TESE
1054

Anderson Marcos Henriques

Algoritmo de Escalonamento de Comunicação para o Fieldbus da Fieldbus Foundation

Dissertação apresentada à Escola Federal de Engenharia de Itajubá para obtenção do título de Mestre em Engenharia.

Área de Concetração:
Engenharia Elétrica
Orientadora:
Prof. Dra. Lùcia Regina Horta Rodrigues Franco

Itajubá
2000

Anderson Marcos Henriques

Algoritmo de Escalonamento da Comunicação para o Fieldbus da Fieldbus Foundation

Dissertação apresentada à Escola Federal de Engenharia de Itajubá para obtenção do título de Mestre em Engenharia.

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca
MAUÁ
EIM

Itajubá

2000

Agradeço aos meus pais Manoel e Elizabeth, a meus irmãos Sandra e Leandro, e a minha esposa Marília pelo apoio e incentivo, na busca de meus ideais e objetivos.
Agradeço a minha orientadora Prof. Dra. Lúcia Regina Horta Rodrigues Franco, pela sua dedicação em me ajudar a compreender a teoria por trás de um tema que é extremamente novo para a área de automação. Além de me estimular e ampliar meu aproveitamento oriundo da execução deste trabalho.
A SMAR e seus engenheiros pela apoio e dedicação para troca de informações.
Ao Projeto RECOPE - Subgrupo Fieldbus da Escola Politécnica da USP em cooperação com a EFEI, financiado pela FINEP, que proporcionou reuniões com especialistas da área.
E a todas as pessoas que de alguma maneira contribuíram, para que este trabalho terminasse com êxito.

ÍNDICE

CAPÍTULO 1 – INTRODUÇÃO 1

CAPÍTULO 2 – FIELDBUS 4

1. Benefícios do Fieldbus 5
2. Camadas de Comunicação 6
2.1. Camada Física 8
2.2. Pilha de Comunicação 9
2.2.1. Camada de Enlace de Dados (Data Link Layer – DLL) 9
2.2.1.1. Comunicação escalonada ou cíclica 11
2.2.1.2. Comunicação não-escalonada ou acíclica 12
2.2.1.3. Tarefas realizadas pelo LAS 12
2.2.1.3.1. Envio do CD no tempo programado 13
2.2.1.3.2. Manutenção da lista viva 13
2.2.1.3.3. Sincronização do tempo da camada de enlace de dados 14
2.2.1.3.4. Passagem de *token* 15
2.2.1.3.5. Redundância do LAS 15
2.2.2. Sub-camada de Acesso Fieldbus (Fieldbus Access Sublayer – FAS) 15
2.2.2.1. Serviço cliente/servidor (Client/Server VCR) 16
2.2.2.2. Serviço de distribuição de relatório (Report Distribution VCR) 17
2.2.2.3. Serviço publisher/subscriber (Publisher/Subscriber VCR) 18
2.2.3. Camada de Especificação de Mensagem fieldbus (Fieldbus Message Specification – FMS) 19
2.3. Camada do Usuário ou de Aplicação do Usuário 19
2.3.1. Bloco de Recurso 20
2.3.2. Bloco Funcional 20
2.3.3. Bloco Transdutor 20
3. Gerência do Sistema 21
3.1. Escalonamento de Bloco Funcional 21

CAPÍTULO 3 – O PROBLEMA 25

1. Abordagem Utilizada 26
2. A Comunicação no Fieldbus 28
2.1. Fatores importantes para a comunicação no fieldbus 29
3. Algoritmo de escalonamento 35

CAPÍTULO 4 – ANÁLISE DO ESTADO DA ARTE 36

1. Trabalhos Relacionados 37
1.1. Escalonamento de Processos em Um Único Processador 38
1.2. Escalonamento de Processos em Múltiplos Processadores 39
2. Paradigma do Escalonamento Estático x Dinâmico 41
3. Descrição Sucinta do Algoritmo Proposto por este Trabalho 44
3.1. Discussão sobre a Abordagem Teórica Feita nos Trabalhos Anteriores Utilizada pelo Algoritmo Proposto 44
3.1.1. Comunicação Entre Dispositivos 44
3.1.2. Ciclo Operacional do Sistema 44
3.1.3. Interoperabilidade entre Aplicações 45

CAPÍTULO 5 – ALGORITMO PROPOSTO **46**

1. Unicidade do Algoritmo Proposto 46
2. Funcionalidade Global do Algoritmo Proposto 47
3. Tópicos Importantes sobre o Comportamento do Algoritmo Proposto 48
4. Etapas do Algoritmo 53
   4.1. Percursos Alternativos em cada Estratégia de Controle 53
   4.2. Construção da Base de Dados 55
   4.3. Algoritmo Escalonador 60
      4.3.1. Obtenção da Árvore de Busca 60
      4.3.2. Escalonamento do Nó 64
   4.4. Tratamento dos Escalonamentos Obtidos 66
      4.4.1. Escalonamento dos Blocos Funcionais 67
      4.4.2. Escalonamento das Mensagens 67

CAPÍTULO 6 – RESULTADOS **68**

1. Programa Gerado para Apresentação dos Resultados 68
2. Exemplo de Algumas Aplicações Simples 70
   2.1. Exemplo 1 70
   2.2. Exemplo 2 73
3. Exemplo de uma Aplicação Complexa 77
4. Exemplo de uma Aplicação Composta 80
5. Resultados Obtidos e Discussão sobre a Eficiência Alcançada 82

CAPÍTULO 7 – CONTRIBUIÇÕES E TRABALHOS FUTUROS **85**

REFERÊNCIAS BIBLIOGRÁFICAS **87**

LISTA DE FIGURAS

Figura 1 – Relacionamento do modelo *Fieldbus* com o modelo OSI [9] 7
Figura 2 – Etapas da comunicação escalonada ou cíclica [9]..... 11
Figura 3 – Etapas da comunicação não escalonada ou acíclica [9] 12
Figura 4 – Adição de um dispositivo a Lista Viva [9] 14
Figura 5 – Serviço Client/Server [9] 17
Figura 6 – Serviço de distribuição de relatório [9] 18
Figura 7 – Serviço Publisher/Subscriber [9] 19
Figura 8 – Visualização das conexões lógicas de uma aplicação 22
Figura 9 – Escalonamento de mensagens e blocos funcionais 23
Figura 10 – Representação alternativa para o escalonamento 23
Figura 11 – Exemplo de uma aplicação de usuário 31
Figura 12 – Processo simples da estratégia de controle mostrada na figura 11 34
Figura 13 – Processo composto da estratégia de controle mostrada na figura 11 34
Figura 14 – Diagrama funcional de escalonamento centralizado estático baseado em tabela [1] 42
Figura 15 – Diagrama funcional de escalonamento centralizado dinâmico [1] 42
Figura 16 – Diagrama funcional de escalonamento descentralizado dinâmico 43
Figura 17 – Fluxograma utilizado por todos os percursos 54
Figura 18 – Fluxograma da função de construção da base de dados 56
Figura 19 – Exemplo de aplicação do usuário 58
Figura 20 – Hierarquia da árvore de busca 62
Figura 21 – Fluxograma utilizado para geração da árvore de busca 63
Figura 22 – Fluxograma para o escalonamento de um nó da árvore de busca 65
Figura 23 – Programa desenvolvido para visualização do escalonamento 69
Figura 24 – Configuração de uma aplicação contendo processos simples 70
Figura 25 – Simulação do escalonamento factível da aplicação da figura 24 72
Figura 26 – Configuração de uma aplicação contendo processos compostos 76
Figura 27 – Simulação do escalonamento factível da aplicação da figura 26 77
Figura 28 – Configuração de uma aplicação contendo uma maior quantidade de BFs e percursos 79
Figura 29 – Simulação do escalonamento factível da aplicação da figura 28 79
Figura 30 – Configuração de uma aplicação contendo estratégias de controle concorrentes 81
Figura 31 – Simulação do escalonamento factível da aplicação da figura 30 82
Figura 32 – Apresentação do tempo de processamento em função da quantidade de MSGs e BFs 84

LISTA DE TABELAS

Tabela 1 – Tabela de escalonamento [9] 22
Tabela 2 – Características temporais, relacionamentos e listas da aplicação mostrada na figura 24 72
Tabela 3 – Características temporais, relacionamentos e listas da aplicação mostrada na figura 26 76
Tabela 4 – Características temporais, relacionamentos e listas da aplicação mostrada na figura 28 80
Tabela 5 – Características temporais, relacionamentos e listas da aplicação mostrada na figura 30 81
Tabela 6 – Informações referentes a eficiência alcançada com o algoritmo proposto 83

LISTA DE SIGLAS

A/D	–	Conversão Analógica para Digital
AI	–	Entrada Analogica
AO	–	Saída Analógica
CD	–	Compel Data
D/A	–	Conversão Digital para Analogica
DD	–	device descriptions
DLL	–	Data Link Layer
DT	–	Live List Update/ Node Activation
FAS	–	Fieldbus Access Sublayer
FMS	–	Fieldbus Message Specification
HSE	–	High Speed Ethernet
ID	–	Identificador de dispositivo
ISO	–	Interconexão de Sistemas Abertos
LAN	–	Local Area Network
LAS	–	Link Active Scheduler
NMIB	–	Network Management Information Base
OSI	–	Open Systems Interconnect
PID	–	Proporcional Integral Derivativo
PN	–	Probe Node
PR	–	Probe Response
PT	–	Pass Token
RT	–	Return Token
SMIB	–	System Management Information Base
TD	–	Time Distribution
VCR	–	Virtual Communication Relationships
VFD	–	Virtual Field Device

RESUMO

O presente trabalho propõe a implementação de um escalonamento para a comunicação em um barramento compartilhado pelas mensagens trocadas entre os dispositivos de um sistema fieldbus, de acordo com a execução dos blocos funcionais distribuídos em tais dispositivos. O algoritmo proposto utiliza uma heurística EDF e a técnica Branch-and-Bound.

O sistema implementado é testado em configurações de aplicações industriais reais com resultados satisfatórios.

ABSTRACT

The present work implements a scheduler for the communication in a shared bus used by the messages exchanged among the devices of a fieldbus system, according to the execution of the functional blocks distributed in such devices. The proposed algorithm uses the EDF heuristic and the Branch-and-Bound technique.

Real industrial applications were tested by the scheduler and the results were satisfactory.

CAPÍTULO 1 – INTRODUÇÃO

Neste capítulo serão resumidos os pontos importantes deste trabalho, objetivando-se assim realçar as características inerentes ao problema tratado por este trabalho, o enfoque da dissertação e suas principais contribuições.

O problema

Um dos problemas presentes em um sistema *fieldbus*, consiste na obtenção de um escalonamento no barramento de comunicação compartilhado pelas mensagens trocadas entre dispositivos, e na obtenção de um escalonamento de blocos funcionais nos dispositivos distribuídos pelo *fieldbus*.

Estes dispositivos dependem do meio físico de comunicação para transmitir suas mensagens que são produzidas pela aplicação, a qual apresenta-se distribuída em um conjunto de dispositivos conectados ao barramento. Com a distribuição da aplicação no barramento, gera-se um outro problema devido à necessidade do escalonamento e sequencialização dos módulos que compõem a aplicação entre os dispositivos. Este escalonamento deve então, estar em conformidade com a sequencialização dos módulos da aplicação, e sincronizado com o escalonamento das mensagens que trafegam sobre o barramento (escalonamento da comunicação).

Portanto, em função das características impostas por este problema, pode-se observar a necessidade da sincronização e sequencialização dos blocos funcionais e mensagens, de tal forma que sejam atendidas ainda as restrições temporais presentes no sistema, tempos ociosos gerados no barramento, relações de precedência de mensagens e blocos funcionais, restrições fim-a-fim, e relacionamentos entre blocos funcionais e mensagens existentes em cada estratégia de controle que compõem a aplicação do usuário.

A dissertação

Os algoritmos clássicos não são bem sucedidos para solucionar este problema, porque, eles não otimizam o uso do barramento e ainda paralelamente o uso dos processadores dos dispositivos. Além de, não utilizarem relações de precedência, tempos ociosos, consistências temporais, restrições fim-a-fim, relacionamentos entre mensagens e blocos funcionais, entre outros, não levam em consideração a possibilidade de qualquer relacionamento de dois tipos de escalonamentos distintos e completamente relacionados, como é o caso do escalonamento

de um recurso compartilhado para a comunicação e o escalonamento de recursos distribuídos (processadores dos dispositivos) dependentes do escalonamento da comunicação no *fieldbus*.

Portanto, até o presente momento os algoritmos utilizados em automação industrial, não satisfazem todos os requisitos impostos pelas restrições temporais existentes no *fieldbus* para a garantia do escalonamento gerado *off-line*.

A Organização da dissertação

O respectivo trabalho, começa especificando e detalhando as características referentes a rede *fieldbus* da especificação *Fieldbus Foundation*, abordada no *Capítulo 2*.

O *Capítulo 3* descreve detalhadamente o problema que tratado por este trabalho.

Alguns trabalhos anteriores são descritos no *Capítulo 4* para que seja feito um posterior relacionamento com os procedimentos utilizados no desenvolvimento do algoritmo proposto. No *Capítulo 5* apresenta-se o algoritmo proposto, discutindo-se algumas considerações que foram feitas em relação ao problema.

No *Capítulo 6* são apresentados os resultados obtidos baseados em algumas aplicações práticas, utilizadas em indústrias que já utilizam o padrão *fieldbus* na comunicação de processos/plantas industriais. É também abordada, a eficiência alcançada com a utilização do algoritmo proposto.

Finalmente, no *Capítulo 7* conclui-se o trabalho ressaltando os pontos chaves que contribuíram para que o problema pudesse ser solucionado através das técnicas utilizadas, e algumas sugestões para futuros trabalhos.

Principais contribuições

Com a execução deste trabalho foram levantados os pontos relevantes para o desenvolvimento de um configurador da comunicação *fieldbus*, envolvendo o escalonamento das mensagens e dos blocos funcionais paralelamente.

Neste trabalho otimizou-se o uso dos processadores dos dispositivos conectados ao *fieldbus*, respeitando a temporização e sequenciamento dos blocos funcionais e mensagens existentes na configuração da aplicação do usuário. Além disso, garantiu-se o atendimento das restrições temporais impostas pelo sistema *fieldbus*, caracterizado pela mesma configuração da aplicação do usuário.

O procedimento utilizado e a forma de tratamento dos percursos contidos em uma configuração de aplicação de usuário, garantem o desenvolvimento de uma grande variedade de aplicações impostas pelos processos ou plantas industriais.

O procedimento utilizado para o escalonamento dos blocos funcionais em função do escalonamento das mensagens, proporcionou que aplicações mais complexas pudessem ser atendidas, viabilizando a aplicabilidade do escalonamento gerado. Ao contrário, se fossem feitos dois escalonamentos distintos, existiria a necessidade da verificação da compatibilidade dos dois escalonamentos, e se estes escalonamentos não fossem compatíveis, deveriam ser gerados novamente, i.e., o algoritmo deveria ser executado de novo até a obtenção de dois escalonamentos compatíveis.

Portanto, para que o algoritmo proposto pudesse estar apto para ser aplicado na prática num sistema *fieldbus*, foi necessário o levantamento e a especificação dos componentes envolvidos na prática com os escalonamentos dos blocos funcionais e mensagens, bem como, o entendimento do comportamento global das restrições temporais existentes na aplicação entre os relacionamentos existentes das mensagens com os blocos funcionais.

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca MAUÁ BIM

CAPÍTULO 2 – FIELDBUS

O *Fieldbus* é basicamente uma arquitetura de comunicação de dados projetada especificamente para permitir medições e controle de processos, montados em ambientes agressivos situados no campo. É um sistema de comunicação digital, multiponto, bidirecional, serial, usado para a interligação de dispositivos de instrumentação e controle de processos industriais de baixo nível [5][9]. Alguns exemplos de dispositivos ou equipamentos de campo são sensores, atuadores e controladores.

Fieldbus é uma rede local (Local Area Network – LAN) utilizada pela indústria de automação de processos e manufatura que permite a distribuição da aplicação de controle nos dispositivos através da rede [9].

Diversas características dos sistemas de comunicação analógicos tradicionais (4-20mA) utilizados pela indústria são absorvidas pelo *fieldbus*, i.e., a padronização da interface física de comunicação, opção de alimentação dos dispositivos pelo barramento de comunicação através de um simples par trançado e a utilização opcional de barreiras de segurança intrínseca [13].

Além dos pontos importantes citados acima, o *fieldbus* disponibiliza uma tecnologia avançada e apta para a sua utilização pela indústria. Como o *fieldbus* utiliza comunicação digital, a quantidade de informações que trafega pelo barramento pode ser aumentada o que possibilita a redução do custo de fiação, pois, múltiplos dispositivos são conectados em um único barramento de comunicação onde informações digitais trafegam em ambas as direções. Várias opções de velocidade estão disponíveis para controle de processos e aplicações de manufatura, garantindo o atendimento aos requisitos específicos de cada tipo de atividade.

Com a distribuição da ação de controle e funções de entrada/saída para os dispositivos de campo pode-se reduzir a carga de processamento na sala de controle, disponibilizando uma maior atividade da sala de controle para as ações de supervisão, manutenção, gerenciamento e funcionalidade operacional do sistema.

Esta tecnologia permite que as funções de controle possam ser distribuídas nos equipamentos de campo, interconectando vários instrumentos, de modo que o usuário possa construir estratégias de controle apropriadas para a sua aplicação, através da ligação de blocos funcionais, sem que este venha a se preocupar com o processo ou procedimento da distribuição da aplicação de controle sobre a rede. Uma estratégia de controle é uma estrutura utilizada para a configuração da ação de controle que necessita ser executada sobre uma

determinada planta ou processo de controle.

A tarefa de distribuição da aplicação de controle fica por conta do sistema configurador, deixando o usuário desacoplado desta tarefa, bastando apenas que o usuário se preocupe com a localização dos blocos funcionais sobre cada dispositivo da rede.

O conceito de bloco funcional foi introduzido, objetivando a simplificação da programação a ser efetuada pelo usuário, tornando-a mais fácil. Deste modo, possibilita que o usuário efetue a construção e visualização de estratégias de controle complexas mais facilmente.

O *fieldbus* satisfaz as necessidades de agilidade, precisão e eficiência, almejados pelas empresas visando facilidade de instalação, de manutenção e rápida expansão (sem placas adicionais nem nova cablagem). Permite a interoperabilidade e torna compatível a utilização de produtos de diferentes fornecedores de processos industriais sem a necessidade do desenvolvimento de drivers especiais para suporte da comunicação. Portanto, a tecnologia utilizada não influencia a comunicação da rede, o que possibilita a combinação de dispositivos de diferentes fornecedores, buscando a otimização do processo de controle através do uso de dispositivos independentemente dos fabricantes, os quais possuem maior eficiência, bem como, garantindo que o sistema proporcione uma maior qualidade do produto do processo industrial.

A rede *fieldbus* possibilita ainda, a auto detecção de defeitos nos cabos durante a varredura da rede, auto-diagnóstico e consistência temporal dos dados que trafegam no barramento, o que proporcionará um número menor de paradas desnecessárias do processo industrial, bem como, um menor número de idas e vindas do campo para a verificação e ajuste de defeitos.

Este trabalho quando foi desenvolvido baseou-se na especificação da *Fieldbus Foundation*, a única normalização disponível para redes *fieldbus* em 1999. De posse das informações pode-se desenvolver um escalonador que atende-se então a alguns requisitos impostos pela norma internacional, disponibilizada a partir de dezembro de 1999.

1. BENEFÍCIOS DO FIELDBUS

Os benefícios obtidos através da aplicação da tecnologia *fieldbus* refletem diretamente no ciclo de vida do sistema de controle.

Um dos benefícios encontrados logo no processo de instalação ou de levantamento

para aquisição desta tecnologia, é observado com: a acentuada redução do número de fiações e painéis utilizados, a redução do número de barreiras de segurança intrínseca, a redução do número de conversores de entrada/saída, o número reduzido de fontes de alimentação e gabinetes.

Durante o processo de configuração e utilização desta tecnologia, tornam-se evidentes os seguintes benefícios proporcionados pelo sistema, pois, permite a configuração, calibração e diagnóstico remoto dos dispositivos, o autodiagnóstico, a avaliação do desempenho, e uma maior quantidade de informações disponíveis para operações de controle.

Como a comunicação utilizada pelo *fieldbus* é digital, pode-se validar os dados que trafegam no barramento, de tal forma que possam ser detectadas distorções e variações indesejáveis do sinal. Sendo o sistema totalmente digital e com um protocolo bem projetado, este torna-se mais robusto e imune a variações eletromagnéticas, permitindo ainda que operadores e engenheiros tomem ações preventivas e corretivas com maior segurança através da detecção de condições anormais ou da necessidade de uma manutenção preventiva.

A qualidade do produto é aumentada em função da redução das conversões A/D e D/A, da integridade do sinal, da validação dos dados e do aumento da precisão das medições efetuadas. Portanto em função destas características pode-se obter um produto com uma maior e mais estável qualidade.

Os benefícios obtidos além orçamento da empresa que utiliza este sistema, resulta diretamente no custo total de instalação do sistema, em um menor custo de manutenção, em um menor custo de expansão e modificação, e no custo de treinamento [13].

Além das vantagens disponíveis atualmente, esta tecnologia não encontra barreiras para evoluir, pois, a instrumentação utilizada é sofisticada e flexível, o que possibilita uma maior facilidade para evoluir futuramente devido à utilização de blocos funcionais padronizados.

2. CAMADAS DE COMUNICAÇÃO

O *fieldbus* baseia-se no modelo de referência OSI da ISO para Interconexão de Sistemas Abertos, isto é, sistemas que são abertos à comunicação com outros sistemas. O modelo OSI (Open Systems Interconnect) representa as várias funções requeridas em uma rede de comunicação, e possui sete camadas: *física, enlace, rede, transporte, sessão, apresentação* e *aplicação*. Cada uma delas possuindo um conjunto de tarefas específicas.

A figura 1, ilustra o relacionamento do modelo *Fieldbus* com o modelo OSI. A pilha de comunicação do *fieldbus* é compreendida pelos níveis 2 e 7 no modelo OSI.

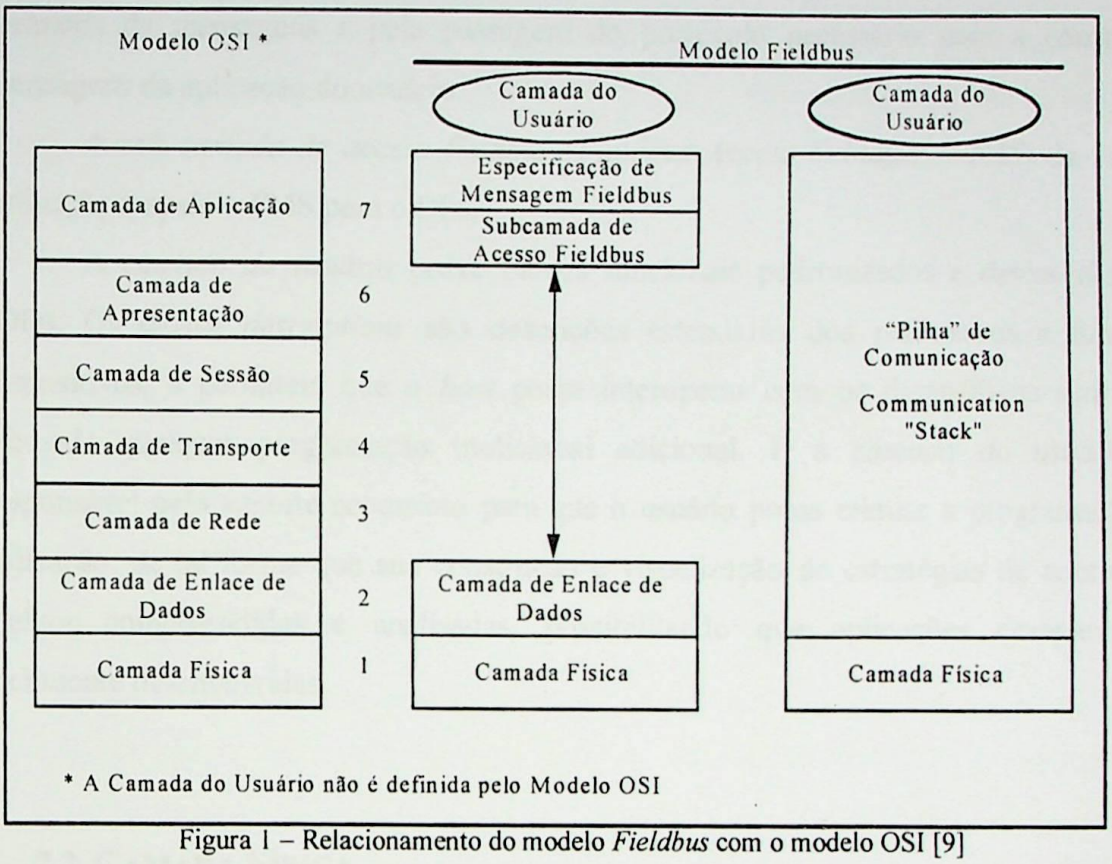


Figura 1 – Relacionamento do modelo *Fieldbus* com o modelo OSI [9]

O *fieldbus* não utiliza as camadas 3 à 6, já que elas tratam basicamente da transferência de dados entre redes. E além disto utiliza mais uma camada, a *camada de usuário*.

Cada camada no sistema de comunicação é responsável por um pedaço da mensagem que é transmitida no *fieldbus*.

A *camada física* corresponde a camada 1 no modelo OSI, e é responsável entre outras coisas, pelo tipo de sinal, meio físico, e velocidade da transmissão. Esta camada recebe as mensagens da pilha de comunicação e as converte em sinais físicos para transmissão no meio de comunicação do *fieldbus* e vice-versa.

A *camada de enlace* (Data Link Layer – DLL) corresponde a camada 2 no modelo OSI, e caracteriza os serviços de transferência de dados entre os dispositivos, além de estruturar as mensagens, garantir sua integridade e controlar o acesso à rede para transmissão destas mensagens. Esta camada está relacionada ao gerenciador de acesso ao barramento através de um escalonador determinístico e centralizado chamado de *Link Active Scheduler* (LAS).

A *sub-camada Fieldbus Message Specification – FMS* da camada de aplicação relaciona-se com algumas funções da camada 7 no modelo OSI, e é responsável pelos serviços de comunicação entre a camada de enlace e a sub-camada de acesso *fieldbus*, formatos de mensagens e pela passagem do protocolo necessário para a construção de mensagens da aplicação do usuário.

A *sub-camada de acesso fieldbus* (*Fieldbus Access Sublayer – FAS*) da camada de aplicação mapeia o FMS para o DLL.

A *camada do usuário* provê blocos funcionais padronizados e *device descriptions* (DD). Os *device descriptions* são descrições estendidas dos parâmetros e funções dos dispositivos, e permitem que o *host* possa interoperar com os dispositivos sem que seja efetuada qualquer programação tradicional adicional. É a *camada do usuário* que é responsável pelo suporte necessário para que o usuário possa efetuar a programação de sua aplicação, de tal forma que sua construção e visualização de estratégias de controle sejam melhor compreendidas e analisadas, possibilitando que aplicações complexas sejam facilmente desenvolvidas.

2.1. CAMADA FÍSICA

A camada física é constituída de duas partes: o meio físico e o método de sinalização. O meio físico atualmente disponível para o *fieldbus* são: fio, sinal de rádio e fibra óptica.

Para o fio de cobre existem duas classificações do *fieldbus*, sendo definidas como H1 para a taxa de comunicação de 31,25Kbps, H2 para a taxa de 1Mbps e 2,5Mbps e HSE para a taxa de 100Mbps (Fast Ethernet) [9].

O H1 possui baixa velocidade, podendo aproveitar a fiação existente dos sistemas analógicos utilizados pelos dispositivos de 4-20mA. Esta classificação suporta a utilização de segurança intrínseca, podendo alimentar os dispositivos pelo próprio par trançado utilizado para comunicação. Algumas aplicações que utilizam esta classificação são: aplicações de controle de temperatura, nível, fluxo, etc.

O H2 é utilizado para aplicações especiais, controle de processos avançados e aplicações de automação de fábrica de alta velocidade, que necessitam menor tempo de resposta e maior taxa de transmissão. Nesta classificação os dispositivos podem necessitar, que a alimentação seja feita independentemente ou externamente, desacoplada da fiação utilizada para a comunicação.

O número de dispositivos para um segmento *fieldbus* depende dos fatores de consumo de potência de cada dispositivo, do tipo de cabeamento utilizado, do uso de repetidores, etc.

O *High Speed Ethernet – HSE* é um backbone de 100Mbps com perspectivas futuras de chegar a 1Gbps [9]. Ele possui compatibilidade direta com equipamentos do padrão *Ethernet* e disponibiliza uma comunicação mais confiável. Esta classificação utiliza-se de fios de cobre e fibras ópticas como meios físicos de transmissão para a comunicação.

O método de sinalização possui uma forma de onda específica, com o qual o sinal/informação é codificado por códigos binários digitais (bits). *Fieldbus* utiliza uma forma de onda trapezoidal diferente, na qual com sua magnitude diminui a sensível interferência de ruídos existentes na onda quadrada tradicional.

2.2. PILHA DE COMUNICAÇÃO

A pilha de comunicação tem a função de codificação e decodificação das mensagens da *camada do usuário*, possuindo controle determinístico da transmissão das mensagens. As técnicas utilizadas para estas funções compõem um mecanismo eficiente e seguro para a transferência de dados.

A seguir serão descritos os pontos importantes de cada camada que compõe a pilha de comunicação, bem como, suas atividades e serviços disponíveis para a comunicação dos dispositivos *fieldbus*.

2.2.1. CAMADA DE ENLACE DE DADOS (DATA LINK LAYER – DLL)

Três tipos de dispositivos são especificados nesta camada. Dispositivos **básico**, ***link master*** e ***bridge***.

