise de rede, um estático e um dinâmico. O primeiro é baseado na política do algoritmo Taxa Monotônica com Servidor Esporádico e o segundo, no algoritmo Próximo Prazo.

No quinto capítulo são apresentados os ambientes computacionais e as características das funções utilizados para avaliar o desempenho das implementações dos módulos de escalonamento. Em seguida, são relatados os resultados dos testes de gerenciamento da execução do subsistema de análise de uma rede elétrica real, realizados com os dois escalonadores nos diferentes ambientes.

No Capítulo 6 são relatadas as conclusões acerca da viabilidade da execução do escalonamento das funções de análise de rede em ambientes computacionais que executem sistema operacional UNIX, finalizando com propostas para novas linhas de pesquisa nesta área.

Capítulo 2

Visão Geral de um Centro de Controle de Energia

2.1 Histórico

O advento dos computadores digitais nos anos 60 trouxe consigo drásticas mudanças no projeto e no uso dos sistemas de controle e supervisão. Os sistemas eletromecânicos, que precederam a introdução dos computadorizados, objetivavam o controle remoto e a simples indicação de estado (aberto/fechado) de equipamentos. A aquisição de grande quantidade de indicações de estados e valores analógicos não era prática e, por isso, os primeiros sistemas eram geralmente conhecidos apenas por *"Supervisory Control"* (Sistema de Controle Supervisório). A função de Controle e Supervisão permitia ao operador abrir e fechar disjuntores, alterar o *tap* de transformadores ou então realizar outras mudanças na rede do sistema de potência [Gau 87].

Quando os sistemas com computadores começaram a surgir, as possibilidades de aumento da aquisição de dados se tornaram aparentes e a expressão *"Supervisory Control and Data Acquisition"* ou *SCADA* passou a ser uma descrição mais apropriada do sistema. A função Aquisição de Dados fornecia o estado dos equipamentos e os valores das medidas analógicas tais como geração, fluxo de potência em linhas, freqüência, etc, os quais eram mostrados e gravados em grupos de registros. Outra importante função de controle que, na época, estava sendo executada nos Centros de Controle era o Controle Automático da Geração (CAG). A função CAG automaticamente regulava a geração para atender a demanda de carga elétrica através do controle da freqüência da carga e, ao mesmo tempo, despachava a geração necessária entre as diferentes unidades da forma mais econômica.

Os efeitos da informatização foram imediatos. A aquisição de dados pôde ser feita de forma mais abrangente, pois o armazenamento e a recupe-

ração dos dados se tornaram independentes dos grupos de registros. O controle de freqüência passou a ser mais flexível e de modificação mais simples e o despacho econômico pôde ser feito de forma mais exata. Além disso, tornou-se possível uma interface homem-máquina através do uso de visores de tubos de raios catódicos (TRCs) preto e branco, mesmo que ainda rudimentares, e a rápida aceleração do crescimento funcional [Rus 79], que foi o principal impacto da conversão para computadores digitais.

Desde então, os Centros de Controle vêm requerendo, cada vez mais, computadores com maior capacidade de armazenamento e de processamento. Dentre os motivos desta crescente demanda, podemos citar o significativo aumento no tamanho e também das partes monitoradas das redes dos sistemas elétricos de potência, o aumento do número de funções e a introdução de interfaces gráficas homem-máquina plenas (*"full graphic man-machine interfaces"*).

Aliado a este aumento de complexidade operativa do sistema, a necessidade de uma operação cada vez mais segura e rentável dos equipamentos tem tornado crescente a importância dos Centros de Controle em tempo real para a operação dos sistemas elétricos. Muito desta necessidade se deve aos elevados valores de investimento envolvidos. De acordo com a referência [Tei 90], os Centros de Controle em tempo real têm por principais funções:

- coordenar a operação energética e elétrica do sistema, de modo a minimizar a incidência e a extensão de falhas de suprimento, e
- otimizar a operação do sistema de produção, minimizando investimentos em centrais hidrelétricas e a queima de combustível em centrais térmicas.

Com a crescente dificuldade de investimentos para a expansão dos sistemas e o conseqüente aumento dos riscos de falha dos mesmos, a importância deste papel dos Centros de Controle torna-se ainda maior.

O crescimento das funções do Centro de Controle trouxe a necessidade de se desenvolverem novos projetos de configurações de computadores para os sistemas de controle de energia. Houve uma evolução dos sistemas monoprocessadores para sistemas onde as funções ficam distribuídas entre os componentes da configuração de acordo com as exigências computacionais de cada uma. Isto porque cada uma requer uma arquitetura de computador diferente para uma operação mais eficiente.

2.2 Modelo Conceitual

Um típico *Energy Management System* (*EMS*), isto é, Sistema de Gerenciamento de Sistemas de Potência, consiste de vários componentes funcio-

nais. A Figura 2.1 mostra um modelo das principais funções de um sistema, que, conceitualmente, são [Rus 79], [Hor 87]:

1. Base de dados: armazenamento dos dados.
2. Aplicativos: estimador de estado, análise de contingências, fluxo de carga ótimo, controle de tensão, fluxo de carga e simulador para treinamento.
3. Controle e despacho da geração: controle automático da geração, despacho econômico e intercâmbio entre áreas.
4. Manutenção de dados: serviços de recuperação dos dados para todas as funções.
5. Aquisição de dados: aquisição de dados e processamento de alarmes.
6. Interface homem-máquina: interface com o operador.

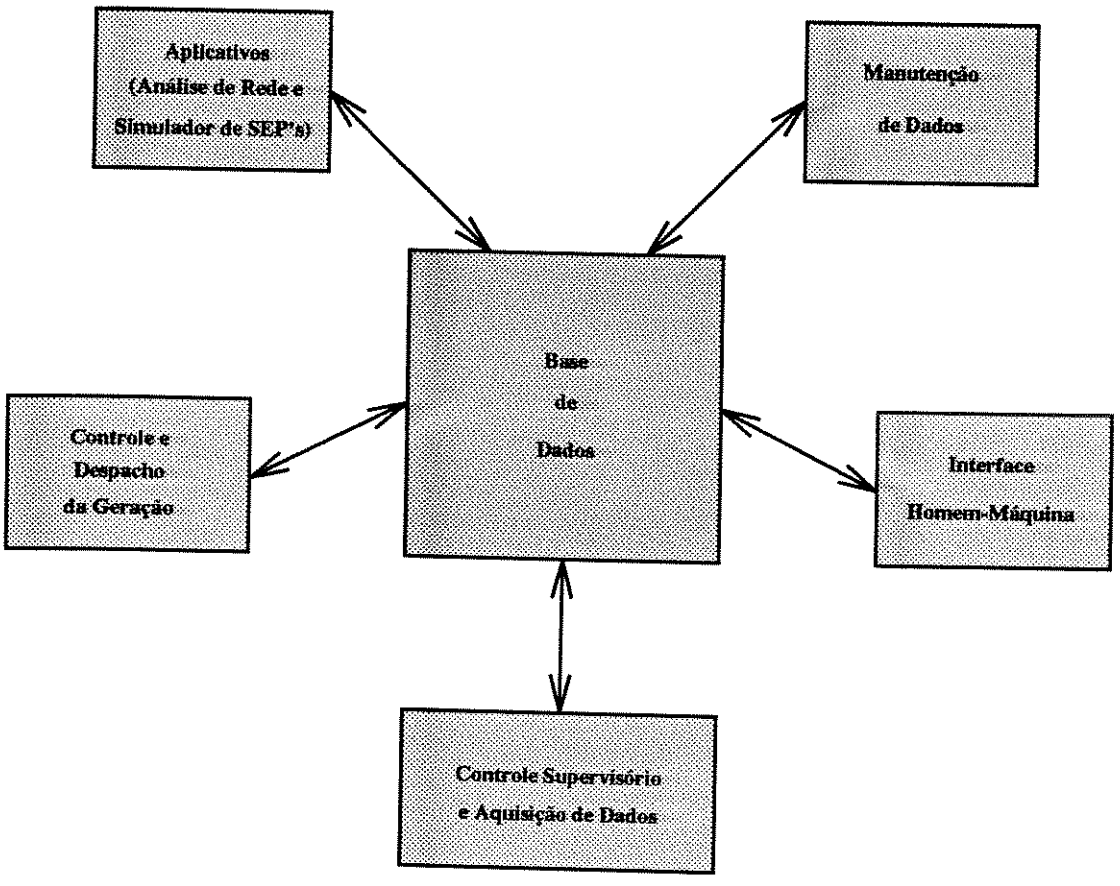

Figura 2.1: Modelo Conceitual de um Centro de Controle de Energia

As exigências de capacidade computacional tais como processamento em ponto flutuante, taxa de transação com a base de dados ou mesmo

tratamento de interrupções, variam de subsistema a subsistema. Isto é, cada componente funcional do modelo do Centro de Controle requer da configuração do sistema uma característica distinta.

Tabela 2.1: Demandas Computacionais de um *EMS* Básico

Componentes Funcionais	Carga Total Aproximada (MIPS)	Processamento Aritmético (MIPS)	Interrupções Externas por segundo	E/S Kilobytes por segundo	Base de Dados (Transações por segundo)
SCADA	1,76	0,09	800	950	5200
Controle e Despacho da Geração	0,22	0,06	-	26	2000
Análise de Rede	2,92	1,17	-	195	2200
Simulador para Treinamento	6,37	2,89	200	751	7500
Comunicações Externas	1,20	-	-	-	-

Fonte: Ferranti International Controls, 1983

A Tabela 2.1, extraída da referência [Dan 85], mostra valores de demandas computacionais típicas dos subsistemas. A partir da tabela, pode-se retirar algumas características importantes dos componentes do modelo do Centro de Controle:

- o processamento em ponto flutuante é predominantemente executado pelo subsistema de aplicativos, ou seja, pelas funções de análise de rede e pelo simulador para treinamento do operador;
- apenas o subsistema de aquisição de dados (função *SCADA*) é responsável por responder a uma alta carga de interrupções;
- o simulador para treinamento e a função *SCADA* têm que executar uma alta taxa de E/S (Entrada/Saída);
- todas as partes do modelo realizam acesso à base de dados com freqüências muito altas e
- a função de simulação de treinamento sozinha requer mais processamento que todas as outras partes do modelo combinadas.

Diante desta rápida análise do modelo conceitual e cientes de que cargas computacionais superiores a 9 MIPS[1] dificilmente podem ser sustentadas em conjunto com 800 interrupções por segundo, pois computadores que são eficientes no tratamento de interrupções não são velozes no processamento em ponto flutuante, fica claro que não apenas um só computador, mas também várias unidades de um mesmo computador, não são capazes de suprir todas as exigências funcionais de um moderno Centro de Controle [Eva 89]. Acrescenta-se a isto o

[1] Milhões de instruções por segundo

fato de que a expansão de alguns componentes do modelo não se faz da mesma forma. Um exemplo é o crescimento da função *SCADA* que deve prever a duplicação de componentes para adição de mais unidades terminais remotas (UTRs) e TRCs (crescimento horizontal), ao passo que o crescimento das funções de análise de rede é vertical, isto é, deve-se expandir ou aumentar o porte do(s) computador(es) para se acomodarem funções adicionais e maiores modelos de sistemas elétricos de potência.

Antes de discutir os aplicativos de análise de rede dos Centros de Controle, será apresentada uma breve exposição de cada um dos componentes do seu modelo, caracterizando-os em termos de suas demandas computacionais sobre o sistema.

2.2.1 A Base de Dados e sua Manutenção

As bases de dados dos primeiros sistemas de controle de energia eram de formato fixo, o que simplificava enormemente seus projetos. Os programas de usuários, no entanto, exigiam informações explícitas sobre a base de dados e sua estrutura. Isto levava a um aumento substancial de complexidade de manutenção, quando pontos eram mudados e novos programas aplicativos adicionados.

Com o advento das memórias de baixo custo, passou-se a investir em novas concepções de bases de dados, as quais tornaram possível uma considerável independência entre as funções de aquisição de dados e os programas aplicativos. As funções de aquisição de dados atualizam a parte de tempo real da base, enquanto, os aplicativos recuperam dados da base e armazenam resultados obtidos a partir deles. Tais bases de dados podem ser mais facilmente expandidas, o que implica em um aumento da importância da função de sua manutenção. Esta função é responsável pela formatação, carregamento e atualização de todos os arquivos de dados estáticos necessários a todas as outras funções [Gau 87].

Durante a fase de projeto da base de dados, deve-se objetivar, principalmente, os seguintes pontos [Rus 79]:

- organizá-la de forma a otimizar o acesso aos dados pelos programas que a utilizam,
- otimizar sua edição para mudanças que ocorram rotineiramente no ambiente de operação e
- otimizar sua geração e manutenção para que novos equipamentos, bem como novas funções, possam ser acomodados.

Enfim, uma das considerações mais importantes no projeto da base

de dados é que ela deve prover meios eficientes para armazenamento e recuperação de informações.

Os sistemas de computadores hoje em dia empregados nos Centros de Controle já resolvem em parte vários dos problemas relativos ao projeto das bases de dados, mas limitações impostas pelo tempo de acesso aos dados ainda são severas. Assim, deve-se considerar com atenção o número de acessos a disco necessários à realização de uma tarefa. Outro fator de projeto relevante é o compromisso que sempre existirá entre o grau de generalização da base de dados e a velocidade de operação, isto é, bases de dados altamente generalizadas freqüentemente apresentam excessivo *overhead* [2], o que as tornam inadequadas para uso em tempo real crítico.

A informação contida na base de dados foi classificada, na referência [Gau 87], nos seguintes tipos:

Tempo real: Medições e informações de estados, que são periodicamente adquiridas via UTRs ou informadas pelo operador. Em cada atualização os valores antigos são sobrepostos. A periodicidade da atualização pode variar de poucos segundos a horas.

Paramétrica: Informações de parâmetros são dados semi-fixos que contêm vários atributos necessários à interpretação dos dados de tempo real.

Calculados: Pseudo-valores que são calculados a partir de outros e então tratados da mesma forma que os dados de tempo real.

Aplicação: Informação que é exclusiva a aplicações específicas. Pode haver constantes, limites, mensagens armazenadas, dentre outras. Um exemplo seriam as mensagens armazenadas para uma função de processamento de alarmes.

2.2.2 A Aquisição de Dados e o Processamento de Alarmes

O único meio de interface entre o sistema de potência e o sistema de controle utilizado no passado era através das unidades terminais remotas. A função primária da UTR é adquirir dados de variáveis do sistema de potência (como fluxos de potência ativa e reativa e magnitudes de tensão) e controlar seus dispositivos (disjuntores e comutadores de *tap*) em pontos remotos [Amel83]. Protocolos de comunicação serial sofisticados com amplos esquemas de detecção de erros foram desenvolvidos para solucionar os problemas resultantes da transmissão de dados e do comando de dispositivos de controle com alta confiabilidade a longas distâncias usando canais de comunicação.

[2] Custo de computação adicional resultante do desempenho de uma operação que não é diretamente produtiva no processo em que aparece

Na falta de padrões, fabricantes de sistemas de controle desenvolveram vários esquemas de detecção de erros diferentes. As UTRs e suas interfaces com a rede de comunicação e com o sistema computacional, portanto, eram freqüentemente implementadas com *hardware* proprietário, isto é, sem padrão. Hoje em dia tem havido um aumento na diversidade de fontes de dados de variáveis do sistema, das quais o sistema de controle pode ou deve adquirir informações. Como exemplos, pode-se citar as saídas de partida e de atuação e dados de falta dos relés de estado sólido, registrador digital de dados de falta e medidores digitais de potência [Pod 93].

A aquisição de dados em um *EMS* é feita sempre de forma periódica. A maioria dos sistemas usados para aquisição de dados em concessionárias de energia consistem na transmissão dos dados das UTRs para o computador mestre, chamado *master station*, somente na ocorrência de uma requisição do mestre para a UTR. Existem duas formas pelas quais as UTRs podem responder às solicitações do computador mestre: uma é devolvendo ao mesmo o valor real ou o estado do ponto ou grupo de pontos requerido, outra é devolvendo à estação mestra somente o ponto ou grupo de pontos que tiverem tido mudança de estado ou variação de seu valor superior a um montante pré-definido desde a última requisição. Esta última opção é conhecida por "transmissão por exceção".

A principal vantagem da transmissão por exceção é a redução do *overhead* de processamento na estação mestra. Outra vantagem é a redução da média de carregamento no circuito de comunicação em relação à primeira opção. Deve-se, entretanto, prover suficiente capacidade ao circuito de comunicação para atender à situação de pior caso, quando, então, uma grande porcentagem de pontos está mudando rapidamente, como quando uma perturbação no sistema elétrico ocorre e a necessidade do despachante por dados exatos e no tempo certo é maior. A transmissão por exceção é mais usada para pontos discretos do que para valores analógicos [Gau 87].

Ainda segundo a referência [Gau 87], o processo de aquisição de dados pode ser considerado como uma agregação de vários subprocessos especializados e altamente relacionados. Estes subprocessos incluem:

- varredura interna e rápida atualização da base de dados interna da UTR;
- consulta periódica à UTR pela estação mestra;
- transmissão dos conjuntos de dados requeridos da UTR para a estação mestra;
- conferência dos dados para detectar erros de transmissão;
- conversão dos dados em unidades de engenharia;
- sobreposição dos novos estados ou valores anteriores na base de dados.

A função de processamento de alarmes é responsável por alertar o operador do Centro de Controle sobre eventos não programados e informá-lo do instante da ocorrência, da localização da estação, da identificação do dispositivo e da natureza do evento. A forma mais comum de saída do processamento de alarmes são as listas cronológicas de alarmes em TRCs, cópias impressas e alarmes sonoros. Recentemente, alguns Centros já implantaram alarmes por voz sintetizada.

Os modernos Centros de Controle, muitas vezes, tratam com milhares de indicações de estados de equipamentos, os quais podem, esporadicamente, produzir alarmes. Além disso, alguns Centros reportam alarmes para o operador de todos os erros de comunicação, das novas tentativas de restabelecimento da mesma e de outras mensagens [Amel83]. De uma forma ou de outra, todos os alarmes são importantes, mas o agrupamento deles pode não ser benéfico, especialmente em grandes Centros de Controle. Isto pode vir a criar um cenário onde o operador tenha dificuldade em diagnosticar o que está acontecendo e, conseqüentemente, afetar sua tomada de decisão. Uma alternativa que tem sido estudada é a de se tratarem tais alarmes antes de apresentá-los aos operadores, de forma a facilitar o entendimento dos eventos que os provocaram, suas causas e conseqüências. O "processamento inteligente de alarmes" consiste em uma análise em tempo real contínua, filtrando os alarmes redundantes e incorretos e apresentando ao operador um quadro mais fácil de ser analisado. Há portanto um ganho de agilidade e de confiabilidade no sistema [Wol 85].

2.2.3 O Controle e Despacho da Geração

A função de Controle Automático da Geração (CAG) controla as unidades de geração de forma a atender a demanda do sistema e manter sua freqüência e intercâmbios. Quando várias concessionárias estão conectadas, cada uma irá realizar seu próprio CAG, independentemente das outras. O CAG de um sistema multi-área faz uso do erro de controle de área (ECA), que é uma combinação linear do erro do intercâmbio em relação ao valor desejado e o desvio na freqüência. Como cada área usa o seu próprio ECA como retroalimentação, todos os erros de freqüência e de intercâmbio são levados a zero [Deb 88].

Os algoritmos mais novos utilizados para controle da geração, baseados na moderna teoria de controle de variáveis de estado, começaram a ser implementados comercialmente pela primeira vez durante o final dos anos 80. O controle da geração geralmente também incorpora o Despacho Econômico, que consiste na alocação ótima da geração entre as unidades geradoras.

O Despacho Econômico, além de fornecer os pontos básicos de operação, fornece também os fatores de distribuição, usados pelo CAG, que dão as proporções pelas quais as variações das cargas do sistema são assumidas pelas unidades geradoras sob controle. A geração é alocada entre as unidades do sistema de tal forma que sejam obedecidos os requisitos de confiabilidade (reserva girante)

e as restrições de operação do sistema (limites de operação) [Mon 83]. O controle do despacho econômico, pelo tradicional método do multiplicador de LaGrange, é tão eficiente que sua demanda de UCP (Unidade Central de Processamento) é desprezível. As novas técnicas baseadas em programação linear são apenas ligeiramente mais pesadas.

2.2.4 A Interface Homem-Máquina

A interface homem-máquina de um sistema de gerenciamento de energia consiste de dispositivos tais como TRCs, *consoles*, painéis, impressoras, alarmes sonoros além dos programas que gerenciam estes dispositivos. A prática mais freqüente tem sido a de se utilizar TRCs gráficas coloridas como interfaces primárias com o operador.

Há três tipos de dados necessários para compor as telas de um visor de TRC. Primeiro são as máscaras estáticas ou *background*, que é a informação que nunca muda para uma determinada tela. Estas informações geralmente são armazenadas de forma compacta, fazendo-se uso das capacidades de instrução de vários controladores de vídeo. O segundo conjunto de dados não aparece, de forma direta, na tela, mas é a informação requerida pelo sistema para produzir os dados que aparecem. Para cada tela deve haver descritores que identifiquem o que é necessário no visor, onde encontrá-lo e como reagir às entradas do operador. Estes descritores identificam os dados telemedidos e os calculados que devem ser mostrados, a forma na qual devem ser apresentados e a ação a ser tomada em resposta a uma seleção do operador via cursor e/ou teclado. E, por fim, o terceiro tipo que são os dados dinâmicos ou variáveis, os quais são exibidos na tela e periodicamente atualizados [Rus 79]

A migração para os visores *full graphics* nos Centros de Controle é inevitável, uma vez que o montante de informação que pode ser transmitido por tela é muito maior que com os visores convencionais (*limited graphics*). Estes subsistemas utilizam estações de trabalho (*workstations*) que oferecem o potencial para uma maior resolução, melhor apresentação gráfica dos dados e a capacidade de transformação de imagem como o uso de janelas, *zoom*, etc.

A tendência em um *EMS* é na direção de interfaces homem-máquina cada vez maiores e um número de *consoles* e TRCs também cada vez maior. Um Centro de Controle típico apresenta de 5 a 50 *consoles*, cada um tendo de dois a quatro TRCs [Hor 87], todos os quais devem ser atualizados com os novos dados com alta periodicidade. Dependendo de onde os dados estão armazenados, pode haver significantes compromissos entre o armazenamento e a velocidade de resposta. O tempo de resposta de um sistema de interface homem-máquina a uma requisição do operador é uma medida altamente visível e significativa da performance do sistema. O esforço de processamento de suporte às telas pode ser também um significante, se não dominante, fator no carregamento do mesmo.

Assim sendo, uma cuidadosa organização dos dados que dão sustentação à interface homem-máquina, bem como a base de dados em tempo real, podem ser críticas.

2.2.5 Análise de Rede e Simulador de Sistemas Elétricos de Potência

Com o crescente aumento dos sistemas de potência e do conseqüente aumento do número de informações adquiridas, a operação dos sistemas em tempo real vem requerendo, cada vez mais, a incorporação de funções de análise de redes, considerando aspectos de regime permanente e dinâmico nos Centros de Controle. Conforme vimos na Tabela 2.1, estas funções sozinhas requerem a maior parte da capacidade computacional do sistema.

Em termos de demanda computacional, os programas de análise de rede variam basicamente na quantidade de processamento e de E/S que eles requerem, a freqüência com a qual são usados e quais restrições são postas em seu uso. As demandas de UCP e de E/S destes programas variam diretamente com o tamanho da rede que está sendo modelada. No caso de grandes sistemas, o uso de esparsidade faz com que seu comportamento seja próximo do linear com o tamanho do modelo, se o código é eficiente.

Os simuladores de sistemas de potência, além de serem usados para treinar novos operadores do sistema, são aplicados também na elaboração de cursos de atualização do pessoal existente. De acordo com a referência [Rus 79], a experiência, até então adquirida, indicava que os seguintes benefícios poderiam ser atingidos com a introdução dos simuladores para treinamento:

- maior aceitação e confiança do operador em um novo *EMS*, do ponto-de-vista das interfaces homem-máquina e programas de controle;
- melhor compreensão técnica da engenharia de sistemas de potência e
- maior familiaridade com determinados sistemas através da prática em seus modelos, especialmente nas áreas de solução de problemas e preparação para emergências.

Como se devem executar todas as funções de um *EMS*, além de se simular o comportamento do sistema em tempo real, é necessário que se execute esta função em máquinas "paralelas" à configuração do *EMS*, isto é, ou em sua porção de $back - up$ ou em um terceiro subsistema autônomo.

2.3 Aplicativos de Análise de Rede Elétrica em Tempo Real

Uma das principais características dos modernos Centros de Controle é a incorporação dos aplicativos de análise de rede. Os aplicativos de análise de rede elétrica, também conhecidos por funções avançadas de análise de rede, são um conjunto de programas de execução em tempo real que visam proporcionar maior segurança e confiabilidade na operação dos sistemas elétricos de potência.

As funções de análise de rede, de acordo com suas funcionalidades dentro do sistema de supervisão e controle da rede de potência, podem ser classificadas como [Fre 92]:

- modelagem da rede em tempo real,
- análise de desempenho da rede e
- síntese de ações de controle a serem sugeridas aos operadores do sistema.

A modelagem da rede em tempo real é obtida a partir dos dados periodicamente adquiridos pelo subsistema *SCADA*. Ela fornece uma representação atualizada do sistema de interesse, o qual inclui, além das partes mais importantes do próprio sistema, a representação das partes relevantes da rede não observável, ou seja, da rede de outras empresas concessionárias de energia e de partes não monitoradas do próprio sistema. A tarefa de modelagem da rede é realizada por um grupo de funções avançadas como o configurador da rede, o analisador de observabilidade, o estimador de estado e pela determinação do modelo da rede externa.

A tarefa de análise de desempenho consiste basicamente em programas de fluxo de potência e de análise de contingências, baseados no modelo em tempo real da rede.

Finalmente, a tarefa de síntese das ações de controle tem por objetivo determinar as melhores estratégias de ação para o operador do sistema. Uma função avançada típica desta tarefa é o fluxo de potência ótimo, que determina ações de controle necessárias para levar o sistema a operar em um determinado ponto que minimize uma função objetivo [Was 91].

Estas tarefas são coordenadas e escalonadas por um módulo de gerenciamento a que chamaremos, por enquanto, de controlador. É o controlador o responsável pelo escalonamento de todos os aplicativos de análise de rede, periódicos ou não.

Estas funções podem ser executadas tanto de forma periódica e/ou mediante significativas alterações no estado do sistema (modo Tempo Real) como também por solicitação do operador (modo Estudo).

O modo Tempo Real tem como objetivo determinar e avaliar o estado operativo e o grau de segurança praticamente correntes do sistema elétrico. Seu tempo de resposta deve ser suficientemente baixo para permitir ao pessoal de operação atuar sobre o sistema elétrico com base nos seus resultados.

O modo Estudo possibilita o estudo e simulação de situações operativas passadas, atuais ou postuladas, permitindo a avaliação do efeito de manobras e de diferentes estratégias de operação, ou ainda uma análise pós-operativa. A sua utilização deve permitir ao pessoal de operação se preparar para um evento programado ou previsto para as próximas horas ou mesmo minutos [Oli 93].

A Figura 2.2 mostra uma seqüência lógica de execução em tempo real [Fre 94a] de algumas funções que compõem o subsistema de análise de rede. A periodicidade de execução destas funções deve ser determinada de acordo com a periodicidade da varredura do subsistema *SCADA* e com a utilização dos recursos computacionais do subsistema de análise de rede. Em uma mesma instância da seqüência lógica, todas as funções se referem aos dados de tempo real da última varredura anterior ao seu instante de ativação. Isto sugere que elas possuam períodos que sejam múltiplos dos ciclos do subsistema *SCADA*.

É importante notar que as funções de análise de rede são independentes, uma vez que a ativação de uma não implica, necessariamente, na ativação de outras e que elas não realizam comunicação entre si. Outro ponto relevante, é que elas, além de serem ativadas periodicamente, podem também ter sua ativação requerida por eventos.

Os bancos de dados, aos quais se refere a Figura 2.2, são as bases onde ficam armazenadas as informações sobre o sistema elétrico requeridas pelas funções de análise de rede. O banco de dados estático recebe este nome porque armazena informações que possuem baixa freqüência de atualização, como é o caso dos parâmetros de linhas e de transformadores. Os dados de alta freqüência de atualização, que são atualizados em tempo real através da varredura feita pelo subsistema *SCADA*, são armazenados em bancos de dados dinâmicos, como é o caso de valores de magnitude de tensão, fluxo de potência em linhas, estados (aberto/fechado) de chaves e disjuntores [Fre 94a].

O banco de dados em tempo real é constituído pelos dados descritivos do sistema, em conjunto com os modelos da rede e os resultados obtidos pelas várias funções e tem por objetivo final atender os requisitos de operação do sistema (Figura 2.3).

A seguir são apresentadas algumas funções avançadas de um moderno Centro de Controle. Note que estas funções não são únicas e que podem ser abordadas por técnicas diferentes.

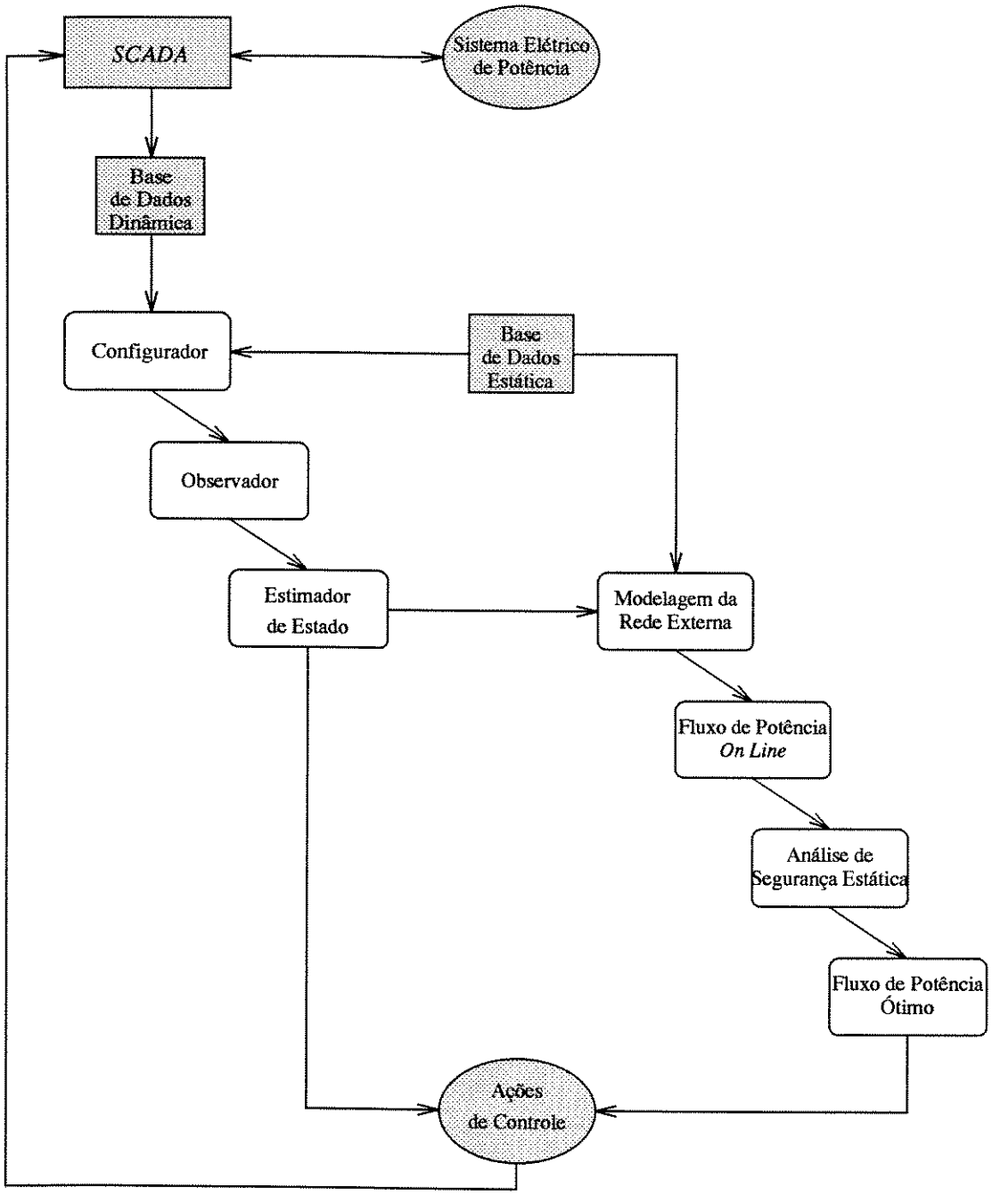

Figura 2.2: Principais Componentes da Análise de Rede Elétrica em Tempo Real

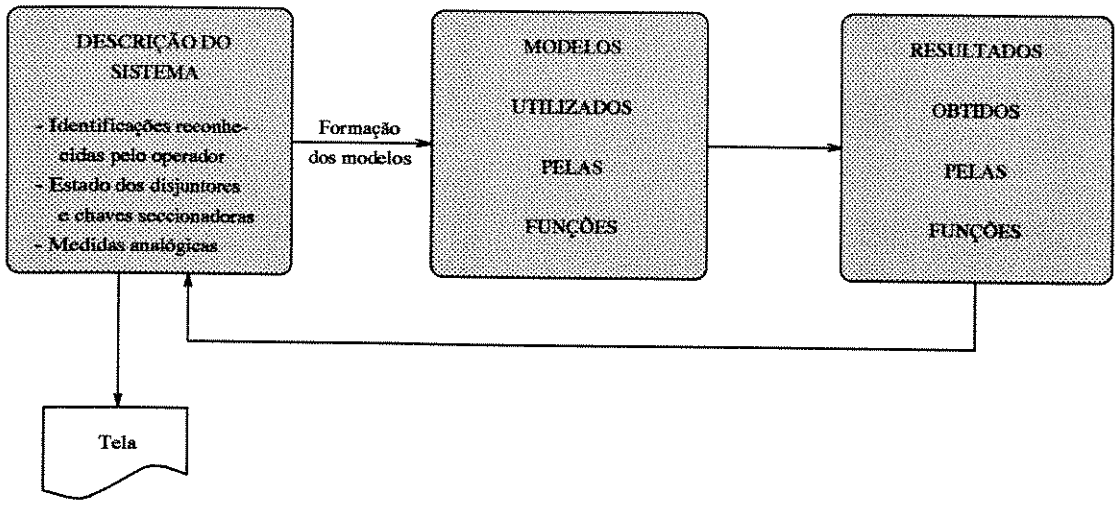

Figura 2.3: Banco de Dados em Tempo Real

2.3.1 Configurador da Rede

O configurador é a função do subsistema de análise de rede que determina a topologia do sistema elétrico em tempo real. Isto é, a partir de dados estáticos (linhas de transmissão, seções de barramento, reatores, geradores, etc) e de dados de tempo real (estados dos disjuntores e chaves - abertos/fechados) da rede, o configurador constrói um modelo que descreve quantos e quais são os nós elétricos da rede e como estes nós estão interligados para um determinado instante.

