Acesso Local sem Fios em Redes de Comunicação

Escalonamento de Tráfego de Tempo Real em Sistemas de Aquisição de Dados e Controlo

José Augusto Afonso

Dissertação submetida à Universidade do Minho

para obtenção do grau de Doutor

em Engenharia Electrónica Industrial,

elaborada sob a orientação do

Prof. Doutor Joaquim José Esteves Neves

Guimarães, Novembro de 2004

Agradecimentos

Ao Professor Joaquim José Esteves Neves, meu orientador de doutoramento, pela disponibilidade e auxílio na preparação desta tese.

À Universidade do Minho, pelas condições de trabalho proporcionadas para a realização do doutoramento.

A todos os colegas do Departamento de Electrónica Industrial que contribuíram com o seu apoio e incentivo para a realização deste trabalho.

Aos meus pais, um agradecimento especial pelo amor e dedicação demonstrados em todos os momentos.

É de referir que este trabalho foi suportado pelo programa PRODEP, concurso Nº 4/5.3/PRODEP/2000.

Resumo

Este trabalho propõe soluções destinadas a optimizar o transporte do tráfego de sistemas de aquisição de dados e controlo, utilizando redes de comunicação sem fios para a interligação de unidades remotas a uma unidade central. O tráfego gerado por estes sistemas normalmente possui requisitos de tempo real, pelo que a rede sem fios utilizada deve ser capaz de suportar os requisitos de qualidade de serviço (QoS) das aplicações.

No entanto, apesar de dar prioridade ao transporte de tráfego com requisitos de tempo real (isócrono), o sistema proposto admite também a transmissão, em simultâneo, de tráfego assíncrono, de forma a permitir a integração de serviços de natureza diferente sobre a mesma plataforma de comunicação.

A largura de banda disponível para operação das redes sem fios é escassa, e tem que ser partilhada por todas as unidades. Além disso, a taxa de erros no canal costuma ser bem mais elevada e variável do que nas redes cabladas. Devido a esses problemas, os mecanismos de controlo de acesso ao meio, controlo de erros e escalonamento de tráfego implementados pela rede sem fios utilizada no sistema devem ser optimizados para garantir a qualidade de serviço das aplicações.

Efectuado um estudo preliminar das tecnologias de redes sem fios disponíveis, de forma a seleccionar aquelas que apresentam as características mais adequadas para servir de base ao sistema proposto, a escolha recaiu sobre as redes HIPERLAN/2 e IEEE 802.11.

Não sendo viável o recurso à utilização de sistemas reais, foi desenvolvido e implementado um modelo de simulação detalhado para cada uma destas redes, de acordo com as respectivas normas. Com base nestes modelos, propõem-se algoritmos de escalonamento do tráfego de dados e controlo que fornecem garantias de qualidade de serviço ao tráfego isócrono, ao mesmo tempo que permitem o transporte de tráfego

assíncrono no sistema. Os resultados obtidos nas simulações permitem avaliar o desempenho do sistema, estando ou não sujeito a erros no canal, e comparar a qualidade de serviço obtida com a rede HIPERLAN/2 e com a rede IEEE 802.11.

A tese inicia-se com a introdução dos conceitos fundamentais à compreensão do trabalho apresentado, à qual segue-se a descrição das diversas tecnologias de redes sem fios existentes, realçando as normas HIPERLAN/2 e IEEE 802.11. Após a descrição dos trabalhos relacionados, apresentam-se os algoritmos propostos. Seguem-se os modelos de simulação desenvolvidos e os resultados obtidos com os sistemas baseados em ambas as redes.

Abstract

This work proposes solutions intended to optimize the transmission of traffic on data acquisition and control systems using wireless networks to interconnect remote units to a central unit. The traffic generated by such systems normally is subject to real time constraints, so the wireless network must be able to support the quality of service (QoS) requirements of the applications.

Although the priority is given to the transportation of real time (isochronous) traffic, the proposed system also allows the transmission of asynchronous data simultaneously, in order to enable the integration of different services under the same communication platform.

The bandwidth available to the operation of wireless networks is scarce, and has to be shared among all the units. Moreover, the channel error rate tends to be higher and more variable than on wired networks. Due to such problems, the medium access control, error control and traffic scheduling mechanisms implemented by the wireless network that is used by the system must be optimized in order to guarantee the quality of service required by the applications.

After a preliminary study of the available wireless networks in order to select those who present the most appropriate characteristics to serve as the basis of the proposed system, the HIPERLAN/2 and IEEE 802.11 networks were chosen.

As the utilization of real systems was not viable, a detailed simulation model for each of the chosen networks was developed and implemented, according to the respective standards. Based on these models, this thesis proposes scheduling algorithms comprising the data and control traffic, which provide quality of service guarantees to the isochronous traffic and allow the transportation of asynchronous traffic on the system at the same time. The results obtained with the simulations allow the evaluation of the performance of the system, with and without channel errors, and

to compare the quality of service of the HIPERLAN/2 network with that of the IEEE 802.11 network.

The thesis begins with an introduction to the fundamental concepts related to the presented work, which is followed by a description of the different wireless networks technologies, with emphasis on the HIPERLAN/2 and IEEE 802.11 standards. After a description of the related work and the proposed algorithms, the simulation models developed and the results achieved with the systems based on both standards are presented.

Índice de Conteúdos

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas

3GPP	3rd Generation Partnership Project
AAL	ATM Adaptation Layer
ABR	Available Bit Rate
AC	Access Category
ACF	Association Control Function
ACH	Access feedback CHannel
ACL	Asynchronous Connection-Less
ADPCM	Adaptive Differential Pulse Code Modulation
AGCH	Access Grant CHannel
AIFS	Arbitration InterFrame Space
AMPS	Advanced Mobile Phone System
AP	Access Point
ARQ	Automatic Repeat reQuest
ASCH	ASsociation control CHannel
ATM	Asynchronous Transfer Mode
AuC	Authentication Center
BCCH	Broadcast Control CHannel
BCH	Broadcast CHannel
BER	Bit Error Rate
BMB	BitMap Block

BMN	BitMap block Number
BPSK	Binary Phase Shift Keying
BRAN	Broadband Radio Access Networks
BS	Base Station
BSC	Base Station Controller
BSA	Basic Service Area
BSS	Basic Service Set
BSS	Base Station Subsystem
BTS	Base Transceiver Station
BWA	Broadband Wireless Access
CAC	Connection Admission Control
CAI	Cumulative Acknowledgement Indicator
CAN	Controller Area Network
CBR	Constant Bit Rate
CC	Central Controller
CCK	Complementary Code Keying
CDMA	Code Division Multiple Access
CDV	Cell Delay Variation
CDVT	Cell Delay Variation Tolerance
CEPT	European Conference of Postal and Telecommunications Administrations
CER	Cell Error Rate
CFB	Contention Free Burst
CFP	Contention Free Period
CL	Convergence Layer
CLP	Cell Loss Priority

CLR	Cell Loss Ratio
CFP	Contention Period
C/I	Carrier to Interference ratio
CMR	Cell Misinsertion Rate
CPCS	Common Part Convergence Sublayer
CRA	Contention Resolution Algorithm
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
CT	Cordless Telephony
CTD	Cell Transfer Delay
CTS	Clear To Send
CW	Contention Window
DCC	DLC Connection Control
DCCH	Dedicated Control CHannel
DCF	Distributed Coordination Function
DCS	Dynamic Channel Selection
DECT	Digital Enhanced Cordless Telecommunication
DFIR	Diffuse Infrared
DFS	Dynamic Frequency Selection
DIFS	Distributed InterFrame Space
DiL	Direct Link
DL	Downlink
DLC	Data Link Control

DLCC	DLC Connection
DLP	Direct Link Protocol
DS	Distribution System
DSSS	Direct Sequence Spread Spectrum
DS-CDMA	Direct Sequence Code Division Multiple Access
EC	Error Control
EDCA	Enhanced Distributed Channel Access
EDCF	Enhanced Distributed Coordination Function
EDD	Earliest Due Date
EDF	Earliest Deadline First
EFR	Enhanced Full-Rate
EIR	Equipment Identity Register
ERC	European Radiocommunications Committee
ETSI	European Telecommunications Standards Institute
ESS	Extended Service Set
EY-NPMA	Elimination Yield Non-Preemptive Priority Multiple Access
FCA	Fixed Capacity Agreement
FCC	Federal Communications Commission
FCCH	Frame Control CHannel
FCH	Frame CHannel
FCFS	First Come First Served
FCS	Frame Check Sequence
FDD	Frequency Division Duplex
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction

FFQ	Fluid Fair Queuing
FHSS	Frequency Hopping Spread Spectrum
FH-CDMA	Frequency Hopping Code Division Multiple Access
FIFO	First In First Out
FP	Fixed Part
FSK	Frequency Shift Keying
FWA	Fixed Wireless Access
GFC	Generic Flow Control
GFSK	Gaussian Frequency Shift Keying
GMSK	Gaussian Minimum Shift Keying
GPRS	General Packet Radio Service
GPS	Generalized Processor Sharing
GSM	Global System for Mobile Communications
HARQ	Hybrid ARQ
HC	Hybrid Coordinator
HCCA	HCF Controlled Channel Access
HCF	Hybrid Coordination Function
HEC	Header Error Check
HEC	Header Error Control
HIPERLAN	High Performance Radio Local Area Network
HLR	Home Location Register
HTTP	HyperText Transfer Protocol
HSCSD	High Speed Circuit Switched Data
IBSS	Independent BSS
IE	Information Element

IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IFS	InterFrame Space
IMT-2000	International Mobile Telecommunications 2000
IP	Internet Protocol
IrDA	Infrared Data Association
ISI	InterSymbol Interference
ISM	Industrial, Scientific and Medical
ISO	International Organization for Standardization
ITU	International Telecommunications Union
LAN	Local Area Network
LCCH	Link Control CHannel
LCH	Long transport CHannel
LCG	Linear-Congruential Generator
LLC	Logical Link Control
LOS	Line Of Sight
LMDS	Local Multipoint Distribution Service
MAC	Medium Access Control
MASCARA	Mobile Access Scheme based on Contention and Reservation for ATM
MBS	Maximum Burst Size
MCR	Minimum Cell Rate
ME	Mobile Equipment
MMDS	Multi-channel Multipoint Distribution Service
MPDU	MAC Protocol Data Unit
MSDU	MAC Service Data Unit

MS	Mobile Station
MSC	Mobile Switching Center
MSDU	MAC Service Data Unit
MT	Mobile Terminal
NAV	Network Allocation Vector
NMT	Nordic Mobile Telephone System
nrt-VBR	Non-Real-Time Variable Bit Rate
NS	Network Subsystem
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open System Interconnection
PAN	Personal Area Network
PCF	Point Coordination Function
PCH	Paging CHannel
PCR	Peak Cell Rate
PDH	Plesiochronous Digital Hierarchy
PDU	Protocol Data Unit
PER	Packet Error Rate
PIFS	PCF InterFrame Space
PLCP	Physical Layer Convergence Procedure
PLR	Packet Loss Ratio
PMD	Physical Medium Dependent
PP	Portable Part
PPM	Pulse Position Modulation
PRADOS	Prioritized Regulated Allocation Delay Oriented Scheduling
PRMA	Packet Reservation Multiple Access

PSTN	Public Switched Telephone Network
PT	Payload Type
QAM	Quadrature Amplitude Modulation
QBSS	QoS Basic Service Set
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RACH	Random Access CHannel
RBCH	RLC Broadcast CHannel
RCH	Random CHannel
RDIS	Rede Digital com Integração de Serviços
RF	Rádio Frequência
RFCH	Random access Feedback CHannel
RG	Resource Grant
RLC	Radio Link Control
RPE-LPC	Regular Pulse Excited-Linear Predictive Coder
RR	Resource Request
RRC	Radio Resource Control
RTS	Request To Send
rt-VBR	Real-Time Variable Bit Rate
SACCH	Slow Associated Control CHannel
SAR	Segmentation and Reassembly
SCH	Short transport CHannel
SCO	Synchronous Connection-Oriented
SCR	Sustained Cell Rate
SDMA	Space Division Multiple Access

SDU	Service Data Unit
SECBR	Severely Errored Cell Block Ratio
SIFS	Short InterFrame Space
SIM	Subscriber Identity Module
SMTP	Simple Mail Transfer Protocol
SS	Spread Spectrum
SSCS	Service Specific Convergence Sublayer
TACS	Total Access Communications System
TBTT	Target Beacon Transmission Time
TCH	Traffic CHannel
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
TG	Task Group
TIM	Traffic Indication Map
TPC	Transmission Power Control
TSPEC	Traffic Specification
TXOP	Transmission Opportunity
UBCH	User Broadcast CHannel
UBR	Unspecified Bit Rate
UDCH	User Data CHannel
UMCH	User Multicast CHannel
UDP	User Datagram Protocol
UL	Uplink
UP	User Priority

UMTS	Universal Mobile Telecommunications System
U-NII	Unlicensed National Information Infrastructure
VBR	Variable Bit Rate
VCI	Virtual Channel Identifier
VLR	Visitor Location Register
VPI	Virtual Path Identifier
WATM	Wireless ATM
WEP	Wired Equivalent Privacy
WFQ	Weighted Fair Queueing
WLAN	Wireless Local Area Network
WLL	Wireless Local Loop
WPAN	Wireless Personal Area Network
WRR	Weighted Round-Robin

Capítulo 1

Introdução

1.1 Enquadramento

Nos tempos actuais assiste-se a uma forte competição entre as tecnologias de suporte das redes cabladas e das redes sem fios. Se os sistemas de transmissão por fibra óptica são cada vez mais usados na rede de transporte, para interligação de comutadores de grande capacidade, os sistemas de transmissão por Rádio Frequência voltam a impor-se nas redes de acesso e de área local, nomeadamente na interligação de equipamentos terminais. A difusão dos sistemas celulares móveis e das redes locais (ou redes de área local) sem fios constituem exemplos flagrantes desta tendência. O sucesso das redes de comunicação sem fios deriva da eliminação dos cabos, que proporciona as seguintes vantagens.

- Aumento da mobilidade.
- Rápida instalação e reestruturação da rede.
- Redução dos custos e inconvenientes associados à instalação de cabos.

Os sistemas celulares móveis surgiram para a transmissão de tráfego de voz em alternativa à rede telefónica fixa, enquanto as redes locais sem fios foram concebidas para substituir, ou complementar, as redes locais convencionais, criadas para transmissão de dados. O tráfego de voz e vídeo possui requisitos de tempo real, impondo, por isso, limitações nos atrasos de transmissão, mas é relativamente tolerante a erros resultantes da perda de pacotes. Já o tráfego de dados normalmente

não tolera erros, mas, por outro lado, não impõe restrições temporais, sendo por isso denominado tráfego assíncrono.

Entretanto, o tráfego de aplicações industriais alia os requisitos de tempo real com a baixa tolerância a erros, possuindo ainda diferentes níveis de prioridade, pelo que as redes locais convencionais não são adequadas à sua transmissão. Sendo assim, estas aplicações são suportadas por redes cabladas específicas, como o CAN (*Controller Area Network*) [FAR99] e outras redes conhecidas pela designação de barramentos de campo [DEC97a]. A extensão das redes de comunicação sem fios aos ambientes industriais permitiria o aproveitamento das suas vantagens.

No entanto, o desenvolvimento de redes sem fios apresenta dificuldades acrescidas em relação às redes cabladas, relacionadas com a propagação em meio livre, destacando-se as seguintes.

- A largura de banda disponível para a operação de um sistema sem fios é limitada, pois o espectro de frequências tem que ser dividido pelos diferentes serviços que partilham o mesmo meio, para possibilitar a sua coexistência.
- Nas redes sem fios normalmente não é possível transmitir e monitorar o canal ao mesmo tempo, pois o sinal transmitido sobrecarregaria os circuitos de recepção. Isto tem repercussão a nível do protocolo de controlo de acesso ao meio.
- As redes cabladas podem ser organizadas em diferentes topologias de acordo com as ligações físicas entre as estações. Nas redes sem fios o sinal de uma estação propaga-se para todas as outras como num barramento.
- As redes sem fios estão sujeitas a taxas de erros no canal muito mais elevadas e variáveis no tempo, devido a factores como interferências, obstrução e efeitos da propagação multipercurso.

Estas dificuldades já foram superadas, em grande parte, graças à actividade de investigação desenvolvida tanto a nível da camada física quanto ao nível da camada de ligação de dados. Hoje em dia já existem redes de comunicação sem fios capazes de transportar tanto o tráfego de tempo real como o tráfego assíncrono proveniente de aplicações convencionais, admitindo diferentes níveis de prioridade. No entanto, os erros no canal continuam a ser um obstáculo à utilização das redes sem fios em

ambientes industriais, devido à baixa tolerância a erros associada às restrições temporais.

A utilização de técnicas de modulação e correcção de erros permite tornar o canal mais robusto, mas não consegue eliminar totalmente os erros. A técnica de detecção de erros e retransmissão (ARQ) é normalmente utilizada no transporte de tráfego assíncrono para remoção dos restantes erros, mas tem sido pouco considerada para uso com tráfego de tempo real, porque o atraso extra derivado das retransmissões pode fazer com que os pacotes não cheguem a tempo ao destino. Entretanto, o débito elevado proporcionado pelas redes locais sem fios mais recentes associado ao curto tempo de propagação característico destas redes, favorecem a utilização desta técnica. No entanto, é necessária a criação de algoritmos de escalonamento que proporcionem a retransmissão rápida dos dados corrompidos por erros no canal, caso contrário a retransmissão do tráfego de tempo real é inútil.

Este trabalho propõe soluções destinadas a optimizar o transporte do tráfego de sistemas de aquisição de dados e controlo utilizando redes de comunicação sem fios. Duas abordagens foram consideradas inicialmente: o desenvolvimento de uma nova rede; ou o desenvolvimento de uma solução apoiada em normas de redes sem fios existentes.

A vantagem do desenvolvimento de uma nova rede é que a margem de manobra é maior, pelo que a rede poderia ser melhor optimizada para atender os requisitos da aplicação desejada. Por outro lado, os custos do desenvolvimento da solução completa são elevados, pois esta deverá integrar de forma efectiva funções de controlo de acesso ao meio, controlo de erros, escalonamento de tráfego, gestão das conexões, gestão da mobilidade, funções da camada física, etc. Uma proposta baseada nesta abordagem teria de se concentrar num conjunto limitado de funções, como o controlo de acesso ao meio [JIA98] [WIL97], devido à complexidade do sistema completo. No entanto, seria difícil avaliar o peso de outros factores, como o *overhead* necessário para a execução das demais funções, que também influenciam o desempenho global do sistema.

Por outro lado, uma solução apoiada em normas existentes poderá incorporar uma grande parte das funções necessárias ao funcionamento da rede sem fios, já

especificadas e testadas para funcionar de forma integrada. O custo do produto final será menor devido à economia de escala, sendo ainda possível a integração de equipamentos de diferentes fabricantes.

Esta última abordagem foi adoptada, tendo envolvido a sequência dos seguintes passos iniciais:

- estudo das diferentes normas de redes de comunicação sem fios existentes;
- selecção das mais promissoras para a transmissão do tráfego de sistemas de aquisição de dados e controlo;
- e proposta de soluções para optimizar o desempenho destas redes no transporte do tráfego de tempo real.

As soluções propostas incidem principalmente sobre o escalonamento de tráfego e a sua implementação, em regra, não exige modificações incompatíveis com as especificações das respectivas redes de suporte.

Não obstante a ênfase deste trabalho ser dada ao transporte de tráfego com requisitos de tempo real, considera-se também a possibilidade da transmissão, em simultâneo, de tráfego assíncrono, de forma a permitir a integração de serviços de natureza diferente sobre a mesma plataforma de comunicação.

As soluções propostas podem ser utilizadas num leque variado de aplicações industriais, que inclui desde sistemas de aquisição de dados em estufas agrícolas [SAN98] [SER97] até à coordenação de equipas de robôs móveis. Estas soluções podem igualmente ser utilizadas no transporte de tráfego de tempo real em aplicações convencionais.

1.2 Objectivos do trabalho

Como o desempenho previsível das soluções propostas para escalonamento de tráfego depende em grande parte da rede de comunicações sem fios utilizada, foi feito um estudo das tecnologias de redes sem fios existentes, por forma a seleccionar as

redes com as características[1] mais adequadas para a transmissão do tráfego de sistemas de aquisição de dados e controlo. Com base nesse estudo, a escolha dos sistemas de transmissão recaiu sobre as redes IEEE 802.11 [IEE99] e HIPERLAN/2 [ETS00].

Os algoritmos de escalonamento do tráfego de dados e de controlo que são propostos visam fornecer garantias de qualidade de serviço ao tráfego de tempo real, de alta prioridade, sobretudo quando sujeito a erros no canal. Ao mesmo tempo, permite-se o transporte de tráfego assíncrono no sistema aproveitando os recursos não utilizados pelo tráfego prioritário.

Estes algoritmos foram elaborados com o objectivo de minimizar não só o atraso das conexões de tempo real, mas também o seu *jitter*[2], sacrificando a eficiência na utilização dos recursos, relativamente à que seria possível sem estas restrições. Outro objectivo consiste em satisfazer os requisitos de qualidade de serviço por conexão individual, e não por agregado de conexões.

Não sendo viável o recurso a sistemas reais de transmissão para análise do comportamento dos algoritmos propostos, foi desenvolvido um modelo de simulação das redes IEEE 802.11 e HIPERLAN/2 que implementa as respectivas especificações, e incorpora os algoritmos de escalonamento propostos. Para cada uma destas redes, o modelo incide com particular detalhe nas funções da camada de ligação de dados e da camada de convergência, tendo igualmente em consideração o *overhead* da camada física. Os resultados obtidos nas simulações permitem avaliar e comparar os desempenhos obtidos com as soluções propostas, tanto para o sistema baseado na rede IEEE 802.11 como para o sistema baseado na rede HIPERLAN/2.

[1] Como o suporte de qualidade de serviço (QoS), o débito de transmissão e o alcance.

[2] O termo *jitter* é utilizado aqui com o sentido de variação do atraso dos pacotes, uma vez que *jitter* também significa variações nas transições cíclicas do relógio extraído da linha de transmissão.

1.3 Organização da tese

Após a presente introdução, o capítulo 2 aborda conceitos básicos associados à operação de redes de comunicação, realçando os aspectos relacionados com as redes de comunicação sem fios.

No capítulo 3, os sistemas de comunicação sem fios são classificados em categorias e suas características são apresentadas, sendo algumas das normas mais representativas em cada categoria descritas com mais pormenor. As características das redes IEEE 802.11 e HIPERLAN/2, propostas para tecnologia de suporte de transmissão do sistema de aquisição de dados e controlo, são analisadas com particular detalhe neste capítulo.

O capítulo 4 discute trabalhos desenvolvidos por outros autores que apresentam relações com o trabalho realizado no âmbito desta tese. Neste capítulo são também apresentados os algoritmos de escalonamento propostos tendo em vista a optimização do transporte do tráfego de tempo real nas redes IEEE 802.11 e HIPERLAN/2.

O capítulo 5 introduz a configuração arquitectónica do sistema analisado e descreve os modelos de simulação de redes de comunicação sem fios desenvolvidos para permitir a análise do desempenho das redes consideradas. Neste capítulo são focados ainda aspectos gerais e particulares associados às simulações dos modelos desenvolvidos.

No capítulo 6 são apresentados os cenários de simulação e os resultados obtidos, tanto com o sistema baseado na rede IEEE 802.11 como com o sistema baseado na rede HIPERLAN/2. Estes resultados servem de base para a discussão e comparação do desempenho destes sistemas em função dos algoritmos de escalonamento de tráfego propostos.

Finalmente, o capítulo 7 apresenta as conclusões e as perspectivas de trabalho futuro.

Capítulo 2

Técnicas comunicacionais

Neste capítulo são introduzidos conceitos que servem de base às redes de comunicação em geral e às redes sem fios em particular, sendo úteis para a compreensão dos restantes capítulos.

2.1 Classificação das redes sem fios

Existem diversas maneiras de se implementar uma rede de comunicação sem fios. Esta secção considera três critérios de classificação destas redes: a estrutura; a comunicação entre as estações; e a coordenação do acesso ao meio. As opções associadas a cada critério são descritas abaixo. Dependendo da aplicação que se deseja dar à rede, umas opções são mais apropriadas do que outras.

Na Figura 2.1 estão representadas as estruturas das redes sem fios, que são geralmente classificadas em duas categorias: redes *ad hoc;* e redes baseadas em infraestrutura.

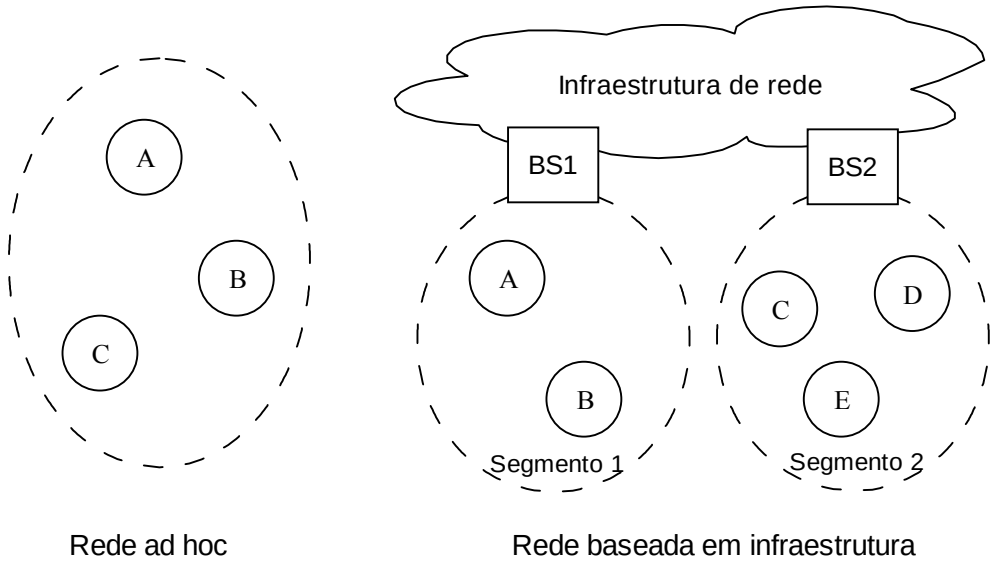


Figura 2.1: Estruturas de redes sem fios.

- **Rede *ad hoc*:** O principal propósito de uma rede *ad hoc* é permitir comunicação entre as estações sem fios que pertençam à mesma. As redes *ad hoc* costumam operar durante períodos relativamente curtos, pelo que geralmente são concebidas de forma a tornar simples e rápido o processo de estabelecimento e dissolução da rede. Um exemplo de aplicação deste tipo de rede é uma reunião de negócios, durante a qual os participantes podem utilizar os seus dispositivos portáteis para partilhar informações.

- **Rede baseada em infraestrutura:** Estas redes são segmentos sem fios de uma rede mais extensa, cujo núcleo é normalmente uma rede cablada. As redes baseadas em infraestrutura possuem uma estação especial, denominada ponto de acesso (AP, *Access Point*) ou estação base (BS, *Base Station*), que serve de interface entre o segmento sem fios e o resto da rede. Exemplos de infraestruturas de rede junto às quais são utilizadas estas redes sem fios são a rede telefónica pública e a rede local de uma empresa.

O meio de transmissão sem fios é um meio de difusão por natureza, ou seja, a transmissão de uma estação pode ser detectada por todas as outras na sua vizinhança[3]. Apesar disso, a rede pode ser configurada de forma que as estações não possam

[3] Desde que não haja estações ocultas na rede.

comunicar directamente entre si. As redes sem fios podem operar utilizando comunicação directa entre as estações ou comunicação centralizada, como é representado na Figura 2.2. Estas configurações correspondem, em termos lógicos, às topologias em barramento e em estrela das redes cabladas, respectivamente.

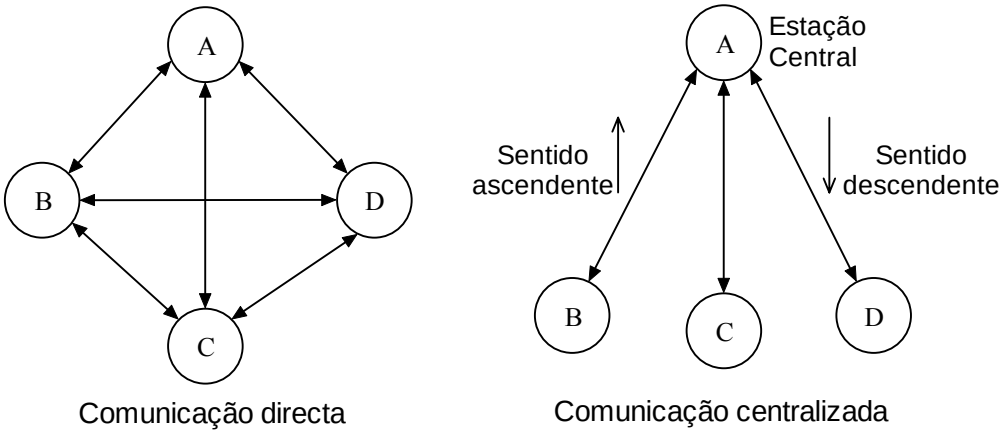


Figura 2.2: Modos de comunicação entre as estações numa rede sem fios.

- **Comunicação directa:** Nesta configuração, as estações que pertencem à rede sem fios comunicam directamente entre si. Nas redes baseadas em infraestrutura que usam esta configuração, apenas as mensagens que têm como origem ou destino o exterior da rede é que precisam passar pela estação base. Este modo de operação é apropriado para aplicações em que a maior parte do tráfego é trocado entre as estações situadas dentro de uma mesma célula.

- **Comunicação centralizada:** Nesta configuração, todas as mensagens que circulam na rede passam por uma estação central[4]. Estas redes admitem apenas dois sentidos de transmissão: o sentido descendente (*downlink*), da estação central para as outras estações; e o sentido ascendente (*uplink*), destas estações para a estação central. A ligação ascendente é do tipo ponto a ponto. Já a ligação descendente geralmente é do tipo ponto-multiponto, permitindo a transmissão simultânea de informação para um grupo de estações na célula (*multicast*) ou para todas as estações (*broadcast*). Para que duas estações terminais da mesma célula

[4] A estação central é tipicamente a estação base de uma rede baseada em infraestrutura, embora esta configuração também seja usada em algumas redes *ad hoc*.

possam comunicar, a estação origem tem que transmitir a sua mensagem para a estação central, que depois retransmite a mensagem para a estação destino. Isso implica a duplicação da largura de banda necessária em relação ao modo de comunicação directa, pelo que a comunicação centralizada é mais apropriada quando a maior parte do tráfego das estações é trocado com a estação central ou com unidades situadas no exterior da célula.

Relativamente à coordenação do controlo do acesso ao meio, esta pode ser feita de forma centralizada ou distribuída.

- **Controlo centralizado:** Neste caso, a rede possui um controlador central (CC, *Central Controller*), com autoridade para regular o acesso ao meio por parte das estações. Uma estação que deseje transmitir deve aguardar a autorização do controlador central. Nas redes baseadas em infraestrutura que operam com controlo centralizado, o papel de controlador central normalmente é desempenhado pela estação base.
- **Controlo distribuído:** Neste caso, o acesso das estações ao meio é decidido de forma distribuída por todas as estações participantes na rede, segundo as regras definidas pelo protocolo de controlo de acesso ao meio utilizado.

Os diferentes tipos de protocolos de controlo de acesso ao meio e suas características são apresentados na secção 2.6.1.

2.2 Qualidade de serviço

Algumas aplicações são relativamente insensíveis à degradação transitória da qualidade de serviço oferecida pela rede. Por exemplo, num serviço de transferência de ficheiros, a redução da largura de banda disponível ou o aumento do atraso dos pacotes podem afectar o desempenho da aplicação, mas não comprometem a sua operação. Devido à capacidade de adaptação a variações na disponibilidade de recursos, tais aplicações são denominadas de elásticas. Essas aplicações contentam-se com um serviço do tipo melhor esforço, no qual a rede compromete-se apenas a tentar transmitir o tráfego gerado pela aplicação, sem no entanto oferecer garantias de desempenho.

Já no caso de aplicações de tempo real, a diminuição da largura de banda disponível ou o aumento do atraso podem inviabilizar a sua operação. Neste caso, a rede necessita de reservar recursos para as aplicações de modo que, mesmo em momentos de maior carga na rede, os requisitos mínimos de desempenho destas aplicações sejam atendidos, ou seja, a rede deve fornecer um serviço com garantias de qualidade de serviço (QoS).

No contexto das redes de comunicação, os parâmetros objectivos[5] de qualidade de serviço [CHA99b] podem ser classificados em quatro categorias:

- **Débito:** é a taxa de transferência de dados entre o emissor e o receptor. O débito máximo que se pode obter está limitado pelo débito da rede. O débito de um fluxo também pode ser afectado pelos outros fluxos que partilham a rede.
- **Atraso:** é o intervalo de tempo que um pacote demora para percorrer a rede desde o emissor até o receptor.
- ***Jitter*:** é a variação do atraso sofrido pelos pacotes no percurso entre o emissor e o receptor.
- **Fiabilidade:** é uma medida da capacidade de entrega dos dados no receptor da mesma forma como foram entregues à rede pelo emissor. O principal parâmetro desta categoria é a taxa de perda de pacotes.

Esses parâmetros permitem quantificar o desempenho fornecido pela rede. Numa rede que ofereça garantias de qualidade de serviço, esses parâmetros servem de base para a negociação, entre as aplicações e a rede, de limites mínimos de desempenho a serem satisfeitos pela rede independentemente da carga. Esses limites podem ser especificados de forma determinística, por exemplo, pela garantia de um débito superior a 100 kbit/s para o fluxo durante o seu tempo de vida; ou probabilística, por exemplo, pela garantia de um atraso inferior a 50 ms para 99 % dos pacotes.

Para que possa oferecer garantias de desempenho para uma aplicação, a rede deve reservar recursos para o respectivo fluxo. A quantidade de recursos a reservar depende

[5] Em oposição aos parâmetros subjectivos associados à percepção da qualidade de serviço por parte do utilizador.

da intensidade de tráfego gerado, pelo que a rede deve ser informada dos parâmetros de tráfego que a aplicação propõe-se a respeitar. Caso a aplicação exceda os limites de tráfego negociados, a rede pode optar por armazenar os pacotes em excesso até estarem em conformidade com o contrato, marcá-los como tendo baixa prioridade para que sejam descartados mais adiante na rede em caso de congestionamento, ou mesmo descartá-los.

A classificação das classes de tráfego e dos parâmetros de qualidade de serviço varia para diferentes organizações internacionais, como o ATM Forum, a ITU, o IETF, o IEEE e o 3GPP [MCD98] [IEE98b] [IET97] [3GP01]. A classificação adoptada nas redes ATM é uma das mais completas, sendo a seguir apresentada.

2.2.1 Qualidade de Serviço em redes ATM

A tecnologia ATM foi concebida desde o início tendo em vista a integração e suporte de serviços com diferentes características e requisitos de qualidade de serviço numa mesma rede de comunicação. Na fase de estabelecimento de conexão, a aplicação negocia com a rede um contrato onde são especificados a categoria de serviço desejada e os respectivos parâmetros de tráfego e de QoS.

Os parâmetros de tráfego especificam os limites impostos à fonte para o tráfego injectado na rede, sendo utilizados para verificação de conformidade do fluxo de células:

- **Peak Cell Rate (PCR):** É o limite superior para o débito da fonte, sendo definido como o inverso do intervalo mínimo entre as células geradas na interface ente as camadas ATM e física.
- **Sustained Cell Rate (SCR):** Representa o limite superior para o débito médio do fluxo.
- **Minimum Cell Rate (MCR):** É o débito mínimo garantido, aplicável para a categoria de tráfego ABR (*Available Bit Rate*).
- **Maximum Burst Size (MBS):** Corresponde ao número máximo de células consecutivas que podem ser transmitidas ao débito máximo (PCR).

- **Cell Delay Variation Tolerance (CDVT)**: Especifica a variação em relação ao instante nominal de inserção de células na rede pela fonte.

Os parâmetros de QoS, por outro lado, definem os requisitos que a rede deve satisfazer no que respeita à qualidade de serviço oferecida:

- **Cell Transfer Delay (CTD):** É o tempo decorrido entre a injecção de uma célula na rede por parte da fonte e a sua entrega ao destino.
- **Cell Delay Variation (CDV):** Exprime a diferença ente os valores máximo e mínimo do CTD.
- **Cell Loss Ratio (CLR):** Expressa a relação entre o número de células que podem perder-se durante o trânsito na rede (por exemplo, devido a congestionamento) e o número total de células do fluxo.
- **Cell Error Rate (CER):** Expressa a relação entre o número de células afectadas por erros e o número total de células.
- **Cell Misinsertion Rate (CMR):** Corresponde à taxa de células inseridas num fluxo e que pertencem a outros fluxos, ou seja, que são incorrectamente encaminhadas.
- **Severely Errored Cell Block Ratio (SECBR):** Exprime a relação entre o número de blocos severamente corrompidos e o número total de blocos.

As categorias de serviço definidas pelo ATM Forum tem implicação na forma como os parâmetros de tráfego e de QoS são utilizados:

- **Constant Bit Rate (CBR):** Destina-se a aplicações de tempo real de débito constante com requisitos exigentes quanto a atrasos e perdas. O débito é caracterizado pelo parâmetro PCR, o atraso pelos parâmetros CTD e CDV e as perdas pelo parâmetro CLR. A rede reserva recursos correspondentes ao PCR mesmo que a aplicação transmita a um débito inferior. Esta categoria é utilizada para emulação de circuitos e transmissão de tráfego de voz, áudio e vídeo de débito fixo.
- **Real-Time Variable Bit Rate (rt-VBR):** Esta categoria destina-se a aplicações de tempo real de débito variável com restrições temporais. As conexões desta

categoria são caracterizadas pelos parâmetros PCR, SCR, MBS, CLR, CTD e CDV. Esta categoria destina-se principalmente a aplicações multimédia, como o transporte de vídeo e áudio interactivos.

- **Non-Real-Time Variable Bit Rate (nrt-VBR):** Esta categoria à semelhante à anterior, porém não apresenta as restrições temporais, sendo caracterizada pelos parâmetros PCR, SCR, MBS e CLR. É adequada para serviços de transporte de áudio e vídeo sem interactividade.
- **Unspecified Bit Rate (UBR):** Esta categoria é utilizada para o transporte de tráfego do tipo melhor esforço, não sendo oferecidas nenhumas garantias de qualidade de serviço.
- **Available Bit Rate (ABR):** Nesta categoria não são dadas quaisquer garantias quanto ao atraso, mas garante-se um débito mínimo (MCR), bem como uma taxa de perdas pequena caso a fonte adapte o seu débito de acordo com a informação de controlo de fluxo enviada pela rede no sentido inverso.

2.3 Escalonamento de tráfego

Quando pacotes associados a diferentes fluxos partilham um mesmo recurso, como, por exemplo, a saída de um multiplexador, torna-se necessário utilizar um algoritmo de escalonamento [KES97] para determinar a ordem em que os pacotes são servidos. No algoritmo de escalonamento mais simples, denominado FCFS (*First Come First Served*), os pacotes dos diferentes fluxos são colocados numa mesma fila de espera, na ordem em que chegam, e são servidos nessa mesma ordem. Este algoritmo, porém, não é capaz de oferecer justiça, protecção entre os fluxos ou garantias de qualidade de serviço.

Para superar essas deficiências, diversos outros algoritmos de escalonamento foram propostos na literatura. Os algoritmos concebidos para o escalonamento de tráfego assíncrono são, na sua maioria, baseados no algoritmo ideal denominado GPS (*Generalized Processor Sharing*), também conhecido pela designação FFQ (*Fluid Fair Queuing*). Em termos lógicos, o GPS mantém uma fila de espera por cada fluxo, e visita as filas ocupadas em sequência, servindo uma quantidade infinitesimal de

dados de cada fila, de modo que, num intervalo de tempo finito, cada fila é visitada pelo menos uma vez. Aos fluxos podem ser associados diferentes pesos, o que permite que a quantidade de dados servida de um dado fluxo seja proporcional ao respectivo peso quando há dados na fila de espera.

Uma forma simples de implementar o algoritmo GPS é através de rotação (*round-robin*). As filas de espera são servidas em sequência, como no GPS, porém, este esquema serve um pacote de cada vez, e não uma quantidade infinitesimal de dados. A rotação é uma boa aproximação do GPS quando os fluxos têm o mesmo peso e os pacotes têm o mesmo tamanho. O algoritmo WRR (*Weighted Round-Robin*) é uma variação da rotação que serve os fluxos na proporção dos seus pesos.

O WFQ (*Weighted Fair Queueing*) é um algoritmo que implementa melhor os critérios definidos pelo GPS, nomeadamente quando os pacotes são de tamanho variável, mas com uma maior complexidade que a do WRR. O algoritmo WFQ calcula o instante em que cada pacote teria o seu serviço terminado, caso o algoritmo GPS fosse usado, e identifica os pacotes com os números calculados, servindo-os na ordem determinada por esses identificadores.

O algoritmo WFQ também pode ser utilizado com tráfego de tempo real. Entretanto, quanto menor o atraso desejado para a conexão, maior terá de ser a largura de banda reservada, mesmo que não seja utilizada [KES97]. Como no meio sem fios a largura de banda é um recurso escasso, este algoritmo de escalonamento não é muito indicado para lidar com tráfego com requisitos de reduzido atraso em redes de comunicação sem fios.

Para o escalonamento de tráfego de tempo real, existem outras alternativas, como o EDD (*Earliest Due Date*), também conhecido como EDF (*Earliest Deadline First*). No EDD, atribui-se um prazo de validade (*deadline*) a cada pacote e o escalonador serve os pacotes pela ordem dos seus prazos de validade. Neste algoritmo, um pacote, ao qual é atribuído um prazo mais próximo do instante de chegada sofre um atraso menor na fila de espera do que outro ao qual tenha sido atribuído um prazo mais distante. Dependendo da carga, pode não ser possível servir todos os pacotes antes de se atingir os respectivos prazos que lhes foram atribuídos.

2.4 Controlo de erros

As redes sem fios estão sujeitas a taxas de erros muito mais intensas e variáveis do que as redes cabladas, devido a diversos factores, como ruído, interferência de outras fontes, obstrução do sinal, desvanecimento (*fading*) e interferência intersimbólica (ISI, *InterSymbol Interference*). Estes dois últimos factores resultam do fenómeno conhecido como propagação multipercurso, que consiste na recepção de um mesmo sinal por múltiplos caminhos de comprimento diferente devido à reflexão do sinal transmitido em objectos, o que faz com que as diferentes cópias do sinal cheguem ao receptor desfasadas entre si. O desvanecimento multipercurso deriva da combinação, construtiva ou destrutiva, das diferentes cópias, que pode provocar flutuações significativas na intensidade do sinal resultante. Já a interferência intersimbólica ocorre quando uma parcela da energia de um símbolo sobrepõe-se à do símbolo seguinte, dificultando a sua detecção.

Existem diversas técnicas de controlo de erros [VAR99] [LIU97] que procuram minimizar os efeitos dos erros no canal sobre a informação transmitida, ao custo do aumento do *overhead*. O controlo de erros pode ser aplicado em diferentes camadas: física, de ligação de dados, de transporte ou de aplicação. Os critérios que condicionam a escolha das técnicas de controlo de erros a empregar numa rede sem fios incluem a largura de banda disponível, o padrão de erros no canal, o tempo de propagação entre o emissor e o receptor, a existência de um canal de retorno, e as possíveis limitações de memória, autonomia e capacidade de processamento das estações. A seguir, descrevem-se algumas das técnicas de controlo de erros utilizadas em redes sem fios.

2.4.1 Equalização

A técnica de equalização é utilizada para compensar a distorção no sinal provocada pela interferência intersimbólica (ISI). O receptor calcula os parâmetros a utilizar na equalização com base nas alterações de amplitude e fase sofridas por uma sequência de treino conhecida, transmitida pelo emissor. Como as características do canal sem fios tendem a variar com o tempo, os parâmetros utilizados na equalização

têm que ser constantemente actualizados, pelo que a sequência de treino costuma ser repetida sempre que há uma transmissão.

2.4.2 Redundância

Nesta técnica, a mesma informação é recebida através de diversas vias independentes. Com isso, a probabilidade de que todas as versões da informação sejam corrompidas por erros é pequena. A diversidade de vias pode ser implementada de várias formas: diversidade temporal, de frequências, de antenas ou de polarização. As duas primeiras técnicas requerem largura de banda extra para a transmissão das diferentes versões, pelo que a sua utilização não é muito difundida em redes sem fios, nas quais a largura de banda é um recurso escasso. Para que um sistema beneficie da diversidade de antenas, os sinais devem apresentar baixa correlação, o que implica que as antenas devem estar separadas de pelo menos metade do comprimento de onda da portadora. Esta técnica é utilizada principalmente no combate ao desvanecimento multipercurso.

2.4.3 Correcção de erros

A técnica de correcção de erros (FEC, *Forward Error Correction*) baseia-se na transmissão de informação redundante pelo emissor, de modo a habilitar o receptor a corrigir os erros nas mensagens recebidas. A técnica de correcção de erros tradicional não adapta dinamicamente o nível de redundância às condições variáveis do canal, pelo que a largura de banda pode ser desperdiçada devido ao *overhead* da técnica de correcção de erros quando as condições do canal são boas. Por outro lado, o nível de redundância pode ser insuficiente para corrigir os erros quando as condições são ruins.

Quando os erros introduzidos pelo canal resultam dum estado de perturbação de duração relativamente curta, a técnica de entrelaçamento (*interleaving*) possibilita a aplicação eficaz da técnica de correcção de erros, ao espalhar os erros que normalmente estariam concentrados em poucos bits. Uma desvantagem da técnica de entrelaçamento é que a sua utilização provoca o aumento do atraso.

2.4.4 Detecção de erros e retransmissão

Na técnica de detecção de erros e retransmissão (ARQ, *Automatic Repeat reQuest*), o emissor inclui em cada pacote um código CRC (*Cyclic Redundancy Check*), calculado em função da informação contida no pacote, que permite ao receptor detectar se o pacote foi corrompido por erros no canal. Em caso positivo, o receptor requisita ao emissor a retransmissão do pacote, seja de forma explícita, pelo envio de uma mensagem de reconhecimento negativo, ou de forma implícita, pelo não envio de uma mensagem de reconhecimento positivo, num intervalo de tempo estabelecido a seguir à recepção do pacote.

Esta técnica é muito mais adaptável às condições variáveis do canal do que a técnica de correcção de erros, porque o seu nível de redundância, expresso pela razão entre retransmissões e transmissões, não é fixo. Por outro lado, a técnica de detecção de erros e retransmissão necessita de um canal de retorno, o que não é necessário na técnica de controlo de erros.

A utilização da técnica de detecção de erros e retransmissão não é aconselhável para a transmissão de tráfego de tempo real quando o tempo de propagação é elevado ou quando o débito de transmissão é pequeno, porque o atraso cumulativo das retransmissões pode fazer com que o atraso final exceda o valor máximo aceitável. Por outro lado, esta técnica é mais fiável do que a correcção de erros para transmissão de tráfego intolerante a erros, pois permite que as retransmissões sejam realizadas até que a informação seja recebida correctamente.

2.4.5 *Hybrid* ARQ (HARQ)

Estes esquemas combinam a correcção de erros com a retransmissão de pacotes. Existem três tipos de esquemas HARQ [ALM02]. No HARQ tipo I, os pacotes incluem um código de correcção de erros. Se este não for suficiente para corrigir os erros no pacote, o receptor requisita a retransmissão do mesmo. Nos esquemas HARQ tipo II e III, introduz-se redundância nos pacotes retransmitidos para correcção de erros. A diferença é que, no HARQ tipo II, o pacote original inclui um código de

detecção de erros (como no ARQ), enquanto no HARQ tipo III o pacote original contém um código de correcção de erros (como no HARQ tipo I).

2.5 Acesso múltiplo

A largura de banda disponível para a operação da rede sem fios pode ser dividida entre as estações, de acordo com as seguintes dimensões: frequência, tempo, código e espaço. Cada uma destas dimensões é associada com uma técnica de acesso múltiplo respectiva [RUB97].

- **Acesso Múltiplo por Divisão de Frequência (FDMA, *Frequency Division Multiple Access*):** Nesta técnica, a largura de banda disponível é repartida em múltiplas bandas de frequências (canais), cada qual com a sua portadora. As estações transmitem em bandas diferentes, o que possibilita a ocorrência de múltiplas transmissões simultâneas no meio.
- **Acesso Múltiplo por Divisão de Tempo (TDMA, *Time Division Multiple Access*):** Neste esquema, as estações partilham a mesma banda de frequências. Tipicamente, apenas uma mensagem consegue ser transmitida com sucesso de cada vez; assim, as transmissões das diferentes estações devem ser programadas para ocorrer em períodos distintos.
- **Acesso Múltiplo por Divisão de Código (CDMA, *Code Division Multiple Access*):** Com esta técnica, as mensagens das diferentes estações são codificadas de tal modo que múltiplas mensagens podem ser transmitidas ao mesmo tempo utilizando a mesma banda de frequências. Com base no código usado pelo emissor, o receptor é capaz de descodificar correctamente a mensagem desejada, apesar da presença de outras transmissões. Entretanto, a codificação faz com que a largura de banda necessária para a transmissão do sinal codificado seja maior do que a necessária para a transmissão do sinal original.
- **Acesso Múltiplo por Divisão de Espaço (SDMA, *Space Division Multiple Access*):** Devido ao efeito de atenuação na propagação das ondas electromagnéticas, a mesma banda de frequências pode ser utilizada em comunicações simultâneas, desde que as estações envolvidas estejam

suficientemente afastadas para que a interferência entre as emissões não seja significativa. As redes celulares aproveitam-se da dimensão espacial ao reutilizarem as bandas de frequências atribuídas para operação segundo padrões geométricos apropriados. Outra forma de explorar a dimensão espacial consiste na configuração de múltiplos sectores numa única célula, que, aliada à utilização de antenas directivas, permite a coexistência de múltiplas transmissões em simultâneo na mesma banda de frequências dentro de uma mesma célula.

A utilização da técnica de acesso múltiplo por divisão de frequência (FDMA), apresenta algumas desvantagens, entre as quais se inclui a pouca flexibilidade na alocação dinâmica de largura de banda às estações de acordo com as suas necessidades, o que é mais problemático quando o tráfego das aplicações possui débito variável. Outra desvantagem consiste na sensibilidade dos sinais transmitidos à interferência de banda estreita. As redes celulares móveis de primeira geração, que empregam tecnologia analógica, são baseadas em FDMA. Nas redes digitais actuais, o FDMA não costuma ser utilizado, a não ser em conjunto com outras técnicas de acesso múltiplo, como o TDMA.

A técnica de acesso múltiplo por divisão de tempo (TDMA) possibilita uma maior flexibilidade na alocação de largura de banda para as diferentes estações quando comparada à técnica de FDMA. Porém, como as estações transmitem sequencialmente, e não em simultâneo, o débito de transmissão é mais elevado, o que aumenta a interferência intersimbólica (ISI)[6]. Praticamente todas as redes sem fios de área local e pessoal existentes, bem como a maior parte das redes celulares móveis de segunda geração, utilizam a técnica de acesso múltiplo por divisão de tempo. O TDMA também é usado em redes de acesso fixo sem fios de banda larga e nas redes celulares móveis de terceira geração, juntamente com o CDMA.

Nas redes sem fios baseadas em TDMA que usam comunicação centralizada, existem duas opções para a multiplexagem das transmissões feitas nos sentidos

[6] A modulação OFDM (*Orthogonal Frequency Division Multiplexing*) [ROH99], utilizada em redes locais sem fios de alto débito, consegue reduzir a interferência intersimbólica através da conversão do fluxo de dados de alto débito da estação em múltiplos fluxos paralelos de mais baixo débito, que são utilizados para modular um igual número de portadoras ortogonais entre si.

ascendente e descendente, denominadas técnicas de *duplex*. Na técnica de divisão na frequência (FDD, *Frequency Division Duplex*), uma banda de frequências é alocada para o tráfego no sentido ascendente e outra para o tráfego no sentido descendente. Já na técnica de divisão no tempo (TDD, *Time Division Duplex*), o tráfego nos dois sentidos é multiplexado no tempo, ocupando a mesma banda de frequências. A técnica de TDD é mais eficiente quando a proporção de tráfego nos dois sentidos é variável.

Existem dos tipos básicos de acesso múltiplo por divisão de código (CDMA), consoante a técnica de espalhamento espectral[7] (SS, *Spread Spectrum*) [GLI97] que é aplicada:

- CDMA por saltos em frequência (FH-CDMA, *Frequency Hopping CDMA*);
- e CDMA por sequência directa (DS-CDMA, *Direct Sequence CDMA*).

Nos sistemas baseados em FH-CDMA, a frequência da portadora no emissor sofre variações discretas e periódicas (dentro de uma banda de frequências predefinida) em função da sequência (código) usada pelo modulador. Para recuperar o sinal modulante, o receptor aplica a mesma sequência, em fase, no desmodulador. Já nos sistemas baseados em DS-CDMA (normalmente de implementação mais complexa), cada bit do sinal é modulado por uma sequência de *chips*[8]. Com isso, o sinal resultante é espalhado por uma banda muito maior do que a necessária para transmitir o sinal original. O receptor utiliza a mesma sequência pseudo-aleatória adoptada pelo emissor para a recuperação do sinal original.

Quando os códigos utilizados pela técnica de CDMA são ortogonais, as transmissões não interferem entre si, mas existe um limite rígido no número máximo de utilizadores que o sistema pode suportar. Quando os códigos não são ortogonais,

[7] Algumas redes locais e pessoais sem fios utilizam o espalhamento espectral para poderem operar sem licença em bandas ISM (*Industrial, Scientific and Medical*), ao mesmo tempo que beneficiam da sua robustez. No entanto, as estações que fazem parte da célula partilham o mesmo código, pelo que o acesso múltiplo não é baseado em CDMA (mas sim em TDMA).

[8] Os bits num código de espalhamento espectral por sequência directa recebem a denominação de *chips*.

não há um limite estrito no número máximo de utilizadores, mas o sistema fica sujeito à interferência mútua entre as fontes, que aumenta linearmente com o número de utilizadores no sistema [SAR00].

Um problema associado ao DS-CDMA é que o receptor tem dificuldade de descodificar o sinal de um emissor distante na presença de um sinal mais forte de outro mais próximo. A resolução do problema requer o ajuste dinâmico da potência de transmissão de cada emissor, feito com o auxílio de informação sobre o nível de sinal fornecida pelo receptor. O DS-CDMA é utilizado em algumas redes celulares móveis de segunda e terceira geração.

2.6 Controlo de acesso ao meio

Num meio partilhado por diversas estações, quando as transmissões de duas ou mais estações sobrepõem-se (colidem), normalmente as mensagens são corrompidas, pelo que os destinatários não conseguem recuperá-las. A retransmissão de mensagens corrompidas aumenta a carga na rede, o que colabora para o aumento da frequência de colisões. Este processo, se não for controlado, pode inviabilizar a comunicação entre as estações. Sendo assim, é necessário que a rede implemente um mecanismo de controlo de acesso ao meio (MAC, *Medium Access Control*) para disciplinar o acesso das estações ao meio e evitar colisões, de forma a melhorar o desempenho do sistema.

Uma grande variedade de protocolos controlo de acesso ao meio tem sido proposta na literatura [CHA00] [ANA98] [AKY99]. Um dos principais critérios para avaliação desses protocolos é o suporte de qualidade de serviço (QoS) oferecido. Neste sentido, duas abordagens diferentes podem ser adoptadas.

- **Diferenciação de serviços:** Neste caso, o tráfego que circula na rede é separado em classes de tráfego, e diferentes prioridades são associadas a cada classe. Essas prioridades permitem que algumas estações tenham maior probabilidade de aceder ao meio do que outras, em função do tráfego que desejam transmitir. Esta abordagem permite oferecer diferentes níveis de qualidade de serviço para diferentes classes de tráfego, porém, as conexões de uma mesma classe não são protegidas umas das outras, pelo que o comportamento agressivo de uma conexão

afecta desfavoravelmente as outras na mesma classe. Mesmo o tráfego de menor prioridade pode afectar a qualidade de serviço das conexões de maior prioridade, dependendo da carga na rede e dos parâmetros do protocolo utilizado.

- **Garantias de qualidade de serviço:** Neste caso, as garantias de qualidade de serviço são fornecidas através de um cuidadoso processo de controlo de admissão de conexões e escalonamento das transmissões, sendo possível satisfazer requisitos de qualidade de serviço por conexão individual. A operação deste tipo de protocolos requer a presença de uma entidade denominada controlador central (CC, *Central Controller*), responsável pelo escalonamento de tráfego e pela reserva de recursos.

Grande parte dos protocolos controlo de acesso ao meio usados em redes locais não fornece suporte de qualidade de serviço, proporcionando apenas um serviço do tipo melhor esforço. Estes protocolos geralmente procuram minimizar o atraso médio dos pacotes, desde o instante em que são colocados na fila de espera do emissor até ao momento em que são entregues com sucesso ao receptor. No entanto, o atraso médio de todos os fluxos de informação tende a aumentar à medida que aumenta a carga global na rede.

Outros critérios utilizados para avaliar o desempenho dos protocolos de controlo de acesso ao meio são a eficiência e a justiça.

- **Eficiência:** A eficiência de um protocolo de controlo de acesso ao meio é uma medida do aproveitamento da largura de banda disponível, sendo normalmente expressa pela razão entre o débito útil e a capacidade do canal. Um protocolo de controlo de acesso ao meio deve procurar maximizar a eficiência sem comprometer a qualidade de serviço oferecida às conexões. Para isso, deve procurar minimizar o *overhead* que introduz.

- **Justiça (*fairness*):** Um protocolo controlo de acesso ao meio é considerado justo se não exibir preferência por nenhuma estação em particular quando múltiplas estações competem pelos recursos, no caso de as diferentes conexões pertencerem a uma mesma classe de tráfego. Entre diferentes classes de tráfego, o protocolo deve atribuir recursos na proporção de suas alocações.

2.6.1 Categorias de protocolos de controlo de acesso ao meio

Os protocolos de controlo de acesso ao meio usados em redes sem fios podem ser classificados em cinco categorias: acesso aleatório, *polling*, reserva fixa, reserva dinâmica implícita e reserva dinâmica explícita.

2.6.1.1 Acesso aleatório

Nestes protocolos, as estações competem pelo acesso ao meio de forma distribuída. A decisão sobre o momento de transmissão é feita com base num algoritmo de resolução de contenção (CRA, *Contention Resolution Algorithm*) específico do protocolo.

O protocolo ALOHA foi o primeiro protocolo de acesso aleatório a ser proposto. Segundo o seu princípio de operação, quando uma estação tem um novo pacote disponível para transmissão, transmite-o imediatamente. A seguir, a estação aguarda durante um período que tem em conta o tempo de propagação entre emissor e receptor pela confirmação do recebimento do pacote, proveniente do receptor. Caso a confirmação não chegue nesse período, o emissor retransmite o pacote após um período aleatório. A colisão, mesmo que parcial, de pacotes transmitidos em simultâneo por diferentes estações faz com que os mesmos sejam corrompidos, pelo que a eficiência teórica máxima deste protocolo é muito baixa (cerca de 18 %).

O protocolo S-AHOHA (*Slotted* ALOHA) é uma variante do protocolo ALOHA na qual o tempo é dividido em *slots* de duração fixa. Quando uma estação quer transmitir um pacote, ela espera pelo início do *slot* seguinte. Isso reduz para metade o período de vulnerabilidade a colisões da transmissão, duplicando a eficiência teórica máxima relativamente ao protocolo ALOHA. Por outro lado, a sua utilização requer a sincronização entre as estações. Este protocolo é usado, por exemplo, na requisição de canais de comunicação em redes celulares móveis.

Caso o tempo de propagação seja pequeno em comparação com o tempo de transmissão dos pacotes, como no caso das redes de área local, quando uma estação começa a sua transmissão, as outras estações na rede podem ficar a saber quase imediatamente e, desta forma, adiar as suas transmissões de forma a evitar a colisão.

Este processo, conhecido como detecção de portadora, está na base do protocolo CSMA (*Carrier Sense Multiple Access*). Este protocolo apresenta um desempenho superior ao do protocolo ALOHA, pois as estações precisam de começar as suas transmissões quase ao mesmo tempo para que ocorra uma colisão. Apesar disso, a probabilidade de colisões ainda pode ser significativa, nomeadamente quando a carga na rede é elevada.

No protocolo CSMA, quando ocorre uma colisão, o meio fica ocupado até que as estações envolvidas terminem suas transmissões, embora os pacotes envolvidos na colisão sejam descartados. Este desperdício de largura de banda pode ser minimizado caso as estações emissoras interrompam as suas transmissões logo que detectem a ocorrência de colisão. Este procedimento é utilizado pelo protocolo CSMA/CD (*Carrier Sense Multiple Access/Collision Detection*), adoptado pela rede Ethernet (IEEE 802.3) [IEE02a].

Infelizmente, a detecção de colisões é impraticável nas redes sem fios. Num meio sem fios, quando uma estação transmite, parte do sinal alcança o estágio de recepção[9]. Devido à proximidade entre os estágios de transmissão e recepção da estação, esta interferência é muito mais forte do que os sinais provenientes de outras estações, inviabilizando a detecção de outras transmissões enquanto a estação transmite. Por isso, as redes locais sem fios utilizam uma variante do protocolo CSMA conhecida pela denominação CSMA/CA (*Carrier Sense Multiple Access/Collision Avoidance*), que procura evitar colisões em vez de detectá-las.

2.6.1.2 *Polling*

Nas redes que utilizam protocolos de *polling* existe uma estação, conhecida pela designação de mestre, que faz o papel de controlador central, ou seja, regula o acesso ao meio por parte das outras estações, que recebem a denominação de escravos. Uma estação só pode transmitir após ser interrogada pelo mestre. Juntamente com a interrogação, o mestre pode incluir dados para a estação. Após a resposta da estação, o

[9] Este fenómeno é denominado *self-interference*.

mestre repete o processo com a estação seguinte na lista de *polling*. A forma mais simples de implementar um protocolo de *polling* consiste em fazer a rotação da interrogação por todas as estações presentes na lista de *polling*, independentemente dos padrões de tráfego. Caso seja frequente a situação na qual uma estação não tenha dados a transmitir em resposta a uma interrogação, a eficiência deste protocolo será baixa, devido ao *overhead* associado às interrogações.

2.6.1.3 Reserva fixa

Os protocolos de controlo de acesso ao meio desta categoria operam com base na reserva de canais de transporte com capacidade fixa para as diferentes conexões. As estações normalmente requisitam os canais de transmissão usando um protocolo de acesso aleatório. Os protocolos de reserva fixa são adequados para o transporte de tráfego de débito constante, mas são ineficientes para o transporte de tráfego de débito variável, pois a largura de banda de um canal não pode ser aproveitada por outras conexões quando o canal não está a ser usado. Mesmo com tráfego de débito constante, a flexibilidade para acomodar conexões com débitos diferentes é pequena. A reserva fixa costuma ser usada em redes de comutação de circuitos, como as redes celulares móveis de primeira e segunda geração.

2.6.1.4 Reserva dinâmica implícita

Estes protocolos foram concebidos com o objectivo de realizar a multiplexagem estatística do tráfego de voz e de dados. A sua motivação principal consiste no suporte de serviços de dados em redes celulares móveis baseadas em TDMA. Nestes protocolos, o tempo é dividido em tramas compostas de *slots*. O período de uma trama coincide com o intervalo de geração dos pacotes de voz, e o tamanho de cada *slot* é apropriado para a transmissão de um pacote de voz ou de dados. As estações competem pelos *slots* vazios, utilizando um processo aleatório. A diferença destes protocolos em relação aos protocolos de acesso aleatório é que um pacote de voz enviado com sucesso num dado *slot* reserva, implicitamente, o respectivo *slot* nas tramas seguintes. Assim, os outros pacotes da conexão podem ser transmitidos livres

de contenção. O *slot* é libertado para utilização das outras conexões a partir do momento em que é deixado vazio. Devido à estrutura rígida da trama, estes protocolos são pouco flexíveis para o transporte de uma mistura de conexões de tempo real com diferentes características. Um exemplo de protocolo de reserva dinâmica implícita é o PRMA (*Packet Reservation Multiple Access*) [GOO89].

2.6.1.5 Reserva dinâmica explícita

Na reserva dinâmica explícita, as estações enviam pedidos de recursos para o controlador central, indicando as suas necessidades de transmissão. Os pedidos de recursos normalmente são feitos usando um mecanismo de acesso aleatório, em *slots* apropriados. Alguns protocolos também permitem o envio de pedidos de recursos em resposta a interrogações do controlador central, evitando assim a contenção, ou a anexação de pedidos às mensagens de dados transmitidas. Com base nos pedidos recebidos, o controlador central realiza o escalonamento do tráfego das estações, tendo em consideração os requisitos de qualidade de serviço das conexões. Periodicamente, o controlador central divulga a informação sobre a atribuição de recursos efectuada, para que as estações possam saber o momento certo para efectuar as suas transmissões. A transmissão de tráfego ATM sem fios (*wireless* ATM) está na base do desenvolvimento destes protocolos. Entretanto, outros tipos de tráfego também podem beneficiar de suas características.

2.6.1.6 Outros protocolos

Além dos protocolos descritos anteriormente, existem ainda os protocolos de passagem de testemunho (*token*), comuns em redes cabladas. Nestes protocolos, a estação em posse do testemunho controla o acesso ao meio. O testemunho é passado de estação para estação, em sequência, para que todas tenham oportunidade de transmitir, pelo que o controlo de acesso é distribuído. Os problemas de eficiência destes protocolos são semelhantes aos dos protocolos de *polling* quando operam com um esquema de rotação. Os protocolos de controlo de acesso ao meio das redes *Token Bus* (IEEE 802.4) [IEE90] e *Token Ring* (IEEE 802.5) [IEE98a] são exemplos de

protocolos de passagem de testemunho utilizados em redes cabladas com topologias em barramento e em anel, respectivamente. No entanto, estes protocolos não são apropriados para redes sem fios [WIL02] [CHA00], pois a perda do testemunho seria frequente, dada a natureza instável das ligações, e o processo de recuperação do testemunho é dispendioso.

2.6.1.7 Conclusões

Os protocolos de acesso aleatório não são adequados para satisfazer os requisitos de qualidade de serviço do tráfego de tempo real, pois cada pacote tem que competir pelo acesso ao meio para que possa ser transmitido. O tempo necessário para a resolução da contenção entre estações depende da carga na rede, pelo que torna complicado garantir limites para o atraso e o *jitter*. O fornecimento de garantias de qualidade de serviço às conexões requer a reserva de recursos e, de preferência, a utilização de controlo centralizado no acesso ao meio. A presença de um controlador central torna possível o escalonamento das transmissões em função dos requisitos das conexões e da disponibilidade de recursos. Das categorias de protocolos apresentadas, a mais adequada para fornecer garantias de qualidade de serviço e lidar com uma mistura de tráfego de débito fixo e variável parece ser a categoria dos protocolos de reserva dinâmica explícita, embora os protocolos de *polling* também mereçam ser considerados.

2.7 Sumário

Este capítulo introduziu conceitos básicos associados aos sistemas de comunicação sem fios que serão apresentados no próximo capítulo. Começou-se pela definição de três critérios de classificação das redes de comunicação sem fios e a apresentação das respectivas opções, que mantém estreita relação com a aplicação que se deseja dar à rede.

Em seguida, foram apresentados aspectos associados à qualidade de serviço e ao escalonamento de tráfego em redes de comunicação, seguindo-se a descrição de

técnicas de controlo de erros utilizadas em redes sem fios, bem como dos diferentes tipos de técnicas de acesso múltiplo. Por fim, abordou-se a problemática do controlo de acesso ao meio, descrevendo-se as diferentes categorias de protocolos e os principais critérios utilizados para a avaliação do seu desempenho.

Capítulo 3

Sistemas de comunicação sem fios

Apresentam-se a seguir diferentes tipos de redes de comunicação sem fios e descrevem-se algumas das implementações mais representativas de cada categoria, com particular ênfase nas redes IEEE 802.11 e HIPERLAN/2, que servem de base às soluções propostas.

3.1 Introdução

3.1.1 Espectro electromagnético

A utilização do espectro electromagnético, tanto pelas redes de comunicação sem fios como por outros sistemas que emitem ondas de rádio, é rigorosamente controlada de modo a possibilitar a partilha do meio de transmissão por todos os utilizadores. Entre as organizações responsáveis pela alocação de bandas de frequência para a operação dos diferentes sistemas, encontram-se o Comité Europeu de Radiocomunicações (ERC, *European Radiocommunications Committee*), vinculado ao CEPT (*European Conference of Postal and Telecommunications Administrations*), na Europa, e a FCC (*Federal Communications Commission*), nos Estados Unidos da América. Normalmente, a operação numa dada banda de frequências requer a negociação com estas organizações e o pagamento de um taxa. Após a análise das características do sistema, como a potência de transmissão e a cobertura geográfica, as

instituições competentes conferem a atribuição da licença de operação, ou seja, o direito de utilização da banda de frequências.

Devido às dificuldades que o processo de licenciamento impõe aos utilizadores individuais, foram definidas bandas de frequências que podem ser usadas sem necessidade de licença, embora não deixem de estar sujeitas a restrições relativamente à potência máxima de transmissão e à técnica de modulação utilizada. Estas bandas são conhecidas pela designação ISM (*Industrial, Scientific and Medical*). Como o nome sugere, essas bandas já eram utilizadas anteriormente por diversos equipamentos, desde aparelhos de instrumentação médica e científica até fornalhas industriais e fornos de microondas. Estes equipamentos, assim como outras redes sem fios que operem na vizinhança, podem provocar interferências nas comunicações de uma rede sem fios. Apesar das técnicas de espalhamento espectral permitirem diminuir os efeitos da interferência, a resolução do problema não é garantida, sendo esta uma desvantagem associada à utilização de uma banda que não requer licença de operação. A banda ISM mais popular, disponível em todo o globo, está localizada nos 2.4 GHz.

3.1.2 Tipos de redes sem fios

As tecnologias de redes sem fios apresentadas neste capítulo são classificadas em quatro categorias:

- os sistemas celulares móveis;
- os sistemas de acesso fixo sem fios;
- as redes de área local sem fios;
- e as redes de área pessoal sem fios.

Os sistemas celulares móveis e os sistemas de acesso fixo sem fios proporcionam o acesso dos utilizadores a uma infraestrutura de rede fixa. Normalmente o equipamento de rede é propriedade de uma operadora, que cobra uma taxa de utilização pelos serviços oferecidos. Os utilizadores precisam adquirir apenas o

equipamento terminal. Estes sistemas geralmente operam em banda de frequências que requerem licença de operação.

Os sistemas celulares móveis surgiram com o objectivo de possibilitar a transmissão de tráfego de voz sem a restrição de mobilidade imposta pelos telefones fixos. O utilizador pode transportar o equipamento terminal consigo, de modo a ter acesso à rede telefónica pública em qualquer local dentro da zona de cobertura da rede. Para possibilitar o aumento da cobertura e do número de utilizadores servidos, a área coberta pela rede é repartida em células[10]. Como o nível de sinal decresce com a distância, as células suficientemente afastadas entre si podem reutilizar as frequências de operação disponíveis sem que haja interferência significativa. Estes sistemas têm evoluído no sentido do aumento do débito disponível para os utilizadores, da diversificação dos serviços oferecidos e da possibilidade de transmissão de tráfego multimédia.

Os sistemas de acesso fixo sem fios permitem o acesso a serviços de comunicação de voz, vídeo e dados. A principal diferença em relação aos sistemas celulares móveis é que a localização do equipamento terminal do subscritor é fixa. Isso permite oferecer serviços de mais alto débito a um custo mais reduzido.

Ao contrário dos sistemas celulares móveis, as redes de área local sem fios surgiram com o objectivo de transmitir tráfego de dados (assíncrono). A principal aplicação consiste na comunicação de dados entre os computadores dentro de uma organização. Nestas redes, como nas redes de área pessoal, o equipamento tende a ser adquirido pelos utilizadores para uso privado, e a banda de frequências de operação pode ser utilizada sem necessidade de licença. Ainda hoje, algumas redes locais sem fios não são muito adequadas para o transporte de tráfego de voz, assim como de outros tipos de tráfego com requisitos de tempo real, devido principalmente aos protocolos de controlo de acesso ao meio utilizados. A evolução das redes locais sem fios tem sido no sentido do aumento do débito disponível e na introdução de suporte de qualidade de serviço para diferentes classes de tráfego.

[10] Vindo daí a designação destes sistemas.

O conceito de redes de área pessoal surgiu mais recentemente, devido à crescente profusão de dispositivos portáteis. Estas redes têm sido concebidas de modo a permitir tanto o transporte de tráfego de voz como de dados. O alcance das transmissões é inferior ao das redes locais, visto que o objectivo das redes de área pessoal é, como diz o seu nome, abranger uma área em torno do utilizador. O custo, a complexidade e o consumo dos equipamentos também tendem a ser inferiores.

O projecto BRAN (*Broadband Radio Access Networks*), do ETSI (*European Telecommunications Standards Institute*), visa desenvolver normas para as novas gerações de redes sem fios de banda larga. Para sistemas com licença, a principal aplicação consiste no fornecimento de serviços para residências e empresas através da cobrança de subscrição. Sistemas sem licença serão utilizados maioritariamente dentro das instalações de empresas e em indústrias.

As redes locais de alto débito HIPERLAN/1 e HIPERLAN/2, desenvolvidas no ambito do projecto BRAN, são descritas nas secções 3.4.2 e 3.7, respectivamente. O projecto BRAN está igualmente a desenvolver os seguintes sistemas.

- **HIPERACCESS**: Proporciona acesso fixo sem fios de banda larga (cerca de 25 Mbit/s) até as instalações do subscritor, sendo capaz de suportar aplicações multimédia. Apresenta um alcance de até 5 km e pode ou não requerer licença para operação, sendo vocacionado para o acesso por parte de residências e empresas de pequeno e médio porte. As bandas de frequência dedicadas à sua operação ainda estão por definir.
- **HIPERLINK**: Destina-se a fornecer ligações de rádio de muito alto débito (até 155 Mbit/s) para interconexão estática de curta distância (até 150 m) entre redes HIPERACCESS ou HIPERLAN, sem necessidade de licença. A operação está prevista para a banda de 17 GHz.

3.2 Redes celulares móveis

3.2.1 Panorama geral

As redes celulares móveis de primeira geração (1G) começaram a operar no início da década de 80. Estes sistemas caracterizam-se pela utilização de tecnologias analógicas para a transmissão do tráfego de voz dos utilizadores. Para dividir a largura de banda disponível entre os múltiplos utilizadores numa mesma célula, estas redes utilizam a técnica de acesso múltiplo por divisão de frequência (FDMA).

Exemplos de sistemas de primeira geração europeus são o *Nordic Mobile Telephone System* (NMT), implementado nos países nórdicos e o *Total Access Communications System* (TACS), utilizado no Reino Unido, Irlanda, Itália, Espanha e Áustria. Na América do Norte, o sistema adoptado é conhecido pela designação *Advanced Mobile Phone System* (AMPS). Actualmente, os sistemas de primeira geração estão obsoletos, tendo sido substituídos quase totalmente pelos sistemas de segunda geração.

Os sistemas móveis celulares de segunda geração (2G), introduzidos no início da década de 90, caracterizam-se pela utilização de tecnologias digitais. Comparados com os sistemas analógicos, os sistemas digitais apresentam uma série de vantagens, destacando-se as seguintes.

- É possível a aplicação de técnicas de correcção de erros ao sinal digital, tornando-o muito mais robusto que o sinal analógico contra os efeitos do ruído e interferência e melhorando a qualidade das ligações.
- Os dados digitais podem ser comprimidos, o que aumenta a eficiência na utilização da largura de banda disponível.
- Ao contrário dos sinais analógicos, os sinais digitais podem ser facilmente criptografados de modo a aumentar a privacidade e segurança das conversações.

Diversos sistemas de segunda geração foram implementados em todo o mundo, sendo o GSM (*Global System for Mobile Communications*) o mais popular. Este sistema, baseado em TDMA, foi implementado inicialmente na Europa, mas

actualmente é utilizado a nível mundial. O GSM é descrito com mais detalhes na secção 3.2.2.

Na América do Norte, diversos sistemas de segunda geração foram adoptados, entre os quais se incluem o D-AMPS (*Digital* AMPS), baseado em TDMA, e o IS-95, também conhecido como cdmaOne, baseado em CDMA.

Apesar do grande sucesso e aceitação no mercado dos sistemas móveis celulares de segunda geração, o débito de transmissão máximo que estes sistemas oferecem é limitado, sendo insuficiente para satisfazer os requisitos cada vez mais exigentes das aplicações e serviços multimédia. Isso motivou o aparecimento dos sistemas móveis celulares de terceira geração (3G), que começaram a ser especificados em 1992, sob a designação IMT-2000 (*International Mobile Telecommunications 2000*) [CHA99a]. Deste esforço resultaram diversos padrões de sistemas 3G, baseados tanto em CDMA como em TDMA, como o UMTS (*Universal Mobile Telecommunications System*), que está a ser implementado na Europa.

Tal como os sistemas celulares móveis, os sistemas de telefonia sem fios (CT, *Cordless Telephony*) destinam-se primariamente à comunicação de voz. Os primeiros sistemas de telefonia sem fios surgiram na década de 70, com o objectivo de substituir a ligação por fios dos telefones convencionais por uma ligação sem fios. Os sistemas CT são formados por um ou mais aparelhos portáteis que se comunicam com uma estação base ligada à rede telefónica fixa. Ao contrário dos sistemas celulares móveis, estes sistemas foram concebidos para proporcionar mobilidade em pequenas áreas, como no interior de residências e escritórios. A evolução dos sistemas de telefonia sem fios levou ao desenvolvimento, na Europa, do sistema DECT (*Digital Enhanced Cordless Telecommunication*), descrito na secção 3.2.3.

3.2.2 GSM

3.2.2.1 Arquitectura de rede

A rede GSM (*Global System for Mobile Communications*) [SCO97] é composta por diversas entidades funcionais, cujas funções e interfaces encontram-se especificadas pelas recomendações da União Internacional de Telecomunicações (ITU, *International Telecommunications Union*) e do ETSI, onde foi definida. A arquitectura da rede GSM pode ser dividida em três partes principais, representadas na Figura 3.1:

- as estações móveis (MS, *Mobile Station*);
- o subsistema da estação base (BSS, *Base Station Subsystem*);
- e o subsistema da rede (NS, *Network Subsystem*).

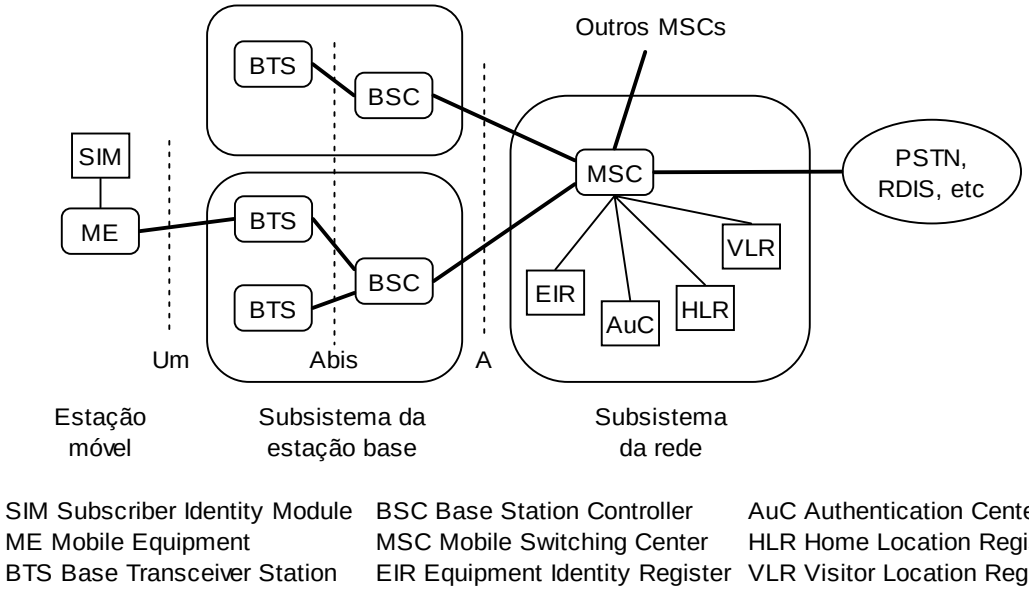


Figura 3.1: Arquitectura da rede GSM.

A estação móvel (MS) é constituída pelo equipamento móvel[11] (ME, *Mobile Equipment*) e por um cartão inteligente denominado módulo de identidade do subscritor (SIM, *Subscriber Identity Module*). O cartão SIM contém os dados associados ao número telefónico do subscritor, permitindo que este tenha acesso aos serviços de rede independentemente do equipamento móvel utilizado. A ligação de rádio entre a estação móvel e o subsistema da estação base (BSS) é denominada interface Um.

O subsistema da estação base é composto por duas partes: as estações base (BTS, *Base Transceiver Station*); e os controladores de estação base (BSC, *Base Station Controller*). A comunicação entre ambas as partes é feita através da interface padronizada Abis, o que possibilita (como no resto do sistema) a interoperação de componentes provenientes de diferentes fornecedores. Os controladores de estação base (BSC) comunicam com a central de comutação móvel (MSC, *Mobile Switching Center*), situada no subsistema de rede (NS), através da interface A.

A central de comutação móvel (MSC) é responsável pela comutação das chamadas dos utilizadores e pelas funcionalidades de gestão das subscrições das estações móveis. Alguns MSCs também efectuam a ligação da rede GSM à rede fixa. O encaminhamento das chamadas dos subscritores na rede GSM é realizado com o auxílio dos registos HLR (*Home Location Register*) e VLR (*Visitor Location Register*), que armazenam informação administrativa dos subscritores, incluindo a localização corrente da estação móvel.