Dispositivos **básicos** são dispositivos inteligentes mas, comuns no que diz respeito a gerência da rede, pois, estes dispositivos não têm capacidade de virem a ser o *Link Active Scheduler* (LAS). O LAS controla a comunicação do barramento, mantendo o tempo do mestre, mantendo a lista de dispositivos mestres, escalonando o tráfico da rede e oferecendo suporte para redundância de dispositivos e de blocos funcionais.

O LAS garante que as mensagens cíclicas (mensagens geradas através da troca de variáveis de medida e de controle) sempre serão escalonadas, em conformidade com as restrições de tempo, que são críticas. Além disso, possibilita que as mensagens acíclicas

(mensagens aleatórias no tempo, como: setpoints, parâmetros de configuração, diagnósticos, etc) sejam escalonadas quando existir a disponibilidade de tempo de transmissão no barramento sem comprometer a transmissão das mensagens cíclicas.

Os dispositivos ***link masters*** tem capacidade para suportar as funções de LAS, podendo em um dado instante vir a ser o LAS do segmento. Cada segmento possui sempre apenas um LAS ativo, ou seja, um único *link master* assume o controle da comunicação do segmento. Cada segmento pode também ter vários *link masters* para que no caso de algum problema ocorrer com o LAS corrente, um dos dispositivos *link masters* do segmento possa vir a assumir as suas funções atuais, com a intenção de não permitir que a comunicação da rede pare.

As ***bridges*** são utilizadas para interconectar segmentos individuais *fieldbus* para a criação de rede maiores.

A comunicação do barramento é caracterizada de duas formas, através do tráfego de dados escalonados ou através do tráfego de dados não escalonados, ou seja, transmissão cíclica ou acíclica, respectivamente.

O LAS controla a comunicação do *fieldbus*, gerenciando a permissão dada a um dos dispositivos que desejam iniciar a transmissão de mensagens(dados) no barramento, de tal forma que somente um único dispositivo transmita em cada instante de tempo, isto é, acesse o barramento. O LAS possui também uma lista de mensagens que somente serão transmitidas entre as transmissões das mensagens cíclicas, esta lista é chamada de **"Lista Viva"**.

Cada dispositivo do barramento é identificado unicamente por um endereço, um nome (*tag*) e por seu ID. O número de endereços disponíveis da rede para cada segmento é de 240.

Com a distribuição da aplicação de controle nos dispositivos conectados ao barramento, e sendo que, cada dispositivo possui um microprocessador com um relógio (*clock*) gerador de sua própria base de tempo (tempo local), há a necessidade de que um mecanismo sincronize os relógios dos dispositivos, ajustando-os. Portanto, logo que a rede é inicializada e o LAS entra em atividade, fixa-se este instante como o instante inicial ou marco zero da base do tempo comum. Periodicamente o LAS difunde a distribuição do tempo comum (*Time Distribution – TD*) para que os relógios dos dispositivos que compõem a aplicação de controle possam ser reajustados. Esta tarefa de distribuição de tempo, resulta em uma correta comunicação escalonada e execução de blocos funcionais escalonados.

2.2.1.1. COMUNICAÇÃO ESCALONADA OU CÍCLICA

A ordenação das transmissões das mensagens no barramento é gerenciada pelo escalonador chamado de LAS (Link Active Scheduler). O LAS é um mecanismo centralizado que controla o acesso ao barramento. O LAS coordena a distribuição da ação de controle dos equipamentos sobre o barramento, permitindo que as mensagens sejam transferidas pelo barramento para outros equipamentos que necessitam de uma respectiva mensagem, através de uma tabela de escalonamento (gerada por um algoritmo de escalonamento).

A comunicação escalonada ou cíclica é estabelecida através de uma tabela de tempos de transmissão, de todos os *buffers* de dados dos dispositivos do segmento que necessitam fazer uma transmissão de tempos em tempos (ocorrências cíclicas).

Quando o tempo de um determinado dado chega, o LAS publica uma mensagem de *Compel Data* (CD) para o dispositivo gerador desta variável (*publisher*). Logo que este dispositivo recebe o CD, publica o dado no barramento. Então, o(s) dispositivo(s) que necessita(m) deste dado agora pode(m) armazená-lo em seu(s) *buffers*. Este(s) dispositivo(s) é(são) chamado(s) de *subscriber(s)*. Portanto, o dado no *buffer* do *publisher* é disponibilizado para todos os dispositivos da rede, após reconhecer um CD enviado para ele. Os *subscribers* então, atentos a comunicação do barramento, esperam o dado ser distribuído pelo *publisher* para que possam consumi-lo.

Como pode ser observado na figura 2, o LAS através de uma tabela de instantes de transmissão seleciona o dispositivo *y* e envia para ele um CD (ação 1). Em seguida o dispositivo *y* transmite o dado para o barramento (ação 2), disponibilizando o dado para que os outros dispositivos possam utilizá-lo.

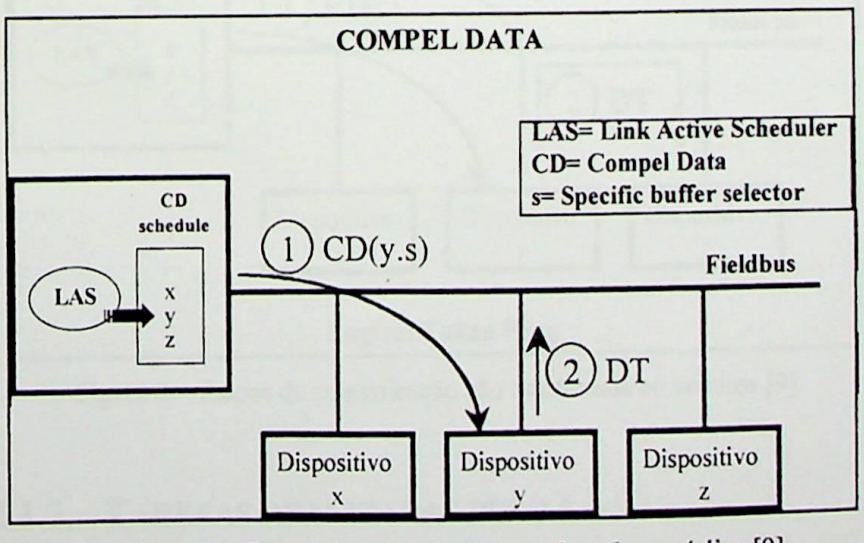


Figura 2 – Etapas da comunicação escalonada ou cíclica [9]

2.2.1.2. COMUNICAÇÃO NÃO-ESCALONADA OU ACÍCLICA

A comunicação não-escalonada ou acíclica é efetuada entre as comunicações escalonadas ou cíclicas e é permitida a todos os dispositivos *fieldbus* do barramento. Quando existir tempo disponível entre as transmissões escalonadas ou cíclicas, o LAS concede a permissão para um dispositivo utilizar o *fieldbus* através da publicação de um *Pass Token* (PT). O dispositivo endereçado pelo PT pode então transmitir mensagens até que estas sejam terminadas, ou então, enquanto o tempo máximo do PT não expirar. O tempo disponível para o PT, é sempre mantido curto para que todos os dispositivos possam ter a chance de transmitir.

Os dispositivos ativos conectados ao barramento são enfileirados em uma Lista Viva de acordo com a prioridade da mensagem que desejam transmitir.

As mensagens podem ser transmitidas para um ou múltiplos dispositivos durante a concessão do PT.

Como pode ser observado na figura 3, o LAS através da Lista Viva (Lista de circulação do PT) seleciona o dispositivo *y* que passará a ter a permissão para transmitir suas mensagens ou dados, enviando o PT para o dispositivo *y* (ação 1), em seguida o dispositivo *y* transmite o dado para o dispositivo *z* (ação 2) e terminada a ação ou expirado o tempo, o dispositivo retorna o *token* para o LAS (ação 3).

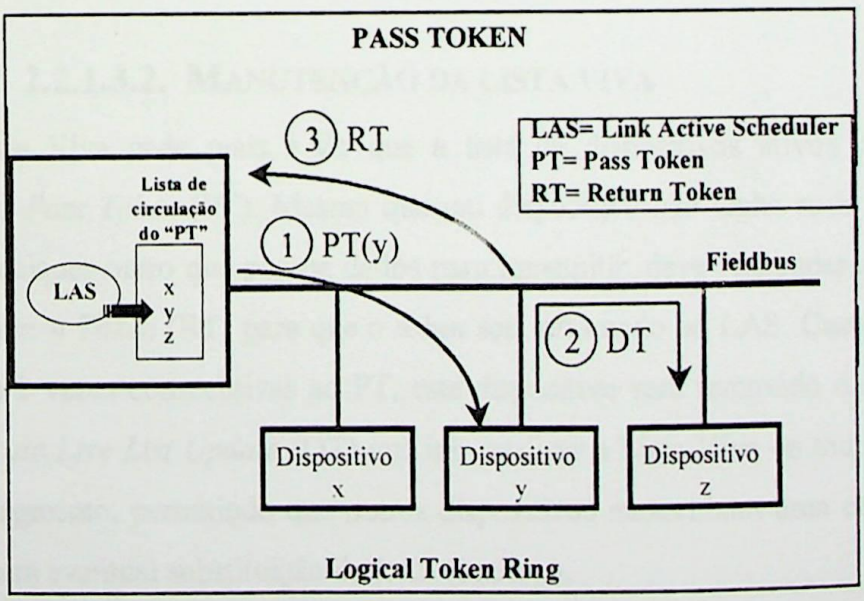


Figura 3 – Etapas da comunicação não escalonada ou acíclica [9]

2.2.1.3. TAREFAS REALIZADAS PELO LAS

O LAS deve executar tarefas rotineiras a partir do *startup* e durante a execução da

aplicação de controle. Nas seções anteriores, foram discutidos alguns pontos importantes envolvendo o LAS e suas tarefas. Mas como as atividades do LAS são essenciais para a manutenção da comunicação da rede e sendo de grande importância para a compreensão do processo de comunicação entre os dispositivos, serão agrupadas nesta seção e explicadas todas as tarefas realizadas pelo LAS. As tarefas que o LAS deve realizar são: envio do *Compel Data* (CD) no tempo programado, manutenção da Lista Viva, sincronização do tempo da camada de enlace de dados, passagem de *token*, e redundância do LAS.

2.2.1.3.1. ENVIO DO CD NO TEMPO PROGRAMADO

O LAS possui uma programação prévia de envio do CD, estabelecida através de uma lista de atividades escalonadas que deverão ocorrer regularmente, de forma cíclica durante a execução da aplicação definida pelo usuário.

Esta tarefa é a atividade de mais alta prioridade em relação aos outros serviços, pois a eficiência de operação da aplicação depende diretamente da comunicação escalonada ou cíclica.

Atividades ou tarefas suplementares para a manutenção, gerência e supervisão do *fieldbus* serão efetuadas entre as transferências de dados escalonadas. No entanto, estas atividades ou tarefas não devem alterar ou atrapalhar as transferências de dados escalonadas já definidas.

2.2.1.3.2. MANUTENÇÃO DA LISTA VIVA

A Lista Viva nada mais é do que a lista de dispositivos ativos no *fieldbus* que respondem ao *Pass Token* (PT). Mesmo que um dispositivo não tenha nada para transmitir, este, como qualquer outro que possua dados para transmitir, deve responder ao PT com uma mensagem *Return Token* (RT) para que o *token* seja retornado ao LAS. Caso um dispositivo não responda 3 vezes consecutivas ao PT, este dispositivo será removido da Lista Viva e o LAS enviará um *Live List Update* (DT) que irá atualizar a Lista Viva de todos os outros *link masters* do segmento, permitindo que outros dispositivos mantenham uma cópia corrente da Lista Viva, para eventual substituição do LAS (*backup*).

Como é permitida a adição a "quente" (com a rede operando) de novos dispositivos, então o LAS deve periodicamente fazer a verificação de endereços de dispositivos que não encontram-se na Lista Viva. Esta verificação é efetuada através do envio da mensagem *Probe Node* (PN) para os endereços. Se um dispositivo estiver presente no endereço, após a recepção do PN, deverá responder com uma mensagem *Probe Response* (PR) para que o LAS possa assim adicioná-lo a Lista Viva. O LAS confirma o ingresso de um novo dispositivo na Lista Viva através de uma mensagem enviada ao dispositivo: *Node Activation* (DT).

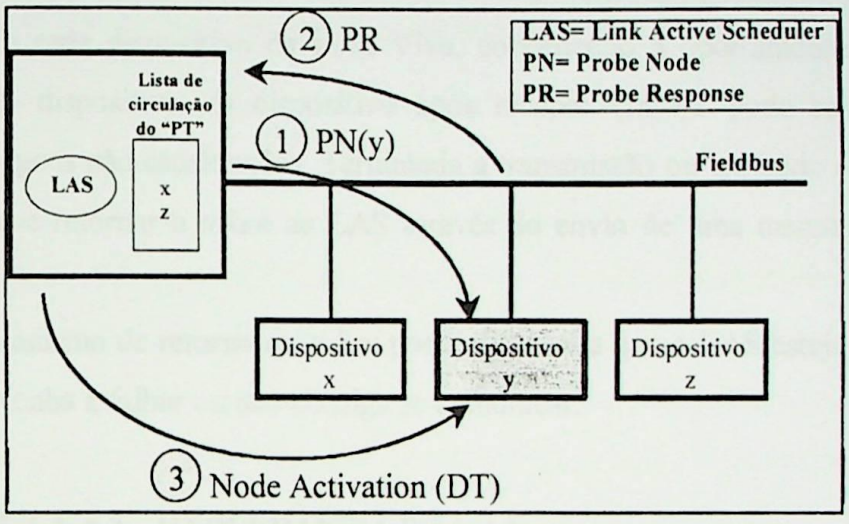


Figura 4 – Adição de um dispositivo a Lista Viva [9]

Como pode ser observado pela figura 4, após a circulação do PT para todos os dispositivos contidos na Lista Viva, o LAS envia uma mensagem PN para pelo menos um endereço que não esteja na Lista Viva, como é ilustrado na figura 4 pela ação 1. O dispositivo y recebendo um PN, responde ao LAS com um PR informando que o endereço possui um dispositivo ativo (ação 2). Adicionado o dispositivo a Lista Viva, o LAS envia uma mensagem DT para a atualização da Lista Viva pelos outros dispositivos do barramento (ação 3).

A verificação do ingresso de novos dispositivos no *fieldbus* é efetuada no mínimo para um endereço, após o ciclo de envio de PTs para todos os dispositivos contidos na Lista Viva.

2.2.1.3.3. SINCRONIZAÇÃO DO TEMPO DA CAMADA DE ENLACE DE DADOS

As comunicações escalonadas no *fieldbus* e as execuções de *function blocks* escalonados pela aplicação do usuário dependem do sincronismo de tempo da camada de enlace de dados de todos os dispositivos. Portanto, o LAS deve periodicamente difundir uma

mensagem de *Time Distribution* (DT) no *fieldbus*, para que todos os dispositivos estejam atualizados com o mesmo tempo de enlace de dados. Assim, regularmente os *clocks* internos dos dispositivos serão atualizados, o que manterá seus *clocks* alinhados com o tempo referencial do barramento emitido pelo LAS.

2.2.1.3.4. PASSAGEM DE *TOKEN*

Esta tarefa é executada pelo LAS, possibilitando que todos os dispositivos tenham a chance de efetuar transmissões não-escalonadas. O LAS então envia uma mensagem de *Pass Token* (PT) para cada dispositivo da Lista Viva, concedendo a oportunidade de acesso ao barramento pelo dispositivo. O dispositivo após receber um PT pode então começar a transmitir mensagens não escalonadas. Terminada a transmissão ou expirado o tempo do PT, o dispositivo deve retornar o *token* ao LAS através do envio de uma mensagem de *Return Token* (RT).

Este mecanismo de retorno do *token* possibilita ainda que o LAS esteja ciente quando um dispositivo venha a falhar ou não consiga se comunicar.

2.2.1.3.5. REDUNDÂNCIA DO LAS

A redundância de LAS é importante para que a comunicação da rede seja garantida, mesmo quando o LAS ativo falhar. Se ocorrer um problema com o LAS corrente, um dos outros *link masters* do segmento do *fieldbus* poderá assumir as atividades do LAS corrente, de tal forma que ele agora venha a ser o LAS ativo do segmento do *fieldbus*.

O *fieldbus* foi projetado para eventuais problemas operacionais, prevenindo-se de paradas ou falhas repentinas que venham a ocorrer.

2.2.2. SUB-CAMADA DE ACESSO FIELDBUS (FIELDBUS ACCESS SUBLAYER – FAS)

O FAS é a interface de comunicação entre a camada de enlace de dados (DLL) e a camada de especificação de mensagem *fieldbus* (FMS). Ele provê serviços ao FMS utilizando-se das comunicações escalonadas e não-escalonadas providas pelo DLL.

Os serviços oferecidos pelo FAS são relacionamentos de comunicações virtuais existentes (*Virtual Communication Relationships – VCR*), que são configuradas depois que o usuário define sua aplicação ou durante a solicitação de um serviço pelo usuário (pertinentes a comunicação não-escalonada ou acíclica).

O FAS provê os serviços ao FMS através de atalhos simplificados dando suporte a todo procedimento de comunicação do DLL sem que o FMS tenha a necessidade de conhecer as mensagens que serão utilizadas pelo DLL para as comunicações escalonadas e não-escalonadas.

Diferentes tipos de VCRs ou serviços estão disponíveis através do FAS: cliente/servidor, distribuição de relatório e publisher/subscriber.

Estes serviços serão explicados e detalhados no itens seguintes.

2.2.2.1. SERVIÇO CLIENTE/SERVIDOR (CLIENT/SERVER VCR)

As operações do FMS que utilizam estes serviços são operações de configuração, supervisão, *Upload/Download* de dispositivos, *Read/Write*, diagnóstico remoto, gerenciamento de alarme, mudança de *setpoints*, etc. Este serviço utiliza-se da comunicação não-escalonada (acíclica) de um endereço de origem para um endereço de destino da rede.

Para que a transferência de dados seja executada, deve-se estabelecer uma conexão entre origem e destino do envio da mensagem. Esta conexão somente é feita, entre endereços permanentes da rede. A partir do momento em que esta conexão é estabelecida, pode-se efetuar a transmissão dos dados.

As mensagens são enviadas e recebidas na sequência em que foram transmitidas, definidas pelas suas prioridades, sem que ocorra a escrita de uma mensagem atual sobre as mensagens anteriormente transmitidas, ou seja, há um enfileiramento das mensagens.

Quando um dispositivo recebe um *Pass Token* (PT) do LAS, este dispositivo pode enviar uma mensagem ou requisição de uma mensagem para outro dispositivo do *fieldbus*. O requisitante é chamado de cliente e o dispositivo que supre ou fornece a mensagem requisitada é denominado servidor. O envio da mensagem do servidor somente pode ser efetuado quando este receber um *Pass Token* (PT).

A figura 5 ilustra o procedimento utilizado para o estabelecimento da comunicação feita através do serviço Cliente/Servidor oferecido pelo FAS à camada FMS.

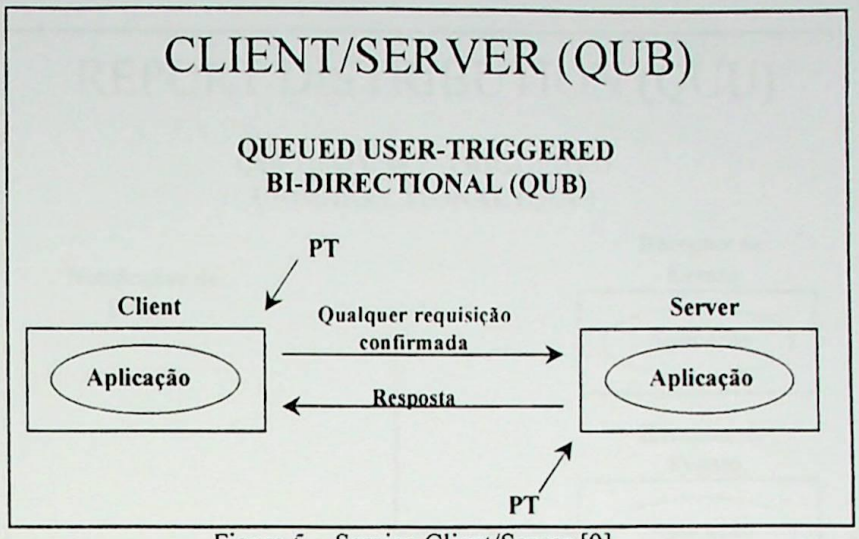


Figura 5 – Serviço Client/Server [9]

2.2.2.2. SERVIÇO DE DISTRIBUIÇÃO DE RELATÓRIO (REPORT DISTRIBUTION VCR)

As operações do FMS que utilizam este serviço são operações de envio de reconhecimento de alarmes, eventos e relatórios. Este serviço utiliza-se da comunicação não-escalonada (acíclica) de um endereço de origem para vários endereços de destino da rede.

A transferência de dados é executada sem o estabelecimento de uma conexão entre os dispositivos de origem e destino da mensagem e sem a confirmação da recepção da mensagem pelos dispositivos. Portanto, a mensagem é transmitida na rede e os dispositivos previamente configurados pela aplicação do usuário, podem recebê-la.

Quando um dispositivo recebe do LAS um *Pass Token* (PT) e possui um relatório de evento ou de tendência, o dispositivo envia esta mensagem para vários endereços especificados pelo VCR. Imediatamente após a transmissão da mensagem, o(s) dispositivo(s) que foi(foram) configurado(s) para receber este relatório deste VCR, armazena(m) a mensagem enfileirando-a sem sobrescrever as mensagens anteriores.

A figura 6 ilustra o procedimento utilizado para o estabelecimento da comunicação feita através do serviço de Distribuição de Relatório oferecido pelo FAS à camada FMS.

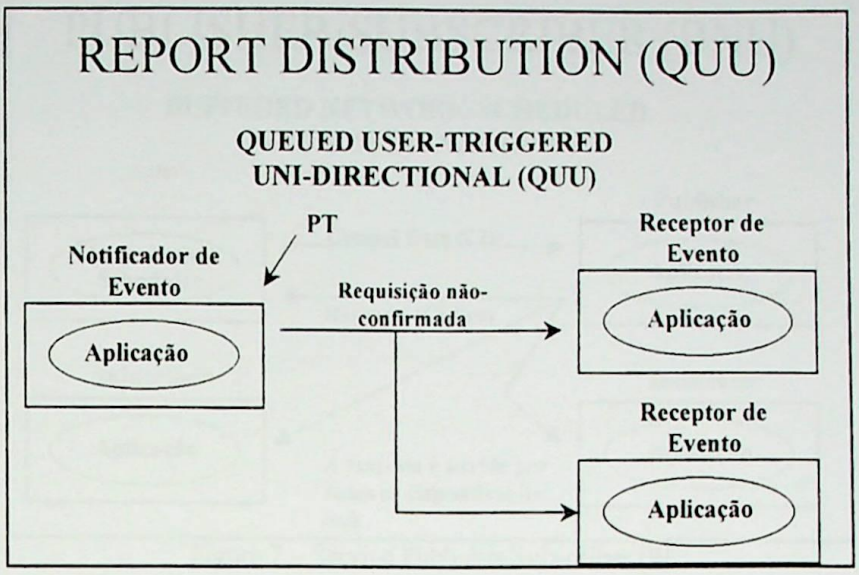


Figura 6 – Serviço de distribuição de relatório [9]

2.2.2.3. SERVIÇO PUBLISHER/SUBSCRIBER (PUBLISHER/SUBSCRIBER VCR)

As operações do FMS que utilizam este serviço são operações de envio de variáveis entre blocos funcionais da aplicação de controle que tenham que acessar o barramento. Este serviço utiliza-se da comunicação escalonada (cíclica) de um endereço de origem para vários endereços de destino da rede sendo orientada à conexão.

Os dados transmitidos são sobrescritos no *buffer* dos dispositivos fazendo com que seja mantida sobre a rede somente a última versão dos dados.

Quando um dispositivo recebe um *Compel Data* (CD) do LAS, este dispositivo deverá publicar sua mensagem escalonada para todos os dispositivos da rede. Logo a seguir, os dispositivos que precisam desta mensagem, podem consumi-la (definindo o dispositivo como *subscriber*) para que o processamento do(s) bloco(s) funcional(is) posterior(es) à requisição da mensagem seja(m) executado(s).

Uma mensagem de CD pode ser um envio de mensagem escalonada (enviada pelo LAS) ou não-escalonada (enviada por *Subscribers*). O método utilizado é identificado por um atributo do VCR estipulado pela aplicação do usuário ao FMS.

A figura 7 ilustra o procedimento utilizado para o estabelecimento da comunicação feita através do serviço Publisher/Subscriber oferecido pelo FAS à camada FMS.

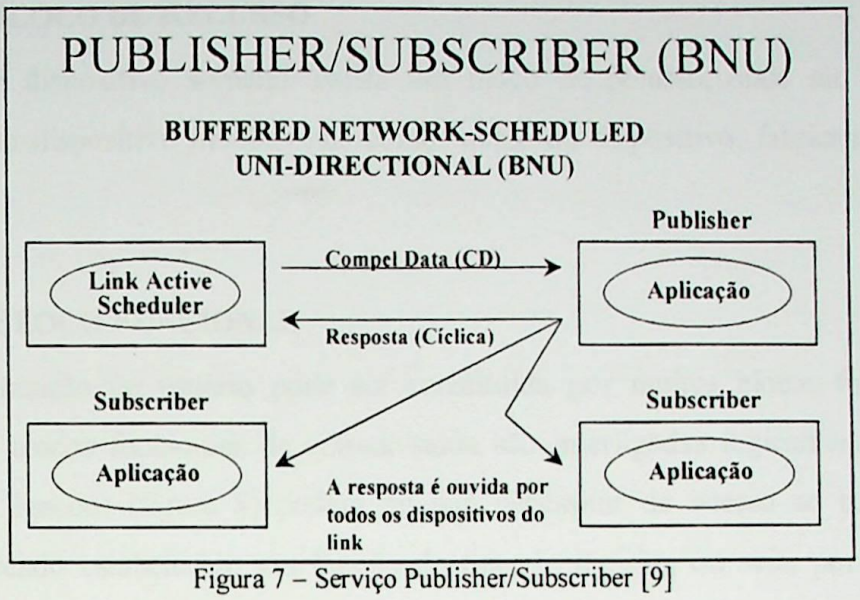


Figura 7 – Serviço Publisher/Subscriber [9]

2.2.3. Camada de Especificação de Mensagem fieldbus (Fieldbus Message Specification – FMS)

O FMS provê serviços à camada de aplicação utilizando-se de um conjunto padronizado de formatos de mensagens que é fornecido para a aplicação do usuário, possibilitando que a aplicação do usuário efetue a transmissão de mensagens entre blocos funcionais através do barramento de comunicação.

O FMS determina os serviços de comunicação, o formato das mensagens e o comportamento dos protocolos de comunicação utilizados para a construção de mensagens fornecidas à aplicação do usuário.

2.3. Camada do Usuário ou de Aplicação do Usuário

A aplicação do usuário é dividida em partes ou em funções que reunidas compõem a estratégia de controle da aplicação. Esta funções da aplicação são denominadas de blocos funcionais que representam diferentes tipos de atividade. A *Fieldbus Foundation* definiu a padronização destas funções baseadas em blocos requeridos pela aplicação do usuário.

Os blocos utilizados pela aplicação do usuário são agrupados em três conjuntos específicos de blocos: bloco de recurso, bloco transdutor e bloco funcional.

A seguir estes blocos serão definidos e apresentados, bem como, sua funcionalidade será detalhada.

2.3.1. BLOCO DE RECURSO

Em cada dispositivo somente existe um bloco de recurso, onde são descritas as características do dispositivo *fieldbus* tais como: nome do dispositivo, fabricante e número serial.

2.3.2. BLOCO FUNCIONAL

Uma aplicação de usuário pode ser constituída por muitos blocos funcionais. As mensagens dos blocos funcionais de entrada/saída são interligadas logicamente através do *fieldbus*. Estas ligações (figura 8) podem ou não necessitar do acesso ao barramento de comunicação, sendo estabelecido em função do tipo de ligação, ou seja, por uma ligação interna ou externa ao dispositivo. Uma ligação externa é uma ligação entre blocos funcionais que se encontram em dispositivos diferentes e pela qual os dados são trocados entre estes blocos funcionais.

Os blocos funcionais são rigorosamente escalonados, sendo executados precisamente em conformidade com a base de tempo estabelecida pelo LAS.

A *Fieldbus Foundation* definiu um conjunto básico de blocos funcionais padronizados para controle convencional. Este conjunto básico é formado por 21 blocos, tais como: entrada analógica, saída analógica, *bias*, seletor de controle, entrada discreta, saída discreta, *manual loader*, proporcional/derivativo, proporcional/integral/derivativo, *ratio*.

Mais 19 blocos funcionais para controle avançado foram padronizados [9].

Além dos blocos funcionais padronizados, pode-se desenvolver blocos funcionais personalizados pelo usuário, de tal forma que sejam carregados nos dispositivos *fieldbus*. Esta possibilidade eleva a aplicação do usuário a uma maior funcionalidade alcançada pelos dispositivos *fieldbus*.

2.3.3. BLOCO TRANSDUTOR

Para cada bloco funcional de entrada/saída utilizado na aplicação do usuário existe um bloco transdutor.

Estes blocos assumem a função de entrada/saída locais para que os blocos funcionais estejam desacoplados das operações de leitura de sensores e comandos de saída de hardware. Além disso, estes blocos possuem informações que dizem respeito ao tipo de sensores utilizados e aos dados de calibração.

3. GERÊNCIA DO SISTEMA

A gerência do sistema faz a sincronização da execução dos blocos funcionais que possuem uma sequência adequada para serem executados, e coordena a comunicação entre mensagens dos blocos funcionais que necessitam ser transmitidas através do *fieldbus*, ou seja, uma ligação externa entre blocos funcionais.

