sid.inpe.br/jeferson/2004/06.22.15.36-TDI

AGENDAMENTO E SIMULAÇÃO PARALELA/DISTRIBUÍDA DE PROCESSOS EM TEMPO REAL/CRÍTICO COM ARQUITETURAS MODERNAS TIPO HLA

Gilberto da Cunha Trivelato

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espacias/Mecânica Espacial e Controle, orientada pelo Dr. Marcelo Lopes de Oliveira e Souza, aprovada em 04 de dezembro de 2004.

URL do documento original:
<http://urlib.net/sid.inpe.br/jeferson/2004/06.22.15.36>

INPE
São José dos Campos
2011

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@sid.inpe.br

sid.inpe.br/jeferson/2004/06.22.15.36-TDI

AGENDAMENTO E SIMULAÇÃO PARALELA/DISTRIBUÍDA DE PROCESSOS EM TEMPO REAL/CRÍTICO COM ARQUITETURAS MODERNAS TIPO HLA

Gilberto da Cunha Trivelato

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espacias/Mecânica Espacial e Controle, orientada pelo Dr. Marcelo Lopes de Oliveira e Souza, aprovada em 04 de dezembro de 2004.

URL do documento original:
<http://urlib.net/sid.inpe.br/jeferson/2004/06.22.15.36>

INPE
São José dos Campos
2011

Dados Internacionais de Catalogação na Publicação (CIP)

Trivelato, Gilberto da Cunha.
T739a Agendamento e simulação paralela/distribuída de processos em tempo real/crítico com arquiteturas modernas tipo HLA / Gilberto da Cunha Trivelato. – São José dos Campos : INPE, 2011.
194 p. ; (sid.inpe.br/jeferson/2004/06.22.15.36-TDI)

Tese (Doutorado em Engenharia e Tecnologia Espacias/Mecânica Espacial e Controle) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2011.
Orientador : Dr. Marcelo Lopes de Oliveira e Souza.

1. Agendadores. 2. Escalonadores. 3. Tempo real. 4. Simulação distribuída. 5. Arquitetura de Alto Nível. I.Título.

CDU 629.78:004.451.26

Aprovado(a) pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de **Doutor(a) em. Engenharia e Tecnologia Espacial/Mecânica Espacial e Controle.**

Dr. Atair Rios Neto
Presidente/INPE, SJCampos-SP

Dr. Marcelo Lopes de Oliveira e Souza
Orientador/INPE, SJCampos-SP

Dr. Paulo Giácomo Milani
Membro da Banca/INPE, SJCampos-SP

Dr. Ulisses Thadeu Vieira Guedes
Membro da Banca/INPE, SJCampos-SP

Dr. Germano de Souza Kienbaum
Membro da Banca/INPE, SJCampos-SP

Dr. Marcelo Curvo
Membro da Banca/Convidado - EMBRAER

Dr. Nei Cardoso Cardenuto
Membro da Banca/Convidado - EMBRAER

Aluno: Gilberto da Cunha Trivelato

São José dos Campos, 12 de abril de 2004.

“Fé inabalável só o é a que pode encarar frente a frente a razão, em todas as épocas da Humanidade”.

ALLAN KARDEC

Aos espíritos nobres e retos que militam com grandeza na Causa do Bem.
Aos espíritos que se reconhecem frágeis e imperfeitos, caindo e erguendo-se, muitas vezes, nas trilhas da existência, sob críticas e censuras, mas sempre resistindo à tentação do desanimo, sem desistirem de trabalhar.
(Plagio de Emmanuel, psicografado por Francisco Cândido Xavier, o mineiro do século XX).

AGRADECIMENTOS

Ao orientador Dr. Marcelo Lopes de Oliveira e Souza pela orientação total e globalizada, incluindo as centenas de noites, domingos e feriados de discussões e estudos sobre as diversas áreas do conhecimento (não expressas nestas páginas), viagens a congressos anuais no exterior, revisão de apresentações em quartos de hotel, inumeráveis almoços e jantares recheados de mais discussões técnicas e também de teimosias de ambas as partes. E também à Rosemary Almeida Lopes, sua esposa, pelo sacrifício e paciência em acompanhar inúmeras dessas discussões.

Ao Instituto Nacional de Pesquisas Espaciais - INPE (www.inpe.br) pela oportunidade contínua de estudos, discussões técnicas e utilização de suas instalações, iniciada há duas décadas quando ingressei no seu quadro de estudantes de Mestrado em Ciências Espaciais e, ainda no mesmo ano, no seu quadro de pesquisa e desenvolvimento.

À Empresa Brasileira de Aeronáutica S.A. - EMBRAER (www.embraer.com.br) pelas condições gerais que me propiciou durante o desenvolvimento deste trabalho entre as quais destaco: i) o ambiente de trabalho e desenvolvimento tecnológico desafiador a cada novo projeto; ii) o régio pagamento de meu salário, benefícios e participações em lucros e resultados, que me permitiu e permite o investimento em livros, equipamentos, programas e viagens a congressos; e, iii) a sua política de horário flexível aliada à compreensão de meus superiores, que me permitiu a compensação de todos os horários que me ausentei da empresa para o cumprimento deste programa.

À empresa Belge Engenharia & Sistemas (www.belge.com.br), na pessoa de seu proprietário e diretor Alain Guy Charles de Norman et d Audenhove, pelo fornecimento do software de simulação discreta ProModel, suporte e sugestões na implementação dos modelos de simulação utilizados neste trabalho.

Aos meus colaboradores na HOMINE INFORMÁTICA E EDUCAÇÃO, na pessoa do analista Leonardo Vieira Barcelos, aos meus amigos, na pessoa do educador Mauro de Menezes, e aos meus familiares, na pessoa da minha irmã e assistente social Delimar Cunha Trivelato; por manterem vivo, mesmo na minha ausência necessária para cumprimento deste programa, o nosso ideal de desenvolvimento de um programa auto-sustentável de melhoria da educação pública brasileira na minha cidade natal Frutal, Minas Gerais.

Aos professores e colegas do INPE pelo conhecimento compartilhado e suporte nos momentos necessários.

Aos colegas da EMBRAER pelo ambiente de trabalho e pelas inúmeras discussões técnicas essênciais para a boa formação profissional.

RESUMO

Este trabalho tem como objetivo estudar o “Agendamento e Simulação Paralela/Distribuída de Processos em Tempo Real/Crítico com Arquiteturas Modernas Tipo HLA”. Para isto apresenta os objetivos, as justificativas, os conceitos básicos, a revisão bibliográfica. A seguir, apresenta novos esclarecimentos para a consolidação de resultados existentes na literatura, uma nova estratégia para agendadores em tempo real e seu impacto sobre os já existentes, 4 novas estratégias de algoritmos agendadores considerando 3 níveis de fidelidade ajustáveis. Nestas analisa as suas vantagens e desvantagens, as ilustra com experimentos projetados com 12, 24 e 36 tarefas com características (período e duração) realistas, e as compara segundo os critérios de fator de utilização U, número de reduções de níveis de fidelidade, número de transições (suspensões/preempções e reentrâncias), e tempo gasto com estas. Baseado nisto tudo, propõe o uso destes algoritmos como aperfeiçoamentos necessários ao “Object Management” e ao “Data Distribution Management” da arquitetura HLA. Finalmente, o trabalho conclui com uma apreciação dos resultados novos apresentados, e com as sugestões e recomendações para o prosseguimento deste trabalho.

SCHEDULING AND SIMULATION OF PARALLEL/DISTRIBUTED PROCESSES IN REAL/CRITICAL TIME IN A MODERN ARCHITECTURE LIKE HLA

ABSTRACT

This work studies the scheduling and simulation of parallel/distributed processes in real/critical time with modern architectures like HLA. To do that, it begins presenting the motivation, the organization of this work, the basic concepts, and the bibliographical review. Then it presents new developments for the consolidation of results existent in the literature, a new strategy for schedulers in real time and its impact on existing ones, 4 new families of scheduler algorithms, considering 3 adjustable levels of fidelity. There, it analyses their advantages and disadvantages, illustrates them with experiments designed with 16, 24 and 36 tasks with realistic characteristics (period and duration), and compare them by criteria such as the factor of utilization U, number of reductions of levels of fidelity, number of transitions (preemptions and reentrances), and time spent with them. Based on all that, it proposes the use of those algorithms as necessary improvements on the Object Management and on the Data Distribution Management of the HLA architecture. Finally, the work concludes with an assessment of the new results presented, and with the suggestions and recommendations for future work.

SUMÁRIO

Pág:

LISTA DE FIGURAS

Pág:

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$<$ - Relação de ordem dos elementos da estrutura T.

τ_1 - Tarefa 1 de um conjunto de tarefas.

$\in$ - Símbolo de pertence.

Δ - Variação de tempo.

Δ_{TS} - Tempo de chaveamento de tarefas.

$\forall$ - Símbolo que representa “qualquer”.

B_i - Intervalo de tempo que a tarefa τ_i foi suspensa por uma tarefa de maior prioridade.

C_i - Duração da tarefa ***i***.

$C_1, C_2, ..., C_m$ - Duração de cada uma das tarefas de um conjunto de tarefas.

C_k^i - Tempo disponível para execução de τ_k em relação a τ_i.

$C_{i(n)}$ - Tempo de processamento da tarefa *i* para o grau de fidelidade *n.*

E_i - Intervalo de tempo que o limite de execução de τ_i foi antecipado por uma tarefa de maior prioridade.

f_i - Frequência da i-esima tarefa.

M - Mínimo múltiplo comum (MMC).

m - Número de tarefas a ser agendado.

$(m \to \infty)$ - Valor de *m* tende a infinito.

n_i - Quantidade de inteiros de uma tarefa em relação ao período de outra.

R_i^k - Resto de *i* em relação a *k*, é o tempo da tarefa τ_i executado após o tempo T_k (período da tarefa *k*) considerando a execução somente das duas tarefas.

S - Relação fixa (ou “escala”)

S_i - Conjunto S_i são os pontos de agendabilidade para a tarefa i.

$S_j^{i(k)}$ - Sustação da execução de C_j^{th} pela execução de C_i^{th} em relação ao instante crítico T_k.

S_k^i - Sustação da execução de C_k^{th} pela execução de todas as tarefas de maior prioridade que k no instante crítico.

$\langle T, < \rangle$ - Estrutura tempo natural.

T - Sequência de números reais inteiros.

t - Tempo natural.

$\langle T', < \rangle$ - Estrutura de tempo virtual.

T' - Sequência de números reais inteiros.

t' - Tempo escalonado ou virtual.

$[0, t]$ - Intervalo de tempo fechado de 0 a *t*.

t_1 - Tempo no instante 1.

$T_{[t_1, t_2)}$ - Intervalo de tempo entre os instantes 1 e 2, incluindo o instante 1.

$T_{t)}$ - Tempo passado excluindo o presente momento.

$T_{t]}$ - Tempo passado incluindo o presente momento

$T_{(t}$ - Tempo futuro excluindo o presente momento

$T_{[t}$ - Tempo futuro incluindo o presente momento

t_{IL} - Tempo de latência de interrupção.

t_{DB} - Tempo de solução de “deadlock”.

t_R - Tempo requisitado ou solicitado (“requested time”).

t_s - Avanço de tempo solicitado.

Ti - Período da tarefa ***i***.

T_m - Período da tarefa ***m***.

$\{T_m / T_i\}$ - Partes fracionais dos períodos das tarefas *m* e *i* de um conjunto de tarefas.

(T_i, T_j) - Par de períodos das tarefas *i* e *j* de um conjunto de tarefas.

$\lceil T_2 / T_1 \rceil$ - Menor inteiro $\geq \frac{T_2}{T_1}$ ou "teto" de T_1 / T_2.

$\lceil T_2 / T_1 \rceil^n$ - Menor inteiro de T_1 em relação à enésima chamada de T_2.

$\lfloor T_2 / T_1 \rfloor$ - Maior inteiro $\leq T_2 / T_1$ ou "chão" de T_2 / T_1 .

$t_{\min(m)}$ - Tempo mínimo de execução disponível para a tarefa m.

t_{pa} - Tempo de processamento do agendador.

t_{pak} - Parte de tempo fixa constante de processamento do agendador.

t_{pat} - Tempo do agendador gasto com alteração do tempo de execução da tarefa.

T_{cpu} - Tempo de processamento necessário para execução do algoritmo.

$U(C_i, T_i)$ - Fator de utilização do processador em função do tempo de execução e do período da tarefa.

$U(C_i, T_i, M)$ - Fator de utilização do processador em função do tempo de execução e do período da tarefa e do MMC.

U - Fator de utilização do processador para o conjunto de tarefas.

U_{lub} - Menor limitante superior do fator de utilização do processador para um conjunto de tarefas.

U_{lub}^m - Menor limitante superior do fator de utilização do processador para um subconjunto de *m* tarefas.

$W_i(t)$ - Carga de trabalho de um conjunto de tarefas.

LISTA DE SIGLAS E ABREVIATURAS

AC	-	“Advisory Circular of FAA”
ADS	-	“Advanced Simulation Program”
ALSP	-	“Aggregate Level Simulation Protocol”
AM1	-	Superfície de agendabilidade máxima 1
AM2	-	Superfície de agendabilidade máxima 2
AMJ	-	“Advisory Material Joint”
ANT	-	“After Next Trigger Scheduling”
ASAPS	-	“As Soon As Possible Scheduling”
ATCMS	-	“Air Traffic Control and Management System”
ATR	-	“After Time Requirement Scheduling”
C4I2	-	“Command, Control, Computer, Communication, Information and Intelligence”
CCRs	-	“Conditional Critical Regions”
CPU	-	“Central Processor Unit”
CTA	-	Centro Técnico Aeroespacial
DARPA	-	“Defense Advanced Research Projects Agency”
DBES	-	“Dynamic Best Effort Scheduling”
DDM	-	“Data Distribution Management”
DIN	-	“German National Standards”
DIS	-	“Distributed Interactive Simulation”
DM	-	“Declaration Management”
DMSO	-	“Defense Modeling & Simulation Office”

DO-178B/ED-12B	-	“Data Order 178 – version ED – release 12B”
DoD	-	“USA Department of Defense”
DPBS	-	“Dynamic Planning-Based Scheduling”
EDFS	-	“Earliest Deadline First Scheduling”
EMBRAER	-	Empresa Brasileira de Aeronáutica S. A.
EUA	-	Estados Unidos da América do Norte
EUROCAE	-	“European Organization for Civil Aviation Electronics”
FAA	-	“Federal Aviation Administration”
FDD	-	“Federation Object Model Data Declaration”
FED	-	“Federation Execution Data”
FOM	-	“Federation Object Model”
FQR	-	“Flush Queue Request”
FTA	-	“Federation Time Axis”
GALT	-	“Greatest Advance Logical Lime”
HLA	-	“High Level Arquitecture”
I/ITSEC	-	“Industry/Interservice, Training, Simulation and Education
IEEE	-	“Institute of Electrical and Electronics Engineers”
INPE	-	Instituto Nacional de Pesquisas Espaciais
IS	-	“HLA Interface Specification”
IST	-	“Institute for Simulation and Training”
JAA	-	“Joint Aviation Authorities”
JSF	-	“Joint Striker Fighter”
LITS	-	“Least Interval Time Stamp”

LL	-	Superfície de Liu e Layland
LP	-	Superfície plana do limite do processador
LT	-	“Logical Time”
M&S	-	“Modeling & Simulation”
MIPS	-	Fluxo de entrada e saída medido em milhões de instruções por segundo
MIPS1	-	Velocidade de processamento da CPU em milhões de instruções por segundo
MIPS2	-	Capacidade de gerenciamento de interrupção medida em milhões de instruções por segundo
MMC	-	Mínimo Múltiplo Comum
MOM	-	“Object Model Specification”
NMR	-	“Next Message Request”
NMRA	-	“Next Message Request Available”
NTSA	-	“National Training Systems Association"
OMT	-	“Object Model Template”
PDPS	-	“Priority-Driven Preemptive Scheduling”
PDS	-	“Priority Driven Scheduling”
PIDS	-	“Priority Inheritance Driven Scheduling”
R1	-	Região abaixo do plano de Liu e Layland ($U \leq U_{lub}(m)$)
R2	-	Região acima do plano limite do processador ($1 < U$)
R3	-	Região acima do plano de Liu e Layland e abaixo do e sobre o plano limite do processador ($U_{lub}(m) < U \leq 1$)
R3a	-	Sub-região de R3 representada pelo interior do sólido formado pela superfície de agendabilidade máxima e o plano limite do processador

R3b	-	Sub-região de R3 representada pelo interior pela superfície de agendabilidade máxima e o exterior do mesmo sólido
RMS	-	“Rate Monotonic Scheduling”
RO	-	“Receive Order”
RTCA	-	“Radio Technical Commission for Aeronautics”
RTI	-	“RunTime Infrastructure”
RTOS	-	“Real Time Operational System”
SAF	-	“Soon As Finished”
SCS	-	“Society for Computer Simulation”
SE	-	“Synthetic Environment”
SFTC	-	“Scheduling with Fault-Tolerance Constraints”
SICT	-	“Scheduling with Imprecise Computation Technique”
SIMD	-	“Single Instruction Multiple Data”
SIMNET	-	“SIMulator NETworking”
SIVAM	-	Sistema de Vigilância da Amazônia
SOM	-	“Simulation Object Models”
SOTR	-	Sistema Operacional de Tempo Real
SRR	-	“Scheduling with Resource Reclaiming”
SSA	-	“System Safety Assessment”
STDS	-	“Static Table-Driven Scheduling”
TAR	-	“Time Advance Request”
TARA	-	“Time Advance Request Available”
TG	-	“Time Granted”
TSO	-	“Time Stamp Order”

VAPS	-	“Virtual Aircraft Prototyping System”
VV&A	-	“Verification, Validation and Accreditation”

CAPÍTULO 1

MOTIVAÇÃO E ORGANIZAÇÃO DESTE TRABALHO

1.1 Introdução

A evolução da computação e das comunicações desde o seu surgimento tem superado as previsões mais otimistas que tenham sido feitas nos instantes passados da sua era. Aplicações inimagináveis há pouco tempo estão presentes em nosso cotidiano. O avanço da área aeroespacial também tem sido expressivo e, em grande parte, podemos associá-lo ao uso de aplicações computacionais e de comunicações mais sofisticadas. Os nossos veículos aeroespaciais, como aeronaves, satélites, naves espaciais, sondas, podem ser comparados em muitas situações a verdadeiros “computadores-transceptores aeroespaciais”. Em muitos casos, como nos sistemas de defesa complexos e de constelações de satélites, o número de unidades de processamento e transceptores supera as dezenas, e os programas embarcados chegam a milhares ou milhões de linhas código.

Simultaneamente, os avanços da modelagem e da simulação permitiram a solução de problemas insolúveis somente com o uso de técnicas analíticas. Problemas complexos de diversas áreas do conhecimento são resolvidos pelo pré-processamento de modelos ou pelo processamento simultâneo ao tempo do sistema. Para ambientes complexos, envolvendo eventos de natureza discreta e contínua, várias arquiteturas foram elaboradas pela comunidade ligada à simulação, culminando com a Arquitetura de Alto Nível – “High Level Arquitecture” (HLA), proposta pelo “USA Department of Defense” (DoD) em 1995-1996 registrada por Kuhl et al (2000).

Apresentamos a seguir algumas áreas onde a modelagem e a simulação tem uma grande importância relativa e ainda assim são esperados grandes avanços no uso dessas técnicas.

Indústria Aeroespacial: modelam-se os processos produtivos de novas aeronaves, com o reuso dos modelos de desenvolvimento como ferramentas de treinamento e marketing,

validação de modelos desenvolvidos para certificação de novos produtos, etc. Isto motiva a criação de um ambiente virtual que permita sua aplicação em todas as fases de desenvolvimento dos veículos aeroespaciais, tais como: especificação, projeto, desenvolvimento, produção, testes, certificação, manutenção, treinamento e marketing.

Operações militares: desejam-se avanços nos graus de realismo de simulações de operações militares para a ajuda na tomada de decisão em combate. Além de investir muito em modelagem e simulação de batalhas complexas, o DoD também tem investido bastante na modelagem do comportamento humano em condições normais e anormais, com sensações e emoções, uso dos 5 sentidos, etc.

Construção e fabricação: simulam-se fábricas operadas por robôs, o aumento da eficiência na fabricação de semicondutores, testes de sistemas de comunicação móveis e sem fio, visualização tri-dimensional de operações de construção, etc.

Distribuição e logística: simulam-se todos os pontos para otimizar os recursos aplicados na distribuição, venda e suporte de produtos, visando expandir as possibilidades de grandes redes de distribuição e vendas, etc.

Saúde pública: o nível de atividades e avanços do uso da modelagem e simulação não corresponde à importância econômica da área. É um campo vasto para expansão em aplicações como: simulação de experimentos no desenvolvimento de vacinas, simulações para alocação de recursos em planejamento de sistemas públicos de saúde, simulações para melhoria nos projetos e processos já em andamento, etc.

Transportes: empregam-se simulações para o estudo, controle e aumento da segurança de sistemas de navegação aérea, simulação de operações para otimização de operação de companhias aéreas, controle do fluxo de veículos em grandes metrópoles, análise dos futuros sistemas de transportes, etc.

Considerando o histórico dessas áreas, as previsões agora são de aplicações ainda mais elaboradas, entre as quais destacamos: constelações de satélites, ciclo de vida de detritos espaciais (“space debris”), sistemas de defesa de mísseis, viagens interplanetárias,

controle de vôo integrado e livre (“free flight”), modelagem, simulação e integração de subsistemas junto ao corpo humano, sistemas imunológicos, etc.

Os sistemas operacionais não evoluíram adequadamente para atender essas novas demandas. De uma forma geral pode-se afirmar que atualmente eles não estão adequadamente preparados para lidar com o agendamento e a simulação de milhões, ou até mesmo bilhões, de tarefas. O problema torna-se mais complexo quando diferentes requisitos de execução são necessários para todas as tarefas, com a criação e destruição de tarefas ligadas a modelos simulados que podem “morrer” ou “nascer” em um ambiente sintético ou real.

Assim, o agendamento e a simulação de tais tarefas é uma área cada vez mais relevante do conhecimento e necessita de novas soluções ou métodos que sustentem o avanço da ciência e das aplicações, como proposto neste trabalho.

1.2 Motivação

A evolução da eletrônica de produção de “chips” cada vez mais densos em número de transistores e em velocidade de processamento aliada com novas e promissoras tecnologias destes componentes produziu equipamentos que podem executar atividades até há pouco tempo nem sonhadas pelo ser humano. Paralelamente os sistemas de controle ou controlados de tempo real tornaram-se mais comuns em nossa vida e também em aplicações remotas impossíveis de serem implementadas sem este avanço. Com a possibilidade de processamento de milhões e até mesmo de bilhões de tarefas ou modelos simulados surgem problemas de natureza diferente dos que encontramos até os dias de hoje. Entre os principais destacam-se os de gerenciamento do tempo e prioridade. Os tempos de latência e de mudança de contexto passam a ter uma importância muito significativa no cômputo geral das tarefas indicando a necessidade de reduzi-los ou evitá-los ao máximo.

Apresentamos sumariamente algumas aplicações que fazem uso desta fenomenal capacidade de processamento e de comunicação e que apresentam indicativos da necessidade de uma nova forma de organização das tarefas ou dos modelos simulados.

1.2.1 Processamento Multitarefa

O uso dos avanços tecnológicos mencionados permitiu o desenvolvimento de sistemas de controle ou controlados de tempo real críticos em aplicações muito complexas. Um sistema operacional de tempo real é elemento essencial nestes sistemas assim como a arquitetura de hardware e a linguagem de programação. As características essenciais de um Sistema Operacional de Tempo Real ("RTOS") são: rápido chaveamento de contexto, tamanho pequeno, rápida resposta a interrupções externas, minimizar intervalos quando as interrupções estão desabilitadas, não utilizar memória virtual, blindar código e dados na memória, alta capacidade de armazenar dados, possuir um relógio de tempo real, possuir capacidade de gerenciamento de tempo e alarmes e prover primitivas para atrasar de um determinado tempo, parar e reativar a execução de uma tarefa. Embora seja possível a existência de um sistema operacional de tempo real com execução de uma tarefa única, eles não fazem sentido para sistemas complexos, pois estes exigem a capacidade de operação multitarefa. Conforme Stankovic (1988) a teoria de agendamento/escalonamento de tarefas em sistemas de tempo real é uma importante área de pesquisa demandando novos algoritmos que sejam bem compreensíveis de forma que o comportamento do sistema no tempo seja compreensível, previsível e de fácil manutenção. Com as aplicações expostas nesta motivação ficará mais evidente esta necessidade de desenvolvimento da teoria de agendamento/escalonamento dinâmico de tarefas de sistemas de tempo real críticos em ambientes variáveis.

1.2.2 Sistemas de Controle e Defesa de Tráfego Aéreo

Estas áreas ilustram situações onde nenhum participante pode ser esquecido no cálculo da solução do problema, embora sejam diferentes em número, importância,

características, tempos aleatórios de entrada e saída no cenário, etc. Nenhum objeto ou ameaça deve ser ignorado no cenário sob o risco de este mesmo objeto ou ameaça ser a causa do problema futuro para o espaço aéreo controlado ou para a aeronave de defesa. Desta forma os recursos computacionais devem ser utilizados para monitorar todos os objetos. Quando se tratar de computação embarcada, uma maneira de aumentar a capacidade de defesa é criar um agendamento de tarefas que considere **diferentes níveis de fidelidade** dos modelos, sendo estes níveis de fidelidade **caracterizados especialmente pelo tempo necessário para processamento das tarefas correspondentes**, idéia central nesta Tese. Caso seja computação em solo, como um centro de controle de aeroporto, essa mesma técnica pode e deve ser utilizada para atender a demanda em momentos de pico causado por anomalias imprevistas ou improváveis, ou para redistribuição dos recursos de comunicação nos momentos de operação normal. Em breve, um centro de controle de um aeroporto será uma grande **federação** onde, além de pousos e decolagens de aeronaves, serão incluídos outros **federados** como: equipamentos auxiliares, caminhões de abastecimento, de carga e descarga, de suporte na pista ou internos e, no futuro, até dos passageiros. De forma semelhante, o espaço aéreo de um país preparado para vôos de rota livre (“free flight”) será uma grande federação com um número flutuante de federados e, também, deverá estar preparado para situações imprevistas ou improváveis.

1.2.3 Simulação de Detritos Espaciais

É a área internacionalmente conhecida como “Space Debris” e que se preocupa com o destino dos detritos espaciais. As simulações nesta área podem envolver inúmeros corpos e com características e condições iniciais distintas, interações (gravitacionais ou por colisão, arrasto, etc.) entre si e/ou com corpos ou meio externos, e em tempo prógrado (para propagação e previsão de colisões, etc.) ou retrógado (para localizar sua origem comum, etc.). Na simulação para previsão de colisão com ou de um conjunto de fragmentos, a não consideração de um elemento com baixa contribuição ou prioridade, pode mascarar exatamente o elemento que causará a catástrofe no futuro. Isto sugere que todas as tarefas devem ser executadas durante todo o tempo de processamento.

1.2.4 Controle e Operação de Constelações de Satélites

Em uma constelação de satélites frequentemente as posições relativas dos mesmos são mais importantes que a posição absoluta de cada uma deles em relação a Terra. Por exemplo, no caso de dois satélites em uma mesma órbita e opostos, se um se desloca de cinco metros em uma direção e o outro desloca de três metros na mesma direção, ao invés de corrigirmos ambos de cinco e três metros respectivamente, podemos corrigir um único de dois metros. O fato de se pensar de uma maneira global representa uma economia. Extrapolando este caso para uma constelação, digamos de 64 satélites, a não consideração de um deles resulta em um desperdício de energia. Este exemplo demonstra uma situação onde nenhum participante pode ser esquecido no cálculo da solução do problema e, portanto, a necessidade de processamento das tarefas relacionadas a todos mesmo que a contribuição de cada um no resultado final seja bastante diferente. A desconsideração de uma pequena contribuição no presente pode representar uma perda muito grande de energia no futuro.

1.2.5 Simulação e Operação de Viagens Interplanetárias

Em uma viagem interplanetária os recursos computacionais são tipicamente embarcados e finitos. Considerando a impossibilidade prática da solução analítica do problema de n ($n > 3$) corpos, e a ocorrência de eventos aleatórios no espaço, torna-se necessário computar em tempo real as posições relativas dos corpos ou até mesmo a simulação dos objetos incluindo ou não o **adiantamento do tempo**. Este é um ambiente típico onde surgem novas tarefas e não podemos adicionar ou trocar os computadores embarcados. A nave está só no espaço e precisa estar preparada para sobreviver em ambientes adversos e inesperados. Uma forma indireta de aumentar o tempo de processamento embarcado é, de forma similar ao pensamento humano, priorizar tarefas através do tempo dispensado a cada uma sem, no entanto deixar de executar nenhuma delas. A forma natural de se implementar isso é simplificar as tarefas menos prioritárias, melhorar a qualidade dos algoritmos de previsão ou ambos.

1.2.6 Controle e Operação de Componentes no Corpo Humano

O desenvolvimento e as aplicações da eletrônica trouxeram muitos avanços para as ciências ligadas ao ser humano. À medida que esses avanços ocorrem, as demandas correspondentes por processamento parecem aumentar em maior velocidade. As áreas de interfaces, simulação e controle de “nano-processadores” embarcados no corpo humano fazem verdadeiras revoluções no imaginário dos diversos tipos de cientistas envolvidos na questão. Um exemplo real nessa área ocorre com o modelo físico de um ser humano e sua correspondente simulação, em andamento no “Institute for Simulation and Training” (IST), Flórida, EUA, desenvolvido com a participação de diversas equipes. Em outros ambientes são modelados e simulados desde comportamentos psicológicos e sociais, passando pelo sistema imunológico do organismo. Quando falamos em células, vírus, bactérias, falamos em bilhões de modelos simulados (entidades) simultaneamente. Em muitos casos, nenhum dos modelos ou tarefas pode deixar de ser processado. Mesmo que sua contribuição naquele momento não seja significativa, o resultado desconsiderando essa contribuição pode ser catastrófico dentro de um determinado período. A organização da execução destas tarefas torna-se impraticável com os métodos utilizados atualmente à medida que o número das mesmas aumenta como nos casos mencionados.

1.3 Objetivos

Desenvolver uma nova estratégia e uma nova família de algoritmos de agendamento/escalonamento de tarefas (“schedulers”) para simulação paralela/distribuída de processos em tempo real críticos com arquiteturas modernas do tipo HLA, que permitam minorar e/ou solucionar os problemas mencionados. Esta família de algoritmos será analisada e comparada com o gerenciamento de tempo da arquitetura HLA, considerando que a mesma é uma arquitetura padrão para simulação. É sugerida a incorporação desta nova família de algoritmos como uma forma de melhoria na arquitetura HLA e também as respectivas alterações necessárias nessa arquitetura. Estes algoritmos deverão considerar e incluir novas variáveis de decisão tais como diferentes níveis de fidelidade e/ou resolução dos modelos simulados.

1.4 Justificativa

Entre os problemas mencionados na motivação deste trabalho, três estão incluídos entre as áreas de interesse do INPE, admitindo contribuições significativas: constelações de satélites, detritos espaciais e viagens interplanetárias. Nas áreas de interesse do Ministério da Defesa e da EMBRAER, o SIVAM é um programa de relevância para o país com um processo de transferência tecnológico que prevê a atualização e evolução do mesmo internamente. Também é de grande interesse e necessidade preparar o controle do espaço aéreo do país para os vôos de rota live. Estes casos são potenciais candidatos ao uso de simulações complexas para a solução de problemas e enfrentar desafios futuros.

Embora a área de modelagem e simulação do corpo humano como um todo seja incipiente e até mesmo inexpressiva no Brasil, temos apresentado um desenvolvimento fantástico na Biotecnologia através do Projeto Genoma. A Biotecnologia é uma área estratégica para o país é forte candidata ao uso de simulação com bilhões de tarefas simultâneas na solução de problemas futuros.

Entre as visões de necessidades futuras nas conferências do ano de 2001 da “Society for Computer Simulation" (SCS) e da “National Training Systems Association" (NTSA) durante a “Industry/Interservice, Training, Simulation and Education Conference” (IITSEC), de 26 a 29 de novembro de 2001, Flórida, EUA, foram apresentadas a simulação com milhões e até bilhões de entidades simultâneas e a solução dos problemas correlacionados e, entre eles, o de gerenciamento da execução das tarefas. Também foi mencionada a necessidade de novas formas de classificação de modelos para simulação.

1.5 Contribuições

Este trabalho apresenta uma nova família de algoritmos de agendamento/escalonamento dinâmico de tarefas (“schedulers”) baseados em uma nova estratégia, que permitam aos sistemas manterem as suas características de tempo real críticos (“hard real time

systems") adequados a ambientes dinâmicos e complexos com milhares e até milhões de tarefas. Os sistemas embarcados utilizados em ambientes com mudanças dinâmicas complexas (sistemas de controle de tráfego aéreo e defesa, o corpo humano ou viagens interplanetárias) e limitados em sua capacidade de processamento, mantém suas propriedades de sistemas de tempo real críticos cumprindo todas as tarefas correntes e incluindo as novas necessárias segundo a família de algoritmos proposta, até o limite extremo de sua capacidade. É proposta uma forma de uso e incorporação desta família de "schedulers" na arquitetura HLA e também alterações necessárias. Eles também podem ser utilizados no núcleo ("kernel") de uma de nova geração de sistemas operacionais. A incorporação desta nova família de agendadores/escalonadores de tarefas nos sistemas operacionais de tempo real permite que eles gerenciem a execução de milhares ou até milhões de tarefas de uma maneira mais próxima ou similar à forma de agendamento do cérebro humano.

A família de agendadores/escalonadores ("schedulers") proposta é original porque é baseada nos paradigmas da possibilidade da extensão do tempo através de níveis de fidelidade dos modelos (representados por tarefas) e na busca da perfeição tanto quanto seja possível. Essa família é de uso geral porque pode ser aplicada para a solução de agendamento/escalonamento genérico de tarefas em diferentes áreas do conhecimento humano. Também é útil porque pode ser implementada como um programa gerenciador de tarefas de mais alta prioridade em um sistema operacional ou até mesmo em um núcleo de um sistema operacional. As alterações sugeridas na arquitetura HLA permitem o desenvolvimento de sistemas de tempo real críticos para sistemas contendo partes reais e simuladas em tempo real.

1.6 Organização Deste Trabalho

Este trabalho foi dividido em mais sete capítulos, descritos a seguir:

- *CAPÍTULO 2 – CONCEITOS BÁSICOS*: Neste capítulo são abordados os conceitos de tempo usados neste trabalho; conceitos e classificação de sistemas de tempo real ou virtual – críticos e não críticos ("hard" ou "soft") e,

particularmente, na área aeronáutica; conceitos de sistemas síncronos e assíncronos; conceitos de sistemas paralelos ou distribuídos; conceitos de programas agendadores/escalonadores de tarefas (“schedulers”), estratégias para classificação de agendadores, outros aspectos para classificação de agendadores, classificação de agendadores de tarefas síncronas e classificação de agendadores de tarefas assíncronas; conceitos de sincronismo, comunicação, sustação e reentrância; o Departamento de Defesa Americano (DoD) e a simulação distribuída contendo termos e definições do DoD e um breve histórico da simulação distribuída no DoD; e, a apresentação da arquitetura HLA, com foco nas regras de funcionamento do HLA, compreensão de tempo na arquitetura HLA, o gerenciamento de tempo e o gerenciamento da distribuição dinâmica de dados.

- *CAPÍTULO 3 - REVISÃO BIBLIOGRÁFICA*: Este capítulo contém a revisão bibliográfica com relação aos principais autores com trabalhos específicos relacionados a este presente trabalho.

- *CAPÍTULO 4 – NOVOS ESCLARECIMENTOS PARA A CONSOLIDAÇÃO DE RESULTADOS EXISTENTES NA LITERATURA:* Apresenta nova contribuição para o esclarecimento e consolidação de teorema fundamental para a teoria dos programas de agendamento/escalonamento de tarefas proposto por Liu e Layland, e contestado recentemente na literatura. Sua estrutura é composta por definições utilizadas na literatura; teoremas e resultados importantes para este trabalho incluindo um algoritmo agendador de prioridade fixa, fator de utilização do processador, menor limitante superior, o agendador de taxa constante restrito, o agendador de taxa constante generalizado e contraprovas falsas na literatura; esclarecimento de equívocos na literatura; restrições do algoritmo de taxa constante; algoritmo com primeiro prazo fatal primeiro; e, restrições do algoritmo com primeiro prazo fatal primeiro.

- *CAPÍTULO 5 – NOVAS ESTRATÉGIAS PARA AGENDADORES/ ESCALONADORES EM TEMPO REAL E SEU IMPACTO SOBRE OS JÁ*

EXISTENTES: Neste capítulo são apresentados os problemas atuais e necessidades de novas estratégias; prova da existência e propriedades da superfície de agendabilidade máxima; visão gráfica do fator de utilização do processador em relação aos períodos e tempos de execução das tarefas, análise segundo as regiões: abaixo do plano de Liu e Layland; acima do plano limite do processador; e entre o plano de Liu e Layland e o plano limite do processador; revisão do critério de carga de trabalho, do critério de limite de taxa constante, e proposição do novo critério do final parcial; fundamentos de novas estratégias de agendamento; novo agendador a taxa constante dinâmico estendido com redução uniforme dos tempos de execução, e com outros novos critérios para redução dos tempos de execução de tarefas, bem como a análise dos problemas deste agendador; novo agendador dinâmico a taxa constante estendido com sintonia dos períodos das tarefas; novo agendador com primeiro prazo fatal primeiro estendido; e, novo agendador dinâmico para sistemas de tempo real críticos.

- *CAPÍTULO 6 – NOVA FAMÍLIA DE ALGORITMOS AGENDADORES/ ESCALONADORES CONSIDERANDO NÍVEIS DE FIDELIDADE AJUSTÁVEIS:* Este capítulo apresenta a arquitetura de programas de tempo real considerada neste trabalho; o novo algoritmo agendador a taxa constante dinâmico estendido; o novo algoritmo agendador com sintonia dos períodos das tarefas; o novo algoritmo do agendador com primeiro prazo fatal primeiro estendido; o novo algoritmo agendador dinâmico para sistemas de tempo real críticos; um experimento para testes e comparação dos algoritmos; resultados experimentais em modo de operação normal, resultados experimentais a uma sobrecarga média, resultados experimentais a uma sobrecarga grande (acima de 100% do processador); e a comparação dos resultados considerando o fator de utilização do processador, o chaveamento de tarefas e a sensibilidade a tempo de chaveamento de tarefas.

- *CAPÍTULO 7 – APLICAÇÃO NA ARQUITETURA HLA*: Apresenta a junção teórica do algoritmo proposto à arquitetura HLA; as alterações propostas nos

federados; as alterações propostas na federação, em particular nos serviços de gerenciamento de tempo (“Time Management” - TM) e de gerenciamento da distribuição de dados (“Data Distribution Management” – DDM); e, a criação de um novo serviço de gerenciamento do nível de fidelidade dos federados (“Fidelity Level Management” - FLM), com exemplos típicos de aplicação.

- *CAPÍTULO 8 – CONCLUSÕES, SUGESTÕES E RECOMENDAÇÕES PARA TRABALHOS FUTUROS*: Conclusões sobre o trabalho desenvolvido, sugestões e perspectivas de trabalhos futuros, e recomendações para novas pesquisas na área.

CAPÍTULO 2

CONCEITOS BÁSICOS

2.1 Conceitos de Tempo Usados Neste Trabalho

A definição de tempo tem sido um assunto bastante discutido nas diferentes áreas do conhecimento humano, indo desde as percepções da Física, passando pela Psicologia e chegando até a Filosofia. Entretanto preocupar-nos-emos somente com as definições pertinentes e aplicáveis no escopo deste trabalho.

Os conceitos de sistemas dinâmicos na Natureza são baseados na sua evolução ao longo da passagem do tempo. Este **tempo**, também chamado **tempo natural**, é uma variável independente e todas as alterações dinâmicas dos sistemas são ordenadas nele. Uma definição matemática ou formal para este tempo é proposta por Zeigler et al (2000) e apresentada a seguir.

"A base de tempo natural é definida como sendo a estrutura $tempo = \langle T, < \rangle$, onde T é uma sequência de números reais e $<$ é uma relação de ordem dos elementos de T. A relação $<$ é transitiva, irreflexível e assimétrica."

A relação de ordem nos permite usar os termos como "**passado**", "**futuro**", e "**presente**". Se considerarmos t como o tempo "presente", então a sequência $T_{t)} = \{\tau \mid \tau \in T, \tau < t\}$ representa o "passado" e a sequência $T_{(t} = \{\tau \mid \tau \in T, t < \tau\}$ representa o "futuro". A sequência $T_{t]} = \{\tau \mid \tau \in T, \tau \leq t\}$ representa o "passado" incluindo o tempo "presente" e a sequência $T_{[t} = \{\tau \mid \tau \in T, t \leq \tau\}$ representa o "futuro" incluindo o tempo presente. De forma similar a sequência $T_{[t_1,t_2)} = \{\tau \mid \tau \in T, t_1 \leq \tau < t_2\}$ representa o intervalo de tempo $[t_1, t_2)$ com início no tempo t_1, incluído no intervalo, e com final no tempo t_2, excluído do intervalo.

Pode ser necessária a representação ou modelo de um sistema, seja matemático, computacional ou lógico, utilizando a variável independente **tempo escalonado**

$t' = S * t$, ou seja, com alguma relação fixa (ou "escala") S em relação ao tempo natural t. **Tempo virtual** é definido como o tempo escalonado utilizado nesse tipo de representação de sistemas. Quando a relação for menor que um (S<1) pode-se dizer que a representação ou simulação do sistema é feita em "câmera lenta" por ser mais lenta que a resposta real do sistema; e, de forma similar, se for maior que um (S>1), a representação ou simulação do sistema é chamada de "acelerada" por ser mais rápida que a resposta real do sistema.

De forma similar à definição anterior de tempo, a **base de tempo virtual** pode ser definida como a estrutura $tempo\ \ virtual = \langle T', < \rangle$, onde T' é uma sequência de números reais, $<$ é uma relação de ordem dos elementos de T' e seus elementos t' são definidos em razão constante com $t \in T$, ou seja $t' = S * t$. As mesmas propriedades da sequência T são mantidas para T'.

2.2 Sistemas de Tempo Real Críticos ou Não-críticos

A história de **sistemas de tempo real** está relacionada com a história da **computação em tempo real**. Isto se deve ao fato de os requisitos de tempo real serem requisitos do sistema em seu ambiente como um todo, e não apenas os requisitos internos do computador.

Os fortes requisitos relacionados aos riscos de vida humana, econômico e financeiro fizeram surgir a necessidade de uma categoria de sistemas com alta confiabilidade e que reagissem como se fossem sistemas contínuos tanto com uma constante de tempo de resposta rápida ou lenta: os **sistemas de controle por computador em tempo real**.

Uma visão histórica sobre o controle por computador em tempo real pode ser vista no livro editado por Bennett e Linkens (1984). Um breve histórico da computação e, em particular da computação de tempo real, pode ser verificado em Martin (1980).

Um exemplo de um problema clássico na área de computação adicionado com requisitos de tempo real é apresentado no Capítulo 1 do livro editado por Stankovic e

Ramamritham (1988). Este caso evidencia de uma forma simples as diferenças fundamentais entre um **sistema de computação tão rápido quanto possível** (“as fast as possible”) e um sistema de computação em tempo real. No Capítulo 2 desta publicação é apresentada uma análise resumida sobre sistemas de tempo real delineando: i) os problemas dos sistemas de computação de tempo real, ii) a necessidade de se estabelecer princípios científicos para o desenvolvimento de sistemas integrados de tempo real, e iii) os erros conceituais comuns na área.

A discussão dos erros conceituais na área de sistemas de tempo real é atualizada no Capítulo 1 do livro editado por Kavi (1992). São apresentados vários tópicos importantes que necessitam pesquisa e desenvolvimento.

Uma boa definição para a **computação em tempo real** pode ser encontrada na norma alemã DIN 44 300: “É o modo de operação de um sistema de computador no qual o programa está permanentemente preparado para processamento de um dado externo, de maneira que o seu resultado estará disponível em um período de tempo pré-determinado; o momento de entrada desse dado externo pode ser aleatoriamente distribuído ou ser determinado anteriormente dependendo das diferentes aplicações”.

Uma boa definição para **sistemas de tempo real** e bastante comum na literatura pode ser encontrada em Stankovic e Ramamritham, IEEE, (1988): “O sistema pode ser chamado de tempo real somente se for possível garantir, na fase de projeto, que todos os requisitos de tempo para execução de tarefas serão atendidos em qualquer modo de operação”.

Considerando as referências citadas, podemos concluir que devemos ter um sistema de computação e comunicação de tempo real para termos uma simulação digital de tempo real ou um sistema de controle de tempo real.

Segundo as definições de Martin (1965), Shin e Ramanathan (1974), Stankovic (1988) e Kavi (1992), na computação de tempo-real os dados entram no computador e são processados numa arquitetura de hardware e software que garantem elevada confiabilidade de seu desempenho garantindo segurança aos elementos vivos e

manufaturados envolvidos; para depois serem direcionados para as respectivas saídas. A computação em tempo-real deve possuir o seguinte conjunto de características: confiabilidade (“reliability”), previsibilidade (“predictability”), agendabilidade (“scheduling”), formação de filas (“queues”), multiplexação (“multiplexing”), multiprocessamento (“multiprocessing”), substituição de computador falhado (“switchover”), operação em modo degradado (“fall-back”), dependabilidade (“dependability”), sincronização (“synchronization”), reusabilidade (“reusability”), conectividade (“connectivity”), e multitarefa (“multitask”). Uma comparação de sistemas de **tempo real** com propriedades de sistemas “**on-line**” e de **tempo-virtual** são apresentadas na TABELA 2.1.

TABELA 2.1 - Classificação das propriedades por tipo de sistema.