A configuração do sistema pode ser feita tanto para a rede interna, rede supervisionada, como para outras partes das redes interligadas, isto é, rede externa. É importante ressaltar que, ao contrário do que ocorre com a configuração da rede interna, na configuração da rede externa a modelagem não é feita baseada em dados de tempo real, uma vez que esta não é supervisionada.

O configurador também realiza a função de situar, dentro do modelo por ele elaborado, a localização das medidas elétricas efetuadas no sistema e transmitidas ao Centro de Controle. Estas informações preparadas pelo configurador da rede são utilizadas por outras funções avançadas como é o caso do analisador de observabilidade (que determina que parte da ilha elétrica modelada pelo configurador é observável) e do estimador de carga (que calcula o estado das partes observáveis desta ilha elétrica).

Antes da ativação de qualquer outra função de análise de rede, é necessária a execução, pelo menos uma vez, do configurador. Todavia, uma vez determinada a topologia do sistema, só se torna necessária nova ativação da função de configuração da rede, no caso de alteração na topologia do mesmo ou de perda de supervisão em alguma parte dele [Fre 92]. A alteração da

topologia ocorre sempre que há manobra de disjuntores e/ou chaves seccionadoras, o que, conseqüentemente, altera o estado do mesmo. A perda de supervisão é conseqüência da falha do subsistema *SCADA* que pode ser devido à falha de comunicação de alguma unidade terminal remota. Mesmo com a ativação do configurador diante de tais alterações no sistema, entretanto, basta que se reconfigurem as porções da rede relacionadas com a alteração topológica ou com a perda de supervisão ocorrida.

2.3.2 Analisador de Observabilidade

O analisador de observabilidade, ou simplesmente observador, determina qual é a parte observável do sistema, a partir do modelo fornecido pelo configurador e de informações do sistema de medição (número, localização e tipo das medidas). A parte observável, ou ilha(s) observável(is), do sistema é aquela na qual se é possível obter seu estado (magnitude e ângulo da tensão) a partir do conjunto de medições disponível no Centro de Controle para uma determinada configuração da rede elétrica.

De acordo com a topologia da rede e com a disposição do conjunto de medições, o sistema elétrico pode dividir-se em uma ou mais ilhas observáveis. A importância do observador reside na necessidade da determinação destas ilhas para a posterior execução do estimador de estado que só se aplica a estas partes do sistema, isto é, às ilhas observáveis. Associado a isto, o fato de tanto a topologia da rede como o conjunto de medições disponíveis variarem com o tempo, justificam o caráter de tempo real da função de análise da observabilidade, que deve ser executada antes da ativação dos aplicativos de análise de rede sempre que houver existido alguma alteração no sistema.

Uma forma eficaz, mas nem sempre viável, de se reduzir o número de ilhas observáveis em um sistema ou mesmo reduzir sua parte não observável é alocar-se estrategicamente as UTRs dentro do mesmo de modo a maximizar sua observabilidade [Fre 92]. Um outro artifício é a alocação de pseudo-medidas em determinados pontos do sistema baseadas em previsão de demanda.

A avaliação da dimensão e do número de ilhas observáveis dentro de um sistema elétrico pode ser realizada, basicamente, através de dois métodos, conhecidos por métodos topológicos e métodos numéricos. A diferença fundamental entre eles é que enquanto os métodos topológicos levam em consideração apenas a topologia da rede e a localização das medidas dentro dela, os métodos numéricos consideram também os parâmetros dos componentes do sistema e as ponderações das medidas. Isto dá aos métodos numéricos uma maior confiabilidade, pois, conforme mostrado em [Mon 92], o fato de não se considerar os parâmetros do sistema, pode levar a análise topológica à determinação errada da observabilidade de determinados sistemas. Outra vantagem dos métodos numéricos é que boa parte do processamento realizado por ele com as matrizes do sistema é aproveitada pela função de estimação de estado.

2.3.3 Estimador de Estado

A função de estimação de estado de uma rede elétrica consiste em se calcular, através de um procedimento iterativo, uma melhor aproximação do vetor de magnitudes e ângulos de tensão de barra para toda uma ilha observável, a partir do modelo da rede e do conjunto de medições disponível. As medidas utilizadas pelo estimador de estado são: magnitudes de tensão nas barras das subestações, fluxos de potência ativa e reativa em ramos, injeções de potência ativa e reativa em barras e, menos freqüentemente, magnitudes de correntes em ramos. Deve-se executar o estimador de estado para cada ilha observável do sistema em análise, arbitrando-se, para cada uma, uma referência angular distinta.

Assim como em todo processo *on-line*, as medidas adquiridas pelo sistema poderão ser afetadas por erros grosseiros. Uma vez que sejam realizadas um número maior de medições que o mínimo necessário, a teoria estatística do estimador de estado pode ser usada para "filtrar" alguns destes erros de medição. Assim, o resultado obtido pelo estimador será mais confiável que o expresso pelas medidas apenas.

Além do aumento da confiabilidade das soluções, o procedimento de estimação de estado é capaz de processar medidas que contenham erros grosseiros: detectando sua presença, identificando-as e eliminando seu efeito no processo de estimação. Os erros grosseiros podem ocorrer devido ao mal funcionamento de instrumentos de medição, erros de comunicação na transmissão de dados, falhas nos transdutores ou outros fatores.

É importante notar que para o bom desempenho do estimador de estado é necessário que o conjunto de medidas contenha um determinado grau de redundância e uma adequada distribuição dentro da rede para permitir a correlação estatística das medidas. A qualidade das medições pode ser expressa em termos de variâncias, que são usadas para ponderar uma medida em relação à outra.

Outra importante característica do estimador de estado é a possibilidade de se calcular o estado $(\mathbf{V}, \theta)$ de barras não supervisionadas, fornecendo ao operador mais informações sobre o sistema elétrico do que as disponíveis no subsistema *SCADA*.

2.3.4 Modelagem da Rede Externa

O estimador de estado é aplicado somente às partes observáveis do sistema, não se levando em consideração o que ocorre no resto da rede. Outras funções, como é o caso do fluxo de potência *on-line*, análise de contingências e fluxo de potência ótimo, utilizam informações tanto do sistema interno como do sistema externo, pois elas são direcionadas à análise da rede como um todo e o

seu resultado final é afetado pelas "reações" das partes não monitoradas da rede a uma alteração na parte monitorada. Na prática, muitas vezes o sistema interno ou de interesse é composto de partes observáveis e de partes não observáveis. O restante do sistema, chamado de sistema externo, é a parte do mesmo da qual não se possuem informações suficientes para avaliação ou que não se tem interesse. A Figura 2.4 ilustra esta divisão.

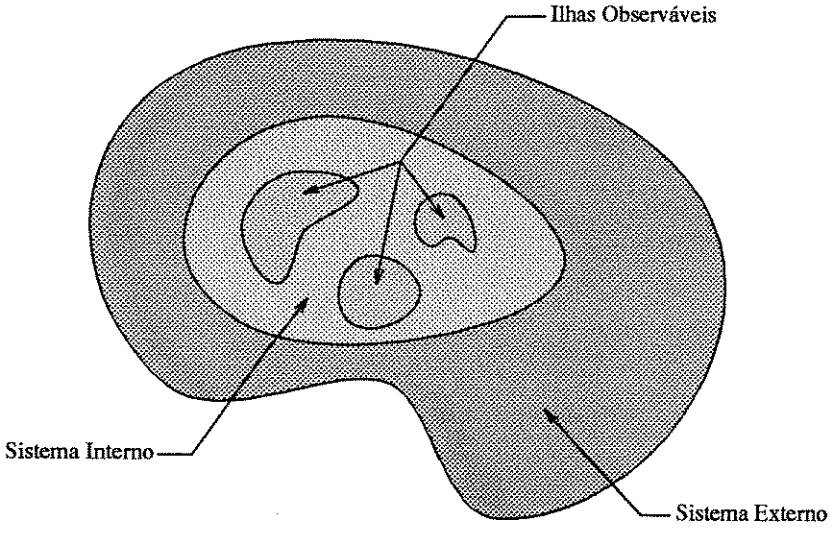

Figura 2.4: Divisões do Sistema Elétrico quanto ao Interesse de Análise

Assim sendo, para estes tipos de funções, torna-se necessária a representação, mesmo que aproximada, das regiões não observáveis através de redes equivalentes. O objetivo da modelagem destas partes do sistema é o de se possibilitar a simulação das "reações" externas quando ocorrem alterações na parte interna, sempre que elas forem importantes para o comportamento da mesma. Como, para as partes não monitoradas do sistema, não se dispõe de informações do estado da mesma em tempo real, o equivalente é obtido utilizando-se apenas os dados da parte da rede supervisionada disponíveis na base de dados de tempo real do Centro de Controle (estado e configuração) e informações sobre a configuração da rede externa. Estas podem ainda ser atualizadas através de troca de informações em tempo real entre Centros de Controle de empresas vizinhas, uma vez que sua configuração varia (mesmo que de forma pouco freqüente), minimizando, assim, erros na geração da representação da rede externa.

A Figura 2.5 mostra um modelo de decomposição de uma rede elétrica. A rede é dividida em três partes: rede interna, fronteira e rede externa.

A rede interna e a fronteira consistem na parte de interesse do sistema elétrico. A rede interna é constituída da parte monitorada do sistema mais uma parte da qual se tem interesse, mas para a qual não se tem tantas informações como na parte supervisionada. A fronteira é constituída pelas barras do sistema de interesse que se ligam à rede externa mais as linhas que interligam

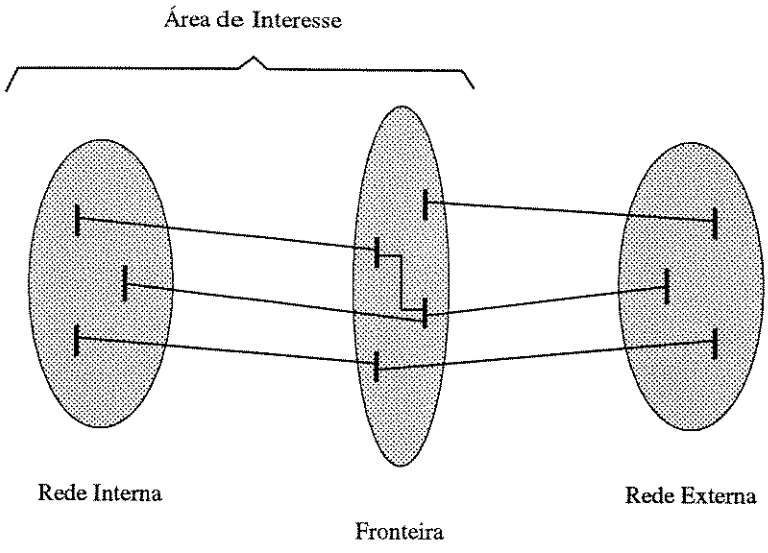

Figura 2.5: Modelo de Decomposição de uma Rede Elétrica

essas barras [Fre 94b]. A rede externa é constituída por todas as outras partes do sistema e, por isso, a redução de suas dimensões através de equivalentes (além do já exposto acima) traz vantagens computacionais significativas.

Uma solução simplória para o problema do equivalente externo seria a substituição da rede externa por injeções de potência adicionais nas barras da fronteira iguais aos fluxos existentes entre as mesmas e as barras externas. A princípio isto não acarretaria em alteração no estado da rede de interesse, mas implicaria em um "mascaramento" das reações externas a perturbações na rede interna. Isto é, uma vez que a rede externa agora está representada por injeções constantes de potência, perturbações na rede interna que antes provocavam reações da rede externa, alterando fluxos entre as barras da fronteira e as barras externas, não serão mais representadas, o que torna tal solução sem valor [Mon 83].

O método do Equivalente Ward Estendido, proposto na referência [Mon 79], para determinação de equivalentes externos tem sido empregado com sucesso em Centros de Controle. Trata-se de uma extensão da versão não linear do método proposto por Ward na referência [War 49]. Quando se utiliza o método do Equivalente Ward Estendido, sua obtenção pode ser feita em três etapas: Na primeira, é determinada a rede equivalente (admitâncias equivalentes que interligam as barras de fronteira entre si); na segunda, são determinadas as barras PV-fictícias e as ligações, também fictícias, que as ligam às barras de fronteira (para a simulação do suporte de reativos do sistema externo); e, finalmente, são calculadas as injeções equivalentes nas barras de fronteira. A Figura 2.6 ilustra a forma geral do Equivalente Ward Estendido.

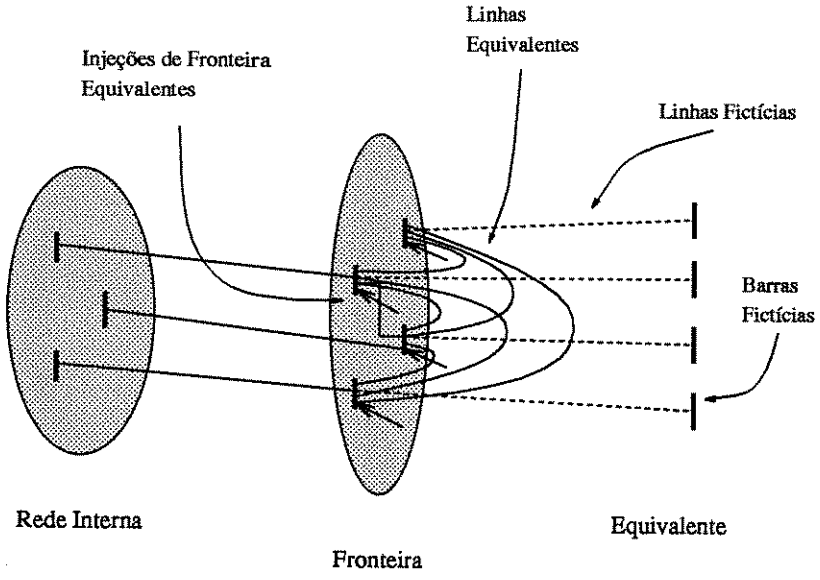

Figura 2.6: Equivalente Ward Estendido

2.3.5 Fluxo de Potência *On-Line*

O fluxo de potência é uma ferramenta tradicional utilizada para análise dos sistemas de potência em regime permanente. Uma vez que a rede tenha sido convenientemente modelada, ele fornece informações completas sobre o sistema de interesse, isto é, o estado ($\mathbf{V}$, θ) da rede, a distribuição dos fluxos de potência, dentre outras.

O programa de fluxo de potência é usado por engenheiros e/ou operadores de sistemas de potência para realização de estudos e planejamento. Já o fluxo de potência *on-line* é uma aplicação que é usada para a função de análise de segurança [IEEE77] e que pode ser usada também para fins de estudo. A finalidade do fluxo de potência *on-line* é possibilitar ao operador a avaliação do comportamento da rede elétrica sob efeito de diferentes níveis de carregamento, despachos de geração, intercâmbio entre áreas, configurações, etc [CEP 78]. A partir dos dados obtidos pelas outras funções de análise de rede, o fluxo de potência *on-line* complementa o modelo de tempo real do sistema de interesse, ajustando as injeções de potência nas barras de fronteira, para o instante no qual foram realizadas as medidas.

É importante que o estado da parte observável do sistema de interesse obtido pelo estimador seja mantido o mesmo durante a execução do fluxo de potência *on-line*. Assim, as barras do sistema observável devem ser tratadas como barras de referência ($\mathbf{V}$, θ). Devem ser também observados os limites e controles operativos da parte não observável da rede.

Existem vários métodos para a implementação do programa do flu-

xo de potência *on-line*. Assim como no caso dos programas de fluxo de potência *off-line*, os que mais se destacam são o método de Newton e suas versões desacoplado e desacoplado rápido, devendo também serem considerados para a aplicação em tempo real.

Há um compromisso entre a exatidão, a confiabilidade e a robustez com o tempo de resposta desejado. Na referência [CEP 78] é apresentada uma discussão minuciosa sobre as características de cada um destes métodos, com suas vantagens e desvantagens, como também critérios para a escolha do algoritmo adequado. É certo que, independentemente do sistema ao qual se deve aplicar a função de fluxo de potência *on-line*, a confiabilidade de convergência é o fator isolado mais importante. Se o algoritmo apresenta problemas neste sentido, ele não deve ser usado. Assim, as alternativas de escolha entre os vários métodos disponíveis é normalmente pequena.

Os métodos desacoplados são mais indicados, uma vez que, além de diminuírem o esforço computacional, eles requerem uma menor capacidade de armazenamento na memória para suas submatrizes de acoplamento da matriz Jacobiana do método de Newton. Outra característica importante dos métodos desacoplados que os tornam atrativos para a aplicação em tempo real é a de apresentarem uma melhor convergência nas primeiras iterações, as quais são as mais importantes se não existir necessidade de uma solução com precisão elevada.

Sob estes mesmos aspectos, a versão do desacoplado rápido que utiliza submatrizes constantes e simétricas, apresentada por Stott e Alsaç [Sto 74] e, posteriormente, ampliada e fundamentada nas referências [Amer89] e [Mon 90], apresenta ainda vantagens de capacidade de memória e de avaliação de contingências pelo método de compensação, sem desvantagem de convergência em relação à versão do método de Newton desacoplado. Devido ao seu desempenho, portanto, o método desacoplado rápido tem sido amplamente utilizado para a solução do problema de fluxo de potência *on-line*.

2.3.6 Análise de Segurança Estática

O objetivo da análise de segurança é avaliar o estado de operação do sistema, isto é, se ele está operando em um estado seguro ou não e, caso o estado seja inseguro, sugerir ações que devam ser tomadas para levar o sistema novamente a um estado de operação seguro. Para a análise em tempo real, as funções de segurança utilizam o modelo do sistema gerado pelas funções de estimação de estado e modelagem da rede externa e, por enquanto, são feitas apenas avaliações em regime permanente, não se realizando ainda análise de segurança dinâmica. De acordo com a referência [Bal 92], a análise de segurança pode ser dividida em tarefas de acordo com sua funcionalidade.

A primeira tarefa, conhecida por Monitoramento de Segurança, seria a identificação do estado atual, normal ou não, do sistema. O monitora-

mento de segurança utiliza as medições de tempo real, como por exemplo, fluxos e injeções de potência ativa e reativa, magnitudes de tensão nas barras, fluxo de correntes nos ramos, estado (aberto/fechado) de disjuntores e chaves seccionadoras, posição de *taps* de transformadores, etc. A partir destes dados, a tarefa de monitoramento de segurança faz uma conferência dos dados analógicos comparando-os aos limites pré estipulados para determinar se o sistema está em estado de emergência ou próximo a ele. Isto pode ser realizado sempre que um novo conjunto de dados de tempo real se torne disponível. No caso de identificação de um estado de emergência, deve-se proceder à execução de ações corretivas adequadas para que o sistema retorne ao estado normal de operação.

A segunda tarefa é responsável por avaliar se o estado normal de operação do sistema é seguro ou inseguro com respeito a um conjunto de contingências, comumente chamada de Avaliação de Segurança. Esta etapa é a fase que demanda maior esforço computacional por requerer a simulação de contingências no sistema para avaliação de seu comportamento. As contingências são simuladas a partir do ponto de operação do sistema calculado pelas funções de estimação de estado e fluxo de potência *on-line*, denominado "caso base".

Por fim, a terceira tarefa, conhecida por Melhoramento de Segurança. É através dela que, uma vez detectado que pelo menos uma contingência pode levar o sistema ao estado de emergência (alerta), isto é, o sistema está operando em um estado inseguro, são determinadas ações preventivas que devem ser tomadas para torná-lo seguro.

As contingências típicas em um sistema elétrico de potência, simuladas durante a etapa de avaliação de segurança, são as perdas simples ou múltiplas de linhas, unidades de geração, transformadores ou de cargas. A análise de contingências por si só é uma tarefa simples e que possui vasta literatura propondo um grande número de métodos para sua simulação. Ela consiste, fundamentalmente, na solução de um fluxo de carga para cada contingência, a partir do caso base, que conste em uma lista previamente especificada. Assim, o estado calculado para cada caso pós-contingência é então comparado aos limites de operação. Para controle em tempo real, entretanto, surgem, na prática, algumas dificuldades para a viabilização da aplicação de análise de contingências devido ao tempo requerido para a execução desta função. Basicamente, os fatores responsáveis são:

- número de casos que devem compor a lista;
- precisão requerida para convergência e pelos modelos (representação de controles);
- esforço computacional requerido pela técnica de solução de fluxo de potência adotada, já que ele é executado uma vez para cada caso da lista.

A solução para a primeira dificuldade exposta no parágrafo anterior é a adoção de uma técnica que reduza a lista de contingências original

"filtrando" as mais importantes. Esta etapa da análise de contingências é chamada de Seleção de Contingências. A elaboração desta nova lista é realizada através de métodos aproximados, se possível, lineares, com técnicas computacionais apropriadas, para se obter resultados rápidos, porém relativamente precisos. Um método de seleção muito usado é o de se executar a primeira iteração do fluxo de potência desacoplado rápido. Esta lista, além de conter em ordem decrescente de severidade os casos de contingências mais importantes, deve também ser atualizada em tempo real, uma vez que as contingências mais severas mudam à medida que o estado do sistema se altera. A seleção de contingências, por ser a parte da análise de contingências que oferece maior facilidade para diminuição do esforço computacional por ela requerido, tem sido objeto de vários estudos.

O problema da precisão requerida pelos modelos e de convergência são tratados através de refinamentos e ajustes adequados nos algoritmos de acordo com as características do sistema, com especial atenção à modelagem dos controles, à simulação dos efeitos dos sistemas externos e aos critérios de testes de convergência apropriados a cada caso simulado.

Finalmente, são realizadas as " avaliações de contingências ", através da utilização de técnicas eficientes de fluxos de potência C.A. e métodos especiais de exploração da esparsidade característica das matrizes de sistemas de potência, como, por exemplo, o método da compensação [Tin 72], o que reduz bastante o problema do esforço computacional demandado. Os casos são simulados, em ordem decrescente de severidade, até que seja satisfeito o critério de parada adotado, que pode ser por tempo, número de casos ou quando não há mais violações de limites no estado pós-contingência.

Como resultado da análise de contingências deve-se gerar uma nova lista que contenha as contingências que, caso ocorram, levem o sistema a um estado de emergência, preferencialmente em ordem crescente de severidade. Baseada nesta lista, é que, segundo a referência [Sto 87], o operador do sistema ou uma função automática de despacho com restrições de segurança pode então tomar a ação de:

- alterar o estado de operação pré-contingência para abrandar ou eliminar a emergência resultante da contingência,
- desenvolver uma estratégia de controle que irá aliviar a emergência, caso ela ocorra, ou
- decidir não fazer nada, baseado no fato da emergência pós-contingência ser pequena e/ou muito improvável.

2.3.7 Fluxo de Potência Ótimo

A solução do problema de fluxo de potência ótimo (FPO) determina o ponto de operação ótimo do sistema para uma determinada condição estática de carga. Isto é feito através da otimização de uma função objetivo sujeita às restrições de operação. O atributo a ser otimizado pode ser o custo de operação, as perdas de potência ativa na transmissão, mínimo desvio de um ponto de operação, etc, ou mesmo uma combinação deles. Baseado no resultado do FPO pode-se determinar ações de controle que levem o sistema a este ponto de operação desejado. Dependendo da confiabilidade da implementação do FPO e da "política" de operação da concessionária, estas ações podem, tanto serem tomadas automaticamente por meio de controles de malha fechada, como serem, apenas, reportadas ao operador do sistema [Bal 92].

Uma extensão do problema do FPO é o fluxo de potência ótimo com restrições de segurança (FPORS) [Als 74]. Além do objetivo do FPO, o FPORS visa determinar um ponto de operação que, quando da ocorrência de uma contingência pertencente à uma lista pré-especificada, o estado pós-contingência permaneça viável. Isto é, busca-se um estado de operação ótimo e seguro.

O fluxo de potência ótimo com restrições de segurança é um problema de otimização não linear de várias variáveis e controle, o que o torna um aplicativo de demanda computacional extremamente grande. Esta característica implica em uma grande dificuldade na sua aplicação como função de tempo real. Já foram desenvolvidos vários métodos para sua solução, que incluem técnicas de programação não linear, programação quadrática sucessiva e programação linear, sem que, contudo, se tivesse alcançado uma solução satisfatoriamente rápida e confiável para a aplicação em tempo real.

Recentemente, tem-se investido no desenvolvimento de técnicas e algoritmos de processamento paralelo para a solução de FPORS. A dificuldade da aplicação de soluções paralelas reside no fato de que nem sempre as melhores versões consistem na simples paralelização de algoritmos seqüenciais e de que deve-se procurar soluções paralelas que independam do equipamento utilizado. Na referência [Rod 94] é proposto um modelo assíncrono, baseado em uma extensão do algoritmo seqüencial de relaxação originalmente proposto por Stott e Marinho [Sto 79], para a solução paralela do problema de FPORS. Neste artigo, é adotado o paradigma *produtor-consumidor* para o intercâmbio de mensagens e dados entre os processadores que realizam as tarefas em que o problema do FPORS é decomposto. Através do bom balanceamento de carga entre os processadores resultante da execução assíncrona, a técnica adotada levou a resultados eficientes e de bom grau de portabilidade.

Enfim, à medida que novos desenvolvimentos na área sejam realizados e que a capacidade de processamento computacional dos Centros de Controle se torne maior, é inevitável que o FPORS se torne tão difundido nos modernos Centros de Controle como hoje é o fluxo de potência convencional [Sto 87].

Capítulo 3

Algoritmos de Escalonamento em Tempo Real para as Funções de Análise de Rede Elétrica

No capítulo anterior, foram apresentadas as funções de análise de rede que têm sido incorporadas aos modernos Centros de Controle. A aplicação destas funções para o controle em tempo real depende da existência de um controlador que gerencie a execução destes aplicativos, sejam eles periódicos ou não.

O controlador é um módulo de gerenciamento que coordena e escalona as funções de tempo real garantindo que elas cumpram seus prazos de execução. A estes controladores chamam-se de escalonadores de sistemas de tempo real. Existem vários tipos de escalonadores amplamente discutidos na literatura relacionada à área. Cada tipo de escalonador se adequa a um determinado tipo de sistema de acordo com as características das funções a serem escalonadas.

Neste capítulo será feita uma breve apresentação do que são sistemas de tempo real, das características básicas de um escalonador de tempo real e de alguns algoritmos de escalonamento. Uma discussão detalhada pode ser encontrada na referência [Mag 86]. Em seguida é apresentada uma discussão sobre qual política de escalonamento é mais adequada ao caso das funções de análise de rede elétrica.

3.1 Introdução a Sistemas de Tempo Real

Um sistema de tempo real (STR) é todo aquele que, estimulado por algum evento externo, deva fornecer uma resposta em um tempo finito especificado. Para que isto seja possível, é preciso que o sistema seja capaz de tratar, imediatamente, as alterações referentes aos eventos que sejam previsíveis de ocor-

rerem, mas para os quais é impossível de se determinar o exato instante em que ocorrerão. No caso de sistemas de tempo-real controlados por computadores, tal atribuição compete a um *software* denominado **núcleo de tempo real** *(kernel)*.

Muitas vezes tais sistemas compreendem tanto atividades que devem ser cumpridas dentro de prazos de tempo previamente especificados como outras que não são críticas do ponto-de-vista do tempo. A estas atividades, com restrições de tempo ou não, denominamos tarefas ou processos. O princípio básico dos sistemas de tempo real é o de que eles devem gerenciar tais tarefas, executando as que são críticas de modo que cada uma cumpra as suas restrições de tempo e minimizando o tempo de resposta daquelas não críticas.

As tarefas também podem ser classificadas como tarefas periódicas ou não periódicas. Tarefas periódicas têm um intervalo de tempo regular entre suas ativações igual ao seu período e possuem um limite de tempo para terminar sua execução, o qual é denominado prazo (*deadline*). Elas devem ser executadas uma única vez dentro deste período constante. Estas tarefas podem existir desde o início da operação do sistema, bem como serem criadas dinamicamente durante a mesma. As tarefas periódicas podem ser críticas ou não em relação ao seu prazo, assim como as tarefas aperiódicas. Porém, estas não possuem periodicidade entre suas instâncias. Quando uma tarefa aperiódica é crítica em relação a seu prazo, ela recebe o nome de tarefa esporádica. A necessidade de satisfazer às peculiaridades de cada uma de suas tarefas críticas contribui de forma direta a tornar o projeto dos sistemas de tempo real uma atividade complexa.

As conseqüências do não cumprimento dos prazos das tarefas irão depender da aplicação à qual se destina o sistema. Para se evitar os atrasos de execução das tarefas, deve-se estudar uma alocação prévia de todos os recursos por elas requisitados.

Podemos concluir que os sistemas de tempo real diferem dos sistemas tradicionais quanto ao fato das restrições de tempo, os prazos, serem associadas às tarefas e, caso não sejam cumpridas, podem vir a provocar conseqüências desastrosas ao sistema. Isto é, para os sistemas de tempo real, precisão e tempo são características fortemente acopladas, ao contrário do que acontece nos sistemas tradicionais.

3.1.1 Características das Tarefas nos STRs

Outras características, que não apenas a periodicidade e as restrições de tempo, podem ser associadas às tarefas de um sistema de tempo real, como por exemplo, os recursos por elas requeridos, sua precedência, concorrência e comunicação entre as mesmas e, ainda, sua própria alocação, dentre outras.

Na maior parte dos sistemas de tempo real com restrições severas de tempo, considera-se que os recursos requeridos por uma tarefa para sua

execução estejam disponíveis assim que a tarefa é ativada. Como as tarefas podem requerer recursos específicos como, por exemplo, unidades de E/S (Entrada/Saída) e processadores especiais, que freqüentemente estão sendo disputados (concorridos), é importante que se considere tal característica para efeito de projeto do sistema. É importante, também, ressaltar que o processador é considerado um requisito primário de toda tarefa.

A precedência de uma tarefa é o seu relacionamento com as demais tarefas do sistema. Uma tarefa é dita preceder outra quando se faz necessário que sua execução chegue ao fim antes que a segunda comece. Quando não há restrições de precedência entre duas ou mais tarefas, elas são ditas serem independentes. As restrições de precedência tomam importância no projeto à medida que o sistema passa a apresentar tarefas cada vez mais complexas, isto é, tarefas que acessem mais e mais recursos. Isto porque estas tarefas são melhor manipuladas através de seu particionamento em tarefas menores, relacionadas por restrições de precedência e cada uma delas requisitando um subconjunto dos recursos.

Pode haver tarefas que se comuniquem umas com as outras visando troca de dados ou mesmo sincronização. Neste caso, é necessário se utilizarem primitivas de comunicação como semáforos, monitores, "*rendez-vous*", troca de mensagens, etc. Esta característica tende a ser muito explorada nos sistemas de tempo real distribuídos, uma vez que proporciona a existência de tarefas cooperantes.

Nos sistemas de tempo real com restrições severas de tempo, as tarefas podem ainda ser classificadas quanto à capacidade de serem interrompidas ou não durante sua execução. Há tarefas que podem ser interrompidas em qualquer instante de tempo por uma tarefa que tenha preempção (precedência) na "posse" do processador e há outras que, uma vez em "posse" da UCP (Unidade Central de Processamento), executam até o seu término.

3.1.2 Requisitos Básicos no Desenvolvimento de STRs

A alocação da UCP a uma determinada tarefa dentre as que se encontram prontas para execução é chamada de **escalonamento**, sendo realizada por um elemento do núcleo denominado de **escalonador** (*scheduler*).

Muitas vezes as características das várias tarefas que compõem uma aplicação de tempo real são conhecidas a *priori*. Assim, a definição da seqüência de execução destas tarefas (escalonamento) pode ser realizada estática ou dinamicamente. No caso das tarefas periódicas, uma especificação estática do escalonamento é normalmente realizada, não acontecendo o mesmo no caso das tarefas não periódicas ou de tarefas periódicas que, freqüentemente, alterem o seu período dinamicamente.

Na prática, entretanto, a maior parte das aplicações de tempo real envolve tanto componentes que podem ser especificados estaticamente como componentes dinâmicos. Assim, para que o sistema seja capaz de responder aos eventos que ocorrerem em instantes e em ordens imprevisíveis, o núcleo de tempo real deve possuir alguns requisitos básicos, os quais dão ao mesmo a capacidade de afetar diretamente o escalonamento das tarefas. Primeiramente, é necessário que o sistema seja informado das ocorrências de eventos importantes no ambiente externo e que ele apresente respostas rápidas a elas. Outra característica importante é que o sistema seja capaz de executar um rápido chaveamento do contexto das tarefas, mantendo sempre mais de uma tarefa simultaneamente na memória.

Muitas vezes, deve ser possível a interrupção da execução de uma tarefa para que seja executada outra e, ainda, que a primeira reassuma no mesmo ponto em que foi interrompida. Isto é, devem existir meios de retardar a execução de uma determinada tarefa por algum período de tempo. O núcleo deve ainda apresentar-se de forma (tamanho) reduzida, procurar minimizar os intervalos de tempo em que as interrupções do sistema se encontram desabilitadas, gerenciar a memória de forma eficiente e fornecer mecanismos adequados para proteção de código e dado na memória.