Além destas funções, a gerência do sistema também manipula outras características importantes do sistema tais como: o tempo de publicação de todos os dispositivos, o chaveamento automático do tempo redundante para o *publisher*, atribuição de endereço automático dos dispositivos, e busca por parâmetros como nome e *tags* do *fieldbus*.

Todas as informações de configuração que a gerência do sistema necessita estão contidas no *Network and System Management Virtual Field Device (VFD)* em cada dispositivo.

O VFD é um mecanismo para acesso da "Base de Informação de Gerência do Sistema" (*System Management Information Base – SMIB*), e também do acesso à "Base de Informação de Gerência da Rede" (*Network Management Information Base – NMIB*).

Devido a relevância deste trabalho com o escalonamento de blocos funcionais que é uma função executada pela gerência do sistema, a seguir será detalhado melhor este assunto, mas isto não quer dizer que a gerência do sistema somente trata do escalonamento de blocos funcionais. A gerência do sistema executa ainda: a distribuição do *clock* da aplicação necessária para a sincronização do tempo da aplicação, atribuição de endereço automático para os dispositivos e o serviço de obtenção do *TAG* que indica a busca por um dispositivo ou variáveis existentes na rede *fieldbus*.

3.1. ESCALONAMENTO DE BLOCO FUNCIONAL

Para a adequada operação do sistema, é necessário a utilização de um programa que é responsável pela construção do escalonamento da execução dos blocos funcionais, pelo escalonamento da transmissão de mensagens existentes sobre o *fieldbus* e gerenciada pelo *Link Active Scheduler* (LAS).

Como resultado do escalonamento obtém-se uma tabela (tabela 1) que contém o instante inicial de execução para os blocos funcionais e instante inicial para a transmissão de mensagens baseados no tempo base (referencial) ou base de tempo absoluto do *schedule* do segmento.

Tabela 1 – Tabela de escalonamento [9]

	Offset do base de tempo absoluto (referencial) do *schedule* do segmento (*mili segundos*)
Execução Escalonada para o Bloco Funcional AI	0
Comunicação Escalonada da Mensagem do AI	20
Execução Escalonada para o Bloco Funcional PID	30
Execução Escalonada para o Bloco Funcional AO	50

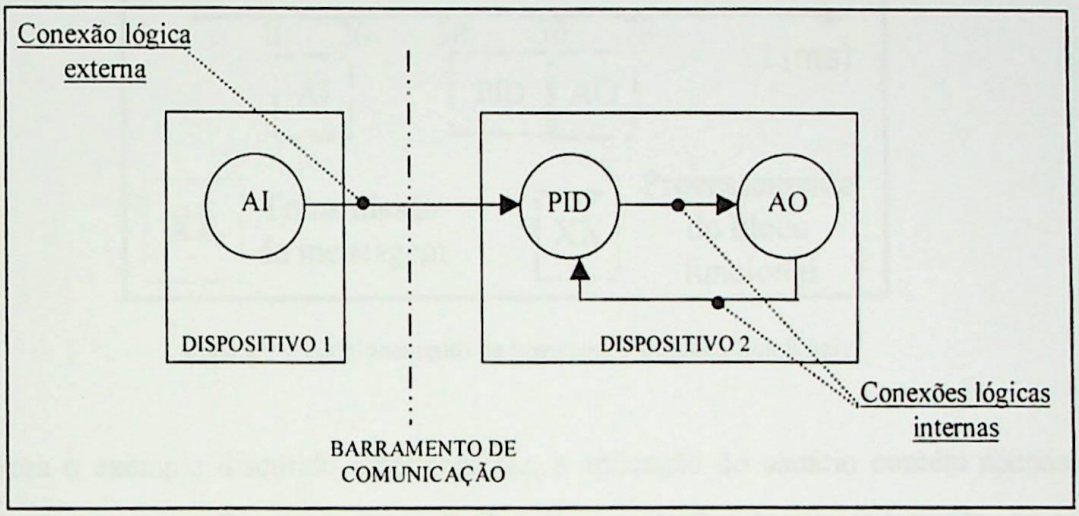


Figura 8 – Visualização das conexões lógicas de uma aplicação

A tabela 1 é resultado do escalonamento da aplicação do usuário definida pela figura 8, onde os instantes iniciais de tempo para blocos funcionais e para mensagens são definidos. Como pode ser visto na figura 8, somente a mensagem de saída do bloco AI será transmitida através do *fieldbus*, pois, representa uma ligação externa entre blocos funcionais, ou seja, a mensagem deve ser transmitida para outro dispositivo através do barramento.

Para a ligação do bloco funcional PID e AO, a mensagem de saída do bloco PID não necessita ser transmitida pelo *fieldbus*, pois, representa uma ligação interna entre blocos funcionais, ou seja, a mensagem fica contida no mesmo dispositivo.

Portanto, como pode ser visto na tabela 1, o bloco funcional AI será executado no instante 0*ms* no processador do dispositivo1 no qual este bloco está localizado, e quando terminada a execução do bloco, com tempo de processamento de 20*ms*, a mensagem do bloco AI será transmitida ao barramento. Terminado o tempo de transmissão da mensagem do bloco AI pelo barramento, o próximo bloco funcional da estratégia de controle será executado, ou seja, o bloco PID que será executado no instante 30*ms* no processador do dispositivo 2 terminando sua execução no instante 50*ms*. Finalmente, o bloco AO, que é o próximo bloco a

ser executado da estratégia de controle será executado no instante 50*ms*, com tempo de processamento de 20*ms*, terminando sua execução no instante 70*ms*.

A figura 9 ilustra o barramento de comunicação sendo utilizado pela transmissão da mensagem do bloco AI, além de representar o escalonamento de blocos funcionais geradores e consumidores da mensagem transmitida.

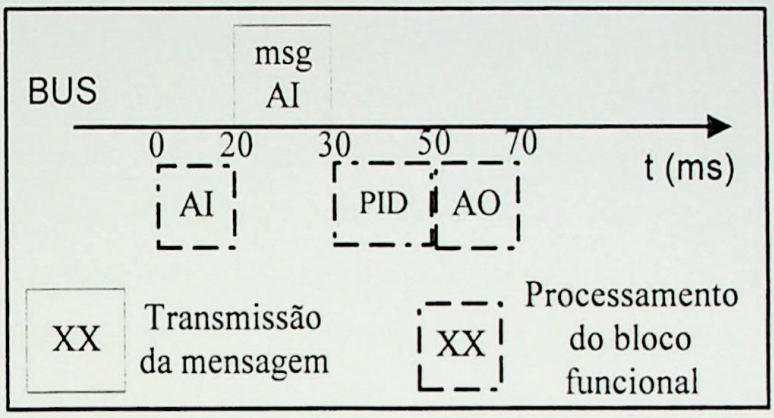


Figura 9 – Escalonamento de mensagens e blocos funcionais

Para o exemplo discutido anteriormente, a aplicação do usuário contém apenas uma estratégia de controle. Por isso, ficou fácil a representação do escalonamento de blocos funcionais sobre o barramento. Quando a aplicação do usuário possuir mais estratégias de controle, ficará difícil a representação do escalonamento de blocos funcionais sobre a mesma linha de tempo do barramento de comunicação. Uma alternativa para a visualização do escalonamento de blocos funcionais e correspondência com a transmissão das mensagens no barramento é ilustrado pela figura 10.

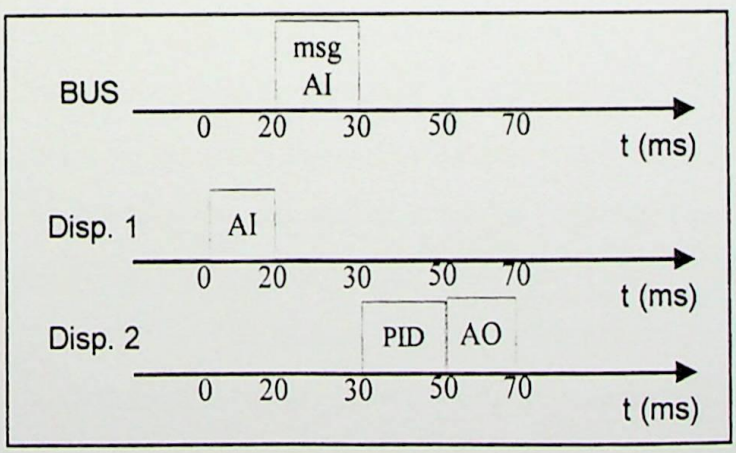


Figura 10 – Representação alternativa para o escalonamento

Como pode ser visto, em paralelo com a linha de tempo do barramento são definidas

linhas de tempo para cada dispositivo ou processador do dispositivo, representando sobre esta base de tempo, o escalonamento dos blocos funcionais de cada dispositivo individualmente. Este tipo de representação possibilita a devida correspondência entre o escalonamento das mensagens do barramento e o escalonamento entre os blocos funcionais de cada dispositivo, bem como a identificação de blocos funcionais contidos em cada dispositivo.

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca MAUÁ BIM

CAPÍTULO 3 – O PROBLEMA

Em um sistema *fieldbus* os dispositivos são conectados num único barramento de comunicação (recurso compartilhado), e a aplicação industrial é distribuída entre os dispositivos do barramento (alocação de processos distribuídos).

O compartilhamento de um recurso ocorre com a disputa de acesso entre mais de um dispositivo pelo único meio de comunicação (barramento). Este compartilhamento do barramento entre dispositivos, requer então um perfeito gerenciamento de acesso dos dispositivos ao barramento, possibilitando que a comunicação entre dispositivos seja garantida, sem a ocorrência de colisões de dados no meio físico de comunicação. Portanto, os dispositivos *fieldbus* são conectados em um único barramento de comunicação e a aplicação do usuário é distribuída nestes dispositivos, de tal forma que o acesso ao barramento seja gerenciado e controlado através de uma tabela de instantes de tempo para transmissão, que cada dispositivo deve obedecer. Somente poderá transmitir ou acessar o barramento, um único dispositivo por vez em um dado instante de tempo disponibilizado para este dispositivo, sendo este previamente calculado para o atendimento da aplicação do usuário.

Para que a aplicação do usuário possa ser distribuída entre os dispositivos, as estratégias de controle devem ser particionadas, permitindo que um conjunto de dispositivos com seus blocos funcionais possam compor a aplicação de controle do usuário.

Uma estratégia de controle desempenha as funções de tratamento de dados e de controle que devem ser executadas numa planta ou processo industrial. Isto possibilita a aquisição dos valores dos sensores, o tratamento dos dados, a execução do controle, e finalmente permite o acionamento dos equipamentos da planta.

A aplicação do usuário pode ser constituída por mais de uma estratégia de controle e depende diretamente da sincronização dos blocos funcionais e mensagens.

Para que a sincronização das mensagens e blocos funcionais possa ser atendida, é necessário o escalonamento das mensagens no barramento em função do processamento dos blocos funcionais nos dispositivos. As mensagens serão transmitidas para o barramento em instantes de tempos fixos estabelecidos numa tabela conforme seu escalonamento.

Portanto, uma mensagem transmitida no *fieldbus* em um dado instante de tempo, é enviada por um e somente um dispositivo, que tenha obtido a permissão de acesso ao barramento para transmissão, através de uma concessão emitida pela entidade que coordena a comunicação no *fieldbus* (LAS), ao ler a tabela de escalonamento.

A estratégia de controle é particionada através dos chamados blocos funcionais. Cada um destes, possui uma relação com suas referidas mensagens que trafegam pelo barramento. bem como, alguns destes devem também obedecer certos pré-requisitos, ou seja, aguardar a transmissão de determinada mensagem sobre o barramento para iniciarem seus processamentos. Esta espera pela transmissão de uma determinada mensagem, significa que esta mensagem deverá estar disponível, i.e., já deverá ter sido transmitida ao barramento para que assim, posteriormente, possa ser utilizada pelos próximos blocos funcionais subsequentes. Portanto, uma estratégia de controle depende da sincronização de mensagens e blocos funcionais do sistema.

Alguns blocos funcionais que compõem a estratégia de controle necessitam que as mensagens sejam produzidas por outros blocos. Estas mensagens podem ou não ser transmitidas através do barramento dependendo do tipo de conexão ou ligação feita entre estes blocos funcionais. Uma **conexão ou ligação interna** entre blocos funcionais não precisa acessar o barramento de comunicação, pois, os blocos funcionais ligados encontram-se no mesmo dispositivo. No entanto, uma **conexão ou ligação externa** entre blocos funcionais precisa acessar o barramento de comunicação para transmissão da mensagem, pois, os blocos funcionais encontram-se em dispositivos diferentes.

O problema apresenta-se com a obtenção de um escalonamento de um barramento de comunicação compartilhado pelas mensagens trocadas entre dispositivos e com a obtenção de um escalonamento de blocos funcionais para cada dispositivo. Além disso, é necessário a sincronização de blocos funcionais e mensagens, bem como, satisfazer às restrições temporais, tempos ociosos de ocupação do barramento, relações de precedência de mensagens e blocos funcionais, e restrições fim-a-fim dos blocos funcionais e mensagens de cada estratégia de controle que compõe a aplicação do usuário.

1. ABORDAGEM UTILIZADA

Como ponto de partida para solucionar o referido problema, utilizaram-se técnicas aplicadas para o escalonamento de tarefas em processadores, bem como, técnicas aplicadas à alocação de processos em processadores distribuídos utilizados na Ciência da Computação, em [5] [8] [10] [11] [12] [17] [18].

Em função destas técnicas, adaptou-se a idéia de escalonamento de tarefas em processadores para o escalonamento de mensagens da comunicação no *fieldbus* como em [4][5] e ainda adaptou-se o procedimento de alocação de processos em processadores

distribuídos para a alocação e escalonamento dos blocos funcionais nos dispositivos *fieldbus*.

Para o escalonamento das mensagens foram feitas as seguintes analogias entre a ciência da computação e a comunicação *fieldbus*:

1) único processador → único barramento
2) tarefa executada no processador → mensagem transmitida no barramento
3) tempo de computação → tempo de transmissão
4) tempo de *release* da tarefa → tempo de *release* da mensagem
5) *deadline* da tarefa → *deadline* da mensagem
6) relação de precedência das tarefas → relação de precedência das mensagens
7) não preempção de uma tarefa no processador → não preempção da transmissão de uma mensagem no barramento

Para o escalonamento dos blocos funcionais foram feitas as seguintes analogias entre a ciência da computação e a comunicação *fieldbus*:

1) processadores distribuídos → dispositivos distribuídos
2) processo executado no processador → bloco funcional executado no dispositivo
3) tempo de processamento do processo → tempo de processamento do bloco funcional
4) instante de *release* do processo → instante de *release* do bloco funcional, especificado pelos *releases* das mensagens
5) *deadline* do processo → *deadline* do bloco funcional estipulado pelo *deadline* das mensagens
6) relação de precedência dos processos → relação de precedência dos blocos funcionais
7) não preempção de um processo no processador → não preempção de um bloco funcional no dispositivo
8) índice do processador no qual o processo será executado → *tag* do dispositivo no qual o bloco funcional será executado

Caracterizada a analogia pôde-se fazer um levantamento e análise para a verificação dos algoritmos aplicáveis ao processo de comunicação no *fieldbus*. Como a comunicação no *fieldbus* apresenta características *hard real time*, seu escalonamento é muito particular. Assim houve necessidade de desenvolvimento de um algoritmo mais complexo, o qual pudesse

absorver alguns procedimentos dos diversos algoritmos que vêm sendo estudados, bem como, pudesse satisfazer todas as funcionalidades do sistema *fieldbus*. Outro ponto importante para estudo e inclusão no algoritmo foi a busca da garantia da consistência temporal do sistema, pois, esta garante que uma variável ou mensagem adquirida no meio de comunicação é um dado validado ou um dado atualizado globalmente para os outros dispositivos em relação a um determinado intervalo de tempo [4].

Quando a consistência temporal do sistema não é considerada, não se pode garantir que a comunicação seja bem sucedida, pois, esta poderá ser degradada, ou em muitos casos, poderá existir a perda da ordem das mensagens, implicando que algumas aplicações não possam ser implementadas em função da complexidade demandada.

O algoritmo desenvolvido baseou-se no algoritmo de Inteligência Artificial desenvolvido por Xu e Parnas que usa uma heurística (EDF) baseada nas especificações das metodologias clássicas de escalonamento, as quais foram adaptadas para responder dentro das restrições de tempo impostas pela consistência temporal da aplicação [7][15].

2. A COMUNICAÇÃO NO FIELDBUS

O respectivo trabalho objetiva a implementação de um algoritmo de escalonamento em conformidade com a especificação do protocolo *Fieldbus Foundation*. Para que este algoritmo seja aplicável, deve garantir o atendimento do tráfego de mensagens cíclicas e acíclicas.

Além do escalonamento das mensagens através do barramento, as aplicações requerem que as restrições temporais sejam atendidas. Estas restrições foram definidas em Franco [4], baseadas nas aplicações do usuário. Elas serão obedecidas neste trabalho e melhor representadas através de alguns exemplos de aplicações apresentados no Capítulo 6. Desta forma deve-se então satisfazer o atendimento à precedência (devido ao encadeamento de blocos funcionais gerado pela aplicação), ao tempo de transmissão ocioso gerado pelo processamento dos blocos funcionais nos dispositivos, às restrições fim-a-fim de blocos funcionais e mensagens, e a consistência temporal da aplicação do usuário [5].

O algoritmo desenvolvido satisfaz às restrições da comunicação das mensagens cíclicas e acíclicas através do escalonamento do barramento. Mensagens cíclicas são mensagens geradas pela troca de variáveis ou mensagens que são trocadas entre os dispositivos durante as ações de medição e controle. O atendimento das restrições temporais

destas mensagens é imprescindível para que a aplicação do usuário opere conforme o esperado [3]. Exemplos de mensagens cíclicas são: variável de um sensor, realimentação de uma variável, acionamento de um atuador, etc.

O tráfego das mensagens acíclicas somente serão satisfeitas quando houver disponibilidade de tempo de transmissão no barramento, ou seja, quando o barramento estiver ocioso entre as transmissões das mensagens cíclicas. As mensagens acíclicas são mensagens aleatórias ou mensagens não-escalonadas com ocorrência incerta no tempo, durante o ciclo operacional do sistema.

O algoritmo proposto escalona as mensagens cíclicas em *pre-run-time* e satisfaz às mensagens acíclicas em *run-time* quando for possível. As mensagens acíclicas são transmitidas entre as transmissões das mensagens escalonadas ou mensagens cíclicas. Estas somente serão atendidas se não produzirem atraso nas mensagens cíclicas escalonadas, pois, com a inclusão deste atraso, as restrições, tempos ociosos do barramento e relações de precedência não poderão ser garantidas.

Alguns exemplos de mensagens acíclicas são: *setpoints*, parâmetros de configuração, diagnósticos, alarmes, etc. Estas mensagens podem ser preemptadas e transmitidas quando o barramento ficar ocioso para as transmissões escalonadas.

Todos estes pontos ou características importantes para a comunicação no *fieldbus*, serão discutidos detalhadamente no tópico a seguir.

2.1. FATORES IMPORTANTES PARA A COMUNICAÇÃO NO FIELDBUS

Uma estratégia de controle composta por mais de um bloco funcional quando distribuída entre os dispositivos, tem que atender às relações de precedências existentes entre seus blocos funcionais. Neste caso, as mensagens trocadas entre os dispositivos terão que trafegar através do barramento, de tal forma que cada mensagem seja escalonada dentro de uma janela de tempo na qual as relações de precedências entre blocos funcionais sejam respeitadas.

Relações de precedências entre mensagens são relacionamentos entre duas mensagens, onde a primeira mensagem (mensagem predecessora) deve ser escalonada para transmissão pelo barramento, antes da outra mensagem (mensagem sucessora) a ela relacionada. Estas relações são determinadas pelas relações de precedência entre blocos funcionais, definidas pelas estratégias de controle.

As **relações de precedência entre blocos funcionais** são utilizadas neste trabalho como base para se caracterizar o instante de tempo adequado para transmissão da(s) mensagem(ns) através do barramento, de tal forma que seja respeitado o sincronismo e o sequenciamento entre mensagens e blocos funcionais.

Relações de precedência entre blocos funcionais são relacionamentos entre dois blocos funcionais, onde o primeiro bloco (bloco funcional predecessor) deve ser escalonado para execução no processador do dispositivo no qual está localizado, antes da execução do outro bloco funcional (bloco funcional sucessor) que possui relação com o bloco funcional predecessor.

As relações de precedências entre mensagens oferecem suporte para o escalonamento das mensagens, de tal forma que ajudam a satisfazer as restrições temporais da aplicação através da ordenação de transmissão das mensagens no barramento. Portanto, para que pudesse ser oferecido o mesmo suporte para o escalonamento dos blocos funcionais, necessitou-se a criação de um relacionamento entre blocos funcionais e mensagens, que garantisse o atendimento das restrições temporais dos blocos funcionais.

A **consistência temporal dos dados** garante que quando uma determinada mensagem for adquirida para utilização, esta será a mensagem mais atualizada ou recente existente no sistema em um determinado intervalo de tempo, ou seja, no macro-ciclo [2].

Restrições temporais são determinadas pelas relações de precedência entre mensagens, pelo período de cada estratégia de controle, pelo tempo de frescor, pelo tempo ocioso no barramento gerado pelo processamento dos blocos funcionais e restrições fim-a-fim [3] [4].

Restrições fim-a-fim são restrições temporais definidas pela estratégia de controle que são estabelecidas através dos requisitos temporais entre as entradas e saídas dos processos.

Uma estratégia de controle é composta por blocos funcionais distribuídos entre dispositivos e suas conexões lógicas como mostrado na figura 11.

Como pode ser observado pela figura 11, nas estratégias de controle todas as conexões lógicas entre blocos funcionais são conexões lógicas externas, onde cada conexão deverá possuir uma mensagem que será transmitida ao barramento, estabelecendo-se assim a comunicação entre blocos funcionais localizados em dispositivos diferentes. Como ressaltado anteriormente, quando os blocos funcionais com suas conexões lógicas estão localizadas em um mesmo dispositivo, estas conexões são chamadas de conexões lógicas internas, pois, suas mensagens não serão transmitidas pelo barramento, i.e., estas mensagens estarão confinadas

dentro do dispositivo indicando uma comunicação entre blocos funcionais locais.

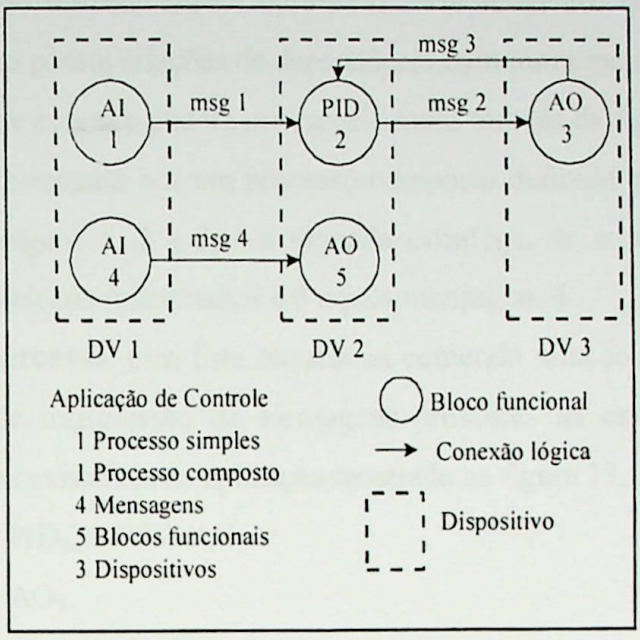


Figura 11 – Exemplo de uma aplicação de usuário

Ainda sendo, observada a figura 11, as relações de precedência entre mensagens e blocos funcionais podem ser levantadas. As relações de precedência entre mensagens são: *msg1 precede msg2* e *msg2 precede msg3*. As relações de precedência entre blocos funcionais são: *AI1 precede PID2* e *PID2 precede AO3* para primeira estratégia de controle, e *AI4 precede AO5* para segunda estratégia de controle.

Como pode ser observado pela figura 11, a relação de precedência entre *AO3* precedendo *PID2* não foi considerada, pois, esta iria gerar um ciclo na primeira estratégia de controle devido a realimentação entre *PID2* e *AO3*. Portanto, os blocos funcionais com saída de realimentação não são levados em consideração como precedentes de uma relação de precedência, pois, estes geram ciclos. Então, em função deste fator, durante a caracterização das relações de precedência não se deve considerar os sentidos reversos (realimentações), levando-se em conta somente os sentidos diretos do encadeamento de blocos funcionais e transmissões de mensagens, i.e., leva-se em conta a sequência de comunicação de mensagens ou processamento de blocos funcionais no sentido do bloco funcional produtor para o bloco funcional consumidor.

Neste trabalho as realimentações não foram escalonadas por falta de tempo, no entanto seu escalonamento torna-se claro, quando as respectivas mensagens oriundas das realimentações são escalonadas após seu bloco funcional gerador, o que determina o tempo de release que é dinâmico, e antes do término do microciclo da estratégia de controle ao qual esta mensagem é dependente. O tratamento a ser considerado para as realimentações é semelhante

ao tratamento dado à mensagem livre presente em um processo simples, porém somente será determinado seu release dinamicamente durante o escalonamento. Uma mensagem livre é uma mensagem que não possui relações de dependência com outra mensagem.

As estratégias de controle podem ser especificadas através da figura 11, como sendo: a primeira estratégia determinada por um processo composto definido pelos blocos funcionais 1, 2 e 3 e pelas mensagens 1, 2 e 3; e a segunda estratégia de controle por um processo simples definido pelos blocos funcionais 4 e 5 e pela mensagem 4.

Chama-se de **percurso**, uma lista sequencial contendo uma sequência de execução de blocos funcionais e de transmissão de mensagens presentes na estratégia de controle da aplicação. Os percursos existentes na aplicação mostrada na figura 11, são:

1) AI_1, MSG1, PID_2, MSG2, AO_3

2) AI_4, MSG4, AO_5

Como pode ser observado pelos percursos acima, no percurso 1 não está presente a mensagem 3 que é uma realimentação existente entre os blocos funcionais AO_3 e PID_2. Isto quer dizer que o percurso é obtido somente no sentido do caminho direto sem que sejam percorridas as realimentações, que podem gerar ciclos durante a caracterização temporal das mensagens.

Os blocos funcionais são modularizados através de funções que o dispositivo deve desempenhar. Eles são descarregados para o dispositivo através do barramento. As conexões lógicas são formadas por mensagens transmitidas entre os blocos funcionais da mesma estratégia de controle.

A garantia das restrições temporais da comunicação podem ser definidas, através da caracterização da janela de tempo de cada mensagem, determinada por um tempo de *release*, e um tempo de *deadline*. O tempo de ocupação da mensagem no meio é determinado através do tempo de transmissão da mensagem.

As mensagens no barramento e os blocos funcionais que serão executados nos processadores dos dispositivos devem respeitar as relações de precedência, para a adequada operação da aplicação. As precedências são as principais responsáveis pela sincronização de troca de variáveis, ou seja, troca de mensagens entre blocos funcionais.

O encadeamento de blocos funcionais, determina a ordem ou sequência de execução destes blocos durante o macro-ciclo.

O macrociclo nada mais é do que o intervalo de tempo necessário para o atendimento das estratégias de controle que se repetem periodicamente durante o ciclo operacional do sistema. Ele é obtido através do mínimo múltiplo comum dos períodos definidos para cada

estratégia de controle [2].

Por exemplo, com as estratégias de controle mostradas na figura 11, onde: a primeira estratégia de controle tem 3 blocos funcionais, um período de *300ms* e a segunda estratégia de controle tem 2 blocos funcionais, um período de *100ms*, tem-se um macrociclo de *300ms* que é o mínimo múltiplo comum entre *100ms* e *300ms*. Portanto, o macrociclo será composto de três vezes o menor período que seria *300ms/100ms*, indicando que a segunda estratégia de controle seria executada 3 vezes dentro do macrociclo e a primeira estratégia somente uma vez.

Como o tempo de transmissão das mensagens cíclicas é muito pequeno, não justifica-se a interrupção da transmissão da mensagem, ou seja, as mensagens não devem ser preemptadas. Portanto, quando uma mensagem cíclica inicia sua transmissão, esta não poderá ser preemptada por qualquer outra mensagem. Estas mensagens irão gerar processos simples e/ou processos compostos para cada estratégia de controle.

Um **processo simples** consiste de mensagens sem relação de precedência, chamadas de mensagens livres, acompanhadas de seus blocos funcionais.

Um **processo composto** consiste de mensagens de entrada(s) e saída(s) com relação de precedência entre mensagens, mas com as mensagens de saída tendo que esperar o processamento de seu respectivo bloco funcional. Portanto, enquanto estes blocos funcionais estiverem sendo processados o barramento fica ocioso, chamado de **tempo ocioso** [4]. Deste modo, outras estratégias de controle poderão acessar o barramento para a transmissão de suas mensagens.

Um processo composto depende das relações de precedência entre mensagens, pois, desta forma pode-se garantir o sequenciamento apropriado das mensagens que serão transmitidas.

A figura 12 mostra um processo simples, onde a mensagem 1 é transmitida sobre o barramento.

A figura 13 mostra um processo composto, onde as mensagens 2 e 3 necessitam esperar o processamento dos blocos funcionais PID(2) e AO(3).

Quando uma variável de entrada de um processo composto (com entrada(s) e saída(s)) termina de ser escalonada para transmissão no barramento, a variável de saída só poderá ser liberada para o escalonamento após o tempo de processamento do respectivo bloco funcional, pois, nenhuma variável de saída de um bloco poderá ser transmitida no barramento sem que seu processamento tenha sido concluído. Portanto, durante o tempo de processamento gera-se uma ociosidade na transmissão dos dados no barramento para a respectiva estratégia de

controle, durante o qual o barramento estará livre para transmitir variáveis referentes a outras estratégias de controle coexistentes no barramento [5].

Aplicações de usuários devem respeitar as consistências temporais absolutas e relativas, mas além disso, estas devem também gerar dados com as respectivas consistências. A relação entre entradas e saídas de conexões externas estabelecem restrições de tempo inerentes à aplicação, definidas como restrições fim-a-fim [5].