	TIPO DE SISTEMA		
	Online	Tempo-virtual	Tempo-real
“Reliability”	IRRELEVANTE	DESEJÁVEL	ESSENCIAL
“Predictability”	IRRELEVANTE	DESEJÁVEL	ESSENCIAL
“Scheduling”	DESEJÁVEL	DESEJÁVEL	ESSENCIAL
“Queues”	DESEJÁVEL	IRRELEVANTE	ESSENCIAL
“Multiplexing”	DESEJÁVEL	IRRELEVANTE	DESEJÁVEL
“Multiprocessing”	IRRELEVANTE	IRRELEVANTE	DESEJÁVEL
“Switchover”	IRRELEVANTE	IRRELEVANTE	ESSENCIAL
“Fall-back”	IRRELEVANTE	IRRELEVANTE	ESSENCIAL
“Dependability”	DESEJÁVEL	DESEJÁVEL	ESSENCIAL
“Synchronization”	DESEJÁVEL	DESEJÁVEL	ESSENCIAL
“Reusability”	IRRELEVANTE	DESEJÁVEL	ESSENCIAL
“Connectivity”	IRRELEVANTE	IRRELEVANTE	ESSENCIAL
“Multitask”	IRRELEVANTE	DESEJÁVEL	ESSENCIAL

FONTE: Kavi (1992)

Do ponto de vista das conseqüências, as características de um **sistema de tempo real crítico** são: nenhum atraso é aceitável em quaisquer circunstâncias, um resultado atrasado é considerado um resultado errado, uma falha no sistema provoca uma catástrofe e o custo da não execução de uma tarefa é infinitamente grande. Um bom

exemplo de um sistema de tempo real crítico é um sistema de controle “fly-by-wire” de uma aeronave instável. De forma análoga, para um **sistema de sistema de tempo real não crítico**, um aumento de custo para o atraso dos resultados e o baixo desempenho causado por atraso ou não execução de tarefas são aceitáveis.

As diferenças entre um sistema de tempo real crítico ou não crítico dependem dos requisitos do sistema: ele é crítico se **não pode** perder o processamento de nenhuma tarefa e ele é não crítico, se **não deve** perder nenhuma tarefa. Existe uma grande discussão na literatura entre as diferenças entre estes dois tipos de sistemas; entretanto na área de aeronáutica, eles têm um significado claro na classificação de sistemas estabelecida pela norma DO-178B/ED-12B, e este será o utilizado neste trabalho. “**Sistema de tempo real crítico** é aquele que sua falha provoca um efeito catastrófico (“catastrophic”), ou seja, a perda da aeronave e a morte de um grande número de passageiros. Sua probabilidade de falha deve ser inferior a 10^{-9}. **Sistema de tempo real não crítico** é aquele cuja falha provoca um dano muito grande (“hazardous”), ou seja, provoca danos parciais na aeronave com ferimentos de passageiros e até mesmo a morte de algum passageiro. Sua probabilidade de falha deve ser inferior a 10^{-7}”.

Devemos salientar que não existe uma similaridade entre sistemas de tempo real críticos ou não-críticos com sistemas operacionais críticos ou não críticos. Não existe a classificação de sistemas operacionais de tempo-real críticos ou não críticos. De qualquer forma um sistema operacional de tempo real é um sistema operacional que pode ser usado para se construir sistemas de tempo real críticos.

2.2.1 Definições na Área Aeronáutica

De acordo com os regulamentos aplicáveis FAA AC 25.1309-1A e JAA AMJ 25.1309, o documento emitido conjuntamente pelo RTCA e EUROCAE, chamado DO-178B/ED-12B, é o guia para desenvolvimento de software embutido em aeronaves aceito pelo FAA e JAA para cumprimento dos requisitos de software. A circular AC 20-115B cancela as versões preliminares da DO-178. A proposta da DO-178B/ED12B é prover diretrizes para a produção de programas digitais de computadores embutidos e

equipamentos que desempenhem suas funções com os níveis de segurança que atendam os requisitos de segurança de vôo. O CTA brasileiro acompanha o FAA e o JAA nas mesmas diretrizes.

O processo de análise de segurança de sistemas, conhecido no meio aeronáutico como “System Safety Assessment” (SSA), a partir dos requisitos operacionais e de segurança de vôo, determina e classifica as condições de falha dos sistemas, depois define os requisitos de segurança que devem ser implementados em software ou hardware. A partir dessas informações os requisitos de sistemas são alocados aos programas (“software”), são definidos os níveis de cada programa, as restrições de projeto e a definição do hardware. Todo este conjunto alimenta o processo do ciclo de desenvolvimento de software que, por sua vez, realimenta o processo SSA com arquitetura e requisitos de software, fontes de erros identificadas e eliminadas e os limites e condições das falhas que possam ocorrer. O processo de definição do hardware e do respectivo sistema operacional de tempo real pode ser antecipado devido à experiência no desenvolvimento de projetos anteriores. Esta interação é representada na FIGURA 2.1.

O processo SSA determina o nível de software adequado a cada componente de software de um determinado sistema sem considerar o projeto do sistema. Se um erro de software pode causar um defeito que contribui para uma condição de falha, então, o nível de integridade do software necessário para a operação segura está relacionado com as condições de falha do sistema. Se o comportamento anômalo de um componente de software contribui para mais de uma condição de falha, então a categoria da condição mais severa de falha daquele componente determina o nível de software daquele componente. As categorias das condições de falha de um sistema são estabelecidas determinando-se a severidade das conseqüências das condições de falha na aeronave e seus ocupantes. É apresentada a seguir a definição dos níveis de software conforme o documento DO-178B, seção 2.2.2, a partir do processo de avaliação de segurança da aeronave, e resumida na TABELA 2.2.

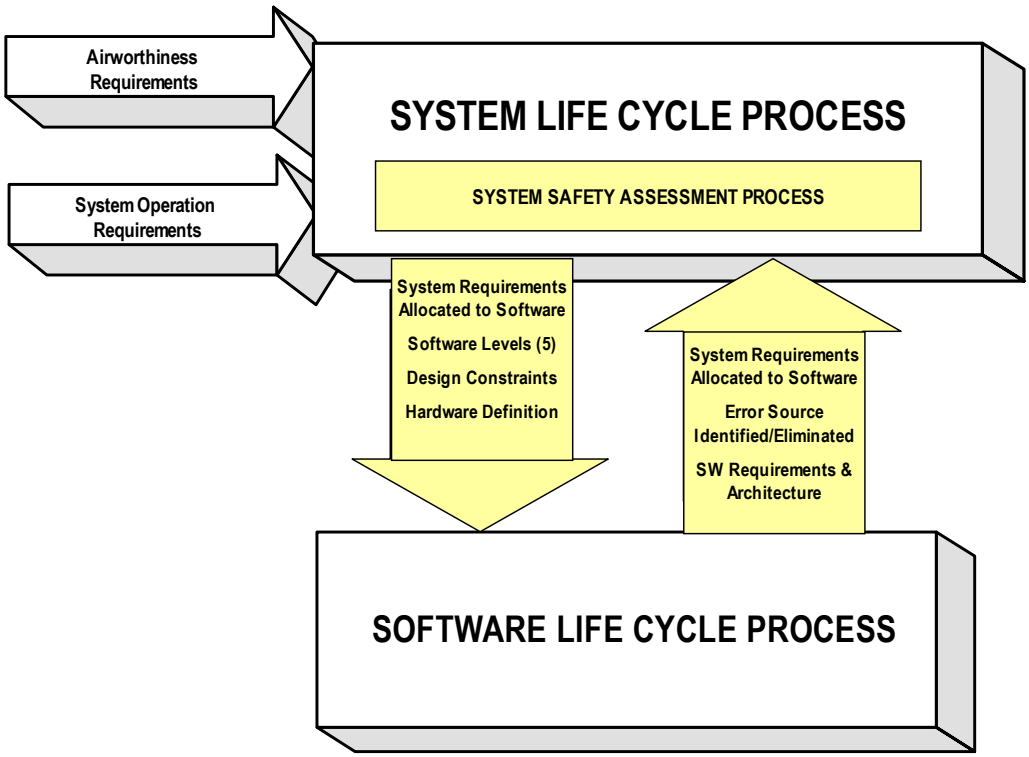

FIGURA 2.1 - Fluxo de informação entre os processos de ciclo de vida de sistema e de software.

FONTE: RTCA (1992)

TABELA 2.2 – Níveis de software conforme DO-178B, seção 2.2.2.

Categoria	Significado do regulamento	Probabilidade	Nível de SW
catastrófica ("catastrophic")	Aeronave destruída, muitas fatalidades	$< 10^{-9}$	A
danosa ("hazardous")	Danos na aeronave, ferimento nos passageiros, alguma fatalidade	$< 10^{-7}$	B
emergência ("major")	Sobrecarga no piloto, dificuldade de resolver as emergências	$< 10^{-5}$	C
pequenos ("minor")	Efeitos pequenos nos tripulantes ou passageiros	N/A	D
sem efeito ("no effect")	Sem efeito na operação da aeronave	N/A	E

FONTE: RTCA (1992)

Nível A - Software cujo comportamento anômalo, conforme demonstrado no processo de avaliação de segurança do sistema, poderia causar ou contribuir para uma falha de uma funcionalidade do sistema resultando em uma condição de falha catastrófica (“catastrophic”) da aeronave levando à perda da mesma e de muitas vidas humanas.

Nível B - Software cujo comportamento anômalo, conforme demonstrado no processo de avaliação de segurança do sistema, poderia causar ou contribuir para uma falha de uma funcionalidade do sistema resultando em uma condição de falha danosa (“hazardous”) da aeronave levando a danos parciais na aeronave com ferimentos de passageiros e alguma eventual fatalidade.

Nível C - Software cujo comportamento anômalo, conforme demonstrado no processo de avaliação de segurança do sistema, poderia causar ou contribuir para uma falha de uma funcionalidade do sistema resultando em uma condição de emergência (“major”) da aeronave colocando o piloto em dificuldades de gerenciamento das emergências.

Nível D - Software cujo comportamento anômalo, conforme demonstrado no processo de avaliação de segurança do sistema, poderia causar ou contribuir para uma falha de uma funcionalidade do sistema resultando em uma condição de falha com pequenos (“minor”) efeitos nos tripulantes ou passageiros da aeronave.

Nível E - Software cujo comportamento anômalo, conforme demonstrado no processo de avaliação de segurança do sistema, poderia causar ou contribuir para uma falha de uma funcionalidade do sistema resultando em uma condição de falha sem efeito (“no effect”) na capacidade operacional da aeronave ou carga de trabalho do piloto.

Várias estratégias de arquitetura dos componentes de software influenciam na determinação do nível daquele componente de software: forma de divisão em componentes, dissimilaridade por uso de diferentes versões, monitoramento de segurança e categoria de componente (carregado em campo, modificável pelo usuário ou configurado pelo usuário).

2.3 Sistemas Síncronos ou Assíncronos

Sistema síncrono é tipicamente um processo de tempo real periódico, representado por uma tarefa síncrona que mostra dados e/ou executa um controle em uma malha fechada em que uma ou diversas tarefas devem ser executadas, podendo ser de freqüências iguais ou diferentes. Essas tarefas que devem ser executadas em todos os ciclos são definidas como tarefas síncronas.

Sistema assíncrono é um processo aperiódico, representado por tarefas assíncronas disparadas por eventos assíncronos, esporádicos ou aleatórios, geralmente de origem externa ao computador considerado. Um exemplo típico de tarefa assíncrona é o acionamento de flapes e trem de pouso de aeronaves militares ou o comando do operador de ligar e desligar um equipamento em um satélite. Em geral um sistema síncrono também contém tarefas assíncronas.

2.4 Sistemas Paralelos ou Distribuídos

Constantemente o conceito de Simulação Distribuída é confundido com Simulação Paralela da mesma forma que Processamento Distribuído é confundido com Processamento Paralelo. Esses conceitos estão intimamente ligados e, desta forma, é necessária a compreensão das diferenças entre computadores de processamento paralelo e computadores de processamento distribuído para a compreensão das diferenças entre as correspondentes simulações.

Recentemente a distinção entre computadores de processamento paralelo e distribuído tornou-se nebulosa com o advento das redes de “workstations”, formadas por um grupo de “workstations” interconectadas através de portas de alta velocidade confinadas em uma mesma sala. Considerando que essas novas técnicas de interconexão não usam os protocolos tradicionais de comunicação, o conjunto formado por elas é considerado como um computador paralelo. O **atraso de comunicação (“latency time”)** entre as máquinas é a propriedade mais importante para diferenciar computadores paralelos e distribuídos. Este atraso é tipicamente de alguns micro-segundos a dezenas de

milisegundos em computadores paralelos e, muito maior em computadores distribuídos principalmente devido às longas distâncias físicas que o sinal deve atravessar e à complexidade do software de protocolos de comunicação necessário para se efetivar a conexão, conforme resumido na TABELA 2.3.

TABELA 2.3 – Diferenças entre computadores paralelos e distribuídos.

Propriedade	Computadores Paralelos	Computadores Distribuídos
Distribuição física	Salas	De prédio até global
Processadores	Semelhantes	Diferentes
Redes de comunicação	Dedicadas	LAN ou WAN comercial
Atraso de comunicação	Menor que 100 ms	De 100 ms até segundos

De uma maneira geral, segundo o tipo de processamento, os computadores podem ser subdivididos nas categorias apresentadas na com as principais diferenças.

Simulações executadas em computadores com vários processadores utilizando **memória compartilhada ("shared memory")**, multicomputadores com memória distribuída ou máquinas "Single Instruction Multiple Data" (SIMD) são definidas como programas de simulação paralela.

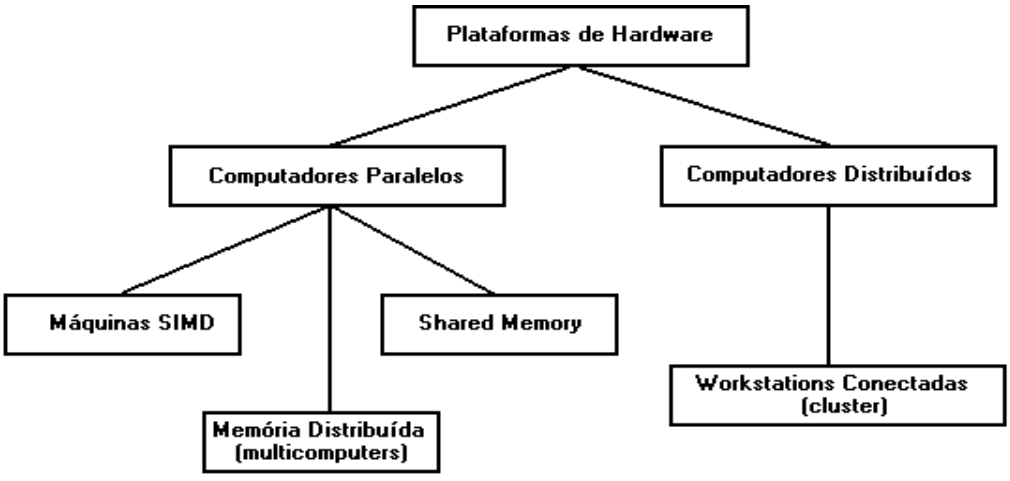

FIGURA 2.2 – Classes de computadores paralelos e distribuídos.
FONTE: Fujimoto (2000)

Simulações executadas em computadores distribuídos são definidas como simulação distribuída. A simulação distribuída é usada com objetivos analíticos ou mais popularmente para a construção de ambientes virtuais distribuídos. Considerando que a construção de ambientes virtuais é a aplicação mais comum encontrada na literatura, às vezes, o termo “simulação distribuída” refere-se exclusivamente a ambientes virtuais distribuídos.

2.5 Programas Agendadores/Escalonadores

Para os sistemas de tempo real, segundo Burns e Wellings (1990), não é suficiente o software estar logicamente correto, pois os programas também devem satisfazer requisitos de tempo delimitados pela natureza dos sistemas físicos. Infelizmente as práticas atuais de engenharia para desenvolvimento de grandes sistemas de tempo real, em geral, ainda são “ad hoc”. Frequentemente, as diversas tarefas são especificadas, desenvolvidas, programadas e testadas pelos requisitos lógicos. Só após isso são verificadas se atendem os requisitos de tempo. Em caso negativo, são feitas revisões e ajustes, incluindo **defasagem de tarefas** (“**skew**”) se necessário. Quando é necessária a atualização ou o acréscimo de tarefas em um sistema de tempo real, a compreensão do agendamento destas tarefas é difícil e, geralmente, provoca a necessidade de troca do hardware. Assim, faz-se necessário o uso de métodos mais formais durante a especificação dos requisitos de tempo em sistemas de tempo real complexos.

Os processos presentes em sistemas de tempo real podem ser classificados como: periódicos ou aperiódicos (esporádicos, aleatórios, etc.), “**soft” ou “hard real time**“. Neles, todos os processos devem ser analisados até mesmo no pior caso de tempo de execução. A noção de **prioridade** é importante em função dos danos que a não execução de cada tarefa possa causar ao sistema e é considerada, em tempo de execução, pelo **administrador/executor/ despachador** (“**dispatcher**”), para indicar qual a ordem em que as tarefas devam ser executadas. Antes de colocar as tarefas na fila de execução, os sistemas operacionais de tempo real executam o agendamento feito pelo **agendador** (“**scheduler**”) das tarefas segundo critérios bem definidos. O agendamento envolve a alocação de tempo e recursos necessários às tarefas para que elas atendam aos

requisitos de desempenho do sistema. Uma boa revisão bibliográfica com conceitos de sistemas de tempo real e apresentação dos diversos tipos de agendadores foi compilada por Farines et al (2000) no evento bi-anual Escola da Computação realizado pelas universidades brasileiras.

2.5.1 Estratégias Para Classificação de Agendadores

A classificação dos programas agendadores segundo diferentes estratégias é bastante vasta na literatura e, em particular, um bom resumo é apresentado por Ramamritham e Stankovic (1994). Essas estratégias são baseadas em: i) se o sistema calcula a agendabilidade, ii) se o faz estática ou dinamicamente, e iii) se o resultado desta análise produz um agendamento ou plano para despachar a execução das tarefas em tempo de execução.

2.5.2 Outros Aspectos Para Classificação de Agendadores

Existem ainda outros aspectos importantes em sistemas de tempo real que justificam o agrupamento em classes de agendadores a ele relacionados: o suporte a tolerância a falhas, o aumento de desempenho com o uso do tempo não utilizado do processador, e a degradação dos cálculos efetuados em condições de sobrecarga.

2.5.3 Classificação de Agendadores de Tarefas Síncronas

Dentre as diversas classes apresentadas anteriormente, destacamos os agendadores mais utilizados na literatura, apresentados também por Burns e Welling (1990), Microsoft Corporation (1995) e Integrated Systems (1995), e utilizados neste trabalho.

As tarefas síncronas de um processo de tempo real periódico, podendo ser de freqüências iguais ou diferentes, devem ser executadas em todos os seus ciclos. Os agendadores desse tipo de tarefa são classificados principalmente segundo os critérios a seguir.

2.5.3.1 Agendadores Baseados em Prioridade

Classe conhecida em Inglês como “Priority Driven Scheduling – PDS”, representa os algoritmos onde são atribuídos níveis diferentes de prioridades para as diversas tarefas. Esta prioridade é definida pelo usuário segundo um critério que lhe pareça mais adequado ao problema: risco, preferência, geração de dados, etc. Cada tarefa pode ter um nível de prioridade, ou diversas tarefas podem ter níveis de prioridade iguais. O número de níveis de prioridade é uma característica do sistema operacional e, geralmente para sistemas complexos, é bem inferior ao número de tarefas a serem executadas. Segundo este critério a tarefa de maior prioridade deve sempre ser executada primeiro. Nos sistemas de tempo real “soft”, a tarefas que podem deixar de ser cumpridas são colocadas como as de menor prioridade.

2.5.3.2 Agendadores Baseados em Prioridade Com Herança

Classe conhecida em Inglês como “Priority Inheritance Driven Scheduling – PIDS”, representa os algoritmos que ampliam o conceito anterior acrescentando-se a inversão automática de prioridades quando a tarefa que está sendo executada depende de uma informação ou recurso de uma tarefa de prioridade inferior. Este é um critério mais apropriado para sistemas não tão críticos onde as tarefas usam recursos comuns, mas é desaconselhável para sistemas ”life-critical”. A necessidade do uso de uma informação ou recurso de uma tarefa de menor prioridade na execução de uma tarefa de maior prioridade é um indicativo forte de um problema de projeto que pode e deve ser resolvido de outra forma.

2.5.3.3 Agendadores de Taxa Constante

Classe conhecida em Inglês como “Rate Monotonic Scheduling - RMS”, representa os algoritmos onde a prioridade de execução é sempre da tarefa de maior freqüência no sistema.

2.5.3.4 Agendadores Com Primeiro Prazo Fatal Primeiro

Classe conhecida em Inglês como “Earliest Deadline First Scheduling - EDFS”, representa os algoritmos onde a prioridade de execução das tarefas é definida pelo momento em que cada tarefa irá terminar. Nestes casos são considerados os términos das tarefas e, aquela cujo final está mais próximo, será executada primeiro. Este é um critério que tenta sempre a execução de todas as tarefas. Entretanto se alguma tarefa deixar de ser cumprida, é difícil identificá-la em tempo de projeto.

2.5.4 Classificação de Agendadores de Tarefas Assíncronas

As tarefas assíncronas de um processo são aquelas disparadas por eventos assíncronos, esporádicos ou aleatórios, geralmente de origem externa ao computador. Alguns tipos são apresentados na FIGURA 2.3 e descritos a seguir.

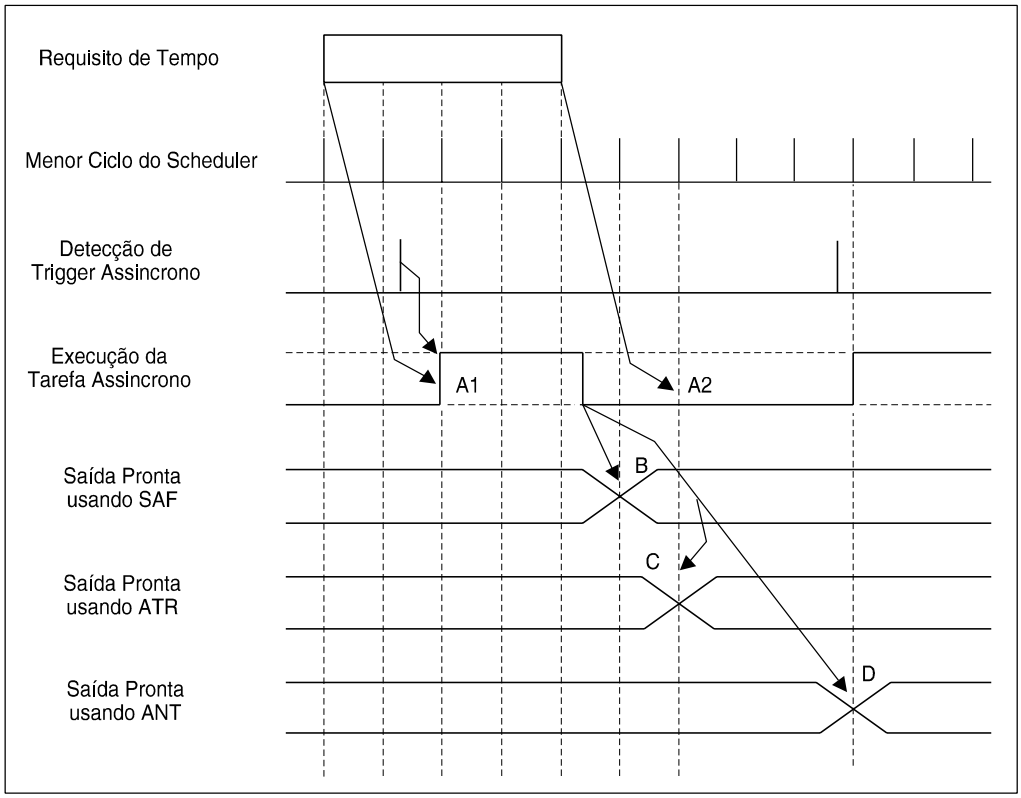

FIGURA 2.3 – Tarefa assíncrona com diferentes tipos de agendamento.
FONTE: Integrated Systems (1984)

As tarefas assíncronas de um sistema de tempo real representam funções complementares de outras tarefas síncronas já existentes. Um exemplo típico destas tarefas é o acionamento de flapes e trem de pouso de aeronaves militares ou o comando do operador de ligar e desligar um equipamento em um satélite.

2.5.4.1 Agendadores o Mais Cedo Possível

Classe conhecida em Inglês como “As Soon As Possible Scheduling – ASAPS” ou “Soon As Finished – SAF”, representa os algoritmos onde uma tarefa assíncrona deve ser executada pelo sistema operacional no tempo mais cedo possível sem afetar a execução de qualquer tarefa cíclica. Evidentemente a análise do pior caso deve ser feita para verificar: a possibilidade da tarefa ser executada ou não, o número mínimo de ciclos necessários para a sua execução e, até mesmo, como será tratada a concorrência com uma outra atividade também assíncrona.

2.5.4.2 Agendadores Depois de Um Tempo Definido

Classe conhecida em Inglês como “After Time Requirement Scheduling – ATR”, representa os algoritmos onde uma tarefa assíncrona deve ser executada pelo sistema operacional e sua saída deve ocorrer em um tempo definido. As análises de possibilidade devem ser feitas da mesma forma que no critério anterior. O resultado final, em geral um comando de saída, deve ser executado em um tempo bem definido. Este caso é aplicado quando se faz um pré-processamento anterior considerando um tempo exato para não se introduzir erros no resultado final (ex. momento de um disparo de jato de gás para correção de órbita, ou momento do lançamento de um míssil).

2.5.4.3 Agendadores Após o Próximo Gatilho

Classe conhecida em Inglês como “After Next Trigger Scheduling - ANT”, representa os algoritmos onde uma tarefa assíncrona deve ser executada pelo sistema operacional e aguardar um próximo sinal de **gatilho ("trigger")** para ser concluída e sua ação enviada

ao sistema. Este critério é importante quando se quer confirmar a execução de uma tarefa eliminando-se eventuais sinais espúrios na sua origem.

2.6 Sincronismo, Comunicação, Sustação e Reentrância

Os sistemas de tempo real críticos necessitam de um **sistema operacional de tempo real (SOTR)** onde os serviços são definidos em termos temporais além dos funcionais. Embora seja possível construir um sistema de tempo real crítico utilizando um **SOTR mono-tarefa ou seqüencial ("single task")** sob condições especiais, um sistema capaz de reagir a eventos externos aleatórios, deve ter a capacidade de processamento multitarefa ("multitask"). **Sistemas de processamento multitarefa ou simultâneos ("multitask") são** aqueles que possuem a capacidade de executar diversas tarefas simultaneamente, necessariamente com cooperação e troca de informações entre elas. **Uma tarefa** ("task"), também chamada de **processo ("process")**, é uma representação de um contexto de execução de um programa contendo espaço de memória próprio, registradores alocados, arquivos abertos, regras de acesso, entre outros recursos que variam para cada sistema operacional. Os estados possíveis para uma tarefa em um sistema operacional de tempo real são apresentados na FIGURA 2.4. A capacidade de operação "multitramas" ("*multithread*") leva a idéia de operação multitarefa para dentro de cada aplicação, de forma que uma simples aplicação possa ser subdividida em "**tramas**" ("***threads***") individuais que possam ser executadas em paralelo. São tarefas mais simples e leves pois estão restritas somente ao contexto de execução e, portanto, associadas aos registradores do processador. Todos os demais recursos são herdados da tarefa que os hospeda e, consequentemente, o tempo de chaveamento de contexto entre "*threads*" é menor que entre tarefas. Devido a isto um sistema de tempo real crítico utilizando "*threads*" é mais eficiente que um utilizando somente tarefas. Um SOTR deve prover para as "*threads*" o mesmo suporte de criação, destruição, sustação e reentrância disponibilizados para tarefas.

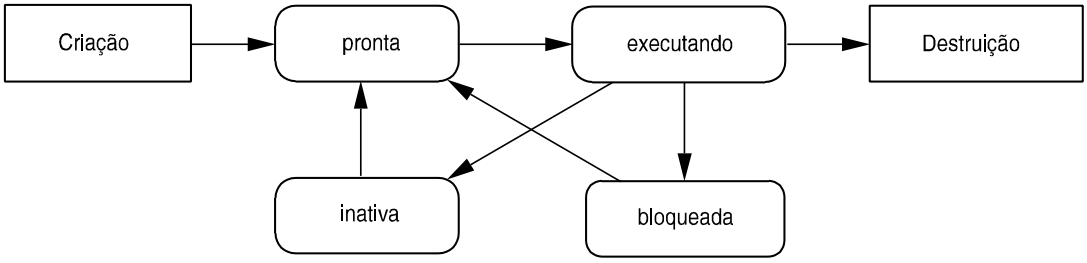

FIGURA 2.4 – Estados das tarefas em um SOTR.
FONTE: Farines et al (2000)

Posto que um sistema de tempo real utiliza tarefas e "*threads*" que cooperam e trocam informações entre si, torna-se necessário o sincronismo e a comunicação entre as mesmas. **Sincronismo** é o cumprimento das restrições temporais no entrelaçamento de ações de diferentes processos. **Comunicação** é a passagem de informação de um processo a outro. Os dois conceitos estão diretamente ligados porque é necessário haver sincronismo para haver comunicação, e a sincronização pode ser considerada uma comunicação sem conteúdo. Basicamente existem duas formas de sincronismo e comunicação entre tarefas: através de variáveis compartilhadas ou através de mensagens.

Variáveis compartilhadas são objetos que mais de um processo pode acessar. A sincronização é explicitada por mecanismos como indicadores de ocupado ("busy waiting" ou "spinning"), ou semáforos ("semaphores"), regiões críticas condicionais ("Conditional Critical Regions" – CCRs), monitores ("monitors"), ou outro mecanismo similar. Quando os processos são executados em computadores diferentes, o mecanismo de **compartilhamento de memória virtual** é implementado por hardware, através de **cartões** "**shared memory**" que possuem um custo mais elevado.

A troca de **mensagens** envolve explicitamente a troca de dados entre dois processos por meio de uma mensagem que passa entre um processo e outro, via operações de enviar e receber mensagens. O objetivo a ser buscado é utilizar uma única construção para o sincronismo e a comunicação. Existe uma grande variedade de combinações possíveis

que são definidas considerando estas três características: modelo e sincronismo, método de passagem de nome (“process naming”), e a estrutura da mensagem.

Sustação ou preempção (“**preemption**”) é a propriedade da tarefa ou “*thread*” que pode ser suspensa por um SOTR a qualquer momento. **Reentrância** é a capacidade que o SOTR deve ter de identificar qual tarefa foi suspensa anteriormente por outra de maior prioridade, e de retomar a execução da 1ª após o término da 2ª. Considerando que estas duas operações utilizam parte do precioso recurso “tempo”, torna-se necessário o conhecimento destes tempos e a análise se eles são desprezíveis ou não. Desta forma é necessário compreender como estes tempos são formados e definir cada um dos seus componentes, como se faz a seguir.

Período de execução de uma tarefa *Ti* é o intervalo de tempo cíclico em que é necessária a execução de uma mesma tarefa.

Tempo de execução de uma tarefa *Ci* é o tempo necessário para o processador executar uma tarefa sem nenhuma interrupção.

Tempo de chaveamento entre tarefas (“task switching time”) é definido como o tempo médio que o sistema gasta para mudar entre duas tarefas ativas e independentes de mesma prioridade e é representado na FIGURA 2.5.

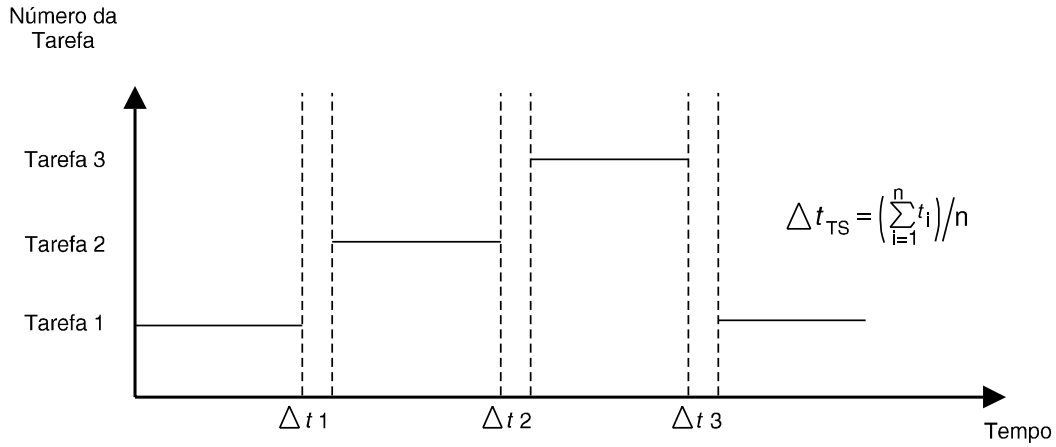

FIGURA 2.5 – Tempo de chaveamento entre tarefas.

FONTE: Furht et al (1991)

Tempo de sustação/preempção (“preemption time”) é definido como o tempo médio que o sistema leva para transferir o controle de uma tarefa de menor para outra de maior prioridade. O conceito é similar ao tempo de chaveamento, entretanto é mais longo porque o SOTR deve reconhecer a solicitação de maior prioridade, acessar as prioridades de ambas as tarefas e depois transferir o controle.

Tempo de latência de interrupção (“interrupt latency time”) é definido como o tempo entre o momento que o processador recebe uma solicitação de interrupção até o momento que executa a primeira instrução da rotina que emitiu a solicitação e é representado na FIGURA 2.6.

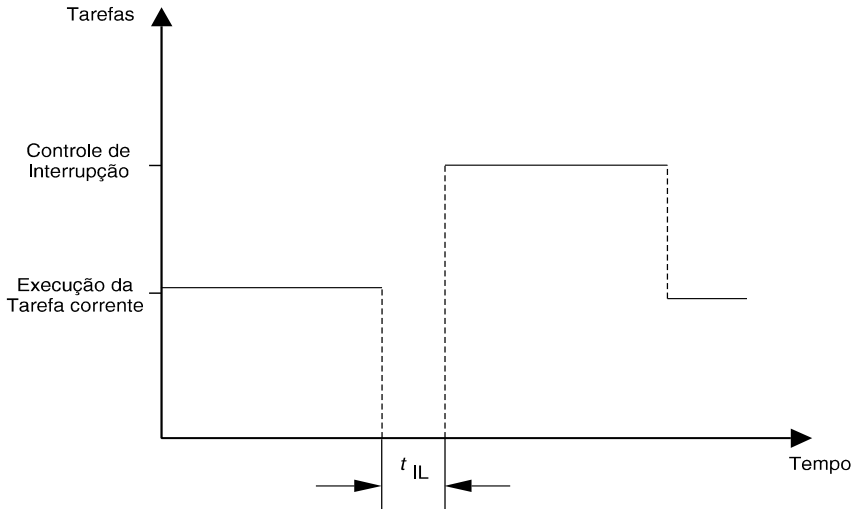

FIGURA 2.6 – Tempo de latência de interrupção.
FONTE: Furht et al (1991)

Tempo de chaveamento de semáforo (“semaphore shuffling time”) é o atraso de tempo entre a liberação de um semáforo por uma tarefa até a ativação de outra tarefa que estava esperando o semáforo e é representado na FIGURA 2.7.

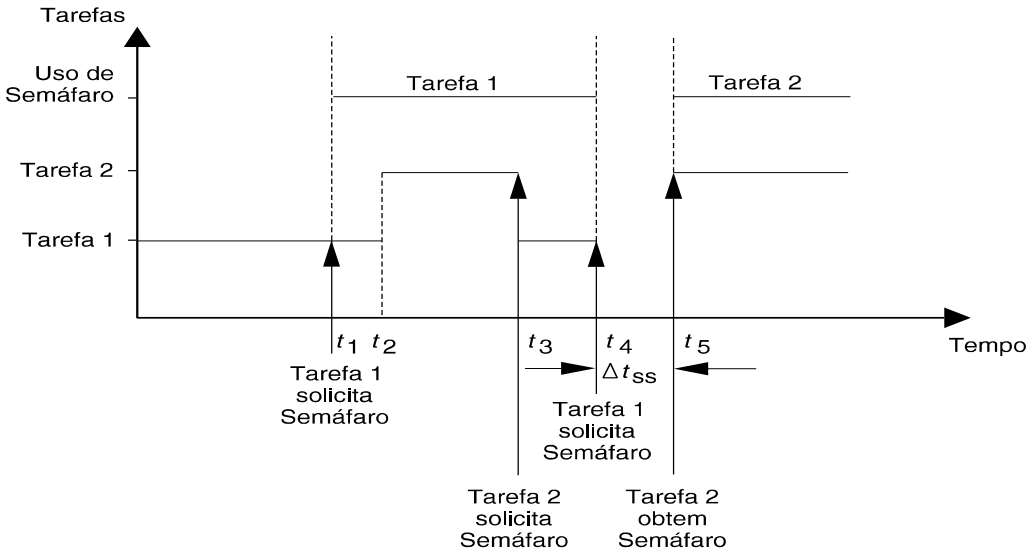

FIGURA 2.7 – Tempo de chaveamento de semáforo.

FONTE: Furht et al (1991)

Tempo de solução de travamento (“deadlock breaking time”) é o tempo médio que o sistema gasta para solucionar um travamento por disputa de recursos. Portanto é a soma do tempo de detecção do problema e o tempo da solução do mesmo conforme representação na FIGURA 2.8.

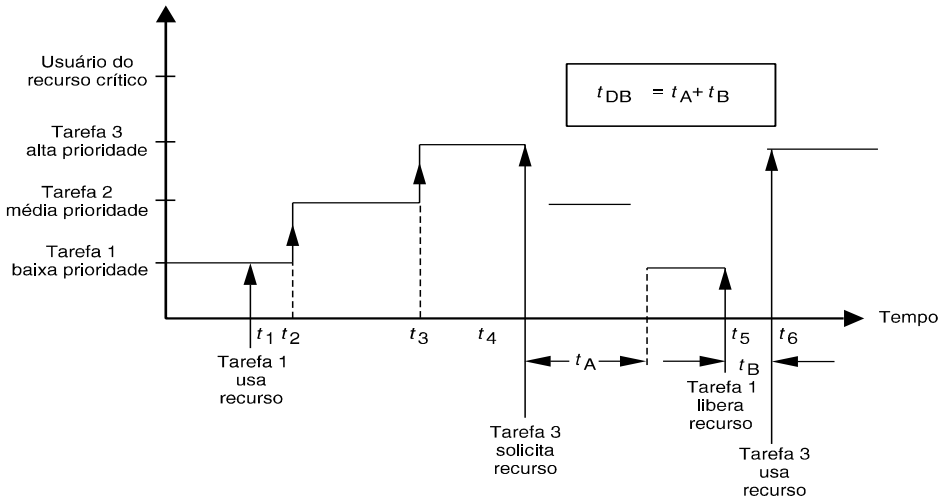

FIGURA 2.8 – Tempo de solução de travamento.

FONTE: Furht et al (1991)

“Datagram Throughput” é o número de kilobytes por segundo que uma tarefa pode enviar a outra via chamadas às primitivas do SOTR.

A soma das frequências dadas pelos inversos dos tempos: tempo de chaveamento entre **tarefas, tempo de sustação/preempção, tempo de latência de interrupção, tempo de** chaveamento de semáforo, tempo de solução de travamento, com a frequência dada pelo “datagram throughput” compõem uma 1ª métrica para SOTRs segundo Fuhrt et al (1991).

Tempo de latência de despacho de um processo (“process dispatch latency time”) pode ser definido como o intervalo de tempo desde quando o sistema recebe uma solicitação de interrupção até o início da execução da tarefa correspondente à resposta do sistema de tempo real. É uma medida frequentemente utilizada para avaliação de sistemas de tempo real. Como a latência (atraso) é referente ao sistema, devem ser considerados outros tempos conforme a FIGURA 2.9**Erro! Fonte de referência não encontrada.**.

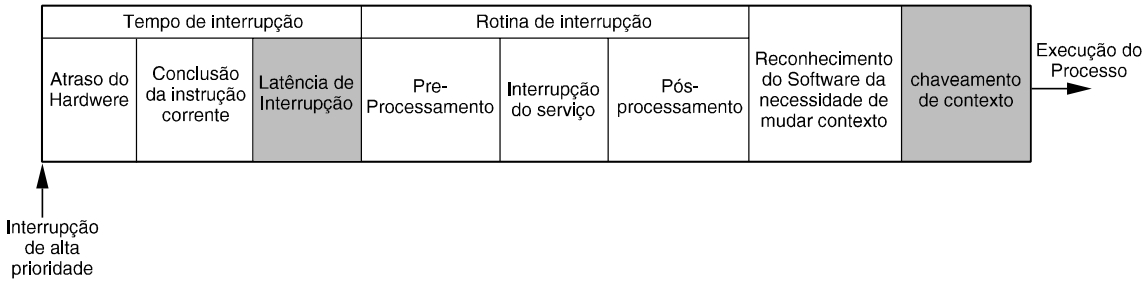

FIGURA 2.9 - Tempo de latência de despacho de um processo.
FONTE: Furht et al (1991)

Tempo de resposta de interrupção (“interrupt response time”) está relacionado com os atrasos tanto do hardware como do software O **atraso de hardware** é o tempo necessário para que a notificação explícita de interrupção emitida pelo hardware chegue ao sistema operacional. Evidentemente o sistema operacional deve **concluir a instrução corrente** e começar o gerenciamento para interrupção das funções (ex. salvar variáveis de estado) provocando uma **latência de interrupção**. Então o sistema passa para a fase

de **rotina de interrupção ("interrupt routine")** em três fases distintas: **pré-processamento**, **interrupção do serviço** e **pós-processamento**. Somente neste instante é que o programa reconhece a necessidade de **mudança de contexto** e, irá efetuá-la.

O tempo de latência de despacho de um processo é uma 2ª métrica do desempenho de sistemas operacionais de tempo real e, em particular, tem uma importância muito grande em aeronaves militares de combate.

Para se medir o desempenho de sistema de tempo real não devemos considerar somente a capacidade e velocidade de processamento disponível. Uma 3ª métrica deste desempenho deve considerar pelo menos a capacidade de resposta, a capacidade de sustação e a velocidade de entrada e saída de dados. Uma boa definição para esta métrica é apresentada por Furht et al (1988) e é dada pelo volume do sólido formado pela representação tri-dimensional das variáveis: a) velocidade de processamento da CPU medido em MIPS1 (milhões de instruções por segundo), b) capacidade de gerenciamento de interrupção medida em MIPS2 (milhões de interrupção por segundo) e, c) fluxo de entrada e saída medido em MIPS3 (milhões de operação de entrada e saída em Mbytes/sec por segundo). A FIGURA 2.10 mostra uma visão gráfica das capacidades de um sistema de tempo real.

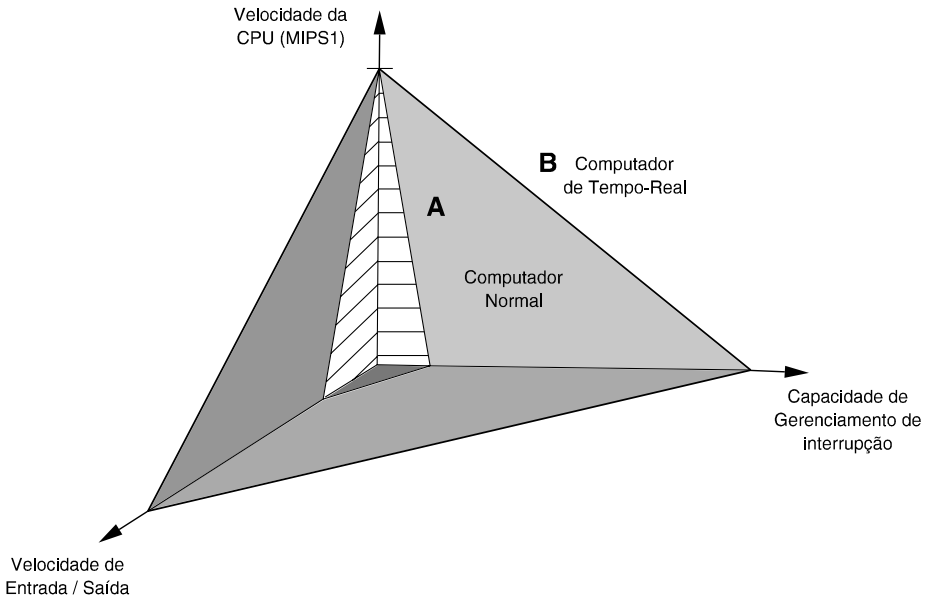

FIGURA 2.10 - Medid*a* de desempenho de computadores de sistemas de tempo real.
FONTE: Furht et al (1991)

As definições de tempo relacionadas explicitamente com as tarefas são apresentadas nos capítulos teóricos em que são necessárias.

2.7 O Departamento de Defesa Americano (DoD) e a Simulação Distribuída

Um dos mais antigos usos da simulação é a aplicação do exercício preparatório da guerra, de onde surgiram os fundamentos da Pesquisa Operacional. No mundo moderno, após o advento dos computadores, o DoD comandando a mais poderosa máquina de guerra jamais visto na Terra, tornou-se o maior usuário e também maior pesquisador da área de simulação. O uso, pesquisas e avanços aconteceram tanto na área de Sistemas Discretos como na área de Sistemas Dinâmicos embora em áreas distintas ou, até mesmo, em forças armadas distintas.

2.7.1 Termos e Definições do DoD

De acordo com o "Defense Science Board" dos EUA, as simulações militares podem ser classificadas em três diferentes categorias:

Simulação ao vivo: ("Live Simulation"). Envolve pessoas reais em sistemas reais. Testes operacionais e exercícios são exemplos deste tipo.

Simulação virtual: ("Virtual Simulation"). Envolve pessoas reais em sistemas simulados. Incluem-se nessa classificação os simuladores de aviões e de tanques. Esse tipo de simulação é muito utilizado para avaliação de controle, decisão e comunicação.

Simulação construtiva: ("Constructive Simulation"). Humanos podem ou não interagir com modelos, mas estes e tudo o mais são simulados. São utilizadas pelos corpos de elite de forças armadas em treinamentos e análise de planos e estabelecimento de novas estratégias de ataque e preparação de missão.

2.7.2 Breve Histórico da Simulação Distribuída no DoD

As primeiras simulações militares distribuídas para ambientes virtuais começaram na década de 80 com o projeto denominado SIMNET (“SIMulator NETworking”) que viria mais tarde tornar-se o padrão “Distributed Interactive Simulation” (DIS) onde eram definidos os padrões para suportar a interoperabilidade entre simuladores de treinamento autônomos em ambientes de simulação distribuídos geograficamente. O segundo maior impulso a partir do SIMNET foi o protocolo ALSP (“Aggregate Level Simulation Protocol”) que expandiu o conceito de interoperabilidade para simulações de jogos de guerra.

