Universidade do Porto

Faculdade de Engenharia

FEUP

ARQUITECTURAS DE COMUNICAÇÃO INDUSTRIAIS PARA SUPORTE A SISTEMAS COMPUTACIONAIS MÓVEIS

VERÍSSIMO MANUEL BRANDÃO LIMA SANTOS

Licenciado em Engenharia Electrotécnica – Electrónica e Computadores
pelo Instituto Superior de Engenharia do Porto

Dissertação submetida para a satisfação parcial dos requisitos do grau de mestre em
Engenharia Electrotécnica e de Computadores
(perfil de Sistemas Digitais e Informática Industrial)

Dissertação realizada sob a supervisão de:

Prof. Doutor Luís Miguel Pinho (Orientador)
do Departamento de Engenharia Informática
do Instituto Superior de Engenharia do Porto

Prof. Doutor Pedro Ferreira Souto (Co-orientador)
do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

Porto, Setembro de 2007

Resumo

Um dos grandes objectivos na análise de sistemas distribuídos com requisitos de tempo real é a possibilidade de prever o comportamento temporal do sistema. Para este efeito são geralmente utilizadas duas abordagens: a análise do pior caso e a análise do caso médio. Cada uma destas abordagens permite extrair informação sobre determinadas características do sistema, sendo complementares entre si pois a utilização de ambas permite uma maior compreensão do comportamento temporal do sistema.

Se por um lado a análise do pior caso permite estabelecer garantias quanto a um patamar mínimo de desempenho que pode ser esperado do sistema, por outro o seu pessimismo conduz quase sempre a resultados que se distanciam muito dos obtidos em situações reais. Para obter um comportamento mais provável do sistema distribuído é normalmente utilizada a simulação de eventos discretos que permite aproximar a utilização efectiva do sistema.

O projecto europeu RFieldbus (IST-1999-11316) teve como objectivo a expansão da arquitectura de uma das redes de comunicação integrantes na norma europeia EN50170 (PROFIBUS, WorldFIP e P-Net) no sentido de incluir novas funcionalidades que se demarcavam pela sua importância. No âmbito desse projecto, foi desenvolvida a arquitectura da rede de comunicação RFieldbus, tendo como base a arquitectura PROFIBUS e foi implementada no Laboratório de Sistemas Críticos do grupo de investigação IPP-HURRAY do Instituto Superior de Engenharia do Porto uma aplicação de teste (MAF – *Manufacturing Automation Fieldtrial*) consistindo numa aplicação típica de um sistema de manufactura de componentes discretos, cujos diferentes componentes integrantes se encontram interligados por uma rede de comunicação industrial RFieldbus. Este sistema evidencia as principais novas funcionalidades desta arquitectura: a comunicação em meio físico sem fios, o eficiente escalonamento de tráfego de controlo, a integração de tráfego multimédia e a mobilidade de dispositivos sem fios.

O RFieldbus, enquanto rede de comunicação industrial integra os sistemas distribuídos existentes na planta fabril e o seu desempenho afecta o desempenho do sistema distribuído. Por este motivo o estudo do comportamento temporal desta rede de comunicação é importante pois contribuirá para a compreensão do comportamento temporal do sistema distribuído em que está inserido. Desta forma, uma aplicação de software foi desenvolvida no âmbito do projecto RFieldbus para calcular o pior caso do comportamento temporal de uma aplicação da rede RFieldbus. Esta aplicação foi utilizada para calcular esta resposta na topologia do MAF. Posteriormente foi observado o comportamento da rede do MAF e comparadas as diferentes respostas obtidas para o pior caso e para as observações do comportamento normal (mais provável) da rede.

Simultaneamente, foi desenvolvido o simulador LLRS (*Lower Layer RFieldbus Simulator), com o* objectivo de permitir a simulação do comportamento temporal em funcionamento normal da rede RFieldbus. O modelo desenvolvido e implementado no simulador utiliza o modelo de comportamento, os parâmetros dos dispositivos, da rede e os ciclos de mensagem que contribuem significativamente para o comportamento temporal de uma rede RFieldbus. Este simulador foi validado com os dados das observações feitas ao comportamento da aplicação do MAF, e permite de uma forma simples e barata, estudar o comportamento temporal de redes RFieldbus com diferentes elementos, topologias, ciclos de mensagem e parâmetros.

Esta dissertação apresenta as bases teóricas, o modelo e o simulador desenvolvidos, os resultados obtidos e sua validação.

Palavras-chave: Redes de campo industriais, simulação de eventos discretos, sistemas distribuídos de controlo por computador, sistemas de tempo real, RFieldbus, PROFIBUS (EN50170), OMNeT++.

Abstract

In the analysis of real-time distributed systems one of the most important objectives is to foresee the timing behaviour of the system. In order to do this, two approaches are commonly used: worst case analysis and typical behaviour analysis, each allowing to extract distinct system characteristics, which are complementary to each other. The use of both approaches allows us to better understand the system timing behaviour.

On one hand, worst case analysis establishes a minimum expected system behaviour, on the other hand its underlying pessimism leads to results that are far from real system behaviour (the most probable to occur in reality). To obtain a more probable behaviour occurring in distributed system, the discrete-event simulation technique is commonly used.

The European RFieldbus project (IST-1999-13316) aimed the expansion of one of the architectures integrated in the European standard EN50170 (PROFIBUS, WorldFIP and P-Net) in order to include new important functionalities. Within the project, the architecture of RFieldbus communication network was developed, based on the PROFIBUS architecture, and a test application (MAF – Manufacturing Application Fieldtrial) based on typical discrete parts manufacturing system has been implemented in the Critical Systems Lab of the IPP-HURRAY Research Group. In this test application, system components are interconnected by an RFieldbus communication network, which comprises and demonstrates the architecture novel functionalities: wireless communication, efficient control traffic scheduling algorithm, multimedia traffic integration and wireless devices mobility.

RFieldbus, as an industrial communication network, is embedded in distributed systems available at the manufacturing floor, thus its behaviour affects the overall system performance. For this reason, the study of the communication network behaviour is important, since it will contribute to the understanding of the distributed system behaviour. A software application has been developed within the RFieldbus Project to compute the worst case behaviour of an RFieldbus network. The application has been used to compute this response for the MAF topology. Later, the MAF behaviour has been observed and both responses have been compared in order to understand the pessimism introduced by worst case analysis.

A discrete-event simulator (LLRS - Lower Layer RFieldbus Simulator) was developed aiming the simulation of RFieldbus typical timing behaviour. The developed model, which was implemented in the simulator, incorporates device behaviour and parameters, network and message cycles that significantly contribute to the RFieldbus network behaviour. The simulator was validated with the observed data on the MAF application, and allows, in a simple and inexpensive way, to study RFieldbus network behaviour considering different elements, topologies, message cycles and parameters.

This dissertation presents the underlying theoreticians, the model, the simulator LLRS, the obtained results and their validation.

Keywords: *Fieldbus*, *communication networks*, *Discrete-Event Simulation*, *Distributed Computer Controlled Systems*, *Real-time systems*, RFieldbus, PROFIBUS (EN50170), OMNeT++.

Résumé

L'un des objectifs les plus importants dans l'analyse des systèmes temps-réel distribués est de prévoir le comportement du système. Il existe principalement deux approches : L'analyse du pire cas et l'analyse du comportement typique, qui permettent d'extraire certaines caractéristiques complémentaires du système pour la classification de son comportement. D'une part, l'analyse du pire cas permet de prévoir un comportement minimal du système ; mais d'autre part induit un pessimisme important sur les résultats qui sont souvent loin du comportement réel du système. Cependant, un comportement plus réaliste du comportement du système peut être obtenu par recours à la technique de Simulation à Evénements Discrets.

Le projet Européen Rfieldbus (IST-1999-11316) a visé le développement d'une architecture parmi les autres intégrées dans la norme européenne EN50170 (PROFIBUS, P-Net et WorldFIP) afin d'inclure de nouvelles fonctionnalités importantes. Dans le cadre de ce projet, l'architecture du réseau de communication de RFieldbus a été développée en se basant sur l'architecture de PROFIBUS et une application d'essai (MAF – *Manufacturing Automation Fieldtrial*) basée sur le système typique de fabrication de pièces. Cette application a été mise en oeuvre dans le laboratoire des systèmes critiques du groupe de recherche IPP-HURRAY. Dans cette application d'essai, les composants du système sont interconnectés par un réseau de communication RFieldbus qui comporte et démontre les fonctionnalités importantes de l'architecture de PROFIBUS, telles que support de communications sans fil, l'algorithme efficace du contrôle du trafic commande, l'intégration du trafic de multimédia et la gestion de la mobilité des dispositifs sans fil.

Etant un réseau de terrain industriel, RFieldbus est embarqué dans les systèmes répartis au niveau du plancher de fabrication, et son comportement affecte la performance du système global. Pour cette raison, la classification du comportement des réseaux de communication est importante dans la mesure où elle contribue à la classification du comportement global du système distribué. Une application logicielle a été développée dans le projet de RFieldbus pour calculer le comportement du pire cas d'un réseau RFieldbus. Cette application a été utilisée pour calculer les temps de réponses dans une topologie de type MAF. Ultérieurement, nous avons observé le comportement de MAF et les deux approches d'analyses du comportement ont été comparées pour évaluer le pessimisme induit par analyse du pire cas.

L'objectif de l'outil de simulation développé (LLRS - *Lower Layer RFieldbus Simulator*) est de simuler le comportement le plus probable d'un réseau RFieldbus. Le modèle développé mis en oeuvre dans le simulateur incorpore le comportement et les paramètres des dispositifs, les cycles des messages et du réseau qui traduisent d'une manière réaliste le comportement du réseau RFieldbus. Ce simulateur fournit une solution simple, efficace à moindre coût pour étudier le comportement des réseaux RFieldbus considérant différents éléments, topologies, cycles de messages et paramètres.

Ce manuscrit présente les approches théoriques d'analyses du comportement, le modèle, le simulateur LLRS, les résultats obtenus et leur validation.

Mots-clés: *Fieldbus, les réseaux de transmission, simulation à evénements discrets, systèmes commandés par ordinateur répartis, systèmes en temps réel, RFieldbus, PROFIBUS (EN50170), OMNeT++.*

Para ser grande, sê inteiro: nada
teu exagera ou exclui.
Sê todo em cada coisa. Põe quanto és
no mínimo que fazes.
Assim em cada lago a lua toda
brilha, porque alta vive.

Fernando Pessoa

Agradecimentos

Em primeiro lugar gostava de agradecer aos meus orientadores, Dr. Luís Pinho e Dr. Pedro Souto, pela supervisão e revisão do meu trabalho, pela amizade e paciência nos momentos mais difíceis.

Aos membros do júri agradeço o tempo e esforço investido na avaliação desta dissertação.

Um agradecimento a todos os colegas e amigos do grupo de investigação IPP-HURRAY pelo seu trabalho, exemplo e companheirismo.

Em especial, ao Dr. Mário Alves, Dr. Luis Ferreira, Dr. Eduardo Tovar, pelas valiosas trocas de ideias que contribuíram para o sucesso deste trabalho. Ao Luis Marques, Filipe Pacheco e ao Paulo Sousa agradeço a amizade e as interessantes trocas de ideias.

Ao Engº (s) Sousa Guimarães, Rui Guedes de Azevedo, José Carlos de Oliveira, José Carlos Portela, Pedro Assis, Paula Viana e Dr. Jorge Mamede pela confiança e amizade, a todos os colegas de grupo de disciplinas e de departamento pela amizade e incentivo.

Ao José Luis Ramalho e Fernando Fontes pela amizade e uma saudação aos colegas de mestrado.

Aos meus colegas e amigos um agradecimento por tornarem a minha vida mais agradável.

Ao Rui Martins e Branca Silva um obrigado pelas interessantes tertúlias e amizade.

A minha avó Mariazinha, pelo seu amor incondicional.

Aos meus primos Paulo Rodrigues, Fernando Elísio, Tildinha, António Pádua, Maria Adelaide e aos meus tios António Lopes e Rosinha, agradeço o seu exemplo e incentivo.

À Susana um obrigado pela amizade e apoio neste período.

Um agradecimento aos meus pais pela sua presença e encorajamento constante em toda a minha vida. De uma forma especial agradeço o apoio, que me ofereceram neste período.

Índice

1 CAPÍTULO - INTRODUÇÃO 1

1.1 CONTEXTUALIZAÇÃO 1
1.1.1 PROFIBUS 2
1.1.2 RFieldbus 3
1.1.3 Estudo da arquitectura da rede de comunicação 3
1.1.4 Ferramentas utilizadas 4
1.2 CONTRIBUTOS DO TRABALHO 4
1.3 ESTRUTURA DA DISSERTAÇÃO 5

2 CAPÍTULO - PROFIBUS 7

2.1 INTRODUÇÃO 7
2.2 ARQUITECTURA DA REDE DE COMUNICAÇÃO PROFIBUS 8
2.3 CAMADA FÍSICA (PHY) 10
2.3.1 Interface da camada PHY 11
2.3.2 Especificação do protocolo da camada PHY 12
2.3.3 Descrição do RS-485 12
2.4 CAMADA DE LIGAÇÃO DE DADOS FDL 13
2.4.1 Formato das tramas 13
2.4.2 Interface da camada FDL 16
2.4.3 Protocolo da camada FDL 17
2.4.4 Endereços 18
2.4.5 Ciclo de mensagem 18
2.4.6 Slot time 20
2.4.7 Idle time 20
2.4.8 Recepção do Token 21
2.4.9 Transmissão do Token 22
2.4.10 GAP Maintenance 23
2.4.11 Algoritmo de despacho de mensagens 24
2.4.12 Máquina de estados do controlador da FDL 27
2.5 CAMADAS SUPERIORES 30
2.5.1 PROFIBUS-DP 30
2.5.2 Arquitectura Protocolar PROFIBUS-DP 30
2.5.3 Estações PROFIBUS-DP 31
2.5.4 Modelo de comunicação PROFIBUS-DP 32
2.5.5 Funções disponibilizadas ao utilizador de cada estação 32
2.5.6 Controlo de acesso ao meio físico 34
2.5.7 Configurações de rede em PROFIBUS-DP 35
2.6 SUMÁRIO 36

3 CAPÍTULO - RFIELDBUS 37

3.1 INTRODUÇÃO 37
3.1.1 Elementos constituintes 38
3.1.2 Wired e Wireless Communication Domains 42
3.2 TOPOLOGIAS 42
3.2.1 Direct Link Network 42
3.2.2 Relay Link Network 44
3.3 MOBILIDADE 45

3.3.1 Intra-cell mobility *46*
3.3.2 Inter-cell mobility *46*
3.4 REDES ESTRUTURADAS 49
3.5 ARQUITECTURA DA REDE DE COMUNICAÇÃO RFIELDBUS 50
3.5.1 Formato do PDU da camada física Wireless *51*
3.5.2 Physical Layer *51*
3.5.3 Data Link Layer *52*
3.5.4 Camadas Superiores *54*
3.6 SUMÁRIO 58

4 CAPÍTULO - SIMULAÇÃO POR OMNET++ 59

4.1 INTRODUÇÃO 59
4.2 SIMULAÇÃO DE SISTEMAS DE EVENTOS DISCRETOS 60
4.2.1 Princípios de simulação de Sistemas de Eventos Discretos (DES) *60*
4.2.2 Mecanismos para o avanço do tempo *60*
4.2.3 Abordagens na descrição da lógica *61*
4.3 DESCRIÇÃO DO AMBIENTE DE SIMULAÇÃO 62
4.3.1 Características gerais *62*
4.3.2 Módulos simples e compostos *62*
4.3.3 Hierarquia de módulos, ... *62*
4.3.4 Mensagens, Gates e Conexões *63*
4.3.5 Parâmetros *64*
4.4 MÓDULOS SIMPLES 64
4.4.1 Modo de funcionamento *64*
4.4.2 Paradigmas de programação *64*
4.4.3 Funções membro virtuais mais utilizadas *65*
4.5 DESENVOLVIMENTO E EXECUÇÃO (*BUILD* AND *RUN*) DO SIMULADOR 65
4.6 SUMÁRIO 66

5 CAPÍTULO - MODELO DE SIMULAÇÃO DEFINIDO PARA CADA COMPONENTE RFIELDBUS 69

5.1 INTRODUÇÃO 69
5.2 RFIELDBUS *MASTER* 70
5.2.1 Estados do controlador da FDL *70*
5.2.2 Classes de tráfego RFieldbus *71*
5.2.3 Tráfego das camadas superiores para tráfego FDL do RFieldbus *71*
5.2.4 Classes de tráfego FDL implementadas *71*
5.2.5 Escalonamento de tráfego *72*
5.2.6 Processamento de cada ciclo de mensagem *73*
5.2.7 Transmissão do token *73*
5.2.8 Recepção do Token *73*
5.3 RFIELDBUS *SLAVE* 73
5.4 RFIELDBUS MOBILITY *MASTER* 73
5.5 RFIELDBUS WIRED/WIRELESS DOMAIN 74
5.6 PHYSICAL MEDIA - WIRED/WIRELESS 74
5.7 RFIELDBUS INTERMEDIATE SYSTEMS 76
5.7.1 Link Station *76*
5.7.2 Base Station *76*
5.7.3 Link Base Station *76*
5.8 SUMÁRIO 77

6 CAPÍTULO - IMPLEMENTAÇÃO DO MODELO EM OMNET++ 79

6.1 INTRODUÇÃO 79
6.2 MENSAGEM DEFINIDA PARA O SIMULADOR LLRS 80
6.3 MÓDULOS IMPLEMENTADOS 81
6.4 MÓDULO COMPOSTO *MASTER* 81
6.4.1 Funcionalidade implementada 81
6.4.2 Estrutura do módulo composto 82
6.4.3 Sub-módulo FDL User (simple module) 83
6.4.4 Sub-módulo Mqueues (simple module) 85
6.4.5 Sub-módulo MAC (simple module) 86
6.4.6 Sub-módulo Phy (simple module) 90
6.5 MÓDULO COMPOSTO *SLAVE* 92
6.5.1 Funcionalidade implementada 92
6.5.2 Estrutura do módulo composto 92
6.5.3 Sub-módulo Gerresp (simple module) 93
6.5.4 Sub-módulo Phy (simple module) 94
6.6 MÓDULO COMPOSTO *SLAVEMOB* 96
6.6.1 Funcionalidade implementada 96
6.6.2 Estrutura do módulo composto 96
6.6.3 Sub-módulo Chanaccess (simple module) 97
6.7 DESCRIÇÃO DOS MÓDULOS SIMPLES 99
6.7.1 Mobility Master 99
6.7.2 Meio físico Wired 100
6.7.3 Meio físico Wireless 102
6.7.4 Base Station (Intermediate System) 104
6.7.5 Link Station (Intermediate System) 105
6.7.6 Link Base Station (Intermediate System) 107
6.8 SUMÁRIO 108

7 CAPÍTULO - CASO DE ESTUDO, APRESENTAÇÃO DE RESULTADOS 109

7.1 INTRODUÇÃO 109
7.2 *MANUFACTURING AUTOMATION FIELDTRIAL* (MAF) 109
7.3 TOPOLOGIA DA REDE A SIMULAR 110
7.4 PARÂMETROS QUE CARACTERIZAM OS ELEMENTOS DA REDE DE COMUNICAÇÃO 110
7.4.1 Master 111
7.4.2 Slave 111
7.4.3 Meio físico Wired 111
7.4.4 Meio físico Wireless 112
7.4.5 LBS 112
7.4.6 MobM 112
7.5 O CONJUNTO DOS CICLOS DE MENSAGEM OBSERVADOS 112
7.6 O ANALISADOR DE REDE PROFIBUS (PROFIBUS *ANALYSER*) 113
7.6.1 Âmbito do uso do PROFIBUS Analyser no MAF 113
7.6.2 O que permite observar o analisador de rede PROFIBUS? 113
7.7 CÁLCULO DA DURAÇÃO DOS CICLOS DE MENSAGEM 114
7.7.1 Duração de um ciclo de mensagem com resposta/ack 114
7.7.2 Duração de um ciclo de mensagem sem resposta/ack 114
7.8 RESULTADOS OBTIDOS 115
7.8.1 Ciclo de mensagem S1 115
7.8.2 Ciclo de mensagem S2 116

7.8.3 Ciclo de mensagem S3 ..... *118*
7.8.4 Ciclo de mensagem S4 ..... *119*
7.8.5 Ciclo de mensagem S5 ..... *119*
7.8.6 Passagem do Token *120*
7.8.7 Parâmetro Slot time *120*
7.8.8 Parâmetro Idle Time T_{ID1} *121*
7.9 AVALIAÇÃO 121
7.10 SUMÁRIO 122

8 CAPÍTULO - CONCLUSÕES E TRABALHO FUTURO 123

8.1 SUMÁRIO DA DISSERTAÇÃO 123
8.1.1 Estudo da arquitectura RFieldbus *123*
8.1.2 Modelo definido para as camadas mais baixas *124*
8.1.3 Implementação do modelo em OMNeT++ *125*
8.1.4 Teste e validação do simulador *125*
8.2 CONCLUSÕES 125
8.2.1 PROFIBUS *126*
8.2.2 RFieldbus *126*
8.2.3 Simulação de DES e OMNeT++ *127*
8.2.4 Validação dos objectivos do trabalho *128*
8.3 TRABALHO FUTURO 129
8.3.1 Observar simultaneamente ambos os meios físicos Wired e Wireless *129*
8.3.2 Modelar as camadas superiores e os estados transitórios *129*
8.3.3 Efectuar este estudo em outras redes de comunicação industriais. *129*

REFERÊNCIAS 131

ACRÓNIMOS 133

SÍMBOLOS 137

ANEXO A - MAF - MANUFACTURING AUTOMATION FIELDTRIAL 139

A.1 SISTEMA MECÂNICO 139
A.1.1 Componentes constituintes *139*
A.1.2 Funcionalidade *140*
A.1.3 Velocidades e dimensões relevantes *141*
A.2 A REDE DE COMUNICAÇÃO 141
A.3 CICLOS DE MENSAGEM DP 142
A.3.1 SCADA *142*
A.3.2 DP Sound Slave *143*
A.3.3 Robuter DP Slave Application *143*
A.3.4 RC2 DP Slave Application *143*
A.3.5 MM1/MM2 ..... *144*
A.3.6 ET1/ET2 ..... *144*
A.3.7 SA1/SA2 ..... *144*
A.4 CICLOS DE MENSAGEM *IP* 145
A.4.1 Disk Colour Detection Master/ Disk Colour Detection Slave *145*
A.4.2 Image Centre/ Image Client *146*
A.4.3 Robot Arm Operator Master/ Robot Arm Operator Slave *146*
A.4.4 Intranet Server *146*
A.4.5 Voice Application *146*

A.5 QUADRO RESUMO DAS CARACTERÍSTICAS DE CADA *MESSAGE STREAM* 147

ANEXO B - DOCUMENTAÇÃO DO MODELO DE SIMULAÇÃO 149

B.1 LISTA DAS CLASSES IMPLEMENTADAS NO SIMULADOR 149
B.2 DESCRIÇÃO DAS CLASSES 149
B.2.1 Classe Mac 149
B.2.2 Classe Mqueues 150
B.2.3 Classe Phy 150
B.2.4 Fdluser 150
B.2.6 Ls 151
B.2.7 Lbs 151
B.2.8 Bs 151
B.2.9 Phywired 151
B.2.10 Phywireless 151
B.2.11 MobM 152
B.2.12 Chanaccess 152

ANEXO C - ESTRUTURA DE MENSAGEM OMNET++ 153

Lista de figuras

Figura 1 - Arquitectura do protocolo de comunicação PROFIBUS 9
Figura 2 - Relação temporal das primitivas do PHY 11
Figura 3 - Carácter UART 13
Figura 4 - *Frames of fixed Length with no Data Field* 13
Figura 5 - *Short Acknowledgement Frame* 14
Figura 6 - *Frames of fixed Length with Data Field* 15
Figura 7 - *Frames* with variable Data *Field* Length 15
Figura 8 - *Token Frame* 15
Figura 9 - Codificação do octeto de endereços 18
Figura 10 - Ciclo de mensagem com resposta 19
Figura 11 - *Slot time* 20
Figura 12 - T_{ID1}, idle time nos ciclos de mensagem com resposta/Ack 20
Figura 13 - T_{ID2}, idle time nos ciclos de mensagem sem resposta/Ack 21
Figura 14 - Recepção do *Token* 22
Figura 15 - Transmissão do *Token* 23
Figura 16 - Processamento de ciclos de mensagem 26
Figura 17 - Máquina de estados do controlador da FDL do *Master* 29
Figura 18 - Arquitectura Protocolar PROFIBUS-DP 31
Figura 19 - Modelo de comunicação 32
Figura 20 - Controlo do acesso ao meio no PROFIBUS DP 34
Figura 21 - Exemplo de um sistema *Mono-Master* em PROFIBUS-DP 35
Figura 22 – Exemplo de um sistema *Multi-Master* em PROFIBUS-DP 36
Figura 23 - Estrutura de uma RFieldbus *Link Station* 39
Figura 24 - Instante para iniciar a retransmissão da trama 39
Figura 25 - Estrutura de uma *Base Station* com um Radio Front-End 40
Figura 26 - Estrutura de uma *Base Station* com dois Radio Front-End 41
Figura 27 - Estrutura de uma *Link Base Station* 41
Figura 28 - Definição de uma célula rádio numa Direct *Link Network* 43
Figura 29 - Extended Direct *Link Network* 43
Figura 30 - Definição de uma célula rádio numa *Relay Link Network* 44
Figura 31 - *Relay Link Network* com segmentos *Wired* 45
Figura 32 - *Relay Link Network* com um segmento *Wired* e LBS 45
Figura 33 - Transição entre domínios estruturados (*Handoff*) 47
Figura 34 - Diagrama temporal do procedimento de gestão da mobilidade 48
Figura 35 - Concatenação de 3 Direct *Link Network* 49
Figura 36 - Concatenação de 3 segmentos *Wired* com BS 49
Figura 37 - Arquitectura do protocolo de comunicações para estações RFieldbus (ES) 50
Figura 38 - Protocol Data Unit (PDU) da camada física rádio 51
Figura 39 - Interface da camada DLX 53
Figura 40 - Camada 2 (FDL) na arquitectura RFieldbus 53
Figura 41 - Filas de mensagens num RFieldbus Master classe 2 55
Figura 42 - Esquema funcional da camada IP ACS 56
Figura 43 - Estrutura do IP *Mapper* 57
Figura 44 - Hierarquia de módulos em OMNeT++ 63
Figura 45 – *Building* and *Running* a simulation 66
Figura 46- Hierarquia dos componentes Rfieldbus 69
Figura 47 - Estrutura do módulo composto *Master* 82
Figura 48 - Estrutura do módulo simples FDL User 83

Figura 49 - Fluxograma da funcionalidade implementada no módulo simples FDL User..... 84
Figura 50 - Estrutura do módulo simples Mqueues..... 85
Figura 51 - Fluxograma da funcionalidade implementada no módulo simples Mqueues 86
Figura 52 - Estrutura do módulo simples MAC 87
Figura 53 - Fluxograma da funcionalidade implementada no módulo simples MAC..... 88
Figura 54 - Fluxograma da funcionalidade implementada no estado *Wait Token*..... 88
Figura 55 - Fluxograma da funcionalidade implementada no estado *Detem Token*..... 89
Figura 56 - Fluxograma da funcionalidade implementada no estado Passa *Token* 90
Figura 57 - Estrutura do módulo simples Phy 91
Figura 58 -Fluxograma da funcionalidade implementada no módulo simples Phy..... 91
Figura 59 - Estrutura do módulo composto *Slave* 92
Figura 60 - Estrutura do módulo simples Gerresp..... 93
Figura 61 - Fluxograma da funcionalidade implementada no módulo simples Gerresp 94
Figura 62 - Estrutura do módulo simples Phy 95
Figura 63 - Fluxograma da funcionalidade implementada no módulo simples Phy..... 95
Figura 64 - Estrutura do módulo composto *SlaveMob* 96
Figura 65 - Estrutura do módulo simples Chanaccess..... 97
Figura 66 - Fluxograma da funcionalidade implementada no módulo simples Chanaccess ... 98
Figura 67 - Estrutura do módulo simples MobM 99
Figura 68 - Fluxograma da funcionalidade implementada no módulo simples MobM..... 100
Figura 69 - Estrutura do módulo simples PhyWired 101
Figura 70 - Fluxograma da funcionalidade implementada no módulo simples PhyWired.... 101
Figura 71 - Estrutura do módulo simples PhyWireless 102
Figura 72 - Fluxograma da funcionalidade implementada no módulo simples PhyWireless 103
Figura 73 - Estrutura do módulo simples *Base Station* 104
Figura 74 - Parâmetro do módulo simples *Base Station* 104
Figura 75 - Fluxograma da funcionalidade implementada no módulo simples *Base Station* 105
Figura 76 - Estrutura do módulo simples *Link Station*..... 106
Figura 77 - Fluxograma da funcionalidade implementada no módulo simples *Link Station* 106
Figura 78 - Estrutura do módulo simples *Link Base Station* 107
Figura 79 - Fluxograma da funcionalidade do módulo simples *Link Base Station* 108
Figura 80 - *Manufacturing Automation Fieldtrial* RFieldbus *Network topology*..... 110
Figura 81 - Duração de um ciclo de mensagem com resposta/ack..... 114
Figura 82 - Duração de um ciclo de mensagem sem resposta/ack 115
Figura 83 - Ciclo de mensagem com iniciator e responder no meio físico WR1 116
Figura 84 – Esquema do sistema mecânico..... 139
Figura 85 – Topologia de rede do MAF..... 141
Figura 86- Aplicações DP 142
Figura 87 - Aplicações multimédia (IP)..... 145

Lista de tabelas

Tabela 1- Perfis de comunicação e de aplicação PROFIBUS 10
Tabela 2 – Classes de tráfego da FDL em PROFIBUS 24
Tabela 3 - Componentes RFieldbus 69
Tabela 4 - Classes de tráfego RFieldbus 71
Tabela 5 - Correspondência do tráfego RFieldbus em tráfego nativo da FDL PROFIBUS 71
Tabela 6 - Classes de tráfego da FDL em PROFIBUS 71
Tabela 7 - Tempo de propagação e duração da transmissão da trama mais pequena 75
Tabela 8 - Campos de dados definidos na mensagem OMNeT++ 80
Tabela 9 - Campos de controlo definidos na mensagem OMNeT++ 80
Tabela 10 - Módulos Implementados em OMNeT++ 81
Tabela 11 - Parâmetros do módulo simples Gerresp 93
Tabela 12 - Parâmetros do módulo simples MobM 99
Tabela 13 – Parâmetros do módulo simples PhyWired 101
Tabela 14 - Parâmetros do módulo simples PhyWireless 103
Tabela 15- Parâmetro do módulo simples *Link Station* 106
Tabela 16 - Parâmetro do módulo simples *Link Base Station* 107
Tabela 17 - Valores dos parâmetros do módulo *Master* no caso de estudo MAF 111
Tabela 18 - Valores dos parâmetros do módulo *Slave* no caso de estudo MAF 111
Tabela 19 - Valores dos parâmetros do módulo PhyWired no caso de estudo MAF 111
Tabela 20 - Valores dos parâmetros do módulo PhyWireless no caso de estudo MAF 112
Tabela 21 - Valor do parâmetro do módulo LBS no caso de estudo MAF 112
Tabela 22 - Valores dos parâmetros do módulo MobM no caso de estudo MAF 112
Tabela 23 - Valores simulados e observados para as message streams do MAF 122
Tabela 24 - Outros valores simulados/observados no MAF 122
Tabela 25 - Velocidades e dimensões relevantes 141
Tabela 26- Abreviaturas da figura da topologia de rede do MAF 142
Tabela 27 - Ciclos de mensagem do MAF 147

Capítulo 1

Introdução

1.1 Contextualização

A globalização da produção de bens e serviços, assim como a globalização do seu comércio, provocou um alargamento dos mercados e um aumento do seu volume, os investidores foram atraídos e com eles a pressão para aumentar os resultados líquidos de operação.

Durante muito tempo duas regras de ouro regiam muitos mercados: aumentar as vendas e criar necessidades de consumo [1]. Actualmente um novo paradigma foi adicionado, a eficiência da empresa enquanto organização. Existem diversas formas de atingir essa eficiência, mas o conceito por trás é a simplificação e controlo global do processo, com o objectivo de manter a organização mais forte, rápida, adaptável, com menores custos de operação face à concorrência e claro manter a confiança dos investidores [1]. Esta pressão constante dos investidores expressou-se na indústria de diversas formas. Uma das mais importantes foi a crescente automatização industrial.

A automatização industrial é um fenómeno suportado pela tecnologia disponível no momento. Durante a revolução industrial foi a tecnologia das máquinas a vapor, mais tarde foram os motores eléctricos, a hidráulica, a pneumática e hoje são as tecnologias de informação.

Este percurso começa com o desenvolvimento de ferramentas para o desempenho de tarefas específicas. Cada ferramenta evolui e surgem novas ferramentas, mas a necessidade de optimizar processos produtivos leva ao desenvolvimento de ferramentas e máquinas cada vez mais complexas. A determinada altura uma planta fabril é constituída pelo conjunto de máquinas (sistemas autónomos) sendo cada uma, responsável por uma ou mais operações do processo produtivo.

Mas a optimização dos processos de fabrico provoca a integração das máquinas/ferramentas entre si, inicialmente através de ligações eléctricas, depois por redes de comunicação industriais, permitindo a sincronização e comunicação entre as diferentes máquinas/ferramentas que passam a cooperar de uma forma mais inteligente entre si, aumentando a produtividade.

O que era um conjunto de sistemas autónomos passa a ser um sistema distribuído [2].

No âmbito dos sistemas distribuídos as redes de comunicação têm uma importância vital pois permitem a sincronização e comunicação entre os diferentes componentes do sistema. A sua eficácia reflecte-se na eficácia do sistema, as funcionalidades oferecidas reflectem as possibilidades que os diferentes componentes dispõem em termos de comunicação.

Porque os sistemas distribuídos dos quais as redes de comunicação industriais são parte integrante, tem muitas vezes requisitos de tempo real, a caracterização do comportamento temporal destas redes de comunicação é importante para que as mensagens trocadas permitam a produção dos resultados correctos antes de um determinado instante de tempo (*deadline*) [3]. Para que se adaptem à utilização em sistemas de tempo real devem ser rápidas, fiáveis e adaptativas [3].

As redes de comunicação industriais são caracterizadas por meios físicos e conectores robustos, pela capacidade de operar em ambientes industriais com ruído e forte interferência electromagnética. Existem redes de comunicação industriais que se destinam a suportar a comunicação dos elementos constituintes de uma máquina industrial (ex. braço robótico), de uma célula ou de toda a planta fabril.

As redes de comunicação industriais surgem com objectivos de controlo, mas na era da informação tentam acompanhar este paradigma. Um manancial de informação é gerado no sistema ao longo de todo processo produtivo [1]. Existe agora a possibilidade de recolher essa informação de forma automatizada, que pode, instantaneamente ou posteriormente ser utilizada para aferir parâmetros de produtividade, de qualidade do processo de fabrico ou do produto, torna-se num dos mais importantes factores que contribuem para o uso de redes de comunicação industrial.

O conhecimento do processo de fabrico permite o controlo e optimização do mesmo. A evolução do processo de fabrico garante uma vantagem competitiva face à concorrência [4].

Assim, a rede de comunicação industrial enquanto parte integrante do sistema distribuído contribui com as suas funcionalidades para o aumento da eficiência desse sistema, a sua flexibilização, e o aumento do controlo sobre o processo.

1.1.1 PROFIBUS

O PROFIBUS [5] é a rede de campo líder nas áreas de *Manufacturing Automation* e *Process Control*, com uma cota de mercado acima dos 20% [5]. Dado que se encontra normalizado nas normas *Fieldbus Standards* EN 50170 [6] e IEC 61158 [7], garante abertura e estabilidade para os utilizadores e vendedores. O âmbito de aplicação desta rede é ao nível de célula (*cell level*) e de campo (*Field level*).

Tecnicamente o PROFIBUS oferece uma vasta gama de possibilidades de comunicação; das quais se destacam os dois perfis de comunicação PROFIBUS-FMS [8] [9] e PROFIBUS-DP [10] e diversos perfis de aplicação.

O PROFIBUS-*FMS* é uma solução que oferece um espectro alargado de funcionalidades e flexibilidade. No entanto, para assegurar a transmissão mais rápida e eficiente de dados, a arquitectura PROFIBUS tem especificado um outro perfil de comunicação chamado PROFIBUS-*DP*, que foi tornado mais eficiente através da simplificação da arquitectura. Foi retirada a camada de aplicação (*layer* 7), sendo utilizadas apenas as camadas física (*layer* 1), de ligação de dados (*layer* 2) e a camada DDLM (*Direct Data Link Mapper*) que permite ao utilizador um acesso mais confortável às funções de transferência de dados da FDL (*Fieldbus Data Link*).

O PROFIBUS-DP destina-se a comunicações de tempo é crítico entre sistemas de automação e periféricos distribuídos. Os perfis de aplicação PROFIBUS são compromissos sobre o uso de serviços não obrigatórios e parâmetros de meio físico para áreas de aplicação específicas.

Em PROFIBUS estão definidas dois tipos de estações, *Masters* e *Slaves* [11]. Os *Master* são estações activas que podem aceder ao meio físico por iniciativa própria quando possuem o *Token*. Os *Slaves* são estações passivas que acedem ao meio físico quando requisitadas por um *Master*. A trama especial que representa o *Token* é trocada entre estações *Master* de forma circular garantindo à estação detentora do *Token* o acesso ao meio físico durante o período em que detém o mesmo. O algoritmo controlo de acesso ao meio é uma versão simplificada do *Timed Token*, que é uma solução eficiente para sistemas de tempo real [12]. Outra característica interessante é distinguir entre

mensagens de alta e de baixa prioridade, que permite diferenciar prioridades do tráfego e tem definidas tramas de tamanhos desde 1 até 255 caracteres FDL, permitindo enviar um *acknowledge* de uma forma eficiente ou enviar mais dados, sem ou recorrendo a pouca segmentação. As comunicações ocorrem em ciclos de mensagem, que são constituídos por uma *action frame* (*request*) enviada pelo *initiator* (*Master*) e um A*cknowledge/Response* enviado pelo responder (*Master* ou *Slave*).

O protocolo de acesso ao meio, os serviços de transferência de dados e os serviços de gestão estão definidos de acordo com as normas DIN 19 241-2, IEC 995, ISO 8802-2 e ISO/IEC JTC 1/SC 4960. A versão 1 da especificação da camada física do PROFIBUS está de acordo com a norma EIA RS-485.

1.1.2 RFieldbus

O RFieldbus [13] (*High Performance Wireless Fieldbus in Industrial Multimédia-Related Environment*) é uma rede de campo (*serial fieldbus*), cuja arquitectura foi definida no âmbito do projecto europeu RFieldbus (IST-1999-11316) e teve como objectivos a expansão das funcionalidades oferecidas pelo PROFIBUS; a introdução do suporte às comunicações em meio físico sem fios (DSSS - *Direct Sequence Spread Spectrum* a 2.4GHz), e o suporte à mobilidade de dispositivos e a integração do tráfego multimédia na arquitectura (TCP/IP).

Esta expansão das funcionalidades implica uma total compatibilidade da funcionalidade nativa PROFIBUS. Falar em RFieldbus é falar em PROFIBUS com as novas funcionalidades, de tal forma que ambos comunicam numa rede RFieldbus (de acordo com as funcionalidades disponíveis em cada estação).

Para além das estações RFieldbus *Master* e *Slave* compreendendo a aplicação PROFIBUS, foram definidas três novas estações: A RFieldbus *Link Station* (interliga os meios físicos com e sem fios), *a* RFieldbus *Base Station* (maximiza a cobertura rádio e aumenta a fiabilidade das comunicações), *e a* RFieldbus *Link Base Station* (combina as funcionalidades das duas estações anteriores) [13].

O RFieldbus permite a criação de redes com domínios de comunicação sem fios, que podem ser estruturadas em vários domínios de comunicação (com e sem fios). A solução de mobilidade definida nesta arquitectura permite não só a mobilidade dos dispositivos sem fios dentro do domínio (*Intra-cell mobility*) como também a transição entre domínios sem fios (*Inter-cell mobility*) [13].

Sobre a camada de ligação lógica (FDL) do RFieldbus foi definida uma estrutura de camadas que suportam a integração do tráfego TCP/IP no tráfego nativo PROFIBUS.

1.1.3 Estudo da arquitectura da rede de comunicação

O trabalho apresentado nesta dissertação consiste no estudo do funcionamento da arquitectura RFieldbus, em especial das camadas, física e de ligação lógica com o objectivo de definir um modelo de simulação (capítulo 5), e implementá-lo num simulador (LLRS) do comportamento temporal destas camadas (capítulo 6).

Depois de desenvolvido e validado o LLRS permite estudar através de simulação o comportamento temporal de outras topologias de rede ou de diferentes perfis de tráfego.

No estudo do comportamento de sistemas com requisitos temporais é frequentemente utilizada a análise do pior caso (*Worst Case*) [12]. Este método tem a vantagem de garantir que em todos os casos o sistema vai ser capaz de reagir em tempo real e satisfazer as *deadlines* impostas por cada evento. A grande desvantagem do método, deriva do seu pessimismo. A resposta temporal determinada por este método tem baixa probabilidade de ocorrer na realidade, conduzindo ao estudo de um comportamento pouco frequente. Já através do modelo de simulação implementado no simulador LLRS é possível observar um comportamento temporal usual na rede de comunicações.

De referir que esta distinção não constitui uma forma de eleger um dos métodos como preferível, mas sim distinguir o âmbito de aplicação de cada um.

1.1.4 Ferramentas utilizadas

No decurso do trabalho o modelo de simulação foi inicialmente aplicado recorrendo à ferramenta de simulação ns-2 [14], especializada na simulação de redes de comunicação IP, e que contém um alargado conjunto de funcionalidades implementadas neste domínio. Inicialmente pensou-se que seria realizável criar novos módulos específicos para simular as camadas baixas do RFieldbus. No entanto após alguns testes verificou-se que esta ferramenta não era a mais eficaz, pois este não é seu campo de aplicação, sendo necessário a adição de novos mecanismos para a simulação de redes não IP.

Desta forma, foi escolhida uma outra ferramenta de simulação de eventos discretos, o OMNeT++ [15], que apresenta um nível de abstracção dos mecanismos de simulação de eventos discretos mais elevado e cuja flexibilidade permite a implementação com um nível de detalhe suficiente para a implementação dos modelos definidos no LLRS.

1.2 Contributos do trabalho

As redes de comunicação industriais são um dos subsistemas constituintes de um sistema distribuído de fabrico. O desempenho de um sistema distribuído resulta da combinação do desempenho de cada um dos subsistemas que o constituem. Assim, a caracterização do desempenho de operação das redes de comunicação industriais é muito relevante pois permite a caracterização do desempenho do sistema distribuído no qual esta será integrada.

O RFieldbus é uma rede de comunicação industrial que deriva do PROFIBUS. Estende as funcionalidades desta conhecida rede de campo, com forte implementação no mercado, permitindo as comunicações sem fios e a integração de tráfego multimédia.

Uma excelente forma de estudar o desempenho de uma rede de comunicação é desenvolver um simulador. A vantagem principal é que após validação o simulador permite uma multiplicidade de estudos, sem necessidade de recorrer a implementações onerosas.

Uma rede de comunicação é um sistema no qual as mudanças do seu estado ocorrem em instantes discretos de tempo. É portanto um sistema de tempo discreto e a sua simulação faz-se implementando um simulador de eventos discretos. Existem diversas ferramentas vocacionadas para a implementação de simuladores de eventos discretos, no desenvolvimento do *Lower Layer* RFieldbus *Simulator* (LLRS) foi utilizado o ambiente OMNeT++.

Desta forma, as principais contribuições do trabalho apresentado nesta dissertação são: a definição de um modelo de simulação das camadas baixas do RFieldbus, a implementação do modelo definido no simulador LLRS recorrendo ao ambiente de simulação de eventos discretos

OMNeT++ e o teste e validação do simulador através da observação do comportamento temporal da aplicação de teste *Manufacturing Automation Fieldtrial* (Anexo A) [16]. Uma contribuição complementar, mas também importante, é o estudo efectuado ao Profibus, apresentado no capítulo 2.

1.3 Estrutura da dissertação

Esta dissertação foi estruturada em 8 capítulos. O capítulo 2 faz uma descrição geral da tecnologia PROFIBUS, apresentando a arquitectura da rede de comunicação com destaque à tecnologia PROFIBUS utilizada no RFieldbus (PROFIBUS-DP) e aos aspectos relevantes ao desenvolvimento do simulador (LLRS) de comportamento temporal das camadas baixas da arquitectura RFieldbus. No capítulo 3 é feita uma descrição geral da tecnologia RFieldbus. São apresentadas as duas topologias de rede (*Direct* e *Relay Link Network*), a solução de mobilidade, alguns exemplos de topologias de rede estruturadas. É também apresentada a arquitectura da rede de comunicação RFieldbus, com especial destaque nas camadas física e de ligação de dados. No capítulo 4 são apresentados os conceitos mais relevantes da teoria da simulação por eventos discretos e o ambiente de simulação utilizado (OMNeT++) na implementação do simulador das camadas baixas da rede de comunicação RFieldbus (LLRS).

O capítulo 5 descreve o modelo de simulação desenvolvido para as camadas baixas do RFieldbus e o capítulo 6 apresenta a implementação do modelo em OMNeT++; a estrutura dos módulos implementados no LLRS, a funcionalidade de cada módulo e a mensagem definida.

No capítulo 7 é apresentado o teste e validação do simulador através da comparação de observações do comportamento temporal da aplicação de teste *Manufacturing Automation Fieldtrial* (MAF) com os resultados obtidos por simulação. De referir que ambos foram obtidos em condições idênticas (mesma topologia, mesmo tráfego e mesmos parâmetros). Estes resultados são comparados, de forma a analisar a validade dos resultados de simulação.

Finalmente, no capítulo 8, é feito um sumário onde são apresentadas as conclusões do trabalho realizado, e apontadas algumas direcção de trabalho futuro.

Em anexo é apresentada uma descrição: da implementação piloto MAF (Anexo A), das classes de simulação implementadas (Anexo B) e da mensagem OMNeT++ que representa o PDU (*Protocol Data Unit*) RFieldbus (Anexo C).

Na escrita desta dissertação tomou-se especial cuidado com a tradução para Português da maioria dos termos técnicos utilizados. A principal excepção acontece nas palavras vulgarmente adoptadas em Português (e.g. software).

No entanto, esta dissertação foca com especial ênfase várias tecnologias com termos técnicos específicos. Desta forma considera-se relevante a não tradução para Português dos termos em inglês que denotem uma funcionalidade ou característica das tecnologias utilizadas. Para facilidade de leitura esses termos são apresentados em itálico.

Capítulo 2

PROFIBUS

2.1 Introdução

PROFIBUS [5][6][7][17] é nome pelo qual é conhecido uma rede de campo (*fieldbus*) destinada à interligação de dispositivos digitais de baixa ou média performance ao nível da célula (*cell level*) e de campo (*Field level*). Exemplos de*stes dispositivos* são: sensores, actuadores, controladores lógicos programáveis, controladores numéricos, consolas de programação, dispositivos de interface homem maquina, etc. O conjunto de modelos, serviços e características que constituem a arquitectura desta rede de comunicação, estão definidos no Volume 2 da norma europeia EN50170 [6].

O objectivo deste capítulo é fazer uma descrição geral da tecnologia PROFIBUS, dando especial atenção à parte da tecnologia PROFIBUS utilizada no RFieldbus e aos aspectos mais relevantes da arquitectura utilizados no desenvolvimento do simulador de comportamento temporal das camadas baixas da arquitectura RFieldbus (LLRS).