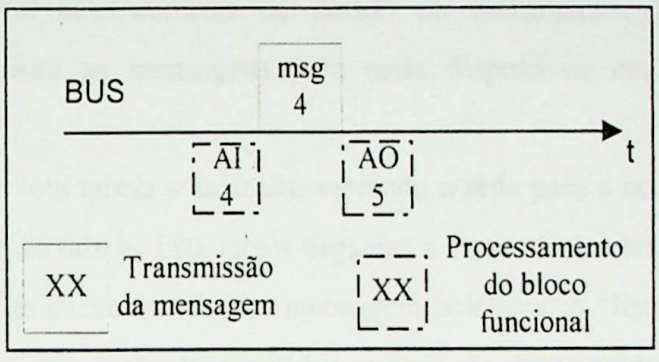


Figura 12 – Processo simples da estratégia de controle mostrada na figura 11

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca MAUÁ BIM

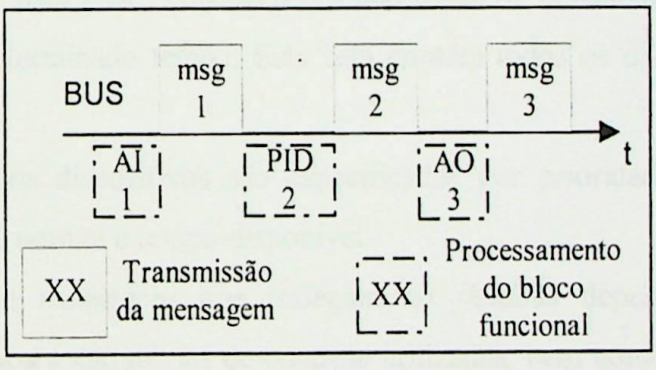


Figura 13 – Processo composto da estratégia de controle mostrada na figura 11

Em conformidade com estas características, desenvolveu-se um escalonador de blocos funcionais e mensagens no *fieldbus* que trata e atende as restrições temporais de processos de controle, garante a sequencialização de blocos funcionais e mensagens, permite a alocação dos blocos funcionais nos processadores dos dispositivos distribuídos pelo *fieldbus*, garante o tratamento e atendimento da multiplicidade de mensagens e blocos funcionais que possuam ciclos diferentes, e ainda caracteriza o sincronismo entre blocos funcionais produtores/consumidores em relação às mensagens transmitidas através do barramento de comunicação.

3. ALGORITMO DE ESCALONAMENTO

O algoritmo de escalonamento da comunicação deve determinar a ordem em que os dispositivos conectados ao sistema poderão transmitir seus dados pelo barramento. A sequência de comunicação (tabela de escalonamento) é importante para que se possa atender satisfatoriamente os requisitos da aplicação [15].

O algoritmo deve produzir uma tabela de escalonamento que é utilizada pelo *Link Active Scheduler – LAS* e deve ser respeitada pelos dispositivos. O LAS é um mecanismo centralizado responsável pelo controle de acesso ao barramento. Este lê a tabela de escalonamento e requisita as mensagens para cada dispositivo em instantes de tempo definidos pela tabela.

O LAS mantém esta tabela atualizada, varrendo a rede para a necessidade de inserção de um novo dispositivo na tabela. Esta tabela organiza a transmissão das mensagens cíclicas e uma "*lista viva*" organiza a transmissão das mensagens acíclicas. A "lista viva" é uma lista de mensagens enfileiradas que serão transmitidas através do barramento, quando ele estiver disponível. Neste caso, o LAS distribuirá uma mensagem especial para um determinado dispositivo existente na "lista viva", concedendo a permissão de transmissão no *fieldbus* para o dispositivo por um determinado tempo. Esta lista contém todos os dispositivos ativos do barramento.

As mensagens dos dispositivos são especificadas por prioridades de transmissão, definidas como: urgente, normal e tempo-disponível.

A quantidade de mensagens que trafegam no *fieldbus* depende diretamente da quantidade de dispositivos e estratégias de controle utilizadas, bem como, da quantidade de ligações externas feitas entre blocos funcionais. A quantidade de mensagens é indeterminada até o momento da configuração da aplicação do usuário no *fieldbus*. Quando a quantidade de mensagens é numerosa, várias opções de escalonamento podem ser definidas gerando um problema de explosão combinatória para a obtenção ou busca de uma solução adequada para o escalonamento das mensagens e blocos funcionais.

CAPÍTULO 4 – ANÁLISE DO ESTADO DA ARTE

Os problemas de escalonamento são mais estudados no campo da ciência da computação, como pode-se observar em Xu e Parnas [18], Liu e Layland [8], Shepard e Gagné [12], Peng, Shin e Abdelzaher [11].

Xu e Parnas [18] desenvolveram uma técnica de escalonamento de processos com tempos de *release*, *deadline*, relações de precedência, e exclusão. Shepard e Gagné [12] modificaram o algoritmo de Xu e Parnas para que desta forma pudessem utilizá-lo em aplicações com múltiplos processadores executando demanda de processos e processos periódicos preemptáveis. Em Liu e Layland [8] são apresentados alguns algoritmos de escalonamento para multiprogramação em ambientes de *hard-real-time*.

O atendimento às restrições temporais durante o escalonamento de sistemas *hard-real-time* no *fieldbus* deve ser garantido. Esta necessidade foi levantada nos trabalhos de Xu e Parnas [17], Decotignie e Raja [3], Vasques e Juanole [16], Franco[4], etc.

O problema inicial encontrado com o caso especial do *fieldbus* é a utilização de um único barramento para a comunicação dos dispositivos. Estes dispositivos dependem deste meio de comunicação para transmitir suas mensagens que são produzidas pela aplicação, a qual apresenta-se distribuída em um conjunto de dispositivos conectados ao barramento. No entanto, com a distribuição da aplicação sobre o barramento, gera-se um outro problema devido à necessidade do escalonamento e sequencialização dos módulos ou pedaços que compõem a aplicação nos dispositivos, permitindo ainda a existência da sincronização com o escalonamento das mensagens que trafegam pelo barramento, ou seja, com o escalonamento da comunicação.

Os algoritmos clássicos não são bem sucedidos para solucionar este problema, porque, eles não otimizam o uso do barramento, bem como, não utilizam relações de precedência, tempos de frescor, tempos ociosos, consistências temporais, e etc. Além disto, eles não levam em consideração o relacionamento de dois tipos de escalonamentos distintos e completamente relacionados, como é o caso do escalonamento de um recurso compartilhado para a comunicação e o escalonamento de recursos distribuídos (processadores dos dispositivos) dependentes do escalonamento da comunicação no *fieldbus*.

Atualmente algumas propostas foram desenvolvidas para o escalonamento do *fieldbus*, mas na realidade a preocupação principal da maioria destes trabalhos busca a minimização da quantidade de mensagens na transmissão, não levando em conta a dependência temporal dos

dados em um sistema *fieldbus*. Nestes trabalhos o problema da consistência e restrições temporais devem ser tratadas pelos processos de aplicação, os quais deverão identificar quando um dado está desatualizado ou não.

Por outro lado, o trabalho apresentado por Franco [5] preocupa-se especificamente com a consistência temporal e restrições fim-a-fim envolvidas na troca de mensagens entre dispositivos *fieldbus*. Com o aumento da complexidade e quantidade de estratégias de controle contidas nas aplicações do usuário, deve-se ter em mente a garantia da consistência temporal e restrições fim-a-fim da aplicação, pois, a falha ou a falta destas características fazem com que os dados enviados pela rede não representem o real estado do processo, comprometendo a aplicação. Tais deficiências existentes hoje no escalonador, somente são aceitáveis, pois as aplicações implementadas são compostas por poucos processos. No entanto, a tendência natural é o crescimento desta quantidade, pois, cada vez mais se exigirá um melhor aproveitamento do barramento para as transmissões e dos processadores dos dispositivos para a execução dos blocos funcionais.

Portanto, até o presente momento, os algoritmos utilizados em *hard real time* não satisfazem todos os requisitos impostos pelas restrições temporais existentes no *fieldbus* para a garantia do escalonamento da comunicação das mensagens e da execução dos blocos funcionais.

Neste trabalho, baseado nas restrições temporais apresentadas pelo trabalho de Franco [5], pode-se especificar as restrições temporais das mensagens em função das restrições temporais do blocos funcionais de uma aplicação, fazendo com que fossem ainda determinadas as restrições fim-a-fim dos blocos funcionais e mensagens, restrições da capacidade dos processadores dos dispositivos *fieldbus*, e o comportamento de mensagens e blocos funcionais nos percursos impostos pela aplicação desenvolvida pelo usuário.

1. TRABALHOS RELACIONADOS

Como o campo de desenvolvimento dos algoritmos de escalonamento em áreas afins é muito amplo, na seção seguinte somente serão discutidos os trabalhos relacionados com o desenvolvimento do escalonador *fieldbus*, pois, pretende-se visualizar a origem das técnicas utilizadas na elaboração do trabalho proposto. Portanto, objetiva-se com isto somente apresentar os pontos importantes de outros trabalhos que tiveram de ser absorvidos, para que fosse elaborado um escalonador que atendesse às restrições temporais impostas pela

comunicação do *fieldbus*.

1.1. Escalonamento de Processos em Um Único Processador

No caso do escalonamento de processos com tempos de *release*, *deadline* e relações de precedência e exclusão desenvolvido por Xu e Parnas [18], objetivou-se a obtenção de um escalonamento em *pre-run-time* de um único processador para um respectivo conjunto de processos, de tal forma que cada um destes processos fosse executado depois de seu tempo de *release*, completando seu tempo de computação antes de seu tempo de *deadline*.

Como apresentado por Xu e Parnas [18], relações de precedência podem existir entre segmentos de processos, onde alguns segmentos requerem informações que são produzidas por outros segmentos de processos. Relações de exclusão são mantidas entre pares de segmentos de processos que são excludentes, i.e., um segmento da relação exclui a execução de um outro segmento contido na relação, de tal forma que este outro segmento não possa interromper a execução do segmento pertinente à relação, prevenindo-se assim a ocorrência de erros causados pelo acesso simultâneo dos recursos compartilhados [18].

Deste modo, cada processo além de ser caracterizado por uma sequência de segmentos, possuindo tempos de *release*, de computação e *deadline*, eles possuem relações de precedência e exclusão arbitrárias definidas em pares ordenados entre os segmentos dos processos, de tal forma que para que o referido processo possa ser executado, este deverá respeitar todas as relações de precedência e exclusão entre seus respectivos segmentos [18].

Portanto, baseando-se no trabalho de Franco [4] que usou Xu e Parnas [18] para o escalonamento de processos em um único processador, pode-se estipular os pontos essenciais para a caracterização dos instantes de tempo importantes ou restrições temporais das mensagens existentes no barramento de comunicação entre os dispositivos *fieldbus*. Refletiu-se no trabalho de Franco [4] a seguinte correspondência já apresentada no capítulo antecessor:

a) único processador → único barramento
b) processo com um segmento → processo simples (mensagem única)
c) processo composto por mais de um segmento → processo composto (mensagens encadeadas)
d) tempo de computação de cada segmento de processo → tempo de transmissão de cada mensagem
e) tempo de *release* do segmento → tempo de *release* da mensagem

f) *deadline* do segmento → *deadline* da mensagem

g) relação de precedência dos segmentos → relação de precedência das mensagens

A seguinte correspondência possibilita que o escalonamento da comunicação no *fieldbus* esteja relacionada com o procedimento de escalonamento de processos em um único processador apresentado por Xu e Parnas.

Portanto, baseado no trabalho de Franco [4], pode-se determinar as restrições temporais das mensagens, impostas pela comunicação no *fieldbus*. Finalmente, o comportamento e aspectos referentes às restrições temporais das mensagens caracterizado no trabalho de Franco [4], teve que ser adaptado e redefinido dinamicamente para este trabalho, pois agora, tem-se a necessidade da obtenção de um escalonamento factível atendendo-se as restrições temporais das mensagens em conformidade com o escalonamento dos blocos funcionais respeitadas suas restrições temporais.

1.2. ESCALONAMENTO DE PROCESSOS EM MÚLTIPLOS PROCESSADORES

Shepard e Gagné [12] em seu trabalho objetivaram o sequenciamento e alocação factível de processos através de um algoritmo de escalonamento *pre-run-time*. Este algoritmo foi baseado no algoritmo proposto por Xu e Parnas [18] sendo desenvolvido para aplicações com múltiplos processadores, onde seriam executados processos preemptáveis possuindo tempo de *release*, tempo de computação, tempo de *deadline*, e relações de precedência e exclusão. A busca exaustiva para a obtenção de uma solução também foi considerada no trabalho de Shepard e Gagné, sendo drasticamente reduzida através da utilização da técnica chamada de *branch and bound* [12].

Segundo a proposta de Shepard e Gagné [12], um escalonamento factível gerado pelo seu algoritmo, atende às especificações de temporização dos processos e ainda todas as relações de precedência e exclusão entre pares de processos. Shepard e Gagné enfatizam que muitas vezes na prática não são considerados os requisitos temporais no desenvolvimento dos sistemas *hard-real-time*. Esta atitude pode então levar o sistema a falhas no que diz respeito ao atendimento das restrições temporais, bem como, quando existe a necessidade de se efetuar a manutenção do sistema, esta manutenção torna-se mais difícil e muito cara de ser efetuada [12].

Portanto, para que os requisitos necessários fossem atendidos, Shepard e Gagné [12]

estipularam o modelamento da aplicação através de um conjunto de processos com suas respectivas especificações temporais e restrições de sincronização. Partindo-se deste ponto com a aplicação modelada, o algoritmo proposto por Shepard e Gagné é processado *off-line* obtendo ao final de sua execução um escalonamento *pre-run-time* que especifica um escalonamento *run-time* para cada processador do sistema. Este escalonamento gerado em *pre-run-time* reduz então significantemente o escalonamento *run-time* dos recursos e o chaveamento de processos entre processadores [12].

Resumindo, o algoritmo proposto por Shepard e Gagné [12] foi desenvolvido para aplicações em múltiplos processadores atendendo demanda de processos periódicos preemptáveis possuindo tempos de *release*, tempos de computação, tempos de *deadline*, e relações de precedência e exclusão. Finalmente, o algoritmo objetiva a obtenção de um escalonamento factível para cada processador do sistema atendendo os seus requisitos temporais.

Os tratamentos do escalonamento *pre-run-time* e *run-time* feitos por Shepard e Gagné, são os mesmos utilizados por este trabalho para o atendimento das restrições temporais e restrições de sincronização existentes no sistema *fieldbus*, sendo considerado o comportamento do algoritmo desenvolvido por eles para o escalonamento de múltiplos processadores, como é o caso dos processadores dos dispositivos *fieldbus* que estão distribuídos pelo barramento de comunicação. Comparativamente com o sistema *fieldbus*, o trabalho discutido a seguir realça um tipo de tratamento para o escalonamento utilizado em sistemas de tempo real distribuídos.

O trabalho de Peng, Shin e Abdelzaher [11] objetiva a busca por uma solução ótima para o problema de alocação da comunicação de tarefas periódicas, disponibilizadas para o processamento em nós heterogêneos de sistemas de tempo real distribuído. Segundo Peng, Shin e Abdelzaher [11] o senso de uma solução ótima ocorre com a minimização do tempo final de alocação de uma tarefa em relação a seu *deadline* que pode conter um atraso, quando o tempo final de alocação é maior que o *deadline* da tarefa. O modelamento do sistema de tarefas é feito por um grafo de tarefas, no qual os módulos de computação e comunicação, atrasos de comunicação, e relações de precedência entre tarefas são claramente identificados [11].

Em função do cálculo apresentado pelo grafo de tarefas e através do algoritmo de busca *branch and bound* podem-se definir os nós da árvore de busca que serão abertos e percorridos, isto é, quando existir uma solução ótima ela será atingida através do caminho a ser percorrido pelos nós abertos existentes a cada nível inferior da árvore de busca.

O grafo de tarefas é uma técnica inteligente para o desenvolvimento de um banco de dados, aonde todas as informações cruciais podem ser obtidas, além de facilitar a representação do sequenciamento e sincronização entre tarefas no sistema. A idéia apresentada por este tipo de desenvolvimento aplicado no trabalho de Peng, Shin e Abdelzaher foi adaptado para o desenvolvimento de um procedimento absorvido pelo algoritmo proposto por esta dissertação.

2. PARADIGMA DO ESCALONAMENTO ESTÁTICO X DINÂMICO

O atendimento das restrições temporais garante que quando um processo estiver sendo executado, ele será finalizado antes de seu respetivo *deadline*. Isto deve ser respeitado por todos os processos do sistema, pois o não atendimento origina uma falha ou problema que pode levar o sistema a uma intolerável degradação de estado, podendo em alguns casos causar uma perda catastrófica da atividade ou propriedade do sistema [17].

Xu e Parnas definem em [17] que tarefas em *hard-real-time* são especificadas como um conjunto de processos sequenciais cooperativos. Nestes sistemas existem dois tipos de processos: processos periódicos e processos assíncronos. Alguns exemplos de sistemas *hard-real-time* são: controle de vôo, controle de tráfego, sistemas táticos embutidos em aplicações militares, controle de produção, robótica, controle de planta nuclear, etc.

Processo periódico consiste de uma computação que deve ser executada repetidamente, uma vez a cada período de tempo (ex.: leitura de sensores e atualização do estado corrente de variáveis internas e saídas). Estes processos possuem *hard deadlines*.

Processo assíncrono ou acíclico consiste na execução de uma computação requisitada por eventos aleatórios internos ou externos, que normalmente não acontecem periodicamente.

Finalizada a execução de um algoritmo de escalonamento, obtém-se um escalonamento para o sistema como resultado do algoritmo. Xu e Parnas definem escalonamento, como sendo "um mapeamento de um conjunto de possibilidades infinitas de unidades de execução de um processo para um conjunto de possibilidades infinitas de unidades de tempo do processador sobre um ou mais eixos de tempo de processador".

Portanto, pode-se dizer que escalonamento é um cronograma de ações, especificadas através de uma previsão temporal definida em função das características dos processos; para que durante a atividade do sistema, estes processos atendam a consistência temporal da comunicação no *fieldbus*. Como apresentado por Xu e Parnas [17], os escalonamentos feitos

em *pre-run-time* (escalonamentos estáticos) são capazes de atender às restrições temporais, enquanto que escalonamentos feito em *run-time* (escalonamentos dinâmicos) não podem garantir o determinismo temporal imposto pelas restrições temporais.

A figura 14 ilustra o comportamento apresentado durante o processo de escalonamento *pre-run-time* ou *off-line*. A figura 15 ilustra o comportamento apresentado durante o processo de escalonamento *run-time*.

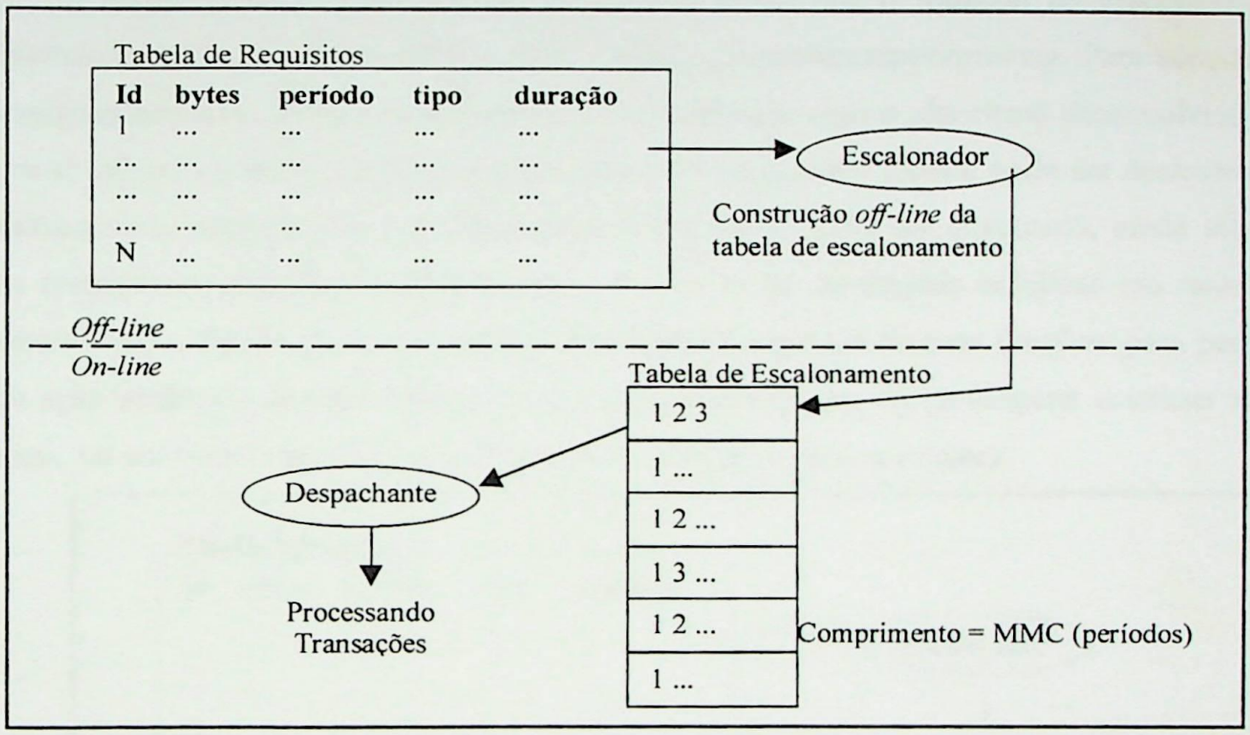


Figura 14 – Diagrama funcional de escalonamento centralizado estático baseado em tabela [1]

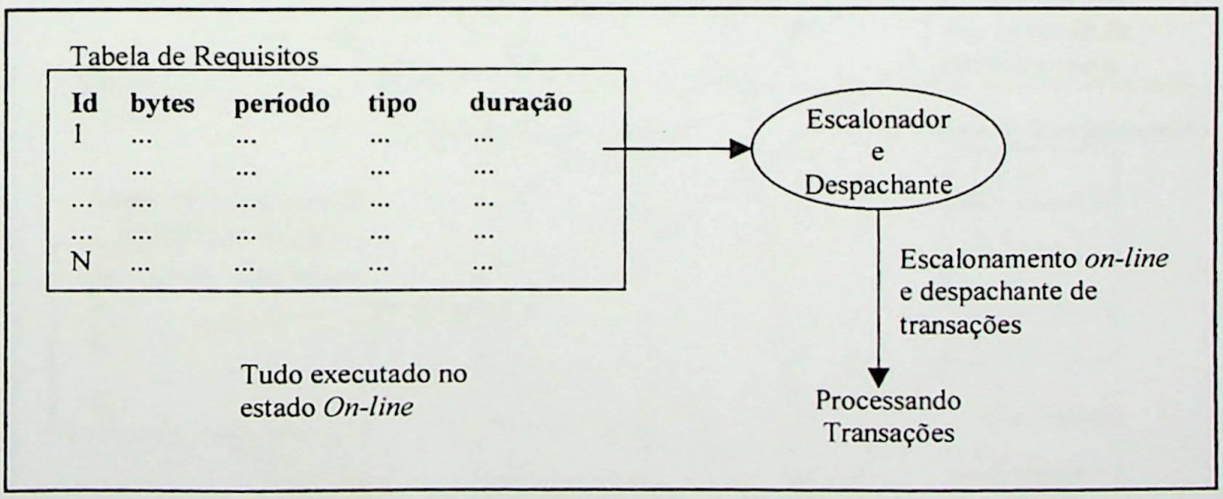


Figura 15 – Diagrama funcional de escalonamento centralizado dinâmico [1]

O problema encontrado com os escalonamentos feitos em *run-time*, é que estes não possuem a priori o conhecimento sobre todas as características dos processos que devem ser escalonados, então fica impossível garantir que todas as restrições temporais possam ser

satisfeitas. Isto acontece com o não atendimento do *deadline* do processo que está sendo escalonado *on-line* ou com os próximos processos que poderão perder seus *deadlines*.

Além deste problema, o tempo necessário para que um escalonador *run-time* compute todas as restrições e opções de escalonamentos factíveis que poderão satisfazer às restrições, é muito grande comparado ao tempo disponibilizado pelo escalonamento feito em *pre-run-time*.

Portanto, este trabalho propõe, do mesmo modo que o trabalho de Franco[4], um escalonamento intermediário entre as duas filosofias descritas anteriormente. Para adequar as mensagens acíclicas, ambos os trabalhos são compatíveis com o algoritmo desenvolvido por Alencar [6], que calcula o último instante que cada mensagem cíclica pode ser deslocada no escalonamento estabelecido pelo algoritmo *pre-run-time*, para ser executada, ainda atender suas restrições e aumentar a possibilidade de inserir as mensagens acíclicas em *run-time*. Como mostra a figura 16, o escalonador descentraliza algumas de suas funções, para permitir uma ação dinâmica do algoritmo proposto, possibilitando que as mensagens acíclicas sejam aceitas, na medida do possível, privilegiando as restrições das mensagens.

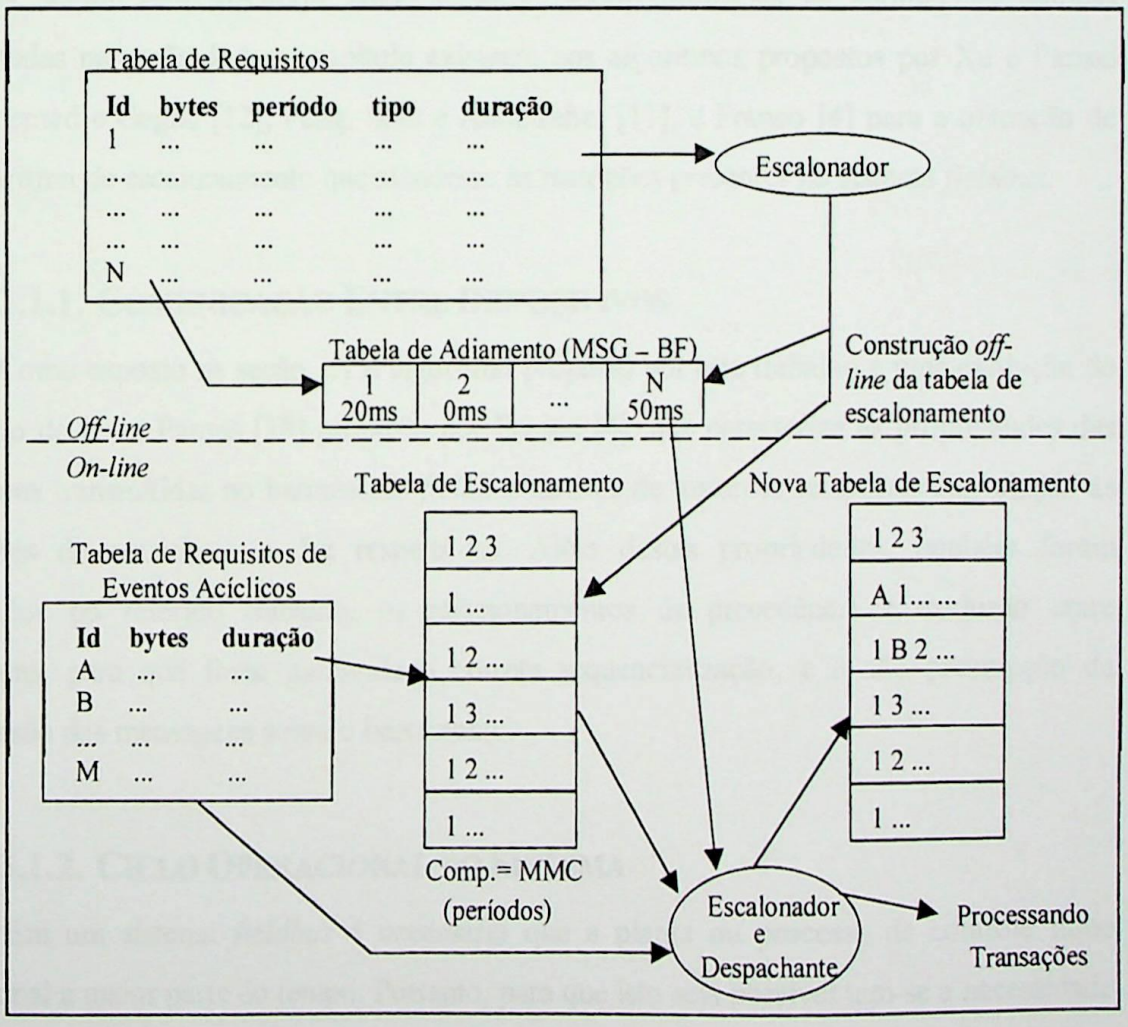


Figura 16 – Diagrama funcional de escalonamento descentralizado dinâmico

3. DESCRIÇÃO SUCINTA DO ALGORITMO PROPOSTO POR ESTE TRABALHO

O algoritmo proposto baseia-se no trabalho de Franco [4] que faz o escalonamento da comunicação entre dispositivos do *fieldbus* e objetiva a ampliação do referido trabalho, buscando fazer o escalonamento dos módulos de programa (blocos funcionais) da aplicação nos processadores dos dispositivos em sincronismo com o escalonamento da comunicação *fieldbus*. Estes módulos ou blocos funcionais, junto com a comunicação entre eles, compõem a aplicação do usuário.

O algoritmo proposto baseou-se nas técnicas apresentadas em trabalhos de outros autores descritos anteriormente (Franco [4], Xu e Parnas [18], Shepard e Gagné [12], e Peng, Shin e Abdelzaher [11]).

3.1. DISCUSSÃO SOBRE A ABORDAGEM TEÓRICA FEITA NOS TRABALHOS ANTERIORES UTILIZADA PELO ALGORITMO PROPOSTO

A seguir será discutido como e aonde foram utilizadas as abordagens teóricas apresentadas na seção 1 deste capítulo existente nos algoritmos propostos por Xu e Parnas [18], Shepard e Gagné [12], Peng, Shin e Abdelzaher [11], e Franco [4] para a obtenção de um algoritmo de escalonamento que atendesse às restrições presentes no sistema *fieldbus*.