O maior avanço da tecnologia teve a sua origem no desenvolvimento do “Modeling & Simulation Master Plan” (inicialmente chamado M&S HLA) iniciado na “Defense Advanced Research Projects Agency” (DARPA) pelo “Advanced Simulation Program” (ADS) que definiu os objetivos básicos desejáveis para a área de simulação no DoD (“Department of Defense”):

- Desenvolver uma infra-estrutura técnica comum (“framework”) de simulação (“High Level Architecture”, Modelos conceituais no espaço de missões; e Padronização de dados);

- Prover modelos do ambiente natural com grande representatividade (Terrenos, Oceanos, Atmosfera e Espaço);

- Prover modelos representativos de sistemas;

- Prover modelos representativos do comportamento humano (indivíduos e grupos ou organizações);

- Estabelecer uma infra-estrutura de M&S para suprir as necessidades básicas de desenvolvedores e usuários finais (Sistemas de campo, VV&A, repositórios, comunicações e centro de coordenação);

- Distribuir os benefícios da simulação (Impacto de qualidade, educacional e uso dual);

Após anos de trabalho coordenados pelo DMSO (“Defense Modeling & Simulation Office”) do DoD e um trabalho de interação de mais de três anos desenvolvido com o IEEE (“Institute of Electrical and Electronic Engineers”), a arquitetura HLA foi definida como norma internacional do IEEE em 2000. A “RunTime Infrastructure” atual, chamado RTI 1.3NG, foi confirmado como padrão industrial, usando processo aberto de desenvolvimento competitivo, de maneira a incrementar e baratear o desenvolvimento de software de simulação.

A HLA é importante porque provê uma arquitetura única que cobre tanto as simulações analíticas e as simulações de ambiente virtuais. Em certos aspectos a HLA representa uma generalização e uma extensão da união dos protocolos DIS a ALSP.

2.8 A Arquitetura HLA

HLA é uma arquitetura integrada de software desenvolvida para a criação de modelos e simulação computacional de sistemas ou para a criação de modelos e simulação computacional de componentes, visando a interoperabilidade e reuso. O padrão HLA atualmente é um padrão formado pelas seguintes normas IEEE:

- **IEEE P1516** – “Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules”;

- **IEEE P1516.1** – “Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules: Federation Interface Specification,” onde estão incluídos: “Federation Management, Declaration Management, Object Management, Ownership Management, Data Distribution Management, Support Services Management, Object Model (MOM) Specification, Federation Execution Data (FED)”;

- **IEEE P1516.2** – “Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – HLA Object Model Template (OMT)”, onde estão incluídos: “Object Model Template (OMT)”, “Federation Object Model (FOM)” e “Simulation Object Models (SOM)”;

As conexões da arquitetura HLA com o tema desta tese acontecem no gerenciamento dos recursos de tempo, mas também podem ser aplicados no gerenciamento e distribuição de recursos que se tornem escassos a partir do aumento do número de federados em tempo de execução de uma federação. Isto ficará evidente nas modificações e aplicações sugeridas no Capítulo 7.

2.8.1 Regras de Funcionamento do HLA

As normas da arquitetura HLA definem cinco regras básicas para federações (1 a 5) e cinco para os federados (6 a 10), a saber:

1) Federações devem estar definidas no “HLA Federation Object Model (FOM)”, seguindo o HLA OMT (“Object Model Template”);

2) Em uma federação, todas as instâncias de objetos associados a uma simulação devem ser definidas como federados e não incluídas na “RunTime Infrastructure (RTI)”. Isso é feito para garantir uma separação entre funcionalidades especificas da simulação e as funcionalidades de propósito gerais;

3) Toda a ligação de dados entre federados de uma FOM ocorre via a RTI;

4) Durante a execução de uma federação, os federados interagem com a RTI seguindo a “HLA Interface Specification (IS)”.

5) Durante a execução de uma federação, todos os atributos de instância devem ter posse definida por pelo menos um federado ao longo do tempo de execução;

6) Todos os federados devem estar definidos no “HLA Simulation Object Model (SOM)”, documentados de acordo com o HLA OMT;

7) Todos os federados devem estar aptos a modificar condições (atualizar, refletir, transmitir e receber) sobre as quais são atualizados os atributos de objetos;

8) Todos os federados devem estar aptos a transferir ou aceitar a posse de atributos dinamicamente durante uma execução de federação, especificadas pelos seus SOMs;

9) Todos os federados devem estar aptos a modificar as condições sobre as quais são providas as atualizações de atributos, especificadas por seus SOMs;

10) Todos os federados devem estar aptos a gerenciar o Tempo Local (o tempo do federado) de maneira a permitir a troca de dados coordenada com outros membros da federação;

De maneira simples, a arquitetura HLA pode ser representada como na FIGURA 2.11.

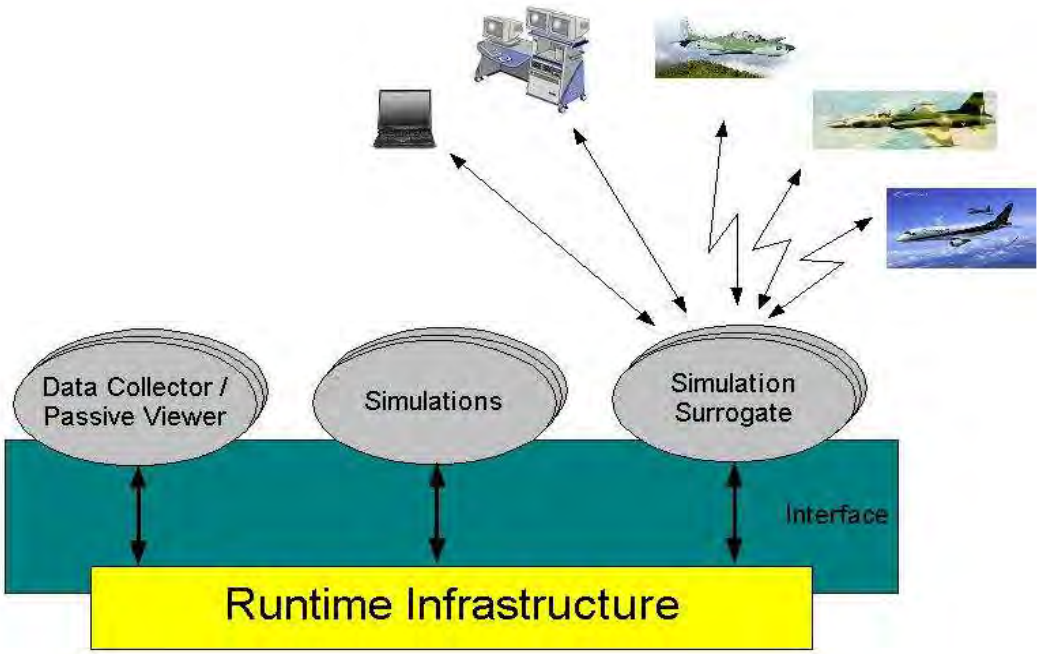

FIGURA 2.11 - Componentes de software em HLA.

2.8.2 Tempo na Arquitetura HLA

Para maior compreensão do gerenciamento de tempo na arquitetura HLA é importante apresentar as definições de sistemas e tempos utilizadas bem como algumas idéias básicas que orientaram o desenvolvimento do seu gerenciador de tempo (“Time Management”).

As definições de sistema físico (“physical system”) e simulação (“simulation”) precedem as definições de tempo utilizadas na arquitetura HLA e são apresentadas por Fujimoto (2000). **Sistema físico** é o sistema atual ou imaginário que está sendo modelado. **Simulação** é o sistema que reproduz o comportamento do sistema físico. Uma representação destes sistemas é a FIGURA 2.12.

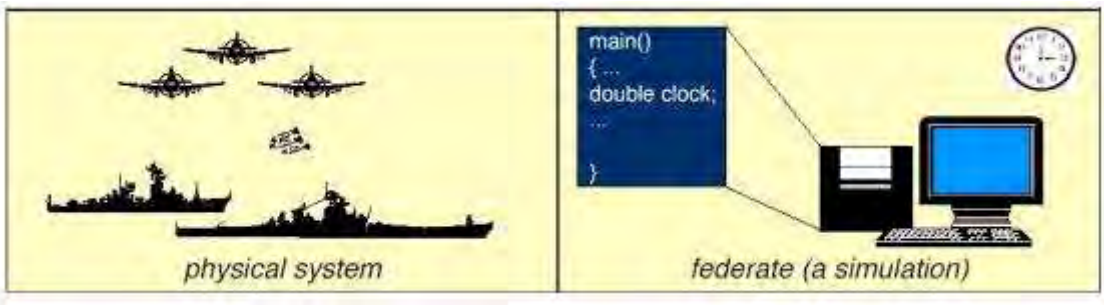

FIGURA 2.12 – Relação entre sistemas físicos e simulação.
FONTE: Fujimoto (2003)

Tempo natural (“physical time”) é o tempo no sistema físico ou natural. Por exemplo, o intervalo de tempo das 16:00 às 18:45 horas do dia 16 de junho de 2002. **Tempo lógico** (“logical time”) ou **de simulação** (“simulation time”) é a representação do tempo natural na simulação. Ele é uma entidade abstrata na RTI, tem um valor inicial, varia positivamente e não tem unidades. Cada federação deve definir uma convenção para o seu significado. **Eixo de tempo da federação** (“FTA - Federation Time Axis”) é a sequência ordenada de valores representando o tempo físico ou natural (ex. valores de ponto flutuante no intervalo [16:00 ; 18:45]. Tempo do federado é um tempo específico do federado no FTA (ex. 17:00). **Tempo de relógio de parede** (“wall clock time”) é o tempo que um observador percebe em um relógio físico durante ou não a execução de uma simulação (ex. 17:00 horas de 07 de abril de 2004). Pode existir ou não uma

relação específica entre o tempo de simulação e o tempo de relógio de parede. Esta relação define três modos de execução: i) **execução a mais rápida possível ("as-fast-as-possible")** – não existe necessariamente nenhuma relação entre o avanço no tempo da simulação e o avanço no tempo de relógio de parede e, portanto, seus passos não são sincronizados; ii) **execução em tempo real ("real-time")** – cada avanço no tempo de simulação é sincronizado com um avanço equivalente no tempo de relógio de parede; e, iii) **execução escalonada em tempo real ("scaled real-time")** - cada avanço no tempo de simulação é sincronizado com um avanço escalonado no tempo de relógio de parede (ex. 2x o tempo de relógio parede). Estas relações estão representadas na FIGURA 2.13.

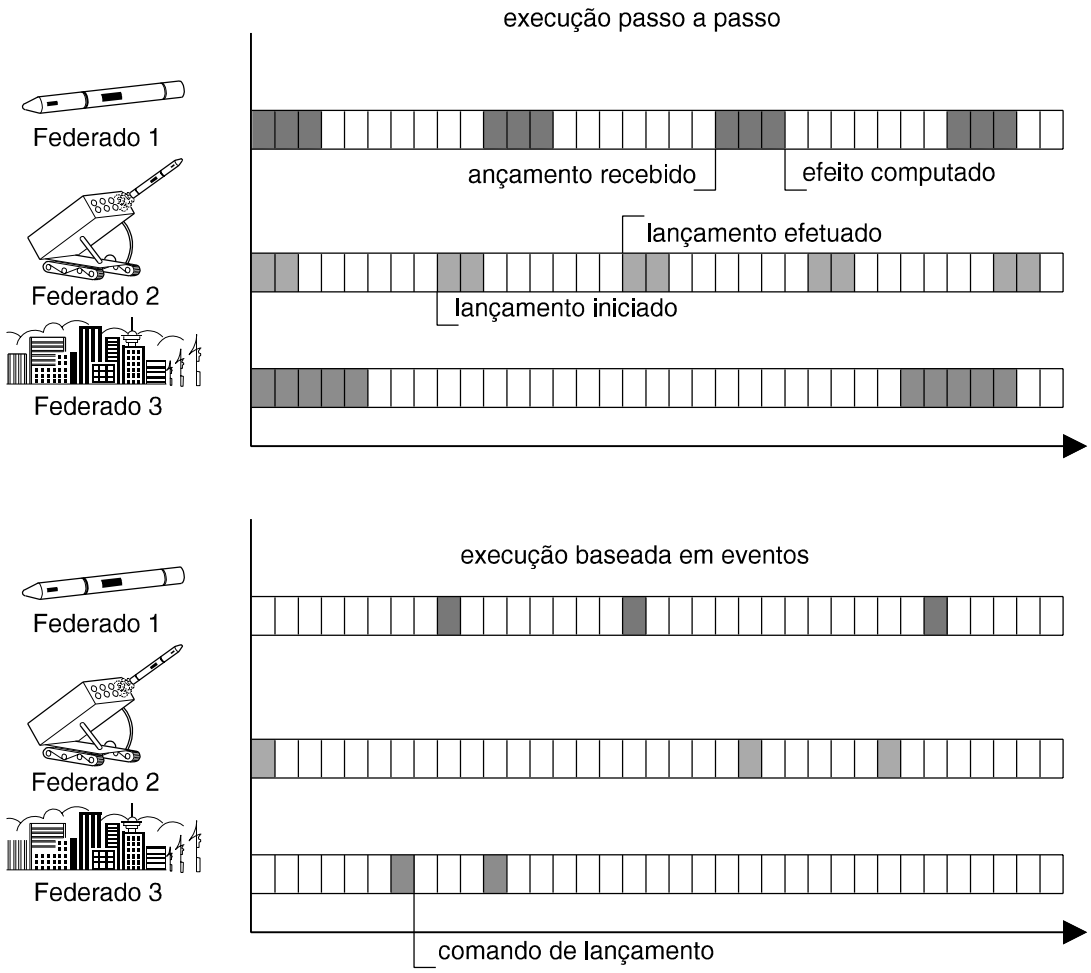

FIGURA 2.13 – Representação no tempo de sistemas passo a passo e baseado em eventos.

Geralmente, o objetivo comum em uma simulação é ter o avanço do tempo da federação aproximadamente na mesma velocidade do tempo de relógio de parede. Em termos gerais, dizemos que executamos uma **simulação em tempo real ("real time simulation")** quando os intervalos entre os eventos na simulação são iguais aos intervalos percebidos por um observador. Ela só será possível se for possível o cálculo do estado para o próximo passo bem antes do tempo de relógio de parede passar. **Simulação passo a passo ("time-stepped simulation")** é aquela que avança o tempo em intervalos de tempo iguais. **Simulação baseada em eventos ("event-driven simulation")** é similar, só que avança o tempo em intervalos de tempo não uniformes. O cálculo do estado para o próximo evento deve ser feito quando o próximo evento estiver na fila e deve ser concluído antes do correspondente tempo de relógio de parede.

2.8.3 Gerenciamento de Tempo

Dentre os diversos serviços da arquitetura HLA, o de interesse direto em nossa pesquisa é o de gerenciamento do tempo e execução das tarefas. Apresentaremos a seguir um breve resumo baseado parte em nossa tradução parcial e, parte em comentários adicionais, do item 8 – "Time Management" da norma **IEEE P1516.1** – "Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules: Federation Interface Specification".

Os serviços do gerenciador de tempo ("Time Management") da arquitetura HLA e seus mecanismos associados permitem a execução da federação garantir que os eventos chegam aos federados na ordem causal correta, e que outra execução da mesma federação produzirá a mesma ordem. O gerenciamento de tempo trabalha com o tempo lógico ou tempo simulado. Estes serviços controlam o tempo lógico da federação e libera os dados marcados com tempo ("time-stamped data").

O objetivo dos serviços de gerenciamento de tempo é permitir à federação garantir que os eventos cheguem aos federados na ordem causal correta, e que outra execução da mesma federação produzirá a mesma ordem. O gerenciamento de tempo trabalha com o

tempo lógico ou tempo simulado. Estes serviços controlam o tempo lógico da federação e libera os dados marcados com tempo (“time-stamped data”).

Segundo Fujimoto (2000), o gerenciamento de tempo foi concebido para suportar e criar mecanismos que controlem o avanço do tempo de federados ao longo do eixo de tempo da federação durante a execução. De maneira geral, o avanço de tempo dos federados pode ser coordenado com os serviços de “Object Management” de tal maneira que essas informações sejam enviadas aos federados (exemplo: estado de atualização e interações) e sejam temporalmente tabuladas e ordenadas. Os serviços de “Time Management” permitem suportar diferentes ordenações e tratamentos de saídas, assim como:

a) Reordenação de requisitos de eventos: Nesse caso, os eventos que chegam a uma simulação são processados pela ordem indicada pelo “Time Stamp”. O processamento de informações na ordem de tempo correto é necessário para uma ampla gama de simulações, como as dirigidas para análises.

b) Mecanismos de avançamento de tempo: Para suportar uma importante classe de simuladores que necessitam que a simulação ocorra no menor intervalo de tempo possível (“unconstrained simulations”).

c) Protocolos de sincronização otimizados: Várias simulações usam técnicas como enviesamento do tempo (“Time Warp”), que permite que o processamento de uma interação possa eventualmente iniciar-se antes do tempo agendado para ocorrer. Nesse caso, se há outros eventos que deveriam ocorrer antes e que sejam dependentes de contexto que precisem ser processados, o evento “adiantado” pode ser desfeito (e seus efeitos) para posterior re-execução no tempo agendado, de maneira a manter a integridade lógica da simulação.

Para permitir esses diferentes mecanismos, múltiplos serviços de emissão de tarefas/eventos são suportados, a saber:

a) Ordem de Tempo Garantida: Eventos são gerados pelo RTI para cada simulação seguindo estritamente a ordem de execução indicada pelo “Time Stamp”.

b) Ordenação de Melhor Esforço: Os eventos são processados numa fila de entrada preferencialmente na ordem que serão despachados. Entretanto, é possível que eventos posteriores possam ser recebidos antes e fora de ordem.

2.8.3.1 Serviços do Gerenciador de Tempo da RTI

Os serviços disponíveis no gerenciador de tempo da arquitetura HLA, num total de vinte e três, são listados a seguir.

Enable Time Regulation – serviço deve ativar a regulagem de tempo de um federado agregado, permitindo desse modo, este federado enviar mensagens TSO.

Time Regulation Enabled – serviço deve indicar que uma solicitação anterior de tempo-regulador foi atendida.

Disable Time Regulation – serviço deve indicar que um federado agregado esta desativando o modo tempo-regulador.

Enable Time Constrained – serviço deve solicitar que um federado agregado chamando este serviço torne-se tempo-restrito.

Time Constrained Enabled – serviço deve indicar que uma solicitação anterior de alteração para o modo tempo-restrito foi atendida.

Disable Time Constrained – serviço deve indicar que um federado agregado não é mais do tipo tempo-restrito.

Time Advance Request (TAR) – serviço deve solicitar o avanço do tempo lógico de um federado agregado e enviar zero ou mais mensagens de liberação para o federado agregado.

Time Advance Request Available (TARA) – serviço similar ao TAR para o tempo lógico T, exceto que: i) a RTI não pode garantir a emissão de todas as mensagens com marcas de tempo T e, ii) após aceito, pode enviar mensagens adicionais com marcas de tempo T se o avanço atual do federado agregado é zero.

Next Message Request (NMR) – serviço deve solicitar o avanço de tempo do federado agregado até a marca de tempo da próxima mensagem TSO que será liberada para o federado, limitando a marca de tempo desta mensagem ao tempo lógico especificado na solicitação.

Next Message Available (NMRA) – serviço similar ao NMR para o tempo lógico T, exceto que: i) a RTI não pode garantir a emissão de todas as mensagens com marcas de tempo iguais a T e, ii) após aceito, pode enviar mensagens adicionais com marcas de tempo iguais a T se o avanço atual do federado agregado é zero.

Flush Queue Request (FQR) – serviço deve solicitar que todas mensagens TSO ordenadas na RTI que um federado agregado irá receber sejam liberadas naquele momento.

Time Advance Grant – serviço deve indicar que a solicitação anterior de um federado agregado para avanço de seu tempo lógico foi atendida.

Enable Asynchronous Delivery – serviço deve instruir a RTI liberar mensagens recebidas RO para um federado agregado que chamou este serviço quando ele estiver no estado “Time Advancing” ou “Time Granted”.

Disable Asynchronous Delivery – serviço deve, para um federado tempo-restrito, instruir a RTI liberar mensagens RO para um federado agregado que chamou este serviço somente quando ele estiver no estado “Time Advancing”.

Query GALT – serviço deve ordenar um GALT solicitado por um federado agregado.

Query Logical Time – serviço deve ordenar o corrente tempo lógico solicitado por um federado agregado.

Query LITS – serviço deve ordenar um LITS solicitado por um federado agregado.

Modify Lookahead – serviço deve ordenar a modificação de um lookahead solicitada por um federado agregado.

Query Lookahead – serviço deve interrogar a RTI sobre o atual lookahead do federado agregado.

Retract – serviço deve ser usado por um federado agregado para notificar a execução da federação que a mensagem anteriormente enviada ao federado agregado deve ser recolhida.

Request Retraction – se a RTI recebe uma chamada legal de *Retract* para uma mensagem que já foi enviada para um federado agregado, o serviço *Request Retraction* deve ser chamado para aquele federado.

Change Attribute Order Type – serviço deve mudar o tipo preferido de ordenamento de um federado agregado durante a execução para todos os serviços futuros *Update Attribute* Values para uma instância específica desses atributos.

Change Interaction Order Type – serviço deve mudar o tipo preferido de ordenamento de um federado agregado durante a execução para todos os serviços futuros *Send Interaction e Send Interaction with Regions* para uma classe específica de interação somente para os federados agregados que solicitaram.

2.8.4 Gerenciamento de Distribuição de Dados

Dentre os diversos serviços da arquitetura HLA, outro de interesse direto em nossa pesquisa é o de gerenciamento de distribuição de dados. Apresentaremos a seguir um breve resumo baseado parte em nossa tradução parcial e, parte em comentários adicionais, do item 9 – "Data Distribution Management" da norma **IEEE P1516.1** – "Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules: Federation Interface Specification".

Os serviços do gerenciador de distribuição de dados ("Data Distribution Management - DDM") da arquitetura HLA e seus mecanismos associados gerenciam a troca de informações entre federados e/ou federações. São serviços opcionais e, portanto, a federação não necessita usá-los. São serviços adicionais ao "declaration management"

que permitem alterações na troca de dados durante a execução da federação. Estes serviços devem ser utilizados pelos federados agregados para a redução da transmissão e recepção de dados irrelevantes entre eles.

As declarações de interesse no DDM são expressas de duas formas de dados**: classe de objeto atributos e classes de interação**. O usuário define um espaço dimensional de interesse, fornecendo os seus limites, que é denominado **região**. A sobreposição das regiões de consumidores e produtores de informações define um espaço limitado para as comunicações relevantes.

2.8.4.1 Definições Para o DDM

As **dimensões** são intervalos definidos por inteiros não negativos. Uma **distância ("range")** é um intervalo contínuo semi-aberto na dimensão definido por um par ordenado contendo o limite inferior e o limite superior. A especificação da região é um conjunto de distâncias. Cada distância na especificação da região deve ser definida em temos de limites inferiores e superiores contidas em [0, dimensão correspondente ao limitante superior da dimensão]. A realização de uma região ocorre quando uma especificação de região é associada a uma instância de atributo de atualização, com o envio de uma interação, ou com uma classe de atributo ou com uma classe de interação para subscrição. A RTI deve prover uma região padrão ("default") que é definida com as distâncias de cada dimensão declarada na "FOM Document Data" (FDD).

2.8.4.2 Novas Relações de Regiões e Especificações de Regiões

A arquitetura contém a descrição das alterações nas relações entre os componentes afetados pelo DDM. Estabelece quatro relações entre especificações de regiões e realização de regiões conforme estabelecido no FDD. Estabelece as relações, através dos serviços do DDM, entre as regiões; entre as classes de objetos, classes de atributos, instâncias de objeto e instância de atributos; e, entre as classes de interações, parâmetros e interações.

2.8.4.3 Calculando Sobreposição de Regiões

Uma região é definida em um espaço n-dimensional; entretanto representamos uma região com duas dimensões na FIGURA 2.14. Devem ser definidas regiões receptoras (“subscribing”) e geradoras (“sending”) para que o DDM, através das interações entre as mesmas, defina quais as comunicações são relevantes ou irrelevantes.

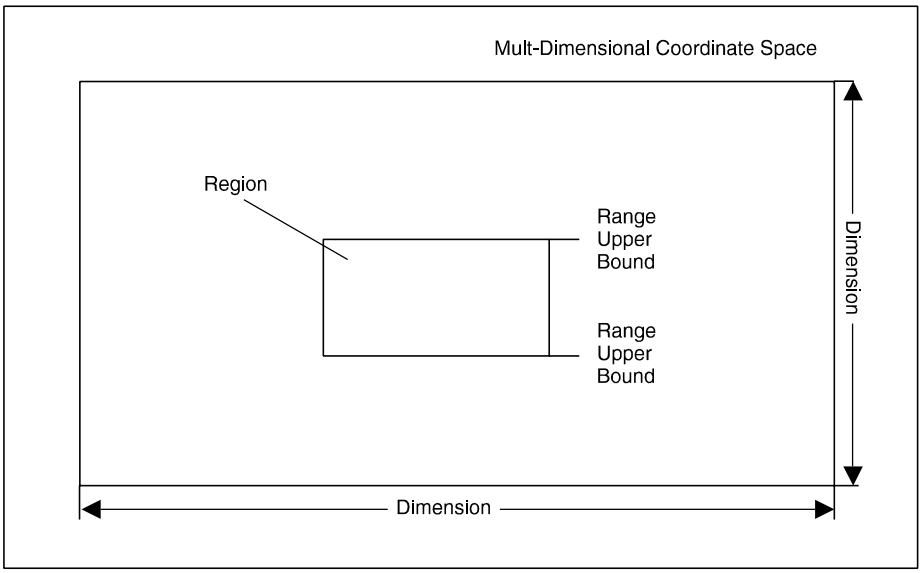

FIGURA 2.14 – Região com duas dimensões.
FONTE: IEEE (2001)

Duas regiões se sobrepõem se e somente se todas as distâncias em todas as dimensões contidas nas duas regiões se sobrepõem par a par. Se duas regiões não têm nenhuma dimensão em comum, elas não devem se sobrepor. A região padrão deve se sobrepor com todas as realizações de regiões não vazias.

2.8.4.4 Reinterpretação de Serviços Pelo DDM

Alguns serviços do DDM podem ser utilizados para realizar funções atendidas pelos serviços DM. Um federado agregado deve interpretar um conjunto de quatro serviços considerando as regiões: *Subscribe Object Class Attributes With Regions, Unsubscribe*

Object Class Attributes With Regions, Subscribe Interaction Class With Regions e Unsubscribe Interaction Class With Regions.

De forma similar o DDM pode ser utilizado para desempenhar funções atendidas com os serviços de gerenciador de objetos. Quando um federado agregado usa o DDM, três serviços do gerenciador de objetos devem ser estendidos para incorporar a expansão da interpretação de como o gerenciador de objetos trabalha em conjunto com os serviços do DDM: *Register Object Instance With Regions, Send Interaction With Regions e Request Attribute Value Update With Regions.*

2.8.4.5 Serviços do Gerenciador de Distribuição de Dados

Os serviços disponíveis no gerenciador de distribuição de dados, num total de doze, da arquitetura HLA são listados a seguir.

Create Region – o serviço deve criar uma região com as dimensões especificadas. A região pode ser usada tanto para atualização ou subscrição.

Commit Region Modifications – o serviço deve informar a RTI sobre alterações nos parâmetros das regiões.

Delete Region – o serviço deve apagar uma região específica. A região em uso para atualização ou subscrição não pode ser apagada.

Register Object Instance With Regions – o serviço deve criar uma única instância de um objeto designador e prover a ligação com a instância do objeto fornecida pela classe do objeto.

Associate Regions For Updates – o serviço deve associar regiões a serem usadas em atualizações com instâncias de atributos de uma instância de objeto específica.

Unassociate Regions For Updates – o serviço deve remover as associações entre as regiões e instâncias de atributos específicas.

Subscribe Object Class Attributes With Regions – o serviço deve especificar uma classe de objetos para a qual a RTI deve iniciar a notificação do federado agregado da descoberta de instanciação de objetos quando pelo menos uma instância de atributos de uma instância de objeto está no escopo.

Unsubscribe Object Class Attributes With Regions – o serviço deve informar a RTI que deve parar a notificação do federado agregado da descoberta de instanciação de objetos e atualização de atributos de uma classe de objetos específica na região especificada.

Subscribe Interaction Class With Regions – o serviço deve especificar a classe de interações que deve ser liberada para o federado agregado, considerando a região.

Unsubscribe Interaction Class With Regions – o serviço deve informar a RTI que ela não deve mais notificar um federado agregado das interações de uma classe especificada que é enviada dentro de uma especificada região.

Send Interaction With Regions – o serviço deve enviar uma interação para uma federação. Os parâmetros da interação somente podem ser aqueles especificados na classe ou todas superclasses, como definido na Federation Object Model Data Declaration (FDD).

Request Attribute Value Update With Regions – o serviço deve usado para estimular a atualização de valores de atributos específicos.

CAPÍTULO 3

REVISÃO BIBLIOGRÁFICA

3.1 Modelagem e Simulação

As sub-áreas do conhecimento representadas pela modelagem e pela simulação estão embutidas de diferentes formas e diferentes níveis nas outras áreas do conhecimento humano. Cada um dessas áreas do conhecimento humano apresenta a sua visão, metodologias, técnicas e ferramentas utilizadas no seu próprio escopo. Modelagem mental é o primeiro passo para a compreensão e solução de um problema e precede a simulação. A simulação está umbilicalmente ligada à primeira, pois fundamentalmente, é a evolução de um determinado modelo no tempo.

A teoria de modelagem para a área de controle de sistemas de controle analógicos é bastante desenvolvida e pode ser verificada em maiores detalhes em Ogata (1970) e Shearer et al (1972). Um estudo mais elaborado sobre modelos discretos (Digitais) de sistemas dinâmicos pode ser encontrado em Ogata (1970) e Franklin & Powell (1980).

A NASA utiliza extensivamente a simulação e em especial a simulação de tempo real para estudos preliminares e antecipação de problemas nos seus programas espaciais. Podemos citar como exemplos de estudos efetuados muito antes de se tornarem projetos reais: i) NASA (1971), onde é feita uma investigação sobre os problemas de dinâmica e controle de uma estação espacial; ii) NASA (1977), onde é apresentado um simulador dinâmico de vôo para reentrada de satélites; e iii) NASA (1986), onde o feito o uso de simulação para auxílio de projeto e análise de operações em condições anormais de um grande túnel antes de sua construção.

Mais tarde, o crescimento exponencial da capacidade de processamento dos computadores digitais e sua penetração em todas áreas da vida humana permitiram uma verdadeira explosão no uso da simulação. Sua importância pode ser verificada através da definição, tanto pela NASA como para o DoD, da simulação como área chave

estratégica para o futuro. A história da simulação deve ser verificada na página da NTSA e nos anais do I/ITSEC.

Entre as visões de necessidades futuras da simulação apresentadas nas conferências desde 1997 da “Society for Computer Simulation" (SCS) e da “National Training Systems Association" (NTSA) durante a “Industry/Interservice, Training, Simulation and Education Conference” (I/ITSEC), na Flórida, EUA, destacam-se a simulação com milhões e até bilhões de entidades simultâneas e a solução dos problemas correlacionados e, entre eles, o de gerenciamento da execução das tarefas. Também foi mencionada a necessidade de novas formas de classificação de modelos para simulação.

Recentemente a comunidade tem trabalhado em uma definição matemática ou formal para a área de modelagem e simulação como pode ser verificada em Zeigler et al (2000). É uma tentativa de se estabelecer formalmente a área incluindo tanto a modelagem de sistemas dinâmicos (“contínua”) e a modelagem de sistemas baseados em eventos (“discreta”).

Um breve histórico e respectiva bibliografia nesta área esta mais detalhada nos relatórios internos do INPE, elaborados por Trivelato (2003) durante este trabalho, para uma melhor compreensão e organização das idéias na área: Técnicas de modelagem e simulação de sistemas dinâmicos, Simulação distribuída: o que é e qual o potencial da HLA e de “web simulation”, Comparação de ambientes de modelagem e simulação, Protocolos de redes para ambientes de simulação distribuída.

3.2 Sistemas de Tempo Real

A História de Sistemas de Controle de Tempo Real está relacionada com a História da Computação de Tempo Real. Isto se deve ao fato de os requisitos de tempo real serem requisitos do sistema em seu ambiente como um todo e não apenas os requisitos internos do computador.

Segundo definições de Martin (1965), Shin & Ramanathan (1974), Stankovic (1988) e Kavi (1992), na computação de tempo-real os dados entram no computador e são

processados numa arquitetura de hardware e software que garantem elevada confiabilidade de seu desempenho garantindo segurança aos valores vivos e manufaturados envolvidos; para depois serem direcionados para as respectivas saídas. Uma visão histórica sobre controle de tempo real pode ser vista no livro de Bennett e Linkens (1984). Um breve histórico da computação e, em particular da computação de tempo real, pode ser verificado em Martin (1980). A discussão dos erros conceituais na área de sistemas de tempo real é atualizada no Capítulo 1 de Kavi (1992). Princípios específicos de arquitetura de computadores para sistemas de tempo real são apresentados no Capítulo 4 de Halang e Stoyenko (1991). Um exemplo de um problema clássico na área de computação adicionado com requisitos de tempo real é apresentado no Capítulo 1 de Stankovic e Ramamritham (1988). Uma boa definição para sistemas de tempo real pode ser encontrada na norma alemã DIN 44 300. No Brasil, uma boa revisão bibliográfica com conceitos de sistemas de tempo real e apresentação dos diversos tipos de agendadores foi compilada por Farines et al (2000) no evento bi-anual Escola da Computação realizado pelas universidades brasileiras.

A referência fundamental para a teoria de agendadores é o trabalho desenvolvido por Liu & Layland (1973). Um excelente trabalho para a área de sistemas de tempo real é o relatório mensal especial publicado pelo IEEE (1994) contendo uma extensa bibliografia cobrindo toda a teoria de agendadores e escalonados para sistemas de tempo real críticos. Conforme Stankovic (1988) a teoria de agendamento/escalonamento de tarefas em sistemas de tempo real é uma importante área de pesquisa demandando novos algoritmos que sejam bem compreensíveis de forma que o comportamento do sistema no tempo seja compreensível, previsível e de fácil manutenção. A classificação dos programas agendadores segundo diferentes paradigmas é bastante vasta na literatura e, em particular, um bom resumo é apresentado por Hamamritham e Stankovic (1994). Dentre as diversas classes apresentadas anteriormente, destacamos os agendadores mais utilizados na literatura, apresentados também por Burns e Welling (1990), Microsoft Corporation (1995) e Integrated Systems (1995), e utilizados neste trabalho.

Os relatórios internos da NASA descrevem, na visão da instituição, uma verdadeira história dos sistemas de tempo real e seus sistemas operacionais. Características de

tempo real para computadores são feitas no relatório NASA (1984). Um exemplo de aplicação em tempo real utilizando este sistema operacional é apresentado no relatório NASA (1985). Um estudo mais detalhado sobre sistemas operacionais de tempo real, utilizando uma extensão baseada no UNIX, é apresentada no livro Furht et al (1991). O sistema operacional LINUX para microcomputadores devido a sua similaridade e compatibilidade com o UNIX e ao fato de ter seu código aberto tem se tornado uma opção para sistemas de simulação. Muitas linguagens de programação chamadas de tempo real não são específicas para atender os requisitos de programação de sistemas de tempo real. No Capítulo 4 de Stankovic e Ramamritham (1988) são apresentados papers sobre essa área. Uma boa definição para a medida da capacidade de sistemas operacionais de tempo real é apresentada por Fuhrt et al (1988).

No segmento de aeronáutica encontramos definições mais formais como a adotada no documento DoD (1978). De acordo com os regulamentos aplicáveis FAA AC 25.1309-1A e JAA AMJ 25.1309, o documento emitido conjuntamente pelo RTCA e EUROCAE, denominado DO-178B/ED-12B, é o guia para desenvolvimento de software embutido em aeronaves aceito pelo FAA e JAA para cumprimento dos requisitos de software. A circular AC 20-115B cancela as versões preliminares do DO-178. Esses requisitos são necessários para a homologação de sistemas de tempo real críticos.

Em arquitetura de sistemas de tempo real distribuídos é necessário incluir também a comunicação entre processos em tempo real. Trabalhos sobre comunicação em tempo real são apresentados no Capítulo 7 de Stankovic e Ramamritham (1988). Um diagrama mais geral apresentando os serviços disponíveis em ambos protocolos é apresentado por Comer (2000).

3.2.1 Arquitetura de Programas de Tempo Real

Os sistemas de tempo real necessitam de programas que tenham características específicas para o cumprimento dos seus requisitos. Apresentamos uma arquitetura simples e robusta desenvolvida pela Integrated Systems Inc. para a família de produtos

do MATRIXx e para o sistema operacional pSOSystem (FIGURA 3.1), que apresentam essas características. Esta é a arquitetura sugerida e utilizada neste trabalho.

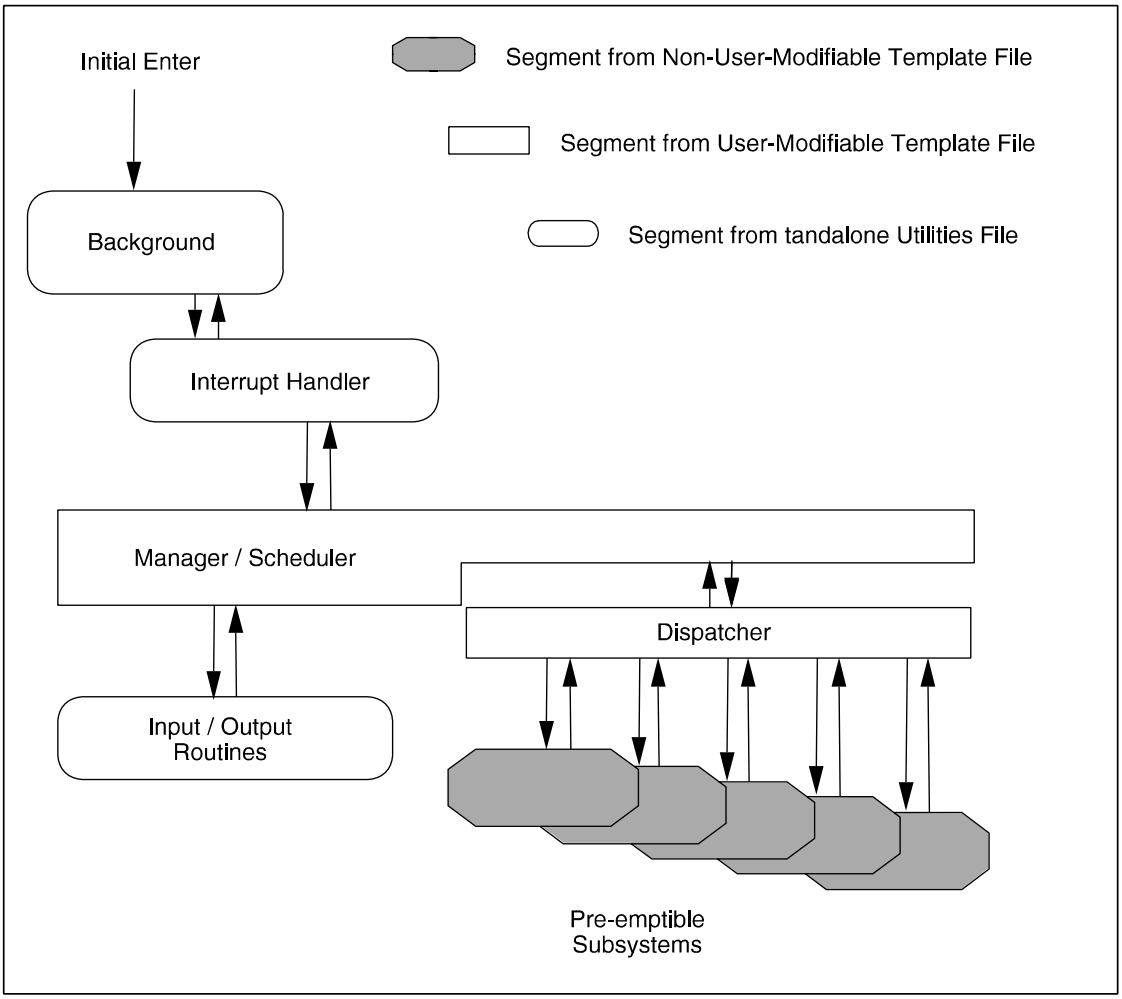

FIGURA 3.1 – Componentes do um programa de tempo real.
FONTE: National Instruments Corporation (2003)

Descrevemos sumariamente as funções básicas de cada um dos seus componentes principais:

- Gerenciador/Agendador (“Manager/Scheduler”) – o gerenciador/agendador é uma rotina de tempo-crítico que controla e executa as funções de entrada e saída da

aplicação, controla diversas funções de limpeza e organização, e gera a lista de tarefas que estão prontas para serem executadas.

- Despachador (“Dispatcher”) – o despachador despacha as tarefas que estão prontas na lista de despacho conforme a política estabelecida pelo agendador.

- Subsistemas (“Subsystems”) – os subsistemas contém o código que implementa as tarefas de tempo real recebendo as entradas e disponibilizando as saídas conforme os tempos de amostragem definidos no sistema.

- Rotinas de Entrada e Saída (“I/O routines”) – as rotinas de entrada e saída fornecem as entradas e escrevem as saídas dentro dos requisitos de tempo estabelecidos pelo sistema.

- Manipulador de Interrupções do Temporizador (“Timer interrupt handler”) – o controlador de interrupções do temporizador chama ou invoca o gerenciador/agendador em intervalos de tempo específicos e gerencia as interrupções externas ao sistema.

- Funções de espera (“Background functions”) – a rotina de espera e serviços gerais, nos intervalos livres do processador, executa as funções não críticas no tempo como atualizar indicadores, autotestes, etc.

3.2.1.1 Controle de Fluxo no Programa de Tempo Real Gerado Pelo AutoCode

Um bom exemplo de aplicação de tempo real é apresentada na FIGURA 3.2, representando o controle de fluxo no programa gerado pelo AutoCode, acompanhado de uma breve descrição de seu funcionamento, nossa tradução parcial do MATRIXx AutoCode Users Guide.

Na partida, o programa inicial (parte das rotinas de utilitários) estabelece a partida do manipulador de interrupção de tempo, linhas de tempo, filas de prioridade, e as condições iniciais para os subsistemas sustados, e o Gerenciador/Agendador (“Manager/Scheduler”) entra no estado de pronto. Como ilustrado na FIGURA 3.2, após

a partida o programa para a executar a rotina de espera (“background”), que fica aguardando a primeira interrupção ou outra chamada do programa inicial.

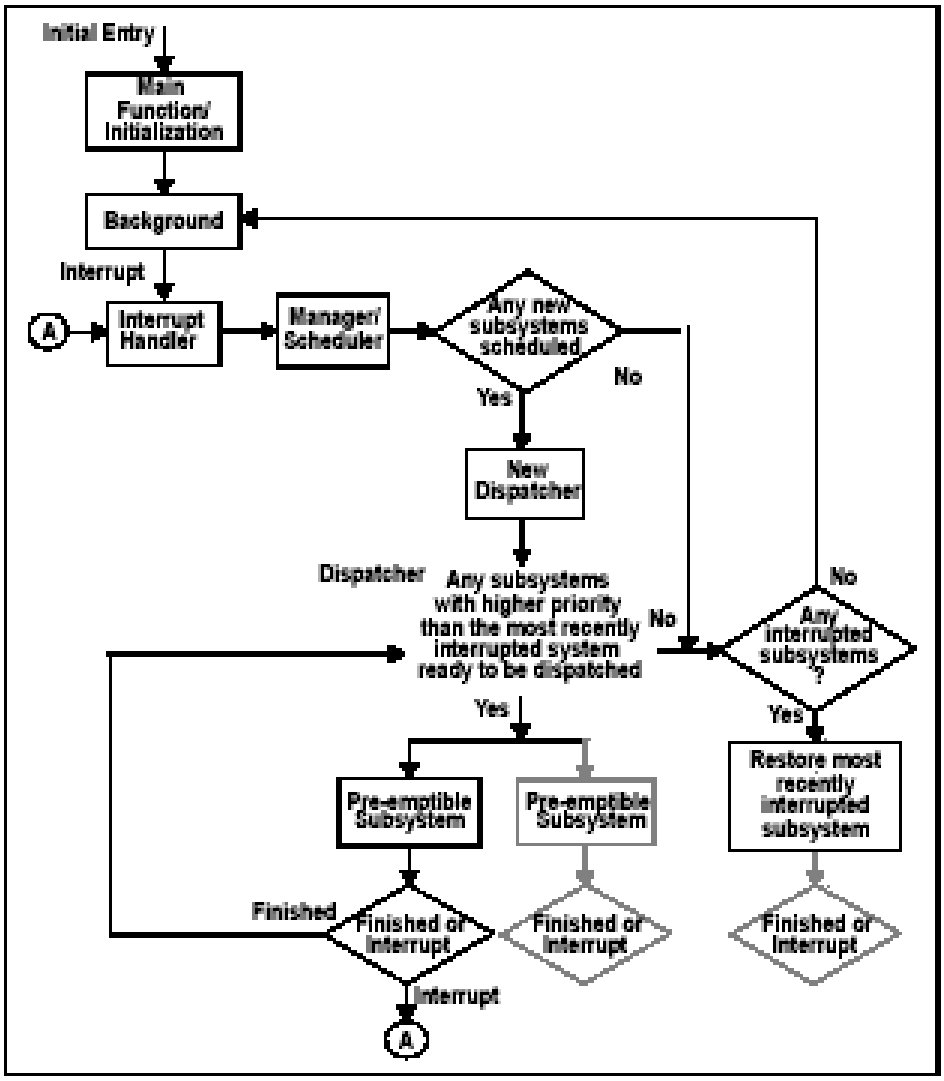

FIGURA 3.2 – Controle de fluxo de um programa gerado pelo AutoCode.
FONTE: National Instruments Corporation (2003)

Quando uma interrupção é recebida, o Manipulador de Interrupções de Tempo (“Timer Interrupt Handler”) salva o contexto interrompido, se necessário, e passa o controle para o Gerenciador/Agendador. O agendador verifica as entradas externas e estabelece a lista de subsistemas a serem despachados. Depois disponibiliza quaisquer saídas externas e desempenha funções de organização interna antes de fornecer a lista de tarefas a serem despachadas ao despachador (“Dispatcher”).

O despachador é basicamente uma grande chave comutadora que passa o controle para o subsistema que tem a maior prioridade na lista de tarefas. Ele sempre verifica se existe algum subsistema despachado previamente, mas interrompido, com maior prioridade antes de despachar um novo subsistema de sua lista de agendados. Se existe, o subsistema mais recentemente interrompido (que deve ser o de maior prioridade sobre todos os outros interrompidos anteriormente e sobre os novos subsistemas agendados) é restaurado pelo gerenciador de interrupção e tem seu processamento continuado.