Na rede GSM, existem ainda dois outros registos usados com propósitos de segurança e autenticação. O registo de identidade do equipamento (EIR, *Equipment Identity Register*) contém uma lista de todas as estações móveis válidas (com permissão de utilizar a rede). O registo de autenticação (AuC, *Authentication Center*) é uma base de dados que armazena uma cópia do código secreto contido em cada cartão SIM da rede, sendo utilizado para autenticação dos subscritores e encriptação dos dados.

[11] Tipicamente, um telemóvel.

A ITU definiu inicialmente duas bandas de frequências de 25 MHz para utilização pelo sistema GSM (entre 890 e 915 MHz, no sentido ascendente, e entre 935 e 960 MHz, no sentido descendente). Mais tarde, uma segunda banda de frequências, em torno de 1800 MHz, também passou a ser utilizada pelo GSM na Europa; enquanto nos Estados Unidos uma banda de frequências em torno de 1900 MHz também foi disponibilizada.

3.2.2.2 Ligação de rádio e estrutura de canais

O acesso múltiplo na rede GSM é realizado pela combinação de TDMA e FDMA. A parte referente ao FDMA envolve a divisão da banda de frequências em 124 portadoras, espaçadas de 200 kHz entre si, que são distribuídas pelas estações base. Cada uma dessas portadoras permite o transporte do tráfego de até oito utilizadores em TDMA. As portadoras no GSM são moduladas utilizando GMSK (*Gaussian Minimum Shift Keying*).

Cada portadora é dividida no domínio do tempo em *slots* de duração igual a 15/26 (0.577) ms. Cada *slot* contém 156.25 bits, pelo que o débito de transmissão bruto do GSM é de 270.833 kbit/s. Oito *slots* consecutivos formam uma trama TDMA, com duração de 120/26 (4.615) ms. Cada canal utiliza um *slot* fixo de entre os oito *slots* disponíveis em cada trama TDMA.

Existem dois tipos de canais: os canais de tráfego (TCH, *Traffic CHannel*) e os canais de controlo. Os canais de tráfego (TCH) são utilizados para o transporte de tráfego de voz e de dados entre a estação móvel e a estação base. Os canais usados por uma conexão nas ligações ascendente e descendente (situados em portadoras diferentes) têm um desfasamento de três *slots*, para que a estação móvel não necessite de transmitir e receber em simultâneo, o que simplifica a electrónica utilizada. Os canais de tráfego são organizados em multitramas compostas por 26 tramas TDMA, cuja duração é de 120 ms. Dessas 26 tramas, apenas 24 são utilizadas pelos canais de tráfego, pois uma trama é ocupada pelo sistema com sinalização e outra trama é reservada. A estrutura dos *slots*, tramas e multitramas para canais de tráfego (TCH) na rede GSM representado na Figura 3.2.

O codificador de voz utilizado pelo GSM é denominado RPE-LPC (*Regular Pulse Excited - Linear Predictive Coder*). Na codificação *full-rate*, a informação de voz é dividida é períodos de 20 ms, que são codificados em 260 bits, o que produz um sinal com débito de 13 kbit/s. Existem também codificadores *half-rate*, que produzem um sinal com débito de 7 kbit/s. Mais recentemente, um algoritmo denominado EFR (*Enhanced Full-Rate*), que produz um sinal com melhor qualidade sonora utilizando os mesmos 13 kbit/s, passou a estar disponível no GSM.

Para protecção dos dados contra erros, o GSM utiliza correcção de erros com codificação convolucional em associação com a técnica de entrelaçamento. Os 260 bits na saída do codificador de voz são transformados em 456 bits com a codificação de canal. Como cada amostra é retirada com um intervalo de 20 ms, resulta um débito de transmissão bruto de 22.8 kbit/s por canal. Os 456 bits são divididos em 8 blocos de 57 bits, que são transmitidos em 8 *slots* consecutivos. O formato de *slot* associado aos canais de tráfico (TCH) contém dois blocos de 57 bits para transporte de dados, como foi mostrado na Figura 3.2, pelo que em cada *slot* é feito o entrelaçamento do tráfico de duas amostras do sinal de voz.

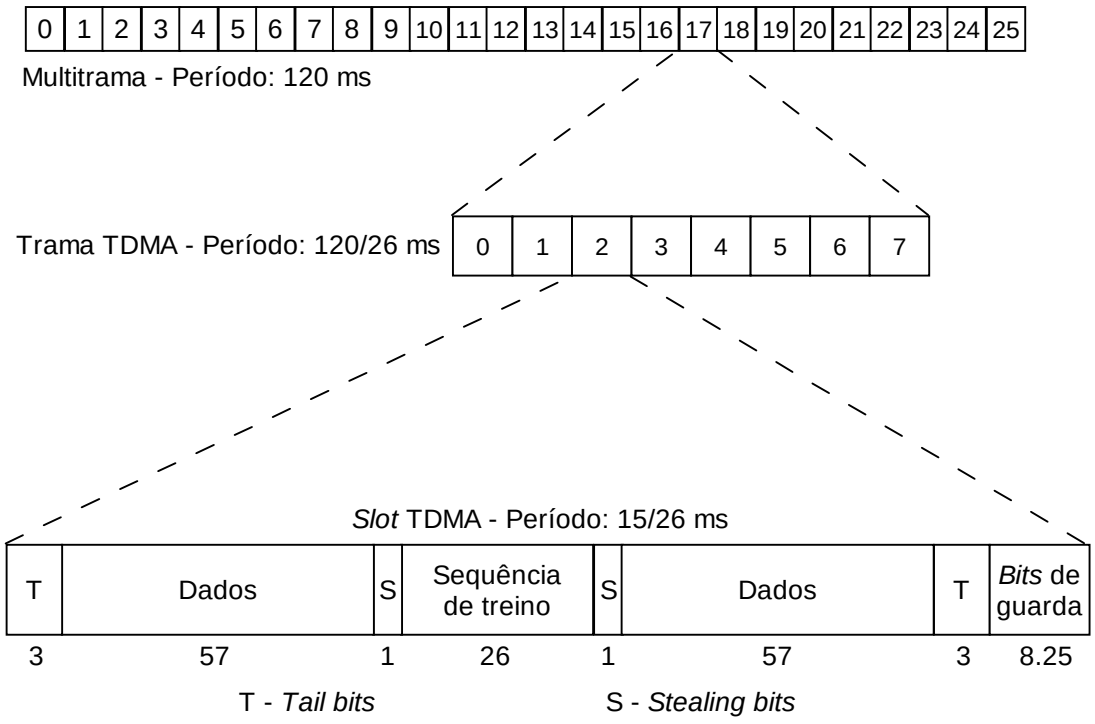


Figura 3.2: Estrutura dos *slots*, tramas e multitramas para canais de tráfego (TCH) na rede GSM.

Os 26 bits da sequência de treino, inseridos no meio de cada *slot* dos canais de tráfego, são utilizados para reduzir a interferência intersimbólica (ISI) por meio da técnica de equalização. Para atenuar o efeito do desvanecimento multipercurso, a técnica de espalhamento espectral por saltos em frequência (FHSS, *Frequency Hopping Spread Spectrum*) pode ser utilizada, embora a estação base não tenha que suportá-la necessariamente. Quando esta técnica é utilizada, as transmissões em cada trama TDMA são feitas utilizando uma frequência de portadora diferente, em conformidade com a sequência de saltos definido pela estação base.

Além dos canais de tráfego, existem diversos tipos de canais de controlo no GSM, entre os quais se incluem os seguintes.

- BCCH (*Broadcast Control CHannel*): É utilizado pela estação base para difusão de informação diversa, como a sua identificação, as alocações de frequência e as sequências de saltos da técnica de FHSS.
- RACH (*Random Access CHannel*): Este canal é utilizado pelas estações móveis para requisitar o acesso à rede. As estações competem pelo acesso utilizando o protocolo *slotted* ALOHA.
- AGCH (*Access Grant CHannel*): É utilizado para notificação da atribuição de um canal de controlo dedicado à estação móvel, para fins de sinalização, a seguir à requisição feita pela estação no canal RACH.
- PCH (*Paging CHannel*): Este canal serve para alertar a estações móvel da chegada de uma chamada.

A estrutura de tramas do GSM é complexa. Além das multitramas de 26 tramas usadas pelos canais de tráfego, existem também multitramas de 51 tramas, usadas pela maioria dos canais de controlo. Estas multitramas são agrupadas em supertramas compostas por 1326 tramas (correspondendo ao mínimo múltiplo comum de 26 e 51), com duração de 6.12 s. No nível superior encontra-se a hipertrama, formada por 2048 supertramas, com uma duração aproximada de 3 horas e 29 minutos.

O GSM define cinco classes de estações móveis em função da potência máxima de pico que pode ser transmitida, que pode variar desde 20 W até 0.8 W. Para minimizar a interferência e diminuir o consumo, tanto a estação móvel como a estação

base operam com a mínima potência necessária para assegurar uma qualidade aceitável do sinal. O nível de potência pode ser aumentado ou diminuído em intervalos de 2 dB, desde a potência de pico da classe até um mínimo de 13 dBm (20 mW). A estação móvel mede o nível do sinal ou a sua qualidade (com base na taxa de erro binários) e passa esta informação para o controlador de estação base (BSC), que decide, em última instância, se o nível de potência deve ser alterado.

3.2.2.3 Gestão da mobilidade

Em termos hierárquicos, no GSM existem quatro níveis de *handover*, referentes à transferência das chamadas entre as seguintes entidades:

- canais (*slots*) numa mesma célula;
- células sob controlo de um mesmo controlador de estação base (BSC);
- células sob controlo de BSCs diferentes, porém, associados a uma mesma central de comutação móvel (MSC);
- e células associadas a diferentes MSCs.

Os dois primeiros tipos, denominados *handovers* internos, envolvem apenas um BSC e são geridos pelo mesmo sem a intervenção da MSC. Os outros dois, denominados *handovers* externos, são geridos pelas MSCs envolvidas. Os *handovers* podem ser iniciados tanto pela estação móvel como pela MSC. Durante os seus *slots* vazios, a estação móvel monitora o canal BCCH de até 16 células vizinhas, formando uma lista das seis melhores candidatas para um possível *handover*. Esta informação é então passada para o BSC e o MSC pelo menos uma vez por segundo.

Para que uma estação móvel possa saber quando há uma nova chamada destinada para si, ela consulta as mensagens de paginação difundidas no canal de paginação (PCH) da célula. Para garantir que a mensagem de paginação seja difundida na célula em que a estação móvel se encontra, uma abordagem extrema obrigaria à transmissão da mensagem de paginação em todas as células que compõem a rede, mas isso representaria um *overhead* significativo. No outro extremo, a estação móvel poderia informar a sua localização a cada mudança de célula, mas isso também teria impacto

na eficiência do sistema, nomeadamente se as mudanças fossem frequentes. Uma solução de compromisso, utilizada pelo GSM, baseia-se no agrupamento das células em áreas. As mensagens de actualização de localização apenas são necessárias quando a estação móvel muda de área, e as mensagens de paginação são enviadas apenas para as células da área em que a mesma se encontra.

3.2.2.4 Extensões do GSM

Além do transporte do tráfego de voz, as redes celulares móveis de segunda geração (2G) possibilitam a transmissão de dados, porém com débitos muito baixos (na ordem dos 10 kbit/s). Com o desenvolvimento de novos serviços e aplicações, e com o aumento do tráfego, nas actuais redes, surgiram extensões aos sistemas 2G que suportam débitos de transmissão de dados mais elevados. Estas extensões representam um passo intermediário em direcção aos sistemas celulares de terceira geração (3G), recebendo por isso a denominação de sistemas 2.5G. Duas destas extensões aplicadas ao GSM são o *High Speed Circuit Switched Data* (HSCSD) e o *General Packet Radio Service* (GPRS) [CAI97].

O *High Speed Circuit Switched Data* é uma extensão bastante simples do GSM. Tal como este, o HSCSD é basicamente uma técnica de comutação de circuitos. Ao contrário do GSM, porém, o HSCSD permite a alocação de mais de um *slot* por trama TDMA para um mesmo utilizador (até quatro *slots* no total), possibilitando assim a obtenção de um débito de transmissão mais elevado. O suporte a ligações assimétricas também é fornecido, de modo que o débito das ligações descendente e ascendente pode ser diferente.

A operação do *General Packet Radio Service* (GPRS) baseia-se no mesmo princípio do HSCSD: a alocação de múltiplos *slots* de uma trama TDMA para um utilizador, de forma a aumentar o seu débito de transmissão de dados. Porém, enquanto o GSM e o HSCSD utilizam a técnica de comutação de circuitos, o GPRS utiliza a técnica de comutação de pacotes. No GPRS, os *slots* são alocados para um utilizador apenas quando os dados estão a ser efectivamente transmitidos, ao contrário do que acontece no GSM e no HSCSD, em que os *slots* são alocados estaticamente durante toda a conexão, mesmo durante os períodos de inactividade. Deste modo, no

GPRS os utilizadores podem estar permanentemente conectados sem serem taxados pelo tempo de conexão, mas somente pelo tráfego transmitido ou recebido. O GPRS permite alcançar débitos de transmissão de até 115.2 kbit/s.

3.2.3 DECT

O sistema DECT (*Digital Enhanced Cordless Telecommunication*) [ETS01c] [DEC97b] opera na banda de frequências entre 1880 e 1900 MHz utilizando modulação GFSK (*Gaussian Frequency Shift Keying*). As células são compostas por uma estação base, também chamada de parte fixa (FP, *Fixed Part*) e um conjunto de terminais, também denominados parte portátil (PP, *Portable Part*).

O esquema de acesso múltiplo utilizado pelo DECT é denominado MC/TDMA/TDD (*Multi Carrier/Time Division Multiple Access/Time Division Duplex*). O sistema DECT opera com dez portadoras, como ilustra a Figura 3.3. Em cada portadora, o tempo é dividido em tramas com a duração de 10 ms, sendo cada trama composta por 24 *slots*.

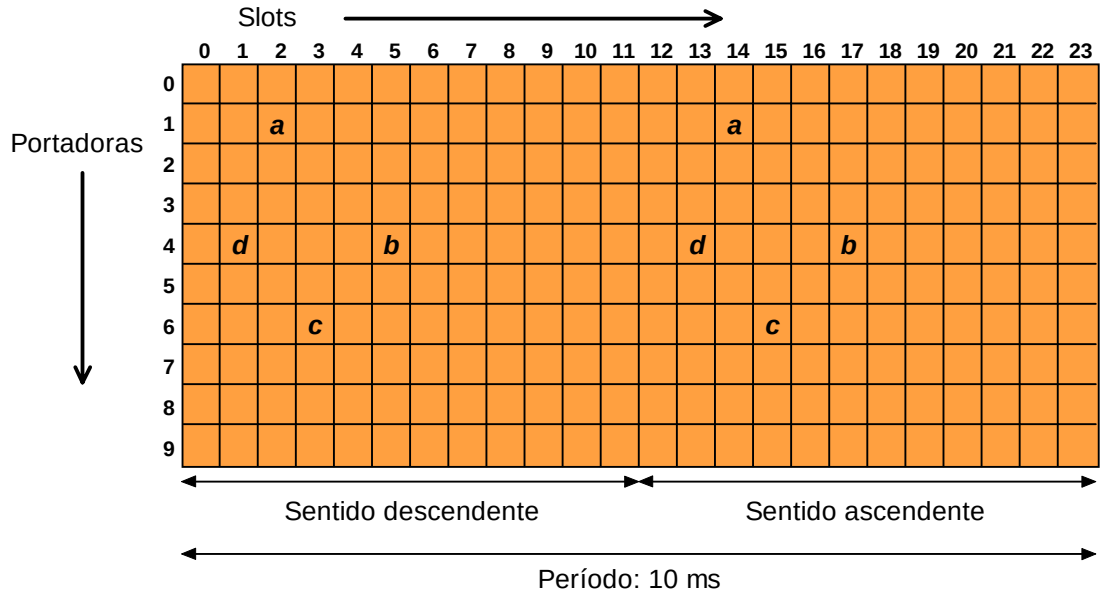


Figura 3.3: Divisão da largura de banda disponível em canais adoptada pelo sistema DECT.

O controlo de acesso ao meio é centralizado, sendo gerido pela estação base. A comunicação também é centralizada, ou seja, os terminais não comunicam directamente entre si. O sistema opera numa configuração baseada em infraestrutura, normalmente ligado à rede telefónica fixa, visto que a principal aplicação do DECT consiste na comunicação telefónica sem fios (CT, *Cordless Telephony*).

No serviço básico de voz, as tramas são dividida em duas partes iguais, sendo os 12 primeiros *slots* utilizados para transmissão no sentido descendente, pela estação base, e os 12 restantes para transmissão no sentido ascendente, pelos terminais. Um par de *slots* separados de 5 ms em cada trama proporciona comunicação simétrica com um débito de 32 kbit/s. Esta configuração é usada normalmente para o transporte de tráfego de voz, utilizando para isso a técnica de codificação de voz ADPCM (*Adaptive Differential Pulse Code Modulation*) [ITU90].

O DECT também possibilita a ocupação assimétrica de *slots*, voltada para transporte de dados. Existem dois formatos de pacote na camada física que podem ser apropriados para a comunicação de dados: o formato P32, que ocupa um *slot;* e o formato P80, que ocupa dois *slots*. A combinação de múltiplos *slots* numa mesma trama pode ser utilizada para aumentar o débito de transmissão de uma conexão. Entretanto, as conexões que utilizam múltiplos *slots* são mais difíceis de gerir, nomeadamente quando é necessária a retransmissão de pacotes corrompidos por erros no canal [BER97]. O débito básico para transmissão de dados, utilizando correcção de erros, é 24 kbit/s [SIL98]. Com a utilização de múltiplos *slots*, conexões assimétricas com débitos de até 552 kbit/s são possíveis numa mesma portadora.

O DECT possui capacidade dinâmica de selecção de canais (DCS, *Dynamic Channel Selection*): os terminais monitorizam frequentemente a qualidade do sinal e, com base nisso, seleccionam o melhor canal disponível. O alcance do DECT estende-se a centenas de metros.

3.3 Acesso fixo sem fios

3.3.1 Panorama geral

Os sistemas de acesso fixo sem fios (FWA, *Fixed Wireless Access*), também conhecidos pela denominação lacete local sem fios (WLL, *Wireless Local Loop*), destinam-se a fornecer serviços de comunicação, mediante subscrição, a equipamentos terminais situados em locais fixos, como residências ou escritórios. Os terminais comunicam com uma estação base, que proporciona o acesso dos terminais à infraestrutura de rede. A cobertura proporcionada pela estação base pode estender-se até algumas dezenas de km, dependendo da frequência de operação do sistema.

Os sistemas de acesso fixo sem fios podem ser classificados em duas categorias: sistemas de banda estreita e de banda larga. Enquanto os sistemas de banda estreita oferecem uma alternativa de ligação à rede telefónica, os sistemas de acesso sem fios de banda larga (BWA, *Broadband Wireless Access*) são capazes de fornecer serviços de voz, vídeo e dados de alto débito, incluindo o acesso de banda larga à Internet.

Espera-se, nos próximos anos, um grande crescimento no mercado dos sistemas de acesso fixo sem fios. Grande parte deste crescimento ocorrerá nos países em desenvolvimento, onde uma parcela significativa da população carece de ligação à infraestrutura da rede telefónica. Outros mercados atraentes para estes sistemas são as zonas rurais e as áreas remotas, onde as receitas obtidas com o reduzido número de potenciais subscritores pode não compensar os custos da instalação de cabos.

Estes sistemas podem ser instalados rapidamente quando comparados com os sistemas cablados. As principais considerações a ter são a aquisição de permissão de operação numa dada banda de frequências e a necessidade de encontrar um ponto elevado para a instalação da antena da estação base. O custo total também tende a baixar, apesar da maior complexidade dos equipamentos, por evitar-se a passagem de cabos. Além disso, os equipamentos de rádio necessitam de ser instalados pelos subscritores apenas na altura da ligação ao serviço, enquanto nos sistemas cablados a instalação dos cabos nos prédios normalmente é feita em antecipação à subscrição do serviço.

Ao contrário do que acontece nos sistemas celulares móveis, a localização dos equipamentos terminais nos sistemas de acesso fixo sem fios é estável. Assim, a instalação das estações base apenas precisa de proporcionar cobertura para as áreas em que a procura imediata dos serviços é necessária. Outra diferença é que o equipamento de rádio pode ser orientado de modo a maximizar a qualidade do sinal durante a instalação. Se necessário, uma antena directiva apontando para a estação base mais próxima pode ser utilizada para melhorar a relação sinal ruído, reduzir a potência de transmissão necessária ou aumentar o alcance das transmissões. Além disso, o efeito do desvanecimento multipercurso tende a ser menor nos sistemas de acesso fixo sem fios.

3.3.2 Sistemas de banda estreita

A principal aplicação dos sistemas de acesso fixo sem fios de banda estreita consiste no fornecimento de acesso à rede telefónica pública (PSTN, *Public Switched Telephone Network*) aos subscritores. Para isso, utilizam-se as ondas de rádio em substituição dos fios de cobre, em parte, ou em todo o percurso entre o equipamento terminal e a central telefónica. Estes sistemas podem ser implementados utilizando diversas tecnologias, desde sistemas de telefonia sem fios (CT, *Cordless Telephony*), como é o caso do DECT, passando por sistemas celulares móveis e por sistemas proprietários. O equipamento terminal destes sistemas tanto pode ser um telefone sem fios, permitindo boa mobilidade, como um equipamento de mesa que integre o telefone e o rádio, ou ainda um equipamento de rádio que forneça uma ou mais linhas e permita ligar telefones convencionais. O equipamento de rádio pode ser instalado no interior ou no exterior do prédio, podendo operar ligado à rede eléctrica ou com baterias. A diversidade de equipamentos possíveis e os variados níveis de serviço que podem ser suportados derivam das diferentes tecnologias de rádio que podem ser utilizadas.

3.3.3 Sistemas de banda larga

Relativamente aos sistemas de acesso sem fios de banda larga, a FCC (*Federal Communications Commission*) alocou cinco bandas de frequência na região entre 2.15 GHz e 2.68 GHz para a operação dos sistemas MMDS (*Multi-channel Multipoint Distribution Service*) nos Estados Unidos da América. Estes sistemas tornaram-se fortes competidores dos sistemas de TV por cabo na oferta de serviços em zonas rurais e áreas remotas. A cobertura de uma célula no sistema MMDS pode atingir até 50 km.

Por outro lado, os sistemas LMDS (*Local Multipoint Distribution Service*) destinam-se a operar na região do espectro acima de 25 GHz. Nos Estados Unidos, um total de cerca de 1.3 GHz de largura de banda foi alocado, na região em torno de 30 GHz, para a operação de sistemas LMDS, enquanto que na Europa a alocação mais significativa foi feita na região em torno de 40 GHz.

A operação na gama das ondas milimétricas impõe algumas restrições aos sistemas LMDS. O efeito da precipitação atmosférica provoca uma forte atenuação no sinal. As ondas milimétricas também são bloqueadas mais facilmente por objectos, pelo que a operação dos sistemas LMDS requer a existência de linha de vista entre emissores e receptores. Como o comprimento de onda é menor do que no caso dos sistemas MMDS, a atenuação com a distância é mais significativa, pelo que o alcance dos sistemas LMDS é limitado a menos de 8 km, o que faz com que seja necessário um número maior de estações base para cobrir uma mesma área. O custo das estações base e dos equipamentos terminais também é mais elevado, devido à necessidade de operação em frequências mais elevadas. Por outro lado, devido à extensa região do espectro disponível, os sistemas LMDS têm a possibilidade de disponibilizar uma largura de banda muito superior ao que pode ser fornecido pelos sistemas MMDS, podendo atingir taxas de transmissão na ordem dos 155 Mbit/s.

Em 1999, o comité de normalização de redes de área local e metropolitana IEEE 802 instituiu o grupo de trabalho 802.16. O objectivo deste grupo consiste em desenvolver projectos que resultem em especificações de sistemas de acesso sem fios de banda larga com as seguintes características [MAR99].

- Ligações de rádio nas regiões de microondas e ondas milimétricas.

- Utilização de espectro sob licença.
- Operação em escala metropolitana.
- Oferta de serviços de rede públicos, mediante o pagamento de uma taxa por parte dos utilizadores.
- Arquitectura do tipo ponto-multiponto com antenas fixas sobre telhados ou torres.
- Capacidade de transporte eficiente de tráfego heterogéneo com suporte de qualidade de serviço (QoS)
- Capacidade de transmissão de tráfego banda larga (débitos acima de 2 Mbit/s).

Estas especificações englobam sistemas operando na região do espectro entre 2 e 66 GHz. O controlo de acesso ao meio é feito pela estação base, que dispõe de um conjunto de esquemas de escalonamento à sua disposição de modo a optimizar o desempenho do sistema. O protocolo de controlo de acesso ao meio é orientado à conexão e proporciona suporte de QoS. As especificações definem ainda camadas de convergência baseadas em ATM e em pacotes, para interacção com as camadas superiores.

No âmbito do projecto BRAN, o ETSI tem vindo a desenvolver normas para as novas gerações de redes sem fios de banda larga, entre as quais se incluem normas para sistemas de acesso fixo sem fios de banda larga, sob a denominação HIPERACCESS, cujo objectivo é proporcionar taxas de transmissão até 25 Mbit/s, com um alcance de cerca de 5 km, sendo voltado para o acesso por parte de residências e empresas de pequeno porte. As bandas de frequência destinadas à sua operação ainda estão por definir.

3.4 Redes de área local

3.4.1 Panorama geral

O mercado de redes de área local sem fios (WLAN, *Wireless Local Area Network*) teve um primeiro impulso em meados da década de 80, com a decisão da

FCC (*Federal Communications Commission*), nos Estados Unidos, de autorizar a utilização pública das bandas ISM (*Industrial, Scientific and Medical*), eliminando com isso a necessidade de obtenção de licenças para a operação dos produtos WLAN. Entretanto, a falta de normas fez com que aparecessem no mercado diversos produtos proprietários incompatíveis entre si, limitando o desenvolvimento da indústria de redes locais sem fios.

A primeira tentativa de definição de um padrão de redes locais sem fios foi feita no final da década de 80. O grupo de trabalho IEEE 802.4 foi mandatado para desenvolver um protocolo de controlo de acesso ao meio baseado em passagem de testemunho para utilização em redes WLAN. Porém, este grupo chegou a conclusão de que o mecanismo de passagem de testemunho era ineficiente para utilização em redes sem fios, pelo que o desenvolvimento de um mecanismo alternativo foi sugerido.

Na sequência desta opção, o comité executivo IEEE 802 decidiu criar o grupo de trabalho 802.11. Com base nos produtos existentes no mercado e no esforço de investigação, este grupo concluiu a primeira versão das normas IEEE 802.11 em 1997. Esta norma define três opções de camadas físicas, sendo que duas delas operam na banda ISM de 2.4 GHz utilizando espalhamento espectral: uma por saltos em frequência (FHSS) e a outra por sequência directa (DSSS, *Direct Sequence Spread Spectrum*). Uma terceira opção de camada física foi definida para operação na banda de infravermelhos. O débito máximo especificado por esta norma situa-se em 2 Mbit/s a nível da camada física, sendo o controlo de acesso ao meio baseado num protocolo de acesso aleatório do tipo CSMA/CA. Um outro protocolo baseado em *polling*, de implementação opcional, também é disponibilizado para suporte a tráfego isócrono.

Em 1999, dois suplementos à versão original da norma foram aprovados. O IEEE 802.11b, baseado na camada física que opera na banda ISM de 2.4 GHz e utilizando espalhamento espectral por sequência directa (DSSS), expande o débito máximo a 11 Mbit/s, mantendo porém a compatibilidade com a versão original baseada em DSSS. Por outro lado, o IEEE 802.11a opera na banda de 5 GHz com modulação OFDM, sendo capaz de atingir um débito máximo de 54 Mbit/s. Todas estas versões operam com os mesmos protocolos de controlo de acesso ao meio, embora os seus parâmetros

sejam diferentes em função da camada física utilizada. O IEEE 802.11 é descrito com detalhes na secção 3.6.

Paralelamente ao desenvolvimento do IEEE 802.11, outro padrão de redes de área local sem fios, denominado HIPERLAN (*High Performance European Radio LAN*), foi desenvolvido pelo comité técnico RES10 (*Radio Equipment and Systems*) do ETSI. Esta primeira versão do HIPERLAN foi concebida para operar na banda de 5.2 GHz com a utilização de modulação de banda estreita, oferecendo um débito de transmissão da ordem de 25 Mbit/s. O controlo de acesso ao meio é baseado num protocolo de acesso aleatório, denominado EY-NPMA, que permite a atribuição de diferentes níveis de prioridade aos pacotes que competem pelo acesso ao meio. Este sistema, conhecido por HIPERLAN/1, é descrito na secção 3.4.2.

Após a definição das normas do HIPERLAN/1, o ETSI decidiu unificar o trabalho de normalização em redes sem fios de banda larga, incluindo as redes locais sem fios e as redes de acesso fixo sem fios, no projecto BRAN (*Broadband Radio Access Networks*). Sob a alçada deste projecto, foi desenvolvido outro padrão de redes locais sem fios de alto débito, denominado HIPERLAN/2, concebido para o transporte eficiente de células ATM (assim como outros tipos de tráfego, como pacotes IP), pelo que também costuma ser classificado como uma rede ATM sem fios (WATM, *Wireless* ATM). O HIPERLAN/2 opera na banda de 5 GHz utilizando modulação OFDM, permitindo alcançar um débito máximo de 54 Mbit/s ao nível da camada física, tal como o IEEE 802.11a. Ao contrário deste, o controlo de acesso ao meio do HIPERLAN/2 é baseado num protocolo de reserva dinâmica explícita. A utilização deste protocolo permite oferecer um suporte de qualidade de serviço (QoS) muito mais completo. A secção 3.7 apresenta uma descrição detalhada do HIPERLAN/2.

3.4.2 HIPERLAN/1

O HIPERLAN/1, desenvolvido pelo comité técnico RES10 do ETSI, utiliza modulação de banda estreita na banda de frequências de 5.2 GHz. Este sistema divide a largura de banda disponível em cinco canais, com as frequências centrais separadas por 23.5 MHz. A operação nos canais 1 a 3 é isenta de licença a nível global, enquanto os canais 4 e 5 não se encontram disponíveis em todos os países. As

transmissões são efectuadas utilizando dois débitos diferentes: o débito baixo (1.47 Mbit/s) é utilizado para transmitir os cabeçalhos dos pacotes e os pacotes de reconhecimento, com a utilização de modulação FSK (*Frequency Shift Keying*); enquanto o débito alto (23.4 Mbit/s) é utilizado para a transmissão dos dados, com a utilização de modulação GMSK (*Gaussian Minimum Shift Keying*). O PDU da camada física é formado por um cabeçalho de débito baixo, 450 bits de treino de débito alto (usados para equalização) e 496 x *n* bits de débito alto para o *payload* mais um número variável de bits de *padding*.

O protocolo de controlo de acesso ao meio do HIPERLAN/1 é baseado num esquema de detecção de portadora e resolução de contenção denominado EY-NPMA (*Elimination Yield - Non-Preemptive Priority Multiple Access*) [FU96]. A transmissão de dados deve ser reconhecida pela transmissão de uma trama ACK por parte da estação destino decorrido um período denominado IFS (*InterFrame Space*). Quando uma estação, que deseja transmitir uma trama de dados, detecta que o canal está livre, por um período correspondente à transmissão de 1700 bits de débito alto, pode começar imediatamente a sua transmissão. Por outro lado, quando a estação deseja transmitir uma trama e o canal está ocupado, inicia-se o denominado período de sinalização activo.

O protocolo EY-NPMA opera em ciclos, constituídos por um período de sinalização activo ao que se segue a transmissão dos dados. O período de sinalização é constituído de três fases, como é representado na Figura 3.4. A primeira fase (*prioritization*) é composta por *slots* de prioridade. A segunda fase (*elimination*) contém o pulso de eliminação e a terceira fase (*yield*) é formada pelos *slots* de *yield*.

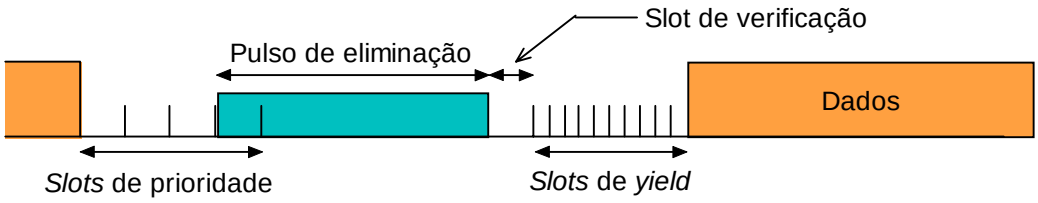


Figura 3.4: Exemplo de funcionamento do protocolo EY-NPMA da rede HIPERLAN/1.

Decorrido um período IFS desde a última trama, as estações no HIPERLAN/1 passam a monitorar o canal durante um certo número de *slots* de prioridade. O número

de *slots* é determinado pela prioridade no acesso ao canal, relacionado com o tipo de tráfego transportado pelo pacote. No total, foram definidos cinco níveis de prioridade, de 0 (maior prioridade) a 4. A prioridade dos pacotes aumenta à medida que o seu tempo de vida decresce. Com isso, o HIPERLAN/1 procura suportar tráfego com requisitos de tempo real. Uma estação, cujo pacote a transmitir tenha prioridade *n,* monitora o canal durante 10 *n* μs antes de emitir o pulso de eliminação. Caso outra estação com maior prioridade comece a emitir antes, a estação adia a sua transmissão.

Desta forma, na segunda fase apenas as estações com mesma prioridade competem pelo canal, pois todas estações com menor prioridade desistiram. As estações que restam começam a emitir seus pulsos de eliminação ao mesmo tempo, mas param de emitir em instantes diferentes, dependendo da duração do pulso, que é definida de forma aleatória. Quando uma estação interrompe sua emissão, volta a monitorar o canal, durante um *slot* de verificação. Caso outra estação ainda esteja emitindo o seu pulso de eliminação, a estação também adia a sua transmissão.

A segunda fase elimina, potencialmente, mais algumas estações na competição pelo acesso ao canal, mas ainda existe a hipótese de duas (ou mais) estações escolherem uma mesma duração para os respectivos pulsos de eliminação. Neste caso, elas seriam incapazes de detectar a transmissão uma da outra ao final da segunda fase. Assim, na terceira fase, as estações que restam ao fim da segunda fase escolhem um período aleatório para monitorização do canal antes de começar a transmissão dos dados. A estação que tenha escolhido o período mais curto vai transmitir, enquanto as restantes vão detectar a ocupação do canal e adiar suas transmissões. As estações que adiaram as suas transmissões durante o período de sinalização activo são livres de competir pelo canal no próximo ciclo de acesso.

O desempenho protocolo de controlo de acesso ao meio do HIPERLAN/1 é afectado pelo problema da estação oculta, que é agravado à medida que se aumenta a intensidade da carga na rede [WEI97]. Entretanto, ao contrário do IEEE 802.11, este sistema não apresenta nenhuma solução para o problema ao nível da camada de controlo de acesso ao meio.

O HIPERLAN/1 pode operar tanto em modo baseado em infraestrutura como em modo *ad hoc*. Além disso, ele suporta a operação por saltos múltiplos (*multihop*), na

qual um pacote pode ser transmitido desde a origem até o destino passando por estações repetidoras sem fios (*forwarders*), sem necessidade de infraestrutura adicional. Cada estação escolhe uma estação vizinha como repetidor, transmitindo-lhe os pacotes cujos destinatários estão fora do seu alcance. Este sistema requer que os repetidores mantenham uma base de dados de encaminhamento de pacotes actualizada, o que é problemático, pois a topologia de uma rede sem fios é dinâmica devido à mobilidade das estações. Além disso, a manutenção da base de dados de encaminhamento por parte dos repetidores requer, periodicamente, a troca de informação com os seus vizinhos, o que diminui o débito útil alcançável pela rede. As estações no HIPERLAN/1 não são obrigadas a desempenhar o papel de repetidor.

3.5 Redes de área pessoal

3.5.1 Panorama geral

O conceito de rede de área pessoal (PAN, *Personal Area Network*), ou rede pessoal, refere-se à comunicação entre múltiplos dispositivos que se encontram nas proximidades de um indivíduo. A motivação para o aparecimento das redes pessoais deve-se ao facto de, no seu dia a dia, cada vez mais as pessoas passarem a conviver com uma maior diversidade de dispositivos electrónicos, à medida em que estes dispositivos tornam-se cada vez menores e mais baratos. As redes pessoais visam proporcionar o intercâmbio de informação entre esses dispositivos, quando necessário, de uma forma conveniente para o utilizador. A comunicação sem fios é apropriada neste caso, pois evita que o utilizador tenha frequentemente de conectar e desconectar os cabos de ligação entre os dispositivos, dado que as conexões entre os mesmos tendem a ser de curta duração. Além disso, a mobilidade dos dispositivos é francamente favorecida.

Conceptualmente, a principal diferença entre as redes pessoais sem fios (WPAN, *Wireless Personal Area Network*)[12] e as redes locais sem fios (WLAN) é que as primeiras tendem a ser centradas em torno de um utilizador, enquanto que as últimas costumam servir múltiplos utilizadores. As redes pessoais costumam ter em comum um alcance limitado a poucos metros e uma configuração simples e rápida das ligações. Entre os dispositivos que podem ser ligados numa rede pessoal encontram-se computadores portáteis e de mesa, *palmtops*, teclados, ratos, impressoras, pontos de acesso a redes locais, telemóveis, auriculares, câmaras fotográficas e de vídeo, relógios de pulso, entre outros.

Para comunicação entre dispositivos portáteis a curtas distâncias, um dos sistemas muito utilizado actualmente baseia-se nas normas emitidas pela *Infrared Data Association* (IrDA) [IRDA]. Este sistema opera por comunicação óptica na banda de infravermelhos, proporcionando ligações ponto a ponto com uma abertura do feixe de cerca de 30 graus, alcance limitado a cerca de 1 m e débito máximo de 4 Mbit/s. Entre as vantagens desta tecnologia estão o seu baixo custo e consumo. Por outro lado, o IrDA requer a existência de linha de vista entre os dispositivos, oferece um alcance pequeno e reduzida mobilidade.

Desenvolvido por iniciativa de membros da indústria de electrónica e telecomunicações, o Bluetooth [HAA00] é outro exemplo de rede pessoal sem fios. As suas características principais são a operação na banda ISM de 2.4 GHz com a utilização da técnica de espalhamento espectral por saltos em frequência (FHSS), um alcance normal de até 10 m, o suporte de canais síncronos e simétricos de 64 kbit/s para transmissão de voz, e de canais assíncronos para transmissão de dados. Estes últimos oferecem débitos assimétricos de até 723.2 kbit/s, numa direcção, e 57.6 kbit/s, na outra, ou débitos simétricos de até 432.6 kbit/s. O Bluetooth é descrito com mais pormenor na secção 3.5.2.

Embora o grupo de trabalho IEEE 802.11 para redes de área local sem fios já existisse, em 1999, o IEEE decidiu formar um novo grupo (802.15), com o objectivo

[12] Dado que todas as redes pessoais existentes operam sem fios, na prática, os termos PAN e WPAN são sinónimos.

específico de desenvolver normas para redes de área pessoal sem fios. Esta opção ficou a dever-se ao facto das aplicações das redes pessoais procurarem soluções muito mais restritivas quanto ao custo, consumo e dimensão dos produtos do que as redes locais. O grupo de trabalho IEEE 802.15 para redes de área local sem fios (WPAN) é constituído pelos quatro grupos de tarefas (TG, *Task Group*) seguintes.

- TG1: WPAN/Bluetooth. Este grupo foi formado com o objectivo de desenvolver uma especificação para redes de área pessoal sem fios baseada no Bluetooth [IEE02b].
- TG2: Coexistência. Este grupo desenvolve recomendações no sentido de facilitar a coexistência de redes pessoais sem fios com redes locais sem fios uma mesma área, de modo a minimizar a interferência mútua.
- TG3: WPAN de alto débito. Este grupo visa desenvolver uma norma de redes pessoais sem fios de alto débito (20 Mbit/s ou superior) que atenda às necessidades de aplicações de imagem digital e multimédia em dispositivos portáteis, enquanto mantém os requisitos de baixo custo e baixo consumo.
- TG4: WPAN de baixo débito. O objectivo deste grupo é trabalhar numa solução de baixo débito (20 a 250 kbit/s) que permita autonomias de meses a anos com baterias e apresente complexidade muito baixa. Possíveis aplicações incluem sensores, brinquedos interactivos, comandos remotos e automação residencial.

O HomeRF [NEG00] é outra iniciativa de membros da indústria, que formaram um grupo de trabalho com o objectivo de desenvolver especificações de uma rede sem fios de baixo custo, tendo em vista principalmente a utilização em ambientes residenciais. O HomeRF foi concebido de modo a suportar a transmissão de tráfego de voz e de dados e possibilitar a interligação dos dispositivos com rede telefónica pública (PSTN) e/ou com a Internet, por intermédio de um ponto de acesso. Como o Bluetooth, o HomeRF utiliza a técnica de acesso múltiplo por salto em frequência (FHSS) na banda ISM de 2.4 GHz. A versão 1.0 do HomeRF, publicada em 1999, suporta conexões de voz com débitos de 32 kbit/s e débitos de transmissão de dados de até 1.6 Mbit/s, permitindo um alcance de 50 m. A versão 2.0, publicada em 2001, estende o débito máximo para 10 Mbit/s.

O protocolo de controlo de acesso ao meio do HomeRF utiliza uma supertrama de duração igual a 20 ms, composta por um período de contenção e dois períodos livres de contenção: um no início (CFP1) e outro no fim (CFP2) da supertrama. Os períodos livres de contenção são formados por pares de *slots*, um para a ligação descendente e o outro para a ligação ascendente. Estes *slots* utilizam um formato baseado no sistema DECT e destinam-se ao transporte de voz com codificação ADPCM. O período CFP2 é usado para transmissão inicial dos pacotes de voz, enquanto o período CFP1 da supertrama seguinte é utilizado para a retransmissão opcional dos pacotes que não foram recebidos correctamente na supertrama anterior. O salto em frequência no HomeRF coincide com o início de uma nova supertrama, pelo que a transmissão inicial e a eventual retransmissão são feitas em frequências diferentes, o que proporciona diversidade temporal e de frequência. Entretanto, cada pacote de voz só pode ser retransmitido, no máximo, uma única vez. A alocação de *slots* para retransmissão no período CFP1 é anunciada pelo ponto de acesso na trama *Beacon*, transmitida no início da supertrama.

O tráfego de dados é transmitido durante o período de contenção na supertrama, situado entre os períodos livres de contenção. Durante esse período, utiliza-se um protocolo de acesso aleatório do tipo CSMA/CA que consiste numa versão simplificada do protocolo utilizado pelo IEEE 802.11.

3.5.2 Bluetooth

O Bluetooth [BLU01] [JOH99] é uma tecnologia de rádio de curto alcance vocacionada para a transmissão de tráfego de voz e dados, cujo objectivo principal é a substituição dos cabos na conexão entre dispositivos electrónicos de uso pessoal. O baixo custo e reduzido consumo das interfaces são propícios à generalização da sua utilização em dispositivos portáteis. A comunicação entre os dispositivos compatíveis com o Bluetooth não requer a existência de linha de vista (LOS, *Line Of Sight*) entre o emissor e o receptor, ao contrário do que acontece com o IrDA, o que constitui duma vantagem da tecnologia Bluetooth, sobre a sua concorrente directa.

As redes Bluetooth operam sem necessidade de licença na banda ISM dos 2.4 GHz, utilizando a técnica de espalhamento espectral por saltos em frequência (FHSS).

A banda de frequências disponível é dividida em 79 canais[13], com separação de 1 MHz entre as frequências das portadoras. Durante a operação da rede, o canal utilizado é alterado após cada transmissão, em função da sequência pseudo-aleatória adoptada. O Bluetooth utiliza modulação GFSK (*Gaussian Frequency Shift Keying*) e proporciona um débito bruto de 1 Mbit/s. O alcance nominal do Bluetooth situa-se entre 10 cm e 10 m, mas pode ser estendido até 100 m com a utilização de um amplificador de potência externo.

Na interligação de dois dispositivos, aquele que inicia a conexão é denominado mestre, enquanto o outro é conhecido como escravo. Um mestre pode manter até sete conexões com escravos ao mesmo tempo, formando uma pequena rede conhecida pela denominação de *piconet*. O mestre é responsável pelo controlo de acesso ao meio dentro da *piconet*, cuja operação é baseada num protocolo de *polling*. As transmissões são feitas, alternadamente, pelo mestre e pelo escravo destinatário da transmissão realizada pelo mestre. Um exemplo de *piconet* é apresentado na Figura 3.5.

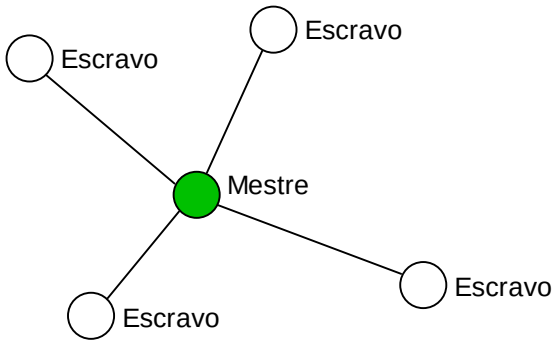


Figura 3.5: Exemplo de uma *piconet* formada por cinco dispositivos.

Cada dispositivo Bluetooth possui um endereço de dispositivo único, formado por 48 bits. A sequência de saltos utilizada numa *piconet* é única, sendo determinada pelo endereço do mestre, enquanto a fase da sequência é determinada pelo relógio do mestre. A relação mestre/escravo é válida apenas para uma conexão específica, dado que qualquer dispositivo Bluetooth pode actuar seja como mestre ou como escravo. Transmissões directas de escravo para escravo não são permitidas dentro de uma

[13] Na Espanha, na França e no Japão estão disponíveis apenas 23 canais.

piconet, pelo que se dois escravos desejam comunicar devem fazê-lo por intermédio do mestre da *piconet* ou, em alternativa, devem formar uma outra *piconet* em que um deles é mestre.

Múltiplas *piconets* numa mesma área formam uma *scatternet*, como é exemplificado na Figura 3.6. As diferentes *piconets* utilizam sequências pseudo-aleatórias de saltos distintas para minimizar a sobreposição das suas transmissões, resultante da utilização de um mesmo canal em simultâneo. Um dispositivo pode fazer parte de diferentes *piconets*; porém, como as unidades de rádio só podem sintonizar uma das portadoras em cada instante, ele só pode comunicar com uma *piconet* de cada vez. O dispositivo pode ser mestre numa *piconet* e escravo em outra, mas não pode ser mestre em duas *piconets* ao mesmo tempo, porque isso faria com que as sequências de saltos de fossem idênticas.

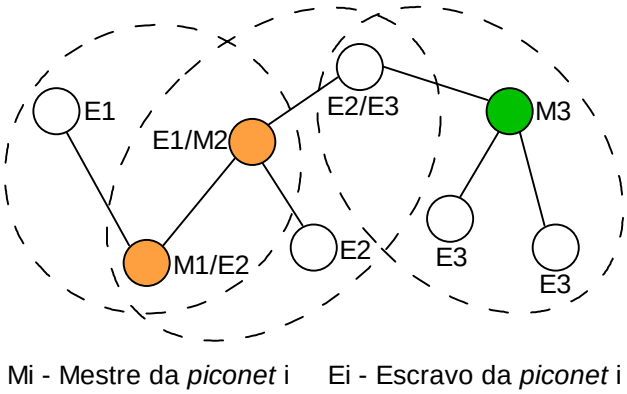


Figura 3.6: Exemplo de uma *scatternet* formada por três *piconets*.

O Bluetooth divide o tempo em *slots* de duração igual a 625 μs, sendo que os pacotes transmitidos podem ocupar um, três ou cinco *slots*. A sequência pseudo-aleatória utilizada para os saltos em frequência avança em cada transição de *slot*, porém, os pacotes que ocupam múltiplos *slots* são transmitidos sem mudança da frequência da portadora. Assim, na transmissão do pacote seguinte, a frequência da portadora salta para o valor da sequência pseudo-aleatória correspondente ao *slot* em que a transmissão é iniciada.

O intervalo mínimo, definido para o Bluetooth, entre o fim da transmissão num sentido e o início da transmissão no sentido oposto (*turnaround time*) é de 200 μs. Este valor é relativamente elevado, porém, permite a utilização de componentes de baixo custo.

Todos os pacotes transmitidos no canal possuem o mesmo formato[14], constituído por três campos, como ilustra a Figura 3.7: o código de acesso (72 bits); o cabeçalho do pacote (54 bits); e o *payload*, cujo comprimento pode variar entre 0 e 2745 bits.

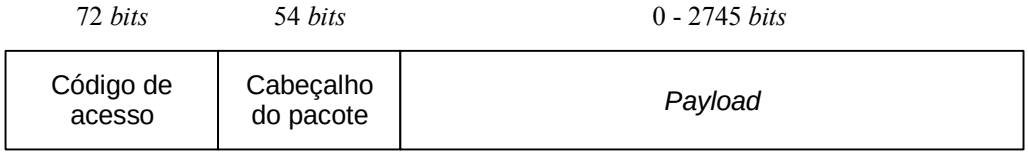


Figura 3.7: Formato do pacote do Bluetooth.

O código de acesso é definido pelo mestre, e identifica os pacotes transmitidos no âmbito de uma *piconet.* Os pacotes transmitidos apenas são aceites por um receptor se a sua identidade é reconhecida nessa *piconet*. O código de acesso também é utilizado para sincronização, compensação do *offset* de corrente contínua e paginação.

O cabeçalho do pacote contém informação utilizada para o controlo da ligação. Os 18 bits de informação do cabeçalho são protegidos por um código de correcção de erros (FEC) de taxa 1/3, o que faz com que o cabeçalho ocupe 54 bits no total. Os campos existentes no cabeçalho, representados na Figura 3.8, possuem as seguintes funções.

- AM_ADDR: Contém um endereço que serve para identificar um escravo activo da *piconet*. Tanto os pacotes enviados do mestre para o escravo quanto os pacotes enviados no sentido contrário transportam o endereço do escravo. O endereço 000 é reservado para mensagens de difusão. Como os endereços são formados por três bits, no máximo sete escravos activos são suportados numa *piconet*. Outros

[14] À excepção do pacote de identidade (ID), usado para sinalização, que contém apenas o campo de código de acesso.

escravos podem fazer parte da *piconet* num estado inactivo de baixo consumo de energia.

- TYPE: Define o tipo de pacote utilizado. A sua interpretação depende do tipo de ligação (síncrona ou assíncrona). Estas ligações são descritas mais abaixo.
- FLOW: É utilizado para controlo de fluxo nas ligações assíncronas.
- ARQN: Serve para reconhecimento do pacote enviado no *slot* anterior. É utilizado pelo esquema de detecção de erros e retransmissão (ARQ).
- SEQN: Contém o número de sequência utilizado pelo esquema de ARQ.
- HEC (*Header Error Check*): Serve para detecção de erros no cabeçalho. Em caso de erro, o pacote é descartado.

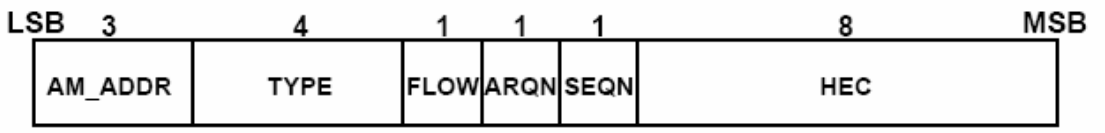


Figura 3.8: Formato do cabeçalho do pacote na rede Bluetooth.

Para a comunicação entre o mestre e os escravos, dois tipos de ligação foram definidos:

- ligação SCO (*Synchronous Connection-Oriented*);
- e ligação ACL (*Asynchronous Connection-Less*).

A ligação SCO é uma ligação ponto a ponto simétrica entre o mestre e um escravo, e utiliza pacotes que ocupam um único *slot*. A ligação é estabelecida pela reserva de pares de *slots* consecutivos (um para cada sentido de comunicação) em intervalos regulares, pelo que se pode dizer que a ligação opera em modo circuito.

A ligação SCO suporta pacotes dos tipos HV1, HV2 e HV3, que transportam tráfego de voz a um débito de 64 kbit/s. Os dois primeiros tipos utilizam taxas de codificação FEC de 1/3 e 2/3, respectivamente, no seu *payload*, enquanto o terceiro transmite a informação de voz sem codificação FEC. A ligação SCO suporta ainda pacotes do tipo DV, que transportam dados e voz em simultâneo. O codificador de voz do Bluetooth gera 10 bytes a cada 1.25 ms. Dado que o comprimento do *payload*

dos pacotes que ocupam um *slot* é de 30 bytes, as conexões HV3 ocupam um *slot* a cada 3.75 ms (6 *slots*), em cada sentido de comunicação. Isso significa que, no máximo, três conexões de voz podem ser suportadas dentro de uma *piconet*. As conexões HV2 reduzem o comprimento útil do *payload* para 20 bytes, pelo que estas ocupam um par de *slots* em cada 4 *slots*. Já no caso das conexões HV1, o comprimento útil do *payload* é de 10 bytes, pelo que estas ocupam um par de *slots* em cada 2 *slots*, fazendo com que toda a largura de banda disponível na *piconet* seja ocupada por uma única conexão de voz.

A ligação ACL é uma ligação ponto-multiponto entre o mestre e os escravos da *piconet* que faz uso dos *slots* que não foram reservados pelas ligações SCO. Os diferentes tipos de pacotes definidos no Bluetooth podem ocupar um, três ou cinco *slots* e podem utilizar uma taxa de codificação FEC de 2/3 no seu *payload* ou serem transmitidos sem codificação FEC. Como o mestre é responsável pelo controlo de acesso ao meio, deve utilizar um algoritmo de escalonamento que tenha em consideração as características do tráfego dos escravos. A Figura 3.9 apresenta um exemplo da coexistência de ligações SCO (síncronas) e ligações ACL (assíncronas) numa mesma *piconet*.

O *payload* dos pacotes transmitidos numa ligação ACL inclui um cabeçalho (1 byte no pacotes que ocupam um *slot* e 2 bytes nos outros casos) e um campo de CRC (16 bits), ausente apenas nos pacotes do tipo AUX1. O débito assimétrico máximo que pode ser obtido é de 723.2 kbit/s, suportando uma ligação de retorno de 57.6 kbit/s; enquanto o débito simétrico máximo suportado é igual a 432.6 kbit/s.

O controlo de erros no Bluetooth baseia-se tanto na correcção de erros (FEC) como na detecção de erros com retransmissão de pacotes (ARQ). No caso do FEC, códigos de taxas de 1/3 ou 2/3 são utilizados. O código FEC de taxa 1/3 é utilizado no cabeçalho do pacote e pode também ser aplicado ao *payload* dos pacotes em ligações SCO. Já o código FEC de taxa 2/3 pode ser aplicado ao *payload* dos pacotes tanto em ligações SCO como em ligações ACL. Nos dois casos, também existe a opção de não proteger o *payload* com um código FEC. Quanto maior for a redundância introduzida pelo código FEC, menor é o espaço que sobra no pacote para a transmissão de informação útil.

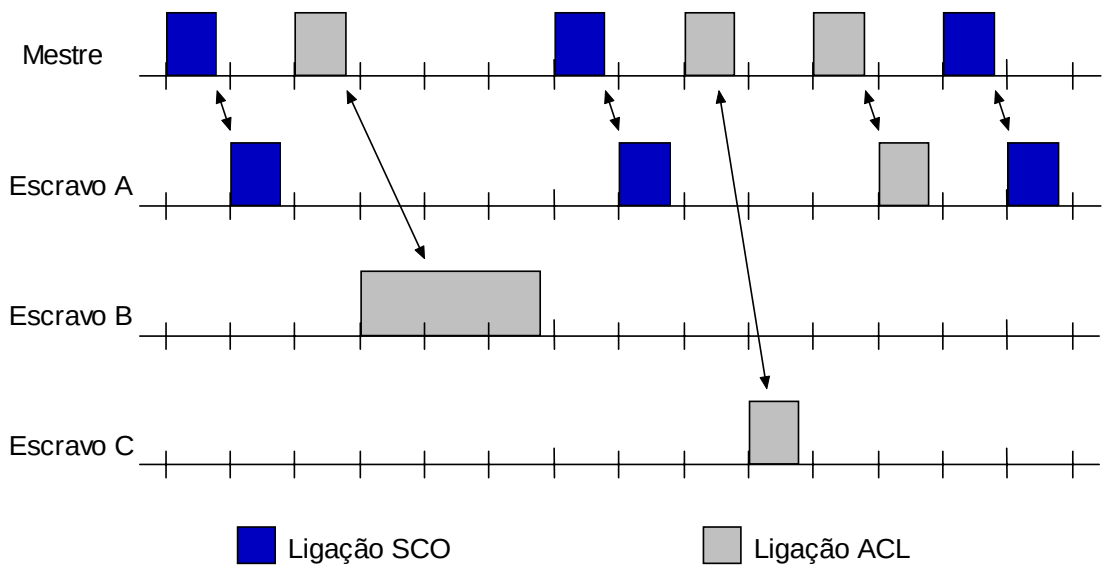


Figura 3.9: Exemplo da coexistência de ligações SCO e ACL numa mesma *piconet*.

O esquema de detecção de erros e retransmissão de pacotes (ARQ) pode ser aplicado somente às ligações ACL. Neste caso, o esquema utiliza o campo de CRC presente no *payload* dos pacotes para detecção de erros. O Bluetooth implementa um esquema denominado *fast* ARQ, no qual o reconhecimento de um pacote, pelo receptor, é feito no *slot* imediatamente seguinte ao término da transmissão do pacote, pelo emissor. Para esse efeito, utiliza-se o campo ARQN presente no cabeçalho dos pacotes, de modo que os dispositivos podem enviar os seus dados ao mesmo tempo que fazem o reconhecimento da transmissão anterior. O número de sequência (SEQN) (1 bit) permite a detectar a recepção de uma segunda cópia sem erros de um mesmo pacote (duplicado). Esta situação pode ocorrer caso o pacote contendo o reconhecimento positivo da primeira cópia seja corrompido por erros.

3.6 A rede IEEE 802.11

3.6.1 Introdução

Os primeiros produtos para redes locais sem fios, utilizando tecnologia proprietária, foram introduzidos no início da década de 90, aproveitando a

disponibilidade das bandas ISM para operação sem necessidade de licença. Estes produtos operavam na banda ISM de 900 MHz, disponível na América do Norte. Algum tempo depois, surgiram produtos a operar na banda ISM de 2.4 GHz e, mais para o final da década, começaram a aparecer produtos que utilizam a banda de frequências de 5 GHz.

O principal problema das soluções proprietárias de redes sem fios foi a moderada aceitação no mercado, por falta de um padrão que garantisse a compatibilidade entre os produtos disponibilizados pelos diversos fabricantes. Para resolver este problema, após vários anos de discussão, o IEEE (*Institute of Electrical and Electronics Engineers*) aprovou a primeira versão das normas IEEE 802.11, publicada em 1997.

3.6.1.1 Arquitectura

De acordo com a denominação utilizada pelas normas do IEEE 802.11, um grupo de duas ou mais estações sob o controlo directo de uma mesma função de coordenação[15] forma um conjunto básico de serviço (BSS, *Basic Service Set*). A área coberta por um BSS é designada por área básica de serviço (BSA, *Basic Service Area*), e pode ser vista como o análogo a uma célula numa rede celular móvel.

As redes IEEE 802.11 podem operar em modo *ad hoc* ou em modo baseado em infraestrutura. Um BSS isolado, em que as estações comunicam apenas entre si formando uma rede *ad hoc* é denominado IBSS (*Independent BSS*).

A Figura 3.10 representa os componentes da arquitectura da rede IEEE 802.11 na configuração baseada em infraestrutura. Este modo de operação requer a presença de uma estação especial no BSS, denominada ponto de acesso (AP, *Access Point*), que serve de interface entre o BSS e o sistema de distribuição (DS, *Distribution System*) e possibilita a comunicação entre as estações (STA) do BSS e entidades externas. O sistema de distribuição (DS) permite interligar múltiplos BSSs formando um conjunto estendido de serviço (ESS, *Extended Service Set*), que aparenta ser um único BSS

[15] Função de coordenação é o nome utilizado pelo IEEE 802.11 para se referir ao mecanismo de controlo de acesso ao meio.

alargado para a subcamada de ligação lógica (LLC) das estações que compõem o ESS. Normalmente, utiliza-se uma rede local convencional como sistema de distribuição, embora outras redes possam ser utilizadas, visto que as normas não entram em detalhes quanto à implementação do sistema de distribuição.

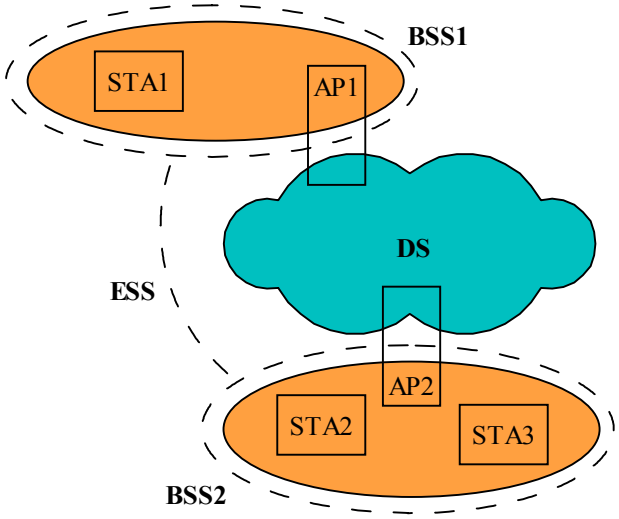


Figura 3.10: Componentes da arquitectura da rede IEEE 802.11.

3.6.1.2 Confidencialidade

Numa rede cablada, apenas as estações fisicamente conectadas ao meio de transmissão estão aptas a captar as transmissões, o que não acontece nas redes sem fios. Qualquer estação compatível pode monitorar o tráfego de uma rede IEEE 802.11 que utilize a mesma camada física sem que seja detectada, bastando para isso estar dentro do raio de alcance da rede. Para aumentar o nível de privacidade do IEEE 802.11, o protocolo de segurança WEP (*Wired Equivalent Privacy*) foi especificado pelo IEEE 802.11. O WEP define um algoritmo criptográfico, de utilização opcional, destinado a proporcionar um nível de confidencialidade à transmissão de dados similar (subjectivamente) à confidencialidade existente numa rede local cablada que não utilize técnicas criptográficas para aumentar a privacidade das comunicações.

3.6.1.3 Associação

Antes que uma estação possa encaminhar mensagens de dados através de um ponto de acesso, ela deve associar-se ao mesmo. O processo de associação permite ao sistema de distribuição determinar a que ponto de acesso uma estação está associada, de modo a poder encaminhar mensagens para a mesma. A cada instante, uma estação não pode estar associada a mais que um ponto de acesso. O processo de associação é sempre iniciado pela estação, pelo envio de uma mensagem de requisição de associação.

3.6.1.4 Autenticação

A autorização de acesso à rede por parte das estações no IEEE 802.11 é negociada através do serviço de autenticação. Se um nível aceitável de autenticação não for estabelecido entre a estação e o ponto de acesso, a associação não é realizada. O IEEE 802.11 suporta diversos mecanismos de autenticação, entre as quais a autenticação por chave compartilhada (*shared key*), em que a identidade da estação é demonstrada pelo conhecimento da chave de encriptação secreta do protocolo de segurança WEP.

3.6.1.5 Gestão de consumo de energia

O IEEE 802.11 proporciona suporte para gestão de consumo de energia das estações. O ponto de acesso armazena os dados que destinados às estações que estão a operar em modo de conservação de energia e difunde informação nas tramas *Beacon* identificando estas estações. Para esse efeito, as tramas *Beacon* contém um mapa de indicação de tráfego (TIM, *Traffic Indication Map*). As estações que operam no modo de conservação de energia acordam periodicamente para escutar a trama *Beacon*. Quando uma estação recebe indicação de que o ponto de acesso tem tramas armazenadas para si, ela interroga o ponto de acesso, requisitando o envio das tramas armazenadas.

3.6.2 Camada física

A norma IEEE 802.11 original [IEE99] especifica três camadas físicas distintas, baseadas em espalhamento espectral por sequência directa (DSSS), espalhamento espectral por saltos em frequência (FHSS) e comunicação óptica (DFIR, *diffuse infrared*). As duas primeiras operam na banda ISM de 2,4 GHz, enquanto a última utiliza a banda de infravermelhos. Todas estas camadas físicas suportam débitos de transmissão de 1 Mbit/s e 2 Mbit/s.

Para permitir que a camada de controlo de acesso ao meio (MAC) do IEEE 802.11 possa operar de forma independente da camada física escolhida, esta última camada é dividida em duas subcamadas: a subcamada PLCP (*Physical Layer Convergence Procedure*); e a subcamada PMD (*Physical Medium Dependent*). A subcamada PLCP faz o mapeamento das tramas MAC num formato adequado à transmissão da informação entre as estações da rede utilizando as funções da subcamada PMD associada, enquanto esta última define as características e métodos de transmissão e recepção de dados sobre o meio sem fios.

A camada física do IEEE 802.11 recebe a trama MAC e anexa o preâmbulo e o cabeçalho da PLCP no início. Ao contrário do que acontece com a trama MAC, o preâmbulo e o cabeçalho da camada física são sempre transmitidos ao débito mínimo da rede[16], independentemente do débito de transmissão utilizado. Isso permite a coexistência de estações que suportem débitos máximos diferentes e simplifica a tarefa de equalização. Porém, a eficiência de transmissão pode ser reduzida significativamente, nomeadamente quando o débito utilizado para transmissão de dados é muito superior ao débito mínimo.

A norma IEEE 802.11b [IEE99b], introduzida em 1999, estende a especificação da camada física baseada em espalhamento espectral por sequência directa (DSSS), aumentando o débito máximo de transmissão para 11 Mbit/s. Esta versão suporta ainda o débito de 5.5 Mbit/s e os débitos de 2 Mbit/s e 1 Mbit/s da versão original, enquanto mantém a retro compatibilidade com a mesma. O aumento do débito de

[16] 6 Mbit/s na rede IEEE 802.11a, 1 Mbit/s nos demais casos.

transmissão é alcançado pela utilização de modulação CCK (*Complementary Code Keying*).

As redes IEEE 802.11b dispõem de 14 canais[17] na banda ISM dos 2.4 MHz para operação, cada qual ocupando 22 MHz. Os canais adjacentes sobrepõem-se parcialmente, pelo que no máximo apenas 3 canais podem ser usados simultaneamente.

Para reduzir o *overhead* associado à camada física, a norma IEEE 802.11b define um formato curto para o preâmbulo e o cabeçalho da PLCP, em alternativa ao formato normal. O formato curto é opcional e só pode ser utilizado para comunicação entre estações que o suportem.

A norma IEEE 802.11a, que define uma outra camada física para além das mencionadas anteriormente, é descrita com mais detalhes na próxima secção.

3.6.2.1 Camada física de alto débito na banda de 5 GHz

A norma IEEE 802.11a [IEE99a] especifica a operação da rede IEEE 802.11 na banda de frequências de 5 GHz, sendo baseada na técnica de modulação OFDM (*Orthogonal Frequency Division Multiplexing*). São utilizadas 52 subportadoras: 48 para o transporte de dados e 4 para sinais piloto. As 48 subportadoras de dados podem ser moduladas utilizando BPSK, QPSK, 16QAM ou 64QAM.

Além das opções de modulação apresentadas, existem diferentes opções de taxas de codificação para correcção de erros (FEC), baseadas na utilização de código convolucional. As taxas de codificação possíveis são 1/2, 2/3 e 3/4. A combinação do esquema de modulação com a taxa de codificação determina o débito de transmissão em uso. A Tabela 3.1 apresenta os oito modos de transmissão disponíveis na rede IEEE 802.11a, juntamente com os débitos de transmissão ao nível da camada física e os parâmetros de modulação e codificação.

[17] Em alguns países, a disponibilidade de canais é menor.

Independentemente do débito de transmissão utilizado, o débito dos símbolos OFDM é constante, sendo igual a 250000 símbolos/s, o que corresponde a uma duração de 4 μs para cada símbolo OFDM. Entretanto, para evitar a interferência intersimbólica (ISI), um período de guarda de 0.8 μs é utilizado, deixando 3.2 μs para a parte útil do símbolo OFDM.

Tabela 3.1: Modos de transmissão da rede IEEE 802.11a

Modo de transmissão	Débito (Mbit/s)	Modulação	Taxa de codificação
1	6	BPSK	1/2
2	9	BPSK	3/4
3	12	QPSK	1/2
4	18	QPSK	3/4
5	24	16QAM	1/2
6	36	16QAM	3/4
7	48	64QAM	2/3
8	54	64QAM	3/4

A subcamada PLCP (*Physical Layer Convergence Procedure*) do IEEE 802.11a adapta a trama da subcamada MAC (MPDU) a um formato (PPDU) apropriado à transmissão pela camada física baseada em modulação OFDM. O formato do PPDU, apresentado na Figura 3.11, inclui o preâmbulo da PLCP, o cabeçalho da PLCP, o PSDU (que corresponde ao MPDU), os *tail* bits e os *pad* bits.

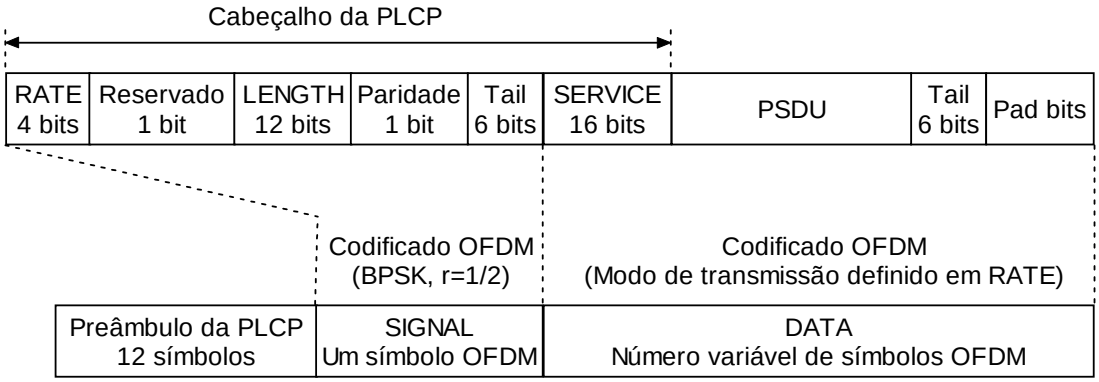


Figura 3.11: Formato do PDU da camada física (PPDU) do IEEE 802.11a.

O preâmbulo da PLCP, usado para sincronização OFDM, tem a duração de 16 μs, sendo formado por duas sequências de treino: a primeira sequência é composta por 10 símbolos curtos; enquanto a segunda sequência contém 2 símbolos longos.

A seguir ao preâmbulo da PLCP encontram-se os campos SIGNAL e DATA. O campo SIGNAL é formado pelo cabeçalho da PLCP, com excepção do campo SERVICE, que faz parte do campo DATA. O campo SIGNAL é transmitido com o modo de transmissão mais robusto disponível (6 Mbit/s), que utiliza modulação BPSK e uma taxa de codificação de 1/2 (Tabela 3.1). Este campo ocupa exactamente um símbolo OFDM, pelo que a sua duração é de 4 μs.

A duração do campo DATA é variável, dependendo do comprimento do PSDU e do débito de transmissão utilizado. Esta informação encontra-se codificada nos campos LENGTH e RATE do cabeçalho da PLCP, respectivamente. Ao contrário do campo SIGNAL, os bits do campo DATA são baralhados antes da transmissão. Parte do campo SERVICE é utilizada para sincronizar o *descrambler* no receptor, enquanto a outra parte é reservada para uso futuro.

A rede IEEE 802.11a foi concebida para operar numa banda de frequências entre 5 e 6 GHz. Nesta região, a frequência central dos canais é definida em múltiplos de 5 MHz a partir de 5 GHz. A relação entre a frequência central do canal (f_{ch}), em MHz, e o número do canal (n_{ch}) é expressa por:

$$f_{ch} = 5000 + 5 \times n_{ch}, \tag{3.1}$$

sendo n_{ch} = 0, 1 ... 200.

O espaçamento entre os canais das redes IEEE 802.11a especificado nas normas é de 20 MHz. Além disso, nem toda a banda de frequências entre 5 e 6 GHz está disponível. Sendo assim, definiu-se um conjunto de canais válidos para operação nas bandas U-NII (*Unlicensed National Information Infrastructure*), nos Estados Unidos, que é apresentado na Tabela 3.2.

Tabela 3.2: Canais válidos para operação de redes IEEE 802.11a nos Estados Unidos.

Banda (GHz)	Número do canal	Frequência central do canal (MHz)
Banda U-NII inferior (5.15 - 5.25)	36	5180
	40	5200
	44	5220
	48	5240
Banda U-NII média (5.25 - 5.35)	52	5260
	56	5280
	60	5300
	64	5320
Banda U-NII superior (5.725 - 5.825)	149	5745
	153	5765
	157	5785
	161	5805

3.6.3 Camada de ligação de dados

O formato geral da trama do nível de controlo de acesso ao meio (MAC) do IEEE 802.11 é representado na Figura 3.12. Os campos endereço 2, endereço 3, controlo de sequência, endereço 4 e corpo da trama estão presentes apenas em alguns tipos de tramas.

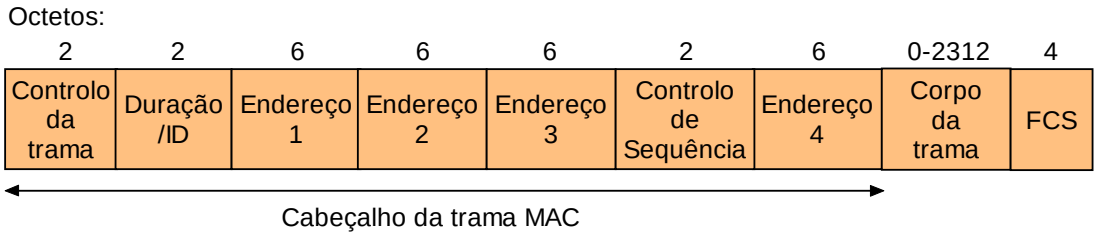


Figura 3.12: Formato geral da trama de nível MAC do IEEE 802.11.

A função de cada campo da trama é descrita sucintamente a seguir.

- O campo de controlo da trama carrega informação de controlo, sendo composto por vários subcampos, entre os quais os subcampos tipo e subtipo, que identificam tanto o formato como a função da trama. O IEEE 802.11 define três tipos de trama: dados, controlo e gestão.
- O campo de duração/ID é utilizado, na maioria dos casos, pelo mecanismo de detecção de portadora virtual, que será referido mais adiante.
- Cada campo de endereço contém um endereço de nível MAC de 48 bits. Dependendo do tipo de trama e do seu eventual percurso pelo sistema de distribuição (DS), os campos de endereço podem referenciar a estação destino (DA, *Destination Address*), a estação origem (SA, *Source Address*), o receptor imediato (RA, *Receiver Address*), o emissor actual (TA, *Transmitter Address*) ou o BSS em que a trama é transmitida (BSSID).
- O campo de controlo de sequência é composto por dois subcampos: o número de sequência e o número de fragmento.
- O campo corpo da trama é um campo de comprimento variável que transporta informação relacionada com o tipo e subtipo de trama. No caso específico da trama de dados, este campo transporta o seu *payload*.
- O campo FCS (*Frame Check Sequence*) carrega um código CRC (*Cyclic Redundancy Code*) de 32 bits usado para a detecção de erros na trama. O FCS é calculado sobre todos os campos do cabeçalho e do corpo da trama.

3.6.3.1 Controlo de acesso ao meio

A camada de controlo de acesso ao meio do IEEE 802.11 implementa dois mecanismos distintos: a função de coordenação distribuída (DCF, *Distributed Coordination Function*); e a função de coordenação pontual (PCF, *Point Coordination Function*). O DCF é o mecanismo básico de controlo de acesso ao meio do IEEE 802.11, sendo um protocolo de acesso aleatório do tipo CSMA/CA (*Carrier Sense Multiple Access/Collision Avoidance*). O PCF, por outro lado, é um protocolo de *polling* que permite às estações o acesso ao meio livre de contenção, o que torna sua utilização mais adequada para o transporte de tráfego de tempo real. Entretanto, a sua implementação não é obrigatória, ao contrário do DCF.

3.6.3.1.1 Função de coordenação distribuída (DCF)

De acordo com as regras do DCF, antes de iniciar a transmissão de uma trama de dados, a estação deve sondar o meio. Se no momento em que a estação deseja iniciar uma nova transmissão o meio não estiver ocupado, e se o meio continuar livre decorrido um período DIFS (*Distributed InterFrame Space*), a estação transmite a sua trama. Caso uma das duas condições anteriores não se verifique, a estação adia a sua transmissão e inicia o processo de *backoff* (Figura 3.13).

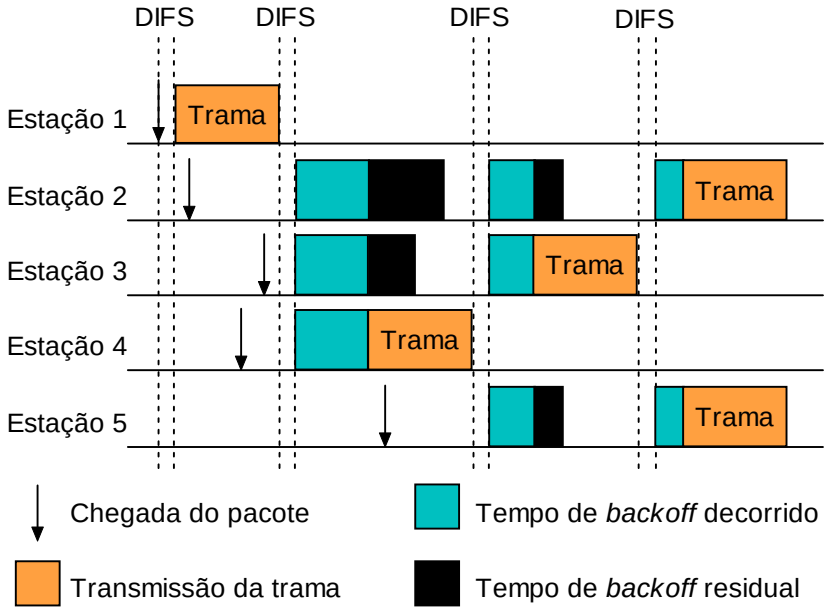


Figura 3.13: Exemplo de operação da função de coordenação distribuída (DCF).

No início do processo de *backoff*, a estação gera um número inteiro aleatório, uniformemente distribuído entre zero e o valor mínimo (CW_{min}) da janela de contenção (CW, *Contention Window*), que é utilizado para inicializar o contador de *backoff*. Quando o meio fica livre e permanece livre durante um período DIFS, a estação começa a decrementar o seu contador de *backoff*, fazendo-o a intervalos fixos (*SlotTime*) enquanto o meio continua livre, até o contador chegar a zero. Se a estação detecta uma transmissão durante esse período, porém, ela interrompe o decremento do contador e aguarda que o meio volte a ficar livre durante um período DIFS, para então recomeçar o decremento do contador de *backoff* a partir do ponto em que parou. Quando o contador chega a zero, a estação inicia a sua transmissão. Caso duas ou mais estações comecem suas transmissões ao mesmo tempo ocorre uma colisão (como é o caso das tramas transmitidas pelas estações 2 e 5, no exemplo da Figura 3.13).

De acordo com as normas do IEEE 802.11, o destinatário de uma trama de dados recebida correctamente responde com a transmissão de uma trama de reconhecimento positivo (ACK), depois de aguardar por um período SIFS (*Short InterFrame Space*), como é mostrado na Figura 3.14, sendo válida a relação:

$$DIFS = SIFS + 2\,SlotTime \qquad (3.2)$$

Como o período SIFS é menor do que o DIFS, a estação receptora pode transmitir a trama ACK sem ter que sondar o meio antes e sem risco de colisão, já que as outras estações estão inibidas de iniciar as suas transmissões durante um período maior. Na rede IEEE 802.11, o reconhecimento é feito cada vez que uma trama de dados é transmitida porque, ao contrário do que acontece nas redes cabladas, a incerteza sobre a recepção correcta de uma trama é muito maior, dado que num meio sem fios não é possível a detecção de colisões, além do que as transmissões estão muito mais expostas a erros no canal.

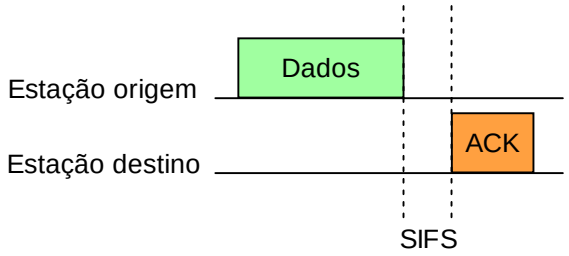


Figura 3.14: Mecanismo de reconhecimento positivo da função DCF.

A estação receptora não transmite uma trama de reconhecimento negativo quando recebe uma trama de dados corrompida por erros no canal. Assim, do ponto de vista do emissor, não há diferença entre este caso e a ocorrência de uma colisão. Se o emissor não receber uma trama de reconhecimento positivo (ACK) em resposta ao envio de uma trama de dados, assume que a trama não foi recebida com sucesso. Esgotado o período de *timeout* para a recepção da trama ACK, o emissor repete o processo de *backoff*, antes de retransmitir a trama de dados. Entretanto, para diminuir a probabilidade de colisão, após cada tentativa falhada de transmissão de uma trama, o valor da janela de contenção (*CW*) é duplicado[18] até atingir um valor máximo (CW_{max}). Após uma transmissão bem sucedida (que tenha recebido reconhecimento positivo), a janela de contenção volta a assumir o valor inicial (CW_{min}). Caso a estação tenha mais dados a enviar, deve executar novamente o processo de *backoff* antes de iniciar a nova transmissão.