3.1.1. COMUNICAÇÃO ENTRE DISPOSITIVOS

Como exposto na seção 1.1 o algoritmo proposto por este trabalho é uma evolução do algoritmo do Xu e Parnas [18] proposto por Franco [4], que caracteriza as propriedades das mensagens transmitidas no barramento *fieldbus* através de instantes temporais em relação às aplicações do usuário e o faz respeitá-las. Além destas propriedades, também foram respeitados no referido trabalho, os relacionamentos de precedência e exclusão entre mensagens, para que fosse garantida a correta sequencialização, e a não preempção da transmissão das mensagens sobre o barramento.

3.1.2. CICLO OPERACIONAL DO SISTEMA

Em um sistema *fieldbus* é necessário que a planta ou processo de controle fique operacional a maior parte do tempo. Portanto, para que isto seja possível tem-se a necessidade de que os procedimentos e configurações da planta ou processo sempre estejam ativos, para que a planta ou processo de controle funcione adequadamente. Baseado nas idéias propostas

por Shepard e Gagné [12], e Franco [4] necessitou-se então que as mensagens transmitidas e os blocos funcionais distribuídos nos dispositivos ficassem presentes o tempo todo durante o ciclo de atividade do sistema. Este ciclo de atividade do sistema garante que a aplicação definida pelo usuário seja executada ciclicamente, fazendo com que exista a necessidade da periodicidade de mensagens transmitidas no barramento e a periodicidade da execução dos blocos funcionais nos dispositivos.

3.1.3. INTEROPERABILIDADE ENTRE APLICAÇÕES

O relacionamento existente entre blocos funcionais de uma ou mais aplicações num mesmo sistema *fieldbus* composto por um número limitado de dispositivos, exige inicialmente que durante o escalonamento dos blocos funcionais nos processadores dos dispositivos seja necessária a definição das propriedades temporais e da sincronização entre mensagens e blocos funcionais presentes nas aplicações. Para que isto ocorra existe a necessidade de uma análise pormenorizada dos componentes envolvidos nos relacionamentos, de tal forma que, se possam definir as características temporais mínimas e a existência de limites de tempos conjugados do sistema. Atendidos estes limites, pode-se então escalonar a mensagem e/ou bloco funcional, de tal forma que atendam às restrições impostas pela(s) aplicação(ões).

Baseado no trabalho de Peng, Shin e Abdelzaher [11], o sequenciamento e a sincronização entre blocos funcionais e mensagens são garantidos através de um procedimento que levanta os percursos existentes na(s) aplicação(ões) e determina todos os relacionamentos existentes entre blocos funcionais e mensagens.

CAPÍTULO 5 – ALGORITMO PROPOSTO

O algoritmo proposto foi desenvolvido a partir do trabalho apresentado por Franco [4] que definiu as restrições temporais importantes, presentes no escalonamento de mensagens cíclicas e acíclicas da comunicação no fieldbus para as aplicações de *hard real-time*, em função dos tempos de execução dos blocos funcionais e precedências entre eles. O algoritmo proposto por este trabalho adaptou o algoritmo original, incrementando-o com o escalonamento dos blocos funcionais nos processadores distribuídos, respeitando as restrições temporais estabelecidas pela aplicação do usuário. Este algoritmo objetiva garantir a sincronização e sequencialização dos blocos funcionais, através do escalonamento das mensagens. Deste modo, além de escalonar as mensagens, o algoritmo escalona os blocos funcionais que estão inter-relacionados pela estratégia de controle **concorrentemente**. O problema torna-se mais complexo, porém a solução encontrada atende às restrições temporais dos dois escalonamentos, evitando-se escalonar os dois conjuntos (blocos funcionais e mensagens) separadamente e verificar, a posteriori, se eles estão adequadamente relacionados.

1. UNICIDADE DO ALGORITMO PROPOSTO

A diferença existente entre o algoritmo proposto e os demais algoritmos desenvolvidos, baseia-se no atendimento das restrições temporais impostas pelo *fieldbus*, ao satisfazer a sequencialização e sincronização entre blocos funcionais, estabelecidas pela aplicação do usuário. Portanto, durante a execução do algoritmo geram-se dois tipos distintos de escalonamento simultaneamente: o escalonamento das mensagens e o escalonamento dos blocos funcionais distribuídos nos processadores dos dispositivos *fieldbus*.

Nenhum trabalho apresentado até o momento, atendeu às restrições temporais dos blocos funcionais e de suas mensagens relacionadas conjuntamente, ou mesmo separadamente.

O escalonamento das mensagens é similar ao escalonamento de processos num processador (recurso compartilhado), como apresentado por Xu e Parnas [18]. Porém, o escalonamento dos blocos funcionais assemelha-se ao escalonamento de tarefas em recursos distribuídos como apresentado por Shepard e Gagné [12], e Peng, Shin e Abdelzaher [11].

O algoritmo permite o desenvolvimento de configurações de aplicações complexas, onde existam uma grande quantidade de mensagens e blocos funcionais, ou quando existir a

necessidade de utilização de um único sistema *fieldbus* rodando mais de um tipo de aplicação de controle ou várias estratégias de controle.

Como em Franco [4], o algoritmo proposto atende às configurações das aplicações do usuário, onde existam mensagens cíclicas e acíclicas, num único segmento do sistema *fieldbus* rodando mais de uma estratégia de controle.

Os blocos funcionais estão limitados pela quantidade de dispositivos disponíveis no sistema, e a transmissão das mensagens estão limitadas pela sobrecarga da largura de banda do canal de comunicação, o que normalmente pode não acontecer, pois, a quantidade de mensagens e o tempo de ocupação no barramento são muito pequenos. Portanto, quando uma solução de escalonamento é obtida todas as restrições temporais puderam ser atendidas, bem como, estes pontos foram levados em consideração durante o processo de escalonamento para que o aproveitamento do sistema torne-se mais eficiente.

Quando existir a necessidade de se configurar aplicações de controle concorrentes, ou ter várias estratégias de controle rodando em um mesmo *fieldbus*; durante o decorrer do processo de escalonamento, as mensagens e blocos funcionais das aplicações ou estratégias de controle serão intercaladas como para o escalonamento de mensagens em Franco [4], de tal modo que o barramento de comunicação e os processadores do dispositivos possam conter uma maior quantidade de mensagens e blocos funcionais. Isto ocasionará, uma otimização da utilização do sistema e um aproveitamento dos dispositivos existentes sobre o barramento de comunicação, fazendo com que o escalonamento gerado pelo algoritmo, suporte a existência de aplicações e estratégias de controle concorrentes em um mesmo sistema, atendendo ainda extensões do segmento do sistema.

2. FUNCIONALIDADE GLOBAL DO ALGORITMO PROPOSTO

Inicialmente o algoritmo proposto traduz as informações da aplicação de tal forma que sejam definidas todas as características temporais das mensagens. A partir destas informações são determinadas as relações de precedência, relacionamentos entre mensagens, relacionamentos entre blocos funcionais e relacionamentos entre blocos funcionais e mensagens, ou seja, um banco de dados construído para que possa ser consultado pelo escalonador e pelo mecanismo de liberação de blocos funcionais e mensagens.

A etapa seguinte ocorre com o escalonamento das mensagens, sendo que para cada mensagem a ser escalonada o mecanismo de liberação de blocos funcionais e mensagens é

Escola Federal de Engenharia
Biblioteca
MAUÁ
BIM

utilizado. Somente será escalonada a mensagem que atender todas as características temporais e já tiver seus relacionamentos com mensagens e blocos funcionais liberados. Escalonada a mensagem, os blocos funcionais que se relacionam com esta mensagem já serão automaticamente escalonados, desde que estes blocos funcionais não possuam outros relacionamentos que ainda não tenham sido liberados pelo mecanismo. O mesmo ocorre com as outras mensagens que possuam relacionamento com a mensagem escalonada: elas somente serão escalonadas a partir do momento em que sejam satisfeitos todos os relacionamentos existentes entre estas mensagens e blocos funcionais.

Terminado o escalonamento das mensagens e de seus blocos funcionais correspondentes, pode-se avaliar se estes escalonamentos satisfazem todas as restrições temporais da aplicação. Caso não satisfaça, um novo escalonamento de mensagens e blocos funcionais é efetuado a partir do escalonamento gerado até a obtenção de um escalonamento factível, ou seja, que atenda todas as restrições temporais da aplicação.

3. TÓPICOS IMPORTANTES SOBRE O COMPORTAMENTO DO ALGORITMO PROPOSTO

Durante o processo de escalonamento utilizam-se algumas técnicas apresentadas por outros trabalhos, como discutido no capítulo 4.

A interoperabilidade entre aplicações distintas ou o atendimento a uma aplicação mais complexa gera a necessidade da implementação dentro do algoritmo de um **mecanismo de liberação de blocos funcionais e mensagens**, que faça o controle do escalonamento da comunicação e do escalonamento dos processadores de acordo com os relacionamentos existentes, de tal modo que os componentes relacionados sejam atendidos a tempo de serem escalonados, prezando ainda o atendimento de todos os outros componentes existentes nas outras aplicações. Este atendimento garante que todas as restrições temporais sejam satisfeitas.

Baseando-se nos trabalhos de Peng, Shin e Abdelzaher [11] e de Tanembaum [14] pode-se desenvolver ainda um mecanismo de liberação com a característica descrita anteriormente, para a liberação dos blocos funcionais e mensagens de aplicações relacionadas ou aplicações complexas. Os blocos funcionais, depois de liberados, são então colocados em filas. Estas filas representam os múltiplos processadores dos dispositivos, de tal forma que sua execução seja sequencializada e correlacionada com outros blocos funcionais através do

mecanismo de liberação.

Como os valores dos *releases*, dos *deadlines* e as relações de precedência (mensagem 1 *precede* mensagem 2) relativos às mensagens interferem nos respectivos valores e relações de precedência dos blocos funcionais (e vice-versa), o algoritmo ajusta os valores e relações referentes às mensagens em função da influência dos valores e relações referentes aos blocos funcionais. Então, o algoritmo passa a considerar somente os valores e relações referentes às mensagens, pois, implicitamente nestes valores e relações já são considerados os valores e relações referentes aos blocos funcionais. Portanto, sendo atendida a consistência temporal das mensagens, automaticamente serão atendidas as restrições temporais dos blocos funcionais, pois, estes estão intimamente ligados com a produção ou geração de mensagens e suas respectivas temporizações.

As mensagens devem respeitar os tempos de *release*, *deadline* e as relações de precedência das estratégias de controle do usuário, escolhidas durante o escalonamento através do algoritmo **EDF** – *Earliest Deadline First algorithm.*

A heurística EDF seleciona a mensagem deste grupo em função de seu *deadline*, ou seja, a mensagem que possuir o menor *deadline* terá prioridade para a transmissão sobre o barramento.

A heurística EDF escalona as mensagens no primeiro instante de tempo possível. Consequentemente este concentrará as mensagens no início do macro-ciclo [6].

A cada instante de tempo t, quando o escalonador encontra uma mensagem que atende a todas as restrições temporais e às relações de precedência, agrupa-a com todas as possíveis mensagens que já podem ser escalonadas no referido instante de tempo. Portanto, durante o processo de escalonamento para cada instante de tempo decorrido tem-se um novo grupo de mensagens selecionadas, as quais formam um grupo de **mensagens elegíveis**.

As restrições temporais do grupo das mensagens elegíveis, estão atendidas, quando para cada mensagem contida no grupo de mensagens elegíveis, estas possuam seu *release* menor ou igual ao instante de tempo considerado, tendo seu *deadline* maior que este instante de tempo e que ainda satisfizerem suas relações de precedência.

A mensagem eleita somente será então escalonada, se esta tiver atendido todos os relacionamentos existentes entre blocos funcionais e mensagens, i.e., se o **mecanismo de liberação de blocos funcionais e mensagens** tiver liberado todos os blocos funcionais relacionados com esta mensagem e as mensagens que se relacionem com a mensagem eleita. Isto significa que os blocos funcionais que devem ser executados antes da transmissão da mensagem eleita, e as transmissões das mensagens relacionadas com a mensagem eleita foram

escalonadas, de tal forma que neste instante a mensagem poderia ser transmitida para o barramento de comunicação, sem prejudicar a sequencialização e a sincronização entre blocos funcionais e mensagens. Portanto, neste instante de tempo a mensagem elegível é escalonada.

Caso a mensagem elegível pelo EDF não tenha sido liberada pelo mecanismo de liberação de mensagens e blocos funcionais, então seu escalonamento será adiado. Se durante a eleição da mensagem pelo EDF ocorrer um empate entre mensagens com um mesmo tempo de *deadline*, a mensagem escolhida será aquela que possuir o maior tempo de transmissão, pois, ela terá maior risco de perder o *deadline* se seu escalonamento ficar para depois.

Durante o processo de escalonamento das mensagens, somente são consideradas como mensagens válidas as mensagens que possuem *release* maior ou igual a zero, as quais podem então fazer parte do grupo selecionado de mensagens elegíveis, ou seja, as mensagens que possuem *releases* dinâmicos (mensagens de saída), fixados durante o escalonamento somente poderão ser eleitas quando estas possuírem um *release* definido dentro da faixa aceitável (maior ou igual a zero). Uma mensagem válida que não atende às características temporais ainda não está localizada dentro de sua **janela de tempo**. Porém, uma mensagem que não é válida possui *release* dinâmico e, portanto, ainda não tem todas as suas características temporais definidas.

Resumindo, os *releases* das mensagens de saídas são fixados dinamicamente durante o processo de escalonamento, e só serão válidos os que momentaneamente apresentarem *release* maior ou igual a zero. Portanto, somente são consideradas para serem mensagens elegíveis, as mensagens que durante o processo de escalonamento tiverem *releases* válidos.

Uma mensagem gerada por um bloco funcional que não possui mensagens e blocos funcionais precedentes a sua execução, será mencionado a partir de agora como mensagem de entrada. Analogamente, as mensagens geradas por blocos funcionais que possuem mensagens e blocos funcionais precedentes a sua execução, serão mencionadas a partir de agora como mensagens de saída.

Os processos compostos precisam passar pelo **processo de validação** de *releases*, porque a(s) mensagem(ns) de saída de um processo somente poderá(ão) ser uma mensagem elegível, depois do escalonamento de sua(s) mensagem(ns) de entrada. Isto implica que as características temporais dos processos compostos não poderão ser especificadas totalmente antes da execução do processo de escalonamento, mas somente durante o processo de escalonamento das mensagens.

Releases e *deadlines* dinâmicos são empregados pelos processos compostos, pois possuem mensagens dependentes de outras mensagens. Portanto, as mensagens consecutivas

à(s) mensagem(ns) elegível(is) do(s) processo(s) composto(s) que por enquanto ainda não são mensagens válidas, somente terão seus *releases* e *deadlines* definidos a partir do momento que a(s) mensagem(ns) elegível(is) do(s) processo(s) composto(s) forem escalonadas, i.e., possuam um instante de tempo fixo para sua transmissão sobre o barramento. Então, nem todos os *releases* e *deadlines* dos processos compostos podem ser definidos completamente, pois, os *releases* das mensagens de saída, que ocorrem posteriormente à mensagem elegível do processo (mensagem(s) de entrada) não podem ser determinados antes que seja iniciado o escalonamento em cada nó da árvore de busca. Além disso, durante o escalonamento de cada nó, os *releases* das mensagens de saída serão gerados dinamicamente para cada nó da árvore de busca, fazendo com que no início do escalonamento **de cada nó**, estes parâmetros sejam invalidados (menores que zero), para que as novas características temporais do nó das mensagens de saída sejam ajustadas, validando-as. Para os *deadlines*, estes podem ou não ser fixados antes ou durante o escalonamento, dependendo dos padrões de aplicação do sistema *fieldbus*.

A partir do momento que a mensagem elegível do processo foi escalonada, os *releases* das mensagens de saída se tornarão válidas, podendo somente ser escalonadas quando seu(s) respectivo(s) blocos funcionais forem processado(s), portanto liberados através do **mecanismo de liberação de blocos funcionais e mensagens**. Isto significa que antes desta mensagem ser transmitida, um tempo de transmissão ocioso imposto pela execução do(s) respectivo(s) bloco(s) funcionai(s) sobre o processador dos dispositivos será gerado no barramento de comunicação pela respectiva estratégia de controle, que contém esta mensagem. Neste caso, o barramento estará livre para transmitir as mensagens de outras estratégias de controle coexistentes no mesmo sistema *fieldbus*.

A pesquisa por um escalonamento que atenda às restrições temporais, à sequencialização e sincronização entre mensagens e blocos funcionais, pode gerar uma explosão combinatória durante a busca de um escalonamento satisfatório que atenda a todos estes parâmetros. A preocupação com a ocorrência de uma possível explosão combinatória gerada pelo algoritmo é reduzida através da utilização de uma técnica de busca chamada de *branch and bound*. Nela, a solução ótima ou factível para a aplicação do usuário será a solução onde o **custo** (o atraso na transmissão de todas as mensagens) for menor ou igual a zero. O valor do atraso é calculado pela diferença entre o instante final de transmissão e o *deadline* da mensagem.

Na solução ótima, o atraso da mensagem mais atrasada é o menor para todo o espaço de busca. Na solução factível, uma solução é encontrada onde o escalonamento gerado

garante todas as características temporais, relações de precedência, e relacionamentos entre mensagens e blocos funcionais, de tal forma que seja uma solução aceitável no ponto de vista de atuação do sistema controlado. Mas contudo, esta solução pode não ser a solução mais otimizada no que diz respeito ao atraso na árvore de busca [12].

O algoritmo parte de uma solução inicial. Esta solução inicial é encontrada pelo escalonamento das mensagens utilizando-se o primeiro conjunto de características temporais e relações de precedência definidos pela base de dados da aplicação especificadas pela aplicação do usuário. Se esta primeira solução inicial não satisfazer às restrições temporais, resultando em uma ou mais mensagens atrasadas em relação a seu *deadline*, novas soluções serão geradas baseadas em novas restrições temporais e relações de precedência mais restritas que as anteriores.

Novas soluções são obtidas através da abertura de novos nós na árvore de busca baseados em uma análise de otimização do último nó aberto. Um novo nó da árvore de busca poderá ser aberto, para cada mensagem presente no último nó aberto, que tenha um *deadline* maior que o *deadline* da mensagem atrasada, e que não possua nenhuma relação de precedência com a mensagem mais atrasada do nó aberto. Assim, a mensagem ao ser escalonada após a mensagem atrasada, ela terá maior chance de atender ao seu *deadline* do que esta última, i.e., a mensagem escolhida será escalonada após a mensagem mais atrasada, na tentativa de antecipar o escalonamento da mensagem mais atrasada.

Um custo é calculado em função do atraso da mensagem e seu *deadline*. Este custo é chamado *lower-bound* e é utilizado como referência para a escolha entre as opções de **nós sucessores (nós filhos)** a serem abertos, que podem melhorar ou resolver o problema de escalonamento.

Como citado anteriormente, a técnica *branch and bound* foi utilizada neste trabalho para minimizar a busca [10]. Além disto, o algoritmo utiliza o *least-lower-bound-next*, que escolhe o ramo (*branch*) que deve ser seguido entre as opções de nós sucessores na árvore de busca. Baseado no custo calculado, a poda (*bound*) é usada para eliminar os **nós sucessores** que não irão gerar uma solução melhor do que a melhor solução encontrada até o instante de tempo pesquisado, decorrido pelo processo de busca por um escalonamento factível. Esta solução consegue atender todas as características temporais, relações de precedência e relacionamentos existentes. Somente o nó filho (nó sucessor) que tiver o menor *least-lower-bound* entre o *least-lower-bound* do nó pai (**nó aberto**) e os *least-lower-bounds* dos nós filhos será selecionado. Caso o menor *least-lower-bound* do nó filho seja igual ao do nó pai, então a busca por uma melhor solução é terminada.

O algoritmo pára quando uma solução factível for encontrada ou quando não existir mais nenhuma mensagem que possa ser trocada (*deadline* maior) com a mensagem mais atrasada do nó aberto (nó pai), i.e., nenhum outro ramo a ser percorrido na árvore de busca.

4. ETAPAS DO ALGORITMO

Nas seções seguintes serão discutidas as funções mais importantes do algoritmo. Serão apresentados também, os módulos necessários para que os dados definidos pela aplicação do usuário possam ser interpretados pelo algoritmo, e como serão tratados os escalonamentos gerados pelo algoritmo.

4.1. PERCURSOS ALTERNATIVOS EM CADA ESTRATÉGIA DE CONTROLE

Quando o usuário desenha sua estratégia de controle, ela pode conter mais de um percurso para transmissão de mensagens entre **blocos funcionais produtores** e **consumidores**.

Para esta dissertação consideram-se blocos funcionais produtores, blocos ou módulos de programa que não precisam receber uma mensagem nem exijam a execução de um bloco funcional antes de sua execução. Estes blocos são blocos geradores, como por exemplo, os blocos transdutores, que tem a função de fazer a leitura de um sensor instalado sobre a planta ou processo de controle industrial e o enviarem para outro bloco funcional [7].

Analogamente, blocos funcionais consumidores são blocos ou módulos de programas que somente tenham a função de consumir uma mensagem, ou exijam que um outro bloco funcional seja executado antes de sua execução. Estes blocos não geram qualquer resposta ou mensagem para outro bloco funcional, e não possuem no percurso nenhuma mensagem de saída, ou outro bloco funcional sendo executado após sua execução. O blocos atuadores são considerados blocos funcionais consumidores, os quais tem somente a função de agir ou atuar sobre a planta ou processo de controle industrial [7].

Um **percurso da estratégia de controle da aplicação** (a partir deste ponto mencionado somente como percurso) é uma lista sequencial que contém uma sequência de execução de blocos funcionais e de transmissão de mensagens. Esta sequência é definida a partir do desenho da estratégia de controle da aplicação do usuário seguindo-se os blocos funcionais através de suas conexões lógicas expressas por setas que indicam o sentido do

percurso escolhido. Assim, forma-se um conjunto de blocos funcionais e mensagens associadas, como pode ser observado pela figura 19.

Cada percurso exige diferentes características temporais das mensagens, relações de precedência e relacionamentos entre blocos funcionais e mensagens. Conforme citado no item 2.1 do capítulo 3, características temporais das mensagens, relações de precedência e relacionamentos entre blocos funcionais e mensagens definem as restrições importantes a serem atendidas para a comunicação no *fieldbus*. Estas restrições devem satisfazer todas as opções de percurso presentes nas estratégias de controle da aplicação do usuário.

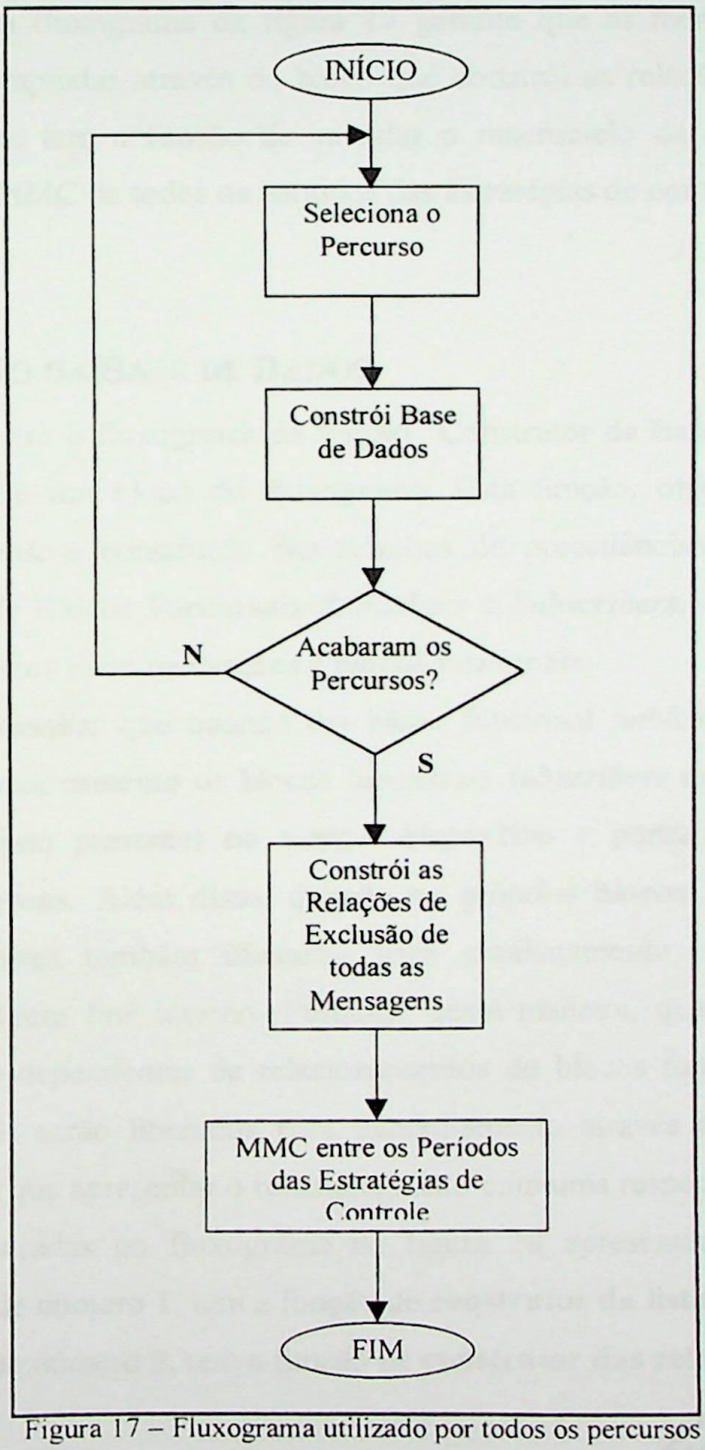


Figura 17 – Fluxograma utilizado por todos os percursos

Portanto, uma aplicação pode conter vários percursos entre os blocos funcionais produtores e os blocos funcionais consumidores, de tal forma que as características temporais sejam alteradas em função das várias opções de sequencialização e sincronização entre mensagens e blocos funcionais existentes na estratégia de controle definida pelo usuário.

O fluxograma da figura 17 apresenta a lógica utilizada para a construção de uma base de dados completa, onde todas as situações presentes na aplicação do usuário são consideradas.

Além da construção da base de dados em função de todos os percursos envolvidos na aplicação do usuário, o fluxograma da figura 17 garante que as mensagens existentes no sistema não serão preemptadas através do bloco que constrói as relações de exclusão entre mensagens. Além disto, tem a função de calcular o macrociclo da aplicação do usuário baseado no cálculo do MMC de todos os períodos das estratégias de controle.

4.2. CONSTRUÇÃO DA BASE DE DADOS

A figura 18 mostra o fluxograma da função "Construtor da Base de Dados" ilustrada na figura 17 através de um bloco do fluxograma. Esta função, objetiva a especificação temporal das mensagens, a construção das relações de precedências entre mensagens, a construção das listas de Blocos Funcionais *Publishers* e *Subscribers*, e a especificação dos relacionamentos existentes entre mensagens e blocos funcionais.

É importante ressaltar que quando um bloco funcional *publisher* é escalonado, ele também libera para escalonamento os blocos funcionais *subscribers* que possuírem um *link* interno, i.e., que estejam presentes no mesmo dispositivo e portanto não dependem da transmissão de mensagens. Além disto, quando os próprios blocos funcionais *publishers* forem escalonados, estes também liberarão para escalonamento os blocos funcionais *subscribers* que possuírem *link* interno. Portanto, desta maneira, quando existirem blocos funcionais *subscribers* dependentes de relacionamentos de blocos funcionais e mensagens, estes blocos funcionais serão liberados para escalonamento através do relacionamento do último bloco funcional que apresentar o relacionamento com uma respectiva mensagem.

As elipses ilustradas no fluxograma da figura 18 apresentam suas funções mais importantes. A elipse de número 1, tem a função de **construtor da lista de blocos funcionais *subscribers***. A elipse de número 2, tem a função de **construtor das relações de precedências entre mensagens**.

