

DEPARTAMENTO DE
ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

**Implementação de uma
Camada de Aplicação
sobre a Rede CAN**

FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Rua dos Bragas, 4099 Porto Codex – PORTUGAL

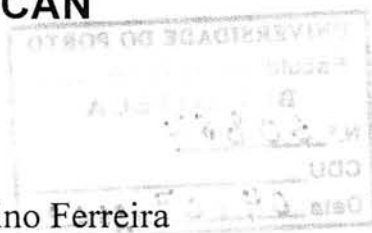
Implementação de uma Camada de Aplicação sobre a Rede CAN

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores



Implementação de uma Camada de Aplicação sobre a Rede CAN

Luis Miguel Moreira Lino **Ferreira**



Licenciado em Engenharia Electrotécnica e de Computadores
pela Faculdade de Engenharia da Universidade do Porto

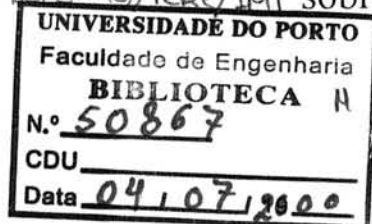
Dissertação submetida para satisfação parcial dos requisitos de grau de mestre em
Engenharia Electrotécnica e de Computadores
(Área de especialização de Informática Industrial)

Dissertação realizada sob a supervisão de
Professor Doutor Artur Capelo Cardoso,
do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

Porto, Março de 1996

Implementação de uma
Camada de Aplicação

CB1.3(043)/FERC/IMP sobre a Rede CAN



Luis Miguel Moreira Lino Ferreira

(llf@tormentas.fe.up.pt)

Dissertação realizada com o apoio do gabinete PRAXIS XXI e da JNICT

*Á Anita e
aos meus Pais*

Agradecimentos

Durante a realização desta dissertação recebi provas de amizade e solidariedade por parte de muitas pessoas. Assim, desejo expressar os meus sinceros agradecimentos:

- ▷ ao Professor Doutor Artur Capelo Cardoso, na qualidade de orientador desta tese pelo total apoio e constante disponibilidade sempre manifestados durante o desenvolvimento daquela;
- ▷ aos meus amigos Armando, Isabel, Filipe, Miguel e Sérgio, pela sua disponibilidade para ler e reler a tese, pelos conselhos e principalmente pela sua amizade;
- ▷ a todos os colegas do Instituto de Sistemas Robótica, em particular à Cristina ao Paulo e ao Mário;
- ▷ à Faculdade de Engenharia da Universidade do Porto e ao Instituto de Sistemas e Robótica pela cedência de equipamentos e instalações imprescindíveis à realização deste trabalho;
- ▷ à JNICT concretamente ao Programa Praxis XXI pelo suporte financeiro.

A todos os meus amigos que me apoiaram, e especialmente aos meus Pais e à Anita.

Resumo

O desenvolvimento das redes de computadores e, principalmente, o abaixamento dos preços do *hardware* e do *software*, veio permitir que se substituíssem as várias ligações ponto a ponto entre sensores, actuadores e a(s) unidade(s) de controlo por redes usualmente chamadas: **redes de campo**.

Entre as redes de campo, o CAN, apesar de pouco conhecido, devido a muitas das suas aplicações serem embebidas, domina perto de 50% do mercado de redes de campo industriais. Este domínio deve-se ao vasto leque de aplicações possíveis para o CAN, às suas potencialidades e aos baixos preços dos seus integrados.

Neste tese é efectuado uma análise ao protocolo CAN assim como a um conjunto de camadas de aplicação constituído pelos: SDS, DeviceNet, CANKingdom, CAL e CANOpen, tendo-se escolhido para implementação o CAL (*CAN Application Layer*). É feita uma análise do CAL utilizando a metodologia orientada ao objecto OMT.

A análise orientada ao objecto é utilizada como base para uma implementação do CAL. Esta implementação disponibiliza as funcionalidades principais, nomeadamente os serviços de gestão da rede, controlo de erro e todos os tipos de objectos CMS (Can Message Specification).

Conclui-se que este trabalho fornece a base para investigações futuras na área do controlo distribuído (DCS) fornecendo-se e caracterizando-se um dos seus constituintes fundamentais, a rede de comunicações e o respectivo protocolo.

Índice

Agradecimentos	vii
Resumo	ix
Índice	xi
Índice de Figuras	xvii
Acrónimos	xxii
1. Introdução	1
1.1 Estrutura	2
2. Redes de Campo	3
2.1 Introdução.....	3
2.2 Controlo por computador	3
2.3 Ligações ponto a ponto	6
2.4 Integração de redes de campo no controlo por computador	7
2.5 Redes de campo.....	8
2.6 Características de redes de campo	12
2.6.1 Topologia	12

2.6.2 Tipo de tráfego	13
2.6.3 Partilha de Objectos ou mensagens	14
2.6.4 Integração	14
2.6.5 Dimensões	14
2.6.6 Ambiente envolvente.....	15
2.6.7 Configuração	16
2.6.8 Manutenção.....	16
2.6.9 Selecção dos Serviços pelo Utilizador	17
2.6.10 Compatibilidade.....	17
2.6.11 Performance e Capacidades	17
2.7 Alguns Exemplos.....	18
2.7.1 Aplicações Industriais.....	18
2.7.2 Aplicações na Indústria Automóvel	18
2.7.3 Aplicações na Indústria Aeroespacial.....	19
2.7.4 Aplicações em Navios	19
3. O CAN (Controller Area Network)	21
3.1 Motivação - Porquê o CAN?.....	21
3.2 Normas do CAN	22
3.2.1 Camada física	24
3.2.2 Mecanismo de acesso ao meio.....	24
3.2.3 Tipos de tramas.....	26
3.2.4 Detecção e Controlo de Erros	28

3.2.5 Filtros de mensagens	30
3.2.6 Implementações	30
3.3 Comparação	33
4. O CAN em Sistemas Distribuídos Tempo Real	37
4.1 Sistemas Tempo- Real	37
4.2 Classificação das tarefas de um sistema de controlo	38
4.2.1 Relativa aos instantes de activação	38
4.2.2 Relativa à Criticalidade dos requisitos temporais	39
4.3 Sistemas e Controlo Distribuído	40
4.4 Elementos para a modelização de uma rede CAN	43
4.4.1 Estrutura de um nó	44
4.4.2 Tipos de tráfego	47
4.4.3 Tempo de transmissão das mensagens	49
4.4.4 Tempo de preparação de mensagens	52
4.4.5 Tempo de colocação da mensagem no <i>buffer</i> de recepção e tempo de espera no <i>buffer</i>	52
4.4.6 Fontes de variabilidade dos tempos de resposta	53
4.5 Análise dos atrasos nas comunicações	55
4.6 Conclusões	57
5. Análise do CAL	63
5.1 Métodos de análise Orientados ao Objecto	58
5.2 Análise OO do CAL	59

5.2.1 O protocolo	60
5.2.2 O CMS	67
5.2.3 NMT	86
5.2.4 O DBT	90
5.2.5 O LMT	97
5.3 Considerações finais	100
6. Implementação do CAL	102
6.1 Implementação de uma análise OO utilizando uma linguagem estruturada ..	102
6.2 Estrutura do protocolo de comunicações	104
6.2.1 Relação com o modelo OSI	105
6.2.2 Descrição da Camada de Ligação Lógica	107
6.3 Análise dos métodos evocação de pedidos e recepção de confirmações	109
6.4 Análise dos métodos para a recepção de indicações e geração de respostas	110
6.4.1 Geração de respostas	111
6.5 Comportamento escolhido	111
6.6 Plataforma de Teste	112
6.6.1 Controlador CAN	113
6.7 Serviços escolhidos	113
6.7.1 Serviços CMS	114
6.7.2 Serviços do NMT	114
6.7.3 Serviços DBT e do LMT	115

6.8 Análise das capacidades do CAL - Medidas de velocidade dos serviços	115
6.9 Conclusões	118
7. Conclusões	120
7.1 Trabalho futuro	121
7.1.1 Uso do campo de identificação do CAN	143
Anexo A	131
Anexo B	159
Referências	167
Bibliografia	176

Índice de Figuras

Figura 1 - Sistema controlado por computador, segundo [Bennet, 94].....	5
Figura 2 - Exemplo de uma instalação utilizando ligações ponto a ponto. ©Honeywell - Micro Switch	6
Figura 3 - Controlo por computador utilizando uma rede para acesso aos dispositivos.....	8
Figura 4 - Instalação idêntica à da Figura 2 utilizando uma rede de campo. ©Honeywell - Micro Switch	9
Figura 5 - a) topologia em estrela, b) topologia em barramento, c) topologia mista.....	13
Figura 6 - Tipos de conectores.....	15
Figura 7 - Exemplo de uma instalação industrial, ©Honeywell - Micro Switch.....	18
Figura 8 - Número de integrados com CAN vendidos (excluindo a industria automóvel). Fonte Selectron GmbH [Rostan, 1997] em 1995.	22
Figura 9 - Alcance na norma do CAN.	23
Figura 10 - Método de acesso ao meio.	25
Figura 11 - Formato da trama de dados com identificador de 11bits.	27
Figura 12 - Trama remota.....	27
Figura 13 - Trama de erro.....	28
Figura 14 - Estados de erro de um dispositivo CAN.	29
Figura 15 - Estrutura de um nó de comunicações genérico.	45
Figura 16 - Atrasos na comunicação.	46
Figura 17 - Caracterização do tráfego acíclico	48
Figura 18 - Representação de um Nó do CAL.....	66

Figura 19 - Interacção entre o Gestor de Mensagens e outros objectos.....62

Figura 20 - Estrutura do dicionário de COBs.63

Figura 21 - Diagrama de estados do objecto **Gestor de Mensagens**.64

Figura 22 - Diagrama de estados do objecto **Gestor de Mensagens**, especialização do estado de **Inicialização**.64

Figura 23 - Diagrama de estados do objecto **Gestor de Mensagens**, especialização do estado de **Recepção de Mensagens**.65

Figura 24 - Diagrama de estados do objecto **Gestor de Mensagens**, especialização do estado de **Transmissão de Mensagens**.66

Figura 25 - Diagrama de estados do objecto **Gestor de Mensagens**, especialização do estado de **Transmissão de Mensagens**.66

Figura 26 - Objectos CMS.....68

Figura 27 - Especialização do diagrama de objectos para variáveis básicas.69

Figura 28 - Modelo dinâmico do objecto **BasicVariableReadOnlyServer**.71

Figure 29 - Modelo dinâmico de uma variável **BasicVariableReadOnlyClient**.72

Figure 30 - Modelo dinâmico de uma variável **BasicVariableWriteOnlyServer**.73

Figure 31 - Modelo dinâmico de uma variável **BasicVariableWriteOnlyClient**.73

Figura 32 - Modelo dinâmico de uma variável **BasicVariableReadWriteServer**.74

Figura 33 - Modelo dinâmico de uma variável **BasicVariableReadWriteClient**.75

Figura 34 - Especialização para variáveis multiplexadas.76

Figura 35 - Especialização para domínios.77

Figura 36 - Diagrama de estados para o objecto **BasicDomainServer**.78

Figura 37 - Diagrama de estados para o objecto **BasicDomainServer**, especialização do estado **Em Download**.79

Figura 38 - Diagrama de estados para o objecto BasicDomainServer , especialização do estado Em Upload	79
Figura 39 - Diagrama de estados para o objecto BasicDomainClient	80
Figura 40 - Diagrama de estados para o objecto BasicDomainClient , especialização do estado Download	81
Figura 41 - Diagrama de estados para o objecto BasicDomainClient , especialização do estado Upload	82
Figura 42 - Diagrama de estados para o objecto BasicDomainClient , especialização do estado Abort	83
Figura 43 - Especialização do diagrama de objectos para eventos.....	84
Figura 44 - Diagrama de estados StoredEventServer	85
Figura 45 - Diagrama de estados StoredEventClient	86
Figura 46 - Diagrama de Objectos do NMT.....	87
Figura 47 - Diagrama de estados de um Remote Node	88
Figura 48 - Diagrama de estados de um objecto Node	89
Figura 49 - Diagrama de objectos do DBT.....	91
Figura 50 - Diagrama de estados do DBT Service Element .(geral).....	92
Figura 51 - Diagrama de estados do DBT Service Element . (CreateUsrDefinitionInd).....	94
Figura 52 - Diagrama de estados do DBT Service Element . (CreateUsrDefinitionReq).....	95
Figura 53 - Diagrama de estados do DBT Service Element . (DeleteUsrDefinitionInd).....	96
Figura 54 - Diagrama de estados do DBT Service Element . (DeleteUsrDefinitionReq).....	96
Figura 55 - Diagrama de objectos do LMT.....	98
Figura 56 - Diagrama de estados do objecto LMT Slave	99
Figura 57 - Camadas necessárias para a implementação do CAL.....	105
Figura 58 - Primitivas utilizadas pelos serviços.....	106
Figura 59 - Camada de aplicação do CAL.....	125

Figura 60 - Diagrama de estados de um nó remoto	128
Figura 61 - Esquema OO de uma <i>I/O Connection</i>	131
Figura 62 - Esquema OO de uma <i>Explicit Messaging Connection</i>	132
Figura 63 - Modelo de objectos de um dispositivo DeviceNet.....	134
Figura 64 - Modelo de objectos de um dispositivo DeviceNet.....	139
Figura 65 - Modelo hierárquico OO de um dispositivo SDS.	141
Figura 66 - Modelo geral de um dispositivo SDS.	142
Figura 67 - Representação da rede utiliza pelo CANKingdom.	146
Figura 68 - Objecto.....	154
Figura 69 - Associação entre objectos.....	154
Figura 70 - Multiplicidade das associações.	155
Figura 71 - Agregação.....	155
Figura 72 - Generalização ou herança.	156
Figura 73 - Relações principais no modelo dinâmico.	157
Figura 74 - Notação do modelo Funcional.	158

Acrónimos

OSI - Open Systems Interconnection

CiA - Can in Automation

ISO - International Standard Organization

CAN - Controller Area Network

CAL - CAN Application Layer

CIM - Computer Integrated Manufacturing

CMS - Can Message Specification

COB - Communication Object

COB-ID - COB Identifier

DBT - Distributor

FIP - Factory Instrumentation Protocol

PROFIBUS - Process Field Bus

LME - Layer Management Entity

LMT - Layer Management

LLC - Logical Link Control

MAC - Medium Access Control.

MDI - Medium Dependent Interface

MCS-5 - Monitoring and Control System

MMS - Manufacturing Message Specification

Acrónimos

NMT - Network Management

NRZ - Non Return to Zero

ODVA - Open DeviceNet Vendors Association

OO - Orientado ao Objecto

OSI - Open Systems Interconnection

PDO - Process data object

SDO - Service data object

SDS - Smart Distributed System

PLS - Physical Layer Signaling.

PMA - Physical Medium Attachment

1. Introdução

A pressão para o aumento da eficiência das unidades industriais tem dado lugar à automatização dos seus sistemas de produtivos, a automatização esta que se fez incrementando à utilização de sistemas controlados por computador. Estes computadores necessitam de obter informação do ambiente exterior através de sensores, processá-la e voltar a actuar sobre o sistema de modo a controlá-lo.

O desenvolvimento das redes de computadores e, principalmente, o abaixamento dos preços do *hardware* e *software* veio permitir que se substituíssem as várias ligações ponto a ponto entre sensores, actuadores e a unidade central de controlo, por redes, usualmente chamadas: **redes de campo**.

A utilização deste tipo de redes veio permitir a diminuição dos custos das instalações industriais e permite a utilização dos sensores e actuadores em todo o seu potencial. Este potencial por sua vez tem tido grande expansão já que a evolução da electrónica dos nós de rede acoplados a sensores e actuadores permite a inclusão de funções cada vez mais sofisticadas como por exemplo: a autocalibração e o diagnóstico.

Várias tem sido as soluções encontradas para a interligação de sensores e actuadores de onde se podem destacar o PROFIBUS, o FIP, o ASI e o Interbus-S. Discretamente e sem se notar uma rede originalmente destinada à indústria automóvel tem dominado o mercado da automação industrial com um quota de cerca de 50%, o CAN.

O CAN é uma rede capaz de atingir velocidades de 10Kbit/s até 1Mbit/s, suportando mensagens de 0 a 8 bytes, com grande imunidade ao ruído, sendo capaz de suportar aplicações com necessidades de tempo real.

O desenvolvimento de ferramentas para a utilização de uma rede CAN em aplicações de controlo por computador, especialmente as destinadas ao suporte de aplicações de controlo distribuído, foram as grande motivações para esta tese que foca o seu estudo nas características desta e na sua utilização em automatização industrial.

1.1 Estrutura

No capítulo 2 serão focados os aspectos e características gerais relativas a redes de campo.

No capítulo 3 será abordado o funcionamento da rede CAN e será feita a comparação das várias normas relativas à camada de aplicação, nomeadamente o CAL, o CANOpen, o DeviceNet e o SDS.

Os problemas relativos à utilização da rede CAN em sistemas distribuídos tempo real são referidos no capítulo 4.

No capítulo 5 é apresentada a análise orientada aos objectos da camada de aplicação CAL.

O capítulo 6 descreve os aspectos relativos a uma implementação prática do CAL.

O 7 e último capítulo apresenta as conclusões e as perspectivas de desenvolvimento e trabalho futuro.

Em anexo podem encontrar-se pequenas explicações relativas às camadas de aplicação CAL, CANOpen, SDS, DeviceNet e CANKingdom, assim como um pequeno resumo da metodologia orientada ao objecto utilizada.

2. Redes de Campo

2.1 Introdução

Neste capítulo será descrito o ambiente no qual as redes de campo se vão inserir, referindo o controlo por computador como um dos elementos principais dos sistemas de automação. De seguida, serão descritas as principais motivações para o uso de redes de campo, as suas características as suas vantagens e desvantagens face a uma solução que utilize ligações ponto a ponto equivalentes. No final serão apresentados alguns exemplos significativos da utilização de redes deste tipo.

2.2 Controlo por computador

Com o aparecimento dos computadores durante a década de 40, e o desenvolvimento de sistemas digitais, surgiu a possibilidade de substituir os sistemas de controlo analógico por sistemas de controlo digital. As primeiras aplicações, durante a década de 60, foram essencialmente na indústria química [Bennet, 94] aplicadas ao controlo de processos químicos em refinarias de petróleo e na produção de produtos químicos.

Em 1974 o aparecimento dos microprocessadores (μP), veio permitir a substituição dos grandes computadores das décadas anteriores por outros mais pequenos, cada um encarregue da sua tarefa e obedecendo às ordens de outros computadores

hierarquicamente superiores. Nas décadas posteriores o desenvolvimento explosivo dos μP aliado a uma redução exponencial do seu preço veio permitir uma maior utilização de μP em sistemas de controlo. As suas aplicações alargam-se com o desenvolvimento dos microcontroladores (μC), que não são mais do que μP especialmente concebidos para aplicações de controlo embebidas, através da junção de μP com dispositivos de entradas e saídas analógicas e digitais, geradores de PWM, interfaces de comunicações, e outros dispositivos feitos à medida para cada aplicação.

Hoje em dia os dispositivos que utilizam sistemas de controlo por computador vão desde máquinas de lavar, passando pelos carros actuais, até grandes e complexas instalações industriais.

Um sistema controlado por computador é caracterizado por recolher informação do meio exterior, processar essa informação e actuar de acordo com a imagem do meio exterior recolhida.

Segundo [Bennet, 94] as actividades de um sistema controlado por computador incluem:

- Aquisição de dados
- Sequenciamento de operações
- Controlo da realimentação de um sistema
- Supervisão
- Análise de dados
- Armazenamento de dados
- Interface com o utilizador

Estas actividades têm por objectivo:

- Eficiência
- Facilidade de operação
- Segurança
- Aumento da qualidade

- Redução do desperdício
- Redução do impacto ambiental
- Redução da carga de trabalho directamente relacionada com o processo

Um sistema controlado por computador pode ser estruturado [Bennet, 94] como mostra a figura 1.

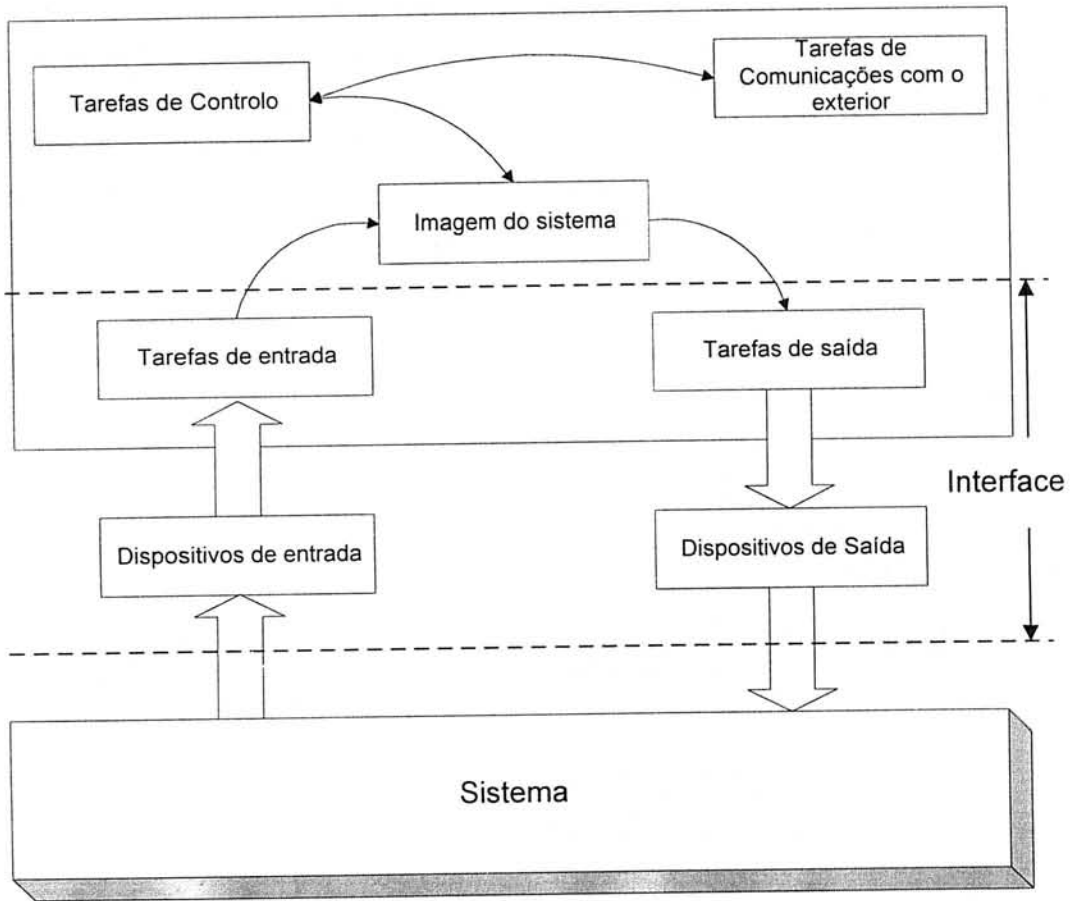


Figura 1 - Sistema controlado por computador, segundo [Bennet, 94].

As tarefas de entrada poderão, por exemplo, através dos sensores, ler ciclicamente o estado do sistema de modo a actualizar a sua imagem. Dentro de algumas restrições¹ as tarefas de entrada disponibilizam às tarefas de controlo uma imagem do processo.

¹ As restrições relacionam-se com o tempo em que é feita a actualização das imagens de uma ou mais variáveis, e com o tipo de actualização: comandada pelo tempo ou por eventos.

Devido à actualização da imagem do sistema ter um determinado atraso em relação ao verdadeiro sistema, surge um dos principais problemas do controlo de sistemas por computador: o tempo e a problemática que lhe está associada.

2.3 Ligações ponto a ponto

Em sistemas de automação industrial é usual a ligação de sensores e actuadores directamente às cartas de E/S dos autómatos programáveis ou controladores (figura 2). Esta característica implica que a ligação de um sensor seja feita através de pelo menos um cabo. No caso de instalações de pequena dimensão o conjunto de cabos que se ligam ao PLC é bastante reduzido. Caso a instalação atinja um dimensão média, com algumas dezenas de sensores, isso implicará que exista igual número de cabos com um comprimento total do conjunto de cabos que pode chegar facilmente às várias centenas de metros. Este elevado número de cabos acarreta custos muito altos na construção do sistema e torna o sistema mais complexo, menos flexível e com elevados custos para a sua reparação e alteração.

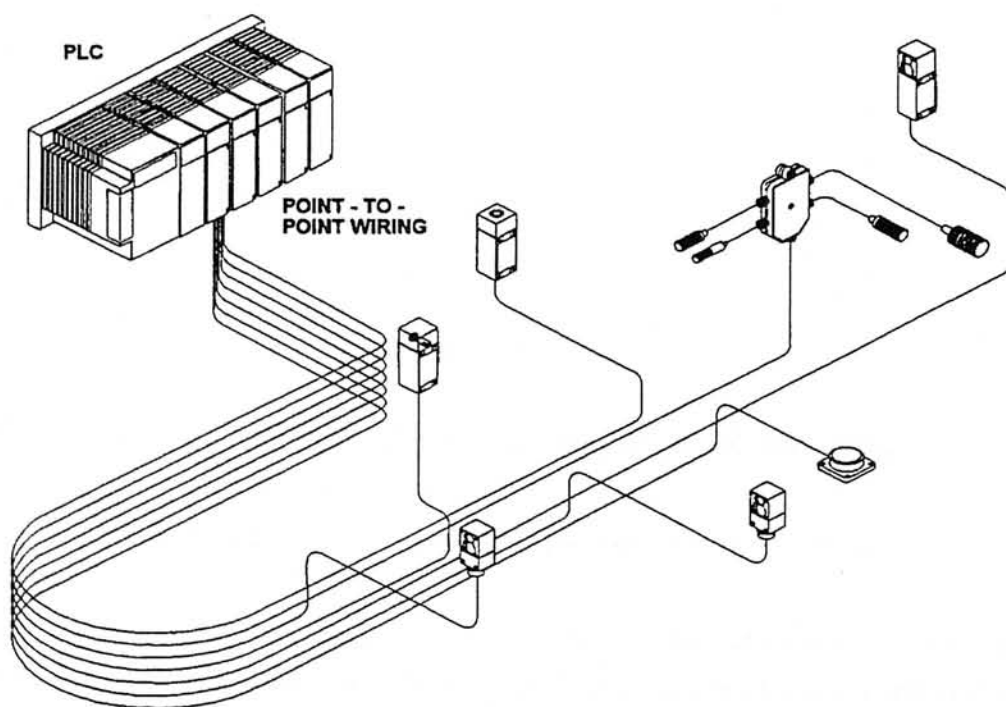


Figura 2 - Exemplo de uma instalação utilizando ligações ponto a ponto. ©Honeywell
- Micro Switch

Usualmente as ligações são feitas utilizando anéis de corrente (4-20mA) para sinais analógicos e sinais diferenciais em tensão para sinais digitais.

A utilização de sinais em corrente (4-20mA) deve-se à maior imunidade ao ruído e às baixas quedas de tensão no cabo. A norma (4-20mA) permite a detecção de cortes no cabo de ligação, uma vez que o valor de 0mA não é um valor admissível e pode ser detectado como um erro pelo controlador.

O uso de sinais diferenciais para o caso dos sinais digitais permite: uma elevada imunidade ao ruído, taxas de transmissão bastante elevadas e interfaces bastante simples e baratas.

2.4 Integração de redes de campo no controlo por computador

Uma rede de campo permite substituir as ligações ponto a ponto por apenas um cabo ao qual se vão ligar todos os outros sensores. A utilização de uma rede de campo permite a distribuição da capacidade de processamento pelos vários nós constituintes de um sistema. Assim, tanto para métodos de controlo hierárquico, como para métodos de controlo distribuído o elemento de comunicações e o protocolo de comunicações usado são elementos imprescindíveis e extremamente importantes para o desempenho do sistema.

Desenvolvendo mais o diagrama de blocos apresentado na figura 1, de modo a substituir as ligações ponto a ponto por uma rede de campo, obtém-se o esquema representado na Figura 3.

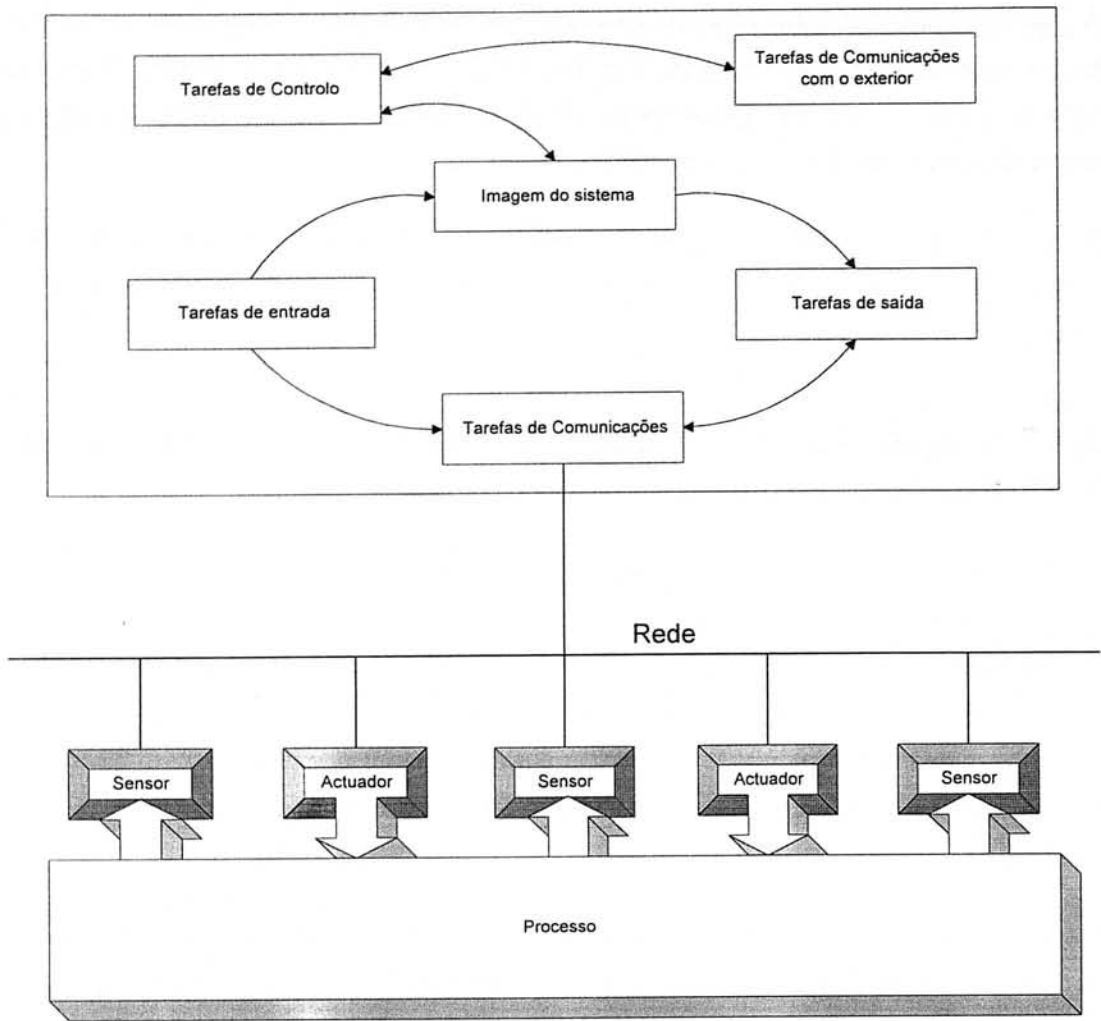


Figura 3 - Controlo por computador utilizando uma rede para acesso aos dispositivos.

2.5 Redes de campo

As redes de campo, utilizadas nos níveis mais baixo da hierarquia do CIM, são redes destinadas à ligação de sensores e actuadores e por vezes, como é o caso do PROFIBUS e do FIP, podem também interligar controladores de célula e PLCs. As redes de campo são também utilizadas em navios, em aviões, em máquinas industriais e nos automóveis.

As primeiras utilizações de redes de campo destinavam-se à substituição de várias ligações ponto a ponto digitais e analógicas. A figura 4 representa uma instalação idêntica à da figura 2 utilizando uma rede de campo.

A redução dos custos do *hardware* e do *software* associado, permite que os fabricantes de sensores produzam sensores com cada vez maior elevada capacidade de

processamento e baixo custo, capazes de executarem um conjunto complexo de funções como, por exemplo: diagnóstico, calibração, configuração, validação de dados, etc. A este tipo de sensores é vulgar chamar-se **sensores inteligentes**.

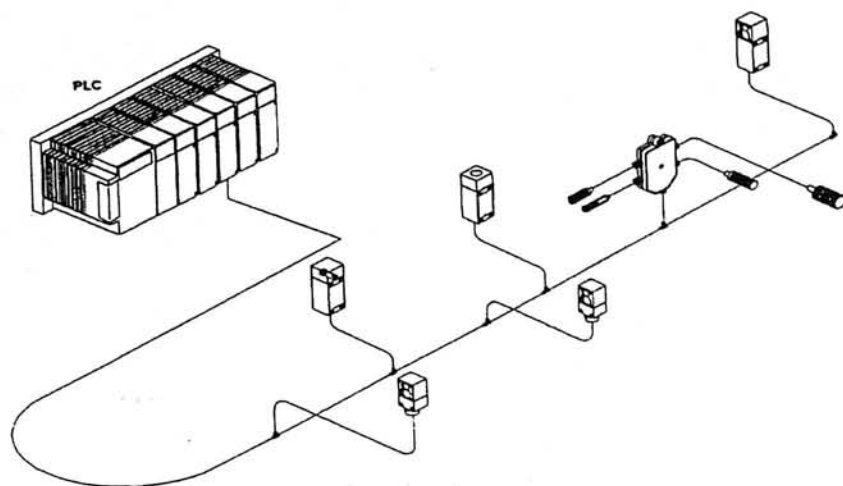


Figura 4 - Instalação idêntica à da Figura 2 utilizando uma rede de campo. ©Honeywell - Micro Switch

Existem, por exemplo, sensores ópticos com capacidade para detectar se o seu óculo se encontra sujo e transmitir ao controlador o seu mau funcionamento. Este simples facto permite que se efectue manutenção preventiva do sensor antes que este avarie e leve à paragem ou a acidentes na instalação em que se encontra integrado.

Alguns dispositivos de E/S conseguem detectar se alguma das suas saídas se encontra acidentalmente em curto-circuito, e enviam uma mensagem de emergência para o controlador a sinalizar o acontecimento. Os sensores inteligentes executam funções que normalmente teriam que ser executadas de forma centralizada pelo controlador do sistema, ou seja, permitem a distribuição de algumas funções de processamento pelos dispositivos constituintes do sistema.

Em instalações de média dimensão, o número de cabos e o seu comprimento total pode ser muito elevado, logo os custos relativos à cablagem podem atingir uma percentagem muito significativa dos custos totais da instalação. Foi esta uma das principais motivações para o desenvolvimento de redes de campo: a redução dos custos da cablagem.

As principais vantagens da utilização de uma rede de campo, relativamente a uma ligação ponto a ponto equivalente são expostas de seguida:

- **Diminuição dos custo da cablagem:**

A utilização de um só cabo para a ligação de todos os dispositivos permite uma redução substancial do comprimento total de cabo e a simplificação da cablagem. É ainda possível a ligação dos nós em várias configurações diferentes (de acordo com a figura 5), nomeadamente em estrela, em barramento, misto de estrela e barramento, etc. A utilização de diferentes tipos de cabos permite um elevado grau de escalabilidade. O tipo de cabo a utilizar depende dos requisitos impostos pela aplicação de acordo com o tamanho da instalação, a velocidade de comunicação e a potência requerida pelo nós² consumida pelos dispositivos do sistema.

- **Sistema de comunicações fiável:**

A informação trocada entre os nós destas redes é transmitida sob a forma digital utilizando protocolos de correcção e detecção de erros. A maior parte dos protocolos existentes para redes de campo permite a utilização de serviços confirmados de modo a aumentar ainda mais fiabilidade das comunicações.

- **Utilização de nós inteligentes:**

A ligação de um sensor ou actuador a uma rede de campo obriga, em alguns casos, à introdução de um microprocessador³. Aproveitando as capacidades deste é possível a realização pelo dispositivo de algumas funções, usualmente feitas apenas pelo controlador central, tal como a linearização do sinal, calibração, armazenamento dos valores adquiridos ao longo do tempo, etc. A utilização das possibilidades referidas permite libertar o controlador de algumas das suas funções relativas aos nós, que podem passar a ser executadas localmente em cada nó ou em concentradores.

- **Processamento distribuído:**

² Os cabos utilizados na interligação de uma rede de campo podem ter algumas linhas adicionais para a transmissão de potência.

³ A ligação de um sensor a uma rede de campo pode dispensar um μC como é o caso dos módulos SLIO [NSSLIO, 95] utilizados no CAN.

A utilização de dispositivos inteligentes em alguns dos nós do barramento permite aliviar a carga a que é submetido o controlador, o que pode levar à utilização de controladores menos sofisticados ou até mesmo, no caso extremo, à dispensa de um controlador central, passando as funções de controlo a serem executadas em cada um dos nós interligados através da rede de campo, utilizando a capacidade em excesso que se encontra disponível nos nós [Carvalho, 95].

- **Acessibilidade da informação:**

Todas as mensagens difundidas pelos dispositivos podem ser acedidas em qualquer local do barramento de uma forma totalmente transparente, independentemente do lugar onde foram produzidos. Por exemplo, possibilita uma reconfiguração rápida da rede se for incluído um novo sensor ou mudando de outro de sítio.

- **Diminuição dos custos de manutenção:**

Através da utilização de ferramentas de monitorização e configuração específicas é possível detectar rapidamente falhas na rede e nos sensores. Dado que se utiliza apenas um cabo é bastante mais fácil encontrar qualquer erro que exista nas ligações do cabo aos vários sensores.

Em alguns casos é possível a configuração completa da rede através de ferramentas apropriadas, o que permite a configuração remota dos nós da rede sem que exista a necessidade de ter acesso físico aos sensores.

- **Custos totais:**

Os custos totais de um sistema que utilize redes de campo são normalmente inferiores aos de um sistema que utilize ligações ponto a ponto, além disso as funcionalidades do sistema e dos dispositivos são geralmente bastante superiores.

As possibilidades abertas devido à utilização de uma rede para interligar todos os nós permite, e em alguns casos obriga, à divisão do sistema num conjunto de unidades funcionais independentes, que cooperam umas com as outras através da rede de campo. Assim, obtém-se como vantagem adicional uma maior facilidade no projecto de todo o sistema, uma vez que cada uma das várias unidades do sistema pode ser desenvolvida em separado, [Fredrikosson, 95].

No entanto existem algumas desvantagens :

- **Partilha do meio:**

A partilha do meio por várias estações implica a existência de mecanismos de acesso ao meio que garantam as restrições temporais de cada mensagem. Este facto obriga à utilização de protocolos de comunicação específicos para aplicações com requisitos tempo real.

- **Coerência da informação distribuída:**

Devido à distribuição inerente a redes deste tipo é necessário a existência de mecanismos que garantam que a informação existente em cada nó, relativamente aos dados de outros nós, seja coerente e actualizada. Outro problema inerente a um sistema distribuído é a sincronização dos relógios, de modo a que o sistema possa ser utilizado em operações como: aquisição de dados ou controlo de movimento em que é necessário conhecer o tempo com exactidão.

- **Menor fiabilidade das ligações:**

A utilização de um só cabo para a ligação de todos os nós pode ser indesejável pois o corte accidental deste pode levar à falha de toda a instalação. Num sistema em que a ligação é feita, ponto a ponto, a falha de uma ligação provoca no sistema uma falha parcial, interrompendo a transmissão de dados pelos sensores afectados. Em sistemas críticos pode ser necessário a introdução de um meio de transmissão extra, montando em cada sensor dois pontos de acesso a barramentos distintos.

2.6 Características de redes de campo

2.6.1 Topologia

A topologia de uma rede de campo, Figura 5, deve poder ser bastante flexível permitindo ligações em barramento, em estrela ou mistas com alguns troços montados em estrela e outros em barramento. A possibilidade de utilização de topologias mistas permite reduzir o comprimento total do cabo.

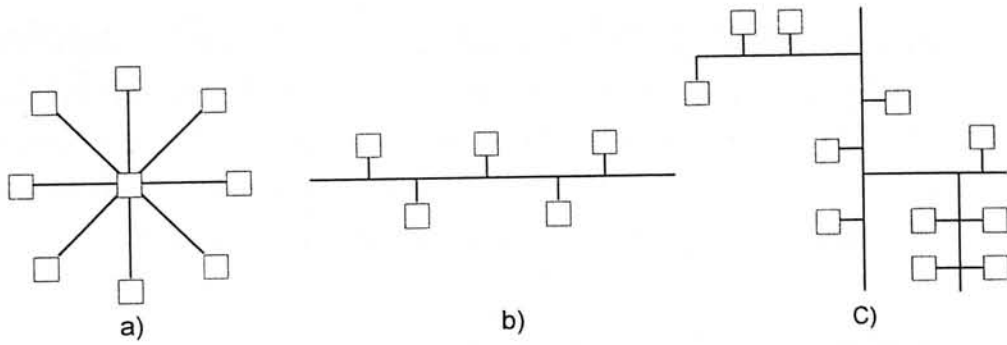


Figura 5 - a) topologia em estrela, b) topologia em barramento, c) topologia mista.

É possível a utilização de diferentes tipos de cabos conforme as exigências de cada sensor e do sistema nomeadamente: as suas dimensões, necessidade ou não de alimentação e velocidade de transferência de dados.

2.6.2 Tipo de tráfego

Uma rede de campo deve ser capaz de assegurar que a transferência de mensagens entre nós se faz cumprindo as restrições temporais associadas a cada uma das mensagens (*deadlines*). As restrições temporais podem ir de alguns μ s na transmissão de eventos numa máquina a alguns segundos na transmissão de variáveis de um processo químico.

É necessário distinguir dois tipos de comunicações:

- **Comunicações críticas:** neste caso é necessário assegurar que a comunicação de uma mensagem seja feita dentro de um janela temporal após a sua produção, caso contrário o valor que é enviado para o consumidor desse objecto deve ser considerado como inválido. O não cumprimento das restrições temporais pode levar à falha do sistema.
- **Comunicações não-críticas:** este tipo de comunicações apenas exigem que a transmissão de dados se efectue de forma correcta.

A existência de comunicações críticas na rede implica o uso de mecanismos que garantam o cumprimento das restrições temporais em qualquer situação de tráfego, o que pode ser conseguido por exemplo através de *polling* ou de prioridades associadas às mensagens, ou seja mecanismos que garantam o determinismo no funcionamento da rede.

O tráfego da rede também pode ser classificado em tráfego cíclico ou periódico e em tráfego acíclico ou aperiódico. É usual a existência no mesmo sistema deste dois tipos de tráfego [SP50, 91]; por exemplo, o tráfego cíclico existe quando um sensor envia repetidamente o valor de uma medida para o controlador central, quando ao tráfego acíclico existe normalmente para a transmissão de eventos tais como alarmes.

2.6.3 Partilha de Objectos ou mensagens

Em algumas aplicações, os dados de dispositivos de medição e controlo (sensores e actuadores) são utilizados por diferentes dispositivos presentes na rede. Logo é necessário que o protocolo de comunicações da rede permita a partilha de dados por vários dispositivos. Este facto leva à conclusão de que a utilização de um protocolo de comunicações baseado apenas na arquitectura cliente/servidor não é muito adequado para o uso em redes de campo, o que leva à utilização de outras arquitecturas nomeadamente o modelo Produtor/Consumidor ou o modelo Produtor/Consumidor/Distribuidor [Cardeira et al, 93]. Estes protocolos permitem a existência de vários consumidores para as variáveis e podem também utilizar o modelo cliente/servidor quando é necessário a comunicação directa com um módulo.

2.6.4 Integração

O CIM implementa a interligação dos vários níveis da hierarquia numa fábrica [Portugal, 95], [Tovar, 95], [Groover, 87]; por isso, uma rede de campo deve ser capaz, quando necessário, de suportar serviços de interligação com camadas superiores e ao mesmo tempo suportar serviços tempo real.

A integração pode fazer-se utilizando subconjuntos do MMS (como é o caso do Profibus e do FIP) ou através da tecnologias proprietárias ao fabricante (caso da arquitectura SINEC da Siemens [Stauß, 95] e [Peyrucat, 95]).

2.6.5 Dimensões

As dimensões físicas dos vários elementos constituintes de uma rede de campo devem poder ser suficientemente pequenas para que seja possível a sua utilização em instalações e sistemas de pequena dimensão. Assim, para que este requisito seja possível é necessário utilizar conectores de pequenas dimensões como por exemplo os conectores mini e micro utilizados nas normas CAN-OPEN [CANOpen, 96] e SDS [SDSphy, 94].

A utilização de diferentes tipos também traz vantagens adicionais em termos de custos e de flexibilidade das ligações.

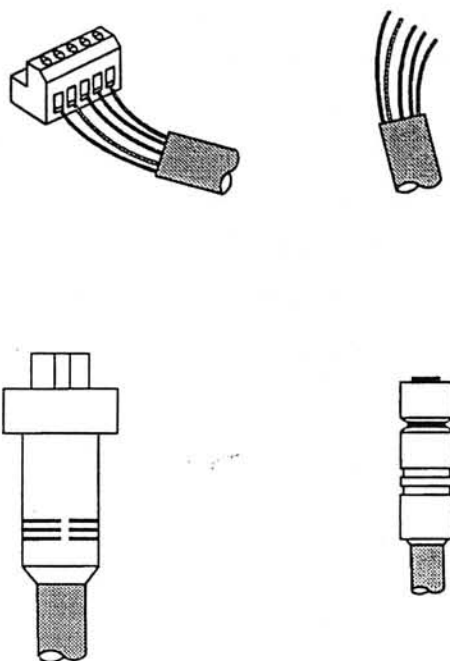


Figura 6 - Tipos de conectores

2.6.6 Ambiente envolvente

A introdução destas redes em ambiente agressivos (*harsh environments*) obriga os dispositivos e a rede a terem capacidades de:

- Retirarem a alimentação directamente da rede: é usual que um ou mais fios do cabo que interliga os vários dispositivos possam fornecer a energia necessária ao funcionamento deste ou como no caso dos sensores ASI ([ASI-Festo, 95] e [ASI-Siemens, 94]), que os sinais de comunicações e alimentação estejam presentes no mesmo cabo.
- Robustez: todos os equipamentos devem ser suficientemente robustos de modo a poder sobreviver em ambientes agressivos mecânica, química e termicamente. Alguns dispositivos especiais podem ser ligados em zonas com ambiente explosivo.
- Ruído Electromagnético: as redes de campo podem geralmente suportar intenso ruído electromagnético, muitas vezes presente em ambiente industrial devido a motores,

contactores, aparelhagem de soldadura, etc. Tal como devem suportar ruído também não devem gerar ruído electromagnético para além das normas legais.

2.6.7 Configuração

A configuração dos vários dispositivos de uma rede de campo poderá ser feita de forma automática através do dispositivo controlador do sistema ou através de um dispositivo especial de configuração ([Reigraf, 96] e [Uphoff, 95]). Alguns nós podem não ter a totalidade das suas funções de configuração acessíveis através da rede, nestes casos a configuração poderá ser feita na altura de fabrico ou na altura da sua montagem do sistema, por exemplo através de *DIP-Switchs*.