[18] Na verdade, a operação realizada é $CW_{i+1} = 2(CW_i + 1) - 1$.

3.6.3.1.2 O mecanismo RTS/CTS

A detecção de portadora em redes sem fios está sujeita ao fenómeno da estação oculta (*hidden station*). Este fenómeno ocorre quando uma estação é capaz de receber o sinal de duas outras estações na sua vizinhança, mas as duas estações não conseguem detectar o sinal uma da outra, seja por estarem muito distantes entre si ou por terem o sinal bloqueado por obstáculos. Para lidar com o problema da estação oculta, o protocolo de controlo de acesso ao meio do IEEE 802.11 inclui um mecanismo opcional, baseado na troca de duas pequenas tramas de controlo antes do envio da trama de dados: a trama RTS (*Request To Send*), enviada pelo potencial transmissor da trama de dados (estação origem); e a trama CTS (*Clear To Send*), enviada pela estação destino em resposta à trama RTS, após um período SIFS. Se a trama CTS não for recebida, a estação origem inicia o processo de *backoff* para a retransmissão da trama RTS. Por outro lado, se a troca das tramas RTS e CTS for bem sucedida, esta estação pode então transmitir a trama de dados, após um período SIFS.

A trama RTS possui um campo de duração que indica o período necessário para transmissão da trama CTS, da trama de dados e da trama ACK, incluindo os períodos SIFS entre as tramas. Esse período é calculado pela estação origem tendo em consideração o comprimento das tramas e o débito de transmissão utilizado. O campo de duração também está presente na trama CTS, sendo preenchido pelo destinatário com base no valor anunciado na trama RTS, descontando-se o tempo necessário para a transmissão da trama CTS. As estações da rede analisam o conteúdo das tramas RTS e CTS, independentemente do destinatário, e utilizam a informação do campo de duração para actualizar os respectivos temporizadores NAV (*Network Allocation Vector*). Este temporizador inibe a transmissão da estação até o tempo programado expirar, mesmo que não haja actividade no meio, pelo que esse mecanismo é denominado detecção de portadora virtual.

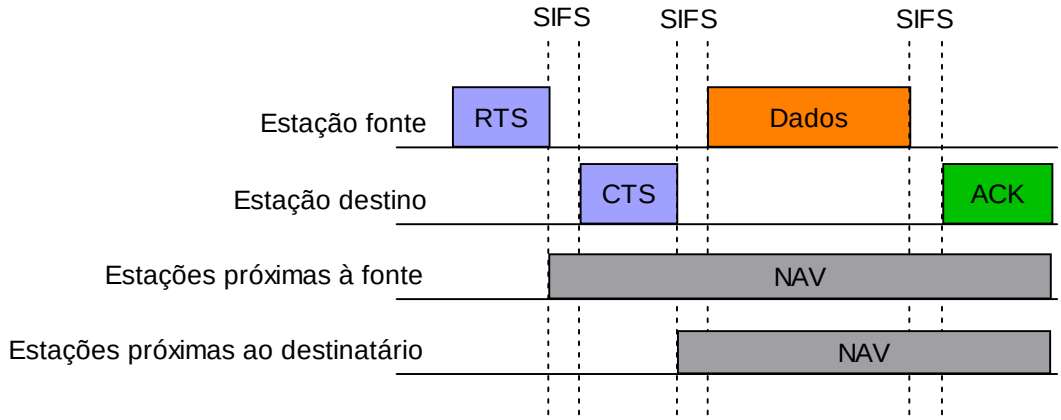


Figura 3.15: Comunicação entre as estações utilizando o mecanismo RTS/CTS.

A transmissão da trama RTS inibe a transmissão das estações na proximidade da estação origem, enquanto que a transmissão da trama CTS inibe a transmissão das estações na proximidade da estação destino. Desta forma, a estação origem tem o caminho livre para a transmissão da sua trama de dados sem o risco de colisão.

Com a utilização do mecanismo de RTS/CTS é possível a ocorrência de colisões entre tramas RTS, tal como podem ocorrer colisões entre tramas de dados sem a sua utilização. Porém, como as tramas RTS são normalmente muito menores do que as tramas de dados, o tempo desperdiçado numa colisão é menor, podendo daí resultar um aumento da eficiência do protocolo de controlo de acesso ao meio, mesmo sem se considerar o problema da estação oculta.

A desvantagem óbvia da utilização do mecanismo de RTS/CTS está no aumento do *overhead*, principalmente na transmissão de tramas de dados de comprimento pequeno. Sendo assim, o mecanismo RTS/CTS é habilitado apenas quando o comprimento da trama de dados a enviar é maior do que valor especificado por um parâmetro denominado *RTS Threshold*.

3.6.3.1.3 Fragmentação

Quando o comprimento do MSDU (*MAC Service Data Unit*) é grande, a sua segmentação, seguida da inserção dos segmentos em múltiplas tramas de dados de dimensão reduzida (fragmentos), contribui para o aumento da fiabilidade da transmissão, pois quanto menor é o comprimento de uma trama, menor é a

probabilidade dela ser corrompida por erros no canal. A fragmentação pode resultar no aumento da eficiência do protocolo quando a taxa de erros binários no canal for significativa, apesar do aumento do *overhead* (cabeçalhos, preâmbulos e períodos de guarda entre tramas) que esta técnica introduz.

A fragmentação é efectuada quando o comprimento da trama excede o valor definido pelo parâmetro *fragmentation threshold*. O comprimento dos fragmentos associados a um dado MSDU é igual ao valor deste parâmetro, com excepção do último fragmento, cujo comprimento depende do comprimento residual do último segmento proveniente do MSDU.

Os fragmentos associados a um mesmo MSDU são transmitidos em sequência, intercalados com as respectivas tramas ACK enviadas pelo receptor. A separação entre tramas consecutivas é de um período SIFS, pelo que o emissor mantém o controlo sobre o meio durante a transmissão dos fragmentos. O meio só é libertado após a transmissão de todos os fragmentos, ou então quando a sequência é interrompida, o que ocorre quando o emissor não recebe o reconhecimento positivo de um dado fragmento. Neste caso, o emissor inicia o processo de *backoff*, ao fim do qual transmite os restantes fragmentos associados ao MSDU, começando pelo primeiro fragmento que não obteve reconhecimento positivo.

Quando o mecanismo RTS/CTS é utilizado, as tramas RTS e CTS são transmitidas apenas uma vez, antes do primeiro fragmento. Os campos de duração destas tramas são usados para indicar a ocupação do meio somente até ao fim da primeira trama ACK, e não até ao fim da última trama ACK, pois isso inibiria a transmissão das outras estações até ao final previsto para a transmissão de todos os fragmentos, mesmo se a sequência fosse interrompida e, consequentemente, o meio ficasse livre antes. Os campos de duração do primeiro fragmento e da primeira trama ACK anunciam que o meio estará ocupado até ao final da segunda trama ACK, e assim por diante, até que todos os fragmentos tenham sido transmitidos.

3.6.3.1.4 Função de coordenação pontual (PCF)

Como o objectivo de proporcionar suporte a serviços com requisitos de tempo real, o IEEE 802.11 inclui a função de coordenação pontual (PCF). Este mecanismo

de controlo de acesso ao meio utiliza um protocolo de *polling*, controlado pelo ponto de acesso (AP, *Access Point*), para determinar qual estação tem direito de transmitir em dado momento. A função PCF é utilizada para o controlo de acesso ao meio durante o período livre de contenção (CFP, *Contention Free Period*), que se alterna com o período de contenção (CP, *Contention Period*), no qual a função DCF é utilizada. A junção destes dois períodos resulta na chamada supertrama, como é representado na Figura 3.16. Além de transportar o tráfego assíncrono, o período de contenção permite o registo de novas estações na lista de *polling*.

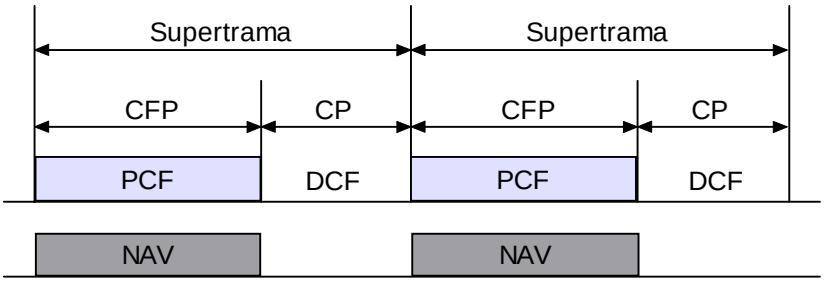


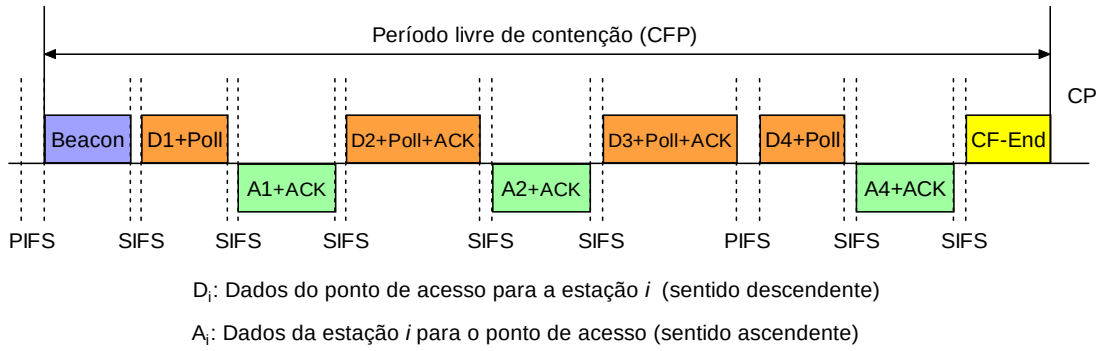
Figura 3.16: Coexistência das funções PCF e DCF na supertrama.

Para dar início a um novo período livre de contenção (CFP), o ponto de acesso aguarda que o canal permaneça livre durante um período PIFS (*PCF InterFrame Space*), após o fim previsto do período de contenção (CP) da supertrama anterior, sendo válida a relação:

$$PIFS = SIFS + SlotTime, \tag{3.3}$$

o que implica que o período PIFS é maior que o SIFS e menor que o DIFS.

O período livre de contenção inicia-se com a transmissão da trama *Beacon* pelo ponto de acesso, como mostra a Figura 3.17. A seguir, é feita a interrogação das estações pertencentes à lista de *polling*. A transmissão da trama CF-End, pelo ponto de acesso, encerra o período livre de contenção. Ao longo deste período, as tramas são espaçadas de um período SIFS.



D_i: Dados do ponto de acesso para a estação *i* (sentido descendente)

A_i: Dados da estação *i* para o ponto de acesso (sentido ascendente)

Figura 3.17: Exemplo de transmissões realizadas durante o período livre de contenção.

A trama *Beacon* é uma trama de gestão utilizada para difundir diversas informações relacionadas com a operação da célula (BSS). Entre as informações associadas à função PCF, encontra-se o intervalo de repetição do período CFP, ou seja, a duração nominal da supertrama[19]. A trama *Beacon* inclui também o parâmetro *CFPMaxDuration*, que indica a duração máxima do período CFP corrente, sendo utilizado pelas estações para actualizar os seus temporizadores NAV. O valor máximo que este parâmetro pode assumir corresponde à duração nominal da supertrama menos o tempo necessário para a transmissão de um MPDU de comprimento máximo durante o período de contenção, incluindo as tramas de controlo associadas.

Durante o período livre de contenção, os seguintes subtipos de tramas de dados podem ser transmitidos: Data, Data+CF-Ack, Data+CF-Poll, Data+CF-Ack+CF-Poll, Null Function (sem dados), CF-Ack (sem dados), CF-Poll (sem dados) e CF-Ack+CF-Poll (sem dados). As tramas contendo o CF-Poll só podem ser transmitidas pelo ponto de acesso, enquanto as outras tramas podem ser transmitidas por qualquer estação que seja capaz de comunicar usando a função PCF. A presença do CF-Poll autoriza a transmissão de dados pela estação destino. A trama Null Function é utilizada pela estação quando é interrogada e não tem dados nem informação de reconhecimento positivo (CF-Ack) para transmitir.

[19] A duração da supertrama traduz-se por um múltiplo inteiro do intervalo entre tramas *Beacon*, pelo que, além da trama *Beacon* transmitida no início do período CFP, outras tramas *Beacon* podem ser transmitidas ao longo da supertrama.

Após receber dados de uma estação, o ponto de acesso pode enviar dados (Data) e interrogar (CF-Poll) uma outra estação ao mesmo tempo que reconhece (CF-Ack) os dados recebidos da primeira. A capacidade do ponto de acesso de combinar a interrogação, o reconhecimento e os dados numa mesma trama foi concebida para aumentar a eficiência do protocolo.

Quando o ponto de acesso interroga uma estação e esta não inicia a sua transmissão no momento esperado (ou seja, decorrido um período SIFS), o ponto de acesso retoma o controlo do canal após um período PIFS, dando sequência à interrogação das estações presentes na lista de *polling*. No exemplo da Figura 3.17, esta situação ocorre quando a estação 3 é interrogada.

Ao ser interrogada pelo ponto de acesso, uma estação pode enviar uma trama de dados para outra estação. Esta responde com uma trama ACK com formato idêntico ao utilizado com a função DCF. Depois, o ponto de acesso reassume o controlo do canal, após aguardar durante um período PIFS. De modo análogo, o ponto de acesso pode transmitir uma trama de dados para uma estação que não reconhece a função PCF durante o período livre de contenção, e esta responde com uma trama ACK da função DCF.

Se no instante previsto para o início do período livre de contenção (CFP) o meio estiver ocupado com uma transmissão inacabada associada ao período de contenção (tráfego DCF), o ponto de acesso tem que esperar pelo fim da transmissão para adquirir o controlo do canal. Entretanto, o período livre de contenção não se pode alongar para além do limite especificado pelo parâmetro *CFPMaxDuration*, pelo que a sua duração máxima permitida, nessa supertrama, sofre uma redução, como é exemplificado na Figura 3.18. Como uma transmissão pode iniciar-se imediatamente antes do término previsto para o período de contenção, o encurtamento do período livre de contenção pode atingir o tempo necessário para a transmissão de um MPDU de comprimento máximo, juntamente com as tramas de controlo associadas. O encurtamento do período livre de contenção provocado pelo alargamento (*stretching*) do período de contenção diminui a largura de banda disponível para o transporte do tráfego PCF, além aumentar o *jitter* das conexões.

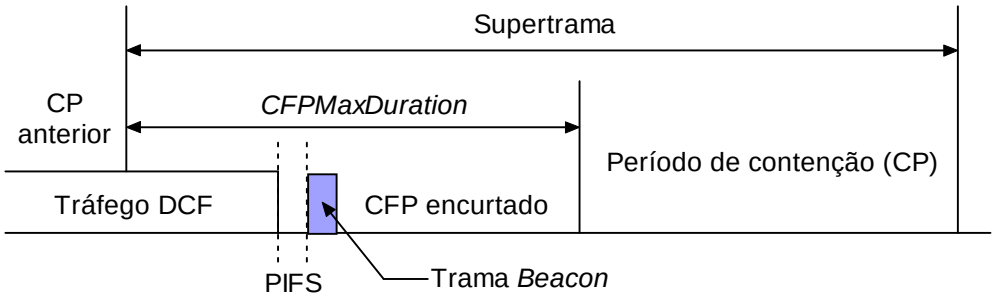


Figura 3.18: Exemplo de encurtamento do período livre de contenção.

3.6.4 Outras extensões do IEEE 802.11

3.6.4.1 802.11e

Na altura de escrita desta tese, o grupo de tarefas E do 802.11 (TGe) encontra-se a desenvolver a norma IEEE 802.11e [IEE02c] [XU03], cujo objectivo é aperfeiçoar o protocolo de controlo de acesso ao meio do IEEE 802.11, de forma a aumentar a eficiência e o suporte de qualidade de serviço.

A norma IEEE 802.11e define um novo modo de operação para o controlo de acesso ao meio, designado função de coordenação híbrida (HCF, *Hybrid Coordination Function*), que combina acesso baseado em contenção e acesso controlado. O mecanismo de acesso baseado em contenção da função HCF é denominado EDCA (*Enhanced Distributed Channel Access*). O EDCA é um aprimoramento da função DCF do IEEE 802.11, sendo também conhecido como EDCF (*Enhanced Distributed Coordination Function*). Já o mecanismo de acesso controlado, denominado HCCA (*HCF Controlled Channel Access*) consiste num aprimoramento da função PCF. A entidade responsável pelo controlo da qualidade de serviço numa célula que suporta o IEEE 802.11e (QBSS, *QoS Basic Service Set*) recebe a designação de coordenador híbrido (HC, *Hybrid Coordinator*) e reside no ponto de acesso.

3.6.4.1.1 Acesso baseado em contenção

O EDCA permite a diferenciação de serviços na rede IEEE 802.11 através da classificação dos fluxos em diferentes classes de tráfego denominadas categorias de acesso (AC, *Access Category*), em função da sua prioridade. Cada categoria de acesso é uma variante da função DCF que compete pelo acesso ao meio utilizando um conjunto distinto de parâmetros, que incluem o AIFS[AC] (análogo ao DIFS da função DCF), o CW_{min}[AC] e o CW_{max}[AC] e são anunciados na trama *Beacon*. Quanto menor o valor destes parâmetros maior é a probabilidade de acesso ao meio para a correspondente categoria de acesso. Desta forma, o suporte de QoS do mecanismo EDCA é obtido estatisticamente pelo aumento da probabilidade de acesso ao meio para os fluxos com maior prioridade e vice-versa. O ponto de acesso pode ajustar esses parâmetros dinamicamente em função das condições na rede para melhor satisfazer a qualidade de serviço das aplicações. Para que o coordenador híbrido tenha prioridade no acesso ao meio em relação às demais estações, o período AIFS[AC] mínimo deve ser maior que o PIFS, pelo que não pode ser inferior ao DIFS, ou seja:

$$AIFS[AC] = SIFS + AIFSN[AC] \cdot SlotTime, \tag{3.4}$$

onde *AIFSN[AC]* é um inteiro maior ou igual a 2.

Cada estação que implementa o IEEE 802.11e (QSTA) pode possuir até quatro categorias de acesso (AC) para o suporte de oito níveis de prioridade (UP, *User Priority*), conforme definidos na norma 802.1D [IEE98b], sendo que cada nível de prioridade é mapeado numa categoria de acesso.

Cada categoria de acesso no interior de uma estação comporta-se como uma estação virtual, com a sua própria fila de espera, competindo pelo acesso ao meio e executando o processo de *backoff* de forma independente. A diferença é que não ocorrem colisões internas quando duas ou mais categorias de acesso numa mesma estação terminam o processo de *backoff* em simultâneo. Neste caso, a estação atribui a oportunidade de transmissão à categoria de acesso mais prioritária, enquanto as demais categorias de acesso envolvidas comportam-se como se tivesse havido uma colisão externa.

3.6.4.1.2 Acesso controlado

O mecanismo de acesso controlado do IEEE 802.11e (HCCA) é baseado em *polling*, tal como a função PCF do IEEE 802.11. No entanto, ao contrário desta, sua acção não está restrita ao período livre de contenção (CFP), pois permite que o ponto de acesso interrogue as estações durante o período de contenção (CP), bastando para isso que o meio permaneça livre por um período PIFS.

No HCCA, o coordenador híbrido é responsável pelo controlo de admissão e pelo escalonamento de tráfego de forma a proporcionar a qualidade de serviço negociada com as estações. Para poder utilizar o HCCA, a QSTA deve submeter uma requisição ao controlador híbrido, contendo a especificação de tráfego (TSPEC, *Traffic Specification*). O elemento TSPEC contém uma série de parâmetros que definem as características de tráfego e expectativas de qualidade de serviço associadas a um dado fluxo unidireccional. Os parâmetros considerados mais relevantes são:

- **Débito médio de dados (ρ):** débito médio de transferência dos pacotes.
- **Limite de atraso (D):** atraso máximo para a transferência do pacote na rede sem fios, incluindo o atraso na fila de espera.
- **Tamanho nominal do MSDU (L):** tamanho nominal dos pacotes.

Cabe ao coordenador híbrido avaliar se há recursos suficientes disponíveis para satisfazer a TSPEC requerida, podendo propor uma TSPEC alternativa (eventualmente com um nível de qualidade de serviço inferior) ou mesmo rejeitar a requisição por completo.

3.6.4.1.3 Oportunidade de transmissão

A oportunidade de transmissão (TXOP, *Transmission Opportunity*) é um intervalo de tempo no qual a estação tem direito de ocupar o canal. Com este parâmetro o coordenador híbrido impõe um limite à duração das transmissões das QSTAs, obtendo maior controlo sobre o acesso ao meio e podendo, com isso, gerir melhor a qualidade de serviço oferecida aos diferentes fluxos. A duração do TXOP para o EDCA é definida por um parâmetro incluído nas tramas *Beacon*, enquanto sua duração no caso do HCCA é especificada na trama de interrogação.

Uma QSTA deve abster-se de transmitir um MPDU se o período necessário (incluindo as tramas de controlo associadas, se for o caso) exceder a duração da sua oportunidade de transmissão. Da mesma forma, a QSTA deve evitar a transmissão se não puder terminá-la antes do instante previsto para o início de transmissão da próxima trama *Beacon* (TBTT, *Target Beacon Transmission Time*). Com isso, consegue-se evitar o problema do encurtamento do período livre de contenção (desde que não existam estações na rede que só reconheçam a função DCF).

O IEEE 802.11e também permite que uma estação transmita múltiplos MPDUs em sequência (CFB, *Contention Free Burst*), contanto que a duração da respectiva oportunidade de transmissão não seja ultrapassada, com o objectivo de aumentar a eficiência do protocolo de controlo de acesso ao meio.

3.6.4.1.4 Outros aprimoramentos

No standard IEEE 802.11 original, duas estações associadas a um BSS que opere em modo baseado em infraestrutura não podem comunicar directamente entre si, mesmo que estejam ao alcance uma da outra. Neste caso, a estação emissora tem que enviar os dados para o ponto de acesso, que os retransmite para a estação destino, pelo que este processo é pouco eficiente quando grande parte da comunicação dá-se entre estações no mesmo BSS. Visando proporcionar a comunicação directa entre as estações num mesmo BSS, a norma IEEE 802.11e inclui um protocolo de ligação directa (DLP, *Direct Link Protocol*), de implementação opcional. Antes de iniciar a comunicação directa, as QSTAs envolvidas devem negociar com o ponto de acesso os parâmetros de transmissão utilizando o mecanismo de sinalização definido no protocolo.

No IEEE 802.11, o *overhead* associado à transmissão das tramas ACK após cada trama de dados recebida sem erros tem impacto sobre a eficiência do protocolo de controlo de acesso ao meio, tanto mais porque a trama ACK, por ser uma trama de controlo, pode ter que ser transmitida utilizando um modo de transmissão inferior ao utilizado pela trama de dados. Com o objectivo de aumentar a eficiência, a norma IEEE 802.11e define um mecanismo de reconhecimento de grupo (*Group ACK*) opcional, que permite que uma estação transmita uma sequência de tramas de dados,

separadas por um período SIFS, sem que seja necessário que o receptor reconheça individualmente cada uma. O reconhecimento das múltiplas tramas de dados passa a ser feita, neste caso, por uma única trama de ACK de grupo.

3.6.4.2 802.11g

O grupo de tarefas G do 802.11 foi formado para desenvolver as especificações de um sistema que opere na banda ISM de 2.4 GHz e suporte débitos de até 54 Mbit/s. Tal como acontece com a norma IEEE 802.11a, a norma IEEE 802.11g é baseada em modulação OFDM, cuja utilização na banda de 2.4 GHz só recentemente deixou de ser proibida pela FCC. As estações baseadas no IEEE 802.11g também devem suportar todos os modos de transmissão do IEEE 802.11b, para que possam operar normalmente numa rede 802.11b, o que implica a implementação da modulação CCK.

Um problema que se coloca neste caso é a detecção das transmissões que utilizam a modulação OFDM pelas estações baseadas no IEEE 802.11b, de modo a evitar colisões. Uma alternativa consiste em preceder a transmissão de dados com as tramas RTS e CTS, transmitidas utilizando modulação CCK, que é reconhecida pelo IEEE 802.11b. Opcionalmente, pode-se utilizar um esquema de modulação híbrido, que conjuga as modulações CCK e OFDM. A modulação CCK é utilizada para transmissão do preâmbulo e do cabeçalho dos pacotes, enquanto a modulação OFDM é usado para a transmissão do *payload*. Com isso, as estações do IEEE 802.11b são alertadas do início de uma transmissão, e informadas da duração da mesma. Depois, os dados são transmitidos ao débito mais elevado proporcionado pelo OFDM. A contrapartida da compatibilidade proporcionada por estas soluções é o decréscimo do desempenho em relação à utilização exclusiva da modulação OFDM, devido ao *overhead* introduzido pelas mesmas.

Outra técnica de modulação opcional incluída no IEEE 802.11g é o PBCC (*Packet Binary Convolutional Coding*), uma solução criada pela Texas Instruments que proporciona débitos de transmissão de até 33 Mbit/s. Esta solução também é híbrida, sendo o preâmbulo e o cabeçalho transmitidos com modulação CCK para preservar a compatibilidade com o IEEE 802.11b.

3.7 A rede HIPERLAN/2

3.7.1 Introdução

As redes HIPERLAN/2 operam normalmente numa configuração baseada em infraestrutura constituída por dois tipos de estações: o ponto de acesso (AP, *Access Point*) e os denominados terminais móveis (MT, *Mobile Terminal*). O ponto de acesso proporciona a ligação dos terminais à infraestrutura de rede e também exerce o papel de controlador central (CC, *Central Controller*), arbitrando o acesso ao meio por parte dos terminais.

Em alternativa, uma rede HIPERLAN/2 pode operar numa configuração *ad hoc*. Neste caso, o ponto de acesso não está presente, mas a função de controlador central mantém-se, sendo realizada por um dos terminais móveis presentes na rede.

O HIPERLAN/2 suporta dois modos de operação: centralizado e directo, sendo a implementação do primeiro obrigatória em todas as estações.

- **Modo centralizado:** Neste caso, todo tráfego passa pelo ponto de acesso, mesmo quando a comunicação dá-se entre terminais móveis pertencentes à mesma célula. Este modo de operação admite apenas a ligação descendente (do ponto de acesso para os terminais) e a ligação ascendente (dos terminais para o ponto de acesso). O pressuposto básico é que a maior parte do tráfego é trocado com dispositivos situados no exterior da célula.

- **Modo directo:** Neste caso, o controlo de acesso ao meio continua a ser efectuado de forma centralizada, pelo controlador central (CC). Entretanto, o tráfego transferido entre terminais móveis situados na mesma célula não tem de passar pelo CC, pois este modo de operação admite a comunicação directa entre os terminais. Este modo é vocacionado para aplicações em que a maior parte do tráfego é trocado dentro de uma mesma célula.

A arquitectura protocolar do HIPERLAN/2 é bastante flexível, de modo a facilitar a interligação com uma variedade de redes fixas. Por exemplo, o HIPERLAN/2 pode ser usado como um segmento de uma rede Ethernet ou como uma rede de acesso aos

sistemas celulares móveis de terceira geração. A pilha protocolar do HIPERLAN/2 é dividida num plano de controlo e um plano do utilizador, seguindo a abordagem utilizada pela Rede Digital com Integração de Serviços (RDIS) [ITU88] [NUN92], como ilustra a Figura 3.19. O plano do utilizador inclui funções para transmissão de tráfego sobre conexões preestabelecidas, enquanto o plano de controlo inclui funções para controlar o estabelecimento, supervisão e dissolução das conexões.

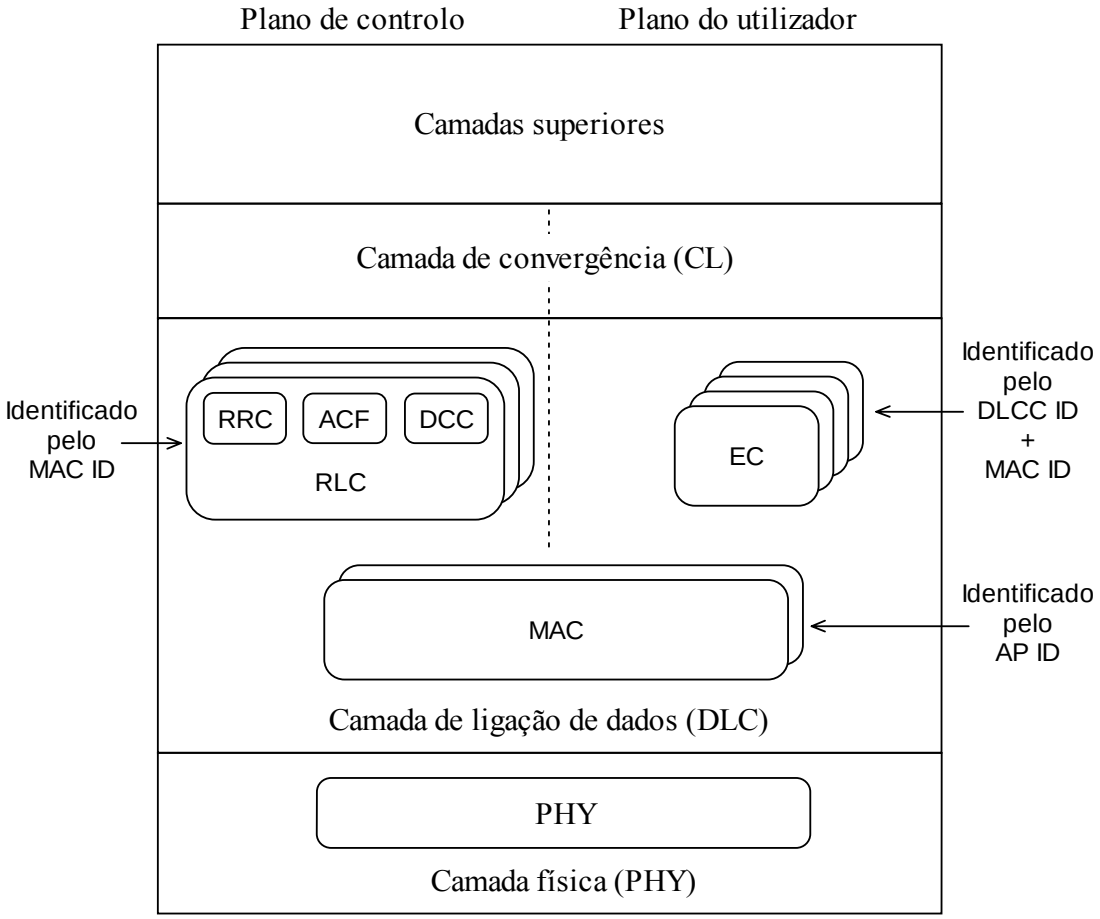


Figura 3.19: Modelo de referência protocolar do HIPERLAN/2.

As três camadas protocolares especificadas pela arquitectura do HIPERLAN/2 são a camada física (PHY), a camada de ligação de dados (DLC, *Data Link Control*) e a camada de convergência (CL, *Convergence Layer*). A camada de ligação de dados (DLC) inclui as funções de controlo de acesso ao meio (MAC, *Medium Access Control*), controlo de erros (EC, *Error Control*) e controlo da ligação de rádio (RLC, *Radio Link Control*).

A operação do HIPERLAN/2 é orientada à conexão. Cada conexão DLC do utilizador é univocamente identificada, dentro de uma célula, pela combinação do identificador de conexão DLC (DLCC ID) com o identificador de MAC (MAC ID). Este último identifica o terminal móvel, sendo atribuído pelo ponto de acesso durante a associação, enquanto o primeiro diferencia as conexões de um mesmo terminal móvel. Uma instância da função de controlo de erros (EC) é criada para cada conexão DLC. As conexões podem ser estabelecidas e desfeitas dinamicamente, caso a camada superior seja orientada à conexão; caso contrário, pelo menos uma conexão DLC deve ser estabelecida para lidar com todo o tráfego do terminal.

A cobertura de uma célula no HIPERLAN/2 estende-se de aproximadamente 30 m (interiores) até 150 m (exteriores).

3.7.2 Camada física

A camada física (PHY) do HIPERLAN/2 [ETS01a] é baseada no esquema de modulação OFDM, que proporciona um bom desempenho contra os efeitos da propagação multipercurso. O débito da camada física pode variar de 6 Mbps até 54 Mbps, através da utilização de diferentes alternativas de modulação e codificação, como mostra a Tabela 3.3. O HIPERLAN/2 opera na banda de 5 GHz e cada canal ocupa 20 MHz, o que permite a coexistência de diversos canais na banda de frequências disponível para a sua operação.

Tabela 3.3: Modos de transmissão da rede HIPERLAN/2.

Modo de transmissão	Débito (Mbit/s)	Modulação	Taxa de codificação
1	6	BPSK	1/2
2	9	BPSK	3/4
3	12	QPSK	1/2
4	18	QPSK	3/4
5	27	16QAM	9/16
6	36	16QAM	3/4
7	54	64QAM	3/4

A principal função da camada física do HIPERLAN/2 é oferecer serviços de transferência de informação à camada de ligação de dados (DLC). Para isso, a camada física implementa diversas funcionalidades de forma a mapear os trens de PDUs[20] da camada de ligação de dados em estruturas denominadas *PHY bursts*, que são usadas para modular o sinal da portadora no emissor. A Figura 3.20 ilustra os estágios envolvidos na geração de um *PHY burst* a partir do respectivo trem de PDUs, descritos a seguir.

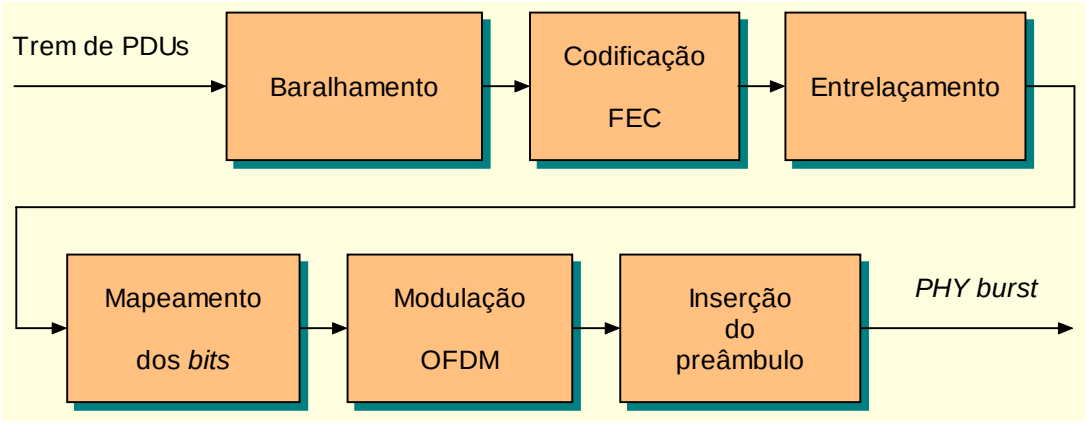


Figura 3.20: Estágios envolvidos na geração de um *PHY burst* pela camada física do HIPERLAN/2, no emissor.

[20] Os trens de PDUs são descritos na secção 3.7.3.1.3.

1. Baralhamento (*scrambling*) do conteúdo do trem de PDUs.
2. Codificação dos bits anteriormente baralhados, de acordo com o esquema de correcção de erros (FEC) associado ao modo de transmissão utilizado. O esquema de correcção de erros utilizado pelo HIPERLAN/2 é baseado em codificação convolucional.
3. Entrelaçamento (*interleaving*) dos bits codificados.
4. Mapeamento dos bits entrelaçados em pontos de constelação da modulação BPSK, QPSK, 16QAM ou 64QAM, dependendo do modo PHY escolhido.
5. Produção do sinal complexo de banda básica da modulação OFDM, pela divisão dos símbolos complexos representando os pontos de constelação, obtidos no estágio anterior, em grupos de 48, e a sua associação às 48 subportadoras OFDM dedicadas à transmissão de dados[21].
6. Anexação do preâmbulo, cujo formato depende do tipo de trem de PDUs, aos símbolos OFDM produzidos pelo estágio anterior, formando assim o respectivo *PHY burst*.

A rede HIPERLAN/2 foi concebida para operar numa banda de frequências na região entre 5 e 6 GHz. Nesta região, a frequência central dos canais é definida em múltiplos de 5 MHz acima de 5 GHz. A relação entre a frequência central do canal (f_{ch}), em MHz, e o número do canal (n_{ch} = 0, 1 ... 200) é expressa por:

$$f_{ch} = 5000 + 5 \times n_{ch}. \tag{3.5}$$

O espaçamento entre os canais do HIPERLAN/2 é de 20 MHz, e nem toda a banda de frequências entre 5 e 6 GHz está disponível. Na Europa, os canais válidos para operação da rede HIPERLAN/2 são os apresentados na Tabela 3.4.

[21] A informação relativa ao grupo de 48 pontos de constelação, juntamente com a informação associada às 4 subportadoras piloto, forma um símbolo OFDM, cujo período é 4 μs.

Tabela 3.4: Canais válidos para a operação da rede HIPERLAN/2 na Europa.

Banda (GHz)	Número do canal	Frequência central do canal (MHz)
Inferior	36	5180
	40	5200
	44	5220
	48	5240
	52	5260
	56	5280
	60	5300
	64	5320
Superior	100	5500
	104	5520
	108	5540
	112	5560
	116	5580
	120	5600
	124	5620
	128	5640
	132	5660
	136	5680
	140	5700

3.7.3 Camada de ligação de dados

A camada de ligação de dados (DLC) do HIPERLAN/2 [ETS00b], que inclui as funções de controlo de acesso ao meio (MAC), controlo de erros (EC) e controlo da ligação de rádio (RLC), suporta um conjunto extenso de mensagens de dados e controlo. Todas essas mensagens são mapeadas nos chamados canais lógicos que, por sua vez, são mapeados nos canais de transporte. Os canais lógicos definem o tipo de informação que carregam, enquanto os canais de transporte definem o formato da mensagem[22].

[22] Os diferentes canais lógicos e de transporte são representados por abreviaturas de quatro e três letras, respectivamente.

Segue-se uma breve descrição dos canais lógicos definidos pelas especificações do HIPERLAN/2.

- **Broadcast Control CHannel (BCCH):** É utilizado pelo ponto de acesso para difusão de informação de controlo referente à operação da célula.
- **Frame Control CHannel (FCCH):** Este canal descreve a estrutura e atribuição de recursos da respectiva trama MAC.
- **Random access Feedback CHannel (RFCH):** O propósito deste canal é informar aos terminais que utilizaram os canais RCH (*Random access CHannel*) na trama anterior dos resultados das suas tentativas de acesso.
- **RLC Broadcast CHannel (RBCH):** Este canal é utilizado para difusão de informação adicional referente à operação da célula, quando necessário.
- **Dedicated Control CHannel (DCCH):** É utilizado para o envio de mensagens de controlo da ligação de rádio (RLC).
- **User Data CHannel (UDCH):** O canal UDCH é utilizado para transmissão de dados do utilizador entre o ponto de acesso e um terminal móvel, no modo centralizado, ou entre dois terminais, no modo directo.
- **User Multicast CHannel (UMCH):** É utilizado para transmissão de dados *multicast*, na ligação descendente (a partir do ponto de acesso) ou na ligação directa (a partir de um terminal).
- **User Broadcast CHannel (UBCH):** É utilizado para transmissão de dados *broadcast*, na ligação descendente (a partir do ponto de acesso) ou na ligação directa (a partir de um terminal).
- **Link Control CHannel (LCCH):** O canal LCCH é utilizado para o envio de mensagens de controlo de erros, no sentido contrário ao dos dados enviados no canal UDCH, e para o envio de mensagens de requisição de recursos (RR), na ligação ascendente.
- **Association control CHannel (ASCH):** É utilizado pelos terminais, na ligação ascendente, para envio de mensagens de associação e *handover*.

Os canais de transporte são os elementos básicos para a construção de trens de PDUs. Os canais de transporte definidos pelo HIPERLAN/2 e suas características são apresentados na Tabela 3.5. Os débitos de transmissão que constam na tabela são referentes à camada física.

Tabela 3.5: Características dos canais de transporte.

Canal de transporte	Sentido	Débito (Mbps)	Comprimento (octetos)
Broadcast control CHannel (BCH)	Descendente	6	15
Frame control CHannel (FCH)	Descendente	6	Múltiplo de 27
Access feedback CHannel (ACH)	Descendente	6	9
Long transport CHannel (LCH)	Todos	Todos[23]	54
Short transport CHannel (SCH)	Todos	6/9/18	9
Random access CHannel (RCH)	Ascendente	6	9

O mapeamento entre canais lógicos e canais de transporte admite apenas algumas combinações, dependendo do tipo de ligação, como é representado na Figura 3.21 para a ligação descendente, na Figura 3.22 para a ligação ascendente e na Figura 3.23 para a ligação directa.

[23] Os débitos a utilizar pelos canais SCH e LCH são sinalizados no canal FCH. O canal SCH suporta apenas os modos de transmissão 1, 2 e 4 definidos na Tabela 3.3, enquanto o canal LCH suporta os sete modos.

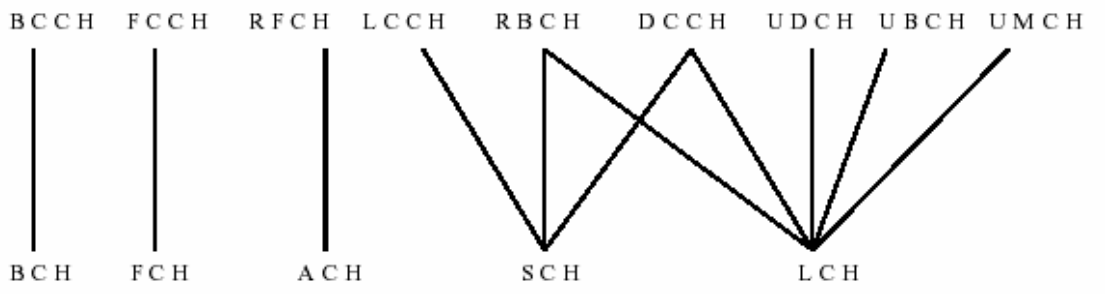


Figura 3.21: Mapeamento entre canais lógicos e canais de transporte na ligação descendente.

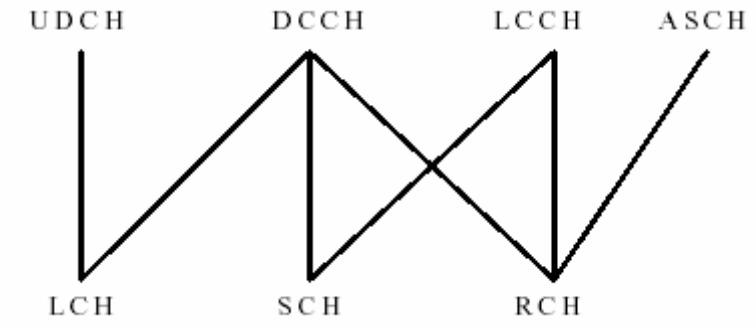


Figura 3.22: Mapeamento entre canais lógicos e canais de transporte na ligação ascendente.

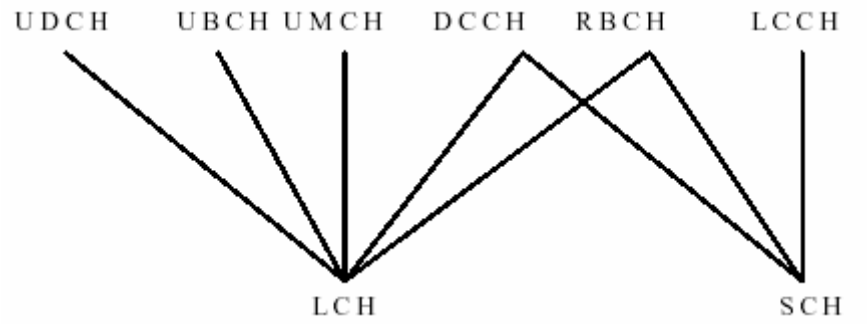


Figura 3.23: Mapeamento entre canais lógicos e canais de transporte na ligação directa.

3.7.3.1 Subcamada de controlo de acesso ao meio

O controlo de acesso ao meio no HIPERLAN/2 é realizado por um mecanismo de reserva dinâmica explícita baseado em TDMA/TDD. Este mecanismo requer a presença de um controlador central (CC), normalmente localizado no ponto de acesso (AP), que realiza o escalonamento do tráfego com base no estado dos seus próprios *buffers* de pacotes e no estado dos *buffers* dos terminais móveis. Para dar a conhecer

ao ponto de acesso o estado de seus *buffers*, os terminais fazem utilizam mensagens de requisição de recursos (RR)[24]. Na posse dessa informação, o ponto de acesso atribui os recursos tendo em consideração critérios de justiça e qualidade de serviço (QoS) [25]. Os terminais móveis são informados da alocação de recursos por intermédio das mensagens de atribuição de recursos (RG, *Resource Grant*) transmitidas pelo ponto de acesso.

O formato básico da trama MAC do HIPERLAN/2, para sistemas com um único sector, é representado na Figura 3.24. A sua duração é fixa (2 ms), ao contrário da duração de cada fase que varia em função do tráfego.

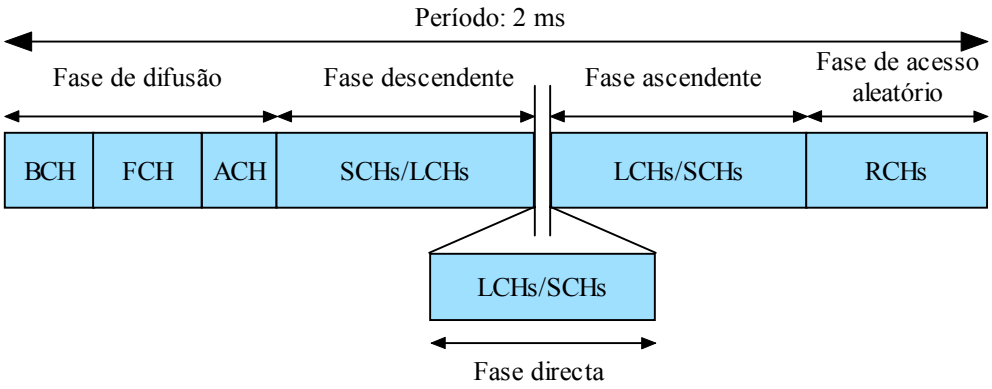


Figura 3.24: Formato básico da trama MAC do HIPERLAN/2.

A fase de difusão, composta pelos canais BCH (*Broadcast control CHannel*), FCH (*Frame control CHannel*) e ACH (*Access feedback CHannel*), contém informação enviada aos terminais móveis pelo ponto de acesso.

O canal BCH/BCCH é utilizado pelo ponto de acesso para difundir informação referente à operação da célula como um todo, incluindo, por exemplo, o número de

[24] Opcionalmente, os terminais podem negociar com o ponto de acesso a atribuição periódica de recursos a uma dada conexão, utilizando o mecanismo de negociação de capacidade fixa (FCA, *Fixed Capacity Agreement*).

[25] O algoritmo de escalonamento utilizado para a atribuição de recursos às estações não é especificado nas normas do HIPERLAN/2.

blocos de elementos de informação (IE) no canal FCH/FCCH e o número de canais RCH na fase de acesso aleatório da trama corrente.

O canal FCH/FCCH fornece uma descrição precisa da forma como os recursos são alocados nas fases descendente, ascendente e directa da trama corrente. É composto por até 15 blocos de elementos de informação (IE). Cada bloco é composto por 3 elementos de informação e por um campo de CRC calculado sobre esses elementos, como ilustra a Figura 3.25.

Cada elemento de informação contém uma mensagem de atribuição de recursos (RG) associada a uma dada conexão e a um dado sentido de transmissão[26]. As mensagens de atribuição de recursos contêm o identificador de MAC (MAC ID), o identificador de conexão DLC (DLCC ID), um apontador para o início da transmissão na trama (*start pointer*), o número de canais SCH e LCH atribuídos e os respectivos modos de transmissão na camada física.

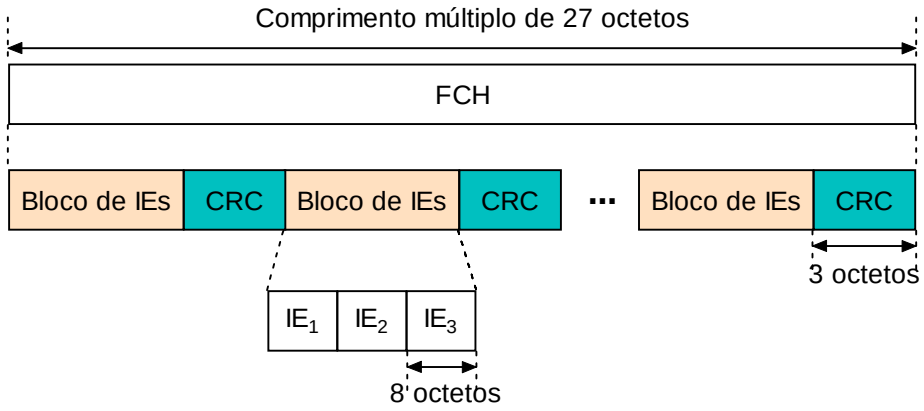


Figura 3.25: Estrutura do canal FCH.

O canal ACH/RFCH é utilizado pelo ponto de acesso para informar aos terminais os resultados das tentativas de acesso aos *slots* da fase de acesso aleatório da trama anterior. A cada *slot* é associado um bit de sinalização, que toma o valor "1" caso o

[26] Existem também elementos de informação para partes vazias da trama e para preencher o último bloco no FCH quando o número de elementos de informação original não é múltiplo de 3.

ponto de acesso tenha recebido com sucesso uma transmissão no respectivo *slot*, e toma o valor "0" caso contrário[27].

As fases descendente (DL), ascendente (UL) e directa[28] (DiL) são preenchidas com dois tipos de PDUs (longos e curtos), que utilizam os canais de transporte LCH (*Long transport CHannel*) e SCH (*Short transport CHannel*), respectivamente.

A principal função do canal LCH consiste no transporte de dados do utilizador, em canais lógicos UDCH, UMCH ou UBCH, dependendo do tipo de conexão. Este canal também pode ser usado para o transporte de mensagens de controlo, em canais lógicos DCCH ou RBCH. O canal LCH admite ainda o transporte de uma mensagem vazia (*dummy* LCH).

O canal LCH possui um comprimento fixo de 432 bits (54 octetos), sendo composto por três campos, como ilustra a Figura 3.26: tipo de PDU LCH (2 bits); *payload* do canal LCH (406 bits); e CRC (24 bits). Os canais lógicos, por sua vez, dividem o campo de *payload* do canal LCH em dois campos: número de sequência (10 bits) e *payload* do canal lógico (396 bits, ou seja, 49.5 octetos).

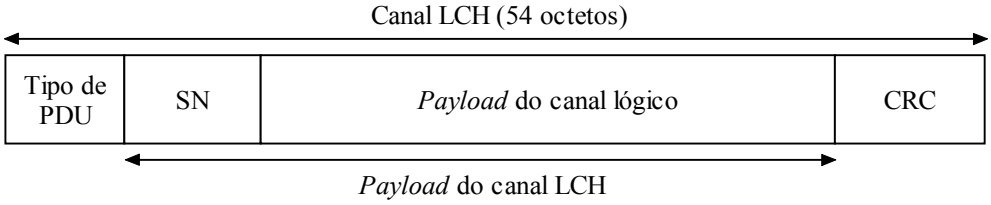


Figura 3.26: Estrutura do canal de transporte LCH e dos respectivos canais lógicos.

O canal SCH destina-se ao transporte de um variado leque de mensagens de controlo, como pode ser visto na Tabela 3.6, entre as quais se incluem as mensagens de ARQ e as mensagens de requisição de recursos (RR), sendo que estas últimas também podem ser transmitidas em canais RCH durante a fase de acesso aleatório.

[27] Ou seja, no caso da transmissão ser corrompida devido a colisão ou erros no canal, ou quando o *slot* é deixado vazio.

[28] A fase directa só está presente quando o modo directo de operação é suportado.

Tabela 3.6: Códigos do campo tipo de PDU SCH.

Tipo de PDU SCH	Utilização
0000	Reservado
0001	Mensagem de ARQ
0010	Mensagem de descarte
0011	Mensagem de requisição de recursos para a fase ascendente
0100	Mensagem RLC de/para o ponto de acesso
0101	Mensagem de requisição de recursos para a fase directa
0110	Não permitido (utilizado pelo canal ACH)
0111	Mensagem RLC no modo directo
1000	*Seed* de encriptação
1001	*Dummy* SCH
1110-1111	Reservado

O canal SCH possui um comprimento de 72 bits (9 octetos), sendo composto pelos seguintes campos, representados na Figura 3.27: tipo de PDU SCH (4 bits); campo de informação (52 bits); e CRC (16 bits).

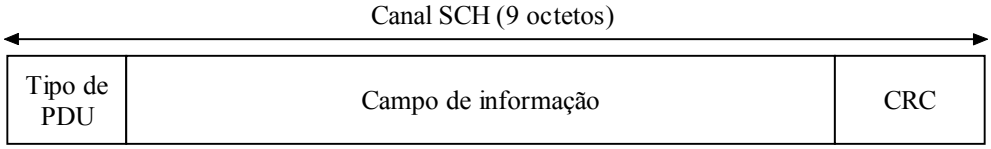


Figura 3.27: Estrutura do canal de transporte SCH.

A fase de acesso aleatório é composta por um conjunto de *slots*[29] que os terminais podem usar para enviar mensagens de requisição de recursos (RR) ou de associação para o ponto de acesso. Essas mensagens são transportadas em canais RCH (*Random*

[29] O número de *slots* disponíveis pode variar em cada trama, conforme a decisão do ponto de acesso, entre um mínimo de 1 e um máximo de 31 *slots*.

access CHannel), de formato idêntico ao dos canais SCH. O acesso aos *slots* na fase de acesso aleatório é sujeito a um processo de contenção, descrito a seguir.

3.7.3.1.1 Processo de contenção na fase de acesso aleatório

O algoritmo que regula as transmissões dos terminais móveis durante a fase de acesso aleatório baseia-se nas seguintes variáveis.

- Contador do número de retransmissões feitas pelo terminal móvel (a). Na primeira tentativa de envio de uma mensagem, seu valor é zero. Quando uma transmissão não é reconhecida pelo ponto de acesso, o contador é incrementado para a próxima tentativa. Quando a transmissão é reconhecida o contador é reinicializado com zero e o processo de contenção termina.
- Tamanho da janela de contenção do terminal móvel (CW_a). Seu valor depende do número de retransmissões, como é descrito abaixo.
- *Slot* em que é feita a transmissão (r_a). Seu valor é obtido de forma aleatória a partir de uma distribuição inteira uniforme no intervalo [1, CW_a].
- Número de *slots* RCH na trama em que r_a é calculado (n).

O primeiro passo do algoritmo de contenção consiste no cálculo do tamanho da janela de contenção (CW_a), que obedece às seguintes regras.

1) Tentativa inicial ($a = 0$):

$$CW_0 = n$$

2) Retransmissões ($a \geq 1$):

$$CW_a = \begin{cases} n, & n \geq 2^a \\ 2^a, & n < 2^a < 256 \\ 256, & 2^a \geq 256 \end{cases}$$

Conhecido o tamanho da janela de contenção, a estação calcula o valor de r_a. O primeiro *slot* da trama na qual o valor de r_a é calculado serve de referência para a determinação da posição em que a mensagem será transmitida. Caso $r_a \leq n$, a

mensagem é transmitida nessa trama; caso contrário, a contagem dos *slots* continua sequencialmente nas tramas seguintes até que a posição de transmissão seja atingida.

3.7.3.1.2 Negociação de capacidade fixa

O mecanismo de negociação de capacidade fixa (FCA, *Fixed Capacity Agreement*) permite a atribuição periódica de recursos a uma dada conexão de forma automática. Com este mecanismo, o terminal estabelece um acordo com o ponto de acesso para que este escalone um certo número de canais LCH e SCH em determinadas tramas. Esta negociação é feita durante o estabelecimento da conexão DLC, ou na altura da modificação dos seus parâmetros. Recursos adicionais podem ser pedidos pelo terminal utilizando mensagens de requisição de recursos (RR), se necessário.

O ponto de acesso atribui o número de canais LCH e SCH que foi negociado de forma periódica, sendo o espaçamento expresso por um número fixo de tramas MAC. O número máximo de canais SCH que pode ser atribuído numa trama é um, e a atribuição dos canais LCH e SCH deve estar alinhada nas tramas. Um exemplo de atribuição de canais utilizando o mecanismo FCA consiste na atribuição de 3 canais LCH a cada 5 tramas e de um canal SCH a cada 10 tramas.

3.7.3.1.3 Mapeamento da trama MAC na camada física

No HIPERLAN/2, a posição de cada transmissão na trama MAC, bem como a sua duração, é rigorosamente controlada pelo ponto de acesso, excepto durante a fase de acesso aleatório, em que a posição de transmissão é regulada pelo algoritmo de controlo de contenção.

Para compor a estrutura de uma trama MAC, é necessário conhecer a duração de cada um dos seus componentes: canais de transporte, preâmbulos e períodos de guarda. O período ocupado por um canal de transporte depende do seu comprimento,

que é fixo[30], e do modo de transmissão utilizado, que é variável no caso dos canais SCH e LCH e fixo nos outros casos, como foi mostrado na Tabela 3.5.

O comprimento dos preâmbulos no HIPERLAN/2 é especificado em número de símbolos OFDM, cuja duração é de 4 μs. O preâmbulo do BCH tem o comprimento de 4 símbolos OFDM, ou seja, sua duração é de 16 μs. No caso de um único sector por célula, os de canais FCH e ACH são transmitido imediatamente após o término do BCH, formando um único trem de PDUs, como mostra a Figura 3.28.

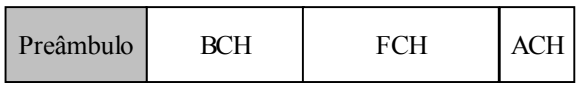


Figura 3.28: Trem de PDUs de difusão e respectivo preâmbulo, para sistemas com um único sector.

Todos os PDUs provenientes de (ou encaminhados para) um dado terminal móvel devem ser reunidos num mesmo trem de PDUs, agrupados por conexão DLC, como é exemplificado na Figura 3.29. Para cada conexão, os canais SCH atribuídos precedem os canais LCH, durante a fase descendente. Já nas fases ascendente e directa ocorre justamente o contrário: os canais SCH sucedem aos canais LCH.

Cada trem de PDUs, na fase descendente, é precedido por um preâmbulo formado por 2 símbolos OFDM. Já na fase ascendente, os trens de PDUs são precedidos por um preâmbulo que pode ser composto por 3 símbolos OFDM (preâmbulo curto) ou por 4 símbolos OFDM (preâmbulo longo), e são intercalados por períodos de guarda com duração mínima de 2 μs. Na fase directa, os trens de PDUs são precedidos por um preâmbulo com comprimento de 4 símbolos OFDM, sendo intercalados por um período de guarda apenas quando os terminais emissores são diferentes.

Para cada conexão DLC, a posição que marca o início da transmissão dos PDUs na trama é indicada pelo campo *start pointer*, presente na mensagem de atribuição de recursos (RG) correspondente, que é inserida no campo FCH/FCCH da trama MAC.

[30] Com excepção do canal FCH.

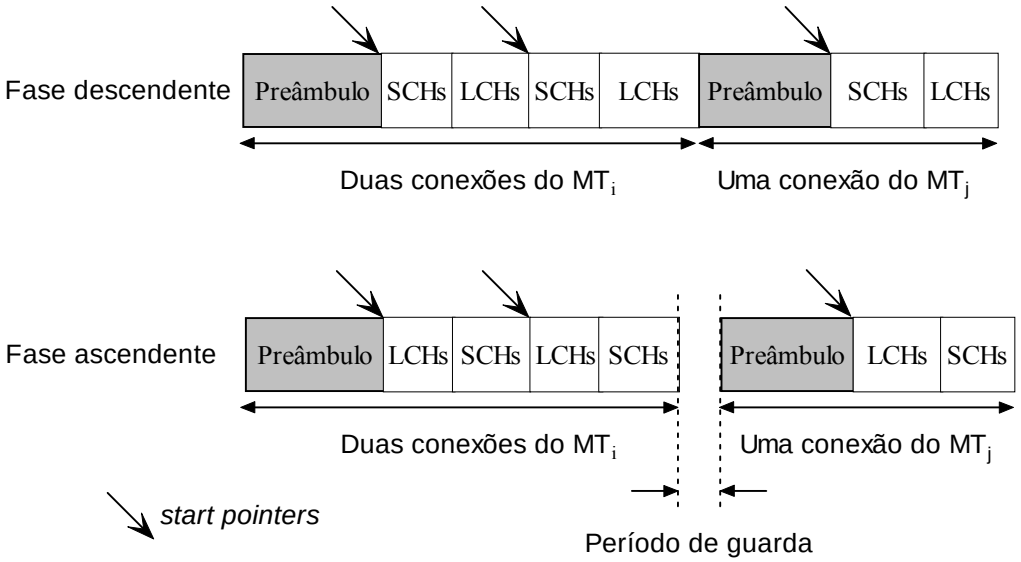


Figura 3.29: Exemplo da estrutura temporal relativa às fases descendente e ascendente.

Na fase de acesso aleatório, os canais RCH são precedidos por um preâmbulo com o mesmo comprimento usado na fase ascendente, e intercalados por períodos de guarda que podem assumir os valores de 2, 2.8, 4 ou 12 μs, como ilustra a Figura 3.30.

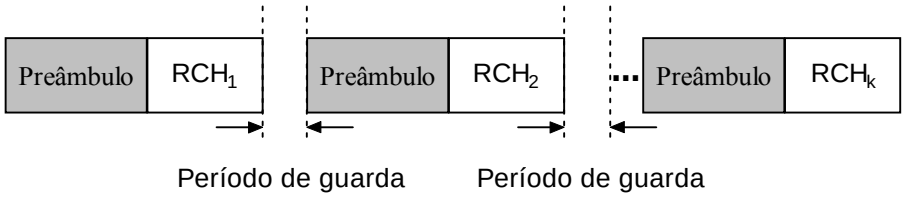


Figura 3.30: Estrutura temporal na fase de acesso aleatório.

3.7.3.2 Subcamada de controlo de erros

A principal tarefa da subcamada de controlo de erros consiste na detecção e recuperação de erros de transmissão que não tenham sido corrigidos pelos mecanismos de correcção de erros da camada física do HIPERLAN/2.

Para proporcionar protecção aos dados transmitidos nos canais de transporte LCH, as especificações do HIPERLAN/2 definem os seguintes modos de controlo de erros a nível da camada de ligação de dados.

- **Modo de reconhecimento (*acknowledged mode*):** O modo de reconhecimento utiliza um mecanismo de detecção de erros e retransmissão (ARQ), descrito na secção 3.7.3.2.1, para proporcionar a transferência fiável de informação entre o emissor e o receptor.
- **Modo de repetição (*repetition mode*):** Este modo proporciona o aumento da fiabilidade da transferência de dados através da repetição das mensagens um número predeterminado de vezes.
- **Modo sem reconhecimento (*unacknowledged mode*):** O modo sem reconhecimento executa uma única transmissão por mensagem, proporcionando uma comunicação com baixa latência, porém, não fiável. Este modo baseia-se apenas na detecção de erros pelo receptor.

O modo de reconhecimento não é utilizado com conexões *broadcast* ou *multicast*, associadas aos canais lógicos UBCH e UMCH, respectivamente, devido ao custo e complexidade inerentes ao processamento de pedidos de retransmissão provenientes de múltiplas fontes. Assim, o modo de reconhecimento é utilizado apenas com canais lógicos UDCH. Já o modo de repetição é especificado para utilização apenas com canais lógicos UBCH, enquanto que o modo sem reconhecimento pode ser utilizado com todos os tipos de canais lógicos.

Para lidar com mensagens cujo prazo de envio tenha expirado, provenientes de aplicações de tempo real, o HIPERLAN/2 define um mecanismo de descarte de mensagens. Este mecanismo permite que o emissor informe o receptor, por intermédio de uma mensagem de descarte, da sua intenção de desfazer-se das mensagens do *buffer* de transmissão cujos números de sequência sejam inferiores a um dado valor D_s. Com isso, o receptor pode proceder à entrega das mensagens pendentes, que estavam à espera das mensagens descartadas, para a camada superior, diminuindo com isso a latência na comunicação.

O modo de reconhecimento disponibiliza uma ligação LCCH bidireccional para transmissão de mensagens de ARQ, do receptor para o emissor, e para a transmissão

de mensagens de descarte, do emissor para o receptor. No modo de repetição, uma ligação LCCH unidireccional é disponibilizada, do emissor para o receptor, para fornecer um caminho para eventuais mensagens de descarte. Já o modo sem reconhecimento não proporciona nenhuma ligação de controlo para suporte à ligação de dados.

3.7.3.2.1 Modo de reconhecimento

O modo de reconhecimento, baseado num protocolo de repetição selectiva, é o principal mecanismo de controlo de erros disponível para as conexões *unicast* do HIPERLAN/2.

Para possibilitar o funcionamento do mecanismo de correcção de erros, as mensagens de dados, transmitidas em canais lógicos UDCH, possuem um campo de número de sequência, de 10 bits, usado para identificar as mensagens da conexão, e um campo de CRC, de 24 bits, usado para detecção de erros nas mensagens.

O número de sequência inserido nas mensagens é incrementado, módulo 2^{10}, sempre que uma nova mensagem é criada. Os 10 bits do campo de número de sequência permitem identificar um conjunto de 1024 mensagens distintas. Para que não haja ambiguidade na identificação das mensagens pelo protocolo de repetição selectiva, o tamanho máximo das janelas de transmissão e recepção é limitado a metade do espaço de números de sequência. O tamanho de janela utilizado (k_s) é negociado durante o estabelecimento da conexão, podendo assumir os valores 32, 64, 128, 256 e 512.

Com o objectivo de reduzir a quantidade de mensagens de ARQ necessárias à operação do protocolo de repetição selectiva, a informação referente aos números de sequência reconhecidos pelo receptor é condensada em estruturas denominadas blocos de *bitmaps* (BMB, *BitMap Block*), que permitem cobrir uma fracção contígua do espaço de números de sequência. Cada bloco de *bitmaps* contém 8 bits, pelo que o espaço de números de sequência comporta 128 blocos. A posição de um dado bloco de *bitmaps* é referenciada por um número de bloco (BMN, *BitMap block Number*).

Num bloco de *bitmaps*, um bit "1" indica o reconhecimento positivo da mensagem cujo número de sequência é referenciado, enquanto um bit "0" indica o

correspondente reconhecimento negativo. Cada mensagem de ARQ contém três blocos de *bitmaps*. A posição do primeiro bloco (BMB1) é identificada por um número de bloco de 7 *bis* (BMN1), utilizando endereçamento absoluto; os outros dois blocos (BMB2 e BMB3) são identificados por números de bloco de 5 bits (BMN2 e BMN3), pela utilização de endereçamento relativo.

A Figura 3.31 apresenta um exemplo da utilização de blocos de *bitmaps* e respectivos números de bloco na composição duma mensagem de ARQ. No bloco 33 (BMN1 = 33), que representa os números de sequência 264 a 271, a mensagem com número de sequência 270 tem reconhecimento negativo, enquanto as restantes mensagens no bloco têm reconhecimento positivo. O valor BMN2 = 8 faz referência ao bloco 41, dado que o endereçamento é feito em relação ao número de bloco anterior (BMN1), cujo valor é 33. No bloco referenciado pelo BMN2, as mensagens com os números de sequência 329 e 330 têm reconhecimento negativo. O endereçamento de BMB3 é relativo a BMB2, e o reconhecimento das mensagens é feito de forma similar aos casos anteriores.

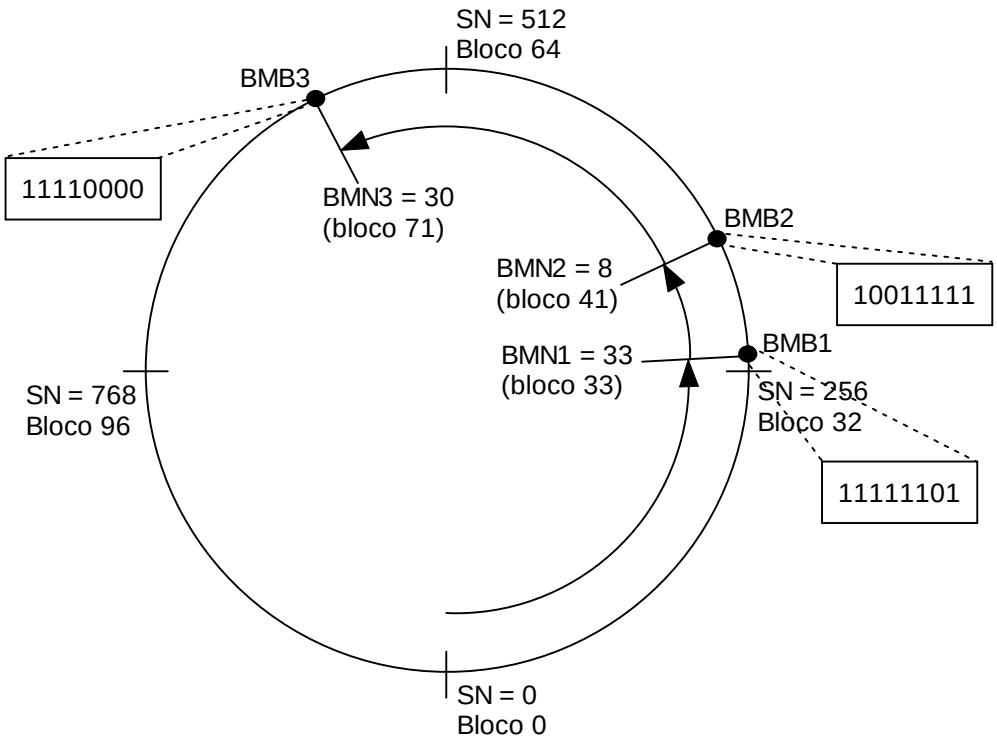


Figura 3.31: Exemplo da utilização de blocos de *bitmaps* em mensagens de ARQ.

Além do reconhecimento positivo explícito das mensagens UDCH referenciadas pelos blocos de *bitmap* da mensagem de ARQ, o receptor pode fazer o reconhecimento implícito das mensagens com números de sequência inferiores aos referenciados pelo BMB1. Para isso, o bit indicador de reconhecimento cumulativo (CAI, *Cumulative Acknowledgement Indicator*) presente na mensagem de ARQ deve estar activo (valor "1"). Noutros casos, as mensagens cujos números de sequência não estão cobertos pelos blocos de *bitmaps* da mensagem de ARQ não podem ser consideradas como tendo recebido reconhecimento positivo.

No exemplo da Figura 3.31, se o bit CAI estiver activo, as mensagens com número de sequência inferior a 270 recebem reconhecimento positivo, pelo que o emissor pode removê-las do seu *buffer* de transmissão. Além disso, o emissor deve actualizar o limite inferior da janela de transmissão com o valor 270, ou seja, o menor número de sequência não reconhecido positivamente pelo receptor.

3.7.3.3 Subcamada de controlo da ligação de rádio

A subcamada de controlo da ligação de rádio (RLC) do HIPERLAN/2 [ETS01b], situada no plano de controlo, engloba os seguintes módulos.

- **Função de controlo de associação (ACF, *Association Control Function*):** Esta função é responsável por tarefas como a associação, desassociação e autenticação dos terminais móveis (MTs) com ponto de acesso (AP), bem como a encriptação dos dados.
- **Controlo das conexões DLC (DCC, *DLC Connection Control*):** Este módulo é responsável pelo estabelecimento e dissolução das conexões do utilizador.
- **Controlo dos recursos de rádio (RRC, *Radio Resource Control*):** É responsável pela supervisão e utilização eficiente da largura de banda disponível. As funções realizadas por este módulo incluem a selecção dinâmica de frequência (DFS, *Dynamic Frequency Selection*), o controlo da potência de transmissão (TPC, *Transmission Power Control*), a função de economia de energia e o *handover*.

O mecanismo de selecção dinâmica de frequência (DFS) permite a alocação automática da frequência de operação do ponto de acesso, com base em medições

realizadas pelo ponto de acesso e pelos terminais móveis, de forma a evitar interferências. Este mecanismo dispensa o planeamento coordenado das frequências de operação das diferentes células do HIPERLAN/2 numa mesma região, sendo importante também para evitar a interferência de outras fontes, como sistemas de radar, que possam utilizar a mesma banda de frequências.

O mecanismo de controlo da potência de transmissão (TPC) deve ser implementado para as fases descendente, ascendente e directa de forma a minimizar potenciais interferências. O objectivo é reduzir a potência de transmissão média pelo menos 3 dB em relação a um sistema que não implemente o TPC. A implementação deste mecanismo passa pelo ajuste da potência de transmissão para o valor mais baixo que ainda seja suficiente para a comunicação fiável entre o emissor e o(s) receptor(es).

Aos terminais móveis que operam com baterias, a função de economia de energia permite requisitar ao ponto de acesso a entrada num estado de baixo consumo de energia, com a definição de um período de hibernação igual a N tramas, sendo $2 \leq N \leq 216$. No término do período de hibernação, três cenários são possíveis: (a) O ponto de acesso desperta o terminal, ao informá-lo da existência de dados pendentes para o mesmo; (b) o terminal desperta por ter dados a enviar; (c) o terminal continua no estado de hibernação por mais um período. A duração do período de hibernação deve ser configurada tendo em conta a necessidade de um atraso menor ou de um consumo mais baixo.

3.7.4 Camada de convergência

A função da camada de convergência (CL) consiste na adaptação entre as camadas superiores e a camada de ligação de dados do HIPERLAN/2, tanto a nível do formato das mensagens como a nível dos serviços disponíveis.

Duas camadas de convergência foram especificadas para o HIPERLAN/2: a camada de convergência baseada em células; e a camada de convergência baseada em pacotes. Futuramente, outras camadas poderão ser definidas. A camada de convergência baseada em células destina-se a oferecer serviços a camadas superiores

que utilizem o formato da célula ATM [ITU99]. Já a camada de convergência baseada em pacotes oferece serviços a camadas superiores que utilizem tanto pacotes de comprimento variável como pacotes de comprimento fixo, porém, superior ao da célula ATM.

Para suportar os diferentes tipos de redes existentes actualmente e ao mesmo tempo permitir o suporte a futuras redes, a camada de convergência é dividida em duas partes: uma parte comum; e uma parte específica, sendo esta representada pela subcamada SSCS (*Service Specific Convergence Sublayer*). A parte específica localiza-se acima da parte comum e sua implementação depende da camada superior utilizada.

3.7.4.1 Camada de convergência baseada em células

A camada de convergência baseada em células [ETS00a] destina-se à integração entre o HIPERLAN/2 e as redes ATM, UMTS e outras redes que utilizem o formato de célula ATM.

A principal função da sua parte comum, representada pela subcamada CPCS (*Common Part Convergence Sublayer*), consiste no mapeamento entre a célula ATM (53 octetos) e o CPCS-PDU[31] (49.5 octetos). Este mapeamento requer que o cabeçalho das células ATM seja incluído numa versão comprimida, da forma descrita abaixo.

Uma conexão ATM é referenciada pelo identificador de trajecto virtual (VPI, *Virtual Path Identifier*) em conjunto com o identificador de canal virtual (VCI, *Virtual Channel Identifier*). A camada de convergência baseada em células permite que células pertencentes a diferentes conexões ATM sejam multiplexadas numa mesma conexão DLC do HIPERLAN/2, pela atribuição de um identificador único, denominado CL-Tag (8 bits), para cada par <VPI, VCI>, durante o estabelecimento da conexão DLC.

[31] O CPCS-PDU é inserido no *payload* da mensagem de dados da camada DLC.

O identificador CL-Tag é inserido no CPCS-PDU, como mostra a Figura 3.32, juntamente com o campo CL-Flags (4 bits) e com os 48 octetos do *payload* da célula ATM. O campo CL-Flags é formado por dois campos do cabeçalho da célula ATM: o tipo de *payload* (PT, *Payload Type*), de 3 bits; e o bit de prioridade de descarte da célula (CLP, *Cell Loss Priority*). Os campos GFC (*Generic Flow Control*) e HEC (*Header Error Control*), que também estão presentes na célula ATM, não são transmitidos pelo HIPERLAN/2.

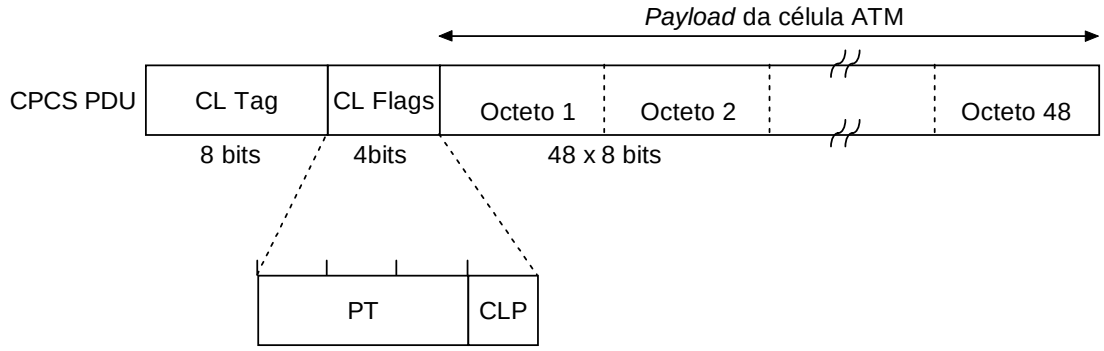


Figura 3.32: Mapeamento da célula ATM no CPCS-PDU da camada de convergência baseada em células.

3.7.4.2 Camada de convergência baseada em pacotes

A camada de convergência baseada em pacotes [ETS00c] é usada para integração entre o HIPERLAN/2 e as diversas redes baseadas em pacotes existentes, como as redes Ethernet (IEEE 802.3), TCP/IP e Firewire (IEEE 1394).

No plano do utilizador, a parte comum contém, além da subcamada CPCS (*Common Part Convergence Sublayer*), a subcamada de segmentação e reunião (SAR, *Segmentation and Reassembly*), situada mais abaixo. No lado do emissor, a subcamada CPCS adiciona um campo de *padding* e uma cauda (*trailer*) aos pacotes recebidos da subcamada SSCS (CPCS-SDU), e passa o pacote resultante (CPCS-PDU) à subcamada SAR, que segmenta estes pacotes em unidades de comprimento fixo. No receptor, realiza-se o procedimento inverso.

A Figura 3.33 representa o formato do CPCS-PDU. O *payload* (CPCS-SDU), de comprimento variável, é preenchido com o pacote recebido da subcamada SSCS. O

campo de *padding* (entre 0 e 47 octetos) não carrega informação. A sua função consiste em fazer com que o comprimento do CPCS-PDU seja múltiplo de 48 octetos. A cauda (4 bits) é formada por 2 bits reservados para utilização futura e outros 2 bits usados para representar o comprimento do *payload* do CPCS-PDU.

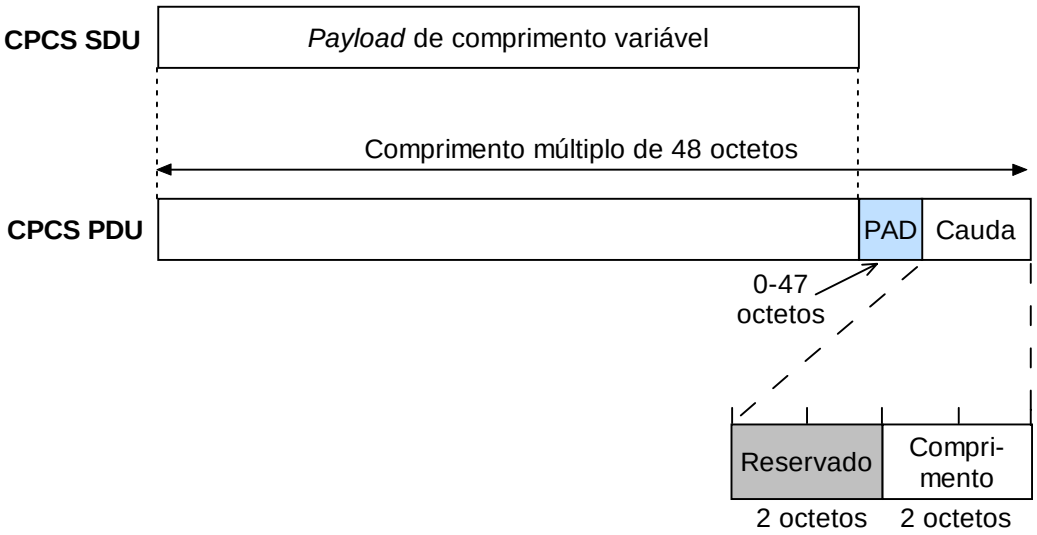


Figura 3.33: Formato do CPCS-PDU da camada de convergência baseada em pacotes.

A subcamada SAR recebe o CPCS-PDU (que corresponde ao SAR-SDU) e procede à sua segmentação, como ilustra a Figura 3.34. Os segmentos, de 48 octetos de comprimento, são inseridos no *payload* do SAR-PDU[32] (49.5 octetos). O cabeçalho do SAR-PDU é composto pelo campo CL-Tag (8 bits), reservado para utilização futura, e pelo campo CL-Flags (4 bits), cujos bits também estão reservados para utilização futura, com excepção do bit 2, denominado *SAR Stop* bit.