Quando o processamento do subsistema é concluído, ele passa o controle de volta para o despachador, que despacha ou restaura o próximo subsistema com a próxima maior prioridade, e assim por diante. Se todos os subsistemas atualmente despachados e subsistemas interrompidos anteriormente terminarem antes da próxima interrupção de tempo, o gerenciador de interrupção retorna o controle para a rotina de espera. Entretanto, se um subsistema ainda estiver sendo processado quando ocorrer a próxima interrupção de tempo, o controle retorna ao gerenciador de interrupção, que novamente passa o controle para o gerenciador/agendador, que é executado novamente.

No caso de subsistemas representados por tarefas codificadas em ADA, o despachador despacha simultaneamente todas as tarefas que estão prontas. As tarefas codificadas em ADA têm prioridades associadas com elas que determinam a disponibilidade de CPU para cada tarefa.

Se o gerenciador/agendador não tem nada para agendar na próxima interrupção de tempo, o agendador passa o controle de volta ao gerenciador de interrupção. O gerenciador de interrupção então restaura quaisquer tarefas que estavam rodando no momento da interrupção. Se qualquer subsistema ou o despachador foi interrompido por meio de sua execução, o gerenciador de interrupção passa o controle de volta a qualquer que estivesse rodando (subsistema ou despachador) no tempo da interrupção. Se não restam despachador e subsistemas, então o controle retorna para a rotina de espera.

3.2.1.2 Sequência de Operações do Agendador

A FIGURA 3.3 ilustra a sequência de operações de uma aplicação agendador de tempo real. O agendador é representado pelo diagrama de bolhas (embora não seja estritamente uma máquina de estados finita), porque durante a fase 9 de despachar subsistemas (“dispatch subsystems”), as operações podem ser interrompidas. Entretanto, durante a seção crítica (bolhas 1-8), ele opera na forma de uma máquina de estado. Na discussão que se segue, o termo agendador refere-se à seção crítica e o termo despachador refere-se à seção inter-rompível.

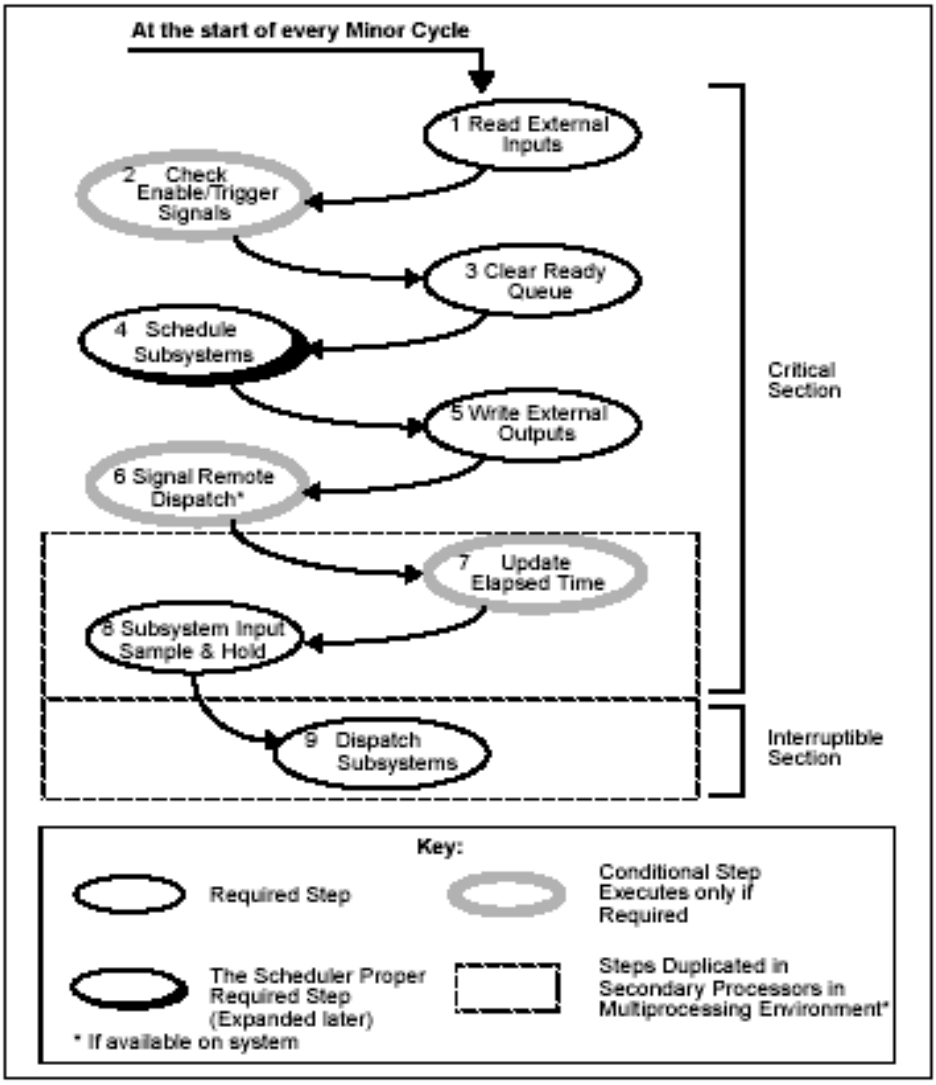

FIGURA 3.3 – Operação de um agendador.

FONTE: National Instruments Corporation (2003)

Porque os primeiros oito passos da operação do agendador é não inter-rompível, passos cítricos, é importante otimizar o código para o máximo desempenho e executá-los com a mesma duração em cada ciclo. Isto minimiza os ruídos de variação de tempo (“jitter”) e evitam problemas de desempenho.

Bolha 1 (Read External Inputs): Na entrada ou re-entrada, o agendador adquire as entradas externas do sistema e elas podem se usadas pelos subsistemas agendados sem um atraso de um ciclo menor (“minor frame”). Por definição, o menor ciclo de tempo de uma aplicação é o menor ciclo do agendador, um intervalo de tempo criado a partir das freqüências de amostragem e dos requisitos de tempo de todos os superblocos no sistema. O agendador é executado exatamente uma vez a cada menor ciclo.

Bolha 2 (Check Triggers and Enables): O agendador prepara para o agendamento dos subsistemas primeiro determinando quais os sistemas disparados (“triggered”) ou ativados (“enabled”) são elegíveis para serem executados durante o atual ciclo. Este passo não é aplicado para sistemas que não possuam subsistemas disparados ou ativados. Para subsistemas ativados, o agendador verifica o estado do subsistema. Se o estado é bloqueado (isto é, pronto para executar, mas aguardando um sinal de ativar), o subsistema está pronto para ser executado. Se o sinal de ativado é verdadeiro, o agendador o inclui na fila de execução. Entretanto, se o subsistema está no estado aguardando (“idle”) e o sinal de ativado é verdadeiro, o agendador deve determinar quando é o tempo correto para este subsistema ser executado. Se ainda não for o tempo, o subsistema espera no estado de aguardando até que seja o tempo dele ser executado. Para subsistemas disparados, se o sinal de disparo é verdadeiro, o agendador verifica o estado do subsistema e procede adequadamente de acordo com o tipo de subsistema. Neste estágio, quando um sinal de disparo é recebido e o subsistema disparado está no estado bloqueado esperando o disparo, o agendador o inclui na fila para execução. Se o subsistema está no modo aguardando, o subsistema é também incluído na fila para execução, mas as saídas são ou não disponibilizadas, dependendo do tipo de subsistema.

Bolha 3 (Clear the Ready Queue): O agendador limpa a fila de pronto para todos os subsistemas. A fila de pronto é estabelecida pelo agendador na bolha 4 e é usada na

bolha 8 para determinar quais os subsistemas que necessitam ter suas entradas atualizadas nos amostradores-seguradores.

Bolha 4 (Schedule the Subsystems): O algoritmo agendador é executado. Para agendar cada subsistema, o agendador verifica o tempo de "overflow" (o tempo que falta para a tarefa ser executada antes da sua próxima solicitação) de cada subsistema. O agendador também verifica para todos os subsistemas, todos os critérios para determinar quando eles devem ser despachados. Estes critérios são:

a) Contínuo ("Continuous") – A tarefa contínua é despachada (através do integrador) todo ciclo que o despachador é chamado. Um elemento do agendador é o integrador (FIGURA 3.4). Ele executa integração contínua de passo fixo dos estados e implicitamente despachar os subsistemas contínuos para atualizarem os seus estados e atualizarem suas saídas. O par integrador/tarefa contínua é tratado por definição como a tarefa mais rápida a ser despachada pelo agendador. Sua frequência pode também ser escolhida desde que seja pelo menos maior ou igual à frequência da tarefa periódica mais rápida no modelo.

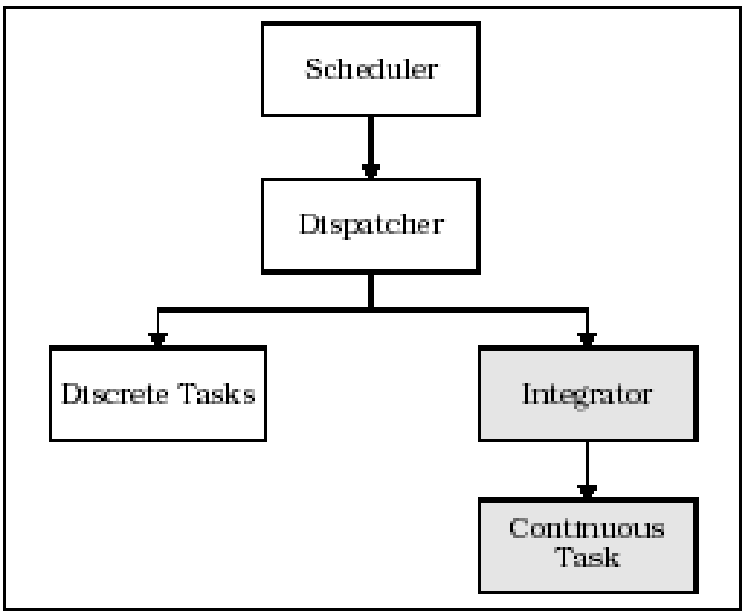

FIGURA 3.4 – Arquitetura do agendador do MATRIXx.
FONTE: National Instruments Corporation (2003)

b) Tarefa periódica livre (“Free-Running Periodic”) – O tempo adequado para execução chegou, como é determinado pela tabela da linha de tempo e pelo tempo passado segundo o contador.

c) Ativados (“enabled”) – O tempo adequado para execução chegou, o sinal de ativado ainda é verdadeiro, e o subsistema estava no estado aguardando no ciclo anterior (estado em que o sinal de ativado é verdadeiro, mas o seu tempo ainda não havia chegado); ou, o sinal de ativado acabou de tornar-se verdadeiro e o subsistema estava no estado bloqueado no ciclo anterior (estado no qual o sinal de ativado é falso).

d) Disparados (“triggered”) – O sinal de disparo sofreu transição de falso para verdadeiro após o início do último menor ciclo. Esta condição também é verificada na bolha 2. Se o tipo do bloco disparado é assíncrono, e se o subsistema está para ser despachado pelo agendador (ou seja, se o sinal de disparo não é uma entrada externa), o subsistema dispara se o sinal de disparo sofreu transição tanto de falsa para verdadeira ou de verdadeira para falsa desde o início do menor ciclo anterior. Se o sinal de disparo para o subsistema assíncrono é uma entrada externa, o subsistema é despachado separadamente do agendador, respeitando suas propriedade específicas.

Baseados nestes critérios, o agendador constrói a fila de pronto e adiciona os subsistemas que devem ser executados na lista de despacho. A diferença entre a lista de despacho e a fila de pronto são aqueles subsistemas que estavam na lista de despacho, mas não foram despachados no ciclo anterior, que são trazidos novamente para a lista de despacho para serem despachados neste ciclo ou mais tarde. Por contraste, a fila de pronto é limpa e reconstruída em todos os ciclos.

Como indicado na bolha 4, o agendador utiliza atributos computacionais dos subsistemas para estabelecer as prioridades para os subsistemas serem despachados. No caso do taxa monotônica, a sequência de prioridade é estabelecida usando atributos computacionais, partindo daquele com maior frequência de amostragem ou requisito de tempo até àquele de menor. Para juntar as diferentes freqüências de amostragem ou

requisitos de tempo, a prioridade de execução é baseada no tipo de subsistema, do maior para o de menor prioridade: contínuo, periódico livre, periódico ativado, assíncrono disparado, disparado no modo mais rápido possível, disparado no modo depois de um requisito de tempo e disparado no modo próximo disparo.

Bolhas 5 & 6 (Write External Outputs and Signal Remote Dispatch): O agendador chama a rotina externa de saída para atualizar todas as saídas de todos subsistemas a cada menor ciclo. Em implementações com multiprocessadores somente, a lista de despacho e o sinal remoto de despacho são alocados ao processador secundário (bolha 6) para indicar a disponibilidade da lista de despacho e definir o início da execução do subsistema.

Bolhas 7 & 8 (Update Elapsed Time and Sample & Hold): Em implementações com multiprocessadores, cada subsistema agendador do processador secundário pode ou não necessitar realizar a atualização do tempo passado ("elapsed time update") - bolha 7, mas será solicitado a "amostrar e segurar" a entrada do subsistema (onde o agendador lê as entradas e as segura para uso do subsistema, bolha 8) e a despachar o subsistema (bolha 9). O agendador atualiza o contador de tempo (bolha 7), se necessário. Bolha 8: o agendador consulta a fila de pronto em cada processador para realizar a função amostrar e segurar as entradas dos subsistemas para os subsistemas que foram incluídos na lista de despacho. Por razões de determinismo, subsistemas já existentes na lista de despacho do ciclo anterior não terão suas entradas amostradas novamente. Este é o último passo da seção crítica.

Bolha 9 (Dispatch Subsystems): O despachador é re-entrante e ele pode ser interrompido em qualquer ponto de suas operações. Este passo de re-entrância aceita a lista de despacho e filas de sub-rotinas para execução. Este passo também inclui a execução de sub-rotinas, que possam ser interrompidas/sustadas a qualquer tempo.

3.3 A Arquitetura de Alto Nível (“The High Level Architecture-HLA”)

Para ambientes complexos, envolvendo eventos de natureza discreta e contínua, várias arquiteturas foram elaboradas pela comunidade ligada à simulação, culminando com a Arquitetura de Alto Nível – (“The High Level Arquitecture” - HLA), proposta pelo DoD em 1995-1996. Um bom histórico é apresentado por Kuhl et al (2000) e Fujimoto (2000).

O maior avanço da tecnologia teve a sua origem no desenvolvimento do “Modeling & Simulation Master Plan” (inicialmente chamado M&S HLA) iniciado na “Defense Advanced Research Projects Agency” (DARPA) pelo “Advanced Simulation Program” (ADS) que definiu os objetivos básicos desejáveis para a área de simulação no DoD.

Após anos de trabalho coordenados pelo “Defense Modeling & Simulation Office - DMSO”, do DoD, e um trabalho de interação de mais de três anos desenvolvido com o (“Institute of Electrical and Electronics Engineers- IEEE”), a arquitetura HLA foi definida como norma internacional do IEEE (2000) , contendo:

IEEE P1516 – “Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules”;

IEEE P1516.1 – “Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules: Federation Interface Specification,” onde estão incluídos: “Federation Management, Declaration Management, Object Management, Ownership Management, Data Distribution Management, Support Services Management, Object Model (MOM) Specification, Federation Execution Data (FED)”;

IEEE P1516.2 – “Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – HLA Object Model Template (OMT)”, onde estão incluídos: “Object Model Template (OMT)”, “Federation Object Model (FOM)” e “Simulation Object Models (SOM)”.

CAPÍTULO 4

NOVOS ESCLARECIMENTOS PARA A CONSOLIDAÇÃO DE RESULTADOS EXISTENTES NA LITERATURA

4.1 Definições Utilizadas na Literatura

As definições básicas são as mesmas apresentadas em Layland (1973). Um **agendador estático** ("static scheduler") define a prioridade de cada tarefa em tempo de projeto e ela permanece a mesma durante todo o tempo de execução enquanto um **agendador dinâmico** ("dynamic scheduler**")** define tais prioridades em tempo de execução e, portanto, elas são variáveis. O **agendador de taxa monotônica** (RMS – "Rate Monotonic Scheduler**")** é estático e define uma prioridade fixa para cada tarefa baseada no seu período: a tarefa de menor período tem a maior prioridade. Mas um **agendador primeiro prazo fatal primeiro** (FDFS – "First Deadline First Scheduler**"** ou EDF – "Earliest Deadline First**"**) é um agendador dinâmico que contém as informações sobre o prazo máximo de encerramento de cada tarefa e define uma prioridade maior para aquela tarefa que tem o seu limite máximo mais próximo. Um **agendador preempitivo** é aquele que consegue suspender imediatamente uma tarefa em execução, salvar seu contexto para reinício mais tarde, e iniciar a execução de outra tarefa com maior prioridade em qualquer tempo que isso seja solicitado. O **prazo fatal** para a solicitação de cada tarefa é definido como o tempo de chegada da próxima solicitação da mesma tarefa. Num algoritmo agendador, nós diz-se que ocorreu uma **sobreposição** ("overflow") no tempo t, se t é o tempo final de uma solicitação não atendida. O **instante crítico** ("critical instant") de cada tarefa é definido como o instante no qual a solicitação para esta tarefa tenha o maior tempo de resposta possível. O próximo instante crítico irá ocorrer no tempo mínimo múltiplo comum (MMC) do período de todas as tarefas. A **zona crítica** ("critical time zone") para uma determinada tarefa é o tempo de intervalo entre o instante crítico e o fim da resposta à correspondente solicitação daquela tarefa. Um conjunto de tarefas **utiliza totalmente** ("fully utilize") um processador se a atribuição de prioridades deste conjunto é realizável e o aumento de uma unidade de tempo na execução de qualquer tarefa torna essa mesma atribuição

irrealizável. O **deslocamento** de uma tarefa ("skew**")** é o atraso entre o tempo de solicitação T_i e o tempo de início da execução da tarefa C_i.

4.2 Teoremas e Resultados Importantes Para Este Trabalho

Embora seja vasta a literatura sobre agendadores estáticos e dinâmicos, serão considerados neste capítulo somente os teoremas que são considerados a base teórica da área e de relevante importância para o trabalho proposto.

O desenvolvimento de resultados analíticos é baseado em premissas e/ou simplificações necessárias para a solução do problema. Posteriormente essas premissas podem ou não ser relaxadas. Em nosso trabalho, os ambientes de sistema de tempo real crítico são baseados nas mesmas premissas utilizadas por Liu e Layland (1973): (A1) as solicitações para execução de todas tarefas que cumpram os requisitos de tempo ("deadlines") são periódicas, com intervalos de tempo constante entre as solicitações; (A2) cumprir requisitos de tempo significa capacidade de execução somente, ou seja, cada tarefa deve ser executada antes da próxima solicitação da mesma tarefa; (A3) as tarefas são independentes no sentido de que a solicitação de uma certa tarefa não dependa do início ou encerramento da execução de qualquer outra tarefa; (A4) o tempo de execução de cada tarefa é constante e não varia com o tempo e é o tempo gasto pelo processador para executar uma tarefa sem qualquer interrupção; e (A5) todas as tarefas não periódicas são especiais e não possuem restrições de execução no tempo. A alteração de uma premissa em um determinado contexto, quando necessária, será esclarecida.

4.2.1 Um Algoritmo Agendador de Prioridade Fixa

Liu e Layland (1973) propuseram o teorema 1: O instante crítico para qualquer tarefa ocorre a qualquer momento em que a tarefa é solicitada simultaneamente com todas as tarefas de maior prioridade.

Prova: Veja Liu e Layland (1973). A prova é clara e de fácil compreensão embora não seja apresentada aqui. O resultado do teorema 1 também sugere que a atribuição de

prioridade é ótima no sentido em que é proposta no teorema 2. Tal atribuição de prioridade será conhecida como taxa monotônica. Como se verá, tal atribuição de prioridade é ótima no sentido em que nenhuma outra forma de atribuição estática será realizável se a atribuição da taxa monotônica não puder ser realizada para este conjunto de tarefas. Se a solicitação para todas as tarefas em seus instantes críticos é atendida antes do seu prazo fatal, então o algoritmo agendador é realizável.

Liu e Layland (1973) propuseram o teorema 2: Se existe uma atribuição de prioridades realizável para um conjunto de tarefas, então a atribuição taxa monotônica é realizável para este mesmo conjunto.

Prova: Veja Liu e Layland (1973). A prova é clara e de fácil compreensão e não será apresentada, mas é esclarecido aqui que o mesmo só é aplicável à família de agendadores estáticos.

4.2.2 Fator de Utilização do Processador

Liu e Layland (1973) definiram fator de utilização do processador como a fração de tempo necessária para a execução de um conjunto de tarefas. Como (C_i / T_i) é a fração de tempo do processador para a execução da tarefa τ_i, para m tarefas, o fator de utilização será:

$$U(C_i, T_i) = \sum_{i=1}^{m} (C_i / T_i). \quad (1)$$

Entretanto, nesta definição, deve se clarificar que (1) pode ser reescrito assim:

$$U(C_i, T_i, M) = \sum_{i=1}^{m} \{(M / T_i).(C_i / M)\} \quad (2)$$

onde M é o mínimo múltiplo comum (MMC) dos períodos de todas as tarefas. Se os períodos de cada tarefa são mutuamente primos, então $M = T_1.T_2 \ldots T_m$. Definindo $n_i = M / T_i$ como o número de vezes que a tarefa i ocorre no intervalo de tempo M, tem-se:

$$U(n_i, C_i, M) = \{\sum_{i=1}^{m} (n_i.C_i)\} / M \ . \qquad (3)$$

Pode se dizer que o fator de utilização do processador é o percentual de tempo gasto pelo processador executando todas as tarefas no intervalo de tempo M . Se não existe nenhum fator comum entre qualquer (T_i, T_j) então, quando m cresce, M cresce mais rápido. Mesmo que defina um conjunto de tarefas com todos os deslocamentos iguais a zero e que utilizem totalmente o processador considerando somente o intervalo de tempo T_m, o fator de utilização do processador só tem significado real no intervalo de tempo M .

4.2.3 Menor Limitante Superior

Liu e Layland (1973) propuseram o teorema 3: Para um conjunto de duas tarefas com prioridade fixa, o menor limitante superior ("least upper bound" - lub) para o fator de utilização do processador é $U_{\text{lub}} = 2.(2^{1/2} - 1)$.

A prova é esclarecida aqui incluindo a representação gráfica do conjunto de tarefas para os dois casos possíveis.

Seja τ_1, τ_2 duas tarefas com períodos T_1 e T_2 e seus tempos de execução C_1 e C_2, respectivamente. Assumindo que $T_1 > T_2$, sem perda de generalidade. De acordo com a atribuição de prioridade taxa monotônica, τ_1 tem maior prioridade que τ_2. Na zona crítica de tempo de τ_2 existem $\lceil T_2 / T_1 \rceil$ ("teto" de T_1 / T_2 = menor inteiro $\geq \frac{T_2}{T_1}$) solicitações para τ_1. Se C_2 for ajustado para a completa utilização do processador, tem-se dois casos:

Caso 1: todas as solicitações de τ_1 que acontecem na zona crítica de T_2 são completadas antes da segunda solicitação de τ_2 - FIGURA 4.1.

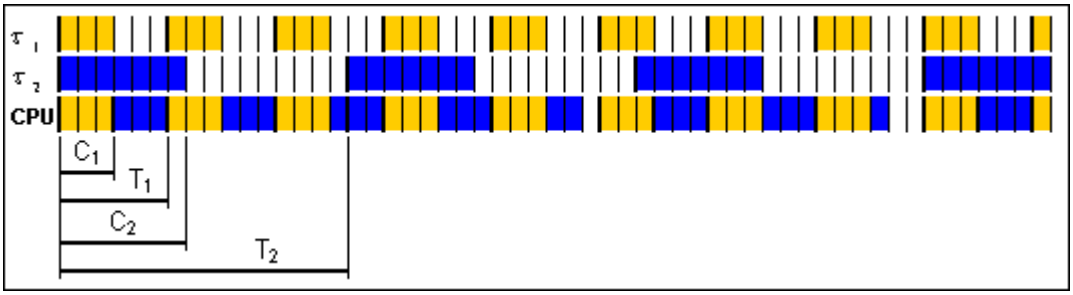

FIGURA 4.1 – Agendamento de tarefas para o caso 1, com $\tau_1 = \{T_1 = 6, C_1 = 3\}$ e $\tau_2 = \{T_2 = 16, C_2 = 7\}$ unidades de tempo.

Usando os resultados de Liu e Layland (1973), onde $C_2 = C_2(C_1, T_1, T_2)$, tem-se:

$$U(C_1, T_1, T_2) = 1 + C_1.\underbrace{\left[(1/T_1) - (1/T_2).\lceil T_2/T_1 \rceil\right]}_{\leq 0}. \qquad (4)$$

Neste caso, o fator de utilização do processador $U(C_1, T_1, T_2)$ é monotonicamente decrescente em C_1.

Case 2: a execução da $\lceil T_2/T_1 \rceil^{ésima}$ solicitação para τ_1 ocorre após a segunda solicitação de τ_2 - FIGURA 4.2.

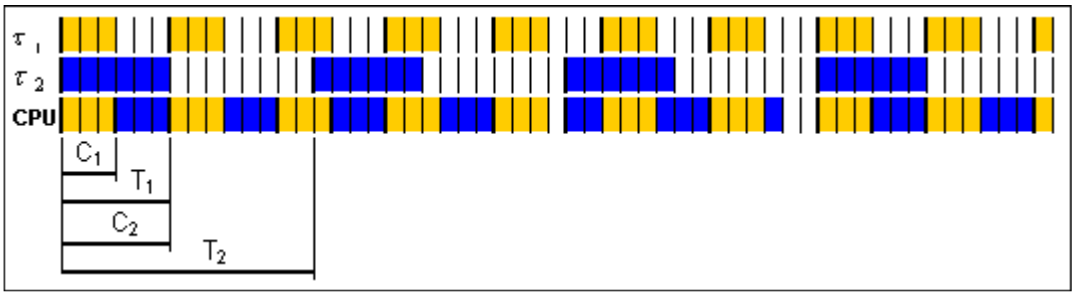

FIGURA 4.2 – Agendamento de tarefas para o caso 2, com $\tau_1 = \{T_1 = 6, C_1 = 3\}$ e $\tau_2 = \{T_2 = 14, C_2 = 6\}$ unidades de tempo.

Usando os resultados de Liu e Layland (1973), onde $C_2 = C_2(C_1, T_1, T_2)$ e $\lfloor T_2/T_1 \rfloor$ é "chão" de T_2/T_1 = maior inteiro $\leq T_2/T_1$, tem-se:

$$U = (T_1 / T_2).\lfloor T_2 / T_1 \rfloor + C_1.\underbrace{[(1/T_1) - (1/T_2).\lfloor T_2 / T_1 \rfloor]}_{\geq 0}. \quad (5)$$

Neste caso, o fator de utilização do processador $U(C_1, T_1, T_2)$ é monotonicamente crescente com C_1.

O mínimo de $U(C_1, T_1, T_2)$ claramente ocorre entre os limites de (3) e (4). Então:

$$C_1 = T_2 - T_1.\lfloor T_2 / T_1 \rfloor. \quad (6)$$

Conforme demonstrado em Liu e Layland (1973) este valor é:

$$U(T_1, T_2) = 1 - (T_1 / T_2).[\lceil T_2 / T_1 \rceil - (T_2 / T_1)].[(T_2 / T_1) - \lfloor T_2 / T_1 \rfloor], \quad (7)$$

e o menor limitante superior de $U(T_1, T_2)$ para todos T_1, T_2 é numericamente $U_{\text{lub}} = 2.(2^{1/2} - 1) \approx 0.83$.

4.2.4 O Agendador de Taxa Constante Restrito

Outra proposição de Liu e Layland (1973) é o teorema 4: Para um conjunto de m tarefas com prioridade fixa, e a restrição de que a razão entre quaisquer dois períodos das tarefas solicitadas seja menor que 2, o menor limitante superior para o fator de utilização do processador é $U_{\text{lub}} = m.(2^{1/m} - 1)$.

Prova: Seja τ_1, τ_2, …, τ_m representar m tarefas periódicas. Seja C_1, C_2, …, C_m os tempos de execução das tarefas e que minimizem o fator de utilização do processador. Assume-se que $T_1 < T_2 < ... < T_{m-1} < T_m$. Seja $U(C_i, T_i)$ o fator de utilização do processador.

Prova-se o teorema como em Liu e Layland (1973) e ilustra-se a prova para o caso $\tau_1 = \{T_1 = 8, C_1^{\text{lub}} = 3\}$, $\tau_2 = \{T_2 = 11, C_2^{\text{lub}} = 4\}$ e $\tau_3 = \{T_3 = 15, C_3^{\text{lub}} = 1\}$.

Pretende-se provar que (lema 1): $C_1^{\text{lub}} = T_2 - T_1$.

Supondo que $C_1 = T_2 - T_1 + \Delta$, $\Delta > 0$, $\Delta = 1$ então, tem-se um novo conjunto de tarefas $\tau_1 = \{T_1 = 8, C_1 = 4\}$, $\tau_2 = \{T_2 = 11, C_2 = 3\}$ e $\tau_3 = \{T_3 = 15, C_3 = 1\}$ conforme mostrado na FIGURA 4.3 e que claramente utilizam totalmente o processador. Esta mudança é equivalente a transferir Δ da tarefa τ_2 para a tarefa τ_1. O primeiro Δ de C_2^{lub} compensa diretamente C_1. O segundo Δ, adicionado ao segundo C_1, atrasa o segundo C_2 de Δ sem sobreposição porque é a mesma quantidade de tempo retirado ao final de C_2.

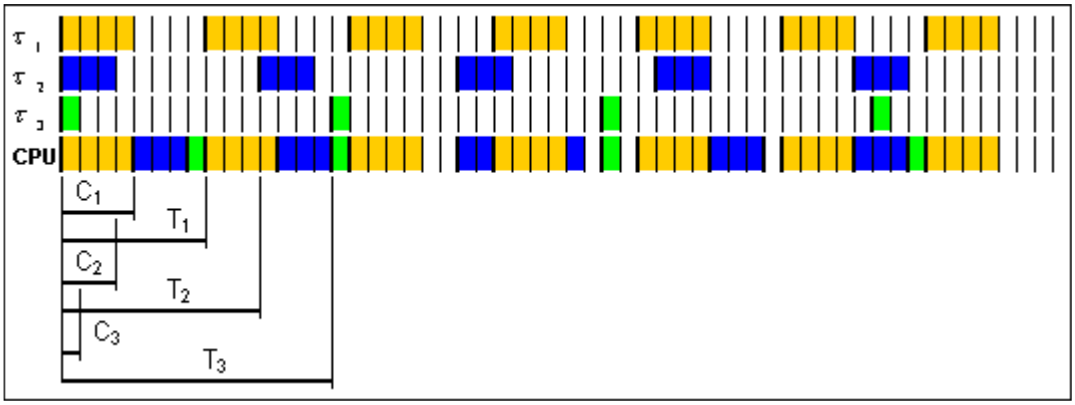

FIGURA 4.3 – Utilização total com $C_1 = T_2 - T_1 + \Delta$.

Reescrevendo o conjunto de tarefas como $C_1^{'}, C_2^{'}, ..., C_m^{'}$, onde:

$$C_1^{'} = T_2 - T_1$$

$$C_2^{'} = C_2 + \Delta$$

$$C_3^{'} = C_3$$

$$\vdots$$

$$C_{m-1}^{'} = C_{m-1}$$

$$C_m^{'} = C_m$$

Se $U^{'}$ representa o correspondente fator de utilização, tem-se:

$$U - U^{'} = (\Delta / T_1) - (\Delta / T_2) > 0 . \quad (8)$$

Um fato importante não mencionado em por Liu e Layland (1973), mas introduzido aqui, é que se τ_i, $i = 2,...m$, satisfaz $C_i > \Delta$, pode-se transferir Δ de qualquer tarefa τ_i $(1 < i \leq m)$ para τ_1 é o resultado ainda será válido, i.e., $U_{\text{lub}} - U^{i} = (\Delta / T_1) - (\Delta / T_i) > 0$. Mais ainda, $U_{\text{lub}} - U^{i} < U_{\text{lub}} - U^{i+1}$ porque $T_i < T_{i+1}$.

Alternativamente, supondo que $C_1 = T_2 - T_1 - \Delta$, $\Delta > 0$, $\Delta = 1$ então, tem-se um novo conjunto de tarefas $\tau_2 = \{T_2 = 11, C_2 = 6\}$ e $\tau_3 = \{T_3 = 15, C_3 = 1\}$ como mostrado na FIGURA 4.4 e que claramente utilizam totalmente o processador. Essa suposição é equivalente a transferir Δ da tarefa τ_1 para a tarefa τ_2. Porque, sob as hipóteses do teorema, sempre haverá duas execuções C_1^{lub} antes da segunda solicitação de τ_2 no intervalo de tempo T_2, e deve-se adicionar em C_2 o dobro do valor Δ subtraído de C_1^{lub}. Na FIGURA 4.4 pode-se verificar que no primeiro intervalo de tempo T_2 tem-se duas vezes a execução de C_1, no primeiro intervalo de T_3 também tem-se duas vezes a execução de C_2, e a primeira execução de C_3 ocorre sempre e justamente antes de se completar o primeiro intervalo de tempo T_1.

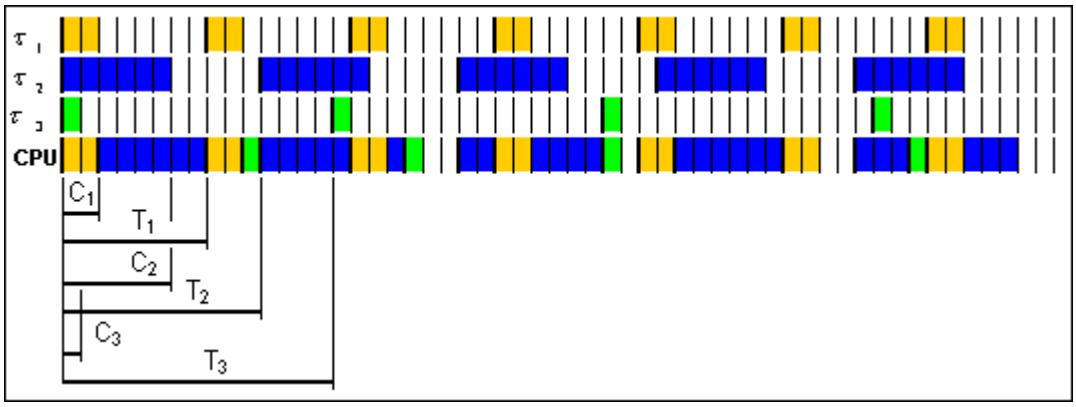

FIGURA 4.4 - Utilização total com $C_1 = T_2 - T_1 - \Delta$.

Reescreve-se o conjunto de tarefas como $C_1^{"}, C_2^{"}, ..., C_m^{"}$, onde:

$$C_1^{"} = T_2 - T_1$$

$$C_2^{"} = C_2 - 2.\Delta$$

$$C_3^{"} = C_3$$

$$\vdots$$

$$C_{m-1}^{"} = C_{m-1}$$

$$C_m^{"} = C_m$$

Se $U^{"}$ representa o correspondente fator de utilização, tem-se:

$$U - U^{"} = -(\Delta / T_1) + 2.(\Delta / T_2) > 0 . \quad (9)$$

Outro fato importante não mencionado por Liu e Layland (1973), mas introduzido aqui, é que se pode retirar Δ da tarefa τ_1 e adicionar $2.\Delta$ a qualquer tarefa τ_i $(1 < i \leq m)$ é o resultado ainda é válido, i.e., $U_{\text{lub}} - U^i = -(\Delta / T_1) + (2.\Delta / T_i) > 0$, e novamente, $U_{\text{lub}} - U^i < U_{\text{lub}} - U^{i+1}$.

Entretanto, se realmente U é o mínimo fator de utilização, então $C_1 = T_2 - T_1$. No exemplo numérico anterior nós pode-se verificar que os dois conjuntos de tarefas modificados $C_1^{'}, C_2^{'}, ..., C_m^{'}$ e $C_1^{"}, C_2^{"}, ..., C_m^{"}$ satisfazem esta condição e são iguais a $C_1^{\text{lub}}, C_2^{\text{lub}}, ..., C_m^{\text{lub}}$.

Quer-se provar que (lema 2): $C_m^{\text{lub}} = T_m - T_{m-1}$.

Liu e Layland disseram, “de forma semelhante pode-se provar que”, conforme mostrado em seu trabalho de 1973:

$$C_2 = T_3 - T_2$$

$$C_3 = T_4 - T_3$$

$$\vdots$$

$$C_{m-1} = T_m - T_{m-1}$$

$$C_m = T_m - 2.(C_1 + C_2 + ... + C_{m-1})$$

Mas eles não provaram isto naquele trabalho. Então, isto é esclarecido usando a FIGURA 4.5.

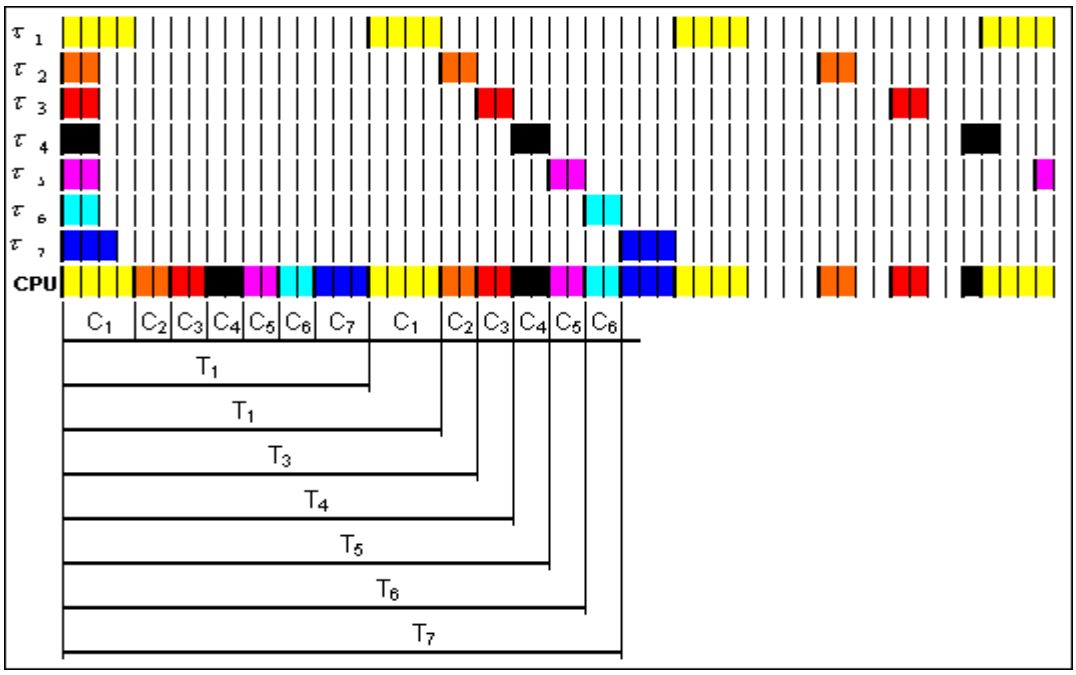

FIGURA 4.5 - Utilização total com m = 7.

O resto da prova é o mesmo apresentado por Liu e Layland (1973); e trás o importante resultado:

$$U^{m}_{\text{lub}(m)} = m.(2^{1/m} - 1)\,. \qquad (10)$$

4.2.5 O Agendador de Taxa Constante Generalizado

Liu e Layland (1973) generalizaram o teorema 4 como o teorema 5: Para um conjunto de m tarefas com ordem de prioridade fixa, o menor limitante superior do fator de utilização do processador é $U^{m}_{\mathrm{lub}(m)} = m.(2^{1/m} - 1)$.

Prova: para isto eles provaram que para qualquer conjunto de tarefas que utilizam totalmente o processador, o fator de utilização do processador U' é menor ou igual a U^{m}_{lub} $(U' \leq U^{m}_{\mathrm{lub}})$ e, consequentemente, para determinar o menor limitante superior deste fator é necessário e suficiente somente considerar os conjuntos de tarefas nos quais a relação entre períodos de quaisquer duas tarefas solicitadas é menor que dois.

4.2.6 Contraprovas Falsas na Literatura

Devillers e Goossens (2000) usaram um exemplo numérico $\tau_1 = \{T_1 = 8, C_1 = 2\}$, $\tau_2 = \{T_2 = 11, C_2 = 3\}$ e $\tau_3 = \{T_3 = 15, C_3 = 5\}$ como um "contra exemplo" para mostrar alguns "erros" no teorema 4. Esclarece-se então o teorema 4 e o "contra exemplo" para ter-se uma melhor compreensão de ambos. Considere o conjunto de tarefas $\tau_1 = \{T_1 = 8, C_1\}$, $\tau_2 = \{T_2 = 11, C_2\}$ e $\tau_3 = \{T_3 = 15, C_3\}$, com os mesmos T_1, T_2, T_3 usados no "contra exemplo" de (2000) e todas as combinações C_1, C_2, C_3 realizáveis. O fator de utilização do processador para todas a combinações C_1, C_2, C_3 realizáveis são apresentados na TABELA 4.1 e na FIGURA 4.6. Os valores C_1 estão nas linhas, os valores C_2 nas colunas, e os valores C_3 são calculados para a total utilização do processador.

TABELA 4.1 – Fator de utilização do processador para todas as combinações C_1, C_2, C_3 realizáveis.

$C_2 \setminus C_1$	1	2	3	4	5	6
1	1,083	0,941	0,933	0,924	0,916	0,908
2	0,907	0,898	0,890	0,882	0,873	-
3	0,864	0,856	0,848	0,839	-	-
4	0,822	0,814	**0,805**	-	-	-
5	0,846	0,838	-	-	-	-
6	0,870	0,862	-	-	-	-
7	0,895	-	-	-	-	-
8	0,919	-	-	-	-	-

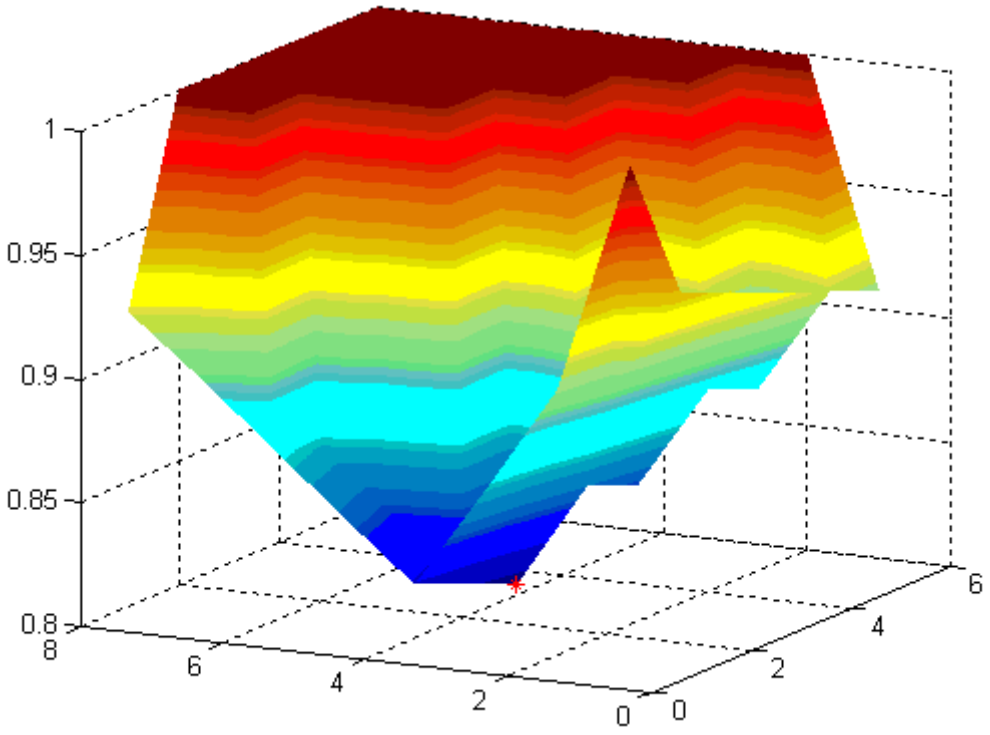

FIGURA 4.6 – Fator de utilização $U(C_1, C_2)$ com C_3 escolhido para utilização total do processador.

O mínimo de U é $U_{\text{lub}} = 0.805$ e ocorre na tarefa $\tau_1 = \{T_1 = 8, C_1^{\text{lub}} = 3\}$, $\tau_2 = \{T_2 = 11, C_2^{\text{lub}} = 4\}$ e $\tau_3 = \{T_3 = 15, C_3^{\text{lub}} = 1\}$. Este conjunto de tarefas é apresentado na FIGURA 4.7.

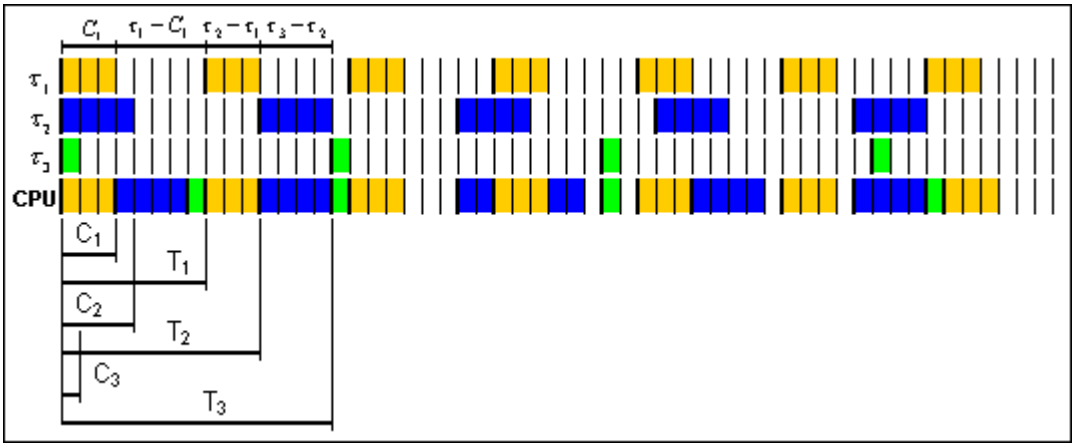

FIGURA 4.7 – Conjunto de tarefas com total utilização do processador e mínimo U.