A configuração de um dispositivo envolve funções como:

- Mudança do seu endereço de rede;
- Configuração do *software* de comunicações;
- Descarga de programas para o dispositivo;
- Descarga de parâmetros do dispositivo; por exemplo, configuração das entradas e saídas digitais, dos níveis de saída analógicos, dos endereços das variáveis produzidas pelo dispositivo, etc.

2.6.8 Manutenção

As tarefas de manutenção do sistema podem vir facilitadas caso os dispositivos tenham capacidades de detecção de falhas e possam posteriormente transmitir essa informação ao controlador, ou caso não seja possível sinalizar através de outros meios, por exemplo utilizando LEDs. No entanto, nos dispositivos muito simples, caso dos módulos SLIO [NSSLIO, 95] utilizados no CAN, a detecção de falhas pode não ser automática e exigir a intervenção de um operador munido de ferramentas especiais para a detecção de falhas.

Devido às possibilidades de configuração referidas no ponto 2.6.7, pode ser necessário voltar a configurar o módulo; em outros casos a simples substituição por outro módulo é suficiente.

2.6.9 Selecção dos Serviços pelo Utilizador

Cada aplicação e cada dispositivo tem as suas necessidades específicas. Cada fabricante, que desenvolve um produto destinado a redes de campo tem, normalmente, a possibilidade de escolher quais os serviços e funcionalidades que cada um dos nós deve ter, de acordo com as necessidades do mesmo. São usualmente definidos pelas diferentes normas de redes de campo, serviços e funcionalidades obrigatórias e opcionais para cada nó, constituindo os chamados perfis, que definem as funções básicas e a maneira de as utilizar para dispositivos do mesmo tipo. Por exemplo, para dispositivos de entrada e saída do Profibus existe sempre o chamado *I/O Device Profile* que especifica as suas funcionalidades e a maneira como podem ser acedidas.

2.6.10 Compatibilidade

A compatibilidade com as soluções ponto a ponto pode ser assegurada através de módulos especiais, a funcionar como interface entre os dois tipos de soluções. É bastante usual encontrar-se em redes de campo módulos com funções de concentrador. Este módulo, tem como função adquirir sinais provenientes de vários sensores, com conexões ponto a ponto, do tipo 4-20mA e disponibilizar essa informação para os restantes dispositivos existentes na rede.

A compatibilidade com outras redes de campo é por vezes feita utilizando pontes, embora não seja frequente. Existem, por exemplo, pontes entre redes CAN e Ethernet [Ekiz *et al*, 96]. Existem também pontes que interligam redes de campo, mas com funcionalidades diferentes ou complementares. No caso do Profibus existem dispositivos destinados a fazer a interface com redes de campo do tipo ASI [Göddertz, 90].

2.6.11 Performance e Capacidades

A performance e as capacidades necessárias para uma rede de campo variam de acordo com a aplicação. Em [SP50, 91] são definidos os requisitos de várias aplicações; nomeadamente, são definidos os tempos de ciclo para a actualização das medidas, o tamanho das mensagens, o número e o tipo de dispositivos necessário, o tamanho e o tipo de cabos e outros pormenores específicos de cada aplicação.

2.7 Alguns Exemplos

A seguir serão apresentados alguns exemplos de aplicações que usam redes de campo.

2.7.1 Aplicações Industriais

As aplicações industriais são muito variadas e vão desde os sistemas de automação industrial até aos sensores no interior de máquinas.

É nestas indústrias que surge a maior parte das redes de campo, essencialmente, devido aos seus menores custos e simplicidade. O exemplo da figura 7 demonstra o variados conjunto de dispositivos que com possibilidade de ligação a uma rede de campo.

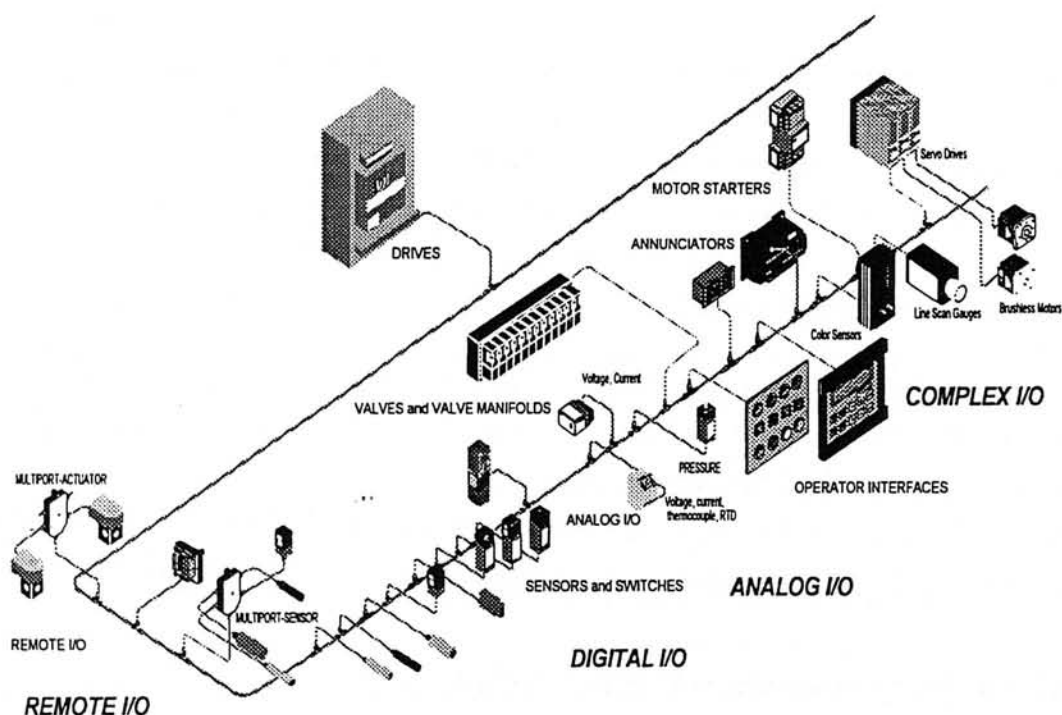


Figura 7 - Exemplo de uma instalação industrial, ©Honeywell - Micro Switch

2.7.2 Aplicações na Indústria Automóvel

A indústria automóvel tem começado a utilizar redes de campo para a ligação de sensores e actuadores nos automóveis. Neste caso, a utilização de redes de campo permite que o processo de montagem seja bastante mais simples quando comparado com

as anteriores ligações ponto a ponto; a sua utilização também permite uma diminuição do peso do carro e uma diminuição dos custos de montagem e de manutenção.

Uma das redes mais usada neste tipo de indústria é o CAN, [Moon, 1996] e [Kattwinkel, 95]. Em [DaimlerBenz, 95] é descrita a utilização do CAN nos novos modelos da classe E da Mercedes. Nestes modelos o CAN interliga os vários sensores utilizados no controlo do motor e na gestão do ambiente interior do carro. Aponta como vantagens principais a redução do custo do carro, do seu peso, da complexidade e uma maior capacidade para transferir informação e para expandir os vários sistemas.

2.7.3 Aplicações na Indústria Aeroespacial

A indústria aeroespacial utiliza redes de campo bastante robustas como é o caso da norma MIL-STD-1553B e da norma ARINC629. Tal como no caso dos carros, estas redes permitem uma grande simplificação da cablagem do avião e uma redução substancial de peso do avião, com consequências muito positivas na economia de combustível e uma consequente redução dos custos de exploração da aeronave.

Este tipo de redes tem grandes exigências de tempo real e de fiabilidade. Algumas das mensagens que circulam em aplicações deste tipo são mensagens críticas cuja perda de uma *deadline* pode levar à perda ou a danos muito elevados no avião. A elevada fiabilidade é uma das exigências desta aplicações; é, necessário assegurar que as mensagens são transmitidas e recebidas sem erros, que a rede permita a existência de redundância no meio e de nós sombra⁴ e que em caso de falha esta seja detectada.

2.7.4 Aplicações em Navios

Em navios é usual existirem grandes conjuntos de sensores e de actuadores, agrupados em vários subsistemas, nomeadamente: carga, radar, navegação, sonar, propulsão, controlo de incêndios, etc.

Estes subsistemas de sensores podem, por exemplo, ser interligados através da norma NMEA 0183, definida especialmente para instrumentos de navegação.

⁴ Nós com sistemas capazes de entrar em funcionamento quando nó principal falha.

Os vários subsistemas do navio podem ser interligados recorrendo à norma MiTS [MiTS, 94] que é uma rede integradora de todos os subsistemas do navio.

A principal necessidade para este tipo de aplicações é a integração da rede de campo com os níveis superiores de controlo no navio. Logo, existe a necessidade de se colocar *gateways* para interligar as redes de campo com as rede superiores.

São aplicações caracterizadas por um elevado número de nós que podem facilmente atingir as centenas. As principais vantagens da utilização de redes de campo em navios são: diminuição do número e comprimentos dos cabos e a diminuição dos custos de manutenção. A diminuição dos custos de manutenção deve-se à redução do número de cabos e conectores, pois menos componentes passam a estar expostos à corrosão existente em ambiente marítimo; além disso, as possibilidades de diagnóstico existentes nos sensores permitem uma mais fácil detecção de sensores avariados em locais pouco acessíveis do navio.

3. O CAN (Controller Area Network)

3.1 Motivação - Porquê o CAN?

O CAN foi concebido pela Bosch [Bosch, 91] com o objectivo de interligar módulos autónomos de forma robusta e fiável em automóveis. As suas primeiras utilizações em automóveis levou a que os fabricantes de circuitos integrados disponibilizassem *chips* com preços bastante competitivos. Este facto aliado à robustez e eficiência do CAN abriu as portas ao seu uso em automação industrial, equipamentos médicos, navios, domótica, robots, máquinas CNC e existem já algumas aplicações em aviões.

O grande número de campos de aplicações, levou igualmente à venda de um elevado número de integrados com o protocolo de CAN incorporado. Uma comparação, relativa ao número de nós vendidos, entre as principais redes de campo presentes no mercado da automação industrial pode ser vista na figura 8 em que se pode verificar que o CAN é a tecnologia mais utilizada dominando quase 50% do mercado mundial.

Apesar do número elevado de nós, e de ser líder do mercado, o CAN continua a ser bastante desconhecido. Isto, deve-se ao facto, de muitos sistemas utilizarem o CAN como barramento de comunicações interno, implementando protocolos de comunicações proprietários. São exemplos destes tipo de aplicações embebidas: os barramentos de expansão para PLCs, máquinas têxteis e de injeção de plástico e aparelhos médicos.

Devido ao elevado número de nós vendidos, os integrados com CAN incorporado têm um preço bastante baixo, o que permite a sua utilização em aplicações cada vez de menor valor.

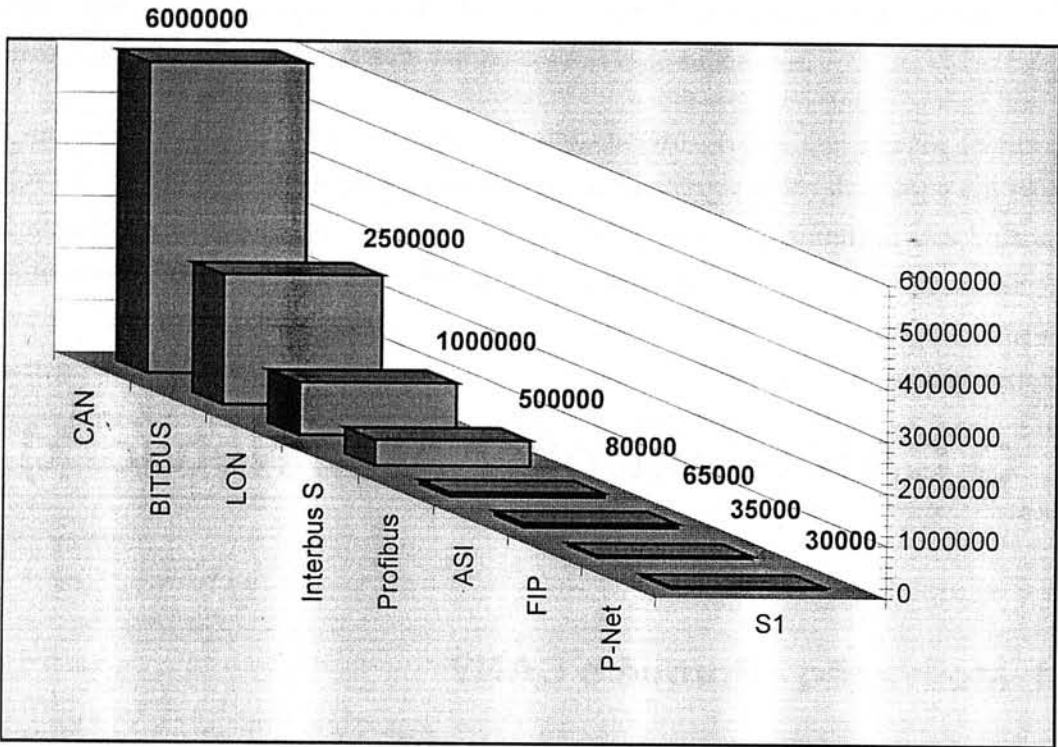


Figura 8 - Número de integrados com CAN vendidos (excluindo a industria automóvel).
Fonte Selectron GmbH [Rostan, 1997] em 1995.

Deste grande número de aplicações diferentes pode concluir-se que o CAN consegue responder a grande número de necessidades que vão desde aplicações em tempo real a aplicações de controlo complexas que envolvem a transferência de volumes médios de dados.

3.2 Normas do CAN

A especificação do CAN está contida em diversas normas em fase de finalização. As normas ISO/DIS11898 e ISO/DIS11519-1 apenas definem as subcamadas MAC (*Medium Access Control*) camada de acesso ao meio e a subcamada PLS (*Physical Layer Signaling*) sinalização da camada física. A figura 9 mostra o alcance das normas CAN utilizando como referência o modelo de comunicações OSI.

A ultima especificação do CAN, ISO/DIS11898 está separada nas partes A e B. A parte A (CAN 2.0A) define os formatos das tramas de 11 bits, enquanto que a parte B (CAN 2.0B) define os formatos de tramas estendidas, com identificadores de 29 bits.

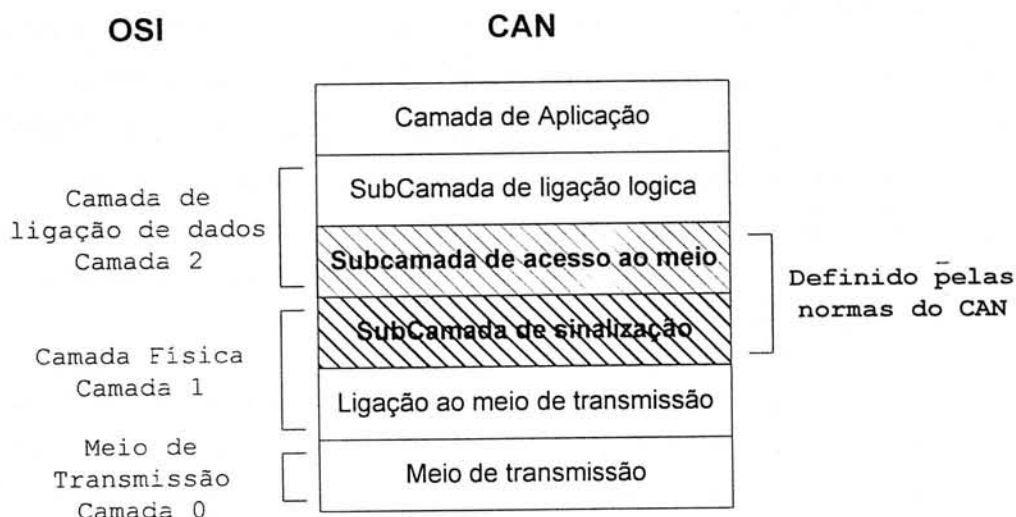


Figura 9 - Alcance na norma do CAN.

O CAN é uma rede baseada em objectos ou mensagens, isto é, a cada objecto está associado um identificador. As outras estações presentes na rede recebem a mensagem, procedem ao teste de aceitação e verificam se a mensagem tem interesse para esse nó ou não. A cada objecto está associado um identificador que irá definir a sua prioridade. Quanto mais baixo o valor do identificador maior é a sua prioridade. Utilizando o identificador do objecto os vários dispositivos presentes no barramento fazem o escalonamento do acesso ao meio, como se irá ver no ponto 3.2.2.

O objectivo inicial do CAN era a sua utilização em automóveis por isso foi especificado de modo a operar em ambientes extremamente duros mantendo as suas performances. Os mecanismos de detecção de erros asseguram que os erros sejam detectados com uma elevada probabilidade. A probabilidade de não detecção de um erro numa trama é de 10^{-13} . Foi calculado [Schofield, 96] que numa instalação a operar a 1Mb/s com uma utilização média de 50% da largura de banda, mensagens com um comprimento médio de 80 bits, ligada 8 horas por dia e 365 dias por ano, pode não ser detectado um erro numa trama uma vez todos os 100 dias.

Este capítulo irá descrever a rede de campo CAN, nomeadamente, a sua camada física e de ligação de dados e no final será feita a comparação das camadas de aplicação CAL e

CANOpen, DeviceNet, SDS e o CANKingdom. Será dado especial relevo ao CAL, pois foi esta, a camada de aplicação escolhida para implementação nesta tese.

3.2.1 Camada física

O CAN utiliza preferencialmente como método de transmissão a codificação em NRZ (*Non Return to Zero*) e sinal diferencial, utilizando apenas dois fios. É possível o funcionamento da rede utilizando adaptadores específicos (*transceivers*) para o CAN, ou utilizando outro tipo de adaptadores como, por exemplo, os utilizados para RS485. O único pré-requisito para a utilização de um meio diferente é a possibilidade de poder obter-se níveis recessivos e níveis dominantes nesse meio.

Os sinais do CAN são representados pelos níveis recessivo e dominante. Quando presente um nível recessivo e um nível dominante no meio é sempre o nível dominante que prevalece. É este princípio que possibilita o mecanismo de acesso ao meio utilizado pelo CAN.

A subcamada de ligação física não é definida pela norma tendo sido adoptados vários tipos de conectores e de meios pelas diversas normas que utilizam o CAN.

3.2.2 Mecanismo de acesso ao meio

Utilizando como exemplo a figura 10 veja-se o mecanismo de acesso ao meio utilizado pelo CAN. No pino de Tx está representado o sinal que é transmitido pela estação e no pino de Rx é representado o sinal que a estação lê do barramento. O módulo A vai tentar transmitir o identificador 1000 1101 111 e o módulo B vai tentar transmitir um mensagem com o identificador 1000 1001 111. Como pode ver-se na figura 10, as duas estações vão transmitindo o seu identificador que é igual até à transmissão do sexto bit. Nesta altura a estação A tenta transmitir um 1 (bit recessivo) e a estação B transmite um 0 (bit dominante), logo devido às propriedades da rede CAN a estação B irá detectar um bit dominante quando havia transmitido um bit recessivo, sai do meio e torna-se uma estação receptora.

O processo completo de acesso ao meio é o seguinte, em primeiro lugar é transmitido um bit dominante, de modo a sincronizar todas as estações que querem tentar aceder ao meio, depois estas transmitem o seu respectivo campo de identificação. As estações que tentam ter acesso ao meio também o monitorizam e quando detectam a presença de um bit dominante tendo transmitido um bit recessivo saem do meio e esperam até que o

barramento esteja de novo em repouso e possa iniciar-se um novo período de arbitragem. A estação que saiu do meio devido a ter perdido o direito de acesso durante o processo de arbitragem, passa a ser uma estação receptora que no final da mensagem deverá, tal como todas as outras, confirmar que a mensagem lhe foi enviada sem qualquer erro. Esta confirmação é feita utilizando o campo de confirmação existente no final da transmissão de uma mensagem. Caso uma estação detecte um erro numa mensagem sinaliza o facto transmitindo um trama de erro.

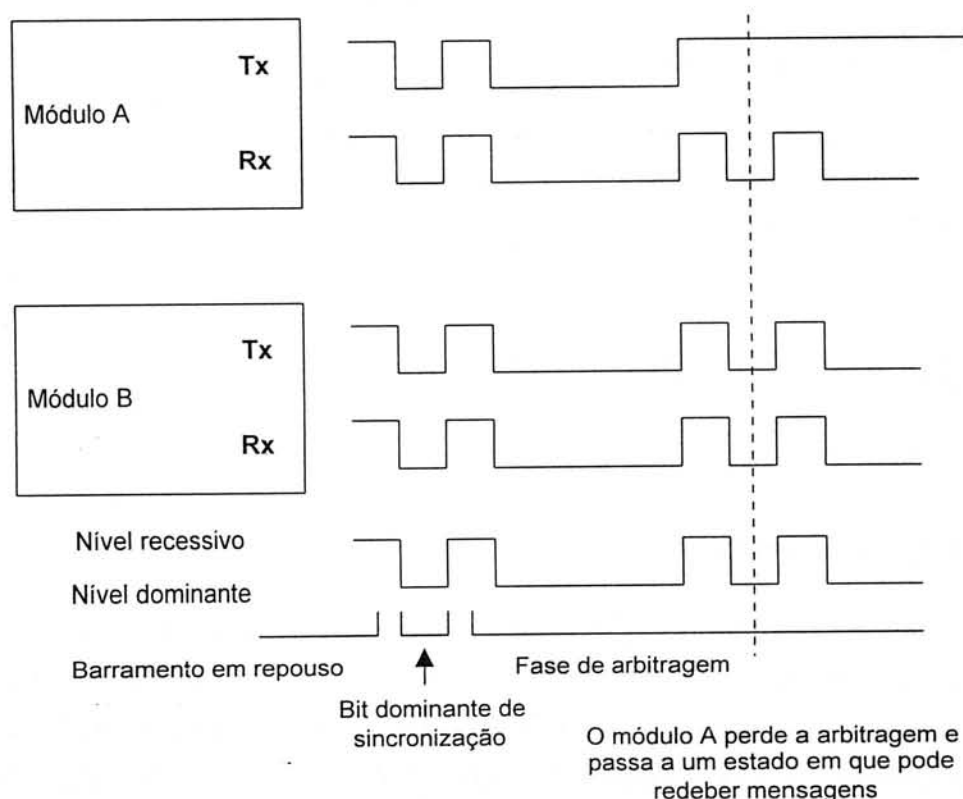


Figura 10 - Método de acesso ao meio.

Uma trama CAN 2.0A e CAN 1.0 (11 bits) permite a existência de 2036 objectos ou mensagens, uma trama do tipo estendido, CAN 2.0B (29 bits) permite a existência de 512 milhões ($2^{29}-16$) de objectos mensagem. Este funcionamento do CAN leva a que as mensagens trocadas sejam vistas como objectos com uma determinada prioridade independentes da estação que os produz. Estes objectos uma vez produzidos podem ser consumidos por uma qualquer estação ligada ao barramento. As estações consumidoras podem utilizar filtros de aceitação de mensagens. Este filtros podem ser configurados de modo a interromper o CPU apenas quando chegam mensagens de interesse para essa estação. É de realçar que o número de filtros de aceitação é limitado e varia com a

implementação (14 tanto C167 [Siemens, 93] como no 82527 [Intel527, 93] e 2 para o MCAN Module [MCAN, 96]) .

O CAN utiliza técnicas de bit *stuffing* de modo a impedir a transmissão de mais do que 5 bits consecutivos idênticos. Esta técnica permite que existam flancos pelo menos de 6 em 6 bits. Estes flancos são usados pelas estações para ressincronização pelos receptores da mensagem.

3.2.3 Tipos de tramas

O CAN disponibiliza ao utilizador quatro tipos de tramas:

- Trama de dados,
- Trama remotas,
- Trama de erros,
- Trama de excesso (*overload*).

3.2.3.1 Trama de Dados

Uma trama de dados destina-se a transferir dados. Esta trama tem o formato da figura 11. Uma trama de dados permite a transmissão de 0 a 8 bytes no seu campo de dados. O comprimento dos dados (DLC) está contido nos últimos 4 bits do campo de controlo, os bits r1 e r2 foram reservados para utilização posterior, nomeadamente, para indicar que a trama é do tipo estendido (29 bits). O bit de RTR é sempre um bit dominante de modo a indicar que a trama é de dados.

O campo de CRC é calculado de acordo com um polinómio específico e assegura uma probabilidade de não detecção de um erro numa trama de 10^{-13} .

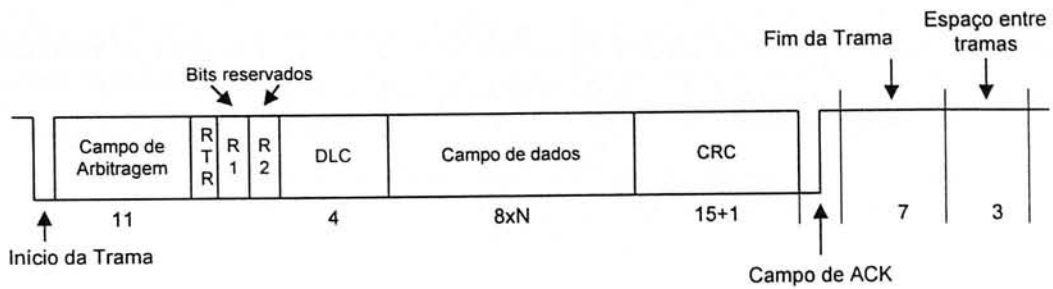


Figura 11 - Formato da trama de dados com identificador de 11 bits.

3.2.3.2 Trama Remota

Uma trama remota permite de forma eficiente e rápida, que um cliente possa pedir a um outro nó configurado como *remote receiver*, o envio de uma trama de dados com o mesmo identificador da trama remota.

O bit de RTR deverá ser recessivo, o DLC da trama remota deverá ser igual ao da trama de dados que vai pedir.

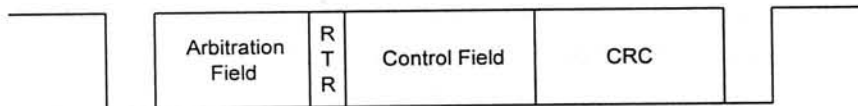


Figura 12 - Trama remota

Após a recepção da trama remota, o servidor vai enviar a resposta de forma automática sem interromper a CPU. A resposta é guardada pela aplicação no *slot* de comunicação configurado como *remote receiver*.

3.2.3.3 Trama de Erro

Uma estação ao detectar um erro envia uma trama constituída por seis bits consecutivos o que viola a norma *bit stuffing* e leva a que as outras estações transmitam também uma trama de erro. Deste modo, pode existir sobreposição das tramas de erro transmitidas por todas as estações. O tipo dos bits que são transmitidos pela estação dependem do estado em que esta se encontra. Uma estação que se encontre no estado *error passive*⁵ transmite uma trama de erro enviando 6 bits recessivos, o que permite a detecção de um erro

⁵ Definido de seguida

apenas se essa estação tiver direito de acesso ao meio. Caso a estação se encontre no estado *error active* a trama de erro é constituída por 6 bits dominantes.

O delimitador da trama de erro é constituído por 8 bits recessivos.

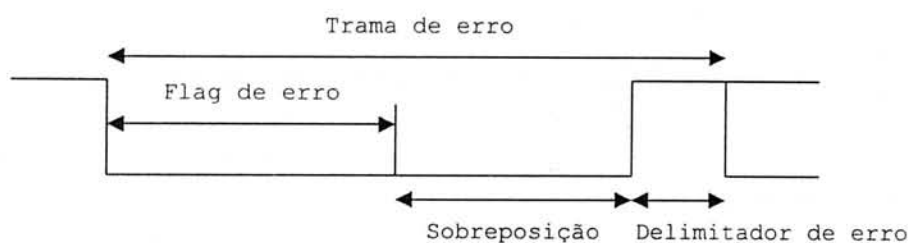


Figura 13 - Trama de erro.

Uma trama deste tipo só é transmitida quando ocorre um erro na transmissão ou na recepção de uma mensagem. Quando existe um erro na mensagem, a estação que a tentou transmitir volta a tentar a sua retransmissão, competindo pelo acesso ao meio com as outras estações. São efectuadas retransmissões até que a estação passe ao estado *bus off*, isto é, são efectuadas no máximo 32 tentativas de retransmissão.

3.2.3.4 Trama de Sobrecarga

A trama de sobrecarga é igual à trama de erro e é transmitida em três situações:

- 1- quando é detectado um bit dominante no campo de intervalo entre duas tramas;
- 2- quando um receptor não está preparado para aceitar uma mensagem e precisa de mais algum tempo para se preparar;
- 3- quando é detectado um bit de dominante no oitavo e último bit do delimitador de erro.

3.2.4 Detecção e Controlo de Erros

A norma CAN disponibiliza alguns mecanismos de controlo e gestão de erros de modo a impedir que um dispositivo a funcionar mal ou avariado provoque erros no barramento.

Em cada nó CAN existem dois contadores: o REC (*Receiver Error Counter*) de 7 bits que vai guardando os erros devidos a recepções erradas, e o TEC (*Transmit Error Counter*) que acumula os erros devidos às transmissões de mensagens por esse nó.

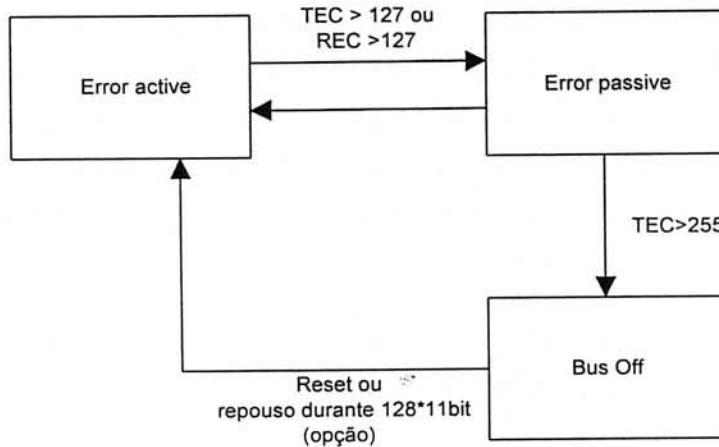


Figura 14 - Estados de erro de um dispositivo CAN.

Ambos os contadores são incrementados quando ocorre um dos seguintes erros:

- Erro de bit - o bit transmitido é diferente do recebido, excepto, quando na fase de arbitragem, uma estação envia um bit recessivo que é detectado como um bit dominante colocado no barramento por outra estação;
- Erro de *stuff* - falta o bit de *stuff*;
- Erro na trama - um bit que deveria ter um valor pré-especificado não tem o valor correcto;
- Erro no CRC - erro no valor de um ou mais bits da mensagem;
- Erro de confirmação - o nó que acabou de transmitir não recebe qualquer confirmação dos outros nós.

Quando um nó, que se encontra no estado **error active**, detecta um erro e o sinaliza através da transmissão de uma trama de erro constituída por seis bits dominantes, os outros nós irão detectar uma violação das regras de *bit stuffing* o que os leva a descartar os dados recolhidos até ao momento, e a transmitir também uma trama de erro.

Os contadores TEC e REC são incrementados de 1 ou de 8 conforme a gravidade do erro detectado e são decrementados de 1 sempre que se verifique uma transmissão ou uma recepção sem problemas.

Se um nó detectar que o valor de um dos seus contadores excede o valor de 127 (TEC e REC > 127), este passa para o estado *error passive*. Neste estado, o nó passa a assumir que um erro no barramento pode muito, provavelmente, ser causado por ele, por isso, sempre que detecta um erro emite uma trama de erro constituída por seis bits recessivos, o que implica que os outros nós apenas detectem uma trama de erro transmitida por este nó, somente, quando está a transmitir.

Um nó passa ao estado *bus off* quando o número de erros de transmissão é igual a 255. Neste estado, um nó está completamente desligado do barramento até que se efectue uma reinicialização do controlador CAN do nó.

3.2.5 Filtros de mensagens

Os filtros de mensagens permitem separar, individualmente, em cada nó, as mensagens que lhe interessam das que não lhe interessam. Deste modo, o processador não tem que estar sempre a ser interrompido para processar mensagens.

Existem dois tipos de filtros:

- Fixo: obrigam a que os bits do identificador sejam todos iguais aos bits da máscara.
- de Máscara: permite definir quais os bits que interessa testar e quais os que não tem interesse testar.

3.2.6 Implementações

Existem várias implementações da norma CAN. As várias versões variam de acordo com a estratégia da empresa, dependem do mercado alvo (por exemplo a indústria automóvel), do preço e da possibilidade de integração com outros microcontroladores da mesma marca.

Uma comparação resumida dos vários integrados e das suas principais diferenças pode ser vista na tabela 1. Esta comparação não pretendeu ser totalmente completa dado que têm aparecido, desaparecido e são anunciados novos integrados com bastante frequência.

Fornecedor	Nome	Protocolo	Grau de integração	Tipo e interface de μC	Caixa Correio (Tipo e N°)	Vários
INTEL	82527	CAN 2.0A/B	nenhum	série, 8/16 bit multiplex/non-multiplex	FullCAN, 14 BasicCAN, 1	4 filtros 2 portos E/S de 8 bitS
INTEL	87C196CA	CAN 2.0A/B	inclui μC	16-bit μC 32K ROM 1.25K RAM	FullCAN, 14 BasicCAN, 1	6 canais E/S de alta velocidade 6 entradas A/D, SSIO/SIO
INTEL	87C196CB	CAN 2.0A/B	inclui μC	16-bit μC 32K ROM 2K RAM	FullCAN, 14 BasicCAN, 1	10 canais E/S de alta velocidade 8 entradas A/D, SSIO/SIO
INTERMETAL	65C02CAN	CAN 2.0A/B			FullCAN	
HITACHI	HCAN	CAN 2.0A/B	nenhum	8/16 bit multiplex/non-multiplex	FullCAN	
INTER-METALL	Custom Dashboard Controller	CAN 2.0A/B	inclui μC	65C02 48K ROM 1.5K RAM	FullCAN, 15 BasicCAN, 14	
MOTOROLA	68HC05X4	CAN 1.2	inclui μC	8-bit μC 4K ROM 176Bytes RAM	BasicCAN	16 portos de E/S temporizador 16 bit, 11/C, IOC
MOTOROLA	68HC705X4	CAN 1.2	inclui μC	8-bit μC 4K EPROM 176Bytes RAM	BasicCAN	16 portos de E/S temporizador 16 bit, 11/C, IOC
MOTOROLA	68HC05X16	CAN 1.2	inclui μC	8-bit μC 16K ROM 256 Bytes RAM	BasicCAN	256 Bytes EPROM contador 16 bit
MOTOROLA	68HC05X32	CAN 1.2	inclui μC	8-bit μC 32K ROM 528 Bytes RAM	BasicCAN	
MOTOROLA	68HC08Ax	CAN 2.0A/B	inclui μC	8-bit μC		
NSC	MM 57C360	CAN 2.0 A 2.0 B Passivo	SLIO	Não tem μC Não tem interface	Registo I/O de 14 Bit	
NSC	MM 57C362	CAN 2.0 A 2.0 B Passivo	SLIO	Não tem μC Não tem interface	Registo I/O de 6 Bit	
NSC	COP884BC	CAN 2.0 A e 2.0 B Passivo	inclui μC	8-bit μC 2K ROM 64 Bytes RAM	BasicCAN, 1	
PHILIPS	82C200 ⁶	CAN 2.0 A	nenhum	8-bit multiplexer Intel/Motorola	BasicCAN, 1	apenas controlador
PHILIPS	P8XC592 ⁷	CAN 2.0 A	inclui μC	8-bit μC 16K (E)PROM 2X256 RAM	BasicCAN	5X8-bit portos de E/S DMA para msg. CAN
PHILIPS	P8XC598	CAN 2.0 A	inclui μC	8-bit μC	BasicCAN	5X8-bit portos

⁶ Esta planeado para 1997 a substituição por um novo controlador de referência SJA1000

⁷ A ser substituído no futuro por novos processadores com arquitectura XA. Não recomendado para novos desenvolvimentos.

				32K (E)PROM 2X256 RAM		de E/S DMA para msg. CAN
PHILIPS	P82C150 ⁸	CAN 2.0 A e 2.0 B Passivo	SLIO	Não tem µC Não tem interface	Registo I/O de 2x8 Bit	pinos E/S configuráveis: 16 digitais 6 analógicas + 10 bit resolução
SIEMENS	81C91	CAN 2.0A e CAN 2.0B passivo	nenhum	8-bit multiplex ou interface série	FullCAN, 15 BasicCAN, 1	
SIEMENS	81C90	CAN 2.0A e CAN 2.0B passivo	nenhum	8-bit multiplex ou interface série	FullCAN, 15 BasicCAN, 1	
SIEMENS	C167C	CAN 2.0A/B	inclui µC	16-bit µC 8K ROM 2-4k RAM	FullCAN, 15 BasicCAN, 1	ID de mensagem programável
SIEMENS	C167CR-16F	CAN 2.0A/B	inclui µC	16-bit µC 8K ROM 2-4k RAM	FullCAN, 15 BasicCAN, 1	ver C167C
SGS	ST10F167 ⁹	CAN 2.0B	inclui µC	16-bit µC	FullCAN BasicCAN	compatível com o Siemens C167C
TEMIC	29C464	CAN 2.0B	inclui µC	8-bit µC 16/32K ROM 512 RAM	FullCAN, 14	
TEMIC	29C461	CAN 2.0B	nenhum	SPI ou 8/16 bit multiplex./não multiplexado	FullCAN, 14	
TEXAS Instruments	TI CAN- Module	CAN 2.0 A e 2.0 B	IC - CELL	8/16 bit não multiplexado	FullCAN, 16	
TEXAS Instruments	TMS 370E08D55	CAN 2.0 A/B	inclui µC	8-bit µC 4K ROM 176Bytes RAM	FullCAN, 16	Watchdog RTI, 2 temporizadores , PWM, 24 bits E/S, I ² C, SCI, 8bit A/D

Tabela 1- Circuitos integrados com CAN.

Na tabela estão referidos integrados de CAN do tipo SLIO. Estes integrados não possuem inteligência nem um µC programável incorporado; permitem a execução de operações do tipo leitura e escrita de portas digitais e analógicas e geração de PWM. Estes integrados estão a ser retirados do mercado devido ao seu principal cliente, a indústria automóvel, ter rejeitado a ideia uma vez que não é possível integrar funções de diagnóstico.

⁸ Será retirado do mercado.

⁹ Apenas disponível a utilizadores seleccionados pela SGS.

3.3 Comparação

As normas ISO que definem o CAN apenas normalizam a camada física e de ligação de dados, por isso, foram surgindo varias normas para a camada de aplicação, com conceitos de implementação também muito variados.

O DeviceNet [**DeviceNet Vol I, 96**], [**DeviceNet Vol II, 96**] foi uma norma definida pela Allen Bradley para a ligação de sensores, actuadores e outros dispositivos a autómatos programáveis. Foi a primeira a surgir e por isso é neste momento a norma com maior implantação no mercado da automação industrial, especialmente nos Estados Unidos. Esta norma é aberta e por isso formou-se um organização de fabricantes e vendedores chamada ODVA (*Open Devicenet Vendors Association*), que tem como funções gerir a norma e licenciar fabricantes.

O SDS (*Smart Distributed System*) é também uma norma destinada, especialmente, ao mercado da automação industrial e define perfis para a ligação de dispositivos simples como: fins de curso, dispositivos de I/O, actuadores, etc. Esta norma foi definida pela Honeywell [**Crovella, 94**], [**SDSphy, 94**], [**SDSprod, 94**], [**SDSdesign, 94**], [**SDSapli, 95**], [**SDSconf, 95**] e encontra-se, neste momento, em fase de implantação no mercado. É também uma camada de aplicação aberta, isto é, pode ser implementada por qualquer fabricante sem custos.

A CiA, (*CAN in Automation*) é uma organização com o objectivo de promover e divulgar o CAN. Esta organização “herdou” a norma CAL [**CAL, 96**], *CAN Application Layer*, da Philips sendo, em 1996, adoptado como norma provisória, o perfil de aplicação para aplicações industriais o **CAN-Open**, que foi desenvolvido no âmbito do projecto Europeu ASPIC integrando várias instituições comunitárias. Em 1996 a CiA alargou-se e passou a suportar outras camadas de aplicação para o CAN como o DeviceNet, o SDS e o CANKingdom.

A firma Sueca Kvaser especializada em controlo, especificou o CAN Kingdom [**CANKing, 94**]. Esta camada de aplicação tem os serviços definidos de forma muito peculiar e original, de modo a que o protocolo de comunicações possa ser entendido por todos os intervenientes na especificação e construção de um sistema. É especialmente destinada à utilização em máquinas e sistemas embebidos.

As duas normas abertas definidas para utilização em automóveis são o J1939, definido pela SAE (*Society of Automobile Engineers*) e o OSEK definido por um conjunto de construtores Europeus no âmbito de um projecto financiado pela União Europeia.

Outras normas foram desenvolvidas para aplicações mais específicas, das quais são exemplo, as normas:

M3S - destinada a interligar dispositivos utilizados por pessoas deficientes como, por exemplo, cadeiras de rodas [Nelisse, 96].

MCS-5 - usada para interligar sensores e actuadores em navios. Define um conjunto de funcionalidades que permite a interligação dos vários dispositivos entre si e às camadas superiores da hierarquia de controlo.

VOLCANO - destinada a ser usada em automóveis.

WESYCAN (Weaving Systems CAN) - Destinado a ser usado em máquinas têxteis. É também uma norma aberta controlado pela empresa Sulzer Ruti Limited.

Caract.inorma	CAL/CANOpen	CANKingdom	DeviceNet	SDS
Baudrate	1Mb/25m 800kb/50 500kb/100m 250kb/250m 125kb/500m 50kb/1000m 20kb/2500m 10kb/5000m	125Kb durante a inicialização Qualquer até 1Mb/s	125Kb/500m 250Kb/250m 500Kb/100m	1Mb/22.8m 500Kb/91.4m, 250Kb/182.8m125K b/475m
Conectores	Tipo D Aparafusados Mini Multipolo	Não definidos	Ligação directa Aparafusados Mini Micro	Ligação directa Aparafusados Mini Micro Tipo D
Norma CAN	2.0A e 2.0B	2.0A e 2.0B	2.0A	2.0A
Uso do identificador	Apenas para identificação de mensagens	Flexível dependente da aplicação e configuração do nó	Dividido em 4 grupos de endereçamento	Endereço do nó + direcção Tx ou Rx + tipo de mensagem
Prioridades	8 níveis	A distribuição dos CAN-IDs é da responsabilidade do utilizador	As mensagens são divididas em 4 grupos ¹⁰	depende do endereço do nó
Uso do campo de dados	Apenas para a transmissão de dados que podem ser comprimidos	Permite o envio de dados comprimidos, cuja descompactação pode ser feita utilizando forms	Variável de acordo com o grupo escolhido para a mensagem	tipo de mensagem + ID do objecto + ID do atributo!
Alimentação através da rede	Sim 24V corrente max não definida	não está definida a interface física	sim ¹¹ 24V max 8A	sim ¹²

¹⁰ De acordo com o grupo atribuído

¹¹ Alimentação Positiva e negativa

¹² Apenas positiva

Perfis de dispositivos definidos	sim	não	sim	sim
Endereçamento dos nós	1-255	(0)1-255	0-63	0-125
Armazenamento do endereço	Dip Switch / Nvram	Dip Switch / Nvram/ Nvram no conector ¹³	Dip Switch / Nvram	Nvram no conector
Mercado alvo	CAL - equipamentos com redes CAN embebidas CANOpen - automação industrial	Máquinas industriais	Automação industrial	Automação industrial
Número de Fabricantes	Elevado, especialmente pequenas empresas Europeias	Reduzido, especialmente empresas escandinavas	Elevado, apresenta-se com um dos líderes do mercado nesta área, em particular nos EUA	Elevado, Em fase de implantação no mercado

Tabela 2 - Quadro comparativo entre as várias normas para o CAN.

Claro que um critério muito objectivo de comparação das redes seria o preço de cada sensor ou actuador, no entanto, esta comparação é extremamente difícil de fazer devido ao elevado número de nós de fabricantes diferentes e também às diferentes funcionalidades exigidas por cada norma. É apenas possível agrupar alguns nós em grupos com funcionalidades idênticas, podendo a comparação apresentar resultados errados e pouco credíveis. Apesar das restrições pode realizar-se uma pequena comparação qualitativa, relativa aos preços dos nós das várias normas, que se apresenta de seguida.

Os nós DeviceNet e SDS apresentam preços bastante semelhantes, dado que baseiam-se em tecnologias muito semelhantes e concorrem geralmente nos mesmos mercados, em termos geográficos e de aplicações.

Os dispositivos CAL e CANOpen podem apresentar preços ligeiramente inferiores em relação aos dispositivos DeviceNet e SDS, devido a esta norma apresentar maior flexibilidade em relação às características construtivas.

O CANKingdom tem ainda uma base instalada muito pequena e as quantidades de produção de um nó CANKingdom são também bastante mais baixas já que se destinam a aplicações em máquinas e robots muito diferentes umas das outras.

¹³ Funcionalidade apenas aconselhada

A escolha por uma destas redes depende essencialmente da aplicação e principalmente da quantidade e qualidade dos dispositivos oferecidos pelos fornecedores locais deste tipo de equipamentos.

O CAN foi a rede escolhida para implementação devido a:

- maior flexibilidade: o CAN define um conjunto de serviços vasto que podem ser implementados de acordo com as necessidades da aplicação.
- maior conjunto de áreas de aplicação: a CiA tem nos últimos meses constituído vários grupos de trabalho para a definição de normas baseadas no CAN para: sistemas hidráulicos, empilhadores, navios, etc.
- possibilidade implementação do CANOpen: sendo o CANOpen uma norma baseada no CAN a sua implementação fica bastante facilitada. Como vantagem adicional do uso desta norma o utilizador fica à sua disposição com um conjunto de dispositivos normalizados de baixo custo.
- funcionalidades alargadas: as funcionalidades oferecidas pelo CAN, nomeadamente o DBT, permitem a sua utilização não apenas como uma rede de sensores e actuadores mas também como uma rede de campo algumas funcionalidades equivalentes ao Profibus e ao FIP.
- grande conjunto de fornecedores: o CAN é utilizado por um grande número de pequenas empresas de automação especialmente na Europa.
- disponibilidade de informação

Por este conjunto de razões e especialmente pela sua flexibilidade e pelo conjunto de empresas de suporte, o CAN foi escolhido para implementação nesta tese.

4. O CAN em Sistemas Distribuídos Tempo Real

A flexibilidade do CAN e o número de camadas de aplicação existentes, destinadas a vários tipos diferentes de sistemas, nomeadamente, instrumentação em navios, sistemas de automação industrial, automóveis, obrigam a um estudo profundo e detalhado das suas características e capacidades.

A adequabilidade da utilização de uma rede de campo tipo CAN em ambientes tão distintos, depende das possibilidades que esta tem para responder às exigências de cada aplicação.

Este capítulo irá caracterizar de uma forma breve os principais conceitos subjacentes aos sistemas tempo real, para de seguida descrever e caracterizar o comportamento de uma rede de campo do tipo CAN e respectiva camada de aplicação, focando especialmente as suas características tempo real.