No emissor, a subcamada SAR atribui o valor "1" ao *SAR Stop* bit do último fragmento (SAR-PDU) proveniente de um pacote, e o valor "0" ao *SAR Stop* bit dos outros fragmentos. Na subcamada SAR do receptor, o valor "1" no *SAR Stop* bit indica que este é o último fragmento associado ao pacote, pelo que os dados armazenados no *buffer* de reunião podem ser enviados à subcamada CPCS.

[32] O SAR-PDU, por sua vez, é inserido no *payload* da mensagem de dados da camada DLC.

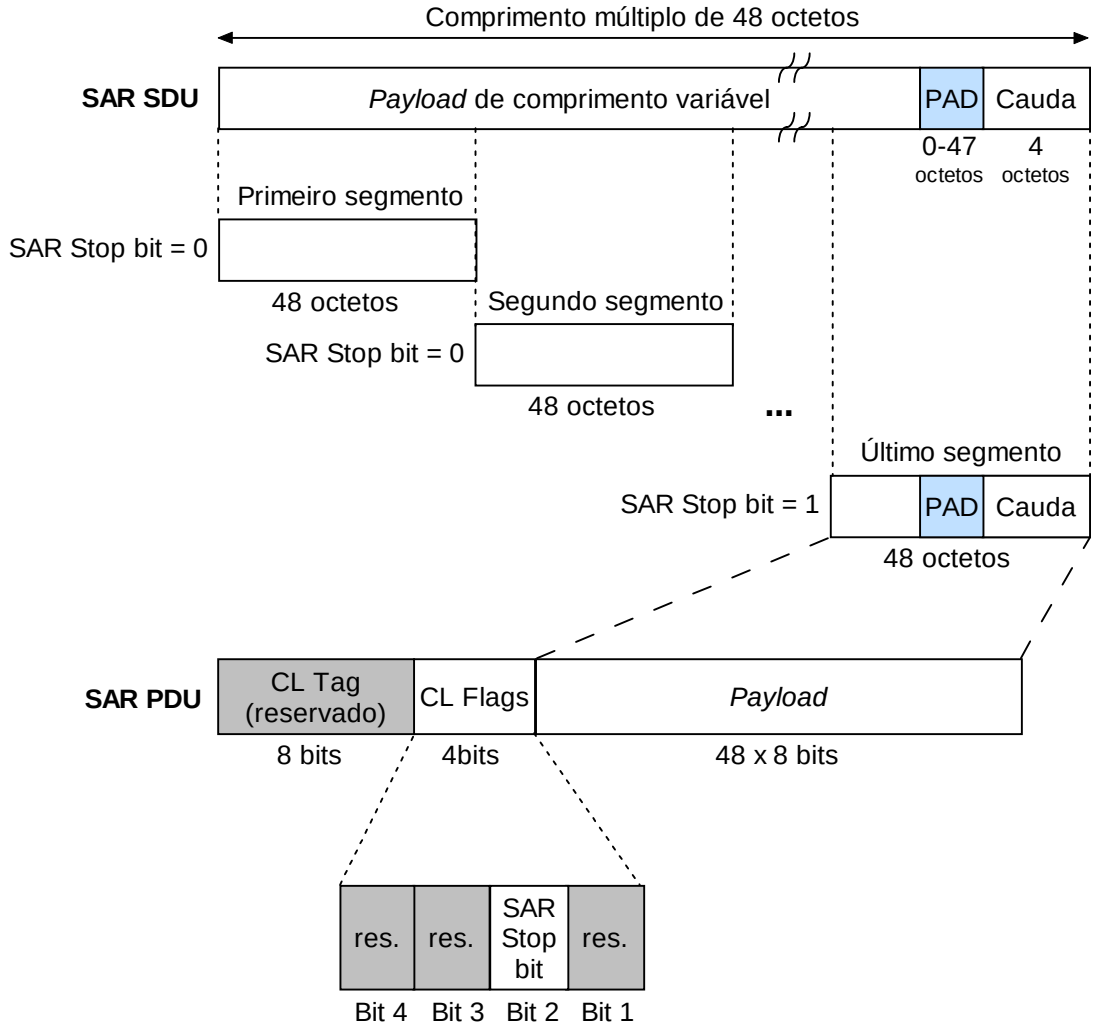


Figura 3.34: Processo de segmentação do CPCS-PDU e composição do SAR-PDU da camada de convergência baseada em pacotes.

3.8 Sumário

Nesta secção foram apresentados os principais sistemas de comunicação sem fios actualmente em operação ou em fase de normalização. Estes sistemas podem ser classificados em quatro categorias: sistemas celulares móveis, sistemas de acesso fixo sem fios, redes locais sem fios e redes pessoais sem fios.

Os sistemas celulares móveis e os sistemas de acesso fixo sem fios proporcionam o acesso dos utilizadores a uma infraestrutura de rede fixa, como a rede telefónica pública. Os utilizadores normalmente pagam pela utilização da rede de comunicação

sem fios e pelos serviços que a mesma disponibiliza. Estes sistemas operam utilizando controlo e comunicação centralizados e requerem licença de operação.

Por outro lado, as redes locais e pessoais normalmente são adquiridas para utilização privada e operam sem necessidade de licença. As redes locais costumam ter um alcance maior que as redes pessoais e servir um grupo maior de utilizadores, apresentando também maior custo e complexidade.

As características e especificações técnicas das redes IEEE 802.11 e HIPERLAN/2 foram abordadas com particular ênfase neste capítulo. Estas redes apresentam boas características para servirem de suporte de transmissão aos sistemas de aquisição de dados e controlo, como débito de transmissão elevado, suporte de QoS, integração de serviços, possibilidade de operação privada e alcance de dezenas de metros.

O débito de transmissão elevado destas redes, em conjunto com o pequeno tempo de propagação, são um incentivo ao desenvolvimento de soluções baseadas na utilização da técnica de detecção de erros e retransmissão (ARQ) que sejam capazes de recuperar rapidamente os dados afectados por erros no canal, possibilitando assim o transporte eficaz do tráfego isócrono típico dos sistemas de aquisição de dados e controlo utilizando uma rede sem fios. Por outro lado, estas redes apresentam algumas diferenças de concepção, nomeadamente a nível da camada de ligação de dados, que tornam interessante a sua utilização para comparação do desempenho das soluções propostas nesta tese, que são apresentadas no próximo capítulo.

Capítulo 4

Escalonamento de tráfego em redes sem fios

A primeira parte deste capítulo apresenta diversos trabalhos publicados por outros autores que se relacionam com o trabalho apresentado neste tese. Estes trabalhos abordam o escalonamento de tráfego em redes de comunicação sem fios e/ou a análise de desempenho das redes IEEE 802.11 e HIPERLAN/2. São referidas as características das análises e soluções propostas, suas limitações e conclusões.

Na segunda parte são apresentadas as soluções desenvolvidas nesta tese: primeiro são apresentados quatro algoritmos de escalonamento propostos para o transporte de tráfego de tempo real em redes IEEE 802.11 cujo objectivo é permitir a retransmissão rápida dos dados corrompidos devido a erros no canal; a seguir, apresenta-se um algoritmo com o mesmo objectivo para a rede HIPERLAN/2, bem como um mecanismo de adaptação do débito de transmissão para limitar o impacto das retransmissões sobre a variação do atraso e um mecanismo de alocação flexível de capacidade fixa.

4.1 Trabalhos relacionados

Os algoritmos de escalonamento usados nas redes cabladas, apresentados na secção 2.3, não são apropriados para uso nas redes sem fios, pois estas redes apresentam características particulares, como uma taxa de erros no canal muito maior e mais variável, além da qualidade e capacidade do canal depender da localização dos terminais. No entanto, os algoritmos convencionais assumem que o canal é isento de

erros. Outro problema é que a estação base, que normalmente é responsável pelo escalonamento nas redes sem fios, desconhece os instantes de chegada dos pacotes aos *buffers* dos terminais, logo não pode utilizar directamente os mecanismos de identificação (*tagging*) dos pacotes nos quais se baseiam os algoritmos convencionais.

Grande parte dos algoritmos de escalonamento propostos recentemente na literatura para utilização em redes sem fios visa proporcionar justiça na partilha da largura de banda disponível. Destes, a maior parte tem como objectivo maximizar a eficiência de utilização da rede, geralmente pela maximização do débito agregado de todos os fluxos, sacrificando a justiça a curto prazo enquanto procura manter a justiça a longo prazo. Com base em estimativas do estado do canal, os fluxos dos terminais são divididos em dois conjuntos: os que percebem um estado bom; e os que percebem um estado mau. Um fluxo neste último estado cede a sua vez de transmitir para os primeiros. O escalonador mantém um registo da largura de banda utilizada pelos diferentes fluxos. Quando um fluxo muda do estado mau para o estado bom, o escalonador compensa este fluxo penalizando os fluxos que foram beneficiados anteriormente, de forma que, no longo prazo, a largura de banda seja distribuída de forma equitativa. Os algoritmos de escalonamento que utilizam esta abordagem [LU99] [EUG98] [SEO00] [RAM98] [JEO99] diferem basicamente quanto ao esquema de compensação utilizado pelo escalonador.

Lin et al. em [LIN00] defendem que o principal objectivo de um algoritmo de escalonamento não deve ser a maximização do débito agregado na rede, mas sim proporcionar um serviço mais flexível aos utilizadores, assegurando um determinado nível de qualidade de serviço a certos fluxos independentemente do estado do canal. Sendo assim, os autores propõem um algoritmo de escalonamento para redes sem fios que procura assegurar justiça no curto prazo ao mesmo tempo que mantém um débito aceitável no canal. Os fluxos que enfrentam condições adversas no canal não são imediatamente suspensos, mas, pelo contrário, recebem uma largura de banda adicional para compensação dos erros logo que estes ocorrem, dentro de certos limites e sujeitos a uma taxação suplementar acordada durante o estabelecimento da conexão.

Adoptando uma abordagem semelhante, Eckhardt e Steenkiste propõem um algoritmo de escalonamento [ECK00] para uso em redes sem fios baseado no algoritmo WFQ, que realiza dinamicamente pequenos ajustes nos pesos dos fluxos

para compensar a ocorrência de erros. Os autores fazem a distinção entre esforço, definido como a parcela de tempo do canal ocupado por um dado fluxo, e resultado, definido como sendo o débito efectivo alcançado pelo fluxo. O método apresentado propõe o aumento do esforço gasto com os fluxos que apresentam erros, dentro de certos limites, para que os mesmos possam alcançar o resultado desejado.

Nos artigos anteriormente discutidos, aborda-se essencialmente o escalonamento de tráfego assíncrono. Além disso, em muitos casos as características das redes sem fios são consideradas de forma abstracta ou idealizada. Nós acreditamos que a implementação prática de um algoritmo de escalonamento, nomeadamente tendo em vista o transporte de tráfego com restrições temporais, deve ter em consideração as particularidades da rede sem fios utilizada, incluindo as funções de controlo de acesso ao meio (MAC) e de controlo de erros (EC).

Os artigos descritos a seguir consideram a transmissão de tráfego de voz em redes IEEE 802.11 utilizando a função de coordenação pontual (PCF). Em todos os casos, o tráfego de voz é modelado por um processo On/Off que representa períodos alternados de conversação e de silêncio exponencialmente distribuídos, embora a duração média de cada um dos períodos varie conforme o artigo.

Visser e Zarki em [VIS95] analisam a multiplexagem estatística do tráfego de voz, considerando o descarte da informação de voz que tenha excedido o prazo de validade à altura da interrogação. Os autores concluem, com base nos resultados obtidos, que o *overhead* elevado introduzido pelas especificações do IEEE 802.11 resulta num número reduzido de conversações possíveis, para o que também contribui o débito de transmissão utilizado na análise (apenas 1 Mbit/s). Nas simulações efectuadas é considerado um modelo de canal de transmissão isento de erros.

Veeraraghavan et al. em [VEE01] consideram a transmissão de tráfego de voz entre dois terminais numa mesma célula com a intermediação do ponto de acesso, demonstrando que o aumento do período entre interrogações permite que mais conexões de voz sejam acomodadas, ao custo do aumento do atraso. Numa análise dos efeitos dos erros no canal sobre a taxa de erros de pacotes, os autores concluem da necessidade de utilização de um mecanismo de controlo de erros na transmissão do

tráfego de voz, ressaltando que as implicações da utilização de retransmissão na recuperação de erros para o tráfego de tempo real necessitam de ser estudadas.

Coutras et al. em [COU00] apresentam uma proposta para o suporte de serviços de tempo real na rede IEEE 802.11 utilizando a função PCF e discutem os detalhes do procedimento de estabelecimento de conexão. Para a satisfação dos requerimentos do tráfego de tempo real, representado pelo tráfego de voz, os autores propõem que cada estação seja interrogada uma única vez a cada supertrama, e que requisições de conexão feitas palas estações, caso sejam aceites, sejam colocadas no final da lista de *polling*. Para evitar que o instante de serviço de uma dada estação *i* numa supertrama seja antecipado quando as estações interrogadas antes da estação *i* não utilizam inteiramente o período alocado para as mesmas, os autores propõem que o ponto de acesso interrogue uma estação assíncrona, normalmente servida durante o período de contenção, até que o próximo instante de interrogação situe-se dentro da janela de serviço da estação *i*. Os autores incorporam na sua análise a variação do atraso causada pelo alargamento do período de contenção (*stretching*), mas não consideram a ocorrência de erros no canal.

Em [CRO97], Crow et al. analisam o desempenho da rede IEEE 802.11 com uma combinação de tráfego de voz e de dados, por meio de simulação. O tráfego de voz é transmitido com a função PCF e o tráfego de dados é transmitido com a função DCF, sendo o débito de transmissão utilizado de 1 Mbit/s. O traçado do complemento da função cumulativa de distribuição do atraso permite determinar a percentagem de pacotes de voz descartados por não terem sido transmitidos dentro dos limites de atraso estipulados, mas o *jitter* das conexões não é analisado. Um modelo de erros de dois estados é utilizado para caracterizar o desvanecimento multipercurso no canal de transmissão, porém, a retransmissão dos pacotes de voz corrompidos por erros no canal não é considerada.

A duração da supertrama utilizada pelos autores é considerável (410 ms) pelo que as estações que transmitem o tráfego de voz são interrogadas várias vezes em cada supertrama. No algoritmo proposto pelos autores, uma estação é removida da lista de *polling* na supertrama após *k* respostas nulas consecutivas resultantes de um período de inactividade, voltando à lista na supertrama seguinte. O objectivo é o aproveitamento da largura de banda que sobra para a transmissão do tráfego de outras

estações. Os resultados mostram que o valor de *k* utilizado tem um impacto reduzido sobre o atraso das conexões de voz, devido à operação em modo On/Off, pois é provável que a inactividade detectada da primeira vez prolongue-se num futuro próximo. Assim, os autores concluem que neste caso a melhor política, numa perspectiva de aumento do débito agregado da rede, é a remoção das estações da lista de *polling* à primeira resposta nula ($k = 1$).

Os trabalhos apresentados acima analisam a transmissão de tráfego de tempo real (voz) nas redes IEEE 802.11, mas desprezam os erros no canal ou não consideram a retransmissão dos pacotes de voz corrompidos por erros, fiando-se na tolerância a erros deste tipo de tráfego. Entretanto, o tráfego gerado por sistemas de aquisição de dados e controlo normalmente possui requisitos muito mais estritos que o tráfego de voz em relação à taxa de perda de pacotes, pelo que a recuperação dos dados corrompidos é ainda mais importante.

Os artigos descritos a seguir analisam o desempenho da rede HIPERLAN/2 e, nalguns casos, também da rede IEEE 802.11, ao nível da camada física e/ou da camada de ligação de dados.

Em [LI00b], Li et al. apresentam uma análise comparativa de desempenho, ao nível da camada de ligação de dados, entre as redes IEEE 802.11a e HIPERLAN/2. O artigo compara o débito fornecido pelas duas normas numa célula com uma única conexão entre um ponto de acesso e um terminal. No caso da rede IEEE 802.11, somente a função DCF é analisada. Os resultados são baseados em modelos analíticos simplificados, por exemplo, no caso da rede HIPERLAN/2, assume-se que 3 canais SCH são reservados em todas as tramas para a conexão: um para as mensagens de ARQ no sentido descendente; um para as mensagens de ARQ no sentido ascendente; e um para as mensagens de requisição de recursos (RR) no sentido ascendente.

Kadelka et al. em [KAD99] analisam o desempenho do protocolo de controlo de acesso ao meio do HIPERLAN/2 num cenário com múltiplos terminais móveis, utilizando um modelo analítico, com o qual concluem que o desempenho do sistema decresce com o aumento do número de terminais activos numa trama. A ocorrência de erros no canal e as respectivas mensagens de controlo não são tidas em consideração. Os resultados limitam-se ao débito agregado das conexões.

Hettich e Schöther em [HET99] comparam o desempenho das redes HIPERLAN/2 e IEEE 802.11a, tanto analiticamente como por meio de simulação. Os resultados incluem tanto tráfego isócrono, representando uma conexão de voz de débito constante, quanto tráfego assíncrono. Entretanto, o artigo não analisa o desempenho das redes em condições de erros no canal. Os resultados indiciam um desempenho superior por parte da rede HIPERLAN/2.

Doufexi et al. em [DOU02] apresentam resultados comparativos entre as redes IEEE 802.11a e HIPERLAN/2. Os resultados de simulação a nível da camada física indicam que o desempenho em termos de taxa de erros binários (BER) das duas normas é semelhante, o que é esperado devido à similaridade das camadas físicas. Já a taxa de erros de pacote (PER) é superior no caso da rede IEEE 802.11a, devido ao maior comprimento dos pacotes utilizados. Os gráficos apresentados para a taxa de erros de pacote em função da relação portadora/interferência (C/I, *Carrier to Interference ratio*) para os diferentes modos de transmissão mostram que, na maior parte dos casos, a taxa de erros de pacote diminui com a utilização de um modo de transmissão inferior, se mantida a mesma relação portadora/interferência. Em [KHU99], Khun-Jush et al. também analisam o desempenho do HIPERLAN/2 a nível da camada física, obtendo resultados semelhantes.

A nível da subcamada de controlo de acesso ao meio, Doufexi et al. comparam o débito agregado de ambas as redes para os diferentes modos de transmissão utilizando um modelo analítico. No caso da rede IEEE 802.11a é analisada apenas a função DCF e com apenas um terminal, pelo que o efeito das colisões sobre o desempenho não pode ser avaliado. Mesmo assim, conclui-se que o HIPERLAN/2 alcança um débito superior ao IEEE 802.11a, nomeadamente com os modos de transmissão superiores, devido ao menor *overhead* introduzido.

Li et al. em [LI00a] descrevem o mecanismo de detecção de erros e retransmissão (ARQ) do HIPERLAN/2 e comparam, através de simulação, o seu desempenho com o de outro mecanismo que foi proposto para utilização no HIPERLAN/2, denominado PRIME ARQ. O cenário utilizado considera o transporte tráfego assíncrono (IP), servido com base na disciplina FCFS (*First Come First Served*). O *overhead* da camada física não foi incluído na simulação. Os resultados apresentados indicam que

o mecanismo adoptado pela norma proporciona um débito/eficiência superior e um atraso inferior ao PRIME ARQ.

Os próximos artigos abordam o escalonamento de tráfego com requisitos de qualidade de serviço em redes ATM sem fios.

Passas et al. em [PAS97] [PAS98] propõem um algoritmo de escalonamento, denominado PRADOS (*Prioritized Regulated Allocation Delay Oriented Scheduling*), que tem como objectivos a regulação do tráfego das conexões com base nos respectivos parâmetros de tráfego e a satisfação dos requisitos de qualidade de serviço, nomeadamente as restrições temporais, das diferentes classes de serviço definidas no ATM.

Este algoritmo foi concebido para utilização numa rede ATM sem fios desenvolvida no âmbito do projecto Magic WAND (*Wireless ATM Network Demonstrator*), em conjunto com um protocolo de controlo de acesso ao meio baseado em reserva dinâmica explícita denominado MASCARA (*Mobile Access Scheme based on Contention and Reservation for ATM*). A estrutura da trama deste protocolo é semelhante à utilizada no HIPERLAN/2, com a diferença de que a duração da trama é variável. O desempenho do algoritmo proposto é comparado com o do algoritmo FCFS, por meio de simulação. Os resultados indicam que a utilização do algoritmo PRADOS é benéfica quando o cenário é composto por conexões de classes diferentes. Entretanto, este algoritmo não tem em consideração a ocorrência de erros no canal.

Kim et al. em [KIM01] propõem um protocolo de controlo da ligação de dados (DLC) específico para o transporte de tráfego CBR em redes ATM sem fios que considera a ocorrência de erros no canal. A recuperação das células ATM corrompidas por erros no canal baseia-se na utilização de um mecanismo ARQ de repetição selectiva. O procedimento de retransmissão de uma célula ATM[33] deve completar-se dentro de um período denominado *intervalo para recuperação de erros*, especificado durante o estabelecimento de conexão, caso contrário a célula é descartada no emissor. Este protocolo incorpora também um esquema de remoção do

[33] Na verdade, cada célula ATM é transmitida encapsulada numa mensagem DLC.

jitter utilizado para reduzir a variação do atraso das células provocado pela retransmissão das células perdidas.

Este protocolo foi implementado num protótipo de rede ATM sem fios denominado WATMNet [RAY97]. O controlo de acesso ao meio do WATMNet baseia-se num protocolo de reserva dinâmica explícita, sendo o tempo dividido em tramas TDMA com duração fixa de 2 ms, tal como acontece com o HIPERLAN/2. No entanto, a estrutura das tramas é diferente; por exemplo, o WATMnet agrupa o tráfego ABR/UBR, VBR e CBR dos diferentes terminais e o transmite em regiões distintas dentro de uma mesma trama.

O protocolo DLC proposto pelos autores mantém um *buffer* de células ATM de tamanho fixo para cada conexão, tanto no emissor como no receptor. Dado o débito das células ATM (α), e o *intervalo para recuperação de erros* (d), a dimensão do *buffer* de células (B), em ambos os lados, é expressa por:

$$B = d\alpha \tag{4.1}$$

Quando uma célula chega ao emissor proveniente da camada ATM é enviada à camada MAC para que seja feita a sua transmissão regular num canal CBR. Ao mesmo tempo, uma cópia da célula é inserida no final do *buffer* do emissor, e todas as células são deslocadas de uma posição. Como a dimensão do *buffer* é fixa, isso faz com que a célula mais antiga no *buffer* seja descartada. Se as células chegam da camada ATM com intervalos constantes, todas as células permanecem o mesmo tempo no *buffer*, correspondendo ao *intervalo para recuperação de erros*.

Quando uma célula chega ao receptor, o seu número de sequência é comparado com os valores mínimo (M_{min}) e máximo (M_{max}) contidos no *buffer* do receptor. Se o número de sequência da célula recebida situa-se entre M_{min} e M_{max} (célula retransmitida), ela é inserida na posição correcta no *buffer*. Uma célula com número de sequência inferior a M_{min} é descartada, por ter passado o limite de atraso imposto. Já uma célula com número de sequência superior a M_{max} é inserida no final do *buffer*, fazendo com que as células antigas sejam deslocadas de uma posição. Nesse instante, a primeira célula é retirada do *buffer* e entregue à camada ATM do receptor.

Neste esquema, a entrega da primeira célula do *buffer* de recepção depende da chegada de uma nova célula para o fim do *buffer*, pelo que a perda desta atrasaria a

transferência da primeira para a camada ATM, aumentando o seu atraso e o *jitter* da conexão. Para evitar que isso aconteça, um *token counter* é usado no receptor. A ideia consiste em inserir uma célula vazia (*dummy*) no fim do *buffer* quando a célula esperada não chega a tempo para possibilitar a retirada da primeira. Como resultado, o fluxo de células entregue à camada ATM no receptor assemelha-se ao fluxo recebido da camada ATM no emissor.

Uma célula vazia no *buffer* do receptor indica que a célula correspondente não foi recebida ou foi recebida com erros. Para que o emissor seja informado da necessidade de retransmissão de células corrompidas, o receptor verifica periodicamente se há células vazias no seu *buffer*. Em caso positivo, ele gera uma mensagem de ARQ contendo um *bitmap* (20 bits) que indica as células em falta a partir de um número de sequência de referência. A retransmissão de células ATM das conexões CBR é feita usando canais UBR da trama.

O artigo apresenta resultados experimentais numa configuração com uma estação base, um terminal e, de início, uma única conexão CBR entre ambos. A taxa de erros binários (BER) observada foi de cerca de $3x10^{-5}$. Como esperado, a utilização de um *intervalo para recuperação de erros* maior possibilita a recuperação de um maior número de células pela camada DLC, embora resulte num atraso extremo a extremo mais elevado para a conexão.

Observou-se que o intervalo de verificação do *buffer* do receptor para geração de mensagens ARQ não tem grande impacto sobre o *overhead* introduzido pelo protocolo DLC, ao contrário do que acontece com a taxa de recuperação de erros, que diminui significativamente com o aumento deste intervalo. Com base nessas observações, os autores concluem que é melhor fazer a verificação do *buffer* do receptor tão frequentemente quanto possível.

Com o aumento do número de conexões CBR estabelecidas entre o terminal e a estação base para até cinco conexões, observou-se um decréscimo significativo na capacidade de recuperação de erros. Os autores atribuem este decréscimo no desempenho à competição entre as múltiplas conexões por largura de banda para as suas retransmissões. Para resolver o problema, sugerem o aumento da largura de banda disponível para as conexões CBR, mas a atribuição fixa de recursos extras teria

impacto sobre a eficiência da rede e a utilização de um mecanismo dinâmico não foi considerada.

Recentemente, verifica-se o aparecimento de uma série de artigos que analisam o desempenho dos mecanismos de acesso ao meio da norma IEEE 802.11e e propõem novas soluções para o transporte de diferentes tipos de tráfego em diversos cenários. A seguir são mencionados alguns destes trabalhos, sem ser-se exaustivo.

Em [GU03], Gu and Zhang descrevem o mecanismo EDCF da norma IEEE 802.11e e avaliam o seu desempenho por meio de simulações, utilizando uma configuração *ad hoc*. Por simplicidade, os autores não consideram parâmetros como a oportunidade de transmissão (TXOP), centrando-se nos parâmetros associados à função de acesso ao meio do EDCF. Os resultados apresentados mostram um atraso menor no acesso ao meio, e um débito maior, quanto mais prioritária é a categoria de acesso, pelo que se conclui que o EDCF é capaz de proporcionar diferenciação de serviços e prioridade no acesso ao meio.

Mangold et al. em [MAN03] e [MAN02] analisam os aprimoramentos no controlo de acesso ao meio introduzidos pela norma IEEE 802.11e, comparando o seu desempenho com do IEEE 802.11 original por meio de simulação. Na análise do mecanismo EDCA são utilizadas quatro classes de tráfego com prioridade decrescente (voz, vídeo, melhor esforço e *background*), porém, com os mesmos parâmetros de tráfego. Num cenário com um ponto de acesso a comunicar com 3 terminais, observou-se que com o aumento da carga na rede o débito atinge um ponto de saturação, com tráfego das classes mais prioritárias obtendo uma maior quota de largura de banda devido ao menor tempo de acesso. Aumentando-se o número de estações, observou-se que o débito de todas as classes reduz dramaticamente devido ao aumento da probabilidade de colisão.

Na avaliação de desempenho do mecanismo HCCA utilizou-se um único fluxo no sentido descendente (para além de outros fluxos transmitidos com o mecanismo EDCA), sendo as tramas deste fluxo imediatamente transmitidas decorrido um período PIFS após a detecção de inactividade no meio. Os resultados apresentados pelos autores mostram que é possível satisfazer os requisitos estritos de atraso máximo definidos pelos autores (300 μs) através da limitação da duração máxima da

oportunidade de transmissão (TXOP) dos outros fluxos. No entanto, o cenário utilizado com o mecanismo HCCA não contempla as condições mais desfavoráveis que podem advir do transporte de tráfego no sentido ascendente, da perda de pacotes, ou da multiplexagem de um número maior de fluxos.

Em [TRU03], Truong e Vannuccini analisam, por meio de simulação, o desempenho do mecanismo EDCF do IEEE 802.11e no transporte de tráfego de voz, vídeo e dados em variados cenários. O tráfego de voz é modelado por um processo On/Off, sendo que nos períodos de actividade gerados pacotes de 160 bytes a cada 20 ms. Os autores consideram um desempenho satisfatório para o transporte do tráfego de voz quando a probabilidade do atraso ser maior que 20 ms é inferior a 1 % e a probabilidade de perda de pacotes (PLR) é inferior a 3 %. Num cenário composto somente por tráfego de voz, os autores concluem que é possível suportar até 70 conexões em simultâneo utilizando-se o débito de transmissão de 24 Mbit/s. Com a adição de tráfego de vídeo e dados, o atraso das conexões de voz aumenta drasticamente, nomeadamente para os débitos de transmissão mais baixos, pelo que o número de conexões de voz suportadas diminui.

Com base nos resultados obtidos, os autores alegam que nos cenários considerados o mecanismo EDCF é capaz de satisfazer os requisitos de qualidade de serviço de aplicações de tempo real e interactivas, concluindo que não há necessidade de utilizar um protocolo de controlo de acesso ao meio mais complexo, como é o caso do mecanismo HCCA. Tendo isso em conta, decidiu-se considerar nesta tese o transporte de tráfego de tempo real utilizando o mecanismo ECDF. Os resultados obtidos com este mecanismo distribuído, bem como a discussão e comparação com o desempenho obtido com os mecanismos centralizados das redes analisadas, são apresentados no capítulo 6.

4.2 Algoritmos de escalonamento propostos

4.2.1 IEEE 802.11

A função de coordenação pontual (PCF) da rede IEEE 802.11 opera com base num protocolo de *polling*. Um modo simples de gerir o acesso ao meio por parte dos terminais com um protocolo deste tipo consiste em interrogar ciclicamente e sem interrupções os terminais presentes na lista de *polling*. No entanto, esta abordagem apresenta diversos problemas, destacando-se os seguintes.

i. O intervalo entre as interrogações a um dado terminal depende do número de terminais na lista de *polling*, bem como da intensidade do tráfego na rede. Sendo assim, o atraso e o *jitter* de uma conexão sofrem uma influência significativa do tráfego das demais conexões.

ii. Quando um terminal é interrogado e não tem dados a transmitir, envia uma mensagem nula. Se esta situação for frequente, o *overhead* do protocolo será elevado, pelo que a sua eficiência será baixa. Além disso a transmissão de respostas nulas provoca um acréscimo indesejado no consumo de energia dos terminais. Se o intervalo de geração dos pacotes no terminal for superior ao intervalo entre as interrogações, a frequência de respostas nulas será elevada.

iii. A utilização de um mecanismo de *polling* sem interrupções não deixa espaço para um período de contenção, necessário para a transmissão de algumas mensagens de gestão. Estas mensagens permitem, por exemplo, a associação de novos terminais à rede ou a sua inclusão na lista de *polling*.

iv. O tráfego assíncrono, que não possui restrições temporais, pode ser transmitido mais eficientemente com um protocolo de acesso aleatório do que com um protocolo de *polling*.

Na abordagem utilizada pelo IEEE 802.11, divide-se o tempo em supertramas compostas por um período livre de contenção (CFP), em que é utilizado um protocolo de *polling* (PCF), e por um período de contenção (CP), em que é usado um protocolo de acesso aleatório (DCF). Isso permite resolver os problemas *iii* e *iv* levantados

acima. No entanto, a solução de outros problemas, depende do modo como o escalonamento do tráfego é realizado.

Caso os terminais sejam interrogados várias vezes a cada supertrama, como é feito em [CRO97], os problemas *i* e *ii* permanecem. Além disso, a pausa nas interrogações durante o período de contenção (CP) tende a aumentar o *jitter*, devido ao acumulo de pacotes nos terminais. Por outro lado, se os terminais forem interrogados apenas uma vez a cada supertrama, o intervalo entre as interrogações será aproximadamente periódico. Isso permite reduzir o *jitter* e o número de respostas nulas dos terminais, desde que a duração da supertrama seja adequada.

A seguir são propostos quatro algoritmos de escalonamento para o transporte de tráfego de tempo real em redes IEEE 802.11. O objectivo destes algoritmos é permitir a retransmissão rápida dos dados corrompidos devido a erros no canal, para que esses dados possam chegar a tempo ao seu destino. Na ausência de erros, as seguintes regras aplicam-se a todos os algoritmos.

- Os terminais que constam da lista de *polling* são interrogados sequencialmente.
- Cada terminal é interrogado uma única vez em cada supertrama.

Nos dois primeiros algoritmos, a retransmissão dos dados corrompidos é feita na supertrama seguinte, enquanto nos outros dois a retransmissão é feita na mesma supertrama.

4.2.1.1 Algoritmo A

Com este algoritmo, a retransmissão dos dados corrompidos é feita na supertrama seguinte, na posição da sequência associada ao terminal, pela retransmissão da trama que contém os dados corrompidos.

A Figura 4.1 apresenta um exemplo do funcionamento deste algoritmo, com destaque para dois terminais (1 e 2, neste caso). Na ausência de erros, o terminal responde a uma interrogação com o envio dos dados recebidos da camada superior desde a transmissão anterior (como é exemplificado para o terminal 2 em todas as supertramas). Porém, se a trama transmitida na supertrama anterior foi corrompida

(terminal 1, supertrama *i*-1), ela é retransmitida na supertrama corrente. Com isso, os dados que eram supostos serem transmitidos nesta supertrama só serão transmitidos na próxima, juntamente com dados associados àquela supertrama.

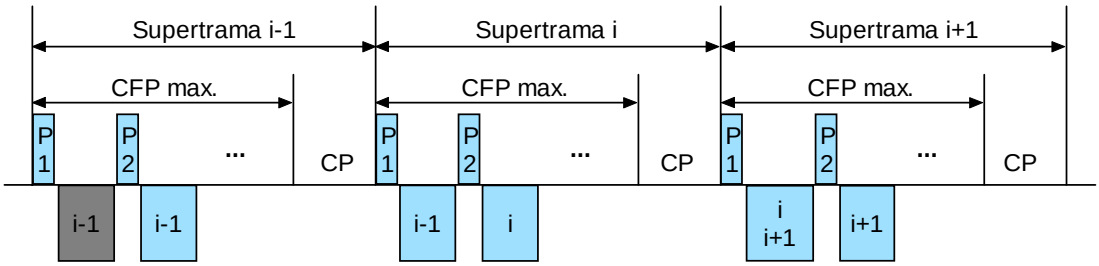


Figura 4.1: Exemplo de funcionamento do algoritmo A.

Caso a retransmissão da trama corrompida não tenha sucesso, uma nova tentativa de retransmissão é feita na supertrama seguinte. Com isso, a transmissão dos novos dados tem que ser adiada mais uma vez.

4.2.1.2 Algoritmo B

Tal como ocorre com o algoritmo A, com este algoritmo a retransmissão dos dados corrompidos é feita na supertrama seguinte, na mesma posição da sequência. A diferença é que, em vez de fazer-se a retransmissão da trama corrompida, transmite-se uma nova trama contendo os dados corrompidos junto com os novos dados. O objectivo é evitar o adiamento da transmissão dos novos dados quando os dados corrompidos são retransmitidos.

A Figura 4.2 apresenta um exemplo do funcionamento deste algoritmo. Quando a trama transmitida na supertrama anterior é corrompida (terminal 1, supertrama *i*-1), a trama da supertrama corrente contém não só os dados corrompidos mas também os novos dados. Caso a transmissão seja bem sucedida desta vez, na supertrama seguinte a situação volta ao normal.

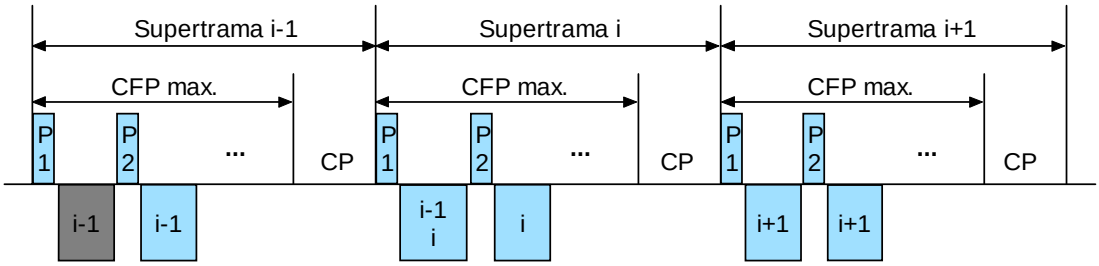


Figura 4.2: Exemplo de funcionamento do algoritmo B.

4.2.1.3 Algoritmo C

Com este algoritmo, as retransmissões são feitas na mesma supertrama, após o ciclo normal de interrogação dos terminais que fazem parte de lista de *polling*. Quando o ponto de acesso recebe uma trama corrompida de um terminal, ele põe o endereço do terminal no final de uma fila FIFO, denominada fila de retransmissões. Depois de todos terminais da lista de *polling* terem sido interrogados, o ponto de acesso começa a interrogar os terminais da fila de retransmissões, que em resposta retransmitem as respectivas tramas corrompidas.

A Figura 4.3 apresenta um exemplo da utilização deste algoritmo, no qual as tramas transmitidas pelos terminais 1 e 3 durante o ciclo normal de interrogação são corrompidas por erros. Depois de interrogar o último terminal da lista de *polling* (terminal *n*), o ponto de acesso volta a interrogar o terminal 1, mas a sua trama é corrompida pela segunda vez, pelo que o seu endereço é colocado novamente na fila de retransmissões. A seguir, o ponto de acesso interroga com sucesso o terminal 3 na segunda tentativa e o terminal 1 na terceira tentativa. Com isso, a fila de retransmissões fica vazia, pelo que o ponto de acesso transmite a trama CF-End para encerrar o período livre de contenção (CFP).

Caso a duração máxima do CFP seja atingida durante a fase de retransmissões, a fila de retransmissões é esvaziada. Os dados corrompidos referentes aos terminais que estavam nesta fila têm então que ser transmitidos na supertrama seguinte, durante o ciclo normal de interrogações, da mesma forma que no algoritmo A.

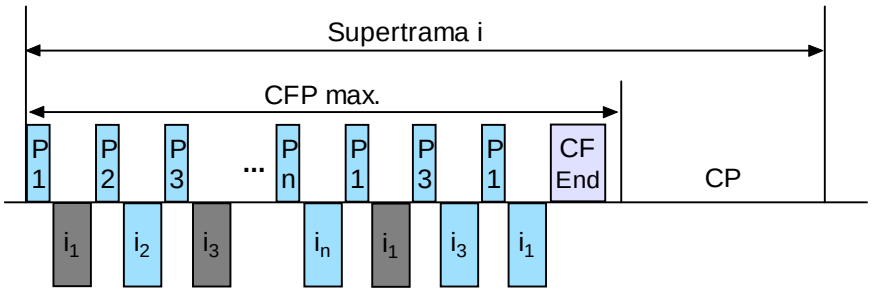


Figura 4.3: Exemplo de funcionamento do algoritmo C.

4.2.1.4 Algoritmo D

Neste caso, quando a trama proveniente de um terminal é recebida com erros pelo ponto de acesso, ele repete a interrogação ao terminal até que a mensagem seja recebida sem erros (ou o limite máximo de retransmissões especificado seja atingido). A Figura 4.4 apresenta um exemplo do funcionamento deste algoritmo.

Com este algoritmo, os primeiros terminais da lista de *polling* são beneficiados, em detrimento dos últimos terminais da lista, cujas transmissões na supertrama podem mesmo ter que ser canceladas por falta de espaço no período livre de contenção (CFP).

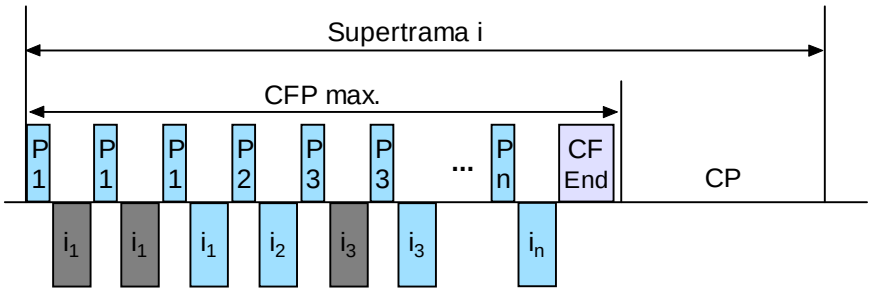


Figura 4.4: Exemplo de funcionamento do algoritmo D.

4.2.1.5 Conclusões

Nos dois primeiros algoritmos a atraso e o *jitter* das conexões tende a ser maior do que nos outros dois, porque os terminais só têm oportunidade de retransmitir os

dados corrompidos na supertrama seguinte. Por outro lado, o *overhead* associado às retransmissões tende a ser menor, porque o ponto de acesso não precisa fazer mais do que uma interrogação por terminal por supertrama.

Os algoritmos A, B e C foram implementados no simulador. A secção 6.2.3 compara o desempenho destes algoritmos com base nos resultados obtidos por simulação. O algoritmo D não foi implementado, por não satisfazer o critério de justiça no tratamento das conexões.

4.2.1.6 Outras considerações

O ponto de acesso deve implementar um mecanismo de controlo de admissão de conexões (CAC, *Connection Admission Control*) para garantir que todas as conexões CBR admitidas na lista de *polling* possam fazer pelo menos uma transmissão por supertrama em condições normais. No entanto, condições excepcionais, provocadas por exemplo pelo excesso de fluxo devido a erros no canal ou pelo efeito do alargamento do período de contenção (Figura 3.18), podem fazer com que o período livre de contenção esgote-se antes que a lista de *polling* tenha sido totalmente percorrida. Duas alternativas são consideradas para lidar com estes casos.

1. Iniciar a interrogação na supertrama seguinte a partir do primeiro terminal da lista de *polling*.
2. Retomar a interrogação na supertrama seguinte a partir do terminal onde ocorreu a interrupção, continuando até que todos os terminais sejam interrogados.

Na primeira alternativa, os últimos terminais da lista de *polling* obtêm uma qualidade de serviço inferior, pois têm menos oportunidades de transmissão do que os primeiros terminais. Já na segunda alternativa, a qualidade de serviço oferecida a todas as conexões é semelhante, porém o *jitter* total tende a aumentar, porque as conexões que não foram atendidas numa supertrama são servidas à frente das outras na supertrama seguinte.

4.2.2 HIPERLAN/2

4.2.2.1 Retransmissão rápida

Nesta secção é proposto um algoritmo de escalonamento vocacionado para o transporte de tráfego de tempo real no HIPERLAN/2 cujo objectivo é possibilitar a retransmissão rápida dos dados corrompidos por erros no canal, para que estes não ultrapassem o limite de atraso requerido pela aplicação. Inicialmente descreve-se o processo normal de retransmissão no HIPERLAN/2 para depois ser apresentado o mecanismo de retransmissão rápida proposto.

A Figura 4.5 representa uma sequência típica de transmissão de mensagens de dados e controlo no HIPERLAN/2. Este exemplo concentra-se nas mensagens associadas a uma conexão específica, omitindo as mensagens associadas às demais conexões. Na trama *x*, o ponto de acesso insere uma mensagem de atribuição de recursos (RG) no canal FCH, para anunciar a atribuição de três canais LCH[34] na fase ascendente (UL) destinados à transmissão de mensagens de dados associadas à conexão. As mensagens com os números de sequência *i* e *i*+2 são corrompidas, pelo que necessitam de ser retransmitidas.

Na trama *y*, o ponto de acesso envia uma mensagem de ARQ para o terminal, durante a fase descendente (DL), num canal SCH, sinalizada pela respectiva mensagem de atribuição de recursos (RG) no canal FCH. O mapa de bits da mensagem de ARQ faz o reconhecimento positivo da mensagem de dados com número de sequência *i*+1 e o reconhecimento negativo das mensagens *i* e *i*+2, permitindo concluir que estas não foram recebidas correctamente.

Neste exemplo, logo que tem conhecimento da necessidade de retransmissão destas mensagens, o terminal envia uma mensagem de requisição de recursos (RR) num canal RCH, durante a fase de acesso aleatório, pedindo que lhe sejam atribuídos

[34] Considera-se que os canais LCH foram requisitados previamente, seja pela utilização de mensagens de requisição de recursos (RR), durante a fase de acesso aleatório ou a fase ascendente, seja através do mecanismo de negociação de capacidade fixa (FCA).

dois canais LCH. Porém, durante a fase de acesso aleatório os terminais competem pelo acesso ao meio, pelo que existe a possibilidade ocorrência de colisões, o que obriga a novas tentativas nas tramas seguintes. Com isso, até mesmo o tráfego de alta prioridade pode ver adiada a requisição dos recursos necessários para a retransmissão dos dados corrompidos. A probabilidade de colisão dependerá da carga na rede.

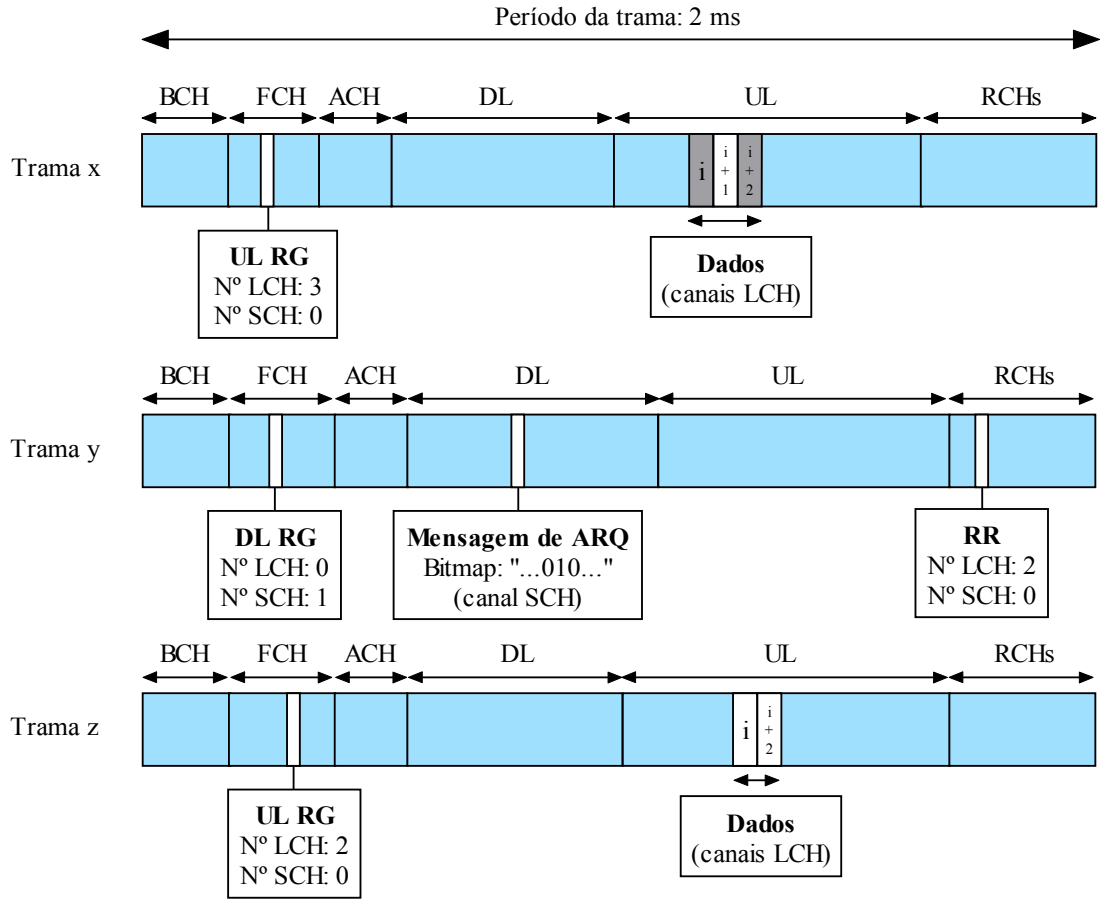


Figura 4.5: Processo normal de retransmissão de mensagens de dados no HIPERLAN/2

Após a recepção efectiva, na trama *z,* da mensagem de requisição de recursos (RR), o ponto de acesso anuncia, no canal FCH da trama, a atribuição de dois canais LCH durante a fase ascendente, usados para a retransmissão das mensagens de dados corrompidas. Este exemplo considera que as mensagens de dados são retransmitidas com sucesso. Caso isso não aconteça, o processo tem que ser repetido.

O atraso na retransmissão das mensagens depende da demora no envio da mensagem de ARQ pelo ponto de acesso após o recebimento das mensagens de dados

corrompidas, bem como do número de colisões sofridas pela mensagem de requisição de recursos (RR). No melhor dos casos, $y = x+1$ e $z = y+1$, o que significa que cada retransmissão de uma dada mensagem é feita duas tramas após a (re)transmissão anterior.

A Figura 4.6 apresenta um exemplo de funcionamento do algoritmo de escalonamento proposto para a retransmissão rápida dos dados corrompidos. Na trama *x*, tal como no exemplo anterior, o ponto de acesso anuncia, no canal FCH, a atribuição de três canais LCH para a conexão. Na mesma trama, as mensagens com os números de sequência *i* e *i*+2, transmitidas pelo terminal durante a fase ascendente (UL), são recebidas com erros pelo ponto de acesso.

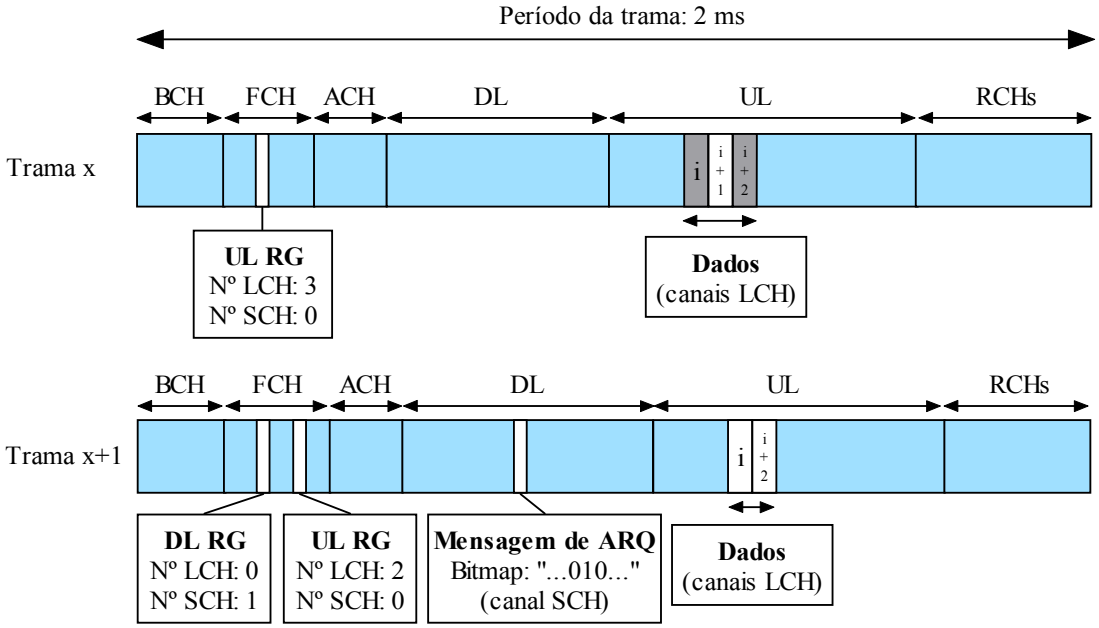


Figura 4.6: Retransmissão rápida de mensagens de dados no HIPERLAN/2

De acordo com o algoritmo proposto, quando o ponto de acesso não recebe uma mensagem válida num canal LCH atribuído para uma conexão, ele escalona imediatamente o envio de uma mensagem de ARQ na trama seguinte. Ao mesmo tempo, o ponto de acesso reserva automaticamente recursos extras para a retransmissão dos dados corrompidos, dispensando o envio de uma mensagem de requisição de recursos por parte do terminal.

Com isso, na trama *x*+1, o ponto de acesso transmite uma mensagem de ARQ num canal SCH durante a fase descendente (DL), bem como a respectiva mensagem de atribuição de recursos (RG), no canal FCH. A mensagem de ARQ permite ao terminal identificar as mensagens de dados que não foram recebidas correctamente pelo ponto de acesso. Como a fase ascendente situa-se depois da fase descendente na trama do HIPERLAN/2, o ponto de acesso pode atribuir nesta mesma trama os dois canais LCH para a retransmissão das mensagens corrompidas, pois o terminal já sabe que mensagens devem ocupar estes canais. A alocação dos canais LCH é sinalizada pela respectiva mensagem de atribuição de recursos (RG) para a fase ascendente (UL), inserida no canal FCH.

Este algoritmo garante que cada retransmissão de uma mensagem corrompida é feita logo na trama seguinte[35], enquanto que pelo processo normal, na melhor das hipóteses, o espaçamento entre as retransmissões é de duas tramas. Este algoritmo também colabora para reduzir a contenção na fase de acesso aleatório, ao dispensar a requisição de recursos para retransmissão por parte do terminal.

As mensagens de atribuição de recursos não especificam se os canais LCH disponibilizados pelo ponto de acesso destinam-se à retransmissão de mensagens corrompidas ou à transmissão de mensagens novas. No entanto, no algoritmo proposto, as mensagens que tiveram reconhecimento negativo têm precedência de transmissão sobre as novas mensagens por dois motivos: primeiro porque as mensagens corrompidas estão atrasadas em relação às demais; depois porque, senão, as novas mensagens viriam a ocupar canais destinados à retransmissão das mensagens corrompidas, sendo transmitidas antes mesmo do previsto.

No exemplo da Figura 4.6, uma única retransmissão foi suficiente para recuperar as mensagens corrompidas. Na prática, a mensagem retransmitida também pode ser corrompida, sendo então necessário realizar mais do que uma retransmissão para que a mensagem seja recebida sem erros. Assim, se no exemplo dado a mensagem *i* fosse corrompida na trama *x*+1, teria que ser retransmitida uma segunda vez na trama *x*+2, e assim por diante.

[35] A não ser que não haja disponibilidade de recursos na trama.

O algoritmo proposto faz com que as retransmissões de uma mesma mensagem ocorram em tramas consecutivas, pelo que o aumento do atraso sofrido pela mensagem é aproximadamente igual[36] ao número de retransmissões necessário para que ela seja recebida correctamente multiplicado pela duração da trama do HIPERLAN/2 (2 ms). A probabilidade de que uma mensagem precise de um certo número de retransmissões (r) para ser recebida correctamente depende da taxa de erros de pacote (PER, *Packet Error Rate*), sendo obtida pela expressão[37]:

$$P(R = r) = PER^{r}(1 - PER) \tag{4.2}$$

A taxa de erros de pacote pode ser calculada em função da taxa de erros binários (BER), ao nível da camada de ligação de dados, e do comprimento do canal de transporte LCH (54 octetos), de acordo com a equação:

$$PER = 1 - (1 - BER)^{54x8} \tag{4.3}$$

Para determinar a probabilidade de que o aumento do atraso provocado pelas retransmissões ultrapasse um dado limite, verifica-se o número mínimo de retransmissões (n) necessário para que isso ocorra e calcula-se o somatório das probabilidades de retransmissão desde n até infinito:

$$P(R \geq n) = \sum_{r=n}^{\infty} P(R = r) \tag{4.4}$$

Substituindo a equação 4.2 na equação 4.4 e calculando o somatório obtém-se:

$$P(R \geq n) = PER^{n} \tag{4.5}$$

Como as mensagens recebidas fora de sequência devem esperar que as mensagens anteriores sejam recebidas correctamente antes de serem enviadas para a camada superior, os resultados analíticos provenientes da equação 4.5 são aproximados, correspondendo a uma simplificação do que acontece no sistema real. No modelo de simulação desenvolvido, esta simplificação não é feita.

[36] O valor exacto depende da posição em que a mensagem é escalonada na trama.

[37] Assumindo uma taxa de erros constante.

O aumento da utilização do canal, no que concerne à retransmissão das mensagens de dados corrompidas, pode ser derivado da seguinte expressão:

$$U_R = \left(\sum_{r=0}^{\infty} (r+1) \cdot P(R=r) \right) \cdot U_0 \tag{4.6}$$

onde U_R é a utilização do canal incluindo as retransmissões e U_0 é a utilização sem retransmissões, equivalendo ao cenário sem erros no canal. Substituindo a equação 4.2 na equação 4.6 e simplificando obtém-se:

$$U_R = \frac{1}{1 - PER} \cdot U_0 \tag{4.7}$$

Esta equação também pode ser utilizada com o IEEE 802.11 nos casos em que há retransmissão mas o tamanho da mensagem de dados não se altera, nomeadamente no transporte de pacotes de comprimento fixo com o uso do algoritmo C apresentado na secção 4.2.1.3.

4.2.2.2 Adaptação do débito de transmissão

O HIPERLAN/2 permite seleccionar, de um conjunto de sete opções, o débito de transmissão a ser utilizado para a transmissão de uma mensagem de dados. Os débitos disponíveis (associados aos respectivos modos de transmissão) resultam de combinações de diferentes técnicas de modulação e taxas de codificação FEC, como foi mostrado na Tabela 3.3. Quanto maior o débito de transmissão utilizado, menor é o tempo necessário à transmissão de uma mensagem, pelo que mais mensagens podem ser transmitidas num mesmo período. Porém, a taxa de erros de pacote (PER) tende a aumentar com o aumento do débito de transmissão [KAD99] [KHU99], ou seja, a utilização de um débito mais elevado aumenta a probabilidade de que a mensagem seja corrompida, e vice-versa.

A retransmissão de mensagens corrompidas aumenta o atraso e a ocupação de largura de banda, pelo que é desejável que o número de retransmissões seja baixo. Assim, a utilização de um modo de transmissão mais robusto quando o estado do canal é mau pode ser compensatória. Por outro lado, quando o estado do canal é bom,

a transmissão a um débito baixo já não se justifica, pelo que a utilização de um débito mais elevado tende a ser mais eficiente.

A seguir, propõe-se um esquema de adaptação do débito de transmissão cujo objectivo consiste em limitar o impacto das retransmissões sobre a variação do atraso sem comprometer a eficiência. Segundo este esquema, as transmissões de dados de uma dada conexão são feitas ao débito normal especificado para a mesma (superior ao débito mínimo). Porém, quando uma mensagem é corrompida por erros no canal, o escalonador selecciona o débito mínimo para a primeira retransmissão da mensagem, bem como para as retransmissões subsequentes, se necessário.

Esta regra admite uma excepção: quando a retransmissão de uma mensagem ocorre na mesma trama em que é feita a transmissão de uma nova mensagem da mesma conexão no mesmo sentido, ambas as mensagens são transmitidas utilizando o mesmo débito, pois a mensagem de atribuição de recursos (RG) do HIPERLAN/2 só permite especificar um único modo de transmissão para todas as mensagens de dados da conexão na trama. Neste caso, há que optar por utilizar o débito normal ou o débito mínimo.

A ideia por trás deste esquema é que os dados contidos nas mensagens que têm que ser retransmitidas estão atrasados em relação aos demais, pelo que devem receber um tratamento especial por parte da rede. Assim, os dados são retransmitidos utilizando um modo de transmissão mais robusto, de forma a aumentar a probabilidade dos mesmos serem recebidos com sucesso. A probabilidade de uma transmissão ser corrompida é maior quando o estado do canal é mau, pelo que, se considerarmos que é provável que o estado mau mantenha-se até o momento da retransmissão, a utilização de um modo mais robusto justifica-se. Este esquema foi implementado no simulador, sendo os resultados obtidos com o mesmo apresentados na secção 6.3.3. Um esquema semelhante poderia ser aplicado à rede IEEE 802.11.

4.2.2.3 Alocação flexível de capacidade fixa

A transmissão do tráfego de uma conexão de débito constante (CBR) requer a atribuição de uma capacidade fixa por parte da rede. Ao permitir que a atribuição de

canais LCH para a transmissão dos dados seja feita de forma periódica e automática, o mecanismo de negociação de capacidade fixa (FCA) do HIPERLAN/2 mostra-se apropriado para lidar com este tipo de tráfego, pois evita que o terminal tenha que enviar continuamente mensagens de requisição de recursos (RR) para o ponto de acesso ao longo da conexão, à medida que os dados são gerados.

Para minimizar o *jitter* das conexões ATM CBR, o ideal seria que o escalonador alocasse os canais LCH com uma periodicidade igual ao intervalo de geração de células. No entanto, o mecanismo de negociação de capacidade fixa do HIPERLAN/2 aloca os canais LCH utilizados para a transmissão de dados de forma rígida, com uma periodicidade que é múltipla do período da trama MAC (2 ms). Na transmissão de tráfego ATM CBR, se o intervalo de geração de células não for múltiplo de 2 ms, não é possível conseguir este mesmo intervalo na atribuição dos canais LCH.

Neste caso, pode-se atribuir os canais com uma periodicidade inferior[38] a este intervalo. Porém, isso resulta num *jitter* elevado para a conexão, como é exemplificado na secção 6.3.5, e obriga à reserva de uma capacidade superior à utilizada. Para solucionar estes problemas, propõe-se um mecanismo de alocação flexível de capacidade fixa, descrito a seguir.

Quando o intervalo de geração de células (*IATime*) não é múltiplo de 2 ms, não é possível alocar os canais LCH com esta periodicidade, pois em algumas tramas do HIPERLAN/2 a posição ideal para a alocação cairia sobre a fase de difusão ou sobre a fase de acesso aleatório, em vez de situar-se sobre a fase ascendente. Sendo assim, no mecanismo de alocação flexível de capacidade fixa proposto, a atribuição dos canais LCH é feita de forma aproximada, tendo em conta a periodicidade ideal (*FCInterval*). Neste mecanismo, a decisão de alocação de um canal LCH numa determinada trama é feita com base no algoritmo representado na Figura 4.7, que é executado pelo escalonador no início de cada trama.

[38] A atribuição com uma periodicidade superior não seria suficiente para satisfazer os requisitos de largura de banda da conexão.

```
se (simTime() > FCPointer)
{
   Aloca canais LCH
   FCPointer <-- FCPointer + FCInterval
}
```

Figura 4.7: Pseudo-código do mecanismo de alocação flexível de capacidade fixa.

No pseudo-código apresentado, simTime() é uma função que retorna o tempo corrente e *FCPointer* é uma variável definida pelo escalonador para cada conexão que utilize este mecanismo, cujo valor corresponde à posição de alocação dos canais LCH utilizando a periodicidade ideal. *FCPointer* é inicializado com o instante de tempo correspondente ao início da trama em que é feita a primeira alocação.

Para minimizar o *jitter*, faz-se:

$$FCInterval = IATime \tag{4.8}$$

Neste caso, um único canal LCH é alocado de cada vez.

Após a primeira alocação, a variável *FCPointer* é incrementada de *FCInterval*. A alocação seguinte ocorre quando a desigualdade (simTime() > *FCPointer*) volta a verificar-se. Com este algoritmo, o mecanismo de alocação flexível de capacidade fixa adia sempre a alocação para a trama seguinte àquela na qual a variável *FCPointer* está posicionada, visto que a desigualdade é verificada no início da trama. Desta forma, assegura-se que o canal LCH não é alocado para a conexão antes da geração de uma nova célula ATM, pois isto resultaria na transmissão de uma mensagem vazia. O valor mínimo do atraso da conexão é obtido quando a variável *FCPointer* aponta para o fim de uma trama (dentro da fase de acesso aleatório), enquanto o valor máximo do atraso ocorre quando o *FCPointer* aponta para o início de uma trama (dentro da fase de difusão), porém, após a desigualdade ter sido verificada. Os resultados obtidos com a utilização deste algoritmo são apresentados na secção 6.3.5.

4.3 Sumário

Na primeira parte deste capítulo foram apresentados trabalhos relacionados publicados por outros autores. Grande parte dos algoritmos de escalonamento propostos na literatura para utilização em redes sem fios são voltados para o transporte de tráfego assíncrono, tendo como objectivo a justiça na partilha da largura de banda disponível entre os diferentes fluxos. Por outro lado, muitos trabalhos que consideram a transmissão de tráfego de tempo real em redes sem fios não modelam os erros no canal ou não consideram a retransmissão dos dados corrompidos.

Foram, por isso, desenvolvidas soluções que visam preencher esta lacuna. Neste sentido, foram descritos algoritmos de escalonamento concebidos para utilização nas redes IEEE 802.11 e HIPERLAN/2, que visam proporcionar a retransmissão rápida dos dados corrompidos por erros no canal. Estes algoritmos foram implementados nos modelos de simulação desenvolvidos para ambas as redes, que são apresentados no próximo capítulo.

Capítulo 5

Modelos de simulação de redes de comunicação

Este capítulo descreve os modelos de simulação das redes IEEE 802.11 e HIPERLAN/2 que foram desenvolvidos. Também são abordados conceitos relacionados com a técnica de simulação, como a verificação e validação de modelos e o tratamento estatístico dos dados, sendo discutida a aplicação dos mesmos aos modelos desenvolvidos.

5.1 Arquitectura do sistema simulado

O sistema de comunicação que serve de base à simulação dos modelos desenvolvidos e para o qual as soluções propostas foram optimizadas foi inspirado na arquitectura do sistema de concentração, transmissão e comutação de dados baseado no modo assíncrono de transferência (ATM) proposta em [NEV99]. Este sistema permite interligar diversas unidades de aquisição de dados e actuação remotas a uma unidade de processamento central, seja por intermédio de sistemas de comunicação cablados ou sem fios, ao mesmo tempo que garante os requisitos mínimos de QoS das aplicações.[39] Os dados relativos a múltiplos sensores ou actuadores podem ser transmitidos separadamente ou multiplexados em células ATM, usando a camada de adaptação do tipo 2 (AAL2).

[39] Devido à capacidade do ATM para lidar com diversas classes de serviço, é possível mapear os requerimentos de qualidade de serviço das aplicações industriais no modelo de QoS do ATM.

Na configuração básica desta arquitectura, as unidades remotas são interligadas à unidade central por meio de cabos, como é exemplificado na Figura 5.1. A unidade central realiza a concentração do tráfego das unidades remotas, podendo, se necessário, proceder à sua transmissão em redes públicas de telecomunicações (nomeadamente redes RDIS ou ATM) ou em redes locais (LAN, *Local Area Network*). As ligações entre as unidades remotas e a unidade central, bem como a ligação desta à infraestrutura de transmissão e comutação utilizam interfaces normalizadas. As células ATM tanto podem ser transmitidas directamente sobre a ligação física entre as unidades como podem ser mapeadas [ITU98a] sobre tramas da hierarquia digital plesiócrona (PDH, *Plesiochronous Digital Hierarchy*) a 2048 kbit/s [ITU01] [ITU98b].

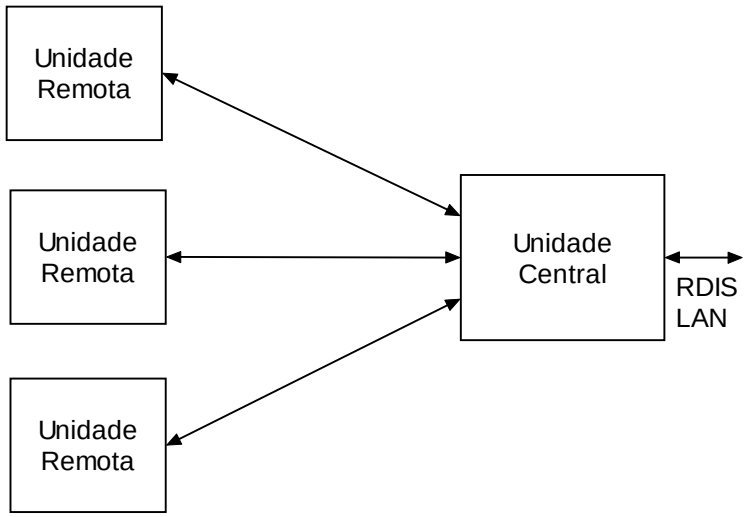


Figura 5.1: Configuração básica do sistema de aquisição de dados e controlo.

A configuração baseada em comunicação sem fios, por outro lado, poderia ser implementada recorrendo-se a sistemas de transmissão em RF, que substituiriam as ligações por cabos entre as unidades por ligações sem fios. Em teoria, a comunicação entre as diferentes unidades poderia ser efectuada através da modulação do sinal gerado em cada unidade numa portadora diferente, suficientemente espaçada das demais de modo a não haver interferência entre os sinais transmitidos. Entretanto, esta solução é claramente inviável, devido à largura de banda total exigida para a sua operação, dado que cada ligação tem uma cadência de 2 Mbit/s.

Uma alternativa consistiria na utilização de ligações direccionadas entre as unidades remotas e a unidade central, o que possibilitaria a reutilização espacial do espectro de frequências, reduzindo a largura de banda necessária para a operação do sistema. Porém, esta solução requer o alinhamento dos emissores com os respectivos receptores, comprometendo a mobilidade das unidades.

Para reduzir a largura de banda necessária e ao mesmo tempo permitir a mobilidade das unidades, a solução proposta extrai a informação útil contida nos sinais gerados nas diferentes unidades e transmite-a, utilizando uma rede sem fios. A configuração do sistema, representada na Figura 5.2, é composta pelas unidades remotas (terminais), pela unidade central (concentrador ou multiplexador), e por dois tipos de adaptadores sem fios: os adaptadores de terminal, ligados aos respectivos terminais; e o ponto de acesso, ligado ao concentrador. Os adaptadores sem fios são responsáveis pela extracção dos dados contidos no sinal gerado pela unidade à qual estão ligados e pela transmissão dos dados sobre o meio sem fios, bem como pela recepção dos dados transmitidos no sentido contrário e reconstrução do sinal original. Para isso, os adaptadores devem implementar os protocolos de comunicação da rede sem fios utilizada como suporte. Com esta configuração, os sinais enviados e recebidos pelas unidades não necessitam de ser alterados, pelo que a passagem do sistema cablado para o sistema sem fios é feita de forma transparente.

Ebert et al. propuseram em [EBE94] uma estrutura similar à apresentada na Figura 5.2 com o objectivo de proporcionar mobilidade aos equipamentos terminais da Rede Digital com Integração de Serviços (RDIS). No entanto, os autores não abordam aspectos como a rede sem fios a utilizar e o desempenho resultante deste sistema.

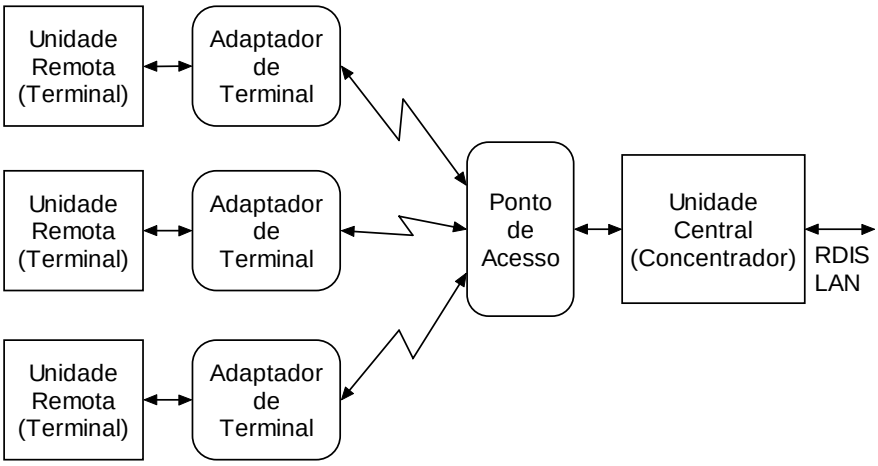


Figura 5.2: Configuração baseada em comunicação sem fios proposta para o sistema de aquisição de dados e controlo.

Com base na configuração apresentada na figura anterior, definiu-se a topologia de simulação representada na Figura 5.3, composta por um módulo concentrador (C), um módulo ponto de acesso (AP), um módulo meio sem fios (WM), e diversos módulos adaptadores de terminal sem fios (W-TA) associados aos respectivos módulos terminais (T). As funções realizadas por estes módulos são descritas na secção 5.2. O atraso de transmissão entre os terminais e os respectivos adaptadores não foi incluído no modelo. Este atraso depende do débito da ligação utilizada e tende a ser desprezável em relação ao atraso extremo a extremo.

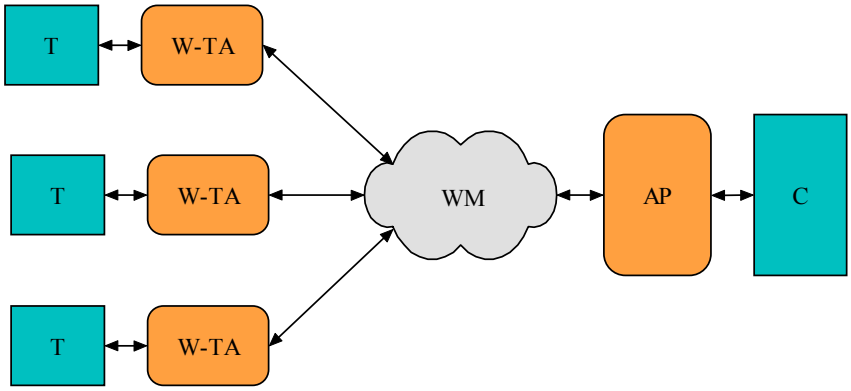


Figura 5.3: Topologia do sistema utilizado nas simulações.

5.2 Modelos de simulação de redes de comunicação sem fios

5.2.1 Introdução

As técnicas de simulação podem ser bastante úteis para analisar o desempenho de sistemas de comunicação. A simulação por eventos discretos permite que um dado sistema seja estudado mais detalhadamente do que recorrendo ao seu modelo analítico, nomeadamente quando a complexidade do sistema é elevada. O nível de detalhe do modelo analítico costuma ser limitado por pressupostos e simplificações destinados a reduzir a complexidade do modelo, que podem fazer com que o modelo obtido não seja representativo do sistema que se deseja analisar.

Esta secção descreve os modelos de simulação desenvolvidos tanto para a rede IEEE 802.11 como para a rede HIPERLAN/2. Estes modelos foram implementados utilizando a linguagem C++, com o auxílio do software de código aberto OMNET++ [VAR00]. Este software é uma ferramenta de simulação de eventos discretos orientada a objectos que proporciona diversas funcionalidades que facilitam a especificação da topologia da rede, a estruturação dos componentes do sistema em módulos e o intercâmbio de mensagens entre os módulos. O OMNET++ possui também uma biblioteca de funções úteis para o tratamento estatístico das amostras recolhidas, bem como estruturas de dados suplementares.

No simulador desenvolvido, os diferentes componentes que formam um sistema de comunicação sem fios foram divididos em módulos, cada qual implementado numa classe do C++, uma vez que a topologia de simulação utilizada pode conter múltiplas instâncias de um mesmo módulo. As funções realizadas pelas estações terminais foram separadas em dois módulos: um módulo terminal e um módulo adaptador de terminal. O módulo adaptador implementa funções da camada física, da camada de ligação de dados e da camada de convergência, enquanto o módulo terminal implementa funções das camadas superiores. As funções associadas ao ponto de acesso, ou estação base, são realizadas por um terceiro módulo. As funções relacionadas com o meio sem fios, através do qual é feita a comunicação entre as

estações, foram implementadas num quarto módulo. Por último, foi implementado um módulo concentrador, cuja única função, no âmbito deste trabalho, consiste em descartar os pacotes gerados pelos terminais que lhe são entregues pelo ponto de acesso.

Os módulos possuem uma implementação para a rede IEEE 802.11 e outra implementação para a rede HIPERLAN/2, com excepção do módulo terminal e do módulo concentrador, cuja implementação é igual para ambas as redes. Para cada um dos módulos, a parte comum foi implementada numa classe base, enquanto as partes específicas de uma dada rede foram implementadas em subclasses derivadas das respectivas classes base.

Nas redes sem fios analisadas, a coordenação de acesso ao meio e o escalonamento de tráfego são mais problemáticos no sentido ascendente, pois no sentido descendente o ponto de acesso tem controlo sobre o meio e conhecimento directo do estado dos seus *buffers* de transmissão. Sendo assim, dá-se mais realce ao tráfego gerado pelas estações terminais e destinado ao ponto de acesso.

As seguintes simplificações foram feitas para limitar a complexidade dos modelos.

- O tempo de propagação dos pacotes entre as estações da rede não foi modelado, pois a sua influência sobre o desempenho do sistema é desprezável para as distâncias envolvidas.
- Os erros introduzidos no canal incidem apenas sobre as mensagens de dados, não sendo considerada a sua influência sobre as mensagens de controlo.

5.2.2 Entidades de tráfego

Na descrição dos modelos de simulação, o termo *pacote* é usado, de modo genérico, como referência ao tráfego gerado pelos terminais, quer se trate, por exemplo, de um pacote IP, uma célula ATM ou um bloco de octetos de uma ligação RDIS. O termo *trama*, por outro lado, poderia ser usado para referenciar os pacotes transferidos ao nível da camada de controlo de acesso ao meio (MAC) de modo a distingui-los dos pacotes gerados pelos terminais. Porém, na rede HIPERLAN/2 este

termo tem outro significado, pelo que se utiliza o termo *mensagem* para referenciar os pacotes de nível MAC.

Sendo assim, existem fundamentalmente dois tipos de entidades de tráfego no simulador desenvolvido.

- Os pacotes do utilizador, comuns tanto para a rede HIPERLAN/2 como para a rede IEEE 802.11. Estes pacotes são transferidos entre os terminais e os respectivos adaptadores e entre o ponto de acesso e o concentrador.
- As mensagens de nível MAC, específicas da rede utilizada, que circulam entre os adaptadores de terminal e o ponto de acesso passando pelo meio sem fios.

O formato dos pacotes é independente da rede sem fios utilizada. O simulador associa os seguintes parâmetros a cada pacote gerado.

- Identificador de terminal/conexão.
- Número de sequência.
- Comprimento do pacote.
- Instante de geração.

Esses parâmetros auxiliam tanto na recolha de estatísticas como na verificação e validação dos modelos de simulação, não tendo necessariamente correspondência com campos presentes no pacote real. O comprimento do pacote também é usado durante o preenchimento do *payload* das mensagens de dados enviadas pelos adaptadores. O parâmetro instante de geração é utilizado pelo simulador para calcular o atraso extremo a extremo sofrido pelo pacote.

Os parâmetros das mensagens de nível MAC variam em função da rede sem fios utilizada na simulação, bem como do tipo de mensagem da rede em questão. Entretanto, os seguintes parâmetros são associados a todas as mensagens pelo simulador.

- Tipo de mensagem.
- Comprimento da mensagem.
- Endereço origem.

- Endereço destino.

Nalguns casos, estes parâmetros têm correspondência directa com campos existentes no MPDU da rede utilizada; noutros casos são utilizados internamente pelo simulador. Os tipos de mensagem implementados no simulador são apresentados nas secções 5.2.5.1 e 5.2.6.1. Cada tipo de mensagem recebe um tratamento diferente nos adaptadores, no meio e no ponto de acesso. O comprimento da mensagem é utilizado no cálculo do tempo de transmissão da mensagem no meio e também para calcular se a mesma foi corrompida por erros. Além destes parâmetros, às mensagens de dados são associados também os parâmetros número de sequência, *payload*, e bit de erro, cuja função é descrita mais adiante.

5.2.3 Parte comum dos módulos simulados

5.2.3.1 Terminal

Os terminais são responsáveis pela geração do tráfego de utilizador inserido no sistema. O simulador permite especificar o comprimento dos pacotes, o débito médio, a distribuição dos intervalos entre chegadas e o tipo de pacote. A informação do tipo de pacote transportado é utilizada para determinar a camada de convergência (CL) que a rede sem fios irá usar, quando aplicável.

O simulador foi implementado de forma a poder lidar com duas classes de tráfego de características diferentes, denominadas classes I e II, em simultâneo. Na implementação corrente, o simulador associa a cada terminal uma única conexão (ou fluxo), de uma das duas classes. O número de terminais associados a cada classe numa simulação é configurável.

O tráfego de classe I está sempre presente na simulação, enquanto o tráfego de classe II é de utilização opcional. Quando ambos estão presentes numa simulação, a classe I corresponde ao tráfego de alta prioridade, enquanto a classe II corresponde ao tráfego de baixa prioridade. Os algoritmos de escalonamento implementados no

simulador visam dar tratamento preferencial ao tráfego de classe I[40] e ao mesmo tempo tratar com justiça as diferentes conexões associadas a uma mesma classe.

5.2.3.2 Adaptador de terminal

O adaptador de terminal proporciona uma interface entre o respectivo terminal e a rede sem fios utilizada. O adaptador encapsula os pacotes recebidos pelo terminal e os transmite através do meio com destino ao ponto de acesso. A comunicação entre os adaptadores e o ponto de acesso é sujeita às regras definidas pelo mecanismo de controlo de acesso ao meio (MAC) utilizado em cada caso.

O adaptador possui um *buffer* FIFO que armazena os pacotes que chegam do terminal enquanto aguardam pela oportunidade de transmissão. A dimensão do *buffer* de pacotes é configurável. O adaptador também possui um *buffer* de mensagens que armazena[41] uma cópia das mensagens de dados transmitidas até que as mesmas recebam o reconhecimento positivo por parte do receptor.

Dependendo do caso, o *payload* das mensagens de dados pode ser composto de três formas diferentes.

- Múltiplos pacotes.
- Um único pacote.
- Um segmento.

Quando o comprimento do pacote é inferior ao comprimento do *payload*, o simulador oferece a possibilidade de agrupamento de múltiplos pacotes no *payload* de uma mesma mensagem. Com este procedimento, consegue-se diminuir o *overhead* imposto pelo protocolo MAC utilizado, o que é particularmente crítico quando o comprimento do pacote é muito pequeno, como no caso do transporte de tráfego RDIS. Antes da transmissão de uma mensagem de dados, o adaptador preenche o seu

[40] Dentro dos limites impostos pelo mecanismo de controlo de acesso ao meio utilizado.