Então o "contra exemplo" de Devillers e Goossens (2000) está errado porque para contrapor o teorema 4, tal "contra exemplo" deveria satisfazer as hipóteses do teorema 4. Mas não se pode usar a combinação $\tau_1 = \{T_1 = 8, C_1 = 2\}$, $\tau_2 = \{T_2 = 11, C_2 = 3\}$ e $\tau_3 = \{T_3 = 15, C_3 = 5\}$ usada naquele artigo porque ela cumpre a utilização total do processador com $U = 0.856$, mas não minimiza este fator. O teorema 4 diz que se deve utilizar somente o conjunto de tarefas $\{\tau_1 = \{T_1 = 8, C_1^{\text{lub}} = 3\}$, $\tau_2 = \{T_2 = 11, C_2^{\text{lub}} = 4\}$ $\tau_3 = \{T_3 = 15, C_3^{\text{lub}} = 1\}\}$ que minimiza o fator de utilização do processador entre todas as combinações C_1, C_2, C_3 possíveis e satisfaz o critério de total utilização.

4.3 Esclarecimento de Equívocos na Literatura

Em literatura específica sobre este assunto, como em Burns e Wellings (1989), p.347, é apresentada a forma simples das equações de Liu e Layland e das conclusões sobre agendamento: "para um sistema com processador único, um conjunto de m tarefas independentes, processos periódicos com restrições somente de tempo de execução, possui algoritmo de agendamento se e somente se o fator de utilização do processador para todas as tarefas é menor que $m.(2^{1/m} - 1)$. Para um valor grande de m este valor tende assintoticamente para 0.693. Então, todo conjunto de tarefas tem algoritmo de agendamento e seu fator de utilização é menor que 69%".

(E1) Olhando rapidamente, à primeira vista, nós pode se pensar erroneamente que se o fator de utilização é maior que $m.(2^{1/m}-1)$ (para m finito) ou maior que 69,3 % $(m \to \infty)$, o conjunto de tarefas não é possível usando RMS. Este resultado assume que todos os períodos $T_1.T_2 \ldots T_m$ da tarefa são mutuamente primos, a pior relação possível entre todos eles. Se existe uma relação mais favorável entre eles, o sistema pode atingir um melhor fator de utilização do processador. Em alguns casos especiais, como um mostrado em Burns e Wellings (1989), p.347, e reproduzido na FIGURA 4.8, $\tau_1 = \{T_1 = 80, C_1 = 40\}$, $\tau_2 = \{T_2 = 40, C_2 = 10\}$ e $\tau_3 = \{T_3 = 20, C_3 = 5\}$, U atinge 100%.

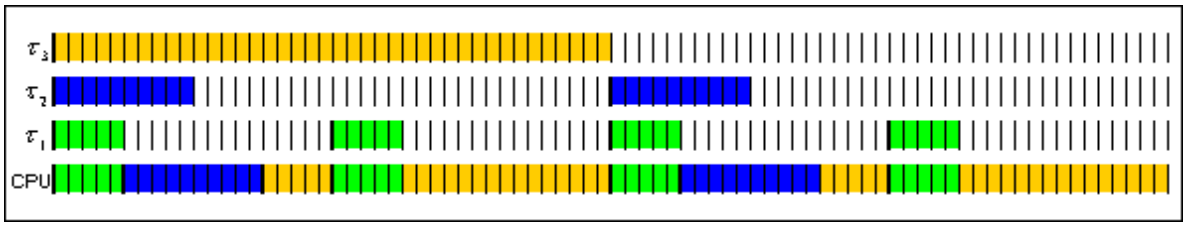

FIGURA 4.8 – Conjunto de tarefas para $\tau_1 = \{T_1 = 80, C_1 = 40\}$, $\tau_2 = \{T_2 = 40, C_2 = 10\}$ e $\tau_3 = \{T_3 = 20, C_3 = 5\}$.

(E2) Por outro lado, também pode se erroneamente pensar que, usando RMS com fator de utilização do processador menor que 69,3 % $(m \to \infty)$ tem-se tempo suficiente para adicionar tarefas porque para valores grandes de m o fator de utilização do processador aproxima-se assintoticamente de 0.693. Em nosso exercício profissional este erro foi detectado como item de especificação de contratos de simuladores. Mais que isso, nós enfrentamos problemas reais onde tínhamos tempo disponível no processador e não pudemos adicionar nada. De fato, para um conjunto de tarefas que utiliza completamente o processador, se pelo menos uma unidade de tempo é adicionada, o algoritmo não será mais realizável mesmo que haja tempo livre do processador. Nesta situação nós deve se analisar um outro algoritmo de agendamento ou, no pior caso, trocar o processador por um mais rápido.

4.4 Restrições do Algoritmo RMS

Nas seções anteriores foi demonstrado que utilizando algoritmos RMS todo conjunto de tarefas é realizável se o fator de utilização do processador é menor que 69,3% usando a pior relação (mutuamente primos) possível entre todos os períodos das tarefas. Pode se melhorar isso escolhendo $T_1.T_2 \ldots T_m$ com boas relações entre (T_i, T_j). Como mostrado em exemplo anterior, a maneira mais simples de se atingir um fator de utilização do processador de 100% usando RMS é forçar as partes fracionais de T_m / T_i, $\{T_m / T_i\} = 0$, para $i = 1,2,...,m-1$. Isto é bom, mas não é suficiente, porque no mundo real nem sempre é possível realizar isso e porque ainda deve ser considerado o custo de chaveamento entre as tarefas.

4.5 Algoritmo Primeiro Prazo Fatal Primeiro (FDFS) ou (EDF)

Na segunda parte do seu trabalho, Liu e Layland (1973) provam que algoritmos FDFS ou EDF podem agendar qualquer conjunto de tarefas que são possíveis por qualquer outro método; e que o menor limitante superior de fator de utilização do processador é uniformemente 100%. Liu e Layland (1973) propuseram e provaram o teorema 7 (numeração deles): para um dado conjunto de m tarefas, o algoritmo que atende ao primeiro prazo fatal primeiro é realizável se e somente se

$$(C_1 / T_1) + (C_2 / T_2) + ... + (C_m / T_m) \leq 1. \qquad (11)$$

Utilizando a notação apresentada em (3), com $M = T_1.T_2 \ldots T_m$ e $n_i = M / T_i$, pode se reescrever (11) da seguinte forma:

$$\{\sum_{i=1}^{m} (n_i.C_i)\} / M \leq 1. \qquad (12)$$

Uma representação gráfica do algoritmo FDFS para o conjunto de tarefas $\tau_1 = \{T_1 = 8, C_1 = 3\}$, $\tau_2 = \{T_2 = 11, C_2 = 4\}$ e $\tau_3 = \{T_3 = 15, C_3 = 1\}$ usado no teorema 4 é apresentada na FIGURA 4.9.

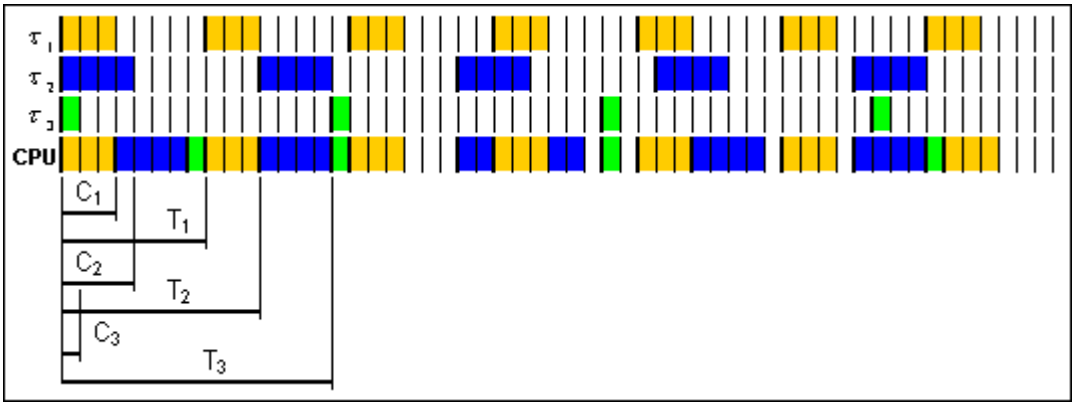

FIGURA 4.9 – FDFS para tarefas $\tau_1 = \{T_1 = 8, C_1 = 3\}$, $\tau_2 = \{T_2 = 11, C_2 = 4\}$ e $\tau_3 = \{T_3 = 15, C_3 = 1\}$.

4.6 Restrições do Algoritmo FDFS ou EDF

Na seção anterior foi mostrado que o algoritmo FDFS ou EDF melhora o algoritmo RMS permitindo um fator de utilização do processador de até 100% para um conjunto de tarefas realizável. Entretanto esta aparente vantagem apresenta alguns problemas importantes.

Primeiro, para permitir o uso máximo do processador ele empurra o não cumprimento de tarefas para o primeiro tempo disponível no futuro e, no limite da possibilidade do agendamento, utiliza a última unidade de tempo antes do tempo M. Desta forma o não cumprimento de qualquer tarefa só será verificado no tempo M. Para um número grande de tarefas com seus períodos diferentes o mínimo múltiplo comum M tende a ser um valor muito grande e, desta forma, pode se dizer que este algoritmo tenta "esconder" o problema.

Segundo, ele apresenta um grande número de suspensões e reentrâncias de tarefas. Para um número elevado de tarefas, a somatória dos tempos envolvidos nestas suspensões e reentrâncias torna-se significativo. A teoria foi desenvolvida para a situação onde estes tempos devem ser considerados, mas aumenta a complexidade do algoritmo além de não terem sido desenvolvidas para aplicações em ambientes complexos com mudanças dinâmicas.

Terceiro, deve se sempre considerar a capacidade de implementação no mundo real. O algoritmo básico já exige uma implementação com mecanismos mais complexos que aumentam ainda mais quando considerados os tempos envolvidos no chaveamento e, para ambientes complexos e dinâmicos, consomem mais tempo com tendência à perda de previsibilidade de execução do mesmo algoritmo.

CAPÍTULO 5

NOVAS ESTRATÉGIAS PARA AGENDADORES/ESCALONADORES EM TEMPO REAL E SEU IMPACTO SOBRE OS JÁ EXISTENTES

5.1 Problemas Atuais e Necessidade de Novas Estratégias

A evolução constante da computação e das comunicações traz novos desafios ao conhecimento humano. Devido a este avanço, em algumas aplicações, o número de unidades de processamento e transceptores supera as dezenas, e os programas embarcados chegam a milhares ou milhões de linhas código. Com a possibilidade de processamento de milhões e até mesmo de bilhões de tarefas ou modelos simulados surgem problemas de natureza diferente dos que encontramos até os dias de hoje. Simultaneamente, os avanços da modelagem e da simulação permitiram a solução de problemas anteriormente insolúveis, somente com o uso de técnicas analíticas, fazendo uso do pré-processamento de modelos ou pelo processamento simultâneo ao tempo do sistema.

Entre estes problemas atuais, conforme citado na motivação deste trabalho, destacam-se: o projeto e operação dos **sistemas de controle e defesa de tráfego aéreo**, modernizando os sistemas de controle de aeroportos, torres de controle, sistemas de comunicação global, preparando o mundo para vôos de rota livre (“free flight”), interconectados com os sistemas de defesa; a **simulação de detritos espaciais** seja para a coleta ou descarte, seja para a segurança de vôos tripulados ou não; o **controle e operação de constelações de satélites**, para aumentar a confiabilidade e também a economia de combustível não renovável; a s**imulação e operação de viagens interplanetárias,** devido à impossibilidade prática da solução analítica do problema de n corpos, e devido à ocorrência de eventos aleatórios no espaço; e o **controle e operação de componentes no corpo humano,** para a melhoria dos problemas de deficiência física.

Pela sua natureza, estes problemas usam sistemas de tempo real críticos e, consequentemente, seus agendadores necessitam das mesmas propriedades já apresentadas no Capítulo 2. Entre as necessidades dos agendadores destacamos: ter, em tempo de projeto, a garantia da realização das tarefas em tempo de execução ("predictability"); eliminar incertezas de tempo, ser dinâmico, ter capacidade dinâmica, boa relação custo/benefício de implementação (simplicidade de cálculo), e capacidade de "criar" tempo até o extremo. Em especial, acreditamos que os agendadores do futuro, utilizados na solução dos problemas apresentados, devam se aproximar da forma de agendamento de tarefas do cérebro humano.

Neste Capítulo 5 apresentaremos uma sequência de melhorias, alterações e extensão dos agendadores existentes, a partir da base sólida do Agendador de Taxa Constante - ATC ("Rate Monotonic Scheduler - RMS"), passando pelos agendadores dinâmicos existentes, até culminar em uma nova estratégia/família aqui proposta que contemple as necessidades atuais expostas e elimine as restrições que cada agendador anterior apresente. Este desenvolvimento é efetuado para sistemas de tempo real que cumpram as mesmas premissas apresentadas no Capítulo 4. O relaxamento de algumas premissas é analisado nos algoritmos propostos no Capítulo 6.

Os agendadores estáticos baseados em tabelas não são incluídos em nossa sequência porque, embora eles sejam altamente previsíveis, eles são também altamente inflexíveis. Primeiro, qualquer modificação efetuada em qualquer tarefa pode tornar o agendador não realizável. Segundo, sua natureza não permite alterações dinâmicas, que são características fundamentais dos sistemas de interesse deste trabalho. As tarefas não periódicas podem ter uma parte do tempo reservado durante o agendamento estático. Para tarefas periódicas só existe um agendamento possível se e somente se existir um agendamento possível para o mínimo múltiplo comum (MMC) dos períodos das tarefas e, o conjunto de tarefas será executado repetidamente a cada intervalo de tempo igual ao MMC.

5.2 Prova da Existência e das Propriedades da Superfície de Agendabilidade Máxima

Os fundamentos da teoria de agendadores, organizados de uma maneira formal, foram apresentados por Liu e Layland (1973). Embora os resultados apresentados por eles tenham sido e ainda sejam utilizados praticamente para toda a teoria moderna dos agendadores, eles foram contestados recentemente por Devillers e Goossens (2000). No Capítulo 4, ampliamos e esclarecemos a prova original, rejeitando os argumentos em contrário de Devillers e Goossens (2000), e consolidando ainda mais os resultados apresentados originalmente por Liu e Layland (1973). Pela sua solidez, estes resultados serão utilizados como ponto de partida da sequência de melhorias, alterações e extensão dos agendadores existentes, que propomos neste Capítulo.

Na primeira parte do trabalho de Liu e Layland (1973), foi desenvolvido e apresentado um critério necessário e suficiente para a existência de uma agendador de tarefas estático com sustação baseado em prioridade chamado de agendador a taxa constante ou, em Inglês, por “Rate Monotonic Scheduler” (RMS).

O teorema 5 de Liu e Layland (1973) é: Para um conjunto de m tarefas com ordem de prioridade fixa, o menor limitante superior (“least upper bound”- lub) do fator de utilização do processador é $U^{m}_{\mathrm{lub}(m)} = m.(2^{1/m} - 1)$.

A grande vantagem deste resultado é estabelecer: 1) o menor limitante superior $U_{lub}(m)$ do fator de utilização U(m, Ti, Ci) do processador na condição de máxima utilização (“full utilization”) do processador em função apenas do número m de tarefas, mas não de suas características (períodos Ti, durações Ci, etc.); e 2) a condição em que ele ocorre. Como o teorema é baseado na condição de **máxima utilização** do processador (“full utilization”) podemos afirmar que os pontos formados pelo conjunto dos limitantes superiores (“upper bounds”) $U_{ub}(m, Ti, Ci)$ do fator de utilização U(m, Ti, Ci) formam uma superfície discreta, côncava, com mínimo em $U_{lub}(m)$, crescente em todas as direções a partir deste ponto até, em algumas, atingir e ultrapassar 100% (ver FIGURA 5.1, TABELA 5.1 e TABELA 5.2) e que para cada conjunto (m, Ti, Ci)

determina o limite entre a agendabilidade ou não das tarefas. Esta superfície será chamada de **a superfície de agendabilidade máxima AM** e suas propriedades serão analisadas com maior detalhe a seguir, para maior compreensão e desenvolvimento.

5.3 Visão Gráfica de U no Espaço C_i e T_i

Uma representação gráfica, no espaço tridimensional, para duas **superfícies de agendabilidade máxima** AM1 e AM2 do fator de utilização do processador U para todas a combinações C_1, C_2, C_3 de cada conjunto de tarefas, conforme TABELA 5.1 e TABELA 5.2, é apresentada na FIGURA 5.1. Os valores de C_1 estão nas linhas, os valores de C_2 nas colunas, e os valores de C_3 são calculados para a total utilização do processador quando for possível o agendamento das tarefas; e, caso contrário, $C_3 = 0$. Os valores destacados na linha azul representam o fator de utilização mínimo para cada C_1.

A TABELA 5.1 contém os fatores de utilização do processador para a tarefa $\tau_1 = \{T_1 = 8, C_1\}$, $\tau_2 = \{T_2 = 11, C_2\}$ e $\tau_3 = \{T_3 = 15, C_3\}$, com $U_{\min} = 0{,}805$.

TABELA 5.1 - Fator de utilização para $\tau_1 = \{T_1 = 8, C_1\}$, $\tau_2 = \{T_2 = 11, C_2\}$ e $\tau_3 = \{T_3 = 15, C_3\}$.

τ_1	8	C1	C1	C1	C1	C1	C1	C1	C1
τ_2	11	1	2	3	4	5	6	7	8
τ_3	15	0,822	0,814	0,805	0,839	0,873	0,908	0,966	1,091
1	U	0,949	0,941	0,933	0,924	0,916	0,908	0,966	1,091
2	U	0,907	0,898	0,890	0,882	0,873	0,932	1,057	1,182
3	U	0,864	0,856	0,848	0,839	0,898	1,023	1,148	1,273
4	U	0,822	0,814	0,805	0,864	0,989	1,114	1,239	1,364
5	U	0,846	0,838	0,830	0,955	1,080	1,205	1,330	1,455
6	U	0,870	0,862	0,920	1,045	1,170	1,295	1,420	1,545
7	U	0,895	0,886	1,011	1,136	1,261	1,386	1,511	1,636
8	U	0,919	0,977	1,102	1,227	1,352	1,477	1,602	1,727
9	U	0,943	1,068	1,193	1,318	1,443	1,568	1,693	1,818
10	U	1,034	1,159	1,284	1,409	1,534	1,659	1,784	1,909
11	U	1,125	1,250	1,375	1,500	1,625	1,750	1,875	2,000

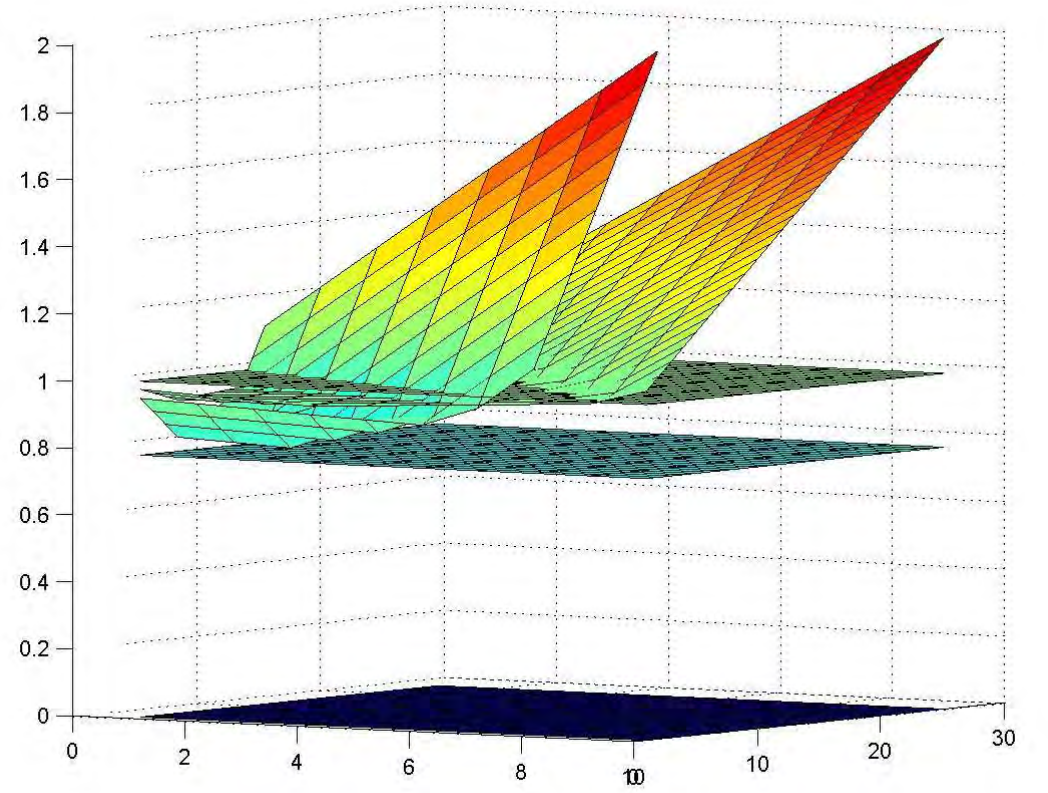

FIGURA 5.1 – Superfície do fator de utilização total do processador para duas tarefas.

A TABELA 5.2 contém os fatores de utilização do processador para a tarefa $\tau_1 = \{T_1 = 10, C_1\}$, $\tau_2 = \{T_2 = 25, C_2\}$ e $\tau_3 = \{T_3 = 55, C_3\}$, com $U_{\min} = 0{,}933$.

Foi adicionada uma superfície plana na FIGURA 5.1, definida como **o plano de Liu e Layland LL**, no menor limitante superior do fator de utilização do processador, isto é, U = $U^m_{\text{lub}(m)} = m.(2^{1/m} - 1)$, consequentemente passando abaixo dos pontos $U_{\min} = 0{,}805$ e $U_{\min} = 0{,}933$. Também foi adicionada uma superfície na mesma figura, chamado de **o plano limite do processador LP**, para $U = 1$, representando o limite de uso de 100% do processador.

TABELA 5.2 - Fator de utilização para $\tau_1 = \{T_1 = 10, C_1\}$, $\tau_2 = \{T_2 = 25, C_2\}$ e $\tau_3 = \{T_3 = 55, C_3\}$.

τ_1	10	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1
τ_2	25	1	2	3	4	5	6	7	8	9	10
τ_3	55	0,933	0,938	0,944	0,949	0,958	0,967	0,976	0,985	0,995	1,040
1	U	0,976	0,967	0,958	0,949	0,958	0,967	0,976	0,985	0,995	1,040
2	U	0,962	0,953	0,944	0,953	0,962	0,971	0,980	0,989	0,998	1,080
3	U	0,947	0,938	0,947	0,956	0,965	0,975	0,984	0,993	1,020	1,120
4	U	0,933	0,942	0,951	0,960	0,969	0,978	0,987	0,996	1,060	1,160
5	U	0,936	0,945	0,955	0,964	0,973	0,982	0,991	1,000	1,100	1,200
6	U	0,940	0,949	0,958	0,967	0,976	0,985	0,995	1,040	1,140	1,240
7	U	0,944	0,953	0,962	0,971	0,980	0,989	0,998	1,080	1,180	1,280
8	U	0,947	0,956	0,965	0,975	0,984	0,993	1,020	1,120	1,220	1,320
9	U	0,951	0,960	0,969	0,978	0,987	0,996	1,060	1,160	1,260	1,360
10	U	0,955	0,964	0,973	0,982	0,991	1,000	1,100	1,200	1,300	1,400
11	U	0,958	0,967	0,976	0,985	0,995	1,040	1,140	1,240	1,340	1,440
12	U	0,962	0,971	0,980	0,989	0,998	1,080	1,180	1,280	1,380	1,480
13	U	0,965	0,975	0,984	0,993	1,020	1,120	1,220	1,320	1,420	1,520
14	U	0,969	0,978	0,987	0,996	1,060	1,160	1,260	1,360	1,460	1,560
15	U	0,973	0,982	0,991	1,000	1,100	1,200	1,300	1,400	1,500	1,600
16	U	0,976	0,985	0,995	1,040	1,140	1,240	1,340	1,440	1,540	1,640
17	U	0,980	0,989	0,998	1,080	1,180	1,280	1,380	1,480	1,580	1,680
18	U	0,984	0,993	1,020	1,120	1,220	1,320	1,420	1,520	1,620	1,720
19	U	0,987	0,996	1,060	1,160	1,260	1,360	1,460	1,560	1,660	1,760
20	U	0,991	1,000	1,100	1,200	1,300	1,400	1,500	1,600	1,700	1,800
21	U	0,995	1,040	1,140	1,240	1,340	1,440	1,540	1,640	1,740	1,840
22	U	0,998	1,080	1,180	1,280	1,380	1,480	1,580	1,680	1,780	1,880
23	U	1,020	1,120	1,220	1,320	1,420	1,520	1,620	1,720	1,820	1,920
24	U	1,060	1,160	1,260	1,360	1,460	1,560	1,660	1,760	1,860	1,960
25	U	1,100	1,200	1,300	1,400	1,500	1,600	1,700	1,800	1,900	2,000

Para a análise da superfície de agendabilidade máxima sob o ponto de vista da agendabilidade ou não do conjunto de m tarefas, onde $T_1.T_2$ ....T_m são os períodos definidos das tarefas e C_1, C_2, …, C_m, variáveis, são os respectivos tempos de execução das mesmas tarefas, utilizaremos analogia entre os espaços n-dimensional e o tridimensional e dividiremos o espaço tridimensional em três regiões : R1) abaixo do plano de Liu e Layland ($U \leq U_{lub}(m)$) inclusive, R2) acima do plano limite do

processador (1 < U), e R3) acima do plano de Liu e Layland até o plano limite do processador ($U_{lub}(m) < U \leq 1$) inclusive.

5.3.1 Abaixo do e Sobre o Plano de Liu e Layland

A região R1 ($U \leq U_{lub}(m)$) abaixo do e sobre o plano de Liu e Layland LL ($U = U^{m}_{lub(m)} = m.(2^{1/m} - 1)$) , representa o espaço de Us para todas as combinações possíveis C_1, C_2, …, C_3, nas quais a agendabilidade é garantida pelo teorema do menor limitante superior de Liu e Layland no espaço tri-dimensional e, por analogia, no espaço n-dimensional, para as combinações C_1, C_2, …, C_m. A principal característica deste critério é que o mesmo é rápido e garantido, permitindo a verificação da agendabilidade ou não do conjunto de tarefas em questão. Mesmo em um ambiente dinâmico, onde tarefas são retiradas ou acrescentadas ao conjunto de tarefas já existente, este critério mantém suas propriedades. Para um número m significativo de tarefas $\lim_{m\to\infty} U^{m}_{lub(m)} = m.(2^{1/m} - 1) \cong 0.693$, sendo que este critério pode ser mais simplificado e reduzido a uma simples comparação U <0.693. Entretanto ele não nos dá informação sobre a agendabilidade das tarefas com Us no semi-espaço complementar, isto é, superior ($U_{lub}(m) < U$) ao plano LL.

5.3.2 Acima do Plano Limite do Processador

A região R2 (1 < U) acima do plano limite do processador LP (U = 1) representa o espaço de Us para todas as combinações de C_1, C_2, …, C_m, nas quais a agendabilidade é impossível sob qualquer estratégia, pois o tempo total das tarefas supera o tempo disponível do processador no intervalo do MMC das tarefas (U >100%). Num ambiente dinâmico, onde forem retiradas tarefas e ficarmos com U <100%, deve ser feita a análise para se verificar a nova região de operação do conjunto de tarefas.

5.3.3 Acima do Plano de Liu e Layland e Abaixo do e Sobre o Plano Limite do Processador

A região R3 acima do plano de Liu e Layland e abaixo do e sobre o plano limite do processador ($U_{lub}(m) < U \leq 1$) representa o espaço de Us para todas as combinações possíveis C_1, C_2, …, C_3 que têm a sua agendabilidade indefinida pelo teorema 5 de Liu e Layland (1973). A sub-região R3a de R3 representada pelo interior do sólido formado pela **superfície de agendabilidade máxima** e o **plano limite do processador** representa o espaço das combinações não agendáveis pelo critério RMS e, pelo teorema 1 de Liu e Layland (1973), por nenhum outro critério. Por outro lado, a sub-região R3b de R3 representada pela superfície de agendabilidade máxima e o exterior do mesmo sólido, dentro desta região R3, representa o espaço das combinações agendáveis pelo critério RMS e por outros explorando ao máximo as características Ti, Ci, etc. das tarefas. O espaço geométrico das combinações agendáveis nesta sub-região R3b representa as perdas de soluções possíveis descartadas quando utilizamos o critério definido pelo teorema de Liu e Layland. Para aumentarmos a utilização do processador devemos aproveitar esta sub-região de soluções agendáveis. Para tanto faz se necessário o desenvolvimento de critérios exatos para determinação da agendabilidade, ou seja, soluções que nos levem a operar na fronteira superior desta sub-região.

5.3.3.1 Revendo o Critério de Carga de Trabalho

O critério da carga de trabalho (“workload”), apresentado por Lehoczky, Sha e Ding (1989), é um critério exato, condição necessária e suficiente, para o cálculo da agendabilidade do algoritmo de taxa constante. Desta forma podemos garantir, com um só critério, a união do espaço de agendabilidade R1 fornecido pelo Teorema de Liu e Layland com o espaço R3b de nosso interesse nesta região R3. Segundo este critério, para determinar se uma tarefa é executável sob as condições mais severas, nós necessitamos considerar a demanda do processador de um conjunto de tarefas como uma função do tempo. Considerando as tarefas τ_1, τ_2, …, τ_i, a expressão

$W_i(t) = \sum_{j=1}^{i} C_j \lceil t / T_j \rceil$, fornece a carga de trabalho acumulativa do processador para estas tarefas no intervalo $[0,t]$ quando 0 é o instante crítico. A divisão da função carga de trabalho ("workload") $W_i(t)$ pelo valor de t representa a porcentagem de utilização do processador no mesmo intervalo de tempo considerando apenas as tarefas de prioridade maior ou igual a T_i: $U_i(t) = W_i(t)/t$. A condição necessária e suficiente é que a carga de trabalho $W_i(t)$ não seja superior ao intervalo de tempo t, ou seja, a tarefa T_i é executável se existir um intervalo de tempo em que a sua utilização $U_i(t)$ seja menor ou igual a 1. A função $U_i(t)$ é monotonicamente decrescente por partes, exceto em um número finito de pontos representados pelos tempos de solicitação das tarefas T_j, que é contínuo à esquerda e salta para um valor mais alto à direita. O resultado do teorema proposto, determina que para a tarefa τ_i ser executável, basta pesquisar a demanda do processador sobre estes mínimos locais, os múltiplos de $T_j < T_i$ para $1 \le j \le i$. Especificamente, $S_i = \{k.T_j \mid j = 1,\ldots,i; k = 1,\ldots,\lfloor T_i / T_j \rfloor\}$. Os elementos do conjunto S_i são os pontos de agendabilidade ("scheduling points") para a tarefa i, ou seja, o limite para execução da tarefa i e os tempos de chegada de todas as tarefas de prioridades superiores a i, antes do limite de execução da tarefa i.

Embora este critério seja exato, do ponto de vista prático, ele equivale a uma busca exaustiva de agendabilidade para todas a tarefas $i, i = 1,\ldots,m$ em todos os pontos de solicitação de tarefas. Se $W_i(t) = 1$, todo o tempo de processamento esta ocupado com a tarefa T_i e com as demais de maior prioridade. Se $W_i(t) < 1$, existem tempos de processamento livres antes de T_i e que serão ocupados pelas tarefas de menor prioridade. Este cálculo é oneroso computacionalmente e impraticável para um conjunto com um número m muito grande de tarefas.

5.3.3.2 Revendo o Critério do Limite de Taxa Constante

O critério limite de taxa constante ("deadline-monotonic"), apresentado por Leung e Whitehead (1982), da mesma forma que o critério anterior, é um critério exato para o cálculo da agendabilidade do algoritmo de taxa constante. Segundo este critério, um conjunto de m tarefas periódicas agendadas pelo algoritmo de taxa constante irão cumprir suas restrições de tempo para todas suas tarefas, em todos períodos, se $\forall i, 1 \le i \le m$, $\frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{(C_i + B_i + E_i)}{T_i} \le i.(2^{1/i} - 1)$, onde B_i é o intervalo de tempo que a tarefa τ_i foi suspensa por uma tarefa de maior prioridade e E_i representa o intervalo de tempo que o limite de execução de τ_i foi antecipado por uma tarefa de maior prioridade. O teorema proposto solicita o cálculo dos valores de B_i e E_i para cada tarefa τ_i, $\forall i, 1 \le i \le m$.

Embora este critério seja exato, do ponto de vista prático, ele transfere o problema para o cálculo dos valores de B_i e E_i para cada tarefa τ_i. O que parece simples à primeira vista esconde uma quantidade e complexidade de cálculos bastante grande como será visto no critério abaixo proposto.

5.3.3.3 Novo Critério do Final Parcial

Como visto, o agendamento de um conjunto de tarefas periódicas existe se existir o agendamento de cada tarefa, considerando as tarefas de maior prioridade que a mesma tarefa. Na prática a questão fundamental é, para cada tarefa τ_i, descobrir quantas unidades de tempo do processador estão livres no período T_i, a partir do instante crítico, considerando somente as tarefas de prioridades maiores que τ_i, e comparar com o tempo de execução C_i desta tarefa. Se o tempo de execução C_i da tarefa for inferior ao número encontrado, é possível o agendamento para a tarefa *i*, caso contrário, não existe este agendamento e a execução da tarefa não cumprirá os seus requisitos de tempo. Faz se necessário então o desenvolvimento uma forma de cálculo deste tempo.

Sejam $\tau_i(C_i, T_i)$ e $\tau_k(C_k, T_k)$, $1 \le i < k \le m$, duas tarefas pertencentes ao conjunto de m tarefas τ_1, τ_2, ..., τ_m com as mesmas propriedades utilizadas no teorema de Liu e Layland. Para calcularmos quanto tempo do processador é disponível para a tarefa τ_k em relação à tarefa τ_i no período crítico T_k, devemos considerar dois casos.

Caso 1: todas as solicitações de τ_i que acontecem na zona crítica T_k são completadas antes da segunda solicitação de τ_k - FIGURA 5.2. Desta forma o tempo disponível para execução de τ_k em relação a τ_i é dado por $C_k^i = T_k - C_i . \lceil T_k / T_i \rceil$.

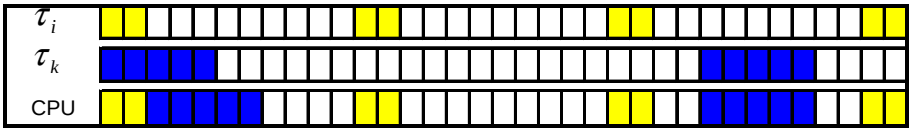

FIGURA 5.2 - Solicitações de τ_i acontecem na zona crítica T_k.

Caso 2: a execução da $\lceil T_k / T_i \rceil^{ésima}$ solicitação para τ_i na zona crítica T_k é completada após a segunda solicitação de τ_k - FIGURA 5.3. O resto da tarefa τ_i que for executado após T_k, já descontado no segundo termo do cálculo, deve ser reconsiderado (terceiro termo do cálculo). Desta forma o tempo disponível para execução de τ_k em relação a τ_i é dado por $C_i^k = T_k - C_i . \lceil T_k / T_i \rceil + R_i^k$, onde $R_i^k = C_i - (T_k - \lfloor T_k / T_i \rfloor . T_i)$, **resto de tempo da tarefa *i* em relação à tarefa *k***, é o resto da tarefa τ_i executado após T_k, quando considerado somente as duas tarefas.

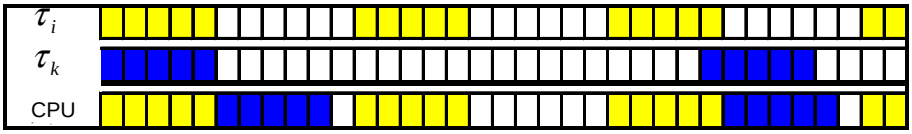

FIGURA 5.3 - Solicitação para τ_i na zona crítica T_k é completada após a segunda solicitação de τ_k.

O resultado do caso 2 pode ser generalizado para ambos os casos se considerarmos somente os **valores positivos** de R_i^k. Assim, ao calcularmos R_i^k, já teremos efetuado automaticamente o teste de interferência da tarefa. τ_k pela tarefa τ_i.

Como R_i^k é dado pela interseção de $C_i^{ésimo}$ com C_k^2 e $T_i < T_k$, temos, $0 \le R_i^k < \min\{C_i^{ésimo}, C_k^2\}$. Para sistemas com muitas tarefas, necessariamente devemos ter os $C_i << T_i$ para $1 \le i \le m$, caso contrário o sistema já nasce com dificuldade de agendamento e devemos aumentar a capacidade computacional como requisito inicial de projeto. R_i^k é mais desprezível à medida que T_k aumenta e, tanto mais, quanto maior for a razão $\lceil T_k / T_i \rceil$.

Analisemos o efeito causado pelo acréscimo de uma tarefa $\tau_j(C_j, T_j)$ com $i < j < k$ ao nosso exemplo. Os casos 1 e 2 anteriores serão distribuídos nos seguintes casos.

Caso 1a: todas as solicitações de τ_j que acontecem na zona crítica T_k são completadas antes da segunda solicitação de τ_k e não coincidem com a execução de $C_i^{ésimo}$ - FIGURA 5.4. Desta forma o tempo disponível para execução de τ_k em relação a τ_i e τ_j é dado por $C_k^{i,j} = T_k - C_i.\lceil T_k / T_i \rceil - C_j.\lceil T_k / T_j \rceil$.

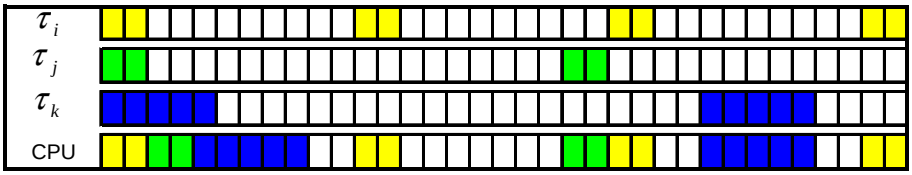

FIGURA 5.4 – Caso 1a

Caso 1b: a execução da $\lceil T_k / T_j \rceil^{ésima}$ solicitação para τ_j na zona crítica T_k é completada após a segunda solicitação de τ_k, mas não tem parte de sua execução sustada pela execução de $C_i^{ésima}$ - FIGURA 5.5. O resto da tarefa τ_j que for executado após T_k, já descontado no segundo termo do cálculo, deve ser reconsiderado no caso anterior de maneira similar à do *caso 2*. Assim, o tempo disponível para execução de τ_k em relação

a τ_i e τ_j é dado por $C_k^{i,j} = T_k - C_i.\lceil T_k / T_i \rceil - C_j.\lceil T_k / T_j \rceil + R_j^k$, onde $R_j^k = C_j - \left(T_k - \lfloor T_k / T_j \rfloor T_j\right)$.

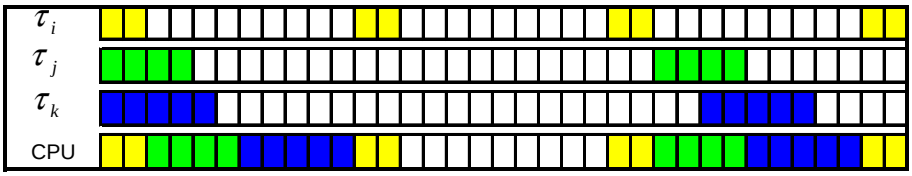

FIGURA 5.5 – Caso 1b.

Caso 1c: todas as solicitações de τ_j que acontecem na zona crítica T_k são completadas antes da segunda solicitação de τ_k, e parte de sua execução é sustada pela execução de $C_i^{ésima}$, mas não o suficiente para sair da zona crítica T_k - FIGURA 5.6. Desta forma o tempo disponível para execução de τ_k em relação a τ_i e τ_j é dado pela mesma relação $C_k^{i,j}$ do caso *1a*.

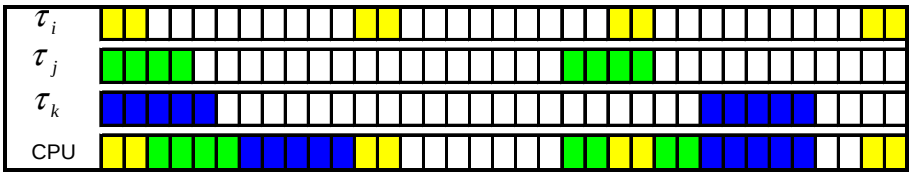

FIGURA 5.6 – Caso 1c.

Caso 1d: a execução da $\lceil T_k / T_j \rceil^{ésima}$ solicitação para τ_j na zona crítica T_k é completada após a segunda solicitação de τ_k, e ainda tem parte de sua execução sustada pela execução de $C_i^{ésima}$ - FIGURA 5.7. Além de considerar o resto R_j^k como no caso 1b, deve-se considerar a **sustação da execução de** $C_j^{ésima}$ **pela execução de** $C_i^{ésima}$ **em relação ao instante crítico** T_k, cuja notação definimos por $S_j^{i(k)}$. Assim, o tempo disponível para execução de τ_k em relação a τ_i e τ_j é dado por $C_k^{i,j} = T_k - C_i.\lceil T_k / T_i \rceil - C_j.\lceil T_k / T_j \rceil + R_j^k + S_j^{i(k)}$, onde $R_j^k = C_j - \left(T_k - \lfloor T_k / T_j \rfloor T_j\right)$ e $S_j^{i(k)} = \lfloor T_k / T_i \rfloor T_i + C_i - \lfloor T_k / T_j \rfloor T_j$, $S_j^{i(k)} \in [0, C_i]$.

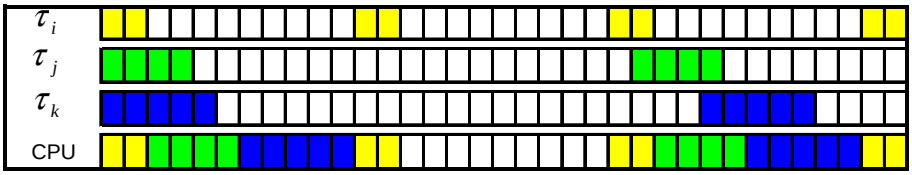

FIGURA 5.7 – Caso 1d.

Caso 2a: todas as solicitações de τ_j que acontecem na zona crítica T_k são completadas antes da segunda solicitação de τ_k e não coincidem com a execução de $C_i^{ésima}$ - FIGURA 5.8. Desta forma o tempo disponível para execução de τ_k em relação a τ_i e τ_j é dado por $C_i^k = T_k - C_i.\lceil T_k / T_i \rceil + R_i^k - C_j.\lceil T_k / T_j \rceil$.

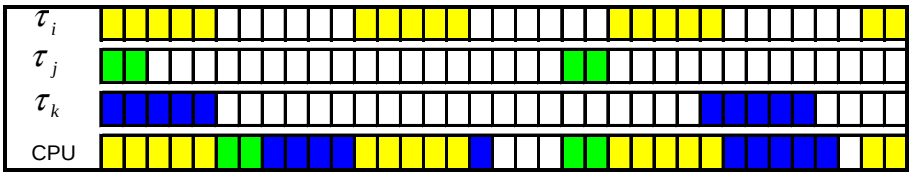

FIGURA 5.8 - Caso 2a.

Caso 2b: a execução da $\lceil T_k / T_j \rceil^{ésima}$ solicitação para τ_j na zona crítica T_k seria completada antes da segunda solicitação de τ_k, mas devido a sustação pela execução de $C_i^{ésima}$, sua execução é completada após - FIGURA 5.9. O tempo disponível para execução de τ_k em relação a τ_i e τ_j é dado por $C_k^{i,j} = T_k - C_i.\lceil T_k / T_i \rceil - C_j.\lceil T_k / T_j \rceil + R_i^k + S_j^{i(k)}$ onde $0 < S_j^{i(k)} < \min\{C_i - R_i^k, C_j\}$ é:

$$S_j^{i(k)} = \lfloor T_k / T_i \rfloor T_i + C_i - \lfloor T_k / T_j \rfloor T_j, \text{ se } \lfloor T_k / T_i \rfloor T_i < \lfloor T_k / T_j \rfloor T_j, \text{ ou}$$

$$S_j^{i(k)} = \lfloor T_k / T_j \rfloor T_j + C_j - \lfloor T_k / T_i \rfloor T_i, \text{ se } \lfloor T_k / T_i \rfloor T_i > \lfloor T_k / T_j \rfloor T_j.$$

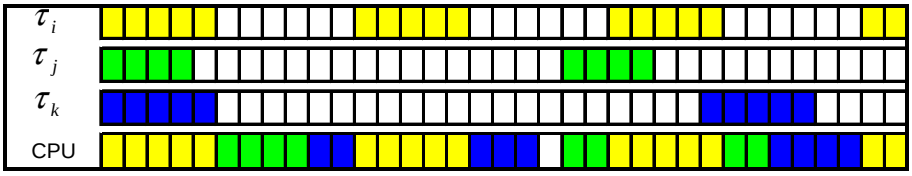

FIGURA 5.9 – Caso 2b.

Caso 2c: a execução da $\lceil T_k / T_j \rceil^{ésima}$ solicitação para τ_j na zona crítica T_k ultrapassa a segunda solicitação de τ_k e, além disso, sofre a sustação pela execução de $C_i^{ésima}$. O tempo disponível para execução de τ_k em relação a τ_i e τ_j é dado por $C_k^{i,j} = T_k - C_i.\lceil T_k / T_i \rceil - C_j.\lceil T_k / T_j \rceil + R_i^k + R_j^k + S_j^{i(k)}$. Mas no cálculo de $S_j^{i(k)}$ devemos compensar o resíduo R_i^k na sustação da tarefa τ_j em duas condições diferentes.

Se $\lfloor T_k / T_i \rfloor T_i < \lfloor T_k / T_j \rfloor T_j$, FIGURA 5.10, $0 < S_j^{i(k)} < \min\{(C_i - R_i^k), C_i\}$ é:

$$S_j^{i(k)} = \lfloor T_k / T_i \rfloor T_i + C_i - R_i^k - \lfloor T_k / T_j \rfloor T_j$$

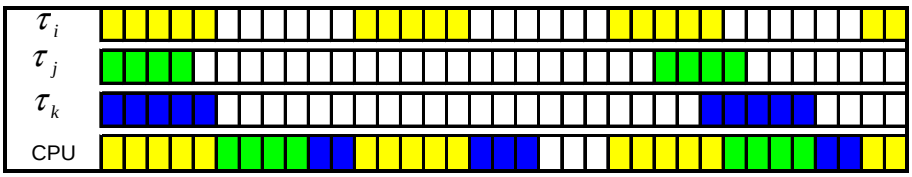

FIGURA 5.10 – Caso 2c com $\lfloor T_k / T_i \rfloor T_i < \lfloor T_k / T_j \rfloor T_j$.