4.1 Sistemas Tempo- Real

[Bennet, 94] define um sistema tempo real como um sistema que tem que produzir respostas correctas dentro de um tempo definido, ou seja um sistema que a cada tarefa

tem associado um limite (*deadline*) no tempo para ser concluída. Se a resposta for produzida para além da sua *deadline* o resultado pode estar errado uma vez que o estado actual do sistema pode ser muito diferente do seu estado quando foram adquiridos os valores para efectuar o processamento da resposta. Também é de notar que o sistema pode ter o seu estado actual muito alterado devido a não ter produzido a resposta necessária a tempo.

Por exemplo, o controlador de um sistema de enchimento de um tanque ao detectar que o nível de um líquido num tanque atingiu o limite de segurança, deve abrir uma válvula de segurança antes que o tanque transborde. Desde a detecção do máximo até que a válvula seja actuada pode, por exemplo, decorrer cerca de 1 segundo, se o tanque for de elevadas dimensões. No caso de um sistema de controlo de uma máquina os tempos de resposta a um evento podem ser da ordem dos μs . Em [Hitex, 94] é descrito um sistema de injeção de combustível de um carro de formula 1, nesta aplicação o sistema de controlo deve garantir que com o motor a uma rotação de 18000rpm abertura e fecho das válvulas do injectores se faça com um tempo de intervalo de $555\mu\text{s}$.

Como pode ver-se por estes três exemplos o mais importante num sistema tempo real é saber se o controlador responde às acções que lhe são pedidas dentro de um tempo previsível e antes que seja atingido o limite para esse processamento (*deadline*).

Concluindo, um sistema pode ser classificado como de tempo real se consegue cumprir, na maior parte das situações, os requisitos temporais impostos pelas várias actividades do sistema.

4.2 Classificação das tarefas de um sistema de controlo

4.2.1 Relativa aos instantes de activação

- **Tarefas Cíclicas ou Periódicas**

O tempo, intervalo entre activações sucessivas de tarefas deste tipo é constante.

Por exemplo, [Bennet, 94] refere que se o fenómeno exterior a medir tiver um comportamento bem conhecido e cuja variação seja também conhecida no tempo, então esse fenómeno pode ser amostrado em intervalos regulares. A aquisição para construção da imagem do sistema, pode ser despoletada através de uma interrupção cíclica do relógio do sistema.

- **Tarefas Aperiódicas ou baseadas em eventos**

Estas tarefas são geralmente activadas por estímulos externos imprevisíveis, designados por eventos. Um evento ocorre em intervalos de tempo não definidos e pode activar uma tarefa, a estas funções activadas por eventos chamam-se tarefas aperiódicas.

Um evento pode ser por exemplo o ligar da chave de ignição de um carro que vai activar uma tarefa aperiódica que controla o arranque do carro. Os eventos servem também para assinalar situações de falha ou alarmes.

Em muitas situações é necessário responder a um evento dentro de um intervalo de tempo determinado. Por exemplo, se um sensor detectar o enchimento de um tanque acima do limite máximo, desencadeia um evento no controlador que vai abrir uma válvula de segurança. O tempo que o controlador leva até emitir a ordem para abrir a válvula de segurança poderá ser da ordem dos décimos de segundo. Já no caso de um disjuntor electrónico, o tempo que este leva a abrir desde a detecção do evento (curto circuito na rede) até que o disjuntor abra, deve andar na casa das dezenas de μs . Assim o tempo que vai deste a detecção da falha e sinalização do respectivo evento varia de acordo com o sistema que está a ser controlado.

- **Tarefas Interactivas**

São tarefas destinadas especialmente à comunicação com seres humanos, apresentam restrições do tipo estatístico. Um exemplo de uma tarefa deste tipo pode ser por exemplo uma caixa automática (tipo multibanco), em que é necessário que a máquina responda ao pedido dentro de um tempo útil, alguns segundos. Estes sistemas têm restrições do tipo a tarefa deve ser completada em média em menos de x segundos.

4.2.2 Relativa à Criticalidade dos requisitos temporais

- **Tarefas não Críticas (*Soft Real Time*)**

Neste tipo de tarefas, o não cumprimento das restrições temporais (*deadlines*) associadas à tarefa acarreta uma pequena penalização para o funcionamento do sistema que se vai tornando maior com o tempo de atraso em relação à *deadline*. O atraso no cumprimento de uma restrição pode não levar o sistema a falhar completamente.

Por exemplo, uma central telefónica tem como especificação que uma em cada dez mil chamadas pode ser encaminhada erradamente. A tarefa que controla o encaminhamento

das chamadas pode falhar algumas vezes, devido ao excesso de carga do sistema, sem que exista demasiada penalização para o utente e para a central telefónica.

- **Tarefas Críticas (Hard Real Time)**

Para este tipo de tarefas o não cumprimento de uma restrição temporal leva à perda do sistema ou a uma penalização muito elevada. Como exemplo de um sistema *hard real time* temos o ABS de um carro. Neste caso, se o valor relativo ao movimento das rodas chegar atrasado ao controlador do ABS, então o accionamento pode ser feito de forma errada o que poderá ter consequências graves para o carro e seus ocupantes.

4.3 Sistemas e Controlo Distribuído

Devido à diminuição dos custos do hardware e ao aparecimento das redes de comunicações, passou a ser possível distribuir funções pelos constituintes do sistema. Um sistema distribuído é constituído por vários nós cooperantes que comunicam através de um rede.

[Milenkovic, 92] define um sistema distribuído como uma colecção de sistemas com capacidade de processamento autónoma capazes de comunicar e cooperar através do seu hardware e software.

Cada nó é um sistema independente constituído por um processador e periféricos. Observe-se os dois exemplos seguintes:

Uma linha de montagem de automóveis é constituída por centenas de robots, cada um com o seu controlador próprio, obedecendo às ordens de um controlador central que, por exemplo, indica a um dos robots para colocar o vidro traseiro e é o controlador do robot que irá fazer todos os movimentos para cumprir a ordem. Este tipo de controlo é chamado de hierárquico distribuído.

Nessa mesma linha de montagem existem vários sensores, cada um dos quais com o seu próprio sistema de controlo, neste caso a distribuição da inteligência (capacidade de processamento) pelos diversos constituintes do sistema permite que um sensor seja capaz de executar várias funções internas como, por exemplo: detecção e execução de autocalibração, detecção de curto circuitos na suas entradas ou saídas, armazenamento dos seus parâmetros em memória não volátil, etc.

Os sistemas referidos no parágrafo anterior efectuam o controlo dito centralizado. O controlo de sistema também poderá ser feito de uma forma heterárquica, tal como foi especificado por [Carvalho, 95] para um controlador de célula flexível de fabrico. Este tipo de controlo não necessita de um controlador central, as tarefas são distribuídas pelos vários controladores autónomos do sistema, o controlador central é retirado e as suas funções passam a ser desempenhadas pelos outros controladores.

Outra solução adopta uma solução mista, em que no mesmo sistema se adoptam métodos de controlo hierárquico e heterárquico, chama-se a esta estrutura de controlo - hierárquico modificado. São os dois últimos tipos de controlo que mais fazem uso da possibilidade de distribuição de funções.

Baseado em [Milenkovic, 92], [Bennet, 94] e [Tanenbaum, 92] passa-se a referir as principais características de um sistema distribuído:

- O processamento é dividido pelos vários nós intervenientes no processo.
- A comunicação entre os vários nós é feita utilizando uma rede em que cada nó pode comunicar com qualquer outro.
- Com a baixa de preços do hardware e software o preço de um sistema distribuído é idêntico ao de um sistema centralizado equivalente¹⁴ ou até mesmo mais baixo.
- Estes sistemas permitem com facilidade e baixo custo a inclusão de novos elementos.
- A falha de um nó geralmente não resulta na falha total do sistema, um nó que falhe poderá ter as suas funções substituídas por outro. Caso não seja possível o sistema poderá desligar-se total ou parcialmente de forma coordenada e sem acidentes.
- A falha da rede acarreta a falha do sistema, o que em sistemas críticos obriga à utilização de redes redundantes, que dupliquem a rede primária e que só entram em funcionamento no caso da falha da rede principal.
- A administração do sistema torna-se mais complexa.

¹⁴ Pode não existir um sistema centralizado equivalente

- A segurança do sistema torna-se um problema uma vez que a informação circula através de uma rede num espaço geográfico que pode ir de uma pequena máquina até continentes.
- Tal como em sistemas não distribuídos é necessário prever mecanismos que garantam uma eficaz partilha de recursos.

Este conjunto de características obriga à utilização de métodos que garantam a coerência da informação em todos os nós, isto é, que a imagem local do sistema seja idêntica à imagem real, ou que pelo menos os erros dessa imagem local para a real possam ser conhecidos.

A utilização de sistemas distribuídos em sistemas com exigências de tempo real obriga a que:

- A comunicação entre nós seja determinística. O sistema de comunicações deve garantir que o atraso devido às comunicações não implica a falha de um *deadline* do sistema.
- Caso o sistema tenha principalmente tarefas síncronas pode ser necessário providenciar mecanismo de sincronização entre nós, a sincronização entre nós poderá ser obtida através de vários mecanismos:
 1. Sincronização de relógio (com um nó mestre): o nó que contém o relógio de referência do sistema envia mensagens com informação sobre o tempo aos outros nós escravos que por sua vez irão ajustar os seus relógios de modo a estarem em sincronismo com o relógio principal [Levi et al, 90] e [Gergeleit et al, 94].
 2. Sincronização por mensagens: este tipo de sincronização implica que um nó mestre envie uma mensagem de sincronização a qual irá implicar algum tipo de resposta pelos outros nós da rede [CANOpen, 96].
 3. Sincronização distribuída: Não existe um relógio central cada nó sincroniza-se em função dos outros [Levi et al, 90].
 4. *Token*: A passagem do *token* entre estações permite a sincronização com os outros nós [Levi et al, 90].

- No caso de sistemas tempo real críticos é necessário assegurar um nível de tolerância a falhas. É necessário prever falhas de:
 1. um nó: caso haja uma avaria num nó o controlador deve ter a possibilidade de detectar essa falha de modo a que possa actuar de acordo com a sua gravidade. Podem verificar-se dois tipos de falhas de um nó falhas silenciosas (falhas em que o nó deixa de transmitir) e falhas não silenciosas (em que o nó passa a transmitir lixo).
 2. da rede: caso a rede fique inoperacional total ou parcialmente.
- Caso o sistema seja do tipo assíncrono, comandado por eventos, é necessário assegurar que todos os eventos podem ser correctamente transmitidos de acordo com as restrições temporais desse evento. Normalmente, neste caso, utilizam-se redes com a possibilidade de utilização de prioridades para o escalonamento das mensagens.

4.4 Elementos para a modelização de uma rede CAN

Geralmente uma rede CAN suporta aplicações distribuídas com características de tempo real, para estas aplicações é necessário conhecer os tempos de execução das tarefas distribuídas pelos vários nós e os atrasos decorrentes do sistema de comunicações. O conhecimento destes tempos permite estabelecer, *à priori*, uma estratégia de controlo adequada ao sistema em estudo. Para isso é necessário conhecer os tempos de:

- Execução das tarefas¹⁵
- Construção da mensagem - tempo necessário para a aplicação gerar e enviar uma mensagem para a fila de mensagens a transmitir
- Limite para a execução das tarefas(*deadline*) - limite de tempo para a execução de uma tarefa.
- Transmissão das mensagens
- Atraso na recepção da mensagem - tempo para processar uma mensagem antes que ela chegue à tarefa de destino

¹⁵ Usualmente o máximo tempo de execução da tarefa

- Tempo de acesso ao meio - tempo necessário para que uma mensagem, à espera na fila de transmissão, possa ter acesso ao meio.

A facilidade de medição e cálculo destes tempos depende em muito do sistema. O tempo de execução de uma tarefa pode ser medido utilizando um qualquer relógio, é preciso apenas garantir que esta medição seja feita sem que outras tarefas entrem em concorrência com a tarefa a medir e que as condições de funcionamento são equivalentes às reais, isto é, deve-se ter atenção à arquitectura do processador, se tem memória *cache*, se utiliza *pipeline* de instruções e ao sistema operativo. Para este fim, também podem ser utilizadas ferramentas que calculem o tempo de execução em função do código da tarefa. Normalmente são utilizados os tempos máximos de execução de uma tarefa, de modo a que a análise posterior seja feita com as condições mais pessimistas. Uma abordagem sucinta deste tema pode ser encontrada em [Tindell1, 95]. Em [Forsyth1, 90] e [Forsyth2, 90] este tema é abordado com maior detalhe, explicando os princípios subjacentes a um programa de cálculo de tempos de execução de tarefas.

4.4.1 Estrutura de um nó

Cada nó poderá conter várias tarefas de aplicações que utilizam concorrentemente a tarefa gestora de transmissões. Esta tarefa encarrega-se de gerir o acesso das aplicações ao subsistema de comunicações. Para o processamento das mensagens recebidas existe uma tarefa responsável pelo atendimento das interrupções geradas pelas recepções. O controlo das transmissões é feito por outra tarefa cuja função é colocar as mensagens na fila de transmissão.

Baseado em [Tindell2, 94] em cada nó de uma rede CAN tem uma estrutura idêntica à da Figura 15.

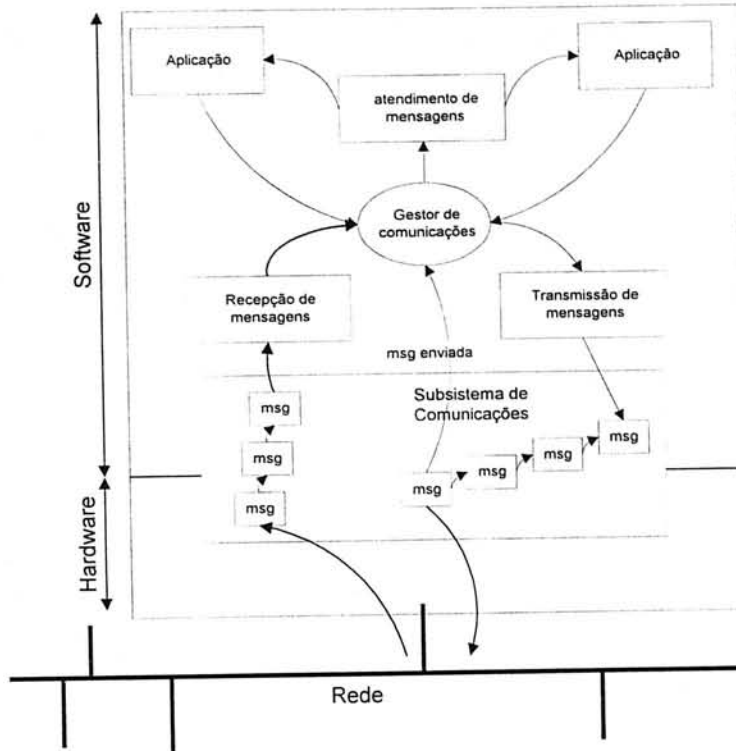


Figura 15 - Estrutura de um nó de comunicações genérico.

No caso do CAN e tal como na figura 15, o subsistema de comunicações não pode ser implementado em hardware e software devido à maioria dos controladores de CAN não implementarem filas de entrada e de saída. É a camada de ligação lógica que controla o funcionamento das filas de entrada e de saída¹⁶.

Por exemplo, o controlador CAN 82527 e o μC C167C disponibilizam apenas 14 objectos de comunicações (*slots*), cada um dos quais pode ser configurado individualmente para receber ou transmitir mensagens. É ainda disponibilizado um objecto de comunicações especial que serve apenas para a recepção de mensagens utilizando uma fila de mensagens com espaço para apenas duas mensagens.

Uma camada de aplicação genérica para o CAN pode programar o controlador para gerar uma interrupção por cada mensagem recebida. A rotina de interrupção apenas coloca essa mensagem na fila das mensagens recebidas até que uma outra tarefa a possa processar.

¹⁶ Desde que a aplicação necessite de utilizar filas de recepção e de transmissão.

A transmissão de mensagens pode ser feita de forma idêntica. A tarefa de aplicação envia as mensagens para as camadas mais baixas do protocolo que armazenam a mensagem numa fila ou, caso seja possível, a colocam num dos objectos de comunicação disponíveis. A mensagem é colocada numa fila quando todos os objectos de comunicações estão ocupados. Após cada transmissão bem sucedida de uma mensagem é gerada uma interrupção e a camada de ligação lógica coloca no objecto de comunicação disponível a primeira mensagem da fila.

O tráfego num sistema distribuído tempo real pode normalmente ser caracterizado de forma quantitativa e temporalmente segura. Isto é, antes de por o sistema em funcionamento o seu construtor consegue, na maior parte dos casos, determinar as exigências impostas pelo tráfego previsto e provar que o sistema consegue cumprir as exigências impostas pelo sistema a controlar. Para que o referido seja possível é necessário caracterizar de forma segura os tempos de todas as acções necessárias à comunicação entre os nós. Existem porém algumas excepções, as características de funcionamento do sistema tornam-se muito difíceis de prever quando o sistema se torna muito complexo ou pode estar sujeito a situações de avaria complexas.

A figura 16 especifica os vários atrasos que uma mensagem tem desde o momento em que é enviada pela aplicação até estar disponível para processamento noutro nó.



Figura 16 - Atrasos na comunicação.

Os tempos da figura 2 são caracterizados da seguinte forma:

- Preparação das mensagens: tempo necessário para que uma camada de aplicação para o CAN interprete o pedido do utilizador e construa uma mensagem.
- Espera no *buffer* antes de iniciar a transmissão: tempo que a mensagem espera no *buffer* de transmissão e no *slot* do controlador CAN até que esta seja enviada.
- Transmissão: tempo de transmissão da mensagem pelo nó emissor, já que, no caso do CAN, o tempo de propagação da mensagem pode ser desprezado devido ao seu baixo valor.

- Colocação da mensagem no *buffer*: após a recepção de uma mensagem é gerada uma interrupção e é necessário retirar a mensagem do *slot* e coloca-la no buffer de recepção.
- Tempo de espera no *buffer*: tempo que a mensagem espera no buffer de recepção até ser processada pela aplicação.

Em alguns casos os dois últimos tempos são nulos devido à camada de aplicação responder imediatamente à mensagem através de tramas remotas.

4.4.2 Tipos de tráfego

Tal como referido em 4.2 num sistema tempo real podem existir tarefas cíclicas e tarefas acíclicas (baseadas em eventos). Num sistema distribuído estes tipos de tarefas podem gerar mensagens cíclicas e mensagens acíclicas.

O tráfego cíclico é caracterizado pelo período entre transmissões sucessivas da mensagem. Existem normalmente pequenas variações no período entre transmissões devido a vários factores nomeadamente:

- Transmissão de mensagens de maior prioridade;
- Existência de uma mensagem a ocupar o meio no instante em que o controlador coloca a mensagem está preparada para competir pelo acesso ao meio;
- Variações do tempo de preparação de uma mensagem pela camada de aplicação;
- Ruído que leva à retransmissão da mensagem.

O tráfego acíclico é caracterizado por não ter um período definido entre transmissões sucessivas. No entanto, vários autores [Schneider, 96] e [Tindell3, 94] caracterizam o tráfego acíclico de acordo com a figura 17. Cada mensagem tem um tempo mínimo para o seu período T , usualmente sucedem-se períodos de activação, em que são transmitidas várias mensagens deste tipo, a períodos de inactividade durante os quais não são transmitidas mensagens deste tipo.

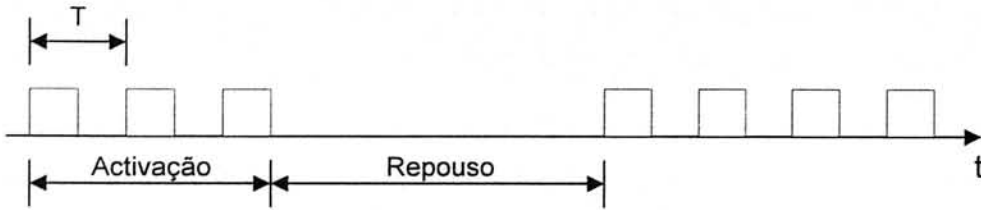


Figura 17 - Caracterização do tráfego acíclico

Um exemplo deste tipo de funcionamento dá-se quando um condutor de um automóvel pisa o pedal do travão para efectuar uma travagem. A partir da activação do pedal o sensor que lê a posição do pedal transmite sucessivamente para o sistema de travagem as novas posições do travão. A mensagem com a posição corrente do travão poderá ser enviada com um período T enquanto durar a travagem.

Devido há possibilidade de uma mensagem de alta prioridade ocupar o meio durante um tempo excessivo existe normalmente a necessidade de limitar o período entre transmissões sucessivas de eventos. É usual a existência de mecanismos nas camadas de aplicação de modo a garantir tempos mínimos entre transmissões sucessivas de uma determinada mensagem.

Um exemplo típico do referido no parágrafo anterior é um sensor óptico que se encontra numa fase de detecção incerta de um objecto por este se encontrar a vibrar. O sensor irá detectar várias vezes o objecto e enviar uma mensagem com uma frequência igual à frequência de vibração do objecto. Este é um dos caso em que se torna necessário prever um período mínimo entre transmissões sucessivas de mensagens representativas de eventos. Este funcionamento também se deve aplicar a qualquer outro tipo de mensagens como mais uma garantia para a limitação da largura de banda utilizada. Também é possível a existência de erros no software que levam à transmissão sucessiva de mensagens, este problema é usualmente referido como o “*Babbling Idiot*”. Um protocolo para a resolução deste problema foi apresentado por [Tindell4, 95]. O protocolo proposto prevê o aumento da prioridade da mensagem que estão a tentar ter acesso ao meio com o tempo e implica alterações no hardware dos controladores de CAN.

As características referidas para o tráfego acíclico permitem a sua modelização considerando o tráfego acíclico como tráfego cíclico com período T . Este tipo de modelização pode, então aplicar-se para o cálculo dos tempos de resposta de uma mensagem. É de notar que esta modelização prevê o pior caso, em que são constantemente enviadas mensagens com um período T , por isso caso se façam os cálculos para dimensionamento de um sistema utilizando este tipo de modelização é

garantido que em condições normais de funcionamento a rede de comunicações não entra em sobrecarga.

4.4.3 Tempo de transmissão das mensagens

É usual a performance de uma rede estar associada ao seu *baud rate*. Esta associação pode induzir em erro, uma vez que, a verdadeira taxa de transferência de dados é bastante mais baixa dado que a trama de comunicação transporta além dos dados um identificador e outros campos de controlo. Uma trama de dados do CAN, é constituída por: início da mensagem (1 bit), o identificador da mensagem (11+1 ou 29+1 bits), o campo de controlo(6 bits), campo de dados(0 a 16 bits), o CRC(15+1 bits) e o campo de fim da mensagem(2 bits). Devido à técnica de bit *stuffing* são acrescentados bits de sinal contrário sempre que seja enviada uma sequência de 5 bits iguais a 1, assim o tempo de transmissão de uma mensagens está limitado pelos valores apresentados na equação 1.

$$\frac{19 + 8 \times n + 25}{B} \leq t_T \leq \frac{19 + 8 \times n + 25 + 0.2 \times (19 + 8 \times n + 15)}{B} \quad (1)$$

Em que B é o *baud rate* e n o número de bytes da mensagem, o factor de 0.2 representa o número máximo de bits de *stuffing*. A Tabela 3 apresenta alguns valores de referência para o tempo máximo de transmissão de uma trama.

	25Kbit/s	50Kbit/s	100Kbit/s	250Kbit/s	500Kbit/s	1Mbit/s
0	2032,0	1016,0	508,0	203,2	101,6	50,8
1	2416,0	1208,0	604,0	241,6	120,8	60,4
2	2800,0	1400,0	700,0	280,0	140,0	70,0
3	3184,0	1592,0	796,0	318,4	159,2	79,6
4	3568,0	1784,0	892,0	356,8	178,4	89,2
5	3952,0	1976,0	988,0	395,2	197,6	98,8
6	4336,0	2168,0	1084,0	433,6	216,8	108,4
7	4720,0	2360,0	1180,0	472,0	236,0	118,0
8	5104,0	2552,0	1276,0	510,4	255,2	127,6

Tabela 3 - Tempo máximo de transmissão de uma trama CAN 2.0A(valores em μ s), número de bytes *versus* velocidade.

A utilização da largura de banda disponível para transmissão de dados (apenas utilizando o campo de dados) é, tal como em outro tipo de redes, bastante baixo, chegando no máximo aos 50% na transmissão de 8 bytes, torna-se ainda mais grave se se utilizarem tramas com identificadores de 29bits CAN 2.0B. Algumas das normas para a camada de

aplicação referidas permitem a utilização do campo de identificação para a transferência de parte dos dados, como a norma CANKingdom que permite uma definição extremamente flexível, configurada pelo construtor do sistema, do conteúdo dos vários campos constituintes de um trama de CAN.

Além das tramas normais o CAN também permite o uso de tramas de comunicação remotas. Um dos objectos de comunicação de um nó é configurado como *remote receiver* e é-lhe colocado um valor no campo de dados, que poderá ser o valor de uma variável existente nesse nó. Após a recepção de uma trama remota esse nó responde enviando uma trama de dados normal com o campo de dados preenchido com os valores lá colocados. Este processo é usado pelo CAL para a leitura de *BASIC VARIABLES*.

No tempo de transmissão deve ser incluído o espaço entre tramas que é de 3 bits, e é necessário ter em conta que existem duas tramas envolvidas na comunicação, a trama remota que faz o pedido e trama de dados de resposta. O tempo de transmissão total para o caso de uma trama remota é dado é limitado pelas equações seguintes:

$$\begin{cases} t_T \leq \frac{19+25+0.2 \times (19+15) + 3+19+8 \times n+25+0.2 \times (19+8 \times n+15)}{B} \\ t_T \geq \frac{19+25+3+19+8 \times n+25}{B} \end{cases} \quad (2)$$

A vantagem principal que advém da utilização de tramas remota consiste na possibilidade de fazer o pedido de um valor directamente ao controlador CAN sem ser necessário interromper o processador do nó para responder ao pedido.

	25Kbit/s	50Kbit/s	100Kbit/s	250Kbit/s	500Kbit/s	1Mbit/s
0	4184.0	2092.0	1046.0	418.4	209.2	104.6
1	4568.0	2284.0	1142.0	456.8	228.4	114.2
2	4952.0	2476.0	1238.0	495.2	247.6	123.8
3	5336.0	2668.0	1334.0	533.6	266.8	133.4
4	5720.0	2860.0	1430.0	572.0	286.0	143.0
5	6104.0	3052.0	1526.0	610.4	305.2	152.6
6	6488.0	3244.0	1622.0	648.8	324.4	162.2
7	6872.0	3436.0	1718.0	687.2	343.6	171.8
8	7256.0	3628.0	1814.0	725.6	362.8	181.4

Tabela 4 - Tempo máximo de transmissão de uma trama remota (valores em μs), número de bytes *versus* velocidade.

Verifica-se que os tempos de transmissão são cerca de 40% mais elevados do que no caso da transmissão de tramas de dados normais.

Como é evidente o tempo de transmissão de mensagens pode ser diminuído aumentando o *baud rate*; no entanto, este aumento causa, também alguns problemas.

4.4.3.1 Selecção do *baud rate*

A selecção do *baud rate* da rede CAN deve ser feita tendo em conta que a selecção de um valor alto permite que a rede tenha capacidade para acomodar o crescimento futuro do sistema mas tem a desvantagem de reduzir o comprimento máximo do cabo de rede, introduzindo maior susceptibilidade ao ruído e exigindo que a interface de rede e a aplicação tenham maior performance.

As duas tabelas anteriores indicam os tempos máximos de transmissão de uma trama em várias situações, estes tempos indicam os limites da rede, no entanto, também é necessário conhecer as exigências que a utilização de um elevado *baud rate* tem para o nó.

As exigências de performance para o nó são ditadas especialmente pelo tempo mínimo entre a recepção sucessiva de tramas. Este tempo mínimo é igual ao menor tempo para a transmissão de uma trama. Na tabela 5 podem ser observados os tempos mínimos de transmissão de uma trama constituída por 0 Bytes para vários *baud rates* diferentes.

Baud Rate	25Kbit/s	50Kbit/s	100Kbit/s	250Kbit/s	500Kbit/s	1Mbit/s
Tempo (μ s)	1760	880	440	176	88	44

Tabela 5 - Tempos mínimos de transmissão de mensagens com 0 Bytes de comprimento em CAN 2.0A (identificador de 11 bits).

Numa situação em que são repetidamente transmitidas mensagens o processador do nó receptor deve poder guardar cada uma das mensagens sem que nenhuma se perca, este requisito obriga o processador a ler uma mensagem da memória do controlador, verificar se lhe é destinada¹⁷, a efectuar algum tipo de pré-processamento¹⁸ e a colocá-la na sua memória num tempo menor do que o tempo mínimo até a chegada de uma outra

¹⁷ Os filtros existentes em hardware, no controlador CAN, podem não ser suficientes

¹⁸ Por exemplo, colocar um semáforo a 1

mensagem. Assim, para mensagens de transmitidas a 1Mbit/s o processamento descrito só pode demorar cerca 44 μ s. Tempo muito baixo quando comparado com as velocidades mais usuais para microcontroladores de 8 e 16 bits. Para o caso de um relógio de 40MHz estão disponíveis 1760 ciclos entre recepções sucessivas de mensagens e para um relógio de 4MHz estão disponíveis apenas 176 ciclos.

Verifica-se então que para garantir que nenhuma mensagem se perde as rotinas de recepção da mensagem devem ser bastante rápidas e tem que ser consideradas como tarefas críticas.

Por outro a selecção de um *baud rate* baixo pode levar à falha de algumas *deadlines* e a rede pode torna-se o *bottleneck* de todo o sistema.

4.4.4 Tempo de preparação de mensagens

Este tempo é essencialmente devido à camada de aplicação que se está a utilizar e por isso bastante variável com a implementação e de norma para norma. Também varia dentro de cada camada com o tipo de serviço utilizado. Observe-se o seguinte exemplo.

Uma aplicação ao evocar o serviço de leitura de uma variável (implementado pelo CAL), primeiro tem que decodificar os parâmetros da função, de seguida irá construir a mensagem de acordo com os atributos da variável e finalmente irá passar a mensagem já construída para a camada de acesso ao meio. A camada de acesso ao meio gere a fila de mensagens à espera de transmissão e coloca a mensagem num dos objectos de comunicação do controlador à espera de ganhar o acesso ao meio.

Partindo dos pressupostos enunciados o cálculo do tempo de preparação de mensagens é dependente: da aplicação, da camada de aplicação utilizada, do tipo de processador e da carga do processador caso se esteja a utilizar um núcleo multitarefa ou da possibilidade de ocorrência de interrupções no decorrer do tempo de preparação da mensagem. Portanto esta parcela não tem um tempo fixo mas varia de acordo com a aplicação a correr no nó que vai transmitir.

4.4.5 Tempo de colocação da mensagem no *buffer* de recepção e tempo de espera no *buffer*

A colocação de uma mensagem no *buffer* pode ser feita utilizando interrupções. Quando chega uma mensagem o controlador CAN gera uma interrupção que activa um rotina

cuja função será a colocação da mensagem no *buffer* de recepção. Dado que uma tarefa deste tipo pode ser bastante simples, o seu tempo de execução pode também ser determinado com facilidade.

O tempo de espera no *buffer* depende sobretudo da aplicação e da altura em que esta tem tempo disponível para retirar a mensagem do *buffer* e processá-la.

O tempo de colocação de mensagens no *buffer* define a taxa máxima a que um nó pode receber mensagens. Claro que a taxa máxima só pode ser mantida por um período de tempo que garante que o *buffer* não enche.

Para o cálculo correcto da taxa máxima de recepção de mensagem é preciso ter em conta que além do tempo para colocar uma mensagem no *buffer* é necessário adicionar o tempo de a camada de aplicação leva até processar cada uma das mensagens recebidas.

4.4.6 Fontes de variabilidade dos tempos de resposta

[Upender *et al*, 96] e [Schneider, 96] detectaram através de simulações que um dos factores que mais influencia o tempo de resposta de uma rede CAN é a serialização de mensagens. Este fenómeno ocorre quando um número elevado de estações tentam ter acesso ao meio no mesmo instante.

Este tipo de cenário tem uma probabilidade de ocorrência bastante baixo, no entanto, este tipo de situações ocorre geralmente em situações de avaria ou acidente, assim quando se dimensiona um sistema é necessário saber que estes fenómenos podem acontecer, quais as suas consequências para o sistema e como se pode resolver.

As causas para a serialização de mensagens são:

- Ruído: o ruído pode momentaneamente corromper a transmissão de dados. Devido aos erros causados existem mensagens que são novamente enviadas automaticamente pelo controlador CAN, o que atrasa a transmissão de outras mensagens de prioridade mais baixa.
- Comunicações tipo pedido de resposta com *broadcast*: esta situação implica um pedido seguido de várias respostas por parte de todas ou de um grande conjunto de estações.

aplicação referidas permitem a utilização do campo de identificação para a transferência de parte dos dados, como a norma CANKingdom que permite uma definição extremamente flexível, configurada pelo construtor do sistema, do conteúdo dos vários campos constituintes de um trama de CAN.

Além das tramas normais o CAN também permite o uso de tramas de comunicação remotas. Um dos objectos de comunicação de um nó é configurado como *remote receiver* e é-lhe colocado um valor no campo de dados, que poderá ser o valor de uma variável existente nesse nó. Após a recepção de uma trama remota esse nó responde enviando uma trama de dados normal com o campo de dados preenchido com os valores lá colocados. Este processo é usado pelo CAL para a leitura de *BASIC VARIABLES*.

No tempo de transmissão deve ser incluído o espaço entre tramas que é de 3 bits, e é necessário ter em conta que existem duas tramas envolvidas na comunicação, a trama remota que faz o pedido e trama de dados de resposta. O tempo de transmissão total para o caso de uma trama remota é dado é limitado pelas equações seguintes:

$$\begin{cases} t_T \leq \frac{19+25+0.2 \times (19+15) + 3+19+8 \times n+25+0.2 \times (19+8 \times n+15)}{B} \\ t_T \geq \frac{19+25+3+19+8 \times n+25}{B} \end{cases} \quad (2)$$

A vantagem principal que advém da utilização de tramas remota consiste na possibilidade de fazer o pedido de um valor directamente ao controlador CAN sem ser necessário interromper o processador do nó para responder ao pedido.

	25Kbit/s	50Kbit/s	100Kbit/s	250Kbit/s	500Kbit/s	1Mbit/s
0	4184.0	2092.0	1046.0	418.4	209.2	104.6
1	4568.0	2284.0	1142.0	456.8	228.4	114.2
2	4952.0	2476.0	1238.0	495.2	247.6	123.8
3	5336.0	2668.0	1334.0	533.6	266.8	133.4
4	5720.0	2860.0	1430.0	572.0	286.0	143.0
5	6104.0	3052.0	1526.0	610.4	305.2	152.6
6	6488.0	3244.0	1622.0	648.8	324.4	162.2
7	6872.0	3436.0	1718.0	687.2	343.6	171.8
8	7256.0	3628.0	1814.0	725.6	362.8	181.4

Tabela 4 - Tempo máximo de transmissão de uma trama remota (valores em μs) , número de bytes *versus* velocidade.

Verifica-se que os tempos de transmissão são cerca de 40% mais elevados do que no caso da transmissão de tramas de dados normais.

Como é evidente o tempo de transmissão de mensagens pode ser diminuído aumentando o *baud rate*; no entanto, este aumento causa, também alguns problemas.

4.4.3.1 Selecção do *baud rate*

A selecção do *baud rate* da rede CAN deve ser feita tendo em conta que a selecção de um valor alto permite que a rede tenha capacidade para acomodar o crescimento futuro do sistema mas tem a desvantagem de reduzir o comprimento máximo do cabo de rede, introduzindo maior susceptibilidade ao ruído e exigindo que a interface de rede e a aplicação tenham maior performance.

As duas tabelas anteriores indicam os tempos máximos de transmissão de uma trama em várias situações, estes tempos indicam os limites da rede, no entanto, também é necessário conhecer as exigências que a utilização de um elevado *baud rate* tem para o nó.

As exigências de performance para o nó são ditadas especialmente pelo tempo mínimo entre a recepção sucessiva de tramas. Este tempo mínimo é igual ao menor tempo para a transmissão de uma trama. Na tabela 5 podem ser observados os tempos mínimos de transmissão de uma trama constituída por 0 Bytes para vários *baud rates* diferentes.

Baud Rate	25Kbit/s	50Kbit/s	100Kbit/s	250Kbit/s	500Kbit/s	1Mbit/s
Tempo (μ s)	1760	880	440	176	88	44

Tabela 5 - Tempos mínimos de transmissão de mensagens com 0 Bytes de comprimento em CAN 2.0A (identificador de 11 bits).

Numa situação em que são repetidamente transmitidas mensagens o processador do nó receptor deve poder guardar cada uma das mensagens sem que nenhuma se perca, este requisito obriga o processador a ler uma mensagem da memória do controlador, verificar se lhe é destinada¹⁷, a efectuar algum tipo de pré-processamento¹⁸ e a colocá-la na sua memória num tempo menor do que o tempo mínimo até a chegada de uma outra

¹⁷ Os filtros existentes em hardware, no controlador CAN, podem não ser suficientes

¹⁸ Por exemplo, colocar um semáforo a 1

mensagem. Assim, para mensagens de transmitidas a 1Mbit/s o processamento descrito só pode demorar cerca 44 μ s. Tempo muito baixo quando comparado com as velocidades mais usuais para microcontroladores de 8 e 16 bits. Para o caso de um relógio de 40MHz estão disponíveis 1760 ciclos entre recepções sucessivas de mensagens e para um relógio de 4MHz estão disponíveis apenas 176 ciclos.

Verifica-se então que para garantir que nenhuma mensagem se perde as rotinas de recepção da mensagem devem ser bastante rápidas e tem que ser consideradas como tarefas críticas.

Por outro a selecção de um *baud rate* baixo pode levar à falha de algumas *deadlines* e a rede pode torna-se o *bottleneck* de todo o sistema.

4.4.4 Tempo de preparação de mensagens

Este tempo é essencialmente devido à camada de aplicação que se está a utilizar e por isso bastante variável com a implementação e de norma para norma. Também varia dentro de cada camada com o tipo de serviço utilizado. Observe-se o seguinte exemplo.

Uma aplicação ao evocar o serviço de leitura de uma variável (implementado pelo CAL), primeiro tem que descodificar os parâmetros da função, de seguida irá construir a mensagem de acordo com os atributos da variável e finalmente irá passar a mensagem já construída para a camada de acesso ao meio. A camada de acesso ao meio gere a fila de mensagens à espera de transmissão e coloca a mensagem num dos objectos de comunicação do controlador à espera de ganhar o acesso ao meio.

Partindo dos pressupostos enunciados o cálculo do tempo de preparação de mensagens é dependente: da aplicação, da camada de aplicação utilizada, do tipo de processador e da carga do processador caso se esteja a utilizar um núcleo multitarefa ou da possibilidade de ocorrência de interrupções no decorrer do tempo de preparação da mensagem. Portanto esta parcela não tem um tempo fixo mas varia de acordo com a aplicação a correr no nó que vai transmitir.

4.4.5 Tempo de colocação da mensagem no *buffer* de recepção e tempo de espera no *buffer*

A colocação de uma mensagem no *buffer* pode ser feita utilizando interrupções. Quando chega uma mensagem o controlador CAN gera uma interrupção que activa um rotina

cujas função será a colocação da mensagem no *buffer* de recepção. Dado que uma tarefa deste tipo pode ser bastante simples, o seu tempo de execução pode também ser determinado com facilidade.

O tempo de espera no *buffer* depende sobretudo da aplicação e da altura em que esta tem tempo disponível para retirar a mensagem do *buffer* e processá-la.

O tempo de colocação de mensagens no *buffer* define a taxa máxima a que um nó pode receber mensagens. Claro que a taxa máxima só pode ser mantida por um período de tempo que garante que o *buffer* não enche.

Para o cálculo correcto da taxa máxima de recepção de mensagem é preciso ter em conta que além do tempo para colocar uma mensagem no *buffer* é necessário adicionar o tempo de a camada de aplicação leva até processar cada uma das mensagens recebidas.

4.4.6 Fontes de variabilidade dos tempos de resposta

[Upender *et al*, 96] e [Schneider, 96] detectaram através de simulações que um dos factores que mais influencia o tempo de resposta de uma rede CAN é a serialização de mensagens. Este fenómeno ocorre quando um número elevado de estações tentam ter acesso ao meio no mesmo instante.

Este tipo de cenário tem uma probabilidade de ocorrência bastante baixo, no entanto, este tipo de situações ocorre geralmente em situações de avaria ou acidente, assim quando se dimensiona um sistema é necessário saber que estes fenómenos podem acontecer, quais as suas consequências para o sistema e como se pode resolver.

As causas para a serialização de mensagens são:

- Ruído: o ruído pode momentaneamente corromper a transmissão de dados. Devido aos erros causados existem mensagens que são novamente enviadas automaticamente pelo controlador CAN, o que atrasa a transmissão de outras mensagens de prioridade mais baixa.
- Comunicações tipo pedido de resposta com *broadcast*: esta situação implica um pedido seguido de várias respostas por parte de todas ou de um grande conjunto de estações.

- Carga elevada: quando a rede está sob carga elevada, as mensagens com menor prioridade tendem a esperar mais tempo até serem transmitidas.
- Mensagens longas: se um nó necessitar de transmitir uma mensagem maior do que 8 Bytes tem que partir a mensagem em pedaços mais pequenos e enviá-la em vários pacotes. Durante o tempo em que decorre a transmissão o acesso ao meio para mensagens de menor prioridade vai ser impedido.
- Deriva dos relógios locais: por vezes, devido à baixa precisão dos relógios dos nós estes atrasam-se ou adiantam-se o que leva à transmissão de mensagens cíclicas no instante errado, que pode coincidir com a transmissão de outras mensagens por outros nós.
- Avaria de um nó: quando um nó avaria de forma não silenciosa (está avariado mas transmite mensagens) a rede pode ficar saturada com mensagens provenientes desse nó e as mensagens de mais baixa prioridade tendem a atrasar-se.

O fenómeno da serialização de mensagens pode levar a grandes atrasos no acesso ao meio pelos nós e como consequência algumas *deadlines* das mensagens podem não ser cumpridas.

Novamente [Upender *et al*, 96] e [Schneider, 96] determinaram que o tempo de atraso nas comunicações também variava devido à incerteza do tempo de execução das tarefas responsáveis pela transmissão da mensagem. Verificaram através de simulações que a percentagem de *deadlines* não cumpridas era menor quanto maior fosse a variabilidade do tempo de execução das tarefas responsáveis pela transmissão da mensagem. Note-se que a simulação efectuada partia do princípio que todas as mensagens tinham um período constante. Esta variação contribui para que não ocorram fenómenos de serialização de mensagens quando existem muitas mensagens cíclicas.

A análise através de simulação efectuada por [Upender *et al*, 96] e [Schneider, 96] permite, de forma geral, para sistemas não críticos verificar se o subsistema de comunicações cumpre as especificações. É bastante útil quando o tráfego do sistema não é bem conhecido *a priori*, dado que se pode facilmente simular as condições extremas. Para sistemas críticos é necessária uma abordagem mais precisa do problema de modo a garantir o cumprimento das *deadlines* de todas as mensagens.

4.5 Análise dos atrasos nas comunicações

Foram feitos vários estudos sobre as estratégias de escalonamento de tarefas entre os quais se podem destacar os trabalhos de [Tindell1, 95] e [Liu et al, 73], em que são descritos os resultados de várias estratégias de escalonamento para sistemas que utilizam prioridades fixas. [Milenkovic, 92] descreve as várias estratégias de escalonamento com e sem prioridade, na óptica dos sistemas operativos.

O objectivo do escalonamento das mensagens de um rede CAN é a atribuição correcta das prioridades às mensagens de modo a que a comunicação se efectue cumprindo as *deadlines* impostas pelo sistema. Existem várias estratégias possíveis para a atribuição de prioridades a mensagens enviadas numa rede CAN [Baba, 95], estas estratégias mais não são do que a adaptação das estratégias usadas para escalonamento de tarefas em processadores, a saber:

- SJF*¹⁹ - Esta política escalona as mensagens, atribuindo a prioridade mais alta (identificador de mensagem mais baixo) de acordo com a *deadline* da mensagem e utiliza como factor de desempate, quando as *deadlines* são iguais, o número de bytes da mensagem(o tempo de transmissão).
- DM²⁰ - São atribuídas as prioridades mais altas às tarefas com menor *deadline*.
- RM²¹ - As prioridades são atribuídas de acordo com o período da mensagens. Mensagens com menor período tem prioridade mais alta.

É ainda sugerido em [Tindell1, 95] a utilização de um algoritmo tipo DM modificado utilizando como técnica para seriação das tarefas o valor de D-J em vez da *deadline*. Em que J representa o tempo máximo para colocar a mensagem m no *buffer* de transmissão.

A análise que se segue é baseada em vários trabalhos publicados por Ken Tindell. Esta análise permite determinar, com exactidão, os tempos máximos de atraso na comunicação entre nós.

¹⁹ Shortest Job First

²⁰ Deadline Monotonic

²¹ Rate Monotonic

O CAN utiliza uma política de acesso ao meio baseada em prioridades. [Tindell2, 94] mostrou que a análise aplicada ao escalonamento de tarefas em processadores pode ser aplicada, com algumas adaptações, a uma rede baseada em prioridades.

O tempo máximo de resposta (R_m) de uma mensagem m é considerado como o tempo máximo desde o lançamento da tarefa da camada de aplicação, que vai colocar a mensagem na fila de transmissão, até à sinalização da chegada da mensagem (por exemplo, através de uma interrupção) ao nó ou nós receptores.

O tempo máximo para colocar a mensagem m no *buffer* de transmissão é representado por J_m .

O tempo de espera até que a mensagem tenha direito de acesso ao meio, devido ao meio estar ocupado por mensagens de maior prioridade e de menor prioridade é representado por W_m . O meio está ocupado por mensagens de menor prioridade, do que a mensagem que está a tentar ter acesso ao meio, apenas quando a mensagem de menor prioridade já está a ser transmitida no instante em a mensagem que se quer transmitir tenta aceder ao meio.

O tempo máximo de transmissão da mensagem é representado por C_m , para o cálculo deste tempo pode ser utilizada a análise efectuada no ponto 4.4.3.

Finalmente o tempo máximo de resposta pode ser calculado através da equação seguinte.

$$R_m = J_m + W_m + C_m \quad (3)$$

Em que W_m é calculado através da equação seguinte.

$$W_m = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{W_m + J_j + \tau_{hit}}{T_j} \right\rceil C_j \quad (4)$$

O conjunto $hp(m)$ é formado pelas mensagens de maior prioridade do que m e que podem impedir que a mensagem m tenha acesso à rede. Os valores de J_j e de T_j representam respectivamente o tempo máximo para colocar a mensagem j no *buffer* de transmissão e o período da mensagem. O valor de B_m pode ser obtido através da equação (5) e representa o tempo máximo que a mensagem m tem que esperar até que o meio seja desimpedido devido à transmissão de uma mensagem com menor prioridade.

$$B_m = \max(C_k) \quad \forall k \in lp(m) \quad (5)$$

Como se pode ver nos dois lados da equação (4) o termo W_m pode ser encontrado o que obriga à resolução da equação por recorrência através da equação (6).

$$W_m^{n+1} = B_m + \sum_{\forall j \in lp(m)} \left\lceil \frac{W_m^n + J_j + \tau_{hit}}{T_j} \right\rceil C_j \quad (6)$$

Para o cálculo do valor de W_m devem ser feitas iterações sucessivas, arbitrando o valor de $W_0 = 0$ como valor inicial, até que o valor de W_m convirja.