[41] Se o esquema de detecção de erros e retransmissão (ARQ) da rede sem fios estiver habilitado.

payload com pacotes recolhidos do *buffer* de pacotes até que este fique vazio ou que o limite máximo definido para o comprimento do *payload* seja atingido.

O sistema efectua o transporte de um único pacote a cada mensagem de dados quando é configurado para tal, bem como quando as características do tráfego e da rede sem fios utilizada impõem esta opção, como na transmissão de tráfego ATM sobre a rede HIPERLAN/2[42].

Quando o comprimento do pacote é superior ao comprimento máximo do *payload*, o pacote tem que ser dividido em segmentos. Cada segmento é transportado numa mensagem diferente. Os segmentos são reagrupados no receptor, recuperando o pacote original. Esta tarefa é realizada pela função de segmentação e reunião (SAR) da camada de convergência da rede HIPERLAN/2. Neste caso, somente após o recebimento sem erros de todos os segmentos pertencentes a um dado pacote é que o mesmo segue para as camadas superiores no receptor.

5.2.3.3 Meio sem fios

Na topologia de simulação, o módulo meio possui uma ligação bidireccional com cada adaptador de terminal e com o ponto de acesso. Como numa rede sem fios o sinal transmitido por uma estação normalmente pode ser recebido por todas as outras na mesma célula, uma das tarefas do módulo meio é fazer a difusão das mensagens provenientes de uma estação pelas saídas associadas às outras estações. Ao encaminhar uma mensagem aos destinatários, o módulo meio tem em consideração o atraso de transmissão, calculado pela divisão do comprimento da mensagem pelo correspondente débito de transmissão.

Para optimizar a simulação, o meio envia as mensagens apenas para as estações que precisam recebê-las. Com este procedimento, consegue-se reduzir significativamente o tempo de execução da simulação, nomeadamente quando o número de terminais é elevado, sem que os resultados obtidos sejam alterados.

[42] Na verdade, a mensagem de dados do HIPERLAN/2 transporta uma versão comprimida da célula ATM.

Outra função do módulo meio consiste em indicar ao receptor quando uma mensagem foi corrompida por erros no canal, simulando assim a detecção de erros no receptor pela verificação do campo CRC da mensagem. A indicação de erro é dada pela activação do parâmetro bit de erro, que é anexado à mensagem. A probabilidade de activação do bit de erro depende do comprimento da mensagem e da taxa de erros binários (BER, *Bit Error Rate*) a que está sujeita a mensagem durante a sua transmissão.

O simulador implementa dois modelos de erros no canal. No modelo mais simples, a taxa de erros binários (BER) é constante durante toda a simulação. No outro modelo (de Gilbert-Elliot), a qualidade do canal alterna entre dois estados: um estado *bom* no qual a taxa de erros binários é baixa (BER_{bom}); e um estado *mau*, que representa a ocorrência de erros de rajada, no qual a taxa de erros é elevada (BER_{mau}).

O modelo de Gilbert-Elliot utiliza uma cadeia de Markov de dois estados, representado na Figura 5.4. As probabilidades de transição de estado (p e q) são inversamente proporcionais aos tempos médios de permanência nos estados *bom* ($\overline{T}_{bom}$) e *mau* ($\overline{T}_{mau}$), respectivamente. O tempo de permanência em cada estado varia de forma aleatória, sendo obtido pela utilização da distribuição exponencial com o parâmetro $\overline{T}_{bom}$ para o estado *bom* e $\overline{T}_{mau}$ para o estado *mau*.

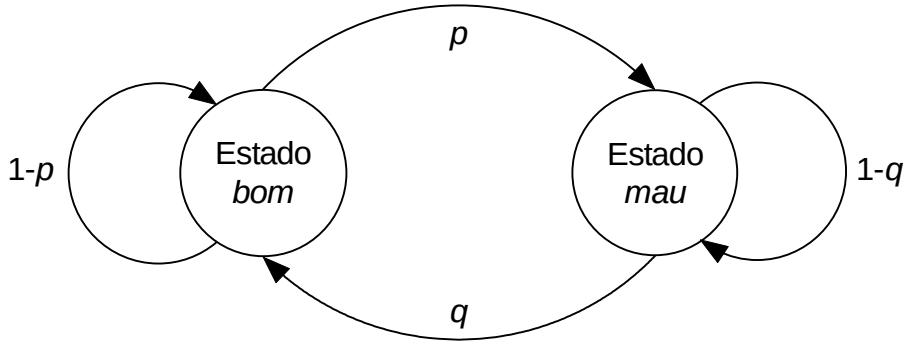


Figura 5.4: Cadeia de Markov representando o modelo de erros de Gilbert-Elliot.

A taxa de erros binários média com este modelo pode ser calculada pela seguinte expressão:

$$BER_{med} = \frac{BER_{bom} \times \overline{T}_{bom} + BER_{mau} \times \overline{T}_{mau}}{\overline{T}_{bom} + \overline{T}_{mau}} \quad (5.1)$$

No simulador desenvolvido, as diferentes ligações terminal/ponto de acesso são tratadas de forma independente, ou seja, quando uma ligação encontra-se no estado *mau*, outras ligações podem estar no estado *bom*.

O módulo meio também detecta colisões entre mensagens. Quando duas ou mais mensagens colidem não são entregues aos respectivos destinatários. Colisões ocorrem, por exemplo, durante o período de contenção (CP) na rede IEEE 802.11 e durante a fase de acesso aleatório na rede HIPERLAN/2.

5.2.3.4 Ponto de acesso

O ponto de acesso comunica com os adaptadores usando o mecanismo de controlo de acesso ao meio (MAC) da rede sem fios utilizada. Nos casos em que o controlo de acesso ao meio é centralizado, o ponto de acesso realiza as funções do controlador central (CC), sendo responsável pelo escalonamento do tráfego.

O ponto de acesso regista o atraso extremo a extremo de todos os pacotes recebidos sem erros. Este atraso diz respeito à diferença entre o instante de entrega do pacote ao concentrador e o instante de geração do pacote no terminal. As demais funções exercidas por este módulo estão intrinsecamente relacionadas com a rede modelada, sendo descritas com mais detalhes nas secções 5.2.5 e 5.2.6.

5.2.4 Recolha de estatísticas

Durante a simulação, as seguintes estatísticas são recolhidas.

- Atrasos sofridos pelos pacotes, registados tanto por conexão como por classe de tráfego.
- Número de pacotes transportados em cada mensagem de dados.
- Número de bytes gerados, transmitidos, recebidos, recebidos sem erros e descartados (na fila de espera do emissor).

- Duração da simulação.
- Utilização do canal por cada classe de tráfego, obtida pelo somatório dos períodos em que o meio fica ocupado com o tráfego das conexões de uma dada classe dividido pela duração da simulação.
- Outras estatísticas relacionadas com os parâmetros da rede sem fios utilizada, como o número de elementos de informação (IE) na fase de difusão e o número de *slots* na fase de acesso aleatório, no caso da rede HIPERLAN/2.

No final da simulação, as estatísticas recolhidas são armazenadas em ficheiros para análise posterior.

Com base na utilização do canal (U), a eficiência da rede (η) no transporte do tráfego de uma dada classe é calculada pela seguinte fórmula:

$$\eta = \frac{\frac{R_a(1-Oh_f)}{U}}{R_w}, \tag{5.2}$$

onde R_a é o débito agregado útil das conexões da classe de tráfego, R_w é o débito de transmissão da rede e Oh_f é o *overhead* fixo da rede, aplicável à rede HIPERLAN/2 (calculado pela equação 6.7).

5.2.5 Detalhes de implementação para a rede IEEE 802.11

5.2.5.1 Tipos de mensagens de nível MAC

As mensagens que passam pelo meio durante a simulação do sistema baseado na rede IEEE 802.11 são divididas em duas categorias, conforme estão associadas ao mecanismo de acesso ao meio centralizado ou distribuído.

As seguintes mensagens estão associadas à função de coordenação distribuída (DCF)[43].

- **DCF-DADOS:** Mensagem de dados do DCF. Esta mensagem transporta os dados gerados pelo utilizador, do emissor para o receptor.
- **DCF-ACK:** Mensagem de reconhecimento positivo do DCF. É enviada pelo receptor em direcção ao emissor, em resposta ao envio de uma mensagem do tipo DCF-DADOS, quando esta última não é corrompida por erros.

As mensagens associadas à função de coordenação pontual (PCF)[44] são as seguintes.

- **PCF-POLL:** Mensagem de interrogação do PCF. É enviada pelo ponto de acesso para o adaptador de terminal referenciado pelo endereço destino da mensagem.
- **PCF-RESPOSTA:** Mensagem de dados do PCF. Esta mensagem, enviada pelo adaptador para o ponto de acesso em resposta à mensagem de interrogação, transporta os dados gerados pelo terminal
- **CF-END:** Mensagem de controlo CF-End. Esta mensagem encerra o período livre de contenção (CFP). É enviada pelo ponto de acesso para os adaptadores. Também faz o reconhecimento positivo da última mensagem do tipo PCF-RESPOSTA da supertrama, quando a mesma não é corrompida por erros.

5.2.5.2 Controlo de erros

No controlo de erros da rede IEEE 802.11, o emissor só transmite uma nova mensagem de dados após a mensagem anterior ter recebido reconhecimento positivo por parte do receptor[45], seja durante o período livre de contenção (CFP) ou durante o

[43] Estas mesmas mensagens também foram utilizadas na simulação do mecanismo EDCA da norma IEEE 802.11e.

[44] Estas mensagens também foram utilizadas com o mecanismo HCCA da norma IEEE 802.11e.

[45] Esta é uma característica dos protocolos de detecção de erros e retransmissão do tipo s*top and wait*.

período de contenção (CP). Sendo assim, os adaptadores só precisam armazenar uma cópia da última mensagem transmitida, pelo que o *buffer* de mensagens dos adaptadores tem dimensão unitária.

O mecanismo de controlo de erros foi implementado de acordo com a descrição efectuada na secção 3.6.3.1, tanto para os mecanismos de acesso ao meio centralizados como para os mecanismos distribuídos.

5.2.5.3 Escalonamento de tráfego

O simulador aceita quatro opções de configuração para a transmissão do tráfego das duas classes no sistema baseado na rede IEEE 802.11.

a) Todo o tráfego é transmitido usando a função DCF.
b) Todo o tráfego é transmitido usando o mecanismo EDCA.
c) O tráfego de classe I é transmitido com a função PCF e o tráfego de classe II é transmitido com a função DCF.
d) O tráfego de classe I é transmitido com o mecanismo HCCA e o tráfego de classe II é transmitido com o mecanismo EDCA.

No primeiro caso, não é feito o escalonamento do tráfego, nem há outra forma de dar prioridade ao tráfego de uma classe sobre o tráfego da outra. Os adaptadores agem de acordo com as regras definidas pela função DCF para decidir, de forma distribuída, quando devem transmitir as mensagens de dados. Neste caso, a função do ponto de acesso consiste basicamente em enviar mensagens de ARQ em resposta às mensagens de dados recebidas, caso estas não tenham sido corrompidas por erros.

O segundo caso é semelhante ao primeiro, mas consegue-se dar prioridade ao tráfego de classe I sobre o tráfego de classe II mediante a utilização de parâmetros de *backoff* diferentes para as duas classes.

No terceiro caso, o ponto de acesso é responsável pelo escalonamento das mensagens transmitidas durante o período livre de contenção (CFP), além de definir a duração máxima deste período e a duração da supertrama. Os adaptadores que transmitem o tráfego da classe I comunicam com o ponto de acesso de acordo com as

regras da função PCF durante o período livre de contenção, enquanto os outros adaptadores utilizam a função DCF durante o período de contenção (CP).

O quarto caso é semelhante ao terceiro, mas com os mecanismos HCCA e EDCA no lugar das funções PCF e DCF, respectivamente.

5.2.6 Detalhes de implementação para a rede HIPERLAN/2

5.2.6.1 Tipos de mensagens de nível MAC

Durante a simulação do sistema baseado na rede HIPERLAN/2, as seguintes mensagens, associadas ao protocolo de controlo de acesso ao meio, circulam entre o ponto de acesso e os adaptadores de terminal.

- **H2-DIFUSÃO:** Mensagem de difusão. Resulta da junção dos canais BCH, FCH e ACH. Esta mensagem é gerada pelo ponto de acesso, sendo destinada a todos os adaptadores da célula.
- **H2-DADOS:** Mensagem de dados. Esta mensagem transporta informação contida nos pacotes gerados pelos terminais. As mensagens de dados são enviadas pelos adaptadores em canais LCH durante a fase ascendente (UL) com destino ao ponto de acesso. Caso um canal LCH seja atribuído a uma conexão e o respectivo adaptador não tenha dados a enviar, ele envia uma mensagem de dados vazia (*dummy* LCH).
- **H2-RR:** Mensagem de requisição de recursos. É enviada pelo adaptador de terminal, normalmente num canal RCH durante a fase de acesso aleatório. Também pode ser enviada num canal SCH durante a fase ascendente. Tem como destino o ponto de acesso.
- **H2-ARQ:** Mensagem de ARQ. Este tipo de mensagem está presente quando é utilizado o modo de reconhecimento (*acknowledged mode*) do HIPERLAN/2 para o controlo de erros. É enviada pelo ponto de acesso num canal SCH durante a fase descendente (DL), com destino a um dos adaptadores.

5.2.6.2 Controlo de erros

Quando o modo de reconhecimento do HIPERLAN/2 é habilitado na simulação, uma cópia de cada mensagem de dados transmitida por um adaptador é armazenada num *buffer* de mensagens até receber o reconhecimento positivo do ponto de acesso. Cada conexão requer um *buffer* de mensagens no respectivo adaptador, cuja dimensão depende do valor da janela de transmissão (*Ks*) utilizado.

Ao receber uma mensagem de dados, o receptor verifica se esta foi corrompida por erros no canal. Os pacotes contidos nas mensagens são entregues às camadas superiores na ordem em que foram gerados, o que implica que as mensagens que chegam fora de sequência são armazenadas num *buffer* de mensagens até que as mensagens anteriores sejam recebidas sem erros[46]. O ponto de acesso mantém um *buffer* de mensagens distinto para cada conexão.

O ponto de acesso mantém ainda uma lista com os números de sequência das mensagens de dados em falta numa sequência (lista NAK) para cada conexão, que serve de base para a composição das mensagens de ARQ. Esta lista também é usada para determinar se uma mensagem recebida deve ser entregue ao concentrador ou armazenada no *buffer* de mensagens. As mensagens de ARQ são enviadas com o bit CAI (*Cumulative Acknowledgement Indicator*) activo

Quando o modo sem reconhecimento (*unacknowledged mode*) do HIPERLAN/2 é utilizado, as mensagens de ARQ não são geradas, os adaptadores não armazenam as mensagens enviadas e o ponto de acesso não espera pelas mensagens anteriores para despachar os pacotes das mensagens recebidas sem erros.

5.2.6.3 Escalonamento de tráfego

O controlador central (CC), situado no ponto de acesso, é responsável pelo escalonamento dos recursos, tanto na fase descendente como na fase ascendente. O

[46] Isso implica um aumento do atraso dos pacotes transportados pelas mensagens fora de sequência.

escalonador define, para cada conexão servida numa trama, o número de canais LCH e SCH atribuídos, os respectivos débitos de transmissão e a sua localização na trama. Além disso, o escalonador é responsável pela definição do número de *slots* disponíveis na fase de acesso aleatório em cada trama.

O escalonamento dos recursos pelo ponto de acesso é feito com base na leitura dos elementos provenientes das diferentes filas FIFO descritas abaixo, que se distinguem pela natureza dos seus elementos. O simulador implementa duas instâncias de cada fila, uma para cada classe de tráfego, com excepção das duas primeiras filas, associadas somente ao tráfego de classe I.

- **Fila FC:** Cada elemento desta fila contém um identificador de terminal/conexão, bem como o número de canais LCH que serão atribuídos à conexão, na fase ascendente da trama corrente, pela utilização do mecanismo de negociação de capacidade fixa do HIPERLAN/2[47]. Para a fila FC ser usada, a opção de atribuição de capacidade fixa, aplicável ao tráfego de classe I, deve estar habilitada no simulador.
- **Fila POLL:** Cada elemento desta fila contém um identificador de terminal que serve para requisitar a atribuição de um canal SCH na fase ascendente da trama corrente. Este canal é destinado ao transporte de uma mensagem de requisição de recursos (RR). A introdução de elementos nesta fila é feita periodicamente. Para esta fila ser usada, a opção de atribuição de canais SCH às conexões da classe I deve estar habilitada no simulador.
- **Fila ARQ:** Cada elemento desta fila dá origem a uma mensagem de ARQ no sentido descendente, transportada num canal SCH. A introdução de elementos nesta fila, regulada pelo algoritmo de escalonamento utilizado, é feita para que o adaptador de terminal possa ser informado da necessidade de retransmissão de mensagens corrompidas. A recepção de uma sequência de mensagens de dados sem erros de uma conexão também faz com que um elemento seja introduzido na

[47] Ou então, em alternativa, pelo uso do algoritmo de alocação flexível de capacidade fixa descrito na secção 4.2.2.3.

fila, para permitir ao adaptador a remoção das mensagens com reconhecimento positivo do seu *buffer* de mensagens.

- **Fila RET:** Esta fila está presente quando o algoritmo de retransmissão rápida, descrito na secção 4.2.2.1, é utilizado. Os elementos desta fila são introduzidos durante o escalonamento das mensagens de ARQ. O número de canais LCH requisitados por um elemento desta fila depende do número de mensagens de dados recebidas com erros do respectivo adaptador.
- **Fila RR:** Armazena as requisições de recursos (RR) feitas pelos adaptadores. As requisições novas são colocadas no fim da fila, excepto quando uma requisição associada ao terminal já existe na fila. Neste caso, a nova requisição substitui a antiga.

Cada uma dessas filas corresponde a uma etapa[48] no escalonamento de recursos. As etapas são percorridas em sequência, de acordo com a ordem de prioridades representada na Figura 5.5.

[48] As etapas presentes na simulação dependem da configuração escolhida.

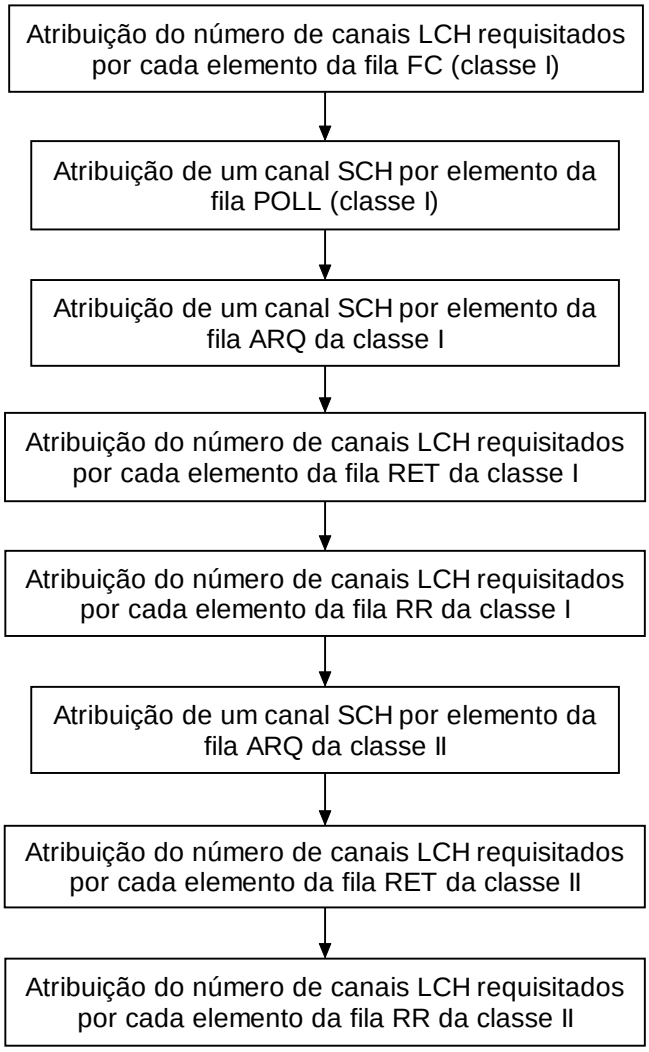


Figura 5.5: Etapas percorridas durante o escalonamento do tráfego efectuado pelo ponto de acesso na rede HIPERLAN/2.

Em cada etapa, o escalonamento de recursos é efectuado de forma iterativa até que um dos seguintes eventos aconteça.

- A respectiva fila fique vazia.
- O espaço disponível na trama se esgote.
- O número máximo de elementos de informação (IEs) no campo FCH da trama[49] seja atingido.

[49] O limite máximo definido pelas especificações do HIPERLAN/2 é de 45 elementos.

A cada iteração, ao ler um elemento de uma dada fila, o escalonador verifica se já existe um elemento de informação (IE) atribuído na trama no mesmo sentido e associado à mesma conexão. Se não existir, além do espaço necessário para a transmissão das mensagens requisitadas pelo elemento, em canais LCH ou SCH, o escalonador tem que verificar se há espaço para o preâmbulo, o período de guarda (para o sentido ascendente) e o elemento de informação.

Quando os recursos requisitados pelo elemento da fila são totalmente atribuídos, este é removido da fila e o escalonamento prossegue com a leitura do elemento seguinte. Por outro lado, quando os recursos disponíveis não são suficientes para atribuir todos os canais requisitados pelo elemento, os seguintes procedimentos são efectuados.

1) O escalonador realiza uma atribuição parcial ou nula, limitada pelos recursos disponíveis.
2) O número de canais requisitados pelo elemento é actualizado, pela subtracção do número de canais atribuídos pelo escalonador. O elemento é mantido na fila.
3) O escalonamento passa à etapa seguinte. As requisições pendentes permanecem na fila à espera de atendimento nas tramas seguintes.

Percorridas todas as etapas, o escalonamento termina. Caso sobre espaço na trama, este pode ser aproveitado para aumentar o número de *slots* disponíveis na fase de acesso aleatório.

5.2.6.4 Requisição de recursos

Os adaptadores de terminal utilizam as mensagens de requisição de recursos (RR) para pedir ao ponto de acesso que atribua um número de canais LCH que possibilite a transmissão dos dados pendentes durante a fase ascendente. O adaptador decide enviar uma nova mensagem de requisição de recursos caso tenha recebido novos

pacotes do terminal desde a última requisição de recursos[50], ou caso seja notificado pelo ponto de acesso (através de uma mensagem de ARQ) da existência de mensagens de dados que necessitam de retransmissão[51].

Em cada trama, imediatamente antes do início da fase de acesso aleatório, o adaptador verifica se é necessário de envio de uma mensagem de requisição de recursos. Caso a decisão seja positiva, o adaptador executa o processo de contenção para definir o *slot* em que será feita a transmissão. Os pacotes que chegam depois do momento de decisão e antes do envio da mensagem de requisição de recursos também são considerados no cálculo do número de canais LCH a requisitar.

O ponto de acesso também pode atribuir canais SCH às conexões na fase ascendente (pela utilização da fila POLL), que podem ser utilizados pelos adaptadores para transmitir mensagens de requisição de recursos. A vantagem deste procedimento é que as mensagens de requisição de recursos são transmitidas livres de contenção, portanto não há risco de colisão. Neste caso, a decisão sobre o envio de uma mensagem de requisição de recursos é tomada imediatamente antes do momento previsto para o início da transmissão.

5.3 Validação dos modelos de simulação

Esta secção apresenta aspectos e conceitos associados às simulações em geral e aos modelos de simulação desenvolvidos em particular. Após a definição dos tipos de simulação, a secção 5.3.1 discute o processo de verificação e validação de modelos de simulação. De seguida, a secção 5.3.2 aborda o tratamento estatístico dos dados de simulação, focando aspectos como a geração de números aleatórios, a remoção do transitório e o critério de paragem da simulação.

As simulações podem ser classificadas em dois tipos no que diz respeito à análise dos dados de saída [ALE00].

[50] A não ser que os recursos para transmissão já tenham sido reservados pelo uso de um mecanismo de atribuição de capacidade fixa.

[51] Caso o mecanismo de retransmissão rápida descrito na secção 4.2.2.1 não esteja habilitado.

- **Simulação de horizonte finito:** Neste caso, a simulação começa num estado inicial específico e termina após a ocorrência de um dado evento. Neste tipo de simulação não se espera que o processo atinja um estado estacionário. Qualquer parâmetro estimado a partido dos dados obtidos será transitório no sentido em que dependerá das condições iniciais.
- **Simulação de estado estacionário:** Neste caso, o interesse situa-se no comportamento do sistema a longo prazo, após o equilíbrio ter sido atingido. Este tipo de simulação normalmente requer a remoção do período transitório inicial.

5.3.1 Verificação e validação de modelos

A implementação de um modelo de simulação normalmente resulta num programa complexo que, caso não sejam tomadas devidas precauções, pode conter erros de programação, dando origem a resultados equivocados. A verificação do modelo procura assegurar que o código associado ao modelo é correctamente implementado.

Um programa de simulação que não contenha erros de programação pode, ainda assim, não representar correctamente o sistema real que se deseja modelar, devido à incorporação no modelo de pressupostos equivocados a respeito do comportamento do sistema. A validação do modelo procura garantir que o mesmo é representativo do sistema real, ou seja, que o modelo reproduz o comportamento do sistema real com fidelidade suficiente para satisfazer os objectivos da análise.

São apresentadas a seguir algumas técnicas de verificação e validação de modelos de simulação, juntamente com exemplos de sua utilização na fase de desenvolvimento do simulador.

- A técnica denominada *antibugging* consiste na introdução de código adicional em pontos-chave do programa, com o objectivo de verificar se certos pressupostos básicos são violados durante a simulação e, em caso positivo, alertar para a existência de erros de programação. Por exemplo, no código associado à rede HIPERLAN/2, após o processo de escalonamento dos recursos na trama, uma

rotina apresenta uma mensagem de erro caso tenham ficado requisições por servir e ao mesmo tempo tenham sobrado recursos na trama.

- Outra técnica utilizada consiste na execução do modelo num cenário simples, por exemplo, com uma única fonte geradora de tráfego. Estes casos podem ser analisados mais facilmente, o que permite que os resultados da simulação possam ser comparados com os resultados teóricos. Como é óbvio, não é garantido que um modelo que funcione num cenário simples venha a funcionar num cenário mais complexo, no entanto, um modelo que não funcione num cenário simples certamente não funcionará num cenário mais complexo.

- A técnica denominada teste de continuidade consiste na execução de um conjunto de simulações com parâmetros de entrada ligeiramente diferentes. Uma pequena alteração no valor de um parâmetro de entrada, mantidos os valores dos outros parâmetros, costuma produzir uma mudança pequena nos resultados obtidos. Alterações bruscas nos resultados devem ser investigadas, pois nestes casos é provável a existência de erros no modelo. Nas simulações efectuadas, a continuidade foi verificada no traçado de gráficos em função do valor de um parâmetro de entrada, como o número de terminais activos.

- Os resultados obtidos nas simulações devem ser independentes dos valores iniciais (*seeds*) utilizados pelos geradores de números aleatórios, ou seja, o modelo deve produzir resultados similares com diferentes valores de *seeds*. Isso foi verificado durante as simulações efectuadas utilizando o método das replicações, descrito na secção 5.3.2.3.

- A cada alteração relevante no código do programa, os seus efeitos foram avaliados através da execução da simulação com um mesmo conjunto de parâmetros e condições iniciais Algumas mudanças no código, quando correctamente implementadas, não devem alterar os resultados, enquanto outras pressupõem alterações dos valores de saída num dado sentido. A comparação dos resultados obtidos antes e depois das mudanças no código permitiu detectar alterações suspeitas, levando à análise do modelo e à detecção precoce de erros.

- No intuito de validar os modelos, fez-se também a comparação entre os resultados publicados por outros autores e os resultados obtidos na simulação dos modelos desenvolvidos (adaptados aos cenários de simulação utilizados pelos autores).

5.3.2 Tratamento estatístico dos dados

Segundo Pawlikowski et al. [POW02], existe actualmente uma crise de credibilidade nos estudos publicados na área de análise do desempenho de redes de comunicação baseados em simulação estocástica, derivada da pouca atenção dada ao tratamento estatístico dos dados de simulação. Os autores sugerem que, além dos resultados obtidos nas simulações, os estudos apresentem informações sobre o tipo de simulação efectuado, o gerador de números aleatórios utilizado, o método de análise dos dados obtidos e os erros estatísticos finais associados aos resultados. Estes aspectos são abordados nesta secção.

5.3.2.1 Números aleatórios

Normalmente o processo de simulação requer a obtenção de amostras provenientes de diferentes distribuições de probabilidade. Isto costuma ser feito em duas etapas. Primeiro, obtém-se uma sequência de números aleatórios uniformemente distribuídas no intervalo [0, 1]. Depois, a sequência é transformada de modo a produzir valores aleatórios associados à distribuição desejada.

A utilização de sequências de números aleatórios é necessária para a execução de diversas tarefas no simulador desenvolvido, designadamente:

- no cálculo do intervalo entre chegadas do tráfego de débito variável gerado pelos terminais;
- nos algoritmos de resolução de contenção das redes sem fios analisadas;
- na determinação das mensagens afectadas por erros no canal;
- e no cálculo dos instantes de transição de estado no modelo de erros de Gilbert-Elliot.

As seguintes propriedades devem ser satisfeitas pelo gerador de números aleatórios utilizado numa simulação.

- Todos os valores possíveis de serem gerados devem ter a mesma probabilidade de ocorrência, ou seja, devem ser uniformemente distribuídos.
- Os valores gerados devem ser suficientemente independentes. Para isso, a correlação entre os sucessivos números gerados deve ser pequena.
- O comprimento do ciclo do gerador deve ser suficientemente largo para que seja possível evitar a reutilização de uma mesma sequência de números aleatórios dentro de uma simulação.

O gerador de números aleatórios utilizado nas simulações [VAR00] pertence à categoria de geradores denominada *multiplicative* LCG (*Linear-Congruential Generator*) [JAI00]. A função geradora usada para obtenção do número seguinte (x_n) na sequência dado o número anterior (x_{n-1}) é a seguinte:

$$x_n = 7^5 x_{n-1} \bmod (2^{31} - 1), \tag{5.3}$$

A função *mod*, que aparece na expressão, retorna o resto da divisão de dois inteiros. Este gerador possui um comprimento de ciclo igual a 2^{31} - 2.

Apesar de serem denominados geradores de números aleatórios, estes geradores produzem uma sequência determinística (pseudo-aleatória), pois, dado o valor inicial, é possível determinar qualquer valor na sequência. A utilização de uma sequência pseudo-aleatória nas simulações apresenta algumas vantagens. Como a sequência gerada não se altera, as diferenças de resultados entre dois cenários diferentes devem-se às configurações utilizadas e não a variações estatísticas. Além disso, a comparação dos resultados obtidos com uma mesma sequência antes e depois de uma alteração no modelo facilita a detecção de erros.

Para evitar a obtenção de resultados incorrectos devido à correlação entre as diferentes tarefas que fazem uso de sequências de números aleatórios, as seguintes medidas foram implementadas:

- Cada tarefa no simulador utiliza um gerador de números aleatórios diferente, evitando com isso a subdivisão de uma sequência.

- A selecção dos valores iniciais (*seeds*) de cada gerador é feita de forma que não haja sobreposição entre as sequências dos diversos geradores.
- A selecção dos valores iniciais em cada replicação, no caso da utilização do método das replicações apresentado adiante, é feita de forma que não haja sobreposição entre as sequências das diversas replicações.

5.3.2.2 Remoção do transitório

Quando um sistema é estável, o seu comportamento dinâmico tende a estabilizar num estado estacionário, independente do tempo, após um período transitório inicial. A duração do período transitório não é conhecida a priori, sendo influenciada por diversos factores, como os parâmetros do sistema e as condições iniciais.

Em grande parte das simulações, os resultados desejados dizem respeito ao comportamento do sistema durante o estado estacionário, sendo necessário providenciar um método de remoção do transitório para evitar que as amostras recolhidas durante este período venham a distorcer o resultado final. Entretanto, não é possível definir com exactidão quando termina o período transitório e começa o período estacionário, pelo que todos os métodos de remoção de transitório são heurísticos.

Um método grosseiro de lidar com o período transitório é conhecido como execução longa. A ideia por trás deste método é fazer com que a execução seja suficientemente longa para assegurar que o efeito do período transitório sobre os resultados torne-se irrelevante. Uma desvantagem deste método consiste no desperdício de recursos derivado da execução da simulação durante mais tempo que o necessário. Além disso, a sua eficácia é questionável na medida em que é problemático saber quão longa deve ser a execução para que se obtenha o efeito desejado. Sendo assim, a utilização deste método não é recomendável.

Outra alternativa proposta na literatura é conhecida pelo nome de inicialização adequada. Neste método, as condições iniciais são escolhidas de forma que o sistema parta de um estado próximo do estado estacionário. No entanto, existe o problema da

determinação do estado inicial adequado. Além disso, a solução que venha a ser encontrada é intrinsecamente dependente do sistema analisado.

Os outros métodos existentes baseiam-se na remoção das amostras recolhidas no início da simulação, variando na forma como avaliam a duração do período transitório. Em [JAI00], o autor descreve alguns métodos propostos para a remoção do transitório, baseados no pressuposto de que a variabilidade observada durante o período estacionário é menor do que durante o período transitório. Nesses métodos, os cálculos para a remoção do transitório são feitos após o término da simulação, o que requer que todas as observações, nalguns casos provenientes de várias replicações, tenham que ser armazenadas em memória. Outro problema é que os critérios de paragem[52] de uma simulação baseiam-se na recolha de amostras somente durante o estado estacionário, porém, estes métodos de remoção do transitório não permitem determinar o término do período transitório durante a simulação, pelo que a simulação têm que terminar num momento arbitrário. Caso o término da simulação seja prematuro o período transitório poderá não ter acabado, pelo que os dados recolhidos serão inúteis, caso contrário, pode haver desperdício de recursos se a simulação durar mais tempo que o necessário.

O método de remoção do transitório utilizado baseia-se no facto de que, nos cenários analisados, o atraso médio dos pacotes tende a crescer até estabilizar, devido aos *buffers* de pacotes das estações estarem inicialmente vazios. As amostras dos atrasos são divididas em lotes com um dado tamanho n_t. Ao término da recolha das amostras de um dado lote, calcula-se a respectiva média:

$$\overline{x}_i = \frac{1}{n_t} \sum_{j=1}^{n_t} x_{ij} \tag{5.4}$$

A seguir, testam-se as seguintes desigualdades: $\overline{x}_{n-1} < \overline{x}_{n-2}$ e $\overline{x}_n < \overline{x}_{n-2}$. Quando ambas as desigualdades são satisfeitas, considera-se que foi atingido o estado estacionário, passando-se à etapa de recolha de amostras que irão fazer parte da estimativa final.

[52] Discutidos na secção 5.3.2.3.

5.3.2.3 Critério de paragem

A análise estatística dos dados recolhidos permite determinar o grau de confiança da estimativa, normalmente expresso pelo intervalo de confiança da estimativa para um dado nível de confiança, ou seja, um intervalo de valores em torno da estimativa que se espera que contenha o valor exacto com uma dada probabilidade *p*, sendo

$$p = 1 - \alpha \,, \tag{5.5}$$

onde α é o denominado nível de significância. O nível de confiança corresponde à probabilidade *p* expressa em percentagem, ou seja, é igual a $100(1-\alpha)$.

Dado o comprimento do intervalo de confiança (*c*) e o valor da estimativa ($\bar{x}$), o intervalo de confiança estende-se entre $\bar{x} - c/2$ e $\bar{x} + c/2$.

A precisão[53] (*r*) da estimativa para um dado nível de confiança é dada pela razão entre a metade do comprimento do intervalo de confiança e o valor da estimativa:

$$r = \frac{c/2}{\bar{x}} \tag{5.6}$$

O comprimento do intervalo de confiança e, consequentemente, o valor da precisão, tendem a diminuir com o aumento do número de amostras colectadas. O término da simulação num instante arbitrário pode levar à obtenção de uma estimativa com precisão insuficiente, quando precoce, ou ao desperdício de tempo e recursos, quando tardio. Sendo assim, convém que a simulação decorra até que seja obtida a precisão desejada.

No simulador desenvolvido, foram implementados dois métodos distintos para a determinação do momento adequado para terminar uma simulação: o método das replicações e o método dos subintervalos [TAH89].

No método das replicações, cada observação é obtida de uma execução diferente, com os mesmos parâmetros, porém, com sequências diferentes de números aleatórios, o que garante a independência das observações. Cada observação consiste na média de n_s amostras obtidas após a remoção do período transitório:

[53] Também conhecida como erro estatístico relativo.

$$x_i = \frac{1}{n_s}\sum_{j=1}^{n_s} x_{ij} \tag{5.7}$$

Dadas n observações (x_1 ... x_n) de uma variável do sistema, obtidas em diferentes replicações, a estimativa final é dada pela média das observações:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{5.8}$$

O comprimento do intervalo de confiança da estimativa[54] é expresso por:

$$c = 2\, t_{[1-\alpha/2,\, n-1]}\frac{s}{\sqrt{n}}, \tag{5.9}$$

onde s é o desvio padrão das observações:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{5.10}$$

e $t_{[1-\alpha/2,\ n-1]}$ é o quantil $(1-\alpha/2)$ da distribuição t de Student com n-1 graus de liberdade. A Tabela A1 apresenta uma listagem de valores de $t_{[S,N]}$.

A obtenção da estimativa final pelo método das replicações é feita com base na precisão e no nível de confiança desejados. O processo implementado no simulador é descrito a seguir.

1) Executa-se a primeira replicação, da qual resulta uma observação, calculada conforme a equação 5.7 a partir de n_s amostras recolhidas após a passagem do período transitório.
2) Executa-se a replicação seguinte com outras sequências de números aleatórios, da qual resulta mais uma observação, obtida como no primeiro passo.
3) Calcula-se a média provisória das observações obtidas até este momento, com base na equação 5.8.

[54] O comprimento do intervalo de confiança é inversamente proporcional a $\sqrt{n_s n}$

4) Calcula-se a precisão da estimativa neste ponto da simulação, com base nas equações 5.10, 5.9 e 5.6.

5) Caso a precisão desejada tenha sido atingida, a simulação termina, caso contrário, retorna-se ao segundo passo.

Uma alternativa ao método das replicações é o denominado método dos subintervalos, também conhecido como método das subamostras ou método da média dos lotes. Neste caso, cada simulação é realizada com uma única execução de longa duração. Após o descarte do período transitório inicial, divide-se as amostras recolhidas em lotes de comprimento n_s, obtendo-se uma observação de cada lote pelo uso da equação 5.7. A estimativa final é dada pela média das observações dos diferentes lotes:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \,, \tag{5.11}$$

onde n é o número total de lotes.

A média das observações é exactamente igual à média do conjunto das amostras de todos os lotes, ou seja, a divisão das amostras em lotes não tem influência sobre a estimativa obtida. O seu propósito é possibilitar o cálculo do intervalo de confiança da estimativa. Uma vantagem do método do subintervalos sobre o método das replicações é que a remoção do período transitório só precisa ser feita uma vez. Outra vantagem é que basta uma execução da simulação para o cálculo da estimativa final. A desvantagem é que o pressuposto de independência das observações deixa de existir, pois as condições iniciais de cada intervalo dependem dos valores finais do intervalo imediatamente anterior. Entretanto, se o tamanho dos lotes for suficientemente longo as observações serão aproximadamente independentes.

O procedimento implementado no simulador para determinação da estimativa final (e o consequente encerramento da simulação) utilizando o método dos subintervalos, dados a precisão e nível de confiança desejados, é descrito a seguir.

1) Após o início da execução, aguarda-se pelo término do período transitório para se passar à segunda etapa.

2) Recolhem-se n_s amostras, das quais resulta uma observação, calculada de acordo com a equação 5.7.
3) Descartam-se as amostras anteriores e recolhem-se outras n_s amostras, que dão origem a uma nova observação.
4) Calcula-se a média provisória das observações obtidas até o momento, de acordo com a equação 5.11.
5) Calcula-se a precisão até este ponto da simulação, com base nas equações 5.10, 5.9 e 5.6
6) Caso a precisão desejada tenha sido atingida, a simulação termina, caso contrário, volta-se ao terceiro passo.

5.4 Sumário

Este capítulo apresentou os modelos de simulação desenvolvidos para as redes IEEE 802.11 e HIPERLAN/2. Após a descrição das entidades de tráfego presentes nos modelos, foram apresentados os módulos que representam os diversos componentes de uma rede e os detalhes de implementação relativos a cada rede em particular.

Em seguida, foram discutidos conceitos associados às simulações, bem como a aplicação dos mesmos aos modelos desenvolvidos, incluindo a verificação e validação de modelos, a geração de números aleatórios, a remoção do transitório e o critério de paragem associado à obtenção da estimativa.

Os modelos de simulação que aqui foram apresentados são utilizados no próximo capítulo para a avaliação do desempenho dos algoritmos de escalonamento propostos no capítulo anterior, tendo em vista o transporte do tráfego característico de um sistema de aquisição de dados e controlo.

Capítulo 6

Simulação dos modelos desenvolvidos

Este capítulo começa por apresentar os parâmetros de tráfego do sistema e os critérios utilizados nas simulações. As secções seguintes apresentam e discutem os resultados obtidos, primeiro para o sistema baseado na rede IEEE 802.11 e depois para o sistema baseado na rede HIPERLAN/2. A parte final do capítulo sintetiza os resultados e compara o desempenho obtido com as duas redes.

6.1 Introdução

6.1.1 Parâmetros de tráfego

O cenário de aplicação requerido pelo sistema de aquisição de dados e controlo descrito na secção 5.1 é composto por um número elevado de fontes de tráfego de tempo real de baixo débito em conjunto com fontes de tráfego assíncrono. Considera-se na análise do sistema que o tráfego isócrono é proveniente de conexões ATM ou RDIS, enquanto o tráfego assíncrono é gerado por aplicações TCP/IP.

A Tabela 6.1 apresenta as características do tráfego gerado pelos terminais tanto para o sistema baseado na rede IEEE 802.11a como para sistema baseado na rede HIPERLAN/2, o que permite a comparação do desempenho obtido nos dois casos. O tráfego de classe I representa conexões ATM (I_a) ou RDIS (I_b) com requisitos de tempo real, enquanto a classe II representa tráfego assíncrono do tipo IP para ser transportado sem garantias de qualidade de serviço.

Tabela 6.1: Características do tráfego usado nas simulações

Classe	I_a	I_b	II
Prioridade	Alta		Baixa
Número de terminais/conexões	30[55]		0-50
Tipo de pacote	ATM	RDIS	IP
Comprimento dos pacotes (octetos)	53	1	1500
Débito médio (kbit/s)	70.67[56]	64	300
Tipo de tráfego	CBR		Poisson

6.1.2 Critérios utilizados nas simulações

Os resultados apresentados neste capítulo foram obtidos em simulação de estado estacionário, salvo indicação em contrário. O critério de paragem é baseado no método dos subintervalos, conforme descrito na secção 5.3.2.3. A simulação termina quando a média das observações referentes ao atraso da conexão do primeiro terminal (com o endereço mais baixo) atinge uma precisão melhor que 10 % para um nível de confiança de 95 %.

Nos histogramas apresentados, a largura das células é de 125 μs. O número de amostras abrangidas por cada célula foi dividido pelo número total de amostras para permitir a visualização da frequência associada a cada célula.

[55] Salvo indicação em contrário.

[56] Este débito corresponde a um débito de 64 kbit/s para os dados encapsulados na célula ATM, assumindo que o *payload* da célula é usado na totalidade (AAL0).

6.2 Sistema baseado na rede IEEE 802.11

Esta secção apresenta os resultados obtidos com a utilização da rede IEEE 802.11 versão "a", que opera na banda de 5 GHz e possui uma camada física semelhante à do HIPERLAN/2. Os resultados apresentados referem-se, em grande parte, à utilização dos mecanismos de acesso ao meio originais do IEEE 802.11. Quando apropriado, apresentam-se também resultados obtidos com os mecanismos de acesso ao meio propostos na norma IEEE 802.11e, nomeadamente quando, no âmbito dos cenários propostos, a sua utilização introduz alterações relevantes nos resultados em relação ao uso dos mecanismos originais

6.2.1 Parâmetros de simulação

Os parâmetros associados à rede IEEE 802.11 podem ser divididos em duas categorias: os parâmetros fixos, cujo valor é único e não pode ser alterado; e os parâmetros configuráveis, cujo valor pode ser atribuído com base num conjunto de valores definido pelas especificações.

A Tabela 6.2 apresenta os parâmetros fixos da rede IEEE 802.11a que são utilizados nas simulações e os seus valores. Já a Tabela 6.3 apresenta os parâmetros configuráveis da rede IEEE 802.11a que são utilizados, bem como os valores atribuídos nas simulações, salvo indicação em contrário. Utilizou-se o mesmo débito de transmissão para as tramas de dados e para as tramas de controlo.

Na simulação do sistema baseado na rede IEEE 802.11a, as conexões da classe I são servidas durante o período livre de contenção (CFP) utilizando a função PCF[57] para o controlo de acesso ao meio, sendo que o *payload* das mensagens de dados é formado pela concatenação dos pacotes[58] à espera no *buffer* do adaptador. Já o tráfego da classe II é transmitido durante o período de contenção (CP) utilizando a função DCF.

[57] Salvo indicação em contrário.

[58] Células, no caso do tráfego ATM, e blocos de octetos, no caso do tráfego RDIS.

Tabela 6.2: Parâmetros fixos da rede IEEE 802.11a utilizados nas simulações.

Descrição	Designação	Valor
Overhead das tramas de *dados*	*DataOvhLength*	34x8 bits
Comprimento da trama ACK	*ACKLength*	16x8 bits
Comprimento da trama CF-End	*CFEndLength*	24x8 bits
Comprimento máximo do *payload*	*MSDUMax*	2304x8 bits
Overhead da subcamada PLCP	*PLCPOvhTime*	20 μs
Comprimento do campo SERVICE	*SERVICELength*	16 bits
Duração do *slot*	*SlotTime*	9 μs
Período SIFS	*SIFSTime*	16 μs
Período PIFS	*PIFSTime*	25 μs
Período DIFS	*DIFSTime*	34 μs

Tabela 6.3: Parâmetros configuráveis da rede IEEE 802.11a utilizados nas simulações.

Descrição	Designação	Valor
Débito de transmissão	*PHYRate*	18 Mbit/s
Valor mínimo da janela de contenção	*CWMin*	15
Valor máximo da janela de contenção	*CWMax*	1023
Duração da supertrama	*SFPeriod*	6 ms
Duração máxima do CFP	*CFPMaxDuration*	5 ms

6.2.2 Tráfego prioritário sem erros no canal

Os resultados apresentados a seguir referem-se à operação do sistema apenas com o tráfego da classe I_a, correspondendo às 30 conexões ATM CBR. O primeiro terminal, ou seja, aquele com o endereço mais baixo, recebeu a denominação MT1, e a conexão que lhe está associada foi denominada RT1. Durante o período livre de

contenção, os terminais com conexões de classe I são interrogados sequencialmente pelo ponto de acesso, em ordem crescente de endereçamento, pelo que a conexão associada ao primeiro terminal (RT1) é a primeira a ser servida, e assim por diante. A duração da supertrama (*SFPeriod*) utilizada é igual ao intervalo de geração de células (6 ms), de modo a minimizar o *jitter*. Para facilitar a visualização dos resultados, todas as conexões começam a gerar tráfego ao mesmo tempo, a partir do instante inicial de simulação ($t = 0$).

A Figura 6.16 apresenta a distribuição dos atrasos para 3 das 30 conexões presentes na simulação[59] num cenário sem erros no canal. Como a conexão associada ao primeiro terminal (RT1) é a primeira a ser servida, o seu atraso é menor do que o das outras conexões. A conexão RT10 é a décima a ser servida na supertrama, enquanto que a conexão RT30, por ser a última a ser servida, sofre o maior atraso entre as conexões. Num caso mais genérico em que o instante de geração da primeira célula de cada conexão seja aleatório, o atraso das diferentes conexões pode situar-se entre um mínimo próximo de zero e um máximo de cerca de 6 ms (que corresponde ao intervalo entre as interrogações). O atraso depende da diferença entre o instante de interrogação do terminal, determinado pelo ponto de acesso, e o instante de geração da célula, que não precisa ser idêntico para todas as conexões como neste caso.

[59] As demais conexões apresentam resultados similares.

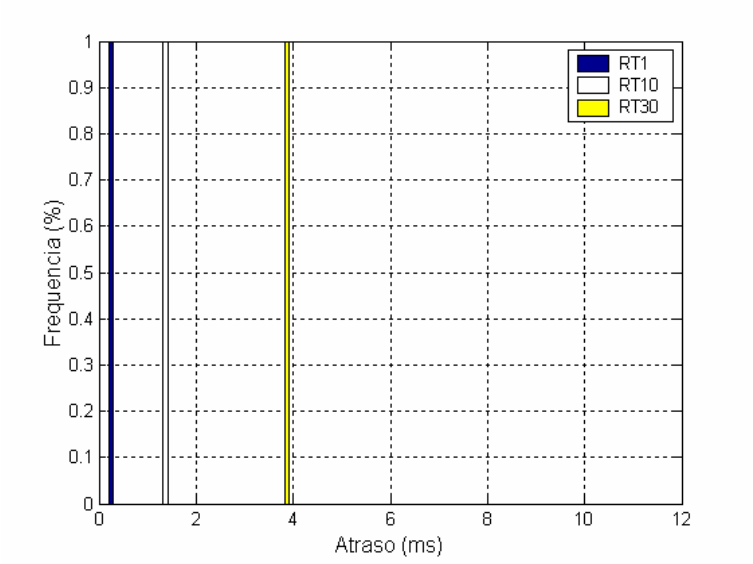


Figura 6.1: Distribuição dos atrasos de três conexões da classe I_a (sem erros no canal).

O tempo de transmissão de uma trama de interrogação (T_{POLL}) ao débito de 18 Mbit/s é de 36 µs, sendo calculado pela seguinte equação:

$$T_{POLL} = PLCPOvhTime + \frac{SERVICELength + DataOvhLength}{PHYRate} \quad (6.1)$$

O tempo de transmissão da trama de resposta do terminal (T_{RESP}) é dado pela equação:

$$T_{RESP} = PLCPOvhTime + \frac{SERVICELength + DataOvhLength + PayloadLength}{PHYRate}, \quad (6.2)$$

onde *PayloadLength* é o comprimento do *payload*, em bits. Neste caso o comprimento do *payload* é igual ao tamanho da célula ATM, ou seja, 424 bits, e o tempo de transmissão da trama de resposta é de 59,6 µs.

O tempo total associado à interrogação e respectiva resposta de um terminal (T_{PR}), incluindo os períodos SIFS entre tramas é de 127.6 µs neste caso, de acordo com a equação:

$$T_{PR} = T_{POLL} + T_{RESP} + 2 \cdot SIFSTime \quad (6.3)$$

Sendo assim, a interrogação aos 30 terminais demora ao todo cerca de 3.83 ms. O débito agregado das 30 conexões CBR é de 2120 Mbit/s, que equivale a cerca de 11.8

% do débito da rede, no entanto, a utilização do canal pelas conexões da classe I é bastante maior (64.7 %), ou seja a eficiência obtida é baixa. A eficiência, calculada com a equação 5.2, é de apenas 18.2 %, devido ao *overhead* introduzido pelo IEEE 802.11, que é mais relevante quando o comprimento do *payload* das mensagens de dados é pequeno. É o que acontece neste cenário, em que o *payload* é formado pelo conteúdo de uma única célula ATM (53 octetos).

Neste caso, o período livre de contenção é encerrado antes de se atingir a duração máxima especificada, sobrando 35.3 % da largura de banda total para ser utilizado para transmissões durante o período de contenção (cerca de 2.1 ms por supertrama). Embora apresente-se aqui resultados relativos ao suporte de 30 terminais, caso a duração máxima disponível para o período livre de contenção fosse aproveitada, o número de terminais com tráfego da classe I poderia atingir um máximo de 38, neste cenário, sem que houvesse degradação na qualidade de serviço oferecida às conexões.

A eficiência do sistema pode ser aumentada com o aumento do período da supertrama. Por exemplo, se usarmos um período de 12 ms, o intervalo entre as interrogações dos terminais duplica e o *payload* das mensagens de dados passa a ser composto por duas células ATM, pelo que o número de mensagens transmitidas durante esse período cai para metade, diminuindo o *overhead*. Por outro lado, o atraso sofrido pelas células ATM aumenta. Neste exemplo, a primeira célula transportada no *payload* de uma mensagem sofre um atraso adicional de 6 ms em relação à segunda célula. Outro factor a ter em consideração é que o aumento do comprimento do *payload* aumenta a probabilidade das mensagens serem corrompidas por erros no canal.

A duração da supertrama do IEEE 802.11 foi fixada em 6 ms para que fosse igual ao intervalo de geração das células ATM do tráfego da classe I, enquanto a duração máxima do período livre de contenção (CFP) foi configurada em 5 ms. Com estes valores, a duração mínima do período de contenção (CP) é de 1 ms, o que é suficiente para a transmissão de uma trama de dados com tráfego da classe II. Entretanto, se fosse utilizado o débito de transmissão mínimo e o MPDU máximo, a duração mínima do período de contenção (CP) teria de ser cerca de 3.3 ms, o que faria com que a duração máxima do período livre de contenção fosse insuficiente para permitir a transmissão das 30 conexões da classe I.

Dado que para este cenário não há diferenças significativas entre a função PCF e o mecanismo HCCA da norma IEEE 802.11e, os resultados obtidos com este último seriam semelhantes.

A seguir, para fins de comparação, são apresentados os resultados obtidos utilizando a função de coordenação distribuída (DCF) do IEEE 802.11 (no qual o controlo de acesso ao meio é efectuado de forma distribuída pelas estações), em vez da função PCF, para transporte do tráfego da classe I. O cenário é semelhante ao anterior, porém, com uma alteração: o instante inicial de geração de pacotes de cada conexão passa a ser obtido de uma distribuição uniforme entre 0 e 6 ms, em vez de ser idêntico para todas as conexões. Esta alteração foi efectuada porque se o sincronismo na geração dos pacotes fosse mantido isso teria um impacto negativo forte sobre o desempenho da função DCF, devido à contenção entre as estações.

O tempo de transmissão de uma trama de dados contendo uma célula ATM ao débito de 18 Mbit/s (incluindo a respectiva trama ACK e os períodos entre tramas) é de 137.6 μs. Se todos terminais transmitirem uma vez a cada 6 ms, o tempo total associado às transmissões será igual a esse valor multiplicado pelo número de terminais. Com 30 terminais, o tempo total é de cerca de 4.13 ms, o que corresponde a 68.8 % do tempo disponível. Este valor é fornecido meramente como um indicativo da carga a que a rede é submetida, não tendo em consideração o tempo perdido em colisões e períodos de *backoff*, nem as suas consequências.

Nos resultados obtidos, observou-se que a distribuições dos atrasos de todas as conexões da classe I são semelhantes, pelo que se optou pela apresentação dos resultados relativos ao agregado de conexões. A Figura 6.2 apresenta a distribuição dos atrasos para as conexões da classe I com 30 terminais activos. O atraso médio registado é de 522 μs e o atraso da maioria dos pacotes é pequeno (abaixo de 1 ms), porém, a distribuição dos atrasos estende-se até valores muito mais elevados (superiores a 20 ms) devido a colisões sofridas por alguns dos pacotes. Isso contrasta com o resultado obtido com a função PCF, em que o atraso associado a algumas conexões é maior (podendo atingir um máximo de 6 ms), mas é constante, ou seja, o *jitter* é nulo.

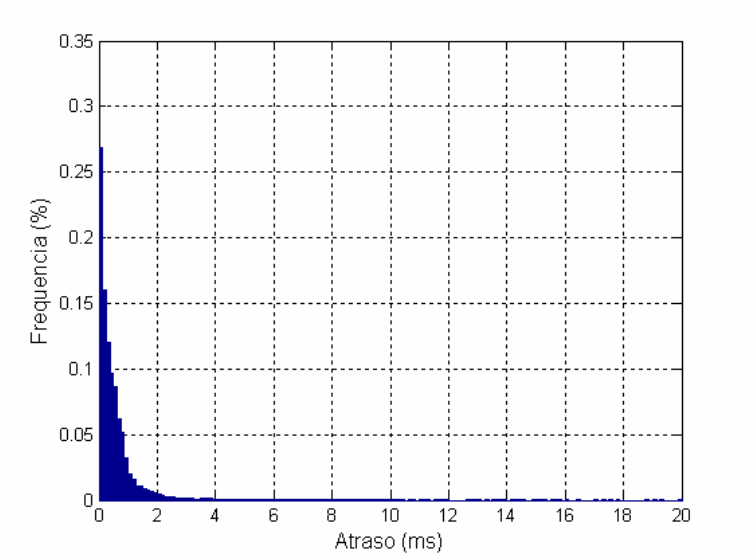


Figura 6.2: Distribuição dos atrasos para as conexões da classe I com 30 terminais utilizando a função DCF (CW_{min} = 15, CW_{max} = 1023).

O complemento da função cumulativa de distribuição (CCDF, *Complementary Cumulative Distribution Function*) da variável aleatória X permite determinar a probabilidade de que o valor da variável ultrapasse um dado limite x. O traçado deste gráfico, tendo o atraso como variável aleatória, permite determinar a probabilidade do atraso dos pacotes exceda um limite (arbitrário), ou seja, permite expressar a taxa de perda de pacotes devido a ultrapassagem do prazo de validade.

A Figura 6.3 representa o complemento da função cumulativa de distribuição do atraso das conexões da classe I. Cada curva no gráfico é proveniente de uma simulação com um número diferente de terminais. Pode-se ver no gráfico, por exemplo, que com 30 terminais cerca de $3x10^{-5}$ pacotes sofrem um atraso superior a 20 ms. O atraso com um número inferior de terminais é significativamente menor devido à redução do número de colisões[60]. Por outro lado, com 35 terminais o atraso é significativamente maior, sendo superior a 20 ms para 8 % dos pacotes.

[60] A análise às conexões individualmente mostra que alguns terminais nem chegam a executar o processo de *backoff*, por encontrarem o canal inactivo sempre que querem transmitir. Isto se deve a condições favoráveis, nomeadamente ao intervalo de geração de pacotes ser idêntico para todas as conexões, às mesmas estarem uniformemente distribuídas, e à carga na rede ser baixa.

Enquanto que com a função PCF o atraso máximo das conexões é de 6 ms até um máximo de 38 terminais (e ainda sobra uma parcela da largura de banda, reservada para a transmissão do tráfego de baixa prioridade), com a função DCF o atraso aumenta significativamente à medida em que aumenta o número de terminais e, consequentemente, a carga na rede.

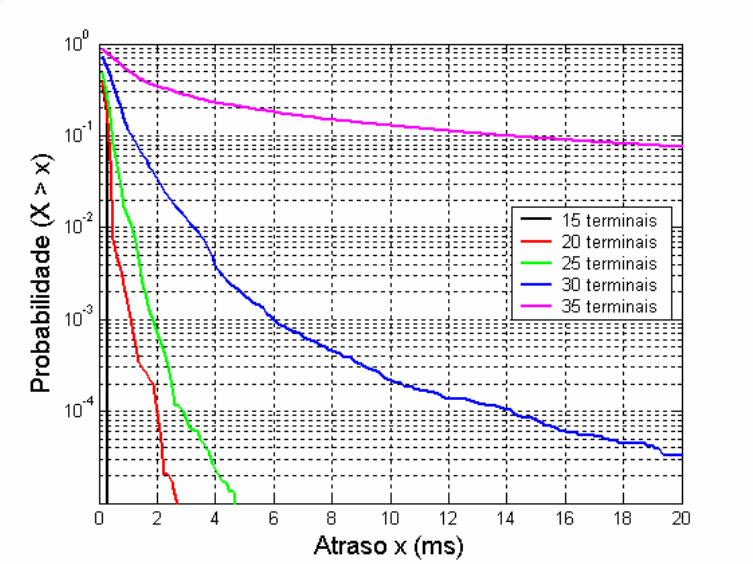


Figura 6.3: Complemento da função cumulativa de distribuição do atraso das conexões da classe I utilizando a função DCF (CW_{min} = 15, CW_{max} = 1023).

Com a função DCF, os parâmetros utilizados para o acesso ao meio são iguais para o tráfego de todas as classes, pelo que o tráfego da classe II concorreria em igualdade de condições com o tráfego mais prioritário da classe I. O mecanismo EDCA da norma IEEE 802.11e, por outro lado, permite dar prioridade de transmissão ao tráfego de uma classe em detrimento do tráfego de outras classes menos prioritárias. No mecanismo EDCA, os parâmetros de *backoff* tipicamente utilizados pela categoria de acesso (AC) de mais alta prioridade são $AIFS = DIFS$, $CW_{min} = 3$ e $CW_{max} = 7$, pelo que se simulou o transporte do tráfego de classe I utilizando estes parâmetros.

A Figura 6.4 representa o complemento da função cumulativa de distribuição do atraso das conexões da classe I utilizando o mecanismo EDCA e os parâmetros especificados acima, para diferentes números de terminais. Observa-se que, com até 25 terminais, consegue-se obter atrasos inferiores aos obtidos com a função DCF,

devido à diminuição do período médio de *backoff*. Porém, com 30 ou mais terminais o atraso atinge proporções muito superiores. Isso acontece porque a redução dos valores da janela de contenção efectuada para o mecanismo EDCA resulta num aumento significativo da probabilidade de colisão quando o número de terminais activos é elevado.

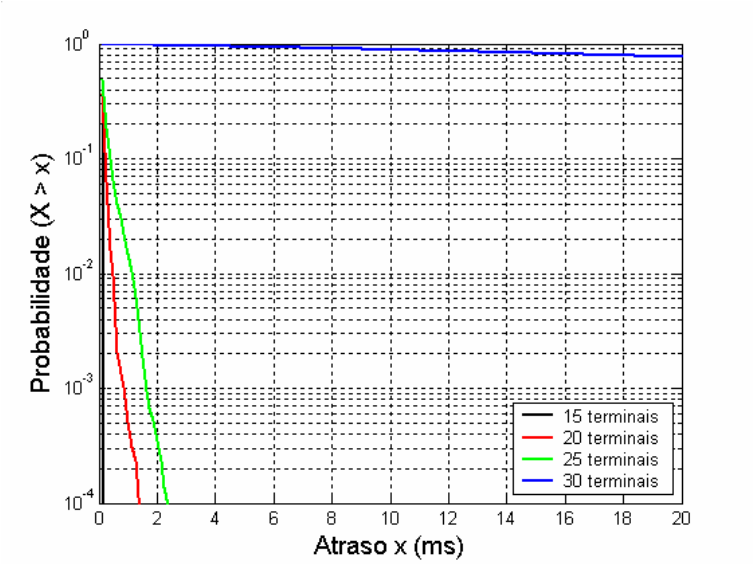


Figura 6.4: Complemento da função cumulativa de distribuição do atraso das conexões da classe I utilizando o mecanismo EDCA (*AIFS[I]* = *DIFS*, $CW_{min}[I] = 3$, $CW_{max}[I] = 7$).

6.2.3 Tráfego prioritário com erros no canal

Os próximos resultados foram obtidos com erros no canal. Estes resultados são expressos em função da taxa de erros binários (BER) ao nível da camada de ligação de dados, que tende a ser menor que a taxa de erros ao nível da camada física, devido à existência de um mecanismo de correcção de erros na camada física do IEEE 802.11a.

Foram implementados os seguintes algoritmos de escalonamento para recuperação dos dados corrompidos por erros no sistema baseado na rede IEEE 802.11, cujos detalhes foram apresentados na secção 4.2.1.

A) Retransmissão dos dados corrompidos na supertrama seguinte, dentro da mensagem original.

B) Retransmissão dos dados corrompidos agrupados com os novos na supertrama seguinte, dentro de uma nova mensagem.

C) Retransmissão da mensagem corrompida na mesma supertrama, após o ciclo regular de transmissões, desde que a duração máxima do período livre de contenção não seja ultrapassada.

Os resultados apresentados nesta secção são referentes à operação do sistema com uma taxa de erros binários (BER) igual a 10^{-4}. De resto, o cenário é idêntico ao utilizado na secção anterior. Para cada algoritmo, foram recolhidas 12000 amostras de cada uma das 30 conexões, sendo apresentadas as distribuições dos atrasos para a primeira conexão (RT1) e para a última conexão (RT30), o que permite verificar se as conexões recebem um tratamento idêntico por parte do sistema. Os resultados apresentados permitem a comparação de desempenho entre os algoritmos propostos para a rede IEEE 802.11 (baseados na função PCF), bem como a comparação posterior com os resultados obtidos com o mecanismo EDCA e com o algoritmo proposto para a rede HIPERLAN/2.

A Figura 6.5 apresenta a distribuição dos atrasos das células da conexão RT1 obtida com a utilização algoritmo A, enquanto a Figura 6.6 apresenta distribuição dos atrasos para a conexão RT30 no mesmo cenário.

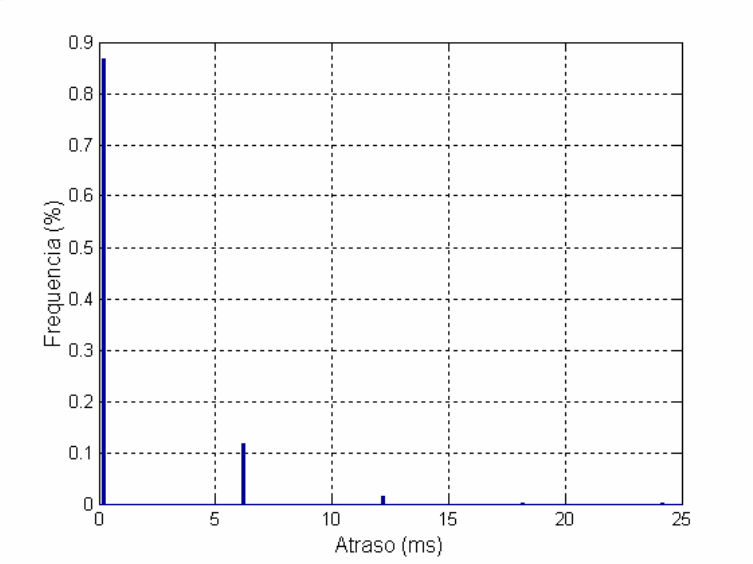


Figura 6.5: Distribuição dos atrasos para a conexão RT1 com erros no canal (BER = 10^{-4}) - Algoritmo A.

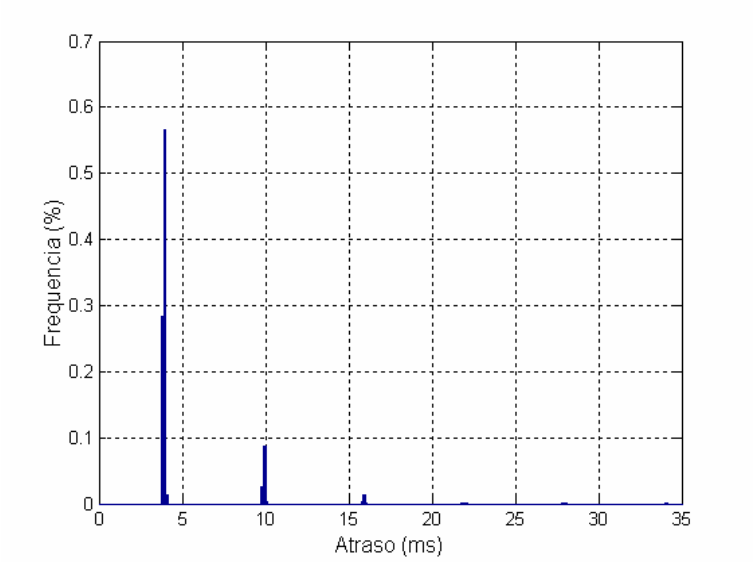


Figura 6.6: Distribuição dos atrasos para a conexão RT30 com erros no canal (BER = 10^{-4}) - Algoritmo A.

O atraso mínimo nos dois casos corresponde à situação em que as células são recebidas sem erros à primeira tentativa, sendo maior para a conexão RT30, como na Figura 6.1, apenas devido ao instante inicial de geração de células utilizado. O espaçamento entre as raias do histograma é de cerca de 6 ms, o que corresponde ao intervalo entre as retransmissões de uma célula. O número máximo de retransmissões registado foi 4 para a conexão RT1 e 5 para a conexão RT30, sendo a diferença casual. As raias do histograma da conexão RT30 apresentam um *jitter* ligeiramente maior porque, sendo servida no final da supertrama, a posição da interrogação é afectada pela variação no comprimento das mensagens de dados das conexões anteriores.

A Figura 6.7 apresenta a distribuição dos atrasos da conexão RT1 utilizando o algoritmo B para gerir as retransmissões, enquanto a Figura 6.8 apresenta a distribuição relativa à conexão RT30. Com este algoritmo, consegue-se diminuir o atraso médio em relação ao observado pelo algoritmo A, porém, a variação do atraso, que é uma métrica mais relevante no transporte de tráfego de tempo real, é semelhante nos dois casos.

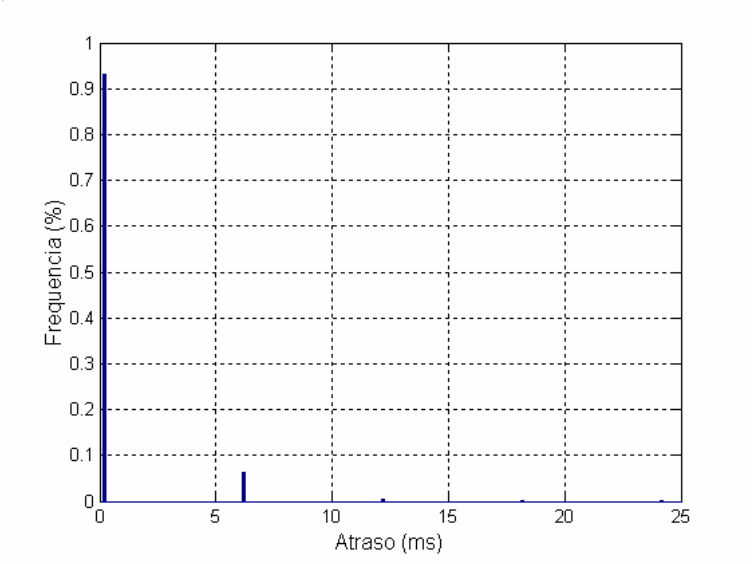


Figura 6.7: Distribuição dos atrasos para a conexão RT1 com erros no canal (BER = 10^{-4}) - Algoritmo B.

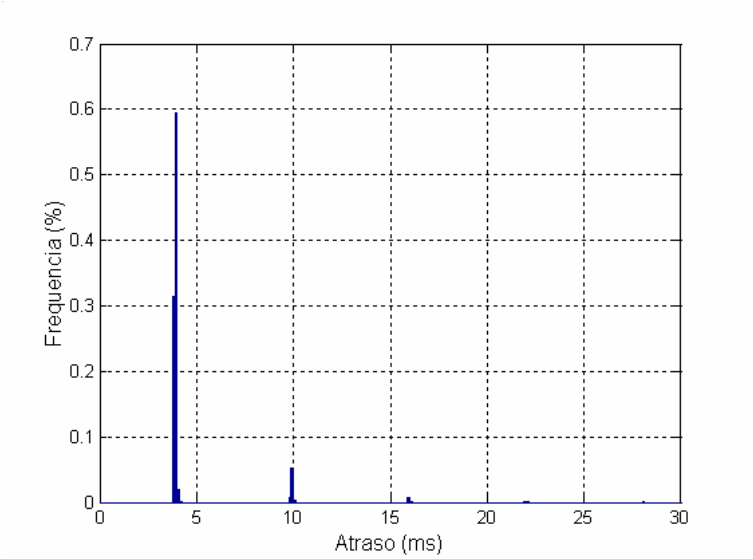


Figura 6.8: Distribuição dos atrasos para a conexão RT30 com erros no canal (BER = 10^{-4}) - Algoritmo B.

A Figura 6.9 apresenta a distribuição dos atrasos para a conexão RT1 utilizando o algoritmo C, e a Figura 6.10 apresenta a distribuição obtida para a conexão RT30 no mesmo cenário. Como as retransmissões são feitas após o ciclo regular de interrogações, o atraso associado às células retransmitidas, em ambos os casos,

aparece nos histogramas a partir de cerca de 4 ms, ou seja, após o fim das transmissões regulares, estendendo-se até aproximadamente 5 ms.

Quando a duração máxima do período livre de contenção é atingida sem que as mensagens pendentes tenham sido retransmitidas, o processo de interrogação dos terminais é abortado, dando lugar ao período de contenção. Neste caso, as mensagens pendentes só são transmitidas no ciclo regular de interrogações da supertrama seguinte. Isto acontece com as conexões RT1 e RT30, o que faz com que os respectivos histogramas contenham amostras cujo atraso é cerca de 6 ms superior ao atraso normal das células que não são retransmitidas (representado pela maior raia do histograma).

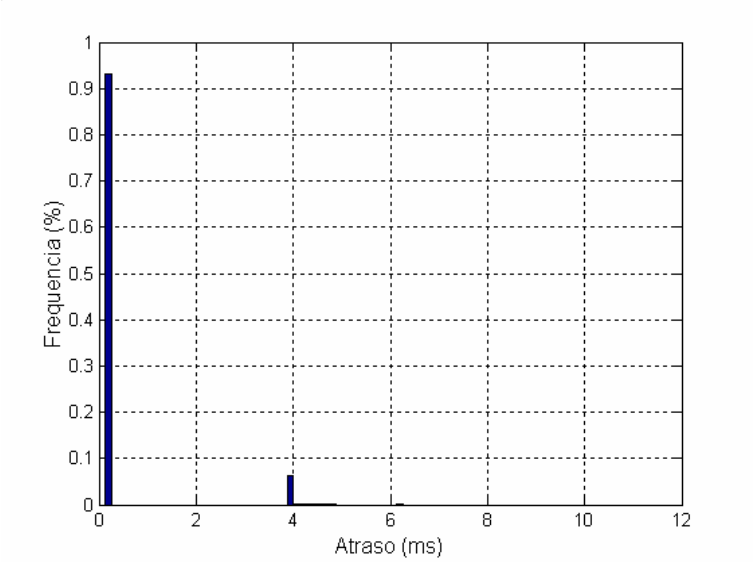


Figura 6.9: Distribuição dos atrasos para a conexão RT1 com erros no canal (BER = 10^{-4}) - Algoritmo C.

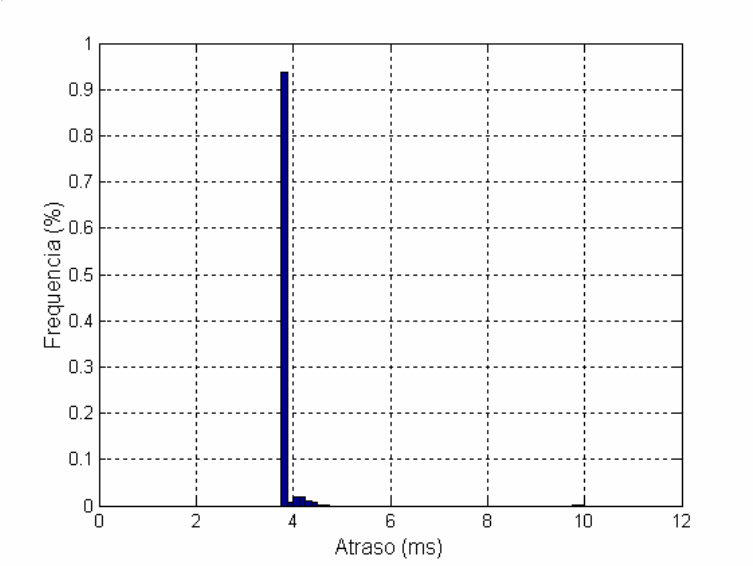


Figura 6.10: Distribuição dos atrasos para a conexão RT30 com erros no canal (BER = 10^{-4}) - Algoritmo C.

Em complemento aos resultados apresentados acima, a Tabela 3.3 apresenta o atraso médio, o desvio padrão e o atraso máximo registados para cada uma das 30 conexões CBR com os três algoritmos propostos.

Tabela 6.4: Atrasos obtidos para as 30 conexões ATM CBR com os três algoritmos implementados e uma taxa de erros binários (BER) de 10^{-4}.

Conexão	Atraso médio			Desvio padrão			Atraso máximo		
	A	B	C	A	B	C	A	B	C
1	1.04	0.59	0.40	2.46	1.74	0.98	24.18	24.23	6.16
2	1.20	0.75	0.50	2.51	1.88	0.91	30.31	36.41	4.86
3	1.33	0.88	0.64	2.49	1.86	0.91	24.46	24.53	4.86
4	1.48	1.03	0.77	2.56	1.95	0.90	24.57	24.64	4.86
5	1.55	1.13	0.89	2.56	1.89	0.88	36.74	24.74	6.67
6	1.74	1.27	0.99	2.59	1.83	0.80	36.84	18.87	6.82
7	1.82	1.37	1.13	2.48	1.82	0.81	25.04	31.02	4.73
8	1.98	1.52	1.22	2.52	1.84	0.73	19.17	25.12	4.86
9	2.10	1.66	1.36	2.50	1.87	0.73	25.23	19.35	4.86
10	2.19	1.75	1.48	2.44	1.76	0.71	31.38	19.43	4.73
11	2.38	1.93	1.59	2.57	1.94	0.67	31.46	37.55	4.73
12	2.45	2.02	1.71	2.47	1.82	0.63	25.63	25.63	4.86
13	2.65	2.19	1.83	2.56	1.89	0.61	31.71	19.83	4.86
14	2.69	2.27	1.95	2.35	1.74	0.58	19.96	19.89	4.86
15	2.87	2.44	2.08	2.51	1.91	0.56	32.02	32.04	4.73
16	3.05	2.58	2.19	2.59	1.91	0.53	26.17	26.14	8.07
17	3.13	2.69	2.31	2.52	1.94	0.49	26.25	44.37	4.73
18	3.23	2.80	2.43	2.49	1.80	0.46	32.35	26.45	4.86
19	3.42	2.98	2.55	2.64	2.01	0.43	44.50	32.62	4.73
20	3.53	3.10	2.67	2.61	2.00	0.40	38.70	44.75	4.73
21	3.69	3.23	2.78	2.57	1.94	0.37	20.85	26.81	8.71
22	3.77	3.32	2.91	2.54	1.83	0.34	26.93	32.98	4.86
23	3.94	3.49	3.02	2.54	1.93	0.31	27.08	27.06	4.86
24	4.07	3.60	3.15	2.54	1.87	0.28	27.19	33.19	4.86
25	4.13	3.72	3.26	2.43	1.83	0.25	27.32	21.39	4.86
26	4.33	3.85	3.38	2.57	1.83	0.23	33.37	27.47	9.37
27	4.43	3.99	3.50	2.46	1.97	0.20	27.59	51.71	9.48
28	4.60	4.14	3.62	2.58	1.97	0.18	27.75	39.89	9.60
29	4.71	4.26	3.74	2.49	1.87	0.14	27.80	27.90	4.86
30	4.84	4.37	3.86	2.60	1.86	0.15	34.02	28.02	9.91
Todas	**2.94**	**2.50**	**2.13**	**2.77**	**2.19**	**1.19**	**44.50**	**51.71**	**9.91**

Uma particularidade do algoritmo C, que pode ser observada na tabela, é que o desvio padrão tende a ser menor para as últimas conexões da lista de *polling*, pois a região em que são feitas as retransmissões na supertrama está mais próxima da posição em que são feitas as transmissões regulares destas conexões, enquanto que nos outros algoritmos os resultados são semelhantes para todas as conexões.

Quanto ao objectivo de proporcionar a retransmissão rápida dos dados corrompidos, os resultados mostram que o algoritmo C é francamente superior aos demais. A utilização do canal com o algoritmo C (69.3 %) é um pouco maior do que com os algoritmos A (65.6 %) e B (65.7 %), porque o *overhead* associado ao algoritmo C é maior, já que neste caso a retransmissões implicam interrogações extras aos terminais, enquanto que nos outros casos os terminais são interrogados apenas uma vez por trama. O aumento na utilização do canal devido às retransmissões, em relação ao cenário isento de erros apresentado na secção anterior é de 1.4 %, 1.5 % e 7.1 % para os algoritmos A, B e C, respectivamente, pelo que o *overhead* introduzido pelos algoritmos é pequeno quando comparado ao *overhead* devido ao protocolo do IEEE 802.11. Para o transporte de tráfego de tempo real, o desempenho muito superior do algoritmo C relativamente à variação do atraso compensa a sua menor eficiência na utilização do canal.

O algoritmo C foi igualmente utilizado noutras simulações, onde se variou o número de terminais. A Figura 6.11 apresenta o complemento da função cumulativa de distribuição do atraso da conexão RT1 com BER = 10^{-4}. Como se pode ver no gráfico, com 25 ou menos terminais consegue-se efectuar com sucesso as retransmissões sem exceder a duração máxima do período livre de contenção. Já com 35 terminais, o espaço disponível para as retransmissões no final do período livre de contenção é menor, o que obriga ao adiamento das retransmissões para as tramas seguintes mais vezes, aumentando o atraso máximo.