Uma vez que as atividades de uma tarefa podem depender das atividades de uma outra, é de fundamental importância que o núcleo também provenha mecanismos de comunicação entre tarefas. Por fim, como várias tarefas devem poder competir pelo processador (ou processadores), o núcleo deve decidir, baseado em algum critério, qual tarefa deve executar primeiro, não esquecendo, entretanto, que tarefas menos importantes também devem ter a possibilidade de execução. Assim, é importante que as tarefas primordiais, ocasionalmente, interrompam a sua execução.

É preciso ressaltar que estes mecanismos são implementados de modo a fornecer um tempo de resposta rápido e, rápido, aqui, é um termo relativo e não é suficiente quando o problema envolve restrições de tempo real.

3.2 Escalonadores em STRs

Tradicionalmente, a técnica usada em sistemas que não se preocupam com o cumprimento das restrições de tempo das tarefas é a técnica de escalonamento com "prioridade dirigida". Nele, a importância das tarefas, que ditará a ordem de execução delas, é definida através de um valor de prioridade que é atribuído a cada uma. Mas, se as prioridades das tarefas são determinadas sistematicamente de tal forma que as restrições de tempo possam ser levadas em consideração, então tal escalonamento pode ser usado para sistemas de tempo real.

Basicamente, o projeto de um escalonador pode ser classificado quanto à capacidade de contemplar ou não as preempções das tarefas sob seu controle. Um escalonador que não contemple as preempções de suas tarefas nunca suspende a execução das mesmas, mesmo que outra tarefa de prioridade superior tenha sido notificada da ocorrência de algum evento externo, que a tenha ativado. Neste caso, uma vez que a tarefa tenha ganho a UCP para execução, ela só perderá o controle do processador para o atendimento de uma interrupção que tenha ocorrido ou caso suspenda ou bloqueie a si própria. Assim, as tarefas executadas em sistemas com estes escalonadores devem sempre cooperar entre si limitando a quantidade de processamento realizado entre suspensões voluntárias. Embora esta limitação dos sistemas que não contemplem as preempções de suas tarefas não traga conseqüências danosas aos sistemas cujos requisitos de tempo não são críticos, em certos casos, ela pode impossibilitar a sua aplicação. Isto ocorre, por exemplo, em sistemas em que a ocorrência de um evento possa sinalizar uma tarefa para iniciar sua execução, mesmo que o seu prazo para término esteja próximo e que o controle da UCP esteja com outra tarefa. Neste caso, faz-se necessário uma política de escalonamento de preempção.

Um escalonador que contemple a preempção de tarefas é capaz de suspender uma tarefa, que esteja sob seu controle, no instante da ocorrência de um evento que desperte outra de maior prioridade. Isto é, em qualquer tempo a tarefa de maior prioridade é a que se executa. A tarefa interrompida, porém, é recuperada posteriormente.

É fácil de se notar que a primeira concepção possibilita uma considerável simplificação no projeto do escalonador, proporcionando uma redução na memória requerida e no tempo de execução. Uma outra vantagem deste escalonamento é que ele reduz a possibilidade de se gerarem problemas de consistência de dados que são compartilhados por mais de uma tarefa. Devido à sua inflexibilidade, no entanto, ele não favorece a elaboração de algoritmos de escalonamento ótimos.

Por fim, os algoritmos de escalonamento, de acordo com a arquitetura do sistema, podem ainda ser divididos em duas categorias: os centralizados e os distribuídos. Um sistema centralizado, como é o caso dos sistemas monoprocessadores e sistemas multiprocessadores com memória compartilhada, possui custo de comunicação entre os mesmos insignificante, quando comparado com o tempo de execução das tarefas. Já os sistemas de computadores interligados em redes locais são exemplos de sistemas distribuídos, pois neles os processadores estão " espalhados" por diferentes pontos, tornando relevante o custo de comunicação entre eles.

Durante o desenvolvimento deste trabalho, abordaremos apenas os escalonadores para ambientes monoprocessadores devido à sua simplicidade e ao fato de atenderem aos requisitos de um escalonador para as funções de análise de rede.

Qualquer que seja o algoritmo de escalonamento adotado, o escalonador deve realizar a escolha da tarefa a ser executada, coordenando todos os recursos do sistema, de forma a satisfazer os seguintes objetivos [Sha 90]:

- garantir que as tarefas com restrições rígidas de tempo satisfaçam seus prazos;
- obter um alto grau de escalonabilidade para as tarefas com restrições de tempo. O grau de escalonabilidade é o valor do fator de utilização do processador[1], onde toda utilização da UCP menor ou igual a esse grau, garanta que todas as tarefas com prazos possam ser cumpridas;
- fornecer bons tempos médios de resposta para as tarefas não periódicas que não possuam prazos e, finalmente;
- garantir estabilidade em momentos de sobrecarga transitória do sistema. Podem acontecer situações em que o sistema fique sobrecarregado por diversos eventos, de tal forma, que se torne impossível satisfazer a todos os prazos, devendo-se garantir pelo menos os de tarefas críticas selecionadas. Um escalonador com esta característica é dito estável.

Quando os sistemas de tempo real possuem tarefas sujeitas a um grande número de restrições, eles podem atingir um grau de complexidade considerável. A determinação da escalonabilidade destas tarefas é um problema NP-completo [Xu 90], por isso, normalmente, a busca de escalonamentos para esses sistemas é feita por algoritmos heurísticos sub-ótimos ou por técnicas de escalonamento *off-line* complementadas com procedimentos *on-line* eficientes [Mel 93].

3.3 Algoritmos de Escalonamento de Preempção por Prioridade

Conforme mencionado anteriormente, os algoritmos de escalonamento de preempção por prioridade, isto é, algoritmos de escalonamento com "prioridade dirigida" que são capazes de suspender uma tarefa devido à preempção de outra, podem ser classificados quanto à definição de prioridades das tarefas. Quando as prioridades são atribuídas às tarefas no instante da inicialização do sistema e não mais alteradas durante a execução do mesmo, diz-se que o algoritmo de

[1] Fator de utilização de processador (U) por n tarefas periódicas T_1, T_2, ..., T_n, onde cada tarefa T_i possui um tempo de execução C_i e um período P_i é definido como sendo

$$U(n) = \sum_{i=1}^{n} \frac{C_i}{P_i} \tag{3.1}$$

escalonamento é estático. Isto requer um conhecimento prévio das características das tarefas que devam ser "despachadas" pelo sistema.

O algoritmo dinâmico é aquele em que a atribuição de prioridades às tarefas é realizada em tempo de execução. A determinação de tais prioridades é feita conforme a política de escalonamento do sistema, podendo ser alterada de acordo com a evolução do mesmo.

Nas próximas seções, apresentaremos alguns algoritmos de escalonamento estáticos: primeiro, o clássico algoritmo *Taxa Monotônica*, desenvolvido por Liu e Layland [Liu 73], e depois algumas extensões de sua filosofia para o tratamento de sistemas que envolvam tanto tarefas periódicas como tarefas aperiódicas portadoras ou não de restrições de tempo. Nas seções seguintes apresentaremos dois algoritmos dinâmicos: O algoritmo *Earliest Deadline* (Próximo Prazo), também proposto por Liu e Layland [Liu 73] e o *Least Laxity First* (Menor Folga Primeiro). No final do capítulo é discutida a aplicação desses algoritmos no problema de escalonamento das funções de análise de rede.

Para a análise dos algoritmos, consideraremos sempre que todas as tarefas periódicas estarão em estado de pronto no instante de tempo inicial e que seus prazos são iguais aos seus períodos.

3.3.1 Taxa Monotônica

O algoritmo *Taxa Monotônica* é um algoritmo de escalonamento estático para tarefas periódicas, independentes, que possam ser interrompidas e para sistemas monoprocessadores. Este algoritmo atribui, em modo *off-line*, prioridades fixas às tarefas de acordo com o inverso de seu período, isto é, as tarefas que apresentem menor período recebem maiores valores de prioridade e as de maior período, menor prioridade. Desta forma, a qualquer instante de execução, fica garantido que a tarefa que estiver sendo executada será sempre a que tiver o menor período, já que a posse da UCP é sempre concedida à tarefa de maior prioridade dentre as que estiverem em estado de pronto.

Liu e Layland, na referência [Liu 73], mostraram que o algoritmo *Taxa Monotônica* é ótimo para processos periódicos e independentes, uma vez que não existe nenhum outro algoritmo de associação de prioridades pré-fixadas que possa escalonar um conjunto de processos do mesmo tipo sem que o *Taxa Monotônica* também o possa. Eles mostraram, ainda, que o algoritmo *Taxa Monotônica* sempre respeitará o prazo de qualquer conjunto de tarefas periódicas e independentes se a condição do seguinte teorema for satisfeita:

Teorema 1 *Para um conjunto de n tarefas periódicas e independentes com prioridade fixa, o mínimo limitante superior do fator de utilização do processador é* $U(n) = n(2^{(1/n)} - 1)$.

Isto é, o *Taxa Monotônica* cumprirá o prazo de tais tarefas sempre que:

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{(1/n)} - 1) = U(n) \tag{3.2}$$

onde

C_i = tempo de execução da tarefa T_i
P_i = período da tarefa T_i
n = número de tarefas

Assim, para o caso de duas tarefas a taxa de ocupação da UCP pode ser de até 83% e para valores de n muito grandes o limite superior de utilização do processador é $\mathbf{U}(\infty) = \ln 2 = 69\%$. O Teorema 1 é uma condição de pior caso, portanto, uma condição de suficiência.

O limite do Teorema 1 é muito pessimista, pois o conjunto de tarefas de pior caso é muito improvável de ser encontrado na prática. Para um conjunto de tarefas aleatoriamente escolhido, o limite provável é de 88%, conforme mostrado na referência [Leh 89]. Para saber se um conjunto de tarefas com fator de utilização maior que o limite do Teorema 1 pode cumprir seus prazos, pode-se usar um teste de escalonabilidade exato baseado no Teorema da *Zona Crítica*, derivado da referência [Liu 73]:

> **Teorema 2** *Para um conjunto de tarefas periódicas e independentes, se todas tarefas cumprirem seus primeiros prazos de término de execução quando elas forem ativadas ao mesmo tempo, então os prazos serão sempre cumpridos para qualquer combinação de tempos de ativação.*

Este teorema serve como base para um teste de escalonabilidade exato para conjuntos de tarefas periódicas e independentes sobre o algoritmo *Taxa Monotônica*. Para saber se cada tarefa pode completar a execução de sua primeira instância antes de seu primeiro prazo, precisa-se checar os "pontos de escalonabilidade". Os pontos de escalonabilidade para a tarefa τ são o prazo da primeira instância da tarefa τ e os fins dos períodos de tarefas de maior prioridade que a tarefa τ, anteriores ao primeiro prazo desta. Para proceder o teste, basta checar se todas as tarefas conseguem completar sua execução em qualquer um dos seus pontos de escalonabilidade. Segundo o Teorema 2, se todas as tarefas conseguirem cumprir seus prazos dentro de seu primeiro período, elas sempre os conseguirão cumprir.

O teste requerido pelo Teorema 2 pode ser representado por um teste matemático equivalente, proposto na referência [Leh 89]:

Teorema 3 *Um conjunto de n tarefas periódicas e independentes escalonável pelo algoritmo Taxa Monotônica irá sempre cumprir seus prazos, para quaisquer tempos de ativação iniciais, se e somente se*[2]

$$\min_{(k,l)\in R_i} \sum_{j=1}^{i} C_j \frac{1}{lT_k} \left\lceil \frac{lT_k}{T_j} \right\rceil \leq 1 \qquad (3.3)$$

onde

C_j e T_j são o tempo de execução e o período da tarefa τ_j, respectivamente, e $R_i = \{(k,l)|1 \leq k \leq i, l = 1, ..., \lfloor T_i/T_k \rfloor\}$.

Os limites superiores de utilização do processador, requeridos para garantia do cumprimento do serviço de escalonamento de tarefas em tempo real, mostrados nesta seção, se aplicam ao caso de um conjunto de tarefas que possuam períodos genéricos. No caso particular, do conjunto de tarefas a serem escalonadas possuírem períodos tais que[3] $\{T_n/T_i\} = 0$, para $i = 1, 2, ..., n-1$, onde n é o número de tarefas, o limite de utilização do processador é igual a 1, ou seja, para um conjunto de tarefas que possuam períodos múltiplos, o limite de utilização é 100%. É importante frisar que isto é uma característica do *Taxa Monotônica* que depende de uma condição inerente ao conjunto de tarefas a serem escalonadas e que, portanto, nem sempre pode ser atingido.

3.3.2 Taxa Monotônica com Processos Aperiódicos

Quando além das tarefas periódicas com restrição rígida de tempo, o sistema apresenta também tarefas aperiódicas, torna-se necessário um algoritmo que não apenas garanta o cumprimento dos prazos das tarefas periódicas, mas que também minimize o tempo de resposta das tarefas aperiódicas.

O algoritmo Taxa Monotônica não contempla o tratamento de processos aperiódicos, mas existem algoritmos desenvolvidos para tal fim que utilizam sua política de escalonamento para o despacho dos processos com restrições de tempo. A seguir, apresentam-se alguns destes algoritmos.

Background

Neste algoritmo as tarefas periódicas são tratadas como no algoritmo Taxa Monotônica. As tarefas aperiódicas recebem prioridades baixas, isto é, menores que as prioridades das periódicas. Isto faz com que as tarefas aperiódicas

[2] $\lfloor x \rfloor$ denota o maior inteiro menor ou igual a x. $\lceil x \rceil$ denota o menor inteiro maior ou igual a x.

[3] $\{T_1/T_2\}$ denota $(T_2/T_1) - \lfloor T_2/T_1 \rfloor$, isto é, a parte fracionária de T_2/T_1.

ganhem a UCP somente quando não há nenhuma tarefa periódica em estado de pronto ou pendente, garantindo que aquelas não interfiram no cumprimento dos prazos das periódicas. Por outro lado, implica também em que os tempos de resposta das tarefas aperiódicas sejam ruins, principalmente quando a carga de processamento do sistema for alta.

Polling Server

Assim como o *Background*, este algoritmo concede valores de prioridade aos processos periódicos segundo a política do Taxa Monotônica. Ele cria, no entanto, um novo processo periódico, que também recebe prioridade como os demais processos periódicos do sistema, o qual irá atender às invocações dos processos não periódicos. Trata-se de um processo servidor de tarefas aperiódicas.

Toda vez que o processo servidor é ativado, ele verifica se há algum processo aperiódico pendente e/ou em estado de pronto, concedendo a utilização do processador ao mesmo. Caso ainda haja processos aperiódicos utilizando o processador, quando o processo servidor atinge seu tempo limite de execução, isto é, sua capacidade de serviço, ele irá suspender a execução daqueles até sua próxima ativação. Se, quando o processo servidor, chamado de processo *Polling*, for invocado, não houver nenhum processo aperiódico em estado de pronto e/ou pendente, ele mesmo irá suspender sua execução até seu próximo período, liberando a UCP para os processos periódicos. Assim, o tempo inicialmente alocado para atender os processos aperiódicos será utilizado pelos processos periódicos.

O tempo de resposta dos processos não periódicos depende do período e da capacidade de serviço do processo *Polling*. Apesar de apresentar tempos médios de resposta melhores do que os do algoritmo *Background*, o algoritmo *Polling Server* também não pode oferecer sempre serviço imediato, uma vez que, quando as invocações dos processos aperiódicos ocorrem logo após o processo *Polling* ter sido suspenso, eles terão que esperar até o próximo período do servidor.

Deferrable Server

O algoritmo *Deferrable Server* (*DS*) apresentado na referência [Leh 87] supera as desvantagens encontradas nos dois algoritmos anteriores, pois devido à sua política de atendimento às requisições aperiódicas ele possui um tempo médio de resposta melhor. Algumas das características do algoritmo *Deferrable Server* são comuns ao *Polling Server*. A principal delas é a de que o *Deferrable Server* cria também um processo periódico servidor, que irá tratar os processos aperiódicos. Todos os processos recebem um valor de prioridade, estaticamente, conforme a política de escalonamento do Taxa Monotônica, inclusive o processo servidor.

A diferença básica entre estes dois algoritmos é que o processo servidor do algoritmo *Deferrable Server* retém sua capacidade de serviço (tempo limite de execução originalmente alocado a ele) não "consumida" durante todo o seu período, ao contrário do servidor *Polling* que libera o tempo alocado a ele quando não há tarefa aperiódica pendente. A capacidade do servidor do algoritmo *Deferrable Server* é restabelecida ao seu valor máximo a cada ativação de seu período.

O fato do servidor ser capaz de reter sua capacidade de serviço proporciona ao mesmo a possibilidade de tratar, durante seu período, qualquer processo aperiódico no instante de sua ativação. Isto, desde que não exceda o limite de tempo determinado por sua capacidade. Assim, o algoritmo *Deferrable Server* apresenta um tempo médio de resposta às requisições dos processos aperiódicos bem melhor que o *Polling Server*, uma vez que, quando não há processo aperiódico pendente no instante da ativação do processo servidor, ele não precisa esperar até a próxima instância do servidor para atender os processos aperiódicos.

Devem ser observados dois aspectos importantes associados ao algoritmo *Deferrable Server*: um positivo e um negativo. São eles:

- sua simplicidade de implementação e
- a restrição de não poder ser tratado como uma tarefa periódica para fins de análise da teoria Taxa Monotônica.

Priority Exchange

O algoritmo *Priority Exchange*, proposto na referência [Leh 87] e posteriormente estendido em [Spr 88], difere do *Deferrable Server* apenas na maneira pela qual eles reservam o tempo de execução da tarefa servidora. O algoritmo *Priority Exchange*, *PE*, consiste, basicamente, em preservar a capacidade de serviço da tarefa servidora, trocando-a pelo tempo de execução de uma tarefa periódica de menor prioridade, sempre que não haja nenhuma tarefa aperiódica pendente e/ou em estado de pronto no instante da invocação da tarefa servidora.

Assim como nos dois últimos algoritmos apresentados, o algoritmo *PE*, cria uma tarefa periódica, chamada servidora, para atender as tarefas não periódicas. As prioridades são determinadas como no algoritmo Taxa Monotônica. Uma vez que a tarefa servidora é invocada, ela verifica se há alguma tarefa aperiódica para ser executada. Em caso afirmativo, ela concede o uso da UCP a esta(s) tarefa(s) até o limite de tempo determinado por sua capacidade. Caso contrário, ela concede o uso do processador à tarefa periódica pendente de maior prioridade e efetua a troca de prioridade, isto é, a tarefa periódica executa no nível de prioridade da tarefa servidora e o tempo de execução alocado para a tarefa servidora é acumulado no nível de prioridade da tarefa periódica. Enquan-

to a capacidade de serviço da tarefa servidora não for completamente utilizada, persistirão as trocas de prioridades com os níveis de prioridade inferiores.

No início de cada período da tarefa servidora, ela tem sua capacidade de serviço restaurada ao valor máximo. Quando não há nenhuma tarefa periódica ou aperiódica para ser executada, a capacidade de serviço, ou tempo de execução acumulado, é descartada.

Como o objetivo do algoritmo *PE* é minimizar o tempo médio de resposta às tarefas não periódicas, sempre que houver concorrência à UCP entre uma tarefa periódica e uma aperiódica com mesmo nível de prioridade, deve-se favorecer a não periódica.

Apesar do algoritmo *Deferrable Server* apresentar maior simplicidade e, portanto, maior facilidade de implementação, o algoritmo *Priority Exchange* pode prover, para um mesmo fator de utilização pelas tarefas periódicas, uma capacidade de serviço para a tarefa servidora maior que o *Deferrable Server*. Além disso, Lehoczky, Sha e Strosnider, na referência [Leh 87], mostraram, matematicamente, que o algoritmo *PE* apresenta maior limite de utilização da UCP para o qual o algoritmo Taxa Monotônica garante escalonar um conjunto de tarefas periódicas.

Sporadic Server

A filosofia utilizada pelo algoritmo *Sporadic Server*, também conhecido como *SS*, é idêntica àquela utilizada pelos algoritmos *DS* e *PE*, exceto na forma pela qual o *SS* "reabastece" o tempo de execução do processo servidor. O *Sporadic Server*, proposto na referência [Spr 89], apresenta as vantagens de ser de baixa complexidade de implementação, como o *DS*, e poder ter um servidor com maior capacidade, como o *PE*.

Em vez de restabelecer o tempo de execução da tarefa servidora periodicamente, o algoritmo *SS* só o faz um determinado tempo depois de alguma tarefa aperiódica tê-lo consumido total ou parcialmente.

Para entendermos melhor o método de reposição de tempo de execução do servidor *sporadic*, definiremos os seguintes termos [Spr 89]:

P_S	O nível de prioridade da tarefa que está executando.
P_i	Um dos níveis de prioridade no sistema. Os níveis de prioridade são numerados em ordem decrescente de prioridade, com P_1 sendo o nível de prioridade mais alto, P_2 sendo o segundo e assim por diante.

Ativo	Este termo é usado para descrever o estado de um determinado nível de prioridade com respeito a um período de tempo. Um nível de prioridade é considerado estar *ativo* se a atual prioridade do sistema, P_S, é igual ou maior que a prioridade P_i.
Inativo	É o oposto do termo *ativo*. Isto é, um nível de prioridade P_i é *inativo* se a prioridade atual do sistema P_S é menor que a prioridade de P_i.
tr_i	É o instante no qual o tempo de execução consumido pelo servidor *sporadic* de nível de prioridade P_i será reposto.

A reposição do tempo consumido do servidor, cuja prioridade é P_i, procede da seguinte forma:

> Se a tarefa servidora tem tempo de execução disponível, o instante de reposição tr_i é atribuído quando o nível de prioridade P_i se torna ativo. O valor de tr_i é igual ao instante atual mais o período da tarefa servidora naquela prioridade. Se a capacidade de serviço foi esgotada, o próximo instante de devolução pode ser atribuído quando esta capacidade se tornar maior que zero e P_i estiver ativo.
>
> A reposição de qualquer tempo de execução consumido para a tarefa servidora deverá ocorrer em tr_i, se o nível de prioridade P_i se tornar inativo ou o tempo de execução da tarefa servidora for esgotado. A quantidade a ser reposta é igual à quantidade de tempo de execução da tarefa servidora consumido.

Assim, quando o processo servidor possui o maior nível de prioridade do sistema e possui tempo de execução disponível, o instante de devolução é determinado sempre que uma tarefa aperiódica for executada.

O algoritmo *Sporadic Server* pode ser estendido para o caso de vários processos servidores, cada um dos quais com um valor de prioridade, proporcionando assim níveis de prioridade distintos entre os processos não periódicos.

Quando os processos aperiódicos possuem restrições rígidas de tempo, para garantir o cumprimento de seus prazos, torna-se necessário a criação de várias tarefas servidoras, uma para cada tarefa aperiódica. Neste caso, deve-se impor um tempo mínimo entre invocações de uma mesma tarefa aperiódica, para que se possa certificar que o processo servidor terá tempo de execução suficiente para o cumprimento do seu prazo. O período da tarefa servidora deve ser sempre menor ou igual a este tempo mínimo entre invocações. Sempre que o prazo

de término de cada tarefa não periódica for maior ou igual ao tempo mínimo entre invocações desta mesma tarefa, o algoritmo *Sporadic Server* garantirá o cumprimento destes prazos.

Deadline Monotonic Sporadic Server

No caso do prazo de alguma das tarefas aperiódicas críticas ser menor que tempo mínimo entre invocações, torna-se necessário adotar uma outra política de atribuição de prioridades para as tarefas servidoras. O algoritmo *Deadline Monotonic Sporadic Server*, *DMSS*, apresenta uma solução para esta situação. Ele baseia-se na mesma filosofia do algoritmo *SS*, exceto no que se refere à atribuição de prioridade da tarefa servidora.

No algoritmo *DMSS*, a prioridade do processo servidor é determinada pelo prazo da tarefa aperiódica à qual ele está associado, e não pelo tempo mínimo de invocações dela. Daí o nome *Prazo Monotônico*. O processo servidor deve ser criado com capacidade de serviço igual ao tempo de execução da tarefa que ele está servindo. É importante ressaltar que as prioridades dos processos periódicos devem continuar sendo atribuídas de acordo com o algoritmo Taxa Monotônica, ficando as prioridades dos processos servidores para serem atribuídas como se os seus períodos fossem iguais aos prazos dos processos aperiódicos associados.

3.3.3 *Earliest Deadline*

Todos os algoritmos apresentados até este ponto são classificados como algoritmos estáticos, isto é, atribuem prioridades às tarefas durante a inicialização do sistema, não as alterando durante o decorrer do mesmo. Já o algoritmo *Earliest Deadline*, apresentado por Liu e Layland na referência [Liu 73], para tarefas periódicas, independentes, que possam ser interrompidas e que não necessitam de recursos específicos para executar, ao contrário dos anteriores, é um algoritmo dinâmico e centralizado. A idéia do *Earliest Deadline* é atribuir sempre o maior valor de prioridade à tarefa que estiver em estado de pronto e apresentar o prazo mais próximo a expirar. A recíproca é verdadeira, isto é, a tarefa com o prazo mais distante fica com a menor prioridade. Em outros palavras, o algoritmo *Earliest Deadline* executa sempre a tarefa que possui o menor prazo para terminar de executar. As novas atribuições de prioridade só ocorrem no instante de uma nova invocação de uma das tarefas do sistema, tendo em vista que só assim se pode alterar os níveis de prioridades entre elas.

Em seu artigo, Liu e Layland [Liu 73] mostraram que uma condição necessária e suficiente para a viabilidade da aplicação do algoritmo *Earliest Deadline* para processos periódicos e independentes, em um sistema monoprocessador, é que o seu fator de utilização da UCP seja menor ou igual a 1, isto

é:

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1 \tag{3.4}$$

onde

C_i = tempo de execução da tarefa T_i
P_i = período da tarefa T_i
n = número de tarefas

Isto mostra que o algoritmo *Earliest Deadline* é ótimo, uma vez que o seu limite do fator de utilização da UCP é 100%, ou seja, ele propicia uma completa utilização do processador.

3.3.4 *Least Laxity First*

O algoritmo *Least Laxity First* é também um algoritmo de escalonamento de preempção por prioridade, dinâmico e centralizado para processos periódicos, sem restrições de precedência ou comunicação (independentes) e que possam ter sua execução interrompida. Como nos demais algoritmos apresentados, ele sempre concede o uso da UCP ao processo de maior prioridade em estado de pronto. O que caracteriza o algoritmo *Least Laxity First* é como ele determina as prioridades de cada processo. Ele atribui as prioridades de forma inversa às folgas de cada processo, isto é, o processo de menor folga recebe a maior prioridade e o de maior folga recebe a menor. Entende-se por folga de um processo o tempo máximo que o processo pode " abster-se " da utilização do processador durante uma instância sem comprometer o cumprimento de seu prazo. No caso do processo começar sua execução no instante atual e não sofrer suspensão durante a mesma, sua folga será o montante de tempo entre o término de sua execução e o seu prazo limite de execução. Formalmente, podemos escrever:

$$F_i(t) = P_i - (t + Tr_i(t)) \tag{3.5}$$

onde

$F_i(t)$ = folga da tarefa T_i no instante t,
P_i = prazo absoluto da tarefa T_i,
t = instante atual e
$Tr_i(t)$ = tempo restante no instante t para a tarefa T_i terminar sua execução.

É importante notar que, apesar da folga do processo que está e-

xecutando manter-se constante, as folgas dos demais processos decrescem. Isto implica na necessidade de um monitoramento a cada unidade de tempo, pois a qualquer instante um processo que está aguardando sua vez pode passar a ter menor folga do que o processo que está executando, logo, maior prioridade; o que pode representar um ponto negativo ao algoritmo sob o ponto de vista de sua implementação.

Assim como o *Earliest Deadline*, o algoritmo *Least Laxity First* é um algoritmo ótimo, no entanto, eles não apresentam a mesma seqüência de execução das tarefas.

3.4 Avaliação dos Algoritmos para a Aplicação de Análise de Rede

A definição de qual algoritmo de escalonamento utilizar depende intimamente do ambiente do sistema e das características das tarefas que irão ser escalonadas por ele. A proposta deste trabalho é a implementação de um algoritmo escalonador em tempo real para as funções de análise de rede elétrica em um Centro de Controle com um ambiente computacional constituído de estações de trabalho, executando o sistema operacional UNIX. Considera-se que seja alocada uma estação desta rede para o subsistema de análise de rede elétrica, parte do modelo do Centro de Controle apresentado no Capítulo 2, que é responsável por executar as funções de análise de rede. Para tal aplicação, como as funções de análise de rede são aplicativos que não realizam comunicação e podem ter sua execução interrompida a qualquer instante, necessita-se apenas de um escalonador centralizado que seja de prioridade dirigida e capaz de contemplar as preempções dos aplicativos de rede sob seu controle, conforme já discutido nas seções anteriores deste capítulo.

Como as funções de análise de rede podem tanto ser executadas de forma periódica (modo Tempo Real) como também por solicitação do operador (modo Estudo), é importante que o escalonador a ser adotado seja capaz de atender às requisições esporádicas das tarefas que compõem este modo, além das requisições periódicas que lhe serão feitas pelas tarefas com prazos. Atender às requisições aperiódicas significa executar tais tarefas, objetivando minimizar o tempo de resposta ao seu atendimento, sem comprometer os prazos das tarefas periódicas.

Isto faz com que o algoritmo Taxa Monotônica, em sua proposta inicial [Liu 73], comprovadamente ótimo dentro de sua classe de algoritmo, não se aplique ao caso das funções de análise de rede, uma vez que entre as considerações feitas para definição de seu ambiente está a de que todas as tarefas possuam apenas requisições periódicas.

As duas primeiras técnicas apresentadas neste capítulo para satis-

fazer a necessidade de se garantirem os prazos das tarefas periódicas críticas e de se atenderem às requisições aperiódicas foram o *Background* e o *Polling Server*. A primeira, por apenas tratar as tarefas aperiódicas, quando não há nenhuma tarefa periódica em estado de pronto, não propicia um bom tempo de resposta àquelas. A segunda, que já introduz o conceito de tarefa servidora, incorpora o tratamento das tarefas aperiódicas ao algoritmo Taxa Monotônica. O principal problema com a técnica *polling* é a incompatibilidade entre a natureza aleatória das tarefas aperiódicas com a natureza periódica do servidor *Polling*: Quando o servidor estiver pronto pode não haver nenhuma tarefa aperiódica a ser atendida, o que o faria liberar o processador às tarefas periódicas e se suspender. Caso aconteça requisição por parte de tarefas aperiódicas logo após este instante, a tarefa aperiódica deverá esperar até o começo do próximo período da tarefa servidora ou aguardar o instante em que o processador se torne ocioso. Apesar de representar uma melhoria em relação ao *Background*, o *Polling Server* ainda pode apresentar média de espera por atendimento e tempo de resposta altos.

Os algoritmos *Priority Exchange* (*PE*) e Deferrable Server (*DS*), introduzidos por Lehoczky *et alii* [Leh 87], superam os obstáculos associados ao atendimento de tarefas aperiódicas pelo servidor *Polling* e pelo *Background*. Apesar de criarem uma tarefa periódica, geralmente de alta prioridade, para o atendimento das requisições aperiódicas como o faz o servidor *Polling*, os algoritmos *PE* e *DS* preservam o tempo de execução alocado para o serviço aperiódico se, uma vez invocada a tarefa periódica, não há tarefas aperiódicas pendentes. Estes algoritmos podem alcançar melhores tempos de resposta para as requisições aperiódicas por causa de sua habilidade de providenciar atendimento imediato às tarefas aperiódicas.

Os algoritmos *PE* e *DS* diferem em sua complexidade e em seu efeito sobre o limite de escalonabilidade para as tarefas periódicas. O algoritmo *DS* é um algoritmo muito mais simples de ser implementado que o algoritmo *PE*, porque o *DS* sempre mantém seu tempo de execução em seu nível de prioridade original, isto é, nunca permuta seu tempo de execução com níveis de prioridades mais baixos como o faz o algoritmo *PE*. O custo desta redução de complexidade em relação ao *PE* é um decréscimo no limite de escalonabilidade de tarefas periódicas, para o pior caso, para um mesmo tamanho do servidor de tarefas aperiódicas.

Ambos, *PE* e *DS*, requerem que seja reservada uma certa utilização de recursos para o atendimento de tarefas aperiódicas com alta prioridade. A esta utilização chama-se de tamanho do servidor, U_s, o qual é a relação entre o tempo de execução do servidor com o seu período. O tamanho e o tipo (*PE* ou *DS*) do servidor determinam o limite de escalonabilidade para as tarefas periódicas, U_p, que é a maior utilização por tarefas periódicas que o algoritmo Taxa Monotônica pode sempre garantir seu escalonamento. Na referência [Leh 87], foram desenvolvidas as equações que relacionam U_p com U_s. Quando o número de tarefas periódicas se aproxima de infinito (pior caso), temos:

$$U_p^{PE} = \ln \frac{2}{U_s^{PE} + 1} \tag{3.6}$$

$$U_p^{DS} = \ln \frac{U_s^{DS} + 2}{2U_s^{DS} + 1} \tag{3.7}$$

A partir das equações 3.6 e 3.7 pode-se concluir que, para uma dada carga periódica, o tamanho do servidor, U_s, é menor para o algoritmo *DS* do que para o *PE*, o que vem a confirmar a desvantagem do *DS* em relação ao *PE*, já mencionada anteriormente.

O algoritmo proposto na referência [Spr 89], o *SS*, apresenta tempo de resposta às requisições periódicas comparável aos algoritmos *DS* e *PE*, a pequena complexidade de implementação do *DS* e servidor do tamanho do servidor do *PE*. Além disso, ao contrário dos algoritmos anteriores, ele é capaz de garantir o cumprimento de prazos de tarefas aperiódicas críticas, desde que estes sejam maiores ou iguais ao seu menor intervalo de tempo entre duas ativações. Caso esta condição não seja observada, é necessário que se utilize o *DMSS* para garantir o escalonamento de tais tarefas.

Na aplicação de análise de rede, as tarefas que compõem o modo Estudo, apesar de serem aperiódicas, são não críticas. Isto significa que não há necessidade de se basear a determinação da prioridade das tarefas aperiódicas em seus prazos, como é feito no *DMSS*.

Isto sugere que, com relação aos algoritmos estáticos, a aplicação dos algoritmos *Sporadic Server* e *Deadline Monotonic Sporadic Server* atende às necessidades impostas pelas características das funções de análise de rede em tempo real e pela configuração do ambiente computacional.