Escola Federal de Engenharia de Itajubá
Biblioteca
MAUÁ
BIM

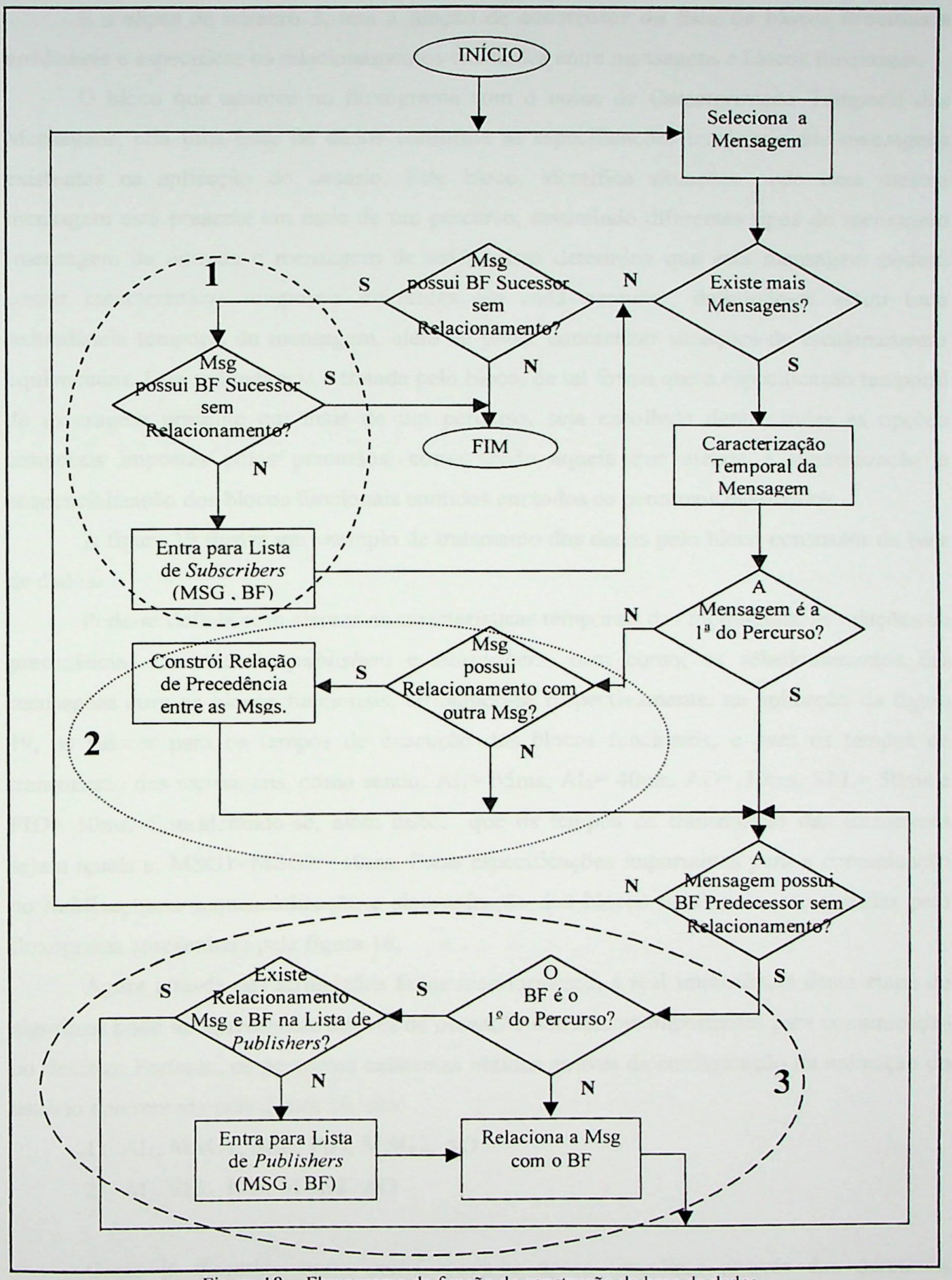


Figura 18 – Fluxograma da função de construção da base de dados

E a elipse de número 3, tem a função de **construtor da lista de blocos funcionais *publishers*** e especificar os relacionamentos existentes entre mensagens e blocos funcionais.

O bloco que aparece no fluxograma com o nome de Caracterização Temporal das Mensagens, cria uma base de dados conforme as especificações temporais das mensagens existentes na aplicação do usuário. Este bloco, identifica situações onde uma mesma mensagem está presente em mais de um percurso, assumindo diferentes tipos de mensagem (mensagem de entrada e mensagem de saída). Isto determina que esta mensagem poderá conter características temporais diferentes em cada percurso, definindo-se assim uma redundância temporal da mensagem, além de poder concretizar situações de escalonamento equivocadas. Esta redundância é tratada pelo bloco, de tal forma que a especificação temporal da mensagem presente em mais de um percurso, seja escolhida dentre todas as opções temporais impostas pelos percursos, como sendo aquela que atende à sincronização e sequencialização dos blocos funcionais contidos em todos os percursos envolvidos.

A figura 19 ilustra um exemplo de tratamento dos dados pelo bloco construtor da base de dados.

Pode-se definir com clareza as características temporais das mensagens, as relações de precedência, as listas de *publishers* e *subscribers*, bem como, os relacionamentos das mensagens com os blocos funcionais, atribuindo-se respectivamente, na aplicação da figura 19, os valores para os tempos de execução dos blocos funcionais, e para os tempos de transmissão das mensagens, como sendo: AI_1= 35ms, AI_2= 40ms, AO= 30ms, SEL= 50ms e PID= 60ms. Considerando-se, além disto, que os tempos de transmissão das mensagens sejam iguais a: MSG1=MSG2= 10ms. Estas especificações importantes para a comunicação no *fieldbus*, para sequencialização e sincronização dos blocos funcionais, são geradas pelo fluxograma apresentado pela figura 18.

Agora através das atribuições feitas anteriormente, a real importância desta etapa do algoritmo pode ser apresentada através da obtenção dos fatores importantes para comunicação no *fieldbus*. Portanto, os percursos existentes obtidos através da configuração da aplicação do usuário apresentada pela figura 19, são:

1) AI_1, **MSG1**, SEL, PID, **MSG2**, AO
2) AI_2, SEL, PID, **MSG2**, AO

Como já discutido nesta seção, pode-se agora visualizar através dos percursos existentes na aplicação, que a MSG2 apresentará redundância em sua especificação temporal, pois ela estará presente em dois percursos. Portanto, um tratamento especial deve ser dado a

ela. A seguir serão apresentados os cálculos para a especificação temporal das mensagens.

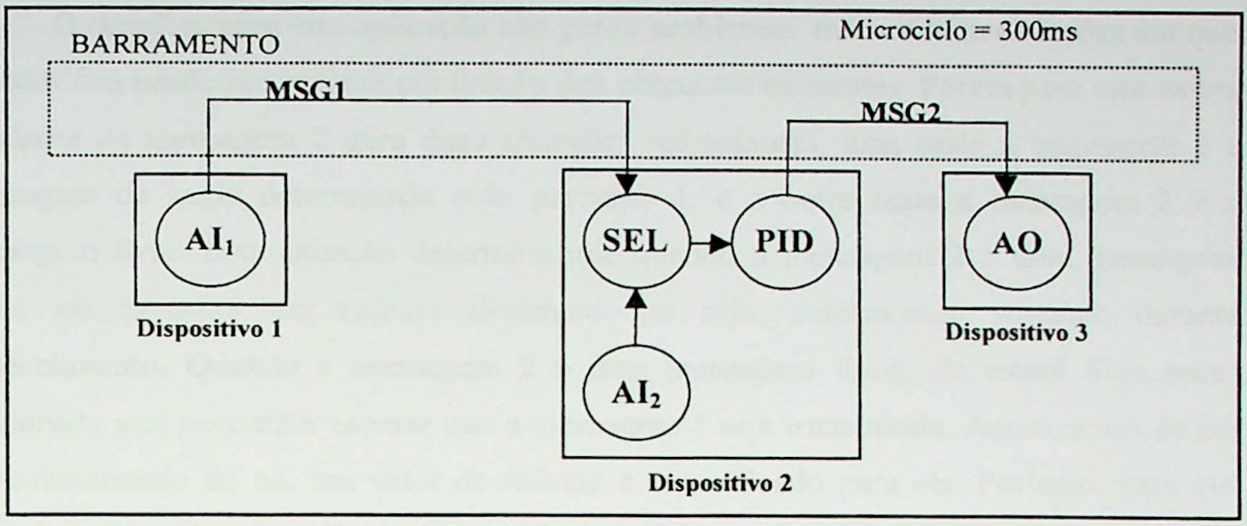


Figura 19 – Exemplo de aplicação do usuário

Os *releases* das mensagens de entrada são definidos como sendo zero, pois para que os blocos funcionais produtores (*publishers*) possam ser escalonados, estes devem ser liberados através da seleção das mensagens para escalonamento. Portanto, para que um bloco funcional seja escalonado, primeiro uma mensagem deve ser eleita para ser escalonada. Assim, os *deadlines* são obtidos através do cálculo:

d_{MSG1}= microciclo – $t_{execução}$ AO – $t_{execução}$ do PID – $t_{execução}$ SEL – $t_{transmissão}$ MSG2

d_{MSG1}= 300ms – 30ms – 60ms – 50ms – 10ms

d_{MSG1}= 150ms

Portanto, para que a mensagem 1 atenda suas características temporais, ela deve ter sua transmissão terminada exatamente no instante de tempo máximo de 150m segundos.

Para o percurso 2 que contém somente a mensagem 2 para ser transmitida, obtém-se como *deadline*:

d_{MSG2}= microciclo – $t_{execução}$ AO

d_{MSG2}= 300ms – 30ms

d_{MSG2}= 270ms

Então, o *release* é obtido através do percurso 2:

r_{MSG2}= microciclo – $t_{execução}$ AI_2 – $t_{execução}$ SEL – $t_{execução}$ do PID

r_{MSG2}= 300ms – 30ms – 50ms – 60ms

r_{MSG2}= **160ms**

O *deadline* para esta aplicação não gerou problemas, mas existem situações em que ele também fica sendo redundante em função dos percursos existentes. Porém para este exemplo, o *release* da mensagem 2 gera duas situações redundantes, uma onde a mensagem é uma mensagem de saída determinada pelo percurso 1, e a outra onde a mensagem 2 é uma mensagem livre. Esta situação determina que quando a mensagem 2 é uma mensagem de saída, ela possuirá um *release* dinâmico, ou seja, determinado somente durante o escalonamento. Quando a mensagem 2 é uma mensagem livre, ela estará livre para ser escalonada sem necessitar esperar que a mensagem 1 seja transmitida. Assim, antes de iniciar o escalonamento do nó, um valor de *release* é especificado para ela. Portanto, para que as duas situações sejam atendidas, deve-se fazer com que a mensagem 2 seja uma mensagem de saída de tal forma, que seu *release* e *deadline* dinâmicos, relacionados com as características temporais da mensagem especificadas pelo percurso 1, respeitem a janela de tempo definida pelo *release* e *deadline* do percurso 2. Desta forma, no exemplo apresentado não serão geradas inconsistências nas especificações temporais, em função da consideração de que a mensagem 2 é uma mensagem válida.

Além disto, ainda faltam especificações de algumas restrições importantes para a comunicação no *fieldbus*. Portanto, para o exemplo da figura 19, tem-se:

- a relação de precedência onde **MSG1** *precede* **MSG2**.
- os relacionamentos entre mensagens e blocos funcionais:

MSG1 $\rightarrow$ AI_1

MSG2 $\rightarrow$ AI_2, SEL, PID

A sequência encadeada de blocos funcionais deve ser respeitada para que o sequenciamento da aplicação seja respeitada.

Além dos respectivos relacionamentos, existem ainda relacionamentos complementares baseados nas listas de *publishers* e *subscribers* necessários para a garantia da sincronização e sequenciamento dos blocos funcionais e mensagens:

publishers	*subscribers*
MSG1 $\rightarrow$ AI_1	**MSG2** $\rightarrow$ AO
MSG2 $\rightarrow$ AI_2	

Concluindo, para que os escalonamentos das mensagens e blocos funcionais estejam adequados e atendam a todas as restrições temporais existentes na comunicação do *fieldbus*, é importante que as informações referentes as restrições temporais estejam consistentes e em conformidade com as características existentes nas estratégias de controle da aplicação do usuário.

4.3. ALGORITMO ESCALONADOR

A seguir serão apresentadas através de fluxogramas, as funções que incorporam o algoritmo escalonador.

4.3.1. OBTENÇÃO DA ÁRVORE DE BUSCA

A árvore de busca nada mais é do que um grafo, aonde os nós são soluções ou escalonamentos gerados pela função de **escalonamento do nó** contida no algoritmo escalonador. Esta hierarquia surge partindo-se de uma solução inicial, i.e., um nó raiz. Caso este nó raiz não atenda a todas as restrições temporais impostas pelo *fieldbus*, novos nós vão surgindo, buscando-se uma solução ou nó que as garanta. Para isto, quando o nó raiz não atende às restrições temporais, elas serão alteradas, diminuindo o grau de liberdade da solução, criando novas relações de precedência entre mensagens para que um novo nó seja gerado. Este novo nó, que se baseia no nó raiz que não atendeu às restrições (referenciado, agora, como nó-pai), procura encontrar uma mensagem que tenha sido escalonada antes da mensagem mais atrasada e que possa ser escalonada depois dela.

Isto é feito para alocar um espaço para o escalonamento da mensagem mais atrasada do nó posicionado anteriormente à posição alocada no nó-pai.

Com a geração deste novo nó, visa-se a obtenção de uma resposta que garanta as restrições, e que seja melhor no que diz respeito ao atraso da mensagem mais atrasada, do que a solução encontrada até seu nível hierárquico superior na árvore de busca. A cada relação de precedência entre mensagens incorporada ao conjunto de relações de precedência especificadas pela aplicação do usuário, um novo nível descendente na hierarquia da árvore é criado. Portanto, para que um novo nível na hierarquia da árvore de busca seja criado, será selecionado somente o nó-filho que tiver um limite mínimo do custo menor que o do nó-pai e o menor limite mínimo do custo entre os dos nós-filhos. A partir deste ponto este nó filho posiciona-se como o nó-pai e possibilitará obter outras soluções através dos nós filhos que

serão gerados através dele. Este processo é executado ciclicamente até que uma solução seja encontrada ou até quando as relações de precedências se esgotarem, o que fatalmente incorrerá na não obtenção de uma solução factível.

O comportamento hierárquico e o escalonamento dos nós da árvore de busca podem ser observados através da figura 4.

A busca por uma solução factível, pode ser uma explosão combinatória. Para minimizá-la, utilizou-se no algoritmo a técnica "*branch-and-bound*"(ramificar e podar), cuja escolha para ramificação é feita pelo custo, i.e., com o menor limite mínimo (*least-lower-bound-next*). O custo usado no problema conforme explicado no item 3, é calculado em função do atraso na transmissão da mensagem em relação a seu *deadline*.

Para que fossem evitados ciclos e as mensagens pudessem ser sequencializadas no barramento corretamente, foi necessário o desenvolvimento de uma função no algoritmo que verificasse a consistência temporal de todas as mensagens transmitidas pelo barramento. Este algoritmo consegue varrer a lista de precedências das mensagens em vários níveis, eliminando inconsistências durante o escalonamento de cada nó da árvore.

A poda ("*bound*") é feita para eliminar os nós sucessores que não gerarão uma solução melhor que a melhor encontrada na busca até então, através de uma função que determina durante a ramificação da árvore de busca, quais nós da lista de nós abertos serão eliminados, ou seja, serão excluídos os nós que são irmãos do novo nó eleito como nó pai conforme explicado no item 3. Com este procedimento, o espaço de busca na árvore é minimizado.

A figura 20 ilustra este procedimento, indicando com um "⊗" os nós que não foram abertos e com um "O" os irmãos eliminados do caminho de busca por uma melhor solução.

Para o exemplo da figura 20, foram gerados quatro nós no caminho da resposta factível, sendo que, dois nós (2 e 6) foram excluídos dentre as opções que compõem o espaço de soluções para a obtenção de um escalonamento factível na árvore de busca. Neste exemplo, como pode ser observado através da atividade de exclusão dos nós "filhos", os nós 2 e 6 não foram eleitos como nós pais deixando assim de serem ramificados, indicados pelo "⊗" na figura 20, pois, estes não iriam melhorar a resposta obtida no nível corrente.

Como observado pela figura 20, os níveis da árvore de busca são determinados pelas relações de precedências incorporadas às relações de precedências existentes definidas pela aplicação do usuário.

Na figura 20, as relações de precedências existentes definidas pela aplicação do usuário são identificadas através das relações apresentadas no primeiro nó aberto da árvore de busca, definido como a raiz da árvore de busca. Portanto, as relações da raiz ou nó 1 são:

MSG1 *precede* **MSG5**, **MSG2** *precede* **MSG6**, **MSG3** *precede* **MSG7** e **MSG4** *precede* **MSG8**.

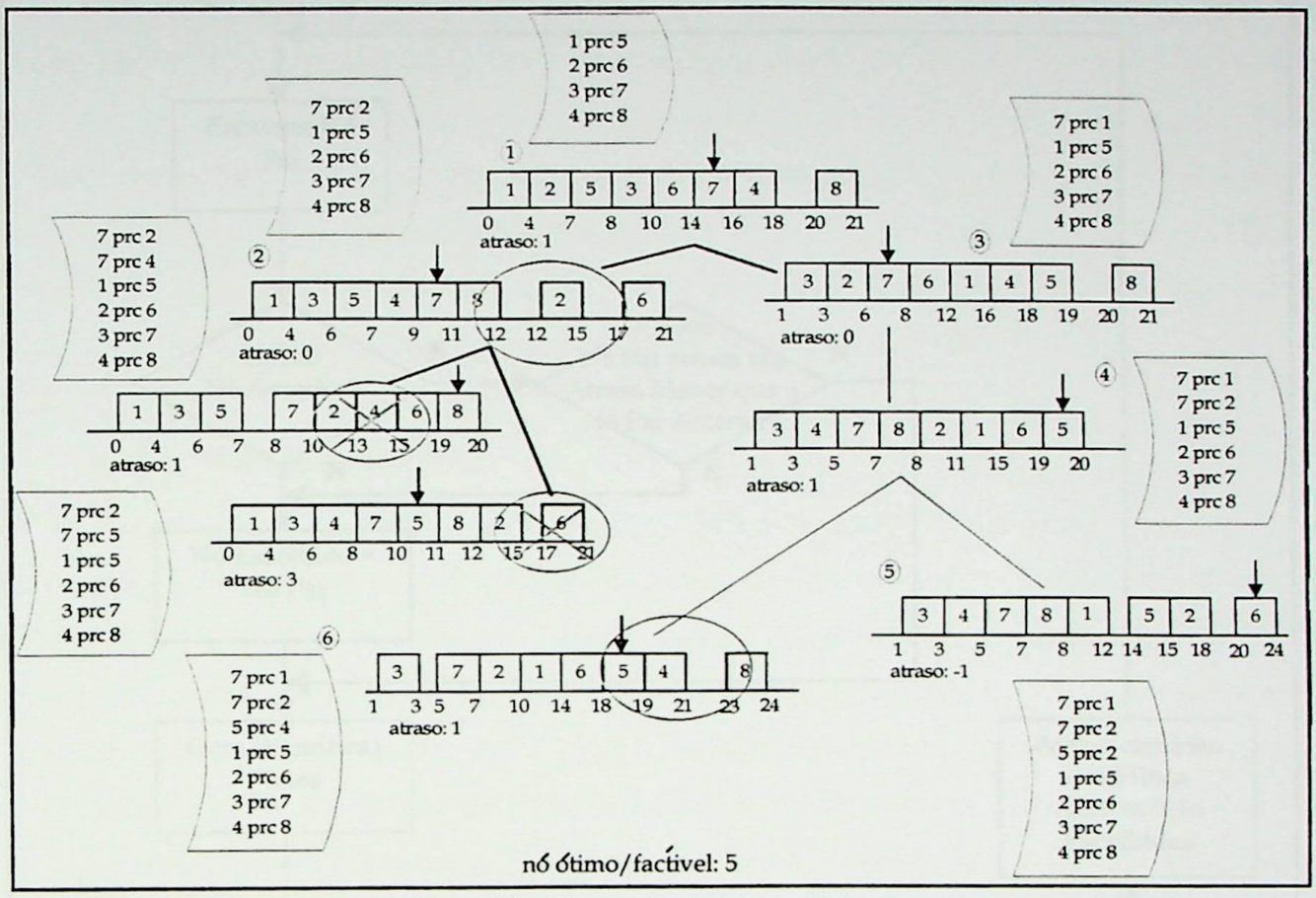


Figura 20 – Hierarquia da árvore de busca

Baseado no nó 5, que foi a solução encontrada e especificada como solução ótima ou factível, pode-se observar que foram adicionados as seguintes relações de precedência: **MSG7** *precede* **MSG1**, **MSG7** *precede* **MSG2** e **MSG5** *precede* **MSG2**. Estas relações indicam o nível da hierarquia da árvore de busca que o nó pertence, ou seja, o nível 3. O nível da raiz é mantido como referência, portanto este não é contado, i.e., considera-se que ele pertence ao nível 0, pois, não possui nenhuma relação de precedência extra gerada automaticamente pelo algoritmo escalonador.

Como pode ser observado pelo fluxograma da figura 21, os nós são escalonados até o momento em que não sejam encontrados mais nós filhos, i.e., outras opções para a otimização do escalonamento tido como uma solução ainda não factível. Neste caso, pode-se melhorar ou otimizar esta solução intermediária, até a obtenção de um escalonamento que atenda a todas as restrições temporais da comunicação no *fieldbus*, ou seja, uma solução factível.

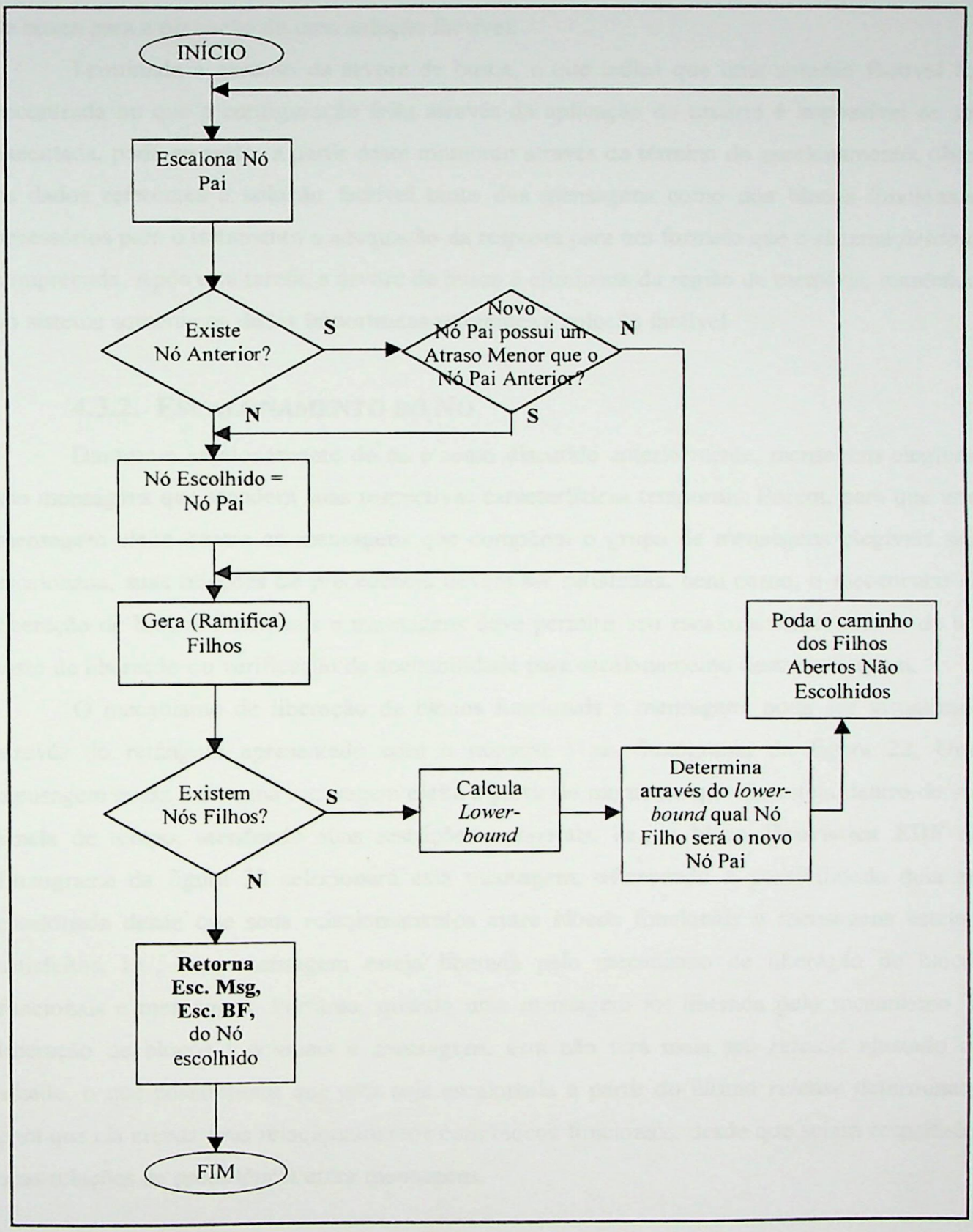


Figura 21 – Fluxograma utilizado para geração da árvore de busca

Para a determinação do caminho a ser percorrido na árvore de busca, o filho que irá melhorar ou otimizar a solução intermediária que contém atraso, é selecionado através do

cálculo do *lowerbound*, o qual faz uma estimativa do caminho a ser seguido (ramos) na árvore de busca para a obtenção de uma solução factível.

Terminada a geração da árvore de busca, o que indica que uma solução factível foi encontrada ou que a configuração feita através da aplicação do usuário é impossível de ser executada, pode-se então, a partir deste momento através do término do escalonamento, obter os dados referentes a solução factível tanto das mensagens como dos blocos funcionais, necessários para o tratamento e adequação da resposta para um formato que o sistema *fieldbus* compreenda. Após esta tarefa, a árvore de busca é eliminada da região de memória, mantendo no sistema somente os dados importantes referentes a solução factível.

4.3.2. ESCALONAMENTO DO NÓ

Durante o escalonamento do nó e como discutido anteriormente, mensagens elegíveis são mensagens que atendem suas respectivas características temporais. Porém, para que uma mensagem eleita dentre as mensagens que compõem o grupo de mensagens elegíveis seja escalonada, suas relações de precedência devem ser satisfeitas, bem como, o mecanismo de liberação de blocos funcionais e mensagens deve permitir seu escalonamento através de um visto de liberação ou verificação de aceitabilidade para escalonamento desta mensagem.

O mecanismo de liberação de blocos funcionais e mensagens pode ser visualizado através do retângulo apresentado com o número 1 no fluxograma da figura 22. Uma mensagem passa a ser uma mensagem eleita a partir do momento que ela esteja dentro de sua janela de tempo, atendendo suas restrições temporais, i.e., o bloco **Heurística EDF** do fluxograma da figura 22 selecionará esta mensagem, oferecendo a possibilidade dela ser escalonada desde que seus relacionamentos entre blocos funcionais e mensagens estejam satisfeitos, i.e., se a mensagem esteja liberada pelo mecanismo de liberação de blocos funcionais e mensagens. Portanto, quando uma mensagem foi liberada pelo mecanismo de liberação de blocos funcionais e mensagens, esta não terá mais seu *release* ajustado ou adiado, o que possibilitará que esta seja escalonada a partir do último *release* determinado para que ela atenda seus relacionamentos com blocos funcionais, desde que sejam respeitadas suas relações de precedência entre mensagens.

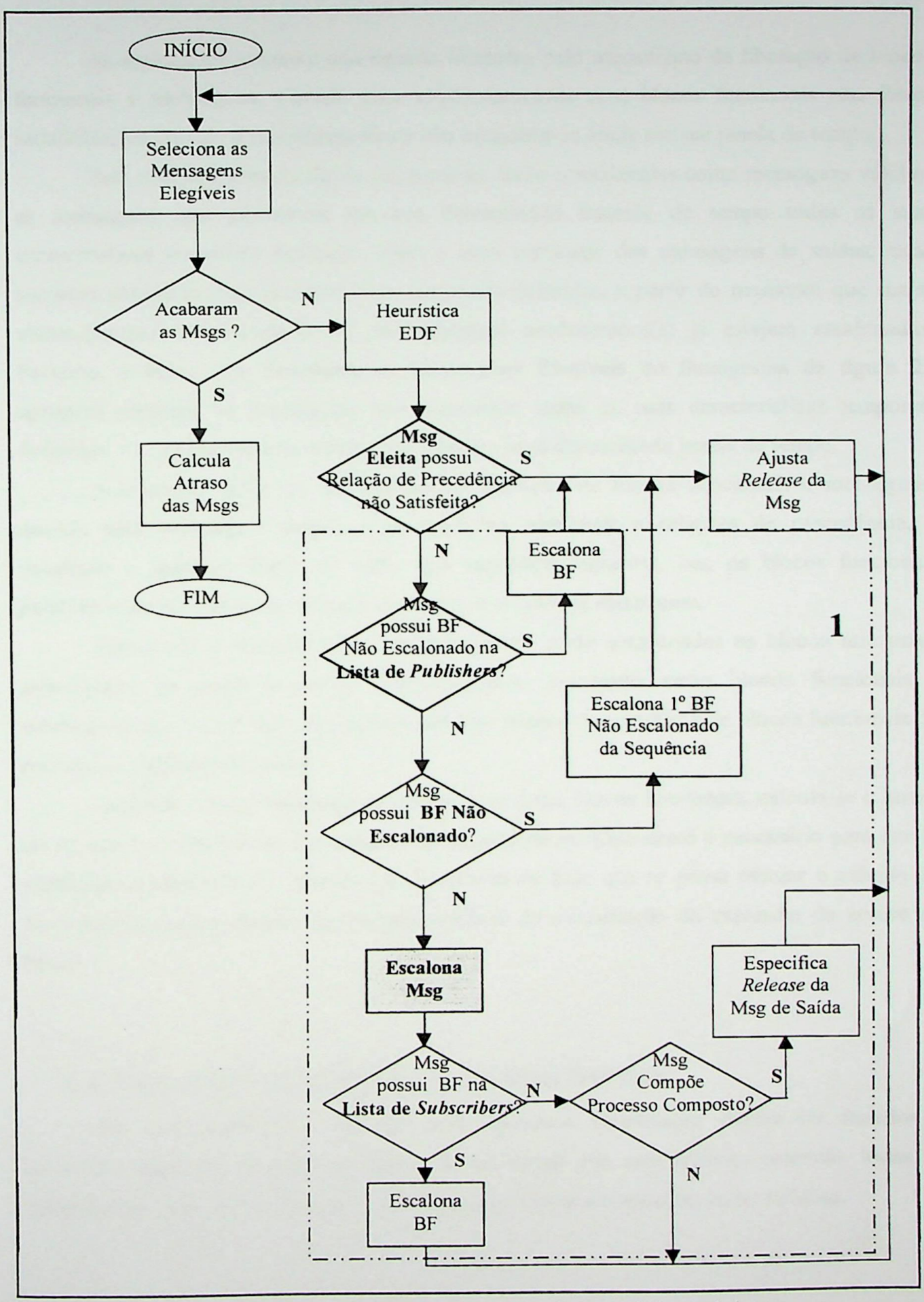


Figura 22 – Fluxograma para o escalonamento de um nó da árvore de busca

As mensagens somente não estarão liberadas pelo mecanismo de liberação de blocos funcionais e mensagens, quando seus relacionamentos com blocos funcionais não forem satisfeitos, ou quando a mensagem ainda não encontrar-se ainda em sua janela de tempo.