Este tipo de análise permite saber se um sistema é escalonável, isto é se se verifica a condição representada pela equação (7) para todas as mensagens :

$$R_m \leq D_m \quad (7)$$

Como restrição adicional é necessário garantir que uma mensagem do mesmo tipo não é enviada enquanto outra se encontra no *buffer* de transmissão à espera para ser enviada.

4.6 Conclusões

Os estudos efectuados pelos vários investigadores referidos provam que o CAN pode ser utilizado como rede de comunicação para sistemas distribuídos tempo real. No entanto é necessário efectuar um estudo profundo do funcionamento da rede em cada caso específico de modo a garantir esse funcionamento correcto isto é cumprindo todas as *deadlines*.

A simulação da rede também pode ser um bom auxiliar no desenvolvimento deste sistemas como o provam a quantidade ferramentas existentes para simulação [Farges et al, 96], [Pisartz et al, 96] e [Vaes et al, 96].

5. Análise do CAL

A análise de um sistema é um dos passos necessários para a sua implementação. Para a análise de um sistema é necessário a utilização de métodos que nos permitam a sua estruturação. É bastante usual que a análise seja feita através de modelos.

Os modelos servem vários propósitos [Rumbaugh, 91]:

- testar antes de implementar;
- comunicação com os intervenientes num projecto;
- visualização do sistema;
- melhorar a documentação;
- redução da complexidade.

A ultima razão, é sem dúvida a de maior interesse para o presente caso. A modelização permite que o sistema seja subdividido em diferentes partes, que podem ser analisadas em separado. Segundo [Fishman, 78] as vantagens inerentes à utilização de modelos são:

- permite ao projectista organizar os seus conhecimentos teóricos e observações empíricas do sistema;

- leva a um melhor conhecimento do sistema;
- salienta a necessidade de detalhe e relevância;
- aumenta a velocidade com que uma análise pode ser realizada;
- permite testar as vantagens de modificações no sistema;
- mais fácil de manipular que o próprio sistema;
- permite controlar mais fontes de variação do que o próprio sistema permitiria.

O seja a utilização de modelos permite melhorar a análise.

5.1 Métodos de análise Orientados ao Objecto

A utilização de uma metodologia orientada para o objecto (OO) permite que a análise de um problema se faça com elevado nível de abstracção dirigindo os seus esforços para a definição dos objectos e das suas funcionalidades, concentrando esforço na análise do sistema e não na implementação. Tal como na análise estruturada, uma análise OO obriga na maior parte dos casos a uma construção hierárquica do modelo, dividindo o problema em vários subproblemas representados por diagramas de objectos cooperantes.

Convém notar que a implementação do CAL tem características de tempo real; é necessário garantir tempos máximos de execução dos diferentes serviços. As facilidades disponibilizadas pelos métodos OO são neste caso bastante interessantes para a fase de análise do CAL, embora para a fase de implementação possam ser apontadas algumas desvantagens.

Em [Pinho, 96] são referidas algumas das desvantagens do desenvolvimento de sistemas tempo real utilizando métodos OO, tais como:

- o desempenho das linguagens de programação OO é menor do que o desempenho de uma linguagem de programação estruturada;
- o ênfase que é posto na abstracção dividindo o sistema em objectos, usando as possibilidades de comunicação entre objectos, diminui o desempenho e a eficiência do sistema levando os compiladores à geração de código de grande dimensão e pouco eficiente;

- outro problema, de cariz mais prático e de particular interesse para este trabalho, é a existência de muito poucos compiladores de linguagens orientadas para os objectos para microcontroladores.

Estas desvantagens podem ser superadas utilizando a maior vantagem de uma análise OO que reside na utilização de métodos de análise e de desenho OO. É possível implementar um projecto desenvolvido utilizando metodologias OO através de uma linguagem estruturada mais eficiente e capaz de gerar código consumidor de menores recursos computacionais.

Entre as várias metodologias OO possíveis de utilização optou-se pela metodologia proposta por Rumbaugh “Object Modeling Technique”, usualmente, designada por OMT. As razões para a utilização do OMT foram:

- disponibilidade de bibliografia e ferramentas para a utilização do método;
- cobre todas as fases de desenvolvimento;
- notação normalizada;
- método intuitivo, de facilidade de utilização e compreensão, uma vez que utiliza uma notação simples e pouco complexa;
- tem vistas sobre a estrutura, o comportamento dinâmico e as transformações de dados;
- é largamente utilizado como norma industrial “*de facto*”;
- experiência anterior de utilização.

5.2 Análise OO do CAL

Em anexo encontra-se uma descrição resumida do CAL, que pode servir como base para a compreensão deste capítulo, no entanto é aconselhável a consulta da norma para uma compreensão total de todos os pormenores descritos. Neste ponto tentar-se-á descrever de forma detalhada o CAL utilizando uma análise orientada para os objectos. O objectivo desta análise é a criação de um modelo que sirva de base para a implementação.

Utilizando a forma como está estruturada a norma, as várias partes constituintes do CAL serão descritas, separadamente. A parte inicial irá descrever as funcionalidades gerais que cada nó deve possuir para comunicar com o controlador de CAN de forma transparente de modo a transmitir, receber, tratar erros e configurar o controlador. São apresentados os modelos de objectos da fase de análise assim como os modelos dinâmicos¹ não sendo apresentado o modelo de fluxo de dados devido a não ser muito relevante para a aplicação em causa.

5.2.1 O protocolo

Em cada nó existe um conjunto de objectos que permitem que este interactue e comunique com os restantes nós do sistema. A estrutura de um nó genérico pode ser vista na figura seguinte.

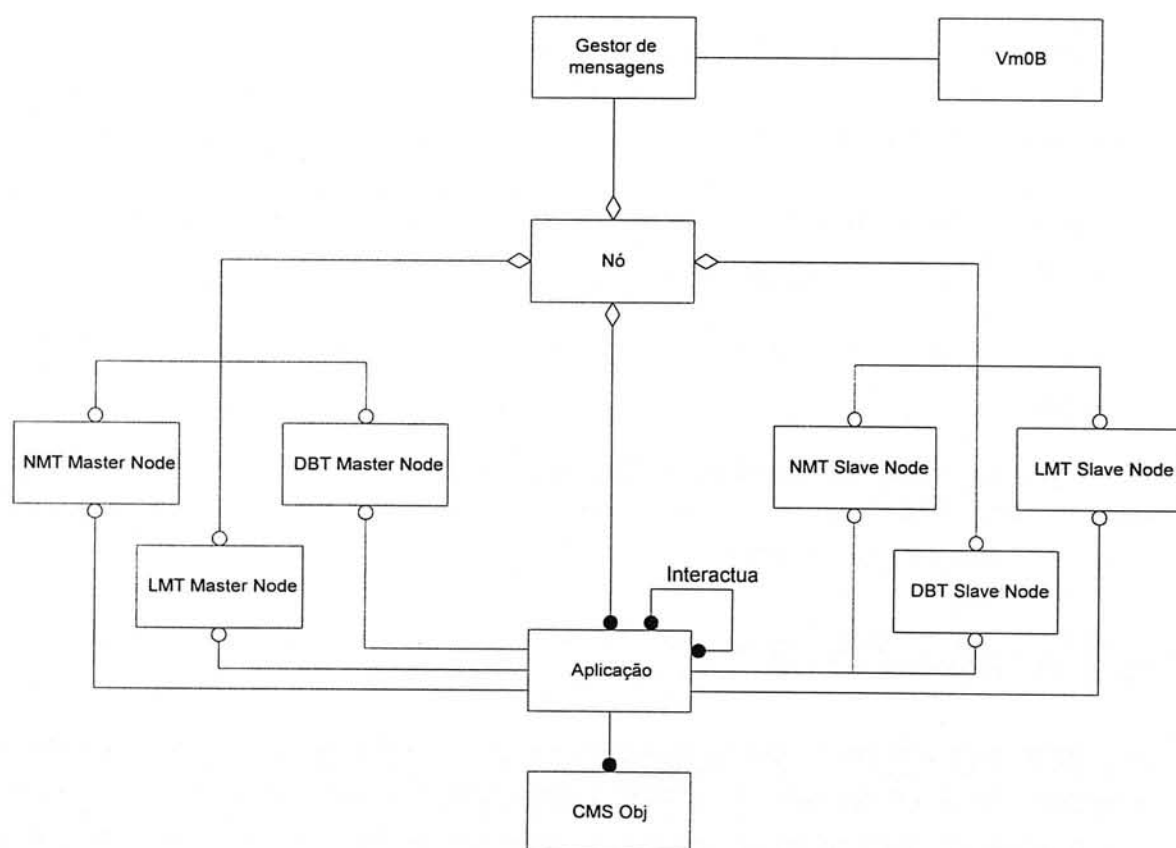


Figura 18 - Representação de um Nó do CAL.

¹ Por vezes simplificados de modo a focar a atenção nos principais problemas

De acordo com a modelização efectuada um nó é constituído por um conjunto de objectos:

- **Aplicação:** este objecto é um objecto genérico que representa as várias tarefas que podem estar a correr no nó;
- **Vm0B:** permite a utilização do controlador de CAN de uma forma independente do *hardware* utilizado e fornece serviços simples de transmissão e recepção de mensagens;
- **Gestor de Mensagens:** interacciona com o objecto Vm0B de modo a transmitir, receber e encaminhar as mensagens para os respectivos objectos destino;
- **CMS Obj:** representa uma superclasse comum a todos os objectos CMS definidos pela norma: variáveis, domínios e eventos;
- **NMT Master Node:** permite que o nó funcione como um mestre de NMT;
- **NMT Slave Node:** possibilita que o nó seja controlado por um mestre NMT presente noutro módulo ou no próprio nó;
- **DBT Master Node:** o nó que contém este objecto é capaz de fazer a distribuição dinâmica de identificadores e de tempos de inibição para todos os COBs utilizados;
- **DBT Slave Node:** este objecto permite a obtenção de identificadores, para os objectos CMS utilizados, a um nó que contenha um objecto **DBT Slave Node**;
- **LMT Master Node:** um nó que contenha este objecto é capaz de actuar como um mestre de LMT, para configurar alguns parâmetros das camadas de aplicação dos outros nós;
- **LMT Slave Node:** permite a sua configuração remota através da rede por um nó mestre LMT.

É de notar que a implementação de um sistema utilizando o protocolo de comunicações CAL exige que existam nesse sistema apenas um objecto **NMT Master Node**, um **DBT Master Node** e um **LMT Master Node**. Os nós que contém o **DBT Master** e o **LMT Master** podem ser nós removíveis (por exemplo, um PC portátil com interface CAN). A configuração dos identificadores e dos parâmetros do nó é feita durante a montagem do

sistema e o nó trata de armazenar a configuração em memória não volátil. Os nós que não disponham desta possibilidade são geralmente configurados através de *DIP switches*.

A gestão das mensagens é um processo complexo que obriga a interações entre o objecto gestor de mensagens, o objecto **Vm0B** assim como com todos os outros objectos capazes de transmitir e receber mensagens (figura 19).

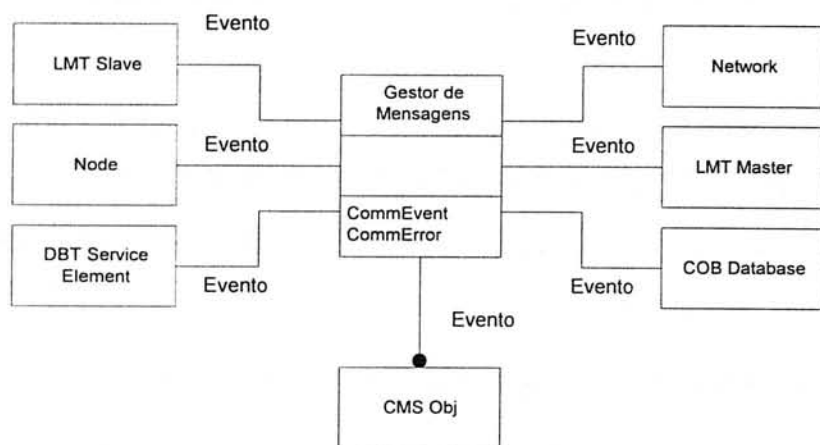


Figura 19 - Interação entre o Gestor de Mensagens e outros objectos

A interação entre estes objectos baseia-se na troca de eventos e na utilização de alguns dos serviços do Vm0B.

O **Gestor de Mensagens** ao receber uma mensagem obtém o seu identificador. Este identificador irá corresponder a um e um só dos COBs utilizados pelos objectos do nó. Após a identificação COB, o Gestor de Comunicações pode enviar um evento para o respectivo objecto que se encarregará de tratar o conteúdo da mensagem. Deve poder, se assim for configurado, guardar o COB num *buffer* de recepção para posterior processamento.

A figura 20 representa a relação entre o gestor de mensagens e o Dicionário de Descritores de COBs.

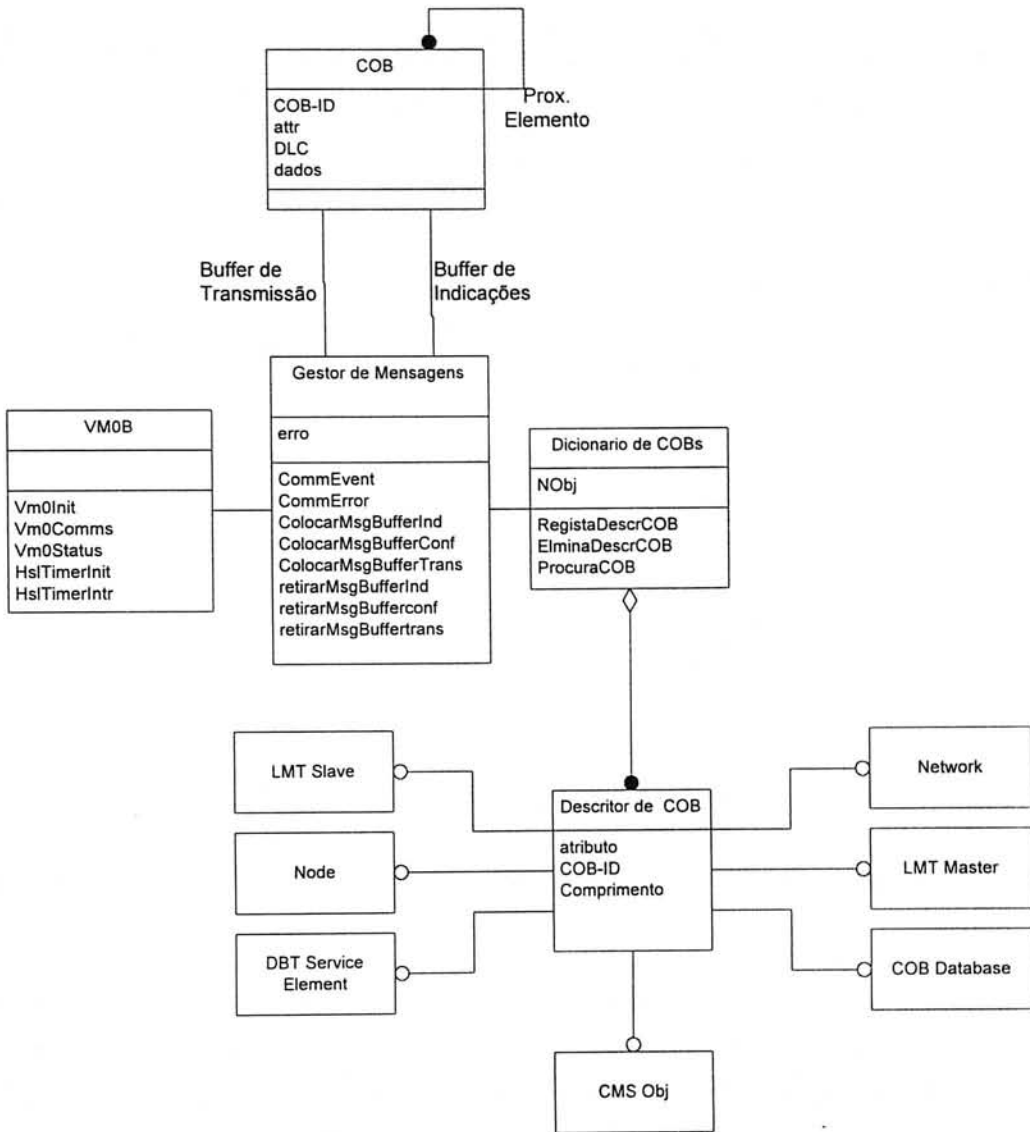


Figura 20 - Estrutura do dicionário de COBs.

De modo a que o gestor de comunicações possa saber a que objecto se destinam as mensagens, tem que utilizar uma estrutura que armazene o significado de cada COB que o nó utiliza. Esta estrutura é representada pelo objecto **Dicionário Descr. de COB** que é constituído por vários objectos **Descritor de COB**. Cada um destes objectos tem uma ligação com o objecto que o criou.

5.2.1.1 Comportamento dinâmico

O único objecto com um comportamento dinâmico importante é o gestor de comunicações; este tem que receber e enviar mensagens e tratar erros de comunicação.

As suas funcionalidades só são possíveis através de um conjunto de estados concorrentes para efectuar a transmissão, a recepção, o controlo de erros e a inicialização.



Figura 21 - Diagrama de estados do objecto **Gestor de Mensagens**.

Os estados apresentados na figura 21 são completamente especificados nas figuras seguintes.

A inicialização do gestor de mensagens deve ser feita pela aplicação. Devem ser inicializados os parâmetros referentes à velocidade do barramento, tamanho dos buffer de transmissão e recepção. Com estes parâmetros o objecto trata de inicializar o objecto Vm0B.



Figura 22 - Diagrama de estados do objecto **Gestor de Mensagens**, especialização do estado de **Inicialização**.

A recepção de mensagens é uma tarefa crítica já que como foi visto é necessário que todo o processamento de recepção de mensagens seja feito em cerca de 50 μ s quando a rede

troca mensagens à velocidade de 1Mbit/s. Na figura 23, estão representados dois tipos de funcionamento possível

- (1) Recebe do objecto Vm0B um evento a indicar que chegou uma mensagem, lê a mensagem e envia um evento para o objecto que a deve receber.
- (2) Quando se opera com elevadas velocidades é necessário ler a mensagem e guardá-la num buffer para posterior processamento e pelo objecto consumidor da mensagem.

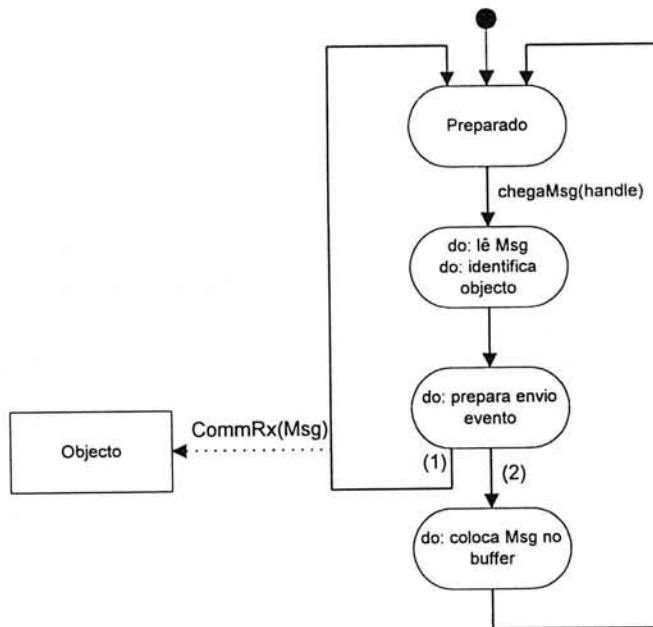


Figura 23 - Diagrama de estados do objecto **Gestor de Mensagens**, especialização do estado de **Recepção de Mensagens**.

Para a transmissão são necessárias duas máquinas de estados separadas: uma que controla a entrada de mensagens no buffer, e, outra que sempre que detecta que o buffer tem mensagens e que o Vm0B está livre tenta transmiti-las.

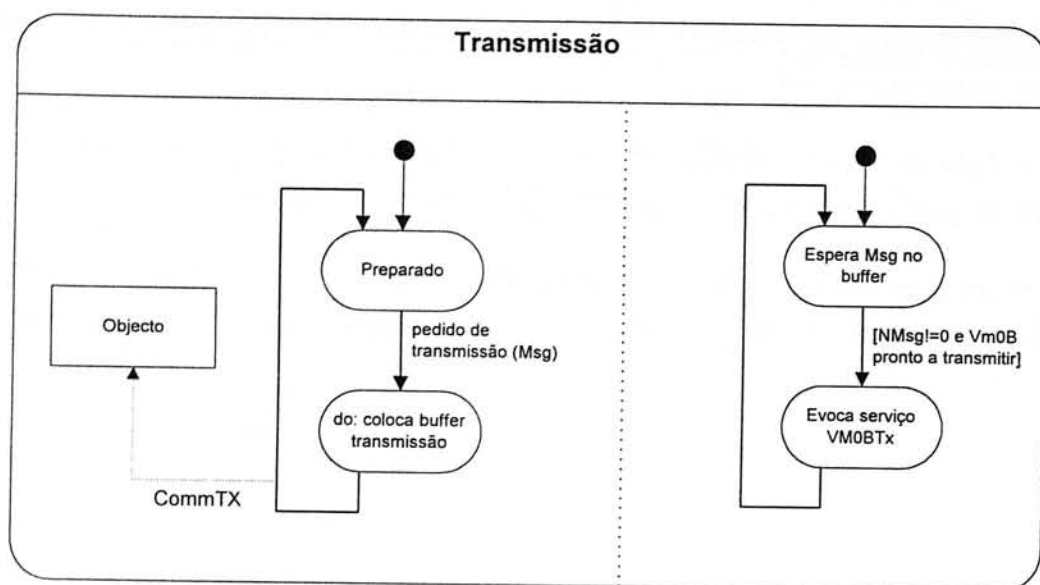


Figura 24 - Diagrama de estados do objecto **Gestor de Mensagens**, especialização do estado de **Transmissão de Mensagens**.

O último diagrama desta série apresenta a forma como são tratados os erros de comunicação. O superestado **Monitoriza Comunicações** espera que ocorra um erro nas camadas inferiores representadas pelo objectos Vm0B, sinaliza esse facto às aplicações que estão a tentar comunicar e actualiza o atributo erro contido no objecto **Gestor de Comunicações**.

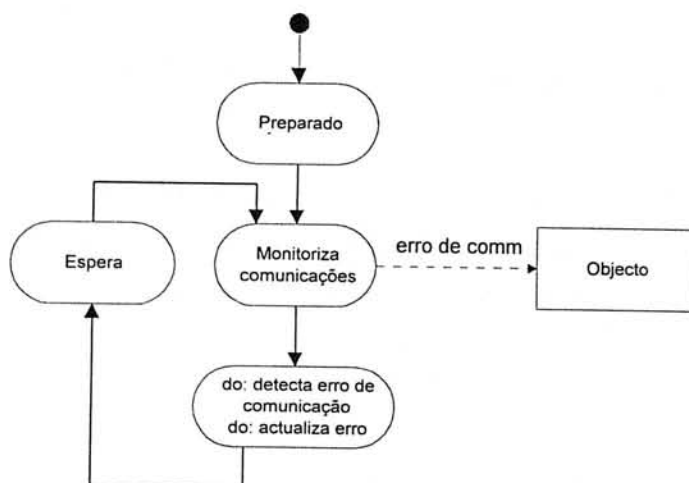


Figura 25 - Diagrama de estados do objecto **Gestor de Mensagens**, especialização do estado de **Transmissão de Mensagens**.

5.2.2 O CMS

O CMS define os vários objectos que serão trocados entre os nós quando o sistema está em funcionamento pleno.

Todos os objectos CMS partilham uma estrutura comum que se modelizou através de uma superclasse chamada de **Obj. CMS**. Esta classe contém os atributos comuns a todos os objectos CMS. Estão definidos na norma [CAL, 96], (DS202-1 p.6)², excepto, o atributo **value** que representa os dados referentes à variável. A sua inclusão neste objecto levanta algumas dúvidas uma vez que pode não ser necessário.

Imagine-se uma variável do tipo Write-Only que mapeia uma saída digital, quando um cliente actualiza o valor dessa variável, o servidor desta, pode apenas modificar o valor da sua saída digital em função do valor enviado sem necessitar de guardar o seu valor.

Outra situação, será por exemplo o de uma variável Read-Only que representa o valor de uma entrada analógica. Se o cliente quiser ler essa variável o seu valor pode ser obtido directamente pelo servidor na entrada analógica e colocado no buffer de resposta remota sem que necessite de ser armazenado em memória.

Estes factos levam à conclusão de que a utilização do atributo de **value** depende da aplicação, no entanto, este atributo é colocado no Obj. CMS de modo a uniformizar o seu uso.

² No resto do texto deste capítulo passa-se apenas a referir a página sempre que houver uma referência à norma CAL.

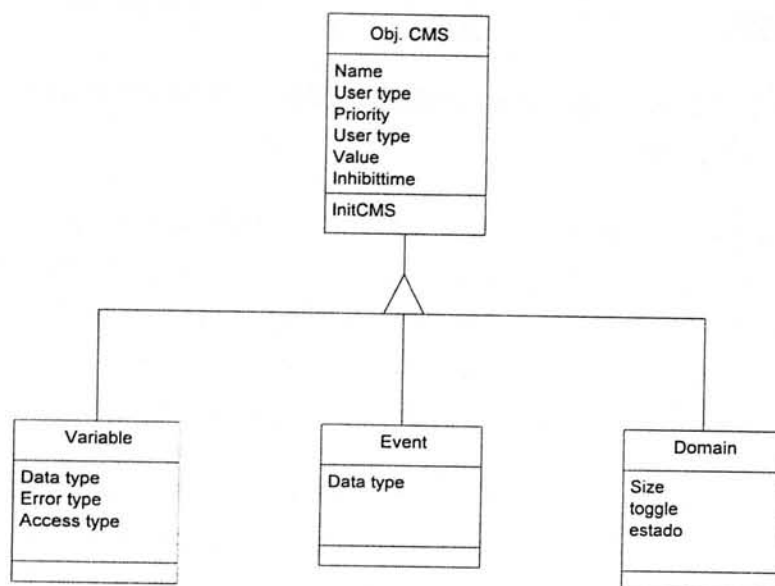


Figura 26 - Objectos CMS.

A especialização do objecto **Obj. CMS** (superclasse) permite definir os vários tipos de objectos que a norma define: variáveis, eventos e domínios. Este objectos sofrem uma nova especialização (subclasses), figuras 27, 34, 35, 43, de modo a acomodar as características que são apenas de cada um dos subtipos.

5.2.2.1 Variáveis Básicas

As variáveis básicas permitem a transferência de dados ou de comandos com um tamanho máximo de 8 bytes e utilizam um ou dois COBs para efectuar os seus serviços.

A figura 27 representa a especialização de um objecto variável em variáveis básicas clientes e servidoras.

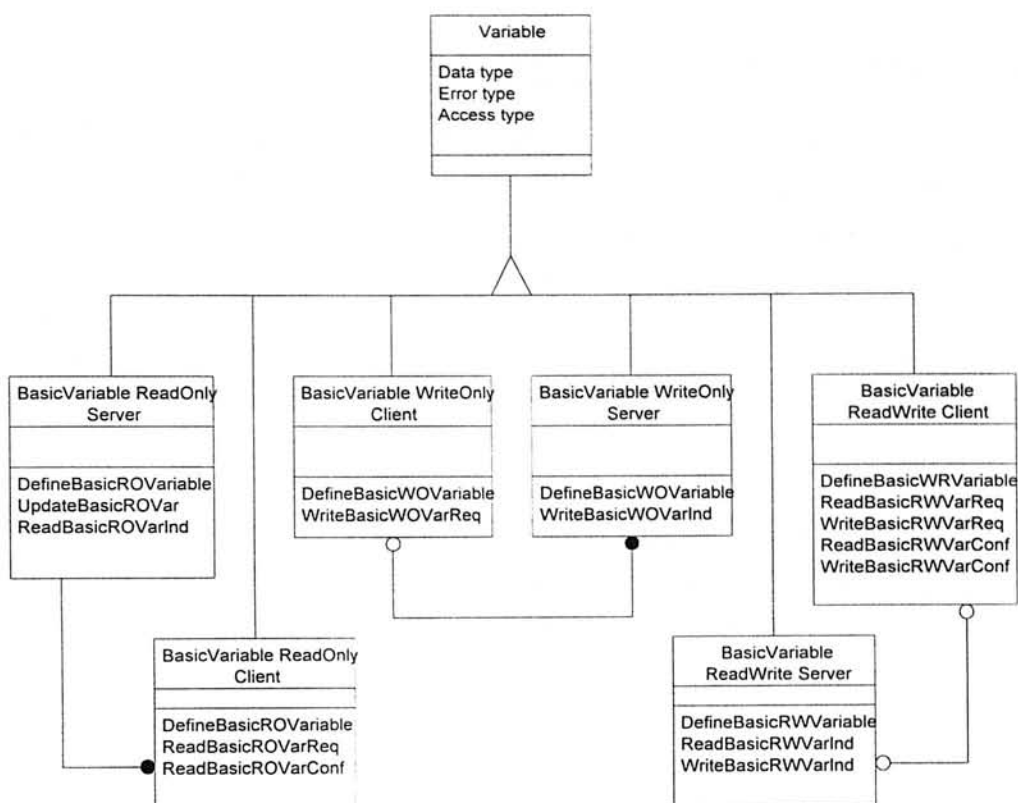


Figura 27 - Especialização do diagrama de objectos para variáveis básicas.

As ligações representadas na figura 27, entre objectos clientes e objectos servidores, são puramente conceptuais, isto é, não existem na realidade mas permitem modelizar alguns aspectos importantes relativos à interacção entre estes. A ligação (comunicação) entre os vários objectos é feita na realidade por via indirecta utilizando os serviços do Gestor de Comunicações e do VmoB. A ligação representa e modeliza o atributo COB-class (DS202-1 p.4), isto é, o número de objectos com que se podem relacionar.

O objecto **BasicVariableReadOnlyServer** tem o seu COB definido como de classe 7, o que quer dizer que pode ter uma instância do objecto servidor **BasicVariableReadOnlyServer**, e N instâncias para o objecto cliente **BasicVariableReadOnlyServer**.

Os objectos **BasicVariableWriteOnlyServer** e **BasicVariableWriteOnlyClient** comunicam entre si utilizando COBs de classe 2 o que permite que exista 1 instância do cliente e N instância para o servidor em todo o sistema.

Finalmente os objectos **BasicVariableReadWriteServer** e **BasicVariableWriteOnlyClient** comunicam utilizando COBs de classe 1 e 4 levando à existência de apenas uma instância no sistema para cada um dos objectos.

O servidor apresenta um conjunto de estados pouco complexo.

Primeiro e, tal como, para todos os outros objectos CMS, é feita a inicialização da variável e é registado o seu COB no dicionário de descritores de COBs³. A chegada do evento Start significa que o objecto e todo o nó passaram a um estado em que todos os objectos CMS estão activos e podem receber e transmitir mensagens.

As transições para os dois estados seguintes são desencadeadas pela chegada de uma indicação para a leitura da variável ou por um pedido para actualização do valor que foi colocado no *slot* do controlador CAN.

Caso haja um erro na execução da actualização da variável, este é sinalizado à aplicação que evocou o serviço. Também poderá ser sinalizado à aplicação o sucesso da evocação do serviço.

Em todos os diagramas de estados existem duas possibilidades relativas à recepção de um indicação, que são:

1. o utilizador pode configurar o software de modo a que as funções de indicação existentes executem o processamento desejado por ele;
2. o utilizador pode configurar o objecto para sinalizar o evento à sua aplicação.

Este dois comportamentos deve-se ao facto da implementação conter funções genéricas para o atendimento de indicações e para o tratamento de erros que podem ser alteradas pelo programador de modo a adaptar o seu funcionamento a outras aplicações.

O funcionamento do estado **Trata erro** é idêntico ao da recepção de indicação. Podendo ser utilizadas as duas formas de tratamento de erro: comunicação à aplicação ou tratamento pelo objecto.

³ Permite que um *slot* do controlador CAN seja configura como *Remote Transmitter* para responder a pedidos do cliente.

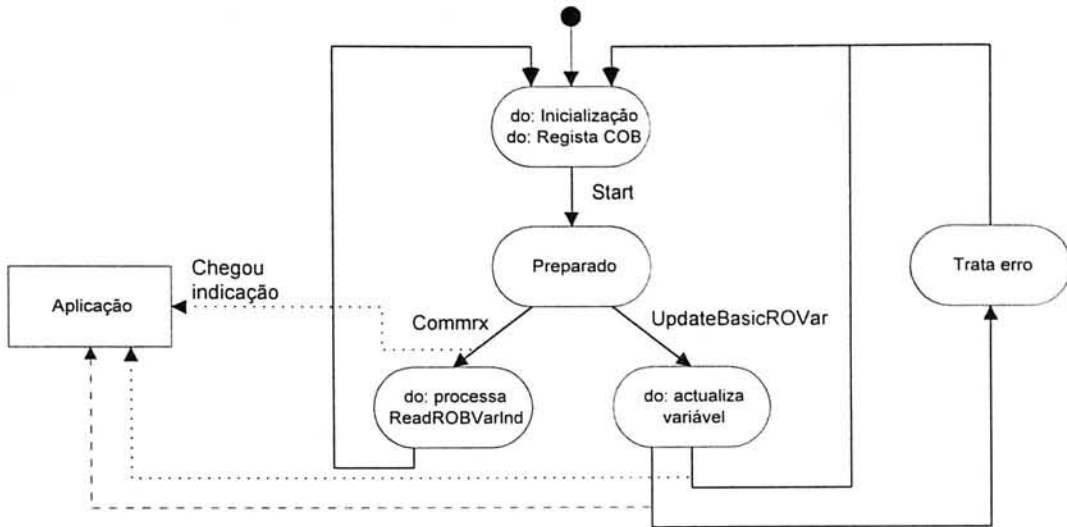


Figura 28 - Modelo dinâmico do objecto **BasicVariableReadOnlyServer**.

O objecto cliente tem um modelo dinâmico ligeiramente mais complexo. Necessita de enviar um pedido de trama remoto e espera pela resposta. Existem duas situações de erro que devem ser sinalizadas à aplicação:

- um erro na transmissão devido a problemas com o *hardware* ou com os objectos encarregados da transmissão de mensagens;
- um erro devido à mensagem não ter chegado dentro de um tempo especificado, erro de *TimeOut*.

Caso a mensagem chegue correctamente, esse evento também é sinalizado à aplicação que efectuou pedido.

A norma define, que mensagens relativas a uma variável não podem ocupar o meio com um número excessivo de transmissões sucessivas, por isso, após o final de uma transmissão com sucesso é necessário que se passe a um estado de inibição da transmissão de mensagens relativas a essa variável.

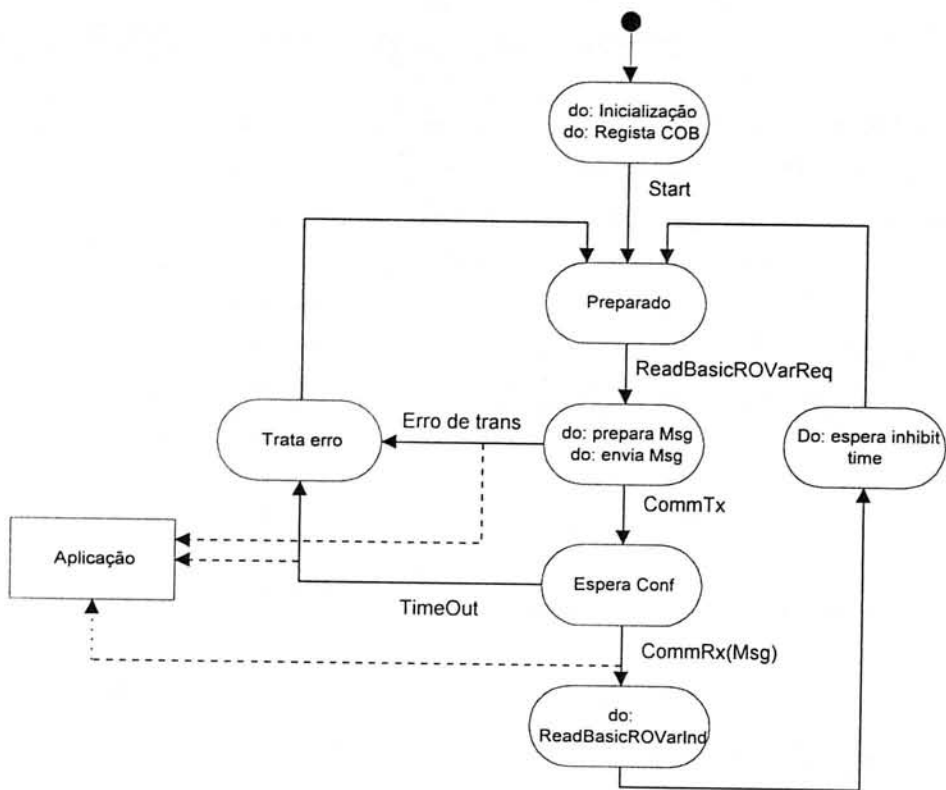


Figure 29 - Modelo dinâmico de uma variável **BasicVariableReadOnlyClient**.

As variáveis do tipo WriteOnly podem ser usadas pelo cliente de modo a que um ou mais servidores executem o mesmo comando. É enviada uma mensagem que é recebida por todos os servidores, sem que estes efectuem qualquer resposta,(DS202-2 p.5).

O objecto **BasicVariableWriteOnlyServer** tem apenas como função receber a mensagem e sinalizar o evento à aplicação respectiva.

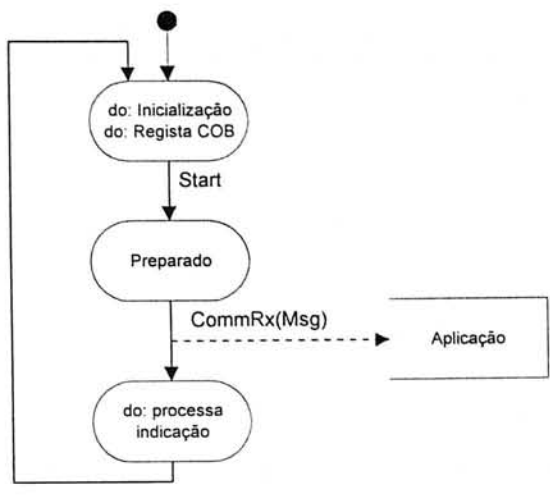


Figure 30 - Modelo dinâmico de uma variável **BasicVariableWriteOnlyServer**.

O objecto cliente para esta variável tem também um funcionamento simples durante o qual envia a mensagem para os outros nós, sinaliza o seu sucesso ou insucesso à aplicação e espera o tempo de inibição respectivo.

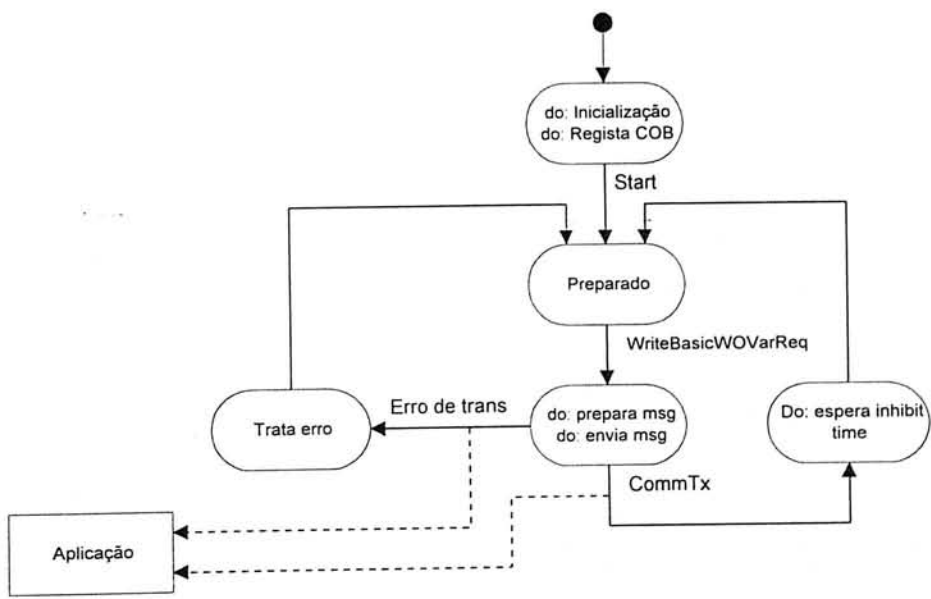


Figure 31 - Modelo dinâmico de uma variável **BasicVariableWriteOnlyClient**.

As variáveis ReadWrite permitem a implementação de serviços confirmados, isto é, o cliente pode, utilizando os serviços de escrita enviar um comando e receber como resposta o resultado da execução desse comando. Podem também servir para ler o valor

de uma variável sem que para isso seja necessário utilizar variáveis ReadOnly que utilizam os *slots* do controlador de CAN (o número destes *slots* não excede os 15 nos controladores C167 e 82527).

O funcionamento do objecto **BasicVariableReadWriteServer** é o seguinte: ao chegar uma mensagem que lhe é dirigida trata de a interpretar de modo a saber se é um pedido de escrita ou um pedido de leitura, ao detectar um destes eventos sinaliza a aplicação. Esta trata o pedido de acordo com o seu estado e os valores enviados pelo cliente. Devolve a **BasicVariableReadWriteServer** o valor que deve ser enviado para o cliente em caso de sucesso ou de falha (é objecto **Aplicação** quem determina o sucesso ou a falha do serviço).

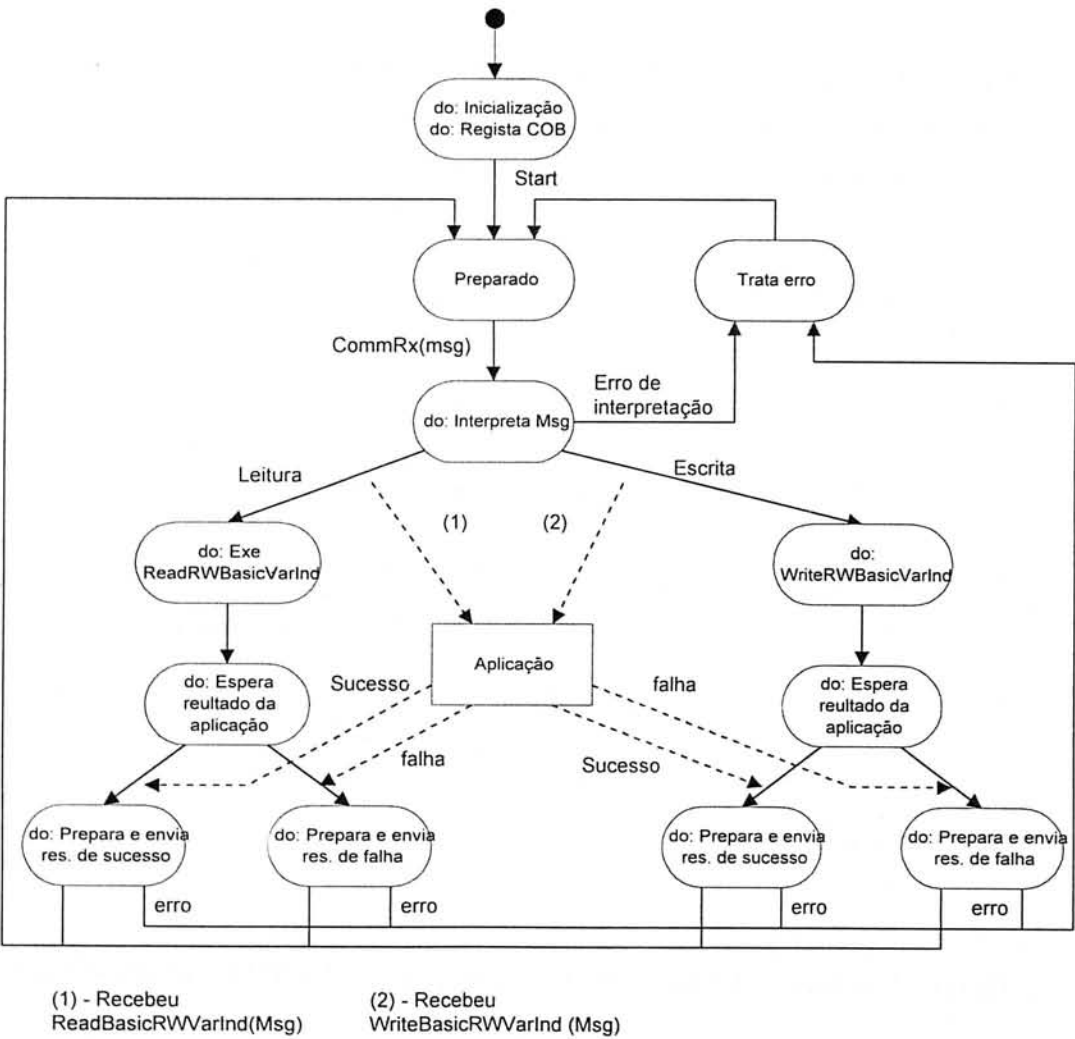


Figura 32 - Modelo dinâmico de uma variável **BasicVariableReadWriteServer**.

O objecto **BasicVariableReadWriteServer** pode processar dois tipos de pedidos: de leitura **ReadBasicRWVarReq** e de escrita **WriteBasicRWVarReq**.

O processamento de ambos os pedidos é idêntico. É enviada a mensagem de pedido, (DS202-2 p.6), com o campo de dados preenchido com os valores requeridos pela aplicação. Fica à espera da mensagem de confirmação. Passa à fase de interpretação mensagem, em que é verificado o primeiro byte de modo a verificar se houve sucesso ou se existe um erro num dos campos especificados pela norma para o primeiro byte. O resultado é devolvido à aplicação que requisitou o serviço.