Na classe de algoritmos de escalonamento dinâmico, foram apresentados dois algoritmos. Tanto o *Earliest Deadline* como o *Least Laxity First*, assim como o Taxa Monotônica, são ótimos dentro de sua classe de algoritmo. Mostra-se que, apesar de não possuírem a mesma ordem de execução das tarefas, eles podem proporcionar uma completa utilização da UCP, isto é, U = 1. Um ponto negativo do algoritmo *Least Laxity First*, do ponto de vista de sua implementação, é o *overhead* que ele impõe devido à necessidade de monitoramento, a cada unidade de tempo, das prioridades das tarefas do sistema, uma vez que mudanças entre elas podem ocorrer a qualquer instante.

Uma vez que o menor limite superior do fator de utilização para o algoritmo *Earliest Deadline* é sempre maior ou igual ao menor limite superior do algoritmo Taxa Monotônica, aquele sempre conseguirá cumprir os prazos das tarefas que o Taxa Monotônica também garantir. A recíproca não é verdadeira, isto é, o Taxa Monotônica pode não garantir as restrições de tempo de tarefas que são garantidas pelo *Earliest Deadline*. No entanto, o fato do *Earliest*

Deadline determinar e atribuir novas prioridades às tarefas em tempo de execução e realizar novas suspensões, pode embutir, devido aos cálculos e aos chaveamentos de contexto, um tempo extra a ser considerado principalmente em momentos de sobrecarga do sistema.

A principal vantagem do Taxa Monotônica é certamente sua *extensibilidade*, pois ele permite um grande número de extensões do tradicional escalonamento de processos periódicos e independentes. Além de poder prover meios convenientes para oferecer rápido atendimento a tarefas aperiódicas críticas ou não, conforme mostrado neste capítulo, o Taxa Monotônica possui outras extensões, como [Sha 91]:

- pode ser usado para assegurar o cumprimento de restrições de tempo das tarefas mais importantes do sistema, mesmo em condições de sobrecarga transitória do mesmo, através do procedimento de transformação de período [Sha 86];
- pode ser modificado para tratar exigências de sincronização de tarefas, contornando problemas de inversão de prioridade e de bloqueio mútuo de tarefas, usando os protocolos de herança de prioridade [Sha 87] e de prioridade topo [Goo 88] [Raj 89];
- pode ser convenientemente usado para escalonar tarefas onde *computação imprecisa* é permitida [Liu 87] [Chu 90] e
- pode ainda permitir a inclusão e exclusão de tarefas durante a execução do sistema e a alteração de parâmetros de tarefas, usando-se o *Mode Change Protocol* [Sha 89].

Baseado nestas considerações e na adequação de seus ambientes ao existente no caso das funções de análise de rede elétrica, optou-se pela implementação dos algoritmos *Sporadic Server* e *Earliest Deadline*, que, a partir de agora, serão chamados de Servidor Esporádico e Próximo Prazo, para realizar o escalonamento de tais funções. No próximo capítulo, são descritas as soluções de implementação destes algoritmos em um ambiente computacional executando o sistema operacional UNIX.

Capítulo 4

Modelos para Implementação do Escalonador das Funções de Análise de Rede Elétrica

Idealmente, os centros de controle deveriam ser dotados de um ambiente computacional que executasse sistemas operacionais que suportassem aplicações de tempo real, isto é, sistemas que possuíssem em seu núcleo um gerenciador ou escalonador de tarefas de tempo real. Assim, a aplicação das funções de análise de rede elétrica em tempo real poderia ser realizada de forma imediata, necessitando-se apenas de um gerenciador para a base de dados.

Na prática, a tendência dos modernos *EMS* é de possuírem arquiteturas de comunicações abertas que permitam que novos *hardwares* sejam conectados e substituídos. Além disso, tais arquiteturas proporcionariam uma maior distribuição das funções dos diversos subsistemas entre os componentes da configuração de computadores e um maior uso de *LANs* (*Local Area Networks*). Neste aspecto, as redes locais de estações de trabalho com sistemas operacionais multiusuários e multitarefas, conectadas por redes *Ethernet* usando protocolos de comunicação *TCP/IP* (*Transmission Control Protocol / Internet Protocol*), têm ganho muito espaço nos modernos centros de controle.

Como normalmente os sistemas operacionais destas estações, disponíveis comercialmente, não possuem núcleo de tempo real, há uma forte motivação para a implementação de um algoritmo de escalonamento em tempo real para as estações de trabalho que gerencie as funções de análise de rede.

Neste capítulo são apresentados dois modelos de implementação de escalonadores de tempo real para o sistema operacional UNIX: o modelo Taxa Monotônica (TM) com Servidor Esporádico e o Próximo Prazo (PP). A implementação foi realizada para os sistemas *SunOS*[1] *Release 4.1.1* e *SunOS Release*

[1] *SunOS* é uma marca registrada da *Sun Microsystems, Inc.*

5.4 (Solaris). Este é compatível com a versão *System V* da AT&T, enquanto, o primeiro, corresponde à uma versão do *4.3BSD* de Berkeley com algumas características da versão *System V*. O código da aplicação foi desenvolvido em linguagem C de alto nível, fazendo-se uso de chamadas do sistema operacional.

4.1 Estratificação das Funções

A implementação do algoritmo escalonador é feita conforme a de um aplicativo, isto é, para o núcleo do sistema operacional o escalonador se comporta como um processo qualquer do sistema. Todas as funções de tempo real são, no entanto, gerenciadas pelo escalonador, antes de serem, efetivamente, ativadas pelo núcleo do sistema operacional. É o escalonador que, além de controlar os instantes de ativação das funções de análise de rede em tempo real, faz as requisições de execução de todas elas ao sistema operacional, determina a de maior prioridade dentre as que estão em estado de pronto e as suspende ou coloca em estado de pronto novamente para atender a uma requisição do operador do sistema. Assim, o escalonador funciona como uma camada a mais entre os aplicativos de rede e o núcleo do sistema operacional, escalonando qual e quando cada função deva ser disparada por este para ser executada pelo processador. A Figura 4.1 ilustra a idéia de camadas.

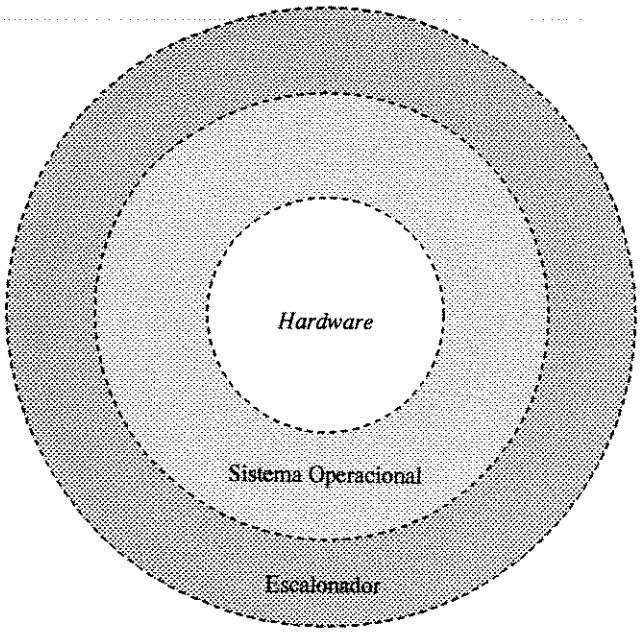

Figura 4.1: Camadas do Sistema

4.2 Modelo Taxa Monotônica com Servidor Esporádico

A Figura 4.2 mostra como são divididas as funções de escalonamento e atendimento de requisições do modo Estudo para o modelo Taxa Monotônica com Servidor Esporádico. O nível 1 é a camada de escalonamento, isto é, possui a função de gerenciar o uso do processador entre os aplicativos periódicos, concedendo sempre o uso do mesmo para o que estiver em estado de pronto de maior prioridade segundo o algoritmo Taxa Monotônica. Neste nível também se enquadra a função de temporização que é responsável por toda a administração de tempo necessária à realização do sistema em tempo real, isto é, este módulo é responsável por realizar a função de *clock* do sistema de tempo real.

O segundo nível é o dos aplicativos periódicos. Cada uma das $n-1$ tarefas τ_j, $j \neq i$, corresponde a um aplicativo de análise de rede do modo Tempo Real. A *i-ésima* tarefa, τ_i, é a tarefa servidora de processos não periódicos, isto é, o servidor esporádico. A existência de um único servidor para todas as tarefas do modo Estudo é suficiente, pois todas elas possuem os mesmos requisitos. O último nível é a camada das tarefas aperiódicas, que correspondem aos aplicativos de análise de rede do modo Estudo.

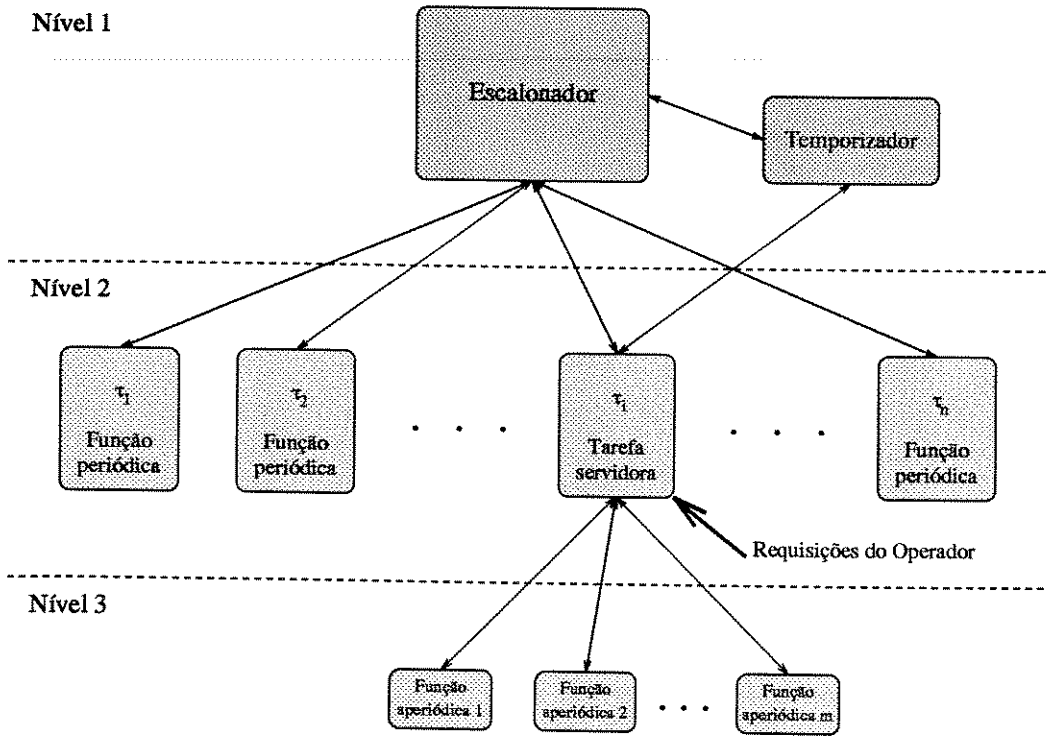

Figura 4.2: Níveis de Aplicativos do Modelo Taxa Monotônica

É importante notar que, para o escalonador, há transparência de

qual, entre as tarefas periódicas, é a tarefa servidora. Esta funciona como se fosse apenas mais uma dentre as funções do modo Tempo Real. Assim, é possível que se altere a política de atendimento às requisições aperiódicas do operador do sistema, substituindo-se a tarefa servidora esporádica por qualquer outra das políticas de atendimento apresentadas no Capítulo 3, sem que se necessite proceder profundas alterações no algoritmo escalonador. Tal procedimento está em concordância com os conceitos de escalonamento do modelo Taxa Monotônica, uma vez que o algoritmo Taxa Monotônica é um escalonador estático de funções periódicas e independentes e que o atendimento às tarefas aperiódicas deve ser realizado através de alguma de suas extensões.

A Figura 4.3 mostra um algoritmo simplificado do modelo do escalonador proposto. Apesar dos módulos escalonador, servidor e temporizador se constituírem de processos distintos, eles possuem relações de dependência, uma vez que, além de se comunicarem, realizam operações de sincronização. Na verdade, mais do que isto, eles possuem relação de descendência, como é mostrado na Figura 4.3. Desta forma, apenas o escalonador é ativado, criando ele próprio o servidor, que por sua vez cria o temporizador. Nas próximas seções serão discutidos cada um dos algoritmos do escalonador, do servidor e do temporizador de forma detalhada.

4.2.1 Módulo Escalonador

A ativação do escalonador deve ser feita através do *console* da estação de trabalho. Uma vez ativado, o escalonador dá início ao procedimento de Inicialização.

Inicialização

Após a definição das variáveis, o escalonador redefine as ações associadas aos sinais do sistema operacional existentes que serão utilizados por ele e pelos processos servidor e temporizador como forma de notificação entre os mesmos de que algum evento ocorreu. Os sinais, às vezes chamados de " interrupções de *software*", suportados pelo sistema operacional UNIX, geralmente ocorrem de forma assíncrona (isto é, o processo receptor não sabe, a *priori*, quando ele irá receber o sinal) [Ste 90]. No âmbito da implementação, o mecanismo de sinais consiste em se definir e elaborar o código para determinar a ação a tomar quando um sinal é recebido. O processo receptor deve executar a ação padrão referente ao sinal, ignorá-lo ou pegá-lo e executar uma sub-rotina que deve ser associada a ele. A princípio, o processo receptor não sabe qual processo envia o sinal. No apêndice A é detalhado o mecanismo de sinais no sistema operacional UNIX.

Para realizar todas as funções de notificação de eventos necessárias ao escalonador tornou-se preciso redefinir 5 (cinco) sinais do sistema operacional.

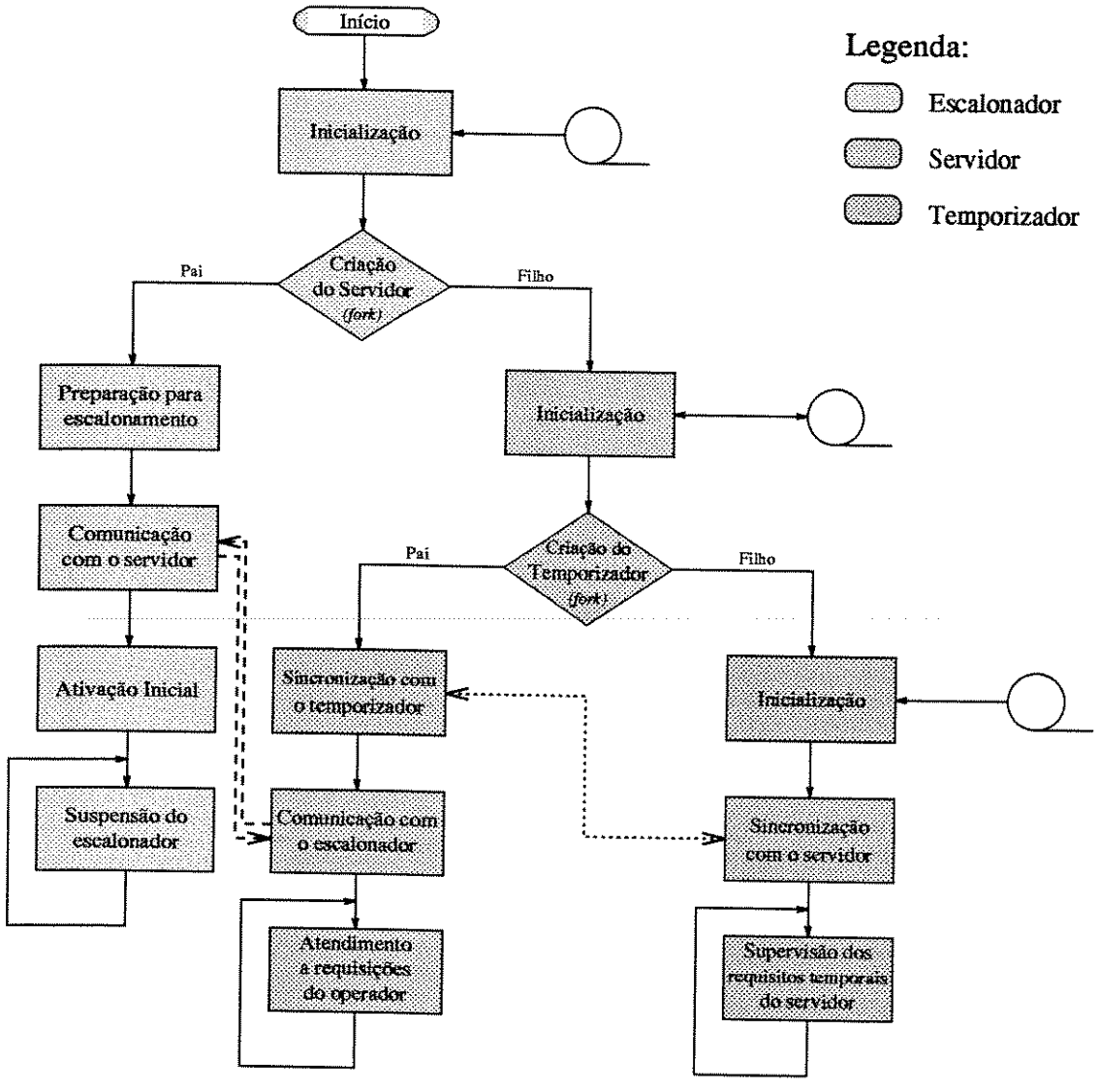

Figura 4.3: Algoritmo do Modelo de Implementação do TM com SE

As funções assíncronas realizadas pelo escalonador, para as quais são tratados os sinais, são o reordenamento da fila de pronto[2], o atendimento à requisição do operador feita através da servidora, o bloqueio/desbloqueio da tarefa servidora e o fim dos aplicativos de análise de rede. Como o UNIX, no caso do *SunOS 4.1.1*, reserva apenas 2 (dois) sinais para serem definidos pelo usuário, houve a necessidade de se tratar outros 3 (três) sinais, redefinindo suas ações padrão, sem que, no entanto, se alterasse o desempenho do sistema operacional. Para isto, usaram-se dois sinais cuja ação padrão é a de ser descartada e um segundo, SIGALRM, teve sua ação reaproveitada para a função de reordenamento. No final desta seção são descritas as quatro funções assíncronas realizadas pelo escalonador.

Como muitas vezes a UCP pode ser disputada entre as funções assíncronas de um mesmo módulo, torna-se necessário prover-se uma forma de garantir a exclusão mútua às regiões críticas destas partes dos processos. Exclusão mútua é uma técnica de definição de um código de entrada e saída de forma que, em determinado instante, somente um único processo esteja acessando o trecho do código denominado região crítica. A região crítica de um processo pode ser entendida como um trecho do código que acessa um recurso compartilhado e/ou que necessita ser protegido com relação a outros processos concorrentes [Ben 90]. Assim sendo, durante a redefinição das ações relativas ao recebimento dos determinados sinais, para a solução dos problemas decorrentes da existência de regiões críticas, torna-se necessário estipular que, ao se receber qualquer um dos sinais, antes mesmo de se dar início ao seu tratamento, os recebimentos de todos os outros sinais devem ser bloqueados até o término do tratamento do primeiro. No UNIX, isto pode ser obtido através da chamada de sistema **sigaction**, conforme apresentado no apêndice A.

Isto feito, o escalonador inicia o passo seguinte: a criação do semáforo que será usado para a sincronização entre os módulos servidor e temporizador. Semáforos são variáveis protegidas cujos valores somente podem ser acessados e alterados por operações indivisíveis. No apêndice B é apresentado o conceito de semáforo e sua aplicação como solução de problemas decorrentes da sincronização e comunicação entre processos concorrentes.

A criação de semáforos é feita através da chamada de sistema **semget**. Para que os outros módulos (servidor e temporizador) também acessem este semáforo criados pelo escalonador, eles devem abri-lo usando a chamada de sistema **semop**. A utilização de semáforos no UNIX, muitas vezes, não é, tão simples, como o são as operações P e V sobre semáforos criadas por Dijkstra [Dij 68], no entanto, é possível desenvolver rotinas, utilizando as chamadas de sistema do UNIX, de forma a tornar as operações com os semáforos de fácil manuseio [Ste 90].

O terceiro passo é criar o canal de comunicação necessário para a troca de informações entre os três módulos, o que é feito através da chamada

[2]Fila de pronto é a lista de processos que já passaram por seu instante de ativação e aguardam a liberação do processador para iniciar ou reiniciar sua execução

de sistema **msgget** [Sun 90c]. Isto é necessário porque os módulos são processos distintos, isto é, cada um possui um identificador próprio o que inviabiliza que se realize a comunicação entre os mesmos através de variáveis globais ou passagem de parâmetros. Para que o canal de comunicação sirva realmente para a troca de informação entre os processos é necessário que os outros módulos também concordem com a comunicação, abrindo o mesmo canal durante a sua execução, conforme será visto adiante. No apêndice C é apresentada uma breve descrição do que é e quais são os métodos providos pelo UNIX para comunicação entre processos.

Em seguida, o módulo escalonador inicializa as variáveis e faz a leitura de um arquivo que contém as informações sobre os aplicativos de análise de rede que constituem o modo de Tempo Real. Este arquivo de processos periódicos deve conter as informações de tempo de execução e período de cada uma das funções de análise de rede que devem ser escalonadas periodicamente. Considera-se que o tempo de pronto das funções de análise de rede é igual ao tempo de chegada das mesmas, isto é, o instante mais cedo que uma função pode iniciar sua execução é o próprio instante em que ela é invocada.

A informação de tempo de execução das funções que o arquivo de processos periódicos deve conter é a do tempo máximo que as respectivas funções podem levar para executar. No cômputo deste parâmetro, deveriam ser levados em consideração os *overheads* introduzidos pela camada escalonadora, como, por exemplo, o tempo de processamento utilizado pelos módulos escalonador, servidor e temporizador, e os *overheads* introduzidos pelo núcleo do sistema operacional, como, por exemplo, o tempo de chaveamento de contexto devido à preempção de funções. Diante dos tempos de execução das funções de análise de rede do modo Tempo Real, no entanto, estes *overheads* tornam-se desprezíveis. O pior tempo de execução dos aplicativos de análise de rede em modo *off-line* já é uma ótima aproximação, conforme será mostrado no próximo capítulo.

O período das funções diz respeito à freqüência com que elas devem ser executadas. Os prazos das funções, isto é, a quantidade de tempo que elas têm para terminar de executar, são considerados serem iguais aos seus respectivos períodos. Tanto o tempo de execução quanto o período das funções devem ser fornecidos, no referido arquivo, em segundos.

Por fim, o arquivo de processos periódicos deve conter também a informação do período da tarefa servidora. Conforme visto no Capítulo 3, o período do servidor esporádico é usado para o cálculo do instante de reposição da "carga" do mesmo. Neste passo, o módulo escalonador já é capaz de determinar se o conjunto de funções apresentadas é escalonável ou não pelo algoritmo Taxa Monotônica.

Com a posse das informações referentes às funções a serem escalonadas, o escalonador já pode calcular a capacidade de atendimento da tarefa servidora. A capacidade de atendimento é a máxima quantidade de tempo em

que o processador pode ser concedido à tarefa servidora por período da mesma, sem afetar o prazo das tarefas periódicas.

O último passo do procedimento Inicialização é a determinação das prioridades das funções apresentadas. Isto é, uma vez que já se possui a informação dos períodos das funções de análise de rede, já se é possível determinar, a partir da política do Taxa Monotônica, qual a prioridade de cada função dentro do sistema. Conforme dito anteriormente, considera-se que todas as funções podem iniciar sua execução já a partir do instante t_I de inicialização do escalonador. Uma vez que a primeira instância de todas as funções estará se referindo aos dados de tempo real do mesmo instante t_I, é desejável que a ordem de execução das mesmas seja conforme a seqüência lógica apresentada na Figura 2.2, o que é obtido com o Taxa Monotônica, pois as primeiras funções devem, obviamente, possuir períodos menores.

Para o caso particular de se possuir duas ou mais funções com o mesmo período, adotou-se, sem perda da generalidade do algoritmo Taxa Monotônica, o seguinte critério: caso haja empate de ativação entre duas ou mais funções de análise de rede de mesmo período, que se refiram aos dados de tempo real de um mesmo instante t_i, a primeira a ser executada será a primeira especificada dentro do arquivo de processos periódicos; bastando, assim, que se especifique dentro do arquivo de processos periódicos, as funções na ordem de sua seqüência lógica, tal e qual se encontram na Figura 2.2.

Na Figura 4.4 é mostrado um algoritmo do procedimento Inicialização, referente à Figura 4.3.

Criação do Servidor

Neste ponto de sua execução, o escalonador cria um novo processo que concorrerá com ele próprio à UCP, o qual executará o módulo servidor das requisições do modo Estudo.

A única forma na qual um novo processo é criado pelo UNIX, exceto quanto ao processo init que é ativado especialmente pelo núcleo, quando o UNIX é inicializado [Bac 86], é por um processo já existente executar a chamada de sistema fork [Sun 90b].

Quando o escalonador invoca a chamada fork, é criada uma cópia de seu código a partir do ponto onde a chamada ocorreu. O processo que executou a chamada, no caso o próprio escalonador, é chamado de "processo-pai", enquanto o novo processo gerado recebe o nome de "processo-filho". Apesar de ser invocada uma única vez, a chamada de sistema fork retorna o identificador do novo processo criado ao processo-pai e retorna o valor zero no processo-filho, que pode obter o identificador do processo-pai através da chamada de sistema getppid [Sun 90b].

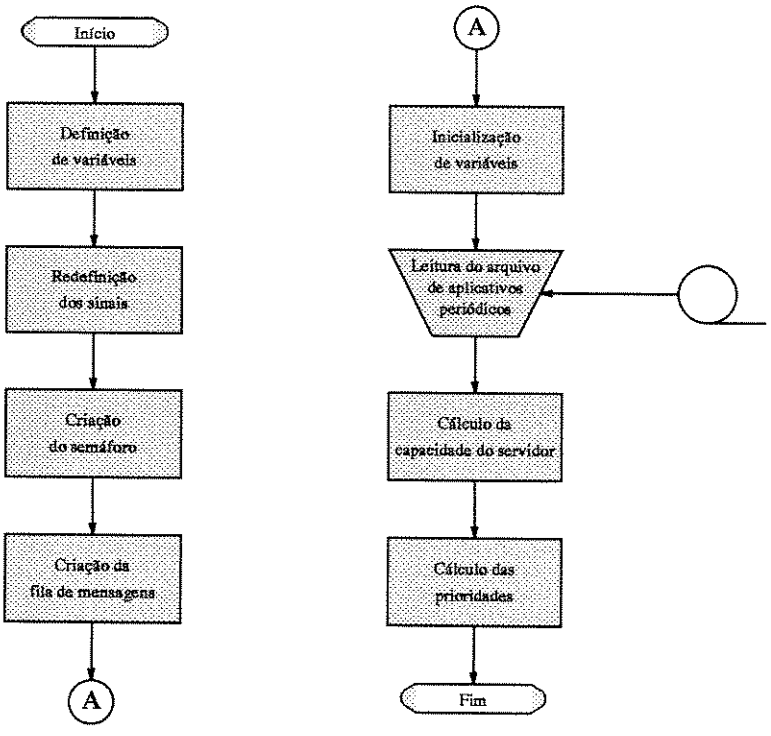

Figura 4.4: Inicialização do Escalonador TM

Assim, para executar o módulo servidor, o escalonador realiza a instrução **fork** para fazer uma cópia de si mesmo e então o processo-filho invoca a chamada de sistema **exec** [Sun 90b], que é o único modo no qual um programa é executado pelo UNIX. A chamada de sistema **exec** substitui o processo atual, chamado de "processo chamativo", pelo "novo programa", não alterando o identificador do processo.

Preparação para Escalonamento

A primeira parte da preparação para escalonamento é executar uma rotina que seja responsável pelo ordenamento das funções de análise de rede dentro da fila de processos prontos. A única tarefa a realizar-se é verificar quais são os processos que estão em estado de pronto no instante específico e qual é a prioridade de cada um, gerando a fila em ordem decrescente da mesma, isto é, colocando-se à frente da fila as funções de menor período.

No caso do instante de inicialização do módulo escalonador, é feita a consideração de que todas as funções estarão em estado de pronto, mas esta rotina de ordenamento será chamada também por algumas das funções assíncronas (funções tratadoras de sinais) do escalonador, tornando-se, por isso, necessário verificar quais aplicativos estão em estado de pronto no determinado instante.

A outra parte da preparação consiste no cálculo do vencimento dos prazos das funções de análise de rede. O escalonador realiza esta operação porque, após a invocação das funções, ele precisará saber qual o próximo ponto de escalonamento do sistema, para, então, solicitar ao núcleo do sistema operacional a notificação da chegada deste instante para o reordenamento da fila de pronto.

Comunicação com o Servidor

Para que o módulo escalonador possa notificar aos módulos servidor e temporizador a ocorrência de determinados eventos e vice-versa, é preciso que cada um deles conheça o identificador de processo dos outros. O identificador de um processo é um inteiro positivo que é assinalado pelo núcleo do sistema operacional sempre que um novo processo é criado. Este identificador é único para cada processo do sistema.

Para realizar esta troca de informação, os módulos escalonador e servidor usam o canal de comunicação criado no procedimento Inicialização do primeiro. Nesta fase, o módulo servidor, além de já haver aberto a mesma fila de mensagens criada pelo escalonador, já deve também ter criado o módulo temporizador, para que possa informar o identificador deste ao escalonador. O identificador do temporizador é retornado ao módulo servidor quando este invoca a chamada de sistema fork para criá-lo.

A forma na qual o mecanismo de filas de mensagens para comunicação entre processos no UNIX transmite os identificadores de um módulo a outro é descrito com maiores detalhes no apêndice C. Após a operação de comunicação com o módulo servidor, o escalonador pode remover a fila de mensagem, pois ela não mais será necessária.

Ativação Inicial

Neste ponto, o escalonador já está apto a iniciar o escalonamento das funções de análise de rede. Conforme mencionado no início desta seção, considera-se que o tempo de ativação de todas as funções, isto é, o instante a partir do qual estão todas prontas para execução, é o próprio instante de inicialização do escalonador. Assim, o que o escalonador tem a fazer é consultar a fila de pronto por ele criada, que contém todas as funções em estado de pronto em ordem de prioridade, e ativar aquela que se encontra em primeiro lugar na fila. Como o módulo escalonador não pode invocar um novo aplicativo e esperar o término de sua execução, pois precisa continuar monitorando os pontos de escalonamento e as requisições de execução de aplicativos do modo Estudo feitas através tarefa servidora, ele primeiro cria um processo-filho (através de uma chamada de sistema fork), que, então, executará a função de análise de rede (através da chamada de sistema exec) e depois continua sua execução normal.

Porém, antes da ativação inicial, a política de escalonamento com servidor esporádico de tarefas aperiódicas requer que o escalonador verifique se o nível de prioridade da tarefa que será executada é maior ou igual ao nível de prioridade da tarefa servidora. Caso seja, é necessário que o escalonador notifique ao módulo temporizador para que ele faça a previsão do primeiro instante de reposição tr_i do tempo de execução consumido pela tarefa servidora, o que é feito através do envio de um sinal, que é tratado no módulo servidor.

Por fim, o módulo escalonador precisa invocar a chamada de sistema **alarm** [Sun 90b] para marcar um envio de sinal do núcleo do sistema operacional, que o notifique da ocorrência do instante de um novo ponto de escalonamento. Quando o escalonador receber este sinal, ele deve tratá-lo executando a função assíncrona de reordenamento da fila de pronto, que será descrita adiante.

Suspensão do Escalonador

Uma vez realizadas todas as operações de inicialização, criação e comunicação com o servidor, disparo inicial das funções do modo Tempo Real, etc, tudo o que o escalonador tem a fazer é aguardar a notificação de eventos tais como o fim da execução de algum aplicativo que detinha o uso do processador, a ocorrência do instante de um novo ponto de escalonamento, a requisição de atendimento a uma solicitação do operador ou mesmo o fim do atendimento à mesma. Porém, é preciso que, além do pronto atendimento a estes eventos, isto seja feito, respeitando-se outros dois princípios básicos:

- a coexistência com o sistema de tempo real, durante toda a sua duração e
- a não exigência do uso dos recursos de processamento do sistema, que devem ser reservados, basicamente, à utilização por parte dos processos de tempo real e das solicitações do modo Estudo.

Um laço infinito contendo uma única instrução, a chamada de sistema **pause** [Sun 90b], pode desempenhar de forma satisfatória todos os três princípios concomitantemente. Isto porque a chamada **pause** suspende o processo que a invocou, conservando o processo na memória do computador sem que, no entanto, ele utilize o processador, até que um sinal vindo de uma instrução **kill** ou do núcleo seja recebido. Ao receber um sinal, a instrução de **pause** retorna, recolocando o módulo escalonador apto a tratá-lo. Ao retornar do tratador do sinal, o escalonador volta ao laço infinito e, portanto, à chamada **pause**.

Funções Assíncronas do Escalonador

Durante a inicialização, o escalonador declara o procedimento a ser tomado quando da ocorrência de determinados eventos. As execuções destes

procedimentos são realizadas quando o escalonador é notificado da ocorrência de algum evento através do recebimento de sinais. Como os sinais são enviados por outros processos, isto pode ocorrer a qualquer instante da execução do escalonador. É importante lembrar que estes procedimentos são realizados como funções atômicas, isto é, indivisíveis, garantindo-se, assim, que um procedimento não é interrompido pelo recebimento de outra notificação. A seguir são descritas cada uma das funções assíncronas.

A função de reordenamento da fila de pronto

A função de reordenamento da fila de pronto é invocada sempre que o sinal SIGALRM é recebido pelo escalonador. Este sinal é encomendado pelo próprio escalonador, que calcula o próximo ponto de escalonamento e requer ao núcleo do sistema operacional o envio do mesmo através da chamada de sistema **alarm** [Sun 90b]. O argumento desta chamada de sistema é o tempo em segundos, a partir do instante da requisição, em que o processo que invocou deseja receber o sinal SIGALRM.