Se $\lfloor T_k / T_i \rfloor T_i > \lfloor T_k / T_j \rfloor T_j$, FIGURA 5.11, $0 < S_j^{i(k)} < \min\{(C_i - R_i^k), (C_j - R_j^k)\}$ é:

$$S_j^{i(k)} = \lfloor T_k / T_j \rfloor T_j + C_j - R_j^k - \lfloor T_k / T_i \rfloor T_i$$

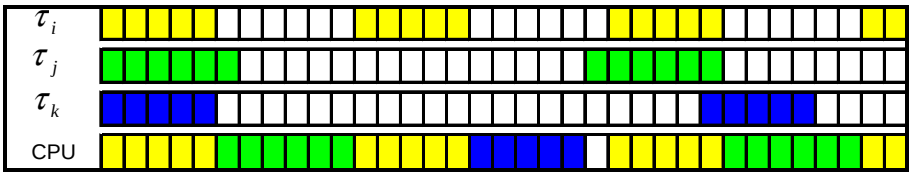

FIGURA 5.11 – Caso 2c com $\lfloor T_k / T_i \rfloor T_i > \lfloor T_k / T_j \rfloor T_j$.

Generalizando os casos expostos anteriormente, segundo a própria definição de agendabilidade, tem-se o seguinte teorema:

Teorema 1: Um conjunto de m tarefas periódicas agendadas pelo algoritmo de taxa constante irão cumprir suas restrições de tempo para todas suas tarefas, em todos períodos, se e somente se, $\forall i, 1 \le i \le m$, $C_i \le T_i - \sum_{k=1}^{k=i-1} C_k.\lceil T_i / T_k \rceil + \sum_{k=1}^{k=i-1} R_k^i + \sum_{k=1}^{k=i-1} [S_k^i]$,

onde $R_k^i = C_k - (T_i - \lfloor T_i / T_k \rfloor T_k)$ e S_k^i é a **sustação da execução de C_k^{th} pela execução de todas as tarefas de maior prioridade que k no instante crítico,** $0 < S_k^i < \min\{(C_k - R_k^i), (C_j - R_j^i\}$, $j < k$.

A expressão para cálculo de C_i máximo é complexa e contem três somatórios que representam um número muito grande de operações, contrariando um regra fundamental para agendadores: tempo pequeno de processamento em relação às tarefas envolvidas. Analisemos a importância relativa destes componentes a partir de um exemplo contendo vários conjuntos de tarefas formados por {(10,3);(T_2,4);(32,5);(55,C_4)}, onde $T_2 \in$ [22,36] é o período da segunda tarefa, e C_4 é o tempo disponível para execução da quarta tarefa, conforme FIGURA 5.12.

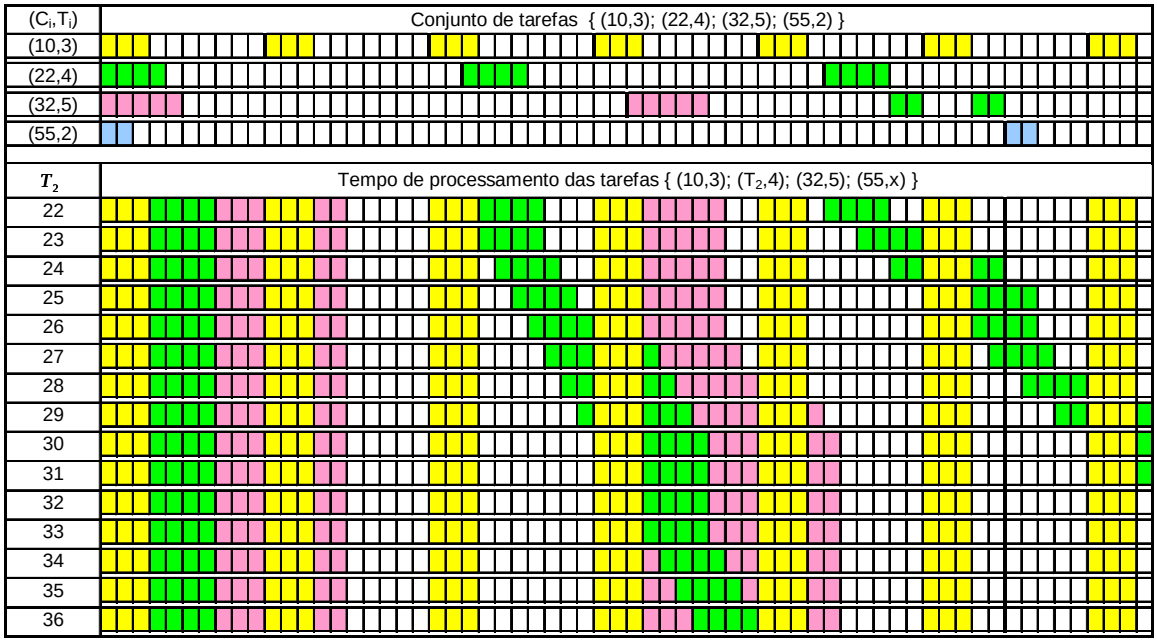

FIGURA 5.12 – Processamento das tarefas {(10,3);(T_2,4);(32,5);(55,C_4)}.

O primeiro somatório $\sum_{k=1}^{k=i-1} C_k \lceil T_i / T_k \rceil$ (i=4 no exemplo) representa o tempo gasto na execução de todas as tarefas de maior prioridade solicitadas antes de T_4 =55, desprezando eventuais partes executadas fora deste intervalo. Isso somente acontece nos casos T_2 =26 (perde 1) e T_2 =27 (perde 3). Neste caso a perda seria de 1,82% em um caso e 5,45% em outro se o objetivo fosse usar toda a capacidade disponível do

processador. Mas este não é o objetivo se tivermos mais tarefas e, portanto, aquela diferença não pode ser classificada como “perda”. Nos casos T_2 =24 e T_2 =25 a parte que ultrapassa o limite é causada pela sustação de T_1 e são recuperadas parcialmente pelo segundo somatório $\sum_{k=1}^{k=i-1} R_k^i$ e complementada pelo terceiro somatório $\sum_{k=1}^{k=i-1} \left[S_k^i \right]$. Apresentamos na FIGURA 5.13 os efeitos da variação de C_2 sobre o erro causado neste componente.

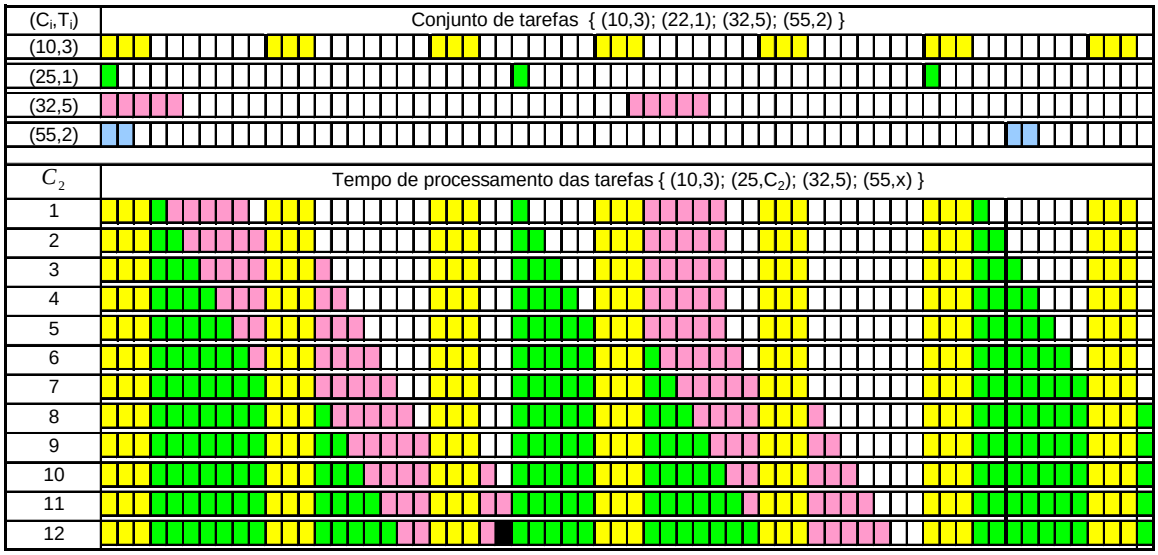

FIGURA 5.13 – Processamento das tarefas {(10,3);(25, C_2);(32,5);(55,x)}.

O efeito da variação C_2 aparece a partir de C_2 = 6, quando o erro passa existir e aumenta com o aumento de C_2. Os erros C_2 = [3,5] são efeitos da suspensão por C_1. Entretanto, como já visto anteriormente, se o exemplo for parte de um universo maior de tarefas, os tempos de cada tarefa devem ser cada vez menores à medida que o número total de tarefas cresce, sob pena de o sistema ser inviável como proposição. É exatamente o cenário dos problemas que queremos resolver: para um sistema com um número muito grande de tarefas, devemos partir de uma capacidade de processamento adequada.

O segundo somatório recupera os “erros” do primeiro se não existir suspensão em instante imediatamente anterior ao resíduo. O efeito do aumento de C_1=3 para C_1=4 pode ser verificado na FIGURA 5.14, casos T_2 =24 e T_2 =25. Os efeitos da suspensão

são ainda inferiores aos erros do primeiro somatório embora o seu cálculo seja ainda mais complexo que o do resto. Considerando ainda os mesmos argumentos para o resto, a correção dos efeitos de suspensão imediatamente anterior aos resíduos, são ainda mais desprezíveis.

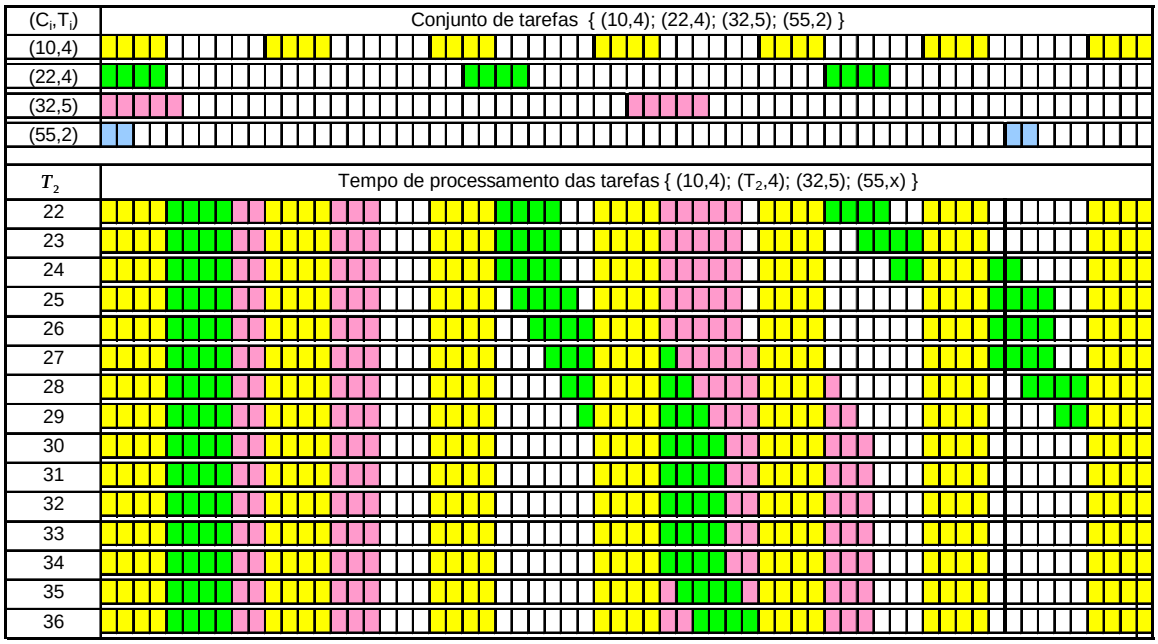

FIGURA 5.14 - Processamento das tarefas {(10,4);(T_2, 4);(32,5);(55,x)}.

Para extensão do agendador de taxa constante, introduziremos um novo critério do final parcial através do Teorema 2.

Teorema 2: Um conjunto de m tarefas periódicas agendadas pelo algoritmo de taxa constante irão cumprir suas restrições de tempo para todas suas tarefas, em todos períodos, se $\forall i, 1 \le i \le m$, $C_i \le T_i - \sum_{k=1}^{k=i-1} C_k . \lceil T_i / T_k \rceil$. Esta é uma condição suficiente.

As hipóteses que suportam a aplicação prática deste teorema são resumidas a seguir:

a) Um agendamento de taxa constante não ocupará todo o tempo de processamento, exceto em relações especiais entre todos os pares (T_i, T_j), mais especificamente, se T_m / T_i , $\{T_m / T_i\} = 0$, para $i = 1,2,...,m-1$.

b) Para sistemas com muitas tarefas, necessariamente devemos ter $C_i << T_i$.

c) Temos $R_k^i < C_i << T_i$. Os restos são pequenos em relação a Ti.

d) Temos $S_k^i < R_k^i < C_i << T_i$. As suspensões de tarefas são pequenas em relação a Ti.

e) O tempo de processamento deve estar distribuído entre todas as tarefas. Isto indica que o processador não pode ser totalmente ocupado antes da tarefa final e, portanto, o somatório dos restos não é "erro", mas sim, a parte reservada para tarefas de prioridades menores.

f) Somente relações bem definidas de (C_i, T_i) para todas as tarefas levam o sistema a operar na fronteira. Não é comum encontrarmos na Natureza sistemas que possuam estas relações.

5.4 Novas Estratégias de Agendamento

Nos problemas reais mencionados como motivação deste trabalho temos um sistema de tempo real crítico de natureza dinâmica com um conjunto de tarefas grande. Sua própria natureza indica que temos tarefas saindo, entrando, ou saindo e entrando em tal conjunto ao mesmo tempo. Portanto faz-se necessário a verificação constante de agendabilidade do atual conjunto de tarefas. Mas saber se é possível ou não agendar todas as tarefas do sistema não é o suficiente para sistemas de tempo real crítico. Temos que tomar alguma medida eficaz, caso contrário teremos um dano indesejável. Neste cenário, as estratégias possíveis são: a) reduzimos a frequência $f_i = 1/T_i$ de uma ou mais tarefas e, portanto, diminuímos suas prioridades; b) reduzimos o tempo de execução C_i de uma ou mais tarefas; ou, c) modificamos simultaneamente a frequência $f_i = 1/Ti$ e o tempo de execução C_i de uma ou mais tarefas.

A frequência de uma tarefa está umbilicalmente ligada à dinâmica natural do componente que a mesma tarefa representa ou está conectada. Desta forma, quando

tentamos alterar a frequência natural de um componente ou sistema estamos falando em mudar a própria natureza do sistema. Em alguns projetos de engenharia essa alteração é necessária e, até mesmo essencial, para a existência ou estabilidade de um sistema que desejamos construir e/ou controlar. Entretanto, para um sistema complexo, composto de muitos componentes, principalmente externos e fora de nosso alcance, como são os casos dos sistemas de tempo real motivadores deste trabalho, a alteração da frequência de um componente é propriedade inalienável do mesmo. Mais ainda, essas propriedades definem os limites em que a frequência de um sistema complexo contendo esses componentes deve operar. Como exemplo, a frequência e velocidade de manobras de hipotéticos mísseis definem o envelope de frequência que o sistema de defesa projetado deve funcionar. Devido a estas razões descartamos, por hora, as estratégias que envolvem modificação da frequência (opções **a** e **c**).

A alternativa possível para a garantia da execução de tarefas em ambientes dinâmicos é a redução do tempo de execução de tarefas (opção b). A alteração dos tempos de tarefas não significa que estamos mudando o tempo do componente ou do sistema, mas sim da tarefa relacionada a este componente ou sistema. As tarefas referem-se a cálculos de complexidade variável ou a modelos que representam de alguma forma o comportamento dos componentes. Em geral, tarefas ou modelos com maior tempo de processamento representam cálculos mais exatos e resultados mais fiéis à realidade (segundo o compromisso fundamental da Modelagem: simplicidade x fidelidade). Sistemas complexos podem manter o desempenho utilizando tarefas ou modelos com diferentes níveis de fidelidade (ou graus de realismo) em diferentes modos de operação.

A título de esclarecimento, utilizaremos como exemplo uma missão de transporte de astronautas para a estação espacial. A partir do lançamento até a entrada em órbita da espaçonave, o cálculo da órbita da estação espacial usando somente os elementos keplerianos é suficiente para o seu sistema de guiagem e controle. Na fase de aproximação, são necessários cálculos mais precisos incluindo eventuais perturbações existentes na órbita. Na fase de acoplamento é evidente que, além das informações de órbita com maior precisão, também são necessárias as informações de atitude.

Desta forma este trabalho propõe novas estratégias dinâmicas baseadas em critérios de redução de tempo de uma ou mais tarefas que garantam, de forma previsível, a agendabilidade de sistemas de tempo real em ambientes dinâmicos e complexos.

5.5 Nova Estratégia de Agendamento a Taxa Constante Dinâmico Estendido

Existem vários critérios para a verificação da existência ou a análise da agendabilidade para agendadores com capacidade de sustação de tarefas como o de taxa constante. Também existe uma grande variedade de estratégias para aproveitamento do tempo livre do processador (inerente ao agendador de taxa constante) sejam para execução de tarefas não críticas, assíncronas, ou para finalizar a execução de tarefas críticas que "não seriam" executadas nos requisitos de tempo.

O objetivo a ser perseguido neste trabalho é propor novas estratégias para agendadores de sistemas de tempo real críticos dinâmicos e, consequentemente, os agendadores devem ser basicamente dinâmicos. Utilizando o agendador de taxa constante como referência propomos uma estratégia para agendamento de tarefas.

Uma nova estratégia proposta é:

a) A fila de tarefas deve ser montada a partir das mesmas regras do agendador de taxa constante;

b) A existência ou não do agendador será verificada sempre que houver uma alteração dinâmica do sistema (alteração do número de tarefas);

c) Essa verificação é efetuada utilizando-se o critério de Liu e Layland $U \leq m.(2^{1/m} - 1)$.

d) Em caso negativo, será aplicado o novo critério de verificação $C_i \leq C_{\max(i)}$, $C_{\max(i)} = T_i - \sum_{k=1}^{k=i-1} C_k . \lceil T_i / T_k \rceil$ para $\forall i, 1 \leq i \leq m$ com a imediata redução de C_k, $k = 1,2,..i$ conforme critérios propostos a seguir neste capítulo.

e) Nova verificação de agendabilidade. Caso negativo, prosseguir com a redução C_k. Caso positivo, retornar ao item d para o próximo valor de *i*.

f) Enviar novo conjunto de tarefas para agendamento (item a).

A aplicação do critério do item *"d"* proporciona a capacidade do sistema reduzir o tempo de execução só de uma tarefa ou subconjunto de tarefas que esteja inviabilizando todo o sistema ao invés de executarmos o algoritmo de redução para todas as tarefas.

5.5.1 Critério de Redução Uniforme dos Tempos de Execução

O critério mais geral de redução do tempo de execução disponível é fazer uma distribuição linear uniforme para todas as tarefas considerando a mesma importância para todas. Neste caso, isto significa fazer uma distribuição igual no **tempo mínimo de execução** considerando o tempo total de execução de cada tarefa no intervalo de tempo correspondente ao MMC, dado por $M = T_1.T_2 \ldots T_m$ para períodos primos e o fator de disponibilidade. O **tempo mínimo de execução disponível** é $t_{\min(m)} = M.m.(2^{1/m} - 1)$.

Para $k = 1,2,..i$, o critério de redução uniforme tem os seguinte passos:

d1) O novo valor de cada tarefa, $C_k^{'}$, é $C_k^{'} = \min\{C_k, \frac{t_{\min(m)}}{m}.\frac{T_k}{T_m}\}$.

d2) O tempo disponível para execução pode ser atualizado para $t_{\min(m)}(k+1) = t_{\min(m)}(k) + (C_k - C_k^{'}).\frac{T_m}{T_k}$.

Este é um critério de fácil aplicação mas apresenta a grande desvantagem de "exigir" que o tempo de execução das tarefas, sem distinção, tenha que se adaptar à condição desejada. Isso significa torcer a Natureza para que ela se enquadre em nossas necessidades de projeto e é, geralmente, irrealizável. A redução do tempo de execução de cada tarefa tem que respeitar os restrições e limites da parte da Natureza que representam.

5.5.2 Novos Critérios Para Redução dos Tempos de Execução de Tarefas

O cérebro humano processa tarefas, internas e externas ao corpo humano, com um grande e indefinível número de níveis que estão relacionados com a importância de cada tarefa no presente estado. Tarefas podem surgir, desaparecer, aumentar ou diminuir de importância, mas nenhuma que seja realmente relevante deixa de ser processada. Sob condições de pressão, o cérebro humano agenda tarefas considerando um número complexo de critérios: tempo de início, tempo de duração, tempo final, prazos a cumprir, custos de cumprimento, custos de não cumprimento ou de cumprimento parcial, impactos no meio ambiente, etc. No caso extremo de falta de recursos, restando somente tarefas vitais, ainda assim ele "desliga" tarefas segundo o critério da sobrevivência. Sendo esta capacidade o resultado de um processo evolutivo de milhões de anos, é também um forte indicativo de que essa característica é importante em sistemas de tempo real críticos complexos.

Devido às restrições e limites impostos aos sistemas pela Natureza, adicionados aos recursos matemáticos e computacionais disponíveis para processamento das tarefas, essas tarefas podem ser classificadas em dois grupos: as que permitem e as que não permitem alterações. Isso sugere a criação de pelo menos dois níveis de tarefas relacionados diretamente com o consumo do recurso disputado.

Os agendadores de computação imprecisa operam utilizando este princípio básico e suas tarefas possuem uma parte obrigatória e outra opcional que deixaria de ser executada na falta de tempo. Estes conceitos foram modificados em algumas áreas para a existência de duas tarefas: uma tarefa básica para a execução normal e outra, alternativa, com menor tempo de processamento, para execução em instantes de sobrecarga do sistema. A primeira alternativa é limitada pela capacidade ou existência de cálculos internos à tarefa com precisão diretamente relacionada ao tempo de cálculo. A segunda, não permite a análise nos cenários de condições básicas presentes no desenvolvimento de sistemas complexos: ruim, normal e bom.

Propomos que todas as tarefas de sistemas de tempo real críticos complexos possuam, por hora, **três níveis de fidelidade**, com ordens de grandeza diferentes, diretamente

relacionados com os recursos disputados. Em sistemas embarcados esse recurso tipicamente é o tempo de processamento, mas em sistemas distribuídos a capacidade de tráfego de informações deve ser considerada. A ordem de grandeza não é obrigatória, mas quando gastamos tempo para reduzir tempo de uma tarefa, o tempo gasto tem que ser insignificante em relação ao tempo reduzido, e esse tempo reduzido têm de ser significativo em relação à própria tarefa. Do ponto de vista prático, as escalas baseadas em ordem de grandeza 1, 10 e 100 representam três níveis de fidelidade importantes para um sistema complexo: a redução da tarefa em um nível, permite a execução de dez vezes a mesma tarefa em um nível de fidelidade abaixo.

Formalmente, cada tarefa $\tau_i\,(C_i, T_i)$ passa a ser representada por $\tau_i\,(C_{i(1)}, C_{i(2)}, C_{i(3)}, T_i)$, onde $C_{i(n)}$, $n = 1,2,3$, representa o tempo de processamento da tarefa *i* para o grau de fidelidade *n*.

A responsabilidade da redução de tempo de processamento é transferida para o critério de definição dos níveis de fidelidade. Para a natureza dos problemas que motivam este trabalho, são propostos os seguintes critérios:

Distância – As tarefas relativas a componentes estáticos ou serviços devem ser classificadas como de fidelidade 1. As tarefas relativas a componentes dinâmicos (ex. federados em uma arquitetura HLA) devem ser classificadas em três níveis de fidelidade relativos às distâncias “perto”, “médio”, e “longe” segundo um referencial e parâmetros de projeto. A referência para um sistema de controle de um aeroporto é a torre de controle e os parâmetros de distância definidos conforme as regras de operação do mesmo. Enquanto para uma aeronave de combate, a própria nave é a referência e os parâmetros de distância são definidos de acordo com o alcance de seus radares.

Aplicando a estratégia proposta na seção 5.5, item **d**, reduzindo os tempos de execução das tarefas segundo o critério no nível de fidelidade relacionado com a distância, são propostos os seguintes passos:

d1) Para $j = 1,2,..i$, toda tarefa de nível de fidelidade 1 deverá ser alterada para nível de fidelidade 2, ou seja, toda tarefa com $C_k = C_{k(1)}$, deve ser alterada para $C_k = C_{k(2)}$.; e atualizar valor de $C_{\max(i)}$.

d2) Nova verificação de agendabilidade com o critério rápido $C_i \leq C_{\max(i)}$. Caso positivo, avançar no critério para o próximo i, em caso negativo, aplicar o mesmo procedimento de redução do nível de fidelidade nas tarefas de nível 2 para 3.

d3) Nova verificação de agendabilidade. Em caso negativo, e todas as tarefas já estiverem no nível de fidelidade 3, entrar no modo de sobrevivência.

Este critério permite a redução do tempo de execução de cada tarefa respeitando as restrições e limites da parte da natureza que representam, mas é mais complexo que o anterior e acrescenta um tempo de processamento de difícil determinação embora seja limitado.

Aproximação – Semelhante ao anterior, trocando as distâncias pela sua derivada, ou seja, velocidade de aproximação (positivo) ou afastamento (negativo) segundo um referencial e parâmetros de projeto. Aplicando a estratégia proposta na seção 5.5, item ***d***, reduzindo os tempos de execução das tarefas segundo o critério no nível de fidelidade relacionado com a aproximação devem ser aplicados os mesmos passos ***d1*** a ***d4*** do critério de distância.

Evidentemente, pode-se fazer construir um critério combinando o da distância e de aproximação com pesos para cada componente definidos em cada projeto. Neste caso os mesmos passos de redução de tempo das tarefas devem ser executados.

Letalidade – Os níveis de fidelidade são definidos conforme o potencial de danos que a tarefa ou componente associado podem causar no sistema. Existem na literatura, para a área de defesa, classificações de letalidade aplicadas a aeronaves, mísseis, bombas, etc, assim como níveis de eficácia de contra-medidas. Essa classificação depende da doutrina de cada país e está incorporada aos seus sistemas de defesa.

Aplicando a estratégia proposta em 5.5, item ***d***, reduzindo os tempos de execução das tarefas segundo o critério no nível de fidelidade relacionado com a letalidade devem ser aplicados os mesmos passos ***d1*** a ***d4*** dos critérios de distância e aproximação.

Todos os três critérios de redução do tempo de tarefas (distância, aproximação, letalidade) permitem uma grande flexibilidade para garantia do agendamento das tarefas até o limite dos recursos disputados. Isso é uma característica importantíssima em sistemas de defesa, sistemas embarcados para atuação em ambientes complexos e dinâmicos. Por outro lado, causam aumento do tempo gasto em processamento e acrescentam a incerteza do tempo gasto no próprio cálculo do agendamento. Essa incerteza pode e deve ser contornada nas formas de implementação do mesmo.

5.5.3 Problemas da Estratégia de Agendamento a Taxa Constante Dinâmico Estendido

Estes algoritmos são adequados em ambientes em que a perda de uma tarefa exigirá um tempo de recuperação muito maior do que se mantivéssemos um resultado anterior ainda que degradado. Estes algoritmos podem ser implementados com técnicas de processamento off-line e on-line.

Entre os problemas dos agendadores a taxa constante dinâmico estendido, nas diferentes formas apresentadas até o presente momento, destaca se: a) o número de alternâncias entre tarefas multiplica-se à medida que aumenta o número de tarefas em sistemas complexos; b) o critério de agendabilidade proposto, embora seja melhor e determinado em função do número de tarefas, usa um algoritmo com grande número de operações matemáticas; e, c) o ganho de flexibilidade por redução do tempo de execução das tarefas é deteriorado pelo tempo e incertezas adicionadas pelo próprio mecanismo de redução de tempo. Estes problemas são causados fundamentalmente pelas partes fracionais de T_m / T_i , $i = 1,2,...,m-1$, e pela falta de verificação instantânea de agendabilidade a cada alteração da duração de uma tarefa. Os passos seguintes buscam alternativas para solução destes problemas.

5.6 Nova Estratégia de Agendamento com Sintonia de (T_i, T_j)

A estratégia proposta de agendamento a taxa constante dinâmico, nas suas diferentes formas, tem o seu critério de agendabilidade dado pela forma arredondada da expressão:

$$C_i \leq T_i - \sum_{k=1}^{k=i-1} C_k . \lceil T_i / T_k \rceil + \sum_{k=1}^{k=i-1} R_k^i + \sum_{k=1}^{k=i-1} \lfloor S_k^i \rfloor .$$

Esse critério pode ter sua resolução aumentada anulando as contribuições dos restos e/ou sustações; e a sua complexidade de cálculo diminuída eliminando as partes fracionais do cálculo do primeiro somatório. Isto é feito escolhendo $T_1 . T_2 \ldots . T_m$ com boas relações entre $\{T_k, T_i\}$ tal que as partes fracionais $T_k / T_i = 0$, onde $k = 1,2,...,i-1$ para todo $i = 2,3,...,m-1$. Isto é equivalente a escolhermos uma série de períodos como uma seqüência exponencial de base ***n*** e multiplicador T_1 dada por $T_i = T_1 . . n^{(i)}$. Se $n = 2$, temos um espectro de freqüências das tarefas onde a próxima tarefa sempre tem a metade da freqüência anterior, o que é bastante representativo e utilizado no em sistemas embarcados de aeronaves militares.

O critério de agendabilidade fica reduzido a $C_i \leq T_i - \sum_{k=1}^{k=i-1} C_k . n^{(i-k)}$ e o fator de utilização a $U\ (i) \leq 1$.

Portanto a nova estratégia proposta pode ser melhorada da seguinte forma:

a) A fila de tarefas deve ser montada a partir das mesmas regras do agendador de taxa constante;

b) A existência ou não do agendador será verificada sempre que houver uma alteração dinâmica do sistema (alteração do número de tarefas);

c) Será aplicado o novo critério de verificação $C_i \leq T_i - \sum_{k=1}^{k=i-1} C_k . n^{(i-k)}$ para $\forall i, 1 \leq i \leq m$ e, em caso negativo, será feita a imediata redução de C_k, $k = 1,2,..i$.

A redução do tempo de execução das tarefas C_k segundo os critérios no nível de fidelidade relacionados com a distância, ou com a aproximação ou com a letalidade é feita nos seguintes passos:

c1) Para C_k, $k = 1,2,..i$, toda tarefa de nível de fidelidade 1 deverá ser alterado para nível de fidelidade 2, ou seja, toda tarefa com $C_k = C_{k(1)}$, deve ser alterada para $C_k = C_{k(2)}$.

c2) Nova verificação de agendabilidade. Em caso negativo, para os casos possíveis, reduzir o nível de fidelidade das tarefas de nível 2 para 3, ou seja, toda tarefa com $C_k = C_{k(2)}$, deve ser alterada para $C_k = C_{k(3)}$.

c3) Nova verificação de agendabilidade. Em caso negativo da nova verificação, entrar no modo de sobrevivência.

Esta estratégia transforma o critério de agendabilidade anteriormente proposto em exato e provoca uma redução muito grande no seu número de operações matemáticas. Não resolve ou diminui a alternância entre tarefas característica do RMS e mantém as incertezas adicionadas pelo próprio mecanismo de redução de tempo. Por outro lado acrescenta restrições na escolha dos períodos das tarefas que, eventualmente, representem uma degradação ou até mesmo não possam ser aceitas. Embora o cálculo do fator de utilização tenha sido simplificado somente para uma comparação numérica, essa vantagem não foi convertida em ganho na estratégia proposta.

5.7 Nova Estratégia de Agendamento Com Primeiro Prazo Fatal Primeiro Estendido

Os Agendadores com Primeiro Prazo Fatal Primeiro - APPFP (“Earliest Deadline First Scheduler” - EDFS), onde a prioridade de execução das tarefas é definida pelo momento em que cada tarefa irá terminar, foram apresentados no item 2.5.3.4.

Seu critério de agendabilidade necessário e suficiente foi apresentado no item 4.5 e é:

$$(C_1 / T_1) + (C_2 / T_2) + ... + (C_m / T_m) \leq 1.$$

Utilizando o mínimo múltiplo comum com $M = T_1.T_2 \ldots T_m$ e $n_i = M / T_i$, podemos reescrever como $\{\sum_{i=1}^{m} (n_i.C_i)\} \leq M$.

Além de ser necessário e suficiente, este critério elimina uma fonte de incertezas adicionadas pelo próprio mecanismo de redução de tempo na medida em que ele permite a imediata verificação da agendabilidade a cada alteração de C_i para $\forall i, 1 \leq i \leq m$. Assim podemos garantir a agendabilidade em condições de sobrecarga usando o máximo tempo possível. O **tempo de processamento do agendador** (t_{pa}) pode ser dividido em uma **parte fixa constante** (t_{pak}) e outra relativa à **alteração do tempo de execução da tarefa** (t_{pat}) ser deduzido, e deduzido do tempo da tarefa quando definimos o **tempo solicitado para CPU** como $T_{cpu} = \sum_{i=1}^{m} (n_i.C_i)$.

A nova estratégia proposta é estender a aplicação do FDF com redução do tempo de execução de tarefas segundo os mesmos critérios de nível de fidelidade relacionados com a distância, aproximação ou letalidade, utilizados no agendador de taxa constante estendido.

Essa outra nova estratégia proposta é:

a) A fila de tarefas deve ser montada considerando o término das tarefas e, aquela cujo final está mais próximo, será executada primeiro;

b) A existência ou não do agendador será verificada sempre que houver uma alteração dinâmica do sistema (alteração do número de tarefas);

c) Essa verificação é efetuada utilizando-se o critério $T_{cpu} \leq M - t_{pak}$ e, em caso verdadeiro, agendar as tarefas, caso falso, continuar a tarefa de agendamento corrente.

d) Cada tarefa de nível de fidelidade 2 deverá ser alterada para nível de fidelidade 3 e seu efeito computado em T_{cpu}, $T_{cpu} = T_{cpu} - C_{i(2)} + C_{i(3)}$ e $M = M - t_{pat}$. Se o novo $T_{cpu} \leq M$, agendar as tarefas.

e) Cada tarefa de nível de fidelidade 1 deverá ser alterada para nível de fidelidade 2 e seu efeito computado em T_{cpu}, $T_{cpu} = T_{cpu} - C_{i(1)} + C_{i(2)}$ e $M = M - t_{pat}$. Se o novo $T_{cpu} \leq M$, agendar tarefas.

f) Cada tarefa de nível de fidelidade 2 deverá ser alterada para nível de fidelidade 3 e seu efeito computado em T_{cpu}, $T_{cpu} = T_{cpu} - C_{i(2)} + C_{i(3)}$ e $M = M - t_{pat}$. Se o novo $T_{cpu} \leq M$, agendar tarefas.

g) O Agendador entra em modo de sobrevivência e descobre o agendamento possível para aquela situação. Para isso armazena em $T_{cpu} = 0$, a soma do tempo de cada tarefa, comparando com o tempo disponível M a cada soma, a partir das tarefas com prioridades originais 1, 2 e 3. Assim que $T_{cpu} > M$, agenda as tarefas do cenário imediatamente anterior.

O agendador com primeiro prazo fatal primeiro estendido, baseado nesta nova estratégia, apresenta grandes vantagens em relação aos anteriores. Ele usa um critério de agendabilidade exata, provoca uma redução muito grande no seu número de operações matemáticas, reduz o número de suspensões e reentrâncias de tarefas em relação ao RMS estendido e compensa as incertezas adicionadas pelo próprio mecanismo de redução de tempo.

Entre as principais desvantagens temos: a) para permitir o uso máximo do processador ele empurra o não cumprimento de tarefas para o primeiro tempo disponível no futuro e, cria uma variação no tempo entre a execução das tarefas; b) o algoritmo básico já exige uma implementação com mecanismos mais complexos que aumentam quando considerados os tempos envolvidos no chaveamento.

5.8 Nova Estratégia de Agendamento Dinâmico de Sistemas de Tempo Real Críticos

Os sistemas de tempo real críticos complexos conhecidos são sistemas de controle ou sistemas controlados. O fator crítico nasce da necessidade de realimentação ou uso de uma informação para um processamento de algum tipo de controle. Embora possamos ter componentes internos ou externos de qualquer natureza, tais como sensores, atuadores, ameaças, defesas, etc, operando em diferentes freqüências, o nosso sistema e seus objetivos necessitam responder a uma frequência fundamental mínima sob pena de não fazer sentido a sua existência. Um sistema de defesa só tem sentido de existir se for capaz de responder mais rápido que as ameaças para o qual é projetado. Um sistema de controle “fly-by-wire” de uma aeronave deve ser capaz de responder em uma freqüência suficiente para se manter o controle da aeronave. Só faz sentido colocarmos um sistema automático de desvio de meteoritos em um ônibus espacial para viagens interplanetárias se o mesmo for capaz de comandar a espaçonave em tempo hábil. O sistema de controle de tráfego aéreo em um aeroporto deve ser capaz de responder em uma frequência mantendo a controlabilidade do sistema. Em resumo, um sistema de tempo real crítico sempre terá uma frequência fundamental em que devemos processar a informação. Outra característica desses sistemas é que também possuem tarefas assíncronas que devam ser atendidas dentro de restrições de tempo.

Em geral, para os sistemas embarcados (ex. aviões de combate e defesa aérea, ônibus espaciais), o recurso mais disputado é tempo de processamento e, eventualmente, o tempo disponível de barramento para enviar e receber mensagens. Para sistemas complexos distribuídos, o recurso mais disputado passa a ser tempo de barramento para comunicação. As estratégias de agendamento propostas neste trabalho podem ser aplicadas a ambos tipos de recursos. Dada a importância destes sistemas ainda é possível adaptar e melhorar as estratégias propostas considerando suas características fundamentais.

Como diretrizes básicas de projeto de um sistema de tempo real complexo, deve-se:

- Selecionar uma frequência fundamental adequada: um sistema de tempo real crítico apresenta uma frequência mínima de operação. Devemos selecionar uma frequência de operação superior que atenda aos requisitos de boas relações entre os períodos das tarefas do sistema.

- Selecionar freqüências de operação de componentes que respeitem essas relações. Isso significa particularizar os períodos das tarefas e fixar o MMC como a frequência fundamental de interesse do sistema.

- Reservar uma percentagem de processamento para cumprimento das tarefas assíncronas suficiente para aplicarmos o critério simplificado do RMS.

- Projetar o ponto de operação do sistema com uma reserva técnica adequada de recursos conforme o ambiente dinâmico que se que operar.

Assim a estratégia proposta para esse tipo de sistema, materializada em algoritmo no próximo capítulo, é:

a) Partir do tempo de discretização adequado.

b) A fila de tarefas deve ser montada a partir das mesmas regras do agendador de taxa constante;

c) A existência ou não do agendador será verificada sempre que houver uma alteração dinâmica do sistema (alteração do número de tarefas);

d) Essa verificação é efetuada utilizando-se o critério $U \leq 69{,}3\% - (t_{pa} / M)$, onde $t_{pa} = t_{pak}$; e, em caso verdadeiro, agendar as tarefas, caso falso, continuar tarefa de agendamento.

e) Cada tarefa de nível de fidelidade 2 deverá ser alterada para nível de fidelidade 3 e seu efeito computado em U, $U = U - \left(C_{i(2)} - C_{i(3)}\right)/M$ e $t_{pa} = t_{pa} + t_{pat}$. Se o novo $U \leq 69{,}3\% - (t_{pa} / M)$, agendar tarefas.

f) Cada tarefa de nível de fidelidade 1 deverá ser alterada para nível de fidelidade 2 e seu efeito computado em U, $U = U - \left(C_{i(1)} - C_{i(2)}\right)/M$ e $t_{pa} = t_{pa} + t_{pat}$. Se o novo $U \leq 69{,}3\% - (t_{pa}/M)$, agendar tarefas.

g) Cada tarefa de nível de fidelidade 2 deverá ser alterada para nível de fidelidade 3 e seu efeito computado em U, $U = U - \left(C_{i(2)} - C_{i(3)}\right)/M$ e $t_{pa} = t_{pa} + t_{pat}$. Se o novo $U \leq 69{,}3\% - (t_{pa}/M)$, agendar tarefas.

h) Agendador entra em modo de sobrevivência e descobre o agendamento possível para aquela situação. Para isso calcula $U = \sum_i (C_i/M)$ para cada tarefa, atualizando $t_{pa} = t_{pa} + t_{pat}$ e comparando com $U \leq 69{,}3\% - (t_{pa}/M)$ a cada soma, a partir das tarefas com prioridades originais 1, 2 e 3. Assim que a comparação anterior for falsa, agenda as tarefas do cenário imediatamente anterior.

As vantagens da aplicação desta estratégia ficam evidentes quando a mesma é aplicada em ambientes complexos de arquitetura distribuída como HLA, conforme nossa proposta de melhoria da mesma arquitetura feita no Capítulo 7.

CAPÍTULO 6

NOVA FAMILIA DE ALGORITMOS AGENDADORES/ ESCALONADORES CONSIDERANDO NÍVEIS DE FIDELIDADE AJUSTÁVEIS

6.1 Arquitetura Proposta Para o Programa de Tempo Real

Uma arquitetura simples e robusta para programas de sistemas de tempo real foi apresentada no Capítulo 3. Para as aplicações críticas sugeridas neste trabalho é fundamental a reserva de tempo destinado às tarefas assíncronas e também às tarefas relacionadas com o controle e gerenciamento dos níveis e fidelidade em um ambiente complexo. As tarefas assíncronas serão implementadas conforme a necessidade e de acordo com as políticas definidas no Capítulo 2: agendador o mais cedo possível, agendador depois de um tempo definido ou agendador após o próximo gatilho. O módulo específico para gerenciamento de níveis de fidelidade tem como função principal registrar os **níveis de fidelidade solicitados**, registrar os **níveis de fidelidade implementados**, incorporar uma nova tarefa à lista previamente agendada e, assim que houver uma redução de tarefas, restaurar o nível de fidelidade solicitado na lista de tarefas enviada ao agendador para processamento.

Sugerimos na FIGURA 6.1 um fluxograma possível para a implementação de um programa de tempo real que incorpore a função agendador. A proposta deste trabalho é desenvolver novos algoritmos para a solução do componente agendador, baseados em novas estratégias. Os demais componentes do programa são importantes e devem ser aderentes ao algoritmo proposto; entretanto não fazem parte do objetivo proposto no presente trabalho.

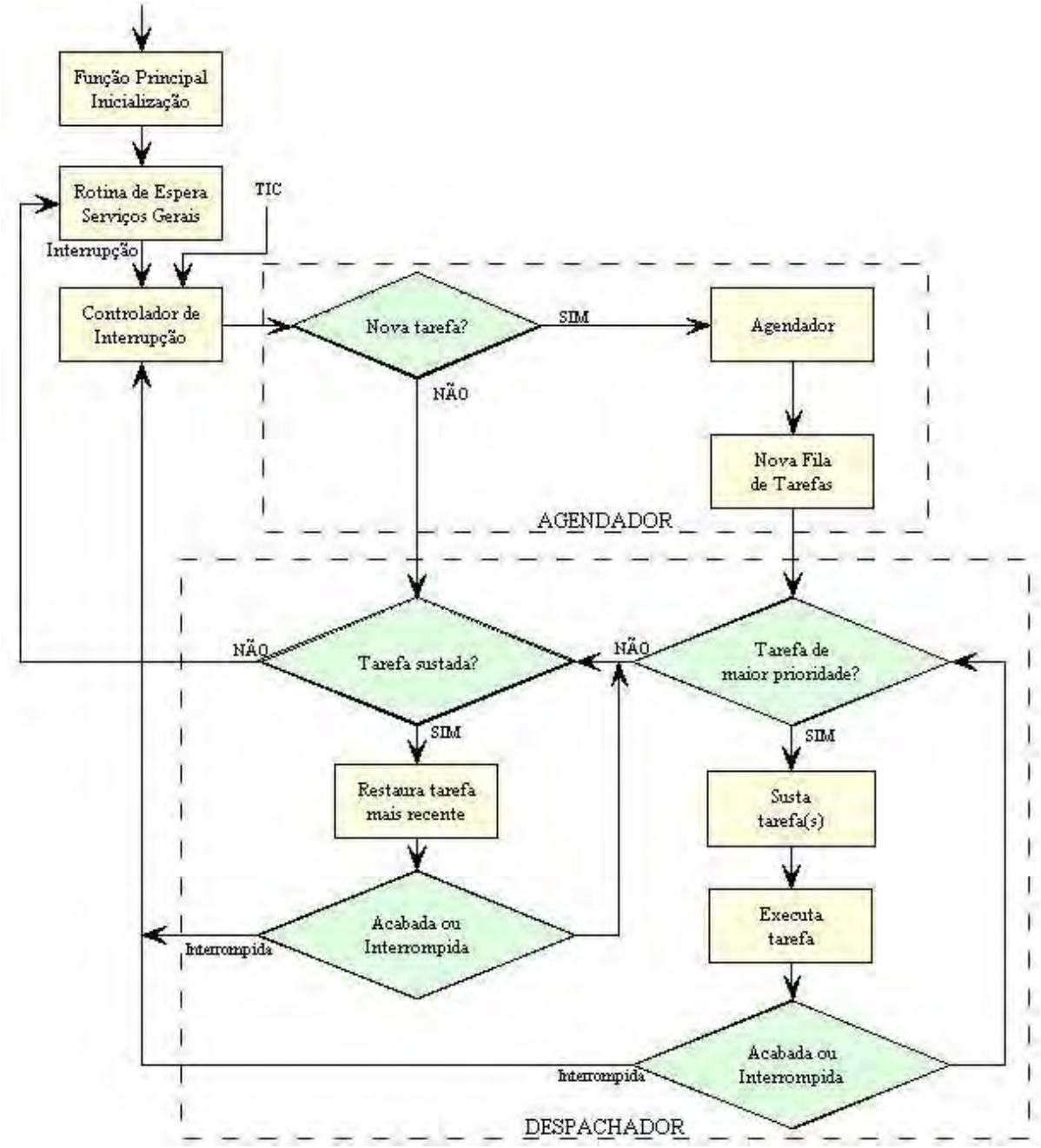

FIGURA 6.1 – Fluxograma de uma programa de sistema de tempo real.

Apresentaremos a seguir os algoritmos propostos para a solução do componente agendador, baseados nas estratégias propostas no Capítulo 5.

6.2 Novo Algoritmo Agendador a Taxa Constante Dinâmico Estendido

O novo algoritmo agendador a taxa constante dinâmico estendido é apresentado na FIGURA 6.2.