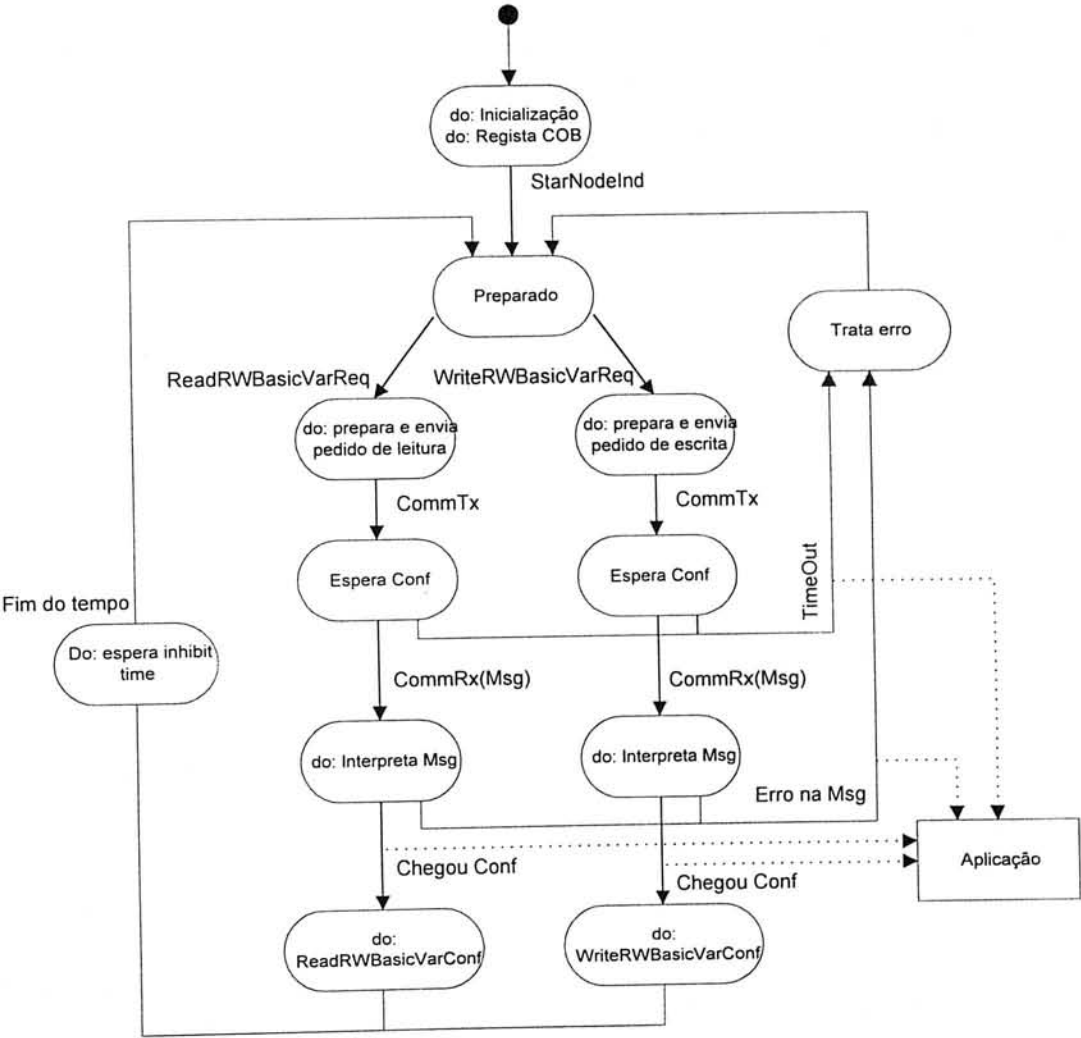


Figura 33 - Modelo dinâmico de uma variável **BasicVariableReadWriteClient**.

5.2.2.2 *Variáveis multiplexadas*

Este tipo de variáveis permite a utilização de apenas um conjunto de dois COBs por um conjunto de objectos que partilham o mesmo atributo `var_set_name`, representado na figura 34 por um atributo de ligação. A distinção entre elas é feita pelo atributo `multiplexor`. Note-se que a instanciação destes objectos só pode ser feita tendo em conta que as várias instâncias devem ter o mesmo valor para os atributos `user_type`, `access_type` e `priority`. Podem, no entanto, ser utilizadas para enviar dados de diferentes tipos.

Os modelos dinâmicos deste conjunto de objectos não serão apresentados dado que são muito semelhantes aos apresentados para variáveis básicas excepto no caso dos objecto **MuxVariableReadOnlyClient** e **MuxVariableReadOnlyClient** que necessitam de processamento adicional por parte do cliente e do servidor, a resposta automática não é possível para o servidor.

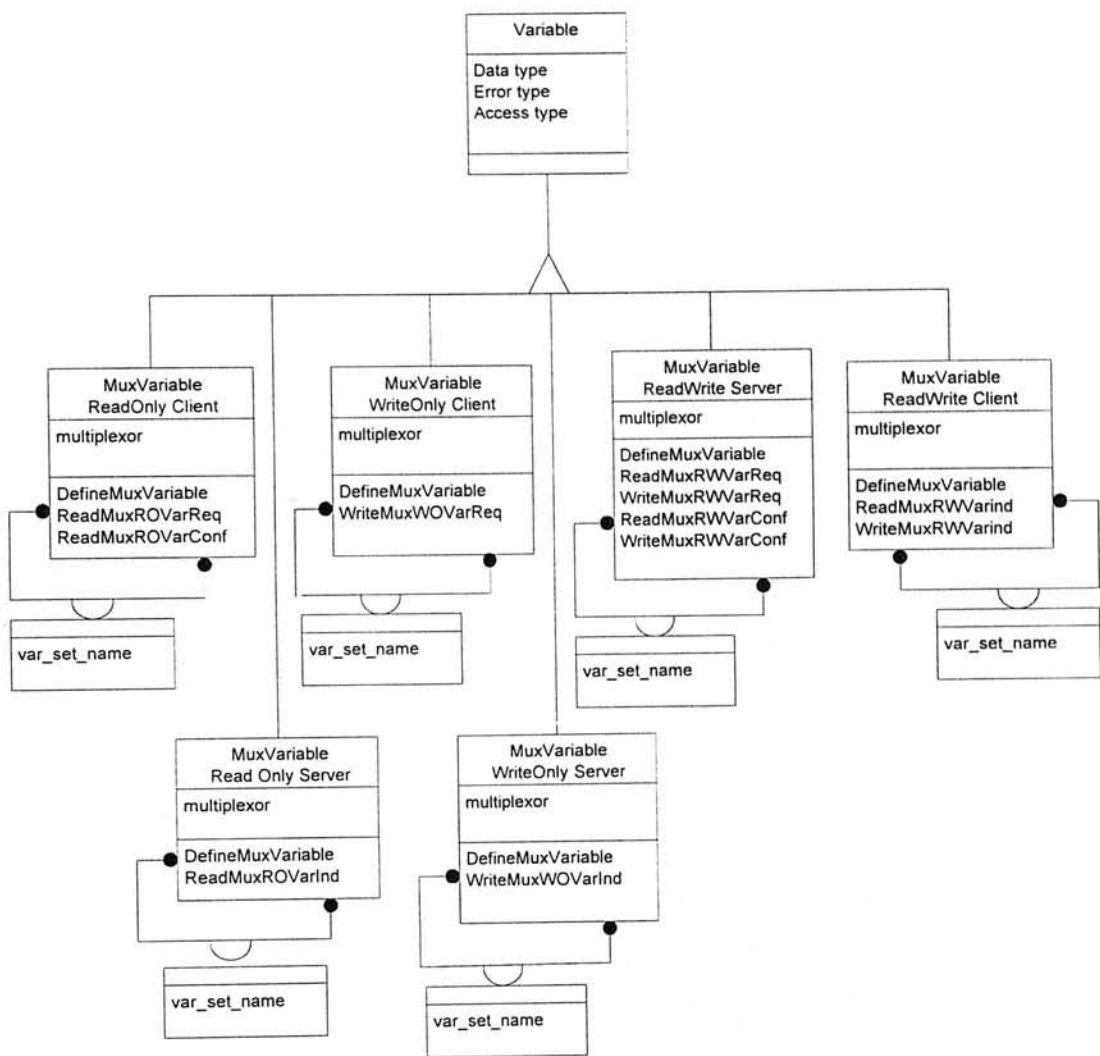


Figura 34 - Especialização para variáveis multiplexadas.

5.2.2.3 Domínios

Os domínios básicos podem ser utilizados para a transferência de grandes blocos de informação.

A sua transferência é feita partindo os dados a enviar, em vários segmentos, que são descarregados (*download*) para o servidor ou são carregados do servidor (*upload*).

Os domínios multiplexados, tal como as variáveis multiplexadas permitem, utilizando o mesmo conjunto de COBs, a transferência de diferentes conjuntos de dados.

A transferência dos dados é da responsabilidade da aplicação, que Inicializa a transferência de um domínio e de seguida controla a transferência dos vários segmentos.

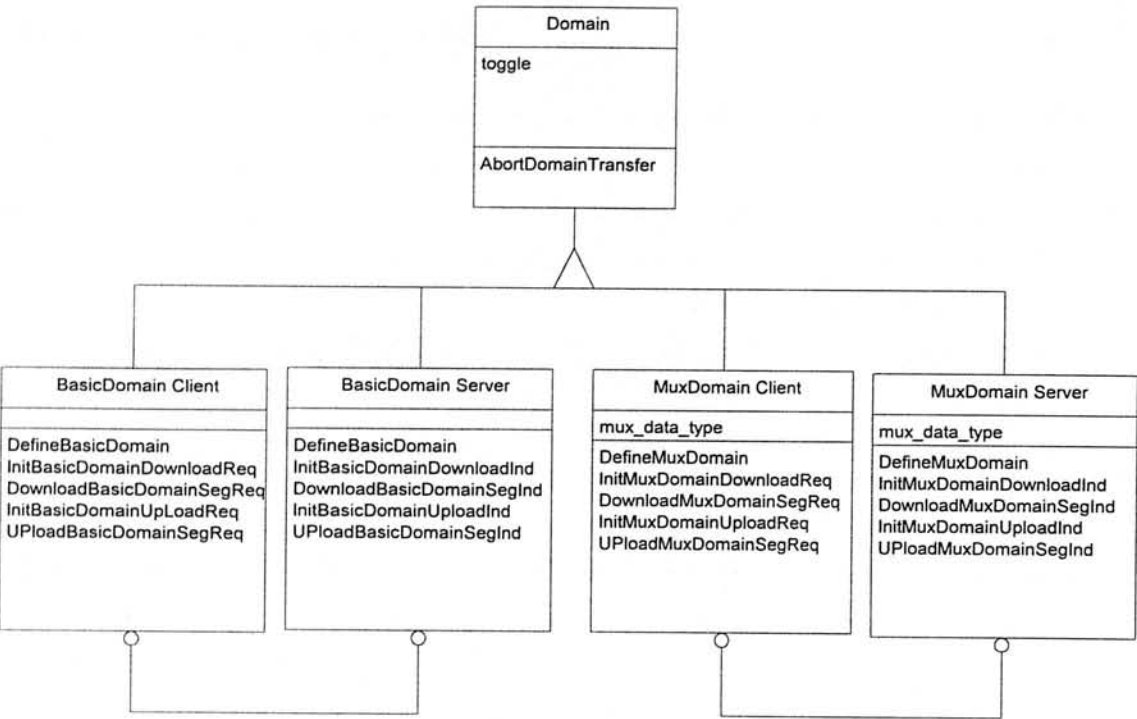


Figura 35 - Especialização para domínios.

O diagrama de estados de um **BasicDomainServer**, figura 36, contempla os três tipos de serviços que este é capaz de efectuar, nas figuras 37 e 38 são desenvolvidos os estados **Em Download** e **Em Upload**.

Não está representado nas figuras seguintes que caso chegue uma indicação ou um pedido de **AbortDomainTranfer** ao cliente ou ao servidor do domínio, os objectos **BasicDomainServer** e **BasicDomainClient** passam de qualquer estado para o estado **Preparado**.

As tramas utilizadas têm um formato semelhante para todos os domínios em que, usualmente, o primeiro byte, representado no diagrama dinâmico pelos elementos *css* (*client comand specifier*) e *scs* (*server comand specifier*), corresponde ao tipo de trama que está a ser enviada. Outro dos seus elementos é o bit *c* que representa se existem mais segmentos para serem descarregados ou se já se chegou ao fim da descarga.

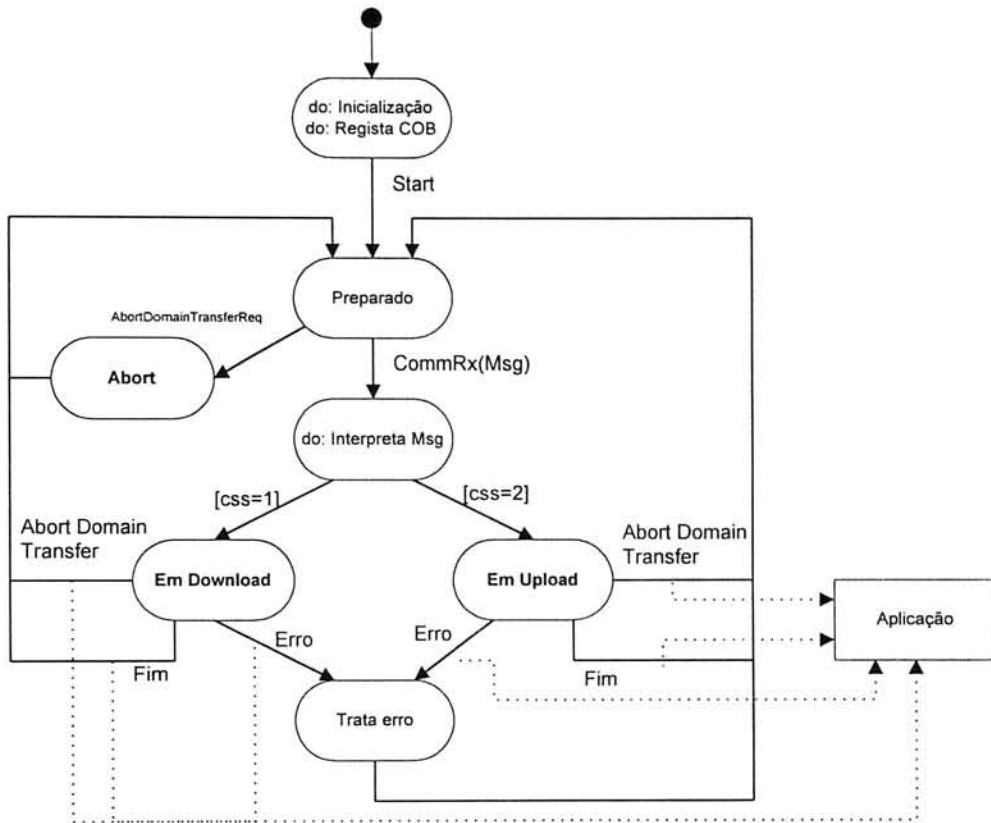


Figura 36 - Diagrama de estados para o objecto **BasicDomainServer**.

O protocolo de *download*, (DS202-2 p.8), especifica a forma como deve decorrer a descarga de um domínio para um servidor. A descarga do domínio também deve ser controlada pela aplicação servidora, que se encarrega de guardar e testar os valores dos dados que vão chegando.

Neste caso, após ter recebido uma trama com o ccs igual 1 o servidor sinaliza a aplicação e fica à espera que lhe seja enviado outro segmento. Ao chegar uma nova mensagem é feita a sua interpretação que pode resultar na chegada de um novo segmento ou na recepção de um erro. Este erro pode resultar do byte de comando ccs recebido ser de um valor diferente do esperado. Qualquer um destes eventos é sinalizado à aplicação. Sai do ciclo quando recebe um comando de fim da mensagem.

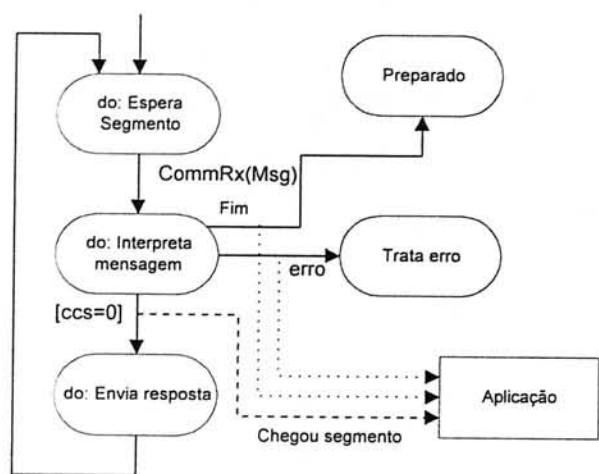


Figura 37 - Diagrama de estados para o objecto **BasicDomainServer**, especialização do estado **Em Download**.

O *Upload* é feito de forma análoga ao *download* mas, neste caso, é a aplicação que se encarrega de fornecer os dados para responder ao cliente.

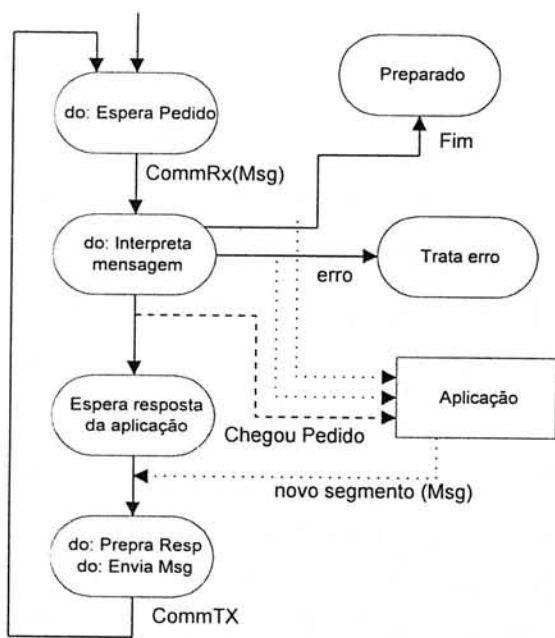


Figura 38 - Diagrama de estados para o objecto **BasicDomainServer**, especialização do estado **Em Upload**.

O objecto **BasicDomainClient** permite o controlo da transferência de domínios entre um nó cliente e um nó servidor. Através dos serviços de **InitBasicDomainDownloadReq** e **InitBasicDomainUploadReq** inicializa o protocolo, os serviços de **DownloadBasicDomainSegReq** e **DownloadBasicDomainSegReq**.

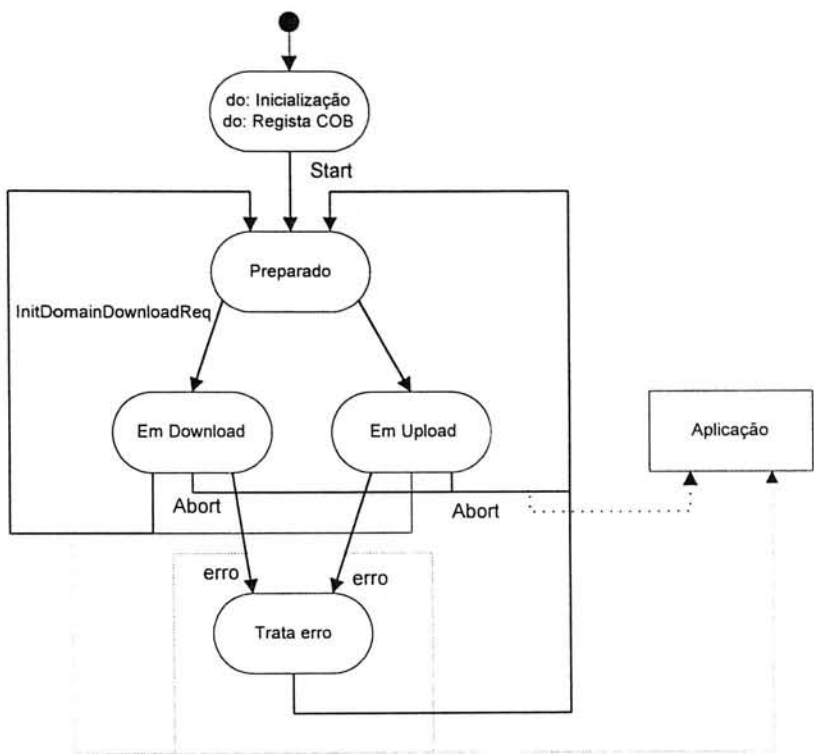


Figura 39 - Diagrama de estados para o objecto **BasicDomainClient**.

Durante o estado de **Em Download** é feita pela aplicação requerente, a transferência dos dados.

Após enviar a primeira mensagem de inicialização, fica à espera da resposta do servidor. Após a recepção de uma resposta com sucesso passa a um estado de espera durante o qual aguarda que a aplicação faça um pedido para a transferência de um segmento. Note-se que o objecto passa de qualquer um dos estados representados na figura para o estado de preparado com a chegada de uma indicação ou de um pedido de abort.

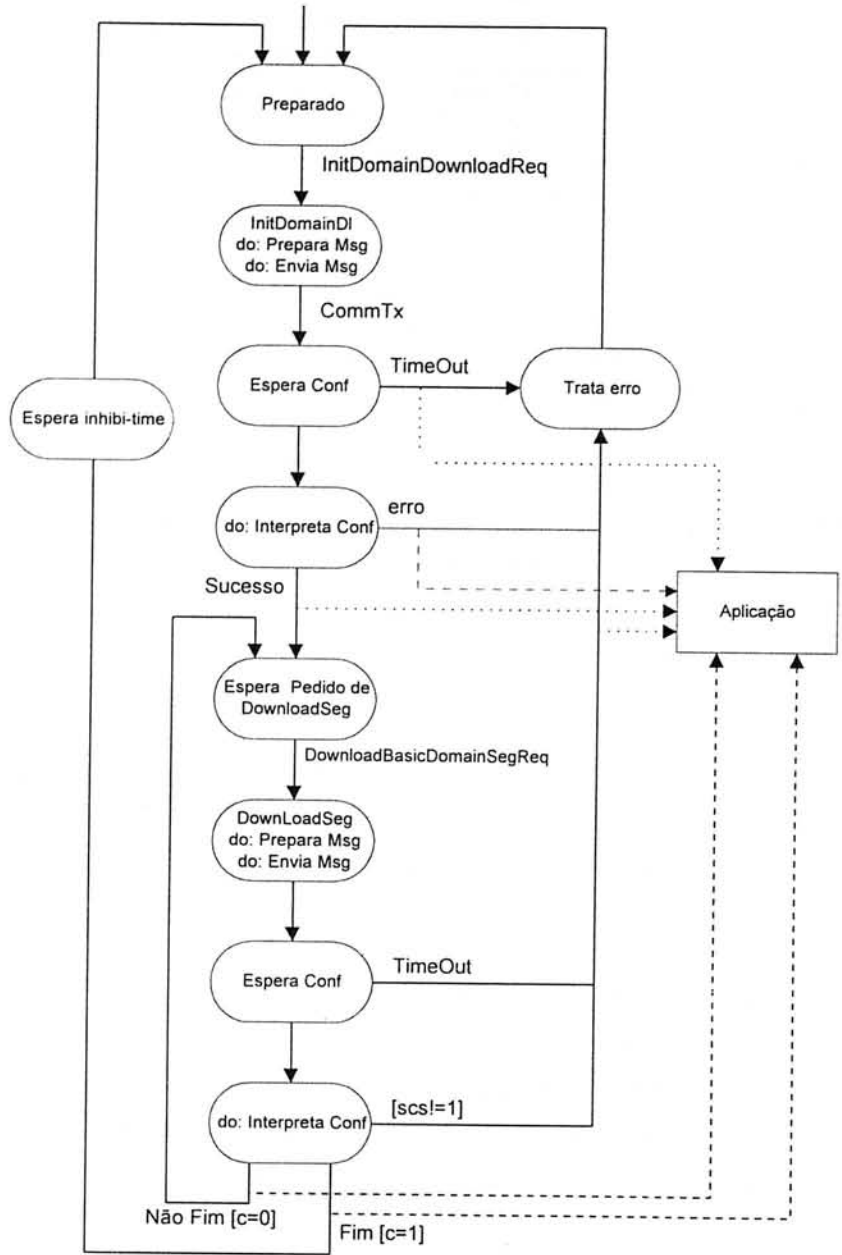


Figura 40 - Diagrama de estados para o objecto **BasicDomainClient**, especialização do estado **Download**.

O *upload* de um domínio é efectuado de forma muito semelhante ao *download*.

Existem alguns aspectos que devem ser considerados:

- a aplicação servidora pode enviar o tamanho dos dados que irão ser transferidos. Este valor é passado à aplicação e é da sua responsabilidade as acções a tomar caso chegue um segmento com a indicação de fim de transferência antes do número de bytes transferidos atingir o valor enviado.

- caso a aplicação servidora não envie o tamanho dos dados, a transferência é feita até que seja enviado um comando de fim de mensagem.
- o processo pode ser terminado e o objecto passa ao estado de **Preparado** devido a um erro, devido a um **AbortDomainTransfer** ou devido a **InitDomainUploadReq** (estas duas últimas razões não estão contempladas no diagrama de estados).

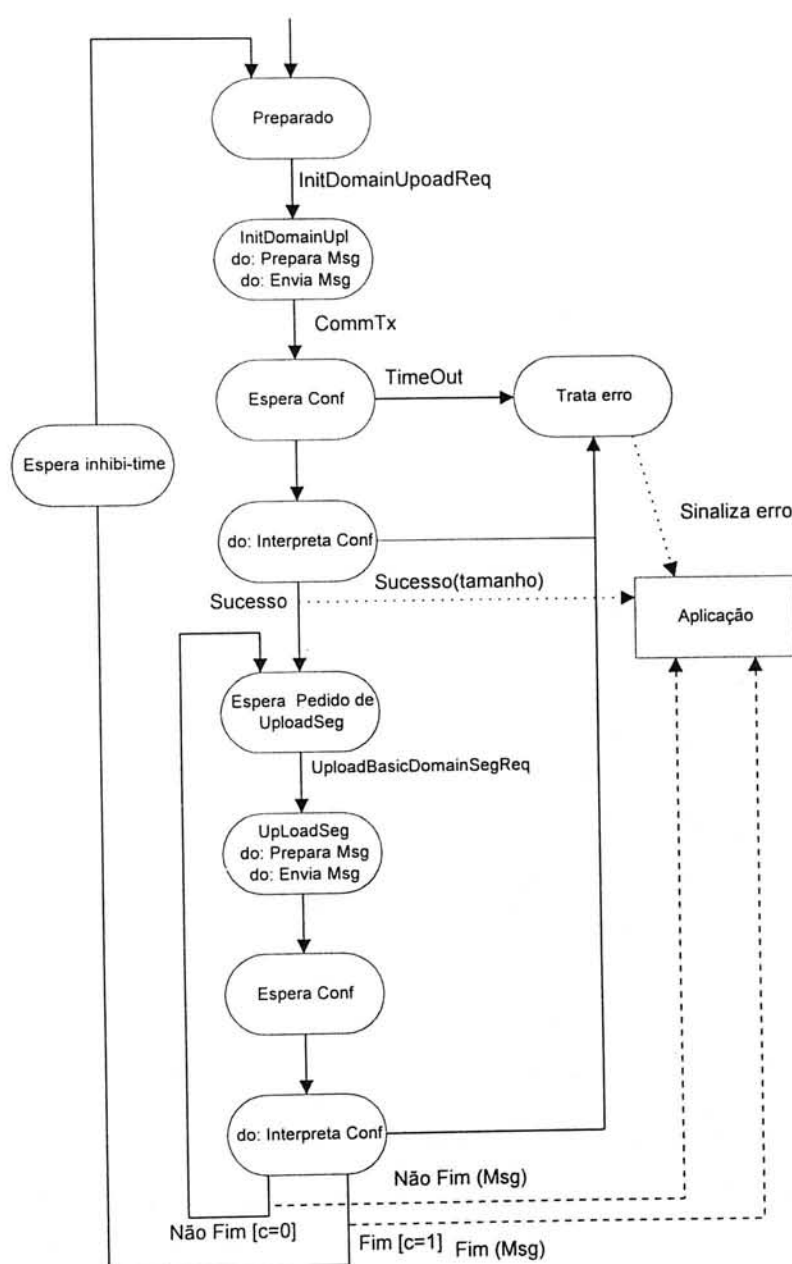


Figura 41 - Diagrama de estados para o objecto **BasicDomainClient**, especialização do estado **Upload**.

Tal como referido no caso da recepção de uma indicação de **AbortDomainTransfer** o objecto **BasicDomainClient** passa de qualquer estado para o estado preparado, figura 42. De modo a que isto seja possível é necessário que exista uma outra máquina de estados para o mesmo objecto que concorre com a máquina de estados principal.

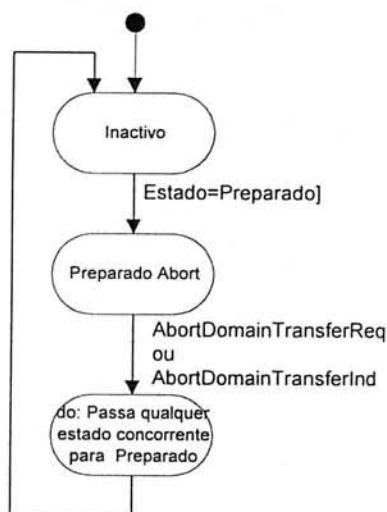


Figura 42 - Diagrama de estados para o objecto **BasicDomainClient**, especialização do estado **Abort**.

5.2.2.4 Eventos

Os eventos são utilizados de forma a transmitir acontecimentos assíncronos no sistema. Por exemplo: que o nível de um líquido num tanque atingiu um determinado nível.

No entanto, e dadas as suas potencialidade podem também ser usados para a transferência de informação, síncrona e assíncrona, tal como foi adoptado pela norma CANOpen.

Os eventos do tipo *Uncontrolled* permitem a notificação de qualquer cliente que esteja interessado em receber esse evento. São bastante semelhantes em termos de implementação às variáveis básicas do tipo *WriteOnly*.

O eventos do tipo *Controlled* permitem que um cliente possa controlar se esse evento irá ser transmitido pelo servidor ou não. Quando o seu estado é activo o servidor notifica, automaticamente, o cliente.

Para eventos do tipo *Stored* um servidor pode notificar um cliente da sua ocorrência ou armazenar o seu valor num *slot* do controlador CAN como *RemoteTransmitter*.

Dada a simplicidade dos eventos do tipo *Uncontrolled* e *Controlled* e a sua semelhança com as variáveis básicas já descritas, anteriormente, não serão apresentados os seus diagramas de estados.

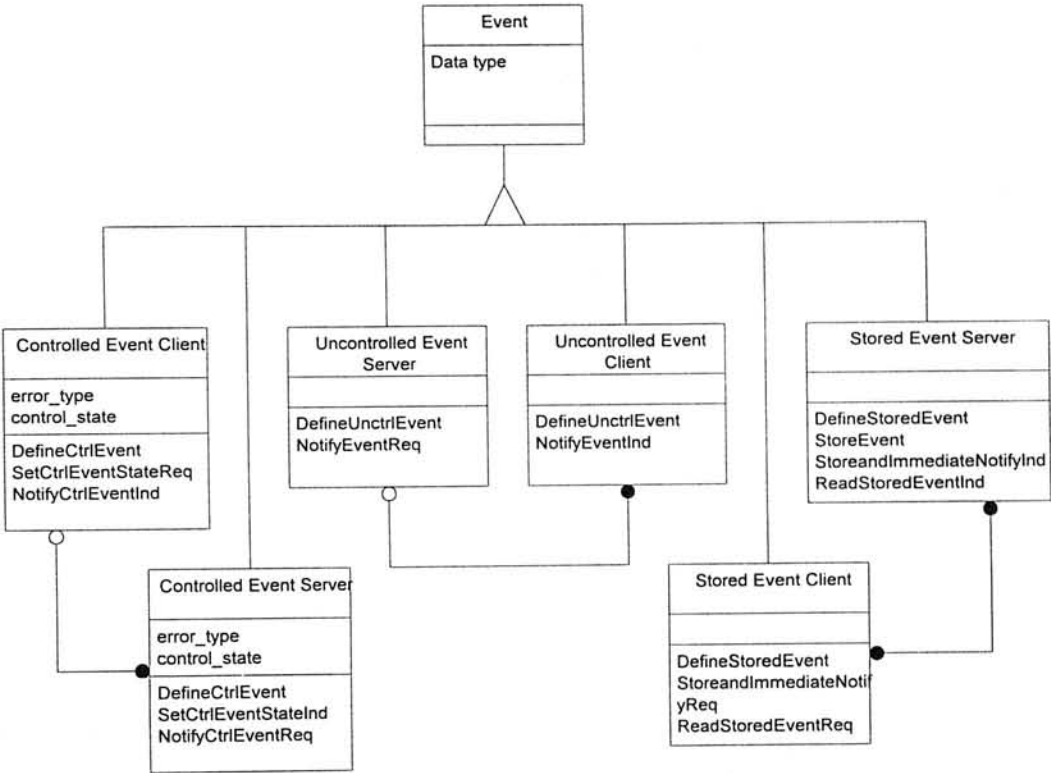


Figura 43 - Especialização do diagrama de objectos para eventos.

A figura 44 representa o diagrama de estados de um objecto **StoredEventServer**. Tal como todos os objecto CMS o processo de inicialização é idêntico. Ao receber o evento de *start* passa para o estado preparado a partir qual pode começar a enviar e receber mensagens. Os seus estados são determinados, essencialmente, pelos seus serviços.

A evocação do serviço **StoreandImmediateNotifyReq** sinaliza ao cliente a ocorrência do evento modelizado pelo objecto; é, também, da responsabilidade da aplicação a decisão de utilizar esse tipo de serviço. A aplicação pode pedir que um novo valor para o objecto seja guardado no *slot* do controlador CAN, de modo a que posteriormente, quando cliente desejar possa ler o valor do evento.

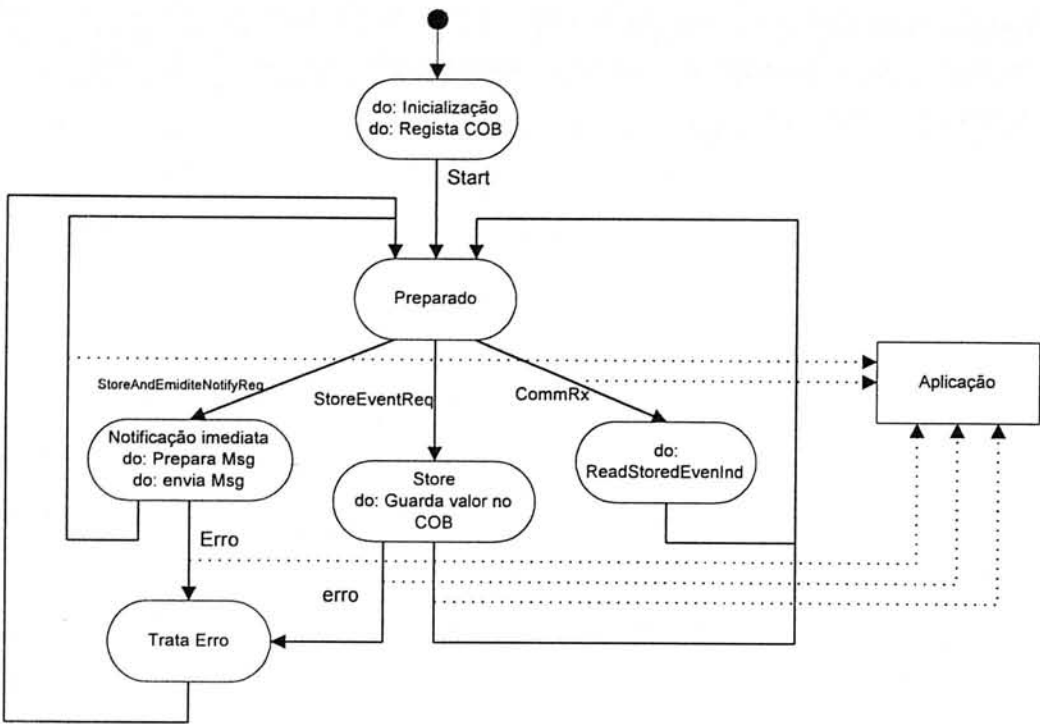


Figura 44 - Diagrama de estados **StoredEventServer**.

Um objecto **StoredEventClient**, figura 45, possibilita a um nó cliente de um evento do tipo *Stored* que estes possam comunicar. Não apresenta nenhuma característica especial que já não tenha sido referida para os outros tipos de objecto CMS.

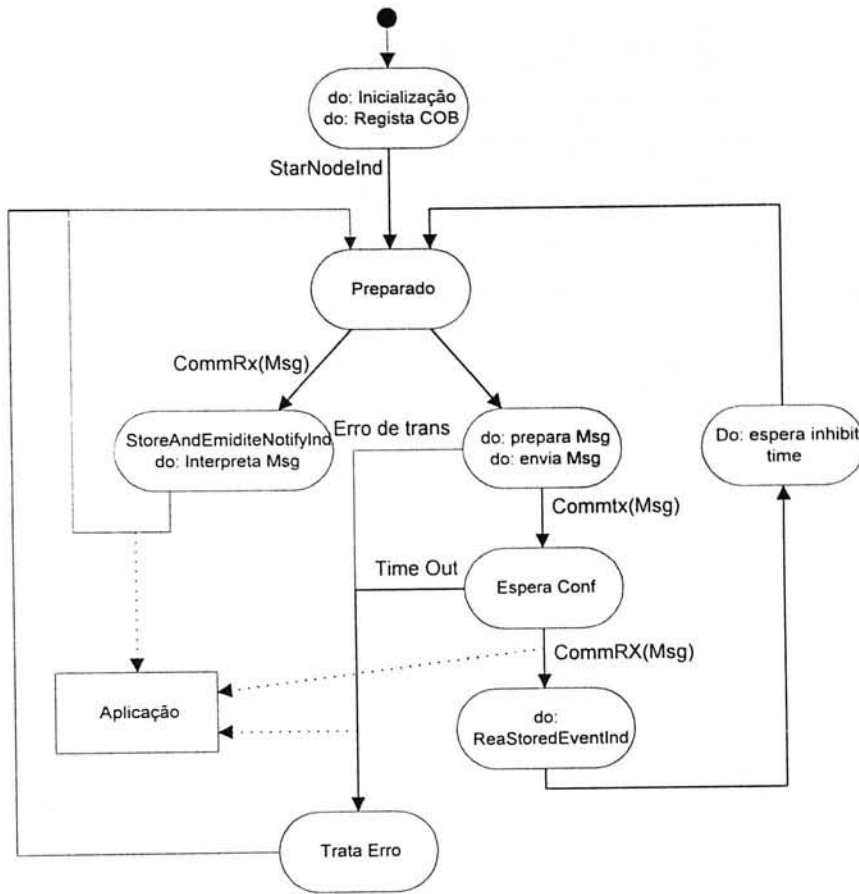


Figura 45 - Diagrama de estados **StoredEventClient**.

5.2.3 NMT

O NMT permite lidar com alguns aspectos de um sistema de controlo distribuído. Possibilita a inicialização dos nós e controla as situações de erro através do protocolo de controlo de erros - *Node Guardian*.

O modelo de objectos apresentado na figura 46 modeliza o conjunto de funcionalidades relacionadas com os objectivos do NMT.

Num sistema de controlo distribuído que utilize o protocolo CAL apenas é permitida a existência de um nó com os objectos: **NMT Master Node**, **Network** e **Remote Node**. Um nó que contenha um objecto **NMT Master Node** é segundo a norma considerado como um nó mestre NMT, o único nó capaz de controlar os erros e a inicialização da rede.

O **NMT Master Node** cria o objecto **Network** que irá fornecer à aplicação os serviços para o controlo do processo de inicialização dos nós remotos. Este objecto é constituído por um conjunto de objectos **Remote Node**, tantos quantos o número de nós da rede. Cada **Remote Node** corresponde, localmente, no nó mestre a um objecto **Node** contido num nó diferente. Note-se que um nó mestre é, também, um nó escravo e, por isso, deve também conter um conjunto **NMT Slave Node** mais um **Node**.

Na rede, cada nó tem uma atributo *NMT_adress* único que identifica esse nó na rede.

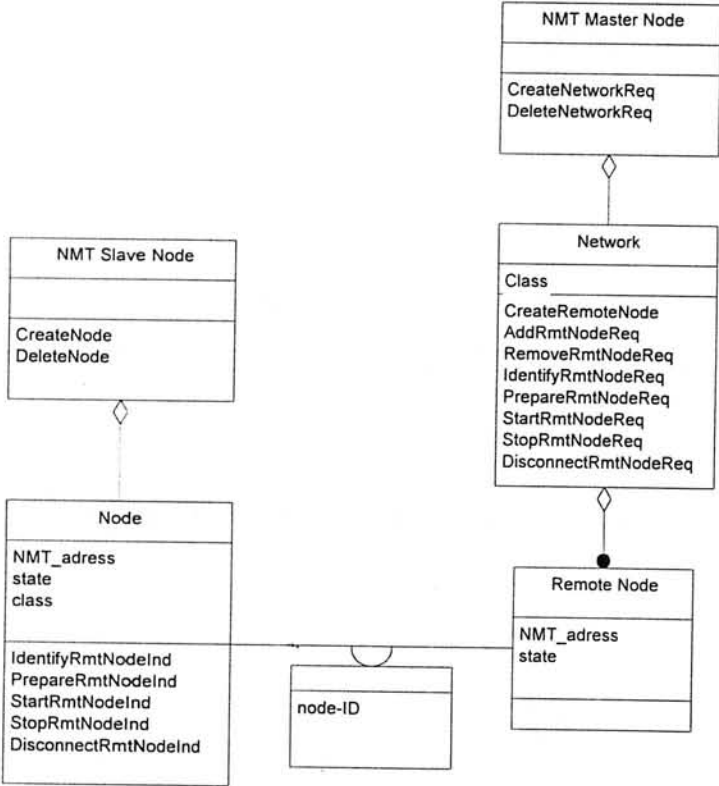


Figura 46 - Diagrama de Objectos do NMT.

O processo de inicialização da rede obriga a que os objectos **Remote Node** e **Node** passem por uma série de estados cuja transição é desencadeada pela ocorrência de eventos relacionados com os serviços.

Primeiro, é criado pela aplicação um objecto **Remote Node** com o mesmo endereço NMT que outro nó ao qual vai corresponder; o **Remote Node** passa para o estado de DISCONNECTED; neste estado o mestre NMT executa o serviço de **ConnectRmtNodeReq** durante o qual o mestre recebe o pedido do servidor com os parâmetros para o protocolo de guarda do nó, informações relativas à classe do nó e se é necessário que se efectue a transferência dos parâmetros de configuração desse nó.

A partir desse momento o protocolo de guarda do nó passa a estar activo e o nó fica no estado de CONNECTED, está representado esse facto na figura por um asterisco “*”. É durante este estado que o nó escravo correspondente pode utilizar os serviços do DBT para obter os identificadores para os seus objectos CMS.

Ao ser evocado o serviço de **PrepareRmtNodeReq**, o mestre NMT pode pedir para que sejam descartados os COBs, anteriormente, obtidos através do DBT, caso os serviços sejam um sucesso, o **Remote Node** entra no estado de PREPARED e fica a aguardar a activação da rede. Apenas no estado OPERATIONAL é que o nó pode executar os serviços CMS.

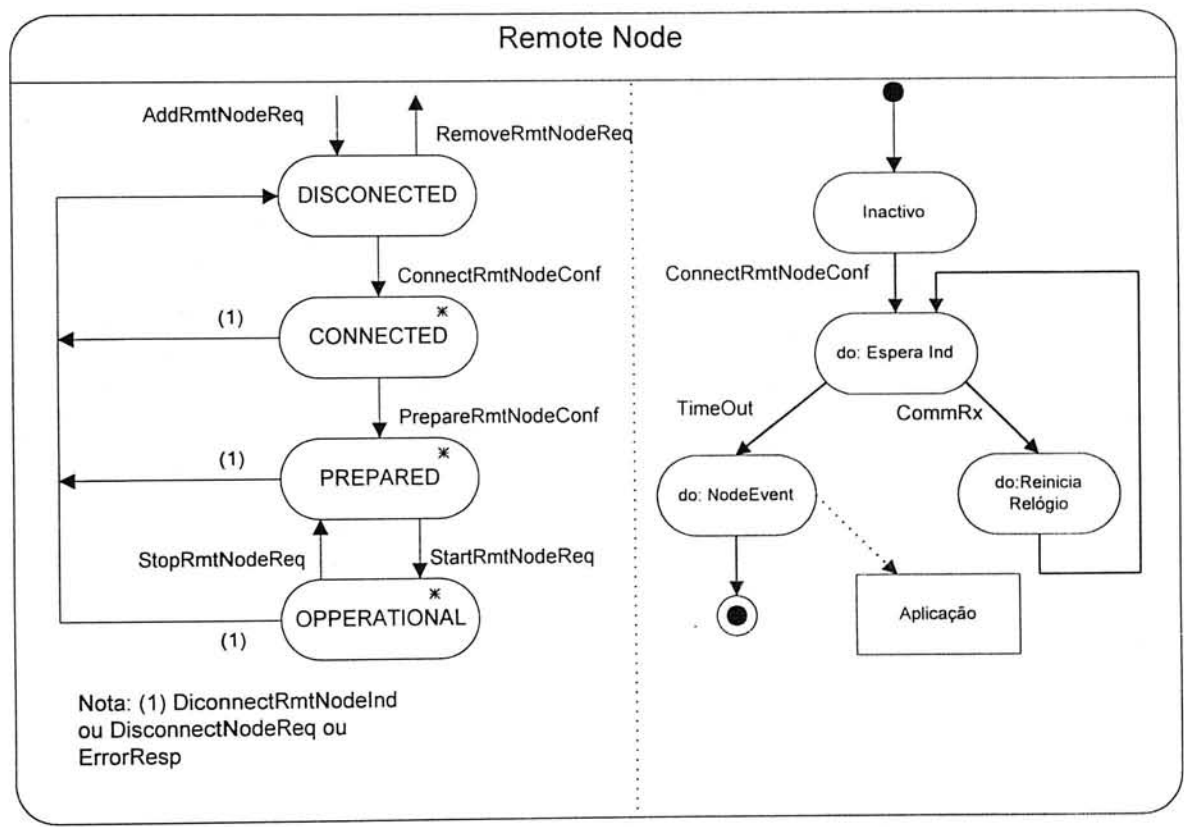


Figura 47 - Diagrama de estados de um Remote Node.

Um objecto da classe **Node** é criado num nó escravo NMT através do protocolo de **CreateNodeReq** que preenche os atributos deste objecto, nomeadamente, a classe e o NMT_Address, o nó passa para o estado de DISCONNECTED. A execução do serviço de **ConnectNode** permite passar o escravo NMT para o estado CONNECTING e indicar se será necessário que o mestre NMT faça a descarga da sua configuração. No estado de CONNECTING o **Node** está pronto para responder aos comandos que serão enviados

pelo mestre NMT. Ao responder, da forma descrita, anteriormente, a uma mensagem correspondente a **ConnectRmtNodeInd**, o nó passa para o estado de **PREPARING** durante o qual pode obter, através do DBT, os identificadores para os seus objectos CMS. A sua resposta a um **PrepareRmtNodeInd** obriga o objecto a passar para o estado de **PREPARED** e fica a aguardar a activação do nó.

