Adaptador de Terminal e Concentrador ATM, baseado num PC com Sistema Operativo Linux

José Manuel Tavares Vieira Cabral
José Gerardo Vieira Rocha
Joaquim José Esteves Neves
José António Ruela Simões Fernandes

Departamento de Electrónica Industrial
Centro Algoritmi
Escola de Engenharia
Universidade do Minho

Março de 2005

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AAL	ATM Adaptation Layer
AAU	ATM User to ATM User Indication
ABR	Available Bit Rate
ABT	ATM Block Transfer
ANSI	American National Standards Institute
ATC	ATM Transfer Capability
ATM	Asynchronous Transfer Mode
B-NT1	Broadband Network Termination 1
B-NT2	Broadband Network Termination 2
BOM	Beginning Of Message
B-TA	Broadband Terminal Adapter
B-TE1	Broadband Terminal Equipment 1
CAN	Controller Area Network
CBR	Constant Bit Rate
CDV	Cell Delay Variation
CLP	Cell Loss Priority
CLR	Cell Loss Ratio
COM	Continuation Of Message
CPCS	Common Part Convergence Sublayer
CRC	Cyclic Redundancy Code
CS	Convergence Sublayer
CSMA-CD	Carrier Sense Multiple Access with Collision Detection
CTD	Cell Transfer Delay
DBR	Deterministic Bit Rate
EDD	Earliest Due Date
EDF	Earliest Deadline First
EOM	End Of Message
ETSI	European Telecommunications Standards Institute
FCFS	First Come First Served
FEC	Forward Error Correction
FFQ	Fluid Fair Queuing
FIFO	First-In-First-Out
FPS	Fast Packet Switching

GFC	Generic Flow Control
GPS	Generalized Processor Sharing
HEC	Header Error Control
IEEE	Institute for Electrical and Electronics Engineers
IMT-2000	International Mobile Telecommunications 2000
ISO	International Standards Institute
IT	Information Type
ITU-T	International Telecommunication Unit - Telecommunication Standardization Sector
IWF	Interworking Function
LAN	Local Area Network
LI	Length Indicator
LSB	Least Significant Bit
MA	Medium Adapter
MAC	Medium Access Control
MAN	Metropolitan Area Network
MCR	Minimum Cell Rate
MID	Message Identifier
MSB	Most Significant Bit
NNI	Network Node Interface
OAM	Operation and Maintenance
OSI	Open Systems Interconnection
PABX	Private Automatic Branch Exchange
PAD	Padding
PC	Personal Computer
PCF	Point Coordination Function
PCM	Pulse Coded Modulation
PCR	Peak Cell Rate
PDU	Protocol Data Unit
PDV	Packet Delay Variation
PL	Physical Layer
PM	Physical Medium
POH	Path Overhead
PRM	Protocol Reference Model
PSTN	Public Switched Telephone Network
PT	Payload Type
QoS	Quality of Service
RDIS	Rede Digital com Integração de Serviços

RLP	Received Loss Priority
SAP	Service Access Point
SAR	Segmentation and Reassembly Sublayer
SBR	Statistical Bit Rate
SCR	Sustainable Cell Rate
SDL	Specification and Description Language
SDH	Synchronous Digital Hierarchy
SDU	Service Data Unit
SN	Sequence Number
SNP	Sequence Number Protection
SOH	Section Overhead
SONET	Synchronous Optical Network
SPC	Stored Program Control
SSCS	Service Specific Convergence Sublayer
ST	Segment Type
STF	Start Field
STM	Synchronous Transfer Mode
STM-N	Synchronous Transport Module - Level N
STS-N	Synchronous Transport Signal - Level N
TC	Transmission Convergence
TCP	Transport Control Protocol
TDM	Time Division Multiplexing
TE2	Terminal Equipment 2
UBR	Unspecified Bit Rate
UNI	User to Network Interface
USB	Universal Serial Bus
VBR	Variable Bit Rate
VC	Virtual Channel
VCI	Virtual Channel Identifier
VP	Virtual Path
VPI	Virtual Path Identifier
WFQ	Weighted Fair Queuing
WLAN	Wireless Local Area Network
WRR	Weighted Round-Robin

1

Introdução

A evolução tecnológica ao nível da micro-electrónica tem conduzido ao aparecimento de novos dispositivos (e.g. sensores, actuadores, sistemas de aquisição de dados) a preços bastante reduzidos. Como consequência, o número de aplicações de controlo distribuído, que envolvem a interligação de uma grande variedade de equipamentos, tem vindo a crescer em áreas como a agro-pecuária, a domótica, o controlo industrial e a indústria automóvel.