Para o fluxograma da figura 22, somente serão consideradas como mensagens válidas, as mensagens que possuírem em um determinado instante de tempo todas as suas características temporais definidas. Para o caso particular das mensagens de saídas, estas somente possuirão suas características temporais definidas, a partir do momento que sua(s) mensagem(ns) de entrada(s) ou mensagem(ns) predecessora(s) já estejam escalonadas. Portanto, o bloco que **Seleciona as Mensagens Elegíveis** do fluxograma da figura 22 agrupará somente as mensagens que possuírem todas as suas características temporais definidas, i.e., as mensagens válidas que atendam uma determinada janela de tempo.

A primeira tarefa do mecanismo de liberação de blocos funcionais e mensagens, quando uma mensagem atende suas restrições temporais e relações de precedência, é escalonar o primeiro bloco de cada uma sequência existente, i.e., os blocos funcionais *publishers* pertencentes ao percurso que envolve a referida mensagem.

Terminado o escalonamento das mensagens, serão escalonados os blocos funcionais *subscribers*, os quais possuem relacionamentos sucessores entre blocos funcionais e mensagens. Isto indica que estes blocos serão os últimos da sequência de blocos funcionais do percurso a serem escalonados.

Acabado o escalonamento das mensagens e dos blocos funcionais, calcula-se o atraso do nó que é o maior atraso de todas as mensagens do nó. Este atraso é necessário para que se verifique se uma solução factível foi encontrada ou para que se possa efetuar o cálculo do *lowerbound*, para a verificação da possibilidade de continuação da expansão da árvore de busca.

4.4. TRATAMENTO DOS ESCALONAMENTOS OBTIDOS

Os escalonamentos retornados pelo algoritmo escalonador devem ser tratados e traduzidos para um formato de tabela, de tal forma que este formato contenha todas as informações necessárias para que o LAS possa gerenciar a comunicação no *fieldbus*.

4.4.1. ESCALONAMENTO DOS BLOCOS FUNCIONAIS

O escalonamento dos blocos funcionais gerado pelo algoritmo escalonador deve apresentar as seguintes informações para a construção da tabela:

1) Parâmetro referente ao instante de tempo inicial que determinado bloco funcional tem que começar a ser executado pelo processador de determinado dispositivo, condizente com a mensagem que deve ser gerada para ser transmitida através do barramento.

2) Parâmetro referente ao número do bloco funcional que deve ser executado no instante de tempo definido pelo item 1. O número do bloco funcional é definido em função do tipo do bloco funcional e do número do dispositivo aonde será executado este bloco funcional.

4.4.2. ESCALONAMENTO DAS MENSAGENS

Da mesma forma, o escalonamento das mensagens gerado pelo algoritmo escalonador deve apresentar as seguintes informações ou parâmetros:

1) O instante de tempo inicial, o qual determinará o início da transmissão da mensagem sobre o barramento.

2) O número da mensagem que será transmitida sobre o barramento no instante de tempo definido pelo item anterior.

A partir do momento que o LAS possuir esta tabela, ele poderá despachar transações permitindo que os dispositivos possam iniciar sua transmissão, um por vez através do barramento.

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca MAUÁ BIM

CAPÍTULO 6 – RESULTADOS

Este capítulo apresenta algumas aplicações do usuário e as soluções obtidas através do algoritmo proposto por este trabalho.

Para visualização dos resultados, foi desenvolvida uma interface gráfica mostrando o escalonamento das mensagens e dos blocos funcionais.

1. PROGRAMA GERADO PARA APRESENTAÇÃO DOS RESULTADOS

Neste trabalho foi desenvolvido um programa contendo uma interface gráfica que possibilita visualizar as linhas de tempo relacionadas com os escalonamentos gerado pelo algoritmo proposto.

Esta interface apresenta as linhas de tempo existentes no sistema, as quais devem respeitar o macrociclo calculado através do MMC das estratégias de controle que compõem a aplicação do usuário. Neste macrociclo, as mensagens e blocos funcionais devem ser escalonadas de tal forma que sejam atendidas as restrições temporais da aplicação e suas respectivas periodicidades.

Na parte superior da figura 23 pode-se observar a primeira janela de apresentação do programa, e na parte inferior a interface gráfica que permite a visualização das linhas de tempo para o escalonamento dos blocos funcionais e mensagens.

Para que o algoritmo proposto ficasse independente de qualquer programa que necessite utilizá-lo, este foi incorporado a uma biblioteca (esc.lib) que permite que qualquer utilitário ou programa do *Windows* possa acessá-lo e utilizá-lo.

Como a fase de configuração da aplicação do usuário faz parte de um programa configurador do *fieldbus*, e para que o algoritmo proposto fosse independente do tipo de configurador utilizado, desenvolveu-se um formato de comunicação para a troca de informações entre o algoritmo proposto e o configurador *fieldbus*. Este formato foi desenvolvido em cooperação com o pessoal técnico de desenvolvimento da empresa SMAR, a qual possui o programa configurador SYSCON que possibilita a configuração de plantas *fieldbus* da SMAR.

Deste modo os dados referentes à aplicação do usuário, são passados neste formato através de um arquivo ou de uma estrutura de dados residente em memória.

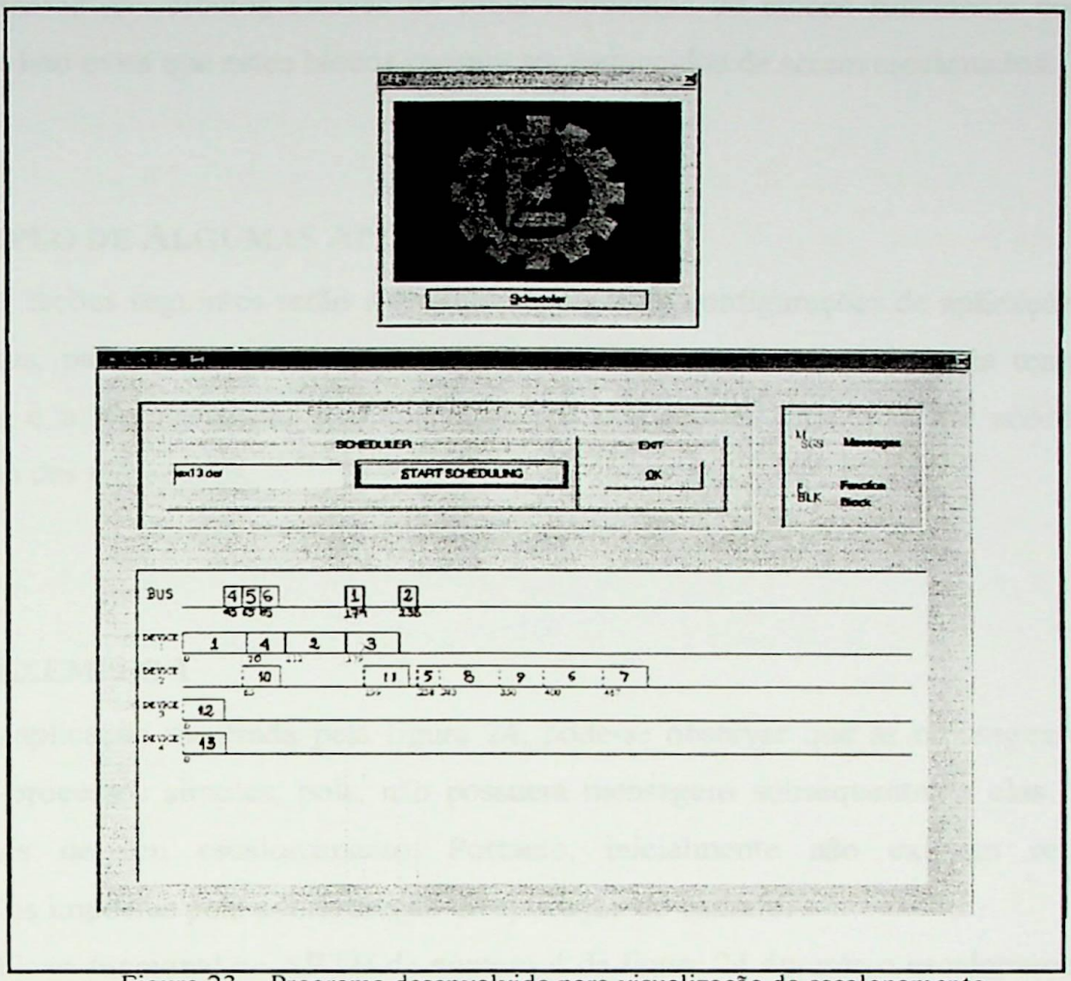


Figura 23 – Programa desenvolvido para visualização do escalonamento

Como pode ser observado pela janela localizada na parte inferior da figura 23, um nome de arquivo deve ser digitado manualmente pelo usuário para que desta forma, o botão da janela "START SCHEDULING" seja habilitado. Habilitado o botão, o algoritmo proposto poderá ser executado a partir de um único clique sobre o botão. Terminada a execução do algoritmo e encontrado um escalonamento factível, a interface gráfica situada na região inferior da janela irá ilustrar o resultado obtido através de uma simulação para o estado *on-line* do sistema. Este estado será delimitado pelo macrociclo que irá conter as estratégias de controle a serem executadas.

Cada dispositivo *fieldbus* conectado ao barramento e fazendo parte da aplicação, possui uma linha de tempo associada com a base de tempo global que representa o tempo de ocupação do processador do dispositivo. Uma linha de tempo também é determinada para o barramento, caracterizando assim o acesso ao barramento pelo dispositivo através da transmissão das mensagens entre dispositivos. Estas linhas de tempo são mostradas na interface gráfica.

Na existência de blocos funcionais sem relações com mensagens, novos

relacionamentos são criados através de relacionamentos de blocos funcionais com blocos funcionais. Isto evita que estes blocos possam ser esquecidos de serem escalonados.

2. EXEMPLO DE ALGUMAS APLICAÇÕES SIMPLES

Nas seções seguintes serão apresentadas algumas configurações de aplicações básicas e complexas, para que em função delas possa ser observada a consistência temporal das mensagens e a sincronização e sequenciamento dos blocos funcionais de acordo com a transmissão das mensagens.

2.1. EXEMPLO 1

Na aplicação mostrada pela figura 24, pode-se observar que as mensagens 1, 2 e 3 compõem processos simples, pois, não possuem mensagens subsequentes a elas que sejam dependentes de seu escalonamento. Portanto, inicialmente não existem relações de precedências impostas pela configuração da aplicação do usuário.

O bloco funcional de ARTH de número 4 da figura 24 durante o escalonamento, deve então esperar que todas as mensagens sejam escalonadas, para que ele possa ser escalonado.

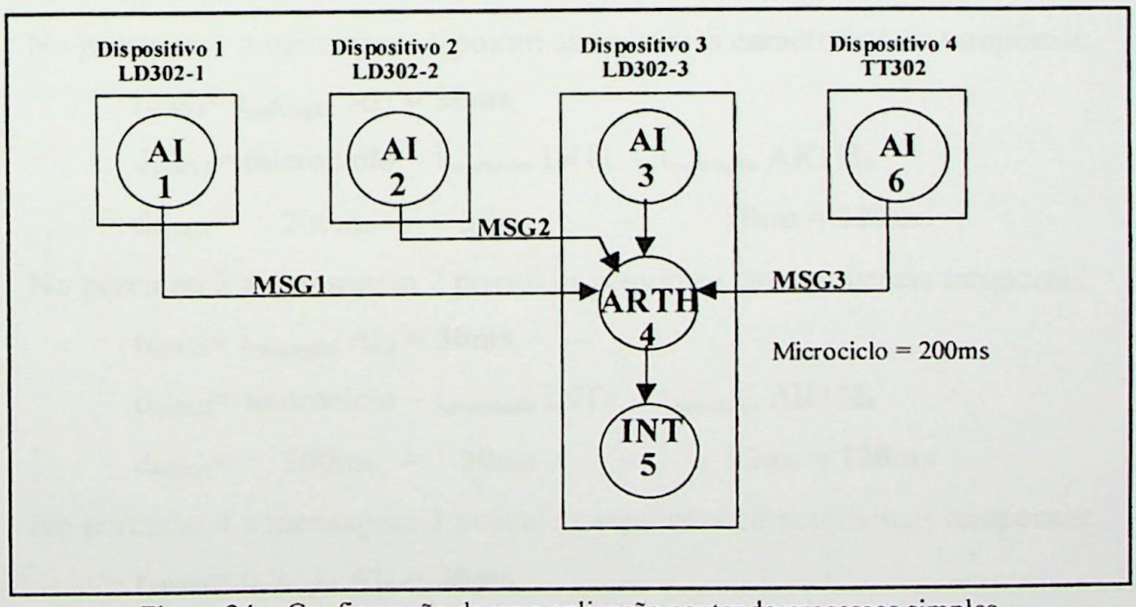


Figura 24 – Configuração de uma aplicação contendo processos simples

Para esta aplicação, o macrociclo adotado foi de 200ms.

A figura 25 mostra a simulação para o estado *on-line* do escalonamento das mensagens sobre o barramento (*BUS*) e os escalonamentos dos blocos funcionais para cada

dispositivo, obtidos através da execução do algoritmo proposto. A visualização dos escalonamentos foi capturada do programa gerado para representação dos resultados desenvolvido neste trabalho.

Como pode ser observado pela figura 25, os números que indicam as mensagens na configuração da figura 24, são escalonados sobre a linha de tempo denominada como *BUS*. No caso dos números dos blocos funcionais indicados na configuração da figura 24, estes são escalonados sobre as linhas de tempo dos dispositivos aos quais estes estão relacionados. Cada um dos dispositivos (DEVICE1, DEVICE2, DEVICE3, DEVICE4) da configuração da figura 24, possuem então uma linha de tempo na qual seus blocos funcionais são escalonados.

Na aplicação foram considerados os seguintes tempos de processamento dos blocos funcionais: $AI_1 = AI_2 = AI_3 = AI_6 = 30$ ms, ARTH = 50 ms e o INT = 30 ms. Foi estipulado como tempo de transmissão das mensagens: MSG1 = 10 ms, MSG2 = 30 ms e MSG3 = 20 ms.

E os percursos existentes obtidos através da configuração da aplicação do usuário apresentada pela figura 24 são:

1) AI_1, **MSG1**, $ARTH_4$, INT_5
2) AI_2, **MSG2**, $ARTH_4$, INT_5
3) AI_3, $ARTH_4$, INT_5
4) AI_6, **MSG3**, $ARTH_4$, INT_5

No percurso 1 a mensagem 1 possui as seguintes características temporais:

$r_{MSG1} = t_{execução}\ AI_1 = \mathbf{30ms}$

$d_{MSG1} = \text{microciclo} - t_{execução}\ INT_5 - t_{execução}\ ARTH_4$

$d_{MSG1} = 200ms - 30ms - 50ms = \mathbf{120ms}$

No percurso 2 a mensagem 2 possui as seguintes características temporais:

$r_{MSG2} = t_{execução}\ AI_2 = \mathbf{30ms}$

$d_{MSG2} = \text{microciclo} - t_{execução}\ INT_5 - t_{execução}\ ARTH_4$

$d_{MSG2} = 200ms - 30ms - 50ms = \mathbf{120ms}$

No percurso 4 a mensagem 3 possui as seguintes características temporais:

$r_{MSG3} = t_{execução}\ AI_6 = \mathbf{30ms}$

$d_{MSG3} = \text{microciclo} - t_{execução}\ INT_5 - t_{execução}\ ARTH_4$

$d_{MSG3} = 200ms - 30ms - 50ms = \mathbf{120ms}$

Portanto,as características temporais e os relacionamentos entre blocos funcionais e

mensagens utilizados, estão especificados através da tabela 2. Na mesma tabela também são definidas as listas de *publishers* e *subscribers* existentes na configuração da aplicação mostrada pela figura 24.

Tabela 2 – Características temporais, relacionamentos e listas da aplicação mostrada na figura 24

Mensagens	***Release, Deadline* (ms)**	**Relacionamentos entre MSGs e BFs**	**Lista de *Publishers***	**Lista de *Subscribers***
MSG1	0, 120	AI_1	AI_1	$ARTH_4$
MSG2	0, 120	AI_2	AI_2	$ARTH_4$
MSG3	0, 120	AI_6	AI_6	$ARTH_4$
Relacionamentos entre BFs			$AI_3 \rightarrow ARTH_4$	$ARTH_4 \rightarrow INT_5$

Observando-se a figura 25, as mensagens foram escalonadas conforme seu tempo de transmissão, i.e., como os *releases* e *deadlines* das mensagens são os mesmos, o EDF elegeu as mensagens para serem transmitidas na ordem decrescente do seu tempo de transmissão. Pretende-se com isto garantir de que começando pelas mensagens maiores seja mais fácil encaixar posteriormente as mensagens menores.

Neste exemplo, existiram dois relacionamentos entre blocos funcionais referentes aos *links* internos existentes no dispositivo 3, que não transmitem nenhuma mensagem para o barramento. Estes relacionamentos garantem que os blocos funcionais 3 e 5 serão escalonados de acordo com os escalonamentos dos outros blocos funcionais. Como pode ser observado através dos relacionamentos existentes, pode-se garantir a sincronização entre as mensagens e blocos funcionais, e a sincronização existente entre blocos funcionais relacionados.

O sequenciamento é obtido através dos percursos existentes, e através destes é que as relações de precedências entre mensagens são definidas.

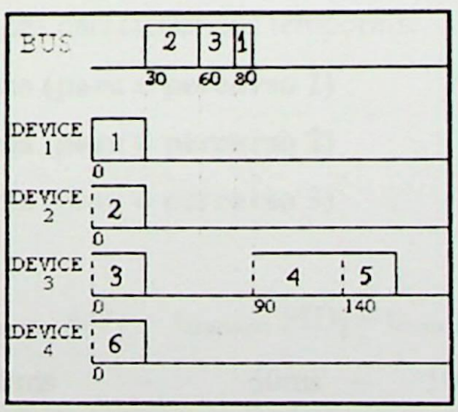


Figura 25 – Simulação do escalonamento factível da aplicação da figura 24

2.2. EXEMPLO 2

Na figura 26 será tratado um outro tipo de configuração de aplicação, onde existem dois processos compostos e uma quantidade maior de percursos a serem percorridos e identificados. Desta forma, novas situações foram inseridas para uma variedade mais ampla de ocorrências existentes na configuração da aplicação do usuário fosse apresentada.

Na configuração da aplicação da figura 26, foram considerados os seguintes tempos de processamento dos blocos funcionais: $AI_1 = AI_4 = AI_7 = 30$ ms, $PID_2 = PID_5 = PID_8 = 60$ ms, $OSDL_6 = 60$ ms e $AO_3 = AO_9 = 30$ ms.

Para o tempo de transmissão das mensagens, estipulou-se: MSG1 = 20 ms, MSG2 = 15 ms, MSG3 = 10 ms e MSG4 = 10 ms.

E os percursos existentes obtidos através da configuração da aplicação do usuário apresentada pela figura 26 são:

1) AI_1, **MSG1**, $OSDL_6$, **MSG3**, PID_2, AO_3
2) AI_1, **MSG1**, PID_2, AO_3
3) AI_1, **MSG1**, $OSDL_6$, **MSG4**, PID_8, AO_9
4) AI_7, **MSG2**, $OSDL_6$, **MSG4**, PID_8, AO_9
5) AI_7, **MSG2**, PID_8, AO_9
6) AI_7, **MSG2**, $OSDL_6$, **MSG3**, PID_2, AO_3
7) AI_4, PID_5, $OSDL_6$, **MSG3**, PID_2, AO_3
8) AI_4, PID_5, $OSDL_6$, **MSG4**, PID_8, AO_9

Como existem múltiplos percursos que definem características temporais redundantes, é interessante apresentar os cálculos em função das mensagens e não dos percursos. Desta forma, tem-se:

⇒Para a **mensagem 1** as seguintes características temporais:

$r_{MSG1} = t_{execução}\ AI_1$ = **30ms (para o percurso 1)**

$r_{MSG1} = t_{execução}\ AI_1$ = **30ms (para o percurso 2)**

$r_{MSG1} = t_{execução}\ AI_1$ = **30ms (para o percurso 3)**

d_{MSG1} = microciclo – $t_{execução}\ AO_3$ – $t_{execução}\ PID_2$ – $t_{execução}\ MSG_3$ – $t_{execução}\ OSDL_6$

d_{MSG1} = 300ms – 30ms – 60ms – 10ms – 60ms

d_{MSG1} = **140ms (para o percurso 1)**

d_{MSG1} = microciclo – $t_{execução}\ AO_3$ – $t_{execução}\ PID_2$

d_{MSG1}= 300ms – 30ms – 60ms

d_{MSG1}= **210ms (para o percurso 2)**

d_{MSG1}= microciclo – $t_{execução}$ AO_9 – $t_{execução}$ PID_8 – $t_{execução}$ MSG_4 – $t_{execução}$ $OSDL_6$

d_{MSG1}= 300ms – 30ms – 60ms – 10ms – 60ms

d_{MSG1}= **140ms (para o percurso 3)**

⇒Para a **mensagem 2** as seguintes características temporais:

r_{MSG2}= $t_{execução}$ AI_7 = **30ms (para o percurso 4)**

r_{MSG2}= $t_{execução}$ AI_7 = **30ms (para o percurso 5)**

r_{MSG2}= $t_{execução}$ AI_7 = **30ms (para o percurso 6)**

d_{MSG2}= microciclo – $t_{execução}$ AO_9 – $t_{execução}$ PID_8 – $t_{execução}$ MSG_4 – $t_{execução}$ $OSDL_6$

d_{MSG2}= 300ms – 30ms – 60ms – 10ms – 60ms

d_{MSG2}= **140ms (para o percurso 4)**

d_{MSG2}= microciclo – $t_{execução}$ AO_9 – $t_{execução}$ PID_8

d_{MSG2}= 300ms – 30ms – 60ms

d_{MSG2}= **210ms (para o percurso 5)**

d_{MSG2}= microciclo – $t_{execução}$ AO_3 – $t_{execução}$ PID_2 – $t_{execução}$ MSG_3 – $t_{execução}$ $OSDL_6$

d_{MSG2}= 300ms – 30ms – 60ms – 10ms – 60ms

d_{MSG2}= **140ms (para o percurso 6)**

⇒Para a **mensagem 3** as seguintes características temporais:

r_{MSG3}= dinâmico **(para o percurso 1)**

r_{MSG3}= dinâmico **(para o percurso 6)**

r_{MSG3}= $t_{execução}$ AI_4 + $t_{execução}$ PID_5 + $t_{execução}$ $OSDL_6$

r_{MSG3}= **30ms + 60ms + 60ms = 150ms (para o percurso 7)**

d_{MSG3}= microciclo – $t_{execução}$ AO_3 – $t_{execução}$ PID_2

d_{MSG3}= 300ms – 30ms – 60ms

d_{MSG3}= **210ms (para o percurso 1)**

d_{MSG3}= microciclo – $t_{execução}$ AO_3 – $t_{execução}$ PID_2

d_{MSG3}= 300ms – 30ms – 60ms

d_{MSG3}= **210ms (para o percurso 6)**

d_{MSG3}= microciclo – $t_{execução}$ AO_3 – $t_{execução}$ PID_2

d_{MSG3}= 300ms – 30ms – 60ms

d_{MSG3}= **210ms (para o percurso 7)**

⇒Para a **mensagem 4** as seguintes características temporais:

r_{MSG4}= dinâmico **(para o percurso 3)**

r_{MSG4}= dinâmico **(para o percurso 4)**

r_{MSG4}= $t_{execução}$ AI_4 + $t_{execução}$ PID_9 + $t_{execução}$ PID_8

r_{MSG4}= **30ms + 60ms + 60ms = 150ms (para o percurso 8)**

d_{MSG4}= microciclo – $t_{execução}$ AO_9 – $t_{execução}$ PID_8

d_{MSG4}= 300ms – 30ms – 60ms

d_{MSG4}= **210ms (para o percurso 3)**

d_{MSG4}= microciclo – $t_{execução}$ AO_9 – $t_{execução}$ PID_8

d_{MSG4}= 300ms – 30ms – 60ms

d_{MSG4}= **210ms (para o percurso 4)**

d_{MSG4}= microciclo – $t_{execução}$ AO_9 – $t_{execução}$ PID_8

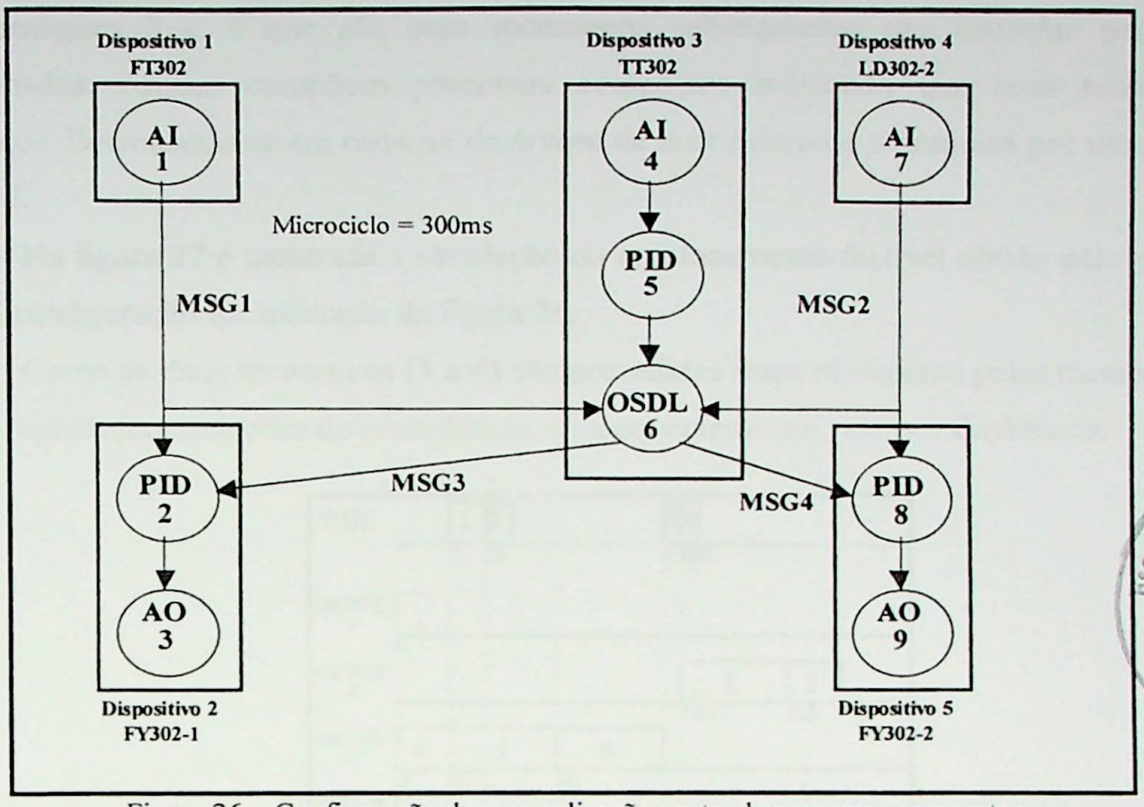
d_{MSG4}= 300ms – 30ms – 60ms

d_{MSG4}= **210ms (para o percurso 8)**

Portanto, para a determinação do release de cada mensagem para que sejam atendidos todos os percursos, estipula-se aquele que é o maior release dentre todos os releases obtidos para cada mensagem dentre os percursos existentes. Quando existe um release que é denominado dinâmico, este se sobressai perante os releases calculados e somente terá seu valor definido durante a fase de escalonamento.

Para a determinação do deadline de cada mensagem para que sejam atendidos todos os percursos, considera-se aquele que é o menor deadline dentre todos os deadlines obtidos para cada mensagem dentre os percursos existentes.

Nos exemplos seguintes deste trabalho será omitida a apresentação dos respectivos cálculos efetuados neste e no exemplo anterior, pois, os próximos exemplos são mais complexos desprendendo de uma grande quantidade de cálculo das características temporais.



Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca MAUÁ BIM

Figura 26 – Configuração de uma aplicação contendo processos compostos

As características temporais e os relacionamentos entre blocos funcionais e mensagens da aplicação da figura 26 são apresentadas através da tabela 3. Na mesma tabela também são definidas as listas de *publishers* e *subscribers*, existentes na configuração da aplicação mostrada pela figura 26.

Tabela 3 – Características temporais, relacionamentos e listas da aplicação mostrada na figura 26

Mensagens	***Release, Deadline* (ms)**	**Relacionamentos entre MSGs e BFs**	**Lista de *Publishers***	**Lista de *Subscribers***
MSG1	0, 140	AI_1	AI_1	—
MSG2	0, 140	AI_7	AI_7	—
MSG3	dinâmico, 210	AI_4, PID_5, $OSDL_6$	—	PID_2
MSG4	dinâmico, 210	AI_4, PID_5, $OSDL_6$	—	PID_8
Relacionamentos entre BFs			$AI_4 \rightarrow PID_5$	$PID_2 \rightarrow AO_3$
				$PID_8 \rightarrow AO_9$
Relações de precedência MSG1 precede MSG3 MSG1 precede MSG4 MSG2 precede MSG3 MSG2 precede MSG4				

Observando-se a configuração da figura 26, após a transmissão das mensagens 1 e 2,

as mensagens 3 e 4 que são suas mensagens subsequentes são liberadas para serem escalonadas. Ambas compõem processos compostos indicando que seus *releases* são definidos dinamicamente em cada nó da árvore de busca durante a pesquisa por uma solução factível.

Na figura 27 é mostrada a simulação do escalonamento factível obtido pelo programa para a configuração da aplicação da figura 26.

Como as duas mensagens (3 e 4) são precedidas respectivamente pelas mensagens 1 e 2, elas apresentam relações de precedência, consequentemente, *releases* dinâmicos.