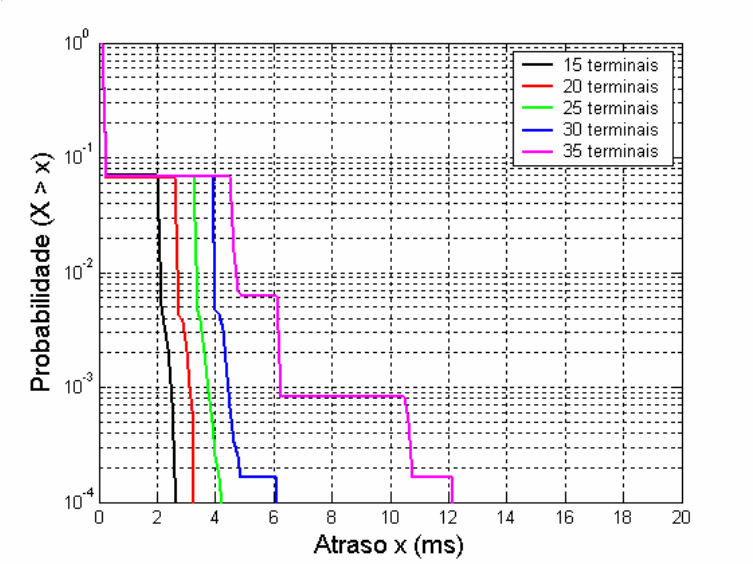


Figura 6.11: Complemento da função cumulativa de distribuição do atraso da conexão RT1 com erros no canal (BER = 10^{-4}) utilizando a função PCF e o algoritmo C.

Nas simulações efectuadas com o algoritmo C, observou-se que as conexões dos primeiros terminais na lista de *polling* têm tendência a apresentar um atraso máximo menor, o que pode ser explicado pelo facto de os primeiros terminais serem colocados à frente na fila de retransmissões. Para aumentar a justiça, uma modificação simples do algoritmo passaria pela ordenação aleatória dos terminais da fila de retransmissões. Uma modificação mais criteriosa passaria por ordenar a lista de forma a dar prioridade de transmissão aos pacotes que estão mais atrasados.

A utilização do mecanismo HCCA da norma IEEE 802.11e não traria modificações significativas nos resultados, para os cenários propostos. Uma diferença é que com o HCCA o ponto de acesso poderia continuar a interrogar terminais pertencentes à fila de retransmissões após o término do período livre de contenção e, assim, diminuir o atraso máximo das respectivas conexões, às custas da redução do espaço disponível no período de contenção para as conexões de mais baixa prioridade.

Os resultados apresentados a seguir dizem respeito à utilização do mecanismo EDCA para a transmissão do tráfego da classe I no mesmo cenário utilizado acima. No mecanismo EDCA, a retransmissão de uma mensagem de dados corrompida por erros no canal processa-se da mesma forma que no caso da perda da mensagem devido a uma colisão, pois em ambos os casos o que o terminal detecta é a não

recepção da trama ACK após um intervalo de tempo especificado. O procedimento nestes casos consiste em duplicar o valor da janela de contenção e iniciar um novo processo de *backoff*.

A Figura 6.12 apresenta o complemento da função cumulativa de distribuição do atraso das conexões da classe I utilizando o mecanismo EDCA com uma taxa de erros BER = 10^{-4} e um número variável de terminais. Os valores de atraso com 15 e 20 terminais são superiores aos observados no cenário sem erros no canal (Figura 6.4), como era esperado, devido às retransmissões, mas ainda são relativamente pequenos. No entanto, a partir de 25 terminais o sistema fica instável, com o atraso das conexões a subir continuamente sem limite.

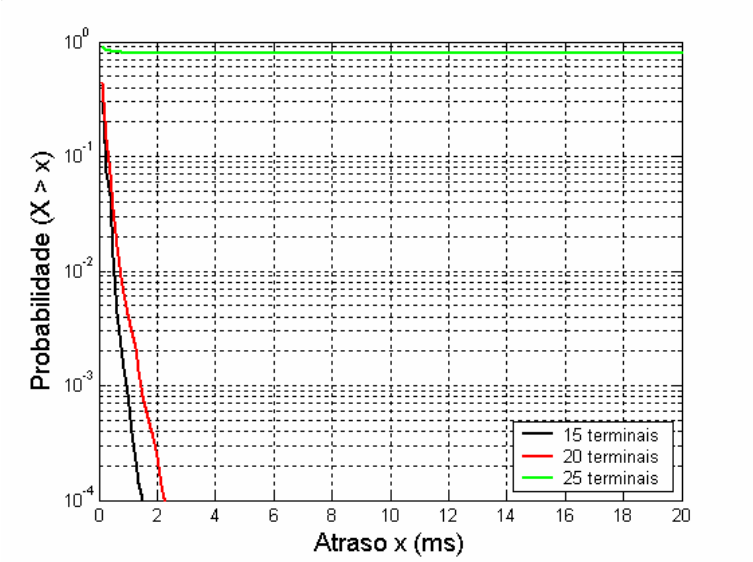


Figura 6.12: Complemento da função cumulativa de distribuição do atraso das conexões da classe I utilizando o mecanismo EDCA (*AIFS[I]* = *DIFS*, $CW_{min}[I] = 3$, $CW_{max}[I] = 7$) com BER = 10^{-4}.

6.2.4 Mistura de tráfego

Os resultados apresentados a seguir referem-se à simulação do sistema com uma mistura de tráfego de classes diferentes, utilizando o modelo de canal isento de erros. Além das 30 conexões ATM CBR presentes anteriormente (classe I_a), um número variável de conexões assíncronas (classe II), caracterizadas pelos parâmetros definidos na Tabela 6.1, foram adicionadas ao sistema.

Estes resultados são provenientes de dois conjuntos de simulações. No primeiro conjunto o tráfego de classe I é transportado usando a função PCF, durante o período livre de contenção, e o tráfego de classe II é transportado usando a função DCF, dentro do período de contenção. No segundo caso o acesso ao canal segue as especificações da norma IEEE 802.11e, substituindo-se a função PCF pelo mecanismo HCCA a função DCF pelo o mecanismo EDCA. Os parâmetros de *backoff* utilizados com o mecanismo EDCA são iguais aos da função DCF no primeiro conjunto de simulações ($AIFS[II] = DIFS$, $CW_{min}[II] = 15$ e $CW_{max}[II] = 1023$), para permitir comparar melhor os resultados, embora normalmente o valor de AIFS utilizado para o tráfego de baixa prioridade seja maior.

A Figura 6.13 apresenta o débito agregado para cada classe de tráfego em função do número de conexões assíncronas (classe II), tanto para o sistema baseado no IEEE 802.11 original como para o sistema que utiliza o IEEE 802.11e. Nos dois casos, o débito das conexões da classe I mantém-se constante, mesmo quando a carga total oferecida excede a capacidade disponível na rede.

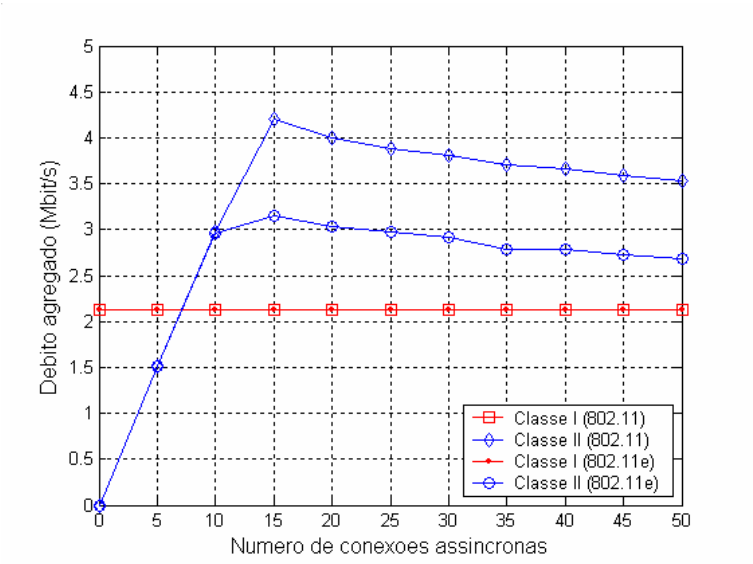


Figura 6.13: Débito agregado das diferentes classes de tráfego em função do número de conexões assíncronas.

Por outro lado, em ambos os casos, o débito agregado das conexões assíncronas atinge o valor máximo em torno de 15 conexões. A partir dessa região no gráfico, o débito chega mesmo a diminuir com o aumento da carga oferecida, pois o número de

terminais que competem pelo acesso ao meio aumenta. Com isso, uma parcela maior do tempo disponível durante o período de contenção (CP) é desperdiçada em colisões entre as mensagens de dados e outra parte deixa de ser aproveitada devido ao aumento do período de *backoff* provocado pelas colisões, diminuindo o tempo efectivamente utilizado para transmissão de dados. Apesar deste desperdício de tempo, não verificado com a função PCF ou o mecanismo HCCA, a eficiência máxima obtida no transporte do tráfego da classe II (com 15 terminais activos) é de cerca de 66 %[61] no primeiro caso e de 49.6 % no segundo, sendo bastante superior à obtida no transporte do tráfego da classe I (18.2 %), devido ao maior comprimento do *payload* das mensagens de dados do tráfego da classe II no cenário proposto.

O débito máximo das conexões assíncronas no caso do IEEE 802.11e é inferior ao registado no caso do IEEE 802.11. O motivo é que no IEEE 802.11e uma estação não deve iniciar uma transmissão se não puder completá-la antes do instante previsto para o início de transmissão da próxima trama *Beacon* (TBTT), pelo que quando isso acontece há um espaço no final do período de contenção que não é aproveitado. A diferença entre os débitos obtidos neste cenário de simulação é significativa porque o comprimento definido para os pacotes da classe II é constante e relativamente grande. Caso fosse utilizado um modelo de tráfego com pacotes de comprimento variável para a classe II, o espaço no final do período de contenção poderia ser aproveitado para a transmissão de pacotes mais curtos, pelo que a menor eficiência do mecanismo EDCA neste cenário não se afigura relevante.

Dado que o comprimento definido para os pacotes do tráfego assíncrono é grande, a utilização do mecanismo de RTS/CTS poderia melhorar o desempenho do sistema relativamente ao transporte deste tráfego, ao diminuir o tempo perdido em caso de colisão. Porém, os potenciais benefícios desse mecanismo não foram explorados, visto terem sido privilegiados cenários de tráfego de tempo real.

Quanto ao efeito das conexões assíncronas sobre o atraso das conexões de tempo real, a Figura 6.14 apresenta o atraso médio e desvio padrão do atraso da conexão

[61] Este valor foi calculado usando a equação 5.2, sendo que a utilização do canal (*U*) é expressa pela parcela do tempo não ocupada pelo período livre de contenção.

RT1 em função do número de conexões assíncronas, nos dois casos. No caso da utilização do IEEE 802.11 original, o atraso tende a aumentar com o aumento do número de conexões assíncronas devido ao efeito de alargamento do período de contenção (*stretching*), descrito na secção 3.6.3.1.4. A partir de certo ponto no gráfico o valor do atraso estabiliza, porque a extensão do período de contenção anterior sobre o período livre de contenção é limitada pelo comprimento dos pacotes transportados utilizando a função DCF e pelo débito de transmissão utilizado. No entanto, o aumento do atraso poderia ser significantemente maior noutro cenário, nomeadamente se o débito de transmissão fosse menor ou fosse utilizada fragmentação. Se o encurtamento do período livre de contenção fosse muito severo, até mesmo o débito oferecido às conexões da classe I poderia ser comprometido, se o tempo restante não fosse suficiente para a interrogação de todas as estações da lista de *polling*.

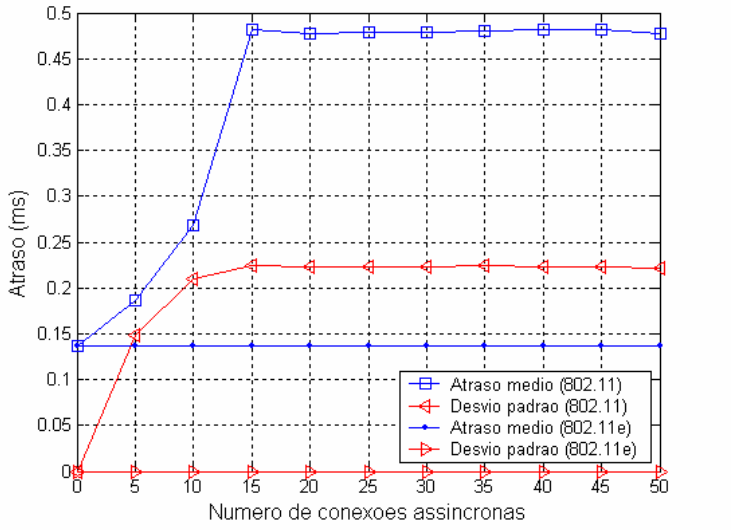


Figura 6.14: Atraso médio e desvio padrão do atraso da conexão RT1 em função do número de conexões assíncronas.

No caso de utilização dos mecanismos de acesso ao meio da norma IEEE 802.11e, o atraso sofrido pelas conexões da classe I é constante, não sendo afectado pelo tráfego assíncrono, pois este não pode invadir o período livre de contenção seguinte, o que é uma vantagem significativa em relação ao IEEE 802.11 original. Refira-se, porém, que esta vantagem não existirá caso façam parte da rede estações antigas que não reconheçam as especificações da norma IEEE 802.11e.

A utilização do mecanismo EDCA para o transporte de uma mistura do tráfego da classe I e da classe II não foi considerado aqui porque os resultados obtidos no transporte do tráfego de classe I isoladamente foram insatisfatórios. De qualquer forma, é de se esperar que o tráfego de classe II tenha influência sobre o atraso do tráfego de classe I, pois durante a transmissão do tráfego de classe II as conexões de classe I que desejem iniciar uma transmissão têm que aguardar que o meio fique livre, embora a demora na liberação do meio possa ser limitada com o uso do parâmetro de duração da oportunidade de transmissão (TXOP) da categoria de acesso associada à classe II. O tráfego de classe II também contribui para o aumento da carga na rede e a probabilidade de colisão, com a consequente degradação da qualidade de serviço oferecida ao tráfego de classe I. Para minimizar a influência do tráfego de baixa prioridade sobre o tráfego prioritário, pode-se fazer:

$$AIFS[II] > AIFS[I] + CW_{\max}[I] \tag{6.4}$$

No entanto, esta abordagem faz com que em caso de saturação da rede o tráfego de classe II fique impossibilitado de aceder ao meio (*starvation*), o que não ocorre com a utilização de um dos mecanismos centralizados do IEEE 802.11 para o transporte do tráfego de classe I, pois as conexões de classe II dispõem do espaço reservado para o período de contenção para transmissão.

6.3 Sistema baseado na rede HIPERLAN/2

6.3.1 Parâmetros de simulação

Os parâmetros de simulação associados à rede HIPERLAN/2 foram divididos em duas categorias, como foi feito anteriormente para a rede IEEE 802.11. A Tabela 6.5 apresenta os parâmetros fixos utilizados na simulação da rede HIPERLAN/2 e os seus valores conforme encontram-se definidos pelas especificações.

Tabela 6.5: Parâmetros fixos da rede HIPERLAN/2.

Descrição	Designação	Valor
Comprimento do canal BCH	*BCHLength*	15x8 bits
Comprimento de um bloco de IEs	*IEBlockLength*	27x8 bits
Comprimento do canal ACH	*ACHLength*	9x8 bits
Comprimento do canal LCH	*LCHLength*	54x8 bits
Comprimento do canal SCH	*SCHLength*	9x8 bits
Comprimento do canal RCH	*RCHLength*	9x8 bits
Duração da trama do HIPERLAN/2	*FrameTime*	2 ms
Duração do preâmbulo do canal BCH	*BCHPreTime*	16 μs
Duração do preâmbulo da ligação descendente	*DLPreTime*	8 μs
Débito de transmissão mínimo	*MinRate*	6 Mbit/s

Os valores utilizados para os parâmetros configuráveis da rede HIPERLAN/2 em todas as simulações realizadas, salvo indicação em contrário, são apresentados na Tabela 6.6.

Tabela 6.6: Parâmetros configuráveis da rede HIPERLAN/2 utilizados nas simulações.

Descrição	Designação	Valor
Débito de transmissão dos canais LCH	*LCHRate*	18 Mbit/s
Débito de transmissão dos canais SCH	*SCHRate*	6 Mbit/s
Período de guarda da ligação ascendente	*ULGuardTime*	2 μs
Duração do preâmbulo da ligação ascendente	*ULPreTime*	16 μs
Turn around time	*TurnTime*	6 μs

Na simulação do sistema baseado na rede HIPERLAN/2, a atribuição de recursos para o tráfego da classe I é feita com base no mecanismo de negociação de capacidade fixa (FCA) do HIPERLAN/2, salvo indicação em contrário. Já no caso do tráfego da classe II, os adaptadores de terminal transmitem mensagens de requisição de recursos

durante a fase de acesso aleatório, à medida que novos dados chegam aos seus *buffers* de pacotes.

As camadas de convergência baseadas em células e em pacotes são utilizadas, respectivamente, para o transporte do tráfego ATM (classe I_a) e IP (classe II). Já o tráfego RDIS (classe I_b) utiliza uma camada de convergência definida por nós.

6.3.2 Tráfego prioritário sem erros no canal

Os resultados apresentados a seguir são referentes à operação do sistema apenas com o tráfego da classe I_a, correspondendo a 30 conexões ATM CBR. O primeiro terminal (com endereço mais baixo) recebeu a denominação MT1 e a conexão que lhe está associada foi denominada RT1. O mecanismo de negociação de capacidade fixa (FCA) do HIPERLAN/2 é utilizado na reserva de recursos para essas conexões. O ponto de acesso aloca um canal de transporte LCH para cada conexão por cada três tramas, ou seja, o intervalo de atribuição de canais LCH pelo mecanismo FCA (*FCInterval*) é igual a 6 ms, sendo igual ao intervalo de geração de pacotes das conexões para minimizar o *jitter*. De forma a distribuir a carga entre as tramas, cada trama transporta 10 conexões, sendo que a alocação dos canais foi feita pela ordem dos seus endereços. Para facilitar a visualização dos resultados, todas as conexões começam a gerar tráfego ao mesmo tempo, a partir do instante inicial de simulação ($t = 0$).

A Figura 6.16 apresenta a distribuição dos atrasos para 3 das 30 conexões presentes na simulação[62] num cenário sem erros no canal, utilizando o modo sem reconhecimento do HIPERLAN/2. Como a conexão associada ao primeiro terminal (RT1) é a primeira a ser servida, o seu atraso é menor do que o das outras conexões. A conexão RT10 é a última a ser servida na mesma trama, enquanto a conexão RT30 é a última a ser servida num mesmo ciclo de três tramas, pelo que ela sofre o maior atraso entre as conexões. Num caso mais genérico, o atraso das diferentes conexões pode situar-se entre um mínimo próximo de zero e um máximo de cerca de 6 ms,

[62] As demais conexões apresentam resultados similares.

dependendo da diferença entre o instante de transmissão determinado pelo ponto de acesso e o instante de geração do pacote, que não precisa ser idêntico para todas as conexões como neste exemplo.

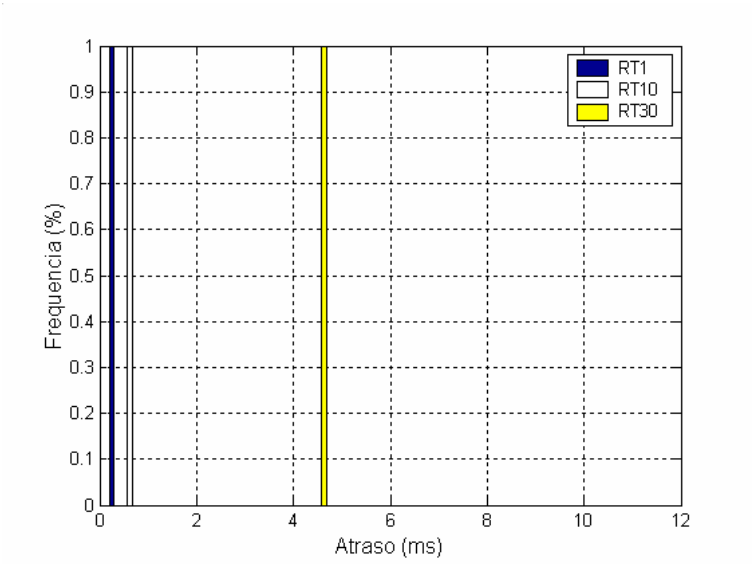


Figura 6.15: Distribuição dos atrasos de três conexões da classe I_a utilizando o modo sem reconhecimento do HIPERLAN/2 (sem erros no canal).

Em termos de utilização do canal, a fase de difusão ocupa, no mínimo, 4.2 % do espaço disponível nas tramas, como pode ser calculado com seguinte expressão:

$$UDifMin = \frac{BCHPreTime + \frac{BCHLength + IEBlockLength + ACHLength}{MinRate}}{FrameTime} \quad (6.5)$$

A utilização mínima dá-se quando o canal FCH é composto por um único bloco de elementos de informação, no entanto, as mensagens de atribuição de recursos (RG) associadas às conexões fazem com que o comprimento da fase de difusão aumente.

Já na fase de acesso aleatório, a utilização do canal é, no mínimo, 1.5 % do espaço disponível nas tramas, quando cada trama contém um único canal RCH:

$$URCHMin = \frac{ULGuardTime + ULPreTime + \frac{RCHLength}{MinRate}}{FrameTime} \quad (6.6)$$

No total, o *overhead* fixo (Oh_f) associado às fases de difusão e de acesso aleatório ocupa 5.7 % da trama do HIPERLAN/2:

$$Oh_f = UDifMin + URCHMin \tag{6.7}$$

As 30 conexões ATM CBR representam um incremento na utilização do canal de 26.7 % neste cenário, que se distribui pela ocupação da fase ascendente para transmissão de dados, incluindo os preâmbulos e períodos de guarda, e pela utilização do canal FCH para sinalização da atribuição de recursos (RG) às conexões.

A Figura 6.16 apresenta a distribuição dos atrasos no mesmo cenário anterior, mas desta vez utilizando o modo de reconhecimento do HIPERLAN/2. Neste caso, o sistema está configurado de modo que o ponto de acesso envie uma mensagem de ARQ por cada 64 mensagens de dados consecutivas recebidas de uma conexão para permitir que o respectivo adaptador de terminal esvazie o seu *buffer* de mensagens e avance a sua janela de transmissão.

Como mostra a figura, quase todos os pacotes de uma mesma conexão apresentam um atraso constante, porém, uma pequena percentagem dos pacotes sofre um atraso ligeiramente maior, causado pelas mensagens de ARQ enviadas pelo ponto de acesso. Uma parte substancial deste aumento é decorrente da alocação de canais de transporte SCH na fase descendente para o transporte das mensagens de ARQ; a outra parte resulta das mensagens de atribuição de recursos (RG) adicionais inseridas no canal FCH. A sincronização entre as diversas conexões ao nível da geração de pacotes faz com que o efeito seja excepcionalmente pronunciado, embora o impacto sobre o *jitter* não seja relevante. Este efeito não foi observado com a rede IEEE 802.11 porque nesta o reconhecimento de uma mensagem de dados enviada por um terminal é anexado à mensagem CF-Poll seguinte, em vez de ser enviado numa mensagem separada.

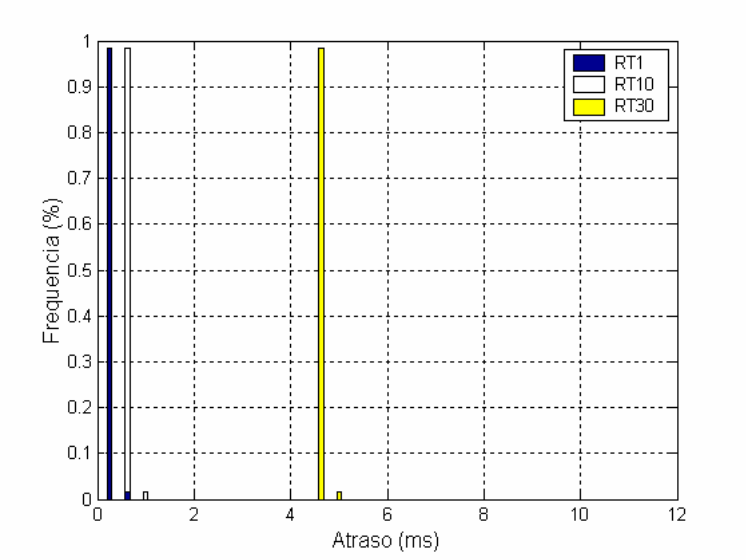


Figura 6.16: Distribuição dos atrasos de três conexões da classe I_a utilizando o modo de reconhecimento do HIPERLAN/2 (sem erros no canal).

A utilização do canal pelas 30 conexões CBR neste cenário é de 26.9 %, sendo repartida pelo aumento do comprimento do canal FCH (mensagens de atribuição de recursos), pela fase descendente (mensagens de ARQ) pela fase ascendente (mensagens de dados). Em relação ao cenário anterior, em que o modo sem reconhecimento era utilizado, o aumento na utilização do canal é de apenas 0.75 %, pelo que o *overhead* introduzido pela transmissão das mensagens de ARQ não é significativo neste caso.

Descontando-se o *overhead* fixo (5.7 %) e a utilização do canal pelas conexões da classe I (26.9 %), ainda sobra, em média, 67.4 % do espaço na trama, que pode ser utilizado para o transporte do tráfego das conexões de baixa prioridade e para o aumento da fase de acesso aleatório. A eficiência no transporte do tráfego da classe I é de 41.3 %, tendo já em conta a parcela que diz respeito ao *overhead* fixo (Oh_f), sendo muito superior à obtida com a rede IEEE 802.11a com o mesmo tráfego (18.2 %). Um factor que contribui para a maior eficiência do HIPERLAN/2 é a menor dimensão do cabeçalho das mensagens de dados. Além disso, o *payload* das mensagens transporta uma versão comprimida célula ATM, devido à utilização da camada de convergência baseada em células, enquanto que no IEEE 802.11 a célula é transportada integralmente.

Devido ao baixo débito das conexões da classe I, os trens de PDUs na fase ascendente são compostos por um único canal LCH, pelo que o *overhead* introduzido pelo período de guarda e preâmbulo, na fase ascendente, e pelas mensagens de atribuição de recursos (RG), no canal FCH, impedem que a eficiência obtida com o HIPERLAN/2 seja maior.

6.3.3 Tráfego prioritário com erros no canal

Os próximos resultados foram obtidos com erros no canal. Estes resultados são expressos em função da taxa de erros binários (BER) ao nível da camada de ligação de dados, que tende a ser menor do que a taxa de erros ao nível da camada física do HIPERLAN/2, pois esta implementa um mecanismo de correcção de erros[63]. Para recuperação dos dados corrompidos por erros é utilizado o mecanismo de retransmissão rápida proposto para a rede HIPERLAN/2, descrito na secção 4.2.2.1. O cenário é idêntico ao utilizado na secção anterior, excepto pela introdução de erros no canal. As distribuições dos atrasos são apresentadas para a conexão RT1, pois os resultados relativos às outras conexões são semelhantes.

A Figura 6.17 apresenta a distribuição dos atrasos obtidos para a conexão RT1 com uma taxa de erros binários (BER) igual a 10^{-5}. Como as mensagens de ARQ das diferentes conexões deixam de estar sincronizadas, não aparece um pico relacionado com estas mensagens no histograma como acontece na Figura 6.16. Neste caso, o pico que aparece entre 2 e 4 ms deve-se à retransmissão das mensagens corrompidas, na trama seguinte. Da mesma forma, a Figura 6.18 apresenta os resultados obtidos para a conexão RT1 com BER = 10^{-4}. A utilização do canal nestes casos é de 27.0 % e 28.2 %, respectivamente.

[63] Similar ao utilizado pela rede IEEE 802.11a.

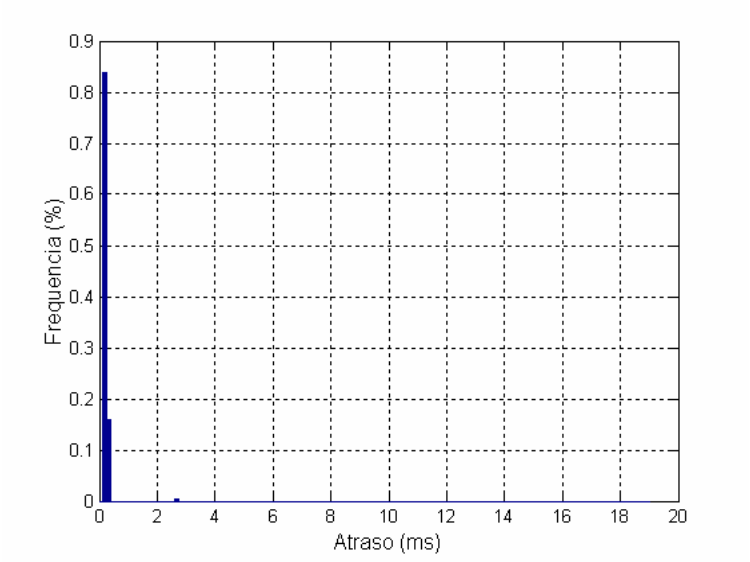


Figura 6.17: Distribuição dos atrasos para a conexão RT1 com erros no canal (BER = 10^{-5}).

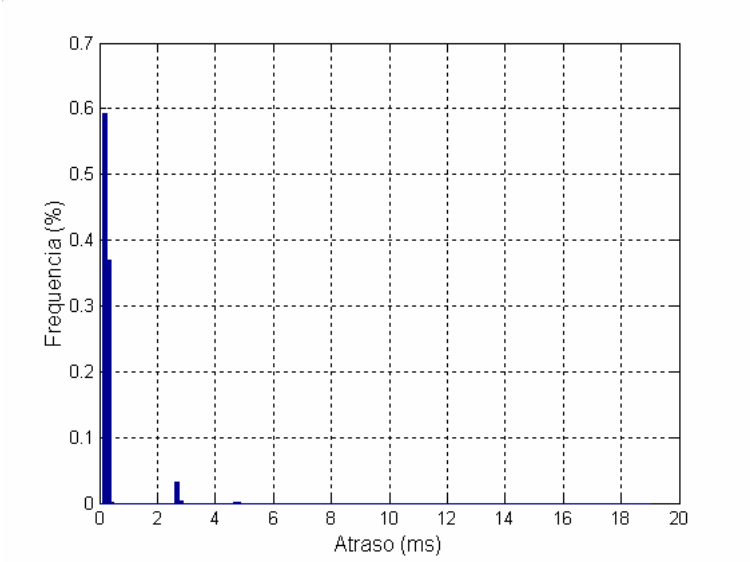


Figura 6.18: Distribuição dos atrasos para a conexão RT1 com erros no canal (BER = 10^{-4}).

A equação 4.5 permite calcular de forma aproximada a probabilidade da variação do atraso ultrapassar um dado limite, podendo ser aplicada ao cenário com a taxa de erros binários BER = 10^{-4}, a que corresponde a taxa de erros de pacote (PER) de 4.23×10^{-2}, de acordo com a equação 4.3. Assim, a título de exemplo, a probabilidade da variação do atraso ser superior a 6 ms (que implica um número de retransmissões $n \geq 3$), é igual a 7.6×10^{-5}, enquanto a probabilidade da variação ser superior a 8 ms ($n \geq 4$) é igual a 3.2×10^{-6}, e assim por diante.

Os resultados apresentados acima indicam que, perante uma taxa de erros moderada, o atraso pode ser mantido sob controlo se forem utilizados os procedimentos definidos no algoritmo de retransmissão rápida, concebidos de forma a limitar a variação do atraso. O acréscimo na utilização do canal no cenário com BER = 10^{-4}, em relação ao cenário sem erros, é de 4.8 %, pelo que o *overhead* associado às retransmissões com este algoritmo é pequeno.

A Figura 6.19 apresenta a distribuição dos atrasos para a conexão RT1 com uma taxa de erros binários BER = 10^{-3}. Neste cenário, o número de retransmissões é elevado, provocando uma variação significativa do atraso, além de ter um impacto considerável na utilização da largura de banda disponível, visto que a utilização do canal pelas conexões é, neste caso, de 48.5 %, o que representa um aumento de 80.3 % em relação ao cenário sem erros.

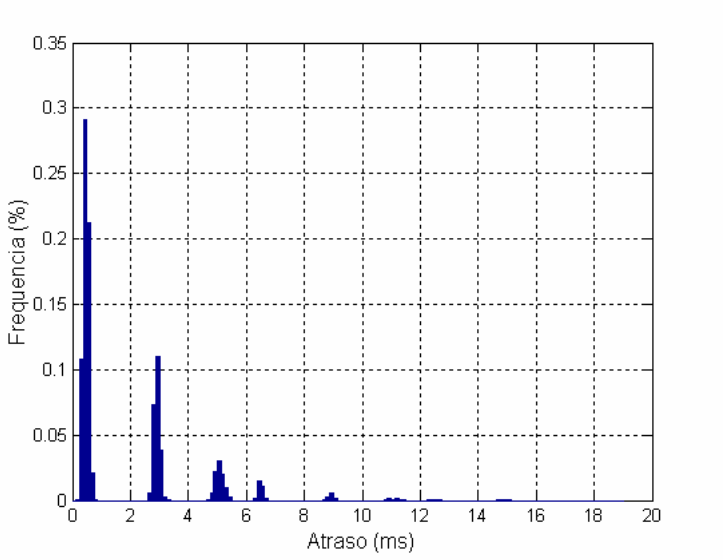


Figura 6.19: Distribuição dos atrasos para a conexão RT1 com erros no canal e com débito dos canais LCH de 18 Mbit/s (BER = 10^{-3}).

Como a taxa de erros de pacote (PER) pode ser reduzida significativamente com a utilização de um modo de transmissão inferior, simulou-se o comportamento do sistema utilizando um débito de transmissão de 6 Mbit/s para os canais LCH. Para estimar a redução na taxa de erros de pacote, foram utilizados os resultados apresentados na Figura 3a da referência [DOU02], na qual uma taxa de erros binários BER = 10^{-3} relativa à utilização do modo de transmissão 4 (18 Mbit/s) corresponde a

uma taxa de erros BER ≅ $3x10^{-5}$ com a utilização do modo 1 (6 Mbit/s), sob as mesmas condições no canal.

A Figura 6.20 apresenta a distribuição dos atrasos para a conexão RT1 utilizando o modo 1 e com uma taxa de erros binários igual a $3x10^{-5}$. O número de retransmissões e, consequentemente, a variação do atraso, são substancialmente reduzidos em relação ao cenário da Figura 6.19.

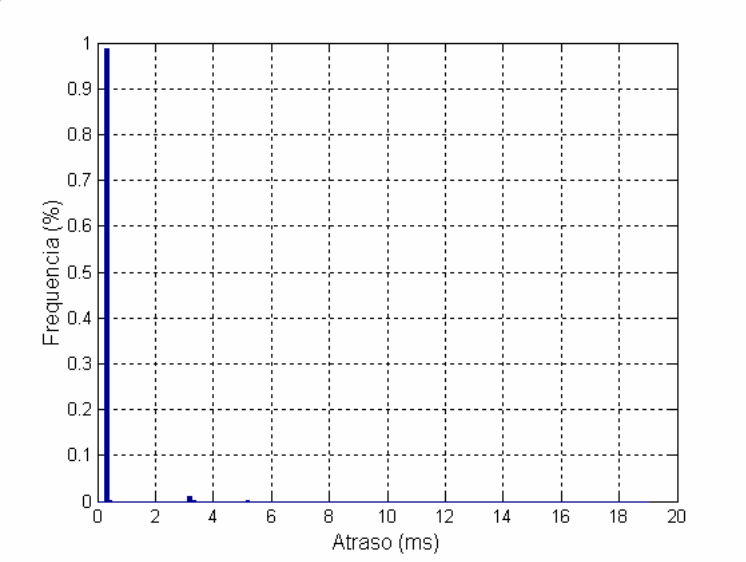


Figura 6.20: Distribuição dos atrasos para a conexão RT1 com erros no canal e com débito dos canais LCH de 6 Mbit/s (BER = $3x10^{-5}$).

O decréscimo no número de retransmissões devido ao uso de um modo de transmissão mais robusto diminui o *overhead* associado à retransmissão das mensagens de dados na fase ascendente, bem como o número de mensagens de ARQ transmitidas na fase descendente[64]. Porém, isto não é suficiente para compensar o incremento do *overhead* causado pelo aumento do tempo de transmissão das mensagens de dados, pois a utilização média das tramas com as conexões CBR aumenta de 48.5 %, com o modo 4, para 51.6 %, com o modo 1.

[64] Isso é confirmado pela redução no número médio de elementos de informação por trama, de 20.23 para 10.36.

6.3.3.1 Modelo de erros de Gilbert-Elliot

Os resultados apresentados a seguir foram obtidos utilizando o modelo de erros de Gilbert-Elliot. O estado *bom* foi caracterizado por um tempo médio de permanência $\overline{T}_{bom}$ = 100 ms e uma taxa de erros binários BER_{bom} = 0, enquanto os mesmos parâmetros para o estado *mau* são $\overline{T}_{mau}$ = 33 ms e $BER_{mau} = 10^{-3}$ (no modo 4). Com estes parâmetros, a taxa de erros binários média (BER_{med}) calculada pela equação 5.1 é igual a 2.48×10^{-4}.

A Figura 6.21 apresenta a distribuição dos atrasos para a conexão RT1 utilizando o modo 4 para transmissão de dados nos canais LCH e com os parâmetros do modelo de Gilbert-Elliot definidos acima. Os resultados mostram que, em relação ao cenário da Figura 6.19, a variação do atraso diminui muito pouco, apesar de a taxa de erros binários média ter diminuído significativamente. Isso acontece porque quando o estado *mau* perdura por várias tramas, como é o caso, os pacotes transmitidos durante esse período são afectados de forma idêntica ao caso em que o estado *mau* está presente o tempo todo.

Por outro lado, quando o sistema está no estado *bom*, não há retransmissão de mensagens, pelo que as raias do histograma associadas às retransmissões têm as suas intensidades reduzidas. Com isso, o atraso médio diminui, bem como a utilização do canal, que desce para 31.5 %. Entretanto, a variação do atraso mantém-se alta, o que é indesejável para o transporte de tráfego de tempo real.

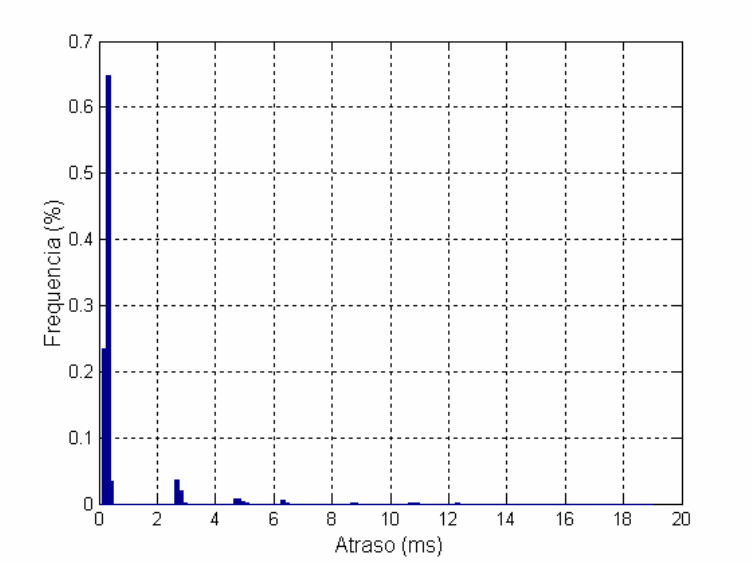


Figura 6.21: Distribuição dos atrasos para a conexão RT1 com modelo de Gilbert-Elliot e com débito dos canais LCH de 18 Mbit/s.

A Figura 6.22 apresenta a distribuição dos atrasos para a conexão RT1 no mesmo cenário da Figura 6.21, com a diferença de que o débito de transmissão nos canais LCH passa de 18 Mbit/s (modo 4) para 6 Mbit/s (modo 1). Com isso, a taxa de erros binários utilizada no estado *mau* passa de 10^{-3} para $3x10^{-5}$, de modo a representar o aumento da robustez das transmissões.

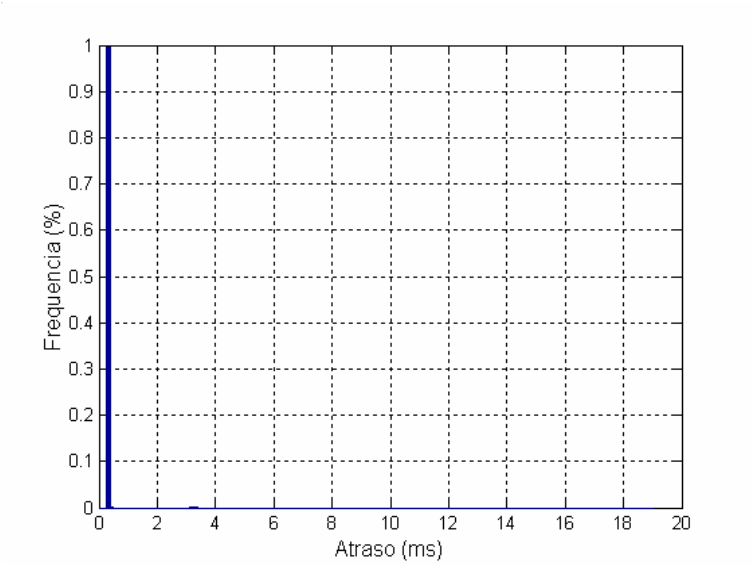


Figura 6.22: Distribuição dos atrasos para a conexão RT1 com modelo de Gilbert-Elliot e com débito dos canais LCH de 6 Mbit/s.

A variação do atraso obtida neste caso é significativamente inferior, porém, o custo disto é que a utilização do canal passa de 31.5 % para 51.0 %, quase igual à obtida no cenário em que o canal permanecia no estado *mau* durante todo o tempo (51.6 %). O que acontece neste caso é que na maior parte do tempo o sistema não precisa de utilizar um modo de transmissão tão robusto, que consome muita largura de banda, pois a qualidade do canal é suficientemente boa para que seja pequena a probabilidade das mensagens serem corrompidas com a utilização de um débito de transmissão mais elevado. Assim, o ideal é utilizar um sistema que adapte o débito de transmissão às condições do canal.

Na Figura 6.23, apresenta-se a distribuição dos atrasos para a conexão RT1 nas mesmas condições dos dois cenários anteriores, porém, utilizando o esquema de adaptação do débito de transmissão proposto na secção 4.2.2.2. Neste cenário, enquanto as transmissões são feitas usando o modo 4 (18 Mbit/s), as retransmissões são feitas usando o modo 1 (6 Mbit/s), excepto quando a retransmissão de uma mensagem acontece na mesma trama em que é feita a transmissão de uma nova mensagem da mesma conexão, caso em que é usado o modo de transmissão da nova mensagem.

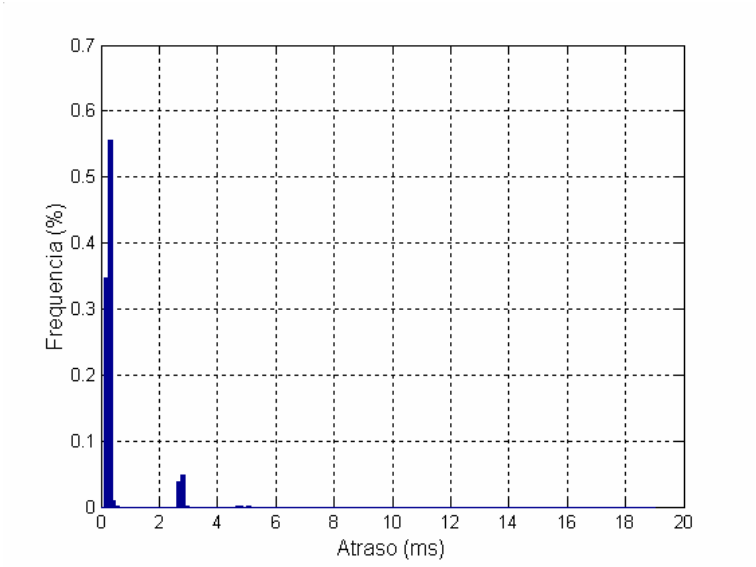


Figura 6.23: Distribuição dos atrasos para a conexão RT1 com modelo de Gilbert-Elliot e com débito dos canais LCH adaptativo.

A variação do atraso com este esquema é maior que a obtida com a utilização do modo 1 durante toda a simulação. No entanto, enquanto que no caso anterior a utilização do canal foi de 51.0 %, neste caso a utilização reduz-se para 32.0 %. Este valor é muito próximo do obtido com a utilização do modo 4 (31.5 %) durante toda a simulação, no entanto, a variação do atraso é muito menor. A explicação para este desempenho reside no facto do aumento do *overhead* com a retransmissão das mensagens ao débito mais baixo ser compensado pela diminuição do número de retransmissões e de mensagens de ARQ. Assim, o esquema proposto consegue reduzir a variação do atraso sem aumentar a utilização do canal.

6.3.3.2 Sumário

A Tabela 6.7 contém um sumário dos resultados obtidos até aqui para as conexões ATM CBR com diferentes condições de erros no canal, diferentes modos de transmissão, e a utilização do modo de reconhecimento do HIPERLAN/2 em conjunto com o algoritmo de retransmissão rápida para o controlo de erros. Em relação ao número médio de elementos de informação (IE) por trama, 10 elementos são referentes à alocação de recursos usando o mecanismo de negociação de capacidade fixa (FCA); enquanto os restantes elementos dizem respeito quer aos canais LCH alocados para as retransmissões, quer aos canais SCH alocados para a transmissão de mensagens de ARQ.

Tabela 6.7: Sumário dos resultados obtidos com as conexões ATM CBR no sistema baseado na rede HIPERLAN/2.

BER no Modo 4	Modo de Transmissão	Utilização do canal (%)	Número médio de IEs	Distribuição dos atrasos
0	4	26.9	10.16	Figura 6.16
10^{-5}	4	27.0	10.22	Figura 6.17
10^{-4}	4	28.2	10.90	Figura 6.18
10^{-3}	4	48.5	20.23	Figura 6.19
10^{-3}	1	51.6	10.36	Figura 6.20
$0/10^{-3}$	4	31.5	12.47	Figura 6.21
$0/10^{-3}$	1	51.0	10.21	Figura 6.22
$0/10^{-3}$	Adaptativo[65]	32.0	11.77	Figura 6.23

6.3.4 Mistura de tráfego

Os resultados apresentados a seguir referem-se à simulação do sistema com uma mistura de tráfego de duas classes diferentes. Além das 30 conexões ATM CBR (classe I_a) presentes nas simulações anteriores, foi inserido no sistema um número variável de conexões assíncronas (classe II), cujos parâmetros estão definidos na Tabela 6.1

A Figura 6.24 apresenta o débito agregado relativo a cada classe de tráfego em função do número de conexões assíncronas presentes no sistema. Como era esperado, o débito das conexões CBR mantém-se constante, mesmo quando a carga total oferecida excede a capacidade disponível na rede, pois o tráfego de tempo real (classe I) recebe tratamento prioritário do escalonador localizado no ponto de acesso, conforme o procedimento descrito na secção 5.2.6.3.

[65] Modo 4 para as transmissões, modo 1 para as retransmissões.

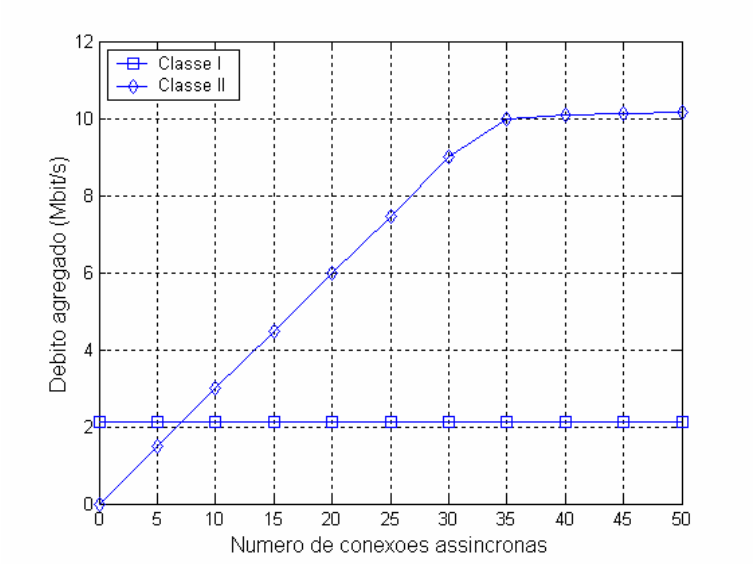


Figura 6.24: Débito agregado das diferentes classes de tráfego em função do número de conexões assíncronas presentes no sistema.

Por outro lado, o débito das conexões assíncronas atinge um ponto de saturação acima de 30 conexões. Quando a carga na rede aproxima-se do ponto de saturação, o atraso das conexões assíncronas aumenta drasticamente, como pode ser visto na Figura 6.25. O atraso continua a aumentar para além do ponto de saturação.

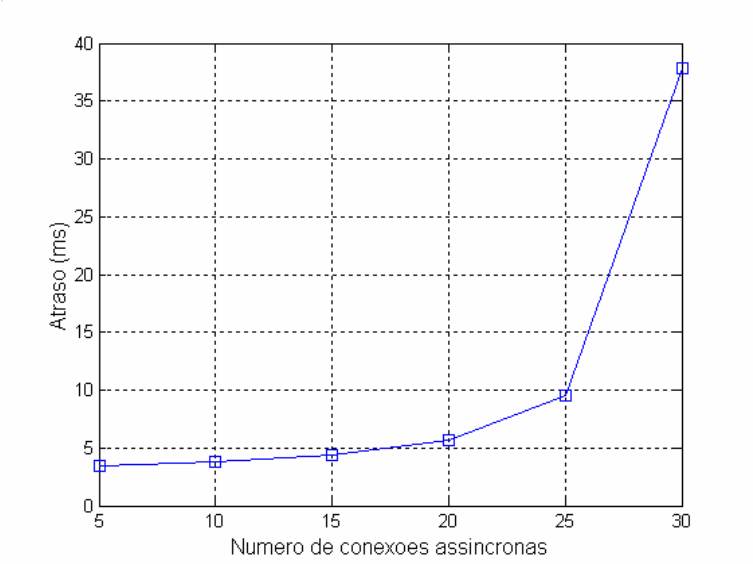


Figura 6.25: Atraso médio das conexões assíncronas em função do número de conexões assíncronas presentes no sistema.

O aumento do atraso faz com que a capacidade dos *buffers* de pacotes nos adaptadores seja atingida, provocando o descarte de pacotes. A Figura 6.26 apresenta a taxa de perda de pacotes (PLR, *Packet Loss Ratio*) das conexões assíncronas (classe II) em função do número de conexões assíncronas presentes no sistema[66]. A possibilidade de redução do débito das fontes de tráfego de débito controlável em caso de congestionamento da rede, efectuada por protocolos como o TCP, permite evitar que a perda de pacotes das conexões assíncronas atinja proporções tão elevadas como neste caso.

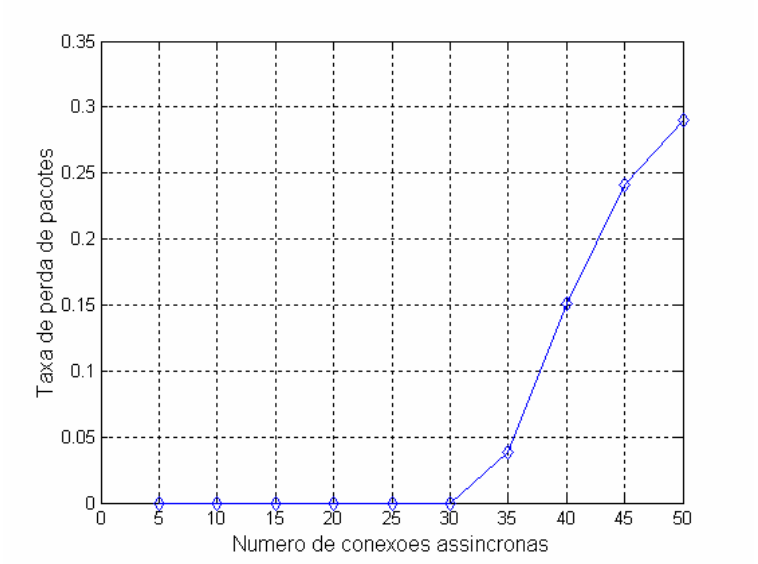


Figura 6.26: Taxa de perda de pacotes das conexões assíncronas em função do número de conexões assíncronas presentes no sistema.

A eficiência máxima registada no transporte do tráfego da classe II é de 80.1 %, sendo bastante superior à obtida com o tráfego da classe I (41.3 %), pois o maior comprimento dos trens de PDUs da fase ascendente diminui o *overhead* introduzido pelo HIPERLAN/2.

A Figura 6.27 apresenta a distribuição dos atrasos do agregado de conexões da classe II com 20 conexões assíncronas presentes no sistema. A última coluna do histograma representa a frequência dos atrasos superiores a 35 ms, pois o histograma é truncado neste valor. A carga da rede, neste caso, ainda é muito inferior do que a sua

[66] A dimensão dos buffers de pacotes nos adaptadores foi configurada em 512 kbytes.

capacidade, como mostra a Figura 6.24; mesmo assim, a variação do atraso é muito superior à obtida com as conexões de tempo real. Este facto é justificado, pois, para a transmissão do tráfego das conexões assíncronas, os terminais têm que competir pelos *slots* da fase de acesso aleatório para requisitar recursos para a transmissão dos novos pacotes ao ponto de acesso. A possibilidade de colisão das tentativas de requisição realizadas pelos diferentes terminais faz com que não haja garantias quanto ao atraso máximo que os pacotes poderão sofrer. Neste cenário, o número médio de *slots* disponíveis na fase de acesso aleatório foi 14.99, e o número médio de retransmissões na fase de acesso aleatório (variável *a*, secção 3.7.3.1.1) foi 0.53. No entanto, o valor máximo de *a* foi 8, ou seja, neste caso extremo, só à nona tentativa a requisição de recursos (RR) chegou ao ponto de acesso. O aumento do número de conexões assíncronas tende a diminuir o número de *slots* disponíveis e aumentar o valor de *a*, aumentando assim a variação do atraso.

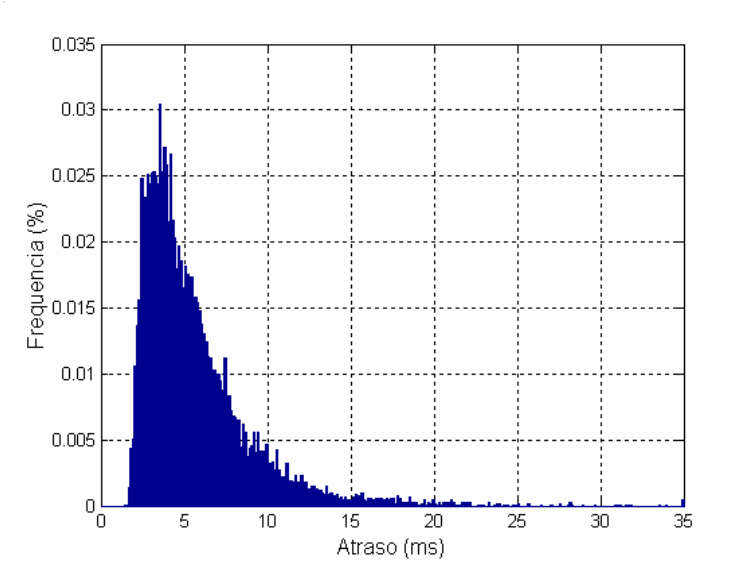


Figura 6.27: Distribuição dos atrasos do agregado de conexões assíncronas (classe II) com 20 conexões assíncronas presentes no sistema.

O impacto do tráfego assíncrono sobre o atraso sofrido pelos pacotes das conexões CBR é pequeno, como pode ser visto na Figura 6.28, onde se mostra o atraso médio e o desvio padrão do atraso da conexão CBR associada ao primeiro terminal (RT1) em função do número de conexões assíncronas presentes no sistema.

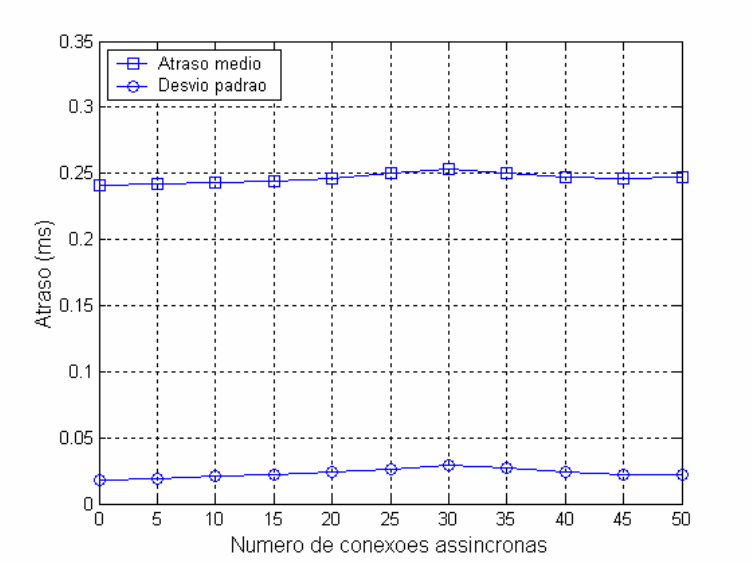


Figura 6.28: Atraso médio e desvio padrão do atraso da conexão RT1 em função do número de conexões assíncronas presentes no sistema.

Partindo de zero, à medida que se incrementa o número de conexões assíncronas no sistema, ocorre um ligeiro aumento do atraso da conexão CBR, provocado pelo aumento do número de conexões assíncronas transportadas em cada trama. O aumento do atraso não se deve ao tráfego assíncrono propriamente dito, pois este é posicionado a seguir ao tráfego CBR na trama, mas sim à expansão do canal FCH. A partir do ponto de saturação do canal, o atraso chega a diminuir ligeiramente. Isso ocorre porque o espaço disponível na trama está todo preenchido, pelo que o aumento da carga provoca o aumento do comprimento dos trens de PDUs associados às conexões assíncronas[67] e, consequentemente, a diminuição do número de conexões transportadas em cada trama, o que se reflecte na diminuição do comprimento do canal FCH.

[67] No mecanismo de escalonamento implementado, quando a requisição de recursos (RR) de uma dada conexão não é completamente atendida, a parte restante passa para o fim da fila, para evitar que uma conexão que tenha requisitado um número muito grande de canais LCH impeça a transmissão das outras conexões assíncronas ao longo de várias tramas.

6.3.5 Tráfego com periodicidade não múltipla de 2 ms

6.3.5.1 Negociação de capacidade fixa

Os parâmetros de tráfego da classe I_a, apresentados na Tabela 6.1, foram definidos tendo em consideração que os dados transmitidos pelas células ATM ocupam todo o espaço existente no seu *payload* (48 octetos). O débito dos dados encapsulados no *payload* da célula ATM foi definido como sendo 64 kbit/s, o que corresponde a um débito de 70.667 kbit/s para as células ATM (53 octetos) e a um intervalo de geração de células igual a 6 ms. Este intervalo, sendo múltiplo do período da trama do HIPERLAN/2 (2 ms), permite que o mecanismo de negociação de capacidade fixa (FCA) do HIPERLAN/2 atribua um canal LCH a cada 3 tramas para a conexão, ou seja, com uma periodicidade igual ao intervalo de geração de células, fazendo com que as células ATM sejam transmitidas após um tempo de espera aproximadamente constante no *buffer* do emissor.

Considere-se agora que a camada da adaptação do tipo 1 (AAL1) [ITU96] do ATM é utilizada. Neste caso, um dos 48 octetos do *payload* passa a ser ocupado pelo cabeçalho da camada de adaptação, pelo que sobram 47 octetos para os dados do utilizador. Se o débito dos dados mantiver-se a 64 kbit/s, o débito a nível da célula ATM aumenta para 72.17 kbit/s e uma célula ATM passa a ser preenchida a cada 5.875 ms. Como o intervalo de geração de células é menor, a atribuição de um canal LCH por cada 3 tramas deixa de ser suficiente para satisfazer o requisito da conexão, pelo que se torna necessário atribuir um canal LCH a cada duas tramas (4 ms)

A Figura 6.29 apresenta a distribuição dos atrasos para a conexão RT1 neste cenário[68]. O atraso, em vez de ser aproximadamente constante, varia entre zero e o intervalo de atribuição de canais LCH (4 ms). Quando uma célula ATM é gerada justamente no momento em que um canal LCH é disponibilizado para a sua

[68] O pico que aparece neste histograma deve-se ao sincronismo na geração de células das conexões e seu efeito sobre as mensagens de ARQ. Num teste, efectuado usando o modo sem retransmissão do HIPERLAN/2, a distribuição dos atrasos obtida foi uniforme.

transmissão, o atraso sofrido pela mesma é mínimo, sendo igual ao tempo de transmissão. Quando o canal LCH seguinte é disponibilizado, 4 ms depois, uma nova célula ainda não foi gerada, pelo que o adaptador envia uma mensagem vazia (*dummy* LCH). Quando a célula é finalmente gerada, tem que aguardar 2.125 ms pelo próximo canal LCH. A célula seguinte é transmitida com um atraso de 0.25 ms, e assim por diante. Como consequência, o atraso que os pacotes sofrem é variável, distribuindo-se entre cerca de zero até ao valor do intervalo de atribuição de canais LCH. Além disso, o envio de mensagens vazias é frequente, o que representa um desperdício de largura de banda: A utilização do canal pelas 30 conexões CBR neste caso é de 39.3 %, sendo 32.8 % superior ao que foi obtido no cenário da Figura 6.16, em que o intervalo de geração das células era múltiplo do período da trama do HIPERLAN/2, apesar de o aumento no débito das conexões ter sido de apenas 2.1 %.

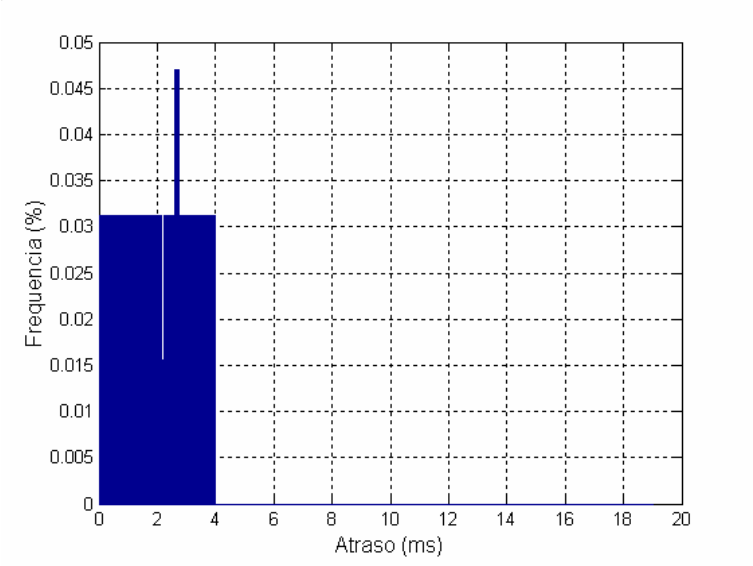


Figura 6.29: Distribuição dos atrasos para a conexão RT1 com o intervalo de geração de células igual a 5.875 ms e a utilização do mecanismo de negociação de capacidade fixa.

6.3.5.2 Alocação flexível de capacidade fixa

O mecanismo de alocação flexível de capacidade fixa, descrito na secção 4.2.2.3, foi proposto com o objectivo de superar as limitações do mecanismo de negociação de capacidade fixa do HIPERLAN/2 quando o intervalo de geração de células não é múltiplo do período da trama do HIPERLAN/2.

A Figura 6.30 apresenta a distribuição do atraso para a conexão RT1, num cenário semelhante ao anterior, porém, com a utilização do mecanismo de alocação flexível de capacidade fixa. Os resultados mostram que a variação do atraso diminui, em relação ao valor obtido com o mecanismo de negociação de capacidade fixa, para cerca de 2 ms (o que corresponde ao período da trama do HIPERLAN/2), não dependendo do intervalo de geração de células. Outro benefício do algoritmo proposto é a redução da largura de banda ocupada pelas conexões: a utilização do canal por parte das conexões CBR, neste caso, é de 26.9 %, sendo similar à obtida no cenário da secção 6.3.2, enquanto que com o mecanismo de negociação de capacidade fixa do HIPERLAN/2 foi de 39.3 %, devido à transmissão de mensagens vazias.

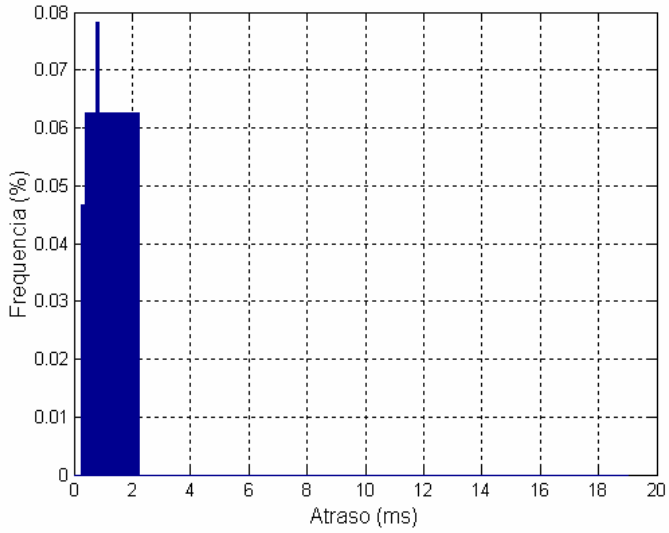


Figura 6.30: Distribuição dos atrasos para a conexão RT1 com o intervalo de geração de células igual a 5.875 ms e a utilização do mecanismo de alocação flexível de capacidade fixa.

A Figura 6.31 apresenta a distribuição dos atrasos para a conexão RT30 no mesmo cenário. O atraso maior deve-se ao facto dos canais LCH para esta conexão serem alocados a partir da segunda trama (o valor inicial da variável *FCPointer* definido para esta conexão foi 2 ms), enquanto que a primeira célula ATM, tanto desta como das demais conexões, ter sido gerada em $t = 0$. A variação do atraso é semelhante à obtida para a conexão RT1, como de resto para as demais conexões CBR, pelo que o mecanismo proposto satisfaz o critério de justiça.

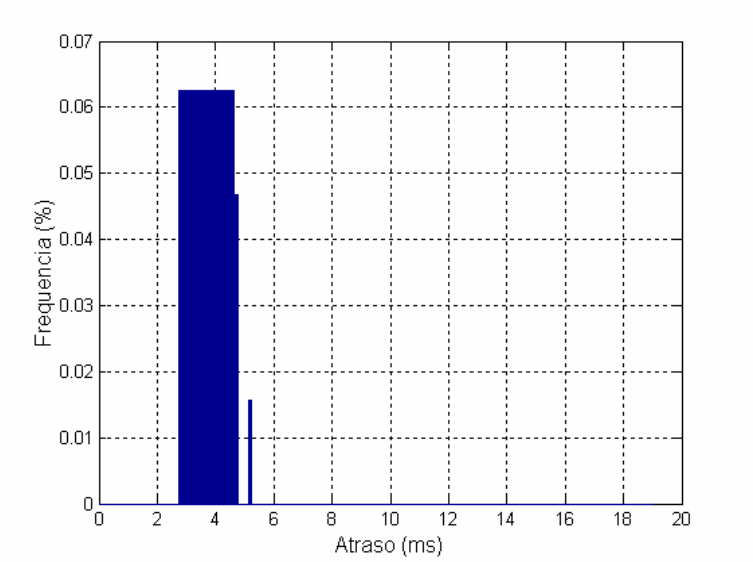


Figura 6.31: Distribuição dos atrasos para a conexão RT30 com o intervalo de geração de células igual a 5.875 ms e a utilização do mecanismo de alocação flexível de capacidade fixa.

Para testar o comportamento deste algoritmo em presença de erros no canal, simulou-se o sistema num cenário semelhante ao anterior, mas desta vez com uma taxa de erros binários (BER) igual a 10^{-3}. A Figura 6.32 apresenta a distribuição dos atrasos para a conexão RT1 com utilização do mecanismo de alocação flexível de capacidade fixa juntamente com o mecanismo de retransmissão rápida neste cenário. A variação do atraso, bem como a utilização do canal (49.4 %), são semelhantes aos obtidos no cenário da Figura 6.19, pelo que se conclui que este mecanismo funciona sem problemas quando há necessidade de retransmitir os dados corrompidos.

Quando o intervalo de geração de células é múltiplo do período do HIPERLAN/2, o mecanismo de alocação flexível de capacidade fixa produz resultados idênticos ao mecanismo de negociação de capacidade fixa do HIPERLAN/2, pelo que pode substitui-lo em qualquer dos casos.

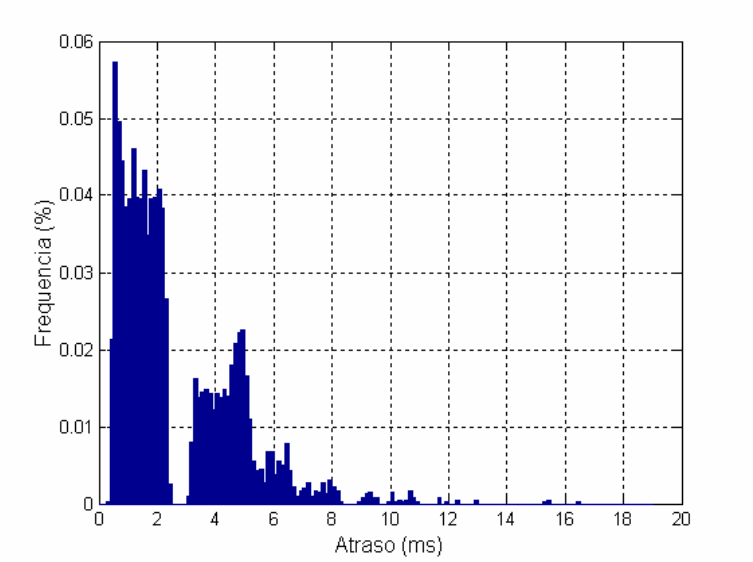


Figura 6.32: Distribuição dos atrasos para a conexão RT1 com o intervalo de geração de células igual a 5.875 ms, com a utilização do algoritmo de alocação flexível de capacidade fixa e com erros no canal (BER = 10^{-3}).

6.3.6 Tráfego RDIS

As especificações do HIPERLAN/2 definem a camada de convergência baseada em células e a camada de convergência baseada em pacotes, que podem ser utilizadas para transporte do tráfego ATM e IP, respectivamente. No entanto, estas camadas não são adequadas para o transporte de tráfego RDIS, pelo que se utiliza uma camada de convergência definida por nós, descrita a seguir, para a adaptação do tráfego RDIS à camada de ligação de dados (DLC) do HIPERLAN/2.

O PDU da camada de convergência RDIS é mapeado no *payload* da mensagem de dados do HIPERLAN/2 (canal lógico UDCH), que tem um comprimento de 49.5 octetos. Considera-se que a dimensão[69] do bloco de octetos é enviada para o controlador central (CC) durante a fase de estabelecimento da conexão DLC do HIPERLAN/2.

[69] A dimensão do bloco corresponde ao número de octetos gerados pela conexão em cada trama RDIS. Por exemplo, a dimensão é 1 no caso do transporte do tráfego de um canal B (64 kbit/s) e 6 no caso de um canal H0 (384 kbit/s).

O formato proposto para o PDU da camada de convergência RDIS, representado na Figura 6.33, é composto por um cabeçalho contendo 12 bits e por um *payload* de comprimento igual a 48 octetos. No lado do emissor, os blocos de octetos associados à conexão são recolhidos de tramas RDIS consecutivas e concatenados no *payload*. O número de blocos transportados no *payload* é indicado por um campo de 6 bits no cabeçalho. Os restantes 6 bits são reservados para sinalização.

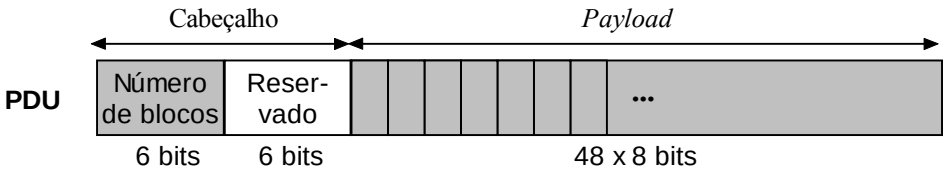


Figura 6.33: Formato do PDU da camada de convergência RDIS proposta.

Os parâmetros do tráfego RDIS utilizado nesta secção são apresentados na Tabela 6.1 (classe I_b). O mecanismo de negociação de capacidade fixa é utilizado no transporte do tráfego RDIS. Cada conexão gera um octeto a cada 125 μs, pelo que o *payload* do PDU da camada de convergência RDIS é inteiramente preenchido a cada 6 ms. Assim, um canal LCH é alocado a cada 6 ms para cada uma das 30 conexões, sendo servidas 10 conexões em cada trama do HIPERLAN/2, como foi feito com o tráfego ATM CBR.

A Figura 6.34 apresenta a distribuição dos atrasos para a conexão RT1 sem erros no canal. Neste cenário, o atraso distribui-se uniformemente entre 0 e 6 ms, aproximadamente, tanto para a conexão RT1 como para as demais conexões. A variação do atraso é maior do que para o tráfego ATM CBR devido ao agrupamento de múltiplos pacotes (blocos) numa mesma mensagem de dados. De qualquer forma, se o tráfego RDIS fosse encapsulado em células ATM, seria afectado de modo análogo, devido ao atraso de empacotamento associado ao preenchimento do *payload* da célula ATM. A utilização do canal é similar com os dois tipos de tráfego (26.9 %).

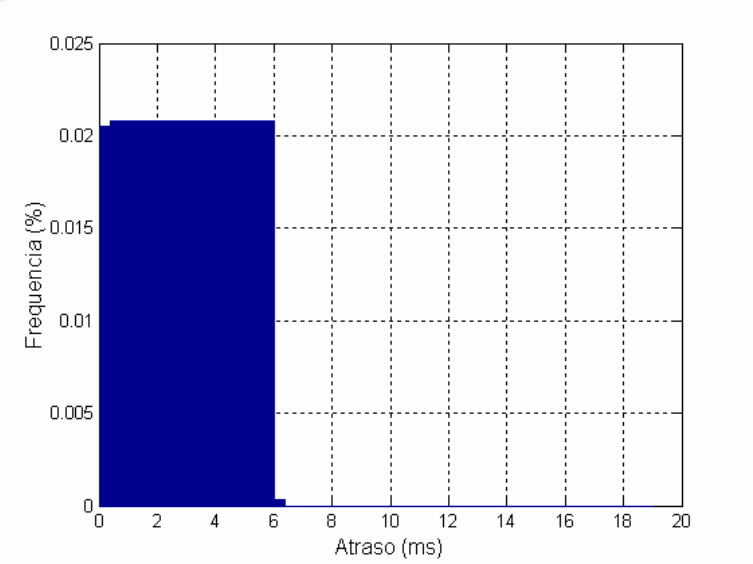


Figura 6.34: Distribuição dos atrasos para a conexão RT1 sem erros no canal, com um intervalo de atribuição de canais LCH (*FCInterval*) igual a 6 ms - Tráfego RDIS.

A Figura 6.35 apresenta o atraso dos pacotes da conexão RT1 em função do instante de geração no terminal, a partir do início da simulação, incluindo o período transitório. No momento da transmissão da primeira mensagem de dados, o *buffer* de pacotes do adaptador de terminal contém apenas dois pacotes, pelo que a mensagem transporta apenas estes pacotes. No entanto, 6 ms mais tarde, quando chega o instante de transmissão da segunda mensagem de dados, o *buffer* já contém 48 pacotes, pelo que o *payload* da mensagem é completamente preenchido, situação que se repete com as mensagens seguintes. Dos pacotes que compõem uma mensagem, os primeiros pacotes a chegar ao *buffer* são os que sofrem os maiores atrasos.

A Figura 6.36 mostra o efeito das mensagens de ARQ periódicas sobre o atraso dos pacotes da conexão RT1. Na trama em que as mensagens de ARQ são transmitidas, a mensagem de dados é transmitida um pouco mais tarde do que é comum, o que se reflecte no atraso dos pacotes que compõem o seu *payload*. A mensagem seguinte é transmitida na posição habitual dentro da trama, pelo que o padrão de atraso dos pacotes volta ao normal.

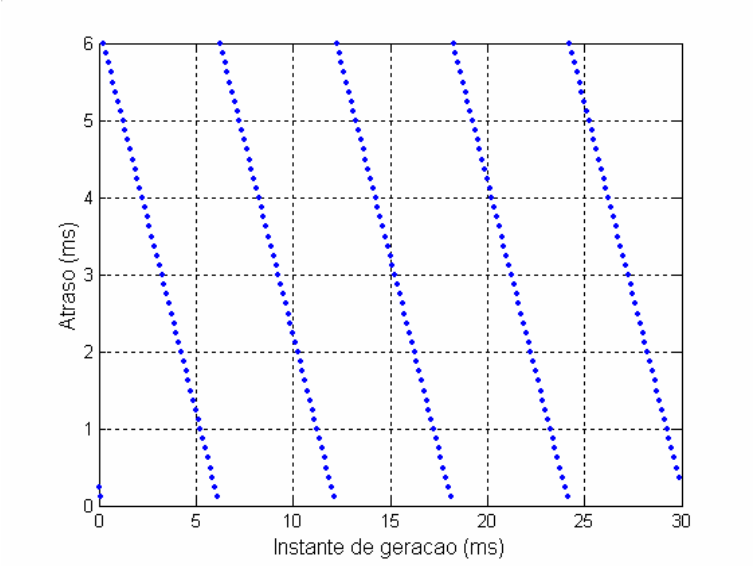


Figura 6.35: Atraso dos pacotes da conexão RT1 em função do instante de geração no terminal (a partir do início da simulação).

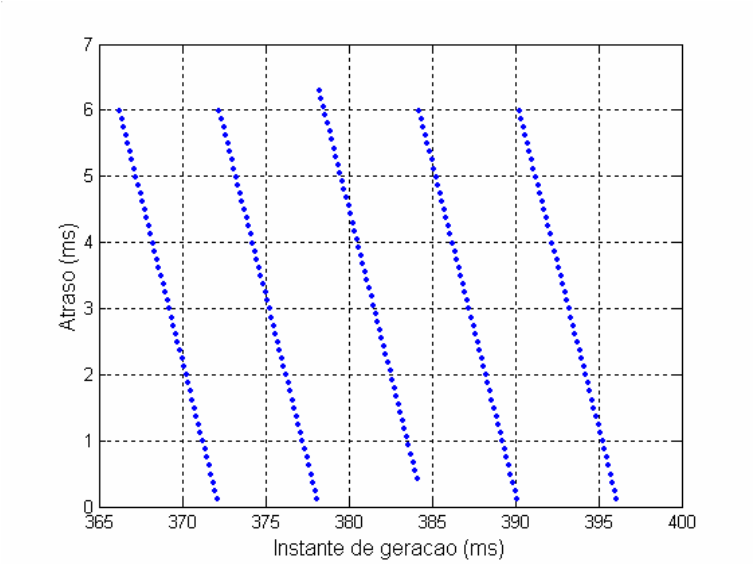


Figura 6.36: Atraso dos pacotes da conexão RT1 em função do instante de geração no terminal (efeito das mensagens de ARQ).

A Figura 6.37 apresenta a distribuição dos atrasos para a conexão RT1 no mesmo cenário, mas desta vez com um intervalo de atribuição de canais LCH (*FCInterval*) igual a 2 ms. A redução deste intervalo permite diminuir a variação do atraso para o tráfego RDIS, de 6 ms para 2 ms, porém, a utilização do canal, que era de 26.9 % com um intervalo de 6 ms, passa para 80.2 %, porque o *payload* das mensagens de dados

deixa de ser ocupado totalmente (a ocupação média do *payload*, neste caso, é de apenas 16 octetos). Sendo assim, a diminuição da variação do atraso obtida com esta abordagem obriga a uma redução drástica na eficiência.

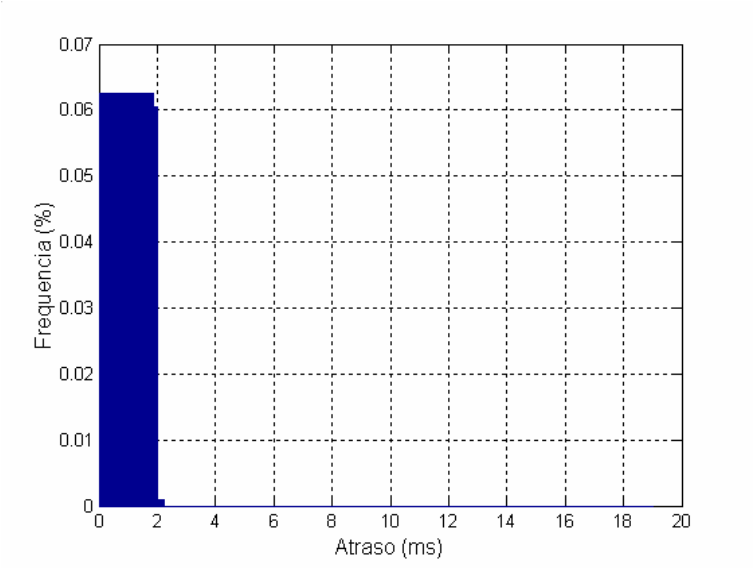


Figura 6.37: Distribuição dos atrasos para a conexão RT1 sem erros no canal, com um intervalo de atribuição de canais LCH igual a 2 ms - Tráfego RDIS.

Para analisar o desempenho do mecanismo de retransmissão rápida, proposto na secção 4.2.2.1, no transporte do tráfego RDIS, simulou-se o comportamento do sistema num cenário semelhante ao da Figura 6.34, mas com a introdução de erros no canal. A Figura 6.38 apresenta a distribuição dos atrasos obtida para a conexão RT1 com uma taxa de erros binários (BER) igual a 10^{-4}. De igual modo, a Figura 6.39 apresenta o resultado obtido com BER = 10^{-3}. A utilização do canal é de 28.3 % e 48.1 %, respectivamente. Os resultados mostram que o mecanismo de retransmissão rápida permite limitar a variação do atraso causada pelos erros no canal para o tráfego RDIS de forma análoga à conseguida para o tráfego ATM CBR.