Em PROFIBUS estão definidos dois perfis de comunicação, PROFIBUS-FMS e o PROFIBUS-DP. O RFieldbus deriva do PROFIBUS-DP.

Inicialmente é feita uma descrição da tecnologia de transmissão de dados ao nível do meio físico e do próprio meio físico[1]. Esta tecnologia é baseada na norma EIA RS-485 [18].

Dado que o estudo efectuado e o simulador LLRS foram desenvolvidos para as camadas 1 e 2 do RFieldbus, que são idênticas as camadas 1 e 2 do PROFIBUS-DP, foi dado um grande detalhe à descrição destas camadas.

É apresentada a arquitectura da rede de comunicação PROFIBUS, com as diferentes camadas constituintes. Para cada camada é descrita a funcionalidade implementada, a forma como essa funcionalidade é disponibilizada ao utilizador do serviço, a *interface* e a forma como essa funcionalidade foi implementada, o protocolo.

Na descrição da camada de ligação de dados (FDL), são apresentados os serviços disponibilizados pela camada, para o envio de dados com confirmação (SDA), envio de dados sem confirmação (SDN) *Unicast*, *Broadcast* e *Multicast*, envio e requisição de dados, ou apenas a requisição de dados (SRD) e o serviço que permite o estabelecimento de um envio e requisição de dados de forma cíclica para um conjunto de estações (CSRD) [19].

---

[1] Versão 1 do meio físico PROFIBUS

Do protocolo da FDL são descritos os formatos das tramas[2] utilizadas, os esquemas de endereçamento, o ciclo de mensagem e parâmetros associados (*slot time* e *idle time*), os algoritmo para gerir o *Token* (distintivo cuja posse garante o acesso ao meio físico), o mecanismo de gestão das entrada e saída dinâmica de estações na rede (GAP *Maintenance*), os tipos de tráfego, o algoritmo de escalonamento de tráfego e a máquina de estados do controlador da FDL [11].

Finalmente é apresentada a arquitectura do perfil de comunicação PROFIBUS que é a base do RFieldbus, o PROFIBUS–DP. São apresentadas as estações definidas, o modelo de comunicação, uma descrição das funções disponibilizadas ao utilizador de cada estação (equivalente ao processo de aplicação) e duas configurações de rede possíveis em PROFIBUS-DP que resumem a aplicação tipo desta rede de comunicação. De referir que existem outros tipos de configuração de rede incluindo estações FMS, mas que não se enquadram no âmbito do estudo apresentado nesta dissertação.

2.2 Arquitectura da rede de comunicação PROFIBUS

A estrutura de camadas é baseada no modelo OSI da ISO para a comunicação aberta entre sistemas. Para minimizar custos e aumentar eficiência da rede as camadas 3 até 6 estão vazias, sendo utilizadas as camadas física, de ligação de dados e de aplicação. A figura seguinte esquematiza a arquitectura do protocolo de comunicação PROFIBUS.

[2] Ao longo desta dissertação os termos trama e PDU serão utilizados para referir as tramas definidas na PROFIBUS FDL.

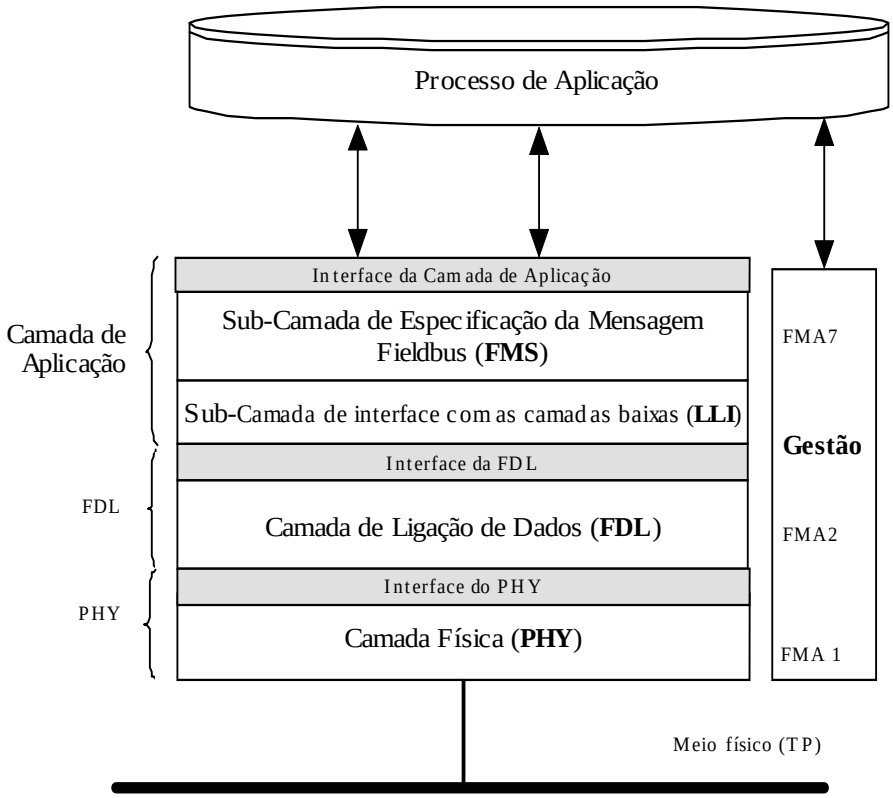


Figura 1 - Arquitectura do protocolo de comunicação PROFIBUS

A camada de aplicação está dividida em duas sub camadas: *Fieldbus Message Specification* (FMS) e a *Lower Layer Interface* (LLI). A FMS descreve os serviços e o modelo associado, do ponto de vista do parceiro de comunicação e fornece um conjunto de serviços de comunicação *Master*/*Master* e *Master*/*Slave*. A LLI permite um acesso facilitado à interface da camada FDL, implementando as seguintes tarefas: a correspondência dos serviços FMS nos serviços fornecidos pela FDL, gestão de erros, gestão das conexões (estabelecimento e cancelamento), supervisão da conexão e controlo de fluxo.

A interface da camada de aplicação (*Application Level Interface*) fornece as funções de comunicação ao processo de aplicação.

Em PROFIBUS estão definidos dois perfis de comunicação. PROFIBUS-*FMS* e PROFIBUS-*DP* e vários perfis de aplicação (Tabela 1). Cada perfil permite a implementação de um sistema *mono-Master* (apenas um *Master* no sistema de comunicação) ou *multi-Master* (vários *Masters* no sistema), comum máximo de 32 estações (*Masters/Slaves*) por segmento e até 126 estações num meio físico constituído por diversos segmentos interligados por repetidores.

O PROFIBUS-*FMS* é uma solução para as tarefas de comunicação ao nível da célula (*cell level*) ou campo (*Field level*) [20]. Oferece um espectro alargado de funcionalidades e flexibilidade. Para assegurar a transmissão rápida e eficiente de dados, a arquitectura PROFIBUS tem especificado um outro perfil de comunicação chamado PROFIBUS-*DP*. Este foi tornado eficiente através da simplificação da arquitectura, retirando a camada de aplicação (camada 7) [10]. Nesta arquitectura são utilizadas apenas as camadas física (camada 1), de ligação de dados (camada 2) e a camada

DDLM (*Direct Data Link Mapper*) que permite ao utilizador um acesso mais confortável às funções de transferência de dados da FDL.

O PROFIBUS-*DP* é uma versão optimizada do protocolo PROFIBUS. Destina-se a comunicações de tempo crítico entre sistemas de automação e periféricos distribuídos. O perfil de comunicação PROFIBUS-*DP* também define funções para diagnóstico. Desta forma, o PROFIBUS-DP define uma nova classe de *Masters*, chamados DPM2 (DP *Master Class* 2), cujas funções são monitorização, diagnóstico e a configuração.

Os perfis PROFIBUS são compromissos sobre o uso de serviços não obrigatórios e parâmetros de meio físico para áreas de aplicação específicas. Os perfis PROFIBUS por área de aplicação são resumidos na tabela seguinte:

Application Profile	*Communication Profile*	
	FMS	*DP*
Communication between PLCs	✓	
Building Automation	✓	
Low voltage switch device	✓	
Process Automation Field device (PA)		✓
Drives Technology		✓
NC / RC control system		✓
Encoder Devices		✓
Sensor/Actuator networks	✓	

Tabela 1- Perfis de comunicação e de aplicação PROFIBUS

A norma PROFIBUS define ainda um outro perfil (*ProfiSafe*) que cobre os requisitos de segurança intrínseca de acordo coma norma IEC 1158-2. Nele é definida a forma como os dispositivos tolerantes a falhas (ex. botões de emergência, botões de pressão, etc) são ligados ao controlador lógico programável PLC via PROFIBUS.

De seguida é feita uma descrição detalhada dos protocolos e interfaces das camadas baixas (camada 1 e 2) do Protocolo de comunicação PROFIBUS.

2.3 Camada Física (PHY)

Na especificação da camada física são descritas as características do meio físico e dos sinais (camada 0 e 1 OSI): o suporte físico à comunicação, com os seus comprimentos e topologia, a interface de linha, o número máximo de estações, a velocidade de transmissão. A especificação desta camada deve fornecer os dados suficientes para que exista uma caracterização total do meio físico no qual se fará a transmissão dos dados, assim como com todos os interfaces funcionais, eléctricos entre as estações e esse meio físico, os interfaces mecânicos, nomeadamente os conectores utilizados.

A versão 1 da especificação da camada física do PROFIBUS está de acordo com a norma EIA RS-485. A versão 2 (de acordo com a norma IEC 1158-2) cobre os requisitos de segurança intrínseca (IS) [21]. Esta versão 2 não será utilizada nesta dissertação. Desta forma a partir deste ponto por camada física[3] está implícita a referência a versão 1 da especificação da camada física do PROFIBUS.

O meio físico (EIA RS-485) consiste num par entrançado com protecção electromagnética, que permite formar redes com topologias de barramento linear com comprimento de linha máximo de 1200 metros e um número de estações que pode ir até às 32 num único segmento, 127 em quatro segmentos interligados com repetidores, com taxas de transferência de dados que podem atingir os 1500kbps.

A protecção electromagnética melhora a compatibilidade electromagnética. Em ambientes com baixos níveis de interferência electromagnética (EMI) pode ser usado par entrançado sem protecção electromagnética (UTP).

2.3.1 Interface da camada PHY

O serviço de dados fornecido pela camada física PHY (fornecedora do serviço) à camada de ligação de dados FDL (utilizadora do serviço) é constituído por duas primitivas, que permitem a transmissão de símbolos que constituem as tramas da FDL [22]. A primitiva *request* permite à camada FDL requerer a transmissão de um símbolo à camada PHY. A primitiva *indication*, invocada na máquina remota, permite à camada PHY comunicar a recepção do símbolo, à camada FDL.

As duas primitivas e respectivos parâmetros que de uma forma abstracta definem o serviço a ser implementado são:

1. PHY_DATA.*request* (FDL_symbol)
2. PHY_DATA.*indication* (FDL_symbol)

A relação temporal destas primitivas é esquematizada na figura seguinte

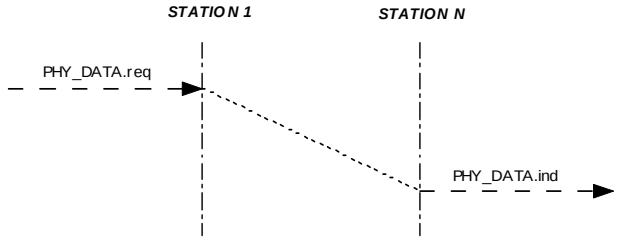


Figura 2 - Relação temporal das primitivas do PHY

A gestão do tempo para a requisição da transmissão do símbolo está a cargo da camada superior FDL.

[3] Meio físico com fios; em RFieldbus existe outro tipo de meio físico, sem fios.

2.3.2 Especificação do protocolo da camada PHY

A definição do serviço da camada física PHY suporta a recepção e transmissão de bits (símbolos da FDL) que são elementos de um carácter UART. Cada símbolo da FDL tem a duração de um *bit time* (t_{BIT}) [22].

A codificação de cada bit da transmissão de dados série e assíncrona é feita no sistema NRZ (não retorna a zero) no qual o estado binário "1" é representado por uma tensão diferencial constante positiva e o estado binário "0" é representado por uma tensão diferencial constante negativa, de acordo com a norma EIA RS485.

2.3.3 Descrição do RS-485

A versão 1 da especificação da camada física do PROFIBUS está de acordo com a norma EIA RS-485 [18]. O RS-485 é um protocolo de camada física de comunicação série assíncrona multiponto, que permite a comunicação em *Half* ou *Full Dupplex* [23]. A norma EN50170 v2 (PROFIBUS) usa a versão *Half Duplex*, com meio físico de um cabo de par entrançado (TP)[4].

A norma RS-485 especifica apenas as características eléctricas e físicas (do meio físico), dos transmissores e receptores e não os protocolos de comunicação de dados.

A transmissão de dados de forma balanceada (tensões diferenciais), confere ao RS-485 grande imunidade a ruído e às interferências (*crosstalk*), dado que: se por um lado a interferência produzir uma tensão de modo comum esta vai ser eliminada na entrada do receptor que é sensível a tensões diferencias e por outro como o meio físico é de par entrançado as correntes induzidas por interferência electromagnética tendem a anular-se devido ao facto dos condutores se encontrarem torcidos entre si.

Com o aumentar das taxas de transmissão de dados (10Mbps,100kbps) as distâncias de comunicação diminuem (10,1200) m respectivamente e aumenta a necessidade de uma correcta terminação do barramento para evitar as reflexões de sinal.[5]

A norma EIA RS-485 define uma impedância de entrada dos circuitos de 12 KΩ que permite a ligação de até 32 transmissores num único segmento. Dado que as comunicações são bidireccionais, existe a necessidade de colocar terminadores para evitar as reflexões (R_t = 120Ω).

---

[4] Existe também a possibilidade de utilizar meio físico de fibra óptica em PROFIBUS, embora esta opção seja menos utilizada.

[5] Em PROFIBUS-DP a máxima taxa de transferencia de dados definida na meio físico é de 1,5Mbps. No entanto actualmente a norma EIA RS-485 já garante funcionamento a 10Mbps. Já existem no mercado *chips* oferecendo taxas de transferencia de dados de 25Mbps (ex SN76ALS176). No âmbito da dissertação apresentada serão utilizadas as taxas de transferência do PROFIBUS-DP/RFieldbus.

2.4 Camada de Ligação de Dados FDL

2.4.1 Formato das tramas

Carácter da UART

Cada trama PROFIBUS é constituída por um ou mais caracteres UART (ISO 1177, ISO 2022). O carácter UART (UC) é um carácter *Start*/*Stop* para a transmissão série assíncrona de dados e tem a seguinte estrutura:

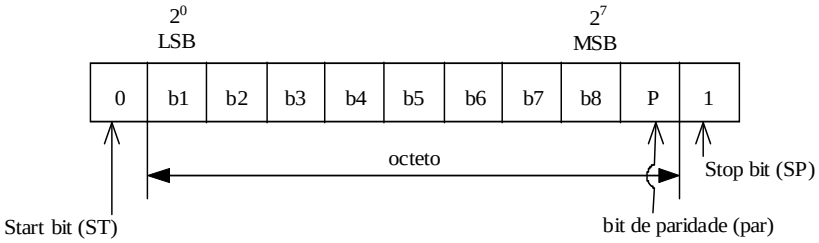


Figura 3 - Carácter UART

O carácter é constituído por um *Start* bit (estado lógico 0) oito *bits* de dados (estado lógico 0 ou 1), um bit de paridade (paridade par) e um stop bit (estado lógico 1). Por cada byte de dados entregue pela camada 2 - FDL são transmitidos 11 bits no meio físico com fios [11].

Formato das tramas

Neste tópico vão ser discutidos os formatos das tramas definidas pela camada FDL (camada 2) [11]. As tramas apresentadas de seguida são utilizadas nos pedidos (*request*) e respostas (*response*), que constituem cada ciclo de mensagem. De referir que as tramas com os *Start Delimiter* SD1, SD2 e SD3 podem ser utilizadas para efectuar um pedido (*action frame*) ou para transmitir uma resposta. A diferença é que, como *action frame*, deve respeitar um tempo mínimo de 33 *bt* (*bit time*) antes da transmissão (SYN). Cada carácter (octeto) destas tramas vai ser inserido no campo de dados do carácter da UART.

Frames of fixed Length with no Data Field

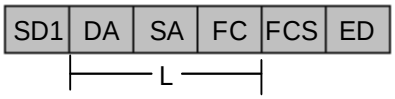


Figura 4 - *Frames of fixed Length with no Data Field*

Figura 5 - *Short Acknowledgement Frame*

- SYN é o mínimo intervalo de tempo durante o qual cada estação deve receber *idle state* (estado binário 1) do meio de transmissão antes de poder aceitar o início de uma *action frame*[6] (*request* , *send*/*request* ou *Token frame*). O seu valor é de 33 *bt*.
- SD1 - *Start Delimiter* 1, octeto com o valor 10_H que permite identificar esta trama como uma trama de tamanho fixo sem campo de dados.
- DA - *Destination address* , octeto com o endereço da estação a que se destina a trama.
- SA - Source *address*, octeto com o endereço da estação que emite a trama.
- FC – este é o oculto de controlo e indica o tipo de trama: se é uma *action frame* (*request* ou *send/request*), uma *response* ou um ACK. Adicionalmente contém também a função e a informação de controlo que previnem a perda e multiplicação de mensagens, ou o tipo de estação e o estado do seu controlador da FDL.
- FCS – *Frame Check Sequence*, este octeto é requerido para garantir uma *Hamming distance*[7] de 4. O valor deste octeto é calculado somando o valor dos campos DA, SA e FC, no caso das tramas de comprimento fixo sem dados e nas restantes a partir dos mesmos campos e DATA_UNIT.
- ED – *End Delimiter*, octeto com o valor 16_H que delimita explicitamente o fim da trama.
- Cada um destes campos ocupa um octeto.
- L – Comprimento do campo de informação, no caso desta trama L = 3 = 1 (DA) + 1 (SA) + 1 (FC).
- SC – é a *Short Acknowledgement Frame,* constituída por um único carácter com o valor $E5_H$.

Nas tramas seguintes apenas são descritos os campos diferentes dos aqui apresentados.

Frames of fixed Length with Data Field

[6] *Action frame* é a primeira trama de uma transação. Pode ser um *request* , um *send*/*request* ou *Token frame.*

[7] *Hamming distance* é o número mínimo de bits, que tem que se alterar numa trama durante a transmissão para que o erro não seja detectável à chegada. No PROFIBUS é 4, isto significa que a alteração acidental de até 3 bits é detectada no receptor.

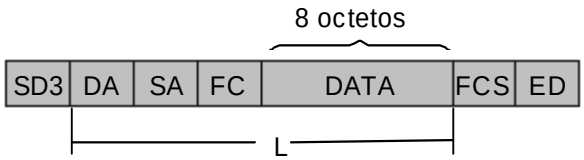


Figura 6 - *Frames of fixed Length with Data Field*

- SD3 - *Start Delimiter* 3, octeto com o valor $A2_H$ que permite identificar esta trama como uma trama de tamanho fixo com campo de dados.
- DATA_UNIT – Campo de dados, nesta trama de comprimento fixo 8 octetos.
- L – Comprimento do campo de dados, nesta trama L = 11 octetos = 8 (DATA_UNIT) + 1 (DA) + 1 (SA) + 1 (FC).

Frames with variable Data *Field* Length

Figura 7 - *Frames* with variable Data *Field* Length

- SD2 - *Start Delimiter* 2, octeto com o valor 68_H que permite identificar a trama com campo de dados de comprimento variável.
- LE (Length Octet) e LER (Length Octet Repeated) – são idênticos, contém o número de octetos de informação (DA, SA, FC e DATA_UNIT) incluídos nesta trama (com campo de dados de comprimento variável). Assim os valores que podem assumir são [4,249], dado que DATA_UNIT tem que ter pelo menos um octeto e com L = 249 o PDU tem 255 octetos de comprimento.
- DATA_UNIT – Campo de dados. Comprimento variável [1,246] octetos
- L – Comprimento do campo de dados. L [4,249]

Token Frame

Figura 8 - *Token Frame*

- SD4 - *Start Delimiter* 4, octeto com o valor DC_H que permite identificar a trama do *Token*.

2.4.2 Interface da camada FDL

Ao utilizador da camada 2 (FDL) são oferecidos os seguintes serviços [19]:

- *Send Data with Acknowledge* (SDA)
- *Send Data with No Acknowledge* (SDN)
- *Send and Request Data with Reply* (SRD)
- *Cyclic Send and Request Data with Reply* (CSRD)

Estes serviços são disponibilizados pela camada FDL à camada superior, que é o utilizador da FDL. O utilizador local é a camada superior à FDL na estação onde é requisitado o serviço. O utilizador remoto é a camada superior à FDL na estação remota, com a qual esta a ser estabelecida a comunicação.

Send Data with Acknowledge (SDA)

Este serviço permite ao Utilizador local (num *Master*), requisitar o envio de dados para um Utilizador remoto [19].

O ciclo de mensagem é implementado com as seguintes tramas:

O *request* (no qual são enviados os dados) é enviado numa *Frame of fixed Length with Data Field* ou numa *Frame with variable Data Field Length*. Para a confirmação da recepção dos dados pelo controlador da FDL remoto é utilizada a *Frame of fixed Length with no Data Field* ou a *Short Acknowledgement Frame*.

Send Data with No Acknowledge (SDN)

Este serviço permite ao Utilizador *local* (num *Master*), requisitar o envio de dados para outra estação remota, para várias estações remotas (*Multicast*) ou para todas as estações remotas (*Broadcast*), simultaneamente. Em cada caso envia os dados para o(s) utilizador(es) remoto(s) [19].

Para enviar os dados é utilizada a *Frame of fixed Length with Data Field* ou a *Frame with variable Data Field Length*.

Send and Request Data with Reply (SRD)

Este serviço permite ao Utilizador *local* (num *Master*), requisitar à FDL o envio de dados para um Utilizador remoto e simultaneamente requerer dados que tenham sido previamente disponibilizados por ele. Permite também apenas requisitar o envio de dados pela estação remota [19].

Para enviar os dados o controlador local da FDL utiliza uma *Frame of fixed Length with Data Field* ou uma *Frame with variable Data Field Length*. Se não enviar dados utiliza uma *Frame of fixed Length with no Data Field*.

Os dados são recebidos numa *Frame with variable Data Field Length*. Em caso de erro no controlador da FDL remoto recebe apenas um ACK negativo numa *Frame of fixed Length with no*

Data Field, ou uma mensagem de erro (*Response FDL Data High/Low no Resource for Send Data*) numa *Frame with variable Data Field Length* ou numa *Frame of fixed length with Data Field*.

Cyclic Send and Request Data with Reply (CSRD)

Este serviço permite estabelecer de uma forma cíclica o envio de dados para uma estação remota e simultaneamente requerer dados dessa estação remota. Existe também a possibilidade de apenas requerer ciclicamente dados dessa estação remota. As estações remotas a serem endereçadas e a sequência de endereçamento estão contidas na *Poll list* [19].

As tramas envolvidas na transferência de dados deste serviço são as mesmas que no serviço SRD [6].

2.4.3 Protocolo da camada FDL

A camada FDL é muito importante em PROFIBUS. A implementação dos serviços disponibilizados pela sua interface implica algumas das funcionalidades mais determinantes do comportamento desta rede de comunicações. As funcionalidades implementadas na FDL são:

- O controlo do acesso ao meio físico partilhado [11]. Em PROFIBUS existem estações *Master* (que são activas) e estações *Slave* (que são passivas). As estações *Master* fazem circular entre si uma trama especial que representa um *Token* e cuja posse garante ao detentor a possibilidade de aceder ao meio físico. Este algoritmo que rege o acesso ao meio físico é uma versão modificada do algoritmo *Timed Token* [24], conhecido pela sua eficácia em sistemas de tempo real. Dada a importância que esta trama tem para o funcionamento da rede de comunicações estão definidos procedimentos específicos para a transmissão e recepção da mesma.
- As alterações dinâmicas à topologia da rede de comunicação devido à entrada e saída de estações na rede de comunicação (GAP Maintenance), implicam a geração da informação (LAS e GAPL) necessária para requisitar o inicio das comunicações nas estações que entram no anel lógico e procedimentos para a alteração do anel lógico nas outras estações [19].
- O ciclo de mensagem é a unidade básica da comunicação. Existem ciclos de mensagem com e sem resposta e em ambos são utilizados temporizadores de controlo para garantir a correcta transmissão da mensagem (*idle* e *slot time*) [19].
- As tramas definidas na FDL as quais constituem os *request* e *response* (quando há lugar) dos ciclos de mensagem [19].
- O algoritmo de escalonamento de mensagens em PROFIBUS, que determina a ordem pela qual as mensagens de alta e baixa prioridade serão transmitidas [19].
- E finalmente o esquema de endereçamento utilizado em PROFIBUS [19].

Ao longo dos sub-capítulos seguintes as funcionalidades atrás referidas são descritas pormenorizadamente.

2.4.4 Endereços

Os endereços vão contidos num octeto (2^8 = 256 endereços). No entanto como o bit 8 do octeto serve para indicar uma extensão de endereço, existem 128 endereços.

O endereço 127 (b1 ao b7 = 1) é o endereço reservado para *Broadcast* e *Multicast* (do ponto de vista da FDL). Este endereço é utilizado no serviço de transferência de dados SDN. Ficam disponíveis 127 endereços [0,126] para atribuir aos *Masters* e *Slaves*

A distinção entre *Broadcast* e *Multicast* faz-se através do ponto de acesso ao serviço utilizado (*service access point*[8]). Nas tramas com DATA_UNIT o bit EXT = 1 (Figura 9) no SA e/ou DA indica que uma extensão de endereço DAE, SAE respectivamente segue no DATA_UNIT logo após o octeto FC. Estas extensões permitem transportar os pontos de acesso ao serviço [11].

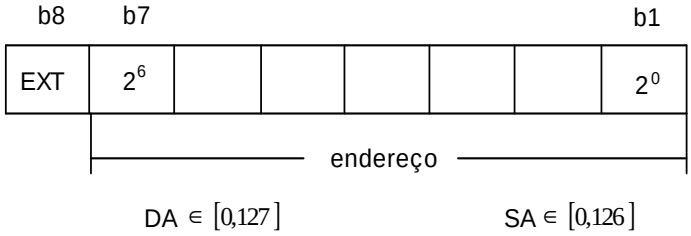


Figura 9 - Codificação do octeto de endereços

2.4.5 Ciclo de mensagem

Os *Masters* são as estações activas em PROFIBUS, podendo iniciar ciclos de mensagem. Os *Slaves* são estações passivas que não podem transmitir por iniciativa própria mas apenas quando são requisitadas por um *Master* (quando recebem uma *request frame*).

Um *Master* inicia um ciclo de mensagem, enviando uma *Action Frame*. Nesta trama são incluídos os endereços do *Master* como emissor da mensagem e do *Slave* como destinatário da mensagem. Este último ao detectar uma mensagem endereçada a si processa-a e envia a correspondente resposta (nos ciclos de mensagem com resposta). Neste caso o *Master* é o *initiator* e o *Slave* é o *responder* [11]. Existem casos em que o *Master* pode ser responder (redes *multi-Master*).

Um ciclo de mensagem (ou transacção) é constituído por uma trama de pedido (*request*) por parte do *Initiator* (que é sempre um *Master*) e o correspondente trama de *acknowledgement* ou resposta enviada pela estação que foi endereçada no *request*.

Existem também ciclos de mensagem sem *acknowledgement*. Neste caso é enviado o *request* e não há lugar a recepção de qualquer *response* [11].

[8] - equivalente a um porto.

Dado que o PROFIBUS funciona num modo de *Broadcast* (todas as estações vêem tudo o que circula na rede) todas as estações estão constantemente a monitorar todos os *request* e respondem apenas quando endereçadas.

Na figura seguinte encontra-se um fluxograma descritivo de um ciclo de mensagem com resposta.

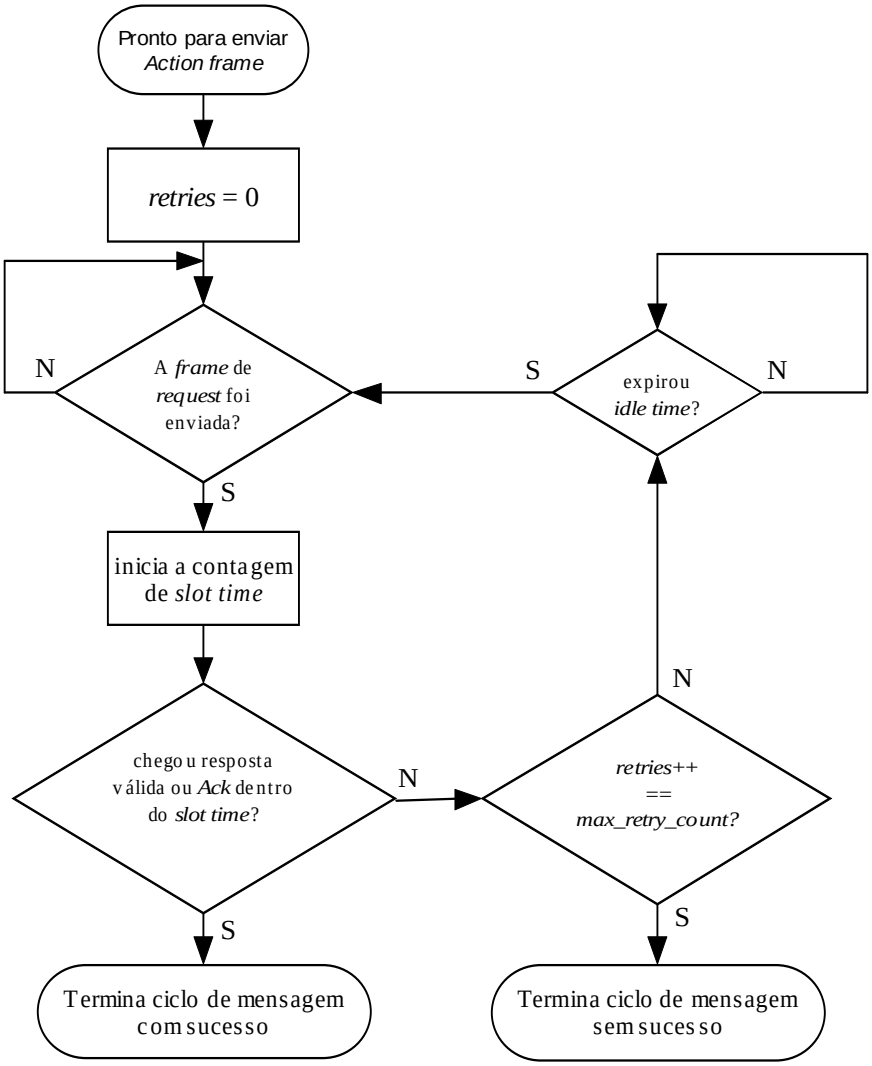


Figura 10 - Ciclo de mensagem com resposta

Uma vez iniciado um ciclo de mensagem é sempre terminado (com ou sem sucesso), mesmo que o tempo que o *Master* tem disponível para comunicação (T_{TH} – Token Holding Time) expire durante a execução do ciclo. O controlador da FDL só verifica o valor de T_{TH} no início do próximo ciclo de mensagem. Uma estação ao enviar o *request* aguarda pela *response* e caso esta não chegue durante o período de tempo disponível (*slot time*) repete o pedido o número de vezes necessárias (até ao limite máximo definido: *max_retry_count*). Esta situação origina muitas vezes um fenómeno

chamado *late Token*, dado que a duração do ciclo de mensagem é superior ao T_{TH} disponível no momento da decisão de processar mais este ciclo de mensagem (T_{TH} pode ser muito pequeno, mas basta que seja maior que zero).

No desenrolar do ciclo de mensagem são utilizados dois *timers* com funções distintas mas igualmente importantes:

2.4.6 Slot time

É o tempo que o *initiator* espera pela chegada da *response* [11]. Quando termina o envio do *request* (instante em que termina o envio do ultimo bit da *request frame*) aguarda durante *slot time* pela chegada da *response*. Se ela não chegar durante este intervalo de tempo reenvia o *request*.

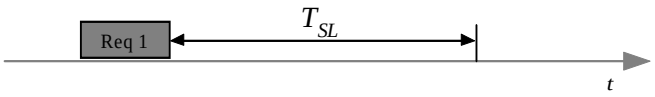


Figura 11 - *Slot time*

2.4.7 Idle time

É um tempo de sincronização entre tramas. Dado que o PROFIBUS assenta numa transmissão de dados assíncrona existe a necessidade de monitorar o *idle state* durante um período de tempo mínimo de forma a permitir uma sincronização correcta dos receptores de linha.

Existem dois casos distintos. As transacções com e sem resposta.

Idle Time nas transacções com resposta/*acknowledge*

No caso das transacções com resposta existe um tempo mínimo (T_{ID1}) que o *Master* aguarda antes de iniciar nova transacção ou de passar o *Token* [11], contado a partir do instante em que finaliza a recepção da *response*, como se pode ver na Figura 12.

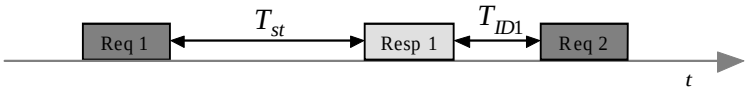


Figura 12 - T_{ID1}, idle time nos ciclos de mensagem com resposta/Ack

Idle Time* nas transacções sem resposta/*acknowledge

No caso das transacções sem resposta existe um tempo mínimo (T_{ID2}) que o *Master* aguarda antes de iniciar nova transacção ou de passar o *Token* [11]. Contado a partir do instante em que termina a transmissão do *request* (Figura 13) .

Figura 13 - T_{ID2}, idle time nos ciclos de mensagem sem resposta/Ack

2.4.8 Recepção do *Token*

Num sistema PROFIBUS com mais do que um *Master* (multi-*Master*) o controlo de acesso ao meio físico é híbrido[9].

Existe uma trama especial (*Token*) cuja posse garante à estação activa (*Master*) a possibilidade de usar o meio físico. Esse *Token* é utilizado por cada *Master* para transmitir os seus dados e posteriormente passado entre *Masters* numa cadeia que forma um anel lógico. Após transmitir os seus dados o *Master* transmite o *Token* ao *Master* seguinte (nessa cadeia lógica) que ao recebe-lo garante o acesso ao meio repetindo-se o processo no *Master* seguinte. Essa cadeia lógica é ordenada por ordem crescente de endereço do *Master*. Ou seja o *Token* começa por ser passado da estação de mais baixo endereço sequencialmente de estação em estação até à estação activa com o endereço mais alto na rede. Este *Master* vai fechar esse anel lógico passando o *Token* para o *Master* com o endereço mais baixo [11].

Para criar este anel lógico cada *Master* conhece o seu endereço (*This Station* - TS) o endereço da estação que lhe passa o *Token* (*Previous Station* - PS) e o endereço da estação para a qual deve passar o *Token* (*Next Station* - NS) [11]. Quando um *Master* recebe um *Token* endereçado a si (DA=TS), passa a ser o detentor do *Token* e pode iniciar a transmissão de ciclos de mensagens. Se ao contrário recebe o *Token* de uma estação que não é a que lhe devia enviar o *Token* (PS) assumirá que ocorreu um erro e descarta o *Token* [25]. No entanto se de seguida voltar a receber um *Token* endereçado a si enviado pela mesma estação do ciclo anterior assume que o anel lógico mudou, recebendo o *Token* e alterando o endereço contido em PS por esta nova estação que agora lhe passa o *Token* [25]. A Figura 14 apresenta o fluxograma demonstrativo deste procedimento de recepção do *Token*.

Durante o período em que *Master* detém o *Token* procede à comunicação com os seus *Slaves* por *polling*. Enquanto tiver comunicações a realizar e tempo disponível de *Token* continua a enviar e requerer dados aos seus *Slaves*. Quando uma das condições não se verifica, passa o *Token* para a estação seguinte.

---

[9] Num sistema mono *Master* o acesso também é híbrido dado que o *Master* faz um *polling* aos *slaves* e passa o *token* para si próprio. Neste caso detém o *token* quase 100% do tempo.

O tempo que cada *Master* tem disponível para processar ciclos de mensagem depende do parâmetro t_{TR}. Devido ao algoritmo de despacho de mensagens (página 24), em PROFIBUS pode ocorrer um *Late Token*, que consiste na recepção do *token* atrasado. Deve-se ao facto de a estação activa anterior poder sempre enviar pelo menos um ciclo de mensagem de alta prioridade ou poder terminar o ciclo de mensagem que inicia (com ou sem sucesso) antes de T_{TH} expirar [26].

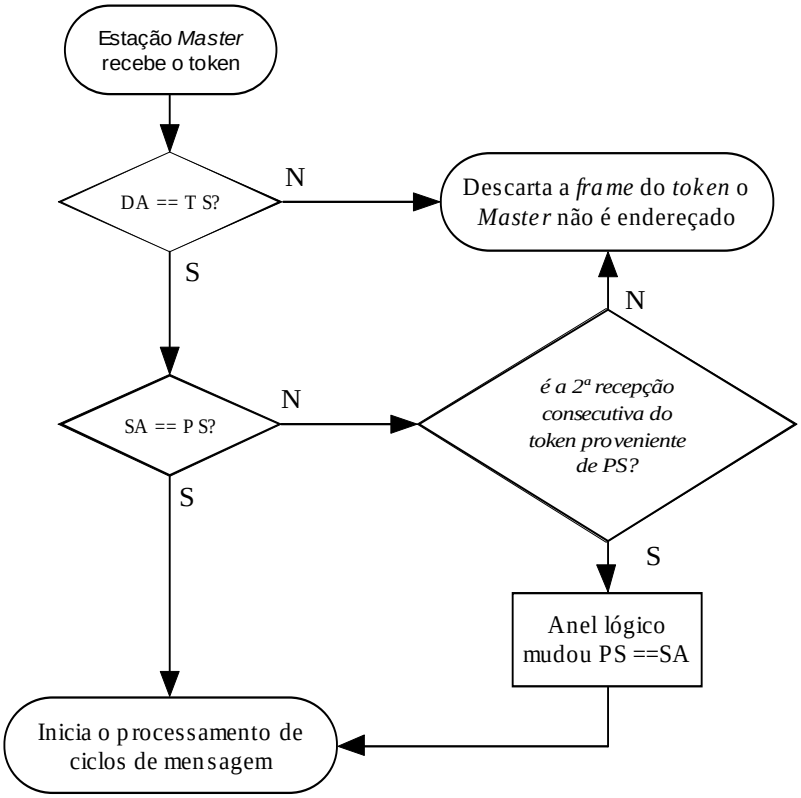


Figura 14 - Recepção do *Token*

2.4.9 Transmissão do *Token*

Por não necessitar de transmitir mais dados ou por não dispor de mais tempo, em determinado instante o *Master* dá início à transmissão do *Token* (figura 15). Dada a importância do *Token* PDU a confirmação da sua correcta transmissão é muito importante. A perda de um *Token* implica a paragem de toda a rede; sem *Token* nenhum *Master* pode transmitir [11].

Após transmitir o *Token* o *Master* aguarda a passagem de *Syn Time* (meio físico em estado *idle* 'nível lógico 1' por um período de 33 *bit time*) e após esse período de tempo começa a procurar actividade no meio físico. Se desde esse instante até aquele em que expira *Slot time* detectar actividade no meio físico (outra estação transmite) considera que o *Token* foi transmitido com sucesso. Caso *Slot time* expire sem o *Master* ter detectado actividade no meio físico, reenvia o *Token* e espera outro *Slot Time*. Se detecta actividade no meio físico dentro deste *Slot time* assume uma correcta transmissão do *Token* e deixa de se comportar como *Master* activo na rede. De outra forma

reenvia o *Token* pela última vez. Se desta vez não detectar actividade no meio físico durante o *Slot time* o *Master* repete o procedimento descrito para o próximo sucessor. Se necessário continua à procura de um sucessor a partir da sua lista de estações activas.

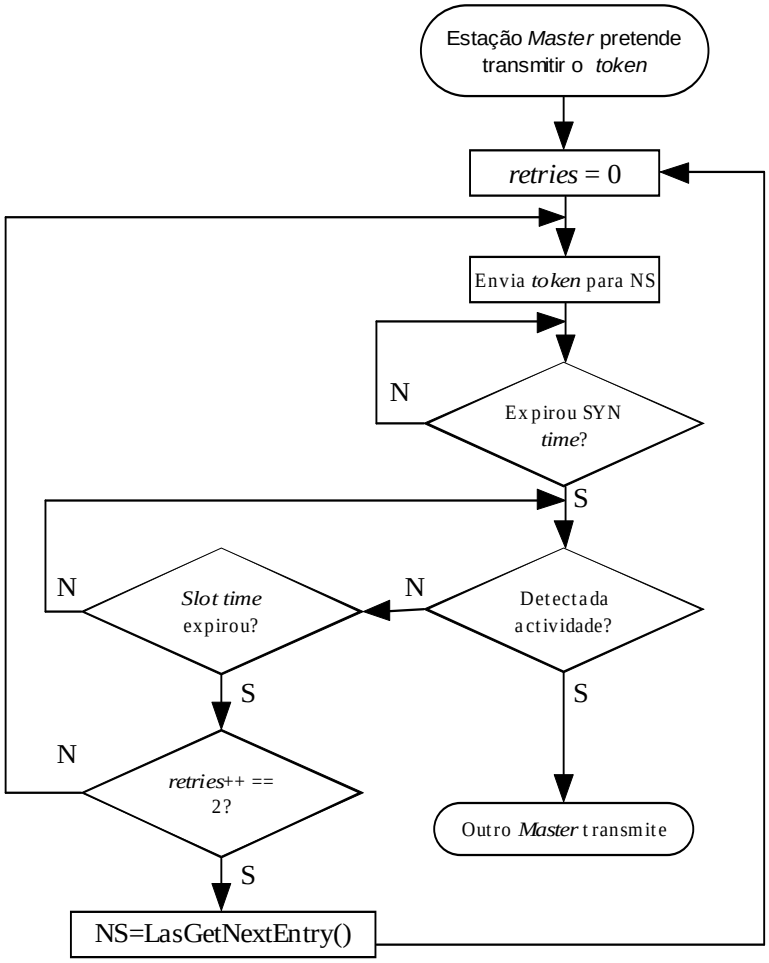


Figura 15 - Transmissão do *Token*

2.4.10 *GAP Maintenance*

No PROFIBUS as estações (*Master* ou *Slave*) podem ser ligadas ou desligadas do meio de transmissão em qualquer instante. O mecanismo para gerir estas alterações dinâmicas da topologia da rede é o *GAP Maintenance* [25]. *GAP* é uma expressão inglesa que significa um espaço entre coisas. **N**este caso o *GAP* é o espaço de endereçamento entre dois *Masters*. Consequentemente a manutenção do *GAP* é a responsabilidade de cada *Master* de verificar a presença ou ausência de estações nesse intervalo de endereçamento. A *GAP List* (*GAPL*) [6]é a lista que contem o estado de todas as estações pertencentes ao *GAP* [11]. O *GAP Update Time* (T_{GUD}) é a periodicidade a partir da qual cada *Master* examina o seu *GAP*. O fim deste período indica a necessidade de proceder a

manutenção do *GAP*. Uma verificação de todas as estações da *GAP List* pode durar várias rotações do *Token*, dado que o tráfego gerado pelo *GAP Maintenance* é tráfego de baixa prioridade [11].

2.4.11 Algoritmo de despacho de mensagens

Tipos de tráfego

Em PROFIBUS existem mensagem de alta (*Hi*) e de baixa prioridade (*Low*). Estas últimas subdividem-se em três subcategorias [25].

Hi	*Hi priority message cycles*
Low	*Poll list* *Low priority message cycles* *GAP Update message cycles*

Tabela 2 – Classes de tráfego da FDL em PROFIBUS

Quando requisitado pela camada superior à FDL é passada a esta última uma *Poll List*. Esta contém uma lista de *Masters* e *Slaves* que deverão ser endereçados por este *Master* durante o período de *Polling* designado por *Cyclic Send/Request Mode*. O tráfego *Poll list* é aquele que é gerado para levar a cabo a execução da *Poll List*.

Os *Low priority message cycles* são ciclos de mensagens não periódicos de baixa prioridade. Exemplos deste tipo de tráfego são ciclos de mensagem no modo '*Acyclic Request* ou *Send/Request Mode*' e o registo de estações que estão activas (*Live List*).

O *GAP Update message cycle* é um ciclo de mensagem não periódico de baixa prioridade tal como um *Low priority message cycle* com a diferença apenas de este acontecer no contexto da manutenção da *GAP List*.

Escalonamento do tráfego

Como referido anteriormente, em PROFIBUS a posse do *Token* garante ao *Master* o acesso ao meio físico. Os *Masters* fazem circular entre si o *Token*. Ao receber o *Token* cada *Master* passa a executar o algoritmo [25] de escalonamento.

Calcular o período de tempo que o *Master* pode deter o *Token* (T_{TH}).

$$T_{TH} = T_{TR} - T_{RR}$$

Equação 1

Onde:

T_{TH}	*Token Holding Time*
T_{RR}	*Real Rotation Time*
T_{TR}	*Target Rotation Time*

O T_{RR} é o tempo que decorre entre duas recepções consecutivas do *Token* e o T_{TR} é o tempo que foi estimado inicialmente para a rotação do *Token*. Para ajudar na descrição deste algoritmo é apresentado um fluxograma na Figura 16.

Quando um *Master* recebe o *Token* pode sempre processar uma mensagem de alta prioridade mesmo que ($T_{TH} < 0$). Após esta mensagem se ainda dispuser de *Token Holding Time* ($T_{TH} > 0$) pode processar as outras mensagens de alta prioridade pendentes. Verifica se ($T_{TH} > 0$) antes de iniciar cada ciclo de mensagem. O *Master* só processa outros ciclos de mensagem se existir T_{TH} disponível ($T_{TH} > 0$) após transmitir todos os ciclos de mensagem de alta prioridade. É importante referir, que quando um ciclo de mensagem é iniciado (de alta ou baixa prioridade) é sempre terminado (com ou sem sucesso) incluindo as retransmissões (caso sejam necessárias) mesmo que T_{TH} expire ($T_{TH} < 0$) durante a execução desse ciclo de mensagem.

O processamento da *Poll* list só é iniciado após todos os ciclos de mensagem de alta prioridade terem sido processados. Um *Poll Cycle* consiste na consulta (através de ciclos de mensagem) a todas as estações contidas na *Poll* List. Após cada *Poll Cycle* completo os ciclos de mensagem de baixa prioridade são processados alternadamente, de acordo com as seguintes regras:

Se o *Poll Cycle* for completado dentro de *Token Holding Time* ($T_{TH} > 0$) os ciclos de mensagem de baixa prioridade são processados durante o restante *Token Holding Time*. Um novo *Poll Cycle* é iniciado na próxima recepção do *Token*.

Se no fim do *Poll Cycle* o *Master* não dispõe de *Token Holding Time* ($T_{TH} < 0$), os ciclos de mensagem de baixa prioridade pendentes são processados na próxima recepção do *Token* que tenha *Token Holding Time* disponível para processar ciclos de mensagem de baixa prioridade. De seguida é iniciado um novo *Poll Cycle*.

Se a execução do *Poll Cycle* durar várias rotações do *Token*, a *Poll* List é processada em segmentos, sem ser interrompida por ciclos de mensagem de baixa prioridade. Estes são sempre processados após o fim de um *Poll Cycle*.