A interligação destes equipamentos pode ser caracterizada, em geral, por apresentar fluxos de baixo débito e por exigir à rede baixos atrasos de transmissão. Devido a estas características, este tipo de tráfego é, por vezes, suportado por redes específicas tais como os vulgarmente designados Barramentos de Campo, de que são exemplos o CAN (CAN, *Controller Area Network*) e o PROFIBUS (PROFIBUS, *Process FieldBus*). Estas tecnologias possuem bastantes limitações, principalmente em aspectos relacionados com a integração de serviços e sistemas, largura de banda e área abrangida.

As actuais redes de comunicação foram, em geral, optimizadas para o suporte de serviços específicos, tendo sido algumas adaptadas para suportar outro tipo de serviços. A tendência actual para uma maior integração de serviços, pressupõe um conceito fundamental para atingir esse objectivo: a Qualidade de Serviço (QoS, *Quality of Service*). A especificação e medição da QoS são problemas complexos, quer do ponto de vista do utilizador quer do ponto de vista da rede. O utilizador deverá adaptar os requisitos das suas aplicações aos níveis de desempenho oferecidos pela rede, enquanto que a rede deverá garantir ao utilizador a plena satisfação dos parâmetros de QoS estabelecidos no início das sessões, ou renegociados posteriormente, de forma a ser possível o correcto desempenho das aplicações suportadas.

1.1 Caracterização do problema e cenários de aplicação

Actualmente é já possível observar aplicações baseadas em computador a serem aplicadas à agricultura e pecuária com as consequentes vantagens ao nível da gestão da produção. Nestas áreas, o futuro aponta para a concepção de arquitecturas sensoriais cada vez mais inteligentes o que implica a existência de mecanismos eficientes de suporte à transferência de informação entre dispositivos e estações de controlo e gestão de todo o processo produtivo.

O estudo da adaptação deste tipo de tráfego em redes de comunicação, tendo em vista o alargamento das áreas abrangidas, do número de equipamentos a interligar e da integração com outro tipo de equipamentos e sistemas foi efectuado baseado no estudo de sistemas e aplicações relacionados com a agro-pecuária. No entanto, apesar das especificidades destes sistemas e aplicações, é possível alargar a sua área de aplicação uma vez que os requisitos destes são bastante semelhantes a outros usados nas diversas aplicações de controlo industrial.

Este estudo identificou a necessidade de interligar um número elevado de dispositivos e sistemas com requisitos de tráfego diversos. Por outro lado, a necessidade de alargar a área abrangida conduz à definição de uma arquitectura de suporte para aplicações de controlo suportadas numa infra-estrutura de rede.

A escolha da tecnologia de rede de suporte a toda a infra-estrutura de comunicação é de capital importância. Com efeito, esta deverá permitir garantir os seguintes requisitos:

- endereçamento de um elevado número de dispositivos,
- garantir a satisfação da QoS associada a cada serviço,
- suportar aplicações em áreas de grande dispersão geográfica.

Em primeiro lugar será necessário inventariar e caracterizar os dispositivos a interligar, de modo a ser possível especificar as características de tráfego (e.g. débito, QoS). Seguidamente será necessário definir uma estrutura de multiplexagem capaz de agregar o tráfego proveniente das várias fontes de forma a ser transmitido na rede duma forma eficiente assim como permitir definir os requisitos de QoS associados ao transporte de cada fluxo de informação. Finalmente, será necessário efectuar o estudo de algoritmos de escalonamento temporal de forma a ser possível multiplexar os vários fluxos em função da QoS definida para cada um.

1.2 Objectivos do trabalho desenvolvido

O estudo da adaptação de tráfego de baixo débito, gerado pelas aplicações de controlo industrial permitiu, por um lado, identificar algumas limitações das capacidades de transmissão, endereçamento, área de cobertura dos actuais sistemas de aquisição de dados e controlo e de interligação remota de diferentes redes industriais e, por outro, constatar que recorrendo às redes públicas de telecomunicações, designadamente às redes ATM (ATM, *Asynchronous Transfer Mode*), poderia ser encontrada uma solução para superar essas limitações.

Neste contexto, o trabalho apresentado nesta tese, explorou vários cenários para interligação remota de sistemas de aquisição de dados e de controlo, tendo estabilizado numa solução caracterizada por uma arquitectura centralizada com dois níveis de concentração de tráfego.

A especificação, a implementação e a análise de desempenho desta solução constituem as três contribuições fundamentais deste trabalho.

Em primeiro lugar, foi definida a arquitectura proposta, que inclui uma aplicação de controlo remoto para interligar os diversos dispositivos através de dois módulos fundamentais: o Adaptador de terminal e o Concentrador. O primeiro destina-se a efectuar a ligação directa ou indirecta dos sensores e / ou actuadores, enquanto que o segundo permite interligar os vários Adaptadores de terminal, efectuando um espécie de multiplexagem de segunda ordem.