Uma vez recebido o sinal, o escalonador procura o aplicativo, ou os aplicativos, de análise de rede que esteja(m) vencendo seu(s) prazo(s) e recebendo um novo e o(s) coloca em estado de pronto. O próximo passo é recalcular a ordem de escalonamento, colocando em ordem de prioridade as tarefas que estão em estado de pronto. Isto feito, caso haja algum aplicativo em estado de pronto, o escalonador verifica se o que está executando no momento é o de maior prioridade. Isto, porque, caso o aplicativo de maior prioridade em estado de pronto não seja o que detém a "posse" do processador, torna-se necessário suspendê-lo. Se a tarefa que deva sofrer a suspensão é a tarefa servidora, então o escalonador deve notificar ao módulo servidor, enviando um sinal, para que ele suspenda a tarefa do modo Estudo que está executando. Em seguida, deve-se conceder o uso do processador ao próximo aplicativo de análise de rede da fila de pronto, o que é realizado por uma sub-rotina que será descrita no final da seção. Após disparar a primeira tarefa da ordem de execução, o escalonador requer novamente, através da chamada de sistema **alarm**, a notificação pelo núcleo do sistema operacional do próximo ponto de escalonamento, quando então executará novamente a função de reordenamento da fila de pronto.

A função de término de aplicativos do modo Tempo Real

Conforme mencionado no apêndice A, o sinal SIGCLD é enviado ao processo-pai quando o estado de um processo-filho muda. A mudança de estado pode ser o término do processo-filho ou simplesmente que o processo-filho foi parado (suspenso) por um sinal SIGSTOP, SIGTTIN, SIGTTOU ou SIGTSTP. Quando o processo-pai não trata o sinal SIGCLD, apesar da ação padrão associada ao seu recebimento ser a de descartá-lo, o processo-filho se torna um processo "zumbi". Um processo zumbi é um processo que terminou sua execução, mas o processo que o gerou ainda não esperou por ele, isto é, não executou a chamada de sistema **wait** [Sun 90b]. Assim, é necessário que este trate o sinal SIGCLD para

evitar que se desperdice recursos do sistema com os processos-filho ao se tornarem "zumbis".

Uma vez que cada instância das funções de análise de rede será um processo-filho do escalonador e que ele precisa ser notificado quando elas terminam, a solução adotada foi a trivial, isto é, a de se aproveitar o sinal de término do processo-filho para notificar ao escalonador o término dos aplicativos de rede. Para isto, basta que se trate o sinal SIGCLD, inibindo o envio do mesmo para a situação de suspensão do processo-filho.

Ao ser notificado que um aplicativo de rede terminou, o escalonador, após bloquear o recebimento de sinais, deve atualizar a fila de pronto, retirando da mesma o processo que acaba de terminar, e desativar o nível de prioridade deste processo. Se houver algum outro aplicativo de rede ou a tarefa servidora na fila de pronto, o escalonador deve executar a sub-rotina de disparo do próximo processo em estado de pronto. Caso contrário, ele deve verificar se o nível de prioridade desativado é maior ou igual ao da tarefa servidora, quando, então, ele deve notificar ao módulo temporizador que o nível da tarefa servidora não está mais ativo. Em seguida, o escalonador pode desbloquear os sinais e retornar ao seu estado de suspensão.

A função de atendimento à requisição do operador feita através da servidora

Sempre que houver uma solicitação de execução de algum dos aplicativos do modo Estudo, feita ao módulo servidor, ele enviará um sinal ao módulo escalonador notificando tal evento. Ao receber este sinal, o escalonador dá inicio ao tratamento do mesmo, bloqueando o recebimento dos demais sinais. A primeira atividade a realizar é colocar a tarefa servidora na fila de pronto e então reordenar a mesma. Antes de chamar a sub-rotina de disparo da próxima tarefa da fila de pronto, o escalonador deve, caso exista algum aplicativo em estado de pronto, verificar se o de maior prioridade agora é diferente do que está executando, para que ele possa, então, suspendê-lo. Após o disparo da primeira tarefa da fila de pronto, o escalonador desbloqueia o recebimento dos sinais.

A função de término de aplicativos do modo Estudo

Quando a tarefa servidora completa o atendimento à requisição de execução de uma tarefa aperiódica, feita pelo operador do sistema, o módulo servidor deve notificar este evento ao escalonador para que ele possa retirá-la da fila de pronto, desativar o nível de prioridade da tarefa servidora e atualizar a ordem da fila. Antes, porém, de finalizar o tratamento deste sinal, o escalonador deve chamar a sub-rotina para disparo da próxima tarefa da fila de pronto, caso haja algum aplicativo ativado e não terminado. Caso contrário, ou seja, caso não haja outra tarefa a executar e como o nível de prioridade desativado é o da tarefa servidora, deve-se notificar ao módulo temporizador que este nível de prioridade não está mais ativo, para que ele possa confirmar o próximo instante de reposição da carga da servidora.

A função de impedimento/liberação de execução do modo Estudo

Existem dois cenários que ainda devem ser analisados: O primeiro é quando a tarefa servidora, já tendo solicitado sua execução, ao se encontrar em estado de pronto e com a "posse" da UCP, esgota sua carga (ou capacidade de serviço) e o segundo, quando a tarefa servidora, tendo ainda alguma requisição do operador a atender, aguarda o instante de reposição de sua capacidade de serviço. Como estes dois eventos serão sempre mutuamente excludentes, pode-se utilizar o mesmo sinal para notificar ao escalonador as suas ocorrências.

Assim, ao receber do módulo servidor este sinal, o escalonador averigua qual, dentre os dois eventos, foi realmente o que ocorreu. Caso tenha sido o primeiro, ele tem que retirar da fila de pronto a tarefa servidora, atualizando sua ordenação. Se a sinalização foi para notificar o instante de recarga da tarefa servidora, o escalonador deve recolocar a mesma na fila de pronto, reordená-la e, em seguida, verificar se o aplicativo que detém a "posse" da UCP é diferente daquele que se encontra no início da fila de pronto, suspendendo-o neste caso.

Qualquer que seja o evento ocorrido, ao final do tratamento do sinal, o escalonador deve, caso haja algum aplicativo em estado de pronto, chamar a sub-rotina de disparo da próxima tarefa da fila de pronto.

Disparo do próximo processo da fila de pronto

A sub-rotina de disparo da primeira tarefa da ordem de execução consiste de um procedimento para se conceder a "posse" do processador à tarefa que estiver em estado de pronto de maior prioridade, isto é, à tarefa ativada e não terminada de menor período.

Logo, a execução deste procedimento só será procedente quando o aplicativo de maior prioridade não for o que estiver executando no momento. O primeiro passo a realizar é verificar se a tarefa procurada se encontra dentre as que estão em estado de suspensão. Caso esteja, ela deve ser notificada para que possa voltar a deter a "posse" do processador, seja ela a tarefa servidora ou outro aplicativo de rede do modo Tempo Real qualquer.

No caso de o aplicativo em estado de pronto de maior prioridade não estar em estado suspenso, deve-se invocar sua execução. Quando for a vez de a tarefa servidora iniciar ou dar prosseguimento ao atendimento das funções do modo Estudo, o módulo escalonador deve notificar o módulo servidor para que este a dispare. Se a tarefa procurada for um dos aplicativos de rede do modo Tempo Real, deve-se criar um processo-filho que invoque sua execução. Este processo-filho, antes da execução da próxima função de análise de rede da lista de pronto, deve desbloquear todos os sinais bloqueados, temporariamente, pelo escalonador, para que o novo aplicativo, que será ativado, não receba este bloqueio como herança. A ação de desbloquear um sinal é realizada pela chamada de sistema **sigprocmask**, conforme apresentado no apêndice A.

É durante este procedimento que se deve comunicar ao módulo temporizador sobre a ativação e desativação do nível de prioridade da tarefa servidora. A ativação do nível prioridade da tarefa servidora ocorre quando o aplicativo terminado possui prioridade menor que a da servidora e o aplicativo a ser executado, prioridade maior ou igual à daquela. A desativação ocorre na situação inversa, isto é, quando a prioridade do aplicativo terminado é maior ou igual e a prioridade do aplicativo a ser executado é menor que a da tarefa servidora.

4.2.2 Módulo Servidor

Conforme mencionado na seção anterior, a execução do módulo servidor é invocada por um processo-filho do módulo escalonador. Ao iniciar sua execução, o módulo servidor deve realizar alguns procedimentos necessários para se preparar para o atendimento das requisições de execução dos aplicativos de rede do modo Estudo. O primeiro procedimento é o de inicialização, descrito a seguir.

Inicialização

O primeiro passo é efetuar o tratamento dos sinais que serão usados pelo módulo servidor para a notificação dos eventos que lhe sejam pertinentes. No caso do módulo servidor, tornou-se necessária a redefinição de 5 (cinco) sinais do sistema operacional. Cada um destes sinais é usado para se notificar um dos seguintes eventos: Liberação para início ou reinício da tarefa servidora, preempção de um aplicativo de análise de rede em relação à tarefa servidora, fim da capacidade de serviço, reposição da carga consumida e fim de execução de aplicativo análise de rede do modo Estudo.

Como os instantes de ocorrência destes eventos não são conhecidos *a priori* pelo módulo servidor, as funções associadas a cada um destes eventos também possuem natureza assíncrona. Estas funções serão discutidas no final desta seção.

O passo seguinte é a abertura do semáforo criado pelo módulo escalonador. Este semáforo será utilizado para o sincronismo entre os diferentes módulos, daí a necessidade de que todos possuam o seu identificador. Em seguida, o módulo servidor deve abrir a fila de mensagens, que também já deve ter sido criada pelo seu processo-pai, o módulo escalonador.

Uma vez definidos e abertos os mecanismos de comunicação que serão usados pelo módulo servidor, pode-se inicializar as variáveis e realizar o procedimento de leitura dos arquivos que contenham as informações sobre as funções de análise de rede. O arquivo de processos aperiódicos precisa conter apenas um

cadastro de todas as funções do sistema que se desejem executar através do modo Estudo com seus respectivos caminhos na árvore de diretórios. Isto porque é o módulo servidor que, uma vez que tenha sido solicitada a execução e o módulo escalonador a tenha permitido, irá ativar as funções aperiódicas. O arquivo com as informações dos processos periódicos é o mesmo lido pelo módulo escalonador, que contém quais são as funções de análise de rede a serem escalonadas em tempo real, seus tempos máximos de execução e seus períodos. A partir destes dados, o módulo escalonador é capaz de calcular a capacidade máxima de atendimento da tarefa servidora por cada período, segundo os conceitos mostrados no Capítulo 3. Para um conjunto de n aplicativos de rede periódicos e com a tarefa servidora sendo a de maior prioridade, tem-se:

$$U_s = 2\left(\frac{n-1}{U_p + n - 1}\right)^{n-1} - 1 \tag{4.1}$$

onde

U_s = capacidade da servidora
U_p = fator de utilização do processador pelos aplicativos periódicos

A equação 4.1, apresentada em [Leh 87], se aplica ao caso de um conjunto qualquer de n processos periódicos escalonados pelo Taxa Monotônica com Servidor Esporádico. Quando o número de processos periódicos é muito grande, a equação 3.6 deve ser usada (caso $n \rightarrow \infty$). É importante lembrar que, para um conjunto de processos periódicos com períodos múltiplos, como é o caso dos aplicativos de rede do modo Tempo Real, o limite do fator de utilização do processador pelas tarefas periódicas e pela tarefa servidora é de 100%.

Após determinado o limite de atendimento da tarefa servidora, o módulo servidor o escreve em um arquivo para que possa ser lido, posteriormente, pelo módulo temporizador. Na Figura 4.5 é mostrado um algoritmo do procedimento Inicialização, referente à Figura 4.3.

Criação do Temporizador

Analogamente à criação do próprio módulo servidor pelo escalonador, o módulo temporizador é invocado por um processo-filho do servidor. A criação deste processo-filho (**fork**) e a posterior invocação do módulo temporizador geram, então, um terceiro processo concorrente, que será responsável por administrar os instantes de esgotamento da carga de atendimento da tarefa servidora e de reposição desta mesma carga.

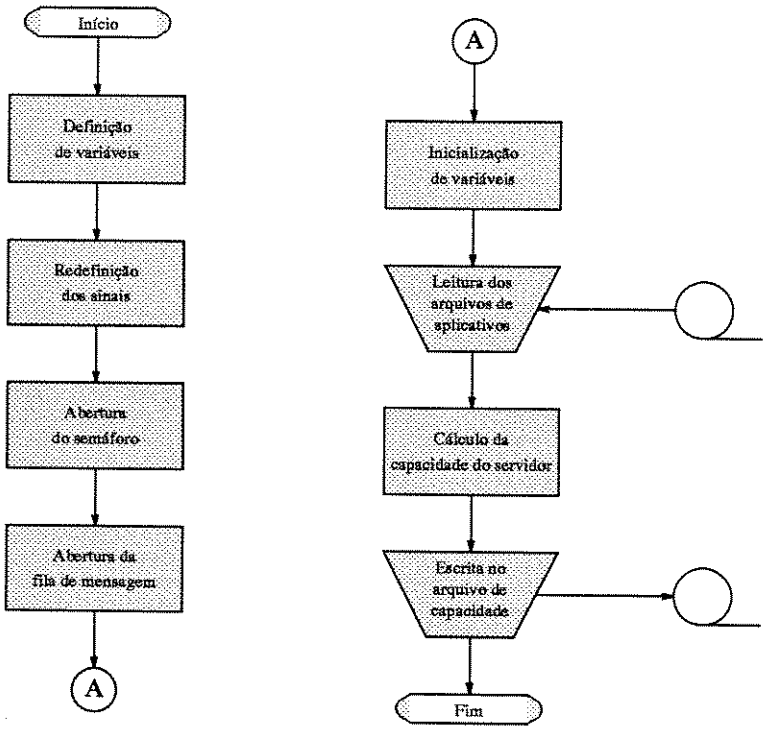

Figura 4.5: Inicialização do Servidor

Sincronização dos Módulos Servidor e Temporizador

Como a função do módulo temporizador é a de gerenciar o tempo de execução da tarefa servidora e os módulos servidor e temporizador tratam-se de processos distintos, é necessário que se garanta que este último esteja pronto para supervisionar os requisitos temporais da tarefa servidora quando esta for ativada. Isto é feito através do semáforo criado pelo módulo escalonador: o servidor deve esperar em uma operação P sobre o semáforo até que o temporizador, ao atingir uma determinada parte de seu código, realize uma operação V sobre este mesmo semáforo, garantindo assim sincronismo entre eles.

Comunicação com o Escalonador

Para que possa haver notificação dos eventos entre os diversos módulos, é preciso que cada um deles conheça o identificador do processo dos outros. Quando se trata apenas de um processo-pai e um processo-filho, isto pode ser realizado pelas chamadas de sistema fork e getppid, como é o caso dos módulos escalonador e servidor e módulos servidor e temporizador. É necessário, no entanto, que se informe ao módulo escalonador o identificador do processo do módulo temporizador, o que é feito através de troca de mensagem usando-se a fila criada pelo escalonador.

A fila de mensagens, além de realizar a função de comunicação dos identificadores, neste caso também é usada para a importante função de sincronizar os módulos escalonador e servidor; garantindo que a execução dos módulos servidor e temporizador já tenham atingido o estágio de suspensão no instante da ativação inicial realizada pelo módulo escalonador.

Atendimento às requisições do operador do sistema

O estado de atendimento às requisições aperiódicas do operador do sistema deve respeitar os mesmos princípios básicos descritos no procedimento de suspensão do escalonador, isto é, o pronto atendimento às requisições aperiódicas, a coexistência com o sistema de tempo real durante toda a sua duração e a não exigência do uso de recursos de processamento do sistema. O atendimento às requisições do modo Estudo é feito através de uma instrução de leitura do terminal de entrada, porém, ela é não reentrante [Ste 93], ou seja, uma vez que ela retorne, não mais reinicia. Assim, como é comum acontecer do módulo servidor receber um sinal, enquanto estiver aguardando uma entrada, o que provocaria o retorno da instrução de leitura, é necessário que esta instrução seja reiniciada sempre que seu retorno não for por uma requisição de execução das funções de análise de rede do modo Estudo.

Quando a instrução de leitura de entrada retornar com êxito, o servidor deverá bloquear o recebimento de todos os sinais, como uma forma de exclusão mútua de uma região crítica entre as suas próprias funções assíncronas, para que ele possa atualizar suas estruturas, colocando o aplicativo solicitado em estado de pronto, e enviar um sinal ao módulo escalonador solicitando o seu escalonamento. Antes de retornar ao início de seu estado de atendimento às requisições do operador, o módulo servidor deverá liberar todos os seus sinais por ele mesmo bloqueados, retornando à "máscara" de sinais anterior.

Funções Assíncronas do Servidor

Assim como no caso do módulo escalonador, as funções assíncronas do servidor, realizadas quando da notificação de determinados eventos, são executadas de forma indivisível, uma vez que quando um sinal é recebido, o recebimento de todos os outros é bloqueado.

A função de liberação para início ou reinício da tarefa servidora

Sempre que o módulo escalonador identificar o instante do escalonamento da tarefa servidora para deter a "posse" do processador, ele enviará um sinal ao módulo servidor. Quando o servidor receber este sinal a primeira vez após a sua última solicitação de execução, ele deve verificar qual é o aplicativo do modo Estudo que se encontra em estado de pronto e criar um novo processo, o qual será responsável pela ativação deste aplicativo. Este processo-filho deverá,

antes da invocação da tarefa aperiódica, desbloquear todos os sinais que tiverem sido bloqueados pelo módulo servidor, para que aquela não os receba como herança. Quando o aplicativo do modo Estudo solicitado já tiver sido iniciado, isto é, encontra-se em estado de suspensão, o que o módulo servidor deve fazer é enviar um sinal para que ele retorne ao estado de execução.

Em ambos os casos, porém, é necessário que o servidor notifique ao módulo temporizador a sua ativação, para que ele inicie a marcação da quantidade de carga consumida pela tarefa servidora no atendimento das requisições do operador.

A função de notificação de preempção à tarefa servidora

Quando a tarefa servidora estiver com a "posse" do processador e ocorrer, em algum ponto de escalonamento, a transição para o estado de pronto de um aplicativo do modo Tempo Real de maior prioridade que a da tarefa servidora, o módulo escalonador deverá requerer a "posse" da UCP à servidora para que ele possa ativar o aplicativo de rede do início da fila de pronto. Para isto, o escalonador envia um sinal ao módulo servidor que, por sua vez, suspende o aplicativo aperiódico que estiver executando e notifica tal evento ao módulo temporizador para que ele interrompa a descarga da capacidade de serviço da servidora, retornando, então, do tratamento deste sinal.

A função de tratamento do fim da capacidade de serviço

Enquanto a tarefa servidora estiver em estado de pronto, ela pode ativar qualquer um dos aplicativos do modo Estudo para atendimento das requisições do operador. Ela possui, no entanto, um limite de tempo no qual ela pode permitir a execução de tais aplicativos, que é controlado pelo módulo temporizador. Quando o temporizador percebe o fim da carga de serviço da servidora, ele envia um sinal a esta que, notificada de tal evento, além de providenciar a suspensão do aplicativo aperiódico que estiver executando, também notifica ao módulo escalonador que já não mais se encontra em estado de pronto.

A função de reposição da carga consumida

Quando chega o instante de repor a carga da tarefa servidora que foi consumida desde sua última recarga, o temporizador notifica ao módulo servidor que ele efetuou a reposição. Ao receber o sinal, o servidor verifica se havia algum aplicativo do modo Estudo executando no instante em que a carga terminou, o que implica na existência de alguma instância de função aperiódica suspensa. No caso negativo, o módulo servidor pode finalizar o tratamento do sinal, pois não há nada a fazer, porém, caso alguma tarefa tivesse sido interrompida, ele precisa enviar um sinal ao módulo escalonador, solicitando a entrada da tarefa servidora na fila de pronto, uma vez que a mesma já está liberada para executar os aplicativos do modo Estudo novamente.

A função de término de execução de aplicativos do modo Estudo

Quando a execução de um aplicativo aperiódico termina, o núcleo do sistema operacional, reconhecendo o término de um processo-filho, envia um sinal ao módulo servidor notificando o fim daquela tarefa. A primeira ação a tomar é identificar a existência ou não de mais aplicativos do modo Estudo em estado de pronto, isto é, já solicitados pelo usuário/operador. Quando houver outro aplicativo a ser ativado, é necessário, novamente, que se crie um processo-filho, que deve efetuar o desbloqueio de todos os sinais e então invocar a execução da nova tarefa aperiódica. Caso contrário, o que o módulo servidor tem a fazer é notificar a ocorrência de tal evento aos módulos escalonador e temporizador. Ao primeiro, para que ele possa retirar a tarefa servidora do estado de execução e, ao segundo, para que ele interrompa a contagem de tempo consumido pela tarefa servidora.

4.2.3 Módulo Temporizador

A função do módulo temporizador é contar o consumo de carga da tarefa servidora, marcar o instante de reposição tr_i da mesma e avisar ao módulo servidor do fim de sua capacidade de serviço e do instante de recarga. A Figura 4.3 mostra um algoritmo simplificado para o módulo temporizador. Nas seções seguintes são descritos os procedimentos.

Inicialização

Assim como nos outros dois módulos, a primeira ação a se realizar é a de se definir quais são os sinais a serem tratados e quais são as funções tratadoras associadas a cada um deles. No final desta seção, são descritas cada uma destas funções assíncronas.

O passo seguinte é o de se abrir o semáforo que será usado para realizar o sincronismo entre os módulos servidor e temporizador. Em seguida, o temporizador deve ler a informação de período da tarefa servidora especificado durante a inicialização do sistema e sua capacidade de serviço, que foi calculada e armazenada pelo módulo servidor, com base nas características do conjunto de funções de análise de rede que compõem o modo Tempo Real do sistema. A Figura 4.6 ilustra o procedimento Inicialização.

Sincronização com o Servidor

A sincronização com o servidor, conforme já mencionado, é necessária para que se garanta que o módulo temporizador esteja apto a cumprir sua função em relação ao módulo servidor e este com o escalonador. Na verdade,

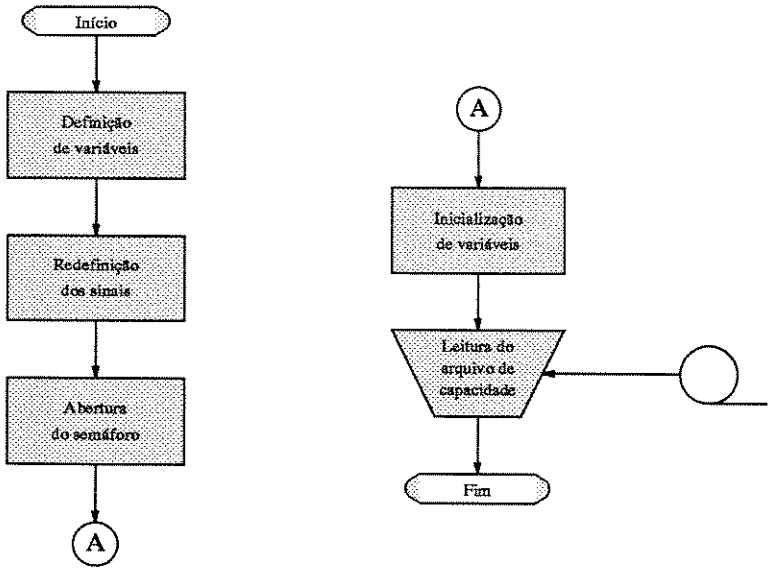

Figura 4.6: Inicialização do Temporizador

o que se pretende com o sincronismo entre os módulos é garantir que, como se tratam de processos concorrentes, um deles já tenha alcançado uma determinada região de seu código quando o outro estiver executando uma outra região já pré-determinada. Isto é realizado através do uso de um semáforo. Possivelmente o módulo servidor estará esperando em uma operação P sobre o semáforo, quando o temporizador atingir a instrução de operação V sobre este mesmo semáforo.

Supervisão das restrições temporais da tarefa servidora

Quando não houver aplicativo de rede do modo Estudo em estado de execução e nem um instante de reposição da carga da tarefa servidora previsto, não há atividade a ser executada pelo módulo temporizador. Assim, o que ele tem a fazer é aguardar, como o faz o módulo escalonador, a notificação de algum evento de forma a respeitar aqueles mesmos princípios obedecidos pelo módulo escalonador enquanto se encontra em estado de suspensão. Se houver, no entanto, qualquer aplicativo do modo Estudo executando ou algum instante de reposição previsto ou confirmado, para que possa executar sua função, o temporizador deve, a cada unidade de tempo, realizar três pequenas tarefas após bloquear o recebimento de quaisquer sinais. Como as funções de análise de rede possuem tempos de execução na ordem de segundos, esta foi a unidade de tempo adotada durante a implementação do escalonador, entretanto, poderia-se usar unidades menores. Cada uma das tarefas é descrita a seguir:

- se houver alguma função do modo Estudo executando, o temporizador deve diminuir uma unidade de tempo da carga da tarefa servidora e incrementar

uma unidade o contador do tempo por ela consumido. Caso a carga tenha terminado, o servidor deve ser avisado, para que ele possa suspender o aplicativo aperiódico, e um novo instante de reposição deve ser confirmado;

- se houver algum instante de reposição já confirmado, deve-se atualizar a estrutura que contenha a informação de quanto tempo ainda falta até o primeiro dentre estes instantes;
- se for o instante de recarregar a capacidade de serviço da tarefa servidora, o temporizador, além de repor o tempo por ela consumido, deve, caso o nível de prioridade da servidora esteja ativo, fazer a previsão de um novo instante de reposição de carga e, se a servidora estiver sem carga, avisar ao módulo servidor. O valor do instante de reposição tr_i previsto deve ser igual ao instante atual mais o período da tarefa servidora.

Finalmente, o módulo temporizador deve desbloquear o recebimento dos sinais e aguardar mais uma unidade de tempo, para que possa refazer, novamente, estas três tarefas.

Funções Assíncronas do Temporizador

O módulo temporizador apresenta apenas três funções assíncronas: as funções de início e de fim de aplicativos do modo Estudo e a função de agendamento do instante tr_i. As funções de início e de fim dos aplicativos aperiódicos têm por objetivo apenas atualizar a informação, no módulo temporizador, se há ou não funções do modo Estudo executando.

Já a função de agendamento do instante de reposição da carga da tarefa servidora, que é disparada quando há transição de estado de ativação do nível de prioridade da tarefa servidora, é responsável por prever ou confirmar um instante tr_i. Quando o nível de prioridade P_i da tarefa servidora se torna ativo e ela possui tempo de execução disponível, o instante de reposição tr_i é atribuído, isto é, previsto, para um instante igual ao atual mais o período da servidora. Quando o nível de prioridade se tornar inativo, deve-se confirmar a reposição da carga da servidora, que deverá ser igual à quantidade de tempo de execução por ela consumido.

4.3 Modelo Próximo Prazo

Nesta seção será apresentada a solução de implementação do escalonador dinâmico Próximo Prazo sobre o UNIX. A Figura 4.7 mostra como foram divididas as funções de escalonamento dos aplicativos de análise de rede dos modos Tempo Real e Estudo. Assim como no caso do escalonador Taxa Monotônica, o primeiro nível da implementação do modelo Próximo Prazo apresenta

um módulo de escalonamento que é responsável pelo gerenciamento da fila de tarefas em estado de pronto e concessão da "posse" do processador ao aplicativo de maior prioridade, que, neste caso, não é o de menor período e sim o que está mais próximo de seu prazo.

A diferença para o caso da implementação do modelo Taxa Monotônica é a de que não há uma tarefa servidora para as requisições do modo Estudo. O pedido de execução de um aplicativo aperiódico é feito através do módulo solicitador, que também se encontra no nível 1, cuja única função é a de comunicar ao módulo escalonador quando e qual aplicativo foi requisitado. É o próprio módulo escalonador que irá invocá-los quando chegarem seus instantes de execução. Logo, tanto os aplicativos de análise de rede do modo Tempo Real quanto os do modo Estudo se encontram em um mesmo nível, pois, fora a forma como entram em estado de pronto, não apresentam diferença para o módulo escalonador.

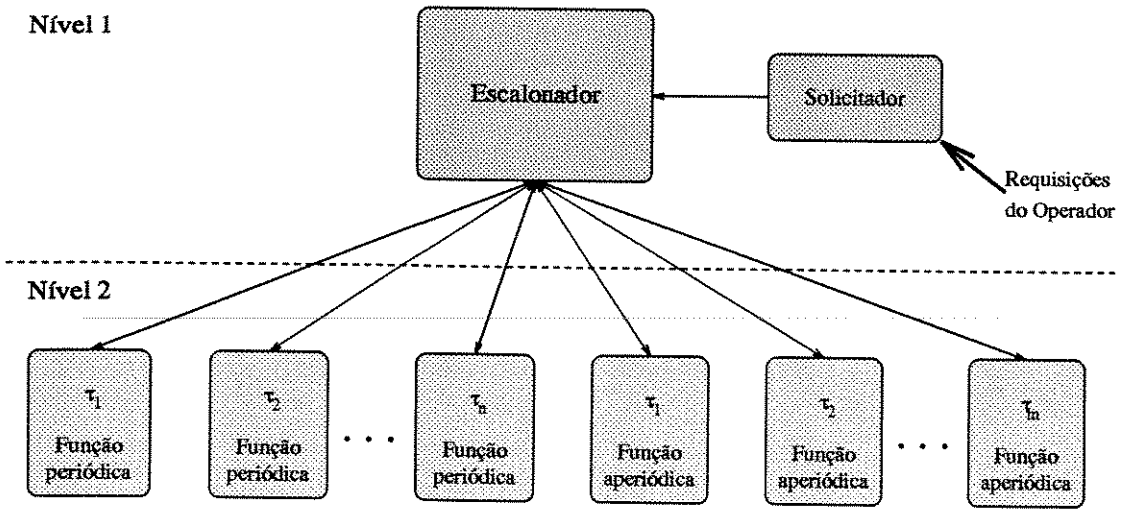

Figura 4.7: Níveis de Aplicativos do Modelo Próximo Prazo

Na Figura 4.8 é apresentado um algoritmo simplificado do modelo de implementação do escalonador Próximo Prazo proposto. Como no caso da implementação do escalonador TM, aqui também, durante a inicialização do sistema, é necessária apenas a ativação do módulo escalonador. A partir deste é criado o módulo solicitador. A sincronização entre os dois módulos concorrentes, por se tratarem de processos distintos, é feita através de troca de mensagens entre os dois, gerando, assim como no caso do escalonador anterior, relações de dependência e descendência entre os módulos do sistema. É importante notar que, para que se pudesse extrair parâmetros mais precisos de comparação entre os dois escalonadores, procurou-se, durante a implementação do escalonador Próximo Prazo, seguir a mesma estrutura do modelo Taxa Monotônica, o que pode ser comprovado analisando-se as figuras 4.3 e 4.8. A seguir serão apresentados com maiores detalhes os dois módulos.

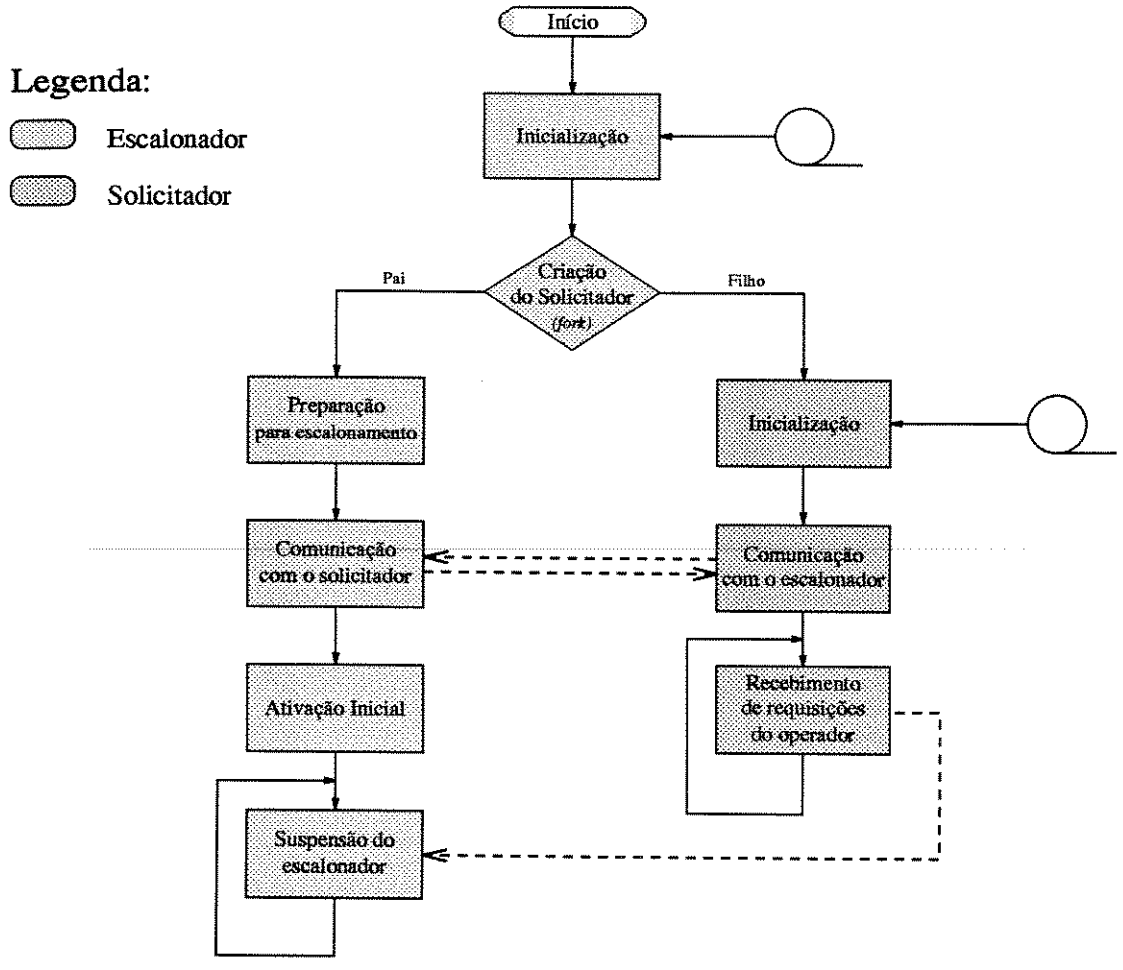

Figura 4.8: Algoritmo do Modelo de Implementação do PP

4.3.1 Módulo Escalonador

O módulo escalonador do modelo PP possui os mesmos procedimentos do módulo escalonador do TM, isto é, os procedimentos de inicialização, criação de um processo-filho que receberá as requisições do operador/usuário, preparação para escalonamento, comunicação com o seu processo-filho, ativação inicial e suspensão do escalonador. Existem, no entanto, dois pontos que marcam a diferença entre estes dois escalonadores, que são: o fato de ser o próprio módulo escalonador que realiza a invocação dos aplicativos do modo Estudo e a necessidade de se gerenciar as prioridades das funções, por serem dinâmicas durante a existência do sistema. Estas diferenças receberão maior ênfase durante a apresentação de cada um destes procedimentos, o que será feito a seguir.