Após todos os ciclos de mensagem de alta e baixa prioridade terem sido processados, se existir *Token Holding Time* disponível ($T_{TH} > 0$) e se T_{GUD} expirou o *Master* processa um ciclo de mensagem de actualização do GAP (apenas um por recepção do *Token*). Caso contrário ($T_{TH} < 0$) a actualização do GAP faz-se na próxima recepção do *Token* após todos os ciclos de mensagem de alta e baixa prioridade terem sido processados.

As *flags Poll_Turn* e *GAP_Turn* são utilizadas no fluxograma da Figura 16 como elementos de memória que permita decidir. Durante a inicialização a *flag Poll_Turn* deve ser colocada como *true* e a *flag GAP_Turn* deve ser colocada como *true*.

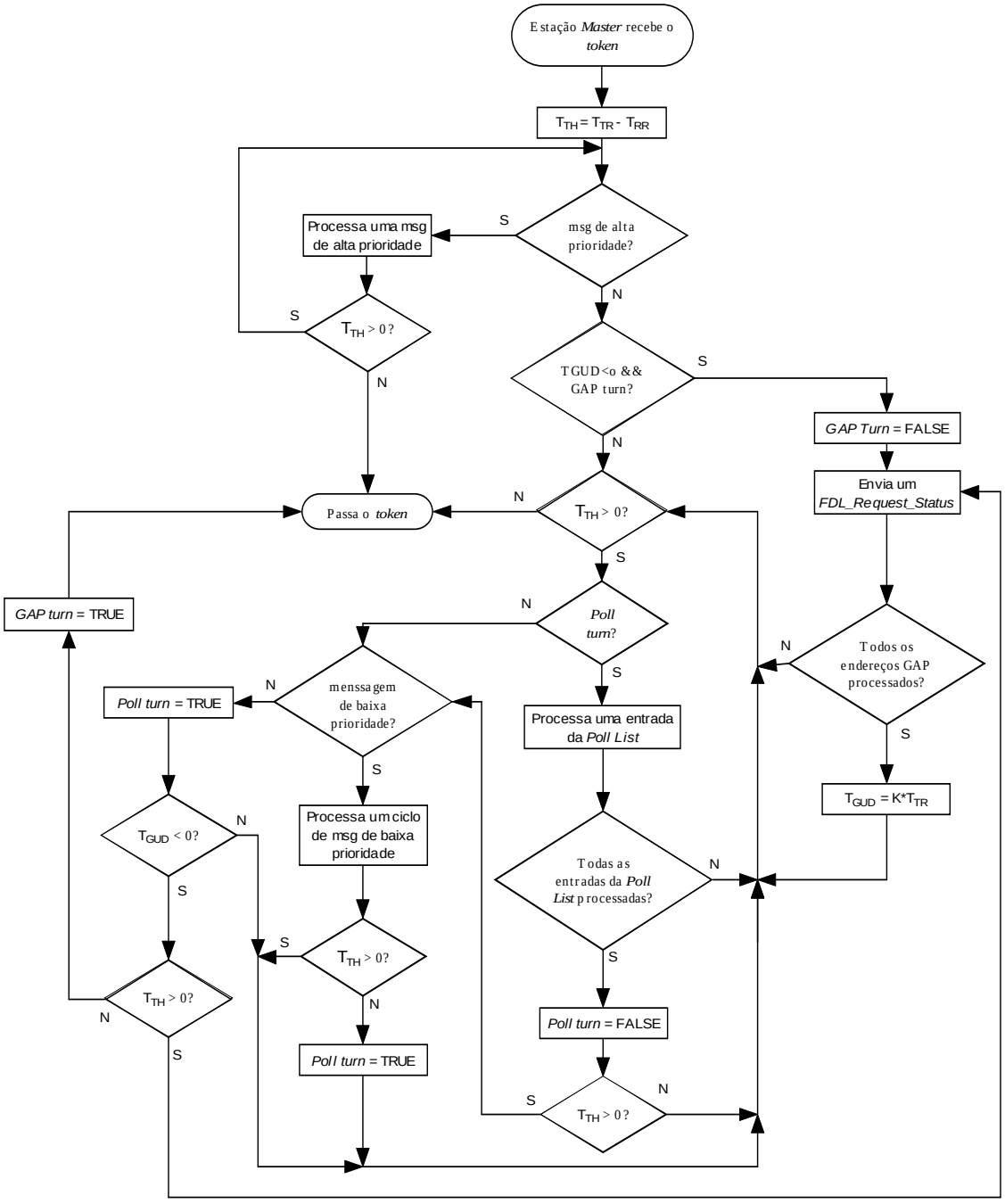


Figura 16 - Processamento de ciclos de mensagem

2.4.12 Máquina de estados do controlador da FDL

O controlador da FDL de um *Slave* só pode assumir dois estados, *Offline* ou *Passive_idle* [11], dado que um *Slave* não pode iniciar uma transmissão, respondendo apenas a pedidos do(s) *Master*(s)

O estado "*Offline*" será descrito de seguida no *Master*. A estação *Slave* entra no estado "*Passive_idle*" quando fica pronta para comunicar (quando dispõe de todos os parâmetros). Neste estado o *Slave* escuta o meio físico e se receber uma *Action Frame* endereçada a si envia a correspondente resposta/ *acknowledgement*.

O comportamento do controlador da FDL de um *Master* pode ser descrito através de dez estados [11] (Figura 17[10]):

- *Offline*

Dado que este estado depende da implementação, não está definido na norma. Neste estado o *Master* não comunica com a rede de comunicação. O controlador da FDL do *Master* assume este estado durante a sequência de *Power On* em que está a inicializar ou quando está a proceder a um auto teste. Terminando qualquer uma destas sequências o *Master* aguarda neste estado enquanto não tiver todos os parâmetros operacionais inicializados.

- *Listen Token*

Neste estado o controlador do *Master* escuta o meio físico de forma a identificar os *Masters* que estão no *Token-Ring* lógico e preenche a lista de estações activas (LAS) analisando os endereços inseridos nos *Token* PDU. Após o preenchimento da LAS (observou duas rotações idênticas do *Token*) o controlador da FDL deve permanecer neste estado aguardando receber um PDU endereçado a si com um "*Request* FDL *Status*" da estação que o precede PS, ao qual responde "*ready to enter logical Token-Ring*". Ao receber o *Token* endereçado a si, transita para o estado "Active_Idle".
Neste estado, se a FDL observa uma ausência de tramas por um período superior ao período de *time-out* assume um erro no *Token-Ring* lógico e transita para o estado *Claim Token* para reiniciar o anel lógico.

- *Claim Token*

Neste estado o controlador da FDL do *Master* tenta iniciar ou reiniciar o anel lógico.

- *Active Idle*

Neste estado o controlador da FDL do *Master* comporta-se como um *Slave*, escutando o meio físico sem se tornar activo. Se receber um pedido de um outro *Master* pode responder. Se receber um *Token* endereçado a si, transita para o estado "*Use_Token*".

- *Await Data Response*

---

[10] Os estados assinalados a cinzento correspondem aos estados de funcionamento em regime permanente da rede.

O controlador da FDL do *Master* entra neste estado após transmitir um pedido (*Action Frame*) e aguarda durante *Slot time* pela recepção de uma resposta válida ao pedido transmitido. Caso esta chegue antes do fim de um *slot time* retorna ao estado "*Use Token*"

- *Use Token*

O controlador da FDL do *Master* entra neste estado após receber o *Token* ou após reiniciar. É neste estado que são processados os ciclos de mensagem de alta e baixa prioridade.

De acordo com o algoritmo de despacho de mensagens atrás descrito (pág. 24), sempre que o controlador da FDL necessita de verificar o *Token Holding Time* (T_{TH}) transita para o estado "*Check Access Time*", do qual retorna caso exista T_{TH} disponível.
Após transmitir uma *Action Frame* o controlador da FDL transita para o estado "*Await Data Response*" no qual aguarda a chegada da resposta antes de decorrer o período de um *slot time*, se isto acontecer retorna a este estado "*Use Token*".

- *Pass Token*

Neste estado o controlador da FDL do *Master* transmite o *Token* PDU para a próxima estação (NS) no anel lógico. De seguida, transita para o estado "*Check Token Pass*".

- *Check Token Pass*

Neste estado o controlador da FDL do *Master* espera durante um *Slot time* pela reacção da estação para quem passou o *Token*. Caso a FDL detecte uma trama válida dentro do *slot time* assume que o *Token* foi transferido com sucesso e transita para o estado "*Active Idle*".

- *Check Access Time*

Neste estado o controlador da FDL do *Master* calcula o *Token Holding Time* (T_{TH}) disponível. Se existir, ($T_{TH} > 0$) retorna ao estado "*Use Token*", caso contrário transita para o estado "*Pass Token*".

- *Await Status Response*

Neste estado o controlador da FDL o *Master* espera durante um *Slot time* por uma trama de *acknowledge*.
O controlador da FDL do *Master* entra neste estado a partir do estado "*Pass Token*" quando não é conhecido um sucessor a quem transmitir o *Token* (inicialização ou manutenção do GAP). Se nada acontecer ou for recebida uma trama corrompida durante um *idle time*, retorna ao estado "*Pass Token*" para passar o *Token* a si próprio ou a um sucessor.

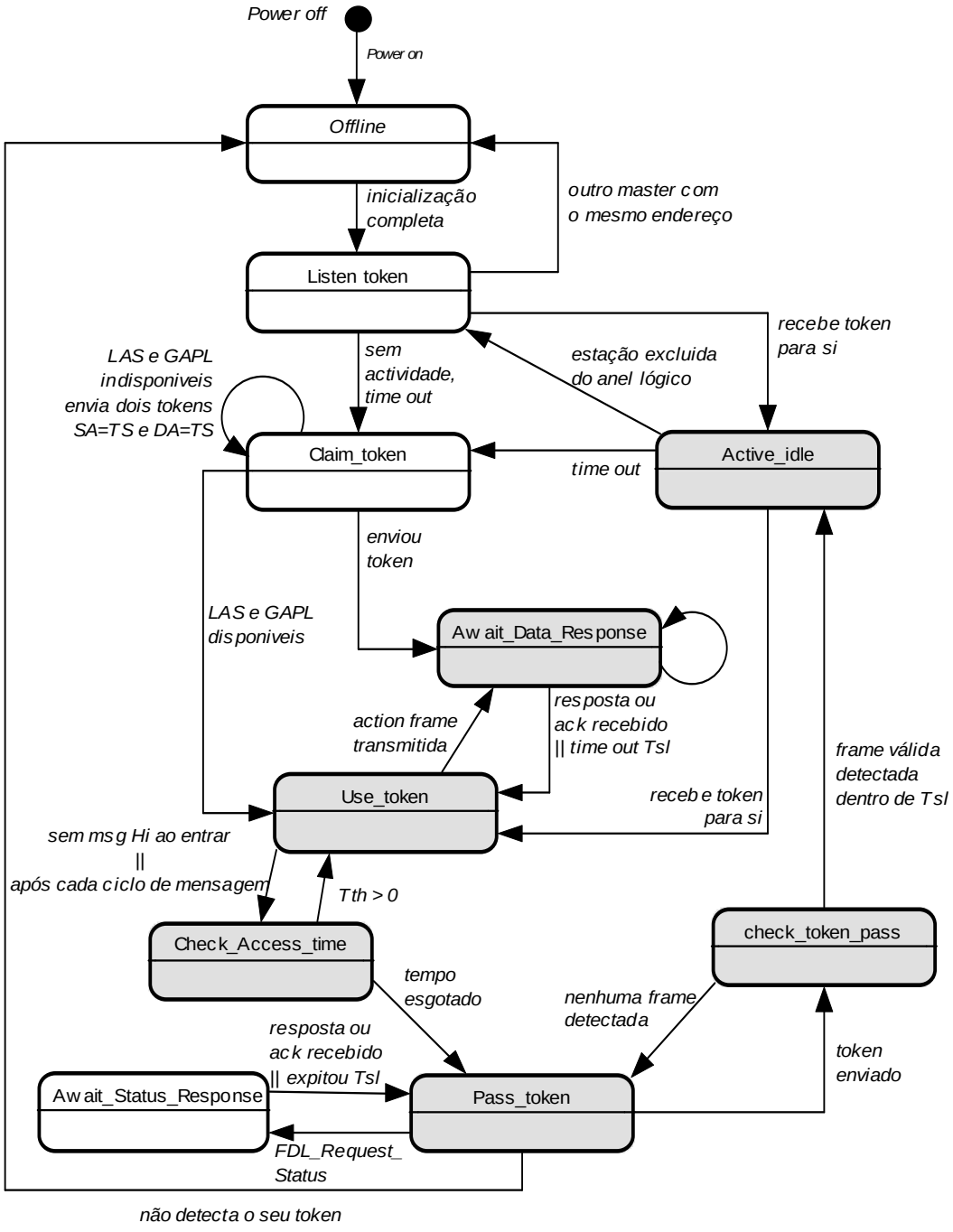


Figura 17 - Máquina de estados do controlador da FDL do *Master*

2.5 Camadas Superiores

O estudo apresentado nesta dissertação incide sobre as camadas baixas PHY e FDL da arquitectura da rede de comunicação PROFIBUS. Por essa razão a descrição dessas camadas foi exaustiva. Como se pode ver na Figura 1, na arquitectura de rede PROFIBUS apenas são definidas as camadas 1, 2 e a camada 7. Esta arquitectura genérica abrange dois perfis de comunicação, o PROFIBUS-FMS e o PROFIBUS-DP. No entanto, a arquitectura da rede de comunicação RFieldbus tem por base a arquitectura do PROFIBUS-DP tornando relevante a descrição da sua arquitectura, o que será feito de seguida.

2.5.1 PROFIBUS-DP

Num sistema de controlo com I/O distribuído (*Distributed Peripherals*) é comum uma configuração na qual a unidade central de controlo (*Master*), se encontra interligada a um conjunto de periféricos distribuídos (*Slaves*), dispostos ao longo da célula (*cell*) ou do campo (*Field*). Nestes casos, a transferência de dados dominante é originada de uma forma centralizada (pelo *Master*) e ocorre através de comunicações cíclicas que actualizam o estado de variáveis desses *Slaves* (*Distributed Peripherals*) no *Master* e vice-versa [10].

O PROFIBUS-DP é um perfil de comunicação cuja arquitectura foi orientada para o cumprimento dos requisitos temporais de controlo deste tipo de aplicação. Para atingir este objectivo de tornar este perfil de comunicação mais apto a cumprir requisitos temporais mais exigentes, a arquitectura foi simplificada ao mínimo indispensável.

No âmbito desta dissertação não serão descritas configurações de rede nas quais coexistem dispositivos PROFIBUS-FMS e PROFIBUS-DP. No entanto convém referir que numa rede exclusivamente PROFIBUS (apenas meio físico com fios e com funcionalidade PROFIBUS) é possível existirem estas configurações e que o acesso ao meio físico se faz também pelo anel lógico formado entre *Masters* (FMS e DP) que passam o *Token* entre si. Cada *Master* quando detém o *Token* processa os seus ciclos de mensagem.

2.5.2 Arquitectura Protocolar PROFIBUS-DP

Por motivos de eficiência o protocolo DP (Figura 18) não tem camada de aplicação (*layer* 7). O PROFIBUS-DP utiliza apenas os serviços de transferência de dados da FDL e da FMA 1/2. Para tornar mais confortável ao utilizador o acesso a interface da FDL é utilizado o DDLM (*Direct Data Link Mapper*) que contém funções de comunicação pré definidas DP.

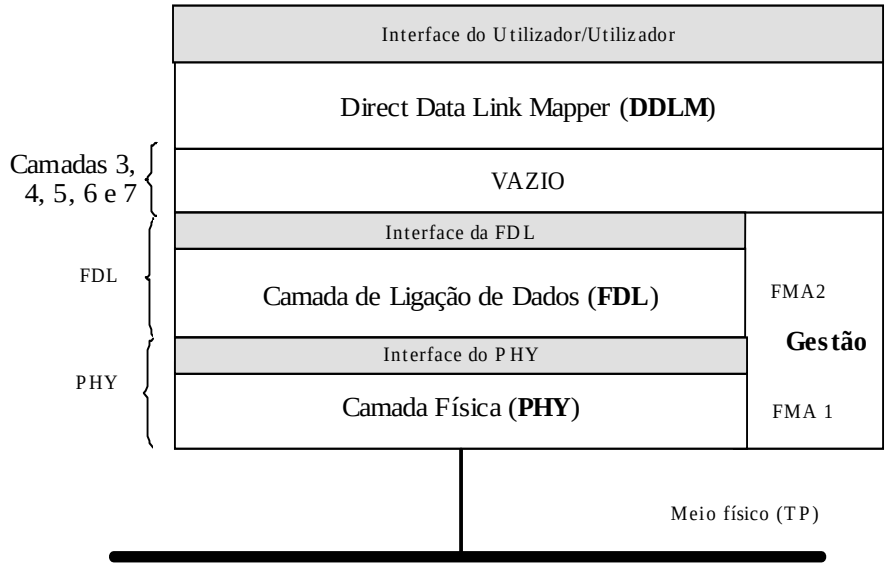


Figura 18 - Arquitectura Protocolar PROFIBUS-DP

2.5.3 Estações PROFIBUS-DP

No âmbito desta arquitectura/aplicação devido à funcionalidade a implementar estão definidas três tipos de estações PROFIBUS-DP [10]:

DP-*Master* (classe 1)

Este DP-*Master* controla diversos DP-*Slaves* de acordo com um algoritmo bem definido [11].

Este DP-*Master* faz um *polling* enviando e recebendo dados para/de o utilizador remoto de cada *Slave* associado. O *Master class* 1 comunica com o *Master class* 2 não só como emissor de pedidos mas também respondendo a pedidos deste.

DP-Master (classe 2)

Um *Master* classe 2 numa rede PROFIBUS-DP é uma consola de programação ou de operação (HMI) [10], contendo um conjunto de funções que suportam a gestão e o diagnóstico de uma rede PROFIBUS–DP. O utilizador do *Master* classe 2 define as operações que devem ser feitas dependendo do tipo de serviço requerido e da estação a ser endereçada.

DP-Slave

O DP *Slave* pode ser endereçado por ambas as classes de *Master*. Implementa um conjunto de funções de resposta.

2.5.4 Modelo de comunicação PROFIBUS-DP

Como se pode ver na Figura 19 a aplicação (*user*) de um DP *Master* (classe 1) e de um DP *Slave* (ex. um PLC como *Master* e um concentrador de I/O como *Slave*) comunicam através da interface de utilizador utilizando aplicações DP predefinidas.

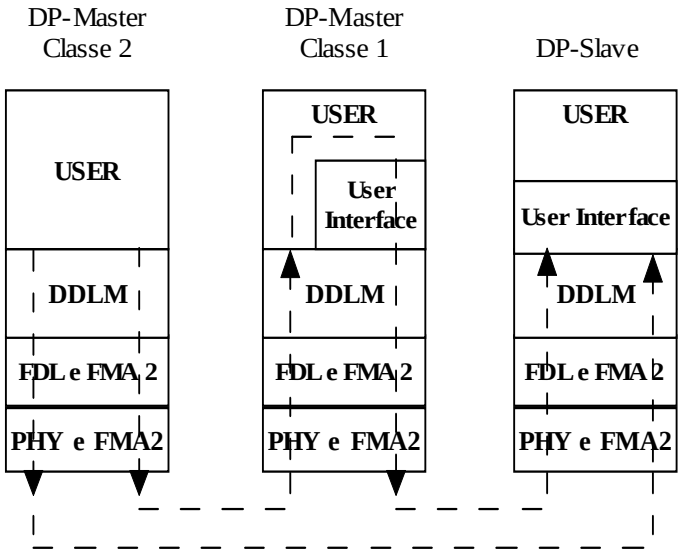


Figura 19 - Modelo de comunicação

2.5.5 Funções disponibilizadas ao utilizador de cada estação

PROFIBUS-DP Master (classe 1)

A interface de utilizador de um *Master* classe 1 implementa as seguintes funções de aplicações *Master-Slave* [10]:

- *read diagnostic information of DP-Slaves*
- *cyclic user data exchange mode*
- *parametrization and configuration checking*
- *submit control commands.*

Ao utilizador são disponibilizadas estas funções pela interface de utilizador. Assim é possível (através da rede de comunicações) pedir a informação de diagnóstico de cada DP-*Slave*, e não só atribuir os parâmetros a cada DP-*Slave* como também verificar as configurações existentes em cada DP-*Slave*. É também possível enviar comandos de controlo e requerer a função que activa a troca cíclica de dados. Esta é a função que estabelece efectivamente a comunicação. Quando esta função é invocada é estabelecido um ciclo de *polling* ao conjunto de DP-*Slaves* atribuídos ao DP-*Master* classe 1, de tal forma que em cada ciclo de comunicação as variáveis de I/O de cada DP-*Slave* atribuído ao DP-*Master* classe 1 são actualizadas no DP-*Master*.

Todas estas funções são processadas de forma independente face ao utilizador (*User*). A interface entre o utilizador e a interface de utilizador (*User interface*) consiste em algumas *service calls* e uma base de dados partilhada por ambos.

PROFIBUS-DP Master (classe 2) vs PROFIBUS-DP Slave

Um DP-*Master* classe 2 pode invocar as seguintes funções para comunicar com um DP-*Slave* [10].

- *read configuration of a DP-Slave*
- *read Input/Output-values*
- *address assignment to DP-Slaves*

Assim o DP-*Master* classe 2 pode ler directamente do DP-*Slave* a sua configuração, pode ler os valores de entradas e saídas de um DP-*Slave* e pode atribuir endereços a DP-*Slaves*. Dado que um DP- *Master* classe 2 é uma consola de programação ou de operação, as funções que ele implementa para comunicar com os DP-*Slaves* são funções que caem nesta categoria, de diagnóstico ou de inicialização/configuração.

PROFIBUS-DP Master (classe 2) vs PROFIBUS-DP Master (classe 1)

Um DP *Master* classe 2 invoca as seguintes funções de um DP-*Master* classe 1 [10]:

- *read the DP-Master's (class 1) diagnostic information of the associated DP-Slaves*
- *upload and download of parameters*
- *activate bus parameters*
- *activate and deactivate DP-Slaves*
- *select the operating mode of a DP-Master (class 1).*

O DP-*Master* classe 2 pode pedir ao DP-*Master* classe 1 que lhe envie a informação dos DP-*Slaves* associados, pode requerer ao DP-*Master* classe 1 para fazer upload e download de parâmetros e de seguida activar os parâmetros transferidos para o DP-*Slave* numa operação anterior, pode requer a activação ou desactivação de DP-*Slaves* e finalmente seleccionar o modo de operação do DP-*Master* classe 1. As funcionalidades que o DP-*Master* classe 2 pode invocar no DP-*Master* classe 1 são funcionalidades de diagnóstico ou de inicialização/configuração.

Estas funções são processadas pelo utilizador do DP-*Master* classe 1. A DDLM oferece ao utilizador e à interface de utilizador um acesso facilitado à interface da FDL (*Layer 2*).

2.5.6 Controlo de acesso ao meio físico

O Controlo do acesso ao meio físico e o protocolo de transmissão de dados estão de acordo com a EN 50170 Volume 2. O PROFIBUS-DP distingue estações activas e passivas. As estações activas formam um *Token-Ring* virtual, sendo o *Token* passado entre *Masters* ao longo de uma sequência circular definida (quando chega à ultima estação retorna à primeira).

As estações passivas não participam neste *Token-Ring* virtual. Estas são consultadas, no *polling* cíclico que cada *DP-Master* faz aos *DP-Slaves* que lhe estão atribuídos [11].

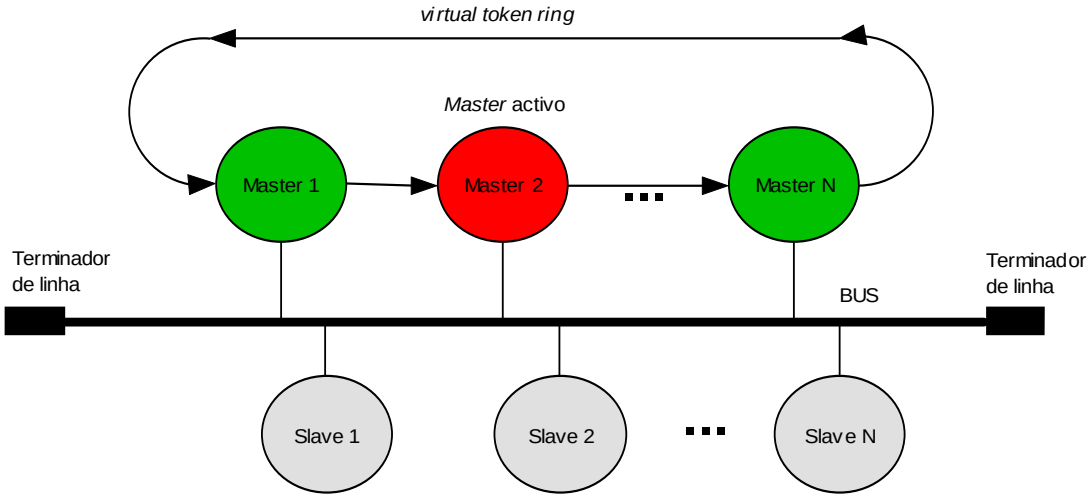


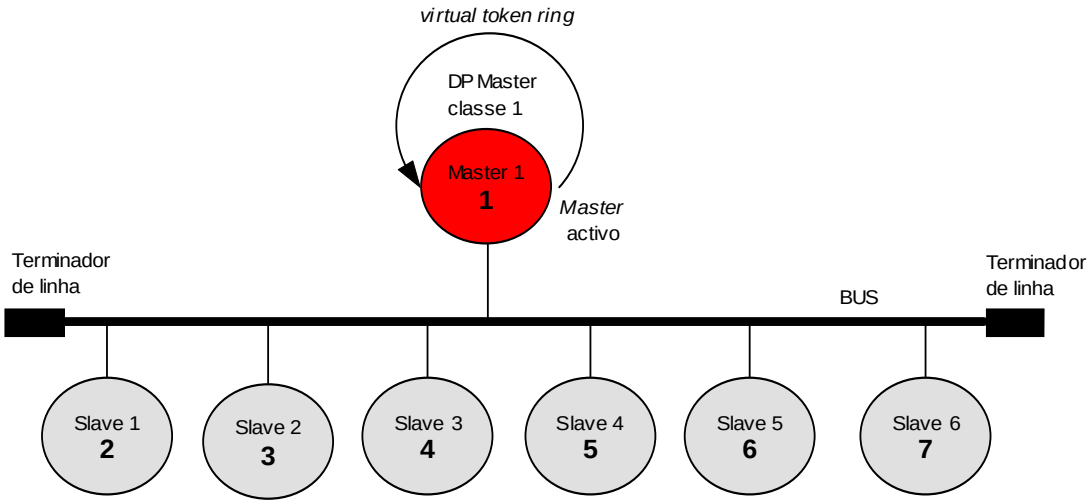
Figura 20 - Controlo do acesso ao meio no PROFIBUS DP

A posse do *Token* garante o acesso ao meio físico. Cada estação ao receber o *Token* usa-o para proceder ao *polling*, ao grupo de *DP-Slaves* que tem atribuídos e para processar outros ciclos de mensagem, posteriormente passa o *Token* para o *DP-Master* seguinte.

Cada DP-*Slave* está associado a um DP-*Master* classe 1 [10]. Um DP-*Master* classe 1 procede ciclicamente a um *polling* aos DP-*Slaves* associados. Um DP-*Master* classe 2 comunica com os DP-*Slaves* de uma forma acíclica.

2.5.7 Configurações de rede em PROFIBUS-DP

Mono-Master



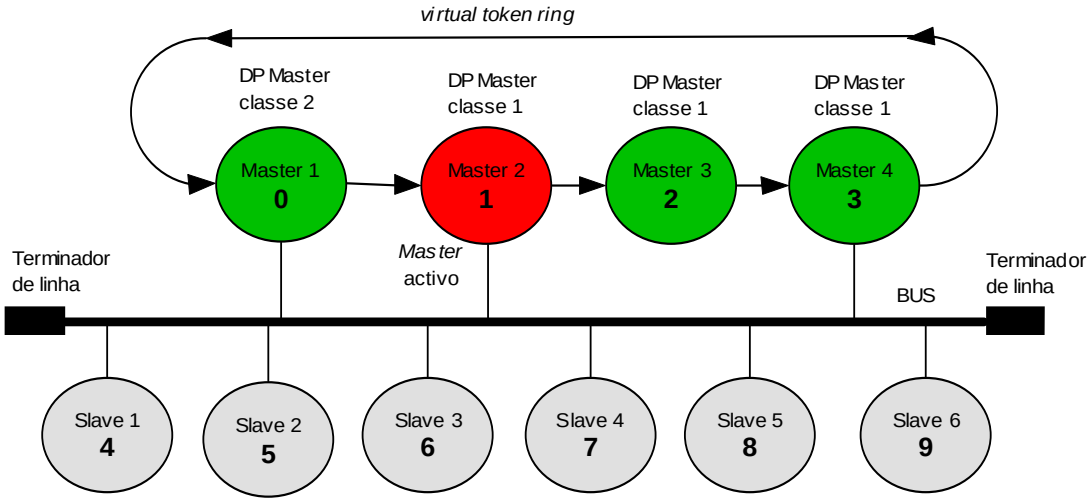
atribuições possiveis:

Master 1 (End 1) --> Todos os Slaves, End [2,7]

Figura 21 - Exemplo de um sistema *Mono-Master* em PROFIBUS-DP

A Figura 21 mostra um sistema *Mono-Master* em PROFIBUS-DP. Este sistema consiste num *DP-Master* classe 1 e até 125 *DP-Slaves* (com repetidores a cada segmento de 32 estações). De acordo com a especificação da camada de ligação de dados da EN 50170 (pág. 24) o *DP-Master* classe 1 passa o *Token* para si regularmente (sempre que não tem mais mensagens a processar ou quando T_{TR} expirou) recebendo-o logo de seguida. Este é um dos motivos pelo qual durante a transmissão do *Token* a estação mantém o receptor a escutar o meio físico enquanto transmite o *Token*, porque está a receber o *Token* que está simultaneamente a enviar (porque o atraso de transmissão pode considerar-se nulo).

Multi-Master



atribuições possiveis:

Master 2 (End 1) --> Slave 1 (End 4) e 4 (End 7)
Master 3 (End 2) --> Slave 2 (End 5) e 5 (End 8)
Master 4 (End 3) --> Slave 3 (End 6) e 6 (End 9)

Figura 22 – Exemplo de um sistema *Multi-Master* em PROFIBUS-DP

A Figura 22 mostra um sistema *Multi-Master* em PROFIBUS-DP. A atribuição de cada *DP-Slave* a um dos *DP-Masters* é feita durante a configuração do sistema e será guardada no conjunto de parâmetros de cada *DP-Master*. É possível ter até 126 dispositivos numa rede.

2.6 Sumário

Neste capítulo foi apresentada a arquitectura da rede de comunicação industrial PROFIBUS. Para a camada física (PHY) e de ligação lógica (FDL) foram descritas as funcionalidades implementadas por cada uma, a interface disponibilizada ao utilizador e os protocolos utilizados na implementação dessas funcionalidades.

Foi também feita uma descrição do perfil de comunicação orientado para redes de I/O distribuído PROFIBUS-*DP*, que é a tecnologia de base da rede de comunicação RFieldbus.

Capítulo 3

RFieldbus

3.1 Introdução

RFieldbus é uma rede de comunicação industrial com uma arquitectura definida no âmbito do projecto Europeu RFieldbus (IST-1999-11316). O objectivo deste projecto europeu foi desenvolver uma nova arquitectura de rede de comunicação industrial a partir de uma das definidas na normal EN50170 tal que essa nova arquitectura suportasse comunicações sem fios, mobilidade e integração de tráfego multimédia [13].

A plataforma de comunicação escolhida foi a EN 50170 V2 – PROFIBUS [6]. A nova arquitectura de rede representa uma evolução face plataforma de base, com a extensão das funcionalidades da mesma. Assim RFieldbus é PROFIBUS com novas funcionalidades, que estão implementadas num conjunto de elementos RFieldbus que podem coexistir e comunicar com dispositivos com a funcionalidade PROFIBUS na mesma rede de comunicação (embora neste caso a comunicação esteja limitada à funcionalidade PROFIBUS). Já na comunicação entre elementos RFieldbus é possível utilizar todas as funcionalidades adicionais implementadas. A implicação directa é que esta nova arquitectura a entrar no mercado permite a integração destas novas funcionalidades em redes PROFIBUS já instaladas [13].

A plataforma rádio escolhida foi o DSSS (Direct Sequence Spread Spectrum a 2.4GHz). No sentido de suportar as comunicações sem fios e a mobilidade de dispositivos, três novos tipos de estações foram definidos. A RFieldbus *Link Station* que permite a interligação de dispositivos sem fios a um segmento PROFIBUS (com fios). A RFieldbus *Base Station* que permite aumentar a cobertura rádio e melhora a fiabilidade das comunicações. A RFieldbus *Link Base Station* que combina as funcionalidades das estações referidas anteriormente numa só estação [13].

Com estas novas estações e considerando agora domínios de comunicação sem fios na rede de comunicação (em PROFIBUS tínhamos só domínios com fios) dois tipos base de topologias sem fios podem ser definidas: A *Direct Link Network* e a *Relay Link Network*. A *Direct Link Network* é uma topologia de rede de comunicação na qual os dispositivos sem fios comunicam directamente entre si (tal como dois rádio amadores) e a *Relay Link Network* é uma topologia de rede de comunicação na qual os dispositivos sem fios comunicam de forma indirecta entre si, através de uma *Base Station* [13].

Este capítulo é dedicado à descrição da arquitectura da rede de comunicação RFieldbus. Para isso começa por descrever cada um dos novos dispositivos introduzidos por esta nova arquitectura de rede. De seguida são definidos os conceitos de domínio de comunicação, com e sem fios (*Wired* e *Wireless*). As topologias atrás referidas são descritas com mais detalhes e são apresentados os aspectos de funcionamento, as vantagens e desvantagens de cada uma das duas topologias. De seguida um sub capítulo é dedicado à explicação da gestão da mobilidade de dispositivos sem fios (nós sem fios e segmentos com fios móveis) e para os casos de mobilidade dentro de um domínio de

comunicação (*Intra-cell mobility*) e mobilidade entre domínios de comunicação (*Inter-cell mobility*). De seguida são explicadas as redes estruturadas com base nestas duas topologias base e posteriormente descrita a arquitectura do protocolo de comunicação RFieldbus. O estudo apresentado nesta dissertação centra-se nas camadas baixas do RFieldbus que são as camadas 1 e 2 do PROFIBUS. As alterações ao protocolo de ligação de dados na camada 2, consistiram na introdução de uma sub-camada DLX de interface com a camada de gestão que disponibilizam um conjunto de serviços a esta camada. Assim o protocolo da FDL é o do PROFIBUS, que foi descrito no capítulo anterior (página 13)

A arquitectura RFieldbus suporta a comunicação em dois tipos de meio físico diferentes. Relativamente ao meio físico com fios (*Wired*) a descrição de protocolo e interface foi feita no capítulo anterior. Neste capítulo é apresentado o protocolo e a interface física do meio físico sem fios com uma descrição das funcionalidades implementadas por cada sub-camada que o constitui, a apresentação do formato da trama utilizada no meio físico sem fios.

3.1.1 Elementos constituintes

Além das estações RFieldbus *Master* e *Slave* que compreendem as aplicações e as funcionalidades de comunicação do PROFIBUS, existem outros três elementos RFieldbus.

- RFieldbus *Link Station*
- RFieldbus *Base Station*
- RFieldbus *Link Base Station*

Sempre que a comunicação tem lugar num meio físico de cabo (no caso do PROFIBUS par entrançado) designamos esse meio físico de *Wired*. Sempre que a comunicação tem lugar num meio físico sem fios designamos esse meio físico de *Wireless*.

A RFieldbus *Link Station* permite a ligação de um meio físico sem fios a um meio físico com fios. A RFieldbus *Base Station* permite maximizar a cobertura rádio bem como aumentar a fiabilidade das comunicações e finalmente a RFieldbus *Link Base Station* que combina a funcionalidade das duas estações anteriores.

Os RFieldbus *End Systems ES* (*Master* e *Slave*) podem ter interface para rede com fios (WRES) ou para rede sem fios (WLES).

RFieldbus *Link Station*

A RFieldbus *Link Station* liga um meio físico sem fios a um meio físico com fios [13]. Esta estação tem uma *interface* para o meio físico com fios e outra para o meio físico sem fios. O que ela faz é repetir (*Relay*) as tramas que recebe de um dos meios físicos para o outro. A figura seguinte mostra a sua estrutura.

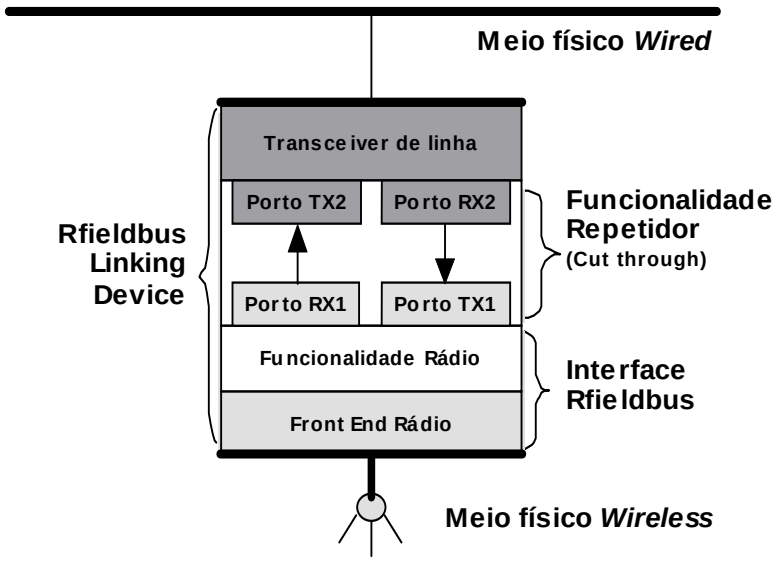


Figura 23 - Estrutura de uma RFieldbus *Link Station*

Dado que o formato das tramas dos meios físicos com e sem fios não são idênticos, a repetição das tramas não é uma repetição de sinais eléctricos. A *Link Station* retira a informação contida na trama recebida e insere-a na trama do outro meio físico. Este processo introduz uma latência na retransmissão (*relaying*) das tramas. Essa latência depende não só das latências físicas do hardware, mas também de um atraso que tem que ser inserido devido ao facto dos meios físicos terem tramas com formatos diferentes e consequentemente com durações de transmissão diferentes. Assim é necessário garantir que a *Link Station* inicia a retransmissão de um PDU num instante tal, que a transmissão do PDU no outro meio físico ocorra sem interrupções [12]. Esta situação acontece quando uma rede com dois meios físicos diferentes, Com e sem fios. Na Figura 24 está representada a duração dos mesmos dados inseridos no PDU do meio físico sem fios (a) e do meio físico com fios (b).

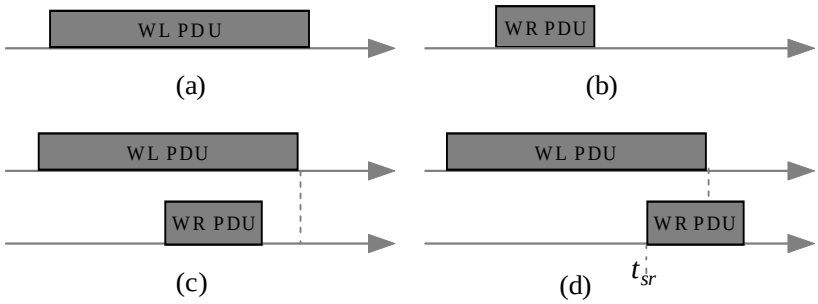


Figura 24 - Instante para iniciar a retransmissão da trama

No sentido de minimizar a latência introduzida pela *Link Station* esta deve iniciar a retransmissão do PDU recebido, quanto antes. No entanto dado que o PDU recebido (a) tem uma duração temporal superior ao correspondente PDU a ser transmitido (b) a *Link Station* só pode iniciar essa transmissão quando for possível faze-la se forma continua. Isto porque um PDU tem que

ser transmitido no meio físico de forma continua (sem interrupções). A situação ilustrada em (c) não é possível pois a *Link Station* termina a transmissão do WR PDU sem receber todos os dados do WL PDU. A situação ilustrada em (d) mostra a situação real em que t_{sr} é o instante de tempo a partir do qual a *Link Station* pode iniciar a retransmissão do PDU.

RFieldbus *Base Station*

Uma *Base Station* permite aumentar a cobertura rádio do meio físico sem fios [13]. Esta estação reenvia para o meio físico todas as tramas que recebe dele. Tal como a *Link Station* faz uma repetição das tramas, mas neste caso repete-os para o mesmo meio físico (sem fios). O que ela faz é reenviar para o meio a mesma trama como sinal rádio reforçado. Neste processo podem surgir colisões, para as evitar existem duas soluções.

A primeira solução é utilizar uma *Base Station* com um repetidor *Store and Forward*. A vantagem desta solução é só necessitar de um *Radio Font-End* pois a estação recebe a trama completa e só depois a reenvia. Assim pode utilizar o mesmo canal rádio para o efeito pois não há possibilidade de existir colisão (a recepção e a retransmissão ocorrem em períodos de tempo diferentes). A desvantagem desta solução é a latência adicional introduzida pela espera da recepção completa da trama até começar a reenvia-la para o meio físico. A Figura 25 ilustra uma *Base Station* cujo repetidor tem uma funcionalidade *Store and Forward*. Esta *Base Station* funciona em *Half Duplex*.

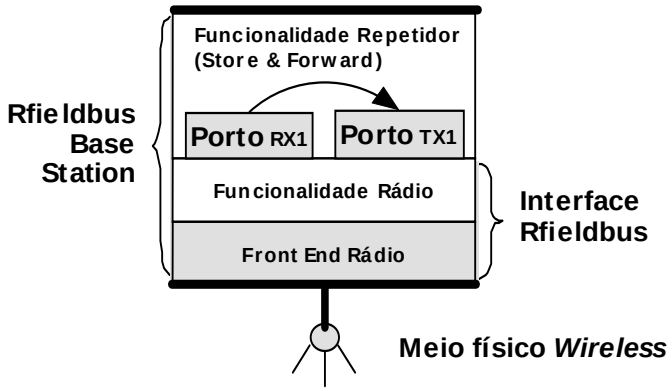


Figura 25 - Estrutura de uma *Base Station* com um Radio Front-End

A segunda solução é utilizar uma *Base Station* com um repetidor *cut-through* que elimina esta latência ao iniciar a transmissão da trama logo que possível. Para isso utilizar dois *Radio Font-End* e dois canais rádio, um para receber tramas do meio físico (no *uplink channel*) e outro para enviar as tramas para o meio físico (no *downlink channel*). A Figura 26 ilustra uma *Base Station* cujo repetidor tem uma funcionalidade *cut-through*. Esta *Base Station* funciona em *Full Duplex*.

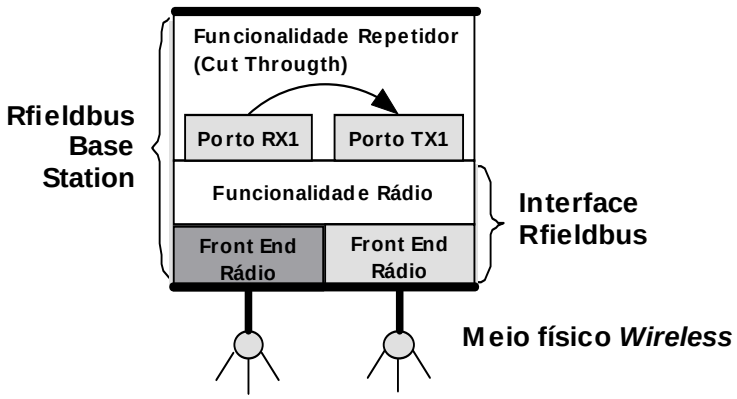


Figura 26 - Estrutura de uma *Base Station* com dois Radio Front-End

RFieldbus *Link Base Station*

A funcionalidade de uma RFieldbus *Base Station* pode ser combinada com a de uma RFieldbus *Link Station*, resultando na estação RFieldbus *Link Base Station* [13]. Assim com apenas uma estação podemos criar um domínio *Wireless* estruturado (ver 3.1.2) e simultaneamente interligar esse domínio *Wireless* com um domínio *Wired* (ver 3.1.2). A Figura 27 ilustra a estrutura de uma RFieldbus *Link Base Station.*

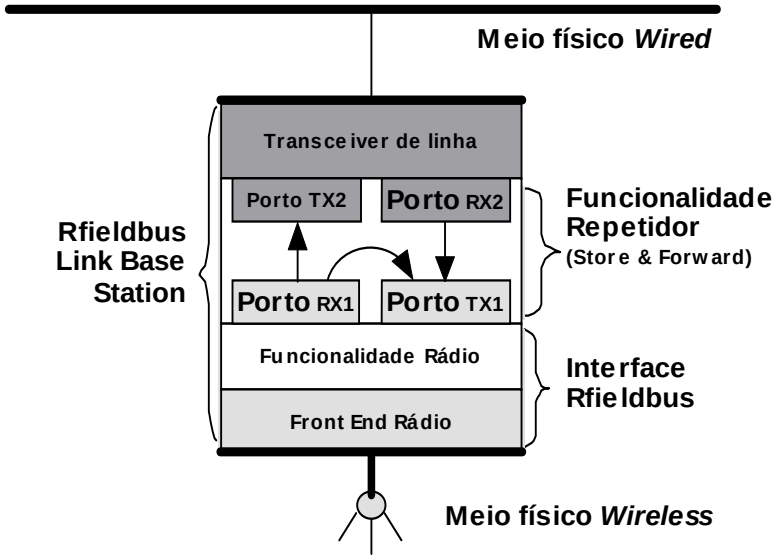


Figura 27 - Estrutura de uma *Link Base Station*

Como se vê na Figura 27, nesta estação as tramas recebidas do meio físico com fios são retransmitidas no meio físico sem fios. As tramas recebidas do meio físico sem fios são retransmitidas para ambos os meios físicos (com e sem fios).

3.1.2 Wired e Wireless Communication Domains

O conjunto de *End Systems* e *Intermediate Systems* que comunicam directamente através de um meio físico com fios é chamado de *Wired Communication Domain*, ou *Wired Domain*. Uma rede de comunicação constituída exclusivamente por *Wired Domains* é chamada *Wired Network* [13].

Uma célula rádio (RC) é definida como uma área de cobertura rádio comum de um grupo de *Wireless End Systems* WLES e *Link Stations* LS. O conjunto de WLES e LS que definem uma célula rádio são chamados *Wireless Communication Domain*, ou *Wireless Domain* (WLD). Uma rede de comunicação constituída exclusivamente por *Wireless Domains* é chamada *Wireless Network* [13].

Um domínio *Wireless* pode ser Ad-hoc (AWLD) ou estruturado (SWLD). A diferença é que num AWLD os WLES's/LS's comunicam directamente entre si enquanto num SWLD o fazem de forma indirecta através de uma BS ou LBS (*Link Base Stations*).

A mobilidade inter-célula só é possível num domínio de comunicação estruturado. O WLES tem que suportar o procedimento de *Handoff* (pág. 46) para poder deslocar-se entre células adjacentes sem perda de conectividade à rede de campo [13]. A necessidade de criar SWLD surge com a necessidade de obter uma maior cobertura rádio, quer pela expansão física da rede (abranger maior área ou volume) ou para obter cobertura em zonas difíceis (devido a obstáculos ou interferência electromagnética).

Existe um outro tipo de domínio, é o *Mobile Wired Domain* MWRD. Consiste num *Wired Domain* com um conjunto de ES associados e com uma LS ou LBS. É particularmente útil em Veículos autoguiados e outras aplicações móveis.