Posteriormente, foi necessário inventariar e caracterizar as diversas fontes geradoras de tráfego do ponto de vista de débito gerado, características temporais e estatísticas de tráfego, para dimensionar os módulos necessários ao suporte de aplicações. Mostrou-se necessário também definir uma estrutura de multiplexagem capaz de suportar o transporte da informação das diversas fontes de tráfego duma forma eficiente, quer do ponto de vista da rede, quer do ponto de vista da satisfação da QoS exigida pelo serviço.

A QoS dos diversos serviços a suportar está directamente relacionada com o atraso que a informação demora a percorrer entre a fonte e o destino. O atraso de transmissão é composto por duas componentes: o atraso de propagação na rede e os atrasos de empacotamento no Adaptador de terminal e no Concentrador. A primeira componente resulta da escolha da rede ATM como suporte e como tal terá de ser negociada uma ligação que garanta a QoS requerida. A segunda, requer o estudo detalhado de algoritmos de escalonamento de forma a que serviços que imponham restrições temporais tenham prioridade mais elevada nas diversas fases dos processos de empacotamento.

No sentido de avaliar correctamente a especificação dos diversos módulos do sistema foi necessário criar modelos computacionais de forma a ser possível simular o seu comportamento e desempenho. Neste contexto, foram avaliados os vários Algoritmos de escalonamento usados no Adaptador de terminal e no Concentrador.

1.3 Organização do documento

Este documento, que apresenta os estudos efectuados que conduziram à especificação, desenvolvimento e avaliação de desempenho duma solução que permite interligar remotamente um elevado número de sensores e actuadores e superar algumas limitações de alguns sistemas de aquisição de dados e controlo, está organizado em 4 capítulos.

Após esta introdução, segue-se o capítulo 2, onde será efectuado o estudo da adaptação de tráfego de baixo débito em aplicações de controlo. Através da apresentação de um modelo de referência genérico, são definidos critérios de QoS para os diversos sistemas e dispositivos a suportar. Seguidamente, é apresentado um estudo comparativo das capacidades de transmissão e de outros parâmetros de QoS, suportados pelas redes de telecomunicações, bem como pelos principais Barramentos de Campo, para justificar a possibilidade de utilização das redes públicas de telecomunicações e da tecnologia ATM, como suporte do modelo proposto e sumariamente introduzido. Finalmente, é apresentado um estudo genérico sobre escalonamento de tráfego, introduzindo os Algoritmos de escalonamento temporal especificados para a garantia da QoS na multiplexagem dos diversos fluxos de baixo débito

No capítulo 3 será apresentada a especificação do sistema proposto, constituída por dois módulos fundamentais: o Adaptador de terminal e o Concentrador. Estes módulos são descritos com detalhe, assim como são apresentadas as funções principais da Aplicação de controlo.

No capítulo 4 analisa-se o desempenho do sistema, apresentando e discutindo resultados dos diversos testes efectuados aos Algoritmos de escalonamento estudados, para definir a solução a adoptar.

No Anexo A são apresentadas as listagens dos programas desenvolvidos.

2

Adaptação de tráfego de baixo débito em aplicações de controlo

Uma vez apresentadas as principais tecnologias de suporte da infra-estrutura de comunicação para a adaptação do tráfego de baixo débito em aplicações de controlo, é agora altura de discutir este problema, em termos de cenários de aplicação, tipos de dispositivos a interligar, referenciais de Qualidade de Serviço (QoS, *Quality of Service*) suportados bem como a arquitectura de referência.

Neste capítulo também se caracteriza funcionalmente o desempenho daquelas tecnologias de rede, em função dos atrasos e das capacidades de transmissão. É igualmente apresentada uma solução concreta para a adaptação do tráfego de baixo débito em aplicações de controlo, que inclui uma discrição sumária do sistema proposto, baseada em Adaptadores de terminal e um Concentrador, que incorporam funções de escalonamento de tráfego.

2.1 Caracterização do problema

Normalmente, uma rede de sensores é formada por um grande número de pequenos dispositivos, cujo principal objectivo é detectar e transmitir alguma característica física do ambiente [Per03]. Estes componentes ou nós podem ser utilizados de forma eficiente, mesmo

que sejam milhares, para atingir um objectivo comum. Uma outra visão que se pode ter deste tipo de redes é a de um conjunto de nós individuais que operam isolados, mas que podem formar uma rede com o objectivo de coleccionar as informações individuais de cada sensor para monitorar algum fenómeno. Por vezes, estes nós movem-se juntamente com o fenómeno observado (e.g, sensores colocados em animais para observar o seu comportamento).