Inicialização

O primeiro passo do procedimento de inicialização é definir as ações que devem ser tomadas quando do recebimento de sinais que tenham por função notificar a ocorrência dos eventos importantes à operação do escalonador, que são: a chegada de um novo ponto de escalonamento, o fim de um aplicativo de análise de rede dos modos Tempo Real ou Estudo e a comunicação da requisição de execução de um aplicativo aperiódico, por parte do operador, através do módulo solicitador. Como não há uma tarefa específica com a função de gerenciar a execução dos aplicativos do modo Estudo, não há mais a necessidade de bloqueio ao atendimento destes, além do fim de aplicativos de ambos os modos serem notificados da mesma forma ao escalonador, gerando, assim, uma redução e, até mesmo, uma simplificação no que se refere às funções assíncronas realizadas pelo módulo escalonador. Assim, não mais se torna necessária a redefinição de sinais do sistema operacional, bastando, para tanto, associar tais ações aos sinais disponíveis do sistema operacional.

Como no caso do módulo escalonador TM, as funções assíncronas do módulo escalonador PP são ativadas pela ocorrência de eventos concorrentes, isto é, o fim de uma das funções, a requisição de execução de um aplicativo do modo Estudo ou mesmo a chegada de um instante que é um ponto de escalonamento podem ocorrer concomitantemente. Para garantir a exclusão mútua às suas regiões críticas, as funções assíncronas devem ser realizadas de forma atômica, necessitando-se, para tanto, bloquear o recebimento de todos os demais sinais, logo que ocorra a chegada de qualquer um deles. As funções assíncronas do módulo escalonador são apresentadas no final desta seção.

O passo seguinte ao da definição dos sinais do sistema operacional é o da criação de uma fila de mensagens para comunicação entre os módulos. Neste passo, o módulo escalonador cria uma fila de mensagens que, posteriormente, será aberta pelo módulo solicitador. Deve-se notar que, por se tratarem apenas de dois módulos, a sincronização entre os dois pode ser realizada através da troca

de mensagens, dispensando-se a criação e o uso de semáforos.

Uma vez criada a fila de mensagens, o módulo escalonador inicializa as variáveis e realiza a leitura de um arquivo que deve conter todas as informações sobre as funções de análise de rede. As informações referentes às funções periódicas são as mesmas requeridas pelo escalonador Taxa Monotônica, isto é, o período e o pior tempo de execução de cada uma, em segundos. Os períodos das funções de análise de rede do modo Tempo Real devem ser fornecidos para que o escalonador possa ter ciência de qual a freqüência que ele deve escalonar cada uma. Considera-se que os tempos de pronto das funções sejam os mesmos que os de chegada. Considera-se, também, que os seus prazos de execução, em cada nova instância, sejam sempre considerados iguais aos instantes em que elas se tornarem prontas nesta instância mais os seus períodos.

Os tempos de execução declarados devem ser os piores tempos de execução obtidos em modo *off-line*. Como a execução das funções de análise de rede para sistemas de potência do porte do sistema Sul-Sudeste Brasileiro, executadas em estações de trabalho, estão na ordem de segundos, não é necessária a inclusão dos *overheads* da camada escalonadora, nem mesmo os de tempos de chaveamento de contexto devido à preempção das funções, o que será mostrado no Capítulo 5. A finalidade da informação dos tempos máximos de execução dos aplicativos do modo Tempo Real é a de se calcular o fator de utilização do processador U_p, que, no caso do algoritmo Próximo Prazo, só não deve exceder ao valor 1, ou seja, 100%.

Após as informações dos aplicativos do modo Tempo Real, deve-se declarar as funções do modo Estudo. A única informação necessária, neste caso, é o próprio nome da função. A utilização do processador por estas funções pode atingir a $1 - U_p$. Assim como no algoritmo Taxa Monotônica, o escalonador Próximo Prazo permite a declaração de um prazo para uma tarefa aperiódica, entretanto, isto foge ao escopo do problema dos aplicativos do modo Estudo do subsistema de análise de rede.

No caso do modelo Taxa Monotônica, a prioridade das tarefas era determinada neste ponto do processo. No modelo Próximo Prazo, no entanto, como as prioridades são atribuídas às funções dinamicamente, só o que o módulo escalonador tem a fazer é ordenar os períodos dos aplicativos do modo Tempo Real, que serão usados para a determinação dos primeiros prazos de cada uma deles. O algoritmo do procedimento Inicialização, constante da Figura 4.8, é ilustrado na Figura 4.9.

Criação do Solicitador

A função do módulo escalonador é a de administrar as prioridades dos aplicativos do sistema, o uso e a concessão do processador e a chegada de novos pontos de escalonamento, escalonando as funções dos modos Tempo Real

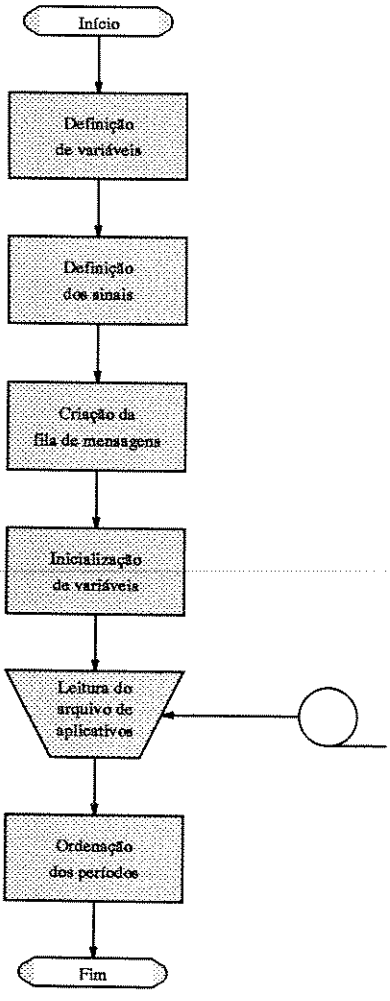

Figura 4.9: Inicialização do Escalonador PP

e Estudo que, por ventura, tenham sido requisitadas. Como ele deve estar atento a todas essas atividades durante todo o tempo de existência do sistema de tempo real, é preciso que haja um outro processo que realize a função de receber as requisições de execução de aplicativos feitas pelo operador. Assim, o módulo escalonador, através da chamada de sistema **fork**, gera um processo-filho, que, por sua vez, invocará a chamada de sistema **exec**, a qual, finalmente, irá executar um novo programa, o módulo solicitador.

Preparação para Escalonamento

A preparação para escalonamento é dividida em duas partes. Como o escalonador precisa conhecer as prioridades dos aplicativos antes de ordená-los na fila de pronto, ele primeiro deve determinar o vencimento dos prazos das funções de análise de rede. Isto porque são eles que irão determinar, para um instante t_i qualquer, a prioridade de cada aplicativo neste mesmo instante t_i. Além disso, o escalonador precisa conhecer estes prazos para que, após a invocação das funções prontas no instante inicial, ele possa solicitar ao núcleo do sistema operacional a notificação do novo ponto de escalonamento, para reordenamento da fila de pronto. Como é feita a consideração de que, na inicialização do sistema, todas as funções de análise de rede do modo Tempo Real já se encontrem em estado de pronto, os seus primeiros prazos são exatamente o instante inicial do sistema acrescido do período de cada uma.

Conforme discutido no Capítulo 2, as funções de análise de rede referentes aos dados de tempo real de uma mesma varredura devem ser escalonadas de acordo com a seqüência lógica da Figura 2.2, não importando a ordem de execução entre elas, quando se referem aos dados de instantes diferentes. Como os períodos de execução destas funções crescem na ordem da seqüência da Figura 2.2, a política de escalonamento do algoritmo Próximo Prazo atende às necessidades de escalonamento das funções de análise de rede que se referem a um mesmo instante, pois os seus prazos também crescem de acordo com aquela seqüência, diminuindo suas prioridades.

No caso particular de empate dos prazos de duas ou mais funções que se refiram aos dados de uma mesma varredura do subsistema de aquisição de dados, o mesmo critério do modelo Taxa Monotônica é adotado, sem perda da generalidade do algoritmo Próximo Prazo. Isto é, caso existam duas ou mais funções com o mesmo prazo para término de execução e que se refiram aos dados de tempo real de um mesmo instante t_i, a primeira a ser executada será a primeira especificada dentro do arquivo que contenha as informações sobre os aplicativos. Assim, para atender às necessidades de escalonamento das funções de análise de rede, basta que elas sejam especificadas, dentro do arquivo de processos, na ordem de sua seqüência lógica.

Uma vez que o escalonador já conhece as prioridades das funções para o instante inicial do sistema, ele deve, então, executar uma rotina de or-

denamento das tarefas em estado de pronto, gerando, assim, a primeira fila de pronto. O ordenamento consiste apenas em se gerar uma lista com as funções, que se encontrem em estado de pronto, em ordem decrescente de prioridade naquele instante, ou seja, no caso da inicialização do sistema, gerar a fila de pronto com os aplicativos de rede em ordem crescente de prazos.

Comunicação com o Solicitador

A comunicação entre o escalonador e o solicitador é feita através da fila de mensagens criada pelo primeiro. Para que a comunicação possa se efetuar de fato, é preciso que o solicitador tenha aberto a mesma fila criada pelo escalonador. Um dos objetivos da comunicação entre os dois módulos é a troca de seus identificadores, para que possam notificar a ocorrência de eventos. O escalonador inicia a troca de mensagens, escrevendo na fila o seu identificador e passando, em seguida, a esperar a mensagem com o identificador do solicitador. Este, por sua vez, deve ler a mensagem enviada pelo escalonador, retirando-a da fila e, então, escrever nela uma mensagem contendo seu identificador.

Este, no entanto, não é o principal objetivo da comunicação entre os módulos, mesmo porque a informação de qual é o identificador do processo-filho poderia ser obtida pelo módulo escalonador através do retorno da chamada **fork** e, o identificador do escalonador, poderia ser obtido pelo solicitador com a chamada de sistema **getppid**. Mas como os dois módulos são processos concorrentes à UCP, de alguma forma, é necessário que se garanta uma mínima sincronização entre os dois, o que é, perfeitamente, alcançado com o esquema de comunicação por troca de mensagens.

Ativação Inicial

Após o procedimento de comunicação com o módulo solicitador, o escalonador já se encontra pronto para conceder o processador ao aplicativo de análise de rede de maior prioridade do sistema, para que ela realize a execução de sua primeira instância. Assim como no caso do modelo Taxa Monotônica, o módulo escalonador não pode esperar pelo término do mesmo, tornando-se necessário gerar um processo-filho, que tenha por função executar o aplicativo de análise de rede.

Enquanto o processo gerado pelo módulo escalonador invoca o aplicativo do modo Tempo Real de maior prioridade em estado de pronto, o escalonador marca o instante do próximo ponto de escalonamento, agendando o envio de um sinal, pelo núcleo do sistema operacional, quando da chegada deste instante.

Suspensão do Escalonador

O procedimento de suspensão do escalonador PP é idêntico ao do TM. Aqui também, tudo o que o escalonador tem a fazer é aguardar a notificação da ocorrência dos eventos de chegada de um novo ponto de escalonamento, fim de execução de aplicativo de análise de rede ou requisição, pelo operador, da execução de algum deles. Para tanto, o escalonador entra em um laço infinito com uma instrução cuja função é suspendê-lo. Ao receber qualquer sinal, o escalonador retorna, executa a função tratadora do sinal recebido e volta a se suspender. Esta atividade cíclica do módulo escalonador garante a pronta execução de qualquer função assíncrona, a coexistência com o sistema de tempo real durante toda a sua duração e, ainda, uma ínfima utilização dos recursos do sistema, que, preferencialmente, devem ser alocados à utilização por parte das funções do modo Tempo Real e modo Estudo.

Funções Assíncronas do Escalonador

Após ter ativado o primeiro aplicativo de análise de rede do modo Tempo Real em estado de pronto, o escalonador entra em estado de suspensão para aguardar a notificação da ocorrência de algum evento relevante à sua função. No caso do escalonador Próximo Prazo, estes eventos são um dos três: chegada de um instante de reordenamento da fila de pronto, isto é, de um ponto de escalonamento, término de execução de um aplicativo do modo Tempo Real ou modo Estudo ou a comunicação de requisição, por parte do operador, de execução de um aplicativo do modo Estudo.

Com a ocorrência de qualquer um desses eventos, o escalonador deve retornar do estado de suspensão, tratar o sinal recebido, executando a ação associada a este sinal no procedimento Inicialização, e, então, entrar em suspensão novamente. Como esses eventos, no entanto, podem ocorrer simultaneamente, ou mesmo ligeiramente defasados, é fundamental, uma vez que o escalonador tenha iniciado o tratamento de qualquer um dos sinais recebidos, que tal procedimento não seja interrompido para tratamento de outro sinal, sob hipótese alguma. Para garantir esta atomicidade na execução do tratamento de um sinal, o que se faz é bloquear o recebimento de um segundo sinal sempre que o tratamento do primeiro ainda não tiver terminado, uma vez que não seja possível inibir o envio daquele.

A seguir são explicadas as três funções assíncronas desempenhadas pelo módulo escalonador.

A função de reordenamento da fila de pronto

No instante da ativação inicial, o módulo escalonador agendou junto ao núcleo do sistema operacional, o envio de um sinal no instante de vencimento do prazo da primeira função de análise de rede. Chegado este instante, o núcleo notifica sua ocorrência ao módulo escalonador que, então, reprepara a fila de

pronto.

O primeiro passo a executar é colocar em estado de pronto todos os aplicativos que iniciem um novo período no dado instante. Em seguida, o módulo escalonador rearranja os pontos de escalonamento, de acordo com a ordem dos novos prazos de cada função e, então, a partir desta nova ordenação, redispõe os aplicativos em estado de pronto dentro da fila.

Com a ocorrência deste ponto de escalonamento, pode acontecer alteração na prioridade de alguns aplicativos e, conseqüentemente, na ordem de escalonamento. Assim, torna-se necessário que o escalonador verifique, no caso de haver algum aplicativo em estado de pronto, se o de próximo prazo é o que está executando no momento. Em caso negativo, o módulo escalonador deve fazer a suspensão da função de análise de rede que detém o uso do processador e invocar a rotina que dispara a tarefa de maior prioridade em estado de pronto, descrita no final da seção.

Por fim, o que resta ao escalonador fazer, antes de desbloquear o recebimento dos demais sinais e retornar ao estado de suspensão, é agendar com o núcleo do sistema operacional a notificação da ocorrência do próximo ponto de escalonamento, determinado a partir dos novos prazos dos aplicativos de rede, que é o próprio prazo do aplicativo agora em execução.

A função de término de aplicativo de análise de rede

A solução adotada para a notificar ao escalonador Próximo Prazo quanto ao término de execução das funções de análise de rede, tanto do modo Tempo Real como do modo Estudo, foi a mesma adotada para o caso das funções periódicas do escalonador Taxa Monotônica. Assim, tudo o que se teve a fazer foi bloquear o envio do sinal do processo-filho, executor do aplicativo de análise de rede, para o caso de sua suspensão, permitindo que o sinal fosse enviado apenas no caso de término da função.

Ao receber a notificação de que o aplicativo em estado de pronto terminou sua execução, periódico ou não, o escalonador retorna do estado de suspensão, bloqueia o recebimento dos demais sinais, retira da fila de pronto o aplicativo terminado, atualizando-a para que, caso ainda haja alguma outra função na fila, a rotina de disparo do próximo processo em estado de pronto possa invocar aquele de maior prioridade. Isto feito, o escalonador pode desbloquear o recebimento dos sinais e colocar-se em suspensão novamente.

A função de atendimento à requisição do operador (modo Estudo)

Quando a execução de uma função do modo Estudo é requisitada, o módulo solicitador não apenas envia um sinal ao módulo escalonador, para notificar a ocorrência da requisição, como também escreve na fila de mensagens, por ambos aberta, qual das funções foi solicitada. Ao receber o sinal enviado pelo solicitador, os demais sinais são bloqueados e o escalonador é retirado do

estado de suspensão. A primeira tarefa a realizar é ler a mensagem que contenha a informação de qual aplicativo do modo Estudo foi solicitado, retirando-a da fila, e, em seguida, colocá-lo em estado de pronto.

No caso das funções de análise de rede em um centro de controle, as funções do modo Estudo não apresentam prazos para término de execução, o que as tornam de baixa prioridade. É necessário, no entanto, realizar a reordenação da fila de pronto, com o objetivo de organizar as funções do modo Estudo em estado de pronto de acordo com o critério de desempate estabelecido no arquivo de descrição dos aplicativos. O passo final é verificar se o aplicativo de próximo prazo em estado de pronto é o que está executando no momento. Caso não seja, deve-se suspender o detentor da "posse" da UCP e invocar a rotina de disparo do próximo processo na ordem de execução, quando, então, o escalonador deve desbloquear o recebimento dos sinais e retornar ao estado de suspensão.

Caso houvesse interesse em se escalonar funções do modo Estudo com prazos, só o que se necessitaria é a imposição de um tempo mínimo entre invocações de uma mesma função.

Disparo do próximo processo da fila de pronto

Toda vez que a ordem de execução dos aplicativos na fila de pronto sofre alguma alteração, o módulo escalonador precisa verificar se a função que detém o uso do processador ainda é a de maior prioridade. Quando isto não acontece, é preciso realizar a suspensão do aplicativo em execução e depois conceder a "posse" da UCP àquele de próximo prazo.

Antes de invocar a execução de uma nova instância da função de análise de rede de maior prioridade, no entanto, é preciso primeiro verificar se ela já não foi ativada e se encontra na lista de funções suspensas. Neste caso, tudo o que se tem a fazer é retorná-la do estado de suspensão, caso contrário, é preciso criar um processo-filho que invoque a execução de uma nova instância da próxima função de análise de rede da fila de pronto. O processo-filho deverá, porém, desbloquear todos os sinais, temporariamente, bloqueados pelo escalonador, para que a nova função a ser executada não receba este bloqueio como herança.

4.3.2 Módulo Solicitador

A função do módulo solicitador é apenas a de receber as requisições de execução das funções do modo Estudo e comunicar ao módulo escalonador quando e qual função foi solicitada. A seguir são descritos os passos do módulo solicitador.

Inicialização

Durante o procedimento de inicialização, o módulo solicitador deve abrir a fila de mensagens que foi criada pelo escalonador, para que a comunicação entre os dois possa se efetuar. Em seguida, ele deve abrir o arquivo que contém as informações sobre os aplicativos do sistema, extraindo aquelas relativas às funções de análise de rede do modo Estudo.

Comunicação com o Escalonador

A informação que o solicitador troca com o escalonador poderia ser obtida sem a necessidade de se usar filas de mensagens. Para obter o identificador do processo do escalonador, necessário ao solicitador para que ele possa notificar a ocorrência de alguma requisição aperiódica, bastaria que ele invocasse a chamada de sistema **getppid**. Isto, porque o processo que executa o módulo solicitador é um processo-filho daquele que executa o módulo escalonador. Além disso, o identificador do processo do solicitador é retornado ao módulo escalonador, logo quando este invoca a criação do primeiro.

Por outro lado, é preciso que se garanta que a execução do módulo solicitador já tenha atingido o estágio de recebimento das requisições aperiódicas do operador, no instante de ativação inicial dos aplicativos do sistema. Assim, o procedimento de comunicação com o escalonador é realizado mais com o objetivo de sincronização entre os módulos, do que propriamente o de troca de informação. Após escrever o identificador de seu processo na fila, o solicitador passa ao procedimento de recebimento de requisições aperiódicas, enquanto o escalonador aguarda a chegada da mensagem, garantindo assim a sincronização.

Recebimento de Requisições do Operador do Sistema

Para realizar o recebimento das requisições aperiódicas de forma imediata, o módulo solicitador o faz através de uma instrução de leitura do terminal de entrada, o que garante também o não comprometimento dos recursos de processamento do sistema, os quais devem ser alocados exclusivamente às funções dos modos Tempo Real e Estudo. Uma vez solicitada a execução de alguma das funções aperiódicas do sistema, o módulo solicitador escreve na fila de mensagens a identificação do aplicativo requerido e notifica, através do envio de um sinal, a ocorrência deste pedido.

Logo que tenha comunicado a requisição ao módulo escalonador, o solicitador deve sempre retornar ao estado de leitura do terminal de entrada, garantindo sua coexistência com o sistema de tempo real.

Quando for interessante se contemplar a existência de funções de

análise de rede do modo Estudo com restrições de tempo, isto é, funções aperiódicas que possuam prazos para término, é neste procedimento que se deve bloquear a solicitação de duas instâncias de uma mesma função que não sejam separadas de um intervalo de tempo mínimo pré-definido, retardando-se a comunicação da segunda requisição ao módulo escalonador.

Capítulo 5

Testes e Resultados

No capítulo anterior, foram apresentadas duas propostas de implementação de modelos de escalonamento para as funções avançadas de análise de rede em um ambiente computacional executando sistema operacional UNIX. A primeira delas é baseada na política de escalonamento estático do algoritmo Taxa Monotônica com Servidor Esporádico para atendimento das requisições do modo Estudo. A segunda é um modelo de implementação para o escalonador dinâmico Próximo Prazo. Neste capítulo comparam-se os desempenhos de ambos, quando submetidos a alguns testes. A escolha dos testes apresentados baseou-se na tentativa de se reproduzir, com a máxima fidelidade, a real situação do subsistema de análise de rede de um Centro de Controle moderno.

5.1 Ambiente Computacional Utilizado

Os subsistemas de análise de rede dos modernos Centros de Controle requerem arquiteturas computacionais de alto desempenho de processamento devido às características das funções de análise de rede. Para a realização dos testes com os escalonadores, utilizou-se um conjunto de estações de trabalho, dedicadas ao subsistema de análise de rede. Três ambientes foram utilizados. O primeiro, ambiente A, é constituído de uma estação *SUN SPARCstation 10* - Modelo 10/40, executando o *Solaris Release 2.4*. Os ambientes B e C são formados por, respectivamente, uma estação *SUN SPARCstation 2* - Modelo 4/75 e uma estação *SUN SPARCserver 330* - Modelo 4/330, ambas sob o *SunOS Release 4.1.1*. A Tabela 5.1 reúne as principais características de cada um dos ambientes utilizados para os testes: memória, freqüência de *clock*, MIPS (Milhões de instruções por segundo), MFLOPS (Milhões de instruções em pontos flutuantes por segundo) e índices da SPEC[1].

[1]SPEC (Standard Performance Evaluation Corporation) é uma instituição sem fins lucrativos formada para " estabelecer, manter e endossar um conjunto padronizado de relevantes *benchmarks* que possam ser aplicados à mais nova geração de computadores de alto desempenho".

Tabela 5.1: Características dos Ambientes Computacionais

Ambiente	Memória Principal (MB)	Freqüência de *clock* (MHz)	MIPS	MFLOPS	SPECint92[2]	SPECfp92[3]
A	64	40,0	90,0	10,6	50,2	60,2
B	16	33,0	28,5	4,2	21,8	22,8
C[4]	32	25,0	16,0	2,6	-	-

5.2 Características das Funções Avançadas de Análise de Rede Elétrica Utilizadas nos Testes

Para testar a adequação dos escalonadores propostos, foram usadas 5 (cinco) funções para o modo Tempo Real e uma para o modo Estudo. A primeira função do modo Tempo Real é a tarefa de atualização da base de dados dinâmica do sistema elétrico, isto é, a tarefa responsável pela atualização das alterações de medições e estados dos equipamentos na base de dados dinâmica, ocorridas entre as duas últimas varreduras do subsistema de aquisição de dados. Por isto, ela deve ter período igual ao da varredura do sistema, que nos modernos Centros de Controle está em torno de 10 segundos, e estar à frente da seqüência lógica de execução das tarefas de um mesmo instante de ativação.

As outras quatro funções do modo Tempo Real são os aplicativos de análise de rede. O primeiro deles é o configurador, que é responsável por determinar a topologia do sistema elétrico em tempo real. Seu período ideal, portanto, é também igual ao da varredura do subsistema de aquisição de dados. O configurador deve ler tanto a base de dados estática como a dinâmica para construir o modelo do sistema.

O segundo aplicativo de análise de rede utilizado engloba as funções de análise de observabilidade e estimação de estado. Esta função determina qual(is) é(são) a(s) parte(s) observável(is) do sistema, a partir do conjunto de medições e do modelo fornecido pelo configurador, e calcula uma melhor aproximação do estado (magnitudes e ângulos das tensões do sistema) das ilhas ob-

[2]SPECint92 é derivado dos resultados de um conjunto de *benchmarks* de inteiros e pode ser usado para estimar o desempenho de uma máquina com uma única tarefa de código com operações de inteiros.

[3]SPECfp92 é derivado dos resultados de um conjunto de *benchmarks* de ponto flutuante e pode ser usado para estimar o desempenho de uma máquina com uma única tarefa de código com operações de ponto flutuante.

[4]Esta estação apresenta índice 11,3 de SPECmark89, que está obsoleto. Não há avaliação de seu desempenho com relação aos parâmetros atuais.

serváveis. Assim, a fim de manter sempre atualizada a informação estimada do estado do sistema, as funções observador mais estimador serão executadas na mesma freqüência do configurador.

O terceiro aplicativo do modo Tempo Real possui a função de estimar o estado das ilhas não observáveis, complementando o modelo de tempo real do sistema de interesse para o instante no qual foram realizadas as medidas. Esta função inclui os valores de injeção de potência ativa e reativa previstos pelo planejamento nas barras não telemedidas, fixando os valores já estimados para as barras observáveis e eliminando aquelas que apresentem erros grosseiros. Esta função faz o papel do fluxo de potência *on-line*. Dado o fato de utilizar dados de planejamento, a oscilação do perfil de tensão calculada para esta parte do sistema elétrico, em regime permanente, é lenta, não havendo necessidade de uma alta freqüência de execução. Para a realização dos testes, foi adotado o período de 20 segundos.

Por fim, o quarto aplicativo de análise de rede, a análise de contingências, tem a função de avaliar se o estado de operação do sistema é seguro ou inseguro com respeito a um conjunto de contingências. Dentre as quatro funções do modo Tempo Real utilizadas durante os testes, esta é a função que demanda maior esforço computacional, devido ao número de casos que devem ser simulados e à precisão requerida para convergência, precisando-se muitas vezes adotar o tempo de execução como critério de parada. Durante a execução dos testes, foi monitorada uma lista contendo, entre simples e múltiplas, 10 contingências consideradas de maior severidade para o sistema em questão. Esta função também pode apresentar uma menor freqüência de execução, tendo sido adotado o período de 40 segundos.

Os piores tempos de execução de todas as funções de análise de rede dependem sempre do ambiente computacional e do porte do sistema que está sendo analisado. O sistema escolhido para os testes descritos neste capítulo corresponde a uma parte do sistema elétrico da Companhia Paulista de Força e Luz, referente à região de Campinas. Como a topologia do sistema pode mudar no tempo, pode haver também alteração na dimensão do mesmo. Para as bases de dados dinâmicas usadas nos testes, o sistema apresentou, em média, 206 barras e 233 ramos (linhas de transmissão e transformadores). Os tempos de execução das funções, para cada um dos diferentes ambientes, são descritos na seção seguinte.

O modo Estudo é constituído da função de fluxo de potência *off-line* com análise de contingências. Através dela, torna-se possível simular situações operativas programadas ou previstas e analisar situações passadas, além de permitir um treinamento dos operadores do sistema. Na Tabela 5.2 são apresentadas as características, o nome pelo qual serão tratadas durante este capítulo e o período de todas as funções testadas.

Tabela 5.2: Características e Períodos das Funções de Análise de Rede

Subsistema	Modo	Funções	Nome da Tarefa	Período (s)
SCADA/BD/AR	Tempo Real	Transferência da Base de Dados	$\mathcal{F}_1$	10
Análise de Rede	Tempo Real	Configurador	$\mathcal{F}_2$	10
		Observador + Estimador	$\mathcal{F}_3$	10
		Observador + Estimador + Fluxo de Potência *On-line*	$\mathcal{F}_4$	20
		Análise de Contingências	$\mathcal{F}_5$	40
	Estudo	Fluxo de Potência *Off-line* + Análise de Contingências	$\mathcal{F}_6$	-

5.3 Testes e Resultados

Nesta seção são apresentados os testes realizados com os modelos de escalonamento nos diversos ambientes anteriormente mencionados. Para cada teste são apresentadas as tabelas com os piores tempos de execução das funções obtidos em modo *off-line*, as execuções das funções de análise de rede ao longo do tempo, quando ativadas pelos escalonadores, e os gráficos de utilização do processador.

5.3.1 Teste I

O Teste I compreende apenas as funções do modo Tempo Real. São apresentados os resultados das execução em tempo real dos modelos de escalonamento Taxa Monotônica e Próximo Prazo para todos os três ambientes.

Ambiente A

A Tabela 5.3 mostra os piores tempos de execução do conjunto de funções de análise de rede quando executadas no ambiente A.

Tabela 5.3: Tempos de Execução do Conjunto de Funções de Análise de Rede para o Ambiente A

Função	Tempo de Execução (s)	Período	Alocação do Processador (%)
$\mathcal{F}_1$	1,3	10	13,00
$\mathcal{F}_2$	1,2	10	12,00
$\mathcal{F}_3$	1,0	10	10,00
$\mathcal{F}_4$	2,2	20	11,00
$\mathcal{F}_5$	1,6	40	04,00
		Total	50,00

Escalonadores TM e PP

Para o conjunto de funções da Tabela 5.3, a operação dos escalonadores Taxa Monotônica e Próximo Prazo é ilustrada na Figura 5.1. A Figura 5.2 mostra o gráfico de utilização do processador durante a execução das funções de análise de rede em uma janela de tempo de 2 minutos.

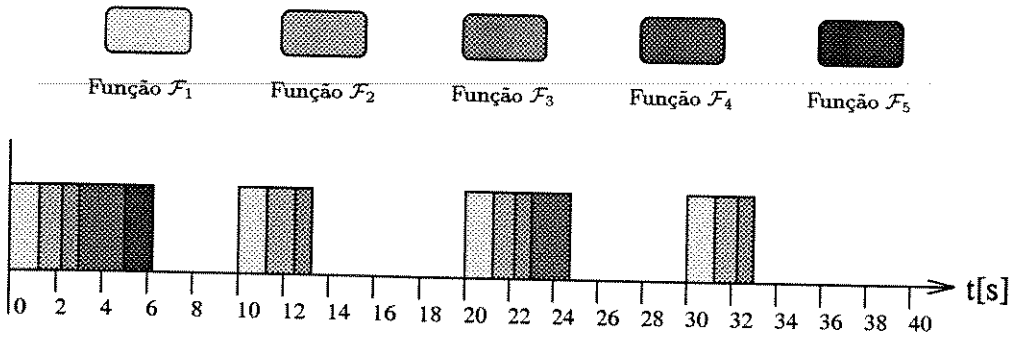

Figura 5.1: Operação dos Escalonadores TM e PP no Ambiente A Durante o Teste I

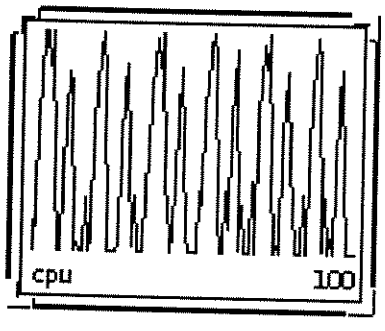

Figura 5.2: Utilização do Processador do Ambiente A Durante o Teste I para os Escalonadores TM e PP

Para este caso, com os dois escalonadores, a execução das funções de análise de rede é feita na mesma ordem. No instante inicial, todas as funções entram em estado de pronto. A primeira a ganhar o uso do processador é $\mathcal{F}_1$, pois é ela que tem o menor período, caso do escalonador TM, ou o próximo prazo, caso do escalonador PP, e é a primeira da seqüência lógica das funções de análise de rede. Seguem-se a ela as funções $\mathcal{F}_2$ e $\mathcal{F}_3$. No instante $t = 3s$, a função $\mathcal{F}_3$ termina sua execução e o escalonador concede o uso do processador à função $\mathcal{F}_4$. O escalonador TM o faz porque ela é a de menor período em estado de pronto ($P_4 = 20s$). O escalonador PP, porque ela é a que tem o primeiro prazo a vencer ($t = 20s$). Ao término de sua execução, a concessão do uso do processador é passada à última função em estado de pronto, a função $\mathcal{F}_5$.

No instante $t = 10s$, chegam novos dados telemedidos e as funções $\mathcal{F}_1$, $\mathcal{F}_2$ e $\mathcal{F}_3$ têm seus prazos vencidos, retornando à fila de pronto. Elas são escalonadas nesta mesma ordem, de acordo com o critério de desempate da seqüência lógica de execução de tarefas que se refiram à uma mesma varredura do subsistema de aquisição de dados (SCADA), com a última em estado de pronto terminando sua execução em $t = 13,2s$. No instante $t = 20s$, elas retornam à fila de pronto junto com a função $\mathcal{F}_4$, que é a última a executar por ser a de maior período (Escalonador TM) ou a de último prazo (Escalonador PP). Ela termina sua execução ainda antes das novas instâncias das três primeiras funções, que são executadas com os novos dados telemedidos a partir do instante $t = 30s$. No instante $t = 40s$, vence o prazo de todas as cinco funções e, então, elas retornam à fila de pronto para a análise dos novos dados, reiniciando o ciclo.