Uma referência que importa fazer é que, de forma alguma a mobilidade permite que um nó ou ramo sem fios móvel, possa transitar de uma rede RFieldbus para outra. Pode como já foi referido mover-se dentro de um domínio sem fios ou entre domínios sem fios estruturados da mesma rede.

3.2 Topologias

A comunicação numa célula rádio pode efectuar-se directamente entre os nós sem fios numa *Direct Link Network* ou via uma *Base Station* numa *Relay Link Network*. De seguida ambas as topologias existentes em RFieldbus serão descritas com mais detalhe.

3.2.1 Direct *Link Network*

A topologia de rede *Direct Link Network* é caracterizada pela comunicação directa entre todos os nós sem fios dentro de determinada área sem o suporte de uma *Base Station* [13]. A cobertura rádio neste domínio sem fios resulta numa área comum de alcance entre todos os nós (para que todos possam falar com todos) chamada de área comum de cobertura rádio ou célula rádio. Nesta topologia os nós gozam de mobilidade desde que se mantenham dentro da área comum de cobertura rádio. A este tipo de mobilidade chama-se *Intra-cell Mobility*. A Figura 28 ilustra o que foi dito.

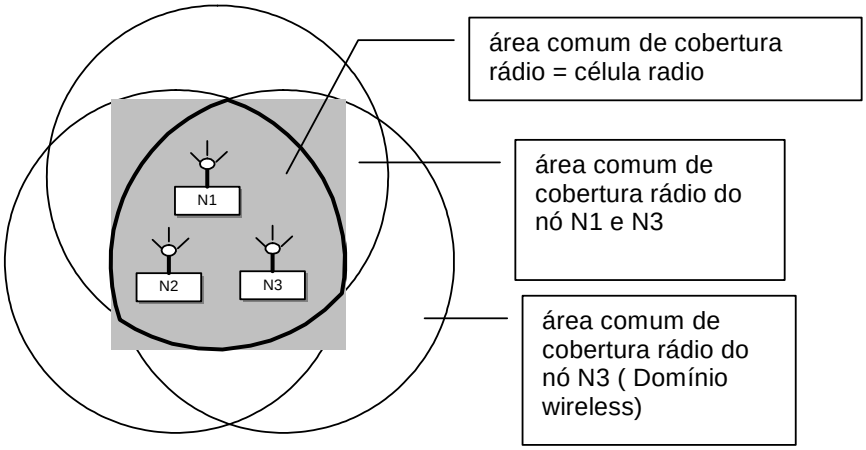


Figura 28 - Definição de uma célula rádio numa Direct *Link Network*

De referir que este caso de *Intra-Cell Mobility* é operacionalmente transparente dado que a mobilidade de um dos nós não implica qualquer alteração nas características da comunicação (considerando que o nó se mantém na área comum de cobertura rádio). A topologia *Relay Link Network* também permite a mobilidade de nós *Wireless*. *Intra-cell* caso estes se desloquem dentre a célula rádio e *Inter-Cell* caso estes se desloquem entre células rádio. Este último tipo de mobilidade implica um mecanismo adicional e por isso ser-lhe-á dedicada um sub capítulo subsequente.

Na Figura 28 está representada a versão mais simples desta topologia constituída apenas por nós *Wireless*. Dado que não são utilizadas *Link Stations* ou *Base Stations* não serão introduzidos atrasos de transmissão adicionais. Assim esta é a versão mais rápida de uma rede *Wireless*.

Se adicionarmos um ou mais segmentos com fios a uma célula rádio obtemos uma topologia chamada *Extended Direct Link Network*. Cada segmento com fios é adicionado, usando uma *Link Station*.

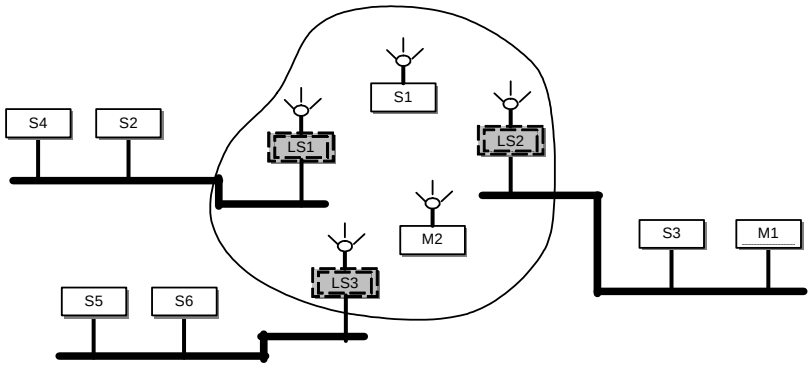


Figura 29 - Extended Direct *Link Network*

Do ponto de vista de célula rádio a *Link Station* e o segmento com fios a ela acoplado podem ser vistos como um nó sem fios especial. Nesta topologia o segmento com fios goza de mobilidade desde que a *Link Station* se mantenha dentro da área comum de cobertura rádio (*Intra-cell*).

Um segmento com fios pode ser ligado a duas ou mais células rádio usando uma *Link Station* para liga-lo a cada uma das células rádio. Em cada uma dessas células só pode residir uma das *Link Station* ligadas ao segmento com fios. Isto permite criar rede estruturadas, adaptadas à realidade da planta fabril onde a rede vai ser instalada.

3.2.2 *Relay Link Network*

A vantagem desta topologia é o aumento da cobertura rádio. Para isso é utilizada uma *Base Station* através da qual passam todas as comunicações entre dois nós sem fios associados ao domínio sem fios formado por essa *Base Station* [13]. Neste caso o domínio sem fios é formado pela área comum entre o alcance do *Wireless End System* e o alcance rádio da *Base Station*. A Figura 30 demonstra este conceito.

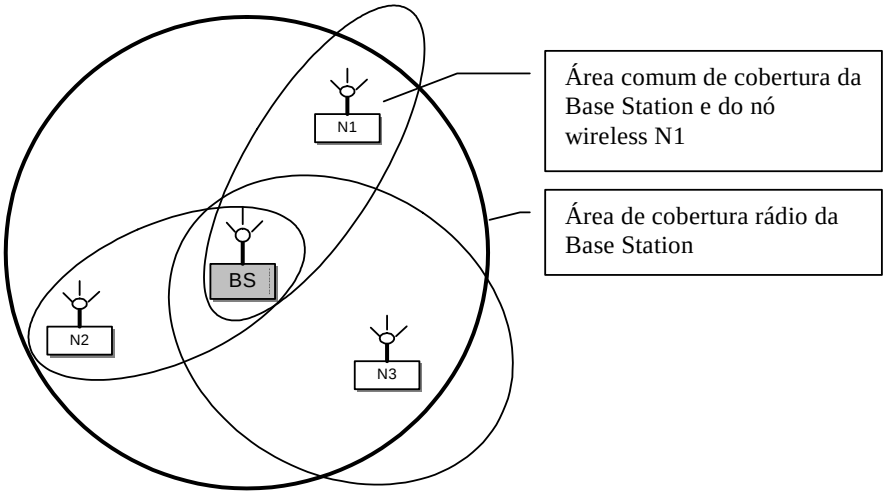


Figura 30 - Definição de uma célula rádio numa *Relay Link Network*

Esta topologia é *Relay Link Network* porque a *Base Station* faz uma retransmissão de todas as tramas que circulam no domínio sem fios. Por este facto é introduzida uma latência adicional em todas as tramas. É esta a troca a fazer para se ter uma maior cobertura rádio. Na Figura 30 temos uma *Relay Link Network* apenas com nós sem fios. Esta é a versão mais simples desta topologia. Dentro do domínio de comunicação da *Base Station* os nós gozam de mobilidade (*Inter-cell*).

Podemos adicionar a uma *Relay Link Network* alguns segmentos com fios. A rede seria, a representada na Figura 31.

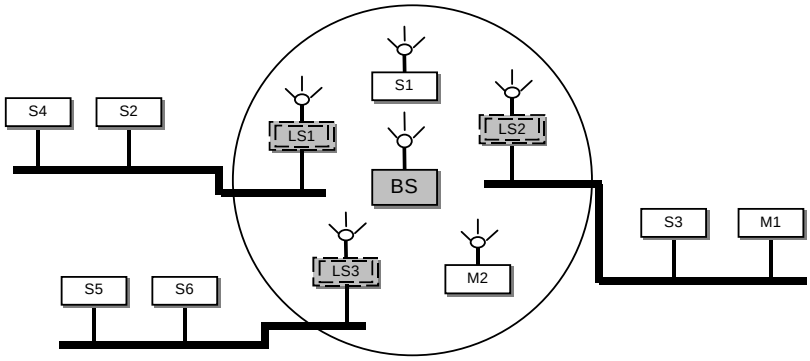


Figura 31 - *Relay Link Network* com segmentos *Wired*

Uma outra possibilidade para uma *Relay Link Network* é criá-la recorrendo a uma *Link Base Station* como ilustrado na figura seguinte.

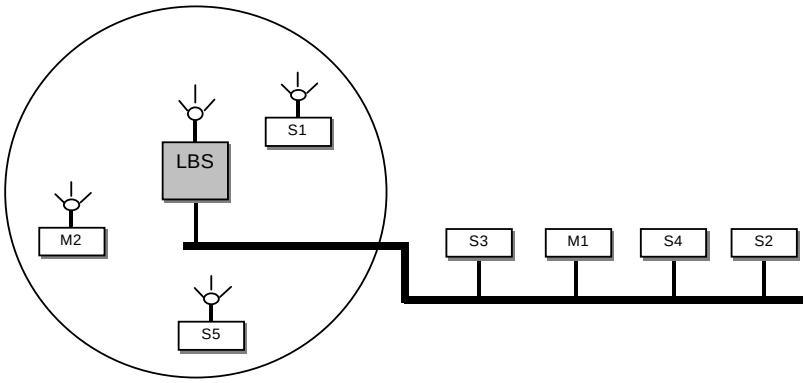


Figura 32 - *Relay Link Network* com um segmento *Wired* e LBS

3.3 Mobilidade

Em RFieldbus os nós *Wireless* podem ter uma posição fixa mas também podem mover-se. Esta mobilidade permite que se desloquem não só dentro de uma célula rádio (*Intra-Cell Mobility*) mas também entre células adjacentes (*Inter-cell* Mobility) [13]. Este ultimo tipo de mobilidade necessita de um mecanismo de *Handoff* [13], que consiste basicamente na gestão do processo de passagem (do ponto de vista da rede) de um dispositivos sem fios de um domínio sem fios para outro adjacente devido ao deslocamento físico deste para a área de cobertura rádio do domínio sem fios de destino.

3.3.1 Intra-*cell* mobility

O suporte de mobilidade intra-célula é bastante simples dado que não existe nenhuma mudança no domínio *Wireless* (WLD). Isto significa que nenhum ES/IS vai sair ou entrar no WLD e o canal rádio é o mesmo.

3.3.2 Inter-*cell* mobility

As células rádio podem reutilizar canais rádio desde que esta reutilização não sobreponha canais idênticos. Transmitir duas mensagens diferentes em dois canais idênticos sobrepostos provoca interferência e a perda dos dados transmitidos.

Células rádio adjacentes têm que operar em canais rádio diferentes para poderem sobrepor-se. Desta forma é possível criar uma malha de canais adjacentes (estruturados) funcionando num mínimo de três frequências diferentes.

Quando os requisitos de mobilidade de pelo menos um dispositivos móvel (WLES e/ou MWRD) são tais que o alcance rádio de um Ad-hoc WLD não é suficientemente abrangente para suportar o deslocamento desse dispositivo móvel, é necessário criar um conjunto de domínios *Wireless* estruturados ao longo dos quais é fornecida conectividade contínua aos MWLES e MWRD ao longo do seu percurso.

A cada mudança entre domínios *Wireless* SWLD o dispositivo móvel efectua uma avaliação da qualidade dos canais rádio disponíveis mudando para o domínio *Wireless* que oferece o melhor canal rádio (*Handoff*).

Dado que a plataforma de comunicação do RFieldbus é o PROFIBUS que é uma rede de comunicação *Broadcast* com um único anel lógico e espaço de endereçamento único não há necessidade de um mecanismo de registo no procedimento de *Handoff*.

A solução proposta para o procedimento de gestão da mobilidade permite um *Handoff* eficaz para todo o tipo de estações móveis, WLES (*Master* e *Slave*) e para os *Mobile Wired Domains* MWRD.

O procedimento de gestão da mobilidade resume-se a uma avaliação dos canais rádio disponíveis e subsequente mudança para o de melhor qualidade [12]. Não obstante, este procedimento garante que não há perda de tramas devido à mobilidade dos dispositivos (em condições normais de funcionamento, ou seja, sem falhas) e permite o cumprimento de requisitos de tempo real. A duração deste procedimento é limitada a um curto intervalo de tempo. A título de exemplo, em topologias de complexidade elevada uma duração do procedimento de gestão da mobilidade é inferior a 2 ms.

Na Figura 33 podemos observar a deslocamento de um MWLES (*Slave*) entre WLD's (CH1 → CH3).

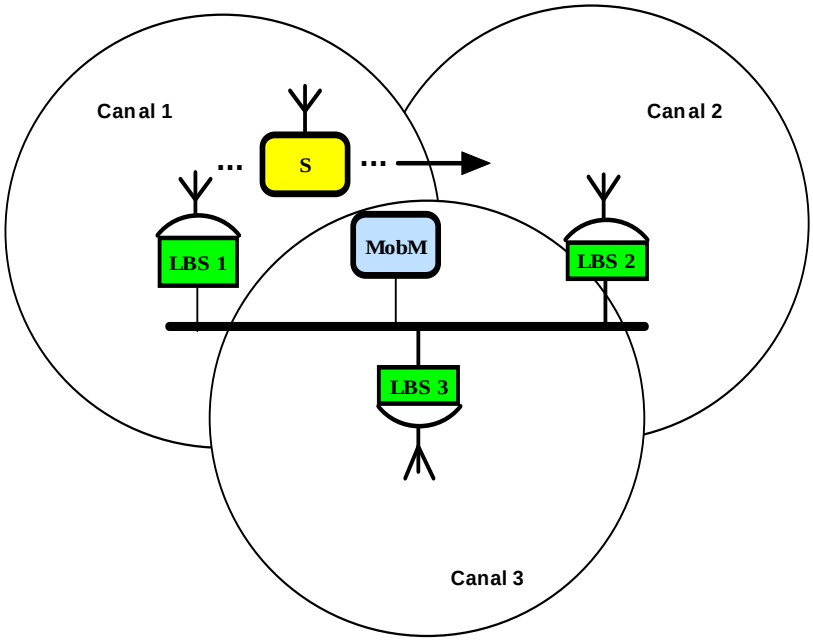


Figura 33 - Transição entre domínios estruturados (*Handoff*)

A responsabilidade de desencadear o procedimento de gestão da mobilidade é do *Mobility Master*. A periodicidade com a qual é iniciado o procedimento de gestão da mobilidade é dependente da velocidade máxima das estações móveis. No caso da Figura 33, como só existe um MWLES é a sua velocidade máxima que vai determinar a periodicidade do procedimento de gestão da mobilidade.

Quando *Mobility Master* pretende desencadear o procedimento de gestão da mobilidade aguarda a chegada de um *Token* endereçado para si e quando detém o acesso ao meio envia uma trama especial sem *acknowledge*, o *BT PDU* (*Beacon Trigger* PDU). A Figura 34 ilustra este processo.

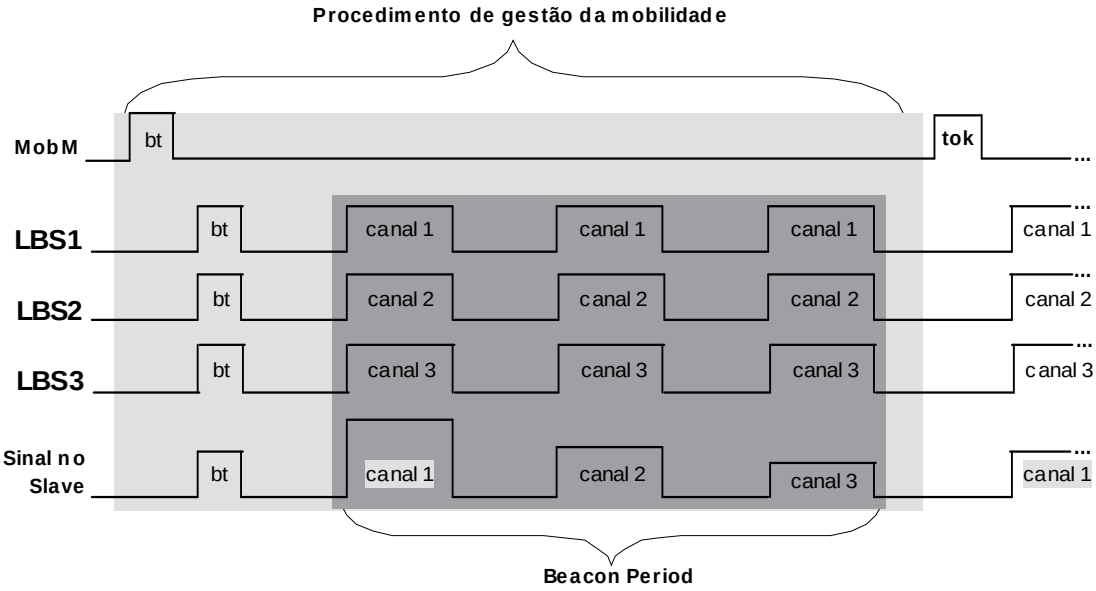


Figura 34 - Diagrama temporal do procedimento de gestão da mobilidade

O *BT PDU* é enviado pelo *Mobility Master* e vai ser recebido e retransmitido pelas (*Link*) *Base* Stations. A recepção do *BT PDU* pelas (*Link*) *Base Stations* faz com que estas comecem a emitir *beacons* (tramas especiais) no seu canal (sua frequência) permitindo às estações móveis avaliar a qualidade dos canais disponíveis (*channel assessment*) e comutarem para o que apresenta melhor qualidade (*Handoff*). Os *Slaves* móveis ao receberem o *BT PDU* sabem que se deu início ao procedimento de gestão da mobilidade e iniciam a avaliação da qualidade dos canais rádio disponíveis.

Considerando o cenário da Figura 33 no qual a estação móvel se desloca em direcção ao SWLD da LBS2 é possível observar na Figura 34 que o *Mobility Master* (MobM) envia o *BT PDU* que é recebido e retransmitido pelas *Link Base Station*. O *Slave* móvel ao receber o *BT PDU* sabe que se deu início ao *Beacon Period* no qual vai avaliar a qualidade dos canais rádio disponíveis através do *beacons* recebidos. No final, o canal rádio com melhor qualidade de sinal é o canal um e é este o canal escolhido. Ou seja no final do procedimento de gestão da mobilidade o *Slave* móvel passa a comunicar no canal rádio do SWLD CH1.

Após completar o procedimento de gestão da mobilidade, o *Master* passa o *Token* para o próximo *Master* no *Token-Ring* lógico. Dado que o ciclo de mensagem associado a este procedimento (*BT PDU*) não tem um *acknowledge* ou *response* que indique o fim do mesmo e de acordo com o protocolo, é calculada a duração máxima do procedimento de gestão da mobilidade (t_{mob}) e a partir deste o valor do t_{id2mob} especifico para o *mobility Master*. Ou seja, após enviar o *BT PDU* o MobM aguarda t_{id2mob} até devolver o *Token*.

3.4 Redes estruturadas

Em teoria todas as topologias de rede atrás mencionadas podem ser combinadas para construir a rede mais ajustada à aplicação final. Os limites nas topologias complexas são impostos pelos requisitos temporais do RFieldbus, o que em situações práticas pode limitar o desempenho da rede de comunicação.

Redes estruturadas são redes obtidas por concatenação das duas topologias base atrás referidas [13]. De seguida serão apresentados dois exemplos deste tipo de redes.

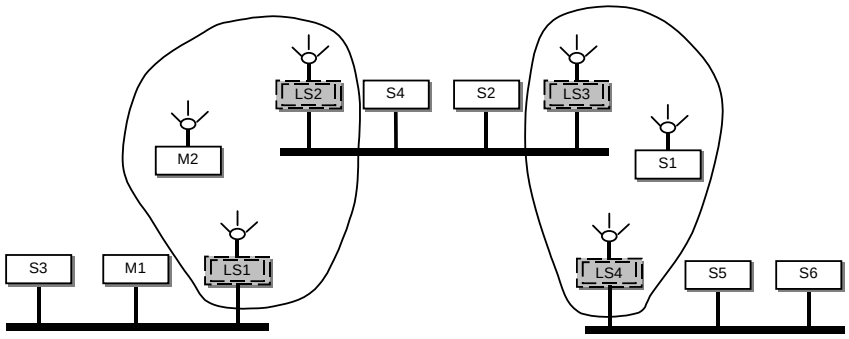


Figura 35 - Concatenação de 3 Direct *Link Network*

Na Figura 35 estão três segmentos com fios interligados por domínios sem fios. Em ambos os casos a ligação faz-se através de domínios sem fios não estruturados (*Direct Link*) no qual as *Link Stations* comunicam directamente entre si.

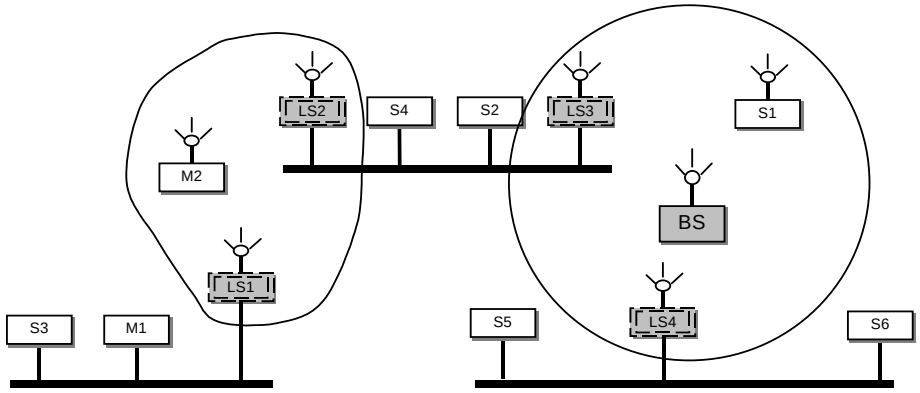


Figura 36 - Concatenação de 3 segmentos *Wired* com BS

Na Figura 36 estão três segmentos *Wired* interligados por domínios *Wireless*. No lado esquerdo a ligação faz-se com um domínio *Wireless* não estruturado (*Direct Link*) no qual as *Link Station* comunicam directamente entre si. No lado direito a interligação dos segmentos faz-se através

de um domínio *Wireless* estruturado por uma *Base Station*. Neste caso todas as comunicações da rede ao atravessarem este SWLD vão passar pela BS.

Em ambos os casos da Figura 35 e Figura 36 há que ter em atenção que dado que as topologias são obtidas por concatenação os atrasos sofridos pelos ciclos de mensagem aumentam e nos casos de ES afastados entre si (do ponto de vista da rede de comunicação) poderemos ter que admitir latências elevadas [13].

3.5 Arquitectura da rede de comunicação RFieldbus

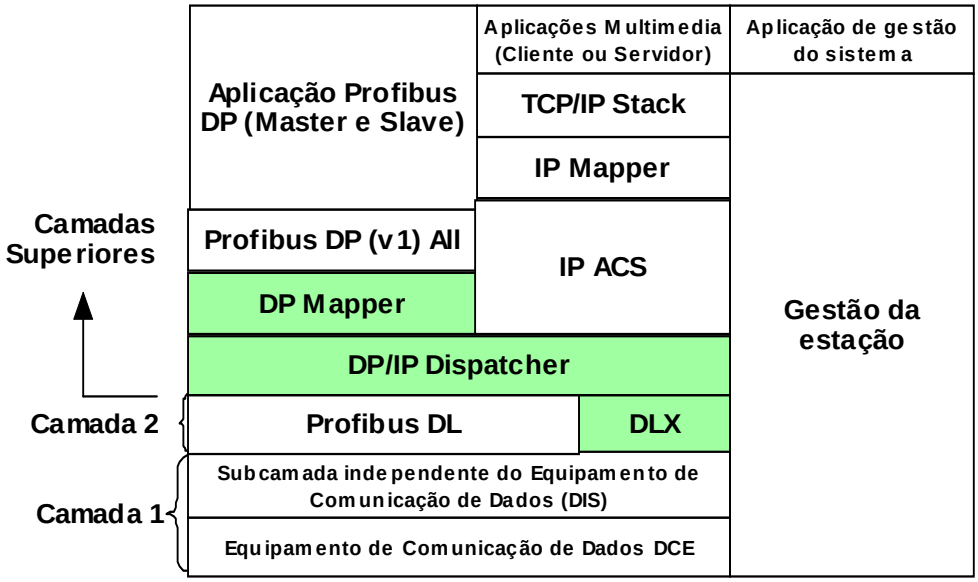


Figura 37 - Arquitectura do protocolo de comunicações para estações RFieldbus (ES)

3.5.1 Formato do PDU da camada física *Wireless*

A estrutura geral do *Protocol Data Unit* (PDU) da camada física rádio é mostrada na Figura 38. É constituído por um preâmbulo gerado pela camada DCE (*Data Communication Equipment*) que é utilizado para efeito de sincronização, um cabeçalho gerado pela camada DIS (DCE *Independent Sublayer*) que transporta a informação relativa ao protocolo PhL RFieldbus e os dados a serem transmitidos entre as *peer* MAC[11] *entities* [27].

DCE Preamble	DIS Header	MPDU (User Data)
54 or 144 usec	72 or 120 Bits	0 - 16376 Bits

Figura 38 - Protocol Data Unit (PDU) da camada física rádio

Considerando um DCE *Preamble* de 54 μS, que a uma taxa de transferência de dados de 2 Mbps usada no *Manufacturing Automation Field Trial* correspondem a 108 *bt* (*bit time*) e um DIS *Header* de 120 bit, temos um cabeçalho de aproximadamente 200 bit para cada *Wireless* PDU [12].

3.5.2 Physical Layer

Especificação do Protocolo da camada PHY (*Wireless*)

Como já referido anteriormente a estrutura protocolar (Figura 37) do RFieldbus suporta dois tipos de meio físico. O par entrançado/RS-485 e a comunicação sem fios/MAC-*less* 802.11b (usando *Direct Sequence Spread Spectrum*). A descrição da camada física do PROFIBUS já foi feita no capítulo anterior. Neste capítulo apenas será descrita a camada física para o meio físico sem fios.

A camada física *Wireless* suporta todas as funções necessárias para transmitir e receber dados via rádio de uma forma eficaz e segura. Esta encontra-se dividida em duas sub camadas, uma relativa

[11] Medium Access Control

ao equipamento de comunicação de dados DCE e outra independente do equipamento de comunicação de dados [27].

Sub camada do equipamento de comunicação de dados (DCE)

Esta camada compreende funcionalidades distintas que podem ser agrupadas em duas sub-camadas. A Ph MAU e a Ph MDS.

A Ph MAU (*Medium Attachment Unit*) é a sub-camada cuja função é fazer a ligação ao meio físico sem fios. Suporta as seguintes funcionalidades: transmissão dos sinais para o meio físico sem fios, o dispositivo de interface com o meio físico sem fios (a antena), a modulação analógica e digital dos sinais, suporte da comutação RX/TX para uma gama de frequências, o controlo da potencia de transmissão e da sensibilidade de recepção.

A Ph MDS (*Medium Dependent Sublayer*) é a sub-camada cuja função é controlar a conexão rádio, a qualidade do sinal rádio, o *channel assessment* (principalmente para a mobilidade inter-célula), uso de diferentes frequências e códigos de canal, encriptação do PDU (quando utilizada) [27].

Sub camada independente do equipamento de comunicação de dados (DIS)

Esta sub-camada DCE (*Data Communication Equipment*) *Independent Sublayer* (DIS), adapta o serviço disponibilizado pela camada Ph MDS ao serviço definido para a camada 1 na norma EN 50170 [6]. Esta interface está descrita em 3.5.2.

Definição do Interface da camada PHY

A interface na camada física PHY é a interface da sub camada DCE DIS [28]. Como já visto anteriormente este interface é o definido pela norma EN 50170 [22]. Ou seja a interface que esta camada oferece é a mesma da camada física do PROFIBUS.

3.5.3 Data Link Layer

Definição do Interface da camada FDL

Sendo a camada FDL do RFieldbus a mesma do PROFIBUS [29], a interface disponibilizada por esta camada é o definido pela norma EN 50170 [19]. Ou seja a interface que esta camada oferece é a mesma da arquitectura PROFIBUS. Esta interface já foi descrita em 2.4.1.

A sub-camada DLX implementa funções associadas a configuração e à mobilidade de dispositivos RFieldbus *Wireless* a sua interface é com a camada adjacente *Station Management* (SM) [29], como se pode ver na figura seguinte.

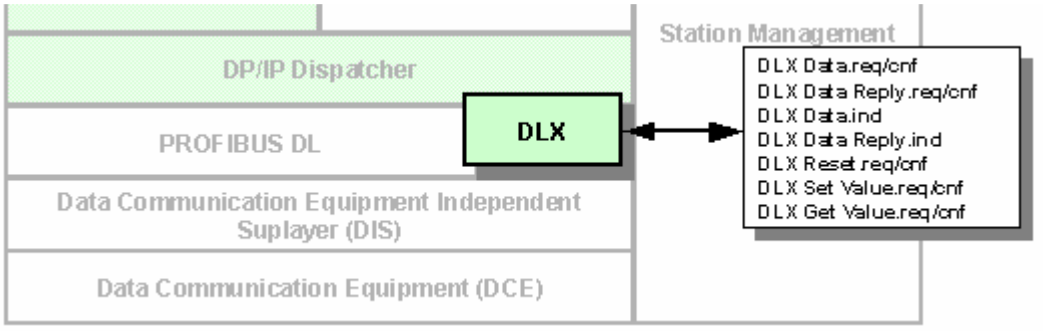


Figura 39 - Interface da camada DLX

As primitivas de serviço e seus parâmetros são comparáveis aos *management services* definidos na PROFIBUS DL [19]. As suas funcionalidades permitem a inicialização, supervisão e detecção de erros nas operações de configuração e da gestão da mobilidade de dispositivos RFieldbus *wireless*.

Os dados dos *end users* não passam por esta camada, passando exclusivamente pela camada FDL definida na EN 50170 [19].

Especificação do Protocolo da camada FDL

Como já referido anteriormente a arquitectura de rede de comunicação RFieldbus deriva do PROFIBUS, sendo uma extensão dessa arquitectura de rede. Ao nível da camada 2 FDL na arquitectura RFieldbus temos a FDL do PROFIBUS e uma sub-camada anexada DLX contendo as extensões ao protocolo da camada 2. Estas extensões são necessárias para suportar a configuração e a mobilidade de dispositivos RFieldbus *Wireless*.

Figura 40 - Camada 2 (FDL) na arquitectura RFieldbus

O protocolo da camada 2 FDL do PROFIBUS já foi descrito no capítulo anterior. Neste capítulo que pretende descrever o RFieldbus serão descritas apenas as extensões ao protocolo, em cada camada. No caso da FDL será descrita a camada introduzida DLX.

Protocolo da camada DLX

No protocolo da camada DLX estão definidas as operações a realizar para implementar os serviços definidos pela camada DLX à camada *Station Management* SM. Serviços esses, que estão associados à configuração e à mobilidade de dispositivos RFieldbus *Wireless* [30], logo não relacionados com os serviços definidos pela PROFIBUS DL [19] que são os serviços que suportam a transmissão de dados. No âmbito do estudo do comportamento temporal das camadas inferiores do

RFieldbus o modelo definido para simulação da rede de comunicações não engloba serviços de gestão.

3.5.4 Camadas Superiores

O estudo apresentado nesta dissertação incide nas camadas inferiores da rede de comunicação RFieldbus, daí a descrição extensiva dessas duas camadas. No entanto para a compreensão da arquitectura RFieldbus é conveniente uma síntese das restantes camadas, o que será feito de seguida.

A camada PROFIBUS DP encontra-se já descrita na página 30 desta dissertação.

DP/IP *Dispatcher*

Esta camada situa-se entre o DP mapper e o IP ACS de um lado, e a camada PROFIBUS DLL do outro (Figura 37).

Na direcção de transmissão são utilizadas cinco filas de mensagens: DP *high priority requests*, DP *low priority requests*, IP *high priority requests* (tráfego IP com qualidade de serviço), DP *best effort requests* e IP *best effort requests* [31]. A camada DP/IP *Dispatcher* transfere os *requests* destas cinco filas de mensagens para as duas filas de mensagens da DLL, com um limite nos ciclos de mensagem, imposto pelo tempo de alocação do *master* t_{MA}. Este é o tempo de alocação calculado para cada *master*. O seu valor é obtido através da soma dos limites temporais de transmissão de cada uma das filas.

$$t_{MA} = t_{DPH} + t_{DPL} + t_{IPH} + t_{BE}$$

Equação 2

Onde:

- t_{DPH} – tempo limite de transmissão da fila do tráfego DP de alta prioridade (H)
- t_{DPL} - tempo limite de transmissão da fila do tráfego DP de baixa prioridade (L)
- t_{IPH} - tempo limite de transmissão da fila do tráfego IP de alta prioridade (H)
- t_{BE} - tempo limite de transmissão da fila do tráfego BE (DP e IP)

Os *requests* são transferidos pelo menos dentro do tempo de ciclo do *dispatcher* t_{DCY}, que corresponde à soma dos t_{MA} de todas as estações *master* da rede e dos tempos de passagem do token. O número de *requests* transferidos de cada fila é determinado pelo tempo limite de transmissão de cada fila (t_{DPH},t_{DPL},t_{IPH},t_{BE}) e pelo tempo alvo para a duração de cada ciclo de mensagem (t_{TMC}) [31]. Este tempo representa o pior caso para a duração do ciclo da mensagem (contando com o *request*, a *response* e o *idle time*).

Na direcção de recepção a camada DP/IP *Dispatcher* passa as confirmações e indicações emitidas pela camada PROFIBUS FDL às camadas PROFIBUS DP e IP ACS respectivamente,

procedendo à identificação do tráfego para este efeito [31]. A título de exemplo na figura seguinte são mostradas as cinco filas de mensagens num RFieldbus *master* classe 2.

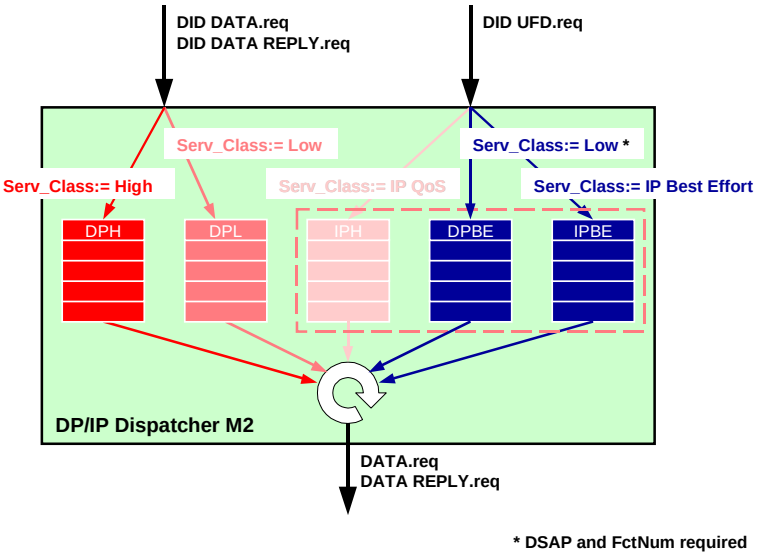


Figura 41 - Filas de mensagens num RFieldbus Master classe 2

A atribuição de tráfego entre filas da camada DP/IP *Dispatcher* e da camada RFieldbus FDL é a seguinte: O tráfego proveniente da fila DPH da camada DP/IP *Dispatcher* é colocado na fila das mensagens de alta prioridade da FDL. O tráfego das restantes filas do DP/IP *Dispatcher* será depositado na fila das mensagens de baixa prioridade da FDL [31].

DP *Mapper*

O DP *Mapper* está situado entre a camada PROFIBUS-DP e a camada DP/IP *Dispatcher* (Figura 37). À camada PROFIBUS-DP, o DP *Mapper* oferece a interface da camada FDL, com a mesma qualidade de serviço, tendo no entanto de reservar alguma largura de banda para o tráfego IP de alta prioridade [31]. Por este motivo o DP *Mapper* tem três funções principais:

- Identificar os *requests* PROFIBUS DP
- Determinar o tempo alvo de cada ciclo de mensagem (t_{TMC})
- Guardar os *requests* DP nas filas de mensagem do DP/IP *Dispatcher*.

A identificação dos *requests* é feita de acordo com as tabelas presentes em [29]. Cada *request* será classificado como tráfego DP_H, DP_L e DP_{BE}. De seguida é calculado o t_{TMC}, que como referido anteriormente é o tempo alvo de cada ciclo de mensagem e representa o pior caso para a duração desse ciclo de mensagem (contando com o *request*, a *response* e o *idle time*). Finalmente o *resquest* é inserido na fila correspondente [29].

IP ACS

Esta camada pertence à parte protocolar que suporta a integração do tráfego multimédia no RFieldbus (Figura 37). O IP ACS reside directamente sob a camada IP *Mapper*. A sua responsabilidade é controlar/limitar a utilização dos recursos de rede pelas aplicações TCP/IP [31]. Para isso utiliza políticas de escalonamento de tráfego capazes de distinguir o tráfego gerado pelas diferentes aplicações TCP/IP. Estas políticas de escalonamento de tráfego devem fornecer a qualidade de serviço desejada às aplicações multimédia, e simultaneamente garantir os requisitos temporais do tráfego de controlo.

Na figura seguinte está esquematizada a constituição da camada IPACS:

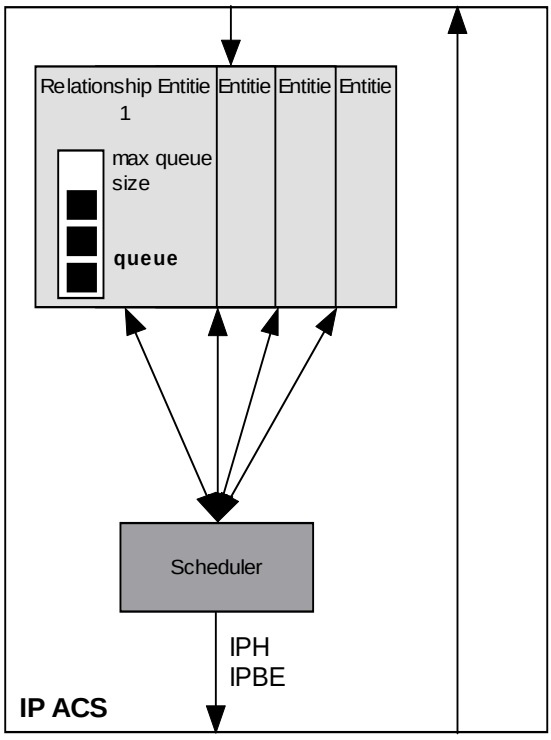


Figura 42 - Esquema funcional da camada IP ACS

A camada de IP ACS é constituída por *Relationship Entities*. Cada uma destas entidades refere-se a uma relação de comunicação IP identificada pelo IP *Mapper* e tem atribuída determinada qualidade de serviço.

Cada *Relationship Entity* contém uma fila (FIFO) que se destina a receber fragmentos IP de uma c conexão específica. Cada fila tem um limite para o número máximo de fragmentos IP a inserir. Atingido este limite, os fragmentos IP que chegarem serão descartados.

Outra entidade, a *Scheduler Entity* é responsável pelo escalonamento dos pedidos contidos nas filas das *Relationship Entities*, para que a qualidade de serviço das aplicações associadas a cada *Relationship Entity* seja cumprida [31].

Cada *Relationship Entity* aceita *requests* do escalonador para entregar tráfego à camada inferior DP/IP *Dispathcher*. Quando a *Relationship Entity* entrega com sucesso um fragmento ao

DP/IP *Dispatcher* confirma ao IP *Mapper*. A *Relationship Entity* fornece ainda ao escalonador *network cost information* acerca dos fragmentos contidos na sua fila.

O escalonador utiliza a interface com o DP/IP *Dispatcher* para determinar se pode depositar o tráfego IP_H e IP_{BE} nas respectivas filas. O escalonador é executado periodicamente, segundo um algoritmo de escalonamento que pode ser definido em *off-line* ou *on-line* [31].

IP *Mapper*

Esta camada pertence à parte protocolar que suporta a integração do tráfego multimédia no RFieldbus (Figura 37). A sua responsabilidade é converter os pacotes IP para/de tramas RFieldbus DLL, bem como suportar de uma forma transparente as relações *peer-to-peer* inerentes ao protocolo IP (em RFieldbus é utilizado o esquema *Master/Slave*) [31].

Concretamente esta camada implementa a correspondência dos serviços TCP/IP em serviços RFieldbus DLL, faz a identificação, fragmentação e posterior reagrupamento dos pacotes IP de/para as tramas DLL. Para cada uma dessas funcionalidades está definida uma entidade.

- *Fragmentation Entity,*
- *Reassembly Entity,*
- *Identification and Routing Entity,*
- *Switching Entity and*
- *ID Generation Entity.*

A figura seguinte esquematiza a estrutura do IP Mapper.

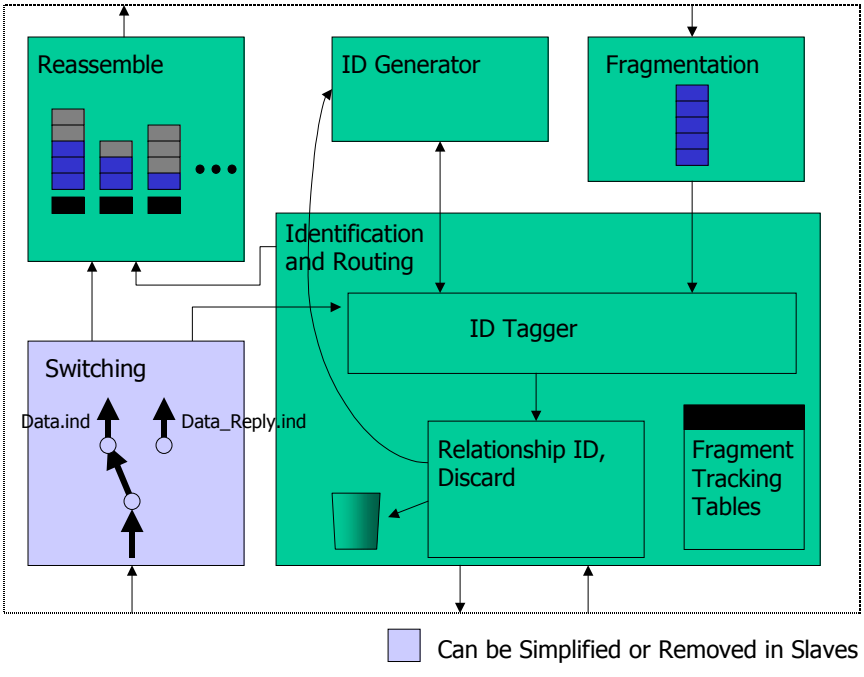


Figura 43 - Estrutura do IP *Mapper*

A *Fragmentation entity* é responsável pela fragmentação dos datagramas IP recebidos da camada superior TCP/IP *stack*. A *Reassembly entity* reconstrói os datagramas IP recebidos a partir dos fragmentos recebidos. A *Identification and Routing Entity* é responsável por identificar o tráfego IP e por proceder ao *routing* do fragmento recebido caso não se destine à estação local.

A responsabilidade da ID *Generation entity* é a gestão da identificação das tramas utilizada para identificar o datagrama IP. Finalmente a *switching entity* que recebe os fragmentos dos pacotes IP vindos de outras estações e os passa para a *Identification and Routing Entity* ou para a *Reassembly entity* [31].

3.6 Sumário

Neste capítulo foi dada uma visão geral da rede de comunicação RFieldbus e detalhados os conceitos mais relevantes utilizados no desenvolvimento do modelo de simulação implementado no LLRS (*Lower Layer* RFieldbus *Simulator*).

Assim após uma apresentação geral do RFieldbus, foram descritas as novas estações introduzidas que permitem o suporte das funcionalidades adicionais de comunicação sem fios, mobilidade de nós e integração de tráfego multimédia.

De seguida foram explicados os diferentes tipos de domínios comunicação da rede, com e sem fios e as apresentadas as topologias base.

Foi também apresentada a solução de gestão da mobilidade utilizada no RFieldbus e posteriormente alguns exemplos de redes estruturadas. Finalmente foi descrita a arquitectura da rede RFieldbus com a especial detalhe nos protocolos e interfaces das camadas inferiores.

Capítulo 4

Simulação por OMNeT++

4.1 Introdução

O OMNeT++ (*Objective Modular Network Testbed in C++*) é um ambiente de simulação de eventos discretos *open source*, baseado em componentes, modular, multi-arquitectura, com *kernel* de simulação embebido e um completo suporte GUI. Este foi o ambiente de simulação de eventos discretos escolhido para implementar o LLRS (Lower *Layer* RFieldbus Simulator).

A sua principal área de aplicação é a simulação de redes de comunicação, no entanto pode ser utilizado para qualquer sistema para o qual seja possível desenvolver um modelo de eventos discretos [32].

Foi desenvolvido para a simulação de tráfego, protocolos de comunicação e filas em redes de telecomunicações, mas devido à sua arquitectura genérica e flexível tem sido utilizado com sucesso em áreas como os sistemas distribuídos, a validação de arquitecturas de hardware, a avaliação da performance de sistemas complexos de software e em processos de gestão.

Porque é *freeware* para a comunidade académica e restantes *non profit users*, tem adquirido grande popularidade junto desta. Mas o OMNeT++ é um ambiente de simulação de eventos discretos entre muitos. Existem ambientes de simulação *freeware* e comerciais, para utilização genérica ou em áreas específicas.

O mais representativo dentro dos ambientes de simulação comercializados é o OPNET. Dentro dos ambientes de simulação não comerciais os mais representativos são PARSEC, ns-2, Ptolemy, C++SIM, *CLASS*.

Existem diversos pontos importantes por onde comparar as funcionalidades oferecidas pelos diferentes ambientes de simulação.

Um desses pontos é a possibilidade ou não, do utilizador implementar novos módulos (que podem servir de *building blocks* para novos módulos) como em OMNeT++, ou pelo contrário está confinado a um conjunto de módulos fornecidos com a ferramenta.

Ferramentas de simulação especializadas, como o ns-2 (dedicada a simulação de redes IP) e *CLASS* (dedicada a simulação de redes ATM), também não se enquadram nesta categoria dado que permitem estender as funcionalidades dos módulos existentes (orientados à simulação de redes especificas a que se destinam).

O OPNET representa o estado da arte em simulação de redes de comunicação. Tal como o OMNeT++ é um ambiente de simulação muito completo mas a sua maior vantagem é ter disponível para compra, um grande número de modelos completos, testados e Validados. Aliás, neste ponto é usual os simuladores não comerciais não poderem competir com os comerciais. No entanto, o OMNeT++ (*freeware*) tem a vantagem de não ter componentes escondidos, permitindo ao utilizador/programador uma total percepção de todo o simulador. Existem em [15] alguns modelos disponíveis para utilização, fruto do contributo de utilizadores da ferramenta de simulação.

4.2 Simulação de Sistemas de Eventos Discretos

Antes de descrever o ambiente de simulação OMNeT++ convém fazer uma breve descrição sobre simulação de sistemas de eventos discretos.