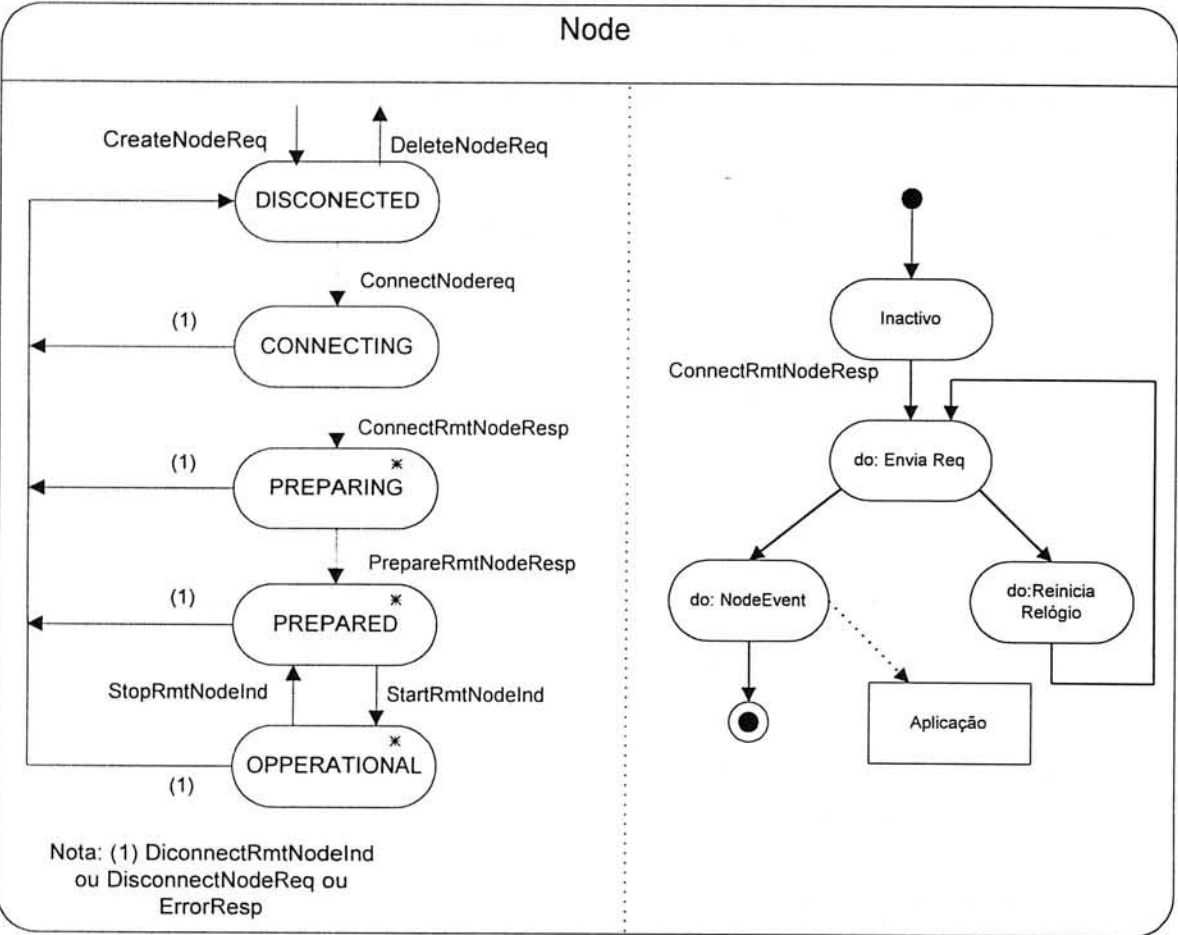


Figura 48 - Diagrama de estados de um objecto **Node**.

Os dois objectos descritos, anteriormente, **Node** e **Remote Node** contém duas máquinas de estados a correr em concorrência: a máquina de estados de controlo do nó e a máquina de estados de controlo do protocolo de guarda do nó.

O protocolo de guarda do nó permite ao mestre NMT o controlo de erros na rede enviado, ciclicamente, para todos os nós com essa capacidade, uma mensagem à qual os nós devem responder enviando o seu estado e um bit de *toggle* que deve mudar entre duas mensagens sucessivas. O período de envio da mensagem de guarda do nó é diferente de nó para nó de acordo com os parâmetros negociados durante a inicialização.

5.2.4 O DBT

O DBT é um dos elementos mais importantes para simplificar a construção de aplicações distribuídas pois permite a atribuição dinâmica de identificadores e tempos de inibição a cada um dos objectos CMS.

A atribuição do identificador para o COB é feita de acordo com o nível de prioridade pedido pelo conjunto de objectos clientes e servidores que utilizam esse COB, evitando conflitos com outros objectos.

O CAL, (DS204-1 p.2) define que conjunto de objectos utilizados para o DBT é opcional. Um módulo pode não ter capacidade de distribuição dinâmica de identificadores. Neste tipo de módulos, a configuração dos identificadores poderá ser modificada através de um outro meio definido pelo fabricante como, por exemplo, DIP-switchs.

Para que o protocolo de distribuição de identificadores funcione é necessário que exista na rede um nó mestre DBT que contenha o objectos **DBT Master Node** e respectivos objectos constituintes **COB Database**, **COB Definition**, **Predefinition** e **User Definition**. Tal como no caso do NMT um nó mestre é também um nó escravo. O nó mestre deverá responder através do **DBT Service Element** a pedidos de **CreateUserDefinitionInd**.

O mestre poderá recorrer ao serviço de **CreatePreDefinition** de modo a obrigar um nó cliente a utilizar um identificador específico. Deste modo, a aplicação poderá, ela própria, definir quais os identificadores a utilizar por cada módulo, se não forem utilizadas pré-definições, o nó mestre trata de distribuir, automaticamente, os identificadores sem que seja necessária a intervenção da aplicação no nó mestre.

A distribuição de identificadores só pode ser feita quando o estado do objecto **Remote Node** é **CONNECTED**.

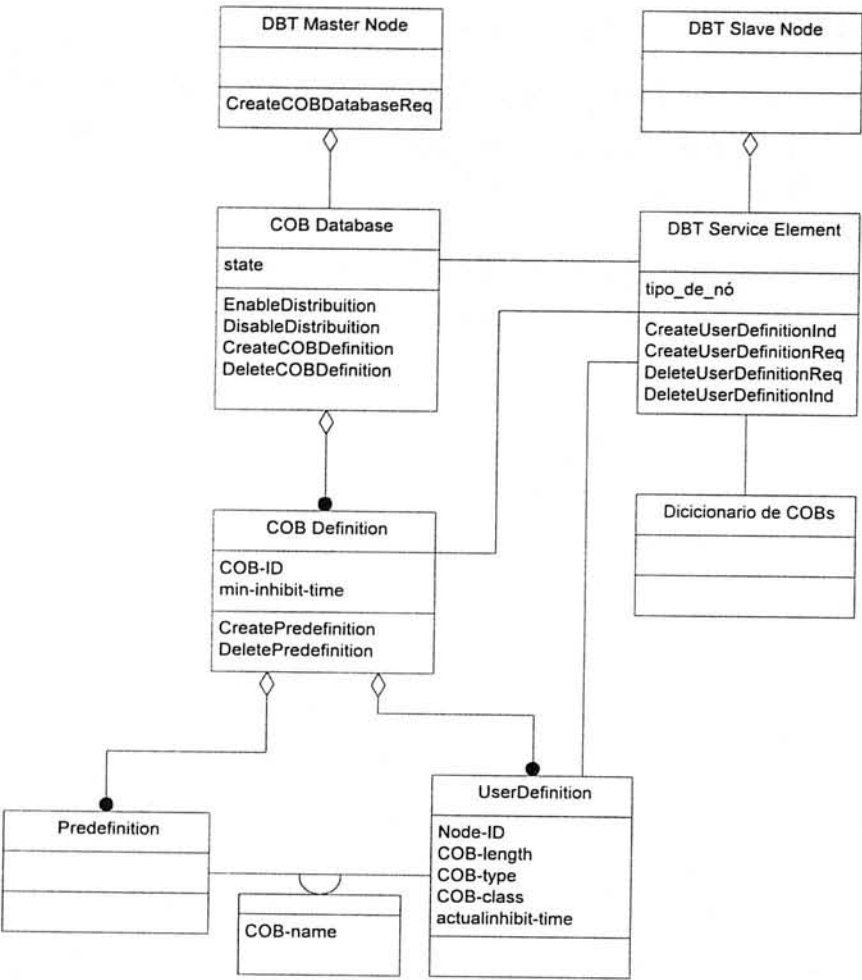


Figura 49 - Diagrama de objectos do DBT.

O diagrama anterior assim como a explicação são referentes a um mestre e a um escravo DBT apenas como capacidade de controlo da distribuição sem capacidades de controlo da consistência, (DS204-1 p. 5).

O diagrama de estados da figura 50, destina-se a um nó mestre. Um nó escravo apenas poderá utilizar os serviços de **CreatUsrDefinitionReq** e **DeleteUsrDefinitionReq**. A distinção entre um objecto **DBT Service Element** pertencente a um nó mestre e outros pertencentes a nós escravo é feita através do atributo de *tipo_de_nó*.

A interpretação da mensagem é feita de acordo com o seu primeiro byte, referido na figura 50 como cs (command specifier). Após a sua identificação pelo **DBT Service Element** do nó mestre, este passa a executar um dos serviços de **CreatUsrDefinitionInd** ou de **DeleteUsrDefinitionInd**.

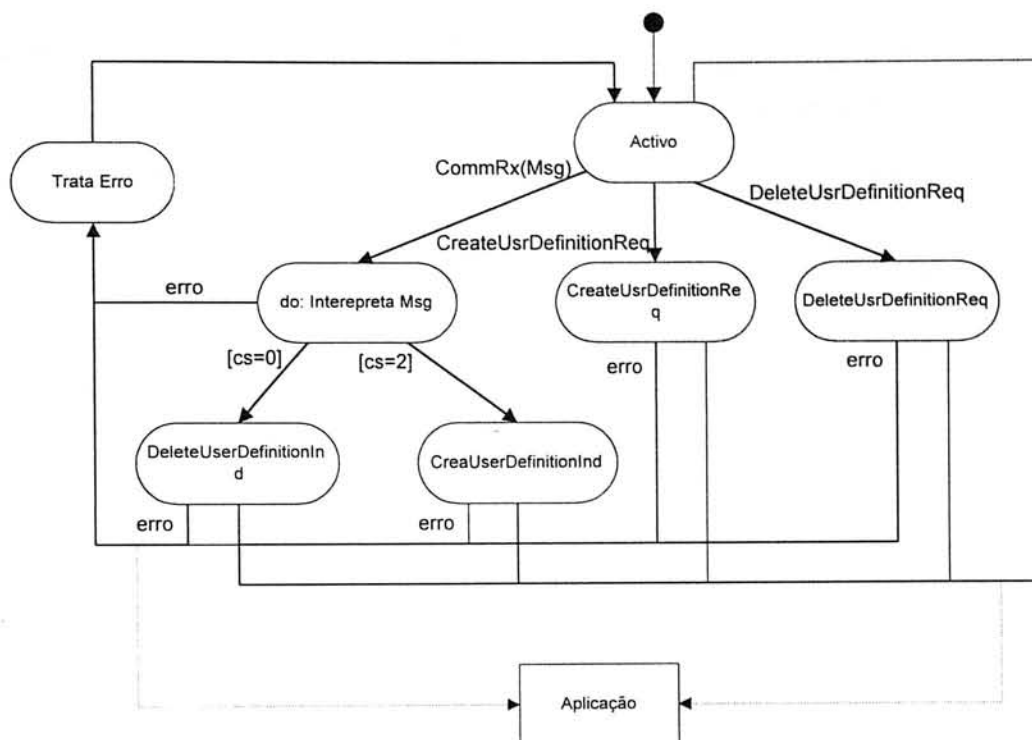


Figura 50 - Diagrama de estados do **DBT Service Element**.(geral).

De seguida, são especificados os estados de **CreateUsrDefinitionInd** e de **CreateUsrDefinitionReq**, figuras 51 e 52.

O processo de **CreateUsrDefinition** passa por várias fases:

1. transferência do COB-name para o mestre DBT
2. envio dos parâmetros do COB - tamanho, tipo, prioridade e classe
3. teste e interpretação dos parâmetros enviados
4. preenchimento da base de dados referente aos COBs.

O sucesso do serviço é definido de acordo com o especificado pela norma, (DS204-1 p. 3 a p. 7 e p.10).

De modo a que mais facilmente se possam interpretar os diagramas convém aqui referir o formato das mensagens trocadas pelo protocolo DBT. O primeiro byte da mensagem, *cs* (*command specifier*), contém a informação relativa ao comando. No caso da mensagem ser de resposta do mestre DBT aparece o referido primeiro byte de comando

seguido de um byte de *status* que indica o sucesso ou a falha do último comando enviado; o terceiro byte da mensagem indica, no caso de erro qual o seu tipo.

Um escravo DBT pode também utilizar o serviço de **DeleteUsrDefinition** para apagar uma definição na base de dados de COBs. Este serviço e as razões da sua utilização são da responsabilidade da aplicação remota.

O objecto **COB Database** apresenta um comportamento dinâmico muito simples. Pode estar em dois estados principais ENABLED e DISABLED, respectivamente, permitindo a distribuição de identificadores e não permitindo.

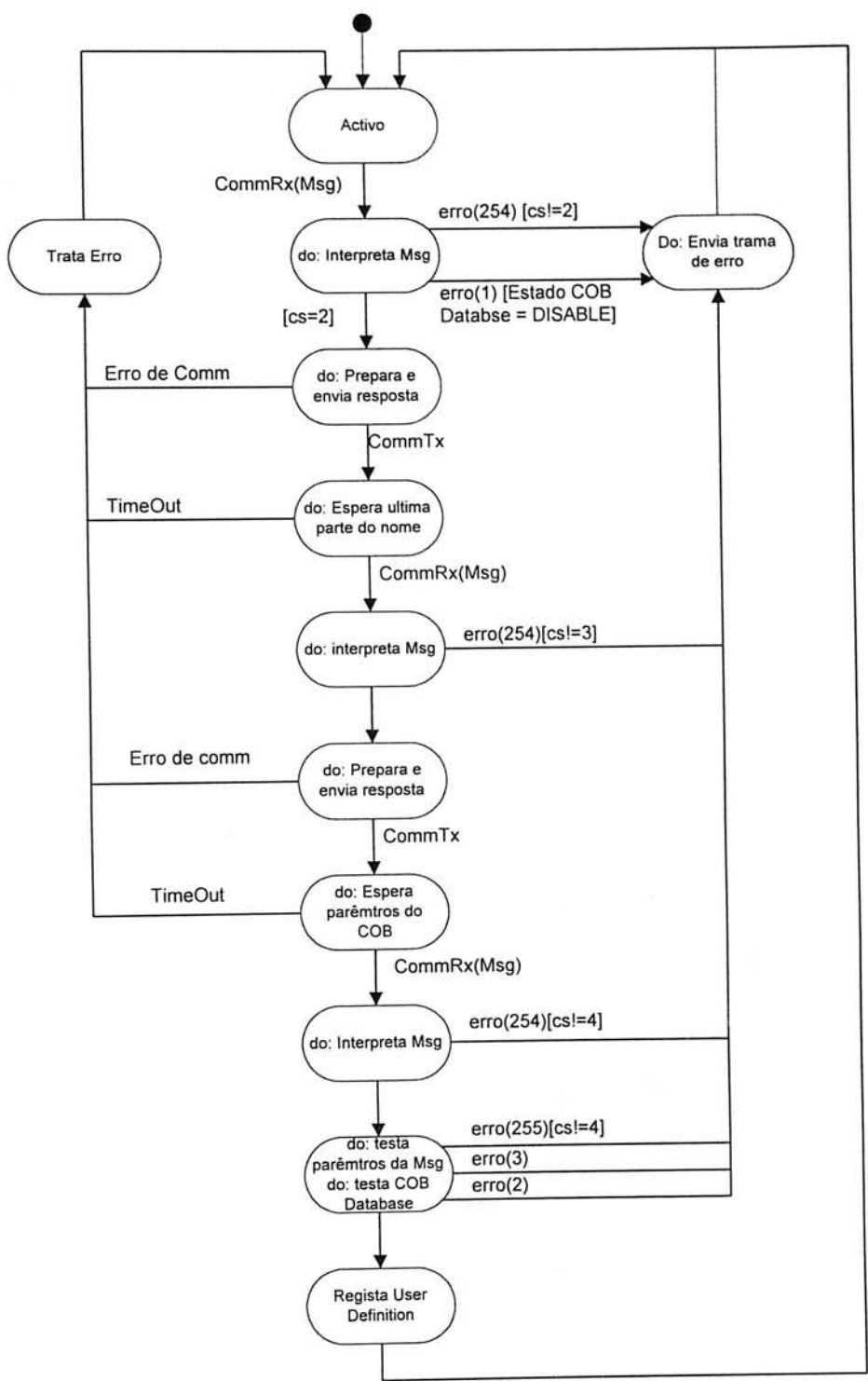


Figura 51 - Diagrama de estados do **DBT Service Element**.
(CreateUsrDefinitionInd).

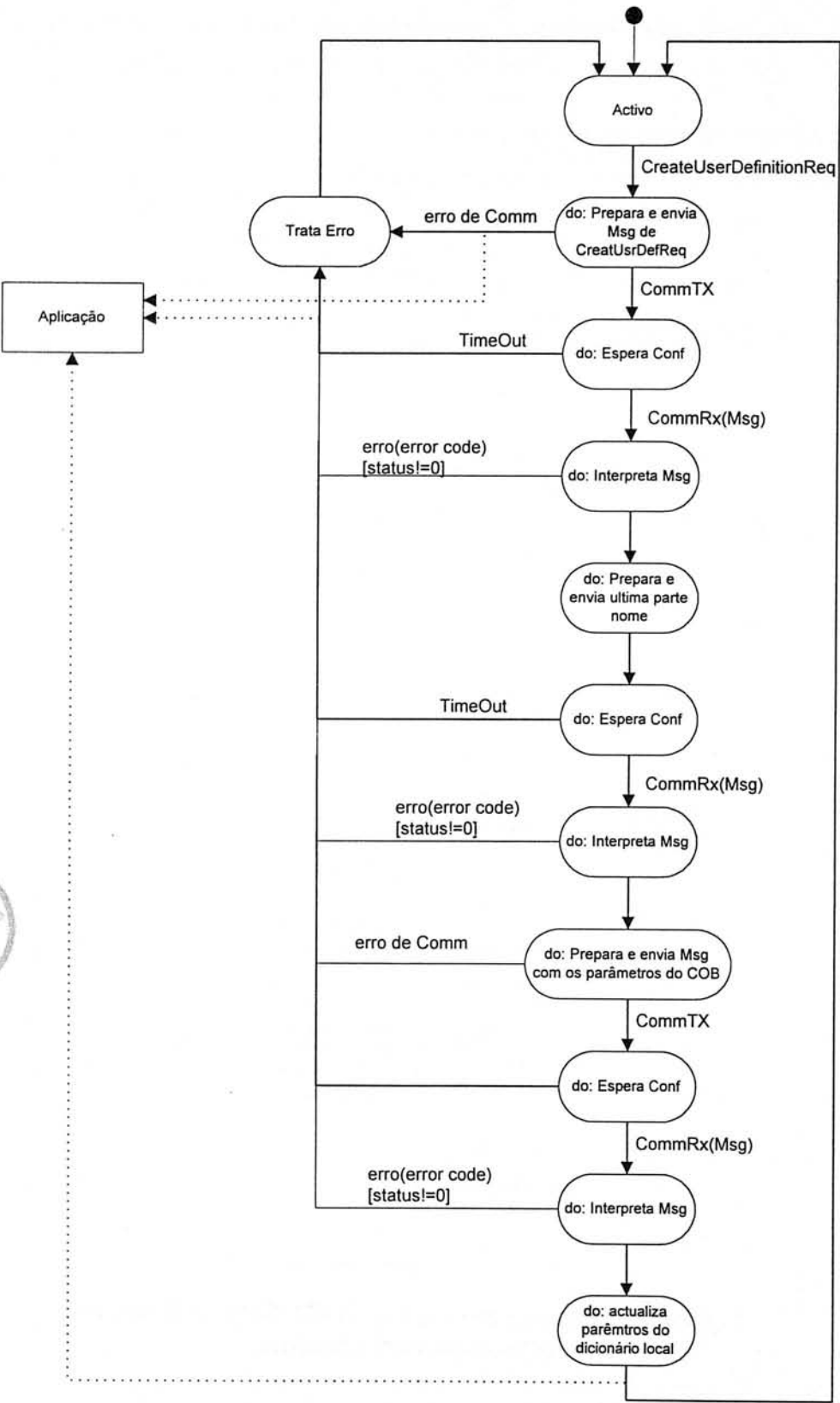


Figura 52 - Diagrama de estados do **DBT Service Element**.
(`CreateUsrDefinitionReq`).

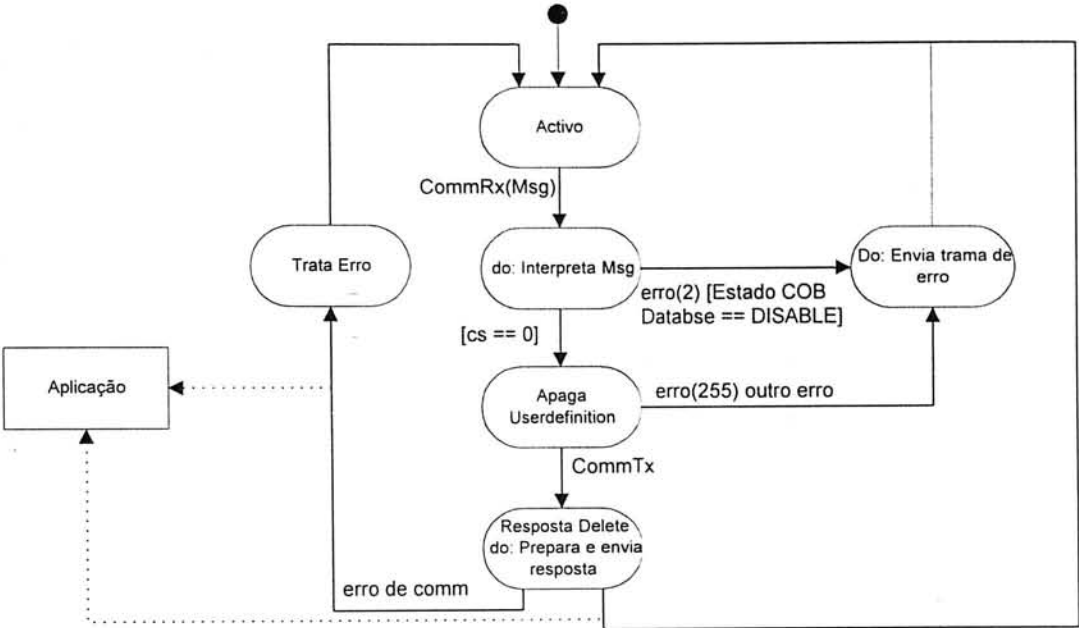


Figura 53 - Diagrama de estados do DBT Service Element.
(DeleteUsrDefinitionInd).

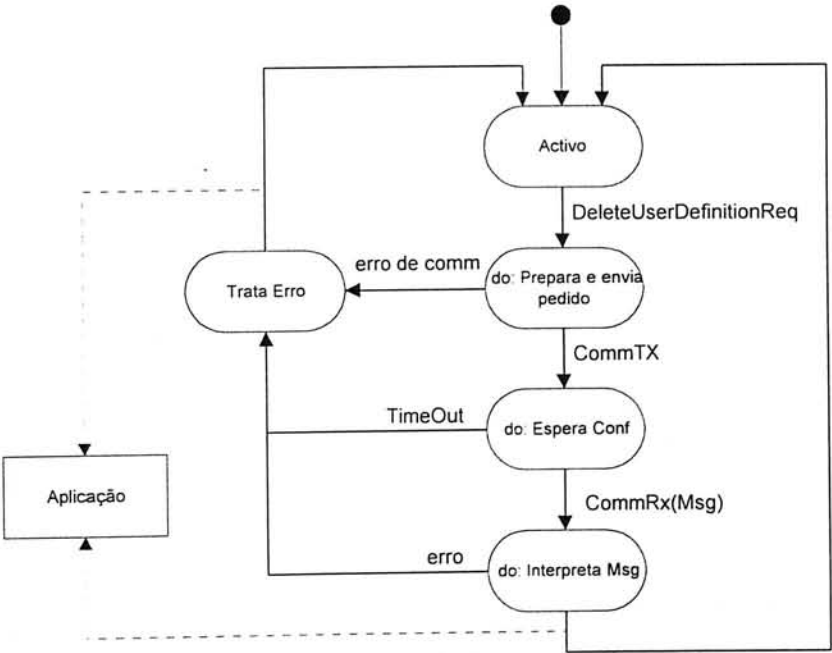


Figura 54 - Diagrama de estados do DBT Service Element.
(DeleteUsrDefinitionReq).

5.2.5 O LMT

O LMT permite a modificação dos parâmetros das camadas locais de cada módulo:

- *NMT-address* - identifica, univocamente, o módulo em toda a rede vai permitir que o módulo responda aos comandos do protocolo NMT;
- tempos de bit da camada física - permitem que a rede trabalhe com várias velocidades e com os parâmetros adaptados às condições de utilização. A activação destes parâmetros é feita de forma coordenada pelo mestre LMT;
- endereço LMT.

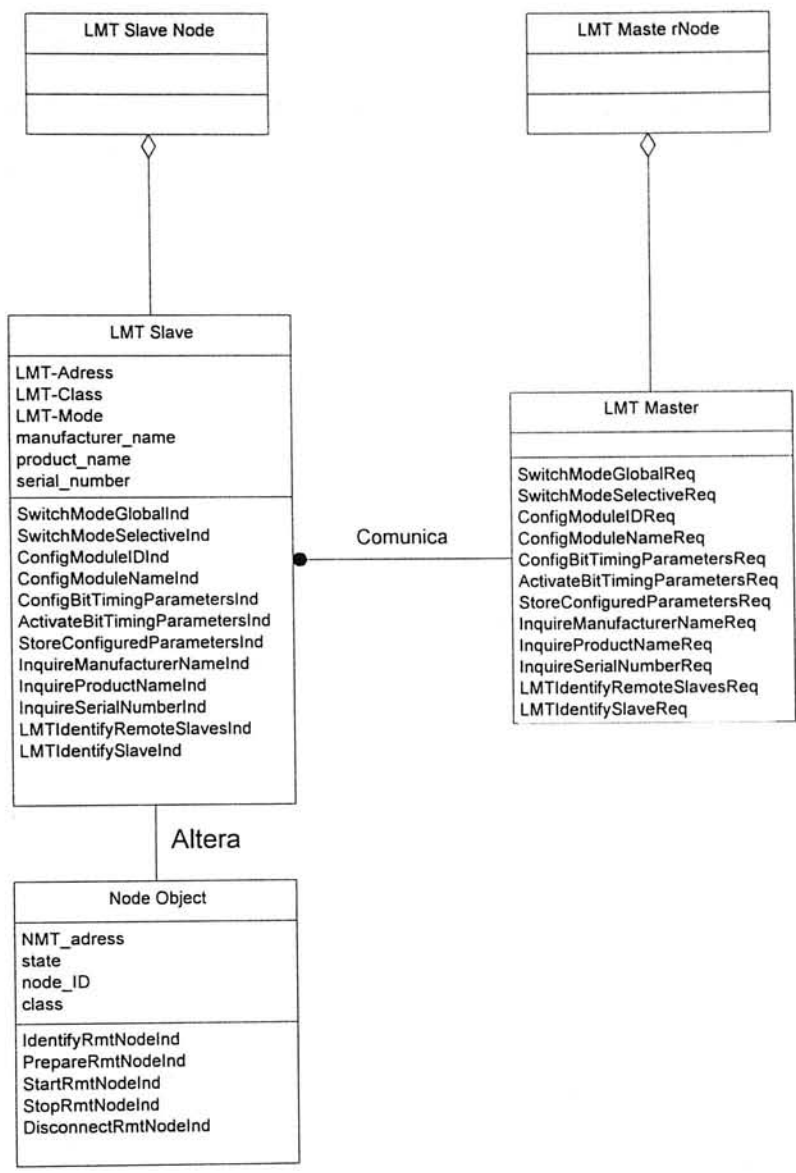


Figura 55 - Diagrama de objectos do LMT.

O diagrama de objectos contempla nós mestre LMT e nós escravo LMT com todas as capacidades. Tal como no NMT e DBT, uma rede CAL só pode conter um nó mestre LMT. Este nó pode ser um nó amovível, isto é, só é geralmente necessário que esse nó seja utilizado durante a fase de montagem da rede para configurar os nós que vão sendo instalados.

De modo a que seja possível configurar, automaticamente, cada um dos nós através da rede é necessário que cada nó tenha um endereço LMT único. A norma, (DS207 p.7), define que esse endereço é constituído pelo nome do fabricante, nome do produto e número de série. É a partir destes atributos que se identifica univocamente um nó e se pode proceder à sua configuração.

No diagrama da figura 56, relativo a um **LMT Slave** é necessário referir a existência de dois estados principais: OPERATION MODE e CONFIGURATION MODE. Durante estes estados o módulo esta respectivamente em operação, já configurado ou em modo de configuração.

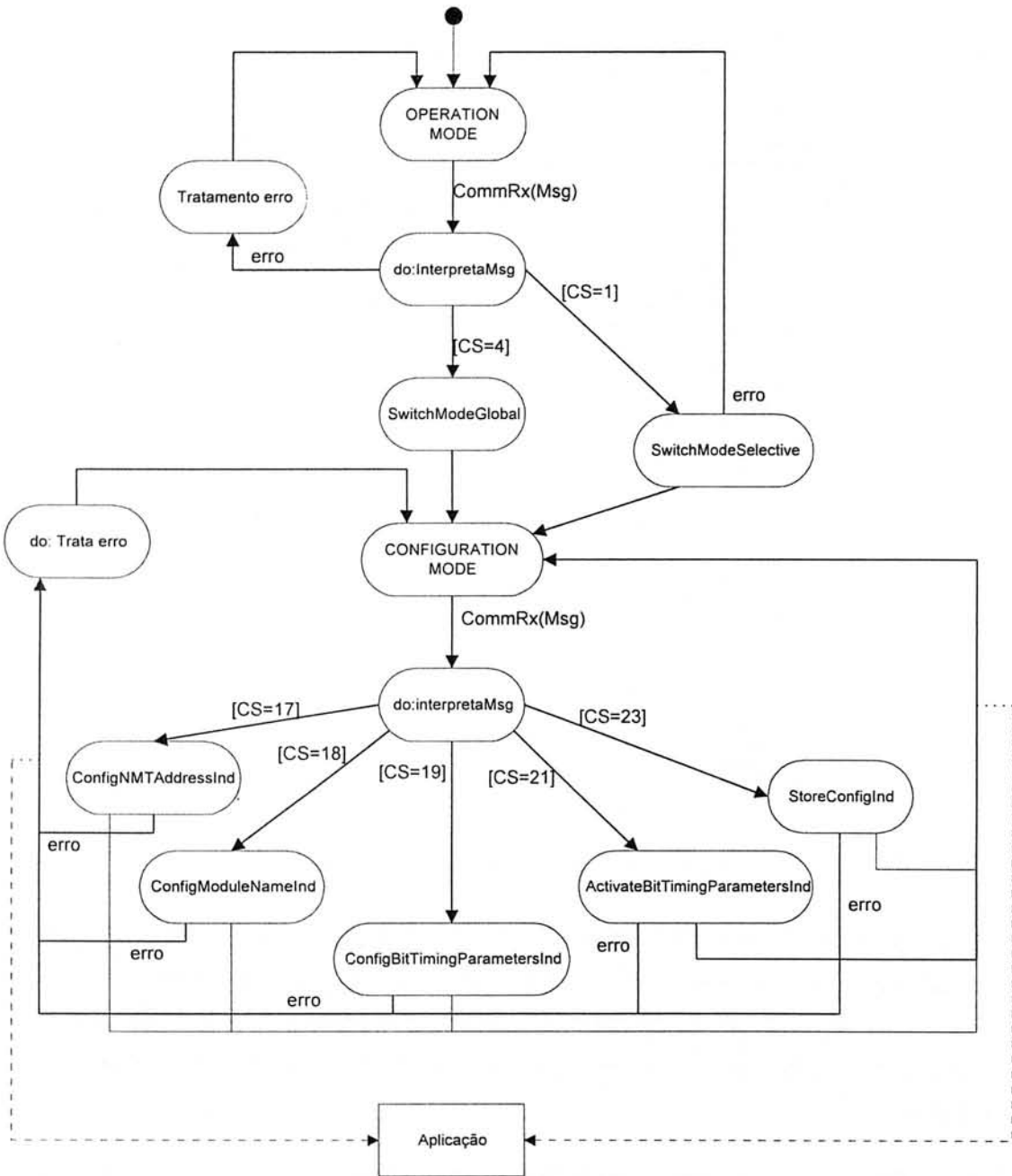


Figura 56 - Diagrama de estados do objecto **LMT Slave**

5.3 Considerações finais

A apresentação da análise não é de forma alguma exaustiva. Tenta-se focar de forma aprofundada os pormenores mais importantes do CAL, deixando algumas partes com um elevado nível de abstracção.

Tem como objectivo servir de base para uma implementação parcial do CAL não orientada aos objectos.

6. Implementação do CAL

Este capítulo irá descrever um dos métodos possíveis para a implementação da camada de aplicação genérica: *CAN Application Layer* (CAL).

Existem vários métodos de comunicações possíveis para a recepção e para a transmissão de mensagens, estes métodos serão avaliados.

6.1 Implementação de uma análise OO utilizando uma linguagem estruturada

A análise do CAL orientada ao objecto efectuada no capítulo anterior fornece uma descrição detalhada do protocolo que serviu de base à implementação. Dado que esta foi realizada utilizando a linguagem C foi necessário fazer algumas adaptações de modo a utilizar, na implementação, a modelização feita.

A inexistência de compiladores de linguagens OO para a maioria dos microcontroladores inviabiliza a implementação puramente OO. As implementações OO com características tempo real, como o presente caso, padecem de várias desvantagens:

- Código menos eficiente: deve-se ao comportamento dinâmico dos objectos, à comunicação entre objectos e a uma gestão de memória mais complexa.

- Maior necessidade de recursos: de modo a poder gerir o comportamento dinâmico dos objectos e o seu encapsulamento.
- Tempos de execução dificilmente previsíveis: o número de tarefas necessárias para a gestão dos objectos leva a tempos de execução pouco previsíveis e de grande variação.

A implementação de uma análise OO utilizando uma linguagem estruturada tal como o C, obedece a alguns passos [Rumbaugh, 91] :

1. Transformar as classes em estruturas: as estruturas em C permitem guardar os atributos das classes.
2. Adaptação da passagem de argumentos para os métodos: deve ser passada a cada método os objectos sobre os quais este vai operar. Os restantes argumentos obedecem às regras normais do C.
3. Alocação de objectos: devem preferencialmente ser alocados estaticamente durante a compilação. Caso o número de instâncias seja desconhecido então a alocação deve ser feita dinamicamente durante a execução.
4. Implementação de herança: deve-se tentar evitar a sua utilização. Pode-se re-analisar o diagrama de objectos de modo a diminuir o número de classes derivadas. Na prática é usual recorrer à utilização de estruturas diferentes para cada especialização da classe.
5. Implementação de polimorfismo: deve-se evitar o seu uso. Caso seja necessário o polimorfismo pode ser resolvido durante a compilação ou detectando o tipo de objecto sobre o qual se quer operar.
6. Implementação de associações podem ser facilmente mapeadas através de ponteiros.
7. Implementação de concorrência: a concorrência não é suportada directamente pela linguagem mas sim pelo sistema operativo que pode disponibilizar serviços para a gestão de tarefas.
8. Encapsulamento das classes: obriga que cada classe seja implementada num módulo separado, evitando o uso de variáveis globais.

Os passos descritos acima obrigam a algumas modificações no modelo de objectos de modo a adapta-lo à implementação. Por exemplo, no caso das variáveis básicas optou-se por utilizar uma estrutura comum na qual são guardados todos seus parâmetros herdados dos objectos **CMS Obj.** e **Variable**.

6.2 Estrutura do protocolo de comunicações

O modelo de comunicações utilizado pelo CAL baseia-se numa arquitectura por camadas das quais a norma [CAL, 96] apenas define as funcionalidades da camada de aplicação, os serviços e o formato das tramas trocadas entre dispositivos, sendo a implementação da camadas inferiores deixada a cargo do utilizador excepto partes da camada física e de transferência de dados definidas pela norma ISO/DIS11898 [CANSpec, 91].

A implementação é estruturada em 3 camadas (figura 57):

- **Hardware:** esta camada representa o controlador CAN. O tipo de controlador a utilizar pode ser muito variado: Siemens C167C, Intel 82527, Intel 82526...
- **DLL:** camada de ligação lógica, esta camada contém os serviços necessários para enviar, receber mensagens e configurar o controlador. Fornece à camada de aplicação um conjunto de serviços básico para a transmissão e recepção de tramas independente do hardware utilizado.
- **CAL:** disponibiliza para a aplicação os serviços descritos no anexo A.

Para a camada DLL foi utilizado com adaptações o software Vm0 fornecido pela I+ME. Esta camada também está estruturada em outras três subcamadas que serão explicadas no ponto 6.2.2.

A camada de aplicação CAL foi implementada baseada na análise efectuada no capítulo anterior, em que o objecto Vm0B representa a interface com a camada de ligação lógica DLL.

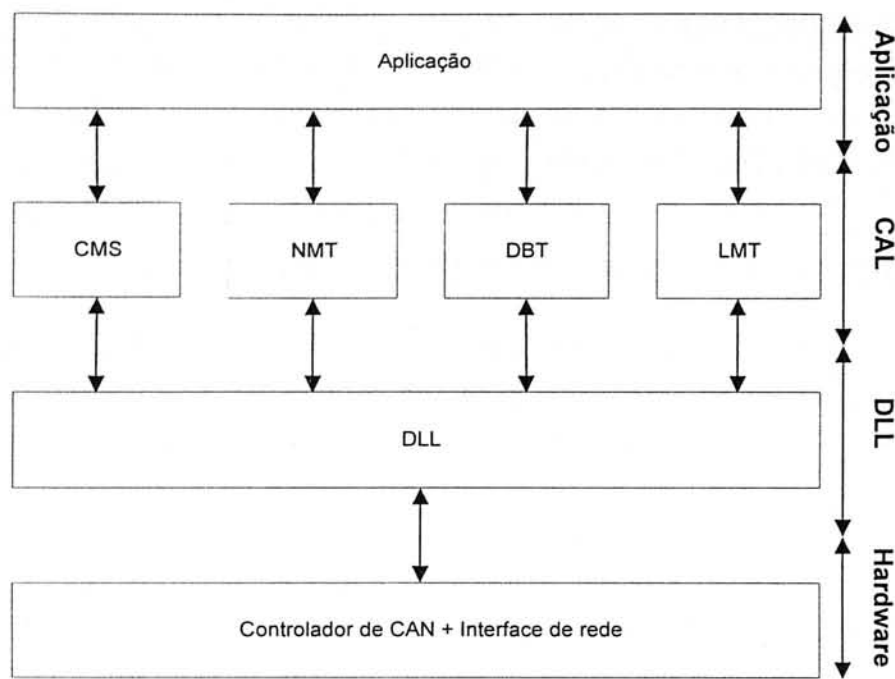


Figura 57 - Camadas necessárias para a implementação do CAL.

Os dois tipos de plataformas utilizadas foram um computador PC com uma placa de CAN utilizando o controlador 82527 e dois nós baseados no C167CR. No caso dos PCs o software seria desenvolvido para DOS e no caso do C167CR não se utilizou qualquer sistema operativo.

6.2.1 Relação com o modelo OSI

As comunicações utilizando o CAL são fundamentalmente orientadas à mensagem, isto é na maior parte dos serviços os dados não são enviados directamente para aplicações em nós específicos, servindo o identificador para identificar qual o conteúdo da mensagem. Cada aplicação presente em cada nó decide que mensagens é que quer consumir. A utilização de um modelo de comunicações orientado à mensagem permite uma utilização mais eficiente dos controladores de rede CAN. As mensagens são sempre trocadas entre cliente(s) e servidor(es).

Para a modelização das comunicações a norma CAL utiliza o mesmo tipo de primitivas que o modelo OSI, de acordo com a figura 58.

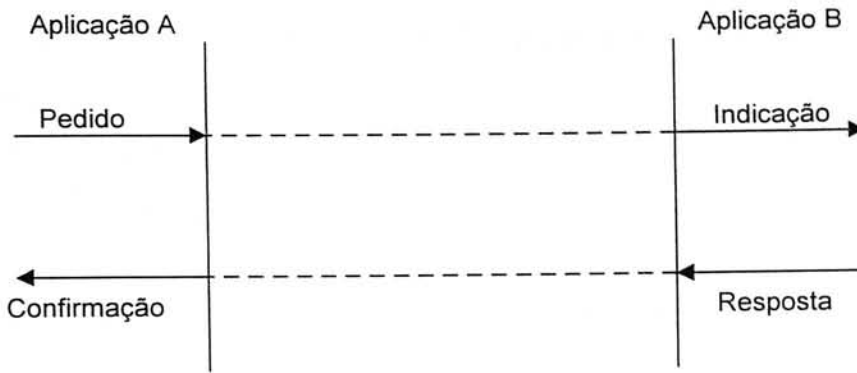


Figura 58 - Primitivas utilizadas pelos serviços

- Pedido - um pedido é lançado pela aplicação para requerer um serviço local ou num nó remoto
- Indicação - notifica a aplicação da ocorrência de um pedido para que seja executado um serviço.
- Resposta - representa a resposta de uma aplicação a uma indicação.
- Confirmação - reporta à aplicação o resultado de um pedido enviado anteriormente.

Os serviços podem ser do tipo:

- Locais - o serviços não envolve comunicação com os outros nós do sistema.
- Confirmado - um cliente envia um pedido ao qual o servidor deve responder enviando o resultado desse pedido
- Não Confirmados - este tipo de serviços é iniciado exclusivamente por iniciativa do cliente
- Iniciados pela camada de aplicação - a elemento de serviço local detecta um evento e notifica a aplicação da sua ocorrência.

Para a modelização das suas funcionalidades norma OSI também define os chamados Elementos de Serviço da Camada de Aplicação (*Application Layer Service Elements*), modelizados como objectos, estes elementos definem um conjunto de funcionalidades independentes, o CMS, o NMT, o DBT e o LMT que são referenciados no anexo A.

6.2.2 Descrição da Camada de Ligação Lógica

A camada de ligação lógica é uma adaptação do software Vm0 (*Virtual Memory level 0*) fornecido pela I+ME. Este software está destinado a ser utilizado em diversos tipos de microprocessadores⁴ escondendo as diferenças do hardware entre cada uma das plataformas. Este facto permite a estruturação das camadas de software superiores de forma idêntica para os diferentes tipos de plataformas, assim como das aplicações.

Este software está estruturado em três subcamadas:

- HSL - *Hardware Specific Layer* - disponibiliza funções para aceder aos vários registos do controlador, funções para aceder e controlar os *timers* e temporizações do microprocessador assim como outras funções mais específicas do *hardware*.
- CCM - *Communication Chip Management* - implementa funções que permitem receber e transmitir mensagens assim como efectua a gestão de erros de comunicação. A sua interface exterior é independente do tipo de controlador CAN utilizado.
- Vm0 - permite a utilização da camada CCM utilizando uma filosofia de comunicação baseada em descritores de mensagens (*handle*).

Esta *software* sofre de algumas deficiências para a aplicações com grandes taxas de transferência dado que não implementa filas transmissão e de recepção com profundidade suficiente, além disso é lento para utilização em aplicações com grandes exigências de tempos real, como por exemplo as postas pela transferência de mensagens com um *baud rate* de 1Mbit/s.

A camada foi adaptada de modo a corrigir as deficiências descritas, foram implementadas funções para a gestão de filas de recepção e transmissão e parte do código foi optimizado retirando as partes redundantes para a implementação do CAL. Esta adaptação da camada passou a ser referência como Vm0B.

Os serviços disponibilizados permitem:

- Configurar a velocidade de transmissão e os parâmetros de transmissão e recepção das mensagens.

⁴ Processadores 80c31, 80c592, 80c196, 80c166, 80c167, etc e em plataformas PC desde que munidas de placas com um controlador CAN.

- Inicializar as camadas de software e reservar espaço para os objectos de comunicações.
- Configurar os 15 objectos de comunicação disponibilizados por cada um dos controladores (167C e 82527).
- Instalar funções para responder às interrupções devidas: a erros, devidas ao fim da transmissão de uma mensagem e recepção de um mensagem.
- Transmitir e receber tramas normais e remotas.
- Controlar e monitorar o funcionamento do controlador da rede CAN.

6.2.2.1 Camada CCM e HSL

A camada HSL disponibiliza 3 tipos de serviços: acesso ao controlador CAN, gestão do relógio do sistema, gestão básica de interrupções de recepção, transmissão e devidas a erros.

A camada CCM oferece serviços que permitem gerir o funcionamento do controlador CAN independentemente do seu tipo. Disponibiliza serviços de inicialização da camada assim como serviços para atendimento de interrupções devidas à transmissão e à recepção de mensagens. Ao utilizador da camada é apresentada uma interface baseada em descritores de mensagens (*handles*), os quais identificam univocamente cada um dos objectos de comunicação utilizados.

- Recepção de mensagens pelo camada CCM

A camada CCM permite a recepção de mensagens através de *polling* ao controlador CAN ou a partir da geração de uma interrupção. Quando a recepção de mensagens é feita por interrupção, o caso mais habitual, o CCM chama uma função definida pelo utilizador.

Caso ocorra um erro durante a comunicação também pode ser chamada uma outra função definida pelo utilizador para gerir a ocorrência de um erro.

- Transmissão de mensagens

Para a transmissão de mensagens é apenas usado um objecto de comunicações. Cada uma das mensagens que se quer transmitir é colocada numa fila, de um elemento,

implementada por software. Quando uma aplicação quer transmitir uma mensagem chama o serviço de `Vm0Tx` que verifica o estado do *slot*, se este estiver livre coloca a mensagem para ser enviada se este está vazio a mensagem é guardada numa fila à espera que o *slot* esteja vazio e possa ser efectuada a transmissão.

6.2.2.2 Camada *Vm0B*

Esta camada é uma adaptação da camada *Vm0* da I+ME e destina-se solucionar alguns problemas apresentados acima, nomeadamente implementa filas de transmissão e filas de recepção e possibilita a gestão de confirmações e de indicações sem as identificar.

Esta camada implementa algumas das funcionalidades que estavam atribuídas ao objecto **Gestor de Mensagens** nomeadamente a gestão da fila de transmissão e de recepção.

6.3 Análise dos métodos evocação de pedidos e recepção de confirmações

- **Serviços síncronos**

Caso o serviço seja do tipo confirmado, o pedido é enviado e a aplicação fica bloqueada à espera que chega a resposta.

Por exemplo, o serviço de **ReadBasicRWVarReq** para ler o valor de uma variável envia o pedido e fica bloqueado à espera que o servidor dessa variável lhe envie a resposta o respectivo valor, após o que a serviço actualiza o valor da variável local e pode actuar de acordo com o valor recebido.

Neste caso convém prever um tempo máximo para a recepção da resposta de modo a que a aplicação não fique bloqueada à espera de uma resposta.

- **Serviços assíncronos**

O serviço evocado envia uma mensagem com o pedido e configura a tarefa de atendimento de mensagens de modo a que esta chame uma função definida pelo utilizador assim que chegar as resposta.

Este método permite libertar a aplicação enquanto não chega uma resposta para o pedido enviado, pode no entanto obrigar a aplicação a gerir a validade do valor do objecto quando o quer utilizar.

6.4 Análise dos métodos para a recepção de indicações e geração de respostas

Existem várias possibilidades para recepção de indicações. O método escolhido depende essencialmente do tipo e das capacidades do sistema operativo utilizado.

- **Colocação em buffer**

Este método permite o processamento de uma mensagem em tempo diferido.