Ambiente B

Os tempos de execução máximos das funções de análise de rede para o ambiente B são mostrados na Tabela 5.4, assim como a alocação do processador.

Tabela 5.4: Tempos de Execução do Conjunto de Funções de Análise de Rede para o Ambiente B

Função	Tempo de Execução (s)	Período	Alocação do Processador (%)
$\mathcal{F}_1$	1,6	10	16,00
$\mathcal{F}_2$	1,5	10	15,00
$\mathcal{F}_3$	1,4	10	14,00
$\mathcal{F}_4$	3,6	20	18,00
$\mathcal{F}_5$	2,3	40	05,75
		Total	68,75

Escalonadores TM e PP

A Figura 5.3 ilustra a operação dos escalonadores TM e PP quando solicitado o escalonamento do conjunto de funções de análise de rede descritos na Tabela 5.4. O uso do processador pelas funções, durante três ciclos inteiros (2 minutos) de suas instâncias, é exibido na Figura 5.4.

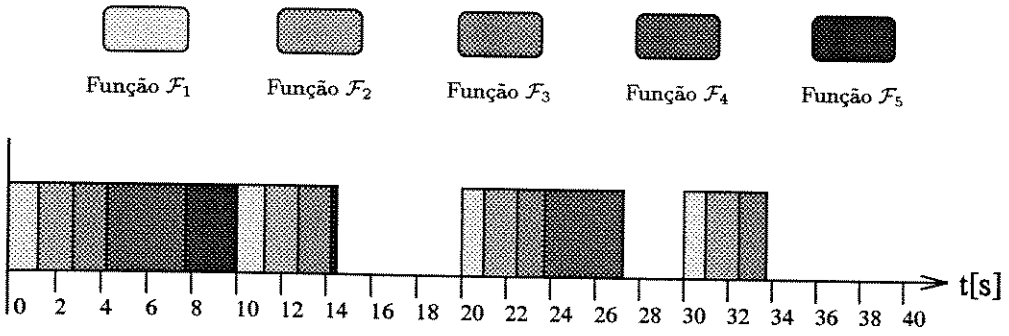

Figura 5.3: Operação dos Escalonadores TM e PP no Ambiente B Durante o Teste I

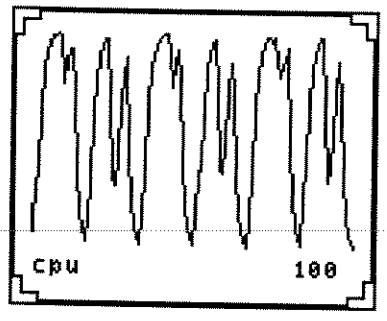

Figura 5.4: Utilização do Processador do Ambiente B Durante o Teste I para os Escalonadores TM e PP

Novamente, a ordem de execução em tempo real das funções de análise de rede determinada pelos dois escalonadores é idêntica. No instante inicial, os dois escalonadores concedem o uso do processador à função $\mathcal{F}_1$. No caso do escalonador TM, ela é a de maior prioridade porque possui o menor período e está à frente na seqüência lógica de execução. No caso do modelo PP, a prioridade mais alta lhe é concedida, para este instante, por apresentar o primeiro prazo a ser vencido. Após sua execução, as funções $\mathcal{F}_2$, $\mathcal{F}_3$, $\mathcal{F}_4$ e $\mathcal{F}_5$, nesta ordem, ganham a "posse" do processador, de acordo com suas prioridades. No instante $t = 10s$, antes do término da função $\mathcal{F}_5$, as funções $\mathcal{F}_1$, $\mathcal{F}_2$ e $\mathcal{F}_3$ entram novamente em estado de pronto e o escalonador (TM ou PP) realiza a suspensão da primeira instância daquela. Quando a função $\mathcal{F}_3$ termina em $t = 14,1s$, a função $\mathcal{F}_5$ é colocada novamente no estado de ativa para finalizar a execução de sua primeira instância. Neste instante $t = 14,6s$, o processador fica ocioso, até a chegada da nova varredura do sistema de aquisição de dados, quando, então, as quatro primeiras funções tornam a ficar em estado de prontas, sendo escalonadas em $t = 20s$, $t = 21s$, $t = 22,4s$ e $t = 23,8s$, respectivamente. Em $t = 30$, mais uma instância da função $\mathcal{F}_1$ é executada, seguida das funções $\mathcal{F}_2$ e $\mathcal{F}_3$. Desta forma,

todas as funções fecham o ciclo cumprindo seus prazos, tanto com o escalonador Taxa Monotônica como com o Próximo Prazo.

Ambiente C

A Tabela 5.5 mostra os piores tempos de execução do conjunto de funções de análise de rede quando executadas no ambiente C.

Tabela 5.5: Tempos de Execução do Conjunto de Funções de Análise de Rede para o Ambiente C

Função	Tempo de Execução (s)	Período	Alocação do Processador (%)
$\mathcal{F}_1$	2,1	10	21,00
$\mathcal{F}_2$	2,4	10	24,00
$\mathcal{F}_3$	2,2	10	22,00
$\mathcal{F}_4$	5,6	20	28,00
$\mathcal{F}_5$	3,6	40	09,00
		Total	104,00

Escalonadores TM e PP

Para o conjunto de funções da Tabela 5.5 é impossível o escalonamento em tempo real, uma vez que o fator de utilização do processador excede o limiar de 100%. Neste caso, para a realização do escalonamento das funções de análise de rede, neste ambiente, é necessário aumentar o período de execução de alguma das funções. Isto pode ser feito, por exemplo, com as funções de fluxo de potência e análise de contingências, sem que se tenha prejuízo apreciável com a perda de informações de tempo real.

A operação dos escalonadores Taxa Monotônica e Próximo Prazo, até o instante em que a primeira função de análise de rede perde seu prazo, é ilustrada na Figura 5.5. Neste caso, não há interesse na seqüência do escalonamento, uma vez que não há como se evitar as sucessivas perdas de prazos. A Figura 5.6 mostra o gráfico de utilização do processador, do instante inicial até a primeira perda de prazo em $t = 40s$, em uma janela de tempo de 2 minutos.

Comparando-se os valores dos piores tempos de execução de cada instância das funções de análise de rede no modo *off-line* com os seus tempos de execução *on-line* em qualquer um dos três ambientes, pode-se afirmar que a parcela de tempo do sistema operacional mais o tempo de execução dos módulos escalonadores (*overhead* de escalonamento), no caso de somente funções periódicas, não é relevante e, portanto, pode ser negligenciado.

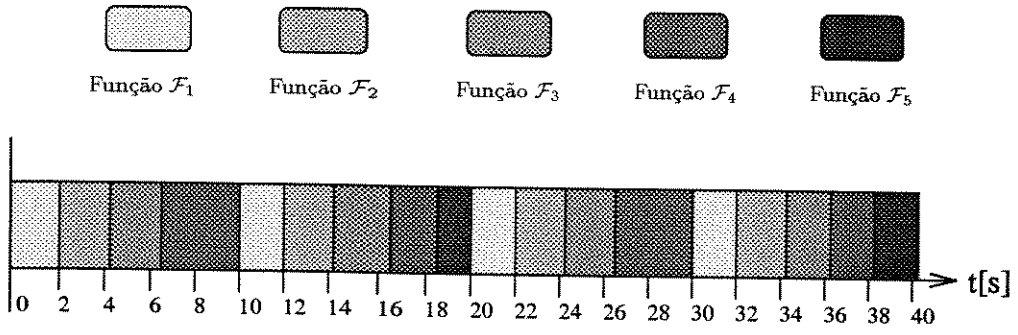

Figura 5.5: Operação dos Escalonadores TM e PP no Ambiente C Durante o Teste I

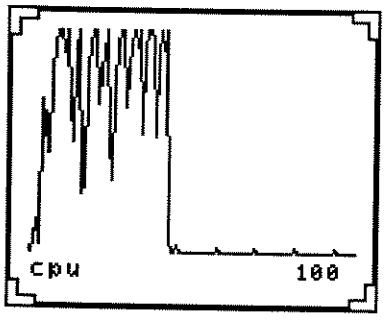

Figura 5.6: Utilização do Processador do Ambiente C Durante o Teste I para os Escalonadores TM e PP

5.3.2 Teste II

O Teste II compreende tanto as funções do modo Tempo Real como a do modo Estudo. Durante o escalonamento das funções do modo Tempo Real, é requisitada a execução da função de análise de contingências, para o estudo de uma lista de contingências no sistema.

O escalonamento dos aplicativos de análise de rede foi realizado nos ambientes A e B com os dois escalonadores. A operação de cada um deles, com os respectivos gráficos de utilização do processador, é mostrada nas seções seguintes.

Ambiente A

Os piores tempos de execução do conjunto de funções de análise de rede para o ambiente A utilizadas no Teste II são os mesmos mostrados na Tabela 5.3. O tempo de execução de uma instância da função de fluxo de carga *off-line* com análise das contingências escolhidas, para o ambiente A, é de 3,0 segundos. São solicitadas duas instâncias da função $\mathcal{F}_6$, a primeira no instante $t = 1s$ e a segunda em $t = 25s$.

Escalonador TM com SE

O período atribuído ao servidor esporádico é igual ao menor período dentre os das funções de análise de rede, isto é, 10 segundos. Isto garante ao servidor o maior tempo de execução por período possível para a situação em que ele possua a maior prioridade do sistema. Assim, como o fator de utilização do processador pelo conjunto de funções do modo Tempo Real é de 50%, pode-se alocar os outros 50% à utilização pelo servidor esporádico (limiar de 100%), o que lhe confere uma capacidade (carga) de 5,0 segundos ($\lfloor 10 segundos \times 50\% \rfloor$).

A Figura 5.7 ilustra a operação do escalonador Taxa Monotônica com Servidor Esporádico no atendimento dos prazos das funções do modo Tempo Real e no atendimento das requisições do modo Estudo. O gráfico de utilização do processador é ilustrado na Figura 5.8. O ciclo central se refere à situação da Figura 5.7.

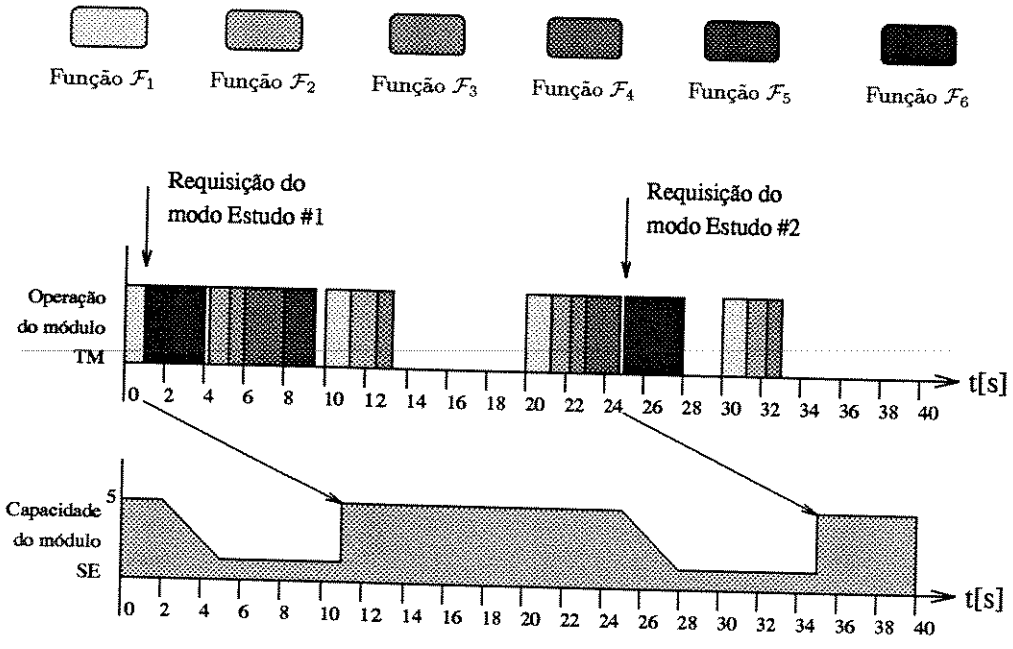

Figura 5.7: Operação do Escalonador TM com SE no Ambiente A Durante o Teste II

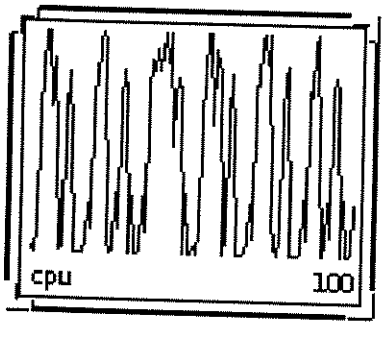

Figura 5.8: Utilização do Processador do Ambiente A Durante o Teste II para o Escalonador TM com SE

Escalonador PP

A operação do escalonador Próximo Prazo e a utilização do processador da máquina do ambiente A, quando solicitado para escalonamento das funções dos modos Tempo Real e Estudo, são mostrados nas Figuras 5.9 e 5.10, respectivamente.

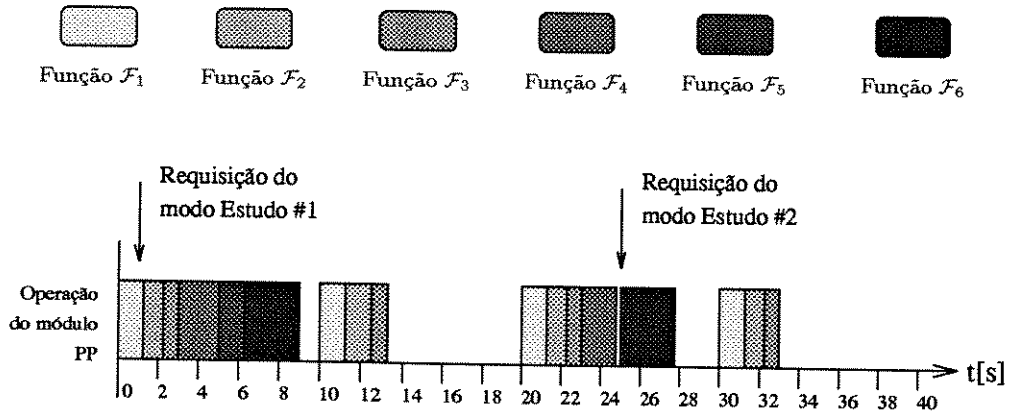

Figura 5.9: Operação do Escalonador PP no Ambiente A Durante o Teste II

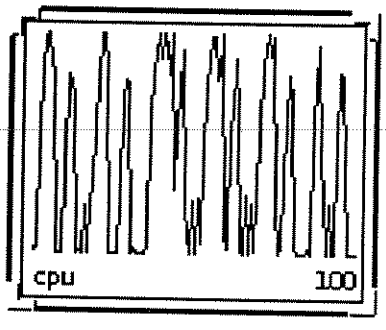

Figura 5.10: Utilização do Processador do Ambiente A Durante o Teste II para o Escalonador PP

Ambiente B

Para o ambiente B, os piores tempos de execução do conjunto de funções de análise de rede utilizados neste teste são os mesmos mostrados na Tabela 5.4 do Teste I. O tempo de execução de cada instância da função do modo Estudo, neste ambiente, é de 5,0 segundos.

Escalonador TM com SE

Assim como no caso do teste no Ambiente A, o período atribuído ao servidor esporádico é igual ao menor período dentre os das funções de análise de rede, que aqui também é 10 segundos, de forma a garantir ao servidor o maior tempo de execução por período possível para a situação em que ele possua a maior

prioridade do sistema. Como para este caso o fator de utilização do processador pelo conjunto de funções do modo Tempo Real é de 68,75%, pode-se alocar 31,25% à utilização pelo servidor esporádico (limiar de 100%). Desta forma, o servidor disporá da carga (capacidade) de 3,0 segundos ($\lfloor 10 segundos \times 31,25\% \rfloor$).

A operação do escalonador Taxa Monotônica com Servidor Esporádico para o escalonamento das funções do modo Tempo Real dentro de seus prazos e no atendimento à requisição do modo Estudo é apresentada na Figura 5.11. O gráfico de utilização do processador é mostrado na Figura 5.12, onde a situação ilustrada na Figura 5.11 corresponde ao ciclo central.

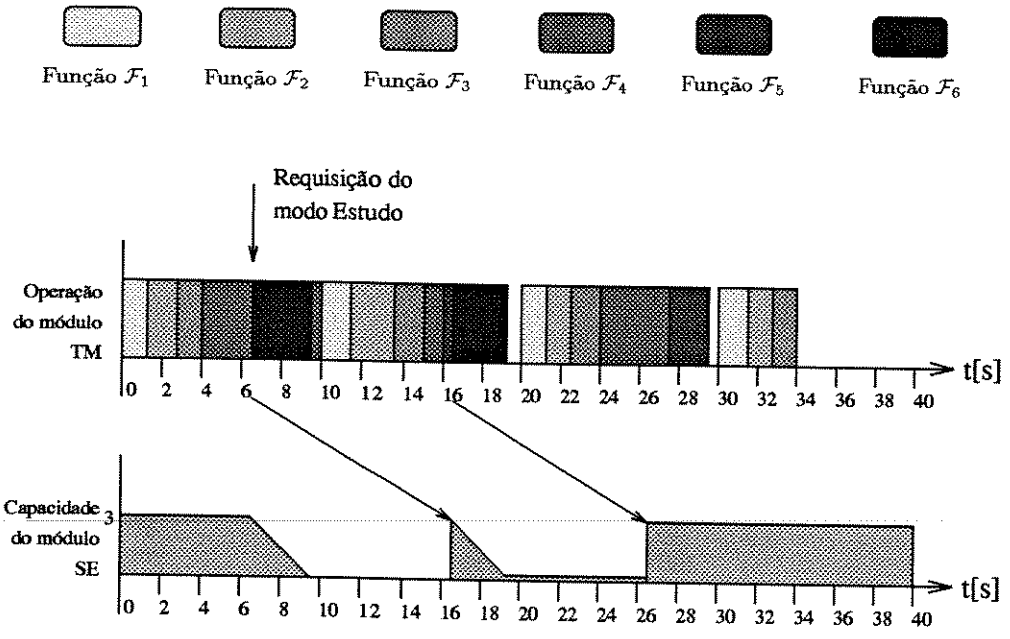

Figura 5.11: Operação do Escalonador TM com SE no Ambiente B Durante o Teste II

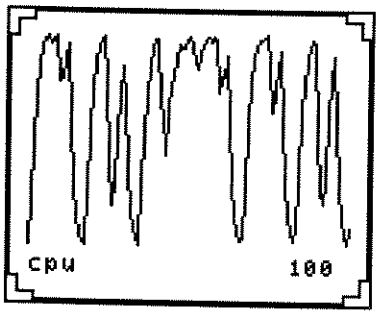

Figura 5.12: Utilização do Processador do Ambiente B Durante o Teste II para o Escalonador TM com SE

Escalonador PP

A Figura 5.13 mostra a operação do escalonador Próximo Prazo no atendimento dos prazos das funções do modo Tempo Real e da requisição do modo Estudo. A Figura 5.14 mostra o gráfico de utilização do processador.

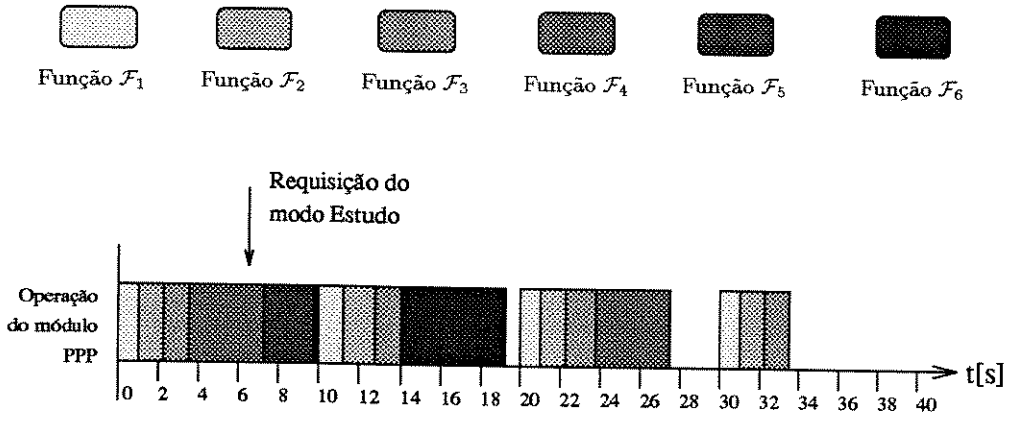

Figura 5.13: Operação do Escalonador PP no Ambiente B Durante o Teste II

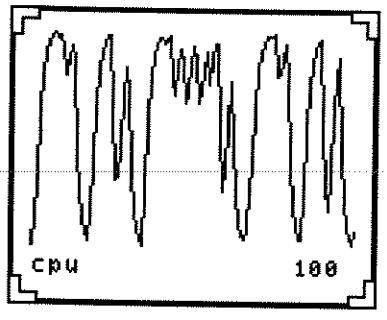

Figura 5.14: Utilização do Processador do Ambiente B Durante o Teste II para o Escalonador PP

A partir da análise da operação dos escalonadores Taxa Monotônica com Servidor Esporádico e Próximo Prazo, pode-se observar que:

- apesar dos tempos de execução *off-line* das funções periódicas não variarem quando escalonadas em tempo real, para as solicitações aperiódicas, há um acréscimo no tempo de execução *on-line*. Isto acontece, principalmente, com o escalonador Taxa Monotônica, devido à necessidade de notificação dos eventos entres os módulos escalonador, servidor e temporizador;
- com o modelo TM com SE, os tempos de resposta às solicitações do modo Estudo, são sempre, no máximo, iguais aos tempos obtidos com o escalonador PP. Isto porque, quando sua capacidade de atendimento não é suficiente para executar a função aperiódica de forma imediata, ele ainda assim poderá sempre dispor do tempo ocioso do processador, o mesmo utilizado pelo modelo Próximo Prazo.

- o período do servidor de funções do modo Estudo deve ser escolhido de forma que sua capacidade de atendimento seja compatível com o tempo de execução das mesmas. Caso contrário, pode ocorrer um atraso desnecessário no atendimento das solicitações aperiódicas.

5.3.3 Teste III

Durante o Teste III, foram usadas as mesmas funções do modo Tempo Real descritas anteriormente. No ambiente A, no entanto, elas tiveram seus períodos de execução alterados para obter um fator de utilização do processador igual a 100%. Os novos períodos de cada uma delas são mostrados na Tabela 5.6. O teste, no ambiente A, foi realizado com o modelo Taxa Monotônica.

No ambiente B, aplicou-se o modelo Próximo Prazo, escalonando, além das funções do modo Tempo Real com seus períodos originais, uma sexta função para o modo Estudo, com característica de processamento intensivo e com tempo execução suficientemente grande para a análise.

Ambiente A

A Tabela 5.6 mostra os piores tempos de execução e os períodos do conjunto de funções de análise de rede usados no Teste III quando executadas no ambiente A.

Tabela 5.6: Tempos de Execução e Períodos do Conjunto de Funções de Análise de Rede usadas no Teste III para o Ambiente A

Função	Tempo de Execução (s)	Período	Alocação do Processador (%)
$\mathcal{F}_1$	1,3	05	26,00
$\mathcal{F}_2$	1,2	05	24,00
$\mathcal{F}_3$	1,0	05	20,00
$\mathcal{F}_4$	2,2	10	22,00
$\mathcal{F}_5$	1,6	20	08,00
		Total	100,00

Escalonador TM

Para o conjunto de funções da Tabela 5.6, a operação dos escalonador Taxa Monotônica é ilustrada na Figura 5.15. A Figura 5.16 mostra o gráfico de utilização do processador durante a execução das funções de análise de rede em seis ciclos completos.

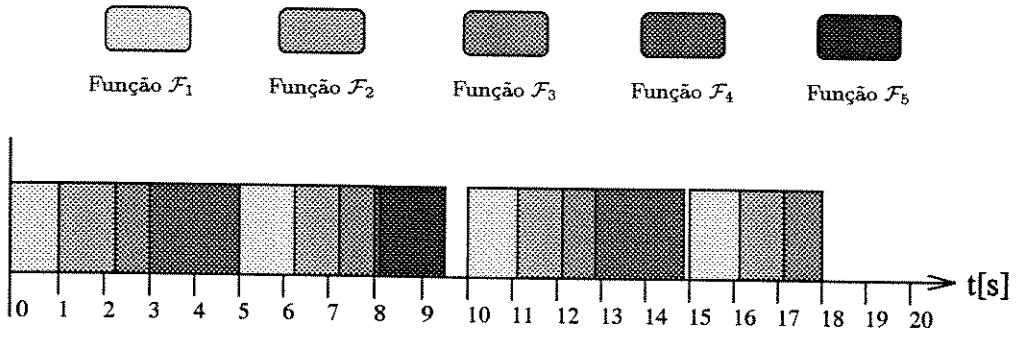

Figura 5.15: Operação do Escalonador TM no Ambiente A Durante o Teste III

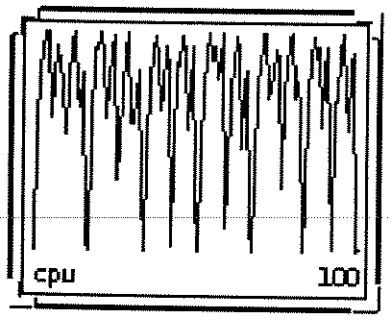

Figura 5.16: Utilização do Processador do Ambiente A Durante o Teste III para o Escalonador TM

Ambiente B

Os tempos de execução máximos das funções de análise de rede para o ambiente B são mostrados na Tabela 5.4. A função do modo Estudo tem tempo de execução ilimitado.

Escalonador PP

A Figura 5.17 ilustra a operação do escalonador PP quando solicitado o escalonamento do conjunto de funções de análise de rede descrito na Tabela 5.4 e a função do modo Estudo. O uso do processador pelas funções, durante três ciclos inteiros (2 minutos) de suas instâncias, é exibido na Figura 5.18. Os "vales" correspondem aos instantes de execução da tarefa de transferência da base de dados.

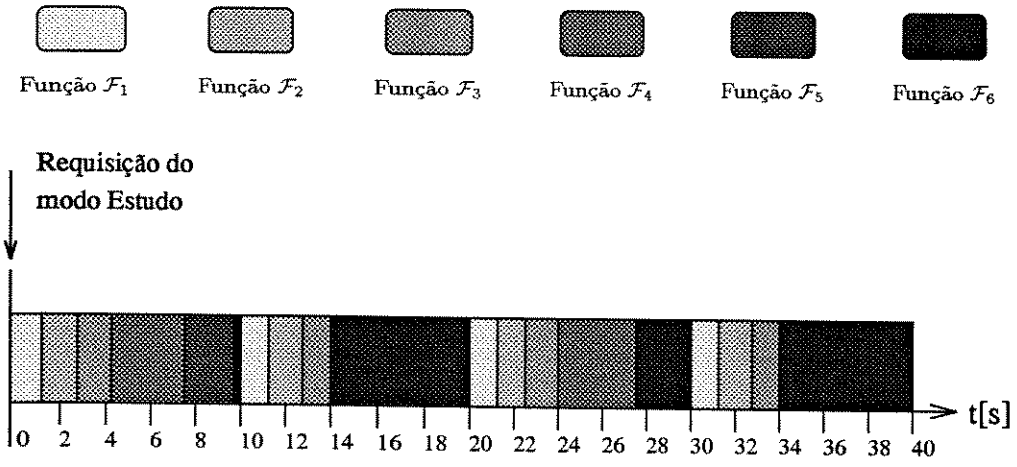

Figura 5.17: Operação do Escalonador PP no Ambiente B Durante o Teste III

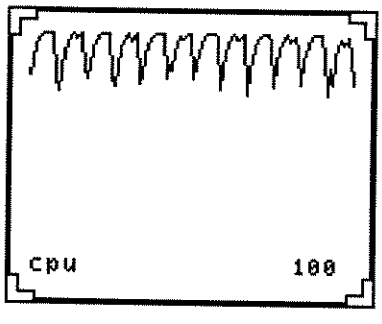

Figura 5.18: Utilização do Processador do Ambiente B Durante o Teste III para o Escalonador PP

O teste com o escalonador TM mostra que, uma vez considerados os piores tempos de execução de cada tarefa, é possível uma completa alocação do processador às funções de análise de rede, devido aos baixos tempos de execução do módulo de escalonamento e de sistema apresentado pelo UNIX. É importante ressaltar que a utilização de 100% do processador só é possível com o algoritmo Taxa Monotônica, devido à particularidade das funções de análise de rede de apresentarem períodos de execução sempre múltiplos. O teste com o escalonador PP vem confirmar também a viabilidade de execução das funções de análise de rede com garantia de cumprimento dos prazos, mesmo com a completa utilização do processador devido a uma requisição aperiódica.

Capítulo 6

Comentários Finais

Com o crescimento dos sistemas elétricos de potência e o conseqüente aumento de sua complexidade de operação, as vantagens advindas das funções de análise de rede têm-se tornado cada vez mais evidentes. Com isto, o esforço no desenvolvimento de tais funções para a aplicação em tempo real vem recebendo especial atenção dos profissionais da área. Hoje, com as novas funções avançadas e com o *hardware* disponível, já se é possível a execução do subsistema de análise de rede concomitantemente aos demais subsistemas que compõem os Centros de Controle de Energia.

Além disso, a partir do conhecimento das técnicas atuais de escalonamento de funções em tempo real, já é possível também afirmar que as funções do subsistemas de análise de rede, com suas características de tempos de execução relativamente constantes, periódicas ou não, podem ser executadas de forma a se garantir o cumprimento de seus prazos.

Com o estudo realizado nos capítulos anteriores, adicionalmente ao acima exposto, pôde-se concluir a perfeita viabilidade da execução do escalonamento de tais funções de forma simples e confiável, com apenas a utilização de ferramentas, como sistema operacional e equipamento, que podem ser facilmente colocadas à disposição em Centros de Controle. Isto é, uma implementação tanto do algoritmo Taxa Monotônica como do Próximo Prazo, sobre o sistema operacional UNIX, é possível e apresenta resultados satisfatórios de eficiência. Isto tem a sua importância, principalmente, dada a grande dificuldade de alocação de computadores, dedicados aos subsistemas de análise de rede, que suportem núcleos de tempo real à arquitetura computacional comumente encontrada nos atuais Centros de Controle dos sistemas de potência.

Após a realização de vários testes com os dois modelos de escalonadores propostos, alguns deles apresentados no capítulo anterior, observou-se, também, que:

- quanto ao problema de escalonamento das funções de análise de rede em tempo real, pode-se dizer que se trata de um problema "bem comportado". Isto é, no que se refere às funções de análise de rede, uma vez que elas apresentam características simples, como:

  - tempos de execução relativamente constantes,
  - períodos de execução sempre múltiplos uns dos outros,
  - tempos de pronto sempre iguais ao tempo de chegada e, principalmente,
  - uma seqüência lógica de execução que acompanha o crescimento de seus períodos,

  há sempre um ciclo de execução das mesmas, bem definido, formado por todas as instâncias das funções dentro do período daquela de menor freqüência;

- devido a estas mesmas características (períodos múltiplos e o mesmo critério de desempate segundo a seqüência lógica de execução), os escalonadores Taxa Monotônica e Próximo Prazo, no caso das funções de análise de rede, não apresentam diferença de operação na ordem de escalonamento quando há apenas funções do modo Tempo Real;

- os tempos de respostas às solicitações do modo Estudo, em algumas situações, podem variar sensivelmente com a utilização do Taxa Monotônica com Servidor Esporádico ou do Próximo Prazo. De acordo com o tempo de execução médio de uma destas solicitações e da capacidade de atendimento que se pode conceder ao Servidor Esporádico, o modelo Taxa Monotônica quase sempre apresenta atendimento mais rápido;

- em uma análise mais profunda, no entanto, em um momento de sobrecarga do subsistema de análise de rede, ocasionado:

  - pela necessidade de reconfiguração de várias subestações,
  - pela detecção, por parte do estimador de estado, de uma lista de erros grosseiros muito extensa
  - pela geração de um número elevado de ilhas observáveis para uma mesma varredura do subsistema de aquisição de dados ou
  - por algum problema de convergência do fluxo de potência *on-line*,

  é possível que, diante de uma solicitação do modo Estudo, haja a perda do prazo de alguma das funções no caso do algoritmo Taxa Monotônica, que não aconteceria com o Próximo Prazo. Para contemplar tal situação, tornar-se-ia necessária a implementação de uma de suas extensões, conforme mencionado no final do Capítulo 3.

- por necessitar de um módulo servidor para as funções do modo Estudo, o algoritmo Taxa Monotônica exigiu um maior esforço de implementação em

relação ao algoritmo Próximo Prazo, ainda que isto não o desabone. A maior complexidade advém da necessidade de uma constante notificação de eventos pertinentes entre os módulos e da monitoração da capacidade de serviço do módulo servidor, que não existe no caso do escalonador Próximo Prazo.

- apesar de não apresentar facilidades para a execução de processos de tempo real (caso do *SunOS Release 4.1.1*), o UNIX apresentou tempos de sistema (*overhead* com as trocas de contexto devido às preempções das funções) muito pequenos. Estes, aliados aos baixos tempos de execução dos módulos de escalonamento (conseqüência da simplicidade dos algoritmos Taxa Monotônica e Próximo Prazo) puderam proporcionar uma completa alocação dos recursos do sistema às funções de análise de rede, quando considerados seus piores tempos de execução.

6.1 Proposta para Futuros Trabalhos

O modelo de implementação proposto é capaz de realizar o atendimento seqüencial das requisições do modo Estudo. Isto é, as requisições de estudo do operador do sistema são atendidas, uma a uma, na ordem em que são solicitadas. A extensão para o caso do atendimento multitarefa das solicitações do operador pode ser de grande valor em alguns Centros de Controle. Além disso, no caso do escalonador Taxa Monotônica, dependendo do interesse da aplicação, pode-se estender o escalonador para as situações descritas no Capítulo 3, como:

- assegurar o cumprimento de restrições de tempo das tarefas mais importantes do sistema, mesmo em condições de sobrecarga transitória do mesmo,
- escalonar tarefas onde "computação imprecisa" é permitida, como pode ser o caso da função de análise de contingências, e
- permitir a inclusão e exclusão de tarefas durante a execução do sistema e a alteração de parâmetros de funções.