Para se simular um sistema é necessário estabelecer um modelo descritivo do comportamento desse sistema (real ou proposto) para que possa ser estudado em condições específicas [33]. Um dos aspectos mais importantes da simulação é permitir modelar comportamentos dos sistemas ao longo do tempo.

Existem sistemas cujo estado varia de uma forma *contínua* com o tempo. São chamados *Continuous time systems*. Noutros, o estado varia apenas em alguns instantes de tempo. São os *Discrete time systems* [34].

4.2.1 Princípios de simulação de Sistemas de Eventos Discretos (DES)

Num modelo de simulação, existem entidades e relações lógicas (*logic statements*) [33]. As entidades são elementos tangíveis e podem ser temporárias ou permanentes. No caso de uma rede de comunicação de dados, um exemplo de uma entidade permanente é uma estação da rede. Um exemplo de uma entidade temporária é uma mensagem transmitida nessa rede de comunicação.

As relações lógicas estabelecem relações entre as entidades (ex. quando uma estação vai emitir uma mensagem para a rede de comunicação). As relações lógicas são uma parte fundamental do modelo de simulação, pois definem o comportamento global do modelo. Cada relação lógica é simples, no entanto a quantidade e o facto de estarem dispersas ao longo do modelo pode aumentar a complexidade.

Um elemento essencial de um sistema de simulação é o executivo de simulação. Este é responsável por controlar o avanço do tempo. O executivo de simulação controla as relações lógicas entre as entidades e o avanço do tempo. Executivo e relógio [33] são componentes chave de um sistema de simulação e o seu comportamento é relativamente simples de implementar.

Um outro elemento é o gerador de números aleatórios que é utilizado para modelar comportamento estocástico no modelo (ex. numa rede de comunicação o tempo de resposta de uma estação está contido num determinado intervalo e pode ser determinado por uma distribuição de probabilidade).

Finalmente existe o elemento de recolha de resultados (e visualização normalmente). Este permite retirar da simulação resultados úteis para a análise do sistema.

4.2.2 Mecanismos para o avanço do tempo

Como referido anteriormente o elemento central de um sistema de simulação é o executivo de simulação. Este gere a passagem do tempo incrementando-o e executando as relações lógicas relevantes ao longo deste. Existem duas abordagens [33] na forma como o executivo faz avançar o tempo de simulação:

- *Time Slicing*
- *Next Event*

Na abordagem por *Time Slicing* o tempo avança em intervalos fixos (*Slice time*). Neste caso o executivo de simulação faz avançar o tempo de simulação de um intervalo de tempo constante, mesmo que nada aconteça no sistema. No caso de um *Slice time* de 1 ms, o tempo de simulação avança 1 ms de cada vez.

Na abordagem por *Next Event* o tempo avança para o instante em que ocorre o próximo evento significante no sistema. Caso não aconteça nada no sistema no próximo segundo o executivo de simulação adianta o tempo de simulação um segundo.

Naturalmente que esta ultima aproximação é mais eficiente e permite a avaliação dos modelos de uma forma mais rápida. O poder computacional é utilizado de uma forma mais eficiente.

4.2.3 Abordagens na descrição da lógica

Existem diferentes formas de representar as relações lógicas num modelo de simulação de eventos discretos. Diferentes abordagens podem ser utilizadas para descrever o mesmo sistema e conduzem aos mesmos resultados.

As diferenças entre abordagens residem na facilidade de implementação, de compreensão do modelo e na eficiência computacional.

Três abordagens a descrição da lógica [33]:

- Eventos
- Actividades
- Processos

A abordagem por eventos descreve um evento como uma mudança instantânea. Tais mudanças ocorrem normalmente aos pares (ex. iniciar uma transmissão no meio físico e terminar essa transmissão). Na abordagem por actividade é descrita uma duração (ex. duração da transmissão no meio físico) e é bastante similar a pares de eventos. A abordagem por processos conglomera um conjunto de eventos e/ou actividades que descrevem o ciclo de operação de uma entidade[12].

A abordagem por eventos é facilmente compreensível e computacionalmente eficiente, mas é mais difícil de implementar do que a abordagem por actividades. A abordagem por actividades também é fácil de perceber, mas computacionalmente não é tão eficiente. Já a abordagem por processos requer mais planeamento para ser implementada de forma correcta mas é considerada computacionalmente eficiente.

[12] - A abordagem por processos aqui descrita é designada em OMNeT++ por *activity*.

4.3 Descrição do ambiente de simulação

4.3.1 Características gerais

As principais características deste ambiente de simulação são a possibilidade do utilizador implementar novos módulos (ou seja não está limitado a um número pré definido de blocos), geração da topologia por software, não existir um limite para o *module nesting* (encapsulamento de módulos dentro de outros módulos), a possibilidade de descrever novas topologias através da linguagem textual NED [32] manualmente, automaticamente ou com o editor gráfico, a possibilidade de implementar os modelos em programação baseada em *thread/coroutine* utilizando *activity()*, ou máquinas de estado finitas utilizando *handleMessage()* no OMNeT++ [35].

4.3.2 Módulos simples e compostos

Existem dois tipos de módulos: os módulos simples e os módulos compostos. No fundo da hierarquia estão os módulos simples. São aqueles em que os algoritmos do modelo de simulação de eventos discretos são implementados, (os eventos ocorrem dentro dos módulos simples: a sua geração e a reacção a eles) pelo utilizador na linguagem de programação orientada a objectos C++ [35]. No caso de um módulo simples é necessário declarar em linguagem NED o módulo como módulo simples, definir os seus parâmetros e *gates*. A programação dos algoritmos faz-se num ficheiro *.cpp, com a redefinição de algumas funções membro.

Os módulos compostos são mais fáceis de criar pois apenas é necessário descrever a sua estrutura. Para isso, há que declarar em linguagem NED os módulos que vão ser encapsulados dentro deste, definir os parâmetros e gates do módulo composto e estabelecer as conexões entre as *gates* dos módulos internos e entre estas e as do módulo composto [32].

4.3.3 Hierarquia de módulos,

Um modelo de simulação em OMNeT++ consiste numa hierarquia de módulos encapsulados. Isto permite ao utilizador reflectir a estrutura lógica do sistema que está a modelar na estrutura do modelo.

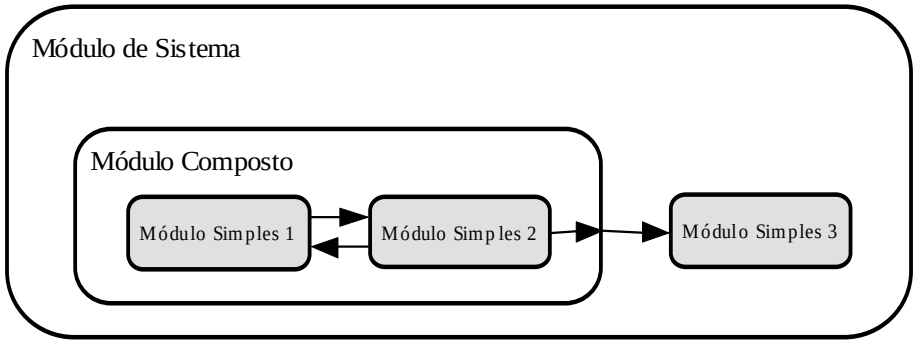


Figura 44 - Hierarquia de módulos em OMNeT++

O simulador é constituído por uma hierarquia de módulos que vão sendo encapsulados dentro de outros módulos (como blocos construtivos de novos módulos). Como já referido, existem dois tipos de módulos, simples e compostos. Os módulos simples são os que estão no nível mais baixo dessa hierarquia e os que contém os algoritmos que representam o modelo de eventos discretos definido para esse módulo.
É nestes módulos que todas as acções decorrem porque são eles que geram e reagem aos eventos.

Os módulos simples em conjunto com outros módulos simples ou módulos compostos podem ser utilizados para desenvolver novos módulos compostos, que por sua vez podem ser utilizados para compor novos módulos compostos, numa sequência sem limite para o número de encapsulamentos de módulos (*module nesting*). A cada novo módulo composto apenas é necessário definir a sua estrutura através da linguagem NED.

Módulos compostos podem representar os elementos da rede de comunicações, sendo estes incluímos num módulo que representa uma topologia da rede de comunicações a simular, até atingirmos o *system module* que é o *top level module* da hierarquia do simulador [32]. Todos os módulos da hierarquia do simulador, são instanciados como sub-módulos ou sub sub-módulos do *system module*.

Existem duas características muito interessantes nesta plataforma de simulação que importa realçar: uma já foi referida anteriormente é o *infinte module nesting* [32]. A outra é não existir qualquer distinção entre módulo simples ou composto no encapsulamento de módulos ou seja é idêntico encapsular um módulo simples ou composto dentro de um novo módulo composto [32]. Ambas, permitem ao utilizador, flexibilidade no desenvolvimento dos módulos, pois é possível substituir um módulo simples por vários encapsulados num módulo composto, ou vice-versa.

Devido ao design orientado ao objecto do OMNeT++, o utilizador através dos conceitos de OO (herança e polimorfismo) pode estender funcionalidades do ambiente de simulação.

O simulador gerado consiste num ficheiro executável. O OMNeT++ integra poderosas capacidades de verificação de erros (*debugging)* e traçagem *(tracing)* no simulador permitindo ao utilizador observar/controlar totalmente o decurso da simulação (todos os objectos criados pelo utilizador são observáveis e modificáveis através do GUI).

4.3.4 Mensagens, *Gates* e Conexões

Os módulos comunicam trocando mensagens através de conexões estabelecidas entre as *gates* de módulos pertencentes ao mesmo nível da hierarquia. Essas mensagens representam normalmente

tramas da rede de comunicação, mas podem conter de apenas uma *flag* a estruturas de dados arbitrariamente complexas.

O tempo local de simulação de um módulo avança quando este recebe uma mensagem. A mensagem pode ser enviada por outro módulo ou pelo próprio módulo (*self messages* podem ser utilizadas para implementar temporizadores).

As gates funcionam como interfaces dos módulos. Uma mensagem é enviada por uma gate de saída de um módulo e é recebida numa *gate* de entrada.

Devido à estrutura do modelo, as mensagens tipicamente atravessam uma série de conexões, desde um módulo simples até outro. Esse conjunto de conexões, que ligam módulos simples, são chamadas rotas.

4.3.5 Parâmetros

Os módulos podem ter parâmetros por três motivos principais: para definir o comportamento do módulo, para criar topologias flexíveis (onde os parâmetros permitem especificar o numero de módulos, a estrutura das conexões, etc.), e para a comunicação entre módulos através do uso de variáveis partilhadas [32].

4.4 Módulos simples

4.4.1 Modo de funcionamento

Nos módulos simples, além da definição do módulo através da linguagem NED é necessário programar o modelo de simulação em C++. Para isso, o utilizador cria uma classe derivada da classe *cSimpleModule* [35], (pertencente à biblioteca de classes do OMNeT++), herdando desta ultima as funcionalidades relacionadas com a simulação, mas à qual tem que ser adicionada a funcionalidade do modelo de eventos discretos.

Isto é feito, redefinindo algumas funções membro virtuais, para que nelas se implementem os algoritmos. A função membro *initialize()*, é utilizada para construir a rede (*Network*). Cria os módulos simples e compostos necessários, cria as ligações entre eles e chama as funções *initialize()* de cada módulo. A função membro *finish()* é chamada automaticamente quando a simulação termina com sucesso. É normalmente utilizada, para gravar as estatísticas recolhidas durante a simulação. Para a geração e processamento dos eventos, existem duas funções, *activity()* e *handleMessage()* que representam paradigmas diferentes na forma de processar os eventos [35].

4.4.2 Paradigmas de programação

As funções membro *activity()* e *handleMessage(cMessage *msg)* são chamadas durante o processamento dos eventos [32]. É dentro de ambas que é implementado o modelo de simulação. *Activity* é uma abordagem de descrição da lógica por actividades enquanto *handleMessage* é por eventos.

Para cada módulo simples o utilizador tem que definir qual delas pretende utilizar. No entanto módulos implementados com *activity()* ou *handleMessage()* podem ser utilizados no mesmo simulador (na mesma hierarquia de módulos).

activity() é uma solução baseada em corrotinas (*threads* cooperativas não *preenptivas*), na qual o código é desenvolvido como em qualquer processo ou *threads* de um sistema operativo. Esta solução torna-se muito interessante porque permite uma descrição intuitiva do algoritmo a implementar e facilita a implementação em diagramas de transição de estados e em redes de *Petri*.

handleMessage() é uma função que é chamada cada vez que uma mensagem chega ao módulo. A função processa a mensagem e retorna logo de seguida. O tempo de simulação (diferente do tempo real que dura a simulação) é potencialmente diferente a cada chamada da função. Não decorre tempo de simulação durante a execução da função *handleMessage()*. A cada evento, chama-se a função *handleMessage()* definida pelo utilizador. Cada evento, pode adicionar ou remover eventos da FES (*Future Events Set*).

4.4.3 Funções membro virtuais mais utilizadas

Dentro destas funções, é criado o comportamento dos *módulos simples* de forma a animar a simulação. Cada uma delas tem um conjunto de funções que são utilizadas neste processo. De seguida serão descritas as funções mais utilizadas quer em *activity()* quer em *handleMessage()* [35].

As funções mais utilizadas em *activity()* são:

- *receive()* para receber mensagens
- *wait()* para suspender a execução durante algum tempo (modelar tempo)
- *send()* para enviar mensagens para outros módulos
- *scheduleAt()* para escalonar um evento (o módulo envia uma mensagem para si próprio)
- *cancelEvent()* para cancelar um evento escalonado com *scheduleAt()* e *end()* para finalizar a execução deste módulo (idêntico a terminar a função *activity()*).

As funções mais utilizadas em *handleMessage()* são:

- *send()* para enviar mensagens para outros módulos
- *scheduleAt()* para escalonar um evento (o módulo envia uma mensagem para si próprio)
- *cancelEvent()* para cancelar um evento escalonado com *scheduleAt()*.

4.5 Desenvolvimento e execução (*Build* and *Run*) do simulador

Na Figura 45, é esquematizada a sequência de desenvolvimento e execução de um simulador [32].

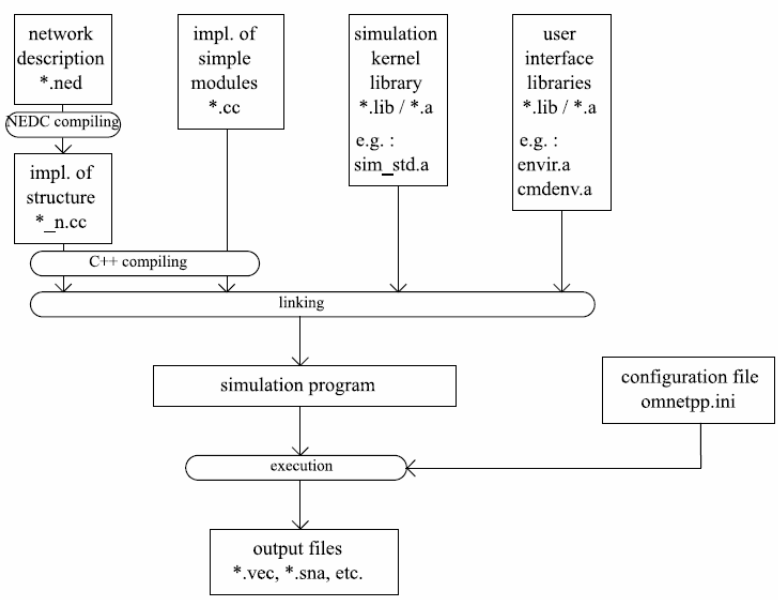


Figura 45 – *Building* and *Running* a simulation

Como se pode ver na Figura 45 os módulos definidos nos ficheiros <NOME>.NED são compilados com o *NEDC compiler* sendo guardados os resultados em ficheiros <NOME>_n.cpp. Os ficheiros resultantes do processo anterior e os ficheiros de implementação dos módulos simples são compilados e integrados com o *kernel* de simulação e as bibliotecas de interface com o utilizador, resultando no ficheiro executável que constitui o simulador. Este ficheiro pode ser executado numa máquina onde o OMNeT++ não está instalado e sem a necessidade dos ficheiros do modelo, dado que estes foram incluídos no executável. Ao correr esse executável, ele busca num ficheiro *.ini, (normalmente OMNeTpp.ini) os valores a atribuir aos parâmetros simulando o sistema e produzindo os resultados da simulação. Estes resultados são escritos em ficheiros de saída.

4.6 Sumário

Este capítulo pretendeu fazer uma descrição dos principais conceitos da simulação de sistemas de eventos discretos e da ferramenta de simulação de eventos discretos (OMNeT++), utilizada para implementar o LLRS.

Para isso foram apresentados os elementos constituintes de um simulador de sistemas de eventos discretos (relações lógicas, executivo de simulação, gerador de números aleatórios e elemento de recolha de resultados), os mecanismos para o avanço do tempo (*Time Slicing* e *Next Event*) e as diferentes abordagens na descrição da lógica (Eventos, Actividades e Processos)

Foram também apresentadas as principais características do OMNeT++, áreas de aplicação, a posição que ocupa no espectro das ferramentas computacionais de simulação de sistemas de eventos discretos, quer face às ferramentas *freeware* (nas quais o OMNeT++ se inclui), quer face a

ferramentas comerciais (categoria na qual se inclui o *software* de referência OPNET que representa o estado da arte em simulação de redes de comunicação).

De seguida foi feita uma descrição da ferramenta de simulação, tipo de módulos (simples e compostos), definição de uma hierarquia de módulos, comunicação por mensagens, gates conexões e parâmetros.

Finalmente foi dada uma especial atenção aos modos de funcionamento dos módulos simples. Foram apresentados os dois paradigmas de programação (*activity* e *handleMessage*) e descritas resumidamente as funções membro virtuais mais utilizadas em cada paradigma de programação.

Capítulo 5

Modelo de simulação definido para cada componente RFieldbus

5.1 Introdução

Neste capítulo, é apresentado o modelo definido para simular o comportamento temporal de cada componente RFieldbus. De seguida (Tabela 3) são apresentados os componentes Rfieldbus modelados (referidos em 3.1.1), bem como esquematizada a sua hierarquia:

RFieldbus *Master* RFieldbus *Slave* RFieldbus *Mobility Master* RFieldbus *Wired Domain* RFieldbus *Wireless Domain* RFieldbus *Wired Physical media* RFieldbus *Wireless Physical media*	
RFieldbus *Intermediate Systems*	*Link Station*
	Link Base Station
	Base Station

Tabela 3 - Componentes RFieldbus

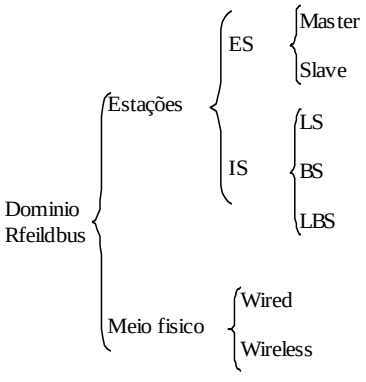


Figura 46- Hierarquia dos componentes Rfieldbus

A Figura 46 esquematiza a hierarquia dos componentes Rfieldbus. Um domínio Rfieldbus é constituído por Estações e meio(s) físico(s). Por sua vez um meio físico pode ser do tipo com fios (wired) ou sem fios (wireless). Já a estação pode ser do tipo ES (End System) ou IS (Intermediate System). Por sua vez uma estação ES pode ser do tipo Master ou Slave e uma estação IS pode ser do tipo LS (Link Station), BS (Base Station) ou LBS (Link Base Station). Só as entidades que não tem nenhuma chaveta à direita no esquema existem (Master, Slave, LS, BS, LBS, Wired e Wireless) todas que se encontram à esquerda de uma chaveta são entidades abstractas cuja utilidade é permitir referenciar um conjunto de entidades físicas.

5.2 RFieldbus *Master*

A estação *Master* é a estação principal em RFieldbus, dado que é ela que inicia e gere cada ciclo de mensagem. A rede de comunicação RFieldbus é do tipo *broadcast*, para controlar o acesso ao meio (pág. 21), as estações *Master* trocam entre si o *Token*, num *Token-Ring* lógico. A estação que detém o *Token* pode utilizar o meio físico para transmitir ciclos de mensagem através dos quais comunica com outras estações da rede (*Master* ou *Slave*). Caso, apenas um *Master* esteja presente na rede este passa periodicamente o *Token* si mesmo.

A camada de aplicação requisita um dos serviços de transferência de dados da FDL (pág. 13), entregando os dados a transmitir a esta camada. Esses dados serão depositados numa das filas de mensagens existentes na camada FDL. Posteriormente, ao receber o *token* (pág. 21), a estação retira uma mensagem de uma das filas, mediante um algoritmo de escalonamento do tráfego (pág. 24) que hierarquiza a transmissão de diferentes classes de tráfego e transmite-a respeitando o algoritmo de processamento dos ciclos de mensagens (pág. 18). Enquanto tiver tempo disponível ($t_{th} > 0$) a estação repete este procedimento de envio de ciclos de mensagem. Quando o tempo disponível se esgota ($t_{th} \leq 0$) a estação procede à transmissão do *token* para a estação seguinte no *Token-Ring* lógico.

5.2.1 Estados do controlador da FDL

As operações referidas anteriormente e sua sequência são descritas através da máquina de estados do controlador da FDL, que se encontra na página 27 desta dissertação. No modelo de simulação definido, os seguintes estados do controlador da FDL não foram incorporados:

O estado *Offline* não foi incorporado porque representa um estado no qual o *Master* não estabelece qualquer comunicação.

Também não foram incorporados os estados que representam a operação transitória da rede de comunicação, durante os quais é construída a topologia completa da rede. O modelo de simulação foi desenvolvido, considerando a rede em plena operação, não contemplando este período transitório no qual as estações fazem as inicializações e tentam aceder à rede de comunicação. É o caso do estado *Listen Token* no qual o *Master* está a aceder ao *Logical Token-Ring*. Neste estado o *Master* preenche a lista de estações activas (LAS) e posteriormente aguarda ser chamado a entrar no *Logical Token-Ring*.

Também dois estados de erro não foram incluídos: o estado *Claim Token* no qual o controlador da FDL tenta iniciar ou reiniciar o anel lógico e o estado *Await Status Response*.

5.2.2 Classes de tráfego RFieldbus

A arquitectura RFieldbus suporta tráfego de controlo (DP) e tráfego multimédia (IP). Existem cinco classes de tráfego em RFieldbus, três para tráfego DP e duas para tráfego IP.

DP_H	DP *High priority traffic*
DP_L	DP *Low priority traffic*
IP_H	IP *High priority traffic*
DP_{BE}	DP *Best Effort priority traffic*
IP_{BE}	IP *Best Effort priority traffic*

Tabela 4 - Classes de tráfego RFieldbus

5.2.3 Tráfego das camadas superiores para tráfego FDL do RFieldbus

É na sub camada DP/IP *Dispatcher* (Figura 37) que os diferentes tipos de tráfego das camadas superiores do RFieldbus são atribuídos a uma das classes de tráfego da camada FDL.

A sub camada DP/IP *Dispatcher* contém cinco filas, uma por cada classe de tráfego RFieldbus. Assim, existe a DP_H , a DP_L , a IP_H , a DP_{BE} e a IP_{BE} *Queue*. Em cada uma das filas será depositado o tráfego correspondente, através da requisição dos serviços fornecidos pela camada DP/IP *Dispatcher* aos utilizadores. Neste caso os utilizadores são de um lado a pilha protocolar do tráfego DP e do outro a pilha protocolar do tráfego IP.

As mensagens contidas em cada uma das cinco filas do DP/IP *Dispatcher* serão atribuídas a uma das duas filas de mensagens da camada FDL. Na tabela seguinte é mostrada essa correspondência.

DP_H	Hi Priority *message cycle*
DP_L	Low Priority *message cycle*
IP_H	
DP_{BE}	
IP_{BE}	

Tabela 5 - Correspondência do tráfego RFieldbus em tráfego nativo da FDL PROFIBUS

5.2.4 Classes de tráfego FDL implementadas

Conforme descrito na página 24, em RFieldbus ao nível da camada FDL, existem quatro classes de tráfego, que são:

Hi	*Hi priority message cycles*
Low	*Poll list* *Low priority message cycles* *GAP Update message cycles*

Tabela 6 - Classes de tráfego da FDL em PROFIBUS

As classes de tráfego *Poll List* e *GAP Update message cycles* não serão implementadas. O tráfego *Poll List* não foi implementado porque não é utilizado em RFieldbus. Isto deve-se ao facto de

o PROFIBUS DP apenas utilizar dois serviços da FDL: o SRD e SDN [6]. Não utilizando o serviço da FDL, CSRD não utiliza a *Poll List*. Dado que o RFieldbus deriva da arquitectura do PROFIBUS-DP também utiliza apenas estes dois serviços de transferência de dados da FDL. Outra forma de verificar isto está em [27], na tabela de correspondência de tráfego do DP/IP *Dispatcher* do *Master Class* 1 os DSAP's utilizados referem-se aos exclusivamente aos serviços SRD e SDN. As observações feitas ao meio físico com o PROFIBUS *Analyser* permitiram verificar que apenas estes serviços eram requisitados à camada FDL.

Os *GAP Update message cycles* não serão implementados porque neste simulador LLRS, não foi definido o suporte de alterações dinâmicas à topologia. Não foi considerada a possibilidade de estações iniciarem ou terminarem a comunicação com a rede em operação. Sem alterações dinâmicas na rede de comunicação não há necessidade de simular um mecanismo para suportar essas alterações. No sistema real este tráfego existe. No entanto a ocupação do meio físico por estes ciclos de mensagem, não é muito significativa pois estes são os últimos ciclos de mensagem a serem escalonados para transmissão e só ocorrem quando não existe tráfego de mais alta prioridade a ser transmitido.

Serão portanto implementadas, os *Hi priority message cycles* e os *Low priority message cycles* [6].

A camada FDL dispõe de duas filas de mensagens, a *Hi priority message Queue* e a *Low Priority Message Queue* [6]. Cada um destes dois tipos de tráfego implementados será depositado na respectiva fila de mensagens da FDL, ou seja os *Hi priority message cycles* serão alojados na *Hi priority message Queue* enquanto os *Low priority message cycles* serão alojados na *Low Priority Message Queue.*

5.2.5 Escalonamento de tráfego

Dado que apenas estão implementadas duas classes de tráfego, o algoritmo de escalonamento de tráfego é mais simples do que o original apresentado na página 24. O algoritmo de escalonamento implementado é apresentado na Figura 55 e respeita o algoritmo de escalonamento desses dois tipos de tráfego.

A cada recepção do *token*, é sempre possível processar um ciclo de mensagem de alta prioridade. Enquanto existirem ciclos de mensagem de alta prioridade e T_{TH} disponível, estes ciclos de mensagem serão processados um a um sendo verificado o T_{TH} disponível no início de cada ciclo de mensagem. Quando T_{TH} expira, a estação transmite o *token* para a estação seguinte no anel lógico. Caso não existam mais mensagens de alta prioridade e ainda exista T_{TH} disponível o *Master* passa a transmitir as mensagens de baixa prioridade. De igual forma antes de iniciar cada ciclo de mensagem verifica o T_{TH} disponível, caso exista inicia o processamento de mais um ciclo de mensagem de baixa prioridade. Caso T_{TH} expire ou não existam mais mensagens de baixa prioridade, o *Master* transmite o *token* para a estação seguinte no anel lógico.

Como referido anteriormente, este algoritmo é uma versão simplificada do algoritmo *Timed Token* que tem bom desempenho em sistemas de tempo real dado que permite uma utilização de quase 100% da banda disponível.

5.2.6 Processamento de cada ciclo de mensagem

O algoritmo de processamento de cada ciclo de mensagem é idêntico para as mensagens de alta ou de baixa prioridade. No caso dos ciclos de mensagem com resposta faz-se de acordo com o fluxograma apresentado na Figura 10. Os ciclos de mensagem sem resposta são mais simples existindo lugar à transmissão do *request*, aguardando a estação *Master* de seguida T_{ID2} até transmitir o *request* do próximo ciclo de mensagem ou o *token* (Figura 13).

5.2.7 Transmissão do token

O algoritmo de transmissão do *token* em RFieldbus está representado por meio de um fluxograma na Figura 15. Dado que o simulador implementado não suporta alterações dinâmicas da topologia da rede o algoritmo de transmissão do token sofre uma alteração face ao original por este facto. O algoritmo implementado encontra-se descrito na Figura 56.

Como no simulador não foram consideradas alterações dinâmicas da topologia, as estações estarão sempre presentes na rede de comunicação. Desta forma qualquer erro num ciclo de mensagem dever-se-á a um erro de comunicação. O algoritmo implementado considera uma variável *retries* mas apenas para contabilização do número de tentativas de transmissão.

Cada estação *Master* conhece o seu endereço (TS), o endereço da estação anterior (PS) e o endereço da estação seguinte (NS) no anel lógico. A trama do *Token* contém os campos: o delimitador SD4, o endereço do remetente SA e o endereço do destinatário DA (pág. 18). Quando pretende enviar o *Token* a estação coloca o seu endereço TS no campo SA e o endereço NS no campo DA e transmite o *token* (algoritmo na página 22).

5.2.8 Recepção do Token

O algoritmo de recepção do *token* está descrito por meio de um fluxograma na Figura 14. Dado que no LLRS não foram consideradas alterações dinâmicas da topologia, não existe a necessidade de construir a LAS e de proceder às alterações ao anel lógico entre *masters*. O algoritmo de recepção do *token* implementado encontra-se descrito no fluxograma da Figura 54.

5.3 RFieldbus *Slave*

A estação *Slave* apenas transmite quando requisitada por uma estação *Master*. O *Slave*, ao receber um *request* endereçado a si, ao qual deve responder (existem ciclos de mensagem sem resposta) prepara a resposta e logo que possível envia-a. A consequência é o atraso temporal (t_{rt}) que decorre entre o fim da recepção do *request* e o início da transmissão da *response*.

5.4 RFieldbus Mobility *Master*

O procedimento da gestão da mobilidade encontra-se descrito na página 45 desta dissertação.

Em fase de instalação da rede, são calculados os parâmetros utilizados na configuração da mesma, através da aplicação *System Planning Tool*[13].

O que é relevante observar relativamente ao comportamento temporal do procedimento de gestão da mobilidade, é a latência introduzida por este procedimento. Diversos parâmetros intermédios, relativos à duração do procedimento da gestão da mobilidade são calculados mas o parâmetro que assume relevância no âmbito da latência introduzida pelo procedimento é o $T_{ID2MobM}$. Este parâmetro vai ser inserido como TID2 no MobM. Quando o MobM recebe um *Token*, verifica se já decorreu o *Period for Triggering Mobility Management Procedure* (T_{FTMMP}). Caso tenha decorrido, dá início ao procedimento de gestão de mobilidade, enviando o *BT PDU* (*unacknowledged*), aguardando de seguida $T_{ID2MobM}$ até devolver o *Token* ao meio físico endereçado à estação seguinte (NS). Caso não tenha decorrido transmite o *Token* para a estação seguinte (NS).

5.5 RFieldbus Wired/Wireless Domain

Um domínio de comunicação (*Wired* ou *Wireless*) é constituído por um conjunto de estações (*End Systems* e *Intermediate Systems*) e um meio físico (*Wired* ou *Wireless*). O Domínio de comunicação é uma entidade abstracta, que associa as estações ao meio físico. Como entidade abstracta que é, não tem influência no comportamento temporal da rede de comunicação servindo apenas para a organização da mesma.

Um RFieldbus *Wired Domain* tem associado um meio físico com fios e um conjunto de estações finais, enquanto que um RFieldbus *Wireless Domain* tem associado um meio físico sem fios e o respectivo conjunto de estações finais.

5.6 Physical media - Wired/Wireless

Em qualquer meio físico a duração da transmissão depende de dois factores. Da taxa de transmissão de dados e do atraso de propagação dos sinais no meio físico.

No RFieldbus, ambos os meios físicos (*Wired* ou *Wireless*), tem um atraso de propagação que pode ser considerado nulo. Desta forma a duração da transmissão é calculada apenas através da taxa de transmissão de dados.

O cálculo da duração da transmissão de cada PDU depende dos dados a transmitir e da taxa de transmissão de dados. A Equação 3 seguinte mostra a relação:

---

[13] - Ferramenta desenvolvida no âmbito do projecto europeu RFieldbus, que se destina a calcular não só os parâmetros de configuração da rede, mas também o pior caso da duração de cada ciclo de mensagem.

$$C^i = \frac{L^i}{R^i}$$

Equação 3 - Fórmula de cálculo da duração de um PDU no meio físico i

Onde C^i é a duração do PDU no meio físico i, ou seja o tempo de transmissão do PDU nesse meio físico. L^i o comprimento do PDU[14] em (k)*bits*, ou (k)*bytes* e R^i a taxa de transmissão de dados nesse meio físico, em (k)*bits*/s ou (k)*bytes*/s respectivamente.

Um meio físico sem fios (WL) tem uma taxa de transmissão de dados de R^{WL}, que para uma determinada trama[15] de comprimento L^{WL} (*bytes*), resultará numa duração da transmissão de C^{WL} (*bytes/s*).

Um meio físico com fios (WR) tem uma taxa de transmissão de dados de R^{WR}, que para uma determinada trama de comprimento L^{WR} (*bytes*), resultará numa duração da transmissão de C^{WR} (*bytes/s*).

O cálculo do tempo de propagação dos sinais no meio físico depende do comprimento do meio.

$$t^i = \frac{x^i}{v^i}$$

Equação 4 - Cálculo do tempo de propagação dos sinais no meio físico i

Onde t^i é o tempo de propagação do sinal no meio físico *i*, x^i é o comprimento do meio físico *i* e v^i a constante que representa a velocidade de propagação dos sinais no meio físico.

A tabela seguinte demonstra isto, comparando o tempo de propagação em cada meio físico (com e sem fios) com a duração da transmissão da trama mais pequena (situação mais desvantajosa) nesse meio físico:

	L^i	v^i	Ri	t^i	SC
Wired (WR)	L^{WR} = 1200 m	v^{WR} = 200*10⁶ m/s	R^{WR} = 100 kbps	t^{WR} = 6 µs	110 µs
	L^{WR} = 10 m	v^{WR} = 200*10⁶ m/s	R^{WR} = 10 Mbps	t^{WR} = 50 ns	1100 ns
Wireless (WL)	L^{WL} = 200 m	v^{WL} = 200*10⁶ m/s	R^{WL} = 2 Mbps	t^{WL} = 1 µs	804 µs

Tabela 7 - Tempo de propagação e duração da transmissão da trama mais pequena

Em ambos os meios físicos o tempo de propagação dos sinais é muito inferior à duração da trama mais pequena a ser transmitida no meio físico. O tempo de transmissão no meio físico foi calculado apenas pelo tempo de transmissão da trama.

---

[14] - cada byte das tramas da FDL é transmitido para o meio físico num carácter UART (11 bits) (página 13).

[15] - PDU = trama.

5.7 RFieldbus Intermediate Systems

Os *Intermediate Systems* são como o nome indica, sistemas intermédios. A sua função é reenviar as tramas recebidas de um meio físico para outro, para que estas cheguem a todos os sistemas finais (*End Systems*). Cada sistema intermédio fá-lo de uma forma específica de acordo com a sua funcionalidade.

5.7.1 *Link Station*

A função de uma *Link Station* (pág. 38) é interligar meios físicos diferentes. Uma *Link Station* está sempre ligada a dois meios físicos, um *Wired* e outro *Wireless*. Quando recebe um PDU de um dos meios físicos, envia-o para o outro meio físico. Entre o instante em que inicia a recepção do PDU num meio físico, até ao instante em que inicia a retransmissão no outro decorre um período de tempo (t_{rd}).

A implementação da *Link Station* pode conduzir a valores de t_{rd} distintos. Considere-se a título de exemplo uma *Link Station* que no lado 1 está ligada ao meio físico *Wired* e no lado 2 ao meio físico *Wireless* e acontece a recepção de um PDU do lado 1.

Se a *Link Station* for implementada como *Store and Forward*, recebe o PDU completo no lado 1 e de seguida inicia a retransmissão no lado 2. Neste caso o $t_{rd} = L^{1}_{PDU}$.

Se a *Link Station* for implementada como *cut-through*, pode iniciar a retransmissão no lado 2 antes de terminar a recepção do PDU lado 1. No entanto, o instante em que inicia a retransmissão não pode ser arbitrariamente pequeno dado que a transmissão de uma trama num meio físico tem que ser efectuada de uma forma contínua (sem interrupções).

5.7.2 *Base Station*

A *Base Station* permite aumentar a cobertura rádio do meio físico *Wireless*. Todos as tramas que a *Base Station* recebe do meio físico *Wireless*, no *uplink channel* são reenviados (com um atraso temporal) para o mesmo meio físico pelo *downlink channel* (com a potência do sinal reforçada).

5.7.3 *Link Base Station*

Esta estação combina o funcionamento de uma *Link Station* com a de uma *Base Station*. Ou seja não só interliga meios físicos diferentes (*Wired* e *Wireless*) como estrutura o domínio *Wireless*. Quando a *Link Base Station* recebe um PDU do meio físico *Wired* retransmite-o no meio físico *Wireless* pelo *downlink channel*. Quando a *Link Base Station* recebe um PDU do meio físico *Wireless* (pelo *uplink channel*) retransmite-o nos meios físicos *Wired* e *Wireless* (*downlink channel*). Como a rede é *Broadcast*, todos os ES tem que receber o PDU daí a retransmissão para o meio físico *Wireless*. Em qualquer um dos casos entre o início da recepção do PDU e o início da retransmissão existe um atraso t_{rd}.

No MAF foram utilizadas duas estações *Link Base Station*, implementadas como *cut-through* para as quais o t_{rd} é conhecido.

5.8 Sumário

Neste capítulo foi apresentado o modelo definido para a simulação do comportamento temporal de cada componente RFieldbus. Inicialmente foram descritos os elementos de uma rede de comunicação RFieldbus e apresentada a sua hierarquia. Para cada um dos componentes foram descritas as funcionalidades implementadas.

Capítulo 6

Implementação do modelo em OMNeT++

6.1 Introdução

O objectivo deste capítulo é descrever a implementação em OMNeT++ [15] do modelo definido no capítulo anterior para cada componente de uma rede RFieldbus [13]. Para cada módulo composto será apresentada a sua estrutura. Para cada módulo simples será descrita a funcionalidade implementada nesse módulo (fluxograma) e apresentados os seus parâmetros.

As duas estações ES, *Master* e *Slave* foram implementadas em módulos compostos, os restantes componentes RFieldbus foram implementados em módulos simples. Devido ao elevado número de funcionalidades a implementar no *Master*, este foi implementado num módulo composto. A estrutura do módulo foi definida de forma a respeitar a estrutura de camadas da pilha protocolar da estação, permitindo separar a implementação das funcionalidades por camadas. A estação *Slave* também foi implementada num módulo composto, mas apenas para que a sua estrutura reflectisse, a estrutura protocolar de uma estação *Slave* RFieldbus.

Será também apresentada a mensagem definida em OMNeT++ para implementar as tramas que constituem os ciclos de mensagem RFieldbus.

Todos os módulos simples (onde está contida a lógica a executar em cada evento) foram implementados numa abordagem de descrição de lógica por processos. Em OMNeT++ a programação da lógica segundo esta abordagem faz-se dentro da função *activity()* da classe que representa o módulo simples.

Esta abordagem foi adoptada por permitir uma descrição intuitiva da lógica e permitir a implementação de máquinas de estados finitas (FSM). No entanto, a descrição da lógica por eventos é considerada computacionalmente mais eficiente (pág. 61). Ao longo de todo o processo de implementação e validação do simulador como para o modelo/tráfego/topologias implementados esta abordagem não resultou em tempos de simulação elevados (pelo contrário) não foi necessário recorrer a uma abordagem mais eficiente que resultaria numa implementação diferente do mesmo modelo.

Uma rede de comunicação RFieldbus é constituída por domínios de comunicação. Cada domínio de comunicação é uma entidade abstracta que associa um conjunto de estações a um meio físico [12]. Em RFieldbus existem dois tipos de meios físicos disponíveis, *Wired* e *Wireless*. Cada domínio de comunicação pode ter associado um destes meios físicos [12]. Para estabelecer comunicações entre diferentes domínios de comunicação os seus meios físicos são interligados por interligados por estações intermédias (IS). Estas fazem uma “repetição” das tramas entre meios físicos diferentes (tramas diferentes, taxas de transmissão de dados diferentes) [12]. Este simulador (LLRS - *Lower Layer RFieldbus Simulator*) foi definido para suportar a simulação de qualquer topologia RFieldbus.

6.2 Mensagem definida para o simulador LLRS

Foi definida uma mensagem no simulador desenvolvido LLRS para representar as tramas que constituem os ciclos de mensagem em RFieldbus. A mensagem foi definida através de uma classe que herda as propriedades da classe *cmessage* [35] incluída no ambiente de simulação OMNeT++.

Como referido anteriormente, a mensagem definida em OMNeT++ contém o *request* e a *response* de um ciclo de mensagem RFieldbus, sendo cada um activado na sua transmissão. Os campos definidos nas tramas RFieldbus são listados na Tabela 8:

Campo	Função
SD1	*Start Delimiter* 1 (10_H)
SD2	*Start Delimiter* 2 (68_H)
SD3	*Start Delimiter* 3 ($A2_H$)
SD4	*Start Delimiter* 4 (DC_H)
SC	*Single Character* ($E5_H$)
DA	*Destination* Adress
SA	*Source Adress*
FC	*Frame Control*
FCS	*Frame Check Sequence*
ED	*End Delimiter* (16_H)
LE	*Octet Length* [4,249]
LEr	*Octet Length repeated* [4,249]
L	*Information Field Length* [4,249]

Tabela 8 - Campos de dados definidos na mensagem OMNeT++

Os campos de dados listados na Tabela 9 são campos de controlo para uso exclusivo no auxílio à simulação em contraste com os da Tabela 8 que são campos existentes nas mensagens a trocar entre estações.

Campo	Função
msg_n	Número de série do ciclo da mensagem (interno ao simulador)
priority	Prioridade da mensagem: 1 – Hi 2 - Low
type	Indica a trama do pedido : 1 - SD1, 2 - SD2, 3 - SD3 e 4 - SD4
type_R	Indica a trama da resposta: 1 - SD1, 2 - SD2, 3 - SD3 e 4 - SD4
req_resp	Indica se está activo o *request* ou a *reponse*: 1 – *request* 2 - *response*

Tabela 9 - Campos de controlo definidos na mensagem OMNeT++

6.3 Módulos Implementados

No LLRS foram implementados módulos simples e compostos para simular a funcionalidade:

Módulos compostos[16]	*Master*	
	Slave Wired /Wireless	
Módulos Simples	Mobility *Master*	
	Meio físico *Wired*	
	Meio físico *Wireless*	
	Intermediate Systems	*Link Station*
		Base Station
		Link Base Station

Tabela 10 - Módulos Implementados em OMNeT++[17]

6.4 Módulo composto *Master*

6.4.1 Funcionalidade implementada

Este módulo representa a operação de uma estação RFieldbus *Master*. Foi implementado como módulo composto devido ao seu alargado conjunto de funcionalidades. A estrutura do módulo foi escolhida de forma a reflectir a estrutura das camadas da arquitectura protocolar e a separar as funcionalidades:

Geração de mensagens. Consiste em produzir o tráfego gerado pelas camadas de aplicação, gerar as tramas que constituem cada ciclo de mensagem e emiti-las de acordo com a sua periodicidade (no caso dos ciclos de mensagem periódicos).

As funcionalidades da camada FDL: a máquina de estados do controlador da FDL operando em regime permanente, a gestão das filas de mensagem, a implementação da gestão do *token* (transmissão e recepção), o algoritmo de despacho de mensagens e a gestão de cada ciclo de mensagem (verificação do *Idle time* e do *Slot time*).

É a camada física, que nas estações Rfieldbus apenas retransmite instantaneamente as mensagens recebidas. Isto deve-se ao facto de o cálculo do tempo de transmissão e a introdução do atraso temporal correspondente à transmissão da trama no meio físico serem implementados no módulo simples correspondente ao meio físico.

---

[16] - Por simplicidade de apresentação, não vão ser referidos os parâmetros obrigatórios nos módulos compostos para passagem de valores aos módulos simples constituintes.

[17] - Os módulos compostos contém módulos simples que serão apresentados na descrição da estrutura dos módulos compostos.

6.4.2 Estrutura do módulo composto

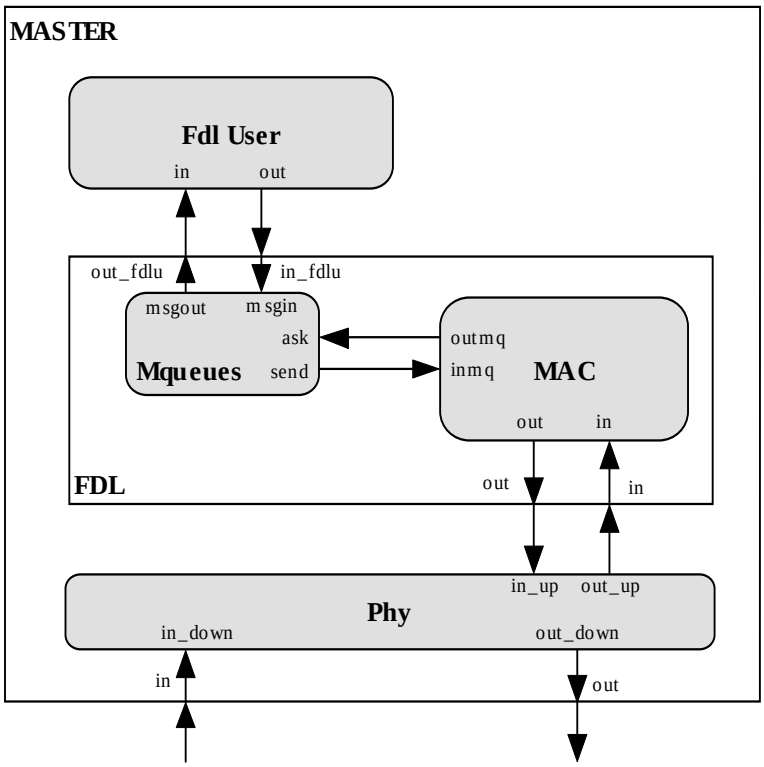


Figura 47 - Estrutura do módulo composto *Master*[18]

Como se pode observar na Figura 47 o módulo *Master* é composto por três módulos: FDL User (Simples), FDL e PHY (Simples). O sub-módulo FDL é também um módulo composto por dois módulos simples: Mqueues e MAC.

De seguida vão ser descritos os módulos simples que constituem o módulo composto *Master*.

[18] - Os módulos simples estão a cinzento, os módulos compostos são transparentes.

6.4.3 Sub-módulo FDL User (*simple module*)

Funcionalidade implementada

Este módulo simples é responsável pela geração de todos os ciclos de mensagem, que serão entregues para transmissão à FDL. Permite gerar ciclos de mensagem com ou sem resposta/*acknowledge*, periódicos ou esporádicos e de alta ou baixa prioridade. Para conhecer os ciclos de mensagem acede a um ficheiro de texto onde estão definidos todos os ciclos de mensagem da rede de comunicação. Cada ciclo de mensagem está definido numa linha desse ficheiro de texto. Os campos de cada linha do ficheiro são apresentados e caracterizados de seguida:

Tamanho_do_Request Tamanho_da_Response Prioridade Tempo_de_Inicio Período Iniciator Response

Nos ciclos de mensagem sem resposta/*acknowledge* o tamanho da resposta é definido como zero. O campo prioridade pode ser 1 ou 2 conforme o ciclo de mensagem seja considerado de alta ou baixa prioridade. O campo tempo de início marca o instante em que o ciclo de mensagem é enviado pela primeira ou única vez (dependendo se é ou não periódica). O campo período contém o período do ciclo de mensagem, no caso não ser periódica o período é zero.