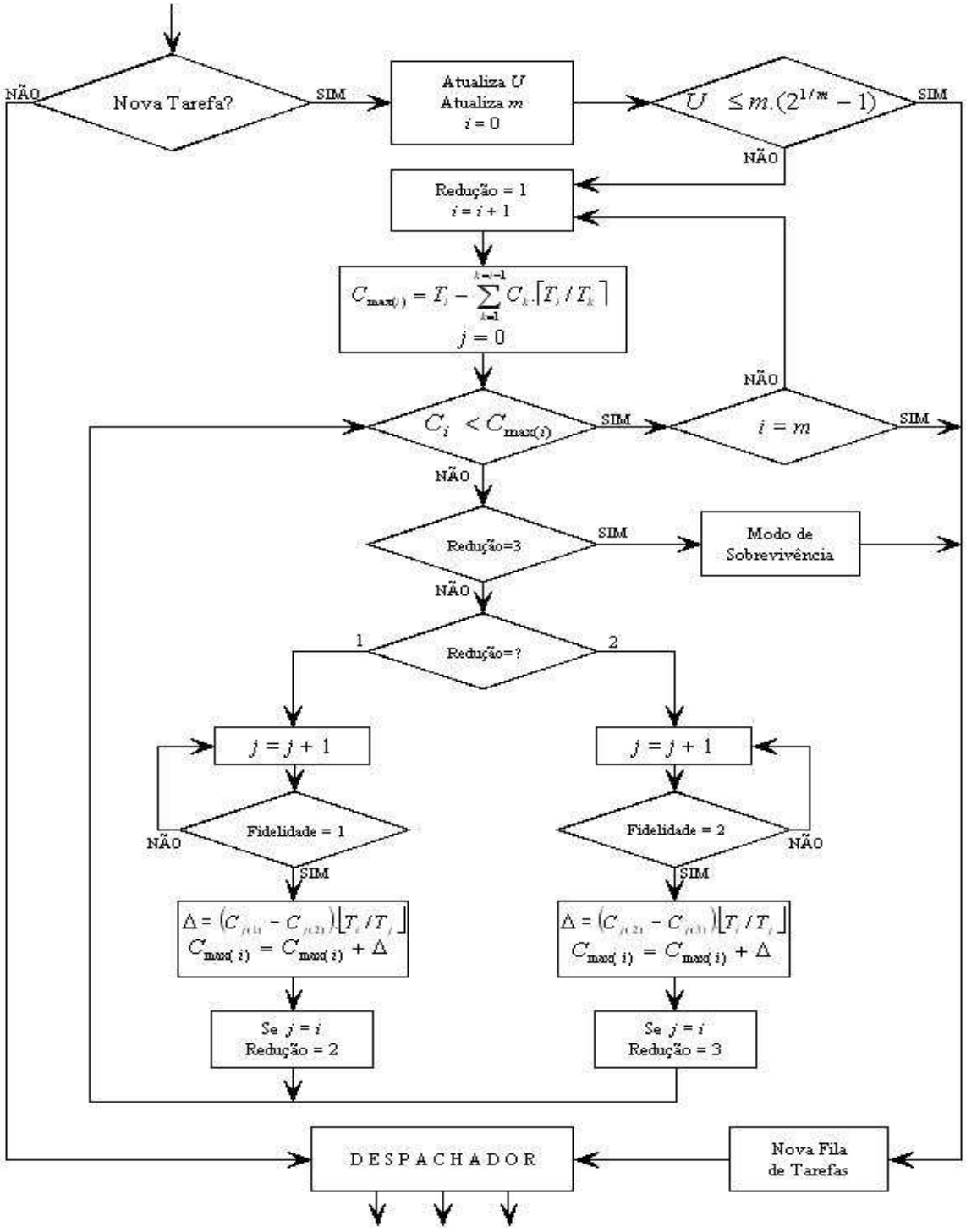

FIGURA 6.2 – Algoritmo agendador a taxa constante estendido.

Este algoritmo necessita a atualização dos valores do fator de utilização e do número de tarefas para o teste rápido de agendabilidade que é representado pelo ramo lateral externo do lado direito. Somente se falhar este teste é que será feita a verificação conforme o critério de agendabilidade proposto para a extensão do algoritmo de taxa constante. Este critério apresenta uma complexidade relativa a um somatório, que por sua vez possui uma divisão, um arredondamento e um produto, para cada tarefa do conjunto de tarefas a ser agendado. Para um conjunto de tarefas agendável, o sistema percorre um ramo menor cuja duração pode ser definida como uma função do número total de tarefas. Se o conjunto de tarefas não é agendável, o agendador busca um ponto possível de agendamento através da redução dos níveis de fidelidade das tarefas de fidelidade maior e, se necessário, de todas as tarefas. Um ajuste aproximado no tempo máximo permitido de execução da tarefa corrente é efetuado se ocorrer a redução do tempo de execução de outra tarefa, para permitir uma verificação rápida de agendabilidade para cada tarefa reduzida. Se, após reduzir o tempo de execução de todas as tarefas ao máximo possível, o critério de agendabilidade não for satisfeito, o agendador entre no **modo de sobrevivência**. Embora possam ser implementados diferentes **modos de sobrevivência**, sugerimos um que construa um subconjunto incremental de tarefas a partir daquela de maior frequência até àquela referente ao ponto imediatamente anterior ao critério de agendabilidade dado pelo teorema de Liu e Layland. Neste caso, torna-se ainda mais importante a compensação dos tempos gastos com processamento do agendamento. De uma forma geral, este algoritmo tende a compensar o tempo gasto com o agendamento obtendo um maior fator de utilização do processador na região entre o plano de Liu e Layland e a superfície de agendabilidade máxima. Um agendador baseado neste algoritmo tende a se tornar inviável à medida que o número de tarefas cresce muito rapidamente porque o tempo gasto no algoritmo de agendamento cresce proporcionalmente.

6.3 Novo Algoritmo Agendador Com Sintonia de (T_i, T_j)

O novo algoritmo agendador com sintonia de (T_i, T_j) é apresentado na FIGURA 6.3. Este algoritmo é fundamentalmente o apresentado anteriormente com a alteração do

critério de agendabilidade. As operações de arredondamento das divisões efetuadas são eliminadas em função das boas relações entre os períodos de todas as tarefas. Considerando que esta operação está incluída em um somatório, esta redução é significativa. Outra vantagem esperada, conforme será verificado posteriormente, é permitir um maior fator de utilização do processador. Na prática, para um conjunto de tarefas bem distribuídas (com pesos iguais), este critério tende a maximizar a taxa de utilização do processador para até 100%. De uma forma geral, este algoritmo permite o agendamento a taxa constante aproximando-se do plano de agendabilidade máxima (U=100%) em contrapartida às restrições de projeto impostas pela sintonia dos períodos das tarefas.

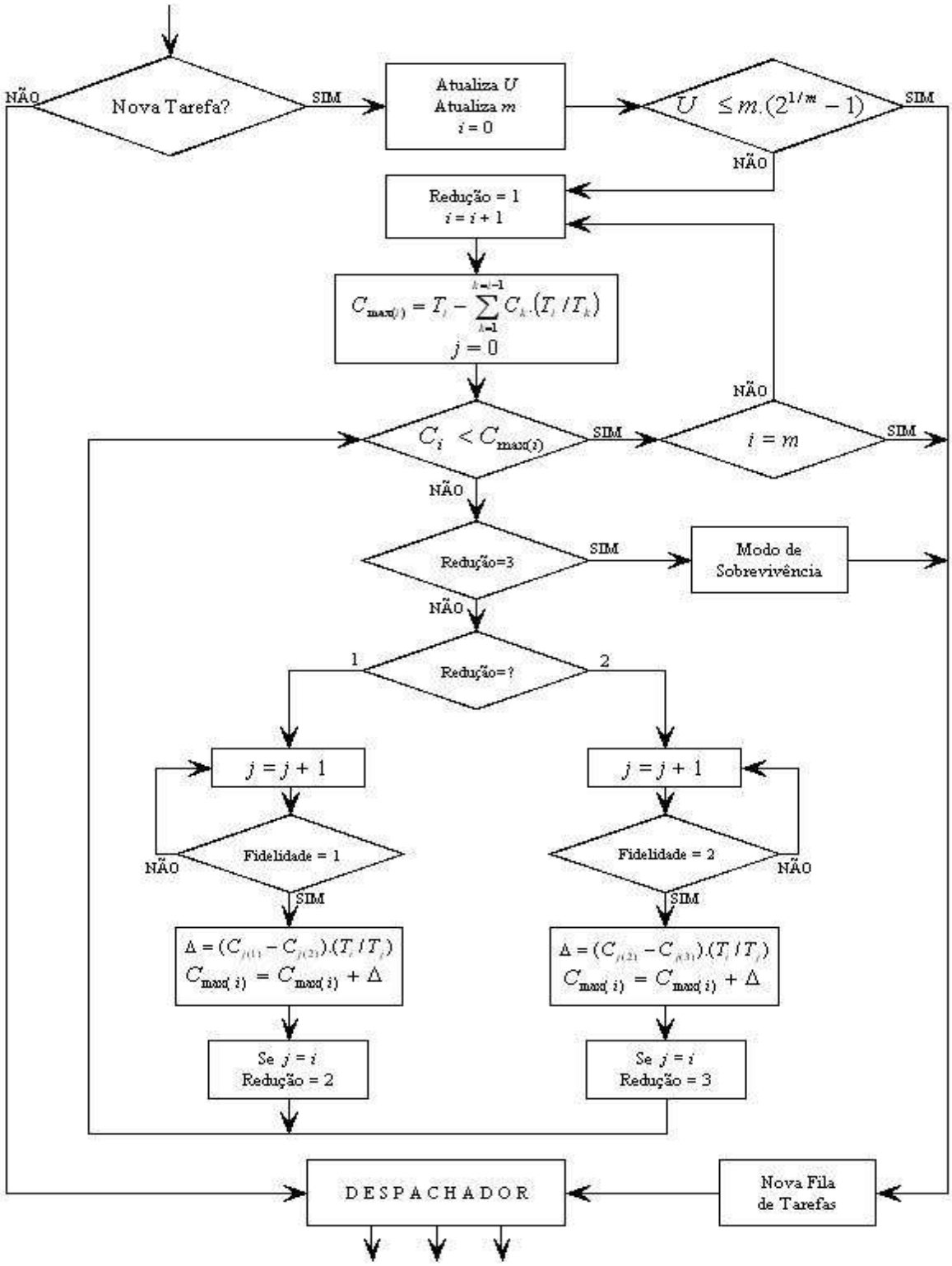

FIGURA 6.3 - Algoritmo agendador com sintonia de (T_i, T_j).

6.4 Novo Algoritmo do Agendador Com Primeiro Prazo Fatal Primeiro Estendido

O novo algoritmo do agendador com primeiro prazo fatal primeiro estendido é apresentado na FIGURA 6.4.

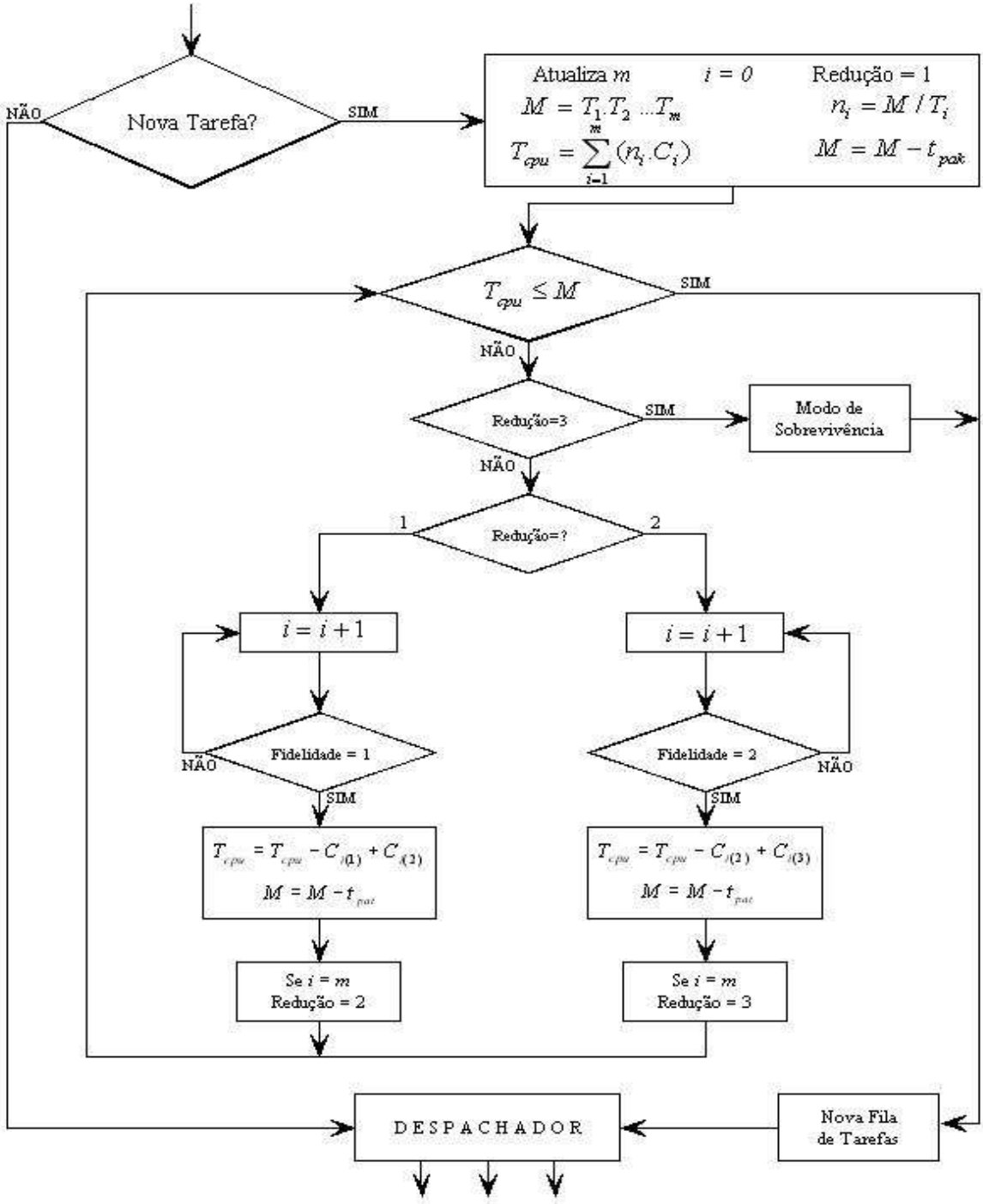

FIGURA 6.4 - Algoritmo agendador com primeiro prazo fatal primeiro estendido.

Este algoritmo apresenta uma parte inicial mais complexa porque necessita calcular o Mínimo Múltiplo Comum-MMC dos períodos das tarefas (pode se usar o produto direto em alguns projetos sob condições) e o tempo gasto por todas as tarefas neste intervalo de interesse. Se o tempo gasto com o agendamento for considerado não nulo, então, também nesta parte inicial, o tempo disponível para as tarefas deve ser reduzido da parte constante do tempo de processamento do agendador. Todo o tempo de cálculo do agendamento pode ser representado em uma parte constante diretamente proporcional ao número total de tarefas do conjunto e uma parte variável proporcional e relativa a cada redução de tarefa efetuada. O critério de agendabilidade é uma comparação de números inteiros e é bastante rápida. Caso o conjunto de tarefas não seja agendável, o agendador busca um ponto possível de agendamento através da redução dos níveis de fidelidade das tarefas de fidelidade maior e, se necessário, de todas as tarefas. O tempo exato do ramo de redução dos níveis de fidelidade é diminuído do tempo disponível para as tarefas restantes antes da nova verificação de agendabilidade para cada tarefa com tempo de execução reduzido. Se, após reduzir o tempo de execução de todas as tarefas ao máximo possível, o critério de agendabilidade não for satisfeito, o agendador entre no modo de sobrevivência. O modo de sobrevivência sugerido constrói um subconjunto incremental de tarefas a partir daquela de maior frequência até àquela referente ao ponto imediatamente anterior ao não cumprimento do critério proposto, sempre diminuindo os tempos gastos com o próprio processamento. Um agendador baseado neste algoritmo usa o fator de utilização U de 100% como limite para a agendabilidade de todas as tarefas independentemente do ponto inicial de agendabilidade do conjunto de tarefas proposto.

6.5 Novo Algoritmo Agendador Dinâmico Para Sistemas de Tempo Real Críticos

O novo algoritmo do agendador dinâmico para sistemas de tempo real críticos é apresentado na FIGURA 6.5. Este algoritmo é estruturalmente e matematicamente similar ao do agendador com primeiro prazo fatal primeiro estendido com uma restrição maior no fator de utilização. Esta restrição maior, abaixo do plano de Liu e Layland, é compensada pela reserva de 30% do tempo de processamento para as tarefas

assíncronas e outras necessárias para a implementação do mesmo em sistemas de tempo real críticos.

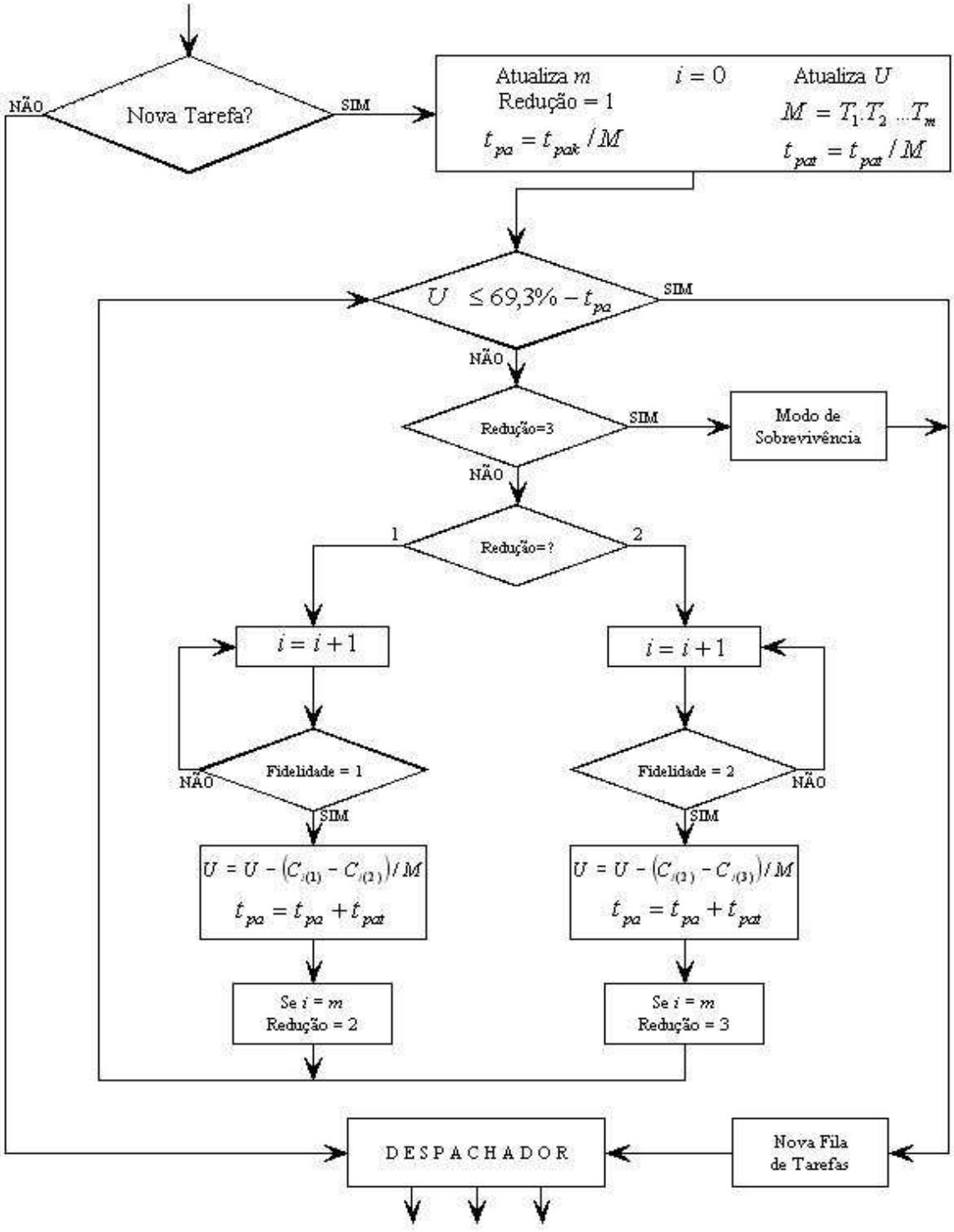

FIGURA 6.5 - Algoritmo agendador dinâmico para sistemas de tempo real críticos.

Ele também apresenta o mesmo comportamento do novo agendador com primeiro prazo fatal primeiro estendido: o cálculo do agendamento é previsível, é rápido, usa o princípio de redução dos níveis de fidelidade das tarefas, utiliza cálculo exato de tempo e possui o mesmo modo de sobrevivência do caso anterior.

Este algoritmo permite a realização das tarefas utilizando os diferentes critérios apresentados no Capítulo 2. O critério “o mais rápido possível-SAF” é adequado para situações de emergência como manobras de escape ou ejeção de assentos, entretanto, devido aos requisitos de sistemas de tempo real, esses sinais necessitam de redundância e dissimilaridade e, quando aplicável, até mesmo por sinais de hardware. O critério “depois de um tempo definido-ATR” deve ser utilizado em situações em que uma pequena variação de tempo no cálculo do controle ou saída provoca um erro inaceitável na saída, como nos momentos de lançamento de bombas ou de manobras de mudança de órbita. O critério “após o próximo gatilho-ANT” deve ser utilizado para evitarmos erros involuntários de operação ou quando necessitamos de uma confirmação de decisão
.

6.6 Experimento Para Testes e Comparação dos Algoritmos

É importante o projeto de um experimento adequado para a verificação e comparação das propriedades e do desempenho dos agendadores propostos. No Capítulo 5 foram utilizados conjuntos de três e quatro tarefas apenas para uma maior facilidade de compreensão e visualização gráfica; mas estes não são adequados para tais verificações e comparações. Neste Capítulo 6 necessitamos de ambientes mais complexos e dinâmicos, contendo muitas tarefas e alterações, para evidenciarmos as diferenças de cada agendador. É muito importante também realizarmos experimentos contendo tarefas com períodos e tempos de duração baseados em situações reais.

O experimento foi projetado considerando-se a capacidade de um pequeno sistema de controle de defesa, que pode estar embarcado em uma aeronave ou em uma fragata, com capacidade operacional prevista de rastreamento simultâneo de 16 ameaças em condições normais de operação. Este módulo de defesa é processado em um computador comercial dedicado, enquanto as demais tarefas são realizadas em outros

processadores específicos. Este sistema de defesa é avaliado: em **condições normais de operação (16 tarefas)**, em **condições de sobrecarga moderada (24 tarefas)** e em **condições de sobrecarga excessiva (36 tarefas)**. Essas condições estão relacionadas ao fator de utilização U do processador respectivamente nas regiões: abaixo do plano de Liu e Layland LL; entre este plano e o plano de agendabilidade máxima LP; e acima do plano de agendabilidade máxima.

O conjunto de tarefas representando as condições normais de operação é apresentado na TABELA 6.1. O intervalo de períodos das tarefas varia de 20 ms a 1280 ms, correspondendo a um intervalo de frequências de 50 Hz até aproximadamente 0,78 Hz. A duração de cada tarefa (em micro-segundos na TABELA 6.1) em maior nível de fidelidade corresponde a 10% do seu período; e para os demais níveis de fidelidade são reduzidos em uma ordem de grandeza. Outros valores de duração podem ser perfeitamente utilizados sem grandes alterações de resultado se o fator de utilização for aproximadamente o mesmo. O intervalo de tempo analisado é o instante crítico para todas tarefas e, portanto, o MMC = 288.000 ms, e o fator de utilização inicial é U = 65,60%.

TABELA 6.1 – Conjunto de tarefas para a operação normal do sistema.

Índice (n)	**Frequência** (Hz)	**Tempo** (ms)	$C_{i(1)}$ (us)	$C_{i(2)}$ (us)	$C_{i(3)}$ (us)	**Fid.**	C_i (us)	U_i	n_i
1	50,00	20	1000	100	10	1	1000	5,00%	14400
2	25,00	40	2000	200	20	1	2000	5,00%	7200
3	12,50	80	4000	400	40	1	4000	5,00%	3600
4	10,00	100	5000	500	50	1	5000	5,00%	2880
5	6,25	160	8000	800	80	1	8000	5,00%	1800
6	5,00	200	10000	1000	100	1	10000	5,00%	1440
7	3,13	320	16000	1600	160	1	16000	5,00%	900
8	2,50	400	20000	2000	200	3	200	0,05%	720
9	2,00	500	25000	2500	250	2	2500	0,50%	576
10	1,67	600	30000	3000	300	1	30000	5,00%	480
11	1,56	640	32000	3200	320	1	32000	5,00%	450
12	1,33	750	37500	3750	375	1	37500	5,00%	384
13	1,25	800	40000	4000	400	1	40000	5,00%	360
14	1,11	900	45000	4500	450	1	45000	5,00%	320
15	1,00	1000	50000	5000	500	3	500	0,05%	288
16	0,78	1280	64000	6400	640	1	64000	5,00%	225
	M =	288000					U =	65,60%	

O índice i representa o número da tarefa, o tempo de execução para os níveis de fidelidade 1, 2 e 3 são dados respectivamente por $C_{i(1)}$, $C_{i(2)}$ e $C_{i(3)}$. O sistema está

operando normalmente com os níveis de fidelidade fornecidos na coluna Fid., a coluna C_i é o tempo de execução da tarefa i no respectivo nível de fidelidade, a coluna U_i é a contribuição para o fator de utilização da tarefa i, e n_i é o número de ocorrências da tarefa i no MMC.

Para o sistema operar no modo moderadamente sobrecarregado foram adicionadas mais oito tarefas conforme a TABELA 6.2. O acréscimo deste conjunto de tarefas solicita um aumento do fator de utilização do processador que leva a demanda total a 96,60%. Assim, os agendadores propostos devem ser utilizados para a elaboração da nova lista de tarefas a ser enviada para o despachador.

TABELA 6.2 - Conjunto adicional de tarefas para a operação do sistema com sobrecarga moderada.

Índice (n)	**Frequência** (Hz)	**Tempo** (ms)	$C_{i(1)}$ (us)	$C_{i(2)}$ (us)	$C_{i(3)}$ (us)	**Fid.**	C_i (us)	U_i	n_i
17	25,00	40	2000	200	20	1	2000	5,00%	7200
18	10,00	100	5000	500	50	1	5000	5,00%	2880
19	5,00	200	10000	1000	100	1	10000	5,00%	1440
20	2,50	400	20000	2000	200	1	20000	5,00%	720
21	1,67	600	30000	3000	300	2	3000	0,50%	480
22	1,33	750	37500	3750	375	1	37500	5,00%	384
23	1,11	900	45000	4500	450	1	45000	5,00%	320
24	0,78	1280	64000	6400	640	2	6400	0,50%	225
							U =	96,60%	

TABELA 6.3 - Conjunto adicional de tarefas para a operação do sistema com sobrecarga excessiva.

Índice (n)	**Frequência** (Hz)	**Tempo** (ms)	$C_{i(1)}$ (us)	$C_{i(2)}$ (us)	$C_{i(3)}$ (us)	**Fid.**	C_i (us)	U_i	n_i
25	50,00	20	1000	100	10	1	1000	5,00%	14400
26	25,00	40	2000	200	20	2	200	0,50%	7200
27	12,50	80	4000	400	40	1	4000	5,00%	3600
28	6,25	160	8000	800	80	1	8000	5,00%	1800
29	5,00	200	10000	1000	100	1	10000	5,00%	1440
30	3,13	320	16000	1600	160	3	160	0,05%	900
31	2,00	500	25000	2500	250	1	25000	5,00%	576
32	1,67	600	30000	3000	300	2	3000	0,50%	480
33	1,56	640	32000	3200	320	1	32000	5,00%	450
34	1,25	800	40000	4000	400	1	40000	5,00%	360
35	1,00	1000	50000	5000	500	1	50000	5,00%	288
36	0,78	1280	64000	6400	640	2	6400	0,50%	225
							U =	138,15%	

Para o sistema operar no modo de sobrecarga excessiva foram adicionadas mais doze tarefas ao ambiente sobrecarregado, conforme a TABELA 6.3. O acréscimo deste conjunto de tarefas solicita um aumento do fator de utilização do processador que leva a demanda total a 138,15%, claramente não agendável. Da mesma forma, os agendadores devem elaborar a nova lista de tarefas a ser enviada para o despachador.

Devido às restrições inerentes ao próprio agendador sintonizado, foram necessárias alterações no conjunto de tarefas anteriores para o seu teste. Para permitir a sintonia de períodos, cada tarefa necessária teve seu período ajustado para o valor mais próximo e, para permitir a comparação com os demais agendadores, o respectivo tempo de duração da tarefa foi ajustado para mantermos o mesmo fator de utilização solicitado ao processador. As tarefas dos modos de operação em condições normais, sobrecarga e de sobrecarga excessiva são dadas, respectivamente, nas TABELA 6.4, TABELA 6.5 e TABELA 6.6. Os destaques de cor na primeira coluna representam as tarefas ajustadas e o instante crítico para todas tarefas é o MMC = 1280 ms.

TABELA 6.4 – Conjunto de tarefas para a operação normal do sistema sintonizado.

Índice (n)	**Frequência** (Hz)	**Tempo** (ms)	$C_{i(1)}$ (us)	$C_{i(2)}$ (us)	$C_{i(3)}$ (us)	**Fid.**	C_i (us)	U_i	n_i
1	50,00	20	1000	100	10	1	1000	5,00%	64
2	25,00	40	2000	200	20	1	2000	5,00%	32
3	12,50	80	4000	400	40	1	4000	5,00%	16
4	12,50	80	4000	400	40	1	4000	5,00%	16
5	6,25	160	8000	800	80	1	8000	5,00%	8
6	6,25	160	8000	800	80	1	8000	5,00%	8
7	3,13	320	16000	1600	160	1	16000	5,00%	4
8	3,13	320	16000	1600	160	3	160	0,05%	4
9	3,13	320	16000	1600	160	2	1600	0,50%	4
10	1,56	640	32000	3200	320	1	32000	5,00%	2
11	1,56	640	32000	3200	320	1	32000	5,00%	2
12	1,56	640	32000	3200	320	1	32000	5,00%	2
13	1,56	640	32000	3200	320	1	32000	5,00%	2
14	0,78	1280	64000	6400	640	1	64000	5,00%	1
15	0,78	1280	64000	6400	640	3	640	0,05%	1
16	0,78	1280	64000	6400	640	1	64000	5,00%	1
	M =	1280					U =	65,60%	

TABELA 6.5 - Conjunto adicional de tarefas para a operação do sistema sintonizado com.sobrecarga.

Índice (n)	Frequência (Hz)	Tempo (ms)	$C_{i(1)}$ (us)	$C_{i(2)}$ (us)	$C_{i(3)}$ (us)	Fid.	C_i (us)	U_i	n_i
17	25,00	40	2000	200	20	1	2000	5,00%	32
18	12,50	80	4000	400	40	1	4000	5,00%	16
19	6,25	160	8000	800	80	1	8000	5,00%	8
20	3,13	320	16000	1600	160	1	16000	5,00%	4
21	1,56	640	32000	3200	320	2	3200	0,50%	2
22	1,56	640	32000	3200	320	1	32000	5,00%	2
23	0,78	1280	64000	6400	640	1	64000	5,00%	1
24	0,78	1280	64000	6400	640	2	6400	0,50%	1
							U =	96,60%	

TABELA 6.6 - Conjunto adicional de tarefas para a operação do sistema sintonizado com sobrecarga excessiva.

Índice (n)	Frequência (Hz)	Tempo (ms)	$C_{i(1)}$ (us)	$C_{i(2)}$ (us)	$C_{i(3)}$ (us)	Fid.	C_i (us)	U_i	n_i
25	50,00	20	1000	100	10	1	1000	5,00%	64
26	25,00	40	2000	200	20	2	200	0,50%	32
27	12,50	80	4000	400	40	1	4000	5,00%	16
28	6,25	160	8000	800	80	1	8000	5,00%	8
29	6,25	160	8000	800	80	1	8000	5,00%	8
30	3,13	320	16000	1600	160	3	160	0,05%	4
31	3,13	320	16000	1600	160	1	16000	5,00%	4
32	1,56	640	32000	3200	320	2	3200	0,50%	2
33	1,56	640	32000	3200	320	1	32000	5,00%	2
34	1,56	640	32000	3200	320	1	32000	5,00%	2
35	0,78	1280	64000	6400	640	1	64000	5,00%	1
36	0,78	1280	64000	6400	640	2	6400	0,50%	1
							U =	138,15%	

6.7 Resultados Experimentais com 16 Tarefas

O sistema está operando em condições normais, e utilizaremos uma pequena perturbação em torno desse ponto de operação (U= 65,60%) para avaliação do desempenho dos algoritmos propostos. Visto que o sistema tem fator de utilização inferior ao limite de Liu e Layland para m=16 (69,3%), o conjunto de tarefas é agendável sem alteração utilizando o agendador RMS ou o EDF. Todos os algoritmos propostos utilizam apenas o ramo externo para verificação da agendabilidade. O fator de utilização nos agendadores RMS estendido e sintonizado é uma função apenas do número *m* de tarefas e seus cálculos são mais rápidos que nos casos dos agendadores

EDF estendido e crítico, onde são necessárias várias operações matemáticas. Nestas condições, os algoritmos RMS estendido e sintonizado apresentam um desempenho melhor que os agendadores EDF estendido e crítico; entretanto, do ponto de vista prático, isto não representa nenhuma vantagem pois o processador apresenta tempo livre nestas mesmas condições. Conforme a análise feita no Capítulo 2, o algoritmo EDF estendido apresentará um menor número de sustação de tarefas em contraposição a uma maior complexidade na implementação do despachador.

6.8 Resultados Experimentais com 24 Tarefas

O sistema estava operando em condições normais com as 16 tarefas anteriores quando foi acrescentado no mesmo instante (instante crítico) o subconjunto de mais 8 tarefas para levar o sistema a operar no modo sobrecarregado. O novo fator de utilização solicitado é U = 96,60%. O resultado dos níveis de fidelidade finais para cada tarefa necessários para o agendamento do sistema é apresentado na TABELA 6.7. São dados o fator de utilização do conjunto final de tarefas agendadas e a quantidade de tarefas com os níveis de fidelidade alterados.

O agendador RMS estendido não passa pelo critério rápido de agendabilidade e, portanto, o critério proposto em 6.2 exige uma complexidade maior de cálculo. Como este agendador é mais restritivo, é necessária a redução dos níveis de fidelidade para o agendamento das tarefas. Após a redução no nível de fidelidade de 3 tarefas, o conjunto de tarefas passou a ser agendável com U = 83,10% superior ao limite de Liu e Layland.

Os agendadores sintonizado e EDF estendido, com U = 96,60%, passam pelo critério de agendabilidade sem a necessidade da alteração dos níveis de fidelidade de qualquer tarefa.

O agendador crítico não passa pelo critério rápido de agendabilidade e necessita a redução do nível de fidelidade de tarefas. Como este agendador é ainda mais restritivo que o RMS estendido, ele provoca uma redução no nível de fidelidade de mais tarefas, no caso 7, resultando em U = 65,10% .

TABELA 6.7 – Níveis de fidelidade para agendamento com sobrecarga moderada.

Índice (n)	Fid. inicial	RMS estendido	Sintonia	EDF estendido	Crítico
1	1	2	1	1	2
2	1	2	1	1	2
17	1	2	1	1	2
3	1	1	1	1	2
4	1	1	1	1	2
18	1	1	1	1	2
5	1	1	1	1	2
6	1	1	1	1	1
19	1	1	1	1	1
7	1	1	1	1	1
8	3	3	3	3	3
20	1	1	1	1	1
9	2	2	2	2	2
10	1	1	1	1	1
21	2	2	2	2	2
11	1	1	1	1	1
12	1	1	1	1	1
22	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
23	1	1	1	1	1
15	3	3	3	3	3
16	1	1	1	1	1
24	2	2	2	2	2
U	96,60%	83,10%	96,60%	96,60%	65,10%
Aterações efetuadas		3	0	0	7

6.9 Resultados Experimentais com 36 Tarefas

O sistema estava operando em condições normais com as 16 tarefas anteriores quando foram acrescentados no mesmo instante (instante crítico) os subconjuntos de mais 8 tarefas do modo moderadamente sobrecarregado mais o outro de 12 tarefas do modo excessivamente sobrecarregado, necessárias para, em conjunto, levarem o sistema a operar no modo de sobrecarga excessiva. O novo fator de utilização solicitado é U = 138,15%.De forma similar ao caso anterior o resultado dos níveis de fidelidade finais para cada tarefa necessários para o agendamento do sistema é apresentado na TABELA 6.8

TABELA 6.8 - Níveis de fidelidade para agendamento com sobrecarga excessiva.

Índice (n)	**Fid.**	**RMS**	**Sintonia**	**EDF**	**Crítico**
1	1	2	2	2	2
25	1	2	2	2	2
2	1	2	2	2	2
17	1	2	2	2	2
26	2	2	2	2	2
3	1	2	2	2	2
27	1	2	2	2	2
4	1	2	2	2	2
18	1	2	2	2	2
5	1	2	1	2	2
28	1	2	3	1	2
6	1	2	1	1	2
19	1	2	2	1	2
29	1	2	1	1	2
7	1	1	2	1	2
30	3	3	1	3	3
8	3	3	1	3	3
20	1	1	1	1	2
9	2	2	1	2	2
31	1	1	1	1	2
10	1	1	1	1	1
21	2	2	3	2	2
32	2	2	1	2	2
11	1	1	2	1	1
33	1	1	1	1	1
12	1	1	2	1	1
22	1	1	1	1	1
13	1	1	1	1	1
34	1	1	1	1	1
14	1	1	3	1	1
23	1	1	1	1	1
15	3	3	2	3	3
35	1	1	1	1	1
16	1	1	1	1	1
24	2	2	1	2	2
36	2	2	2	2	2
U	138,15%	79,65%	97,65%	97,65%	66,15%
Aterações efetuadas		13	9	9	16

Sob condição de sobrecarga excessiva faz-se necessária a redução de níveis de fidelidade de tarefas para a agendamento das tarefas usando qualquer agendador. O

agendador RMS estendido reduz o nível de fidelidade de 13 tarefas produzindo um agendamento com U = 79,65%, ainda superior ao limite de Liu e Layland.

Os agendadores sintonizado e EDF estendido, reduzem o nível de fidelidade das mesmas 9 tarefas produzindo um agendamento com U = 97,65%, bem próximo ao plano de agendabilidade máxima.

O agendador crítico reduz o nível de fidelidade de mais tarefas, em número de 16, resultando em U = 66,15% .

6.10 Comparação dos Resultados

Todos os algoritmos propostos garantem o agendamento das tarefas utilizando para tal, o mecanismo de redução dos níveis de fidelidade das tarefas. O custo desta garantia é exatamente a complexidade no projeto de sistemas com a preparação dos respectivos níveis de fidelidade para cada tarefa. Essa é uma alternativa viável para o desenvolvimento de sistemas de tempo críticos em ambientes dinâmicos.

Quando o sistema opera com fator de utilização do processador inferior ao limite de Liu e Layland, todos os agendadores apresentam um custo computacional praticamente irrelevante. O número de sustações inferior (**Erro! Fonte de referência não encontrada.**) no caso do algoritmo EDF estendido não se justifica neste modo de operação, por duas razões: i) o número de sustações é baixo em todos os casos e ii) não tem sentido aumentar a complexidade de implementação para ganhar tempo de processamento quando não é necessário mais tempo de processamento.

Quando o sistema opera com fator de utilização do processador acima do limite de Liu e Layland e abaixo do plano de agendabilidade máxima, aparecem as diferenças entre os diversos agendadores propostos.

Os agendadores RMS estendido e o crítico provocam uma redução do nível de fidelidade de um número maior de tarefas. O RMS estendido apresenta um maior fator de utilização, mas o seu custo computacional é bastante superior ao do agendador crítico. Enquanto o primeiro usa um critério que inclui operações algébricas dentro de

uma operação de somatório para cada tarefa, o segundo utiliza apenas uma comparação simples e, portanto, extremamente rápida. Para um ambiente com grande número de tarefas, operando no modo sobrecarregado, o ganho de processamento do agendador RMS estendido não é significativo em relação ao processamento do agendador crítico, considerando o seu custo computacional. É necessária uma análise deste aspecto em relação às condições do processador onde os referidos algoritmos serão implementados.

Os agendadores sintonizado e EDF estendido, em oposição aos agendadores RMS estendido e o crítico, não provocam a redução do nível de fidelidade de nenhuma tarefa porque ambos garantem a agendabilidade das tarefas até o limite do fator de utilização do processador (U=100%). O agendador sintonizado apresenta um custo computacional bem superior ao EDF estendido pela mesma razão da comparação anterior no cálculo do critério de agendabilidade enquanto o EDF estendido necessita de um despachador mais complexo. Este custo computacional da maior complexidade do despachador do EDF, em geral, é inferior ao custo computacional acrescido no critério utilizado pelo agendador sintonizado.

Quando o sistema solicita operação com fator de utilização do processador acima do plano de agendabilidade máxima (sobrecarga excessiva) são válidas as mesmas comparações efetuadas no caso anterior, acrescentando as influências do número de sustações efetuadas entre os agendadores baseados no RMS (RMS estendido, sintonizado e crítico) e o agendador EDF estendido.

Para a análise do número de sustações de tarefas provocado pelos dois tipos de despachadores necessários à implementação dos agendadores propostos (o 1^{o}. tipo para os agendadores RMS estendido, sintonizado e crítico; e o 2^{o}. tipo para o agendador EDF estendido) foi desenvolvido um simulador de um núcleo despachador de uma CPU, através de dois modelos discretos baseados em eventos, e implementados no ambiente de simulação discreta do software ProModel.

Utilizamos um cenário de referência constituído pelas mesmas 36 tarefas finais do agendador RMS sintonizado, que é praticamente o mesmo cenário do agendador EDF estendido. Foram feitas simulações para cada conjunto de tarefas, contendo de 16 a 36

tarefas, partindo do cenário inicial e adicionando, uma a uma, em ordem de maior frequência, as demais tarefas. Todas as simulações foram efetuadas com início no instante crítico e duração do período crítico para a tarefa de número 36 (1280 ms). Os resultados estão apresentados na FIGURA 6.6.

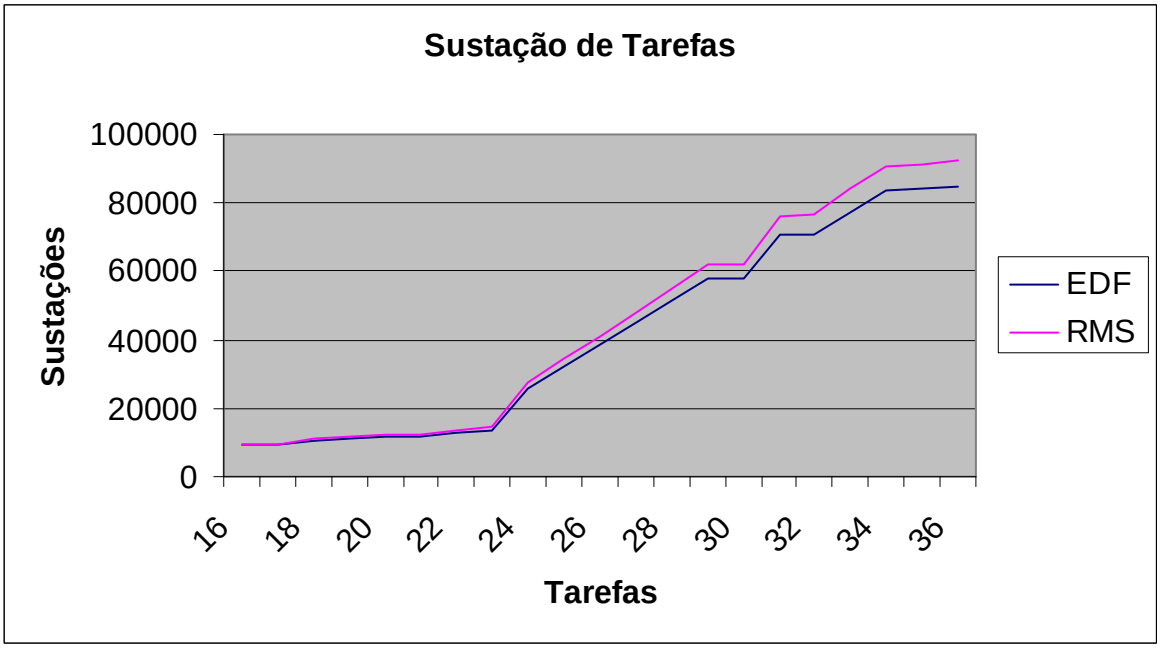

FIGURA 6.6 – Sustação de tarefas para ambiente dinâmico.

Como previsto, o número de sustações de tarefas para o despachador do agendador EDF estendido é inferior ao número de sustações quando utilizamos o despachador do agendador RMS sintonizado. A diferença aumenta na medida em que o número de tarefas m aumenta. Para uma comparação das vantagens de se utilizar um agendador baseado em um ou outro tipo de despachador devemos fazer a análise em função do ambiente computacional que iremos implementar o programa do sistema de tempo real, considerando todos os tempos descritos no Capítulo 2. Considerando a diferença da ordem de 500 sustações entre os dois casos, para as 36 tarefas no seu período crítico, comparado com o total de 33.600 execuções de todas as tarefas neste mesmo período, podemos afirmar que a agendabilidade é bastante sensível quanto à técnica e forma de implementação em baixo nível dos mesmos agendadores e despachadores.

CAPÍTULO 7

APLICAÇÃO NA ARQUITETURA HLA

7.1 Razões Para Uso da Arquitetura HLA na Indústria Aeroespacial e de Defesa

HLA é uma arquitetura integrada de software desenvolvida para a criação de modelos e simulação computacional de sistemas ou para a criação de modelos e simulação computacional de componentes, visando a interoperabilidade e reuso. O padrão HLA atualmente é um padrão formado pelas seguintes normas IEEE: 1516, 1516.1 e 1516.2, conforme apresentado no Capítulo 2.

Sem fazermos uma apologia do uso cego da tecnologia a qualquer custo, sem a necessária análise do custo/benefício de cada caso, apresentaremos algumas áreas de aplicação da arquitetura HLA de nosso interesse baseados em: i) nos estudos efetuados; ii) nas visitas técnicas, participações em conferências e feiras da indústria aeronáutica, espacial e de defesa; iii) nos trabalhos apresentados; e, iv) nas discussões técnicas; ocorridas durante o desenvolvimento deste trabalho. Os argumentos estão ordenados de uma maneira sequencialmente causal no tempo embora a interdependência entre eles seja mais complexa.