Vislumbra-se que no futuro toda a supervisão dos Centros de Controle será alimentada a partir de dados oriundos do subsistema de análise de rede, tornando imprescindível que tal aplicação seja tratada como crítica, isto é, executada em processadores redundantes para que seja tolerante a falhas.

Outra linha de pesquisa de relevante interesse é o escalonamento das funções de análise de redes quando se utiliza processamento distribuído e/ou paralelo. A teoria de sistemas de tempo real distribuído, embora ainda seja um tema em aberto [Sta 95], tem avançado nos últimos anos e sua aplicação às funções do subsistema de análise de rede podem trazer benefícios ainda mais sensíveis à operação do sistema elétrico.

Bibliografia

[Als 74] ALSAÇ, Ongun, STOTT, Brian. " Optimal Load Flow with Steady-State Security ". *IEEE Transactions on Power Apparatus and Systems*, V. 93, pp. 745-751, May/June 1974.

[Amel83] AMELINK, H., Hoffmann, A. G.. "Current Trends in Control Centre Design". *International Journal on Electrical Power & Energy Systems*, V. 5, No. 4, pp. 205-211, Oct. 1983.

[Amer89] AMERONGEN, Robert A. M. van. " A General-Purpose Version of the Fast Decoupled Load Flow ". *IEEE Transactions on Power Systems*, V. 4, No. 2, pp. 760-770, May 1989.

[Bac 86] BACH, M. J.. "The Design of the Unix Operating System". Englewood Cliffs, New Jersey: *Prentice Hall*, 1986.

[Bal 92] BALU, Neal et al.. " On-Line Power System Security Analysis ". *Proceedings of the IEEE*, V. 80, No. 2, pp. 262-280, Feb. 1992.

[Ben 90] BEN-ARI, M. "Principles of Concurrent and Distributed Programming". [Hemel Hempstead, UK]: Prentice Hall International, 1990.

[CEP 78] CEPEL. "Funções de Monitoração e Controle em Tempo Real de Sistemas Elétricos de Potência", IEEE TELECOM. Rio de Janeiro: CEPEL, set., 1978. Material de curso.

[Chu 90] CHUNG, J. Y., LIU, J. W. S., LIN, K. J.. "Scheduling Periodic Jobs that Allow Imprecise Results". *IEEE Transactions on Computers*, V. 39, No. 9, pp. 1156-1174, Sept. 1990.

[Dan 85] DANIELS, Howard A., Mayur, Neelal. "More than Mainframes". *IEEE Spectrum*, pp. 54-61, Aug. 1985.

[Deb 88] DEBS, Atif S.. "Modern Power Systems Control and Operation". Norwell, Massachusetts: *Kluwer Academic*, 1988.

[Dij 68] DIJKSTRA, E. W.. "Co-operating Sequential Processes". In F. Genuys (ed.), *Programming Languages*. New York: Academic Press, 1968.

[Eva 89] EVANS, Jerry W.. "Survey of Energy Management System Architectures". *IEEE Computer Applications in Power*, pp. 11-16, Jan. 1989.

[Fre 92] FREIRE, Luciano M., GARCIA, Ariovaldo V., MONTICELLI, Alcir J.. "Modernização Incremental do Centro de Operação do Sistema da CPFL". In: *Simpósio de Automação de Sistemas Elétricos*, 1., 1992, Campinas, SP. Anais ... Campinas: CPFL, 1992. p. 318-328.

[Fre 94a] FREIRE, Luciano M., GONÇALVES, Paulo C.. "FAR - Funções de Análise de Rede". In: *ERDOS - Encontro Regional de Despachantes de Operação do Sistema*, 2., 1994, Ibitinga, SP. Anais ... Campinas: CPFL, 1994.

[Fre 94b] FREIRE, Luciano M. et al.. " Integração e Avaliação das Funções de Análise de Redes do Centro de Controle da CPFL ". In: *Simpósio de Automação de Sistemas Elétricos*, 2., 1994, Belo Horizonte, MG. Anais ... Belo Horizonte: CEMIG, 1994.

[Gau 87] GAUSHELL, Dennis J., DARLINGTON, Henry T.. "Supervisory Control and Data Acquisition". *Proceedings of the IEEE*, V. 75, No. 12, pp. 1645-1658, Dec. 1987.

[Goo 88] GOODENOUGH, John B., SHA, Lui. "The Priority Ceiling Protocol: A Method for Minimizing the Blocking of High Priority Ada Tasks". In: *International Workshop on Real-Time Ada Issues*, 2., 1988, Moretonhampstead. (ACM Ada Letters, V. 8, No. 7, 1988).

[Hor 87] HORTON, Jerry S., GROSS, David P.. "Computer Configurations". *Proceedings of the IEEE*, V. 75, No. 12, pp. 1659-1669, Dec. 1987.

[IEEE77] IEEE Tutorial Course. " Energy Control Center Design ". [New York]: IEEE, 1977. Apresentado no Summer Meeting, July 17-22, 1977, Cidade do México, México.

[Leh 87] LEHOCZKY, John P., SHA, Lui, STROSNIDER, J. K.. " Enhanced Aperiodic Responsiveness in Hard Real-Time Environments ". In: *IEEE Real-Time Systems Symposium*, 1987. Proceedings... [S.l.]: IEEE, 1987. pp. 261-270.

[Leh 89] LEHOCZKY, John P., SHA, Lui, DING, Ye. "The Rate Monotonic Scheduling Algorithm: Exact Characterization And Average Case Behavior". In: *IEEE Real-Time Systems Symposium*, 1989. Proceedings... [S.l.]: IEEE, 1989. pp. 166-171.

[Liu 73] LIU, C. L., LAYLAND, James W.. "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment". *Journal of the Association for Computing Machinery*, V. 20, No. 1, pp. 46-61, Jan. 1973.

[Liu 87] LIU, J. W. S., LIN, K. J., NATARAJAN, S.. "Scheduling Real-Time Periodic Jobs Using Imprecise Results". In: *IEEE Real-Time Systems Symposium*, 1987. Proceedings... [S.l.]: IEEE, 1987. pp. 252-260.

[Mag 86] MAGALHÃES, Maurício F.. "Software para Tempo Real". Campinas: *UNICAMP*, 1986.

[Mel 93] MELO Jr., B. Alencar de. " Uma Estratégia de Escalonamento de Processos Periódicos e Esporádicos em Sistemas de Tempo Real Crítico Monoprocessados ". Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas (FEE/UNICAMP). Campinas, SP, jan. 1993.

[Mon 79] MONTICELLI, Alcir J. et al.. "Real-Time External Equivalents for Static Security Analysis". *IEEE Transactions on Power Apparatus and Systems*, V. 98, No. 2, pp. 498-508, Mar./Apr. 1979.

[Mon 83] MONTICELLI, Alcir J.. "Fluxo de Carga em Redes de Energia Elétrica". São Paulo: *Edgard Blücher*, 1983.

[Mon 90] MONTICELLI, Alcir J., GARCIA, Ariovaldo V., SAAVEDRA, Osvaldo R.. "Fast Decoupled Load Flow: Hypothesis, Derivations, and Testing". *IEEE Transactions on Power Systems*, V. 5, No. 4, pp. 1425-1431, Nov. 1990.

[Mon 92] MONTICELLI, Alcir J., GARCIA, Ariovaldo V., SLUTSKER, I. W.. "Handling Discardable Measurements in Power System State Estimation". *IEEE Transactions on Power Systems*, V. 7, No. 3, pp. 1333-1340, Aug. 1992.

[Oli 93] OLIVEIRA, Mario E., PITTA, Ronaldo L.. " Análise de Redes no Novo Centro de Supervisão e Controle de FURNAS: Experiência de Implantação e Perspectivas Futuras ". In: *EDAO - Encontro para Debates de Assuntos da Operação*, 4., 1993, São Paulo, SP. Anais ... São Paulo: CESP, 1993.

[Pod 93] PODMORE, Robin. "Criteria for Evaluating Open Energy Management Systems". *IEEE Transactions on Power Systems*, V. 8, No. 2, pp. 466-471, Mar. 1993.

[Raj 89] RAJKUMAR, R.. "Task Synchronization in Real-Time Systems". Thesis (Ph.D.) - Carnegie Mellon University (CMU). Pittsburgh, Pennsylvania, Aug. 1989.

[Rod 94] RODRIGUES, Marcelo, SAAVEDRA, Osvaldo R., MONTICELLI, Alcir J.. " Asynchronous Programming Model for the Concurrent Solutions of the Security Constrained Optimal Power Flow Problem ". *IEEE Transactions on Power Systems*, V. 9, No. 4, pp. 2021-2027, Nov. 1994.

[Rus 79] RUSSELL, Jerry C., MASIELLO, Ralph D., BOSE, Anjan. "Power System Control Center Concepts". In: *PICA - IEEE Power Industry Computer Applications Conference*, 11., 1979, Cleveland. Proceedings... [New York]: IEEE, 1979. pp. 170-176.

[Sha 86] SHA, Lui, LEHOCZKY, John P., RAJKUMAR, R.. "Solutions for Some Practical Problems in Prioritized Preemptive Scheduling". In: *IEEE Real-Time Systems Symposium*, 1986. Proceedings... [S.l.]: IEEE, 1986. pp. 181-191.

[Sha 87] SHA, Lui, RAJKUMAR, R., LEHOCZKY, John P.. "Priority Inheritance Protocols: An Approach to Real-Time Synchronization". Technical Report - Departament of Computer Science, Carnegie Mellon University. Pittsburgh, Pennsylvania, 1987.

[Sha 89] SHA, Lui et al.. "Mode Change Protocols for Priority-Driven Preemptive Scheduling". *The Journal of Real-Time Systems*, V. 1, pp. 243-264, 1989.

[Sha 90] SHA, Lui, GOODENOUGH, John B.. "Real-Time Scheduling Theory and Ada". *Computer Magazine, IEEE Computer Society*, V. 23, No. 4, pp. 53-62, Apr. 1990.

[Sha 91] SHA, Lui, KLEIN, Mark H., GOODENOUGH, John B.. "Rate Monotonic Analysis for Real-Time Systems". Technical Report - Software Engineering Institute, Carnegie Mellon University (CMU/SEI). Pittsburgh, Pennsylvania, Mar. 1991.

[Spr 88] SPRUNT, Brinkey, LEHOCZKY, John P, SHA, Lui.. " Exploiting Unused Periodic Time for Aperiodic Service Using the Extended Priority Exchange Algorithm ". In: *IEEE Real-Time Systems Symposium*, 1988. Proceedings... [S.l.]: IEEE, 1988. pp. 251-258.

[Spr 89] SPRUNT, Brinkey, SHA, Lui, LEHOCZKY, John P.. " Aperiodic Task Scheduling for Hard-Real-Time Systems ". *The Journal of Real-Time Systems*, V. 1, No. 1, pp. 27-60, 1989.

[Sta 95] STANKOVIC, John A., SPURI, Marco, NATALE, Marco Di, BUTTAZZO, Giorgio C.. " Implications of Classical Scheduling Results for Real-Time Systems ". *Computer Magazine, IEEE Computer Society*, V. 28, No. 6, pp. 16-25, June 1995.

[Ste 90] STEVENS, W. Richard. " UNIX Network Programming ". Englewood Cliffs, New Jersey: *Prentice Hall*, 1990.

[Ste 93] STEVENS, W. Richard. " Advanced Programming in the UNIX Environment ". Reading, Massachusetts: *Addison-Wesley*, 1993.

[Sto 74] STOTT, Brian, ALSAÇ, Ongun. "Fast Decoupled Load Flow". *IEEE Transactions on Power Apparatus and Systems*, V. 93, pp. 859-869, 1974.

[Sto 79] STOTT, Brian, MARINHO, J. L.. "Linear Programming for Power Systems Network Security Applications". *IEEE Transactions on Power Apparatus and Systems*, V. 98, No. 3, pp. 837-848, May/June 1979.

[Sto 87] STOTT, Brian, ALSAÇ, Ongun, MONTICELLI, Alcir J.. "Security Analysis and Optimization". *Proceedings of the IEEE*, V. 75, No. 12, pp. 1623-1644, Dec. 1987.

[Sun 90a] System Services Overview. *Sun Microsystems*, EUA, 1990.

[Sun 90b] SunOS Reference Manual. *Sun Microsystems*, EUA, 1990.

[Sun 90c] Programming Utilities & Libraries. *Sun Microsystems*, EUA, 1990.

[Tei 90] TEIXEIRA, M. J. et al.. " Uma Nova Geração de Centros de Controle de Energia ". In: *Congresso Brasileiro de Automação*, 8., 1990, Belém, PA. Anais ... Belém: UFPa, 1990. p. 397-403.

[Tin 72] TINNEY, W. F.. "Compensations Methods for Network Solutions by Optimally Ordered Triangular Factorization", *IEEE Transactions on Power Apparatus and Systems*, V. 91, pp. 123-127, Jan./Feb. 1972.

[Xu 90] XU, J., PARNAS, D. L.. "Scheduling Process with Release Times, Deadlines, Precedence, and Exclusion Relations". *IEEE Transactions on Software Engineering*, V. 16, No. 3, pp. 360-369, 1990.

[War 49] WARD, J. B.. " Equivalent Circuits for Power Flow Studies ". *AIEE Transactions*, V. 68, pp. 373-382, 1949.

[Was 91] WASLEY, Ronald G., STADLIN, Walter O.. "Network Applications in Energy Management Systems". *IEEE Computer Applications in Power*, V. 4, No. 1, pp. 31-36, Jan. 1991.

[Wol 85] WOLLENBERG, Bruce F., "Feasibility Study for an Energy Management System Intelligent Alarm Processor". In: *PICA - IEEE Power Industry Computer Applications Conference*, 14., 1985, San Francisco. Proceedings... [New York]: IEEE, 1985. pp. 249-254.

Abstract

This work presents two real-time schedulers for network analysis functions in energy management systems. The proposed schedulers handle functions such as network configuration, state estimation and on-line power flow, both in periodic mode as well as in asynchronous mode (interruption handling). The first model is a fixed priority scheduler, designed using the rate monotonic scheduling algorithm with a sporadic server, and the second one, based on the earliest deadline scheduling algorithm, is a dynamic scheduler. Time constraints and independence of the network functions were studied and then considered in the definition of the scheduler characteristics.

The schedulers implementations described in this work refer specifically to the network analysis functions running on a single processor in Unix workstations. Tests have been performed with part of the CPFL network (Campinas region) and the results are reported in the dissertation.

Keywords: *Network Analysis Functions, Real-Time Scheduling, Energy Management Systems.*

Apêndice A

Sinais do Sistema Operacional UNIX

Conforme mencionado no Capítulo 4, um sinal é uma notificação a um processo que um evento ocorreu. Eles podem ser enviados tanto de um processo a outro (ou a ele mesmo) como do núcleo do sistema operacional a um processo. Todo sinal no sistema operacional UNIX tem um nome, que é especificado no arquivo <signal.h>. Na Tabela A.1 são apresentados todos os sinais do UNIX, com seus nomes, sua descrição e sua ação padrão. Na coluna "Nota" é descrito se o sinal é específico à versão *4.3BSD* de Berkeley ou à versão *System V* da AT&T. Quando os sinais são comuns aos dois sistemas, a célula será apresentada em branco [Ste 90].

Os sinais podem ser gerados de cinco formas, que são:

- pela chamada de sistema kill [Sun 90b], que permite que um processo envie um sinal a outro processo ou a ele mesmo.

  int kill(int *pid*, int *sig*);

  onde

  pid é o inteiro que identifica o processo recebedor do sinal e

  sig é o inteiro que corresponde ao sinal desejado.

  Alguns sinais realmente terminam o processo recebedor, outros são usados para informar ao processo recebedor sobre alguma condição, que é então "tratada" por este. Um processo não é apto a enviar um sinal a qualquer outro processo do sistema. Para enviar um sinal, é preciso que os processos remetente e recebedor tenham o mesmo ID (identificador) de usuário ou o processo remetente seja do superusuário.

- através do comando (de mesmo nome) kill [Sun 90b], que é um programa que toma o argumento da linha de comando e realiza uma chamada de sistema kill.

Tabela A.1: Sinais do UNIX

Nome	Nota	Descrição	Ação Padrão
SIGALRM		Alarme do relógio	Termina
SIGBUS		Erro de barramento	Termina gerando imagem
SIGCLD		"Morte" de um processo filho	Descartado
SIGCONT	*4.3BSD*	Continuar após SIGSTP	Descartado
SIGEMT		Instrução EMT	Termina gerando imagem
SIGFPE		Instrução FPE	Termina gerando imagem
SIGHUP		Fechamento de um terminal	Termina
SIGILL		Instrução ilegal	Termina gerando imagem
SIGINT		Caracter de interrupção	Termina
SIGIO	*4.3BSD*	E/S é possível em um descritor de arquivo	Descartado
SIGIOT		Instrução IOT	Termina gerando imagem
SIGKILL		"Mata"	Termina
SIGPIPE		Escrita em pipe sem ninguém para ler	Termina
SIGPOLL	*System V*	Evento selecionável em dispositivo de *streams*	Termina
SIGPROF	*4.3BSD*	Alarme de perfil de tempo	Termina
SIGPWR	*System V*	Falta de energia	Termina
SIGQUIT		Caracter de abandono	Termina gerando imagem
SIGSEGV		Violação de segmentação	Termina gerando imagem
SIGSTOP	*4.3BSD*	Parada	Pára (suspende) processo
SIGSYS		Argumento errado para chamada de sistema	Termina gerando imagem
SIGTERM		Sinal de término de *software*	Termina
SIGTRAP		Armadilha de investigação	Termina gerando imagem
SIGTSTP	*4.3BSD*	Sinal de parada gerado do teclado	Pára (suspende) processo
SIGTTIN	*4.3BSD*	Leitura em *background* do terminal de controle	Pára (suspende) processo
SIGTTOU	*4.3BSD*	Escrita em *background* no terminal de controle	Pára (suspende) processo
SIGURG	*4.3BSD*	Condição urgente presente em socket	Descartado
SIGUSR1		Sinal definido pelo usuário 1	Termina
SIGUSR2		Sinal definido pelo usuário 2	Termina
SIGVTALRM	*4.3BSD*	Alarme de tempo virtual	Termina
SIGWINCH	*4.3BSD*	Mudança de tamanho de janela	Descartado
SIGXCPU	*4.3BSD*	Limite de tempo de UCP excedido	Termina
SIGXFSZ	*4.3BSD*	Limite de tamanho de arquivo excedido	Termina

- através de alguns caracteres do teclado, como por exemplo os caracteres de interrupção (tipicamente $Control-C$ ou *Delete*) e de abandono (tipicamente $Control - backslash$). O caracter de interrupção gera um sinal SIGINT, terminando um processo que está executando. O caracter de abandono termina um processo que está executando e gera uma imagem da memória do processo que pode ser usada para uma posterior análise do mesmo - ele gera um sinal SIGQUIT. Estes caracteres podem ser ajustados para serem quase qualquer um dos caracteres do teclado que se desejar. Além destes dois, a versão *4.3BSD* provê ainda um caracter de suspensão (tipicamente $Control - Z$), que gera um sinal SIGTSTP, imediatamente, e um caracter de suspensão com retardo (tipicamente $Control - Y$). Estes sinais gerados pelo terminal são enviados a todos os processos do grupo de controle do terminal e não somente ao processo que está executando. Eles geralmente são enviados do núcleo do sistema para um processo.

- por algumas condições de *hardware*, como por exemplo os erros de aritmética de ponto flutuante, que geram um sinal SIGFPE, ou o referenciamento de um endereço fora do espaço de endereços de um processo, o qual gera um sinal SIGSEGV. As condições de *hardware* específicas e os

sinais que elas geram, podem diferir de uma versão do UNIX para outra. Estes tipos de sinais também são normalmente enviados do núcleo do sistema operacional para um processo.

- por algumas condições de *software* que são notificadas pelo núcleo do sistema operacional, como é o caso do término ou a suspensão de um processo filho, que gera o sinal SIGCLD, o qual é enviado ao processo pai.

Uma vez que tenha recebido um sinal, seja do núcleo do sistema operacional ou de um outro processo, o processo recebedor pode:

- criar uma função (ou procedimento) que é chamada sempre que um tipo específico de sinal ocorre. Esta função, chamada de manipuladora de sinal, pode fazer o que quer que o processo queira para lidar com a condição. Isto é chamado "pegar" o sinal.
- ignorar o sinal, uma vez que não seja o SIGKILL nem o SIGSTOP.
- ou permitir que o padrão aconteça, como indicado na coluna " Ação Padrão" da Tabela A.1. Normalmente um processo é terminado ao receber um sinal, com alguns sinais gerando uma imagem da memória de um processo em seu diretório corrente de trabalho, mas, no *4.3BSD*, a ação padrão para os sinais SIGURG, SIGCONT, SIGIO e SIGWINCH é para serem ignorados e para os sinais SIGSTOP, SIGTSTP, SIGTTIN e SIGTTOU é para parar os processos.

Para especificar como ele quer tratar um sinal, o processo deve invocar a chamada de sistema **signal** [Sun 90b]; exceto para os sinais SIGIO, SIGPOLL e SIGURG, pois eles requerem que o processo execute outras chamadas de sistemas caso este queira pegar o sinal.

```
#include <signal.h>

int (*signal (int sig, void (*func) (int)))(int);
```

O que esta declaração diz é que **signal** é uma função que retorna um ponteiro para uma função que retorna um inteiro. O argumento *func* especifica o endereço de uma função que não retorna nada (void). Há dois valores especiais para o argumento *func*: SIG_DFL para especificar que o sinal é para ser tratado da forma padrão e SIG_IGN para especificar que o sinal é para ser ignorado. A chamada de sistema **signal** sempre retorna o valor anterior de *func* para o sinal especificado.

Quando uma função é chamada para tratar um sinal, o número do sinal é passado como o primeiro argumento da função. Isto permite que uma única função possa tratar vários sinais, determinando em tempo de execução qual sinal tenha ocorrido. Alguns sinais gerados pelo *hardware* passam argumentos adicionais para a função manipuladora do sinal. Se a função manipuladora de

sinal retorna, o processo que recebeu o sinal continua sua execução do ponto em que foi interrompido.

A.1 Sinais Confiáveis

Nas primeiras versões do UNIX, o mecanismo de sinais não era confiável. A existência de condições de corrida fazia com que sinais pudessem ser perdidos, isto é, um evento poderia gerar um sinal, mas o processo nunca seria notificado. De forma a se prover sinais mais confiáveis, foram acrescentadas as seguintes características tanto na versão *BSD* quanto na versão *System V*:

- as funções tratadoras de sinal permanecem instaladas depois da ocorrência de um sinal. As primeiras versões do UNIX redefiniam a ação associada a um sinal como SIG_DFL no instante imediatamente anterior ao da invocação da função tratadora de sinal do usuário. Isto implicava que qualquer ocorrência do mesmo sinal antes que a função tratadora de sinal pudesse invocar a chamada de sistema **signal** novamente, fosse perdida;

- um processo é apto a impedir a ocorrência de determinados sinais quando desejado, sem que, no entanto, ele seja descartado, o que ocorreria caso se usasse a ação SIG_IGN. O que se procura é que o sinal seja lembrado e remetido quando se estiver pronto a atendê-lo. Na versão *BSD* esta característica é conhecida por bloquear um sinal e na versão *System V*, por reter um sinal;

- enquanto um sinal está sendo remetido a um processo, este sinal é bloqueado (retido) em relação ao mesmo, ou seja, se um sinal é gerado uma segunda vez, enquanto o processo está tratando a sua primeira ocorrência, a função tratadora do sinal não é chamada novamente. O sinal será lembrado e a função tratadora será invocada outra vez, caso a mesma retorne normalmente após o tratamento da primeira ocorrência deste sinal.

A primeira e a terceira característica foram incorporadas à chamada de sistema **sigaction**, a qual permite que se examine ou modifique a ação associada a um sinal particular, excedendo, assim, à chamada **signal** das versões anteriores do UNIX.

A segunda característica foi acrescentada às novas versões com a introdução de uma outra importante chamada de sistema, a chamada **sigprocmask**. Através dela, um processo pode, a qualquer instante de sua execução, examinar e/ou alterar sua "máscara" de sinais. A máscara de sinais de um processo é o grupo de sinais que se encontram com o seu recebimento, temporariamente, bloqueado.

Maiores detalhes sobre o mecanismo de sinais no UNIX podem ser encontrados nas referências [Ste 90], [Ste 93] e [Sun 90a].

Apêndice B

Exclusão Mútua e Sincronização Usando Semáforos

Em um sistema concorrente, os processos que compõem o sistema são superpostos relativamente ao tempo, isto é, uma operação pode ser iniciada, em função da ocorrência de algum evento, antes do término da operação que estava executando anteriormente. Assim, estes processos devem colaborar de maneira a compartilhar informações e recursos do sistema. De modo a garantir uma correta operação dos processos sobre os recursos é necessário que estes processos sejam adequadamente sincronizados.

Às notações e técnicas de programação utilizadas para expressar um paralelismo potencial e para solucionar os problemas de sincronização e comunicação resultantes deste paralelismo, dá-se o nome de "programação concorrente". O problema básico da programação concorrente é a identificação de quais atividades podem ser realizadas concorrentemente.

Muitas vezes os processos que constituem o sistema concorrente compartilham um mesmo recurso, que, em um determinado instante de tempo, só pode ser acessado por um único processo. O trecho do código do processo que acessa um recurso compartilhado e que necessita de proteção com relação a outros processos denomina-se "região crítica", enquanto que a técnica de definição de um código de entrada e saída de forma a que, em um determinado instante, somente um único processo esteja executando sua região crítica, isto é, acessando o recurso compartilhado, denomina-se " exclusão mútua ".

A exclusão mútua é um dos mais importantes problemas na programação concorrente devido ao fato de ser a abstração de muitos dos problemas de sincronização. A abstração do problema de exclusão mútua pode ser expressa como apresentado na Figura B.1. O protocolo representa o preço a ser pago pela concorrência.

Além de que uma terminação anormal do processo fora da região

```
    .
    .
    .
processamento
protocolo de entrada
seção crítica
protocolo de saída
processamento
    .
    .
    .
```

Figura B.1: Forma de Solução do Problema de Exclusão Mútua

crítica não deva afetar nenhum outro processo, os requisitos básicos da abstração proposta são:

- satisfazer a propriedade de exclusão mútua,
- impedir a ocorrência de *deadlocks*[1] e,
- na ausência de disputa, possibilitar a entrada de um processo na sua seção crítica imediatamente.

Dijkstra [Dij 68] abstraiu as noções chaves de exclusão mútua no conceito de "semáforo". Semáforo é uma variável protegida cujo valor pode ser acessado e alterado somente pelas operações P e V e pela operação de inicialização. Os semáforos binários somente podem assumir os valores 0 e 1 e os contadores podem assumir valores não negativos.

A operação P no semáforo s, escrita P(s), opera como mostrado na Figura B.2 e a operação V no semáforo s, escrita V(s), opera como mostrado na Figura B.3.

```
if s > 0
    then s := s-1;
    else (espera em s);
```

Figura B.2: Operação P(s)

```
if (um ou mais processos estão esperando em s)
    then (permita que um dos processos prossiga);
    else s := s+1;
```

Figura B.3: Operação V(s)

[1] *Deadlocks* são cenários onde dois ou mais processos bloqueiam-se uns aos outros, fazendo com que o sistema ao qual pertencem não possa mais responder a nenhum sinal ou requisição

As operações P e V são indivisíveis, ou seja, a exclusão mútua ao semáforo s é garantida dentro das operações P(s) e V(s).

Um processo que deseja entrar em sua seção crítica, digamos *P1*, executa um protocolo de entrada que consiste somente da operação P(s). Se s=1, então s pode ser subtraído de 1 e *P1* entra em sua seção crítica. Quando *P1* sai de sua seção crítica e executa o protocolo de saída, que consiste apenas da operação V(s), o valor de s será novamente 1. Entretanto, se *P2* tenta entrar em sua seção crítica antes de *P1* ter saído, *P2* ficará suspenso em s. Quando *P1* finalmente sair, a operação V(s) acordará *P2*.

A implementação atômica das operações sobre o semáforo impede o escalonamento entre o teste de s e a atribuição para s.

De relevante interesse, é a possibilidade de sincronização de processos através de semáforos. Isto porque, freqüentemente, um processo deseja ser "notificado" a respeito da ocorrência de algum evento. Supondo que um outro processo seja capaz de detectar que o evento ocorreu, uma solução seria a mostrada na Figura B.4. Deve ser observado que o mecanismo funciona mesmo na hipótese de que o processo *P2* detecte e sinalize o evento antes do processo *P1* executar P(evento_de_interesse).

```
program Sincronização_via_Semáforo;
var s: semáforo;

task body P1 is
begin
    Seção_Não_Crítica_1;
    P(s);
    Outra_Seção_Não_Crítica_1;
end P1;

task body P2 is
begin
    Seção_Não_Crítica_2;
    V(s);
    Outra_Seção_Não_Crítica_2;
end P2;

begin (* Programa Principal *)
    Início_Semáforo(s,0);
    cobegin
        P1; P2;
    coend;
end.
```

Figura B.4: Sincronização de Tarefas através de Semáforos

Apêndice C

Comunicação entre Processos no UNIX

Quando se trata de programação tradicional, a comunicação entre diferentes módulos de um mesmo processo pode ser feita usando-se variáveis globais, chamadas às funções e passagem de parâmetros. No caso de programação em redes, a comunicação entre diferentes processos pode ser feita basicamente de duas formas: Através de troca de mensagens e/ou através de compartilhamento de memória.

No caso de troca de mensagens, as variáveis são locais a cada processo e o acesso a elas se dá através do envio/recebimento de mensagens. Quando uma mensagem chega ao seu destino, ela é armazenada sendo entregue ao processo destino quando este requisitá-la. Caso a mensagem não esteja disponível no momento em que for requisitada, o processo pode ter sua execução suspensa até a chegada da mesma.

No caso de compartilhamento de memória, é necessário a disponibilidade de mecanismos de sincronização (semáforos, por exemplo) com o objetivo de garantir a consistência dos dados armazenados. A forma de evitar o conflito na utilização de um recurso é através de um acesso ordenado ao recurso compartilhado: Primeiramente, a solicitação do recurso, depois a sinalização quando da liberação do mesmo. O recurso é alocado ou não ao processo em função da sua disponibilidade.

Quando os sistemas operacionais de multiprogramação foram desenvolvidos, havia a preocupação de que nenhum processo interferisse em outro. Assim, para que dois processos se comuniquem é necessário que ambos concordem e que o sistema operacional provenha algumas facilidades para a comunicação entre eles.

O uso de comunicação entre processos (*Interprocess Communication - IPC*), não se restringe apenas ao caso de programação em rede. Há muitos

exemplos onde *IPC* pode e deve ser usada entre processos em um sistema de um único processador. Uma descrição detalhada dos mecanismos de *IPC* no UNIX pode ser encontrada na referência [Ste 90].

C.1 Filas de Mensagens

Muitos sistemas operacionais modernos apresentam alguma forma de troca de mensagem entre processos. Alguns sistemas operacionais restringem a passagem de mensagens, de tal forma, que um processo só pode enviar uma mensagem para outro processo especificado. O UNIX *System V* não tem tal restrição. Nele, todas as mensagens são armazenadas no núcleo e têm um " identificador de fila de mensagem " associado. Ele é o identificador, chamado *msqid*, que identifica uma fila de mensagens particular. Os processos lêem e escrevem mensagens em filas arbitrárias. Não há exigência que algum processo esteja esperando por uma mensagem chegar em uma fila antes que outro processo esteja permitido para escrever na mesma, ao contrário do que ocorre com *pipes* e *FIFOs*, onde não faz sentido ter um processo escritor sem que exista também um processo leitor. Assim, é possível ter um processo que escreva algumas mensagens em uma fila, termine sua execução e então tenha as mensagens lidas por outro processo mais tarde.

Toda mensagem em uma fila tem os seguintes atributos:

- *type* - inteiro longo;
- *length* - tamanho da porção de dados da mensagem (pode ser zero);
- *data* - dados (se o tamanho for maior que zero).

Uma nova fila de mensagens é criada ou uma fila de mensagens existente é acessada com a chamada de sistema **msgget**. O valor retornado por **msgget** é o identificador da fila de mensagem, *msqid*, ou -1 se ocorreu algum erro.

Uma vez aberta a fila de mensagem com **msgget**, coloca-se uma mensagem na fila usando a chamada de sistema **msgsnd** [Sun 90c] e se lê tal mensagem da mesma usando a chamada de sistema **msgrcv** [Sun 90c]. Um dos argumentos da função **msgrcv**, o inteiro longo *msgtype*, especifica qual mensagem na fila é desejada:

- Se *msgtype* é zero, é retornada a primeira mensagem da fila. Desde que toda mensagem seja armazenada como em uma fila onde o primeiro a entrar é o primeiro a sair (*first in, first out*), um *msgtype* igual a zero especifica que deve ser retornada a mensagem mais antiga da fila.

- Se *msgtype* é maior que zero, é retornada a primeira mensagem com o tipo igual a *msgtype*.
- Se *msgtype* é menor que zero, é retornada a primeira mensagem com o menor tipo que seja menor ou igual ao valor absoluto do argumento *msgtype*.

Para se remover uma fila de mensagens do sistema usa-se a chamada de sistema **msgctl** [Sun 90c], a qual prove uma variedade de operações de controle sobre filas de mensagens.

O propósito de se ter um tipo associado a cada mensagem é o de permitir que vários processos possam multiplexar mensagens através de uma única fila. Outra característica provida pelo atributo de tipo de mensagens é a possibilidade que tem o receptor de ler as mensagens em outra ordem que não seja a de primeira entrar, primeira a sair. Com *pipes* e *FIFOs*, os dados devem ser lidos na ordem em que foram escritos; já com filas de mensagens, pode-se ler as mensagens em qualquer ordem que seja consistente com os valores associados ao tipo das mesmas. Isto é, em essência, pode-se assinalar prioridades às mensagens associando uma prioridade a um tipo ou faixa de tipos. Além disso, pode-se, ainda, ler qualquer mensagem de um determinado tipo de uma fila, retornando-se, imediatamente, caso não haja nenhuma mensagem do tipo especificado nesta fila.