Estrutura do módulo

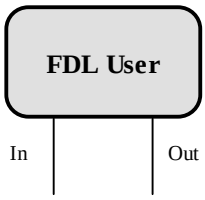


Figura 48 - Estrutura do módulo simples FDL User

Parâmetros

Este módulo não tem parâmetros.

Fluxograma

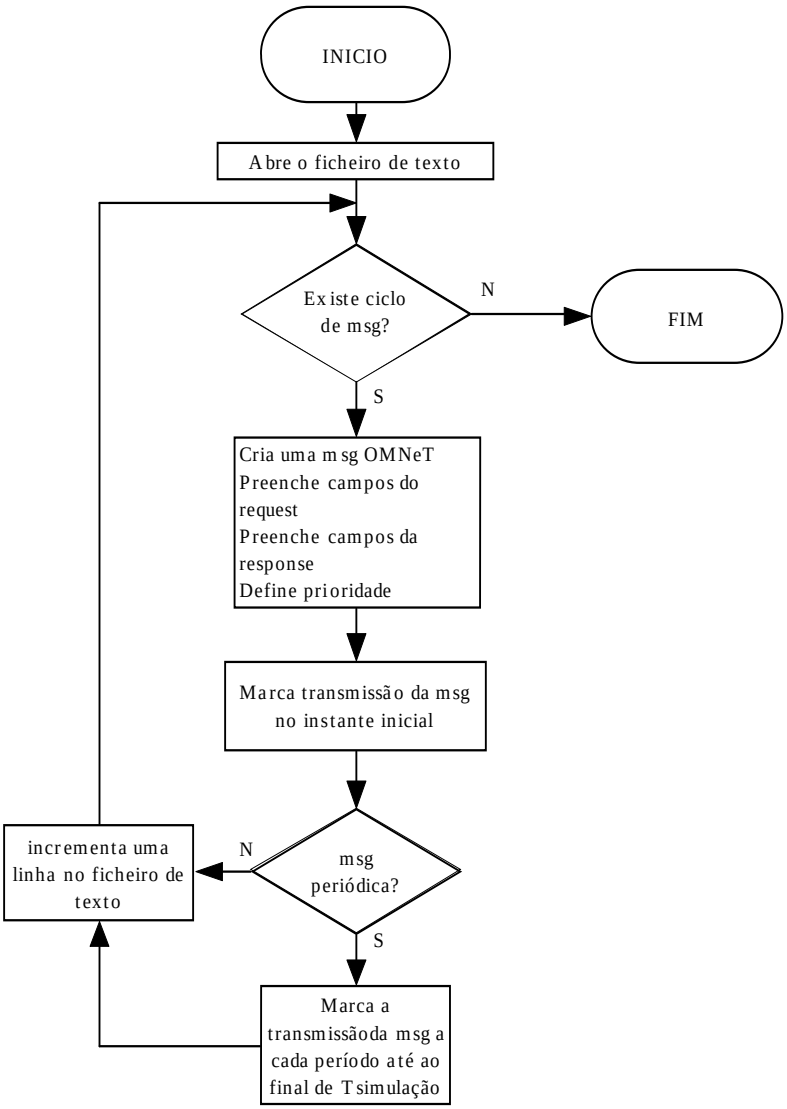


Figura 49 - Fluxograma da funcionalidade implementada no módulo simples FDL User

6.4.4 Sub-módulo Mqueues (*simple module*)

Funcionalidade implementada

O módulo simples *Mqueues* contém as duas filas de mensagem do tráfego da camada FDL. A *Hi Priority message Queue* e a *Low Priority message Queue*.

A funcionalidade implementada pelo módulo simples *Mqueues* representa a deposição das mensagens pelo utilizador da FDL (através da interface desta, serviços SRD e SDN) nas respectivas filas de mensagem e também, a entrega de mensagens para transmissão ao módulo simples MAC mediante pedido deste.

Estrutura do módulo

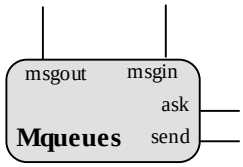


Figura 50 - Estrutura do módulo simples Mqueues

Parâmetros

Não tem parâmetros.

Fluxograma

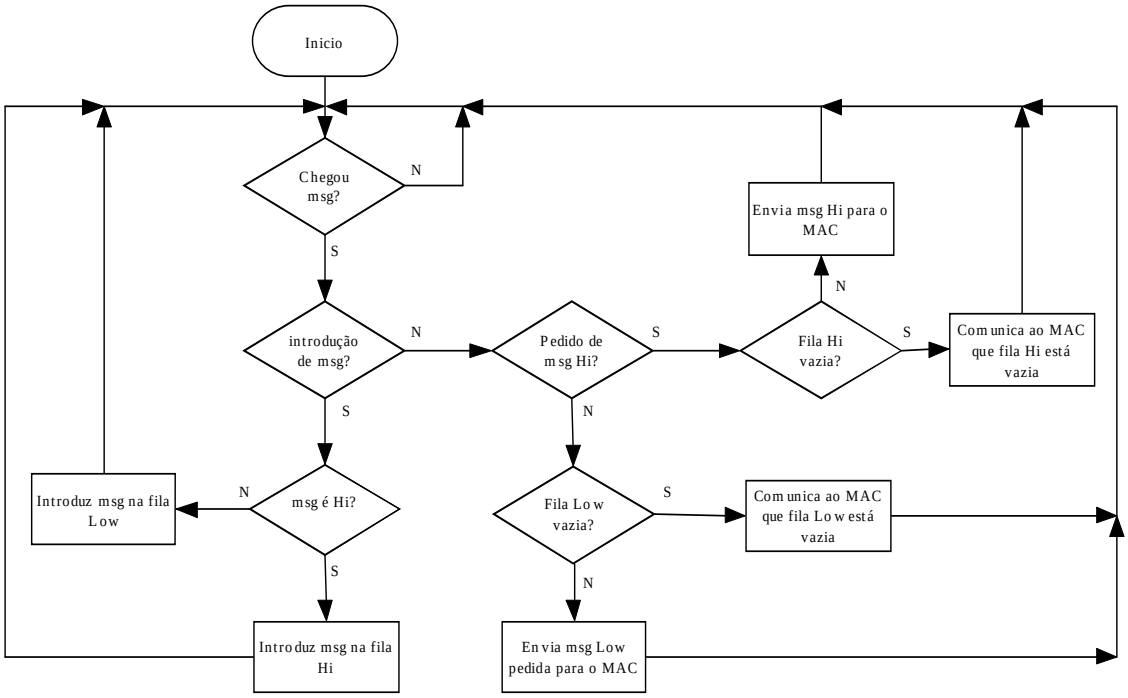


Figura 51 - Fluxograma da funcionalidade implementada no módulo simples Mqueues

6.4.5 Sub-módulo MAC (*simple module*)

Funcionalidade implementada

A funcionalidade implementada no *simple module* MAC engloba os seis estados do controlador da FDL implementados (coloridos a cinza na Figura 17). São eles o estado *Active Idle*, *Await Data Response*, *Use Token*, *Pass Token*, *Check Token Pass* e *Check Access Time*. O controlador da FDL foi implementado com apenas dois estados, um representando o estado *Active Idle* (recepção do *Token*) e o outro englobando todos os outros estados implementados. Neste segundo estado estão as funcionalidades do escalonamento das mensagens, o cálculo do T_{TH} disponível (*Check Access Time*), o processamento de cada ciclo de mensagem (*Use Token* e *Await Data Response*) e a transmissão do *Token* (*Pass Token* e *Check Token Pass*).

Estrutura do módulo

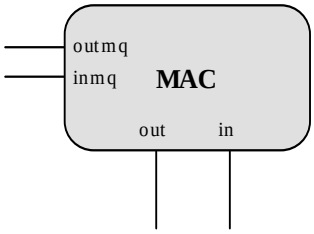


Figura 52 - Estrutura do módulo simples MAC

Parâmetros

Parâmetro	Descrição
adressM	Endereço da própria estação (TS)
NS	Endereço da estação a quem passa o *Token*
PS	Endereço da estação de quem recebe o *Token*
adressS[]	Endereços dos *Slaves* associados a este *Master*
t_{ID1}	*Idle time* 1, ciclos de mensagem com resposta/Ack
t_{ID2}	*Idle time* 2, ciclos de mensagem sem resposta/Ack
t_{SL}	*Slot time*
max_retry_count	Número máximo de tentativas de retransmissão de cada ciclo de mensagem
Bitrate	Taxa de transmissão de dados no meio físico
t_{tr}	*Target Rotation Time*, tempo de referência para a rotação do *Token*

Fluxograma

O *simple module* MAC é o que tem o algoritmo mais complexo implementado. Consequentemente o fluxograma que o representa é também complexo. Para ser possível a sua representação na dimensão de uma página o fluxograma foi dividido, sendo primeiro apresentado um fluxograma geral e posteriormente fluxogramas que detalham alguns processos (como sub rotinas em programação). Na Figura 53 está representado o fluxograma com o algoritmo implementado no *simple module* MAC. Neste existem dois processos WAIT TOKEN e DETEM TOKEN cujos fluxogramas serão detalhados na Figura 54 e na Figura 55 respectivamente. Finalmente no fluxograma do processo DETEM TOKEN existe o processo PASSA TOKEN cujo fluxograma será detalhado na Figura 56.

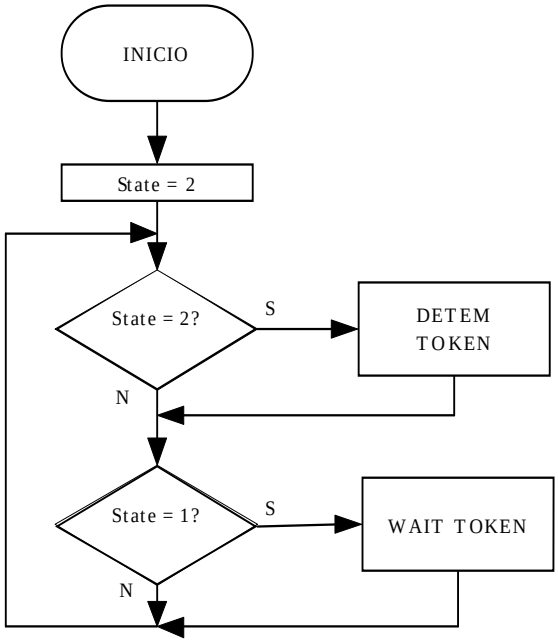


Figura 53 - Fluxograma da funcionalidade implementada no módulo simples MAC

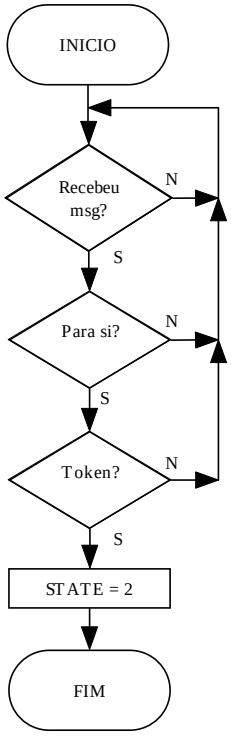


Figura 54 - Fluxograma da funcionalidade implementada no estado *Wait Token*

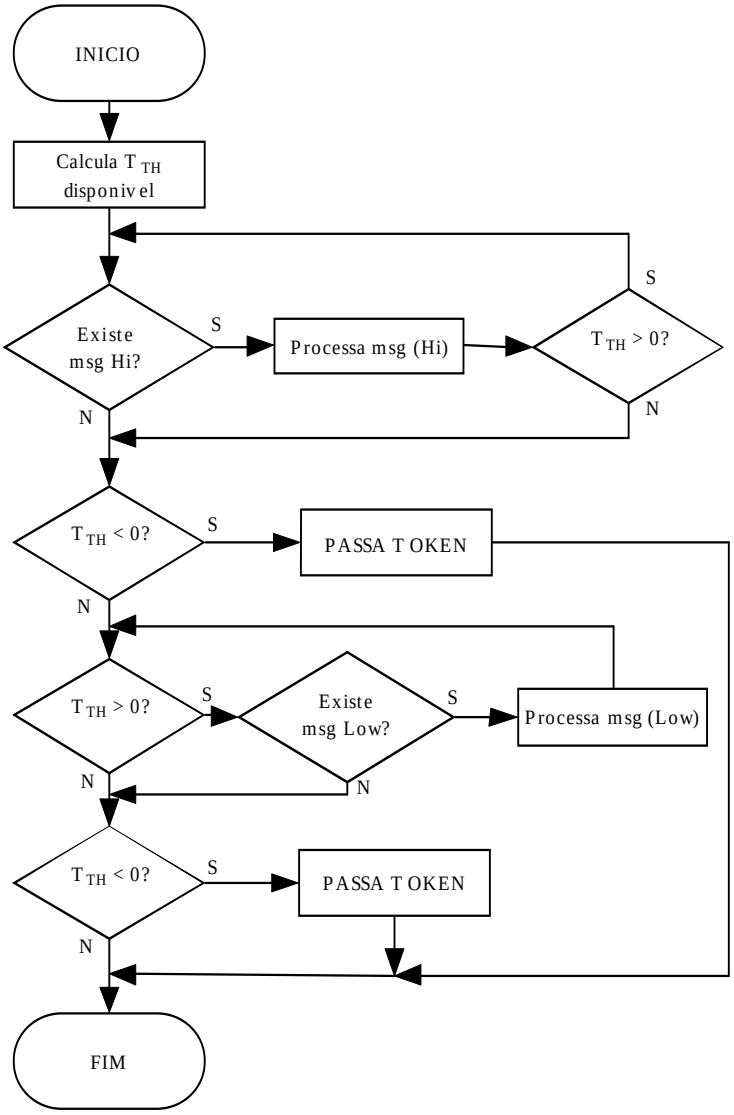


Figura 55 - Fluxograma da funcionalidade implementada no estado *Detem Token*

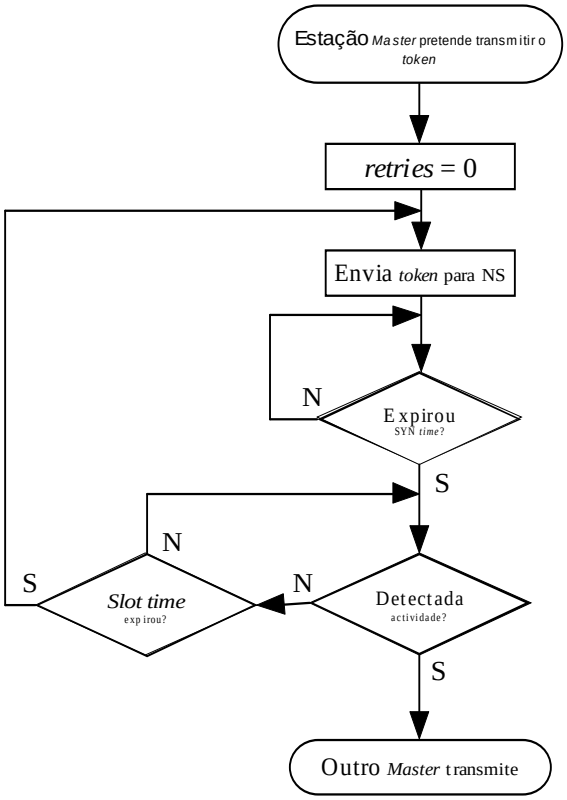


Figura 56 - Fluxograma da funcionalidade implementada no estado Passa *Token*

De acordo com 5.2.6, a descrição do processo "Processa mensagem (*Hi)* e (*Low)*" no fluxograma da Figura 55 encontra-se na Figura 10 para os ciclos de mensagem com resposta e na Figura 13 para os ciclos de mensagem sem resposta.

6.4.6 Sub-módulo Phy (*simple module*)

Funcionalidade implementada

O Phy é um módulo simples que não implementa qualquer funcionalidade RFieldbus. Apenas foi colocado, para a estrutura dos *End Systems* (ES) implementados, reflectir a estrutura real da arquitectura de uma estação *Master* RFieldbus.

Estrutura do módulo

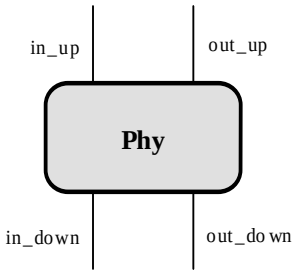


Figura 57 - Estrutura do módulo simples Phy

Parâmetros

Este módulo não tem parâmetros.

Fluxograma

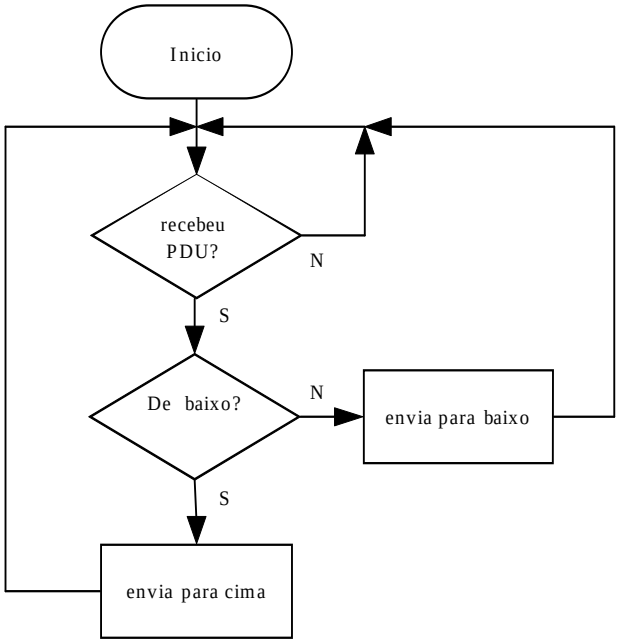


Figura 58 -Fluxograma da funcionalidade implementada no módulo simples Phy

6.5 Módulo composto *Slave*

6.5.1 Funcionalidade implementada

O *Slave* é uma estação passiva, ou seja não pode tomar a iniciativa de transmitir as tramas no meio físico [6], respondendo apenas a *request*s endereçados a si por um *Master*.

Por opção do autor as *frames* de pedido e resposta foram englobadas na mesma mensagem OMNeT++ (página 80), embora estando uma activa, a cada instante. A vantagem tem a ver com a geração do tráfego pois o conjunto das *message streams* da rede de comunicação tem que ser conhecido (para poder ser simulado) e é mais simples gerar o ciclo de mensagem completo no *Master* do que ter um gerador de tráfego no *Master* e outro no *Slave* cada um a gerar a sua parte da mesma *message stream*.

Assim a funcionalidade implementada por um *Slave* nos ciclos de mensagem com resposta é; quando recebe um *request* endereçado a si, activa a *response* na mensagem OMNeT++, aguardar t_{rt} e envia-a para o meio físico

6.5.2 Estrutura do módulo composto

Para implementar a funcionalidade do *Slave*, o seguinte módulo composto foi definido.

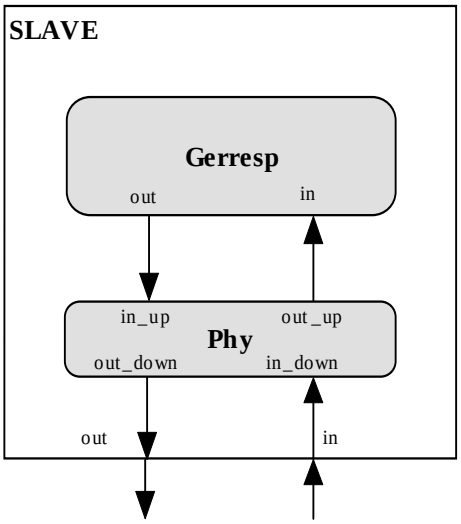


Figura 59 - Estrutura do módulo composto *Slave*

Como se pode observar na Figura 59, o módulo *Slave* é composto por dois módulos simples. O *Phy* e o *Gerresp*. De seguida será descrita para cada um destes módulos simples a funcionalidade implementada recorrendo a um fluxograma, a estrutura do módulo (*gates*) e os seus parâmetros.

6.5.3 Sub-módulo Gerresp (*simple module*)

Funcionalidade implementada

O Gerresp é o módulo responsável pela funcionalidade do *Slave*. Este recebe todas as tramas e verifica se é um *request* dirigido a este *Slave*, quando chegar um desactiva na mensagem OMNeT++ o *request* e activa a *response*, de seguida espera t_{rt} e quando este tempo expirar transmite a *response*.

Estrutura do módulo

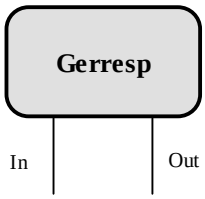


Figura 60 - Estrutura do módulo simples Gerresp

Parâmetros

Parâmetro	Descrição
adressS	Endereço do *Slave*, endereço da própria estação
t_{rt}	*Responder turnaround time*, intervalo de tempo que decorre entre o instante em que termina a recepção do *request* e o instante em que inicia a transmissão da *response*

Tabela 11 - Parâmetros do módulo simples Gerresp

Fluxograma

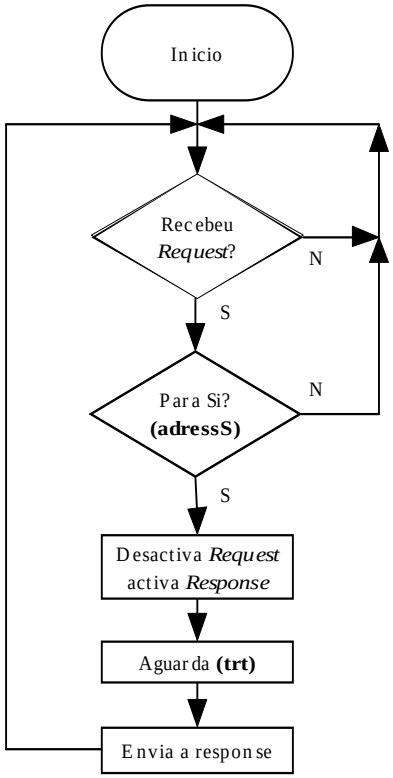


Figura 61 - Fluxograma da funcionalidade implementada no módulo simples Gerresp

6.5.4 Sub-módulo Phy (*simple module*)

Funcionalidade implementada

O Phy é o mesmo módulo simples descrito no *Master*. Não implementa qualquer funcionalidade RFieldbus, apenas foi colocado para a estrutura dos *End Systems* (ES) implementados reflectir a estrutura real da arquitectura de uma estação *Slave* RFieldbus.

Estrutura do módulo

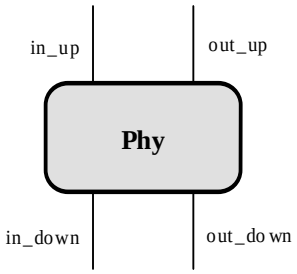


Figura 62 - Estrutura do módulo simples Phy

Parâmetros

Este módulo não tem parâmetros.

Fluxograma

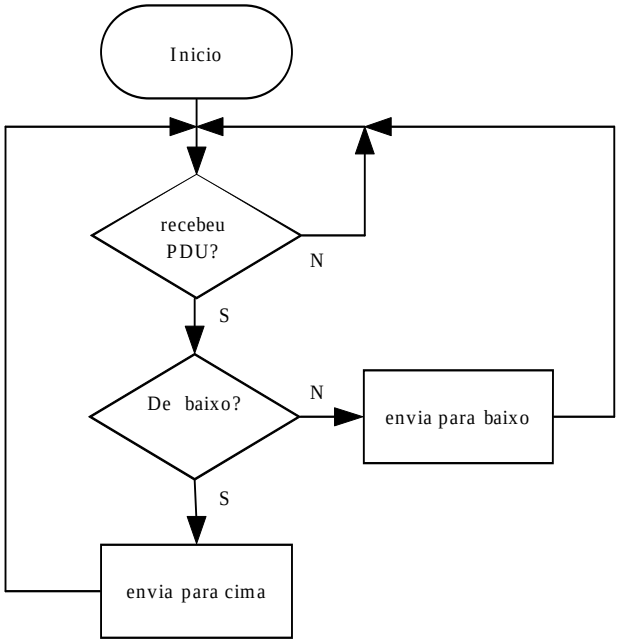


Figura 63 - Fluxograma da funcionalidade implementada no módulo simples Phy

6.6 Módulo composto *SlaveMob*

6.6.1 Funcionalidade implementada

O módulo *SlaveMob* representa um slave móvel. A estrutura deste módulo composto difere da estrutura do módulo composto *Slave* porque lhe foi adicionado um módulo simples (*Chanaccess*) cuja função é gerir o acesso entre dois canais rádio disponíveis.

A funcionalidade implementada é totalmente idêntica à de um slave associado a um meio físico com fios, fazendo o módulo simples *Chanaccess* a comutação entre dois canais rádio disponíveis.

6.6.2 Estrutura do módulo composto

Para implementar a funcionalidade do *Slave móvel*, foi implementado o seguinte módulo composto.

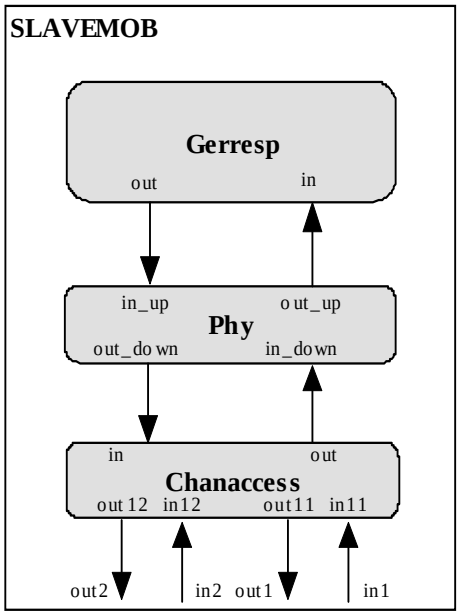


Figura 64 - Estrutura do módulo composto *SlaveMob*

Como se pode observar na Figura 64, o módulo *SlaveMob* é composto por três módulos simples. O *Phy* e o *Gerresp* já foram descritos anteriormente. De seguida será descrita a funcionalidade implementada recorrendo a um fluxograma, a estrutura e os seus parâmetros do módulo simples *Chanaccess*.

6.6.3 Sub-módulo Chanaccess (*simple module*)

Funcionalidade implementada

Este módulo implementa um algoritmo que simula a avaliação dos canais rádio por parte dos *slaves* móveis no período de gestão da mobilidade. Para isso aguarda a chegada do *beacon trigger PDU* (início do período de gestão da mobilidade) e gera um número aleatório que definirá qual o canal que apresenta melhor qualidade de sinal.

Estrutura do módulo

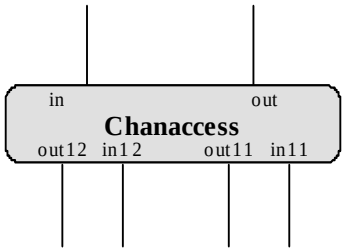


Figura 65 - Estrutura do módulo simples Chanaccess

Parâmetros

Este módulo não tem parâmetros.

Fluxograma

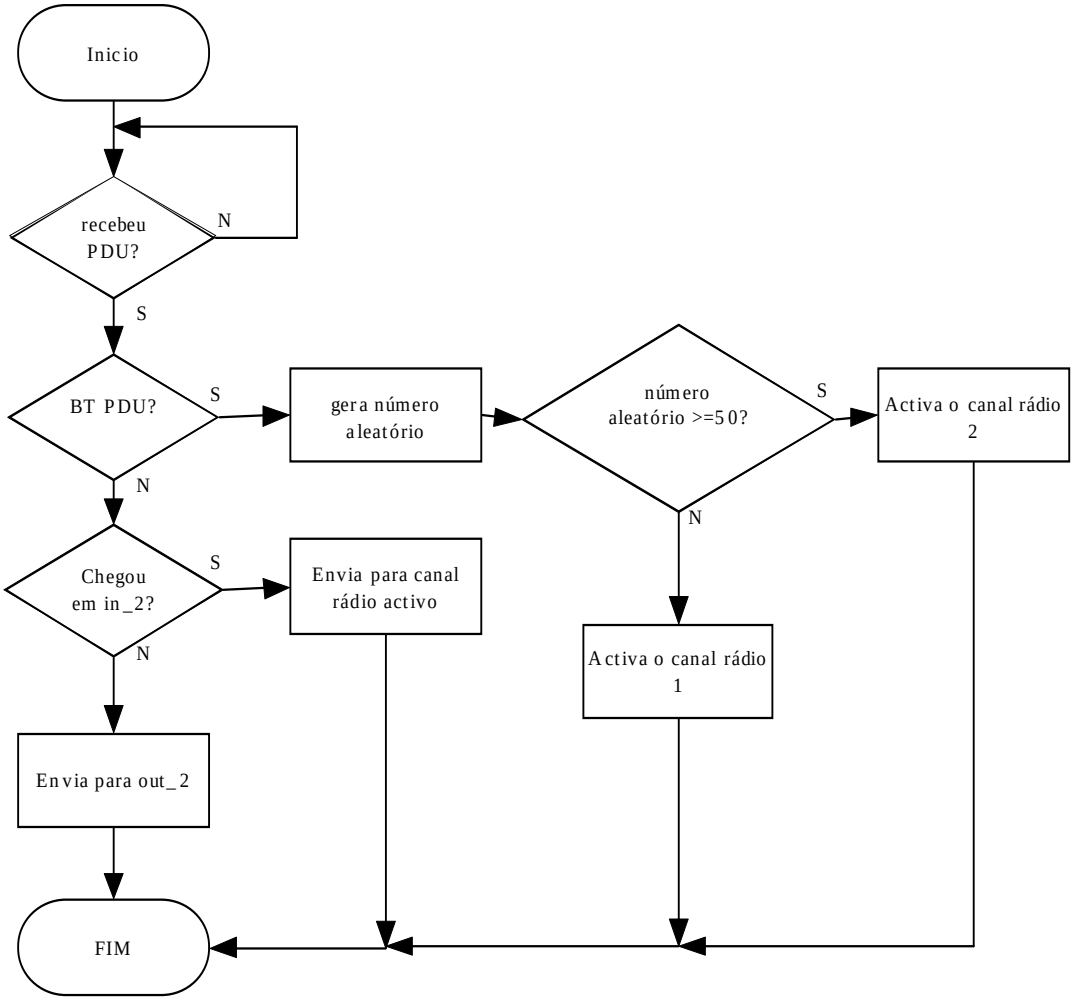


Figura 66 - Fluxograma da funcionalidade implementada no módulo simples Chanaccess

6.7 Descrição dos módulos Simples

6.7.1 Mobility *Master*

Funcionalidade implementada

De acordo com a funcionalidade do *Mobility Master*, este, a cada recepção do *Token* verifica se decorreu o *Period for Triggering Mobility Management Procedure* (T_{FTMMP}). Se não decorreu, devolve o *Token*. Caso tenha decorrido, desencadeia o procedimento de gestão da mobilidade com o envio da trama especial *BT PDU* (*unacknowledged*) e aguarda $T_{ID2mobm}$ até devolver o *Token* à próxima estação activa [12].

Estrutura do módulo

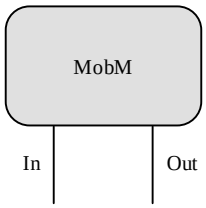


Figura 67 - Estrutura do módulo simples MobM

Parâmetros

Parâmetro	Descrição
TS	Endereço do Mobility *Master*
NS	Endereço da estação para a qual passa o *Token*
PS	Endereço da estação que lhe envia o *Token*
T_{FTMMP}	Período ao fim do qual deve desencadear o procedimento de gestão da mobilidade
$T_{ID2mobm}$	*Idle time* a inserir após o envio da trama especial *BT PDU*

Tabela 12 - Parâmetros do módulo simples MobM

Fluxograma

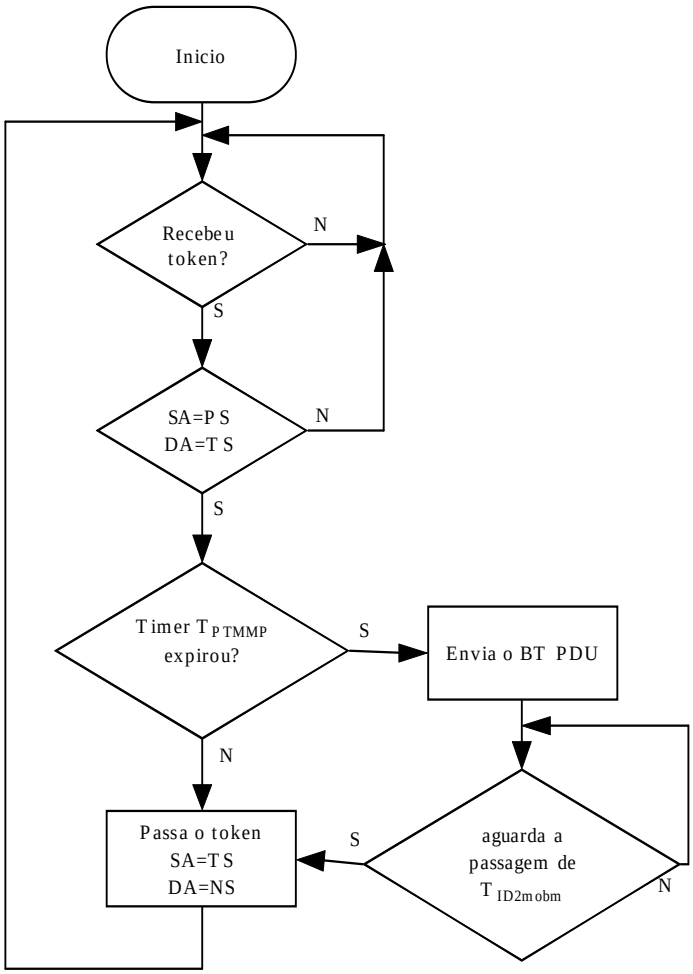


Figura 68 - Fluxograma da funcionalidade implementada no módulo simples MobM

6.7.2 Meio físico *Wired*

Funcionalidade implementada

Como referido anteriormente, cada domínio de comunicação tem associado um meio físico e um conjunto de estações. Nesse meio físico serão transmitidas as tramas geradas por cada estação do domínio. De acordo com o modelo definido para o meio físico (pág. 74) o que este implementa é o tempo de transmissão de cada trama.

Estrutura do módulo

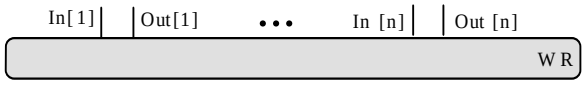


Figura 69 - Estrutura do módulo simples PhyWired

Parâmetros

Parâmetro	Descrição
bit_rate_WR	taxa de transmissão de dados do meio físico *Wired* (WR), permite calcular o tempo de transmissão de cada trama nesse meio físico
Prob_error_WR	Probabilidade de erro no meio físico *Wired* (WR), ou seja, a probabilidade de acontecer um erro na transmissão de uma trama no meio físico

Tabela 13 – Parâmetros do módulo simples PhyWired

Fluxograma

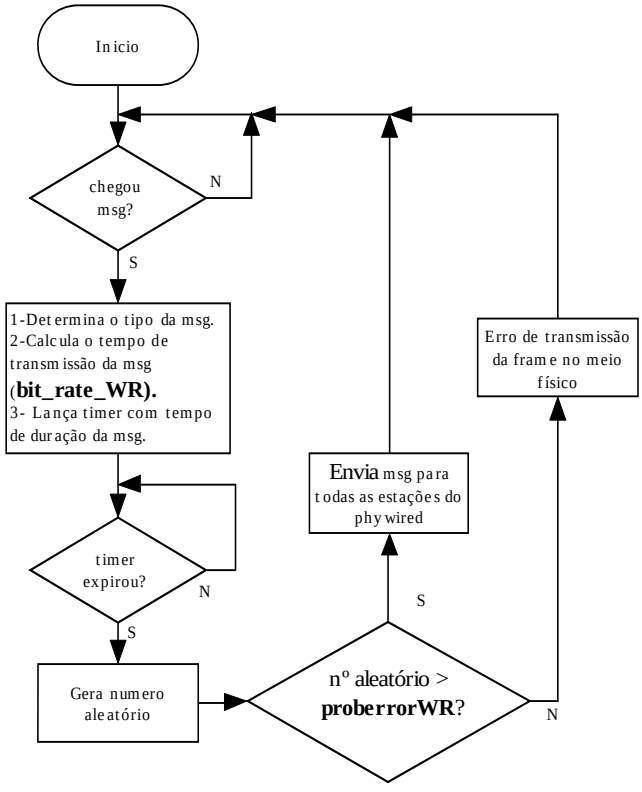


Figura 70 - Fluxograma da funcionalidade implementada no módulo simples PhyWired

6.7.3 Meio físico *Wireless*

Funcionalidade implementada

A funcionalidade implementada neste módulo simples que representa o meio físico *Wireless* (pág. 74) é muito parecida à implementada para o meio físico com fios. No entanto existem algumas diferenças. Os cálculos são efectuados sobre os parâmetros que caracterizam o meio físico sem fios, o método de cálculo do tempo de transmissão da trama é diferente dada estar definida outra estrutura de trama (pág. 51) neste meio físico.

Também devido ao facto de os meios físicos sem fios poderem ser ou não estruturados, o módulo simples que representa o meio físico sem fios deve suportar a funcionalidade de ambos os casos. Como referido na página 42 um domínio *Wireless* não estruturado conduz a uma topologia de rede *Direct Link Network* na qual as comunicações *Wireless* se efectuam apenas num canal rádio dentro da área de cobertura comum de todas as estações do domínio. Num domínio *Wireless* estruturado por uma (*Link*) *Base Station*, as comunicações são efectuadas em dois canais (pág. 44). Nomeadamente a (*Link*) *Base Station* recebe tramas no *uplink channel* e envia tramas no *downlink channel*. Já os ES enviam no *uplink channel* e recebem no *downlink channel*. Assim num domínio *Wireless* estruturado o *uplink channel* é utilizado por uma estação para transmitir dados (exclusivamente) para a (*Link*) *Base Station*, enquanto o *downlink channel* serve para a (*Link*) *Base Station* fazer *Broadcast* de tramas para todas as estações associadas a esse meio físico.

O módulo simples implementado em OMNeT++ que representa o meio físico *Wireless*, detecta que uma trama está a ser transmitida no *uplink channel* através de uma *flag* existente na mensagem. Neste caso aguarda o tempo de transmissão e entrega a trama apenas à (*Link*) *Base Station*. Antes de devolver a trama ao meio físico *Wireless* esta estação desactiva a *flag* o que faz com que o meio físico *Wireless* a transmita em *Broadcast* para todas as estações associadas ao meio físico.

Estrutura do módulo

Figura 71 - Estrutura do módulo simples PhyWireless

É no primeiro conjunto de gates (In[1] Out[1]) que será ligada a (*Link*) *Base Station* no caso do domínio ser estruturado.

Parâmetros

Parâmetro	Descrição
bit_rate_WL	taxa de transmissão de dados do meio físico *Wireless* (WL), permite calcular o tempo de transmissão de cada trama nesse meio físico
Prob_error_WL	Probabilidade de erro no meio físico *Wireless* (WL), ou seja, a probabilidade de acontecer um erro na transmissão de uma trama no meio físico

Tabela 14 - Parâmetros do módulo simples PhyWireless

Fluxograma

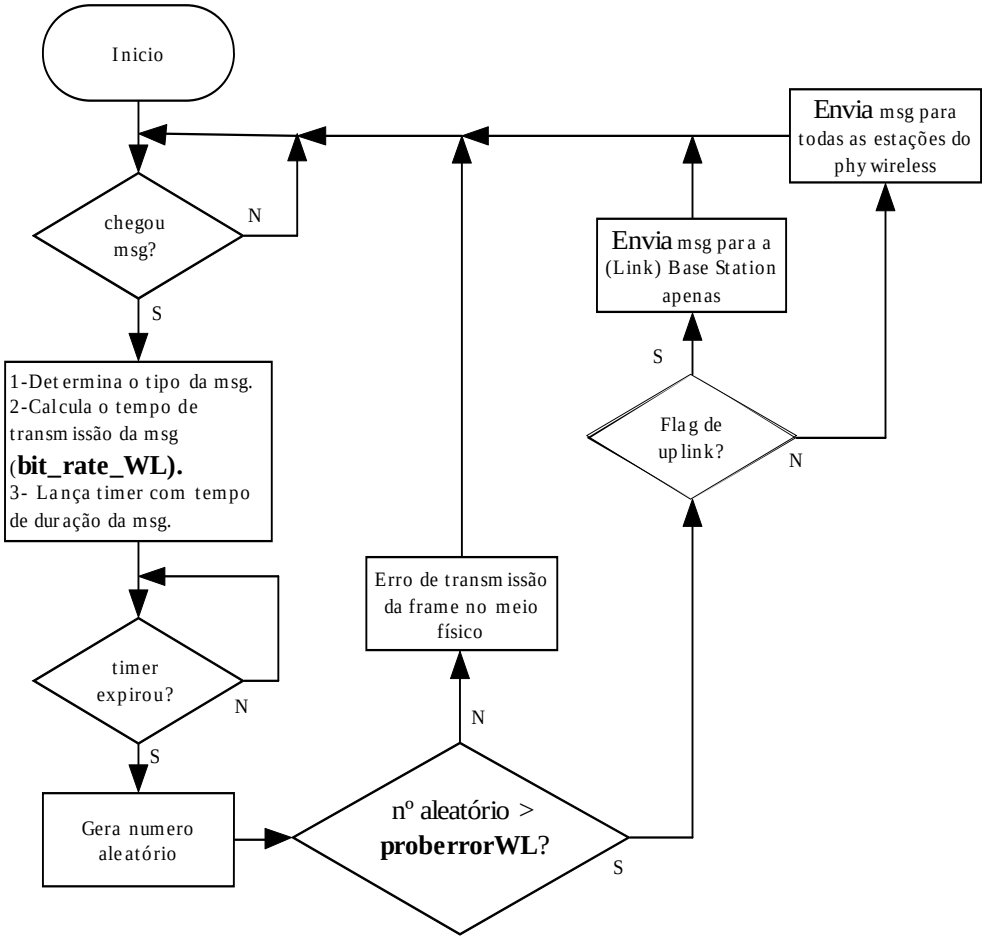


Figura 72 - Fluxograma da funcionalidade implementada no módulo simples PhyWireless

6.7.4 *Base Station* (*Intermediate System*)

Funcionalidade implementada

A *Base Station* é uma estação que permite estruturar as comunicações num domínio *Wireless*. Desta forma cada dispositivo móvel passa a poder comunicar numa área (ou volume) comum definida pelo seu alcance e o alcance da *Base Station*. Num Domínio *Wireless* estruturado as comunicações efectuam-se em dois canais; *uplink* e *downlink channel*. A *Base Station* recebe todas as tramas do meio físico *Wireless* no *uplink channel* e retransmite-as para o mesmo meio no *downlink channel*. Como os *End System* (ES) transmitem no *uplink channel* e recebem no *downlink channel*, todas as comunicações neste domínio passam na *Base Station*, que as retransmite reforçando a potência do sinal.

A funcionalidade implementada no módulo simples que representa a *Base Station* é a recepção de tramas no *uplink channel* e a sua retransmissão no *downlink channel* com um atraso t_{relay}.

Estrutura do módulo

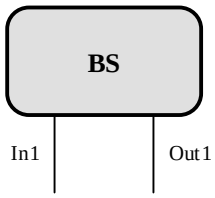


Figura 73 - Estrutura do módulo simples *Base Station*

Parâmetros

Parâmetro	Descrição
t_{relay}	Tempo de atraso introduzido pela *Base Station* na retransmissão da trama entre os domínios de comunicação que interliga. Este tempo de atraso é medido entre o instante em que inicia a recepção e o instante em que inicia a retransmissão.

Figura 74 - Parâmetro do módulo simples *Base Station*

Fluxograma

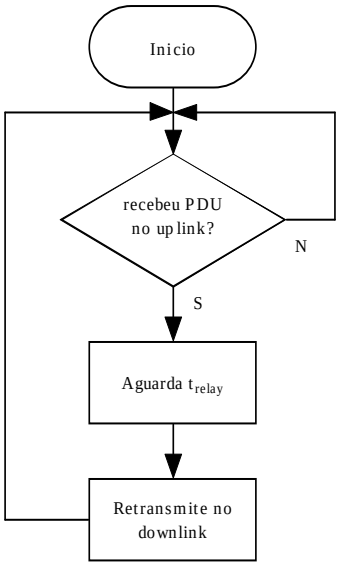


Figura 75 - Fluxograma da funcionalidade implementada no módulo simples *Base Station*

6.7.5 *Link Station* (*Intermediate System*)

Funcionalidade implementada

De acordo com o modelo definido na página 76 a *Link Station* interliga dois meios físicos diferentes, retransmitindo as tramas que recebe de um dos meios físicos para o outro. Meios físicos diferentes implicam tramas de formatos diferentes. Numa implementação real a *Link Station*, assim como a *Link Base Station* irá extrair a informação da trama recebida e inseri-la na trama respeitando o formato do outro meio físico. Como no simulador implementado, toda a trama e seu conteúdo, são encapsulados no interior da mensagem OMNET++ definida neste simulador e os módulos que representam os meios físicos calculam do tempo de transmissão das tramas com base na informação contida na mensagem OMNET++, não há necessidade de fazer a extracção dessa informação nos modelos da *Link Station* e da *Link Base Station*. Desta forma são reduzidas as operações a efectuar sobre a mensagem na simulação.

Estrutura do módulo

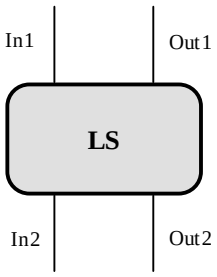


Figura 76 - Estrutura do módulo simples *Link Station*

Parâmetros

Parâmetro	Descrição
t_{relay}	Tempo de atraso introduzido pela *Link Station* na retransmissão da trama entre os domínios de comunicação que interliga. Este tempo de atraso é medido entre o instante em que inicia a recepção e o instante em que inicia a retransmissão.

Tabela 15- Parâmetro do módulo simples *Link Station*

Fluxograma

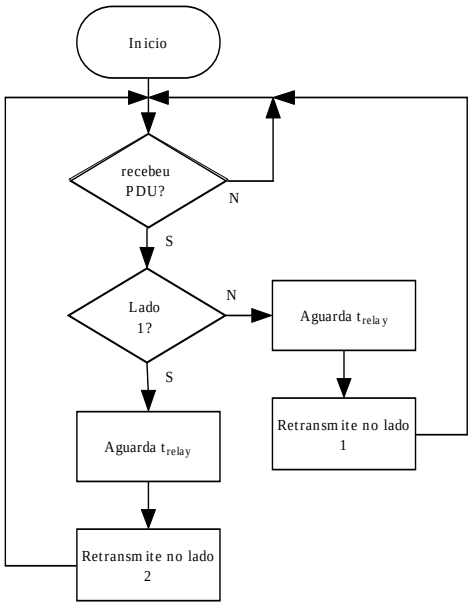


Figura 77 - Fluxograma da funcionalidade implementada no módulo simples *Link Station*

6.7.6 *Link Base Station* (*Intermediate System*)

Funcionalidade implementada

A *Link Base Station* é uma estação que implementa as funcionalidades das duas estações anteriores; *Link Station* e *Base Station*.

Esta estação permite interligar dois meios físicos diferentes e simultaneamente estruturar um meio físico *Wireless*. Quando recebe uma trama do meio físico *Wired*, aguarda t_{relay} e retransmite a trama para o meio físico *Wireless* utilizando o *downlink channel*. Quando recebe uma trama do meio físico *Wireless* (através do *uplink channel*), aguarda t_{relay} e retransmite simultaneamente a trama no meio físico *Wired* e no *downlink channel* do meio físico *Wireless*.