Redistribuição de tempo entre fases de projeto: É notório o aumento do volume e complexidade de softwares embarcados e de suporte em sistemas de engenharia, em particular, de engenharia aeroespacial e de sistemas de defesa. O uso de tecnologias avançadas tem provocado um consequente deslocamento de horas em cada fase de um programa. O tempo dispensado com ensaios seja em solo ou em vôo, tem aumentado duplamente: i) o absoluto devido ao aumento da complexidade e, o relativo, devido à diminuição do tempo de outras fases. Por exemplo, no caso do JSF, com o uso da especificação dos displays feita em VAPS, o tempo de projeto e codificação foi praticamente reduzido à margem de erro gerencial.

Complexidade dos ensaios: A complexidade dos ensaios tem crescimento combinatório, muito maior que o linear, devido ao aumento das funcionalidades.

Conseqüentemente, o tempo para desenvolvimento de ambientes de ensaios pode tornar o produto não competitivo caso não seja feito reuso destes ambientes ou destes componentes. Para o reuso destes componentes faz-se necessário o uso de uma arquitetura comum.

Ensaios de sistemas de defesas: Os sistemas de defesa atuais tornaram-se bastante complexos com a integração de datalinks e com a incorporação de redes terrestres com diferentes tipos de estações de solo para um vasto tipo de aplicações. Em última instância de sofisticação temos os complexos ambientes de C4I2. Dado o alto grau de complexidade, risco e custo, grande parte das funcionalidades só podem ser ensaiadas ou demonstradas em solo ou em ambientes sintéticos (SE).

Custo dos ensaios: Além das situações em que os ensaios em vôo tornam-se impraticáveis, o impacto do número de vôos necessários para ensaios é grande para sistemas de defesa complexos. O número de testes, para um ou até mesmo para mais de um ensaio, por vôo passa a ser um item de controle fundamental nos custos do programa. Maior número de testes e de maior complexidade aumentam a probabilidade de se repetir um ensaio em vôo ou parte dele. Imediatamente torna-se necessário um plano de testes mais elaborado e, conseqüentemente, uma preparação maior, em solo, dos ensaios em vôo.

Preparação dos ensaios: A preparação, em solo, dos ensaios em vôo de sistemas de defesa complexos exige ambientes de simulação incluindo aeronave e ambiente. Os engenheiros de vôo devem usar estes ambientes para reprodução e análise de fases importantes, bem como “briefing” com a equipe, incluindo o piloto de ensaio. A simulação, a “priori” do ensaio que será efetuado é ferramenta essencial para o treinamento de pilotos e conseqüente melhoria no aproveitamento dos ensaios em vôo. Devem ser considerados os aspectos de segurança e confiabilidade dos ensaios.

Ambientes de ensaios: Os ambientes de ensaio tornam se ineficazes ou com custo proibitivo sem o reuso de componentes e de uma arquitetura comum com os demais dispositivos do sistema de defesa: estações de solo diversas, simuladores, treinadores, ambientes de C4I2, etc... Sistemas de defesa contam com diversos parceiros e

fornecedores de diferentes países. A melhor forma, talvez a única, de se usar e reusar uma arquitetura e/ou componentes comuns, é a adoção de uma arquitetura padrão e até mesmo de padrões para sub-componentes ou sub-sistemas.

Arquitetura padrão e padrões de fato: Com a preocupação essencial de redução de custos o DoD americano suportou a pesquisa e o desenvolvimento de uma arquitetura durante uma década: o HLA. Esta arquitetura tornou-se um standard IEEE adotada nos EUA e Europa pelos produtores/compradores de sistemas de defesas que utilizam simulação. Além do HLA, uma série de padrões de fato vêm colaborando na redução de custos, prazos e riscos em projetos deste tipo: OpenGl (HMI), OpenFlight (vôo), DTED (terreno), etc... As principais características desta arquitetura são: Interoperabilidade; Reuso; Conectividade; Arquitetura padrão – aberta; Infra-estrutura que permite monitoração de informações, conexão de simulações e conexão de participantes reais; Maturidade – foi baseada na experiência da comunidade de defesa americana com a participação e discussão dos diversos grupos envolvidos no assunto; Aderência tanto a projetos legados e orientados a objeto (distribuídos ou não); e, Reuso de ambientes. A adoção de uma arquitetura única permitiria um reuso de ambientes de integração e ensaios nos diferentes programas da empresa. Conseqüentemente, teremos uma redução de prazos e custos e uma melhoria nos processos da empresa. Além das aplicações diretamente ligadas a ensaios de sistemas complexos, processos em outras áreas serão beneficiados e, até mesmo, criadas novas oportunidades de negócios, mencionados a seguir.

Demonstrações: Ambientes integrados de simulação podem ser usados para demonstração da capacidade tecnológica da empresa ou de cumprimento de requisitos. Existem requisitos que, devido ao alto valor dos ensaios em vôo, podem ser substituídos, no todo ou em parte, por demonstrações e ensaios em ambientes sintéticos. Esse percentual torna-se maior à medida que os modelos sejam certificados.

Certificação: Ambientes dessa natureza são ferramentas imprescindíveis na certificação de aeronaves. O uso de uma arquitetura comum reduz prazo e custos.

Marketing: O marketing de um sistema de defesa complexo exige uso de ferramentas complexas. Como convencer um potencial cliente do avançado conteúdo tecnológico de um produto e sua eficácia utilizando um meio sem tecnologia? Qual seria o custo de se desenvolver esta ferramentas de marketing sem uma boa base? Como demonstrar a qualidade de um complexo sistema de defesa sem simulação?

Pré-projetos: A área de pré-projetos é fundamental para qualquer empresa no mercado aeroespacial e de defesa. Quais ferramentas estão disponíveis hoje? Quais estarão no futuro? Os ambientes de simulação incluindo sistemas, displays, aeronaves, ambientes e terrenos, são essenciais para o futuro das empresas da área.

Treinamento: A qualidade do treinamento civil melhoraria com produtos de maior qualidade na área de simuladores e com produtos de menor custo para dispositivos mais simples. Isto sempre foi considerado parte do nosso produto civil. Com a sofisticação das aeronaves militares e, respectivos sistemas de defesa, esses dispositivos de treinamento tornaram-se fundamentais. Qual o custo se treinar um piloto somente em vôo? Este treinamento seria eficaz para aeronaves com interfaces complexas. É possível capacitar um esquadrão a operar adequadamente sistemas equivalentes ao nosso SIVAM sem ambientes de simulação?

C4I2: O uso de padrões conforme diretrizes expostas facilitaria a integração das aeronaves brasileira e seus respectivos treinadores em ambientes C4I2. Essa é uma tecnologia essencial para uma empresa se estabelecer no mercado de defesa.

ATCMS: Um mercado potencial muito grande é a preparação e transformação do sistema de controle do tráfego aéreo para permitir o vôo livre. As pesquisas nessa área demandam ambientes sintéticos simulados.

7.2 Junção Teórica

Entre as principais áreas de aplicação sugeridas para uso da arquitetura HLA estão a pesquisa e desenvolvimento de ambientes complexos de simulação para controle de vôo livre e de sistemas de defesa complexo. A descrição da aplicação de uma federação para

vôo livre foi apresentada por Roza e Gool (2000). A avaliação dos serviços de suporte da RTI para sistemas de tempo real é apresentada por Ludwig e Wuerfel (2001). A avaliação da demonstração da visão dos sistemas de defesa americanos para o ano 2020 (“Joint Vision 2020”), sob aprovação do congresso americano, é apresentada por Ceranowicz et alli (2003). A demonstração destes conceitos foi efetuada com um grande experimento de simulação durante os meses de julho a agosto de 2002 envolvendo mais de 13.500 pessoas distribuídas em diferentes lugares dos Estados Unidos, mais de 35.000 entidades integradas, envolvendo simulações ao vivo, virtuais e cognitivas, integradas com partes de sistemas de defesa americano e controle de espaço aéreo. Este grande experimento de simulação incluiu os diversos tipos de protocolos de comunicação já existentes, destacando-se os suportados pelos padrões de simulação DIS e HLA. Devido ao custo e importância destes experimentos e, principalmente, aos objetivos propostos de uso no futuro indicam a necessidades de cumprimento dos requisitos de sistemas de tempo real.

A implementação de um sistema dessa natureza e complexidade com requisitos de sistemas de tempo real necessitam de mecanismos ou serviços que garantam, em tempo de projeto, o funcionamento dos mesmos sistemas. Os problemas e dificuldades para a garantia da comunicação entre as milhares de entidades do sistema são análogos aos problemas de agendamento de milhares de tarefas para processamento, substituindo o recurso escasso de tempo de processamento por tempo e volume de dados (banda de passagem). As mesmas estratégias desenvolvidas para garantia de execução de tarefas em ambientes dinâmicos e complexos podem ser aplicadas para garantia da comunicação das entidades. No caso da arquitetura HLA, comunicação entre os federados e a RTI. Este tipo de aplicação evidencia ainda mais a aplicação do conceito de níveis de fidelidade proposto em nosso trabalho e incluído nas estratégias de agendamento também propostas.

A arquitetura HLA possui o serviço de Data Distribution Management para a implementação da otimização no envio de dados entre os federados, possibilitando o envio somente dos dados de interesse entre os federados. Conforme apresentado no Capítulo 2, este é um serviço adicional e opcional e cada federado que o solicite

necessita definir uma região (espaço dimensional) de interesse. Não ocorrem interferências diretas entre os federados porque eles são isolados um do outro pelos serviços disponíveis da RTI. Os serviços do DDM, ao nível de federação, ficam responsáveis pela intersecção de todas as regiões de todos os federados, a definição das comunicações relevantes entre todos os federados e o retorno dessa informação a cada federado. O princípio de intersecção de regiões de espaços dimensionais definidos pelo usuário é muito poderoso, entretanto não provê garantia de comunicação entre os diferentes federados. Esta garantia é fundamental para a aplicação da arquitetura HLA em sistemas de tempo real como ambientes de comando e controle e sistemas de controle de tráfego aéreo. Assim propomos alterações na arquitetura HLA para incorporar serviços que permitam a aplicação dos agendadores propostos, agora aplicados aos recursos de tempo de comunicação (ou banda de passagem), como forma de garantir o cumprimento de requisitos de tempo real usando a arquitetura HLA.

7.3 Alterações Propostas na Arquitetura HLA

O HLA OMT provê um Template para documentação relevante de HLA sobre objetos federados (classes de simulação), seus atributos e as interações que podem ocorrer entre os objetos de uma federação. Ele provê um mecanismo "entendido por todos" para especificar a troca de conteúdo público e gerar coordenação entre os membros de uma federação. **Proposta**: Incorporar o HLA OMT para incluir o atributo de **níveis de fidelidade ("fidelity level")** para os federados contendo as informações dos recursos de comunicação necessários em cada nível. O valor padrão é de 3 níveis de fidelidade. Seria permitido ampliar os níveis de fidelidade.

RunTime Infraestructure (RTI) provê serviços aos federados similares ou análogos aos serviços que um sistema operacional distribuído prove para suas aplicações. O Object Management é o grupo de serviços de RTI para criação, modificação e destruição de objetos e suas interações produzidas. Os serviços do gerenciador de distribuição de dados ("Data Distribution Management - DDM") da arquitetura HLA e seus mecanismos associados gerenciam a troca de informações entre federados e/ou

federações. Estes serviços devem ser utilizados pelos federados agregados para a redução da transmissão e recepção de dados irrelevantes entre eles.

Proposta: Implementar o serviço de **gerenciamento dos níveis de prioridade ("Fidelity Level Management" - FLM)** responsável pela definição dos níveis de prioridade que cada federado deve operar em função dos critérios (distância, velocidade, letalidade, etc.) definidos pelo usuário para aquela federação ou sistema.

Com a implementação de serviços propostos neste trabalho ou similares, a arquitetura HLA proveria o usuário de importantes mecanismos para o desenvolvimento e implementação de sistemas de tempo real em ambientes complexos.

CAPÍTULO 8

CONCLUSÕES, SUGESTÕES E RECOMENDAÇÕES PARA TRABALHOS FUTUROS

8.1 Conclusões

Este trabalho teve como objetivos: i) desenvolver uma nova estratégia e uma nova família de algoritmos de agendamento/escalonamento de tarefas (“schedulers”) para simulação paralela/distribuída de processos em tempo real críticos com arquiteturas modernas do tipo HLA considerando e incluindo novas variáveis de decisão, tais como diferentes níveis de fidelidade e/ou resolução dos modelos simulados; e, ii) analisar e comparar a nova família de algoritmos e os seus impactos nos serviços disponíveis na arquitetura HLA, sugerindo melhorias e as respectivas alterações necessárias nessa arquitetura para a incorporação desta nova família de algoritmos aqui proposta.

No Capítulo 2 foram desenvolvidos os conceitos básicos em diversas áreas de conhecimento para o desenvolvimento do trabalho proposto. Essa atividade mostrou-se essencial para formação de uma visão sistêmica e de integração entre as áreas de modelagem, simulação, simulação de sistemas dinâmicos, simulação discreta baseada em eventos, simulação paralela e distribuída, computação de tempo real, programação, linguagens de programação para sistemas de tempo real, sistemas operacionais de tempo real, programação, engenharia de software, redes de computadores, sistemas de comunicação, sistemas de controle, sistemas de defesa, sistemas de treinamento, integração e testes de sistemas e certificação de sistemas segundo regulamentos e normas aplicados nas áreas de Aeronáutica, Espacial e de Defesa. Durante o desenvolvimento deste trabalho percebeu-se um distanciamento entre essas diversas áreas, cada uma com seu próprio linguajar, em diferentes níveis, embora todas elas sejam importantes para o ciclo de desenvolvimento de sistemas de tempo real críticos complexos. A integração e nivelamento do conhecimento entre essas áreas são essenciais para o desenvolvimento da engenharia e indústria aeroespacial na atualidade.

No Capítulo 3 foi feita uma revisão bibliográfica complementar à apresentada no Capítulo 2. Constatamos a semelhança entre os problemas enfrentados na área de sistemas de tempo real críticos nas décadas de 70 e 80 com os problemas enfrentados atualmente no desenvolvimento de sistemas de tempo real complexos, distribuídos ou não.

No Capítulo 4 foram discutidos e esclarecidos que os teoremas fundamentais da teoria dos programas de agendamento/escalonamento de tarefas proposto por Liu e Layland para os agendadores de taxa constante (RMS) e primeiro prazo fatal primeiro (EDF) são teorias sólidas para simulação e controle de tempo real. Esses algoritmos são amplamente conhecidos e discutidos na literatura de tempo real, mas neste trabalho: 1) nós discutimos e esclarecemos alguns aspectos da proposição original e nós melhoramos tais proposições explicitamente reescrevendo algumas de suas passagens ou transformações incompletas ou mesmo escondidas; e 2) nós discutimos e esclarecemos alguns erros cometidos em algumas contraprovas ou críticas, substituindo os mesmos por novos exemplos que confirmam os algoritmos e ilustram os pontos discutidos e esclarecidos.

No Capítulo 5 foi apresentada a prova da existência e propriedades da superfície de agendabilidade máxima utilizando recursos gráficos e a proposição de um novo critério do final parcial para agendabilidade aproximada. Foram desenvolvidos os fundamentos teóricos e apresentadas as seguintes novas estratégias de agendamento de tarefas incluindo a proposição e variação de níveis de fidelidade das tarefas: 1) extensão dinâmica do critério de taxa constante com variação dos níveis de fidelidade das tarefas; 2) extensão dinâmica do critério de taxa constante com sintonia dos períodos das tarefas e com variação dos níveis de fidelidade das tarefas; 3) extensão do critério primeiro prazo fatal primeiro com variação dos níveis de fidelidade das tarefas; e, 4) formulação de um critério dinâmico adequado a sistemas de tempo real com variação dos níveis de fidelidade das tarefas, de forma a garantir as características de tempo real críticos. As estratégias propostas são a base teórica dos algoritmos de agendamento propostos e testados no Capítulo 6.

No Capítulo 6 foram apresentados os novos algoritmos para agendamento de tarefas adequados a ambientes dinâmicos e complexos com milhares e até milhões de tarefas baseados em uma nova estratégia, que permitam aos sistemas manterem as suas características de tempo real críticos (“hard real time systems”). Os novos agendadores propostos são: 1) o algoritmo agendador dinâmico a taxa constante estendido; 2) o algoritmo agendador dinâmico com sintonia dos períodos das tarefas; 3) o algoritmo agendador dinâmico com primeiro prazo fatal primeiro estendido; e, 4) o algoritmo agendador dinâmico para sistemas de tempo real críticos.

Em modos normais de operação os agendadores RMS estendido e sintonizado são mais rápidos que os agendadores EDF estendido e crítico, mas, do ponto de vista prático, isto não representa nenhuma vantagem, pois o processador apresenta tempo livre nestas mesmas condições. A seleção do melhor algoritmo a ser utilizado fica fortemente relacionada com a complexidade de implementação e com os atrasos de tempo considerados.

Em modos de operação com sobrecarga moderada ou excessiva os agendadores RMS estendido e crítico são mais lentos e apresentam um fator de utilização menor quando comparados aos agendadores sintonizado e EDF estendido. Tanto o agendador sintonizado ou EDF estendido apresentam um fator de utilização do processador próximo ao máximo possível. Por outro lado o agendador sintonizado apresenta restrições de projeto devido aos requisitos das relações das tarefas e o EDF estendido exige a implementação de um despachador dinâmico e, portanto, mais complexo.

Em modos de operação com sobrecarga excessiva o agendador RMS estendido e o crítico reduzem o nível de fidelidade de um número maior de tarefas pois os seus limites de fator de utilização do processador são menores que aqueles utilizados pelos agendadores sintonizado e EDF estendido (próximos de 100%).

Todos os algoritmos propostos garantem o agendamento das tarefas utilizando para tal, o mecanismo de redução dos níveis de fidelidade das tarefas. O custo desta garantia é exatamente a complexidade no projeto de sistemas com a preparação dos respectivos

níveis de fidelidade para cada tarefa. Essa é uma alternativa viável para o desenvolvimento de sistemas de tempo críticos em ambientes dinâmicos.

O agendador RMS estendido é o que apresenta um maior custo computacional entre todos os agendadores propostos. Embora não tenhamos testado os agendadores sob este ponto de vista específico, a complexidade computacional adicional neste caso, indica que o ganho de processamento do mesmo em relação ao agendador crítico pode ser anulado pela própria complexidade daquele custo computacional. É necessária uma análise deste aspecto em relação às condições do processador onde os referidos algoritmos serão implementados.

O agendador sintonizado apresenta um custo computacional bem superior ao EDF estendido enquanto este necessita de um despachador mais complexo. Este custo computacional da maior complexidade do despachador do EDF, em geral, é inferior ao custo computacional acrescido no critério utilizado pelo agendador sintonizado.

Os resultados das simulações efetuadas indicam, como previsto, que o número de sustações de tarefas para o despachador do agendador EDF estendido é inferior ao número de sustações quando utilizamos o despachador do agendador RMS sintonizado. A diferença aumenta à medida que o número de tarefas m aumenta. Para uma comparação das vantagens de se utilizar um agendador baseado em um ou outro tipo de despachador devemos fazer a análise em função do ambiente computacional no qual iremos implementar o programa de sistema de tempo real, considerando todos os tempos de atrasos envolvidos. Em ambientes complexos, devido ao próprio número de tarefas e ao número de sustações para todos os algoritmos podemos afirmar que a agendabilidade é bastante sensível quanto à técnica e forma de implementação em baixo nível dos mesmos agendadores e despachadores.

Considerando: i) os resultados expostos acima; ii) o fato que os sistemas de tempo real críticos são sistemas de controle ou sistemas controlados; iii) a obrigatoriedade prática de sistemas de tempo real críticos reagirem a solicitações assíncronas; e, iv) os rigores para o desenvolvimento e certificação de sistemas de tempo real críticos (“hard” ou “soft”), acreditamos que o algoritmo crítico proposto é o mais adequado para este tipo

de aplicação. Durante o desenvolvimento deste trabalho constatamos em diversos trabalhos apresentados e discutidos em conferências organizadas pelo AIAA ou pelo DoD que as principais aplicações destes agendadores, na área aeroespacial, seriam o controle de tráfego aéreo para vôo livre, e a integração de ambientes de defesa complexos.

No Capítulo 7 propomos as alterações necessárias para a incorporação desses agendadores na arquitetura HLA: 1) a criação de um atributo de nível de fidelidade para cada federado relacionado ao tempo de processamento e tempo de comunicação com a RTI; e, 2) e a criação de um novo serviço de gerenciamento do nível de fidelidade dos federados (“Fidelity Level Management”).

8.2 Sugestões e Recomendações Para Trabalhos Futuros

Considerando principalmente a escassez de recursos humanos e materiais para a pesquisa e desenvolvimento no Brasil iremos concentrar as nossas sugestões de trabalhos futuros nas principais aplicações de interesse da indústria aeroespacial: o controle de tráfego aéreo para vôo livre; integração e desenvolvimento de ambientes de defesa complexos; e pesquisas de técnicas de coletas de detritos espaciais através da simulação de ambientes complexos com centenas de tarefas.

Na área de controle de tráfego aéreo para vôo livre, uma realidade na próxima década e um mercado gigantesco a ser explorado, sugerimos o desenvolvimento de ambientes de simulação de aeroportos e espaços aéreos baseados na arquitetura HLA e a sua respectiva integração com elementos reais, simulados virtualmente por computadores ou simulados realmente por equipamentos (simuladores aviônicos, treinadores de procedimento, rigs de integração, estações de planejamento, etc.). Um projeto piloto desta natureza tem a capacidade de expor os diferentes problemas das diferentes áreas do conhecimento forçando a comunicação e integração dos diversos grupos de conhecimento fundamentais no desenvolvimento da indústria aeroespacial nacional.

Devido à extensão territorial brasileira, os projetos integrados de defesa tendem cada vez a ficarem maiores e mais complexos. Com a entrada em operação das novas

aeronaves e sistemas já adquiridos pelo governo brasileiro iremos enfrentar uma série de novos problemas e dificuldades. Por outro lado, a integração entre sistemas e subsistemas é uma exigência maior a cada dia que passa. Os mesmos ambientes de modelagem, simulação, integração, testes, validação, certificação e marketing aplicados no projeto de espaço aéreo podem e devem ser utilizados para a pesquisa e desenvolvimento operacional com o desenvolvimento de novos algoritmos de interceptação, comando e controle da defesa nacional. A integração dos diversos simuladores do Ministério de Defesa seja do Exército, da Marinha ou da Aeronáutica, traria benefícios de maior qualidade no treinamento operacional das equipes com redução dos custos de manutenção e operação destes mesmos sistemas. Além disso, permitiriam a especificação, através da simulação, de novos sistemas de maior nível de complexidade e integração.

Também devido à sua extensão territorial e à sua privilegiada posição geográfica, a evolução do estudo dos detritos espaciais (“space debris”) e a formulação das novas políticas de operações de satélites no futuro são de importância estratégica para o futuro do nosso país. Em grande parte, as mesmas ferramentas e ambientes necessários às pesquisas anteriores, podem e devem ser reutilizados para a simulação e pesquisa nesta área.

Aos novos pesquisadores e desenvolvedores que forem trabalhar nessas áreas e, que tiverem especial interesse pela teoria dos agendadores, recomendamos o uso, o mais cedo possível, de simulações em ambientes discretos baseados em eventos. As planilhas, incluindo suas diferentes cores são essenciais para a formação de idéias, fixação de conceitos e validação de modelos com pequeno número de tarefas; entretanto são inadequadas e impraticáveis para ambientes complexos. Só a simulação discreta baseada em eventos nos possibilita a avaliação e computação dos atrasos envolvidos no agendamento e despacho de tarefas.

Considerando: i) a importância estratégica do desenvolvimento e integração de complexos sistemas de tempo real para as indústrias e centros da região; ii) as dificuldades financeiras para investimentos de longo prazo em nosso país; iii) a escassez

de recursos humanos em número e qualidade quando comparado com as grandes potências mundiais; recomendamos um esforço coordenado entre a empresas da região, os centros de pesquisa e desenvolvimento, em particular do INPE e CTA, com o apoio das agências de pesquisa e desenvolvimento, para organização e implantação de grupos de pesquisas nas áreas sugeridas neste trabalho como uma forma de resguardar interesses da indústria nacional aeroespacial e de defesa.

REFERÊNCIAS BIBLIOGRÁFICAS

Banks, J. (editor). **Handbook of simulation**. New York: John Wiley & Sons, Inc., 1998. 849 p.

Banks, J.; Carson II, J. S. **Discrete-event system simulation**. New Jersey: Prentice Hall, 1984. 514 p.

Burns, A.; Wellings, A. **Real-time systems and their programming languages.** Addison-Wesley, Reading, MA, USA, 1990, 575p.

Ceranowicz, A; Torpey, M; Helfinstine, B; Evans, J; Hines, J. Reflections on building the joint experimental federation. In: I.ITSEC Conference, Orlando, 2003. **Proceedings…** Orlando: NTSA, 2003.

Comer, D. E. **Interconnecting with TCP/IP principles, protocols, and architectures – vol. 1.** New Jersey: Prentice Hall, 2000. 750 p.

Dahman, J. S.; Fujimoto, R. M.; Weatherly, R. M. The Department of Defense High Level Architecture. In: Winter Simulation, Colorado, 1997. **Proceedings…** New York: Association for Computing Machinery, 1997. p. 142-149.

Devillers, R.; Goossens, J. **Liu and Layland's Schedulability Test Revisited,** Information Processing Letters 73, 2000. 5 p.

Delong, C. (Editor). **Mil-Std-1553 databus systems integration handbook (SAE-12).** Washington: Society of Automotive Engineers, 1991.

Farines, J.M.; Fraga, J.S.; Oliveira, R. S. **Sistemas de Tempo Real.** São Paulo, Brasil: IME-USP, 2000. 201p.

Fishwick, P. A. Web-based simulation: some personal observations. In: WINTER SIMULATION CONFERENCE, 1996, Coronado. **Proceedings...** New York: Association for Computing Machinery, 1996. p. 772-779.

Franklin, G. F.; Powell, D. J.; Workman, M. L. **Digital control of dynamic systems.** Portland: Book News Inc, 1985.

Fujimoto, R. M. **Parallel and distributed simulation systems.** New York: John Wiley & Sons, 2000. 300 p.

Fujimoto, R. M. Ten years of PADS: where we've been, where we're going. In: Workshop on Parallel and Distributed Simulation (PADS '96), 10., San Diego, 1996. **Proceedings…** New Jersey: IEEE Press, 1996.

Fujimoto, R. M. Zero Lookahead and repeatability in the high level architecture. In: SPRING SIMULATION INTEROPERABILITY WORKSHOP, 1997, Orlando. **Proceedings…** San Diego: Society for Computer Simulation, 1997.

Furht, B.; Grostick, D.; Gluch, D.; Rabat, G.; Parker, J.; McRoberts, M. **Real-time unix systems: design and application guide**. Boston: Kluwer Academic Publishers, 1995. 316 p.

Institute for Electrical and Electronic Engineers (IEEE). **IEEE standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federation Interface Specification**. New York, NY, USA: IEEE Press, 2000. 129 p.

Institute for Electrical and Electronic Engineers (IEEE). **IEEE standard for Modeling and Simulation (M&S) HIGH Level Architecture (HLA) - Object Model Template (OMT) specification**. New York, NY, USA: IEEE Press, 2000. 466 p.

Institute for Electrical and Electronic Engineers (IEEE). **IEEE standard for Modeling and Simulation (M&S) HIGH Level Architecture (HLA) - Framework and Rules**. New York, NY, USA: IEEE Press, 2000. 21 p.

Integrated Systems. **MATRIXx product family user's manual**. Santa Clara, 1995.

Johansson, R. **System modeling and identification.** Englewood Cliffs: Prentice Hall, 1993. 512 p.

Kuhl, F. K.; Weatherly, R. and Dahmann, J. **Creating computer simulation systems**. Englewood Cliffs: Prentice Hall, 2000. 212 p.

Linkens, D. A. (ed.). **CAD for control systems**. New York: Marcel Dekker, 1993. 584 p.

Liu, C. L.; Layland, J.W. Scheduling algorithms for multiprogramming in hard-real-time environment. **Journal of the ACM**, v. 20, n.1, p. 46–61, 1973.16 p.

Ludwig, A; Wuerfel, R. DMSO RTI Evaluation. In: I.ITSEC CONFERENCE, 2001, Orlando. **Proceedings**. Orlando: NTSA, 2001.

Martin, J. **Data communication technology**. Englewood Cliffs: Prentice Hall, 1995. 691 p.

Ogata, K. **System dynamics**. Englewood Cliffs: Prentice Hall, 1993. 512 p.

Page, E. H. Beyond Speedup: PADS, the HLA and web-based simulation. In: WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION, 13., 1999, Atlanta. **Proceedings...** San Diego: Society for Computer Simulation, 1999.

Page, E. H.; Buss, A.; Fishwick, P. A.; Healy, K.; Nance, R. E.; Paul, R. J. Web-Based Simulation: Revolution or Evolution? **ACM Transactions on Modeling and Computer Simulation**, v. 10, n. 1, 15p. 2000. New York: Association for Computing Machinery, 2000.

Page, E. H.; D. J. Medeiros, E. Watson, J. Carson and M. Manivannan. The Rise of Web-Based Simulation: Implications for the High Level Architecture. In: WINTER SIMULATION CONFERENCE, 1998, Washington,. **Proceedings…** San Diego: Society for Computer Simulation, 1998. p. 1663-1668.

Page, E. H.; Griffin, S. P.; Rother, S. L. Providing conceptual framework support for distributed web-based simulation within the high level architecture. In: SPIE: ENABLING TECHNOLOGIES FOR SIMULATION SCIENCE II, 1998, Orlando. **Proceedings…** San Diego: Society for Computer Simulation, 1998. p. 287-292.

Page, E. H.; Opper, J. M. Investigating the application of web-based simulation principles for a next-generation computer generated forces model. **Future Generation Computer Systems.** Upper Saddle River, USA: Prentice Hall, Jan. 1999.

Papini, M; Barracos, P. Real-time simulation control and HIL with COTS computing clusters. In: AIAA GNC, AFM, AND MST CONFERENCES, 2000, Denver. **Proceedings...** Denver: AIAA, 2000.

Prudêncio, S. V. **Simulação digital em tempo real de um sistema de controle de atitude magnético autônomo de um satélite**. São José dos Campos. 167 p. Dissertação (Mestrado em Engenharia e Tecnologias Espaciais) – Instituto Nacional de Pesquisas, 2000.

Rigby, W. H.; Dalby, T. **Computer interfacing**. Englewood Cliffs: Prentice Hall, 1995. 232 p.

Roza, M.; Gool, P.V. Simulating free flight in hla federations. In: AIAA GNC, AFM, AND MST CONFERENCES, 2000, Denver, CO, USA. **Proceedings…** Denver: AIAA, 2000.

SBS Avionics Technologies. **ARINC User's Manual**. Albuquerque: SBS Avionics Technologies, 2001. 214 p.

Shearer, J.L.: Murphy, A. T., Richardson, H. H. **Introduction to system dynamics**. Reading: Addison-Wesley Pub. Co., 1967.

Smetana, F. O. **Flight vehicle performance and aerodynamic control.** Reston: American Institute of Aeronautics and Astronautics, Inc, 2001. 359 p.

Takahashi, Y.; Rabins, M. J.; Auslander, D. M. **Control and dynamic systems.** Reading: Addison-Wesley Pub. Co., 1970.

The Mathworks, Inc. **MATLAB product family manuals**. Natick, 1996.

Torpey, M., Helfinstine, B. and Evans, J. Reflections on building the joint experimental federation, 2002, Orlando. Proceedings of IITSEC 2002 Conference, Orlando, FL, USA.

Trivelato, G. C; Souza, M. L. O. Comparing MATRIXx and MATLAB for Modeling, Designing and Simulating Flight Control Systems. In: AIAA GNC, AFM, AND MST CONFERENCES, Denver, 2001. **Proceedings**. Quebec: AIAA, 2001.

Trivelato, G. C; Souza, M. L. O. Comparando MATRIXx and MATLAB para modelagem simulação de veículos. In: COBEM, 2001, Uberlândia. **Proceedings...** Uberlândia: ABCM, 2001.

Trivelato, G. C. **Comparação de ambientes de modelagem e simulação**. São José dos Campos. 167 p. INPE-9622-NTC/355 (Nota técnica) – Instituto Nacional de Pesquisas, 2003. 16p.

Trivelato, G. C. **Protocolos de redes para ambientes de distribuída**. São José dos Campos. 167 p. INPE-9622-NTC/356 (Nota técnica) – Instituto Nacional de Pesquisas, 2003. 27p.

Trivelato, G. C. **Simulação distribuída:** o que é e qual o potencial da "HLA" e "WEB simulation". São José dos Campos. 167 p. INPE-9622-NTC/357 (Nota técnica) – Instituto Nacional de Pesquisas, 2003. 26p.

Trivelato, G. C. **Técnicas de modelagem e simulação de sistemas dinâmicos.** São José dos Campos: INPE, 2003. 167 p. (INPE-9622-NTC/358) (Nota técnica).

Trivelato, G. C; Souza, M. L. O. Improving the rate monotonic and the first deadline first schedulers for real time simulation and control of aerospace vehicles. In: AIAA GNC, AFM, AND MST CONFERENCES, 2003, Austin. **Proceedings…** Quebec: AIAA, 2003.

Trivelato, G. C; Souza, M. L. O. Simulators and simulations: their characteristics and applications to the simulation and control of aerospace vehicles. In: SAEBrasil, 2003, São Paulo. **Proceedings**. São Paulo: SAE, 2003.

U. S. Defense modeling and simulation office. **1993 DMSO survey of semi-automated forces.** Alexandria: DMSO, 1993.

U. S. Department of Defense. **High level architecture object model template, version 1.3.** Washington: DOD, 1998.

U. S. Department of Defense. **High level architecture rules, version 1.3.** Washington: DOD, 1998.

U. S. Department of Defense. **Modeling and simulation glossary**. Washington: DOD, 1998.

U.S. Defense Modeling and Simulation Office. **1993 DMSO survey of semi-automated forces.** Alexandria: DMSO, 1993.

White, F. M. **Fluid mechanics.** New York: McGraw-Hill Book Company, 1979. 701 p.

Zeigler, B. P. **Object-oriented simulation and hierarchical, modular models**. Orlando: Academic Press, 1990.

Zeigler, B. P. Praehofer, H. and Kim, T. G.; **Theory of Modeling and Simulation - Second Edition**. New York: Academic press, 2000. 510 p.

zeigler, b. p. **theory of modeling and Simulation**, New York: John Wiley, 1976.

APÊNDICE A

O AMBIENTE DE SIMULAÇÃO DISCRETA PROMODEL E O MODELO IMPLEMENTADO

A.1 O Ambiente de Simulação Discreta ProModel

O ProModel é um ambiente de modelagem e simulação discreta desenvolvido especificamente para suportar o projeto e operação de sistemas de produção mas pode ser utilizado para a simulação discreta de uma forma geral. No ProModel um sistema de produção é visto como um arranjo de locais de processamento como máquinas ou estações de trabalho onde entidades como partes são processadas. Um sistema também deve incluir recursos de suporte tais como operadores ou equipamentos de manipulação de material para colaborar no processamento ou no movimento das entidades.

ProModel permite a modelagem de eventos aleatórios incluindo chegadas aleatórias e falhas aleatórias de equipamentos. A lógica de processamento e outras lógicas de decisão podem ser baseadas em uma das muitas regras fornecidas ou em lógicas definidas durante o modelamento.

Durante a simulação, o ProModel mostra uma representação animada do sistema e coleta as estatísticas das medidas de desempenho que serão mais tarde tabuladas e apresentadas em gráficos.

O ProModel é constituído por uma base de elementos já previamente modelados. Esses elementos são utilizados para representar a estrutura e operação do sistema a ser simulado. Os elementos básicos são divididos naqueles que definem os objetos do sistema, naqueles que definem a operação do sistema e naqueles usados para especificar parâmetros ou lógicas de operação.

A.1.1 Objetos de Sistema (“System Objects”)

Os objetos de sistema são divididos em quatro categorias: locais, entidades, recursos e rotas.

a) Locais (“locations”):

Os locais representam pontos fixos no sistema onde as entidades são enviadas para processamento, fila, ou para tomada de decisão de um futuro encaminhamento. Locais devem ser utilizados para modelar elementos tais como estações de trabalhos, locais de depósito e de transferência. Locais opcionalmente têm restrições de capacidade e podem ser tanto locais unitários (máquina capaz de processar uma ou mais partes de cada vez) ou locais multi-unitários (máquinas paralelas efetuando a mesma operação).

Os locais devem ser representados por locais gráficos na janela de apresentação. Eles podem ter atributos e atrasos agendados ou não. Cada local tem regras definidas para selecionar as entidades que chegam e para enviar as entidades de saída.

Dois locais especiais que permitem o movimento de entidades assim como o processamento da os depósitos e as filas.

b) Entidades (“entities”):

Entidades são itens que são processados pelo sistema, tais como produtos, suporte, ou até mesmo formulários. Entidades podem ser agrupadas para formar uma nova entidade, divididas em duas ou mais entidades ou até mesmo convertidas em uma ou mais novas entidades. Entidades podem ter velocidade, dimensões ou outros atributos definidos pelo usuário.

Entidades movem de um local para outro usando o caminho definido pela rota ou o caminho definido pela rede. O movimento pode requisitar o uso de um recurso.

c) Recursos (“resources”):

Recursos podem ser pessoas ou equipamentos necessários para transportar entidades, realizar operações ou para realizar manutenção em locais ou outros recursos.

Um recurso consiste em uma ou mais unidades com características comuns tais como um grupo de trabalhadores ou uma frota de caminhões. Cada recurso móvel deve ser associado a uma rota na malha para mover entre os locais. Recursos também podem ter tempo de atraso. Os recursos podem ser dinâmicos ou estáticos.

d) Rotas (“path networks”):

Rotas devem ser usadas para definir um caminho a ser seguido por uma entidade ou por um recurso entre locais. Rotas são ser pessoas ou equipamentos necessários para transportar entidades, realizar operações ou para realizar manutenção em locais ou outros recursos. As rotas são opcionais a menos que existam recursos dinâmicos no modelo. Um modelo pode ter muitas rotas. Um ou mais entidades ou tipos de recursos podem compartilhar a mesma rota. O movimento ao longo da rota deve ser definido em termo de tempo ou pela velocidade e distância (opção padrão).

A.1.2 Operações do Sistema (“System Operation”)

O sistema possui os seguintes tipos de operação: processos, chegadas, atrasos e trocas.

a) Processos (“processes”):

Processos definem o encaminhamento de entidades através do sistema e quais operações acontecem para cada entidade em cada local. Encaminhamento de entidades é uma capacidade restrita de forma que nenhuma entidade será capaz de reivindicar ou mover para o próximo local até que a capacidade esteja disponível naquele local.

Antes de o usuário especificar a lógica de processamento ele deve criar todos os locais e entidades que devem ser usados no processamento. Se, entretanto, ele referenciar um local ou uma entidade que ainda não foi criada, ele será forçado a adicioná-lo(a) no local corrente ou na lista de entidade. Posteriormente, ele pode ir ao referido módulo e definir o símbolo gráfico para o novo local ou nova entidade.

b) Chegadas (“arrivals”):

Chegadas definem o ponto de entrada das entidades no sistema. Podem ser atribuídas entidades de qualquer tipo em qualquer quantidade para chegar em um local, ou definir chegadas para ocorrer em tempos agendados, em intervalos periódicos, aa taxas crescentes ou decrescentes que sigam um roteiro repetitivo, ou como o resultado de um

evento em um modelo. Locais de chegada descartam entidades que excedem a capacidade disponível no local.

c) Atrasos (“downtimes”):

Um atraso torna um local ou um recurso inoperativo ou fora de serviço de forma que nenhuma função esteja disponível. Atrasos podem representar interrupções agendadas tais como deslocamentos, paradas ou manutenção agendada; ou eles podem representar interrupções não agendadas ou aleatórias tais como falhas de equipamentos. Atrasos podem ter prioridades sustadas ou não sustadas e podem solicitar o uso de um ou mais recursos para manutenção ou reparação.

d) Trocas (“shifts”):

A troca é um intervalo de tempo durante o qual o local ou recurso é “obrigado a parar”. Um ou mais intervalos podem ser definidos para ocorrer durante uma troca. Trocas ou paradas são definidos para a semana. Quando usando trocas na simulação, a duração da simulação deve ser especificada em termos de uma data inicial e final. Se um particular local ou recurso não tem uma troca endereçada, este local ou recurso está sempre disponível, exceto pelos seus próprios atrasos.

A.1.3 Elementos lógicos (“Logic Elements”)

Os elementos lógicos provêm várias maneiras de controlar o comportamento dos elementos modelados (locais, processos, etc.). São suportados os seguintes elementos lógicos: Variáveis (“Variables”), atributos (“Attributes”), vetores (“Arrays”), funções de sistema (“System functions”), funções matemáticas (“Math functions”), distribuições (“Distributions”), tabelas de função (“Function Tables”), sub-rotinas (“Subroutines”), macros (“Macros”), números indexados (“Name-index numbers”), expressões (“Expressions”) e declarações (“Statements”).

Todos os modelos implementados no ProModel devem conter pelo menos componentes dos seguinte tipos: locais, entidades, processos e chegadas.

A.2 O Modelo Implementado

Para o desenvolvimento teórico dos algoritmos propostos na literatura e em particular neste trabalho foi assumido que o tempo de chaveamento de contexto era desprezível ou praticamente nulo e, portanto não considerado. Para um conjunto de tarefas muito grande, que é a área específica de nosso interesse, essa assunção deve ser verificada em tempo de projeto. Desta forma torna-se necessário o cálculo do número de chaveamento de contexto ocorrido no processador quando utilizamos os algoritmos propostos para um determinado conjunto de tarefas. O cálculo analítico deste número é impossível. As formas restantes para tal são o uso de planilhas ou simulação discreta. Como o uso de planilhas é impraticável, a solução possível e viável e o desenvolvimento de um modelo de simulação discreta que incluía as funções desenvolvidas pelo despachador e pelo processador.

O modelo foi desenvolvido em duas partes distintas. A primeira, em ambiente de planilha, com a implementação dos diversos algoritmos propostos, tem como resultado a lista de tarefas agendadas. A segunda, em ambiente de simulação discreta ProModel, com a implementação do despachador e do processamento das tarefas no processador. A transferência dos resultados da primeira para a segunda parte e feita através de acesso de leitura que o modelo no ambiente ProModel faz em planilhas Excel (recurso existente no ambiente).

A primeira parte implementa exatamente os quatro algoritmos propostos no Capítulo 6 deste trabalho. A segunda parte é descrita a seguir e está representada na FIGURA A. 1.

O nosso modelo tem 36 locais de chegada e, consequentemente, possui capacidade para despachar e executar até trinta e duas tarefas. Os locais de chegada são numerados de 1 a 36 e recebem as chegadas das tarefas agendadas nas suas respectivas freqüências. Cada tarefa recebe os atributos de período e de tempo do próximo ciclo (“deadline”).

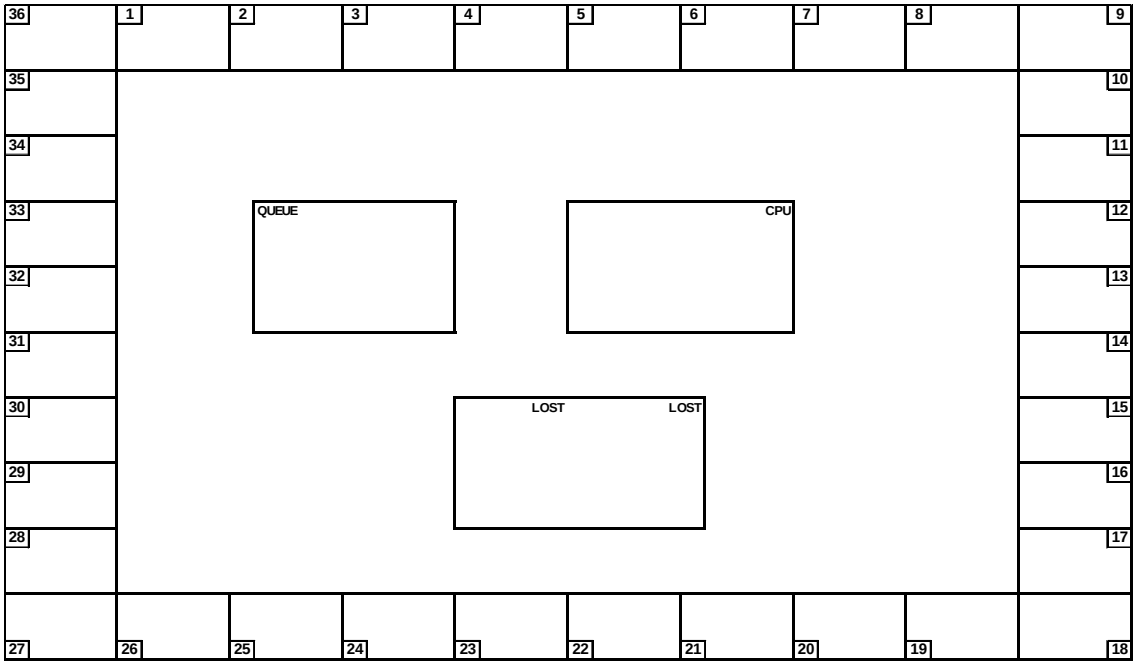

FIGURA A. 1 – Modelo implementado do despachador e processador.

Todas as tarefas são granuladas em unidades de tempo e entram em uma fila única de pronto (local de depósito identificado por QUEUE). Neste local, elas são encaminhadas para o processador (local CPU) ordenadas de acordo com o atributo. O atributo utilizado no caso de taxa monotônica (“rate monotonic - RMS”) é o período e, no caso de prazo fatal primeiro (“earliest deadline first - EDF”), tempo do próximo ciclo. Quando duas solicitações de uma mesma tarefa acontecer neste local, as unidades de tempo ainda não processadas são encaminhadas para um local de depósito (LOST) e uma solicitação desta tarefa é anotada como não atendida. O local processador (CPU) executa uma unidade de tempo da tarefa e devolve a mesma para a fila de pronto sem nenhum atraso. Na próxima unidade de tempo o processador recebe novamente a tarefa de maior prioridade conforme critério apropriado (RMS ou EDF). No processador é registrado o número de chaveamento de contexto que ocorre quando uma tarefa é suspensa: o resultado que representa o número de unidades de tempo desprezadas.