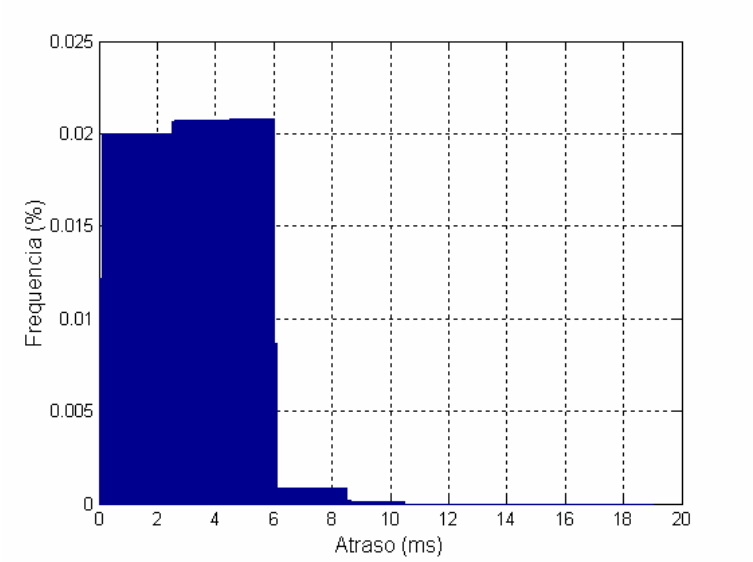


Figura 6.38: Distribuição dos atrasos para a conexão RT1 com erros no canal (BER = 10^{-4}) - Tráfego RDIS.

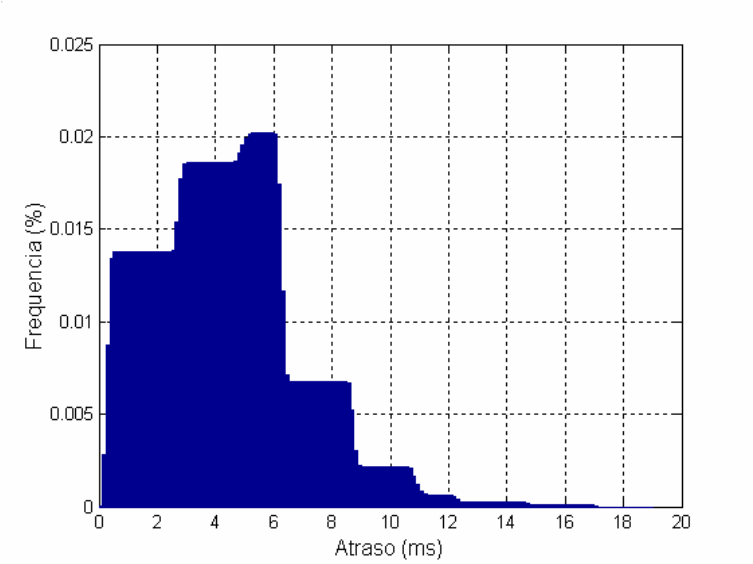


Figura 6.39: Distribuição dos atrasos para a conexão RT1 com erros no canal (BER = 10^{-3}) - Tráfego RDIS.

6.4 Sumário

Este capítulo apresentou e discutiu os resultados obtidos na simulação dos modelos e algoritmos desenvolvidos para as redes IEEE 802.11 e HIPERLAN/2 no ambiente de aplicação de um sistema de aquisição de dados e controlo.

No caso do IEEE 802.11, apresentou-se o comportamento obtido tanto com mecanismos de acesso ao meio centralizados como distribuídos. Observou-se que, quando a carga na rede é pequena, conseguem-se bons resultados no transporte de tráfego de real utilizando o mecanismo EDCA (ou a função DCF), tanto no cenário sem erros como no cenário com erros no canal e retransmissão. No entanto, este mecanismo não possibilita um controlo de admissão tão efectivo como o mecanismo HCCA (ou a função PCF, em menor grau), de forma que à medida que a carga na rede aproxima-se da saturação, o desempenho tende a deteriorar-se consideravelmente, enquanto que os fluxos admitidos através do mecanismo HCCA mantém o mesmo nível de qualidade de serviço.

Nos cenários de mistura de tráfego da classe I e da classe II, observou-se que com os mecanismos HCCA e EDCA da norma IEEE 802.11e, respectivamente, o atraso sofrido pelas conexões da classe I não é afectado pelo tráfego da classe II, pois consegue-se evitar o alargamento do período de contenção, ao contrário do que ocorre com o uso dos mecanismos originais do IEEE 802.11. Além disso, no IEEE 802.11e o controlador central pode estabelecer limites para a duração das oportunidades de transmissão. Estas características possibilitam um maior controlo sobre a qualidade de serviço oferecida às conexões.

Nos cenários sem erros no canal, a rede HIPERLAN/2 mostrou uma eficiência superior à rede IEEE 802.11a, tanto no transporte do tráfego isócrono (classe I) quanto no transporte do tráfego assíncrono (classe II). Dentro da mesma rede, o tráfego da classe II foi transportado com mais eficiência do que o tráfego da classe I, devido ao maior comprimento do *payload* da mensagem de dados (IEEE 802.11) ou do trem de PDUs (HIPERLAN/2), o que reduz o peso do *overhead* introduzido pela rede.

Na transmissão do tráfego ATM CBR (classe I_a), o número de retransmissões necessário para a recepção correcta dos dados com a rede IEEE 802.11 é mais elevado do que com a rede HIPERLAN/2, para a mesma taxa de erros binários (BER) no canal, como pode ser verificado pela comparação da Figura 6.5 com a Figura 6.18, ambas obtidas com BER = 10^{-4}. Isto ocorre porque o comprimento das mensagens de dados no IEEE 802.11 é maior, logo a taxa de erros de pacote (PER) também é maior.

A razão principal do comprimento das mensagens de dados do IEEE 802.11 ser maior, deve-se ao facto do *overhead* do cabeçalho e da cauda (34 octetos) ser bem maior que no HIPERLAN/2 (4.5 octetos). Outro factor de diferenciação é justificado pelo IEEE 802.11 proceder à transmissão integral da célula ATM (53 octetos) no *payload* da mensagem, enquanto o HIPERLAN/2 transmite uma versão comprimida da célula (49.5 octetos). Refira-se ainda que nos algoritmos A e B propostos para a rede IEEE 802.11 algumas mensagens carregam mais do que uma célula ATM, devido às retransmissões, o que também colabora no aumento do comprimento médio das mensagens.

Estes resultados indicam que a transmissão de mensagens curtas é importante para reduzir a taxa de erros de pacote e, consequentemente, o número de retransmissões quando o canal está exposto a erros. A rede HIPERLAN/2 beneficia disto sempre, pois os pacotes longos, como no caso do tráfego IP, são segmentados para transmissão. A rede IEEE 802.11 também pode fazer o mesmo com a utilização do mecanismo de fragmentação, porém, o seu *overhead* é bem maior, resultando numa eficiência menor.

O débito máximo obtido para as conexões assíncronas com a rede IEEE 802.11a foi muito inferior ao obtido com a rede HIPERLAN/2. A justificação principal deste facto deve-se à largura de banda disponível para as conexões assíncronas após o escalonamento do tráfego CBR ser bastante inferior com a rede IEEE 802.11. Além disso, a eficiência no aproveitamento da largura de banda disponível por parte desta rede também foi menor. No caso do HIPERLAN/2, quase toda a largura de banda disponível foi efectivamente usada para a transmissão de dados, pois os *slots* de contenção na fase de acesso aleatório só ocuparam o espaço deixado vazio na trama[70]. Já no IEEE 802.11, o processo de contenção ocorre durante a fase de transmissão de dados, pelo que parte da largura de banda disponível é desperdiçada em colisões entre as mensagens de dados e outra parte não é aproveitada devido ao aumento do período de *backoff* provocado pelas colisões.

[70] Esta abordagem favorece a maximização do débito agregado em detrimento do tempo de resposta na requisição de recursos, pelo que pode ser desejável aumentar o número mínimo de *slots* na fase de acesso aleatório para diminuir a probabilidade de colisão.

O algoritmo C proposto para a rede IEEE 802.11 mostrou um desempenho superior aos algoritmos A e B. Tanto este algoritmo como o algoritmo de retransmissão rápida proposto para a rede HIPERLAN/2 mostraram-se eficazes para limitar a variação no atraso do tráfego de tempo real na presença de erros, desde que a taxa de erros binários não seja excessivamente elevada.

Quando a degradação das condições no canal é muito severa, a passagem do modo de transmissão normal para um modo inferior, mais robusto, permite reduzir a variação do atraso, ao reduzir o número de retransmissões necessário para a recepção correcta das mensagens corrompidas. No entanto, o modo mais robusto é ineficiente quando as condições no canal são boas, devido à maior ocupação de banda na transmissão das mensagens, não se justificando a sua utilização permanente porque o estado *mau* geralmente é temporário. Os resultados apresentados para a rede HIPERLAN/2 mostram que o esquema de adaptação do débito de transmissão proposto, baseado na utilização do modo inferior apenas para a retransmissão das mensagens corrompidas, permite reduzir a variação do atraso em relação à utilização constante do modo normal, enquanto mantém o mesmo nível de eficiência.

Quando o intervalo de geração de células do tráfego ATM CBR não é múltiplo do período da trama do HIPERLAN/2 (2 ms), a utilização do mecanismo de alocação de capacidade fixa proposto, em substituição do mecanismo de negociação de capacidade fixa definido nas especificações do HIPERLAN/2, permite diminuir a variação do atraso e eliminar o desperdício de largura de banda associado à transmissão de mensagens vazias.

As especificações do HIPERLAN/2 não definem uma camada de convergência adequada à adaptação do tráfego RDIS ao formato da mensagem de dados utilizada pelo HIPERLAN/2, pelo que foi definida uma camada de convergência para este efeito. A variação do atraso no transporte do tráfego RDIS foi maior do que a obtida com o tráfego ATM CBR, devido ao atraso de empacotamento associado ao preenchimento do *payload* da mensagem de dados. De resto, o desempenho dos mecanismos de escalonamento propostos é semelhante nos dois casos.

Capítulo 7

Conclusões e perspectivas de desenvolvimento futuro

As redes de comunicação sem fios representam uma alternativa interessante às redes cabladas no suporte de transmissão do tráfego de sistemas de aquisição de dados e controlo, pois apresentam diversas vantagens, como o aumento da mobilidade, a redução dos custos e a rapidez de instalação e reestruturação da rede derivados da eliminação dos cabos.

O tráfego gerado pelos sistemas de aquisição de dados e controlo normalmente possui requisitos de tempo real em conjunto com uma baixa tolerância a erros, mas como as redes de comunicação sem fios existentes não estão, à partida, adaptadas ao transporte deste tipo de tráfego, devido nomeadamente à problemática dos erros no canal, muito mais grave do que nas redes cabladas, torna-se necessário optimizar o seu desempenho para que possam proporcionar um suporte adequado aos requisitos de qualidade de serviço destas aplicações.

Para servir de base ao sistema de transmissão do sistema de aquisição de dados e controlo, foram consideradas as possibilidades de utilização de duas tecnologias de redes sem fios distintas: a rede IEEE 802.11a e a rede HIPERLAN/2, ambas operando na banda de 5 GHz. Uma vez que, tanto na rede IEEE 802.11 como na rede HIPERLAN/2, os mecanismos de escalonamento de tráfego não se encontram normalizados, propôs-se alguns algoritmos de escalonamento, tanto do tráfego de dados como do tráfego de controlo, destinados a fornecer garantias de qualidade de serviço ao tráfego de tempo real, sobretudo quando sujeito a erros no canal. Neste sentido, o objectivo principal destes algoritmos é permitir a retransmissão rápida dos

dados corrompidos devido a erros no canal, para que esses dados possam chegar a tempo ao seu destino. Ao mesmo tempo, os algoritmos propostos permitem o transporte de tráfego assíncrono no sistema, aproveitando os recursos não utilizados pelo tráfego prioritário.

O escalonamento do tráfego foi analisado de forma integrada com os mecanismos de controlo de acesso ao meio e de controlo de erros. Desta forma, as características específicas destes mecanismos, para cada uma das redes sem fios utilizada, foram tidas em consideração no desenvolvimento dos algoritmos de escalonamento e na avaliação de desempenho do sistema.

Para possibilitar a avaliação da eficácia das soluções propostas, foram desenvolvidos e implementados modelos de simulação detalhados, tanto para a rede IEEE 802.11 como para a rede HIPERLAN/2, de acordo com as respectivas especificações. A integração das unidades remotas e da unidade central do sistema de aquisição de dados na rede de comunicação sem fios deu origem a um sistema baseado na rede HIPERLAN/2 e outro baseado na rede IEEE 802.11.

Foram igualmente desenvolvidos e implementados modelos de três tipos diferentes de protocolos de controlo de acesso ao meio: protocolos de acesso aleatório (função DCF e mecanismo EDCA da rede IEEE 802.11); protocolos de *polling* (função PCF e mecanismo HCCA da rede IEEE 802.11); e protocolos de reserva dinâmica explícita (HIPERLAN/2). Estes protocolos foram implementados de raiz, sem a utilização de rotinas desenvolvidas por terceiros, o que permitiu a aquisição de experiência no desenvolvimento de modelos de simulação variados. Além disso, foram definidas rotinas para geração de tráfego, modelação de erros no canal, tratamento estatístico das amostras e registo dos resultados.

Futuramente , pretende-se incorporar outras funcionalidades ao simulador, de modo a permitir a extensão das suas capacidades, por exemplo para análise do desempenho de outros sistemas de comunicação, existentes ou ainda por definir. Outra possibilidade de expansão de capacidades de simulação poderá aplicar-se à geração de tráfego de tempo real, onde actualmente, por simplicidade, apenas foram utilizados modelos de tráfego periódico com débito constante. No entanto, como as redes industriais também transportam tráfego aperiódico e tráfego de débito variável,

um trabalho futuro passa pela optimização do escalonamento de tráfego para estes casos.

Nestes casos, a atribuição de capacidade fixa para as conexões não é eficiente, pelo que é preferível realizar a atribuição de recursos de forma dinâmica. Neste contexto, no que concerne à utilização da rede IEEE 802.11, o trabalho futuro passa pela análise de algoritmos de escalonamento propostos no âmbito do TGe e de outras soluções propostas recentemente na literatura [GRI03] [LIM04] [RAJ03], tendo em vista não só a sua comparação, mas eventualmente uma complementaridade entre estas soluções e os algoritmos de escalonamento propostos nesta tese.

No que concerne à rede HIPERLAN/2, a requisição de recursos para transmissão sob pedido (apropriada para a transmissão de tráfego de débito variável) normalmente é feita durante a fase de acesso aleatório, não sendo garantido um atraso controlado, devido ao risco de colisão. Além disso, não existe um mecanismo de diferenciação de prioridades que permita atribuir ao tráfego prioritário uma maior probabilidade de sucesso no acesso ao meio, ao contrário do que acontece no IEEE 802.11e com o mecanismo EDCA. Para contornar este problema, o ponto de acesso pode atribuir canais SCH às conexões durante a fase ascendente, para a requisição de recursos livre de contenção, num esquema semelhante, de certo modo, ao *polling* efectuado na rede IEEE 802.11. Este mecanismo foi incorporado no simulador, mas ainda não foi testado com modelos de tráfego de débito variável.

Para evidenciar as qualidades das soluções propostas para resolução dos problemas criados pela concentração de fluxo, apenas foi necessário simular o tráfego transmitido no sentido ascendente, pelo que as rotinas necessárias à comunicação no sentido descendente não foram completamente implementadas no simulador. No futuro pretende-se colmatar esta lacuna para permitir a determinação de eventuais adaptações nos algoritmos de escalonamento propostos para optimizar o transporte do tráfego descendente e a análise de desempenho do sistema com tráfego bidireccional.

O sistema que foi implementado no simulador opera utilizando comunicação centralizada, ou seja, as unidades remotas comunicam directamente apenas com a unidade central. No entanto, em muitas redes industriais as estações terminais podem comunicar directamente entre si, pelo que seria interessante avaliar o desempenho das

soluções propostas neste cenário de aplicação. Como tanto a rede IEEE 802.11 como a rede HIPERLAN/2 suportam a comunicação directa, a concretização desta ideia poderá passar basicamente pela adaptação dos modelos de simulação desenvolvidos de modo a incorporarem as funcionalidades necessárias.

Para diminuir o *overhead* associado às interrogações na rede IEEE 802.11 poder-se-ia substituir diversas tramas CF-Poll por uma única trama (*superpoll*) contendo os endereços das estações a serem servidas em sequência. No entanto, uma implementação eficiente deste mecanismo precisaria de lidar com outros problemas, como o efeito da estação oculta e o reconhecimento das mensagens transmitidas. Uma proposta neste sentido, com a avaliação do seu desempenho por simulação, constitui outra hipótese de desenvolvimento futuro.

Por fim, estes desenvolvimentos só ficarão concluídos com a implementação e teste das soluções propostas nos sistemas reais, que apenas foram alvo de simulação no trabalho apresentado nesta tese.

Bibliografia

[3GP01] 3GPP Doc. 3G TS 23.107 v3.1.0, "QoS Concept and Architecture", September 2001.

[AFO03] J. Afonso and J. E. Neves, "Performance Simulation of HIPERLAN/2 with Low Debit Traffic for Real Time Data Acquisition and Control Applications", *4th Conference on Telecommunications*, Aveiro, Portugal, pp. 139-142, June 2003.

[AKY99] I. F. Akyildiz, J. McNair, L. C. Martorell, R. Puigjaner and Y. Yesha, "Medium Access Control Protocols for Multimedia Traffic in Wireless Networks", *IEEE Network*, pp. 39-47, July/August 1999.

[ALE00] C. Alexopoulos and A. F. Seila, "Output Analysis for Simulations", *Winter Simulation Conference WSC 2000*, Orlando, FL, USA, pp. 101-108, December 2000.

[ALM02] J. N. T. Almeida, "High-Speed Wireless Mobile LAN Communications: Improved Error Control Mechanisms for Real-Time Services", PhD thesis, Universidade do Porto, June 2002.

[ANA98] G. Anastasi, L. Lenzini, E. Mingozzi, A. Hettich and A. Krämling, "MAC Protocols for Wideband Wireless Local Access: Evolution Towards Wireless ATM", *IEEE Personal Communications*, pp. 53-64, October 1998.

[BER97] M. O. Berger, O. Kubitz and T. Rochner, "Asymmetric Data Communication using DECT in an Industrial Environment", *IEEE Vehicular Technology Conference VTC'97*, pp. 2099-2102, May 1997.

[BIN00] B. Bing, "High-Speed Wireless ATM and LANs", Artech House, 2000.

[BLU01] Bluetooth Special Interest Group (SIG), "Specification of the Bluetooth System - Core", February 2001.

[CAI97] J. Cai and D. J. Goodman, "General Packet Radio Service in GSM", *IEEE Communications Magazine*, pp. 122-131, October 1997.

[CHA99a] P. Chaudhury, W. Mohr and S. Onoe, "The 3GPP Proposal for IMT-2000", *IEEE Communications Magazine*, pp. 72-81, December 1999.

[CHA99b] D. Chalmers and M. Sloman, "A Survey of Quality of Service in Mobile Computing Environments", *IEEE Communications Surveys*, Second Quarter 1999.

[CHA00] A. Chandra, V. Gummalla and J. O. Limb, "Wireless Medium Access Control Protocols", *IEEE Communications Surveys & Tutorials*, Second Quarter, 2000.

[COM90] D. E. Comer, "Internetworking with TCP/IP - Vol I: Principles, Protocols and Architecture", Prentice-Hall International, 1990.

[COU00] C. Coutras, S. Gupta and N. B. Shroff, "Scheduling of real-time traffic in IEEE 802.11 wireless LANs", *Wireless Networks*, Vol. 6, pp. 457-466, December 2000.

[CRO97] B. Crow, I. Widjaja, J. G. Kim and P. T. Sakai, "IEEE 802.11 Wireless Local Area Networks", *IEEE Communications Magazine*, pp. 116-126, September 1997.

[DEC97a] J.-D. Decotignie, “Industrial Electronics Handbook”, Section 18.4 - Fieldbus, CRC Press, 1997, pp. 403-408.

[DEC97b] DECT Forum, “DECT: The standard explained”, 1997.

[DOU02] A. Doufexi et al., “A Comparison of the HIPERLAN/2 and IEEE 802.11a Wireless LAN Standards”, *IEEE Communications Magazine*, pp. 172-180, May 2002.

[EBE94] J.-P. Ebert, B. Rathke and A. Wolisz, “A Conceptual Framework to integrate Mobility into ISDN”, *International Conference Wireless Computer Networks WCN’94*, Le Hague, Netherlands, pp. 986-993, September 1994.

[ECK00] D. A. Eckhardt and P. Steenkiste, “Effort-limited Fair (ELF) Scheduling for Wireless Networks”, *IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000.

[ETS00] ETSI TR 101 683 V1.1.1, “Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; System Overview”, February 2000.

[ETS00a] ETSI TS 101 763-1, “Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Cell based Convergence Layer; Part 1: Common Part”, April 2000.

[ETS00b] ETSI TS 101 761-1, “Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Data Link Control (DLC) Layer; Part 1: Basic Data Transport Functions”, November 2000.

[ETS00c] ETSI TS 101 493-1 V1.1.1, “Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Packet based Convergence Layer; Part 1: Common Part”, April 2000.

[ETS01a] ETSI TS 101 761-1, “Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) Layer”, February 2001.

[ETS01b] ETSI TS 101 761-1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Data Link Control (DLC) Layer; Part 2: Radio Link Control (RLC) sublayer", April 2001.

[ETS01c] ETSI EN 300 175-1, "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 1: Overview", February 2001.

[EUG98] T.S. Eugene Ng, I. Stoica and H. Zhang, "Packet Fair Queueing Algorithms for Wireless Networks with Location-Dependent Errors", *IEEE INFOCOM '98*, San Francisco, USA, March/April 1998.

[FAR99] M. Farsi, K. Ratcliff and M. Barbosa, "An Overview of Controller Area Network", *Computing & Control Engineering Journal*, pp. 113-120, June 1999.

[FU96] K. Fu, Y. J. Guo and S. K. Barton, "Performance of the EY-NPMA Protocol", *Wireless Personal Communications*, Vol. 4, No. 1, pp. 41-50, 1997.

[GLI97] S. Glisic, "Spread Spectrum CDMA Systems for Wireless Communications", Artech House, 1997.

[GOO89] D. J. Goodman, R. A. Valenzuela, K. T. Gayliard and B. Ramamurthi, "Packet Reservation Multiple Access for Local Wireless Communications", *IEEE Transactions on Communications*, Vol. 37, No. 8, pp. 885-890, August 1989.

[GRI02] A. Grillo and M. Nunes, "Performance evaluation of IEEE802.11e,", *IEEE PIMRC'02*, Lisboa, Portugal, September 2002.

[GRI03] A. Grilo, M. Macedo e M. Nunes, "A Scheduling Algorithm for QoS Support in IEEE802.11E Networks", *IEEE Wireless Communications*, pp. 36-43, June 2003.

[GU03] D. Gu and J. Zhang, “QoS Enhancement in IEEE802.11 Wireless Local Area Networks”, *IEEE Communications Magazine*, pp. 120-124, June 2003.

[HAA00] J. C. Haartsen, “The Bluetooth Radio System”, *IEEE Personal Communications*, pp. 28-36, February 2000.

[HET99] A. Hettich and M. Schröther, “IEEE 802.11 or ETSI BRAN HIPERLAN/2: Who will win the race for a high speed wireless LAN standard?”, *European Wireless Conference EW’99*, Munich, Germany, October 1999.

[HIG98] G. N. Higginbottom, “Performance Evaluation of Communication Networks”, Artech House, 1998.

[IEE90] IEEE 802.4 (ISO/IEC 8802-4-1990), “Token-Passing Bus Access Method and Physical Layer Specifications”, 1990.

[IEE98a] IEEE 802.5 (ISO/IEC 8802-5:1998), “Token Ring Access Method and Physical Layer Specification, 1998.

[IEE98b] IEEE 802.1D, “IEEE standard for local and metropolitan area networks - Common specifications - Media access control (MAC) Bridges”, 1998.

[IEE99] IEEE 802.11 (ISO/IEC 8802-11:1999), “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, 1999.

[IEE99a] IEEE 802.11 (8802-11:1999/Amd 1:2000(E)), “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - High-speed Physical Layer in the 5 GHz Band”, 1999.

[IEE99b] IEEE 802.11, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Higher-Speed Physical Layer Extension in the 2.4 GHz Band”, 1999.

[IEE02a] IEEE 802.3, “Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications”, 2002.

[IEE02b] IEEE 802.15.1 “Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs), 2002.

[IEE02c] IEEE 802.11e draft/D4.0, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS)”, November 2002.

[IET97] J. Wroclawski, “The Use of RSVP with IETF Integrated Services”, IETF RFC2210, September 1997.

[IRDA] Infrared Data Association, http://www.irda.org.

[ITU88] ITU-T Rec. I.412, “ISDN user-network interfaces - Interface structures and access capabilities”, 1988.

[ITU90] ITU-T Rec. G.726, “40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM)”, 1990.

[ITU96] ITU-T Rec. I.363.1, “B-ISDN ATM Adaptation Layer specification: Type 1 AAL”, August 1996.

[ITU98a] ITU-T Rec. G.804, “ATM cell mapping into Plesiochronous Digital Hierarchy (PDH)”, 1998.

[ITU98b] ITU-T Rec. G.704, “Synchronous frame structures used at 1544, 6312, 2048, 8448 and 44736 kbit/s hierarchical levels”, 1998.

[ITU99] ITU-T Rec. I.361, “B-ISDN ATM layer specification”, February 1999.

[ITU01] ITU-T Rec. G.703, "Physical/electrical characteristics of hierarchical digital interfaces", 2001.

[JAI00] R. Jain, "The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling", Wiley, 2000.

[JEO99] M. R. Jeong, H. Morikawa and T.Aoyama, "Fair Scheduling Algorithm for Wireless Packet Network", *ICPP International Workshop on Internet IWI'99*, pp. 280-285, 1999.

[JIA98] S. Jiang, "Wireless Communications and a Priority Access Protocol for Multiple Mobile Terminals in Factory Automation", *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 1, pp. 137-143, February 1998.

[JOH99] P. Johansson, N. Johansson, U. Körner, J. Elg, and G. Svennarp, "Short range radio based ad-hoc networking: performance and properties", *IEEE International Conference on Communications ICC'99*, Vancouver, Canada, pp. 1414-1420, 1999.

[KAD99] A. Kadelka, A. Hettich and S. Dick, "Performance Evaluation of the MAC protocol of the ETSI BRAN HIPERLAN/2 standard", *European Wireless Conference EW'99*, Munich, Germany, pp. 157-162, October 1999.

[KES97] S. Keshav, "An Engineering Approach to Computer Networking, Chapter 9: Scheduling", Addison-Wesley, 1999.

[KHU99] J. Khun-Jush, P. Schramm, U. Wachsmann and F. Wenger, "Structure and Performance of the HIPERLAN/2 Physical Layer", *IEEE Vehicular Technology Conference VTC'99*, Amsterdam, Netherlands, pp. 2667-2671, September 1999.

[KIM01] H. Kim, S. K. Biswas, P. Narasimhan, R. Siracusa and C. Johnston, "Design and Implementation of a QoS Oriented Data-Link Control

Protocol for CBR Traffic in Wireless ATM Networks", *Wireless Networks*, Vol. 7(5), Kluwer Academic Publishers, pp. 531-540, September 2001.

[KUB97] O. Kubbar and H. T. Mouftah, "Multiple Access Control Protocols for Wireless ATM: Problem Definition and Design Objectives", *IEEE Communications Magazine*, November 1997.

[LI00a] H. Li et al, "Automatic Repeat Request (ARQ) Mechanism in HIPERLAN/2", *IEEE Vehicular Technology Conference VTC'2000*, pp. 2093-2097, September 2000.

[LI00b] H. Li, G. Malmgren and M. Pauli, "Performance Comparison of the Radio Link Protocols of IEEE 802.11a and HIPERLAN/2", *IEEE Vehicular Technology Conference VTC'2000*, September 2000.

[LIM04] L. W. Lim et al., "A QoS Scheduler for IEEE 802.11e WLANs", *IEEE Consumer Communications and Networking Conference CCNC'2004*, Las Vegas, Nevada, USA, pp. 199-204, January 2004.

[LIN00] P. Lin, B. Bensaou, Q. L. Ding and K. C. Chua "A Wireless Fair Scheduling Algorithm for Error-Prone Wireless Networks", *ACM International Workshop on Wireless Mobile Multimedia WoWMoM'2000*, Boston, USA, August 2000.

[LIU97] H. Liu, H. Ma, M.E. Zarki and S. Gupta, "Error control schemes for networks: An Overview", *Mobile Networks and Applications*, Vol. 2, No. 2, pp. 167-182, 1997.

[LU99] S. Lu, V. Bharghavan and R. Srikant, "Fair Scheduling in Wireless Packet Networks", *IEEE/ACM Transactions on Networking*, Vol. 7, No. 4, pp. 473-489, August 1999.

[MAN02] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz and L. Stibor, "IEEE 802.11e Wireless LAN for Quality of Service", *European Wireless Conference EW'02*, Florence, Italy, February 2002.

[MAN03] S. Mangold et al., "Analysis of IEEE 802.11e for QoS Support in Wireless LANs", *IEEE Wireless Communications*, pp. 40-50, December 2003.

[MAR99] R. B. Marks, "The IEEE 802.16 Working Group on Broadband Wireless", *IEEE Network*, pp. 4-5, March/April 1999.

[MCD98] D. E. McDysan and D. L. Spohn, "ATM Theory and Application", McGraw-Hill, 1998.

[MIC97] H. Michiel and K. Laevens, "Teletraffic Engineering in a Broad-Band Era", *Proceedings of the IEEE*, Vol. 85, No. 12, pp. 2007-2033, December 1997.

[NEG00] K. J. Negus, A. P. Stephens and J. Lansford, "HomeRF: Wireless Networking for the Connected Home", *IEEE Personal Communications*, pp. 20-27, February 2000.

[NEV99] J. E. Neves, "Modular Architecture for High Flexibility ATM Based Control System", *Asia-Pacific Conference on Communications APCC'99*, Beijing, China, October 1999.

[NIC03] P. Nicopolitidis, M. S. Obaidat, G. I. Papadimitriou and A. S. Pomportsis, "Wireless Networks", John Wiley & Sons, 2003.

[NUN92] M. S. Nunes and A. J. Casaca, "Redes Digitais com Integração de Serviços", Editorial Presença, 1992.

[PAS98] N. Passas et al., "MAC Protocol and Traffic Scheduling for Wireless ATM Networks", *ACM Mobile Networks and Applications*, No. 3, pp. 275-292, 1998.

[PAH97] K. Pahlavan, A. Zahedi and P. Krishnamurthy, "Wideband Local Access: Wireless LAN and Wireless ATM", *IEEE Communications Magazine*, pp. 34-40, November 1997.

[PAS97] N. Passas et al., "Quality-of-Service Oriented Medium Access Control for Wireless ATM Networks", *IEEE Communications Magazine*, pp. 42-50, November 1997.

[POW02] K. Pawlikowski, H.-D. J. Jeong and J.-S. R. Lee, "On Credibility of Simulation Studies of Telecommunication Networks", *IEEE Communications Magazine*, pp. 132-139, January 2002.

[PRY95] M. de Pricker, "Asynchronous Transfer Mode - Solution for Broadband ISDN", 3rd edition, Prentice Hall, 1995.

[RAJ03] S. Rajesh et al., "QoS Algorithms for IEEE 802.11e Implementation", *Asia Pacific Conference on Communication*, Malaysia, pp. 213-217, September 2003

[RAM98] P. Ramanathan and P. Agrawal, "Adapting Packet Fair Queueing Algorithms to Wireless Networks", *ACM/IEEE International Conference on Mobile Computing and Networking MOBICOM'98*, Dallas, Texas, USA, October 1998.

[RAP02] T. S. Rappaport, "Wireless Communications - Principles and Practice", 2nd Edition, Prentice Hall, 2002.

[RAY97] D. Raychaudhuri et al., "WATMnet: A prototype wireless ATM system for multimedia personal communication", *IEEE Journal on Selected Areas in Communications*, 15(1), pp. 83–95, January 1997.

[ROH99] H. Rohling, T. May, K. Brüninghaus and R. Grünheid, "Broad-Band OFDM Radio Transmission for Multimedia Applications", *Proceedings of the IEEE*, Vol. 80, No. 10, October 1999.

[RUB97] I. Rubin, "Chapter 46: Multiple Access Methods for Communications Networks". In J. D. Gibson (Ed.), "The Communications Handbook", CRC Press, 1997.

[SAN98] R. Santos, "Estação Multisensorial para Estufas Agrícolas", MSc thesis, Universidade do Minho, 1998.

[SAR00] H. Sari, F. Vanhaverbeke and M. Moeneclaey, "Extending the Capacity of Multiple Access Channels", *IEEE Communications Magazine*, pp. 74-82, January 2000.

[SCO97] J. Scourias, "Overview of the Global System for Mobile Communications", 1997. http://ccnga.uwaterloo.ca/~jscouria/GSM/gsmreport.html.

[SEO00] Y. Yi, Y. Seok and J. Park, "$W^2F^2Q^2$: Packet Fair Queueing in Wireless Packet Networks", *ACM International Workshop on Wireless Mobile Multimedia WoWMoM'2000*, Boston, USA, August 2000.

[SER97] C. Serôdio et al., "MNet-DACS: Multi-level Network Data Acquisition and Control System", *IEEE International Symposium on Industrial Electronics ISIE'97*, Guimarães, Portugal, pp. 39-43, July 1997.

[SIL98] L. Silva, A. Muchaxo and H. Sarmento, "Interworking DECT with Ethernet for Low Cost Wireless LAN", *IEEE Conference on Universal Personal Communications*, Florence, Italy, pp. 727-731, October 1998.

[STA94] W. Stallings, "Data and Computer Communications", Maxwell MacMillan International Editions, 1994.

[TAH89] H. A. Taha, "Operations Research: An Introduction", Maxwell MacMillan International Editions, 1989.

[TRU03] H. L. Truong and G. Vannuccini, “Performance Evaluation of the QoS Enhanced IEEE 802.11e MAC Layer”, *IEEE Vehicular Technology Conference VTC‘2003*, Orlando, Florida, USA, pp. 940-944, April 2003.

[VAR99] U. Varshney, “Error Control Techniques for Wireless ATM Networks”, *IEEE International Performance Computing and Communications Conference IPCCC’99*, Phoenix, Arizona, USA, pp. 104-110, February 1999.

[VAR00] A. Varga, “OMNET++ Discrete Event Simulation System”, User Manual, Department of Telecommunications, Technical University of Budapest, 2000.

[VEE01] M. Veeraraghavan, N. Cocker and T. Moors, “Support of voice services in IEEE 802.11 wireless LANs”, *IEEE INFOCOM’2001*, Anchorage, Alaska, USA, pp. 488-497, April 2001.

[VIS95] M. A. Visser and M. El Zarki, “Voice and Data Transmission over an 802.11 Wireless Network”, *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC’95*, Toronto, Canada, pp. 648-652, September 1995.

[WEI97] J. Weinmiller, M. Schlager, A. Festag and A. Wolisz, “Performance Study of Access Control in Wireless LANs - IEEE 802.11 DFWMAC and ETSI RES 10 HIPERLAN”, *Mobile Networks and Applications*, Vol. 2, No. 1, pp. 55-67, 1997.

[WES98] E. K. Wesel, “Wireless Multimedia Communications: Networking Video, Voice and Data”, Addison-Wesley, 1998.

[WIL97] A. Willig, “A MAC Protocol and a Scheduling Approach as Elements of a Lower Layers Architecture in Wireless Industrial LANs”, *IEEE International Workshop on Factory Communication Systems WFCS'97*, Barcelona, Spain, pp. 139-147, October 1997

[WIL02] A. Willig, “Analysis of the PROFIBUS Token Passing Protocol over Wireless Links”, *IEEE International Symposium on Industrial Electronics ISIE’02*, L'Aquila, Italy, July 2002.

[XU03] S. Xu, “Advances in WLAN QoS for 802.11: an Overview”, *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC’03*, Beijing, China, pp 2297-2301, September 2003.

Apêndice A: Tabelas estatísticas

Tabela A1: Quantis da distribuição *t* de Student.

	S									
N	**0.75**	**0.8**	**0.85**	**0.9**	**0.95**	**0.975**	**0.99**	**0.995**	**0.997**	**0.998**
1	1	1.376	1.963	3.078	6.31	12.71	31.82	63.66	106.1	159.2
2	0.816	1.061	1.386	1.886	2.92	4.303	6.965	9.925	12.85	15.76
3	0.765	0.978	1.25	1.638	2.353	3.182	4.541	5.841	6.994	8.053
4	0.741	0.941	1.19	1.533	2.132	2.776	3.747	4.604	5.321	5.951
5	0.727	0.92	1.156	1.476	2.015	2.571	3.365	4.032	4.57	5.03
6	0.718	0.906	1.134	1.44	1.943	2.447	3.143	3.707	4.152	4.524
7	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	3.887	4.207
8	0.706	0.889	1.108	1.397	1.86	2.306	2.896	3.355	3.705	3.991
9	0.703	0.883	1.1	1.383	1.833	2.262	2.821	3.25	3.573	3.835
10	0.7	0.879	1.093	1.372	1.812	2.228	2.764	3.169	3.472	3.716
11	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	3.393	3.624
12	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.33	3.55
13	0.694	0.87	1.079	1.35	1.771	2.16	2.65	3.012	3.278	3.489
14	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.234	3.438
15	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.197	3.395
16	0.69	0.865	1.071	1.337	1.746	2.12	2.583	2.921	3.165	3.358
17	0.689	0.863	1.069	1.333	1.74	2.11	2.567	2.898	3.138	3.326
18	0.688	0.862	1.067	1.33	1.734	2.101	2.552	2.878	3.113	3.298
19	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861	3.092	3.273
20	0.687	0.86	1.064	1.325	1.725	2.086	2.528	2.845	3.073	3.251
21	0.686	0.859	1.063	1.323	1.721	2.08	2.518	2.831	3.056	3.231
22	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819	3.041	3.214
23	0.685	0.858	1.06	1.319	1.714	2.069	2.5	2.807	3.027	3.198
24	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.014	3.183
25	0.684	0.856	1.058	1.316	1.708	2.06	2.485	2.787	3.003	3.17
26	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	2.992	3.158
27	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	2.982	3.147
28	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	2.973	3.136
29	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	2.965	3.127
30	0.683	0.854	1.055	1.31	1.697	2.042	2.457	2.75	2.957	3.118
31	0.682	0.853	1.054	1.309	1.696	2.04	2.453	2.744	2.95	3.109
32	0.682	0.853	1.054	1.309	1.694	2.037	2.449	2.738	2.943	3.102
33	0.682	0.853	1.053	1.308	1.692	2.035	2.445	2.733	2.937	3.094
34	0.682	0.852	1.052	1.307	1.691	2.032	2.441	2.728	2.931	3.088
35	0.682	0.852	1.052	1.306	1.69	2.03	2.438	2.724	2.926	3.081
40	0.681	0.851	1.05	1.303	1.684	2.021	2.423	2.704	2.902	3.055
50	0.679	0.849	1.047	1.299	1.676	2.009	2.403	2.678	2.87	3.018
60	0.679	0.848	1.045	1.296	1.671	2	2.39	2.66	2.849	2.994
120	0.677	0.845	1.041	1.289	1.658	1.98	2.358	2.617	2.798	2.935
Inf	0.674	0.842	1.036	1.282	1.645	1.96	2.326	2.576	2.748	2.878