Estrutura do módulo

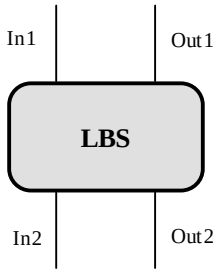


Figura 78 - Estrutura do módulo simples *Link Base Station*

Parâmetros

Parâmetro	Descrição
t_{relay}	Tempo de atraso introduzido pela *Link Base Station* na retransmissão da trama entre os domínios de comunicação que interliga. Este tempo de atraso é medido entre o instante em que inicia a recepção e o instante em que inicia a retransmissão.

Tabela 16 - Parâmetro do módulo simples *Link Base Station*

Fluxograma

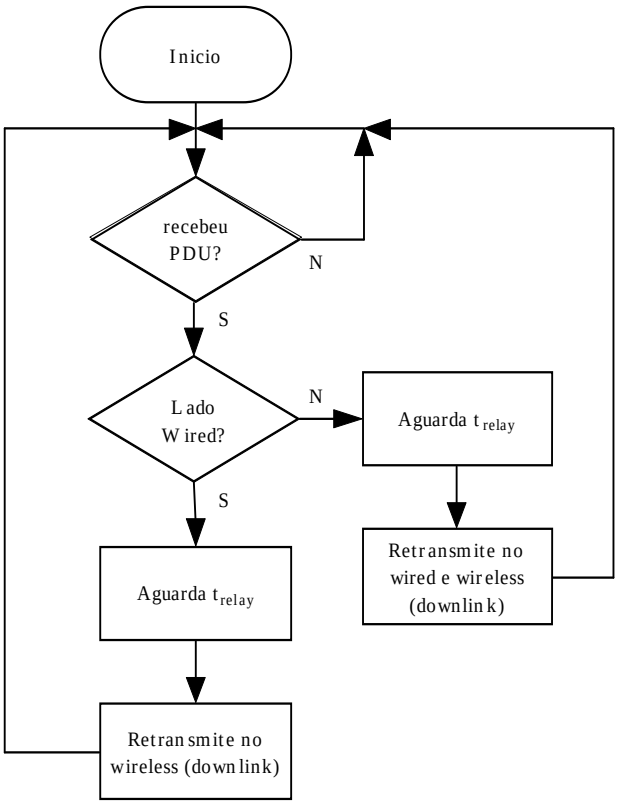


Figura 79 - Fluxograma da funcionalidade do módulo simples *Link Base Station*

6.8 Sumário

Neste capítulo foi apresentada a implementação do modelo definido no capítulo anterior. Cada componente RFieldbus foi implementado num módulo OMNeT++. Para cada módulo foi apresentada a sua estrutura, uma descrição detalhada das funcionalidades implementadas e os seus parâmetros.

Capítulo 7

Caso de estudo, apresentação de resultados

7.1 Introdução

Neste capítulo são apresentados os resultados obtidos por simulação no LLRS do comportamento temporal da rede do projecto piloto *Manufacturing Automation Fieldtrial* (MAF) [16], apresentado no Anexo A. A sua validade será confirmada por confrontação com os resultados observados através do analisador de rede PROFIBUS associado ao domínio *Wired* do MAF [36].

É apresentado o MAF, a topologia de rede a simular, o conjunto de parâmetros que caracterizam os elementos dessa topologia e o conjunto dos ciclos de mensagem.

São explicadas as fórmulas de cálculo da duração dos ciclos de mensagem com e sem resposta, feitas considerações acerca de limitações existentes na observação do comportamento desta topologia com o analisador de rede e apresentados os resultados observados e simulados para validar os resultados de simulação.

7.2 *Manufacturing Automation Fieldtrial* (MAF)

O projecto europeu RFieldbus visou o desenvolvimento de uma nova arquitectura, baseada na da rede de campo PROFIBUS, na qual foram incluídas a comunicação sem fios e integrado o tráfego multimédia. Com o intuito de demonstrar as características técnicas desta nova arquitectura foram implementados dois projectos piloto, um orientado engenharia de processos contínuos e outro à engenharia de processos discretos [37].

O *Manufacturing Automation Fieldtrial* (MAF) [16] foi o projecto piloto orientado à engenharia de processos discretos, implementado no Laboratório de Sistemas Críticos do Instituto Superior de Engenharia do Porto (LASCRI-ISEP) [38] no âmbito do projecto europeu RFieldbus [13].

No anexo A foi incluída uma descrição detalhada do MAF: o sistema mecânico, seus componentes e funcionamento, bem como dos correspondentes ciclos de mensagem gerados na operação do MAF.

O caso de estudo apresentado nesta tese consistiu na comparação dos resultados observados no MAF com os resultados obtidos por simulação de um sistema com características idênticas às do MAF (no simulador LLRS).

7.3 Topologia da rede a simular

A topologia de rede implementada neste caso de estudo é a do *Manufacturing Automation Field Trial* [A.1.3]. Na figura seguinte é apresentada uma captura do ecrã do simulador LLRS na qual é possível observar essa mesma topologia.

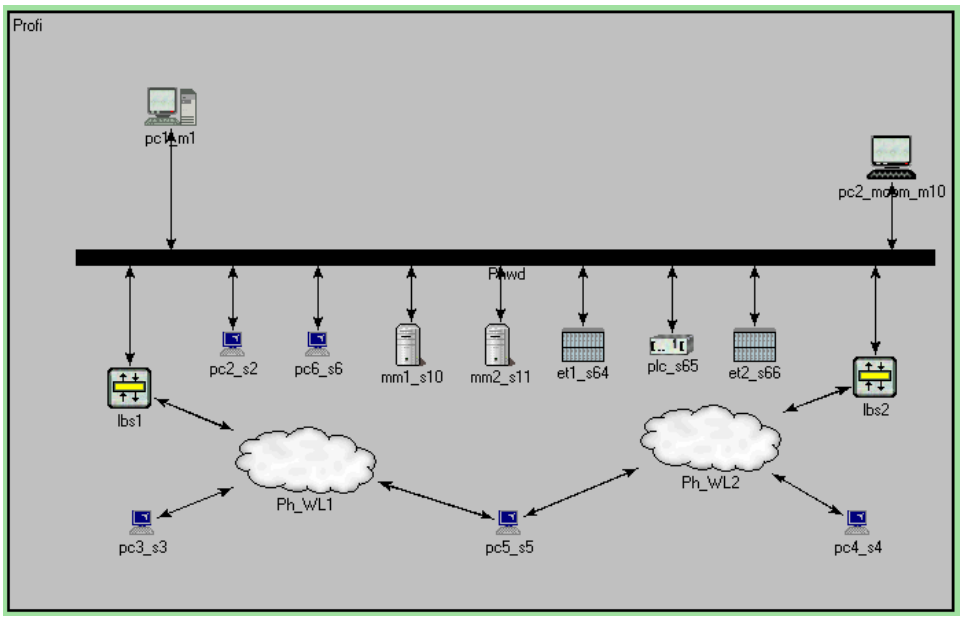


Figura 80 - *Manufacturing Automation Fieldtrial* RFieldbus *Network topology*

7.4 Parâmetros que caracterizam os elementos da rede de comunicação

São de seguida apresentados os parâmetros que caracterizam o comportamento de cada um dos elementos do MAF. Os parâmetros temporais são calculados pela System Planning Application (SPA) considerando a topologia de rede, as características dos ciclos de mensagens e os diferentes tipos de meios físicos [39] de forma a garantir a operação em tempo real da rede de comunicação.

7.4.1 Master

Parâmetro	Valor
adressM	1
NS	10
PS	10
adressS[]	{2,3,4,5,6,11,12,64,65,66}
t_{ID1}	396 bt
t_{ID2}	207 bt
t_{SL}	2950 bt
max_retry_count	3
Bitrate	1.5 Mbps
t_{tr}	350 ms

Tabela 17 - Valores dos parâmetros do módulo *Master* no caso de estudo MAF

7.4.2 *Slave*

Parâmetro	Valor
adressS	{2,3,4,5,6,11,12,64,65,66}
t_{rt} PC1 (1)	271 bt
t_{rt} PC2 (2)	271 bt
t_{rt} PC2 (10)	271 bt
t_{rt} PC6 (6)	271 bt
t_{rt} PC3 (3)	251 bt
t_{rt} PC4 (4)	251 bt
t_{rt} PC5 (5)	251 bt
t_{rt} MM1 (10)	12 bt
t_{rt} MM1 (11)	12 bt
t_{rt} ET1 (64)	12 bt
t_{rt} PLC (65)	12 bt
t_{rt} ET1 (66)	12 bt

Tabela 18 - Valores dos parâmetros do módulo *Slave* no caso de estudo MAF

7.4.3 Meio físico *Wired*

Parâmetro	Valor
bit_rate_WR	1.5 Mbps
Prob_error_WR	0.01

Tabela 19 - Valores dos parâmetros do módulo PhyWired no caso de estudo MAF

7.4.4 Meio físico *Wireless*

Parâmetro	Valor
bit_rate_WL	2 Mbps
Prob_error_WL	0.01

Tabela 20 - Valores dos parâmetros do módulo PhyWireless no caso de estudo MAF

7.4.5 LBS

Parâmetro	Valor
t_{relay}	25 µs

Tabela 21 - Valor do parâmetro do módulo LBS no caso de estudo MAF

7.4.6 MobM

Parâmetro	Valor
TS	10
NS	1
PS	1
T_{FTMMP}	1 s
$T_{ID2mobm}$	4305 bt

Tabela 22 - Valores dos parâmetros do módulo MobM no caso de estudo MAF

7.5 O conjunto dos ciclos de mensagem observados

O conjunto dos ciclos de mensagem que ocorrem nesta topologia são descritos no Anexo A - MAF - Manufacturing Automation Fieldtrial nas páginas 142 (DP) e 145 (IP). Na Tabela 27 estão resumidas as características de todos os ciclos de mensagem do MAF. Os resultados de observação são retirados do estudo realizado em [39], onde foi concluído que as características dos ciclos de mensagem eram basicamente idênticas em todas as amostras. Na altura em que o trabalho experimental desta tese foi efectuado o MAF já não dispunha de uma rede de comunicações RFieldbus dado já incorporar novos tipos de redes de comunicação do tipo Ethernet e WiFi. Desta forma, os resultados apresentados na Tabela 27 são uma amostra (em várias observadas) dos ciclos de mensagens presentes na rede de comunicação [39].

7.6 O Analisador de rede PROFIBUS (PROFIBUS *Analyser*)

O analisador de rede PROFIBUS é um dispositivo que permite observar o funcionamento desta rede de comunicação. Para além desta funcionalidade básica, fornece também um conjunto de funções analíticas para gerar estatísticas e observar erros.

O analisador de rede PROFIBUS permite observar o tráfego do meio físico (*on-line*) ao qual está associado (*Wired*). Permite também guardar estes dados e observa-los posteriormente em *off-line* [36].

7.6.1 Âmbito do uso do PROFIBUS *Analyser* no MAF

Para observar o funcionamento temporal da rede de comunicações no caso de estudo foi utilizado um analisador de rede PROFIBUS. No caso de estudo a topologia da rede RFieldbus é constituída por três domínios de comunicação, um domínio *Wired* e dois *Wireless*. O analisador de rede está associado ao domínio com meio físico *Wired* permitindo a observação directa do comportamento temporal, no entanto nos outros dois domínios (meio físico *Wireless*) o comportamento não é observável, sendo possível inferi-lo apenas. Surgem duas situações distintas.

No caso dos ciclos de mensagem em que o *initiator* e o *responder* estão ambos no mesmo meio físico (e é aquele ao qual o analisador de rede esta associado) é possível observar todos os tempos envolvidos na transacção. Já no caso dos ciclos de mensagem em que *initiator* e *responder* estão em meios físicos diferentes (domínios diferentes) e um desses meios físicos não é observável, já só se pode observar a parte da transacção que decorre no meio físico que tem o analisador. Quanto ao resto da transacção que decorre no outro meio físico, apenas se conhece o tempo global que dura essa parte da transacção (desde que a trama do *request* deixa o meio físico *Wired* até que a trama da *response* entra no meio físico *Wired*). O conhecimento dos tempos que constituem esse tempo global depende do cálculo teórico de algumas das componentes que totalizam esse tempo, para por diferença ao tempo total, determinar as outras.

7.6.2 O que permite observar o analisador de rede PROFIBUS?

Com este analisador de rede é possível observar:

1. Duração das tramas no meio físico ao qual o analisador esta associado (C_{Lreq}, C_{Lresp}).
2. O *System Turnarround Time* de todos os ciclos de mensagem iniciados no meio físico (*Wired*) ao qual o analisador esta associado (t_{st}).
3. O *Responder Turnarround Time* dos *Responders* que se encontram no meio físico (*Wired*) ao qual o analisador esta associado.
4. O T_{ID1} e T_{ID2} inserido pelos *Masters* do meio físico ao qual o analisador esta associado (*Wired*).
5. O tempo que dura uma rotação do *Token* (T_{RR}).
6. O *Slot time* (t_{SL}) que os *Masters* associados ao meio físico onde está o analisador inserem (igual em todos).

7. Observar o *tráfego* despachado pelo *Master*.

7.7 Cálculo da duração dos ciclos de mensagem

Um ciclo de mensagem abrange a transmissão das tramas que constituem a mensagem e o período de inactividade no meio físico *idle time* que o *Master* tem que respeitar antes de poder iniciar o próximo ciclo de mensagem ou passar o *Token* [12].

7.7.1 Duração de um ciclo de mensagem com resposta/*ack*

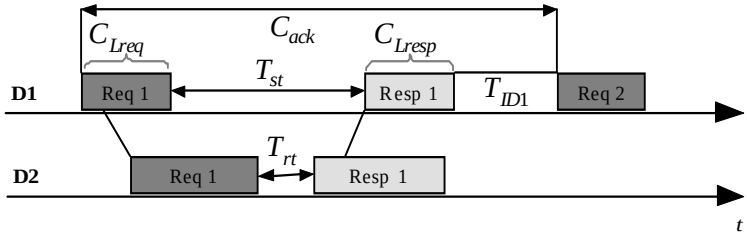


Figura 81 - Duração de um ciclo de mensagem com resposta/ack

$$C_{ack} = C^1_{Lreq} + t_{st} + C^1_{Lresp} + T^1_{ID1}$$

Equação 5

Da figura é possível observar que a duração de um ciclo de mensagem com *acknowledged/response* depende da duração das tramas de *request* e *acknowledged/response* no domínio D1, do tempo que decorre desde o fim da transmissão do *request* até ao inicio da *response* e do *idle time* que o *Master* tem que inserir (neste caso T_{ID1}).

7.7.2 Duração de um ciclo de mensagem sem resposta/*ack*

Dado que neste tipo de ciclos de mensagem o *initiator* emite o *request* e não espera por um *acknowledged/response* o cálculo da duração deste tipo de ciclo de mensagem (C_{unk}) pode ser efectuada da forma seguinte:

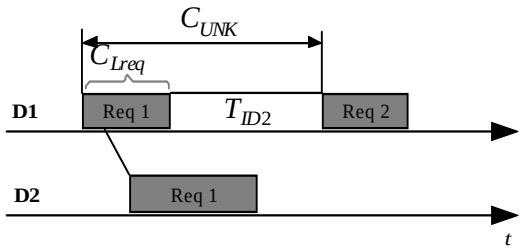


Figura 82 - Duração de um ciclo de mensagem sem resposta/ack

$$C_{unk} = C^{1}_{Lreq} + T^{1}_{ID2}$$

Equação 6

Apesar da passagem do *Token* ser uma transacção sem *acknowledge*, após a passagem do mesmo o *Master* insere um *idle time* T_{ID1}.

7.8 Resultados obtidos

A topologia do caso de estudo tem dois meios físicos diferentes (*Wired* e *Wireless*) com taxas de transmissão de dados diferentes. A duração de um bit é diferente em cada tipo de meio físico. Nos resultados apresentados, todos os tempos em bt, são *bit time* do meio físico *Wired*. São apresentados os resultados obtidos em ciclos de mensagem que decorrem apenas no domínio de comunicação *Wired*. S1 no caso da comunicação com um dispositivo RFieldbus, S4 e S5 no caso da comunicação com dispositivos PROFIBUS. São também confrontados os resultados obtidos para os ciclos de mensagem S2 e S3 que decorrem em dois domínios de comunicação *Wired* e*Wireless*. Posteriormente são confrontados os resultados obtidos nos parâmetros *Slot time* e *Idle time* (T_{ID1}). Através da análise desses resultados foi possível validar a eficácia do simulador implementado.

7.8.1 Ciclo de mensagem S1

Este ciclo de mensagem está associado à aplicação *DP Sound Slave* e é trocado entre o *Master* PC1 e o *Slave* PC6. É um ciclo de comunicação DP e ocorre no domínio de comunicação *Wired*.

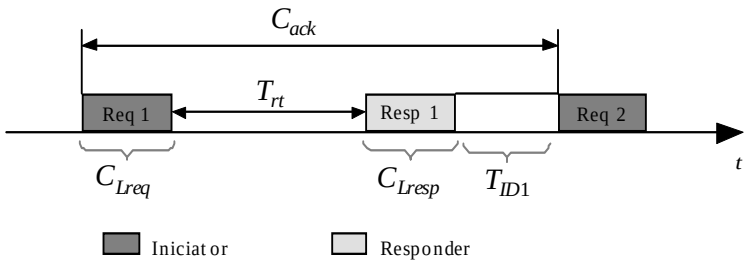


Figura 83 - Ciclo de mensagem com iniciator e responder no meio físico WR1

Simulação:

$$C_{Lreq}^{wr} = 121 \text{ bt} \qquad C_{Lresp}^{wr} = 121 \text{ bt}$$

$$C_{ack}^{wr} = 909 \text{ bt}$$

$$C_{ack\_withoutidletime}^{wr} = 513 \text{ bt}$$

$$T_{ID1}^{wr} = 396 \text{ bt}$$

$$T_{st}^{wr} = T_{rt}^{wr} = 271 \text{ bt}$$

Resultados experimentais:

$$C_{Lreq}^{wr} = 121 \text{ bt} \qquad C_{Lresp}^{wr} = 121 \text{ bt}$$

$$C_{ack}^{wr} = 916 \text{ bt}$$

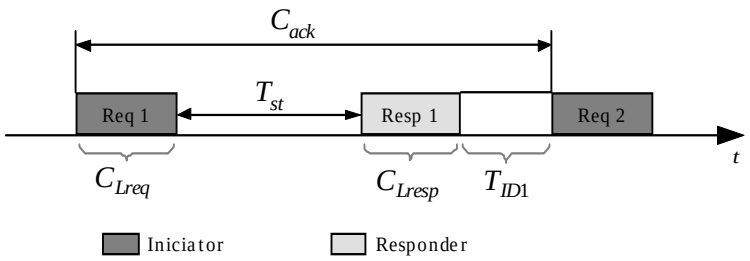
$$C_{ack\_withoutidletime}^{wr} = 519 \text{ bt}$$

$$T_{ID1}^{wr} = 397 \text{ bt}$$

$$T_{st}^{wr} = 271 \text{ bt}$$

7.8.2 Ciclo de mensagem S2

Este é o ciclo de mensagem está associado à *Robuter Slave Connection* e é trocado entre o *Master* PC1 e o *Slave* PC3. É um ciclo de comunicação DP e ocorre nos domínios de comunicação *Wired* e *Wireless*.



Onde:

$$T_{st} = T_{relay} + C_{Lreq}^{wl} + T_{rt}^{wl} + C_{Lresp}^{wl} + T_{relay}$$

Equação 7

Simulação:

C_{Lreq}^{wr} = 121 bt C_{Lresp}^{wr} = 121 bt

C_{Lreq}^{wl} = 216 bt C_{Lresp}^{wl} = 216 bt

T_{relay} = 37,5 bt T_{rt}^{wl} = 251 bt

T_{st} = 758 bt (Equação 7)

$C_{ack\_withoutidletime}^{wr}$ = 1000 bt

T_{ID1}^{wr} = 396 bt

C_{ack}^{wr} = 1396 bt

Resultados experimentais:

C_{Lreq}^{wr} = 121 bt C_{Lresp}^{wr} = 121 bt

C_{ack}^{wr} = 1400 bt

$C_{ack\_withoutidletime}^{wr}$ = 1004 bt

T_{ID1}^{wr} = 396 bt

$$T_{st}^{wr} = 758 \text{ bt}$$

7.8.3 Ciclo de mensagem S3

Este é o ciclo de mensagem está associado à *RC2 Slave Connection* e é trocado entre o *Master* PC1 e o *Slave* PC4. É um ciclo de comunicação DP e ocorre nos domínios de comunicação *Wired* e *Wireless*.

Simulação:

$$C_{Lreq}^{wr} = 121 \text{ bt} \qquad C_{Lresp}^{wr} = 121 \text{ bt}$$

$$C_{Lreq}^{wl} = 216 \text{ bt} \qquad C_{Lresp}^{wl} = 216 \text{ bt}$$

$$T_{relay} = 37{,}5 \text{ bt} \qquad T_{rt}^{wl} = 251 \text{ bt}$$

$$T_{st} = 758 \text{ bt} \quad \text{(Equação 7)}$$

$$C_{ack\_withoutidletime}^{wr} = 1000 \text{ bt}$$

$$T_{ID1}^{wr} = 396 \text{ bt}$$

$$C_{ack}^{wr} = 1396 \text{ bt}$$

Resultados experimentais:

$$C_{Lreq}^{wr} = 121 \text{ bt} \qquad C_{Lresp}^{wr} = 121 \text{ bt}$$

$$C_{ack}^{wr} = 1400 \text{ bt}$$

$$C_{ack\_withoutidletime}^{wr} = 1004 \text{ bt}$$

$$T_{ID1}^{wr} = 396 \text{ bt}$$

$$T_{st}^{wr} = 758 \text{ bt}$$

7.8.4 Ciclo de mensagem S4

Este ciclo de mensagem está associado à comunicação entre o *Master* PC1 e um dos *drives* dos motores (*MicroMaster*) dos tapetes transportadores, presente na rede de comunicação como *Slave* com o endereço 10. É um ciclo de comunicação DP e ocorre no domínio de comunicação *Wired*.

Simulação:

$$C_{Lreq}^{wr} = 231 \text{ bt} \qquad C_{Lresp}^{wr} = 231 \text{ bt}$$

$$C_{ack}^{wr} = 870 \text{ bt}$$

$$C_{ack\_withoutidletime}^{wr} = 474 \text{ bt}$$

$$T_{ID1}^{wr} = 396 \text{ bt}$$

$$T_{st}^{wr} = T_{rt}^{wr} = 12 \text{ bt}$$

Resultados experimentais:

$$C_{Lreq}^{wr} = 231 \text{ bt} \qquad C_{Lresp}^{wr} = 231 \text{ bt}$$

$$C_{ack}^{wr} = 875 \text{ bt}$$

$$C_{ack\_withoutidletime}^{wr} = 478 \text{ bt}$$

$$T_{ID1}^{wr} = 397 \text{ bt}$$

$$T_{st}^{wr} = 12 \text{ bt}$$

7.8.5 Ciclo de mensagem S5

Este ciclo de mensagem está associado à comunicação entre o *Master* PC1 e o outro *drive* dos motores (*MicroMaster*) dos tapetes transportadores, presente na rede de comunicação como *Slave* com o endereço 11. É um ciclo de comunicação DP e ocorre no domínio de comunicação *Wired*.

Simulação:

$$C_{Lreq}^{wr} = 231 \text{ bt} \qquad C_{Lresp}^{wr} = 231 \text{ bt}$$

$$C_{ack}^{wr} = 870 \text{ bt}$$

$$C^{wr}_{ack\_withoutidletime} = 474 \text{ bt}$$

$$T^{wr}_{ID1} = 396 \text{ bt}$$

$$T^{wr}_{st} = T^{wr}_{rt} = 12 \text{ bt}$$

Resultados experimentais:

$$C^{wr}_{Lreq} = 231 \text{ bt} \qquad C^{wr}_{Lresp} = 231 \text{ bt}$$

$$C^{wr}_{ack} = 875 \text{ bt}$$

$$C^{wr}_{ack\_withoutidletime} = 478 \text{ bt}$$

$$T^{wr}_{ID1} = 397 \text{ bt}$$

$$T^{wr}_{st} = 12 \text{ bt}$$

7.8.6 Passagem do *Token*

Simulação:

$$C^{wr}_{Ltoken} = 33 \text{ bt}$$

$$T^{wr}_{\text{st\_token}} = T^{wr}_{ID1} = 396 \text{ bt}$$

Resultados experimentais:

$$C^{wr}_{Ltoken} = 33 \text{ bt}$$

$$T^{wr}_{\text{st\_token}} = T^{wr}_{ID1} = 345 \text{ bt}$$

7.8.7 Parâmetro *Slot time*

Simulação:

$$T^{wr}_{SL} = 2950 \text{ bt}$$

Resultados experimentais:

$$T_{SL}^{wr} = [2915, 2965] \text{ bt}$$

7.8.8 Parâmetro *Idle Time* T_{ID1}

Simulação:

$$T_{ID1}^{wr} = 396 \text{ bt}$$

Resultados experimentais:

$$T_{ID1}^{wr} = 396 \text{ bt}$$

7.9 Avaliação

Os resultados apresentados na secção anterior demonstram que os tempos simulados para a circulação das tramas na rede de comunicação se aproximam dos observados na aplicação de teste MAF.

Não são apresentados os valores simulados e observados na aplicação de teste do MAF para cinco ciclos de mensagem dado terem características idênticas a outros ciclos de mensagem.

Como se pode observar em ambas as tabelas os tempos obtidos por simulação e observados são muito próximos. No caso da transmissão nos meios físicos os tempos simulados e observados são idênticos dado não existir variação nos valores observados. Apenas no parâmetro *Slot Time* os valores observados variam num intervalo e por esse motivo o valor do parâmetro foi ajustado para o ponto central desse intervalo.

Desta forma, nota-se que o comportamento simulado equivale ao observado na aplicação de teste MAF. Com esta ferramenta disponível é assim possível analisar o comportamento temporal de uma rede de comunicação RFieldbus com diferentes topologias, parâmetros e ciclos de mensagens. A vantagem da obtenção desta informação sobre o comportamento do sistema usando o LLRS é poder realizar a análise antes/sem a implementação do sistema real como no caso do MAF. Este é aliás um paradigma associado à simulação.

	C^{wr}_{Lreq}	T^{wr}_{rt}	T^{wr}_{st}	C^{wr}_{Lresp}	T^{wr}_{ID1}	C^{wr}_{ack}	
S1	121	277	277	121	397	916	S
	121	271	271	121	396	909	O
S2	121	X	758	121	396	1400	S
	121	X	758	121	396	1396	O
S3	121	X	758	121	396	1400	S
	121	X	758	121	396	1396	O
S4	231	12	12	231	397	875	S
	231	12	12	231	396	870	O
S5	231	12	12	231	397	875	S
	231	12	12	231	396	870	O

S – Simulado O – Observado X - não se aplica[19]

Tabela 23 - Valores simulados e observados para as message streams do MAF

	C^{wr}_{Ltoken}	$T^{wr}_{\text{st\_token}}$	T^{wr}_{SL}	T^{wr}_{ID1}	
Passagem do token	33	345	X	345	S
	33	396	X	396	O
Slot Time	X	X	[2915, 2965]	--	S
	X	X	2950	--	O
Idle Time	X	X	X	396	S
	X	X	X	396	O

S – Simulado O – Observado X – não se aplica

Tabela 24 - Outros valores simulados/observados no MAF

7.10 Sumário

Neste capítulo foram apresentados e validados os resultados obtidos por simulação no LLRS do comportamento temporal da rede do MAF. A validação desses resultados foi feita por confrontação com observações feitas na aplicação de teste MAF.

Foram também apresentados os dados de entrada do simulador, a topologia de rede a simular, os parâmetros, os ciclos de mensagem, explicadas as fórmulas de cálculo da duração dos ciclos de mensagem com/sem resposta e feitas considerações acerca de limitações existentes na observação do comportamento desta topologia com o analisador de rede.

[19] - *Message Stream* envolvendo comunicação nos dois meios físicos, apenas é conhecido t_{st}.

Capítulo 8

Conclusões e trabalho futuro

8.1 Sumário da Dissertação

As redes de comunicação têm uma importância vital no desempenho dos sistemas distribuídos com requisitos de tempo real. Dado que estas permitem a comunicação (sincronização, transferência e recepção de comandos/informação) entre os diferentes componentes cooperantes do sistema, a caracterização do seu comportamento temporal é importante, pois este influenciará o comportamento do sistema distribuído e consequentemente o cumprimento ou não dos seus requisitos de tempo real.

Esta dissertação apresentou um estudo, efectuado à arquitectura da rede de comunicação industrial RFieldbus. O estudo incidiu sobre o comportamento temporal das camadas baixas (1 e 2) desta arquitectura.

8.1.1 Estudo da arquitectura RFieldbus

A primeira fase deste trabalho consistiu no estudo dos requisitos colocados a uma rede de comunicação de tempo real. É possível salientar a necessidade de baixa latência para as comunicações prioritárias, o âmbito de aplicação da rede de comunicação, os tipos de tráfego que a rede pode transportar, a previsibilidade do seu funcionamento, e se apresenta uma arquitectura genérica englobando um largo espectro de aplicações ou pelo contrário é restrita a uma aplicação especifica.

Passada esta fase inicial, o estudo focou na arquitectura do PROFIBUS-DP que é a arquitectura na qual a rede RFieldbus se baseia. Da arquitectura do PROFIBUS-DP foram estudadas exaustivamente as camadas 1 e 2 porque são parte integrante da arquitectura RFieldbus. No capítulo 2 são apresentadas as interfaces e protocolos dessas camadas. A camada 2 pela sua relevância na definição do comportamento do RFieldbus mereceu uma atenção especial. O controlador da camada 2 rege o seu comportamento por uma máquina de estados (pág. 27). Nos diferentes estados são implementadas as funcionalidades de transmissão (pág. 22), recepção (pág. 21) do token e os algoritmos de escalonamento de mensagens (pág. 24). Também foram estudados os formatos das tramas utilizadas em ambos os meios físicos (*Wired* e *Wireless*), o esquema de endereçamento utilizado, os parâmetros utilizados (pág. 20), a gestão do ciclo de mensagem (pág. 18), e as filas onde são armazenadas as mensagens aguardando transmissão (pág. 54).

O capítulo 2 apresenta também uma descrição das camadas superiores que constituem o perfil de aplicação PROFIBUS-DP (pág. 30)

Posteriormente foi analisada a arquitectura RFieldbus (capítulo 3), focando nas suas novas funcionalidades: a possibilidade de comunicação num meio físico *Wireless*, a possibilidade dos dispositivos *Wireless* serem móveis e a integração de tráfego multimédia. Foram assim apresentadas as restantes camadas da arquitectura RFieldbus, nomeadamente o DP/IP *Dispatcher*, o DP *Mapper*, o IP ACS e o IP *Mapper* (pág. 54) as novas estações introduzidas na arquitectura RFieldbus: a RFieldbus *Link Station*, a RFieldbus *Base Station*, a RFieldbus *Link Base Station* (pág. 38) e o RFieldbus *Mobility Master* (3.3.2) e as topologias passíveis de serem implementadas com esta rede de comunicação (pág. 42).

No capítulo 4 foram apresentados conceitos da teoria da simulação por eventos discretos e a ferramenta OMNeT++. Foram estudados os módulos simples e compostos, a hierarquia dos módulos (pág. 62), os parâmetros (pág. 64), as mensagens, *gates* e ligações (pág. 63). Os módulos compostos que servem apenas como blocos construtivos enquanto nos módulos simples são implementados os algoritmos de simulação. Relativamente aos módulos simples são descritos os dois paradigmas de programação (*activity* e *handlemessage*) (pág. 64) e explicadas as principais funções membro virtuais (*receive*, *wait*, *send*, etc) (pág. 64 e 65). É apresentado o esquema de desenvolvimento e execução do simulador.

Após este estudo havia que definir um modelo de simulação para as camadas baixas do RFieldbus, que reflectisse os principais atrasos sofridos pelas tramas dos ciclos de mensagens da rede a simular.

8.1.2 Modelo definido para as camadas mais baixas

O objectivo foi o de determinar quais as funcionalidades que originavam os principais atrasos sofridos pelas tramas constituintes de cada ciclo de mensagem ao longo do seu trajecto.

No caso da estação *Master* os atrasos começam nas filas de mensagens (pág. 54) da FDL (camada 2) onde cada mensagem é depositada após requisição do serviço disponibilizado na interface com a camada superior (pág. 13). Posteriormente cada mensagem aguarda um determinado tempo até ser escalonada, dependente do tempo que a estação demora a receber o token (algoritmo de passagem do *token* PROFIBUS, baseado no *timed token*) (pág. 22 e 21), da classe de tráfego a que pertence (pág. 54) e da sua posição na fila de mensagens (algoritmo de escalonamento de mensagens) (pág. 24). O *responder*, a partir do momento em que termina a recepção da trama demora um determinado tempo até iniciar a transmissão da correspondente resposta (t_{rt}) (pág. 73).

No que respeita ao meio físico, a transmissão de uma trama dura um período de tempo determinado pela taxa de transferência de dados desse meio físico, pelo tamanho da trama e pelos erros ocorridos na transmissão da mesma nesse meio físico (pág. 74).

Os *Intermediate Systems* (IS) ao iniciarem a recepção de uma trama demoram um período de tempo até iniciar a sua retransmissão para o outro meio físico (pág. 74). Consoante o tipo de IS o seu comportamento varia. Uma *Link Station* (LS) retransmite as tramas recebidas de um dos meios físicos para o outro (com a latência atrás referida). Já uma *Base Station* BS estrutura as comunicações num domínio de comunicação *Wireless* dado que recebe todas as comunicações desse meio físico retransmitindo-as posteriormente com o sinal reforçado. A *Link Base Station* LBS fornece ambas as funcionalidades numa só estação. As consequências temporais dos diferentes comportamentos dos IS são no caso da retransmissão de tramas o atraso introduzido (Figura 24) e relativamente ao modo de funcionamento (pág. 76) os diferentes caminhos definidos para a trama (diferença entre domínios *Wireless* estruturados e não estruturados).

Relativamente à gestão da mobilidade, a *Intra-Cell Mobility* (pág. 46) é transparente em termos operacionais. No caso da *Inter-Cell Mobility* (pág. 46) o atraso provocado por este procedimento é resultado da passagem do *token* para o *Mobility Master* (para que este dê inicio ao

procedimento de gestão de mobilidade), da duração do procedimento de gestão da mobilidade e da passagem do *token* pelo *Mobility Master* para a estação activa seguinte.

8.1.3 Implementação do modelo em OMNeT++

O modelo de simulação desenvolvido foi implementado na ferramenta de simulação de sistemas de eventos discretos OMNeT++. Foram implementados os seguintes módulos: *Master* (pág. 81), *Slave* (pág. 92), *Mobility Master* (pág. 99), Meio físico *Wired* (pág. 100), Meio físico *Wireless* (pág. 102), *Link Station* (pág. 104), *Base Station* (pág. 104), *Link Base Station* (pág. 107). Os componentes representativos dos ES (*Master* e *Slave*) foram implementados como módulos compostos. Os restantes foram implementados como módulos simples.

Para cada módulo composto é apresentada a sua estrutura. Para cada módulo simples é descrita a sua funcionalidade recorrendo a um fluxograma e são apresentados os seus parâmetros.

É também apresentada a mensagem (pág. 80) definida em OMNeT++ no âmbito do simulador *Lower Layer RFieldbus Simulator* LLRS.

O módulo *Master* integra três módulos: o Phy que implementa a camada física, o módulo FDL e o módulo FDL *User* que é um gerador de tráfego ao nível da interface da FDL. O módulo FDL contém dois módulos que implementam diferentes funcionalidades. O módulo *Message Queues* que implementa as duas filas de mensagens onde são guardas as tramas que aguardam transmissão após serem depositadas por requisição do serviço na interface da camada FDL e o módulo MAC que implementa a máquina de estados do controlador da FDL, o escalonamento das mensagens, o cálculo do T_{TH} disponível, o processamento de cada ciclo de mensagem e a transmissão do *Token.*

8.1.4 Teste e validação do simulador

Para testar e validar o LLRS foi simulada a aplicação de teste MAF. Para o efeito foi definida no simulador a topologia do MAF (pág. 110), utilizados os parâmetros que caracterizam cada elemento da rede RFieldbus (pág. 110) e geradas todas as *message streams*. Os valores obtidos por simulação foram comparados com os valores observados através do PROFIBUS *Analyser* na aplicação de teste MAF.

Como referido, o PROFIBUS *Analyser* permite fazer observações ao meio físico *Wired*. Foram observados ciclos de mensagem que envolviam *responders* com placas RFieldbus *Wired*, com placas RFieldbus *Wireless* e também dispositivos equipados com interfaces de comunicação apenas com a funcionalidade PROFIBUS.

Para diferentes tipos de ciclos de mensagem foram medidos, a duração do *request* C_{Lreq}, a duração da *response* C_{Lresp}, o *responder turnaround time* t_{rt}, o *system turnaround time* t_{st}, o idle time t_{ID1} e a duração do ciclo de mensagem C_{ack}. Também foram confrontados os parâmetros *Slot time* t_{SL} e *Idle time* t_{ID1} (pag. 115).

8.2 Conclusões

Com o trabalho apresentado nesta dissertação foram cumpridos os objectivos iniciais de analisar o comportamento temporal das camadas mais baixas da rede de comunicação RFieldbus e

de implementar um modelo de simulação para permitir o estudo do comportamento desta rede com diversos componentes, parâmetros e topologias.

Para além disso, foi também possível durante a execução do trabalho obter um melhor conhecimento sobre das redes de comunicação RFieldbus e PROFIBUS (na qual se baseia a rede RFieldbus) e das metodologias e ferramentas de simulação de eventos discretos.

8.2.1 PROFIBUS

A arquitectura desta rede de comunicação é muito abrangente, oferece uma vasta gama de possibilidades de comunicação, das quais se destacam os dois perfis de comunicação PROFIBUS-FMS e PROFIBUS-DP e os diversos perfis de aplicação. Características como a possibilidade de entrada e saída de estações na rede durante a comunicação, taxas de transferência de dados até 12 Mbps, redes com extensão física até 1200 m, tramas com tamanhos de 1 até 255 bytes, diferentes classes de tráfego, o algoritmo de controlo de acesso ao meio físico, o algoritmo de escalonamento de mensagens e os serviços de transferência de dados da camada 2 que permitem a transmissão de dados em *Uni*, *Multi* ou *Broadcast*, são algumas da inúmeras possibilidades oferecidas ao utilizador final. Estas características fazem deste *serial fieldbus* líder nas áreas de automação e controlo de processos, com uma cota de mercado acima dos 20% sendo o mais importante a nível europeu e um dos mais importantes a nível mundial. O âmbito de aplicação deste *fieldbus* é ao nível de célula (*cell level*) e de campo (*Field level*).

Dado que se encontra normalizado nas normas *Fieldbus Standards* EN 50170 [6] e IEC 61158, garante abertura e estabilidade para os utilizadores e vendedores.

Em PROFIBUS, o algoritmo de controlo do acesso ao meio físico é baseado no *Timed Token* que é uma solução muito eficiente para sistemas com requisitos de tempo real pois permite a comunicação em tempo real para o tráfego prioritário (síncrono). A vantagem do algoritmo utilizado em PROFIBUS é que a transmissão das diferentes classes de tráfego define prioridades através do algoritmo de escalonamento de mensagens que garante um intervalo de tempo para a transmissão do tráfego de prioridade elevada de cada estação, sendo o restante tráfego transmitido no tempo remanescente.

8.2.2 RFieldbus

A arquitectura RFieldbus é uma extensão da arquitectura PROFIBUS. Os dispositivos RFieldbus são totalmente compatíveis com os PROFIBUS podendo coexistir na mesma rede. No entanto a comunicação entre dispositivos RFieldbus e PROFIBUS só é possível quando os dispositivos RFieldbus utilizam apenas as funcionalidades nativas PROFIBUS. Por este motivo, dispositivos RFieldbus podem por exemplo, ser utilizados para estender funcionalidades de redes PROFIBUS instaladas e em funcionamento.

Esta nova arquitectura baseia-se no PROFIBUS-DP que é uma rede largamente utilizada no seu mercado, cuja solução técnica já deu provas quanto ao desempenho e fiabilidade e representa um grande avanço no mercado das redes de comunicação industrial. Adiciona as possibilidades de transferir tráfego multimédia na rede, de as comunicações decorrerem em meio sem fios, com segurança, em que os dispositivos podem estar numa posição fixa ou serem móveis. A solução de mobilidade do RFieldbus permite mesmo uma mobilidade sem grandes restrições devido à inclusão das estações base (BS e LBS) que criam domínios estruturados de comunicação sem fios.

Sendo o PROFIBUS a base da arquitectura RFieldbus mantém-se um eficiente algoritmo de escalonamento do tráfego que garante o cumprimento das restrições de tempo real do tráfego prioritário. Na restante largura de banda pode agora ser transferido tráfego multimédia ao qual pode ser garantida alguma qualidade de serviço.

Este alargamento de funcionalidades do PROFIBUS representa um esforço comum a muitas redes de comunicação no sentido de alargar as possibilidades fornecidas pela arquitectura, garantindo desta forma a sobrevivência da rede no mercado. Um exemplo é a *Industrial Ethernet*, que tem feito um grande esforço para garantir o cumprimento dos requisitos temporais do tráfego de controlo de forma a eliminar a necessidade de uma rede específica para que se cumpram os requisitos temporais das aplicações.

8.2.3 Simulação de DES e OMNeT++

A simulação em computador consiste na implementação de um modelo de um sistema real ou proposto para que o comportamento do sistema possa ser estudado em condições específicas. A simulação é uma técnica que tem vantagens e desvantagens. As principais desvantagens são o modelo nunca ser absolutamente exacto, ter que ser validado para que exista uma garantia de que representa a realidade, normalmente ser construído para gerar informação sobre um ou alguns parâmetros do sistema real e não a sua totalidade.

A principal vantagem é a possibilidade de fazer testes que seriam impossíveis de fazer num sistema real (por diversos motivos: custos, impossibilidades físicas, etc).

A simulação de sistemas de eventos discretos é uma das ferramentas disponíveis para a análise do comportamento temporal de redes de comunicação. Os resultados obtidos por este método são normalmente relativos a comportamentos médios desse sistema. Já a análise do pior caso permite estabelece um patamar mínimo de confiança na resposta temporal do sistema.

No caso do simulador implementado o modelo definido serve para analisar o comportamento temporal de uma rede RFieldbus, considerando os diferentes elementos (sistemas finais, intermédios e meios físicos), a topologia da rede, o conjunto dos parâmetros que caracterizam cada um desses elementos e dos ciclos de mensagem existentes. Com este simulador o utilizador pode testar diferentes topologias, alterar parâmetros e o tráfego da rede.

Os resultados de simulação obtidos no LLRS para o caso de estudo respeitam a valores médios observados na aplicação de teste RFieldbus *Manufacturing Automation Fieldtrial*. Estes resultados são muito próximos dos que foram observados na aplicação MAF, o que permitiu validar o modelo de simulação. No entanto dado que os resultados obtidos por simulação respeitam ao comportamento mais provável da rede (médio) estes não podem ser utilizados para estabelecer um patamar de confiança quanto à resposta temporal da rede de comunicações.

Desta forma as análises do comportamento médio e de pior caso são complementares. Se por um lado a análise do pior caso permite estabelecer um patamar mínimo de confiança na resposta temporal do sistema, por outro é fortemente pessimista quando comparada com a resposta mais provável de ocorrer na rede de comunicação (caso médio). Inversamente a simulação DES permite estimar comportamentos mais próximos dos que são normalmente observados na rede de comunicação mas não permite estabelecer um patamar mínimo de confiança na resposta temporal do sistema.

Das variáveis envolvidas no modelo de simulação das camadas baixas RFieldbus, algumas podem ser consideradas determinísticas (ex. duração das tramas nos meios físicos). Outras no entanto, são aleatórias e uma extensão deste trabalho aconselharia a sua classificação estatística (ex. tempo de resposta do *responder*, parâmetro *Slot Time*, etc).

Inicialmente, foi tentada uma implementação do modelo de simulação definido recorrendo à ferramenta de simulação ns-2. A adequabilidade desta ferramenta para a implementação de um modelo com esta especificidade é baixa, dado esta ser uma ferramenta dedicada à simulação de redes IP. Ou seja os módulos oferecidos formam uma base de trabalho para a simulação deste tipo de redes. A inviabilidade desta tentativa resultou na escolha do OMNeT++ que se revelou muito mais simples e intuitivo.

Em OMNeT++ os conceitos de simulação de sistemas de eventos discretos estão implementados de uma forma muito explícita e com um grau de abstracção superior relativamente à ferramenta de simulação.

A experiência na sua utilização permite prever um bom futuro a esta ferramenta pela sua simplicidade[20], pela facto de ser *freeware* para a comunidade académica e pelo facto de começarem a ser disponibilizados mais modelos de redes de comunicação. Recentemente foi disponibilizada uma base de trabalho para a análise da performance dos protocolos de comunicação e das configurações das redes de comunicação industriais, *Ethernet*, *ControlNet* e *DeviceNet*. Brevemente será também integrada a rede *LonTalk* [40].

8.2.4 Validação dos objectivos do trabalho

O modelo de simulação definido está validado e representa a operação temporal das camadas mais baixas da rede de comunicação RFieldbus. Isto implica que o modelo definido para a simulação do comportamento temporal de ambos os meios físicos *Wired* e *Wireless* gera um comportamento (tempo de transmissão) similar. O modelo definido para cada uma das estações *Intermediate Systems* IS (LS, BS e LBS) também representa o comportamento de cada uma das estações intermédias. Relativamente aos *End Systems* ES (*Master* e *Slave*) os modelos definidos representam o comportamento de cada uma das estações finais. Também o modelo definido e implementado para o procedimento de gestão da mobilidade nomeadamente a estação responsável, o Mobility Master, reflectem o comportamento observado.

A implementação do modelo no LLRS conduziu a um simulador que no caso de estudo originou um comportamento temporal muito próximo do observado no MAF (pág. 121). As diferenças devem-se ao facto do modelo não ser perfeito, mas sim uma aproximação aos componentes reais. Outro factor é a aplicação de teste (MAF) conter dispositivos que são protótipos e cujo *firmware* não sofreu um extensivo processo de validação. Isto implicou os atrasos adicionais observados que estavam a ser introduzidos pelas placas Rfieldbus, quer como *iniciator*, quer como *responder*.

[20] Relativa a outros. Trata-se de um ambiente de simulação de sistemas de eventos discretos orientado a objectos.

8.3 Trabalho futuro

O trabalho apresentado nesta dissertação permitiu obter uma ferramenta adequada para a análise do comportamento temporal de redes RFieldbus. No entanto, foram também identificadas novas áreas de trabalho futuro que se podem aplicar aos resultados obtidos. Entre estas destacam-se:

- Observar simultaneamente ambos os meios físicos *Wired* e *Wireless*.
- Modelar as camadas superiores e os estados transitórios da rede de comunicação industrial Rfielsbus.
- Efectuar este estudo em outras redes de comunicação industriais.

8.3.1 Observar simultaneamente ambos os meios físicos *Wired* e *Wireless*.

A validação dos resultados obtidos por simulação foi feita pela observação do comportamento temporal das tramas no meio físico *Wired*, visto no MAF apenas estar disponível um analisador de rede PROFIBUS.

Um analisador para o meio físico *Wireless* permitiria observar totalmente o comportamento temporal da rede RFieldbus. Desta forma seria possível verificar o modelo e validar o comportamento temporal do meio físico *Wireless*, das estações ES a ele ligadas e dos IS (necessidade de sincronizar os relógios de ambos os analisadores de rede para permitir medir as latências na retransmissão das tramas entre os dois domínios).