Quando um nó está configurado para o consumo de determinada mensagem ele recebe a mensagem e coloca-o num *buffer* à espera que a aplicação tenha tempo para o seu processamento. O nó pode colocar as mensagens que lhe são destinadas num *buffer* através de interrupção ou através de *polling* ao controlador do sistema.

Este método adapta-se sem dificuldades a sistemas operativos monotarefa e a sistemas comandados por máquinas de estados, tal como PLCs. O processamento das indicações apenas é feito quando a aplicação tem tempo sendo muito difícil de prever o tempo que a aplicação demora até efectuar a resposta à indicação.

A sua principal vantagem é a facilidade de implementação e de utilização.

- **Atendimento através de interrupção**

Cada vez que chega uma mensagem é gerada uma interrupção e durante o tempo de processamento da interrupção é feito processamento da mensagem, construindo uma resposta ou processando o conteúdo da mensagem.

O tempo de processamento deve ser bastante pequeno de modo a não afectar a recepção das próximas mensagens. Devem ser tomadas precauções de modo a evitar fenómenos semelhantes à inversão de prioridades [Tindell1, 95], deve-se evitar o processamento de uma nova interrupção antes do fim do processamento da interrupção anterior.

- **Activação de uma tarefa**

Este processo apenas se pode aplicar a aplicações a correr em sistemas operativos multitarefa.

Quando chega uma mensagem que o nó quer consumir o sistema operativo lança ou activa uma tarefa específica para o tratamento da mensagem que chegou.

Este tipo de recepção permite que exista uma tarefa específica para o tratamento de cada mensagem. É também possível que para mensagens do mesmo tipo se utilize a mesma tarefa.

6.4.1 Geração de respostas

As respostas são sempre geradas de acordo com o resultado do processamento efectuado pela rotina de indicação.

Tal como descrito o processamento de uma indicação pode ser feito de forma automática pela camada de aplicação ou pode envolver a aplicação a correr no nó.

6.5 Comportamento escolhido

O desenvolvimento deste software de comunicações tem como principais condicionantes que o desenvolvimento se faça utilizando um sistema operativo monotarefa tal como o DOS ou então que o desenvolvimento se faça sem utilizar um sistema operativo como é o caso de se estar a desenvolver para microcontroladores 167C.

Os objectivos subjacentes ao desenvolvimento do software são:

- Adaptabilidade a múltiplas aplicações;
- Deve funcionar em sistemas operativos monotarefa;
- Possibilidade de evoluir de modo a funcionar em sistemas operativos multitarefa;
- Possibilidade do utilizador definir o modo como são atendidas as indicações;
- Possibilidade do utilizador definir o modo como são recebidas as respostas a pedidos;
- Baixo *overhead* para o processador;
- Definição dos serviços e do funcionamento de acordo com a norma CAL e com a possibilidade de ser adaptável para funcionar com dispositivos de outras normas.

Atendendo a estes requisitos optou-se utilizar a evocação síncrona de mensagens e possibilitando com pequenas alterações o funcionamento assíncrono, desde que após evocação do pedido se configure o gestor de mensagens para chamar uma função definida pelo utilizador.

O funcionamento síncrono é perfeitamente utilizável quando o sistema está submetido a uma carga não muito elevada. Vários trabalhos [Upender et al, 96] e de [Schneider, 96] simulam uma rede CAN submetida a vários tipos de tráfego e calculam os tempos de atraso a que cada mensagem pode estar sujeita.

A resposta a indicações é feita através da colocação da mensagem num buffer depois é activada uma rotina que se encarrega de retirar a mensagem do buffer e chamar a respectiva função de atendimento da indicação. Este comportamento é equivalente a enviar um evento de *CommRx* para o objecto que vai receber essa mensagem. Este funcionamento deve-se à necessidade de permitir o funcionamento em vários tipos de sistemas diferentes, desde microcontroladores de 8bits tipo 8051 passando pelas arquitecturas 16 bits e microprocessadores da família Intel 80x86.

6.6 Plataforma de Teste

Nestes trabalhos foram utilizadas duas plataformas de hardware distintas: o PC com uma placa de interface para CAN e dois nós baseados no 167CR.

A placa CAN utilizada é baseada no Intel 82527, é uma placa bastante simples que apenas mapeia os endereços dos controladores na memória do PC, o que permite o acesso aos registos do 82527 através de operações de escrita e de leitura. A placa é do tipo AT e por isso sujeita à velocidade do barramento de 4.88MHz

Os módulos contendo o 167CR apresentam as seguintes características:

- de 256K de memória RAM;
- de 256K de memória ROM;
- relógio de 20MHz;
- interface CAN ISO DIS 11898;
- interface de RS232;
- configuração através de *jumpers*.

O controlador utilizado nas duas plataformas tem uma arquitectura bastante semelhante, que será descrita no próximo ponto.

6.6.1 Controlador CAN

O microcontrolador C167CR [Siemens, 93] tem integrado um controlador CAN com funcionamento muito semelhante ao do 82527 [Intel527, 93], por isso a descrição seguinte adapta-se aos dois controladores utilizados .

Este tipo de controlador disponibiliza espaço para o armazenamento de 15 mensagens com 8 bytes de comprimento. Cada uma das mensagens tem associado um conjunto próprio de registos para configuração. Ao conjunto constituído pela mensagem e pelos seus registos chama-se de objecto de comunicação, também referido como *slot* de comunicação. Note-se que os 15 objectos não constituem um fila.

Cada objecto de comunicações pode ser configurado para receber ou para transmitir excepto o objecto 15 que apenas pode ser configurado para receber mensagens.

Estes controladores tem a possibilidade de utilizar filtros para a recepção de mensagens e deste modo o CPU apenas é interrompido quando chega uma mensagem cujo identificador o controlador foi configurado para receber.

Existe a possibilidade de serem geradas interrupções sempre que é enviada ou recebida uma mensagem e também quando se verifica um erro no controlador.

A principal desvantagem desta estrutura é que o CPU tem que retirar a mensagem que chegou a cada *slot* antes que chegue outra mensagem dado que não são implementadas filas de mensagens.

6.7 Serviços escolhidos

Um dos principais objectivos desta tese seria a desenvolvimento de uma camada de aplicação que permitisse a construção de aplicações distribuídas utilizando uma a rede CAN.

Os serviços escolhidos destinam-se a possibilitar as funcionalidades mínimas de modo a que se possam facilmente construir aplicações dedicadas e posteriormente seja possível desenvolver aplicações capazes de se integrarem em sistemas que necessitem de todas as funcionalidades disponibilizadas pelo CAL e o CANOpen.

Os serviços implementados oferecem a possibilidade de operar com os três tipos principais de objectos definidos pelo CMS e permitem a utilização do NMT.

6.7.1 Serviços CMS

- **Serviços de variáveis**

O tráfego principal em aplicações de automatização industrial é constituído fundamentalmente por mensagens curtas (máximo de 8 Bytes) [SP50, 91], serão mensagens deste tipo que irão ocupar a maior parte da largura de banda da rede de campo.

As variáveis básicas permitem utilizar as capacidades do CAN para operar com mensagens remotas assim como com mensagens normais. Os serviços implementados permitem lidar com todos os tipos de variáveis básicas de acordo com os modelos dinâmicos apresentados nos ponto 5.2.2.1

Note-se que as variáveis básicas do tipo WriteOnly são utilizadas pelo CANOpen para a transmissão de comandos entre os quais as mensagens de sincronização.

- **Serviços de eventos**

Foram escolhidos para implementação os eventos do tipo *Stored*, uma vez que são utilizados pelo CANOpen para implementar os PDO (*Process Data Objects*). Estes serviços funcionam de maneira muito semelhante à das variáveis básicas. A sua implementação não apresenta diferenças significativas em relação ao descrito anteriormente no ponto 5.2.2.4.

- **Serviços de domínios**

Foram implementação domínios básicos e multiplexados.

6.7.2 Serviços do NMT

Os serviços NMT têm as funções de controlo do funcionamento da rede, de gestão de erros e de descarregar a configuração para os módulos. Foram implementados os serviços obrigatórios, tanto para os clientes como para os servidores, de controlo da rede e de gestão de erros. Os serviços de configuração podem ser futuramente implementados utilizando os serviços de descarga de domínios básicos como base.

Os serviço de controlo da rede possibilitam a sua inicialização de acordo com o processo previsto pelo CAL.

O controlo de erros é feito utilizando as potencialidade fornecidas pela camada HSL para a gestão dos temporizadores.

6.7.3 Serviços DBT e do LMT

Dado que estes serviços não estão implementados a distribuição de identificadores deve ser feita de forma estática e perfeitamente livre, deve-se no entanto obedecer às normas impostas pelo CAL. Em cada módulo é feita a verificação de modo a que não sejam utilizados os mesmos identificadores em dois objectos diferentes. Os identificadores utilizados pelos serviços do NMT e do DBT não podem ser utilizados por objectos-CMS e encontram-se pré-definidos.

A configuração da *baud rate* da rede e dos parâmetros dos tempos de bit podem ser configurados localmente. Estão pré-configuradas, para utilização fácil o conjunto de velocidades e de parâmetros de bit definidos pelo CANOpen para a camada física. As velocidades pré-configuradas são de 1Mbit/s, 800Kbit/s, 500Kbit/s, 250Kbit/s, 125Kbit/s, 50Kbit/s, 20Kbit/s e de 10Kbit/s.

O endereço LMT e NMT devem ser especificados localmente durante a configuração da camada de aplicação.

6.8 Análise das capacidades do CAL - Medidas de velocidade dos serviços

Foram efectuadas algumas medições relativas aos tempos de serviço para os objectos CMS com maior interesse para aplicações tempo real, nomeadamente os serviços relativos a variáveis básicas e a eventos.

As medições efectuadas servem apenas como valores de referência não são valores exactos, aplicam-se ao *Kit* de desenvolvimento para o C167CR da I+ME com as características enunciadas no ponto 6.6 e PC equipados com um processador Pentium a 133Mhz.

As medições foram efectuadas utilizando no C167CR um dos seus temporizadores internos e no PC um placa AT equipada com temporizadores.

As variação encontradas nos valores nas medições de tempo no PC devem-se a:

- utilização da memória *cache*: a memória cache é mais rápida do que a memória externa do computador, no entanto nesta memória apenas são armazenadas pequenas quantidades de dados e programas, neste caso 256K. Quando os blocos de memória a que se quer aceder não estão na *cache* as operações de leitura e escrita tornam-se mais demoradas [Wilkinson, 92]. A partir do momento em que os dados já estão em memória a execução do programa torna-se mais rápida. Devido a este funcionamento é normal que os tempos de execução variem de forma aleatória.
- sistema operativo: o sistema operativo tem que desempenhar tarefas de gestão da memória, gestão dos periféricos, de interrupções, dos discos, do relógio, etc. Estas tarefas são normalmente executadas utilizando interrupções de alta prioridade que podem ocorrer durante a execução de outra rotinas.
- variações na execução do código: conforme o tipo e o conteúdo das mensagens que são enviadas os algoritmos podem comportar-se de maneira diferente, executando código diferente. Além disso podem ocorrer interrupções devidas ao protocolo de guarda do nó.
- incerteza associada ao relógio: a precisão do relógio usado era de $\pm 500\text{ns}$, a somar a este erro é necessário acrescentar os tempos de acesso ao barramento e o tempo de execução do código para parar o contador⁵.

A medição de tempo utilizando o C167CR está submetida a menores incertezas, um vez que não está ser utilizado nenhum sistema operativo. Assim as variações apenas se ficam a dever a:

- variações na execução do código
- incerteza associada ao relógio: o relógio disponível no C167CR que permite a medição de tempos com uma precisão de 400ns.

As medições foram efectuadas para os serviços de acesso a variáveis do tipo *Basic* e eventos do tipo *Stored*. Foram sempre utilizadas mensagens de 8 bytes constituídas pelo valor de hexadecimal de 55. Este valores foram escolhidos de modo a que o tempo de transmissão da mensagem não venha afectado pelo inclusão de *stuff bits*. A velocidade da rede foi sempre de 1Mbit/s.

⁵ Pode ser compensado mas deve-se ter em atenção os fenómenos provocados pela *cache*

As tabelas seguintes são um resumo dos tempos dos principais serviços, os tempos estão especificados da seguinte forma: máximo - média - mínimo. O tempo de mensagem foi calculado pelas equações (1) e (2). Foram obtidos através da transmissão de 10000 mensagens espaçadas de um tempo aleatório entre cada uma, de modo a evitar fenómenos de sincronização das mensagens com os eventos do sistema operativo.

Uma variável `ReadOnly` utiliza mensagens remotas para a execução da sua tarefa por isso a execução deste serviço é feita sem qualquer intervenção do nó remoto que apenas tem que colocar a mensagem que irá ser lida no *slot* correspondente. A medição deste tempo é feita a partir do momento em que é evocado o serviço até ao instante em chega a resposta e se finaliza o serviço. A função de confirmação apenas trata de actualizar o valor da variável.

Serviço \ Tempo	Tempo de Mensagem	Tempo Total 167CR	Tempo Total PC
ReadBasicROVarReq	155	278 - 282 - 305	218 - 222 - 330

Tabela 6 - Tempos de execução do serviço ReadBasicROVarReq (µs).

O serviço de **WriteBasicWOVarReq** não tem confirmação por isso a medição dos seus tempo revela-se um pouco mais difícil. Para a sua medição correcta seria necessário utilizar sinais exteriores. Pode-se no entanto continuar a medir o seu tempo se se alterar ligeiramente o protocolo de modo a que o nó seja ao mesmo tempo cliente e servidor da variável.

O tempo necessário apenas para executar o serviço até à sua transmissão é de 9µs e 16µs respectivamente no PC e no 167CR. Para a recepção da indicação e actualização da variável são necessários 47µs no PC e 101µs no 167CR.

Serviço \ Tempo	Tempo de Mensagem	Tempo Total 167CR	Tempo Total PC
WriteBasicWOVarReq	108	223 - 255 - 230	159 -161 -179

Tabela 7 - Tempos de execução do serviço WriteBasicWOVarReq (µs).

Tal como no caso do serviço **ReadBasicROVarReq** para este serviço foi usado o mesmo método de medição, desde o momento em que o serviço é evocado até que chegue a resposta e se saia da função.

Serviço \ Tempo	Tempo de Mensagem	Tempo Total 167CR	Tempo Total PC
ReadBasicRWVarReq	2x108	424 - 428 - 438	331 - 335 - 342
WriteBasicRWVarReq	2x108	424 - 428 - 439	331 - 335 - 342

Tabela 8 - Tempos de execução do serviço ReadBasicRWVarReq (μ s).

A implementação do serviço de ReadStoredEvent é muito semelhante ao serviço de ReadBasicROVarReq por isso os seus tempos são muito semelhantes.

Serviço \ Tempo	Tempo de Mensagem	Tempo Total 167CR	Tempo Total PC
ReadStoredEvent	155	277 - 280 - 305	217 - 222 - 329

Tabela 9 - Tempos de execução do serviço ReadStoredEvent (μ s).

Os tempos de execução dos serviços são em grande parte devidos à camada de ligação lógica. As medições efectuadas relativas apenas aos tempos de execução das funções de recepção e transmissão da camada de ligação lógica estão exposta na tabela 10.

Plataforma \ Serviço	Transmissão	Recepção
167CR	6	47
PC	16	101

Tabela 10 - Tempos de execução dos serviços de transmissão e recepção da camada de ligação lógica (μ s).

Este tempos são devidos ao grande número de acessos aos registos do controlador CAN. Este facto justifica também porque é que os tempo de execução no PC não são muito superiores ao do 167CR como seria esperar. O tempo de acesso ao controlador no caso do PC está em muito limitado pela velocidade do barramento AT.

6.9 Conclusões

A utilização da análise OO do CAL descrita no capítulo 5 permite que o desenvolvimento se faça de forma modular utilizando as estruturas descritas para os objectos e principalmente os seus diagramas dinâmicos.

A análise orientada ao objecto utilizada permitiu que a implementação fosse concebida de modo a operar em qualquer sistema desde que presentes as funcionalidades do objecto Vm0B que se encarrega de esconder o hardware das camadas superiores.

O conjunto de serviços implementados permite a operação de um sistema de controlo que utilize os serviços e os objectos do CMS assim como a maior parte dos serviços do NMT. A sua construção modular possibilita que sejam incluídas novas funcionalidades no futuro.

7. Conclusões

A rede CAN revela-se um poderoso meio para a interligação de dispositivos devido às características de tempo real que apresenta e à sua grande fiabilidade. A quantidade e a diversidade dos circuitos integrados com controlador de CAN permite a sua utilização fácil e barata em múltiplas aplicações dedicadas. Os trabalhos efectuados por vários investigadores levaram à caracterização do seu comportamento com bastante segurança, provando a sua adequabilidade para sistemas tempo real, estudando o seu comportamento em situações de falha assim como foram efectuadas simulações relativas ao seu comportamento com vários tipos de tráfego.

A análise efectuada ao conjunto de camadas de aplicação existentes permite que se considere esta rede como de largo espectro dado que pode ser utilizada em aplicações usualmente referidas como redes de sensores e actuadores, assim como em redes cujas funcionalidades já se aproximam das redes de campo. As suas capacidades permitem a sua utilização em ambientes extremamente diferenciados, por exemplo: indústria automóvel, navios, aviões, equipamentos médicos, automação industrial, barramentos de PLC, máquinas, domótica, etc.

Durante o trabalho implementou-se uma versão da camada de aplicação CAL. O CAL (*Can Application Layer*) foi escolhido por ser a camada de aplicação que apresenta maiores potencialidades industriais. Apresenta perfis definidos para várias aplicações,

disponibiliza funcionalidades variáveis conforme necessário e semelhanças com as funcionalidades preconizadas pelo MMS (*Manufacturing Message Specification*).

A análise do CAL utilizando uma metodologia OO permitiu a modelização da camada de aplicação CAL, a sua interacção com as aplicações e a definição das funcionalidades da camada de ligação lógica. A sua estrutura modular e as capacidades de organização hierárquica da informação facilitam a implementação e respectiva documentação. A análise conseguida foi adaptada para implementação utilizando linguagens não OO, obtendo-se uma implementação eficiente e que permite a disponibilizar todas as funcionalidades preconizadas.

Os serviços implementados permitem o funcionamento da rede em aplicações dedicadas envolvendo módulos utilizando o software desenvolvido assim como módulos normalizados existentes no mercado, desde que convenientemente configurados os identificadores e o endereços NMT através de outros meios.

As medições efectuadas caracterizam a implementação e levam à conclusão de que a utilização de computadores com arquitectura PC e DOS pode não ser a mais adequada para aplicações exigentes, dada a grande variabilidade dos tempos de execução das tarefas. A utilização de PC em aplicações de controlo deve ser feita utilizando sistemas operativos tempo real específicos tal como o OS-9000 ou o QNX. No entanto a utilização de PCs poderá ser bastante interessante para aplicações de monitorização e configuração da rede.

Finalizam a trabalho desenvolvido fornece uma base para investigações futuras na área dos sistemas de controlo distribuídos (DCS) fornecendo e caracterizando um dos seus constituintes fundamentais a rede de comunicações e o respectivo protocolo.

7.1 Trabalho futuro

Como trabalho futuro pode-se dividir em trabalho a curto prazo e trabalho a longo prazo.

A curto prazo:

- continuação da implementação do CAL utilizando núcleos multitarefa de tempo real;
- teste de exaustivo de todas as suas funcionalidades;
- verificação de compatibilidade com outra implementações do CAL;

- utilização em aplicações práticas preferencialmente com incorporando dispositivos de vários fabricantes.

A longo prazo:

- desenvolvimento do CANOpen baseado no CAL;
- construção de ferramentas de apoio à construção aplicações baseadas no CAL e o CANOpen nomeadamente:
 - ⇒ simulação do comportamento;
 - ⇒ aplicações de configuração da rede;
 - ⇒ aplicações de teste da rede;
- estudo aprofundado das características da rede e da camada de aplicação focando especialmente as suas características utilizáveis em sistemas de tempo real distribuídos.

A quota de mercado das redes do tipo CAN rondam, neste momento, 50% de todas as redes de campo. É com muita expectativa que se aguarda a aplicação maciça na indústria automóvel, que certamente fará descer ainda mais os preços dos integrados tornando-se cada vez uma rede de campo de eleição.

Anexo A - Descrição das camadas de aplicação baseadas no CAN

CAL - CAN Application Layer e CANOpen

O CAL [CAL, 96] foi definido pela CiA, tendo a versão 1.1 sido lançada a 1 de Fevereiro de 1996. Juntamente com esta norma foi posteriormente definido um perfil para aplicações industriais designado por CANOpen [CANOpen, 96]. A descrição que se segue é baseada nestes dois *standards* complementares.

Esta norma é baseada no modelo OSI e bastante semelhante ao MMS em termos dos vários elementos de serviço que define.

Esta camada de aplicação está estruturada em 4 elementos de serviço: o CMS (*CAN Message Specification*), o DBT (*Distributor*), o LMT (*Layer Management*) e o NMT (*Network Management*), de acordo com a Figura 59.

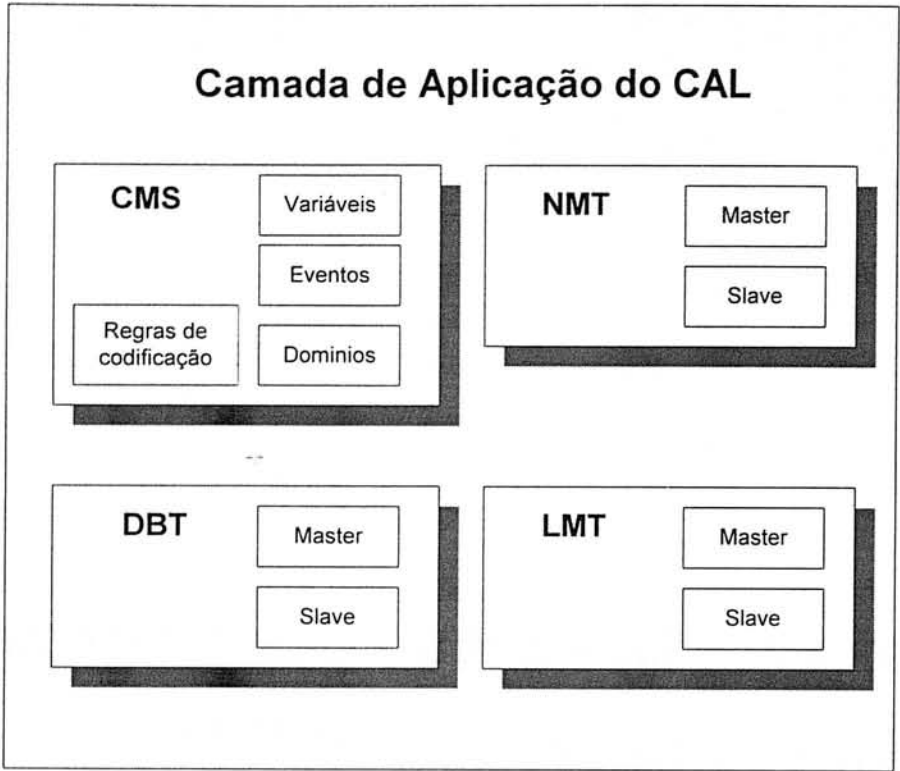


Figura 59 - Camada de aplicação do CAL.

CMS: *CAN Message specification*, oferece um ambiente orientado ao objecto destinado ao desenvolvimento de aplicações. Disponibiliza os seguintes objectos: variáveis, domínios e eventos. Define as regras de codificação e decodificação de dados assim como a maneira como cada uma das funcionalidades de um módulo podem ser acedidas através da rede.

NMT: *Network Management*, permite que o NMT Master trate da inicialização da rede e possibilita a detecção de falhas nos outros nós os NMT Slaves através de protocolos de guarda do nó.

DBT: *Distributor*, o DBT Master distribui, dinamicamente, os identificadores (COB-ID) por todos os DBT Slaves de modo que uma vez atribuída uma prioridade [0,7] pelo programador a um objecto do CMS (variáveis, eventos e domínios), o DBT trata de atribuir, univocamente, o respectivo COB-ID.

LMT: *Layer Management*, permite que um módulo o LMT Master controle alguns dos parâmetros de todos os LMT Slaves através da rede, como o *Baudrate* e o seu endereço

LMT. Permite, ainda, a mudança de estado do dispositivo para um estado de configuração.

Camada Física

O CAL define grande número de velocidades possíveis associadas a vários comprimentos de cabo, para um módulo, a saber: 1Mb/25m, 800kb/50m, 500kb/100m, 250kb/250m, 125kb/500m, 50kb/1000m, 20kb/2500m, 10kb/5000m. Para cada uma das velocidades são definidos os parâmetros relativos aos tempos de bit. É obrigatório, apenas, que um módulo suporte a velocidade de 20kb. Para grandes distâncias é recomendado o uso de acopladores ópticos.

O tipo de cabos está, ainda, por definir; apenas é definido o tipo de conectores que pode ser do tipo mini, em D, aparafusados e multipólo.

Os cabos de ligação disponibilizam dois condutores para a distribuição de alimentação de 24V pelos diferentes módulos, não estando definidos os valores máximos da corrente.

O protocolo apenas está especificado para utilização de controladores de CAN pelo menos do tipo 2.0A.

Modelo de comunicações

O CAL funciona com comunicações do tipo cliente/servidor com a possibilidade de utilização de modelos tipo produtor/consumidor, em que um nó produz um objectos CMS que pode ser consumido por outros nós configurados para os receber.

Uso do campo de identificação

O CAL apenas usa o campo de identificação para o endereçamento dos vários objectos ou mensagens. São definidos 7 níveis de prioridade para os objectos, e é da responsabilidade do DBT a atribuição dos identificadores aos vários objectos de acordo com a prioridade definida pelo utilizador.

Os restantes endereços são utilizados pelos vários serviços disponibilizados pelo CAL.

Identificador	Serviço
0	NMT <i>start/stop</i>

1-220	Objectos CMS de prioridade 0
....	
1541-1760	Objectos CMS de prioridade 7
1761-2015	Protocolo de guarda do nó.
2016-2019	Reservados
2020	Serviços do LMT
2021	Serviços do LMT
2022	NMT serviço de identificação do nó
2023	Serviços do DBT
2024	Serviços do DBT
2025	Serviços do NMT
2026	Serviços do NMT
2027	Reservado para o teste do módulo
2028-2031	Reservados

Tabela 11 - Mapa da atribuição dos identificadores do CAL.

Uso do campo de dados

O campo de dados tem a sua utilização definida pelos vários serviços, não estando normalizado.

É apenas de notar que o CMS disponibiliza funcionalidades para o empacotamento dos dados orientado ao bit, por exemplo, se se quiser transmitir uma estrutura (teste) do tipo seguinte em que a primeira variável têm 5 bits e a segunda 7.

```
STRUCT OF
    INTEGER5      a,
    INTEGER7      b
teste
```

A codificação para transmissão desta estrutura seria do tipo: no primeiro byte iriam os 5 bits da variável a, nos últimos 3 bit apareceriam os 3 bits mais significativos de b, no segundo byte seriam enviados os restantes 4 bits mais 4 bits sem significado. Esta forma de tratar as variáveis permite poupar largura de banda, no entanto obriga a um elevado esforço de decodificação por parte do receptor da mensagem o que leva a que na maior parte das implementações do CAL a parte referente ao empacotamento das mensagens possa ser desactivada pelo programador, de modo, a que este possa implementar rotinas específicas de desempacotamento de mensagens.

Serviços

Cada um dos elementos de serviço CMS, LMT, NMT e DBT contém o seu conjunto de serviços, que lhe permite cumprir as suas tarefas, nomeadamente a leitura de variáveis, a configuração dos endereços e a distribuição de identificadores.

Inicialização e configuração

O processo de inicialização (figura 60) começa quando um nó é ligado à rede e este faz as inicializações internas necessárias. Depois fica à espera que o NMT *master* lhe envie um pedido para se conectar à rede, o nó responde pedindo ao NMT *master* os tempos de inicialização do protocolo de guarda do nó, juntamente com o envio da classe do nó. Após esta primeira fase entra no estado de PREPARING, durante o qual pode obter os identificadores para os seus objectos. No final dessa fase o nó passa ao estado OPERATIONAL, durante o qual irá desempenhar as funções para as quais foi construído.

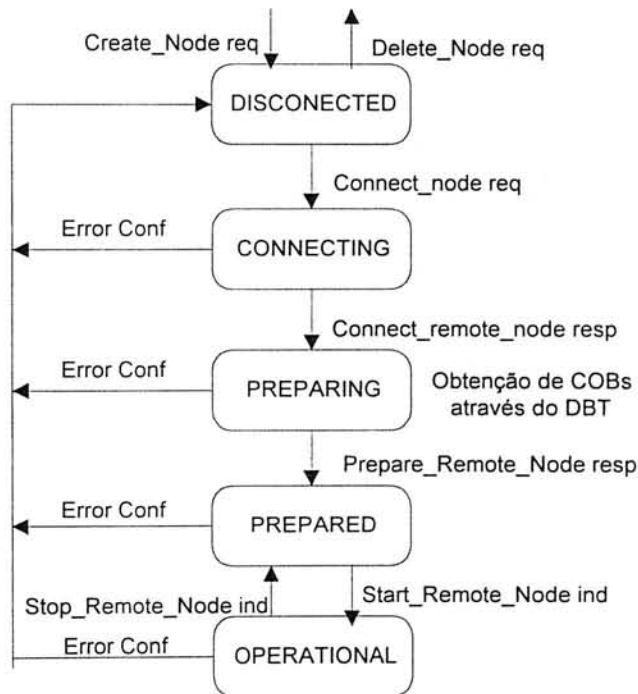


Figura 60 - Diagrama de estados de um nó remoto

Perfis de comunicação

O CAL serve de suporte a vários perfis de comunicações entre os quais se destaca o CANOpen para aplicações industriais, para aplicações em sistemas hidráulicos e para sistemas móveis.

Dado que a vertente principal desta tese serem as aplicações industriais este perfil que é aqui descrito sumariamente.

O perfil de comunicações CANOpen serve de interface entre o CAL e as funcionalidades de um dispositivo. Descreve os conceitos e modos possíveis para a comunicação tempo real e respectiva configuração, além disso, descreve alguns dos mecanismos possíveis para a sincronização entre diferentes dispositivos. O CANOpen é baseado, apenas, em algumas das funcionalidades do CAL, ou seja, não é necessário uma implementação completa do CAL para que um dispositivo CANOpen possa comunicar com outros dispositivos com o mesmo perfil.

Os perfis dos dispositivos definem as funcionalidades de cada dispositivo de acordo com a normalização imposta pelo dicionário de objectos definido pelo CANOpen. É também de referir que foi feito um esforço pelas entidades normalizadoras, neste caso a CiA de modo a que os perfis dos dispositivos se assemelhassem aos perfis de dispositivos utilizados pelo PROFIBUS e pelo Interbus-S.

O modelo de comunicações suporta a transmissão de mensagens síncronas e assíncronas.

As mensagens síncronas permitem a sincronização de vários dispositivos na rede de modo a possibilitar operações de aquisição de dados e controlo do movimento.

As mensagens assíncronas são destinadas a assinalar eventos.

As mensagens podem ser subdivididas em quatro grupos:

- Mensagens de Administração (controlo das camadas, controlo da rede, distribuição de identificadores)
- Service Data Messages
- Process Data Messages
- Mensagens pré-definidas (sincronização, Time Stamp, mensagens de emergência)

As mensagens de administração são definidas de acordo com a norma do CAL e permitem implementar as funcionalidades dos serviços que lhe estão associadas (NMT, LMT, DBT e CMS).

As *Service Data Messages* são usadas para aceder aos objectos definidos no dicionário como objectos de um dispositivo. O dicionário de objectos contém alguns dados imprescindíveis ao funcionamento do dispositivo como, por exemplo: nome do fabricante, número de série e a definição das mensagens que troca com outros dispositivos.

As *Process Data Messages* permitem a transferência de dados em tempo real.

O perfil do dispositivo descreve as funcionalidades gerais que dispositivos do mesmo tipo devem suportar. Neste momento, ainda só estão definidos perfis para dispositivos de E/S [**ProfileI/O, 95**], mas já se encontram em fase de definição perfis para *drives* de motores, actuadores hidráulicos, PLC, etc.

DeviceNet

A norma DeviceNet define as suas funcionalidades utilizando um modelo orientado para o objecto particular, isto é, não segue nenhuma das metodologias mais conhecidas. Assim, nesta norma, todos os serviços são modelizados como funções dos objectos; além das funções, cada objecto, contém, também, atributos que definem as suas características.

A descrição apresentada neste ponto baseia-se em [**DeviceNet Vol I, 96**] e em [**DeviceNet Vol II, 96**].

As principais características desta rede são:

- Possibilidade de ramos no barramento (embora com comprimento limitado).
- Número máximo de nós: 64.
- Possibilidade de remoção de um nó sem danificar a rede.
- Alimentação disponível nos cabos de ligação dos sensores, até 16A de pico 8A nominal a 24V.
- Protecção contra erros na ligação dos cabos.

- Velocidade 500Kbaud (100m), 250Kbaud (250m), 125Kbaud (500m).
- Comunicação em *Master/Slave* ponto a ponto.
- Fragmentação para o transporte de grandes blocos de informação.
- Detecção de identificadores MAC-ID⁶ duplicados.

Camada Física

A camada física desta norma prevê a utilização de uma topologia em barramento com a possibilidade de incorporar linhas de quebra (*drop lines*).

Pode operar em três velocidades e três comprimentos diferentes a saber: 500Kb (100m), 250Kb (250m), 125Kb (500m), todos os casos utilizando cabo grosso (*thick*). Caso se opte pela utilização do cabo fino (*thin*) a norma não prevê comprimentos superiores a 100m e fica também limitada a corrente máxima admissível.

Recomenda o uso dos *transceivers* PCA82C250 com ou sem isolamento óptico adicional.

A ligação do barramento aos dispositivos pode ser feita utilizando ligações directas, aparafusadas, conectores mini e conectores micro.

Os cabos de ligação permitem a alimentação dos dispositivos através de dois pinos extra de V^+ e de V^- ; a corrente nominal máxima é de 8A.

Métodos de estabelecimento de conexões

O DeviceNet define um esquema de conexões de modo a facilitar o desenvolvimento de aplicações. As conexões podem ser modelizadas através dos objectos de comunicações "*connection object*".

Existem dois tipos distintos de "*connection object*":

⁶ Esta norma chama este nome aos identificadores

- **Conexões de I/O** - Permite a comunicação entre um objecto produtor e um ou mais objectos consumidores. Cabe ao utilizador a definição do conteúdo de cada uma das mensagens trocadas.

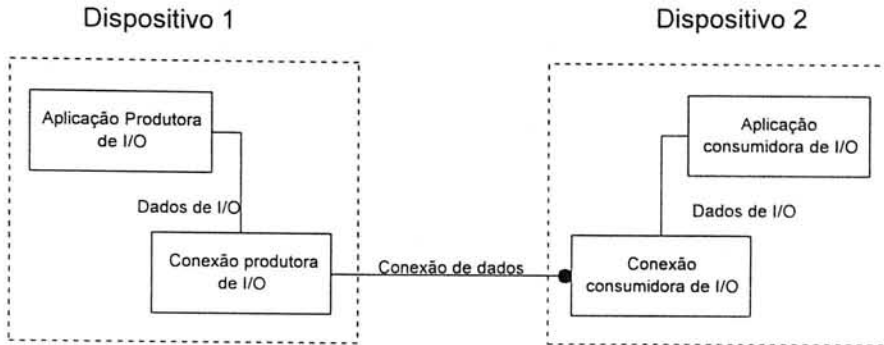


Figura 61 - Esquema OO de uma *I/O Connection*.

- **Conexão Explícita de Mensagens** - Destina-se a possibilitar um canal de comunicações genérico entre dois dispositivos. Permite a comunicação típica de pedido/resposta. Este tipo de conexão pode ser utilizado, por exemplo, na configuração de um dispositivo.

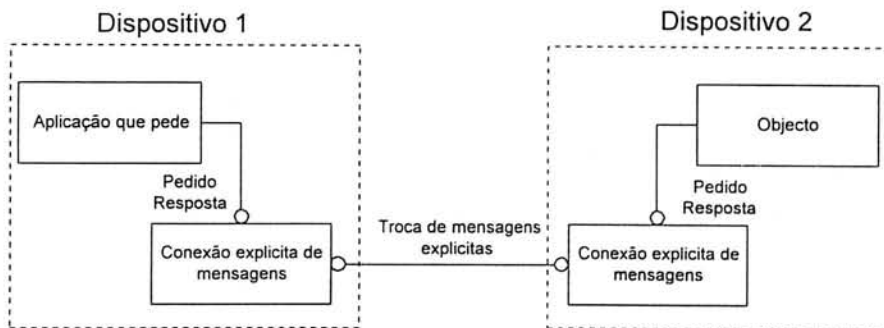


Figura 62 - Esquema OO de uma *Explicit Messaging Connection*.

Conexões Mestre/Escravo pré-definidas

O estabelecimento de Conexões Mestre/Escravo pré-definidas permite a inclusão de dispositivos simples na rede, uma vez, que são simplificados os mecanismos para o estabelecimento de conexões. Este tipo de conexão destina-se, especialmente, à ligação de sensores simples como: fins de curso, fotoelétricos, proximidade, actuadores, etc.

Utilizam para a comunicação um conjunto de identificadores dos grupos 1 e 2 com um significado pré-definido. As mensagens do grupo 1 permitem ao escravo responder a um comando enviado pelo mestre; as mensagens do grupo 2 são utilizadas pelo mestre para enviar comandos para os escravos.

Neste tipo de conexões estão definidos serviços para fazer o *polling* dos vários dispositivos, enviar comandos orientados ao bit (cada bit representa um dos dispositivos) e detecção de MAC-ID duplicados.

Os escravos respondem aos vários serviços enviando mensagens que representam o seu estado.

Modelo de comunicações genérico

Através do uso dos dois tipos de objectos definidos acima a norma DeviceNet define um modelo abstracto para as aplicações em DeviceNet de acordo com a figura 63 .

Na Figura 63 estão representados os seguintes objectos:

- Gestor de mensagens sem conexão - ⁷ - trata do envio e da recepção de mensagens explícitas sem conexão.
- Conexão⁸ - Gere os recursos internos associados com conexões I/O e com conexões explícitas de mensagens.
- Conexão de I/O, Conexão explícita de mensagens ⁹ - Gere a comunicação.
- *DeviceNet* - Permite configurar o estado de uma conexão à rede.
- Ligação consumidor de I/O ou Explícita - Usado por um objecto de conexão para transmitir dados.
- Ligação produtora de I/O ou Explícita - Usado por um objecto de conexão para receber dados.

⁷ *Unconnected Message Manager (UCMM)*

⁸ *Connection Class*

⁹ *Connection Object*

- Gestor de mensagens¹⁰ - Distribui as mensagens explícitas pelos objectos respectivos.
- Aplicação - Implementa as funções principais do produto.

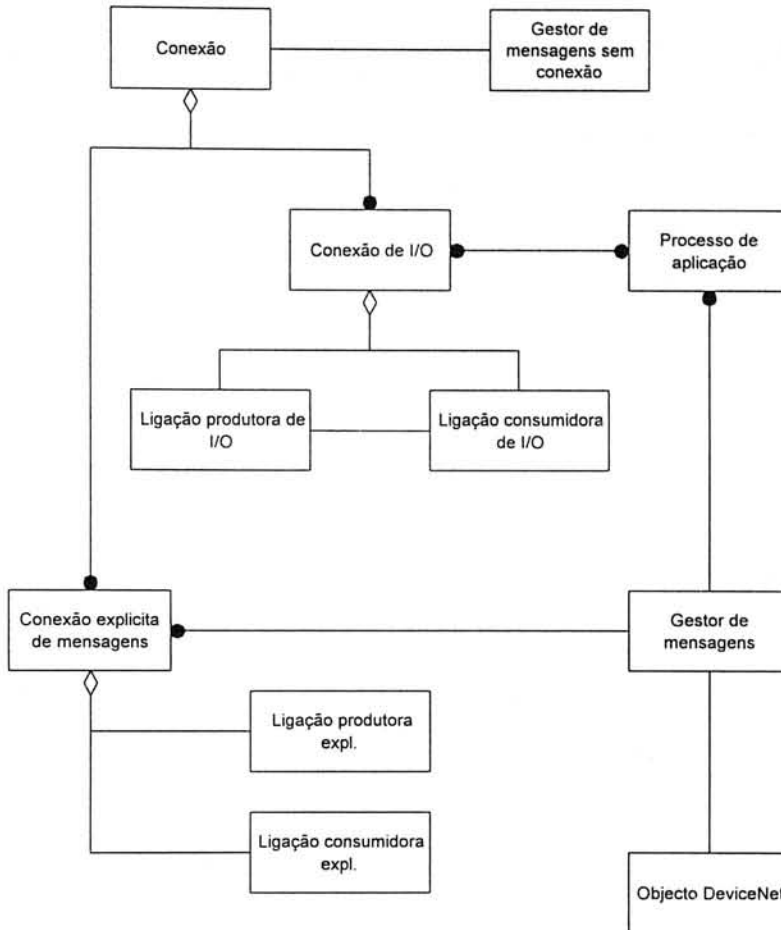


Figura 63 - Modelo de objectos de um dispositivo DeviceNet.

Uso do campo de identificação do CAN

O DeviceNet especifica quatro grupos distintos de mensagens de acordo com a tabela 12. É de notar que esta especificação permite o envio de alguns dados no campo de identificação, tais como, o identificador da mensagem, o identificador do destinatário e o identificador da fonte da mensagem.

¹⁰ Message Routes

Bits do Identificador											Intervalo	Uso	
10	9	8	7	6	5	4	3	2	1	0			
0	Grupo 1 Msg ID				MAC-ID Fonte						000-3FF	Grupo 1	
1	0	MAC-ID				Grupo 2 - Msg ID						400-5FF	Grupo 2
1	1	Grp 3 Msg ID				MAC-ID Fonte						600-7BF	Grupo 3
1	1	1	1	1	Grupo 4 Msg ID						7C0-7EF	Grupo 4	
1	1	1	1	1	1	1	x	x	x	x	7F0-7FF	Não usadas	

Tabela 12 - Uso do campo de identificação no DeviceNet.

As mensagens do grupo 1 não têm nenhuma função atribuída pela norma do DeviceNet. O grupo 2 define os endereços para a troca de mensagens entre *peers* assim como endereços especiais para gestão de conexões mestre/escravo e detecção de duplicação de MAC-ID. O grupo 3 destina-se à troca de mensagens em conexões explícitas. O grupo 4 é reservado pela norma.

Uso do campo de dados

A utilização do campo de dados varia caso o tipo da mensagem seja explícita ou seja uma mensagem de I/O. No caso de uma mensagem de I/O o protocolo não define o que deve ser passado no campo de dados, os dados são da inteira responsabilidade das aplicações.

No entanto uma mensagem explícita tem o seu formato definido de acordo com a tabela 13 .

Byte\bit	7	6	5	4	3	2	1	0
0	Frag	XID	MAC-ID					
1	R/R	Código do Serviço						
2	Definido pelo Serviço							
....	Definido pelo Serviço							
7	Definido pelo Serviço							

Tabela 13 - Campo de dados nas mensagens explícitas

- Frag - Indica se a mensagem é fragmentada ou não
- XID - De modo a identificar a resposta com o pedido

- MAC-ID - da fonte ou do destino conforme o MAC-ID presente no campo de identificação da mensagem
- R/R - Indica se a mensagem é um pedido ou uma resposta
- Código do Serviço - Identifica o serviço a ser executado.

Definição dos Serviços

A definição do protocolo é orientada aos objectos por isso a definição de cada uma das funcionalidades de um dispositivo é feita através da definição de objectos. Para tal são utilizadas as propriedades que uma modelização por objectos oferece, nomeadamente a definição de atributos e funções, relações entre objectos (herança, agregação, etc).

Objecto ligação produtora

Atributos	Descrição
<i>state</i>	Estado em que se encontra o objecto

Tabela 14 - Atributos de uma ligação produtora.

Serviços	Descrição
Send	Transmissão de dados através da rede
Get_Attribute	Leitura de um atributo
Set_Attribute	Modificação de um atributo

Tabela 15 - Serviços de uma ligação produtora.

Objecto ligação consumidora

Atributos	Descrição
<i>state</i>	Estado em que se encontra o objecto

Tabela 16 - Atributos de uma ligação consumidora.

Serviços	Descrição
Get_Attribute	Leitura de um atributo
Set_Attribute	Modificação de um atributo

Tabela 17 - Serviços de uma ligação consumidora.

Objecto conexão

Devido à extensão dos serviços e dos atributos apenas serão especificados os principais.

Atributos	Descrição
<i>state</i>	Estado em que se encontra o objecto
<i>produced_connection_id</i>	Colocado no identificador de CAN quando a conexão está a transmitir
<i>consumed_connection_id</i>	Identifica a mensagem recebida
<i>initial_comm_characteristics</i>	Define o grupo a que pertencem as mensagens desta conexão
<i>production_inhibit_time</i>	Especifica o tempo mínimo entre a produção de novos dados

Tabela 18 - Atributos de um objecto conexão.

Serviços	Descrição
Get_Attribute	Leitura de um atributo
Set_Attribute	Modificação de um atributo
Reset	Reinicializa o objecto
Apply_Attributes	Activa a configuração efectuada

Tabela 19 - Serviços de um objecto conexão

Objecto DeviceNet

Devido à extensão dos serviços e dos atributos apenas serão especificados os principais.

Atributos	Descrição
MAC ID	Endereço do nó
<i>baud_rate</i>	velocidade da rede
<i>bus_off_couter</i>	Número de vezes que o controlador entrou no estado de BUS_OFF
<i>BOI</i>	Indica que foi activa uma interrupção devida ao controlador de CAN ter entrado no estado de BUS_OFF.

Tabela 20 - Atributos de um objecto DeviceNet.

Serviços	Descrição
Get_Attribute	Leitura de um atributo
Set_Attribute	Modificação de um atributo

Allocate_Master/Slave_Connection_Set	Pede para utilizar um conexão <i>Master/Slave</i> pré-definida.
--------------------------------------	---

Tabela 21 - Serviços de um objecto DeviceNet.

Inicialização e configuração da rede

A configuração dos vários nós não se encontra especificamente definida pela norma. Pode ser feita por um nó principal (*master*) que acede aos vários objectos definidos na figura 63.

Perfis de comunicação

O objectivo da definição de perfis de comunicação é de possibilitar a troca de dispositivos do mesmo tipo, mas de fabricantes diferentes e ao mesmo tempo simplificar o trabalho de projecto destes.

Um perfil de comunicação define os objectos principais que cada dispositivo é obrigado a ter, o formato de entrada e saída de dados no dispositivo, a interface pública para os dados produzidos e consumidos e a configuração do dispositivo.

Os objectos presentes em cada um dos perfis pode ser modelizado de acordo com a figura seguinte.