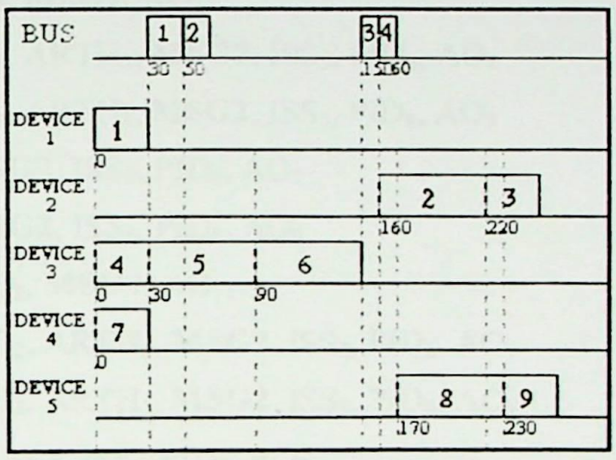


Figura 27 – Simulação do escalonamento factível da aplicação da figura 26

Além disso, como mostrado pela figura 27, os blocos funcionais 4, 5 e 6 que representam *links* internos ao dispositivo 3, não apresentam relacionamentos entre blocos funcionais como ocorreu com a aplicação da figura 24. Portanto, o dispositivo 3 transmite as mensagens 3 e 4 as quais caracterizam os relacionamentos necessários para que os blocos funcionais 4, 5 e 6 possam ser escalonados. Desta forma, como os relacionamentos foram absorvidos pelas mensagens, não necessitando a criação de um relacionamento de bloco funcional para bloco funcional, pois, este tipo de relacionamento somente é utilizado quando algum bloco funcional que possui um *link* interno (dispositivo que não transmite mensagem) não participa dos relacionamentos entre mensagens e blocos funcionais.

3. Exemplo de uma Aplicação Complexa

A figura 28 representa uma aplicação que possui uma quantidade razoável de blocos funcionais, possuindo ainda uma maior quantidade de percursos e uma maior de blocos funcionais e mensagens. A maior quantidade de percursos implica em uma maior quantidade

de relacionamentos envolvidos entre mensagens e blocos funcionais, e entre os próprios blocos funcionais. Estes relacionamentos garantem a sincronização ou sincronismo dos blocos funcionais e mensagens, e seus respectivos sequenciamentos.

Como esta aplicação é uma aplicação utilizada na prática por uma indústria real por força de cláusulas de confidencialidade, serão omitidos os dados referentes aos tempos de processamento dos blocos funcionais e mensagens.

Os percursos existentes obtidos através da configuração da aplicação do usuário apresentada pela figura 28 são:

1) AI_1, ALM_4, PID_2, **MSG1**, AO_{11}

2) AI_1, ALM_4, PID_2, $ARTH_3$, **MSG2**, ISS_5, PID_6, AO_7

3) AI_1, ALM_4, PID_2, $ARTH_3$, **MSG2**, ISS_5, PID_8, AO_9

4) AI_1, $ARTH_3$, **MSG2**, ISS_5, PID_6, AO_7

5) AI_1, $ARTH_3$, **MSG2**, ISS_5, PID_8, AO_9

6) AI_{12}, **MSG4**, PID_2, **MSG1**, AO_{11}

7) AI_{12}, **MSG4**, PID_2, $ARTH_3$, **MSG2**, ISS_5, PID_6, AO_7

8) AI_{12}, **MSG4**, PID_2, $ARTH_3$, **MSG2**, ISS_5, PID_8, AO_9

9) AI_{12}, **MSG4**, ALM_{10}, ISS_5, PID_6, AO_7

10) AI_{12}, **MSG4**, ALM_{10}, ISS_5, PID_8, AO_9

11) AI_{12}, **MSG4**, ISS_5, PID_6, AO_7

12) AI_{12}, **MSG4**, ISS_5, PID_8, AO_9

13) MAI_{13}, **MSG5**, PID_6, AO_7

14) MAI_{13}, **MSG6**, PID_8, AO_9

As características temporais tratadas (nesta aplicação a mensagem 4 possui características temporais redundantes, por exemplo: d_{MSG4}= 412ms / d_{MSG4}= 613ms) e os relacionamentos entre blocos funcionais e mensagens estão especificados através da tabela 4.

Na figura 29 pode-se observar que todos os blocos funcionais foram devidamente escalonados, sendo executados em sincronismo com as respectivas mensagens.

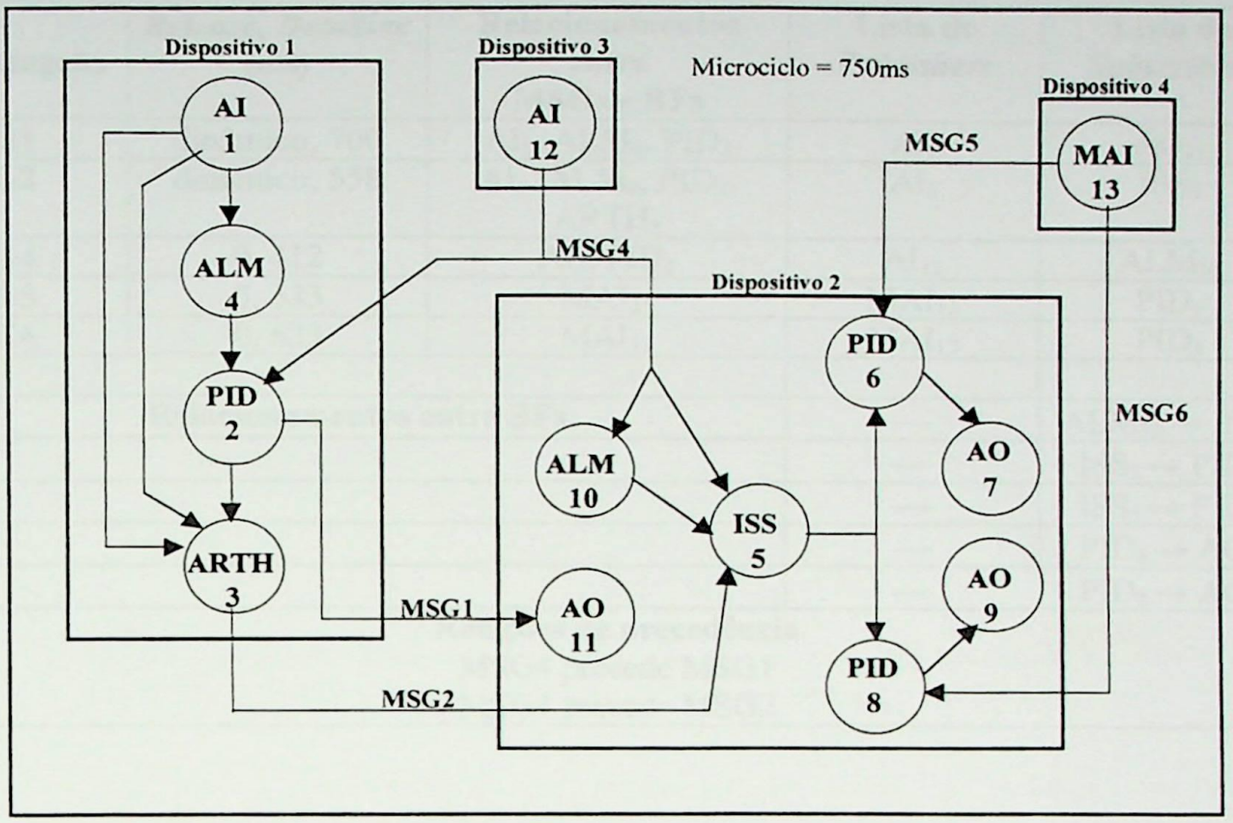


Figura 28 – Configuração de uma aplicação contendo uma maior quantidade de BFs e percursos

Analisando-se a figura 29, pode-se observar que somente quando a mensagem 2 foi transmitida para o barramento, os blocos funcionais 5, 8, 9, 6 e 7 foram escalonados, respeitando-se assim a todos os percursos envolvidos com os respectivos blocos funcionais.

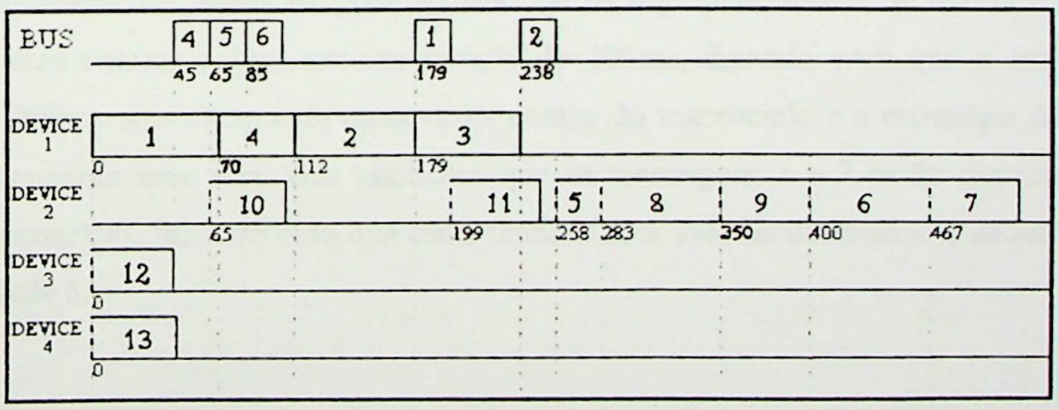


Figura 29 – Simulação do escalonamento factível da aplicação da figura 28

Tabela 4 – Características temporais, relacionamentos e listas da aplicação mostrada na figura 28

Mensagens	*Release, Deadline* (ms)	Relacionamentos entre MSGs e BFs	Lista de *Publishers*	Lista de *Subscribers*
MSG1	dinâmico, 700	AI_1, ALM_4, PID_2	AI_1	AO_{11}
MSG2	dinâmico, 558	AI_1, ALM_4, PID_2, $ARTH_3$	AI_1	ISS_5
MSG4	0, 412	AI_1, PID_2	AI_{12}	ALM_{10}
MSG5	0, 633	MAI_{13}	MAI_{13}	PID_6
MSG6	0, 633	MAI_{13}	MAI_{13}	PID_8
Relacionamentos entre BFs			—	$ALM_{10} \rightarrow ISS_5$
			—	$ISS_5 \rightarrow PID_6$
			—	$ISS_5 \rightarrow PID_8$
			—	$PID_6 \rightarrow AO_7$
			—	$PID_8 \rightarrow AO_9$
Relações de precedência MSG4 precede MSG1 MSG4 precede MSG2				

4. EXEMPLO DE UMA APLICAÇÃO COMPOSTA

Uma aplicação é chamada de composta neste trabalho, quando ela for formada por mais de uma estratégia de controle com microciclos diferentes. Quando isto ocorre, os microciclos diferentes requerem o cálculo de MMC entre eles para determinação do macrociclo da aplicação.

Na aplicação da figura 30, pode-se observar os seguintes valores de microciclo: 200ms e 400ms. Estes valores geram um macrociclo de 400ms, fazendo com que a estratégia de controle de 200ms seja executada duas vezes dentro do macrociclo e a estratégia de controle de 400ms, somente uma vez. Isto implicará que as mensagens 1 e 2 serão executadas duas vezes no macrociclo, fazendo com que estas tenham dois valores de *release* e *deadline*, como mostra a tabela 5.

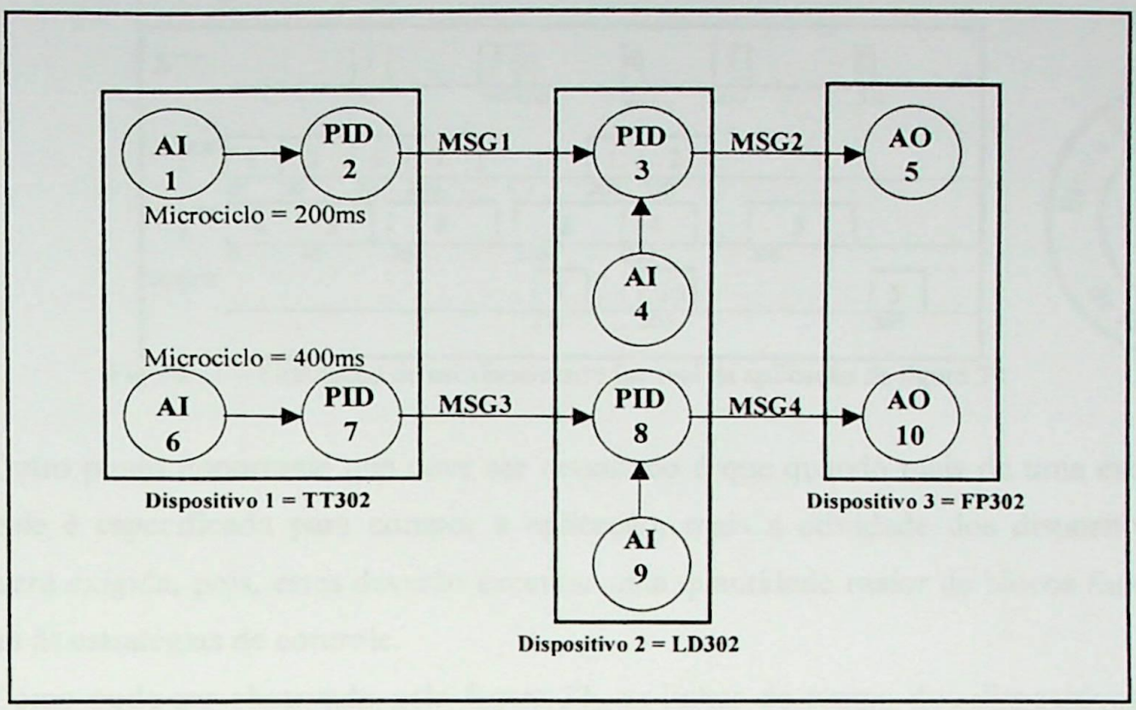


Figura 30 – Configuração de uma aplicação contendo estratégias de controle concorrentes

As características temporais e os relacionamentos entre blocos funcionais e mensagens da aplicação estão especificados através da tabela 5.

Tabela 5 – Características temporais, relacionamentos e listas da aplicação mostrada na figura 30

Mensagens	***Release, Deadline* (ms)**	**Relacionamentos entre MSGs e BFs**	**Lista de *Publishers***	**Lista de *Subscribers***
MSG1	0, 100 e 200, 300	AI_1, PID_2	AI_1	—
MSG2	dinâmico, 170 e dinâmico, 370	AI_4, PID_3	AI_4	AO_5
MSG3	0, 300	AI_6, PID_7	AI_6	—
MSG4	dinâmico, 370	AI_9, PID_8	AI_9	AO_{10}
Relações de precedência MSG1 precede MSG2 MSG3 precede MSG4				

Analisando-se a figura 31 pode-se observar que a estratégia de controle definida pelos blocos funcionais AI_1, AI_4, PID_2, PID_3 e AO_5 distribuídas pelos dispositivos são novamente escalonados a partir do instante de tempo 200ms, atendendo à multiplicidade da estratégia de controle de 200ms dentro do macrociclo.

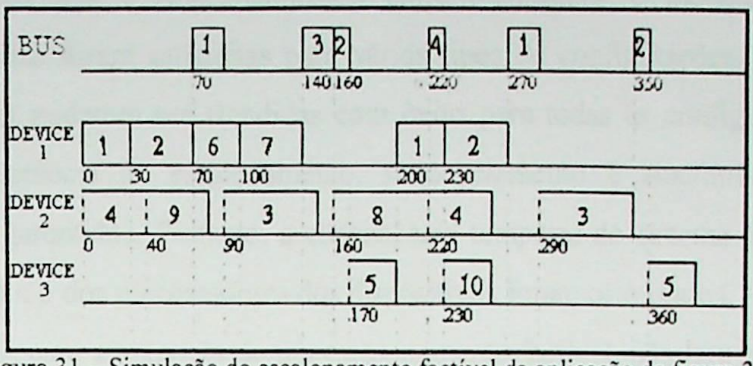


Figura 31 – Simulação do escalonamento factível da aplicação da figura 30

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca MAUÁ BIM

Outro ponto importante que deve ser ressaltado é que quando mais de uma estratégia de controle é especificada para compor a aplicação, mais a atividade dos dispositivos no sistema será exigida, pois, estes deverão executar uma quantidade maior de blocos funcionais referentes às estratégias de controle.

Como pode ser observado pela figura 31, as linhas de tempo dos dispositivos 1 e 2 estão praticamente preenchidas em comparação com as linhas de tempo dos exemplos das aplicações simples e da aplicação complexa. Isto determina que quanto mais estratégias de controle estão sendo executadas em um sistema *fieldbus*, mais se exige dos dispositivos e se tem uma maior quantidade de comunicação entre dispositivos.

5. Resultados Obtidos e Discussão sobre a Eficiência Alcançada

O algoritmo proposto foi executado para diversos tipos de aplicações onde algumas destas aplicações eram casos reais específicos desenvolvidos por empresas. Para que os escalonamentos obtidos através do algoritmo proposto pudessem ser visualizados, foi necessário o desenvolvimento de um programa para apresentação dos resultados. Este programa, apresenta o escalonamento factível através de uma simulação do estado *on-line* para o primeiro período de macrociclo do sistema, exibindo as mensagens e os blocos funcionais.

A ocorrência de ciclos durante o processo de busca é eliminada, pois, a pesquisa por uma solução somente continua enquanto existirem mensagens com *deadlines* maiores que o *deadline* da mensagem mais atrasada. Este conjunto é definido no nó raiz. Para cada novo nó aberto, uma destas mensagens deste conjunto é escolhida, até, no máximo, para todas as suas mensagens. De forma que cada novo nó não pode fornecer a mesma solução que um nó criado anteriormente, evitando-se assim o ciclo.

Analisando-se os resultados obtidos, foi observado que as restrições temporais, as

relações de precedência e relacionamentos entre mensagens e blocos funcionais das aplicações analisadas foram satisfeitas para vários tipos de configurações. Desta forma, as mensagens cíclicas puderam ser atendidas com êxito para todas as configurações testadas, bem como, a coerência de escalonamento, sequenciamento e sincronismo dos blocos funcionais foram garantidos. Portanto, a consistência temporal do sistema foi satisfeita, e o uso do barramento e o dos processadores dos dispositivos foram otimizados.

Além disso, para a obtenção de um escalonamento factível, não houve necessidade da abertura de muitos nós na árvore de busca, devido à utilização da técnica *branch-and-bound* e do uso de uma heurística adequada baseada no conhecimento do problema. Isto foi comprovado através da execução do algoritmo para vários tipos de aplicações, ilustrando estes resultados através dos exemplos deste capítulo, como mostra as informações contidas na tabela 6.

Tabela 6 – Informações referentes a eficiência alcançada com o algoritmo proposto

	Quantidade de MSG	Quantidade de BF	Tempo de Processamento (μs)
Exemplo da Seção 2.1	3	6	248
Exemplo da Seção 2.2	4	8	347
Exemplo da Seção 3	5	13	367
Exemplo da Seção 4	6	15	717

Todos os exemplos discutidos acima foram executados em um computador PENTIUM III 450MHz com 64 Mb de memória. Os 3 primeiros exemplos de aplicações, os quais apresentaram tempos inferiores em relação ao último exemplo, geraram somente 1 nó na árvore de busca no caminho da solução factível. No entanto, o último exemplo gerou 2 nós no caminho da solução factível. Como esperado, pode ser notado que quanto mais estratégias de controles concorrentes existirem na aplicação do usuário, mais aumenta a complexidade, demandando um maior tempo de processamento do algoritmo proposto, para a obtenção de uma solução factível.

Pode-se ainda visualizar em função destes dados e através da figura 32, que o tempo de processamento varia mais abruptamente em função da quantidade de mensagens existentes na aplicação do que com a quantidade de blocos funcionais.

Para o levantamento da eficiência alcançada, observou-se que o tempo necessário para o processamento do algoritmo proposto e a quantidade de nós abertos, baseado no pior caso fica na escala de mili segundos enquanto que para o algoritmo proposto é de micro segundos.

Assim, o algoritmo teve em média uma redução em torno de 34% no tempo de pesquisa em relação a uma busca exaustiva. Porém, observou-se que esta eficiência pode ser mais expressiva com o aumento da quantidade de mensagens, pois com o processo de poda da árvore de busca, cada vez mais, podem-se eliminar soluções não satisfatórias, que não otimizariam o barramento e o uso da CPU dos dispositivos, com escalonamentos não factíveis.

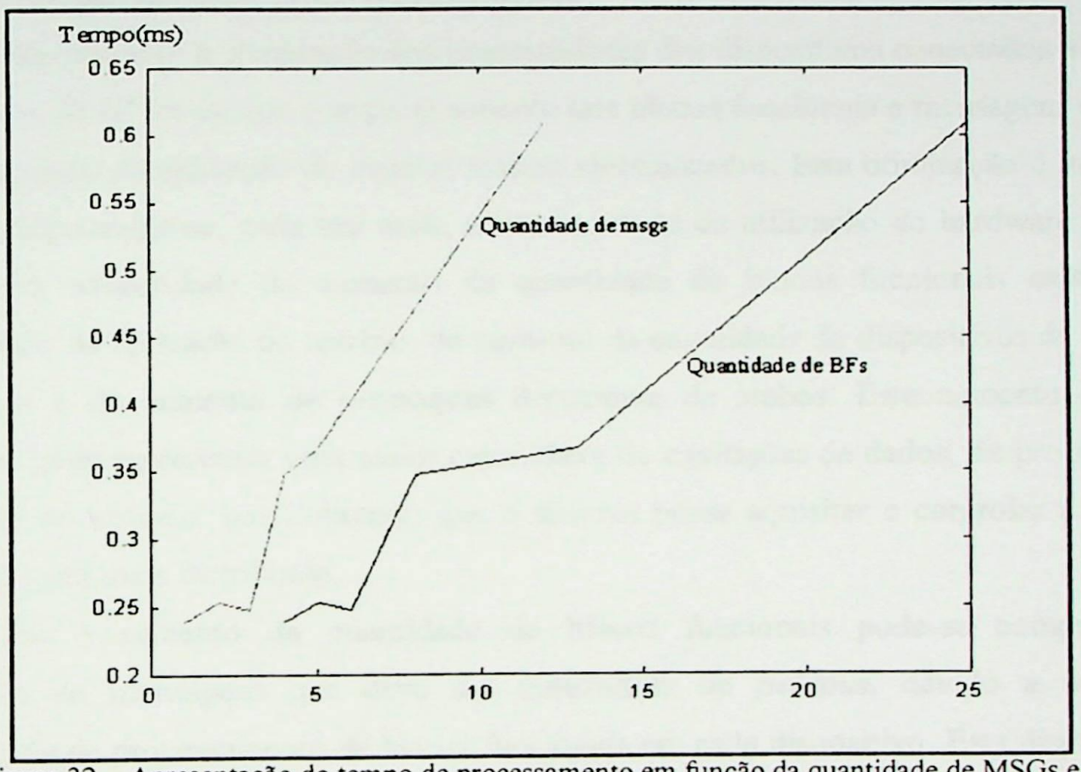


Figura 32 – Apresentação do tempo de processamento em função da quantidade de MSGs e BFs

CAPÍTULO 7 – CONTRIBUIÇÕES E TRABALHOS FUTUROS

Com a execução deste trabalho pôde-se ampliar o domínio de aplicabilidade de um sistema *fieldbus*, apresentando como resultado a obtenção de características importantes existentes nas aplicações do usuário, que devem ser levadas em consideração para que aplicações complexas possam ser implementadas.

Neste trabalho a otimização dos processadores dos dispositivos conectados ao fieldbus foi atendida, de tal forma que o sequenciamento dos blocos funcionais e mensagens existentes na configuração da aplicação do usuário fossem sincronizados. Esta otimização é importante, pois na prática exige-se, cada vez mais, a maximização da utilização do hardware instalado, gerado pela necessidade do aumento da quantidade de blocos funcionais existentes na configuração da aplicação do usuário, do aumento da quantidade de dispositivos de campo no barramento e do aumento de mensagens decorrente de ambos. Este aumento de blocos funcionais pode representar uma maior capacidade de aquisições de dados, de processamento e atuações no sistema, possibilitando que o sistema possa aquisitar e controlar a planta de forma cada vez mais distribuída.

Com o aumento da quantidade de blocos funcionais pode-se comprometer a quantidade de mensagens que deve ser transmitida no *fieldbus*, devido a uma maior concorrência de processamento de blocos funcionais em cada dispositivo. Esta disputa gerada no processador de cada dispositivo teve de ser tratada e considerada durante o desenvolvimento dos módulos do escalonador para que uma ampla quantidade de aplicações pudesse ser desenvolvida na prática pela indústria ao usar o referido escalonador.

O atendimento das restrições temporais e a obtenção de um escalonamento de blocos funcionais sincronizado com o escalonamento das mensagens, garantem a aplicabilidade do algoritmo proposto na prática.

A sincronização entre as mensagens e os blocos funcionais é garantida através da criação de relacionamentos gerados em função dos percursos envolvidos com a estratégia de controle. O procedimento utilizado e a forma de tratamento dos percursos garantem assim o desenvolvimento de uma grande variedade de aplicações impostas pelos processos ou plantas industriais.

O procedimento proposto para o escalonamento dos blocos funcionais em função do escalonamento das mensagens, proporciona uma complexidade maior ao problema, mas automatiza a busca da solução casada, ao invés de só verificar a posteriori se as soluções

geradas separadamente para o escalonamento das mensagens e blocos funcionais são compatíveis.

Portanto, para que o algoritmo proposto pudesse estar apto para ser aplicado na prática sobre um sistema fieldbus, foi necessário o desenvolvimento de funções adicionais e módulos que garantem a atividade operacional do sistema.

Como proposta para a execução de trabalhos futuros observou-se a possibilidade de se pesquisar: a aplicação ou adaptação do escalonador para novos protocolos como *High Speed Ethernet*, a eficiência do uso de outras heurísticas para a seleção das mensagens como *Least Laxity*, a eficiência do uso de outros algoritmos para o atendimento das mensagens acíclicas *on-line*, e a adaptação do algoritmo para mais de um segmento sincronizados na rede, utilizando *Bridges*.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ALMEIDA, L.M.P. **Flexibility and timeliness in fieldbus-based real-time systems.** Portugal, Out. 1999. 196 p. tese(doutorado) – Universidade de Aveiro.

[2] CAVALIERI, S.; Di STEFANO, A.; MIRABELLA, O. Optimization of acyclic bandwidth allocation exploiting the priority mechanism in the fieldbus data link layer. **IEEE Transactions on Industrial Electronics**, v.40, n.3, p.297-306, Jun 1993.

[3] DECOTIGNIE, J-D.; RAJA, P. Fulfilling Timing Constraints in Fieldbus. **In: International Conference on Industrial Electronics**, IEEE, 1993. p.519-24.

[4] FRANCO, L.R.H.R. **Escalonamento de Mensagens para a Comunicação no Fieldbus**. São Paulo, 1998. 124p. tese(Doutorado) – Escola Politécnica, Universidade de São Paulo.

[5] FRANCO, L.R.H.R.; Rillo M. Escalonamento de Mensagens no Fieldbus. **I Workshop em Sistemas de Tempo Real**. 16° Simpósio Brasileiro de Redes de Computadores. Rio de Janeiro, Maio 1998.

[6] JÚNIOR, A.M.; MAGALHÃES, M.F. Uma Estratégia Eficiente para o Escalonamento "On-Line" de Processos Esporádicos em Sistemas de Tempo Real Crítico. **I Workshop em Sistemas de Tempo Real**. 16° Simpósio Brasileiro de Redes de Computadores. Rio de Janeiro, Maio 1998.

[7] LAINE, T. **Modelisation d'applications reparties pour la configuration automatique d'un bus de terrain**. Nancy, 1991. 146p. Tese (doutorado) – École Nationale Superieure d'Electricité et de Mécanique de Nancy – Institut National Polytechnique de Lorraine.

[8] LIU, C.L.; LAYLAND, J. Scheduling algorithms for multiprogramming in a hard-real-time environment. **Journal of the ACM**, v.20, n.1, p.46-61, Jan. 1973.

[9] MITSCHKE, S. **Tecnologia Fieldbus Foundation Advanced**. Itajubá, Fieldbus Foundation, 1999.

[10] PAPADIMITRIOU, C.H.; STEIGLITZ, K. **Combinatorial Optimization – Algorithms and Complexity**. Capítulo 18, Englewood Cliffs, Prentice-Hall, 1982, XVI, 496 p.

[11] PENG, D.; SHIN, K.G.; ABDELZAHER, T.F. Assignment and scheduling communicating periodic tasks in distributed real-time systems. **IEEE Transactions on Software Engineering**, v.23, n°12, p.745-757, Dec. 1997.

[12] SHEPARD, T.; GAGNÉ, J.A.M. A pre-run-time scheduling algorithm for hard real-time systems. **IEEE Transactions on Software Engineering**, v.17, n°7, p.669-677, Jul. 1991.

[13] SMAR. **System 302**. Sertãozinho,1999. n.p.

[14] TANEMBAUM, A.S. **Sistemas Operacionais Modernos**. 1.ed. Rio de Janeiro, Prentice Hall do Brasil, 1995.

[15] THOMESSE, J.P.; LAINE, T. The fieldbus application services. **In: International Conference on Industrial Electronics, Control and Instrumentation**, 15º, Philadelphia, 1989. Proceedings. New York, IEEE, 1989. p.526-30.

[16] VASQUES, F.; JUANOLE, G. Fieldbus MAC mechanisms for hard-real-time data communications support. **Proc. Open Bus Systems**, Munich – Germany, p.225-32, Nov. 29-30, 1993.

[17] XU, J.; PARNAS, D.L. On satisfying timing constraints in hard-real-time systems. **IEEE Transactions on Software Engineering**, v.19, n.1, p.70-84, Jan. 1993.

[18] XU, J.; PARNAS, D.L. Scheduling processors with releases times, deadlines, precedence, and exclusion relations. **IEEE Transactions on Software Engineering**, v.16, n°3, p.360-9, Mar. 1990.