Apêndice B: Listagem do programa de simulação desenvolvido

```
//*************************************************************************
// Descrição da topologia do sistema simulado.
//*************************************************************************

// CE1
//
// Canal E1 entre terminais e adaptadores.
//
channel CE1
    // datarate 1920000
endchannel

// cTerm
//
// Terminal, gera os dados
//
simple cTerm
    gates:
        in: in;
        out: out;
endsimple

// cAdapt
//
// Adaptador de terminal. Recebe os pacotes do terminal
// e comunica com a estação base atraves do meio.
//
simple cAdapt
    gates:
        in: from_term;
        out: to_term;
        in: from_meio;
        out: to_meio;
endsimple

// cMeio
//
// Meio sem fios. Interface entre a estação base e os adaptadores.
// Modela o meio de difusão, o tempo de transmissão e os erros.
//
simple cMeio
    gates:
        in: from_unit[];
        out: to_unit[];
endsimple

// cEB
//
// Estação base. Controla o acesso ao meio e comunica com os
// adaptadores de terminal e o MUX atraves do meio sem fios.
// Gera as estatisticas pelo lado do MUX.
//
simple cEB
    gates:
        in: from_meio;
        out: to_meio;
        in: from_MUX;
        out: to_MUX;
endsimple

// cMUX
```

```
//
// Destroi os pacotes recebidos.
//
simple cMUX
    gates:
      in: in;
      out: out;
endsimple

// cNet
//
// terminais + adaptadores + meio + estacao base + MUX
//
module cNet
    parameters:
        trafego1 : string,
        trafego2 : string,
        data_rate1, data_rate2,
        data_length1, data_length2,
        adapt_type : string,
        meio_type : string,
        EB_type : string,
        num_term1, num_term2 : numeric const,
        transient_criteria_time : bool,
        wind_ini,
        stop_criteria_time : bool,
        wind_collect,
        BER_good, BER_bad, t_good, t_bad,
        display_update_time : bool,
        display_update, frag_threshold,
        retransmite : bool;
    submodules:
        // cMUX tem que ser o primeiro, pois seu initialize() contém código que deve
        // ser executado anteriormente ao dos outros submódulos neste projecto.
        MUX: cMUX;
        term: cTerm[num_term1 + num_term2];
        adapt: adapt_type[num_term1 + num_term2] like cAdapt;
        meio: meio_type like cMeio;
            gatesizes:
                from_unit[num_term1 + num_term2 + 1],
                    to_unit[num_term1 + num_term2 + 1];
        EB: EB_type like cEB;
    connections:
        for i=0..num_term1 + num_term2 - 1 do
            term[i].out --> CE1 --> adapt[i].from_term;
            term[i].in <-- CE1 <-- adapt[i].to_term;
            adapt[i].to_meio --> meio.from_unit[i];
            adapt[i].from_meio <-- meio.to_unit[i];
        endfor;
        EB.to_meio --> meio.from_unit[num_term1 + num_term2];
        EB.from_meio <-- meio.to_unit[num_term1 + num_term2];
        EB.to_MUX --> MUX.in;
        EB.from_MUX <-- MUX.out;
endmodule

// Net
//
// Uma instância do módulo cNet
//
network Net: cNet
endnetwork



;**************************************************************************
; Parâmetros de configuração da simulação (parte comum).
;**************************************************************************

include ../seedsruns.ini

[Parameters]
Net.transient_criteria_time = false
Net.wind_ini = 200  ;1000
Net.stop_criteria_time = false
```

```
Net.wind_collect = 2000 ;20000
Net.display_update_time = true
Net.display_update = 1
Net.BER_good = 1E-4 ;0
Net.BER_bad = 1E-4  ;0
Net.t_good = 0.100
Net.t_bad = 0.033
Net.frag_threshold = 15000  ;-1
Net.retransmite = true

Net.trafego1 = "CBR_SINC"   ;ATM
Net.data_rate1 = 70.6666666666667E+3
Net.data_length1 = 424

;Net.trafego1 = "CBR_SINC" ;RDIS
;Net.data_rate1 = 64E+3
;Net.data_length1 = 8

Net.trafego2 = "Poisson"
Net.data_rate2 = 300E+3
Net.data_length2 = 12000

Net.num_term1 = 30

[Cmdenv]
display-update = 1000
module-messages = no
verbose-simulation = no

[Tkenv]
default-run = 14
use-mainwindow = yes
print-banners = yes
animation-speed = 1.0

[General]
network = Net

[Run 1]
Net.num_term2 = 0

[Run 2]
Net.num_term2 = 5

[Run 3]
Net.num_term2 = 10

[Run 4]
Net.num_term2 = 15

[Run 5]
Net.num_term2 = 20

[Run 6]
Net.num_term2 = 25

[Run 7]
Net.num_term2 = 30

[Run 8]
Net.num_term2 = 35

[Run 9]
Net.num_term2 = 40

[Run 10]
Net.num_term2 = 45

[Run 11]
Net.num_term2 = 50

[DisplayStrings]
*.term[*] = "p=60,60,col,60;b=30,30,rect;o=#008000"
*.adapt[*] = "p=120,60,col,60;b=30,30,oval"
```

```
*.meio = "p=240,120;i=cloud_b"
*.EB = "p=360,120;b=50,50,oval"
*.MUX = "p=430,120;i=mux"



;**************************************************************************
; Valores iniciais usados nos geradores de números aleatórios.
;**************************************************************************

[General]
gen0-seed = 1
gen1-seed = 1768507984
gen2-seed = 33648008
gen3-seed = 1082809519
gen4-seed = 703931312
gen5-seed = 1856610745
gen6-seed = 784675296
gen7-seed = 426676692
gen8-seed = 1100642647
gen9-seed = 1359921031

[Cmdenv]
runs-to-execute = 1



;**************************************************************************
; Parâmetros de configuração específicos da rede IEEE 802.11.
;**************************************************************************

include ../default.ini

[Parameters]
Net.adapt_type = "cAdapt802_11"
Net.meio_type = "cMeio802_11"
Net.EB_type = "cEB802_11"



;**************************************************************************
; Parâmetros de configuração específicos da rede HIPERLAN/2.
;**************************************************************************

include ../default.ini

[Parameters]
Net.adapt_type = "cAdaptDemand"
Net.meio_type = "cMeioDemand"
Net.EB_type = "cEBDemand"



//****************************************************************************
// Critério de paragem pelo método das replicações.
// Executa a simulação repetidas vezes até atingir a precisão desejada.
//****************************************************************************

#include "omnetpp.h"

#include <iostream.h>   // cout
#include <fstream.h>    // ifstream
#include <stdlib.h>     // system()
#include <direct.h>     // _chdir(), só para Windows

#include <stdio.h>
#include <string.h>

// Máximo 85 repetições com estes parâmetros:
#define NUM_GEN     10
#define DIST_SEED   10000000
// Máximo 31 repetições com esta tabela de t_quantiles
#define MAX_REP     20
#define PREC_MIN    0.1
```

```
int main(int argc, char *argv[])
{
    // Confidence Level 95%
    double t_quantile[] = {0,12.706, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306,\
                              2.262, 2.228, 2.201, 2.179, 2.160, 2.145, 2.131, 2.120,\
                              2.110, 2.101, 2.093, 2.086, 2.080, 2.074, 2.069, 2.064,\
                              2.060, 2.056, 2.052, 2.048, 2.045, 2.042};

    long is, run_ini, run_fin;
    // seed[0] a seed[NUM_GEN - 1] - Usadas na replicação corrente.
    // seed[NUM_GEN] - Vai ser a seed[0] da replicação seguinte.
    long seed[NUM_GEN + 1];
    cStdDev stats;
    double observ[MAX_REP];
    // usar cout com endl (em vez de '\n'), senão não faz flush.
    if ((argc != 2) && (argc != 4))
    {
        cout << "Uso: dyna_sim <rede> [<run_ini> <run_fin>]" << endl;
        exit(1);
    }

    if (argc == 4)
    {
        run_ini = atol(argv[2]);
        run_fin = atol(argv[3]);
    }
    else
    {
        run_ini = 10;
        run_fin = 10;
    }


    char str_aux[100];
    FILE *fseed, *fpar, *fstat;
    float media;// porque o fscanf retorna um float (4 bytes), não um double (8 bytes)
    for (int i = run_ini; i <= run_fin; i++)
    {
        seed[0] = 1;
        stats.clearResult();
        for (int m = 1; m <= MAX_REP; m++)
        {
            sprintf(str_aux, "seedtool g %d %d %d > seed.txt", seed[0], DIST_SEED,
                              NUM_GEN);
            system(str_aux);

            fseed = fopen("seed.txt", "r");

            for (is = 1; is <= NUM_GEN; is++)
                fscanf(fseed, "%d", &seed[is]);

            fclose(fseed);

            printf("\n------------------------------------------------------------\n");
            printf("Replicacao Numero:%3d\n\n", m);

            fpar = fopen("seedsruns.ini", "w");

            fprintf(fpar, "[General]\n");
            for (is = 0; is < NUM_GEN; is++)
            fprintf(fpar, "gen%d-seed = %d\n", is, seed[is]);
            fprintf(fpar, "\n[Cmdenv]\n");
            fprintf(fpar, "runs-to-execute = %d\n", i);

            fclose(fpar);

            // Tem que por "\\" para interpretar como "\",
            // tanto no sprintf como no _chdir.
            // Assim, 4 "\" viram dois, que viram 1.
            sprintf(str_aux, ".\\\\%s", argv[1]);
            _chdir(str_aux);
            system(argv[1]);
```

```
            fstat = fopen("_media.txt", "r");
            fscanf(fstat, "%f", &media);
            printf("\nMedia %d = %.6f\n", m, media);
            stats.collect(media);
            fclose(fstat);
            observ[m-1] = media;

            _chdir("..");

            seed[0] = seed[NUM_GEN];

            if (m >= 3)
            {
                double autocovariance = 0;
                for (int i = 0; i < (m - 1); i++)
                    autocovariance += (observ[i] - stats.mean()) * (observ[i + 1] -
                                stats.mean());
                autocovariance /= (m - 2);
                cout << "Autocovariance = " << autocovariance << endl;
                cout << "Variance = " << stats.variance() << endl;
                cout << "Razao = " << (autocovariance / stats.variance()) << endl;
            }
            if (m >= 2)
            {
                double stats_prec = t_quantile[m - 1] * stats.stddev() / (stats.mean()
                                * sqrt(m));
                printf("\nPrecisao parcial = %5.2f%s, com %d repeticoes.\n", 100.0 *
                                stats_prec, "%", m);
                if (stats_prec < PREC_MIN)
                    break;
            }
        }
        printf("\n----------------------------------------------------------------\n");
        printf("Media Final: %.6f", stats.mean());
        printf("\n----------------------------------------------------------------\n");
    }

    exit(0);
    return 0;
}



//*************************************************************************
// Cabeçalho comum às redes.
//*************************************************************************

#include "omnetpp.h"
#include <string.h>

#ifdef _802_11 // Não aceita "." (ponto) no token
    #include "802.11.h"
#endif
#ifdef _DEMAND
    #include "demand.h"
#endif

#define RE1             1920E+3
#define ATRASO_MAX      0.5
//#define DIM_UL_QUEUE  128000 // 16kbytes (2s a 64kbit/s)
#define DIM_UL_QUEUE512000 // 64kbytes (1.7s a 300kbit/s)
#define NUM_GEN         10
#define NUM_CLASSES     2
#define BER_FACTOR      0.03// 1E-3 => 3E-5

//#define _TRANSIENTE

//#define _DESCARTA_PKT_ATRASO

class cPktInfo : public cMessage
{
public:
    long term_index, seq_num, frag_num, classe;
#ifdef _DEMAND
```

```
    double trr;
#endif

    cPktInfo(char *name = NULL) : cMessage(name) {}     // constructor
    virtual cObject *dup() { return new cPktInfo(*this); }
};

class cGlobal : public cGlobalEspecifico
{
public:
    double FCInterval, ilto;
    // Geradores de números aleatórios
    long BER_ger, BER_trans_ger, contention_ger;
    long IA_ger[NUM_CLASSES], length_ger[NUM_CLASSES], data_trans_ger[NUM_CLASSES];
    // Parâmetros do sistema
    long num_term, num_term1;
    bool retransmite;
    bool display_update_time, stop_criteria_time, transient_criteria_time;
    double wind_ini, wind_collect, display_update;
    double BER_good, BER_bad, t_good, t_bad;
    char trafego1[32], trafego2[32];
    long frag_threshold;

    long seed[NUM_GEN];
    double idle_time;
    char par_str[100]; // parâmetros da simulação para os nomes dos ficheiros

    double pkt_rec[NUM_CLASSES], pkt_sent[NUM_CLASSES];
    long pay_rec[NUM_CLASSES], pay_sent[NUM_CLASSES], pay_ack[NUM_CLASSES];
    long pay_ger[NUM_CLASSES], pay_disc[NUM_CLASSES];

    double start_time;
    long num_exec;
    bool usa_rch[NUM_CLASSES];  // Demand

    static void insereTituloComent(FILE *fa);
    static void insereDadosComent(FILE *fa);
};

extern cGlobal g;

class cAdapt : public cSimpleModule
{
    Module_Class_Members(cAdapt,cSimpleModule,0);

    long i;
    long classe;
    double RLCH, TLCH;
    long ul_pay_length, ul_queue_length;
    cQueue ul_queue, *ul_pay_queue, ul_pkt_queue;
    long max_payload;
    long pkt_seq_num;

    virtual void initialize();

    virtual void handle_UL(cMessage *msg);
    virtual void preenchePayload(bool acrescenta);
    virtual cMessage * composePacket(char *nome, int tipo);
};

class cMeio : public cSimpleModule
{
    Module_Class_Members(cMeio, cSimpleModule, 0);

    long i;
    double *BER_transition, *BER_presente;
    bool *bad_channel;
    cMessage *msg_ant;
    char str_aux[100];

    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};
```

```
class cEB : public cSimpleModule
{
    Module_Class_Members(cEB,cSimpleModule,0)

    long i;
    long reg_term, reg_term2;
    cQueue ul_queue;
    cStdDev *dterm, *bterm; // ponteiros para array de cStdDev
    cStdDev dclasse[NUM_CLASSES], bclasse[NUM_CLASSES];
    cStdDev *dsp_trans, *dsp_collect, *bsp_collect;
    char time_s[30];
    cStdDev nies, nrchs;
    double tidle;
    FILE *fbt, *fdt;
    double display_check, collect_check, observ[32], mean_ant;
    long amostras, samples_ant, num_observ;
    cStdDev stat_observ;
    int src_addr;
    bool passou_inicio;
    cDoubleHistogram *dDist_term, *dDist_term2, *dDist_classe[NUM_CLASSES];
    double d_med_prec;

    virtual void initialize();
    virtual void finish();

    virtual void checkPoint();
    virtual void clearResults();
    virtual void despachaPacote(cMessage *msg);
    virtual void regAtrasos(char *fname, cStdDev dterm, cStdDev bterm);
    virtual void regDistAtrasos(char *fname, cDoubleHistogram *dDist);
    virtual void regPacotes(char *fname, long classe);
    virtual FILE * abre_testa(bool &file_exists, char *fname);
};

char * catStr(char *str1, char *str2);

double iaTime(char *trafego, double data_rate, long data_length, long classe);



//*****************************************************************************
// Cabeçalho específico da rede IEEE 802.11.
//*****************************************************************************

#define _PCF_DCF
//#define _DCF_DCF

//#define _NEGATIVE_ACK // Em caso de erro de CRC. Não está na norma.

//#define _BACKOFF_ANTES// Decrementa backoff (1 vez), mesmo com canal sendo tomado.

#define _ACRESCENTA_PAYLOAD // Ao repetir pacote, acrescenta dados ao payload

//#define _REPETE_NAKS_FIM

//#define _ROLA_LISTA_POLLING

#ifdef _PCF_DCF
    #define MAC "PCF_DCF"
#else
    #define MAC "DCF_DCF"
#endif

class cGlobalEspecifico
{
public:
};

// message kind values (message types):
enum
{
    // Ambos:
        UL,
        DL,
```

```
    // DCF:
        DCF_DADOS,
        ACK,
        SEND_DCF_DADOS,
        SEND_ACK,
        SENSE_CHANNEL,
        ACK_TIMEOUT,
    // PCF:
        POLL,
        RESP,
        CF_END,
        SEND_PKT,
        INI_CFP,
};

// *** Parâmetros do standard IEEE 802.11 ***

// DataOvhLength - Overhead do MAC (Header + Tail) da trama de dados

// ACKLength - Tamanho da trama de ACK (só para o modo DCF)
// No modo PCF a trama de ACK é igual à trama de dados

// PLCPOvhTime - Tempo do overhead do PLCP (Physical Layer Convergence Procedure)
// Independente da taxa de transmissão adoptada
// Ver formato na pp. 13-14 do IEEE Std 802.11b-1999

// SERVICELength - Parte do PLCP dependente da taxa de transmissão
// Existente apenas no standard IEEE 802.11a

//-------------------------------------------------------------------
// Parâmetros comuns aos standards "a" e "b"

#define DataOvhLength   272     // 34 octetos (30 + 4)
#define ACKLength       128     // 16 octetos
#define CFEndLength     192     // 24 octectos

#define MSDUMax     18432   // 2304 octetos
//-------------------------------------------------------------------

/*
//-------------------------------------------------------------------
// Parâmetros do standard IEEE 802.11b
// PHY DSSS, Banda 2.4 GHz

#define REDE"IEEE802.11b"

#define PLCPOvhTime     96E-6   // formato curto
//#define PLCPOvhTime   192E-6  // formato longo

#define SERVICELength   0

#define SlotTime    20E-6
#define SIFSTime    10E-6
#define PIFSTime    20E-6
#define DIFSTime    50E-6

#define CWMin           31      // Valores utilizados
#define CWMax           1023    // no modo DCF

#define PHYRate         1E+6//11E+6
//-------------------------------------------------------------------
*/


//-------------------------------------------------------------------
// Parâmetros do standard IEEE 802.11a
// PHY OFDM, Banda 5 GHz

#define REDE"IEEE802.11a"

#define PLCPOvhTime     20E-6

#define SERVICELength   16
```

```
#define SlotTime    9E-6
#define SIFSTime    16E-6
#define PIFSTime    25E-6
#define DIFSTime    34E-6

#define CWMin           15      // Valores utilizados
#define CWMax           1023    // no modo DCF

#define PHYRate         18E+6   //12E+6
//---------------------------------------------------------------------


// Valores obtidos em função dos parâmetros para utilização na simulação:

// LPHY - Overhead da camada física, convertido em bits.
const long LPHY = (long) (PLCPOvhTime * PHYRate) + SERVICELength;

// TotalOvhLength - Overhead total (PHY + MAC) da trama de dados
const long TotalOvhLength = LPHY + DataOvhLength;

// LOVHACK - Overhead total da trama de ACK (para o modo DCF)
const long LOVHACK = LPHY + ACKLength;

// LOVHCFEND - Overhead total da trama de CF-END (para o modo PCF)
const long LOVHCFEND = LPHY + CFEndLength;

// Tempo máximo de intercâmbio POLL / RESPONSE (para um dado PHYRate)
//const double TPRMAX = 2 * (SIFSTime + PLCPOvhTime + (SERVICELength + DataOvhLength +
MSDUMax) / PHYRate);

#define SFPeriod        6E-3
//#define SFPeriod      24E-3

// TCPmin não influencia os resultados se não for necessário abortar o polling (CFP)
#define TCPmin          1E-3



//*****************************************************************************
// Cabeçalho específico da rede HIPERLAN/2.
//*****************************************************************************

#define REDE"HIPERLAN2"
#define MAC     ""

#define H2_NUM_RCH_INI      1   // Máximo 31
#define H2_RETARDO_CONTENDA 50E-3
#define H2_ESC_LCH_MAX      -1
#define MAX_SEM_CAI         64  //256

#define _CONTENDA_RAPIDA
//#define _CONTENDA_ATRASADA
//#define _SEM_CONTENDA

#define _NUM_RCH_VAR

//#define _IGNORA_RCH_COLIS

#define _REGISTA_RR_DELAY

// Escalonamento para a classe I
#define _FCA
//#define _POLL_ANTES
//#define _RR_COM_LCH
//#define _RR_SO_NO_RCH

//#define _FCA_FLEX // Alocação flexível de capacidade fixa
//#define _FCA_PLUS

#define _C0_SEM_RCH

#define _REP_AUTO_EB

//#define _C0_REPETE_RMIN
```

```
//#define _ESC_ORDEM_TSTAMP

enum {DL_RG, UL_RG};

enum {LCH, SCH};

class cIE
{
public:
    long tipo; // DL_RG, UL_RG
    long mac_id, num_sch_g, num_lch_g;
    // Usando o tempo absoluto em vez do slot, para evitar ter que fazer a conversão
    double start_pointer, RLCH;
};

class cGlobalEspecifico
{
public:
    cIE ie_array[45];
    bool rch_feedback[32];
    double tfim_trama;
    cStdDev *drstall, *astall;
};

// message kind values (message types):
enum
{
        UL,
        DL,
        H2_DIFUSAO,
        H2_DADOS,
        H2_ARQ,
        H2_RR,
        SEND_H2_DIFUSAO,
        SEND_H2_DADOS,
        SEND_H2_ARQ,
        SEND_H2_RR,
        HANDLE_RCH,
        CLEAR_ACH,
        AUTORIZA_CONTENDA
};

#define BCHLength       120     // 15 bytes
#define IEBlockLength   216     // comprimento de 1 IE block do FCH (27 bytes)
#define ACHLength       72      // 9 bytes
#define SCHLength       72      // comprimento do canal SCH (9 bytes)
#define LCHLength       432     // comprimento do canal LCH (54 bytes)
#define FrameTime       2E-3    // período da trama do HIPERLAN/2
#define BCHPreTime      16E-6   // preamble do BCH
#define ULGuardTime     2E-6    // guard time do UL/RCH
#define ULPreTime       16E-6   // preamble do UL/RCH
#define DLPreTime       8E-6    // preamble do DL
#define TurnTime        6E-6    // turnaround time
#define MinRate         6E+6
#define SCHRate         6E+6
#define LCHRate         18E+6
#define H2_KS           512     // Tx/Rx Window Size

const double OneRCHMACTime = ((double) SCHLength) / MinRate;
const double OneRCHTime = ULGuardTime + ULPreTime + OneRCHMACTime;
const double LCHTime = ((double) LCHLength) / LCHRate;
const double SCHTime = ((double) SCHLength) / SCHRate;
const double BCHTime = ((double) BCHLength) / MinRate;
const double IEBlockTime = ((double) IEBlockLength) / MinRate;
const double ACHTime = ((double) ACHLength) / MinRate;

#define DataOvhLength       48 // 1.5 (header LCH) + 1.5 (header CL) + 3 (CRC) octetos



//*****************************************************************************
// Funções comuns às redes.
//*****************************************************************************
```

```
#include "comum.h"

cGlobal g;

void cGlobal::insereTituloComent(FILE *fa)
{
    char str_aux[100];
    fprintf(fa, "%3s", "%");

    fprintf(fa, "%5s%10s%8s", "nt1", "rate1", "length1");
    fprintf(fa, "%5s%10s%8s", "nt2", "rate2", "length2");

    fprintf(fa, "%12s", "BER_bad");
    fprintf(fa, "%12s", "BER_good");
    fprintf(fa, "%12s%12s", "T_bad", "T_good");

    fprintf(fa, "%6s", "ARQ?");

    fprintf(fa, "%10s", "Dim_FIFO");

    fprintf(fa, "%10s", "deadline");

    fprintf(fa, "%15s", "C0_REP_RMIN?");
    fprintf(fa, "%12s", "BER_FACTOR");
#ifdef _DEMAND
    fprintf(fa, "%12s", "RLCH");
    fprintf(fa, "%12s", "RSCH");

    fprintf(fa, "%5s", "#RCH");

    fprintf(fa, "%3s", "%");

    fprintf(fa, "%15s", "C0_SEM_RCH?");
    fprintf(fa, "%15s", "FCInterval");
    fprintf(fa, "%12s", "escal_RR");
    fprintf(fa, "%12s", "ESC_LCH_MAX");
    fprintf(fa, "%12s", "ordem_esc");
    fprintf(fa, "%12s", "contenda");
    fprintf(fa, "%12s", "retardo");
    fprintf(fa, "%15s", "REP_AUTO_EB?");
    fprintf(fa, "%15s", "NUM_RCH_VAR?");
    fprintf(fa, "%12s", "MAX_SEM_CAI");
#else // !_DEMAND
    fprintf(fa, "%12s", "PHYRate");
    fprintf(fa, "%13s", "T_superframe");
    fprintf(fa, "%13s", "frag_tresh");
    fprintf(fa, "%15s", "NEGATIVE_ACK?");
    fprintf(fa, "%15s", "BACKOFF_ANTES?");
    fprintf(fa, "%15s", "ACRESC_PAY?");
    fprintf(fa, "%15s", "REP_NAKS_FIM?");
    fprintf(fa, "%15s", "ROLA_POLLING?");
#endif // !_DEMAND

    fprintf(fa, "%3s", "%");

    fprintf(fa, "%12s%12s", "trans_crit", "wind_ini");

    fprintf(fa, "%12s%12s", "stop_crit", "wind_coll");

    for (int i = 0; i < NUM_GEN; i++)
    {
        sprintf(str_aux, "seed_%d", i);
        fprintf(fa, "%12s", str_aux);
    }

    fprintf(fa, "%10", "trafego1");
    fprintf(fa, "%10", "trafego2");

    fprintf(fa, "\n");
}

void cGlobal::insereDadosComent(FILE *fa)
{
```

```
    fprintf(fa, "%3s", "");

    fprintf(fa, "%5d%10d%8d", g.num_term1, (long) simulation.module(1)-
            >par("data_rate1"), (long) simulation.module(1)->par("data_length1"));
    fprintf(fa, "%5d%10d%8d", g.num_term - g.num_term1, (long) simulation.module(1)-
            >par("data_rate2"), (long) simulation.module(1)->par("data_length2"));

    fprintf(fa, "%12.3e", g.BER_bad);
    fprintf(fa, "%12.3e", g.BER_good);
    if (g.BER_good != g.BER_bad)
        fprintf(fa, "%12.6f%12.6f", g.t_bad, g.t_good);
    else
        fprintf(fa, "%12s%12s", "-1", "-1");

    fprintf(fa, "%6d", g.retransmite);

    fprintf(fa, "%10d", DIM_UL_QUEUE);

#ifdef _DESCARTA_PKT_ATRASO
    fprintf(fa, "%10.3f", ATRASO_MAX);
#else
    fprintf(fa, "%10s", "-1");
#endif

#ifdef _C0_REPETE_RMIN
    fprintf(fa, "%15s", "1");
#else
    fprintf(fa, "%15s", "0");
#endif
    fprintf(fa, "%12.6f", BER_FACTOR);

#ifdef _DEMAND
    fprintf(fa, "%12d", (long) LCHRate);
    fprintf(fa, "%12d", (long) SCHRate);

    fprintf(fa, "%5d", H2_NUM_RCH_INI);

    fprintf(fa, "%3s", "%");

#ifdef _C0_SEM_RCH
    fprintf(fa, "%15s", "1");
#else
    fprintf(fa, "%15s", "0");
#endif

    fprintf(fa, "%15.6f", g.FCInterval);

#ifdef _FCA
    #ifdef _FCA_FLEX
        fprintf(fa, "%12s", "FCA_FLEX");
    #elif defined _FCA_PLUS
        fprintf(fa, "%12s", "FCA_PLUS");
    #else
        fprintf(fa, "%12s", "FCA_NORMAL");
    #endif
#endif
#ifdef _POLL_ANTES
    fprintf(fa, "%12s", "POLL_ANTES");
#endif
#ifdef _RR_COM_LCH
    fprintf(fa, "%12s", "RR_com_LCH");
#endif
#ifdef _RR_SO_NO_RCH
    fprintf(fa, "%12s", "RR_só_RCH");
#endif
    fprintf(fa, "%12d", H2_ESC_LCH_MAX);

#ifdef _ESC_ORDEM_TSTAMP
    fprintf(fa, "%12s", "timestamp");
#else
    fprintf(fa, "%12s", "chegada");
#endif

#ifdef _CONTENDA_RAPIDA
```

```
    fprintf(fa, "%12s", "rapida");
    fprintf(fa, "%12d", 0);
#endif
#ifdef _CONTENDA_ATRASADA
    fprintf(fa, "%12s", "atrasada");
    fprintf(fa, "%12.3e", H2_RETARDO_CONTENDA);
#endif
#ifdef _SEM_CONTENDA
    fprintf(fa, "%12s", "sem");
    fprintf(fa, "%12d", 0);
#endif

#ifdef _REP_AUTO_EB
    fprintf(fa, "%15s", "1");
#else
    fprintf(fa, "%15s", "0");
#endif

#ifdef _NUM_RCH_VAR
    fprintf(fa, "%15s", "1");
#else
    fprintf(fa, "%15s", "0");
#endif

    fprintf(fa, "%12d", MAX_SEM_CAI);

#else // !_DEMAND
    fprintf(fa, "%12d", (long) PHYRate);

    fprintf(fa, "%13.3e", SFPeriod);

    fprintf(fa, "%13d", g.frag_threshold);

#ifdef _NEGATIVE_ACK
    fprintf(fa, "%15s", "1");
#else
    fprintf(fa, "%15s", "0");
#endif

#ifdef _BACKOFF_ANTES
    fprintf(fa, "%15s", "1");
#else
    fprintf(fa, "%15s", "0");
#endif

#ifdef _ACRESCENTA_PAYLOAD
    fprintf(fa, "%15s", "1");
#else
    fprintf(fa, "%15s", "0");
#endif

#ifdef _REPETE_NAKS_FIM
    fprintf(fa, "%15s", "1");
#else
    fprintf(fa, "%15s", "0");
#endif

#ifdef _ROLA_LISTA_POLLING
    fprintf(fa, "%15s", "1");
#else
    fprintf(fa, "%15s", "0");
#endif
#endif // !_DEMAND

    fprintf(fa, "%3s", "%");

    if (g.transient_criteria_time)
        fprintf(fa, "%12s%12.3f", "tempo", g.wind_ini);
    else
        fprintf(fa, "%12s%10.0fxN", "amostras", g.wind_ini);

    if (g.stop_criteria_time)
        fprintf(fa, "%12s%12.3f", "tempo", g.wind_collect);
    else
```

```
        fprintf(fa, "%12s%12.0f", "amostras", g.wind_collect);

    for (int i = 0; i < NUM_GEN; i++)
        fprintf(fa, "%12d", g.seed[i]);

    fprintf(fa, "%10s", g.trafego1);
    fprintf(fa, "%10s", g.trafego2);

    fprintf(fa, "\n");
}

double iaTime(char *trafego, double data_rate, long data_length, long classe)
{
    if (!strcmp(trafego, "Poisson"))
        return data_length * genk_exponential(g.IA_ger[classe], 1 / data_rate);
    else if (!strncmp(trafego, "CBR", 3))
        return data_length / data_rate;
    else
    {
        printf("Erro: Tipo de trafego invalido - %s\n", trafego);
        exit(1);
    }
}

char * catStr(char *str1, char *str2)
{
    strcpy(g.par_str, str1);
    strcat(g.par_str, str2);
    return g.par_str;
}



//*************************************************************************
// Módulo terminal.
//*************************************************************************

#include "comum.h"

class cTerm : public cSimpleModule
{
    Module_Class_Members(cTerm,cSimpleModule,0)

    long seq_num, term_index, classe;
    char trafego[32];
    double data_rate;
    long data_length;
    double ia_time;

    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

Define_Module(cTerm);

void cTerm::initialize()
{
    term_index = this->index();  // ou apenas index();

    seq_num = 0;
    if (index() < g.num_term1)
    {
        strcpy(trafego, g.trafego1);
        data_rate = simulation.module(1)->par("data_rate1");
        data_length = simulation.module(1)->par("data_length1");
        classe = 0;
    }
    else
    {
        strcpy(trafego, g.trafego2);
        data_rate = simulation.module(1)->par("data_rate2");
        data_length = simulation.module(1)->par("data_length2");
        classe = 1;
    }
```

```
    if (!strcmp(trafego, "CBR_UNI")) // Atraso inicial uniforme de 0 a IAT
        scheduleAt(simTime() + genk_uniform(g.IA_ger[classe], 0, 1) * iaTime(trafego,
                    data_rate, data_length, classe), new cMessage);
    else if (!strcmp(trafego, "CBR_UNI_N")) // Atraso inicial uniforme de 0 a IAT x
                    num_term1
        scheduleAt(simTime() + g.num_term1 * genk_uniform(g.IA_ger[classe], 0, 1) *
                    iaTime(trafego, data_rate, data_length, classe), new cMessage);
    else if (!strcmp(trafego, "CBR_SINC_N")) // Atraso inicial proporcional ao índice
        scheduleAt(simTime() + index() * iaTime(trafego, data_rate, data_length,
                    classe), new cMessage);
    else if (!strcmp(trafego, "CBR_SINC")) // Atraso inicial zero
        scheduleAt(simTime(), new cMessage);
    else// Tráfego Poisson e outros
        scheduleAt(simTime() + iaTime(trafego, data_rate, data_length, classe), new
                    cMessage);
}

void cTerm::handleMessage(cMessage *msg)
{
    char msg_name[32];
    sprintf(msg_name, "UL %d", seq_num);
    cPktInfo *ul = new cPktInfo;
    ul->setKind(UL);
    ul->term_index = term_index;
    ul->seq_num = seq_num;
    ul->frag_num = 0;
    ul->classe = classe;
    ul->setLength(data_length);
    ul->setTimestamp();
    send(ul, "out");
    seq_num++;

    scheduleAt(simTime() + iaTime(trafego, data_rate, data_length, classe) , msg);
}



//*****************************************************************************
// Módulo concentrador/multiplexador.
//*****************************************************************************

#include "comum.h"
#include <stdio.h>

class cMUX : public cSimpleModule
{
    Module_Class_Members(cMUX,cSimpleModule,0)

    cModule *sistema;

    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

Define_Module(cMUX);

void cMUX::initialize()
{
    sistema = simulation.module(1);
    // Parâmetros de configuração da simulação
    g.retransmite = sistema->par("retransmite");
    g.BER_good = sistema->par("BER_good");
    g.BER_bad = sistema->par("BER_bad");
    if (g.BER_good > g.BER_bad)
        error("BER good > BER bad");
    g.t_good = sistema->par("t_good");
    g.t_bad = sistema->par("t_bad");
    g.num_term1 = sistema->par("num_term1");
    g.num_term = g.num_term1 + (long) sistema->par("num_term2");
    g.wind_ini = sistema->par("wind_ini");
    g.wind_collect = sistema->par("wind_collect");
    strcpy(g.trafego1, sistema->par("trafego1"));
    strcpy(g.trafego2, sistema->par("trafego2"));
    g.display_update_time = sistema->par("display_update_time");
```

```
    g.transient_criteria_time = sistema->par("transient_criteria_time");
    g.stop_criteria_time = sistema->par("stop_criteria_time");
    g.display_update = sistema->par("display_update");
    g.frag_threshold = sistema->par("frag_threshold");

    printf("  %s", REDE);
    if (strlen(MAC) > 0)
        printf(" - %s", MAC);
    printf("\n  Numero terminais 1 = %d \t\t", g.num_term1);
    printf("Numero terminais 2 = %d\n", g.num_term - g.num_term1);

    // Armazena seeds iniciais para registo posterior nos finish().
    // Apresenta os valores usados para as seeds no display.
    printf("  seeds:");
    for (int i = 0; i < NUM_GEN; i++)
    {
        g.seed[i] = genk_randseed(i);
        printf("%11d", g.seed[i]);
    }
    cout << endl;

    g.BER_ger = 0;
    g.BER_trans_ger = 1;
    g.contention_ger = 2;
    g.IA_ger[0] = 4;
    g.length_ger[0] = 5;
    g.data_trans_ger[0] = 6;
    g.IA_ger[1] = 7;
    g.length_ger[1] = 8;
    g.data_trans_ger[1] = 9;

    long max_payload1;
    if (((long) simulation.module(1)->par("data_length1")) == 424) // ATM
        max_payload1 = 424;
    else
        max_payload1 = 384; // 48 octectos

    // ILTO (Inter LCH Time Óptimo)
    g.ilto = max_payload1 / (double) sistema->par("data_rate1");

#ifdef _DEMAND
    #ifdef _FCA_FLEX
        // Usa intervalo ideal
        g.FCInterval = g.ilto;
    #else
        // Usa intervalo múltiplo de 2 ms
        g.FCInterval = FrameTime * floor((g.ilto + 1E-6)/ FrameTime);
        cout << g.FCInterval << endl;
    #endif
#endif
}

void cMUX::handleMessage(cMessage *msg)
{
    int msg_type = msg->kind();
    switch (msg_type)
    {
    case UL:
    {
        delete msg;
    }
    break;
    default:
        error("Mensagem invalida no MUX: %d", msg_type);
    }
}



//***************************************************************************
// Módulo adaptador de terminal (parte comum).
//***************************************************************************

#include "comum.h"
```

```
void cAdapt::initialize()
{
    ul_queue_length = 0;
    ul_pay_length = 0;

    pkt_seq_num = 0;

    if (index() < g.num_term1)
        classe = 0;
    else
        classe = 1;
}

void cAdapt::handle_UL(cMessage *msg)
{
    cPktInfo *ul = (cPktInfo *) msg;

#ifdef _DESCARTA_PKT_ATRASO
    cPktInfo *ul_aux;
    while (!ul_queue.empty())
    {
        ul_aux = (cPktInfo *) ul_queue.tail();
        if ((simTime() - ul_aux->timestamp()) < ATRASO_MAX)
            break;
        // cout << ul_queue.length() << endl;
        g.pay_disc[classe] += ul_aux->length();
        ul_queue_length -= ul_aux->length();
        delete ul_queue.pop();
    }
#endif

#ifdef _DEMAND
    ((cPktInfo *) ul)->trr = 0;
#endif

    g.pay_ger[classe] += ul->length();

    if (ul_queue_length + ul->length() < DIM_UL_QUEUE)
    {
        ul_queue_length += ul->length();
        if (ul->length() > max_payload) // Tem que segmentar
        {
            long extras = (ul->length() / max_payload);
            long resto = (ul->length() % max_payload);
            if (resto == 0)
            {
                extras--;
                if (ul->length() > 0)
                    resto = max_payload;
                else
                    resto = ul->length();
            }
            for (i = extras; i > 0; i--)
            {
                cPktInfo *frag = (cPktInfo *) ul->dup();
                frag->setLength(max_payload);
                frag->frag_num = i;
                ul_queue.insert(frag);
            }
            ul->setLength(resto);
        }
        ul_queue.insert(ul);
    }
    else
    {
        g.pay_disc[classe] += ul->length();
        delete ul;
    }
}

void cAdapt::preenchePayload(bool acrescenta)
{
    if (acrescenta)
```

```
    {
        ul_pay_queue = (cQueue *) (cObject *) ((cMessage *) ul_pkt_queue.tail())-
                        >par("pay_queue");
        ul_pay_length = (long) ((cMessage *) ul_pkt_queue.tail())->par("pay_length");
    }
    else
    {
        ul_pay_queue = new cQueue;
        ul_pay_length = 0;
    }

    cPktInfo *ul;
    while (!ul_queue.empty())
    {
        ul = (cPktInfo *) ul_queue.tail();
        if (max_payload < (ul_pay_length + ul->length()))
            break;
        ul = (cPktInfo *) ul_queue.pop();
        ul_queue_length -= ul->length();
        ul_pay_queue->insert(ul);
        ul_pay_length += ul->length();
    }

    if (acrescenta)
    {
        ((cMessage *) ul_pkt_queue.tail())->par("pay_length") = ul_pay_length;
    #ifndef _DEMAND
        ((cMessage *) ul_pkt_queue.tail())->setLength(TotalOvhLength + ul_pay_length);
    #endif
    }
}

cMessage * cAdapt::composePacket(char *nome, int tipo)
{
    cMessage *ul_pkt = new cMessage(NULL, tipo);
    ul_pkt->addPar("dest_addr") = g.num_term;
    cPar *par_aux = new cPar("pay_queue");
    par_aux->takeOwnership(true);
    par_aux->setObjectValue(ul_pay_queue);
    ul_pkt->addPar(par_aux);
    ul_pkt->addPar("pay_length") = ul_pay_length;
    // seq_num é usado no mecanismo de ARQ (do HIPERLAN/2)
    ul_pkt->addPar("seq_num") = pkt_seq_num;
    pkt_seq_num++;
    ul_pkt->addPar("classe") = classe;

    char dname[32];
#ifdef _DEMAND
    // dest_addr e mac_id são implíctitos, mas sua
    // inclusão na msg facilita o processamento
    ul_pkt->addPar("mac_id") = this->index();
    sprintf(dname, "%s %d", nome, pkt_seq_num);
    ul_pkt->setLength(LCHLength);
#else
    ul_pkt->addPar("src_addr") = this->index();
    sprintf(dname, "%s(%d)", nome, ul_pay_queue->length());
    ul_pkt->setLength(TotalOvhLength + ul_pay_length);
#endif
    ul_pkt->setName(dname);

    return ul_pkt;
}



//*****************************************************************************
// Módulo adaptador de terminal (parte específica da rede IEEE 802.11).
//*****************************************************************************

#include "comum.h"

class cAdapt802_11 : public cAdapt
{
    Module_Class_Members(cAdapt802_11, cAdapt, 0);
```

```
    // DCF
    int CW;
    bool contending, fez_backoff, esperou_slot_time;
    double backoff;
    cMessage *send_data, *sense_channel, *send_ack, *ack_timeout;
    double idle_time_ant;

    // PCF
    bool wait_ack;
    cMessage *send_pkt;

    bool sack, rack; // sack não é usado (adapt não recebe dados)

    virtual void initialize();
    virtual void handleMessage(cMessage *msg);

    virtual void sendULPacket(char *nome, int tipo);
};

Define_Module_Like(cAdapt802_11, cAdapt);

void cAdapt802_11::initialize()
{
    cAdapt::initialize();

    // Ambos
    if ((g.frag_threshold < MSDUMax) && (g.frag_threshold > 0))
        max_payload = g.frag_threshold;
    else
        max_payload = MSDUMax;
    // rack tem que começar true para o envio da primeira mensagem em send_UL_packet()
    rack = true;

    // DCF
    CW = CWMin;
    contending = false;
    send_data = new cMessage(NULL, SEND_DCF_DADOS);
    send_ack = new cMessage(NULL, SEND_ACK);
    sense_channel = new cMessage(NULL, SENSE_CHANNEL);
    ack_timeout = new cMessage(NULL, ACK_TIMEOUT);
    esperou_slot_time = false;

    // PCF
    wait_ack = false;
    send_pkt = new cMessage(NULL, SEND_PKT);
}

void cAdapt802_11::handleMessage(cMessage *msg)
{
    int msg_type = msg->kind();
    switch (msg_type)
    {
    case UL: // só pode vir de term
    {
        ev << "UL" << endl;

        handle_UL(msg);

    #ifdef _PCF_DCF
        if (classe == 1)
        {
    #endif
        if (!contending)
        {
            contending = true;
            if (g.idle_time < simTime())  // canal livre
            {
                backoff = 0;
                fez_backoff = false;
                scheduleAt(simTime() + DIFSTime, sense_channel);
            }
            else
            {
```

```
            backoff = genk_intuniform(g.contention_ger, 0, CW);
            fez_backoff = true;
            scheduleAt(g.idle_time + DIFSTime, sense_channel);
        }
        idle_time_ant = g.idle_time;
    }
#ifdef _PCF_DCF
    }
#endif
}
break;
case SENSE_CHANNEL: // DCF
{
    ev << "SENSE_CHANNEL" << endl;
    if (g.idle_time == idle_time_ant)  // Canal livre durante DIFSTime/SlotTime
    {

    #ifndef _BACKOFF_ANTES
        if (esperou_slot_time)
        {
            backoff--;
            esperou_slot_time = false;
        }
    #endif
        if (backoff == 0)
            scheduleAt(simTime(), send_data);
        else
        {
    #ifdef _BACKOFF_ANTES
            backoff--;
    #endif
            scheduleAt(simTime() + SlotTime, sense_channel);
            esperou_slot_time = true;
        }
    }
    else if (g.idle_time > idle_time_ant)
    // Alguma transmissão ocupou (ou ainda ocupa) o canal
    {
        esperou_slot_time = false;
        scheduleAt(g.idle_time + DIFSTime, sense_channel);
        if (!fez_backoff)
        {
            backoff = genk_intuniform(g.contention_ger, 0, CW);
            fez_backoff = true;
        }
    }
    else
        error("g.idle_time < idle_time_ant - Não pode antecipar o g.idle_time");
    idle_time_ant = g.idle_time;
}
break;
case DCF_DADOS: // DCF (Não ocorre)
{
    ev << "DCF_DADOS" << endl;
    if ((int) msg->par("dest_addr") == this->index())
        scheduleAt(simTime() + SIFSTime, send_ack);
    if (!msg->hasBitError())
    {
        sack = true;
    }
    else
        sack = false;
    // repassa DLs para os terminais aqui
    delete msg;
}
break;
case ACK: // DCF
{
    ev << "ACK" << endl;
    cancelEvent(ack_timeout);
    CW = CWMin;
    if ((int) msg->par("dest_addr") == this->index())
        rack = msg->par("ack");
    if (((g.retransmite) && (!rack)) || (!ul_queue.empty()))
```

```
            // NAK ou mais mensagens a enviar
        {
            backoff = genk_intuniform(g.contention_ger, 0, CW);
            fez_backoff = true;
            scheduleAt(simTime() + DIFSTime, sense_channel);
            idle_time_ant = g.idle_time;
        }
        else
            contending = false; // espera novas mensagens
        delete msg;
    }
    break;
    case ACK_TIMEOUT: // DCF
        // Colisão ou recepção com erro (!_NEGATIVE_ACK)
    {
        ev << "ACK_TIMEOUT" << endl;
        rack = false;
        CW = 2 * (CW + 1) - 1;
        if (CW > CWMax)
            CW = CWMax;
        if ((g.retransmite) || (!ul_queue.empty()))
        {
            backoff = genk_intuniform(g.contention_ger, 0, CW);
            fez_backoff = true;
            double tlivre = ((g.idle_time > simTime()) ? g.idle_time : simTime());
            scheduleAt(tlivre + DIFSTime, sense_channel);
            idle_time_ant = g.idle_time;
        }
        else
            contending = false; // espera novas mensagens
    }
    break;
    case SEND_DCF_DADOS: // DCF
    {
        ev << "SEND_DCF_DADOS" << endl;
        sendULPacket("DCF_DADOS", DCF_DADOS);
        scheduleAt(simTime() + (((double) ul_pay_length + TotalOvhLength + LOVHACK) /
                    PHYRate) + SIFSTime + SlotTime, ack_timeout);
    }
    break;
    case SEND_ACK: // DCF (Não ocorre)
    {
        ev << "SEND_ACK" << endl;
        cMessage *ack_pkt;
        ack_pkt = new cMessage(NULL, ACK, LOVHACK);
        if (sack)
            ack_pkt->setName("ACK");
        else
            ack_pkt->setName("NAK");
        ack_pkt->addPar("src_addr") = this->index();
        ack_pkt->addPar("dest_addr") = g.num_term;
        ack_pkt->addPar("ack") = sack;
        send(ack_pkt, "to_meio");
    }
    break;
    case POLL: // PCF
    {
        if ((long) msg->par("dest_addr") == this->index())
            scheduleAt(simTime() + SIFSTime, send_pkt);
        if (wait_ack)
        {
            rack = msg->par("ack");
            wait_ack = false;
        }
        delete msg;
    }
    break;
    case CF_END: // PCF
    {
        if (wait_ack)
        {
            rack = msg->par("ack");
            wait_ack = false;
        }
```

```
        delete msg;
    }
    break;
    case SEND_PKT: // PCF
    {
        sendULPacket("RESP", RESP);
        wait_ack = true;
    }
    break;
    case RESP: // PCF
    {
        delete msg;
    }
    break;
    default:
        error("Mensagem invalida no adapt: %d", msg_type);
    }
}


void cAdapt802_11::sendULPacket(char *nome, int tipo)
{
    cMessage *ul_pkt;
    if (!g.retransmite)
    {
        preenchePayload(false);
        ul_pkt = composePacket(nome, tipo);
    }
    else
        if (rack)
        {
            ul_pkt_queue.clear();
            preenchePayload(false);
            ul_pkt = composePacket(nome, tipo);
            ul_pkt_queue.insert(ul_pkt->dup());
        }
        else // repete acrescentando
        {
            #ifdef _ACRESCENTA_PAYLOAD
                preenchePayload(true);
            #endif
            ul_pkt = (cMessage *) ul_pkt_queue.tail()->dup();
        }

    send(ul_pkt, "to_meio");
}



//*****************************************************************************
// Módulo adaptador de terminal (parte específica da rede HIPERLAN/2).
//*****************************************************************************

#include "comum.h"

class cAdaptDemand : public cAdapt
{
    Module_Class_Members(cAdaptDemand, cAdapt, 0);

    long TxBoWs;
    cLinkedList NAK_list;
    long num_lch_r, num_lch_r_t, data_length, bmax;
    cMessage *send_h2_rr, *send_h2_dados, *handle_rch;
    long num_ieb, num_ie, num_rch, h2_a, h2_ra, rch_index, num_lch_g, num_sch_g;
    double tlast_rr;
    long rr_counter, rch_counter;
    bool contenda_autorizada;
    cMessage *autoriza_contenda;
    double FCPointer;

    virtual void initialize();
    virtual void handleMessage(cMessage *msg);

    virtual void processa_dr();
```

```
    virtual long numLCH();
};

Define_Module_Like(cAdaptDemand, cAdapt);

void cAdaptDemand::initialize()
{
    cAdapt::initialize();

    //FCPointer = FrameTime * floor(((g.FCInterval + 1E-12) * index()) / (g.num_term1
                    * FrameTime));
    FCPointer = FrameTime * floor(((FrameTime * floor((g.FCInterval + 1E-12) /
                    FrameTime) + 1E-12) * index()) / (g.num_term1 * FrameTime));
    //cout << index() << "\t" << FCPointer << endl;

    TxBoWs = 0;

    contenda_autorizada = true;
    num_lch_r = 0;

    if (index() < g.num_term1)
        data_length = (long) simulation.module(1)->par("data_length1");
    else
        data_length = (long) simulation.module(1)->par("data_length2");

    if (data_length == 424) // ATM
        max_payload = 424;
    else
        max_payload = 384;  // 48 octetos

    if (data_length < max_payload)
        bmax = max_payload / data_length;
    else
        bmax = 1;
    rr_counter = this->index() + 1;
    rch_counter = 0;
    rch_index = -1;
    h2_a = 0;
    h2_ra = -1; // desactivado
    send_h2_rr = new cMessage(NULL, SEND_H2_RR);
    send_h2_dados = new cMessage(NULL, SEND_H2_DADOS);
    handle_rch = new cMessage(NULL, HANDLE_RCH);
    autoriza_contenda = new cMessage(NULL, AUTORIZA_CONTENDA);
}

void cAdaptDemand::handleMessage(cMessage *msg)
{
    int msg_type = msg->kind();
    switch (msg_type)
    {
    case UL: // só pode vir de term
    {
        ev << "ADAPT: UL" << endl;
        handle_UL(msg);
    }
    break;
    case H2_DIFUSAO:
    {
        ev << "ADAPT: BROADCAST" << endl;
        num_rch = msg->par("num_rch");
        num_ie = msg->par("num_ie");
        //cout << num_rch << '\t' << num_ie << endl;
        if (num_ie > 0) // vê se tem RGs para este terminal
            for (int i = 0; i < num_ie; i++)
                if (g.ie_array[i].mac_id == this->index())
                    if (g.ie_array[i].tipo == UL_RG)
                    {
                        num_lch_g = g.ie_array[i].num_lch_g;
                        RLCH = g.ie_array[i].RLCH;
                        TLCH = ((double) LCHLength) / g.ie_array[i].RLCH;
                        num_sch_g = g.ie_array[i].num_sch_g;
                        if (num_sch_g > 1)
                            error("num_sch_g > 1");
                        scheduleAt(g.ie_array[i].start_pointer, send_h2_dados);
```

```
                    break;
                }
                /*
                else if (g.ie_array[i].tipo == DL_RG)
                    cout << "DL_RG" << endl;
                else
                    error("Tipo não definido!");
                */

    // Verifica recebimento do RR no RCH (pode ter havido colisão),
    // em caso negativo, incrementa "a" (contador de retransmissões).
    // Regista estatísticas de "a" e do RR delay.
    if (rch_index > 0) // se enviou RR no RCH na trama anterior ...
    {
        if (g.rch_feedback[rch_index] == false) // fracasso
        {
            #ifdef _SEM_CONTENDA
                error("Perdeu RR no modo sem contenda!");
            #else
                //cout << "EB nao recebeu RR anterior" << endl;
                h2_a++;
            #endif
        }
        else // sucesso
        {
            #ifdef _CONTENDA_ATRASADA
                contenda_autorizada = false;
                scheduleAt(simTime() + H2_RETARDO_CONTENDA, autoriza_contenda);
            #endif
                //cout << "EB recebeu RR na posição " << rch_index << endl;
                g.astall->collect(h2_a);
            #ifdef _REGISTA_RR_DELAY
                processa_dr();
            #endif
                h2_a = 0;
                num_lch_r = num_lch_r_t;
        }
        rch_index = -1;
    }


    #ifdef _FCA
        if ((classe == 0) && (simTime() > FCPointer))
        {
            FCPointer += g.FCInterval;
        }
    #endif


    // Escalona (restante do ) HANDLE_RCH no fim da UL phase
    if ((num_rch > 0) && (g.usa_rch[classe]))
        scheduleAt(g.tfim_trama - num_rch * OneRCHTime, handle_rch);
    delete msg;
}
break;
case HANDLE_RCH:
{
    ev << "ADAPT: HANDLE_RCH" << endl;
    #ifdef _SEM_CONTENDA
        h2_ra = rr_counter - rch_counter;
        if (h2_ra <= num_rch)
        {
            if ((numLCH() > num_lch_r))
            {
                double trr = g.tfim_trama - OneRCHMACTime - (num_rch - h2_ra) *
                                OneRCHTime;
                scheduleAt(trr, send_h2_rr);
            }
            rr_counter += g.num_term;
        }
        rch_counter += num_rch;
    #else // !_SEM_CONTENDA
        long h2_CWa;
        // Calcula novo Ra (posição de transmissão do RR),
```

```
        // se ainda não tiver calculado (h2_ra < 0) e
        // se chegaram dados novos dos terminais desde o último RR com sucesso.
    #ifdef _CONTENDA_RAPIDA
        if ((numLCH() > num_lch_r) && (h2_ra < 0))
    #endif // _CONTENDA_RAPIDA
    #ifdef _CONTENDA_ATRASADA
        if ((numLCH() > num_lch_r) && (contenda_autorizada) && (h2_ra < 0))
    #endif // _CONTENDA_ATRASADA
        {
            // Algoritmo especificado na norma HIPERLAN/2
            if (h2_a == 0)
                h2_CWa = num_rch;
            else
            {
                long pow2_a = (long) pow(2, h2_a);
                if (pow2_a < num_rch)
                    h2_CWa = num_rch;
                else if (pow2_a > 256)
                    h2_CWa = 256;
                else
                    h2_CWa = pow2_a;
            }
            h2_ra = (long) genk_intuniform(g.contention_ger, 1, h2_CWa);
            //cout << "CWa = " << h2_CWa << "\tra = " << h2_ra << endl;
        }

        if (h2_ra > 0)  // se tem RR a enviar ...
        {
            if (h2_ra <= num_rch)
            {
                // Escalona RR nesta trama
                double trr = g.tfim_trama - OneRCHMACTime - (num_rch - h2_ra) *
                                OneRCHTime;
                scheduleAt(trr, send_h2_rr);
            }
            else
                h2_ra -= num_rch;
        }
    #endif // !_SEM_CONTENDA
}
break;
case SEND_H2_RR:
{
    ev << "ADAPT: SEND_H2_RR" << endl;
    char msg_name[32];

    num_lch_r_t = numLCH();
    if (num_lch_r_t <= 0)
        cout << index() << " requisitou " << num_lch_r_t << endl;

    sprintf(msg_name, "RR(%d)", num_lch_r_t);
    cMessage *rr = new cMessage(msg_name, H2_RR, SCHLength);
    // incluído na msg para facilitar processamento
    rr->addPar("dest_addr") = g.num_term;
    // h2_ra : "-1" (desactivado), "0" (RR no SCH), "> 0" (RR no RCH).
    rr->addPar("rch_index") = rch_index = h2_ra;
    if (rch_index > 0)
        rr->addPar("rate") = MinRate;
    else
        rr->addPar("rate") = SCHRate;
    rr->addPar("mac_id") = this->index();
    rr->addPar("num_lch_r") = num_lch_r_t;
    rr->addPar("classe") = classe;

    #ifdef _REP_AUTO_EB
        if (ul_queue.empty())
        {
            cout << "OPA!" << endl;
            endSimulation();
            exit(0);
        }
        rr->addPar("tstamp") = ((cMessage *) ul_queue.tail())->timestamp();
    #else
        rr->addPar("tstamp") = 0;
```

```
    #endif

    send(rr, "to_meio");
    tlast_rr = simTime();

    if (rch_index == 0)  // RR no SCH
    {
        num_lch_r = num_lch_r_t;
        #ifdef _REGISTA_RR_DELAY
            processa_dr();  // Senão processa após confirmação do recebimento
        #endif
    }

    h2_ra = -1;
}
break;
case SEND_H2_DADOS:
{
    ev << "ADAPT: SEND_H2_DADOS" << endl;
    cMessage *dados;
    for (int i = 0; i < num_lch_g; i++)
    {
        if (!NAK_list.empty()) // Reenvia mensagem de dados em erro
        {
            #ifndef _REP_AUTO_EB
                num_lch_r--;
            #endif
            long *NAK = (long *) NAK_list.pop();
            for (cQueueIterator iter(ul_pkt_queue, 0); !iter.end(); iter--)
            {
                dados = (cMessage *) iter();
                if (((long) dados->par("seq_num")) == *NAK)
                {
                    cMessage *dados_dup = (cMessage *) dados->dup();
                    dados_dup->addPar("RPHY") = RLCH;
                    sendDelayed(dados_dup, i * TLCH, "to_meio");
                    break;
                }
            }
            delete NAK;
        }
        else if (!ul_queue.empty()) // Envia mensagem de dados nova
        {
            num_lch_r--;
            preenchePayload(false);
            dados = composePacket("DADOS", H2_DADOS);
            if (g.retransmite)
                ul_pkt_queue.insert(dados->dup());

            dados->addPar("RPHY") = RLCH;
            sendDelayed(dados, i * TLCH, "to_meio");
        }
        else // Envia mensagem vazia (serve de aviso ao EB)
        {
            #ifdef _FCA_PLUS
                FCPointer = g.tfim_trama;
            #endif
            num_lch_r--;
            dados = new cMessage("DUMMY", H2_DADOS);
            dados->setLength(LCHLength);
            dados->addPar("seq_num") = -1;
            dados->addPar("mac_id") = this->index();
            dados->addPar("dest_addr") = g.num_term;
            dados->addPar("classe") = classe;

            dados->addPar("RPHY") = RLCH;
            sendDelayed(dados, i * TLCH, "to_meio");
            cout << index() << " enviou DUMMY" << endl;
        }
    }
    // RR no SCH
    if ((num_sch_g > 0) && (numLCH() > num_lch_r))
    {
        // Cancela possível contenda em curso no RCH e ao mesmo tempo
```

```
            // sinaliza à EB que este RR é transmitido no SCH
            // (com addPar("rch_index") = rch_index = h2_ra)
            h2_ra = 0;
            h2_a = 0;  // Para o caso de estar no meio de uma retransmissão.
            scheduleAt(simTime() + num_lch_g * TLCH, send_h2_rr);
        }
    }
    break;
    case H2_ARQ:
    {
        // msg->par("mac_id") == this->index(), pois o meio só manda para este adapt.
        long ACK = msg->par("ACK");
        if (msg->hasPar("NAK_list"))
        {
            cLinkedList *list = (cLinkedList *) (cObject *) msg->par("NAK_list");
            if (list->length() < NAK_list.length())
                error("NAK_list.length(): EB < Adapt");
            while (!NAK_list.empty())
                delete NAK_list.pop();
            while (!list->empty())
                NAK_list.insert(list->pop());
        }

        if (((long) msg->par("CAI")) == 1)
        {
            while (!ul_pkt_queue.empty())
            {
                long seq_num = (long) ((cMessage *) ul_pkt_queue.tail())-
                                >par("seq_num");
                if (seq_num < ACK)
                    delete ul_pkt_queue.pop();
                else
                    break;
            }
            if (TxBoWs > ACK) // Não deve acontecer com CAI = 1
            {
                if (msg->hasPar("NAK"))
                    cout << "Do NAK" << endl;
                error("TxBoWs > ACK (%d > %d)", TxBoWs, ACK);
            }
            TxBoWs = ACK;
        }

        delete msg;
    }
    break;
    case AUTORIZA_CONTENDA:
    {
        contenda_autorizada = true;
    }
    break;
    default:
        error("Mensagem invalida no adapt: %d", msg_type);
    }
}

void cAdaptDemand::processa_dr()
{
    cPktInfo *ul;
    cQueueIterator iter(ul_queue, 0);
    while (!iter.end())
    {
        ul = (cPktInfo *) iter();
        if (ul->trr > 0)
            for (int i = 0; i < bmax; i++)
            {
                iter--;
                if (iter.end())
                    break; // for
            }
        else
            if (ul->timestamp() < tlast_rr)
            {
                ul->trr = tlast_rr;
```

```
                iter--;
            }
            else
                break; //while
    }
}

// numLCH() retorna o número de canais LCH necessários para transmissão
// do tráfego em espera, sujeito à limitação do controlo de fluxo.
long cAdaptDemand::numLCH()
{
    long janela;
    if (g.retransmite)
        janela = H2_KS - (pkt_seq_num - TxBoWs);
    else
        janela = 1000000;
    long backlog = (long) ceil(((double) ul_queue.length()) / bmax);

#ifdef _REP_AUTO_EB
    return ((janela < backlog) ? janela : backlog);
#else
    return (((janela < backlog) ? janela : backlog) + NAK_list.length());
#endif
}



//*****************************************************************************
// Módulo meio (parte comum).
//*****************************************************************************

#include "comum.h"

void cMeio::initialize()
{
    g.num_exec++;

    for (i = 0; i < NUM_CLASSES; i++)
    {
        g.pkt_rec[i] = 0;
        g.pkt_sent[i] = 0;
        g.pay_rec[i] = 0;
        g.pay_sent[i] = 0;
        g.pay_ack[i] = 0;
        g.pay_ger[i] = 0;
        g.pay_disc[i] = 0;
    }
    g.start_time = simTime();
    msg_ant = NULL;

    bad_channel = new bool[g.num_term];
    BER_presente = new double[g.num_term];
    BER_transition = new double[g.num_term];

    // Inverso do que se quer no início.
    for (i = 0; i < g.num_term; i++)
    {
        bad_channel[i] = true;
        BER_presente[i] = g.BER_bad;
        BER_transition[i] = simTime();
    }
}

void cMeio::handleMessage(cMessage *msg)
{
    if ((g.BER_bad > 0) && (msg->hasPar("pay_length")))
    {
        long pay_length = (long) msg->par("pay_length");
        #ifdef _DEMAND
            long addr = msg->par("mac_id");
            pay_length = 384; // 48 octetos
        #else
            long addr = msg->par("src_addr");
        #endif
```

```
        if (g.BER_bad != g.BER_good)
            while (BER_transition[addr] <= simTime())
            {
                bad_channel[addr] = !bad_channel[addr];
                if (bad_channel[addr])
                {
                    BER_transition[addr] += genk_exponential(g.BER_trans_ger,
                                            g.t_bad);
                    BER_presente[addr] = g.BER_bad;
                }
                else
                {
                    BER_transition[addr] += genk_exponential(g.BER_trans_ger,
                                            g.t_good);
                    BER_presente[addr] = g.BER_good;
                }
            }
        if (BER_presente[addr] > 0)
        {
            double BER = BER_presente[addr];
            #ifdef _DEMAND
                if ((msg->hasPar("RPHY")) && (((double) msg->par("RPHY")) == MinRate))
                    BER = BER * BER_FACTOR;
            #endif
            if (genk_uniform(g.BER_ger,0,1) >= pow(1 - BER, (double) (DataOvhLength +
                                                pay_length)))
                msg->setBitError(true);
        }
    }
}



//***************************************************************************
// Módulo meio (parte específica da rede IEEE 802.11).
//***************************************************************************

#include "comum.h"

class cMeio802_11 : public cMeio
{
    Module_Class_Members(cMeio802_11, cMeio, 0);

    // DCF
    double end_time, NAV_ACK;

    // PCF
    long last_polled;

    FILE *fm;

    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    virtual void finish();
};

Define_Module_Like(cMeio802_11, cMeio);

void cMeio802_11::initialize()
{
    cMeio::initialize();

    // DCF
    end_time = 0;
    g.idle_time = 0;

    // PCF
    last_polled = 0; // Tem que ser o endereço de um terminal válido.

    fm = fopen("ocupa_meio.txt", "w");
}

void cMeio802_11::handleMessage(cMessage *msg)
{
```

```
cMeio::handleMessage(msg);

long src_addr = msg->par("src_addr");
long dest_addr = msg->par("dest_addr");
double transm_time = ((double) msg->length()) / PHYRate;
double deliver_time = simTime() + transm_time;

int msg_type = msg->kind();
switch (msg_type)
{
// DCF_DADOS - DCF, Adapt -> EB
case DCF_DADOS:
{
    long classe = msg->par("classe");

    g.pay_rec[classe] += ((long) msg->par("pay_length"));
    g.pay_sent[classe] += ((long) msg->par("pay_length"));
    if (!msg->hasBitError())
        g.pay_ack[classe] += ((long) msg->par("pay_length"));

    if (simTime() < end_time) // colisão
    {
        end_time = ((end_time > deliver_time) ? end_time : deliver_time);
        g.idle_time = end_time;

        g.pay_rec[classe] -= ((long) msg->par("pay_length"));
        if (!msg->hasBitError())
            g.pay_ack[classe] -= ((long) msg->par("pay_length"));
        delete msg;
        if (msg_ant)
        {
            g.pay_rec[classe] -= ((long) msg_ant->par("pay_length"));
            if (!msg_ant->hasBitError())
                g.pay_ack[classe] -= ((long) msg_ant->par("pay_length"));
            delete cancelEvent(msg_ant);
            msg_ant = NULL;
        }
    }
    else
    {
        NAV_ACK = SIFSTime + ((double) LOVHACK) / PHYRate;

        end_time = deliver_time;
        g.idle_time = deliver_time + NAV_ACK;

        sendDelayed(msg, transm_time, "to_unit", dest_addr);
        msg_ant = msg;
    }
}
break;
// ACK - DCF, EB -> Adapt
case ACK:
{
    sendDelayed(msg, transm_time, "to_unit", dest_addr);
}
break;
// POLL - PCF, EB -> Adapt
case POLL: // Enviado para dest_addr e last_polled (contém info de ACK)
{
    // Meio optimizado - só envia para o destinatário atual e para
    // o anterior, pois este último tem a info de ACK a receber.
    if (last_polled != dest_addr)
        sendDelayed(new cMessage(*msg), transm_time, "to_unit", last_polled);

    sendDelayed(msg, transm_time, "to_unit", dest_addr);
    last_polled = dest_addr;

    g.idle_time = deliver_time;
}
break;
case CF_END: // Envia para last_polled (contém info de ACK)
{
    sendDelayed(msg, transm_time, "to_unit", last_polled);
}
```

```
    break;
    // RESP - PCF, Adapt -> EB
    case RESP: // enviado somente para EB
    {
        long classe = msg->par("classe");
        g.pay_rec[classe] += (long) msg->par("pay_length");
        g.pay_sent[classe] += (long) msg->par("pay_length");
        if (!msg->hasBitError())
            g.pay_ack[classe] += (long) msg->par("pay_length");

        sendDelayed(msg, transm_time, "to_unit", g.num_term);

        g.idle_time = deliver_time;
    }
    break;
    default:
        error("Mensagem invalida no meio: %d", msg_type);
    }
}

void cMeio802_11::finish()
{
    fclose(fm);
}



//*****************************************************************************
// Módulo meio (parte específica da rede HIPERLAN/2).
//*****************************************************************************

#include "comum.h"

class cMeioDemand : public cMeio
{
    Module_Class_Members(cMeioDemand, cMeio, 0);

    long num_rch;
    double tlim_dados;
    double end_time;

    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

Define_Module_Like(cMeioDemand, cMeio);

void cMeioDemand::initialize()
{
    cMeio::initialize();

    end_time = 0;
}

void cMeioDemand::handleMessage(cMessage *msg)
{
    cMeio::handleMessage(msg);

    int msg_type = msg->kind();
    switch (msg_type)
    {
    case H2_DIFUSAO:
    {
        num_rch = msg->par("num_rch");
        tlim_dados = g.tfim_trama - num_rch * OneRCHTime - LCHTime + 1E-12;
        // Envia para todos os terminais, no modo de menor débito (MinRate)
        double transm_time = ((double) msg->length()) / MinRate;
        for (int i = 0; i < g.num_term; i++)
            sendDelayed(new cMessage(*msg), transm_time, "to_unit", i);
        delete msg;
    }
    break;
    case H2_RR:
    {
```

```
        double transm_time = ((double) msg->length()) / ((double) msg->par("rate"));
    #ifdef _IGNORA_RCH_COLIS
        sendDelayed(msg, transm_time, "to_unit", g.num_term); // ignora colisões
    #else
        if (simTime() >= end_time)
        {
            sendDelayed(msg, transm_time, "to_unit", g.num_term);
            msg_ant = msg;
        }
        else // COLISÃO!
        {
            // cout << "COLISAO!" << endl;
            delete msg;
            if (msg_ant)
            {
                delete cancelEvent(msg_ant);
                msg_ant = NULL;
            }
        }
        end_time = simTime() + transm_time;
    #endif
    }
    break;
    case H2_DADOS:
    {
        long classe = msg->par("classe");
        if (simTime() < end_time)
        {
            cout << simTime() << "\t" << end_time << endl;
            error("Colisao de pacotes de dados!"); // Não deve acontecer
        }
        if (simTime() > tlim_dados)
        {
            cout << simTime() << "\t" << tlim_dados << endl;
            error("Dados enviados durante o período de RCH");
        }

        if (((long) msg->par("seq_num")) > 0) // Não é dummy
        {
            long ul_pay_length = msg->par("pay_length");
            g.pay_rec[classe] += ul_pay_length;
            g.pay_sent[classe] += ul_pay_length;
            if (!msg->hasBitError())
                g.pay_ack[classe] += ul_pay_length;
        }

        double TLCH = ((double) LCHLength) / ((double) msg->par("RPHY"));
        sendDelayed(msg, TLCH, "to_unit", (long) msg->par("dest_addr"));
        // 1E-12 é uma margem de segurança aquando da comparação acima
        end_time = simTime() + TLCH - 1E-12;
    }
    break;
    case H2_ARQ:
    {
        sendDelayed(msg, SCHTime, "to_unit", (long) msg->par("mac_id"));
    }
    break;
    default:
        error("Mensagem invalida no meio: %d", msg_type);
    }
}



//***************************************************************************
// Módulo ponto de acesso/estação base (parte comum).
//***************************************************************************

#include "comum.h"

// Com esses valores (800, 0.1) cell width é 0.125 ms
#define NUM_CELLS       800 //1000
#define UPPER_RANGE     0.1
```

```
double dmean_ant;
double dmean_ant_ant;
double dmean_ini;

/*  // Confidence Level 90%
    double t_quantile[] = {0, 6.314, 2.920, 2.353, 2.132, 2.015, 1.943, 1.895, 1.860,\
                              1.833, 1.812, 1.796, 1.782, 1.771, 1.761, 1.753, 1.746,\
                              1.740, 1.734, 1.729, 1.725, 1.721, 1.717, 1.714, 1.711,\
                              1.708, 1.706, 1.703, 1.701, 1.699, 1.697};
*/
    // Confidence Level 95%
    double t_quantile[] = {0,12.706, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306,\
                              2.262, 2.228, 2.201, 2.179, 2.160, 2.145, 2.131, 2.120,\
                              2.110, 2.101, 2.093, 2.086, 2.080, 2.074, 2.069, 2.064,\
                              2.060, 2.056, 2.052, 2.048, 2.045, 2.042};
    const long num_quantile = sizeof(t_quantile) / sizeof(double);

#define PREC_MIN0.1
#define MAX_OBSERV  num_quantile

void cEB::initialize()
{
    reg_term = 0; // (primeiro), ou g.num_term - 1 (último);
//  reg_term2 = 9; // Décimo
    reg_term2 = g.num_term1 - 1;

    if (g.wind_ini > 0)
        passou_inicio = false;
    else
        passou_inicio = true;

    display_check = g.display_update + 1E-12;
    collect_check = g.wind_collect;
    num_observ = 0;

    #ifdef _TRANSIENTE
        fbt = fopen("bt.txt", "w");
        if (!fbt)
            error("Abertura de bt.txt");
        fprintf(fbt,"%s%14s%8s%15s\n" , "%", "t_recebido", "b", "b_medio");
        fdt = fopen("dt.txt", "w");
        if (!fdt)
            error("Abertura de dt.txt");
        fprintf(fdt, "%s%14s%15s%15s%15s\n", "%", "t_gerado", "t_recebido", "atraso",
                                                    "atraso medio");
    #endif

    // aloca memoria para g.num_term objectos do tipo cStdDev
    // (e cria os objectos vazios)
    dterm = new cStdDev[g.num_term];
    bterm = new cStdDev[g.num_term];

    for (i = 0; i < NUM_CLASSES; i++)
    {
        dDist_classe[i] = new cDoubleHistogram("Delay Distribution", NUM_CELLS);
        dDist_classe[i]->setRange(0.0, UPPER_RANGE);
    }
    dDist_term = new cDoubleHistogram("Delay Distribution", NUM_CELLS);
    dDist_term->setRange(0.0, UPPER_RANGE);
    dDist_term2 = new cDoubleHistogram("Delay Distribution", NUM_CELLS);
    dDist_term2->setRange(0.0, UPPER_RANGE);

    printf("%11s%12s%11s%11s%11s", "simTime()", "b_medio", "d_medio", "d_stddev",
                                        "amostras");
    printf("%10s%10s%10s%10s%10s%10s\n", "pay_rec", "pay_ack", "pay_colis", "pay_ger",
                                        "pay_disc", "horario");

    dmean_ant = -1;
    dmean_ant_ant = -2;

    dsp_trans = &dterm[0];
    dsp_collect = &dterm[0];
    bsp_collect = &bterm[0];
}
```

```
void cEB::finish()
{
    time_t ltime = time(NULL);
    strcpy(time_s, ctime(&ltime));
    time_s[24] = 0; // Tira o "\"

    char rede_mac[30];
    strcpy(rede_mac, REDE);
    if (strlen(MAC) > 0)
    {
        strcat(rede_mac, "_");
        strcat(rede_mac, MAC);
    }
    strcat(rede_mac, "_");

    bool file_exists;
    FILE *f;

    long samples_g = 0;
    double sum_g = 0;
    double sqrSum_g = 0;
    printf("%5s%7s%11s%11s%11s%11s\n", "Term", "Classe", "Media", "Std_Dev", "Maximo",
                    "Samples");
    for (i = 0; i < g.num_term1; i++)
        printf("%5d%7d%11.6f%11.6f%11.6f%11d\n", i, 0, dterm[i].mean(),
                    dterm[i].stddev(), dterm[i].max(), dterm[i].samples());
    for (i = g.num_term1; i < g.num_term; i++)
        printf("%5d%7d%11.6f%11.6f%11.6f%11d\n", i, 1, dterm[i].mean(),
                    dterm[i].stddev(), dterm[i].max(), dterm[i].samples());
    for (i = 0; i < NUM_CLASSES; i++)
        printf("%5s%7d%11.6f%11.6f%11.6f%11d\n", "Todos", i, dclasse[i].mean(),
                    dclasse[i].stddev(), dclasse[i].max(), dclasse[i].samples());

    f = fopen(catStr(rede_mac, "conexoes.txt"), "a");
    fprintf(f, "%s %20s\n", "%", &time_s[4]);
    fprintf(f, "%5s%7s%11s%11s%11s%11s\n", "Term", "Classe", "Media", "Std_Dev",
                    "Maximo", "Samples");
    for (i = 0; i < g.num_term1; i++)
        fprintf(f, "%5d%7d%11.2f%11.2f%11.2f%11d\n", i, 0, dterm[i].mean()*1E3,
                    dterm[i].stddev()*1E3, dterm[i].max()*1E3, dterm[i].samples());
    for (i = g.num_term1; i < g.num_term; i++)
        fprintf(f, "%5d%7d%11.2f%11.2f%11.2f%11d\n", i, 1, dterm[i].mean()*1E3,
                    dterm[i].stddev()*1E3, dterm[i].max()*1E3, dterm[i].samples());
    for (i = 0; i < NUM_CLASSES; i++)
        fprintf(f, "%5s%7d%11.2f%11.2f%11.2f%11d\n", "Todos", i,
                    dclasse[i].mean()*1E3, dclasse[i].stddev()*1E3,
                    dclasse[i].max()*1E3, dclasse[i].samples());
    fprintf(f, "\n");
    fclose(f);

/*
    // Cálculo das estatísticas globais a partir das estatísticas das partes.
    for (i = 0; i < g.num_term; i++)
    {
        samples_g += dterm[i].samples();
        sum_g += dterm[i].sum();
        sqrSum_g += dterm[i].sqrSum();
    }
    cout << "Media " << reg_term << " = " << dterm[reg_term].mean() << endl;
    cout << "Media global = " << sum_g / samples_g << endl;
    cout << "Desvio global = ";
    cout << sqrt((sqrSum_g - (sum_g * sum_g) / samples_g) / (samples_g - 1)) << endl;
*/

    #ifdef _TRANSIENTE
        fprintf(fbt, "%s %20s\n", "%", &time_s[4]);
        fclose(fbt);
        fprintf(fdt, "%s %20s\n", "%", &time_s[4]);
        fclose(fdt);
    #endif

    f = abre_testa(file_exists, catStr(rede_mac, "parametros.txt"));
```

```
    if (!file_exists)
        cGlobal::insereTituloComent(f);
    fprintf(f, "%s %20s\n", "%", &time_s[4]);
    cGlobal::insereDadosComent(f);
    fclose(f);

    f = abre_testa(file_exists, catStr(rede_mac, "eficiencia.txt"));
    if (!file_exists)
    {
        fprintf(f, "%s%12s", "%", "tlivre (%)");
#ifdef _DEMAND
        fprintf(f, "%12s%12s%12s%12s%12s%12s", "ie_min", "ie_med", "ie_max",
                    "rch_min", "rch_med", "rch_max");
#endif
        fprintf(f, "\n");
    }
    fprintf(f, "%s %20s\n", "%", &time_s[4]);
    fprintf(f, "%13.6f", (tidle / (simTime() - g.start_time)));
#ifdef _DEMAND
    fprintf(f, "%12d", (int) nies.min());
    fprintf(f, "%12.6f", nies.mean());
    fprintf(f, "%12d", (int) nies.max());
    fprintf(f, "%12d", (int) nrchs.min());
    fprintf(f, "%12.6f", nrchs.mean());
    fprintf(f, "%12d", (int) nrchs.max());
#endif
    fprintf(f, "\n");
    fclose(f);

    regAtrasos(catStr(rede_mac, "atrasos_term.txt"), dterm[reg_term],
                bterm[reg_term]);
    regAtrasos(catStr(rede_mac, "atrasos_term2.txt"), dterm[reg_term2],
                bterm[reg_term2]);
    regAtrasos(catStr(rede_mac, "atrasos_classe_0.txt"), dclasse[0], bclasse[0]);
    regAtrasos(catStr(rede_mac, "atrasos_classe_1.txt"), dclasse[1], bclasse[1]);
#ifdef _REGISTA_RR_DELAY
    regAtrasos(catStr(rede_mac, "atrasos_RR.txt"), *g.drstall, *g.astall);
#endif

    regDistAtrasos(catStr(rede_mac, "dist_atrasos_term.txt"), dDist_term);
    regDistAtrasos(catStr(rede_mac, "dist_atrasos_term2.txt"), dDist_term2);
    regDistAtrasos(catStr(rede_mac, "dist_atrasos_classe_0.txt"), dDist_classe[0]);
    regDistAtrasos(catStr(rede_mac, "dist_atrasos_classe_1.txt"), dDist_classe[1]);

    regPacotes(catStr(rede_mac, "pacotes_classe_0.txt"), 0);
    regPacotes(catStr(rede_mac, "pacotes_classe_1.txt"), 1);

    FILE *fstat;
    fstat = fopen("_media.txt", "w");
    fprintf(f, "%f", dsp_collect->mean());
    fclose(fstat);

    delete [] dterm;
    delete [] bterm;

    for (i = 0; i < NUM_CLASSES; i++)
    {
        delete dDist_classe[i];
    }
    delete dDist_term;
    delete dDist_term2;
}

void cEB::checkPoint()
{
    double display_compara;
    if (g.display_update_time)
        display_compara = simTime();
    else
        display_compara = dsp_collect->samples();

    if (display_compara > display_check)
    {
```

```
    printf("%11.6f%12.6f%11.6f%11.6f%11d", simTime(), bsp_collect->mean(),
            dsp_collect->mean(), dsp_collect->stddev(), dsp_collect->samples());
    printf("%10d%10d%10d%10d%10d", g.pay_rec[0], g.pay_ack[0], (g.pay_sent[0] -
            g.pay_rec[0]), g.pay_ger[0], g.pay_disc[0]);
    time_t ltime = time(NULL);
    char time_s[30];
    strcpy(time_s, ctime(&ltime));
    time_s[19] = 0;
    printf("%10s\n", &time_s[11]);
    display_check += g.display_update;
}


if (!passou_inicio)
{
    if (g.transient_criteria_time)
    {
        if (simTime() > g.wind_ini)
        {
            clearResults();
            passou_inicio = true;
            dmean_ini = dsp_trans->mean();
            cout << "Media inicial = " << dmean_ini << endl;
        }
    }
    else
    {
        if (dsp_trans->samples() > g.wind_ini)
        {
            printf("%11.6f%12s%11.6f%11.6f%11d\n", simTime(), "", dsp_trans-
                    >mean(), dsp_trans->stddev(), dsp_trans->samples());
            if ((dmean_ant < dmean_ant_ant) && (dsp_trans->mean() <
                    dmean_ant_ant))
            {
                passou_inicio = true;
                cout << "***** Fim do periodo transitorio *****" << endl;
                dmean_ini = dsp_collect->mean();
                cout << "Media inicial = " << dmean_ini << endl;
            }
            dmean_ant_ant = dmean_ant;
            dmean_ant = dsp_trans->mean();
            clearResults();
        }
    }
}
else
{
    double stop_compara;
    if (g.stop_criteria_time)
        stop_compara = simTime() - g.start_time;
    else
        stop_compara = dsp_collect->samples();

    if (stop_compara >= collect_check)
    {
        if (num_observ == 0)
        {
            observ[num_observ] = dsp_collect->mean();
            amostras = dsp_collect->samples();
        }
        else
        {
            amostras = dsp_collect->samples() - samples_ant;
            observ[num_observ] = dsp_collect->mean() * dsp_collect->samples() -
                                    mean_ant * samples_ant;
            observ[num_observ] /= amostras;
        }
        stat_observ.collect(observ[num_observ]);
        cout << observ[num_observ] << "\t" << amostras << "\t" << (num_observ + 1)
                << endl;
        cout << dsp_collect->mean() << "\t" << dsp_collect->samples() << endl;
        cout << stat_observ.mean() << "\t" << stat_observ.samples() << endl;
        num_observ++;
        collect_check += g.wind_collect;
```

```
            mean_ant = dsp_collect->mean();
            samples_ant = dsp_collect->samples();
            if (num_observ >= 3)
            {
                double autocovariance = 0;
                for (i = 0; i < (num_observ - 1); i++)
                    autocovariance += (observ[i] - stat_observ.mean()) * (observ[i +
                                      1] - stat_observ.mean());
                autocovariance /= (num_observ - 2);
                cout << "Autocovariance = " << autocovariance << endl;
                cout << "Variance = " << stat_observ.variance() << endl;
                cout << "Razao = " << (autocovariance / stat_observ.variance()) <<
                            endl;
            }
            if (num_observ >= 2)
            {
                double precisao = t_quantile[num_observ - 1] * stat_observ.stddev();
                precisao /= (stat_observ.mean() * sqrt(num_observ));
                printf("\nPrecisao parcial = %5.2f%s, com %d observacoes.\n", 100.0 *
                            precisao, "%", num_observ);
                if (precisao < PREC_MIN)
                    endSimulation();
            }
            if (num_observ == MAX_OBSERV)
            {
                cout << "Numero maximo de observacoes atingido. Terminando simulacao"
                            << endl;
                endSimulation();
            }
        }
    }
}

void cEB::clearResults()
{
    for (i = 0; i < NUM_CLASSES; i++)
    {
        g.pkt_rec[i] = 0;
        g.pkt_sent[i] = 0;
        g.pay_rec[i] = 0;
        g.pay_sent[i] = 0;
        g.pay_ack[i] = 0;
        g.pay_ger[i] = 0;
        g.pay_disc[i] = 0;
    }
    g.start_time = simTime();

    for (int i = 0; i < g.num_term; i++)
    {
        dterm[i].clearResult();
        bterm[i].clearResult();
    }

    if (!g.display_update_time)
        display_check -= dsp_collect->samples();

    for (i = 0; i < NUM_CLASSES; i++)
    {
        dclasse[i].clearResult();
        bclasse[i].clearResult();
        delete dDist_classe[i];
        dDist_classe[i] = new cDoubleHistogram("Delay Distribution", NUM_CELLS);
        dDist_classe[i]->setRange(0.0, UPPER_RANGE);
    }
    delete dDist_term;
    delete dDist_term2;
    dDist_term = new cDoubleHistogram("Delay Distribution", NUM_CELLS);
    dDist_term->setRange(0.0, UPPER_RANGE);
    dDist_term2 = new cDoubleHistogram("Delay Distribution", NUM_CELLS);
    dDist_term2->setRange(0.0, UPPER_RANGE);

    #ifdef _REGISTA_RR_DELAY
        g.drstall->clearResult();
        g.astall->clearResult();
```

```
    #endif
    nies.clearResult();
    nrchs.clearResult();
    tidle = 0;
}

void cEB::despachaPacote(cMessage *msg)
{
#ifdef _DEMAND
    src_addr = msg->par("mac_id");
#else
    src_addr = msg->par("src_addr");
#endif
    cQueue *pay_queue = (cQueue *) (cObject *) msg->par("pay_queue");
    while (!pay_queue->empty())
    {
        cPktInfo *ul = (cPktInfo *) pay_queue->pop();

        if (ul->frag_num == 0)
        {
            double d = simTime() - ul->timestamp();
            dclasse[ul->classe].collect(d);
            dDist_classe[ul->classe]->collect(d);
            dterm[src_addr].collect(d);
            if (src_addr == reg_term)
            {
                dDist_term->collect(d);
                #ifdef _TRANSIENTE
                    fprintf(fdt, "%15.6f%15.9f%15.9f%15.9f\n", ul->timestamp(),
                                simTime(), d, dterm[src_addr].mean());
                #endif
            }
            else if (src_addr == reg_term2)
                dDist_term2->collect(d);

            #ifdef _REGISTA_RR_DELAY
                if (ul->trr > 0)
                    g.drstall->collect((double) ul->trr - ul->timestamp());
                else
                    g.drstall->collect(0);
            #endif
        }

        delete ul;
    }
    delete msg;
}

void cEB::regAtrasos(char *fname, cStdDev dstats, cStdDev bstats)
{
    bool file_exists;
    FILE *f;
    f = abre_testa(file_exists, fname);
    if (!file_exists)
    {
        fprintf(f, "%s%12s%10s%10s%12s%10s%12s", "%", "tsim", "n_b", "b_min", "b_med",
                   "b_max", "b_dev");
        fprintf(f, "%10s%12s%12s%12s%12s", "n_d", "d_min", "d_med", "d_max", "d_dev");
        fprintf(f, "\n");
    }
    fprintf(f, "%s %20s\n", "%", &time_s[4]);
    fprintf(f, "%13.6f", simTime() - g.start_time);
    fprintf(f, "%10d", (int) bstats.samples());
    fprintf(f, "%10d", (int) bstats.min());
    fprintf(f, "%12.6f", bstats.mean());
    fprintf(f, "%10d", (int) bstats.max());
    fprintf(f, "%12.6f", bstats.stddev());
    fprintf(f, "%10d", (int) dstats.samples());
    fprintf(f, "%12.6f", dstats.min());
    fprintf(f, "%12.6f", dstats.mean());
    fprintf(f, "%12.6f", dstats.max());
    fprintf(f, "%12.6f", dstats.stddev());
    fprintf(f, "\n");
    fclose(f);
```

```
}

void cEB::regDistAtrasos(char *fname, cDoubleHistogram *dDist)
{
    char str_aux[100];
    int j;
    bool file_exists;
    FILE *f;
    f = abre_testa(file_exists, fname);
    if (!file_exists)
    {
        fprintf(f, "%13s%13s%10s", "%overflows", "cell_width", "samples");
        for (j = 0; j < dDist->cells(); j++)
        {
            sprintf(str_aux, "cel_%d", j);
            fprintf(f, "%11s", str_aux);
        }
        fprintf(f, "\n");
    }
    fprintf(f, "%s %20s\n", "%", &time_s[4]);
    fprintf(f, "%13d", dDist->overflowCell());
    fprintf(f, "%13.6f", UPPER_RANGE/NUM_CELLS);
    fprintf(f, "%10d", (long) dDist->samples());
    if (dDist->cells() == 0)
        dDist->collect(0);
    for (j = 0; j < dDist->cells(); j++)
        fprintf(f, "%11d", (long) dDist->cell(j));
    fprintf(f, "\n");
    fclose(f);
}

void cEB::regPacotes(char *fname, long classe)
{
    bool file_exists;
    FILE *f;
    f = abre_testa(file_exists, fname);
    if (!file_exists)
    {
        fprintf(f, "%16s", "%t_sim");
        fprintf(f, "%12s%12s%12s%12s%12s", "pay_rec", "pay_sent", "pay_ack",
                   "pay_ger", "pay_disc");
        fprintf(f, "\n");
    }
    fprintf(f, "%s %20s\n", "%", &time_s[4]);
    fprintf(f, "%16.9f", simTime() - g.start_time);
    fprintf(f, "%12d", g.pay_rec[classe]);
    fprintf(f, "%12d", g.pay_sent[classe]);
    fprintf(f, "%12d", g.pay_ack[classe]);
    fprintf(f, "%12d", g.pay_ger[classe]);
    fprintf(f, "%12d", g.pay_disc[classe]);
    fprintf(f, "\n");
    fclose(f);
}

FILE * cEB::abre_testa(bool &file_exists, char *fname)
{
    FILE *f;
    f = fopen(fname, "r");
    if (f)
    {
        file_exists = true;
        fclose(f);
    }
    else
        file_exists = false;
    f = fopen(fname, "a");
    if (!f)
        error("Abertura de %s", fname);
    return f;
}
```

```
//*****************************************************************************
// Módulo ponto de acesso (parte específica da rede IEEE 802.11).
//*****************************************************************************

#include "comum.h"

class cEB802_11 : public cEB
{
    Module_Class_Members(cEB802_11, cEB, 0);

    // DCF
    cMessage *send_ack;

    // PCF
    cMessage *send_pkt, *ini_cfp;
    long term_polled, cont_poll;
    double tfim_PCF, tnext_CFP;
    cLinkedList rep_list;
    bool envia_rep;

    bool sack, rack; // rack não é usado (EB não recebe ACK)

    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    virtual void finish();

    virtual void sendPoll(long term_polled);
};

Define_Module_Like(cEB802_11, cEB);

void cEB802_11::initialize()
{
    cEB::initialize();

    rack = true;

    // DCF
    send_ack = new cMessage(NULL, SEND_ACK);

    // PCF
    term_polled = 0;
    send_pkt = new cMessage(NULL, SEND_PKT);
    ini_cfp = new cMessage(NULL, INI_CFP);

#ifdef _PCF_DCF
    cout << "TSuperframe = " << SFPeriod << "\t" << "TCPmin = " << TCPmin<< endl;
    scheduleAt(simTime(), ini_cfp);  // inicia o primeiro CFP
#endif
}

void cEB802_11::handleMessage(cMessage *msg)
{
    int msg_type = msg->kind();
    switch (msg_type)
    {
    case DCF_DADOS: // DCF
    {
        ev << "DCF_DADOS" << endl;
        cQueue *pay_queue = (cQueue *) (cObject *) msg->par("pay_queue");
        src_addr = msg->par("src_addr");
        int b = pay_queue->length();
        bclasse[(long) msg->par("classe")].collect(b);
        bterm[src_addr].collect(b);
        #ifdef _TRANSIENTE
        if (src_addr == reg_term)
            fprintf(fbt,"%15.9f%8d%15.9f\n" , simTime(), b, bterm[src_addr].mean());
        #endif
        if (!msg->hasBitError())
        {
            sack = true;
            despachaPacote(msg);
        }
        else
```

```
    {
        sack = false;
        delete(msg);
    }
    scheduleAt(simTime() + SIFSTime, send_ack);
}
break;
case SEND_ACK:  // DCF
{
    ev << "SEND_ACK" << endl;
#ifndef _NEGATIVE_ACK
    if (sack)
    {
#endif
    cMessage *ack_pkt;
    ack_pkt = new cMessage(NULL, ACK, LOVHACK);
    if (sack)
        ack_pkt->setName("ACK");
    else
        ack_pkt->setName("NAK");
    ack_pkt->addPar("src_addr") = g.num_term;
    ack_pkt->addPar("dest_addr") = src_addr;
    ack_pkt->addPar("ack") = sack;
    send(ack_pkt, "to_meio");
#ifndef _NEGATIVE_ACK
    }
#endif
}
break;
case RESP: // PCF
{
    if (((long) msg->par("pay_length")) == 0) // Mensagem vazia
    {
        sack = true; // Ignora erros e não processa payload
        cout << "Recebeu DUMMY!" << endl;
    }
    else
    {
        cQueue *pay_queue = (cQueue *) (cObject *) msg->par("pay_queue");
        src_addr = msg->par("src_addr");
        int b = pay_queue->length();
        bclasse[(long) msg->par("classe")].collect(b);
        bterm[src_addr].collect(b);
        #ifdef _TRANSIENTE
        if (src_addr == reg_term)
            fprintf(fbt,"%15.9f%8d%15.9f\n" , simTime(), b,
                            bterm[src_addr].mean());
        #endif
        if (!msg->hasBitError())
        {
            sack = true;
            despachaPacote(msg);
        }
        else
        {
            sack = false;
            delete(msg);
            #ifdef _REPETE_NAKS_FIM
                rep_list.insert(new long(src_addr));
            #endif
        }
    }
    scheduleAt(simTime() + SIFSTime, send_pkt);
}
break;
case SEND_PKT: // PCF
{
    bool envia_cf_end = false;
    if (!envia_rep)
        if (cont_poll == 0)         // Varreu toda a lista
            envia_rep = true;       // Envia repetidos

    if (envia_rep)
        if (rep_list.empty())       // Mas, se não há repetidos ...
```

```
                envia_cf_end = true;// envia CF-END

        if ((!envia_cf_end) && (simTime() > tfim_PCF))
        // Aborta polling (entrou no CP)
        {
            if (!envia_rep)
                cout << "Abortando primeiro POLL a partir do terminal " << term_polled
                        << endl;
            else
            {
                cout << "Abortando POLL repetido. Faltam " << rep_list.length() << "
                            terminais: ";
                for (cLinkedListIterator iter(rep_list, 0); !iter.end(); iter--)
                    cout << *((long *) iter()) << " ";
                cout << endl;
            }
            rep_list.clear();
            envia_cf_end = true;
        }

        if (envia_cf_end)
        {
            cMessage *cf_end = new cMessage(NULL, CF_END, LOVHCFEND);
            cf_end->addPar("ack") = sack;
            cf_end->addPar("src_addr") = g.num_term;
            cf_end->addPar("dest_addr") = -1; // Não interessa o valor (broadcast)
            send(cf_end, "to_meio");
            tidle += (tnext_CFP - simTime() - ((double) LOVHCFEND / PHYRate));
        }
        else if (!envia_rep) // envia 1º POLL ao terminal nesta supertrama
        {
            sendPoll(term_polled);
            term_polled = ++term_polled % g.num_term1;
            cont_poll--;
        }
        else // envia POLL repetido ao terminal a partir da rep_list
        {
            sendPoll(*((long *) rep_list.tail()));
            delete rep_list.pop();
        }
    }
    break;
    case INI_CFP: // PCF
    {
        // Começa o varrimento da lista de polling
        #ifndef _ROLA_LISTA_POLLING
            term_polled = 0;
        #endif
        cont_poll = g.num_term1; // Número de terminais na lista de polling
        envia_rep = false;
        double tlivre = ((g.idle_time > simTime()) ? g.idle_time : simTime());
        scheduleAt(tlivre + PIFSTime, send_pkt);
        g.idle_time = tlivre + PIFSTime; // Evita transmissão DCF
        tnext_CFP = simTime() + SFPeriod;
        tfim_PCF = tnext_CFP - TCPmin - (2 * SIFSTime + ((double) (2 * TotalOvhLength
                    + LOVHCFEND)) / PHYRate);
        scheduleAt(tnext_CFP, ini_cfp);  // inicia o CFP seguinte
    }
    break;
    case DL: // Não implementado (tráfego descendente)
    {
    }
    break;
    default:
        error("Mensagem invalida na EB: %d", msg_type);
    }

    checkPoint();
}

void cEB802_11::finish()
{
    cout << "Parcela livre dos recursos (para o CP) = " << tidle / (simTime() -
                g.start_time) << endl;
```

```
    cEB::finish();
}

void cEB802_11::sendPoll(long term_polled)
{
            char pname[20];
            sprintf(pname, "POLL(%d)", term_polled);
            cMessage *poll = new cMessage(pname);
            poll->setKind(POLL);
            poll->setLength(TotalOvhLength);
            poll->addPar("src_addr") = g.num_term;
            poll->addPar("dest_addr") = term_polled;
            poll->addPar("ack") = sack;

            send(poll, "to_meio");
}



//*****************************************************************************
// Módulo ponto de acesso (parte específica da rede HIPERLAN/2).
//*****************************************************************************

#include "comum.h"

class cEBDemand : public cEB
{
    Module_Class_Members(cEBDemand, cEB, 0);

    long num_ie, ie_index, num_DL_RG;
    double tindep, tdep_ie, tdep_rg;
    cIE l_ie_array[45];
    cQueue rr_queue[NUM_CLASSES], fca_queue, poll_queue;
    cQueue ARQ_pkt_queue[NUM_CLASSES], rep_queue[NUM_CLASSES];
    cMessage *send_h2_broadcast, *clear_ach;
    long num_rch, poll_index, num_poll;
    char str_aux[100];
    long *TxBoWs;
    long *last_seq_num, *inc_RR;
    cLinkedList *NAK_list;
    bool send_ACK;
    cQueue *ul_pkt_queue;
    double *FCPointer;
    bool *ajustou_atraso;
    long num_lch_fca;

    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    virtual void finish();

    virtual void armazenaRR(cQueue &rr_queue, cMessage *msg);
    virtual bool contemTerm(cQueue &queue, long addr);

    virtual long num_ieb_f(long num_ie);
    virtual long inc_num_ieb(long num_ie);
    virtual long numChAv(long mac_id, long tipo_RG, long tipo_ch, double RLCH);
    virtual bool escalonaLCHs(cQueue &aqueue, long num_lch_max, bool move_fim, double
                              RLCH);
    virtual void escalonaARQs(cQueue &ARQ_pkt_queue, cQueue &rep_queue);
};

Define_Module_Like(cEBDemand, cEB);

void cEBDemand::initialize()
{
    cEB::initialize();

    #ifdef _C0_SEM_RCH
        g.usa_rch[0] = false;
    #else
        g.usa_rch[0] = true;
    #endif
    g.usa_rch[1] = true;
```

```
    // ul_pkt_queue - armazena as mensagens que não podem ser
    // despachadas por faltarem mensagens anteriores.
    ul_pkt_queue = new cQueue[g.num_term];

    NAK_list = new cLinkedList[g.num_term];

    last_seq_num = new long[g.num_term];
    inc_RR = new long[g.num_term];
    TxBoWs = new long[g.num_term];
    for (i = 0; i < g.num_term; i++)
    {
        last_seq_num[i] = -1;
        inc_RR[i] = 0;
        TxBoWs[i] = 0;
    }

    ajustou_atraso = new bool[g.num_term1];
    FCPointer = new double[g.num_term1];
    for (i = 0; i < g.num_term1; i++)
    {
        ajustou_atraso[i] = false;
        // Inicializa FCPointer das conexões com o objectivo
        // de distribuir a carga pelas tramas:
        // Escopo: Tramas abrangidas por g.FCInterval
        //FCPointer[i] = FrameTime * floor(((g.FCInterval + 1E-12) * i) / (g.num_term1
                            * FrameTime));
        // Escopo: Tramas inteiramente abrangidas por g.FCInterval (jitter menor)
        FCPointer[i] = FrameTime * floor(((FrameTime * floor((g.FCInterval + 1E-12) /
                            FrameTime) + 1E-12) * i) / (g.num_term1 * FrameTime));
    }

    num_lch_fca = (long) ceil(g.FCInterval / (g.ilto + 1E-6));
    cout << "num_lch_fca = " << num_lch_fca << "\tIntervalo FCA/POLL_ANTES = " <<
                g.FCInterval << endl;

    poll_index = 0;
    g.astall = new cStdDev;
    g.drstall = new cStdDev;
    num_poll = 0;

    send_h2_broadcast = new cMessage(NULL, SEND_H2_DIFUSAO);
    clear_ach = new cMessage(NULL, CLEAR_ACH);
    scheduleAt(simTime() + BCHPreTime, send_h2_broadcast);  // primeira trama MAC
    cout << "num_rch = " << H2_NUM_RCH_INI << "\t" << "num_poll = " << num_poll <<
                endl;
}

void cEBDemand::handleMessage(cMessage *msg)
{
    int msg_type = msg->kind();
    switch (msg_type)
    {
    case SEND_H2_DIFUSAO:
    {
        ev << "SEND_H2_DIFUSAO" << endl;

        #ifdef _SEM_CONTENDA
            num_rch = (g.num_term > H2_NUM_RCH_INI) ? H2_NUM_RCH_INI : g.num_term;
        #else
            num_rch = H2_NUM_RCH_INI;
        #endif

        // Apaga o escalonamento de recursos da trama anterior
        for (i = 0; i < 45; i++)
        {
            l_ie_array[i].num_lch_g = 0;
            l_ie_array[i].num_sch_g = 0;
        }

        g.tfim_trama = simTime() - BCHPreTime + FrameTime;

        tindep = BCHPreTime + BCHTime + ACHTime + TurnTime + num_rch * OneRCHTime;
        num_ie = 0;
```

```
// FCH ocupa 1 IEB mesmo sem IEs (num_ie = 0)
tdep_ie = IEBlockTime;
tdep_rg = 0;
num_DL_RG = 0;

#ifdef _FCA
    cMessage *fca;
    for (i = 0; i < g.num_term1; i++)
        if (simTime() > FCPointer[i])
        {
            fca = new cMessage;
            fca->addPar("mac_id") = i;
            fca->addPar("num_lch_r") = num_lch_fca; //Era 1
            fca_queue.insert(fca);
            FCPointer[i] += g.FCInterval;
        }

    if (!escalonaLCHs(fca_queue, -1, false, LCHRate))
        error("Nao ha recursos suficientes para o FCA");
#endif // _FCA

#ifdef _POLL_ANTES
    cMessage *poll;
    for (i = 0; i < g.num_term1; i++)
        if (simTime() > FCPointer[i])
        {
            poll = new cMessage;
            poll->addPar("mac_id") = i;
            poll->addPar("num_sch_r") = 1;
            poll_queue.insert(poll);
            FCPointer[i] += g.FCInterval;
        }
    while ((!poll_queue.empty()) && ((num_ie + num_poll) < 45))
    {
        cMessage *poll = (cMessage *) poll_queue.tail();

        if (numChAv(poll->par("mac_id"), UL_RG, SCH, 0) == 0)
            break;

        l_ie_array[ie_index].num_sch_g += 1;
        tdep_rg += SCHTime;
        delete poll_queue.pop();
    }
    if (!poll_queue.empty())
        error("Nao ha recursos suficientes para o POLL_ANTES");
#endif // _POLL_ANTES

escalonaARQs(ARQ_pkt_queue[0], rep_queue[0]);
#ifdef _C0_REPETE_RMIN
    escalonaLCHs(rep_queue[0], -1, false, MinRate);
#else
    escalonaLCHs(rep_queue[0], -1, false, LCHRate);
#endif
escalonaLCHs(rr_queue[0], H2_ESC_LCH_MAX, true, LCHRate);

escalonaARQs(ARQ_pkt_queue[1], rep_queue[1]);
escalonaLCHs(rep_queue[1], -1, false, LCHRate);
escalonaLCHs(rr_queue[1], H2_ESC_LCH_MAX, true, LCHRate);

// Transfere os elementos l_ie_array (não ordenados) para
// g.ie_array (ordenado primeiro DL_RG, depois UL_RG).
long j = 0;
for (i = 0; i < num_ie; i++)
    if (l_ie_array[i].tipo == DL_RG)
        g.ie_array[j++] = l_ie_array[i];
for (i = 0; i < num_ie; i++)
    if (l_ie_array[i].tipo == UL_RG)
        g.ie_array[j++] = l_ie_array[i];
if (j != num_ie)
    error("Algoritmo errado");

// Definição o valor do start pointer de cada
// elememento de informação na trama.
```

```
double start_pointer = simTime() + BCHTime + num_ieb_f(num_ie) * IEBlockTime +
                       ACHTime;

for (i = 0; i < num_DL_RG; i++)
{
    start_pointer += DLPreTime;
    long classe = ((g.ie_array[i].mac_id < g.num_term1) ? 0 : 1);
    cMessage *arq = (cMessage *) ARQ_pkt_queue[classe].pop();
    // Estes não são usados
    // g.ie_array[i].mac_id = arq->par("mac_id");
    // g.ie_array[i].tipo = DL_RG;
    // g.ie_array[i].num_lch_g = 0; // Já é zero originalmente
    // g.ie_array[i].num_sch_g = 1;
    g.ie_array[i].start_pointer = start_pointer;
    sendDelayed(arq, start_pointer - simTime(), "to_meio");
    start_pointer += SCHTime;

    long mac_id = arq->par("mac_id");
    if (NAK_list[mac_id].empty())
        arq->addPar("ACK") = last_seq_num[mac_id] + 1;
    else
    {
        arq->addPar("ACK") = *((long *) NAK_list[mac_id].tail());

        cLinkedList *copia = new cLinkedList;
        for (cLinkedListIterator iter(NAK_list[mac_id], 0); !iter.end();
                iter--)
        {
            long *longo = (long *) iter();
            copia->insert(new long(*longo));
        }
        cPar *par_aux = new cPar("NAK_list");
        par_aux->takeOwnership(true);
        par_aux->setObjectValue(copia);
        arq->addPar(par_aux);
    }
    arq->addPar("CAI") = 1;
    TxBoWs[(long) arq->par("mac_id")] = arq->par("ACK");
}

start_pointer += TurnTime;
for (i = num_DL_RG; i < num_ie; i++)
{
    start_pointer += ULGuardTime + ULPreTime;
    g.ie_array[i].start_pointer = start_pointer;
    ev << "index = " << i << endl;
    ev << "num_lch_g = " << g.ie_array[i].num_lch_g << endl;
    ev << "num_sch_g = " << g.ie_array[i].num_sch_g << endl;
    ev << "start_pointer = " << g.ie_array[i].start_pointer << endl;
    start_pointer += g.ie_array[i].num_lch_g * (((double) LCHLength) /
                         g.ie_array[i].RLCH);
    start_pointer += g.ie_array[i].num_sch_g * SCHTime;
}

double tfim_dados = g.tfim_trama - num_rch * OneRCHTime;

nies.collect(num_ie);
tidle += (tfim_dados - start_pointer);

// Verificação de erros no escalonamento
if (((!rr_queue[0].empty()) || (!rr_queue[1].empty())) &&
    ((tfim_dados - start_pointer + 1E-12) > (IEBlockTime + ULGuardTime +
                                             ULPreTime + LCHTime)))
{
    cout << "Recursos nao aproveitados totalmente." << endl;
    cout << "num_ie = " << num_ie << endl;
    cout << "num_DL_RG = " << num_DL_RG << endl;
    cout << "tindep = " << tindep << endl;
    cout << "tdep_ie (num_ie) = " << tdep_ie << endl;
    cout << "tdep_rg = " << tdep_rg << endl;
    cout << (tfim_dados - start_pointer) << " <> " << (IEBlockTime +
                ULGuardTime + ULPreTime + LCHTime) << endl;
    cout << rr_queue[0].length() << endl;
    cout << rr_queue[1].length() << endl;
```

```
        }

        #ifdef _NUM_RCH_VAR
            // Aumenta o num_rch, aproveitando os recursos não aproveitados
            long num_rch_av = (long) ((g.tfim_trama - start_pointer + 1E-12) /
                              OneRCHTime);
            if (num_rch_av < num_rch)
                error("num_rch_av < num_rch");
            num_rch = ((num_rch_av > 31) ? 31 : num_rch_av);
        #endif
        nrchs.collect(num_rch);

        // Envia mensagem de difusão.
        cMessage *broadcast = new cMessage("BROADCAST", H2_DIFUSAO);
        broadcast->setLength(BCHLength + num_ieb_f(num_ie) * IEBlockLength +
                              ACHLength);
        broadcast->addPar("num_rch") = num_rch;
        broadcast->addPar("num_ie") = num_ie;
        send(broadcast, "to_meio");
        scheduleAt(simTime() + FrameTime, send_h2_broadcast);  // próxima trama MAC

        // Escalona CLEAR_ACH, antes de receber novos RRs no RCH, mas depois
        // do recebimento do BROADCAST (que inclui o ACH) pelos adapts.
        scheduleAt(g.tfim_trama - num_rch * OneRCHTime, clear_ach);
    }
    break;
    case H2_DADOS:
    {
        ev << "DADOS" << endl;
        bool repetido;
        src_addr = msg->par("mac_id");
        long classe = msg->par("classe");

        long pkt_seq_num = (long) msg->par("seq_num");

        if (pkt_seq_num == -1) // É dummy
        {
            #ifdef _FCA_PLUS
                FCPointer[src_addr] = g.tfim_trama;
            #endif
            delete msg;
            break;
        }

        if (pkt_seq_num <= last_seq_num[src_addr])
            repetido = true; // Pacote repetido
        else
        {
            last_seq_num[src_addr] = pkt_seq_num;
            repetido = false;
        }
        // Com tráfego ATM, só tem uma mensagem em pay_queue.
        cQueue *pay_queue = (cQueue *) (cObject *) msg->par("pay_queue");
        int b = pay_queue->length();
        bclasse[classe].collect(b);
        bterm[src_addr].collect(b);
        #ifdef _TRANSIENTE
        if (src_addr == reg_term)
            fprintf(fbt,"%15.9f%8d%15.9f\n" , simTime(), b, bterm[src_addr].mean());
        #endif
        if (!msg->hasBitError())
        {
            if (g.retransmite)
            {
                if (repetido)
                {
                    // if (src_addr == reg_term) cout << "R" << pkt_seq_num << "\t" <<
                        flush;
                    i = 1;
                    // remove da NAK_list
                    for (cLinkedListIterator iter(NAK_list[src_addr], 0); !iter.end();
                            i++, iter--)
                        if (*((long *) iter()) == pkt_seq_num)
                        {
```

```
                    delete NAK_list[src_addr].remove(iter());
                    break;
                }
            if (iter.end())
                error("Não encontrou o pacote repetido na NAK_list!");

            // Insere na ul_pkt_queue, na ordem certa
            for (cQueueIterator iter_q(ul_pkt_queue[src_addr], 0);
                    !iter_q.end(); iter_q--)
            {
                cMessage *dados = (cMessage *) iter_q();
                if (((long) dados->par("seq_num")) > ((long) msg-
                        >par("seq_num")))
                {
                    ul_pkt_queue[src_addr].insertAfter(dados, msg);
                    break;
                }
            }
            if (iter_q.end()) // É o último da fila
                ul_pkt_queue[src_addr].insert(msg);

            long despacha_ate;
            if (i == 1) // primeiro
            {
                if (NAK_list[src_addr].empty())
                    despacha_ate = last_seq_num[src_addr];
                else
                    despacha_ate = *((long *) NAK_list[src_addr].tail()) - 1;
                while (!ul_pkt_queue[src_addr].empty())
                    if (((long) ((cMessage *) ul_pkt_queue[src_addr].tail())-
                            >par("seq_num")) <= despacha_ate)
                        despachaPacote((cMessage *)
                            ul_pkt_queue[src_addr].pop());
                    else
                        break;
            }
        }
        else // (!repetido)
        {
            if (NAK_list[src_addr].empty())
            {
                // if (src_addr == reg_term) cout << pkt_seq_num << "\t" <<
                    flush;
                despachaPacote(msg);

                if ((last_seq_num[src_addr] + 1 - TxBoWs[src_addr]) >=
                        MAX_SEM_CAI)
                    if (!contemTerm(ARQ_pkt_queue[classe], src_addr))
                    {
                        cMessage *arq = new cMessage("ARQ", H2_ARQ,
                                                     SCHLength);
                        arq->addPar("mac_id") = src_addr;
                        ARQ_pkt_queue[classe].insert(arq);
                        // cout << "Aplica-se" << endl;
                    }
            }
            else
            {
                // if (src_addr == reg_term) cout << "(" << pkt_seq_num << ")"
                        << "\t" << flush;
                ul_pkt_queue[src_addr].insert(msg);
            }
        }
    }
    else // (!g.retransmite)
        despachaPacote(msg);
}
else // (msg->hasBitError())
{
    if (g.retransmite)
    {
        // if (src_addr == reg_term) cout << "P" << pkt_seq_num << "\t" <<
                flush;
        if (!contemTerm(ARQ_pkt_queue[classe], src_addr))
```

```
                {
                    cMessage *arq = new cMessage("ARQ", H2_ARQ, SCHLength);
                    arq->addPar("mac_id") = src_addr;
                    ARQ_pkt_queue[classe].insert(arq);
                }

                if (!repetido)
                    NAK_list[src_addr].insert(new long(pkt_seq_num));

                inc_RR[src_addr]++;
            }
            delete msg;
        }
    }
    break;
    case H2_RR:
    {
        ev << "RR" << endl;
        long rch_index = (long) msg->par("rch_index");
        if (rch_index != 0) // se não veio num SCH (veio num RCH) ...
            g.rch_feedback[rch_index] = true; // confirma recepção no ACH
        armazenaRR(rr_queue[(long) msg->par("classe")], msg);
    }
    break;
    case CLEAR_ACH:
    {
        ev << "CLEAR_ACH" << endl;
        for (int i = 1; i < 32; i++)
            g.rch_feedback[i] = false;
    }
    break;
    default:
        error("Mensagem invalida na EB: %d", msg_type);
    }

    checkPoint();
}

void cEBDemand::finish()
{
    cout << "num_ie_med = " << nies.mean() << "\t\t";
    cout << "num_ie_max = " << nies.max() << "\t\t";
    cout << "num_rch_med = " << nrchs.mean() << endl;
    cout << "Parcela desocupada dos recursos (antes da expansao do RCH) = " << tidle /
(simTime() - g.start_time) << endl;

    delete send_h2_broadcast;
    delete clear_ach;

    delete [] ul_pkt_queue;

    delete [] last_seq_num;
    delete [] inc_RR;
    delete [] TxBoWs;

    for (i = 0; i < g.num_term; i++)
        while(!NAK_list[i].empty())
            delete NAK_list[i].pop();
    delete [] NAK_list;

    cEB::finish();
}

long cEBDemand::num_ieb_f(long num_ie)
{
    if (num_ie <= 0)
        return 1;
    else
        return (1 + (num_ie - 1) / 3);
}

long cEBDemand::inc_num_ieb(long num_ie)
{
    return (num_ieb_f(num_ie) - num_ieb_f(num_ie - 1));
```

```
}

long cEBDemand::numChAv(long mac_id, long tipo_RG, long tipo_ch, double RLCH)
{
    // Retorna o número de canais disponíveis de um dado tipo (LCH ou SCH),
    // em uma dada direcção (DL_RG ou UL_RG), para um dado mac_id.
    // Se houver canais e o mac_id for novo, actualiza num_ie, ie_index,
    // tdep_ie, l_ie_array[].mac_id e l_ie_array[].tipo.
    for (ie_index = 0; ie_index < num_ie; ie_index++)
        if (l_ie_array[ie_index].tipo == tipo_RG)
            if (l_ie_array[ie_index].mac_id == mac_id)
                break; // for
    double tdep_ie_t; // tenta, não sabe se o tdep_ie vai mudar.
    if (ie_index == num_ie)
    {
        tdep_ie_t = tdep_ie + inc_num_ieb(num_poll + num_ie + 1) * IEBlockTime;
        if (tipo_RG == DL_RG)
            tdep_ie_t += DLPreTime;         // PHY DL Phase
        else if (tipo_RG == UL_RG)
            tdep_ie_t += ULGuardTime + ULPreTime; // PHY UL Phase
        else
            error("Tipo de RG nao definido");
    }
    else // senão o num_ie não aumenta, logo tdep_ie não muda.
        tdep_ie_t = tdep_ie;

    double Tch;
    if (tipo_ch == LCH)
        if (ie_index < num_ie)  // Considera o débito definido inicialmente
            Tch = ((double) LCHLength) / l_ie_array[ie_index].RLCH;
        else                    // Define o débito usado pela conexão nesta trama
            Tch = ((double) LCHLength) / RLCH;
    else if (tipo_ch == SCH)
        Tch = SCHTime;
    else
        error ("Tipo de canal nao definido");

    long num_ch_av = (long) ((FrameTime - tindep - tdep_ie_t - tdep_rg + 1E-12) /
                         Tch);

    if (num_ch_av <= 0)
        return 0; // sai sem executar o resto

    if (ie_index == num_ie)
    {
        l_ie_array[ie_index].mac_id = mac_id;
        l_ie_array[ie_index].tipo = tipo_RG;
        l_ie_array[ie_index].RLCH = RLCH;
        num_ie++;
        tdep_ie = tdep_ie_t;
    }

    return num_ch_av;
}

void cEBDemand::armazenaRR(cQueue &rr_queue, cMessage *msg)
{
    cMessage *rr;
    long mac_id = msg->par("mac_id");
    double tstamp = msg->par("tstamp");
    #ifdef _ESC_ORDEM_TSTAMP
        // Remove da queue o RR anterior de um adapt. (se houver)
        for (cQueueIterator iter(rr_queue, 0); !iter.end(); iter--)
        {
            rr = (cMessage *) iter();
            if ((long) rr->par("mac_id") == mac_id)
            {
                delete rr_queue.remove(rr);
                break;
            }
        }
        // Insere por ordem de timestamp()
        {
        for (cQueueIterator iter(rr_queue, 0); !iter.end(); iter--)
```

```
        {
            rr = (cMessage *) iter();
            if ((double) rr->par("tstamp") > tstamp)
            {
                rr_queue.insertAfter(rr, msg); // era Bef.
                break;
            }
        }
        if (iter.end())
            rr_queue.insert(msg);
        }

    #else // !_ESC_ORDEM_TSTAMP
        // Armazena RRs numa FIFO, substituindo RRs antigos se o mac_id já
        // estiver na lista e inserindo no final se não estiver
        for (cQueueIterator iter(rr_queue, 0); !iter.end(); iter--)
        {
            rr = (cMessage *) iter();
            if ((long) rr->par("mac_id") == mac_id)
            {
                rr_queue.insertBefore(rr, msg); // era Aft.
                delete rr_queue.remove(rr);
                break;
            }
        }
        if (iter.end())
            rr_queue.insert(msg);
    #endif // !_ESC_ORDEM_TSTAMP
}

bool cEBDemand::contemTerm(cQueue &queue, long addr)
{
    for (cQueueIterator iter(queue, 0); !iter.end(); iter--)
    {
        if (((long) ((cMessage *) iter())->par("mac_id")) == addr)
        {
            return true;
            error("Nao deve passar por aqui!");
        }
    }
    return false;

}

bool cEBDemand::escalonaLCHs(cQueue &aqueue, long num_lch_max, bool move_fim, double
RLCH)
{
    while ((!aqueue.empty()) && ((num_ie + num_poll) < 45))
    {
        long num_lch_esc, num_lch_av;
        cMessage *amsg = (cMessage *) aqueue.tail();
        long num_lch_r = amsg->par("num_lch_r");
        long mac_id = amsg->par("mac_id");

        if ((num_lch_av = numChAv(mac_id, UL_RG, LCH, RLCH)) == 0)
            break; // while

        if (num_lch_max <= 0)
            num_lch_esc = num_lch_av;
        else
            num_lch_esc = num_lch_max;

        if (num_lch_r <= num_lch_esc)
        {
            l_ie_array[ie_index].num_lch_g += num_lch_r;
            tdep_rg += num_lch_r * (((double) LCHLength) / l_ie_array[ie_index].RLCH);
            delete aqueue.pop();
        }
        else
        {
            l_ie_array[ie_index].num_lch_g += num_lch_esc;
            tdep_rg += num_lch_esc * (((double) LCHLength) /
                        l_ie_array[ie_index].RLCH);
            amsg->par("num_lch_r") = num_lch_r - num_lch_esc;
```

```
            if (move_fim) // Coloca os restantes no fim da fila
            {
                amsg->par("tstamp") = simTime(); // compromisso
                aqueue.insert(aqueue.pop());
            }
        }

        #ifdef _RR_COM_LCH
            // Inclui um SCH no RG se ainda não incluiu e houver espaço.
            //if (l_ie_array[ie_index].num_sch_g == 0)
            if ((l_ie_array[ie_index].num_sch_g == 0) && (mac_id < g.num_term1))
                if ((FrameTime - tindep - tdep_ie - tdep_rg + 1E-12) > SCHTime)
                {
                    l_ie_array[ie_index].num_sch_g = 1;
                    tdep_rg += SCHTime;
                }
        #endif // _RR_COM_LCH
    }
    if (!aqueue.empty())
        return false;
    else
        return true;
}

void cEBDemand::escalonaARQs(cQueue &ARQ_pkt_queue, cQueue &rep_queue)
{
    long limite_ARQ = 45;
    cQueueIterator i_ARQ(ARQ_pkt_queue, 0);
    while ((!i_ARQ.end()) && ((num_ie + num_poll) < limite_ARQ))
    {
        cMessage *arq = (cMessage *) i_ARQ();
        long mac_id = (long) arq->par("mac_id");

        if (numChAv(mac_id, DL_RG, SCH, 0) == 0)
                break;

        l_ie_array[ie_index].num_sch_g = 1;
        tdep_rg += SCHTime;
        num_DL_RG++;

        #ifdef _REP_AUTO_EB
            if (inc_RR[mac_id])
            {
                cMessage *rep;
                // Soma mais inc_RR lchs na mensagem rep, se houver o mac_id
                for (cQueueIterator iter(rep_queue, 0); !iter.end(); iter--)
                {
                    rep = (cMessage *) iter();
                    if ((long) rep->par("mac_id") == mac_id)
                    {
                        rep->par("num_lch_r") = (long) rep->par("num_lch_r") +
                                                inc_RR[mac_id];
                        break; // for
                    }
                }
                if (iter.end())  // não há => cria a mensagem rep e insere no fim
                {
                    rep = new cMessage;
                    rep->addPar("mac_id") = mac_id;
                    rep->addPar("num_lch_r") = inc_RR[mac_id];
                    rep_queue.insert(rep);
                }
                inc_RR[mac_id] = 0;
            }
        #endif // _REP_AUTO_EB
        i_ARQ--;
    }
}
```