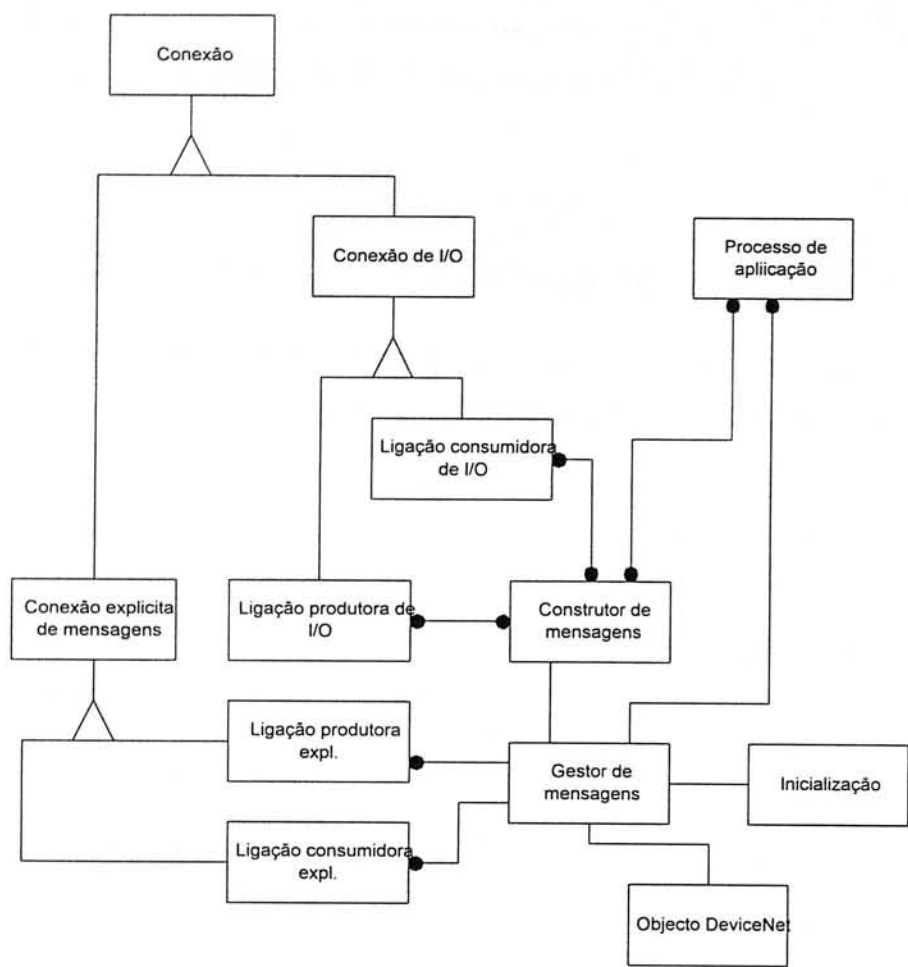


Figura 64 - Modelo de objectos de um dispositivo DeviceNet.

Além dos objectos genéricos já referidos no ponto 0 existem outros objectos utilizados na definição dos perfis são eles:

- Objecto Inicialização: trata da inicialização do dispositivo.
- Objecto Construtor de mensagem: define a maneira como as mensagens devem ser construídas.

Utilizando este modelo genérico de objectos são definidos perfis para os dispositivos do tipo: leitores de código de barras, drives AC e DC, codificadores de posição, contactores, etc.

SDS - Smart Distributed System

O SDS é uma rede de campo especialmente destinada para aplicações simples em automação industrial, em que os dispositivos são, por exemplo: interruptores de fim de curso, relés, motores, I/O analógico/digitais, válvulas electropneumáticas, conversores para sensores 4-20mA, etc.

Esta rede é bastante simples com uma especificação muito detalhada que cobre aspectos tais como:

- camada física: o tipo de cabos de ligação, os conectores, os *transceivers*.
- Camada de aplicação: é definido um conjunto de serviços básicos.
- Perfis: cada tipo de aparelho deve ser construído segundo o perfil definido pela norma.

Camada Física

A camada física desta norma prevê a utilização de uma topologia em barramento com a possibilidade de incorporar linhas de quebra (com o máximo de 3,6m).

Pode operar em 4 velocidades e 4 comprimentos diferentes : 1Mb (22.8m) 500Kb (91.4m), 250Kb (182.8m), 125Kb (475m).

Recomenda o uso dos *transceivers* PCA82C250 e SI9200 com ou sem isolamento óptico adicional.

A ligação do barramento aos dispositivos pode ser feita utilizando ligações directas, aparafusadas, conectores mini, conectores micro e conectores tipo D¹¹.

Os cabos de ligação permitem a alimentação dos dispositivos através de um pino extra, o pino de V⁺.

¹¹ especialmente destinados a ser utilizados na ligação a PC e PLC

Modelo de comunicações

Cada dispositivo SDS é constituído por um componente físico que tem atribuídos alguns atributos especiais tal como: nome do fabricante e versão do software.

Cada componente físico é constituído por 1 a 126 componentes lógicos, aos quais está atribuído um endereço. O objecto embebido permite aceder às funcionalidades do dispositivo. Os objectos **Componente Físico** e **Componente Lógico** são representados na hierarquia SDS através do objecto **Comportamento Mínimo** (figura 66Error! Reference source not found.Error! Reference source not found.).

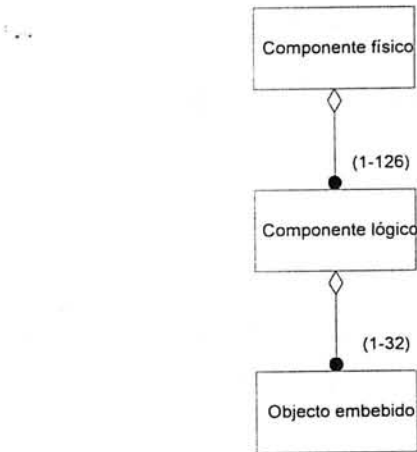


Figura 65 - Modelo hierárquico OO de um dispositivo SDS.

Cada objecto embebido é definido através da herança dos atributos e funções das superclasses superiores tal como se pode verificar para o caso da figura 66, em que o objecto **Saídas Binárias Múltiplas** é definido através da herança de atributos e funções dos objectos **Dipositivo Comum de E/S** e **Saídas Binárias Múltiplas**.

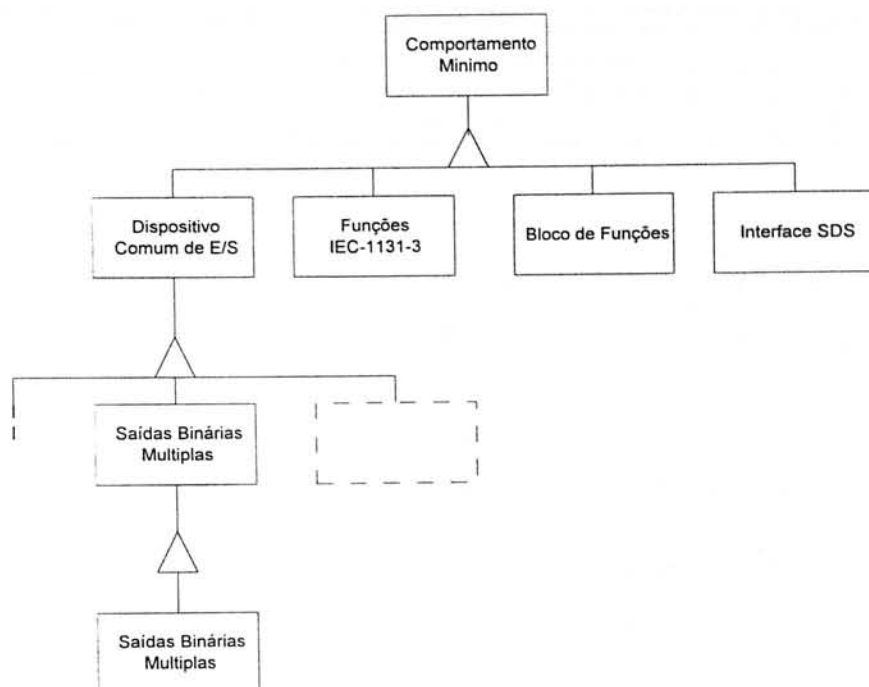


Figura 66 - Modelo geral de um dispositivo SDS.

Os atributos e funções destes objectos podem ser acedidas através do endereço do dispositivo lógico e do identificador do objecto, através dos serviços de *Read/Write* para aceder aos atributos e dos serviços de *Action/EventReport* para aceder às funções do objecto.

Cada um dos objectos utilizados inclui informação relativa aos seus atributos e distingue as funções de cada objecto em *acções* e em *eventos*. Uma acção representa uma função que se pode efectuar sobre o objecto como, por exemplo, a mudança de endereço. Um serve para reportar ao controlador um acontecimento detectado pelo dispositivo como, por exemplo, a mudança de um fim de curso do estado ligado para desligado.

Serviços SDS

Os serviços disponibilizados pelo SDS apresentados e explicados pela tabela abaixo.

Read	Permite ler o valor de um atributo de um objecto contido num módulo.
Write	Permite escrever um valor para um atributo dum objecto contido num módulo.
Event Report	Comunica um evento a outros módulos.
Action	Permite que um utilizador execute uma acção num módulo remoto.
Change Of State On	Especialização de um evento que informa o utilizador de que houve uma mudança do estado para ligado.

Change Of State Off	Especialização de um evento que informa o utilizador de que houve uma mudança do estado para desligado.
Write On State	Coloca o estado de um saída I/O para o estado ON.
Write Off State	Coloca o estado de um saída I/O para o estado OFF.

Tabela 22 - Serviços do SDS.

Juntamente com estes serviços também são especificados os perfis de comunicação para os dispositivos standard de modo a permitir, sem reconfiguração da rede, a troca de dispositivos.

7.1.1 Uso do campo de identificação do CAN

Esta norma bastante simples define apenas dois tipos de tramas (APDU¹²), as tramas longas e as tramas curtas. Apenas está prevista a utilização de identificadores de 11 bits.

Bits do Identificador															
10	9	8	7	6	5	4	3	2	1	0	RTR	DLC3	DLC2	DLC1	DLC0
Dir	Endereço (0-125)							Tipo de APDU				DLC = 0			

Tabela 23 - Identificador de APDUs pequenos.

Os APDUs pequenos (figura 23) permitem o envio de mensagens muito curtas como, por exemplo, assinalar uma mudança de estado de um sensor de ligado para desligado. Este tipo de APDU não permite o envio de qualquer dado.

Bits do Identificador															
10	9	8	7	6	5	4	3	2	1	0	RTR	DLC3	DLC2	DLC1	DLC0
Dir	Endereço (0-125)							Tipo de APDU				DLC >0			

Tabela 24 - Identificador de APDUs grandes.

Os APDUs grandes destinam-se, especialmente, a aceder aos atributos dos objectos.

¹² Application Layer Protocol Data Unit

Uso do campo de dados

Como já foi referido no ponto anterior apenas os APDUs do tipo longo contêm dados, por isso, a tabela seguinte refere-se apenas a esse tipo de tramas.

Byte\bit	7	6	5	4	3	2	1	0
0	Req/Res/Error		frag.	Objecto embebido				
1	Identificador da variável							
2	Definido pelo Serviço							
....	Definido pelo Serviço							
7	Definido pelo Serviço							

Tabela 25 - APDUs Grandes

Este tipo de trama permite aceder a um atributo dentro de um objecto endereçando esse atributo através do identificador do objecto embebido e o identificador da variável. Este tipo de APDU pode ser fragmentado.

Inicialização e configuração da rede

A inicialização e configuração da rede deve ser feita pelo controlador principal. Não se encontra definida pela norma e depende da aplicação e do conjunto de dispositivos utilizados.

A configuração da rede pode ser feita através de um dos controladores do sistema ou através de dispositivos especiais portáteis de configuração da rede. Estes dispositivos geralmente são portáteis, configuram o dispositivo e este trata de armazenar os dados numa flashEprom¹³.

Perfis de comunicação

Em [SDSComp, 95] são definidos os perfis de comunicação. Estes perfis estão separados em quatro subconjuntos ou classes principais de acordo com a figura 66.

A classe Dispositivo Comum de E/S define a maneira de comunicar com os vários dispositivos de entrada e saída simples.

¹³ Alguns dispositivos mais simples não necessitam de armazenar dados de configuração, ou são configurado através de DIP-Switchs.

As outras classes, representadas na Figura 64 não são ainda definidas pela versão da norma utilizada para a escrita desta tese.

CANKingdom

O CANKingdom não é uma camada de aplicação completa, mas antes um conjunto de primitivas que o construtor do sistema pode usar de modo a que possa ter a sua própria camada de aplicação, em que todos os parâmetros do sistema são definidos durante a fase de inicialização.

O CANKingdom foi desenhado segundo o seguinte princípio: *“The modules are to serve the network”*. Este princípio significa que um módulo uma vez ligado à rede apenas deve anunciar que está ligado, após isso serão os outros nós da rede a fornecer a informação necessária para que o nó ligado possa funcionar. Este conceito é o oposto do conceito de *“The network serves the module”*, neste caso um módulo construído segundo este princípio já sabe qual irá ser o seu papel na rede no momento em que é ligado e pede os meios para que possa realizar as tarefas a que está destinado.

Utilizando esta norma o desenhador do sistema pode ter na mesma rede módulos escravos construídos segundo normas de CAN diferentes, apenas necessitando de saber como é que são transmitidos os dados desses dispositivos e como deve responder aos seus pedidos.

Esta norma é especialmente destinada a sistemas de controlo de máquinas, deste modo a linguagem que descreve a norma é muito particular e tem como objectivo que esta seja compreendida por todos os intervenientes no processo de construção da máquina.

Modelização utilizada pelo CANKingdom

Tal como o próprio nome sugere esta norma modeliza o sistema como se fosse um reino, com um *rei* que “vive” na *capital* e controla várias *idades* (outros módulos). O *rei* e as várias *idades* comunicam através do *sistema postal* utilizando as *centrais dos correios* (C.C.) que asseguram as comunicações através do *sistema de distribuição* (rede de comunicações CAN), constituído por várias centrais de correios (chips CAN dos vários módulo).

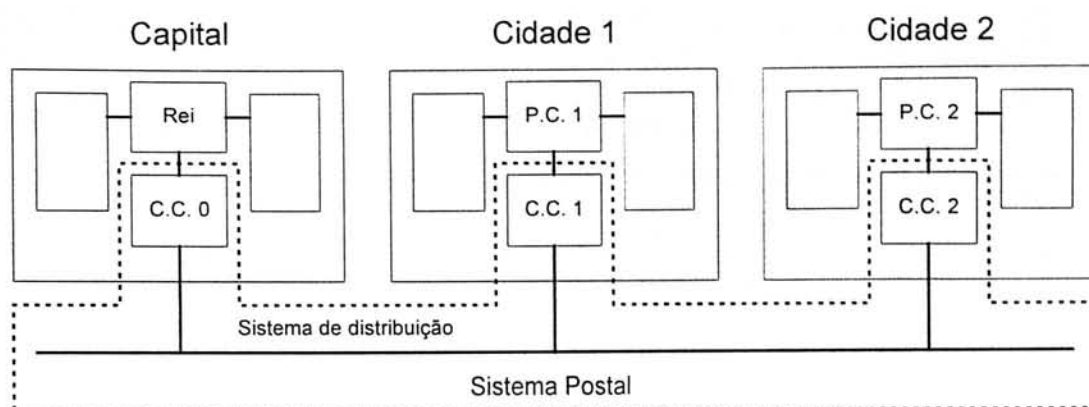


Figura 67 - Representação da rede utiliza pelo CANKingdom.

Utilizando o mesmo tipo de terminologia apresentada na Figura 67 a norma também define utilizando o mesmo tipo de metáfora os seguintes blocos básicos:

Nome	Descrição
Envelopes	Identificadores do CAN (11 ou 29 bits)
Pages	Bloco de dados do CAN, max 8 bytes
Lines	Um byte dos dados do CAN
Bit	Um bit
Letters	Mensagens do CAN, <i>Page + Envelope</i>
Forms	Descreve a informação contida em cada <i>Letter</i> , o desenhador de um módulo pode definir o conteúdo das <i>forms</i> que esse módulo pode receber e enviar ou pode ser o Rei a definir o conteúdo das <i>forms</i> .
Documents	Um <i>document</i> é um conjunto de <i>forms</i> que permite decodificar mensagens enviadas utilizando o mesmo <i>envelope</i> .
Folders	O <i>folder</i> representa uma ligação entre um <i>document</i> e o <i>Sistema Postal</i> . Podemos ter por exemplo um <i>folder</i> para receber e outro para enviar. Os <i>folders</i> ligam um <i>document</i> a um <i>envelop</i> .
Lists	Cada cidade pode ter um conjunto pré-definido de <i>documents</i> que o <i>Rei</i> poderá usar caso necessite. As <i>list</i> definem algumas das funcionalidade básicas do módulo. De modo a aumentar a flexibilidade as <i>lists</i> podem ser definidas pelo Rei

Tabela 26 - Blocos básicos utilizados no CANKingdom.

Uso do campo de identificação do CAN

Esta norma é extremamente flexível na utilização do campo de identificação. Permite o envio de *letters*(mensagens) de um certo tipo com um identificador próprio ou permite o envio de dados no *envelope*(identificador).

Os *envelopes* de valor fixo são geralmente associados a *folders*, são aí colocados automaticamente utilizando filtros especiais fornecidos pelo fundador do reino. A utilização de filtros¹⁴ pelas centrais dos correios (controladores de CAN) permite que os *mayors* das várias cidades recebam apenas as mensagens que lhe são destinadas.

Utilizando os filtros referidos é possível que uma cidade receba dados utilizando os *envelopes* (campo de identificação do CAN), a este tipo de *envelopes* chama-se *compressed envelopes*.

Para a descodificação dos *envelopes*, tanto *compressed* como normais, são utilizadas *forms* que descrevem a informação contida em cada *letter* (mensagem mais indentificador).

O *compressed envelope* é usado de acordo com o exemplo da tabela seguinte

Bits do Identificador																													
2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	
9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										
Endereço cidade										Dados																			

Tabela 27 - Exemplo da utilização do campo identificador em *compressed envelopes*.

Como se pode ver o primeiro campo de um *compressed envelopes* pode conter um endereço para descodificação, é através dos filtros de mensagens referidos que uma cidade consegue distinguir se o *compressed envelope* lhe está destinado. A descodificação de campo de dados é feita utilizando as *forms* que lhe foram enviadas durante a inicialização pelo fundador do reino.

Os *compressed envelopes* são especialmente úteis quando se utiliza a rede CAN a funcionar com tramas estendidas de 29 bits.

Uso do campo de dados

Sendo a flexibilidade a palavra que mais qualifica esta rede mais uma vez existe toda a flexibilidade na utilização do campo de dados nesta norma.

¹⁴ implementados em hardware em alguns uC

Utilizando a página 18 dos documentos do *Rei* o fundador do reino pode configurar todas as cidades com as *forms* correctas para a descodificação das mensagens enviadas por outros dispositivos do sistema.

Esta norma possibilita o envio de dados em formato comprimido (*compressed letters*), isto é, se por exemplo se se pretender actuar várias saídas digitais existentes em vários dispositivos de E/S diferentes pode configurar-se os vários dispositivos de modo a que cada bit dentro da mensagem corresponda a uma das saídas. Esta técnica permite poupar largura de banda, dado que não é necessário enviar para cada dispositivo uma mensagem individual e, além disso, a mensagem a enviar é substancialmente mais pequena (devido a cada saída digital ser representada por um determinado bit dentro do campo de dados de mensagens).

A utilização conjunta de *compressed letters* e de *compressed envelopes* vai ainda acrescentar um grande poder e flexibilidade a todas as comunicações utilizadas nesta norma.

As características referidas neste ponto e no ponto anterior vão permitir que dispositivos construídos de acordo com esta norma possam comunicar com outros dispositivos construídos de acordo com outras normas. De modo a que os dispositivos CANKingdom comuniquem com dispositivos CAN-OPEN apenas é necessário ao fundador do reino configurar correctamente as *forms* existentes nos dispositivos CANKingdom.

Serviços

Utilizando os blocos construtores definidos no ponto 0 a norma define os serviços básicos que cada módulo deve conter. Estes serviços (definidos através de *documents*) estão divididos em *documents* do Rei, do *Mayor*, de transferência de blocos e mensagens de tempo. De seguida serão analisados em mais detalhe cada um dos *documents* referidos.

É de notar que esta norma não define serviços mas sim o formato e o significado das mensagens que podem ser trocadas entre os vários módulos; para isso, utiliza os blocos construtores referidos anteriormente.

Documents do Rei

Os documentos do Rei destinam-se a inicializar o reino. Após a fase de inicialização deixam de ser necessários. Os vários documentos do Rei encontram-se expostos na tabela seguinte :

Page	Descrição
0	É usada para informar os <i>mayors</i> endereçados que a fase de inicialização acabou e que devem colocar a sua cidade em determinado modo de funcionamento.
1	Através deste serviço o <i>Rei</i> atribui um identificador(<i>envelope</i>) para ser utilizado pelos documentos do <i>mayor</i> de uma cidade.
2	Permite atribuir um <i>envelope</i> a um <i>folder</i> numa cidade e configurar os parâmetros de utilização deste.
3	Atribui uma cidade a um ou mais grupos, o que permite que esta seja endereçada através de endereços de grupo.
4	Remove uma cidade de um grupo.
5	Configura uma cidade para reagir à recepção de uma determinada página contida num <i>folder</i> , enviando outra página de outro <i>folder</i> .
8	Configura os parâmetros do controlador CAN numa cidade.
9	Permite modificar o endereço de uma cidade.
10	Atribui a um <i>envelope</i> um tempo mínimo de inactividade entre transmissões sucessivas. Resolve os problemas causados por um controlador com alta prioridade que está sempre a tentar enviar mensagens.
11	Esta página permite configurar a transmissão do tempo que posteriormente pode ser utilizado para sincronizar os módulos intervenientes na computação distribuída.
12	Configura a transmissão de mensagens de tempo .
16	Coloca um <i>document</i> de um módulo num <i>folder</i> desse módulo.
17	A partir deste serviço o <i>Rei</i> cria um <i>document</i> , uma <i>form</i> ou uma <i>line</i> utilizando elementos presentes nas listas de um módulo.
18	Cria uma <i>compressed form</i> , utilizando os listas presentes no módulo.
19	Envia para um módulo uma máscara para <i>envelopes</i> do tipo comprimido.
20	Envia para um módulo o filtro que este deve utilizar para a descodificação de <i>Envelopes</i> .

Tabela 28 - Serviços do *Rei*.

As páginas 6,7 e de 13 a 15 são páginas reservadas.

Documents do Mayor

Estes *Documents* permitem a identificação dos vários *mayors* pelo *Rei*.

Page	Descrição
0	Contém o código EAN-13 que identifica o módulo.
1	Contém o numero de série da cidade.

Tabela 29 - *Documents* definidos pelo *mayor*.

Documents de transferência de blocos

Quando uma cidade quer transmitir mais do que 8 bytes de informação é necessário que esta o faça de forma fraccionada utilizando blocos. Para atingir este objectivo e ultrapassar a limitação do tamanho das mensagens do CAN, a cidade pode utilizar funções especiais para transferir blocos de tamanho elevado.

Page	Descrição
0	Permite a um <i>mayor</i> pedir para que lhe seja enviado um bloco a outro <i>mayor</i> .
1	O <i>mayor</i> que irá enviar o bloco informa o outro <i>mayor</i> do tamanho do bloco a ser enviado.
2	Permite ao receptor do bloco informar o emissor de que este pretende receber as próximas <i>n pages</i> .
3 e 4	Formato de um bloco, cada bloco é dividido em conjuntos de 6 bytes.
5	Permite abortar a transmissão de um bloco.

Tabela 30 - *Documents* de transferência de blocos.

Documents de mensagens de tempo

Em sistemas de controlo tempo real é necessário que alguns dos nós tenham o seu relógio sincronizado. Para este tipo de sistemas é definido a maneira como devem ser enviadas as mensagens que representam o tempo.

Page	Descrição
X	Define o formato de transferência do tempo circular entre módulos.
X	Define o formato de transferência do tempo standard entre módulos.

Tabela 31 - *Documents* de mensagens de tempo.

Perfis dos dispositivos

Esta norma não define qualquer perfil de comunicação para os dispositivos, este facto permite uma grande flexibilidade aos integradores do sistema, uma vez que é possível utilizar dispositivos construídos de acordo com a norma DeviceNet, CAL, SDS [Lennartsson, 90]. É apenas necessário definir, convenientemente, as várias mensagens trocadas entre os dispositivos e o seu significado.

Anexo B - Metodologia OMT

Um modelo é uma abstracção de algo, desenvolvido com o propósito de o compreender antes de o implementar. Um modelo deve omitir os detalhes não essenciais, de modo a ser mais fácil de manusear do que a entidade original. Deve por isso centrar a atenção apenas nas características relevantes para o objectivo da análise em questão abstraindo-se da implementação.

A modelização efectuada recorre à metodologia de James Rumbaugh, “*Object Modeling Technique*” (O.M.T.) [Rumbaugh, 91]. Esta metodologia recorre a três tipos de modelos para descrever um sistema: o modelo de objectos que descreve a estrutura estática, o modelo dinâmico que descreve o comportamento temporal e o modelo funcional que descreve a transformação de informação.

Estes modelos não são independentes permitindo analisar um mesmo sistema de ângulos diferentes. Algumas das ideias subjacentes a esta metodologia, são:

- a **abstracção**, que consiste em focar o interesse nas propriedades essenciais dos objectos, ignorando as suas características acidentais;
- o **encapsulamento**, que consiste em separar os aspectos externos de um objecto, aqueles que são acessíveis por outros objectos, dos detalhes internos de implementação, aqueles que estão “escondidos”;

- **combinação de dados e comportamento**, deste modo quem chama uma determinada operação não necessita de saber quantas implementações dessa operação existem;
- a **partilha**, que existe a vários níveis;
- a **herança** de estrutura de dados e comportamento que permite que uma estrutura comum seja partilhada por várias classes semelhantes sem redundância, o que se pode traduzir, não só em aproveitamento de código, mas também na clareza em reconhecer que operações diferentes à partida são na realidade a mesma.

Esta metodologia consiste num ciclo de desenvolvimento ao longo do qual os modelos vão evoluindo. A primeira fase é a análise, que compreende o modelo do domínio de aplicação, depois o projecto que acrescenta ao modelo anterior dados para a solução do problema, e por fim a implementação que adapta os modelos anteriores a uma solução informática. Cada fase do ciclo de desenvolvimento é a base da seguinte, que se pode centrar na sua finalidade específica (separação de preocupações). O objectivo deste trabalho compreende apenas a fase de análise.

De seguida é feita a descrição dos vários modelos do método OMT de modo a facilitar a compreensão da análise efectuada ao CAL:

Modelo de Objectos

Este modelo descreve a estrutura dos objectos do sistema, a sua identidade, as suas inter-relações, os seus atributos e as operações que lhes estão associadas. Os objectos do modelo correspondem a entidades físicas ou conceptuais do sistema real. Todos os objectos são distinguíveis pelo simples facto de existirem e não pelas propriedades que possam ter. Para aplicar a técnica O.M.T., os objectos são agrupados em classes de características semelhantes. Diz-se que um objecto particular é uma instância da sua classe de objectos.

Uma classe de objectos descreve um grupo que partilha certas características: propriedades (atributos), comportamento (operações), relações com outros objectos e significado semântico. As classes são representadas por rectângulos. Os diagramas de classes descrevem classes de objectos, incluindo as respectivas relações. Essas relações são verdadeiras para todas as instâncias eventualmente existentes.

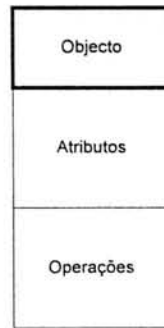


Figura 68 - Objecto

Os atributos de um objecto têm as seguintes características:

- valores(de um dado tipo), não podem ser outros objectos;
- o seu nome é único na classe;
- podem ter um tipo e um valor por omissão;
- para cada instância os valores dos objectos podem ser diferentes.

Os atributos dos objectos de uma classe são enumerados na segunda parte da caixa que representa a classe. Operações são funções ou transformações aplicadas a objectos. Todas as operações referem-se, implicitamente, a um objecto. O comportamento da operação depende da classe do objecto. As operações são representadas na terceira parte da caixa que representa a classe.

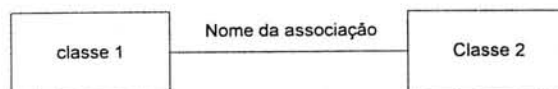


Figura 69 - Associação entre objectos

Uma ligação é uma conexão física ou conceptual entre dois objectos. Uma associação descreve um conjunto de ligações com a mesma estrutura e semântica. Associações são bidireccionais e podem ser implementadas por apontadores: um atributo de um objecto com uma referência explícita a outro. Graficamente, as associações são representadas por linhas com anotações (nome e multiplicidade). Associações e ligações podem ser descritas com mais pormenor, por exemplo, com atributos de ligação, que são propriedade das ligações de uma associação que não pode ser associada a nenhuma das

classes intervenientes. Em certas situações pode ser vantajoso modelizar uma associação como uma classe, em particular para lhe associar operações e outras classes.

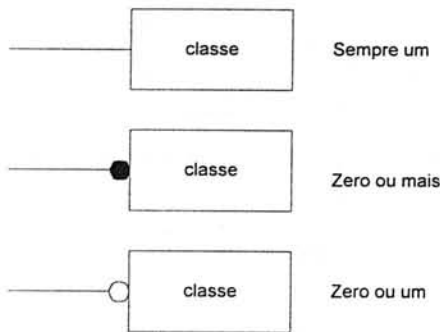


Figura 70 - Multiplicidade das associações.

A multiplicidade é indicada por “círculos” nos extremos da associação junto às “caixas” das classes.

Uma agregação representa a relação *parte de*, em que objectos que representam componentes estão associados a um objecto que representa a montagem completa. São representadas por um losângulo encostado à classe que representa a montagem final.

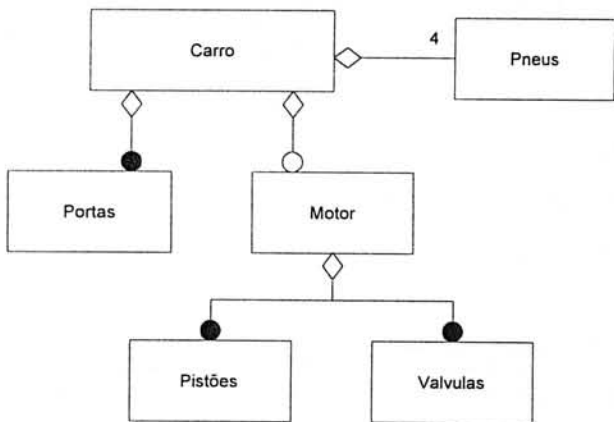


Figura 71 - Agregação

A generalização é a relação existente entre uma classe (a superclasse) e as suas versões mais refinadas (subclasses). Cada subclasse herda todas as características da superclasse: relação *is a*. A generalização é uma relação transitiva e consiste numa relação entre classes; herança, é um mecanismo de partilha de atributos e operações usando o princípio da generalização. Uma subclasse pode redefinir elementos da superclasse, mas

preservando a assinatura dos métodos e tipo dos atributos: o elemento redefinido tem o mesmo nome.

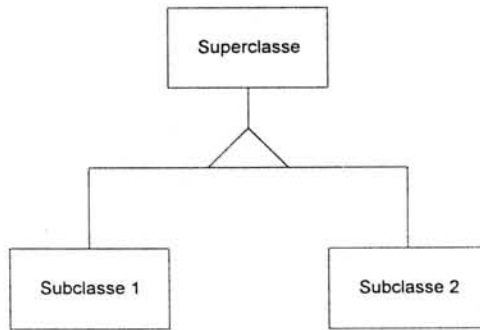


Figura 72 - Generalização ou herança.

Modelo Dinâmico

Este modelo descreve todos os aspectos do sistema que se relacionam com: a passagem do tempo e o sequenciamento de operações, os eventos que sinalizam as modificações, as sequências de eventos, o estado do sistema e a inter-relação de eventos e estados. O modelo dinâmico descreve sequências de operações (que envolvem os objectos do modelo de objectos) sem definir o que as operações fazem ou como são implementadas. O modelo dinâmico é expresso por diagramas estruturados de estados. Os principais conceitos usados são eventos, que representam estímulos externos, e, estados que representam os valores dos objectos.

Um evento é algo que acontece num dado instante. Um evento não tem duração (em termos da granularidade da escala de tempo usada).

O estado de um objecto especifica a sua resposta aos eventos que recebe. Esta resposta pode incluir acções e modificações de estado.

Um diagrama de estados relaciona eventos e estados. O diagrama é um grafo em que os nós representam estados e os arcos transições etiquetadas com o evento que as provoca. Quando um evento ocorre, a transição dispara e o objecto passa instantaneamente para o estado final.

Uma condição é uma função booleana de valores de objectos. Uma condição é válida num dado intervalo de tempo. As condições podem ser usadas como guardas de transições. A transição só pode ocorrer se a condição for verdadeira.

As operações podem ser associadas a transições e estados. Uma actividade é uma operação que demora algum tempo a ser executada e está associada a um estado. Uma acção é uma operação instantânea que está associada a um evento.

Uma actividade de um estado pode ser expandida num diagrama de estados de nível mais baixo, em que cada estado representa um passo dessa actividade. Um diagrama hierárquico representa uma generalização do conceito de estado. Os diferentes (sub-) estados estão numa relação de exclusividade: o sistema está exactamente em apenas um desses estados.

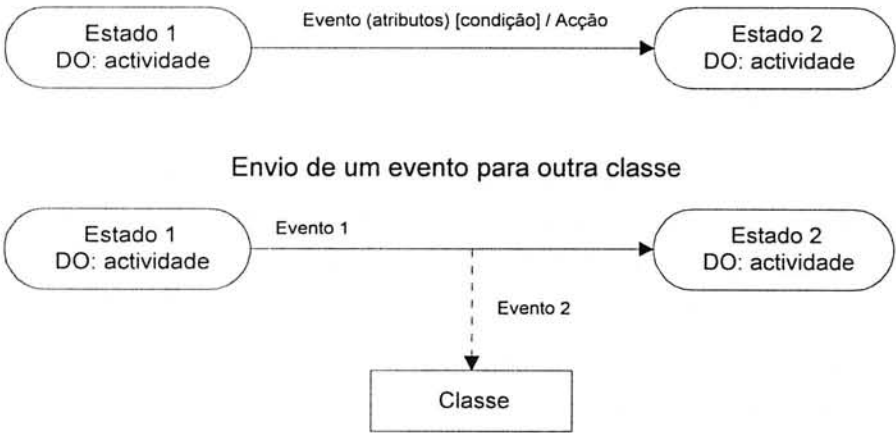


Figura 73 - Relações principais no modelo dinâmico.

Modelo Funcional

Este modelo descreve os aspectos do sistema relacionados com a transformação de dados, as funções, o mapeamento de valores, as restrições e as dependências funcionais; captura a funcionalidade do sistema, sem descrever a relação temporal das operações ou a sua implementação.

O modelo funcional é composto por múltiplos diagramas de fluxo de dados (DFDs) que mostram o fluxo dos valores a partir da entrada, “através” das operações, até às saídas.

O modelo funcional especifica o resultado da computação sem especificar como ou quando é que as operações são efectuadas. Este modelo especifica, ainda, o significado

das operações do modelo de objectos, das acções do modelo dinâmico e as restrições associadas ao modelo de objectos.

Um diagrama de fluxo de dados é composto por processos que transformam a informação, fluxos de dados que movimentam os dados, objectos produtores e/ou consumidores de informação e bases de dados que armazenam passivamente a informação.

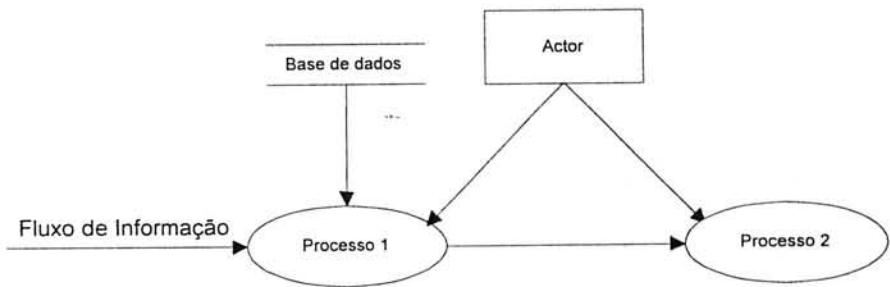


Figura 74 - Notação do modelo Funcional.

Referências

- [ASI-Festo, 94] Festo Pneumatic
Actuator Sensor Interface - New Products
1994
- [ASI-Siemens, 94] Siemens
Actuator Sensor Interface - System Description
1994
- [Astrom et al 90] karl J. Astrom, Bjorn Wittenmark
Computer Controlled Systems - Theory and Design - 2ª edição
Prentice Hall International Editions
1990
- [Bennet, 94] Stuart Bennet
Real-Time Computer Control - An introduction
Prentice Hall International Series in Systems and control Systems
1994
-

- [Baba, 95] M. Dani Baba, E. T. Powner
Scheduling Performance in Distributed Real-Time Control
Systems
Proceedings fo the 2nd International CAN Conference
1995
- [Bosch, 91] Robert Bosch GmbH
CAN Specification Versio 2.0
1991
- [CAL, 96] CiA - CAN in Automation, Internatinal Users and Manufactures
Group e.V.
CAL - CAN Aplicacion Layer for Industrial Aplicacions
CiA Draft standard 201 - 207 Version 1.1
1996
- [CANOpen, 96] CiA Draft Standard 301, Ver 3.0
CANOpen - CAL-based Communication Profile for Industrial
Systems
1996
- [CANKing, 94] Kvaser
CANKingdom
1994
- [CANSpec, 91] Robert Bosh GmbH
CAN Specification Version 2.0
1991
- [Cardeira et al, 93] Carlos Cardeira, Jean-Pierre Thomesse
O Bus de Campo Fip
1993
- [Carvalho, 95] José Augusto Carvalho
Faculdade de Engenharia da Universidade do Porto
1995

- [COP68X, 94] Data sheet COP684BC/ COP884BC Single-Chip CMOS
Microcontroller
National Semiconductor
1994
- [Crovella, 94] Robert M. Crovella
Honeywell - Micro switch
SDS Smart Distributed System: A CAN Protocol for Plant Floor
Control
- [Crovella, 94] Robert M. Crovella
Honeywell - Micro switch
SDS Smart Distributed System: A CAN Protocol for Plant Floor
Control
1994
- [Ekiz *et al*, 96] H. Ekiz, A. Kutlu, M. D. Baba, E. T. Powner
Design and implementation of a CAN / Ethernet Bridge
Proceedings of the 3rd International CAN Conference
1996
- [Embacher, 95] Martin Embacher
Replacing dedicated protocol controllers in low speed CAN
network with cost efficient and customizable micro-
controller solution.
National Semiconductor
1995
- [DaimlerBenz, 95] DaimlerBenz
ESP Takes to the Road
DaimlerBenz - HighTech Report
1995
- [DeviceNet Vol I, 96] ODVA - Open DeviceNet Vendor Association
DeviceNet Volume I, release DeviceNet Communication Model
and Protocol ,release 1.2.
1996

- [DeviceNet Vol II, 96] ODVA - Open DeviceNet Vendor Association
DeviceNet Volume II: Device Profiles and Object Library, release
1.2
1996
- [Farges et al, 96] E. Farges, L. Massimeli
Validation of CAN Real-Time Embedded Automotive Protocols
throughout SDL Software Models and MSC Test Scenarios
Proceedings of the 3rd International CAN Conference
1996
- [Fishman, 78] George S. Fishman
Principles of Discrete Event Simulation
John Wiley and Sons
1978
- [Forsyth1, 90] Charles Forsyth
Using the Worst Case Execution Analyser
Task 8 vol. D Deliverable to European Space Agency 'Hard Real
Time Operating Systems Kernel Study'
1990
- [Forsyth2, 90] Charles Forsyth
Implementation of the Worst Case Execution Analyser
Task 8 vol. D Deliverable to European Space Agency 'Hard Real
Time Operating Systems Kernel Study'
1990
- [Fredrikosson, 95] Lars-Berno Fredrikosson, Kvaser AB
Controller Area Networks and the Protocol CAN for Machine
Control Systems
<http://www.kvaser.se>
1995

- [Gergeleit *et al*, 94] Martin Gergeleit, Hermann Streich
Implementing a Distributed High-Resolution Real-Time Clock
Using de CAN Bus
<http://pmt05.et.uni-magdeburg.de/NT/CAN>
1994
- [Intel527, 93] Intel Corporation
82527 - Serial communication Controller - Architectural
Overview
1993
- [Lennartsson, 90] Kent Lennartsson
Fundamental parts in SDS, DeviceNet & CANKingdom - A
Comparison
Proceedings of the 2nd International CAN Conference
1995
- [Levi *et al*, 90] Shem-Tov Levi, Ashok K. Agrawala
Real Time System Design
McGraw-Hill International Editions - Computer Science Series
1990
- [MCAN, 96] Kenneth Terry
The Motorola MCAN Module
Motorola - Semiconductor Engineering Bulletin
1996
- [Milenkovic, 92] Milan Milenkovic
Operating Systems - Concepts and design - 2^a edição
McGraw-Hill International Editions - Computer Science Series
1992
- [MITS, 94] MITS - Maritime Standard
MITS Architecture V3.0
1994

- [Nelisse, 96] ir. M. W. Nelisse
CAN Development Environment
Proceedings of the 3rd International CAN Conference
1996
- [Moon, 1996] Tony Moon
Control in Automotive Body Systems Networks
Proceedings of the 3rd International CAN Conference
1996
- [NSSLIO, 95] Data sheet MM57C360/MM57C362 Single-Chip micro CMOS
CAN-linked I/O
National Semiconductor
1995
- [Peyrucat, 95] Jean-François Peyrucat
Siemens et Allen Bradley: deux stratégies pour les réseaux
Mesures, 675 Mai
1995
- [Pinho, 96] Miguel Pinho
Faculdade de Engenharia da Universidade do Porto
1996
- [Pisartz et al, 96] J. Pisartz, Dr. M. Schneider
Modelling of Embedded CAN Applications
Proceedings of the 3rd International CAN Conference
1996
- [Portugal, 95] Paulo Portugal
Desenvolvimento de um nó inteligente de I/O analógicas e digitais
para uma rede ethernet, com suporte no protocolo TCP/IP
Faculdade de Engenharia da Universidade do Porto
1995

- [Reigraf, 96] Thomas Reigraf
Development Methodology and Tools for CAN Networks in
Motor Vehicles
Proceedings of the 3rd International CAN Conference
1996
- [Rumbaugh, 91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen
Object-Oriented Modeling and Design
Prentice-Hall International, Inc
1991
- [Schneider, 96] S. Schneider
Performance Analysis for Automotive CAN Systems
Proceedings of the 3rd International CAN Conference
1996
- [Schofield, 96] M. J. Schofield
<http://www.omegas.co.uk/CAN>
1996
- [SDSComp, 95] Honeywell - Micro switch
Smart Distributed System
Component Model Specification
1995
- [SDSphy, 94] Honeywell - Micro switch
Smart Distributed System
Physical Layer Specification
1994
- [SDSprod, 94] Honeywell - Micro switch
Smart Distributed System
Product Selection Guide
1994

- [SDSdesign, 94] Honeywell - Micro switch
Smart Distributed System - Design and Implementation Guide
1994
- [SDSapli, 95] Honeywell - Micro switch
Smart Distributed System - Application Layer Protocol
Specification
1995
- [SDSconf, 95] Honeywell - Micro switch
Smart Distributed System - Conformance Test Procedure
Specification
1995
- [Siemens, 93] Siemens A.G.
The On-Chip CAN-Module
Microcomputer components C167C
1993
- [SP50, 91] IEC 65C WG6 / ISA SP 50
Functional Fieldbus Requirements
1991
- [StauB, 95] Gerhard StauB
Industrial communications for all automation tasks
Engineering & Automation XVII n° 5
1995
- [Tanenbaum, 92] Andrew S. Tanenbaum
Modern Operating Systems
Prentice-Hall International, Inc.
1992
- [Tindell1, 95] Ken Tindell
Real Time Systems and Fixed Priority Scheduling
<ftp:minster.york.ac.uk:/pub/realtime/papers>
1995

- [Tindell2, 94] K. Tindell, A. Burns, A. Wellings
Analisis of Real-Time Communications
ftp:minster.york.ac.uk:/pub/realtime/papers
1994
- [Tindell3, 94] K. Tindell, A. Burns
Garanteeing Message Latencies on CAN
ftp:minster.york.ac.uk:/pub/realtime/papers
1994
- [Tindell4, 95] K. Tindell, Hans Hansson
Babbling Idiot, The Dual-Priority Protocol and Smart CAN
Controllers
Proceedings of the 2nd International CAN Conference
1995
- [Törngren *et al*, 93] Martin Törngren, Ulf Backman
Evaluation of Real Time Communication Systems for Machine
Control
Proceedings of the Swedish National Association on Real-Time
System Conference
1993
- [Upender *et al*, 96] B. Upender, A. Gean
Variability of CAN Network Performance
Proceedings of the 3rd International CAN Conference
1996
- [Uphoff, 1995] Jens Uphoff
Network Tools for Realtime Modelling of CAN Systems
Proceedings of the 2nd International CAN Conference
1995
- [Vaes *et al*, 96] P. Vaes, B. Meuris, V. Vande Keere, J. Vandewege
Fortress, a CAN Based Development Environment for Real-Time
Embedded Applications
Proceedings of the 3rd International CAN Conference
1995

- [Wilkinson, 92] Barry Wilkinson
Computer Architecture - Design and Performance
Prentice Hall
1992

Bibliografia

- [Audsley] N. Audsley, A. Burns
Predictability Dependable Computer Systems, Vol 2
ESPRIT BRA Project (3092)
- [CANHome, 96] CAN Home Page
<http://pmt05.et.uni-magdeburg.de/NT/CAN>
1996
- [ECHELON, 93] ECHELON
LonTalk Protocol
LonWorks Engineering bulletin
1993
- [ECHELON, 95] ECHELON
Enhanced Media Access Control with LonTalk Protocol
LonWorks Engineering bulletin
1995
-

- [Göddertz, 90] Joachim Göddertz
Profibus
Klockner-Moeller GmbH - Thecnical Publication for Industry and
Commerce
1990
- [Groover, 87] Mikell P. Groover
Automation Production Systems, and Computer Integrated
Manufacturing
Prentice-Hall, Inc
1987
- [Liu et al, 73] Scheduling Algorithms for Multiprogramming in Hard Real-Time
Environment
JACM, Volume 20, Number 1
1973
- [Mellor et al 86] Mellor S., e Ward P.
Structured Development for Real-Time Systems, Vols. 1-3,
Prentice-Hall
1986
- [NRTT, 96] NRTT
[http\\www.nrtt.demon.uk\](http://www.nrtt.demon.uk/)
1996
- [Kattwinkel, 95] Thomas Kattwinkel
Tuning up automotive networks with the CAN bus
Siemens Components, vol XXX Nov/Dec 6/95
1995
- [Ogata, 82] Engenharia de Controle Moderno - Não referido
Katsuhiko Ogata
Prentice-Hall
1992



- [Rostan, 97] Martin Rostan, Martin Jaeggi
Communication with CANOpen
CiA forum "CAN for Newcomers"
1997
- [Sousa, 96] Armando Sousa
Desenvolvimento de um Controlador Avançado para Sistema
Embebido
Faculdade de Engenharia da Universidade do Porto
1996
- [Tovar, 95] Eduardo Tovar
Redes de Comunicação Industriais do tipo field Bus, Integração
em Ambiente CIM
Faculdade de Engenharia da Universidade do Porto
1995