8.3.2 Modelar as camadas superiores e os estados transitórios

No caso da estação *Master* seria interessante criar um modelo de simulação das camadas DP/IP *Dispatcher*, DP *Mapper*, IP ACS e IP *Mapper*, permitindo fazer o estudo do comportamento temporal global da arquitectura da rede de comunicação RFieldbus. De igual forma, o modelo definido poderia ser complementado com as funcionalidades da camada FDL de inicialização da rede e de alterações dinâmicas à topologia da rede (estados transitórios de operação da rede de comunicação).

Este trabalho não se enquadrava no âmbito desta dissertação, mas seria um complemento interessante para analisar o comportamento temporal do tráfego gerado entre camadas de aplicação nos parceiros de comunicação, durante todo o período de operação da mesma.

8.3.3 Efectuar este estudo em outras redes de comunicação industriais.

As camadas mais baixas (1 e 2) das redes de comunicação industriais têm uma importância fulcral no comportamento da rede de comunicação. A extensão do estudo apresentado nesta dissertação a outras redes de comunicação industriais permitiria estabelecer comparações entre elas, no que respeita a parâmetros de desempenho e previsibilidade.

Referências

[1] Rockwell Automation, Inc., "Making Sense of e-Manufacturing: a Roadmap for Manufacturers", Cleveland, Ohio, USA. Industry White Paper, 2002.

[2] Tanenbaum A., Distributed Systems: Principles and Paradigms, Prentice Hall, 1992.

[3] Stankovic J., “Real-Time computing”. Byte, pág 155-162, Agosto 1992.

[4] Mashford K., “Next generation manufacturing”. IET Manufacturing Engineer, vol. 82(6), pág 31-4, Dezembro 2003.

[5] PROFIBUS Technical Description. Disponível na Internet: http://www.profibus.com/pb/technology/description/. Último acesso em Setembro de 2007

[6] CENELEC, EN 50170 - General Purpose Field Communication System. Volume 2 - PROFIBUS, 1996.

[7] IEC, IEC 61158 - Digital data communications for measurement and control Fieldbus for use in industrial control systems, 2003.

[8] CENELEC, EN50170 - General Purpose Field Communication System, Volume 2 - Part 5-2 Application Layer Service Definition, 1996

[9] CENELEC, EN50170 - General Purpose Field Communication System, Volume 2 - Part 6-2 Application Layer Protocol Specification, 1996

[10] CENELEC, EN50170 - General Purpose Field Communication System, Volume 2 - Part 8-2 User Specifications, 1996

[11] CENELEC, EN50170 - General Purpose Field Communication System, Volume 2 - Part 4-2 Data Link Layer Protocol Specification, 1996

[12] Alves M., “Real-Time Communications over Hybrid Wired/Wireless PROFIBUS-Based Networks”, Tese de Doutoramento. Faculdade de Engenharia da Universidade do Porto, 2003.

[13] Alves M. et al., Deliverable D1.3 - General System Architecture of the RFieldbus. Projecto Rfieldbus (IST-1999-11316), 2000.

[14] The Network Simulator ns-2. Disponível na Internet: http://nsnam.isi.edu/nsnam/index.php/Main_Page. Último acesso em Setembro de 2007.

[15] OMNeT++ Discrete Event Simulation System. Disponível na Internet: http://www.omnetpp.org. Último acesso em Setembro de 2007.

[16] Alves M., Brandão V., Specification of the Manufacturing Automation Field Trial. Relatório Técnico, Projecto Rfieldbus (IST-1999-11316), 2001.

[17] PROFIBUS International. Disponível na Internet: http://www.profibus.com. Último acesso em Setembro de 2007.

[18] EIA/TIA, RS-485 Differential Data Transmission System Basics, 1998

[19] CENELEC, EN50170 - General Purpose Field Communication System, Volume 2 - Part 3-2 Data Link Layer Service Definition, 1996.

[20] CENELEC, EN50170 - General Purpose Field Communication System, Volume 2 - Part 1-2 General Description of EN 50170 volume 2, 1996.

[21] Alves M. et al., Dependability Aspects in Profibus Layers 1/2. Brief Technical Report, Projecto Rfieldbus (IST-1999-11316), 2000.

[22] CENELEC, EN50170 - General Purpose Field Communication System, Volume 2 - Part 2-2 Physical Layer Specification and Service Definition, 1996.

[23] Stanek J., Introduction to RS 422 & RS 485. Disponível na Internet: http://www.hw-server.com/ docs/rs485.html. Último acesso em Setembro de 2007.

[24] Grow R. A timed token protocol for local area networks, em Proceedings of Electro 1982, Token Access Protocols, El Segundo, CA, EUA, Maio 1982.

[25] Monforte S.; et al, Main Characteristics of the PROFIBUS Data Link Layer. Relatório Técnico. Projecto Rfieldbus (IST-1999-11316), 2000.

[26] Tovar E., "Supporting Real-Time Communications with Standard Factory-Floor Networks", Tese de Doutoramento. Faculdade de Engenharia da Universidade do Porto, 1999.

[27] Adamczyk H. et al., Deliverable D2.1.1 - Physical Layer Specification - Part 2 Protocol & Operational Characteristics Definition. Projecto Rfieldbus (IST-1999-11316), 2001.

[28] Adamczyk H. et al., Deliverable D2.1.1 - Physical Layer Specification - Part 1 Service Definition. Projecto Rfieldbus (IST-1999-11316), 2001.

[29] Adamczyk H. et al., Deliverable D2.2.1 Data Link Layer Specification, Service Part (Part 1). Projecto Rfieldbus (IST-1999-11316), 2001.

[30] Adamczyk H. et al., Deliverable D2.2.1 Data Link Layer Specification, Protocol Part (Part 2). Projecto Rfieldbus (IST-1999-11316), 2001.

[31] Alves M. et al., Deliverable D3.1 Higher Layer Specification. Projecto Rfieldbus (IST-1999-11316), 2001.

[32] Varga A., OMNeT++ User Manual. Disponível na Internet: http://www.omnetpp.org/doc/ manual/usman.html. Último acesso em Setembro de 2007.

[33] Ball P., Introduction to Discrete Event Simulation, Fevereiro de 2001. Disponível na Internet: http://www.dmem.strath.ac.uk/~pball/simulation/simulate.html. Último acesso em Setembro de 2007.

[34] Cassandras C., Lafortune S. , Introduction to Discrete Event Systems, Kluwer Academic Press, 1999.

[35] Varga A., OMNeT++ API Reference. Disponível na Internet: http://www.omnetpp.org/doc/ api/main.html. Último acesso em Setembro de 2007.

[36] Softing, PROFIBUS Analyzer. Disponível na Internet: http://www.softing.com. Último acesso em Setembro de 2007.

[37] Rebakos N. et al., Deliverable D5.1 Specification of pilot applications and Field trials. Projecto Rfieldbus (IST-1999-11316), 2001.

[38] Demonstrador RFieldbus. Disponível na Internet: http://www.hurray.isep.ipp.pt/rfpilot/. Último acesso em Setembro de 2007.

[39] Nieuwenhuyse K., Behaeghel S., Timing performance of a hybrid Wired/Wireless PROFIBUS-based Network. Relatório Técnico, Instituto Superior de Engenharia do Porto/Katholieke Hogeschool St.-Lieven, 2003. Disponível na Internet: http://www.cister.isep.ipp.pt/ asp/show_doc2.asp?id=161. Último acesso em Setembro de 2007.

[40] FIELDBUS Framework for OMNeT++. Disponível na Internet: http://developer.berlios.de/ projects/fieldbus. Último acesso em Setembro de 2007.

Acrónimos

Acrónimo	Descrição
ACK	Acknowledgement
ACS	Admission Control and Scheduling
AGV	Automatic Guided Vehicle
ATM	Asynchronous Transfer Mode
AWLD	Ad-hoc Wireless Domain
bt	Bit time
BT PDU	Beackon Trigger PDU
CSRD	Cyclic Send and Request Data with Reply
DA	Destination Adress
DAE	Destination Address Extension
DCE	Data Communication Equipment
DDLM	Direct Data Link Mapper
DIS	DCE Independent Sublayer
DLL	Data Link Layer
DLX	eXtensions to the PROFIBUS Data Link layer
DP_{BE}	Tráfego DP *Best Effort*
DP_{H}	Tráfego DP de alta prioridade
DP_{L}	Tráfego DP de baixa prioridade
DSAP	Destination Service Access Point
DSSS	Direct Sequence Spread Spectrum
DTE	Data Terminal Equipment
ED	End Delimiter, valor 16_{H}
EIA	Electronics Industries Association
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
ES	End System
esp	Esporádica
ET	Expansion Terminal
FC	Frame Control
FCS	Frame Check Sequence
FDL	Fieldbus Data Link
FES	Future Events Set
FMA	Fieldbus Management
FMS	Fieldbus Message Specification
FSM	Finite State Machine
GAP	Em PROFIBUS representa o espaço de endereçamento existente entre TS e NS
GAP List	Lista contendo o estado de todas as estações do GAP
GUI	Graphical User Interface
HMI	Human Machine Interface
I/O	Imput Output
IEC	International Electrotechnical Commission
IP	Internet Protocol

IP_{BE}	Tráfego IP *Best Effort*
IP_{H}	Tráfego IP de alta prioridade
IPP	Instituto Politécnico do Porto
IS	Intermediate System
IS (PROFIBUS)	Intrinsic safety
ISEP	Instituto Superior de Engenharia do Porto
ISO	International Organization for Standardization
IT	Information Technology
Jpeg	Joint Photographic Experts Group
L	Comprimento do campo de informação
LAS	List of Active Stations
LASCRI	Laboratório de Sistemas Críticos
LBS	Link Base Station (Structuring Intermediate System)
LD	Linking Device equivalente a Link Station
LLI	Lower Layer Interface
LLRS	Lower Layer RFieldbus Simulator
LS	Link Station (Intermediate System)
LSAP	Link Service Access Point
MAF	Manufacturing Automation Fieldtrial
MAU	Medium Attachment Unit
MDS	Medium Dependent Sublayer
MobM	Mobility Master
MWLES	Mobile Wireless End System
MWRD	Mobile Wired Domain
NED	Linguagem utilizada em OMNeT++ para descrever as topologias
NEDC	NED to C compiler
NRZ	Nonreturn to Zero Signals
NS	Next Station
NS	Endereço da estação a quem passa o Token
ns-2	Network Simulator 2
OMNeT++	Objective Modular Network Testbed in C++
OO	Object Oriented
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PHY	Physical Layer
PLC	Programmable logic controller
PROFIBUS	Process Fieldbus
ProfiSafe	Tecnologia suplementar à norma PROFIBUS que reduz a probabilidade de falhas na transmissão de informação entre controladores e dispositivos tolerantes a falhas
PS	Previous Station
RC	Radio Cell
RFieldbus	Radio Fieldbus
RS	Recommended Standard
RX	Receive Signal
SA	Sender Adress

SAE	Source Address Extension
SCADA	Supervisory Control and Data Aquisition
SD	Start Delimiter
SDA	Send Data with Acknowledge
SDN	Send Data with No Acknowledge
SIS	Structuring Intermediate System
SM	Station Management
SPA	System Planning Application
SRD	Send and Request Data with Reply
SWLD	Structured Wireless Domain
SYN	Intervalo de sincronização de pelo menos 33 bt a receber do meio físico o estado idle "1B" até poder aceitar o início de uma *Action Frame*
tcc	Todos os Ciclos de Comunicação
TCP	Transmission Control Protocol
TP	Twisted Pair
TS	This Station
TX	Transmit Signal
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
UTP	Unshielded Twisted Pair
WLD	Wireless Domain
WLES	Wireless End System
WLES	Wireless End System
WRD	Wired Domain
WRES	Wired End System

Símbolos

Símbolo	Descrição
bit_rate_WL	Taxa de transmissão de dados do meio físico Wireless (Rfieldbus).
bit_rate_WR	Taxa de transmissão de dados do meio físico Wired (Profibus).
C_{ack}	Duração de um ciclo de mensagem com resposta/ack (Profibus).
$C_{ackwithoutidletime}$	Duração de um ciclo de mensagem sem incluir o tempo idle time (Profibus).
C^{x}_{Lreq}	Duração da transmissão do request no meio físico x (Profibus).
C^{x}_{Lresp}	Duração da transmissão da response no meio físico x (Profibus).
C_{unk}	Duração de um ciclo de mensagem sem resposta/ack (Profibus).
max_retry_count	Número máximo de tentativas de retransmissão de cada ciclo de mensagem (Profibus).
Prob_error_WL	Probabilidade de erro no meio físico Wireless WL (LLRS).
Prob_error_WR	Probabilidade de erro no meio físico Wired WR (LLRS).
T_{FTMMP}	Period for Triggering Mobility Management Procedure – período a partir do qual dá início ao procedimento de gestão de mobilidade.
T_{GUD}	Gap Update Time – periodicidade a partir da qual cada Master examina o seu GAP.
T_{ID1}	Idle time 1 - inserido após o envio de um Token ou após a recepção da response de um ciclo de mensagem com confirmação (acknowledge) (Profibus).
T_{ID2}	Idle time 2 - inserido após o envio de um PDU por uma estação Master após um unacknowledged request PDU (Profibus).
TID2MobM	Idle time 2 no MobM (Rfieldbus).
T_{MOB}	Duração máxima do procedimento de gestão da mobilidade (Rfieldbus).
T_{RD}	Tempo de atraso introduzido nos IS – Este tempo é medido ente o instante em que o IS inicia a recepção da trama e o instante em que inicia a sua retransmissão (RFieldbus).
trelay	Tempo de atraso introduzido pelos IS na retransmissão da trama entre os domínios de comunicação que interliga. Este tempo de atraso é medido entre o instante em que inicia a recepção e o instante em que inicia a retransmissão (LLRS).
T_{RR}	Real Rotation Time (Token) – tempo que decorre entre duas recepções consecutivas do Token (Profibus).
T_{RT}	Responder's turnaround time – é o tempo que decorre desde que uma estação finaliza a recepção de um PDU com um pedido até ao instante em que inicia a transmissão do PDU com a resposta correspondente (Profibus).
t_{SL}	Slot time - após o envio de um request ou da passagem do Token, a estação Master espera T_{SL} pela resposta da estação endereçada ou por esta ser tornar activa (no caso da recepção do Token) (Profibus).
T_{SR}	Start Relaying Instant – instante de tempo a partir do qual um IS pode iniciar a retransmissão da frame. Existe a necessidade de se calcular este instante para garantir a transmissão continua da frame dado que os IS interligam meios físicos com tramas de comprimentos diferentes (RFieldbus).
T_{ST}	System turnaround time – tempo que decorre desde o fim o request de um ciclo de mensagem até ao instante inicial da correspondente response (Profibus).
T_{TH}	Token Holding Time – tempo que cada master detém para processar ciclos de mensagens. $T_{TH} = T_{TR} - T_{RR}$ (Profibus).
T_{TR}	Target Rotation Time (Token) - tempo que foi estimado inicialmente para a rotação do Token (Profibus).

Anexo A – MAF - Manufacturing Automation Fieldtrial

Neste anexo é apresentado o *Manufacturing Automation Fieldtrial* (MAF) [16] implementado no Laboratório de Sistemas Críticos do Instituto Superior de Engenharia do Porto (LASCRI-ISEP) [38] no âmbito do projecto europeu RFieldbus [13].

É feita uma descrição do sistema mecânico idealizado, da sua funcionalidade, da rede de comunicação industrial e dos correspondentes ciclos de mensagem associados.

A.1 Sistema mecânico

O desenvolvimento deste sistema foi baseado em sistemas largamente utilizados no fabrico industrial de peças discretas. O esquema do sistema mecânico está desenhado na Figura 84. O sistema foi concebido para transportar, classificar e distribuir peças de acordo com o seu tipo.

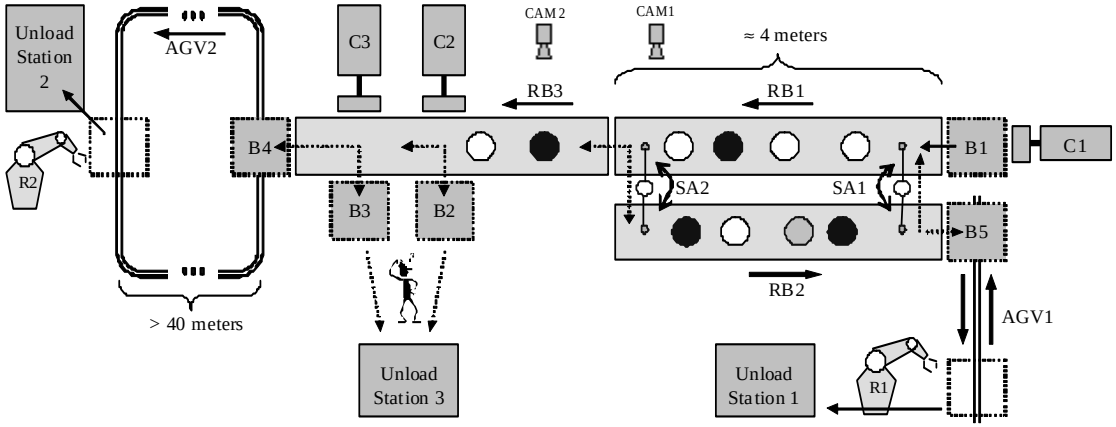


Figura 84 – Esquema do sistema mecânico

A.1.1 Componentes constituintes

O sistema mecânico é constituído pelos seguintes componentes:

- Peças brancas, pretas e cinzentas (não conformes)
- 3 Tapetes transportadores: RB1, RB2, RB3
- 5 *Buffers* de armazenamento: B1, B2, B3, B4, B5
- 3 Cilindros pneumáticos: C1, C2, C3
- 2 Braços de rotação com cilindros pneumáticos e copos de sucção: SA1, SA2
- 2 Veículos automáticos: AGV1 (autónomo), AGV2 (guiado por linha no chão)
- 2 braços robóticos (R1, R2)

- 3 estações de descarga: U1, U2, U3
- 2 câmaras de vídeo

A.1.2 Funcionalidade

O sistema desenvolvido (MAF) é um subsistema constituinte da planta de fabrico industrial de determinado tipo de peças discretas. São-lhe entregues peças sequencialmente, que o sistema classifica e separa tendo na saída dois tipos de peças conformes (pretas e brancas) e peças não conformes (cinzentas).

A este subsistema chegam um conjunto de peças que vão ser classificadas e separadas [16]. O critério de classificação escolhido foi a cor das peças. Na entrada do sistema as peças vão chegando sequencialmente. A chegada das peças foi implementada com um *buffer* de armazenamento B1. O *buffer* B1 contém peças pretas, brancas e cinzentas que são inseridas sequencialmente no tapete de transporte RB1 pelo cilindro pneumático C1. A cadência à qual C1 faz entrar peças no sistema é definida dinamicamente. RB1 transporta as peças até à zona onde se encontra o braço de rotação SA2 que transfere apenas as peças cinzentas para o início do tapete de transporte RB2. As peças cinzentas seguem o longo de RB2 com destino ao *buffer* de armazenamento B5. Quando B5 está cheio ou não presente (em descarga) as peças cinzentas são armazenadas dinamicamente fazendo-as circular ao longo de RB1 e RB2 em circuito fechado com o uso de SA1 e SA2. Quando B5 se encontra cheio, o AGV1 desloca-se para a estação de descarga U1 para efectuar a operação de descarga, retornando de seguida a posição inicial.

As peças pretas e brancas são transferidas automaticamente de RB2 para RB3 que as transporta até à zona onde se encontra C2 que procede à extracção das peças brancas para B2. Quando B2 enche o operador é avisado para proceder à descarga de B2 na estação de descarga U3 e repô-lo na sua posição inicial. B3 é um *buffer* redundante para escoar peças brancas. Quando B2 se encontra cheio ou em processo de descarga C2 não procede à extracção de peças sendo C3 que passa a faze-lo para B3. Após uma operação de descarga, B2 volta a ser o *buffer* de saída activo sendo as peças brancas novamente escoadas para este *buffer*.

Caso, ambos os *buffers* estejam não operacionais, (cheios ou em fase de descarga) as peças brancas são armazenadas dinamicamente em RB1 e RB2, como acontece com as peças cinzentas. As peças pretas são depositadas em B4. Quando B4 se encontra cheio, o AGV2 desloca-se até à estação de descarga U2 para o braço robótico R2 efectuar automaticamente a operação de descarga do *buffer*. Se este *buffer* estiver cheio ou em fase de descarga as peças pretas devem ser armazenadas dinamicamente em RB1 e RB2, como acontece com as peças cinzentas, até que B4 seja reposto vazio na sua posição.

Sempre que SA1 está a transferir uma peça entre os tapetes RB1 e RB2, o funcionamento de C1 é inibido. Sempre que o *buffer* de entrada fica vazio, é avisado o operador para proceder a sua carga. Em qualquer situação anormal, o sistema pára sendo requerida a intervenção do operador.

Dimensões das peças:	diâmetro 150-200 mm altura 20-30 mm
Peso das peças:	< 40 g
Capacidade máxima do *buffer* de entrada (B1):	30-40 peças (brancas, pretas e cinzentas)
Capacidade máxima dos *buffers* de saída (B2,B3,B4,B5):	10 peças
Cadência de entrada de peças (B1, C1):	de 1 peça a cada 5 s até 1 peça a cada 3 s
Dimensões dos tapetes de transporte (RB1, RB2, RB3):	Comprimento 3 metros Largura 40 cen*tíme*tros
Velocidades dos tapetes de transporte (RB1, RB2, RB3):	de 3 metros por minuto (5 cm/s) até 12 metros por minuto (20 cm/s)

Tabela 25 - Velocidades e dimensões relevantes

A.1.3 Velocidades e dimensões relevantes

Na Tabela 25 são apresentadas as dimensões, velocidades e taxas mais relevantes que caracterizam este sistema mecânico [16].

A.2 A rede de comunicação

O MAF é suportado por uma rede de comunicação RFieldbus (PROFIBUS-*DP*) [16]. É supervisionado por uma aplicação central que se encontra no *Master* (PC1) [16]. Esta aplicação recolhe informação para efeitos de controlo e de observação (*monitoring*), fazendo *polling* directamente a cada dispositivo bem como a algumas aplicações cooperantes. Exemplos de dispositivos são o Controlador Lógico Programável (PLC), os *drives* dos motores (MM1/MM2), e os módulos de IO distribuído PROFIBUS (ET1/ET2).

Como se pode ver na Figura 85 a rede de comunicações foi estruturada em três domínios de comunicação. Um *Wired* (WR), que é um segmento PROFIBUS e dois domínios *Wireless* (WL1 e WL2) [16]. No caso do AGV2 que está associado ao PC5 a distância que este percorre implica ainda a necessidade de usar *Link Base Stations* para aumentar o alcance das comunicações criando dois domínios de comunicação *Wireless* estruturados (WL1 e WL2). Neste caso este nó *Wireless* é móvel e a sua mobilidade implica uma mudança de domínio de comunicação (WL1 → WL2 ou vice-versa) e por isso existe no PC2 o *Mobility Master*. Como descrito em 3.3 a sua função é gerir o procedimento de gestão da mobilidade.

De referir que as estações *Wireless* foram implementadas não como *Wireless end systems* mas como segmentos *Wired* (PROFIBUS) com um interface *Wireless* (LS).

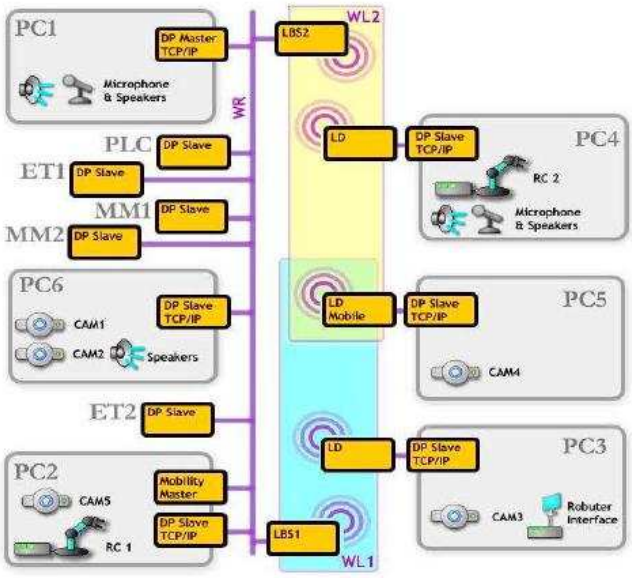


Figura 85 – Topologia de rede do MAF

WR	Segmento *Wired*, cabo PROFIBUS
WL1	Segmento *Wireless* RFieldbus (Área principal)
WL2	Segmento *Wireless* RFieldbus (Área secundária)
DP *Slave*	Dispositivos *Slave* PROFIBUS-DP (standard)
DP *Slave* TCP/IP	RFieldbus *Slave* (com suporte multimédia)
Mobility *Master*	RFieldbus *Mobility Master*
LBS	*Link Base Station*
LD	*Linking Device* (*Link Station* - Sistema intermédio)
LD *Mobile*	*Linking Device Mobile* (Sistema intermédio com suporte a mobilidade)

Tabela 26- Abreviaturas da figura da topologia de rede do MAF

A.3 Ciclos de mensagem DP

A Figura 86 esquematiza as aplicações PROFIBUS-*DP* existentes no MAF. Seguidamente vão ser descritos os ciclos de mensagem [39] gerados por cada uma das aplicações PROFIBUS-*DP*.

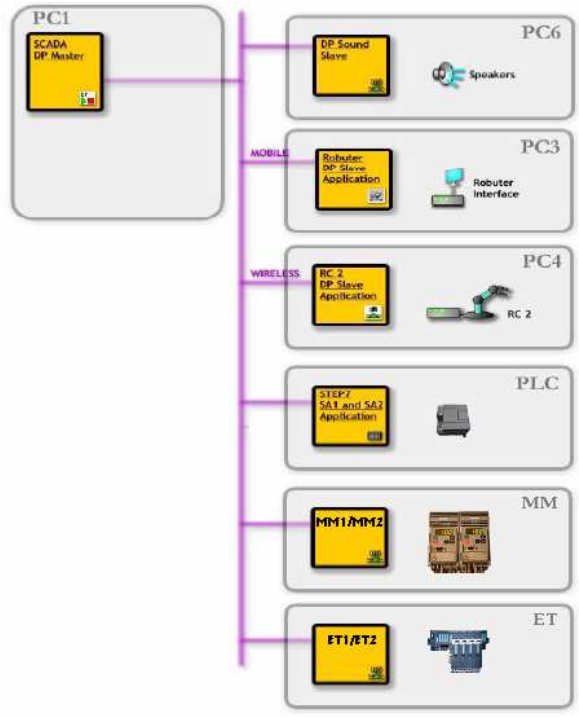


Figura 86- Aplicações DP

A.3.1 SCADA

O PC1 é o *Master* do sistema PROFIBUS-*DP*. Nele reside a aplicação *SCADA*. Esta aplicação faz a aquisição de dados e o controlo ao nível de supervisão. Por controlo ao nível de

supervisão entende-se uma ordem de alto nível que será comunicada às aplicações cooperantes, que por sua vez se encarregarão de efectuar o conjunto de operações necessárias para realizar a operação pretendida.

A aplicação SCADA é a aplicação mais importante no MAF pois tem a responsabilidade de coordenar todo o sistema. Ela é responsável pela aquisição de dados, pela aplicação dos algoritmos de controlo e pela emissão dos comandos correspondentes, isto de uma forma mais directa no caso dos I/O's binários (exemplo das ET's) mas também através da comunicação com as aplicações cooperantes.

Adicionalmente a aplicação SCADA fornece à aplicação HMI a informação sobre os eventos que ocorrem no sistema (valores lidos nos sensores, valores enviados para os actuadores, outros dados de controlo, erros, etc.) para que esta aplicação possa não só apresentar estes dados num formato amigo do utilizador (*user friendly*) como também disponibilizar parte desta informação a outras aplicações do sistema.

Dentro da aplicação SCADA existe uma sub aplicação SCADA DP *Master* que faz a aquisição/actualização de dados (DP).

Com a excepção do ciclo de mensagem de início do procedimento de gestão da mobilidade (originado pelo MobM) todas os outros ciclos de mensagem que circulam na rede são enviados ou endereçados ao *Master* (inerente à arquitectura do PROFIBUS-*DP*)

A.3.2 DP Sound Slave

A aplicação *DP Sound Slave* encontra-se no PC6, que está configurado como PROFIBUS *Slave*. Esta aplicação permite ao sistema emitir sons de aviso usando uma simples transmissão PROFIBUS. O ciclo de mensagem associado a esta aplicação é gerado todos os ciclos de comunicação e consiste num *request* do *Master* com um campo de dados de dois bytes e uma *response* idêntica. Este ciclo de mensagem será designado por S1 (Tabela 27).

A.3.3 Robuter DP Slave Application

A aplicação *Robuter DP Slave* encontra-se no PC3, que está configurado como PROFIBUS *Slave*. Esta aplicação permite ao sistema controlar o processo de descarga do *buffer* B5, através do veículo auto guiado (AGV1/*robuter*), emitindo a ordem para proceder à descarga do *buffer* B5 após o PC3 ter informado o SCADA que B5 se encontra cheio.

O ciclo de mensagem associado a esta aplicação é gerado todos os ciclos de comunicação e consiste num *request* do *Master* com um campo de dados de dois bytes (usado para sinalizar o controlador da *Robuter*) e uma *response* idêntica (contendo *local*ização e estado da *robuter*). Este ciclo de mensagem será designado por S2 (Tabela 27).

A.3.4 RC2 DP Slave Application

A aplicação *RC2 DP Slave Application* encontra-se no PC4, que está configurado como PROFIBUS *Slave*. Esta aplicação coopera com a aplicação do controlador do braço robótico para iniciar e manipular o robot. Adicionalmente o controlador do robot envia para esta aplicação informação sobre a detecção da chegada do AGV2.

O ciclo de mensagem associado a esta aplicação é gerado todos os ciclos de comunicação e consiste num *request* do *Master* com um campo de dados de dois bytes e uma *response* idêntica. Dado que o *Master* (*Initiator*) e o PC4 (*Responder*) se encontram em meios físicos diferentes este ciclo de mensagem decorre em dois tipos diferentes de meios físicos. Este ciclo de mensagem será designado por S3 (Tabela 27).

A.3.5 MM1/MM2

Ambos os *MicroMaster* MM1 e MM2 são variadores de velocidade, utilizados no MAF para variar a velocidade dos tapetes transportadores RB1, RB2 e RB3.

O ciclo de mensagem gerado entre o *Master* e cada um dos *MicroMaster* consiste num *request* do *Master* com um campo de dados de doze bytes e uma *response* idêntica, gerada todos os ciclos de comunicação. Estes ciclos de mensagem serão designados por S4 e S5 (Tabela 27).

A.3.6 ET1/ET2

As ET são módulos de *IO* distribuído. Em cada um dos dois módulos usados, são concentrados os sinais de *input* e *output* digitais dos sensores a actuadores utilizados no *MAF*, que serão lidos/escritos pelo PC1 usando para isso um ciclo de mensagem. A ET1 contém os *IO's* associados aos tapetes RB1 e RB2 e a ET2 contém os *IO's* associados ao tapete RB3.

O ciclo de mensagem gerado entre o *Master* e cada uma das *ET's* consiste num *request* do *Master* com um campo de dados de três *bytes* e uma *response* com um campo de dados de quatro *bytes*, gerado todos os ciclos de comunicação. Estes ciclos de mensagem serão designados por S6 e S7 (Tabela 27).

A.3.7 SA1/SA2

O controlo de ambos os *swivel arms* é feito *local*mente no *PLC S7-200* da *Siemens*, mediante o controlo do PC1.

O ciclo de mensagem com a informação de controlo a ser trocada entre o PC1 e o PLC consiste num *request* do *PC1* com um campo de dados de dezasseis *bytes* e uma *response* idêntica. É gerado todos os ciclos de comunicação e será designado por S8 (Tabela 27).

A.4 Ciclos de mensagem *IP*

A Figura 87 esquematiza as aplicações multimédia *IP* existentes no MAF.

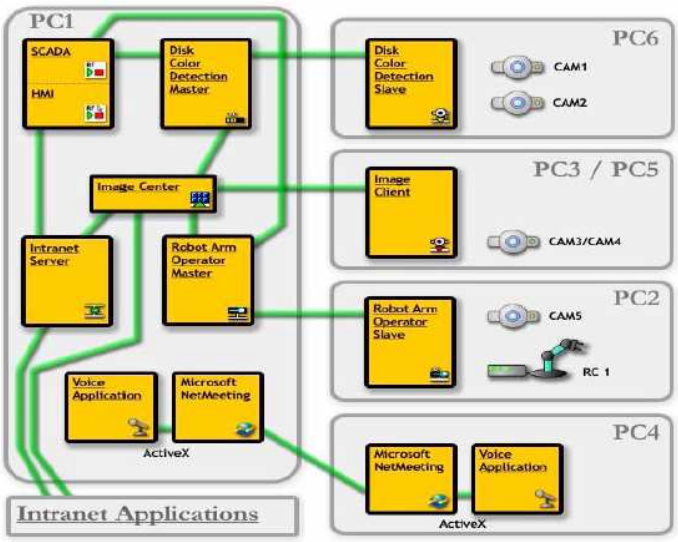


Figura 87 - Aplicações multimédia (IP)

Seguidamente vão ser descritos os ciclos de mensagem [39] gerados por cada uma das aplicações multimédia (*IP)*. Dado que o estudo apresentado nesta dissertação é sobre o comportamento temporal das camadas inferiores da rede de campo RFieldbus, os ciclos de mensagem dentro do *PC1* e entre o *PC1* e a intranet não serão descritos.

A.4.1 Disk Colour Detection Master/ Disk Colour Detection Slave

A aplicação *Disk Colour Detection Slave* (*PC6*) funciona como servidor *TCP* e a aplicação *Disk Colour Detection Master* como cliente *TCP*. Existe uma *conexão* TCP/IP estabelecida entre ambas. A aplicação *Slave* usa uma câmara para fotografar os tapetes transportadores. As imagens são analisadas *local*mente e sempre que uma peça é detectada a imagem é enviada à aplicação *Master* que a utilizará para determinar a cor da peça detectada no tapete. As imagens têm um tamanho compreendido entre dois a três *Kbytes*. Quando a aplicação *Slave* verifica a saída da peça previamente detectada envia uma mensagem de quatro *bytes*.

Este ciclo de mensagem tem uma periodicidade de três segundos (dado esta ser a cadência máxima de entrada de peças em RB1) e será designado por S9 (Tabela 27).

A.4.2 Image Centre/ Image Client

A aplicação *Image Client* (*PC3 e PC5*) envia periodicamente (um segundo) uma fotografia da câmara que se encontram sobre o *AGV* (*AGV1* e *AGV2*). A aplicação *Image Centre* (*PC1*) recebe cada uma dessas imagens, disponibiliza todas as imagens das câmaras do *MAF* para a *intranet* e permite a visualização dessas imagens pelo operador do *PC1*. As imagens são mandadas em *UDP*.

As imagens têm um tamanho compreendido entre dois a três *Kbytes*. Estes ciclos de mensagem têm uma periodicidade de um segundo e serão designados por S10 e S11 (Tabela 27).

A.4.3 Robot Arm Operator Master/ Robot Arm Operator Slave

A aplicação Robot *Arm Operator Slave* controla o braço robótico *RC1* e a câmara vídeo da área de acção de *RC1* (descarga de B5). A aplicação *Robot Arm Operator Master* comunica as ordens à aplicação *Slave*, recebe as respostas a as imagens das operações de descarga de B5. As operações, *requests* e *responses* são enviados como cadeias de carácteres. As imagens estão comprimidas no formato *jpeg*, o seu tamanho varia entre oito e dezasseis *Kbytes* e o seu envio acontece esporadicamente quando ocorre uma operação de descarga de *B5*. Comandos e imagens são enviados através de uma conexão TCP/IP que interligada ambas as aplicações. Este ciclo de mensagem será designado por S12 (Tabela 27).

A.4.4 Intranet Server

A aplicação *Intranet Server* corre no *PC1* como se pode observar na Figura 87. Fornece informação acerca do sistema aos dispositivos ligados à *intranet* do *MAF* bem como a possibilidade de interagir com o mesmo.

A.4.5 Voice Application

A aplicação de voz usa o *Microsoft NetMetting* para enviar áudio adquirido pelos microfones existentes no *PC1* e no *PC4*. É possível os operadores destas estações remotas estabelecerem uma comunicação áudio *Full Duplex*. O *Microsoft NetMetting* usa conexões *TCP* e *UDP*. A aplicação é simétrica o que significa que pode ser iniciada por qualquer um dos operadores. O ciclo de mensagem é esporádico e será designado por S13 (Tabela 27).

A.5 Quadro resumo das características de cada *message stream*

Stream	Type	Size of *request* s (PhL bytes)	Size of answer (PhL bytes)	Period	Description	*Initiator* Name	*Initiator* Adress	Responder Name	Responder Adress
S1	DP	11	11	tcc[21]	Sound *Slave Application*	PC1	1	PC6	6
S2	DP	11	11	tcc	Robuter *Slave* Connection	PC1	1	PC3	3
S3	DP	11	11	tcc	RC2 *Slave* Connection	PC1	1	PC4	4
S4	DP	21	21	tcc	Speed Roller Belts	PC1	1	MM1	10
S5	DP	21	21	tcc	Speed Roller Belts	PC1	1	MM2	11
S6	DP	12	13	tcc	Sensors-Actuators	PC1	1	ET1	64
S7	DP	12	13	tcc	Sensors-Actuators	PC1	1	ET2	66
S8	DP	25	25	tcc	Swivel Arms	PC1	1	PLC	65
S9	IP	6	255	≥3s	Disk Color Detection	PC1	1	PC6	6
S10	IP	6	255	1s	Video Monitoring	PC1	1	PC3	3
S11	IP	6	255	1s	Video Monitoring	PC1	1	PC5	5
S12	IP	39	255	esp[22]	Robot Arm Operator	PC1	1	PC2	2
S13	IP	255	255	esp	Voice Netmeeting	PC1	1	PC4	4

Tabela 27 - Ciclos de mensagem do MAF

[21] tcc – todos os ciclos de comunicação
[22] esp - esporádica

Anexo B - Documentação das classes do modelo de simulação

Neste anexo são apresentadas as principais funcionalidades das classes do modelo de simulação desenvolvido.

B.1 Lista das classes implementadas no simulador

1) Mac
2) Mqueues
3) Phy
4) Fdluser
5) Gerresp
6) Ls
7) Lbs
8) Bs
9) Phywired
10) Phywireless
11) MobM
12) Chanaccess

B.2 Descrição das classes

B.2.1 Classe Mac

virtual void Mac::activity ( );

Neste método foram implementados os algoritmos de acesso ao meio físico, nomeadamente o algoritmo de escalonamento de tráfego descrito no fluxograma da Figura 55, o processamento de ciclos de mensagem com resposta (Figura 10), o processamento de ciclos de mensagem sem resposta (Figura 13), fazendo respeitar os tempos envolvidos: *Slot time* (Figura 11), *idle time* (Figura 12, Figura 13). Foi também implementado o algoritmo de transmissão (Figura 56) e recepção do *token* (Figura 54).

int Mac::message_cycle (profi *msg, simtime_t slot_time, simtime_t t_id1, simtime_t t_id2,

int max_retry_count, simtime_t tbit);

Método (utilizado no método activity) que implementa o algoritmo de processamento de ciclos de mensagem com resposta (Figura 10) e sem resposta (Figura 13). Para ciclos de alta e baixa prioridade.

void Mac::pass_token (int SA, int DA, simtime_t t_id1, simtime_t tbit, simtime_t slot_time);

Método que implementa o algoritmo de passagem do token.

B.2.2 Classe Mqueues

virtual void Mqueues::activity ( );

Este método implementa as filas de mensagem que guardam as mensagens da FDL (Tabela 5) implementa o algoritmo representado no fluxograma da Figura 51. Suporta a inserção de mensagens de alta e baixa prioridade provenientes da requisição do serviço FDL pela camada superior, bem como a entrega da mensagem pedida para transmissão (Nota: o algoritmo de escalonamento é implementado no MAC).

B.2.3 Classe Phy

virtual void Phy::activity ( );

Este método implementa uma retransmissão instantânea da mensagem que recebe. Foi definido para que visualmente a estrutura do simulador reflicta o modelo em camadas existente, não sendo a sua função necessária à simulação (pode ser retirado, pois não afecta o funcionamento do simulador). A funcionalidade implementada está de acordo com o fluxograma da Figura 58.

B.2.4 Fdluser

virtual void Fdluser::activity ( );

Este método implementa um gerador de tráfego cujo algoritmo está de acordo com o fluxograma da Figura 49. Processa automaticamente o ficheiro de texto contendo as streams a simular.

B.2.5 Gerresp

virtual void Gerresp::activity ( );

Este método implementa a funcionalidade do *slave*. Concretamente, ao receber um *request* endereçado a si, constrói a resposta a enviar, aguarda o tempo de resposta da estação e envia-a de acordo com o algoritmo descrito no fluxograma da Figura 61.

B.2.6 Ls

virtual void ls::activity ( );

Este método implementa o algoritmo de retransmissão das tramas entre meios físicos diferentes (entre meio físico com fios e sem fios não estruturado, *Direct Link Network*), como descrito no fluxograma da Figura 77.

B.2.7 Lbs

virtual void lbs::activity ( );

Este método implementa o algoritmo de retransmissão de tramas entre meios físicos diferentes (entre meio físico com fios e meio físico sem fios estruturado, *Relay Link Network*, como descrito no fluxograma da Figura 79.

B.2.8 Bs

virtual void bs::activity ( );

Este método implementa o algoritmo de retransmissão de tramas entre o canal de *uplink* e *downlink* num meio físico sem fios estruturado, *Relay Link Network*, como descrito no fluxograma da Figura 79.

B.2.9 Phywired

virtual void Phywired::activity ( );

Este método implementa o meio físico com fios. Ao receber uma trama, calcula o seu tempo de transmissão (Equação 3), aguarda esse tempo de transmissão e envia-a para todas as estações associadas (*broadcast*) como descrito no fluxograma representado na Figura 70.

B.2.10 Phywireless

virtual void Phywireless::activity ( );

Este método implementa o meio físico sem fios. A funcionalidade implementada está de acordo com o fluxograma descrito na Figura 72. O módulo sabe se o meio físico sem fios se encontra num domínio não estruturado *Direct Link Network* (Figura 28) ou estruturado *Relay Link Network* (Figura 30) através do parâmetro LBS_BS. No caso dos domínios estruturados, ao primeiro par de gates do módulo Phywireless (0) é sempre ligado um módulo LBS ou BS.

Quando o módulo Phywireless recebe um PDU através desse par de gates (LBS ou BS a emitir no canal de *downlink*) vai aguardar o tempo de transmissão e retransmitir o PDU para os restantes pares de gates (*broadcast* no canal de *downlink*).

Quando o módulo Phywireless recebe um PDU através de qualquer outro par de gates (canal de *uplink*), aguardar o tempo de transmissão e retransmite o PDU para o primeiro par de gates (para a LBS ou BS, no canal *uplink*).

B.2.11 MobM

virtual void MobM::activity ( );

O MobM é um *master*, o seu funcionamento rege-se pelo fluxograma representado na Figura 68. No estado detem token apenas envia o *Beacon Trigger* PDU para desencadear o procedimento de gestão da mobilidade e aguarda o *idle time* (T_{ID2} MobM) antes de devolver o token ao meio físico.

void MobM::pass_token (int SA,int DA);

Método que implementa o algoritmo de passagem do token.

B.2.12 Chanaccess

virtual void Chanaccess::activity ( );

A funcionalidade implementada neste método está descrita no fluxograma da Figura 66 e faz com que este módulo ao ser adicionado a um *slave* o torne num *slave* móvel. Aguarda a recepção do BT_PDU, despoletando de seguida a avaliação dos canais rádio disponíveis. No caso implementado gera um número aleatório que definirá qual dos canais é o que apresenta melhor qualidade de sinal.

Anexo C - Estrutura de Mensagem OMNeT++

De seguida é apresentada a mensagem definida em OMNeT++ para representar as tramas RFieldbus.

```
//---------------------------------------------------
// File: profi.msg
//
// Message definitions for Profibus
//
// Author: VBLima
//---------------------------------------------------

message profi
 {
 fields:

        int msg_n; // numero de serie da mensagem

        int stream_n; //stream no MAF

        int priority = 1; // 1=hi 2=low

        int type;
        int type_R;

        int req_resp; //req_resp=1 request req_resp=2 response

//*************************************************************
// vai tudo na mensagem (o request e o response), o slave depois
// copia os campos (_R) para os originais e faz o set de:
// type, req_resp

        //REQUEST
        int SD1;
        int SD2;
        int SD3;
        int SD4;
        int SC;
        int DA;
        int SA;
        int FC;
        int FCS;
        int ED;
        int LE;
        int LEr;
        int DATA_UNIT;
        int L;
        int tamanho;

        //RESPONSE
        int SD1_R;
        int SD2_R;
        int SD3_R;
        int SD4_R;
        int SC_R;
        int DA_R;
        int SA_R;
        int FC_R;
        int FCS_R;
```

```
            int ED_R;
            int LE_R;
            int LEr_R;
            int DATA_UNIT_R;
            int L_R;

      //*********************************************************
      //
      //     type - é esta a var que vai permitir identificar qual o tipo de msg,
      //reportar os estados das filas, pedido de msgs
      //         mensagens profibus
      // 0       nenhuma  trama  (serve  para  o  type_r  para  indicar  que  não  tem
resposta)
      // 1       sd1
      // 2      sd2
      // 3      sd3
      // 4      sd4
      // 5      sd1_sc
      //
      //       estados das filas
      // 10     queue_hi vazia?
      // 11      queue_low1 vazia?
      // 12      queue_low2 vazia?
      // 13     queue_low3 vazia?
      //
      // 20      mensagem não inserida, deve ser reenviada ao layer superior
      //
      // 30     pedido de mensagem de alta prioridade
      // 31     fila de alta prioridade vazia
      // 32     pedido de mensagem de baixa prioridade
      // 33     fila de baixa prioridade vazia
      //
      // 3 + SC=3   Special Frame enviada pelo MobM que indica às BS e LBS que podem
      //iniciar o proc de mobilidade
      //            despoleta no modulo Chanaccess o proc de mobilidade q pode levar
      a //mudança de dominio
      //
      //
      //
      //
      //*********************************************************
      //  FORMATOS DAS TRAMAS
      //       ********SD1****************
      //       int sd1_SD1 = 16;
      //       int sd1_DA;
      //       int sd1_SA;
      //       int sd1_FC;
      //       int sd1_FCS;
      //       int sd1_ED = 22;
      //
      //       ********SD2****************
      //       int sd2_SD2 = 104;
      //       int sd2_LE;
      //       int sd2_LEr;
      //       int sd2_DA;
      //       int sd2_SA;
      //       int sd2_FC;
      //       int sd2_DATA_UNIT;
      //       int sd2_FCS;
      //       int sd2_ED = 22;
      //       int sd2_L;  // L campo de informação - entre [4, 249] octetos
      //
      //       ********SD3****************
```

```
//       int sd3_SD3 = 162;
//       int sd3_DA;
//       int sd3_SA;
//       int sd3_FC;
//       int sd3_DATA_UNIT = 8; // tamanho fixo, em octetos
//       int sd3_FCS;
//       int sd3_ED = 22;
//       int sd3_L = 11; // information field length
//
//       ********SD4***************
//       int sd4_SD4 = 220;
//       int sd4_SA;
//       int sd4_DA;
//       int sd4_tamanho = 3; // em bytes
//
//       ********SD1_SC*************
//       int SC = 229;
//*************************************************************

};
```