As redes de sensores possuem como características principais: o sensor, o observador e o fenómeno. O sensor é o dispositivo que implementa a monitoração física de um fenómeno produzindo uma resposta mensurável a mudanças em condições físicas, tais como temperatura, campo magnético e luz [Meg02]. Muitos modelos de complexidade variada podem ser construídos baseados na necessidade da aplicação e características dos dispositivos. Uma rede de sensores é uma ferramenta para medir e passar informação sobre o fenómeno para o observador dentro do limite de desempenho desejado. O observador efectua consultas que podem ser estáticas (os sensores são programados para fornecer dados de acordo com um padrão específico) ou dinâmicas. A rede pode participar na sintetização da consulta. Por exemplo, filtrando alguns dados dos sensores ou fundindo diversas medidas num valor. As optimizações nestes três níveis são possíveis para melhorar o desempenho.

Um sistema de controlo integra um grande número de sensores, actuadores e respectivas entidades de controlo. Por isso, mesmo que a interligação de cada par de dispositivos terminais apresente débitos baixos e exija atrasos de transmissão moderados, os débitos globais suportados pela rede poderão atingir valores elevados e atrasos que, se não forem controlados, poderão tornar-se inaceitáveis.

2.1.1 Cenários de Aplicação

Estão identificadas inúmeras aplicações em que é necessário que os sistemas de aquisição de dados e controlo suportem a cobertura de áreas de grande dimensão e mesmo geograficamente dispersas. Eis alguns exemplos:

- estufas agrícolas,
- unidades de agro-pecuária,
- indústria têxtil,
- estações meteorológicas,
- sistemas de detecção de fogos florestais.

Os sensores podem também ser usados para monitorar ambientes que sejam de difícil acesso ou perigosos, tais como o fundo do oceano, vizinhanças de actividades vulcânicas, territórios inimigos, áreas de desastres e campos de actividade nuclear. Estes podem também ser usados em tarefas interactivas, como encontrar e detonar explosivos, ou procurar sobreviventes de desastres naturais.

Sistemas sensoriais e periciais

Os sistemas sensoriais e de actuação [Cun04] têm vindo a ser aplicados com sucesso em robots para fins agrícolas. Actualmente, existem várias aplicações destes sistemas podendo referir-se como exemplo a aplicação de tratamentos e a administração de adubos foliares em estufas e os tractores de campo para aplicação controlada de químicos. Um exemplo deste tipo de equipamentos é o de um tractor de aplicação de pesticidas que possui 8 sensores de ultra-sons instalados num braço para controlo individual da altura dos injectores ao solo com vista a aumentar a eficiência do emprego dos tratamentos [Ant00].

A utilização de tecnologias que permitam obter informação sobre os processos fisiológicos e químicos que ocorrem nas plantas, tais como: informação relativa à fotossíntese, potenciais hídricos, fluxos de água e de nutrientes nos caules e distribuição na matéria assimilada, apesar do seu grande interesse, está actualmente condicionada por razões de índole prática, pela grande quantidade de sensores que é necessário interligar para assegurar a representatividade do estado da cultura, e de ordem técnica devido a limitações tecnológicas que impossibilitam obter informação directa de muitos dos fenómenos fisiológicos importantes. Algumas destas técnicas baseiam-se na digitalização de imagens das folhas para medir as suas áreas, na aplicação de impulsos térmicos no caule e na utilização da ressonância magnética nuclear para medir os fluxos de água e nutrientes [Pee96], [Shi96].

A aplicação de sensores de imagem e de técnicas de aquisição e processamento de imagem por computador na agricultura têm também vindo a acentuar-se ao longo dos últimos anos, podendo referir-se como exemplo o desenvolvimento de dispositivos munidos de sistemas de visão guiada para aplicação selectiva de herbicidas [Lei02].

Uma outra área de aplicação que recentemente começou a ser automatizada é a da classificação e selecção de frutos com base nos atributos da dimensão e cor. Nesta área de aplicação da visão por computador, o ser humano é substituído por máquinas para efectuar as operações de inspecção visual que, com base na cor, mede a qualidade do fruto e com base na dimensão efectua a selecção do calibre [Cru02].

A recolha de grandes quantidades de informação requer sistemas de processamento capazes de gerar em tempo-real as decisões mais adequadas ao processo produtivo. Dada a interdependência e a complexidade dos vários factores envolvidos na gestão dos processos agro-industriais, os sistemas de supervisão, os sistemas periciais e a inteligência artificial têm vindo a assumir um papel fundamental na implementação de estratégias de automação e gestão integradas. Diversos sistemas periciais de ajuda ao produtor no diagnóstico e resolução de problemas específicos, como sejam o diagnóstico de doenças, entre outros, têm vindo a ser aplicados com sucesso à agricultura [Gre94], [Bea95].

De uma forma simples, um sistema pericial pode ser definido como um programa de computador que substitui um especialista humano na assistência a um utilizador no diagnóstico de problemas, na selecção de alternativas e na gestão de sistemas. O sistema é composto por um módulo de conhecimento, um de inferência que analisa o anterior e uma interface com o utilizador, como ilustrado na Figura 2-1.

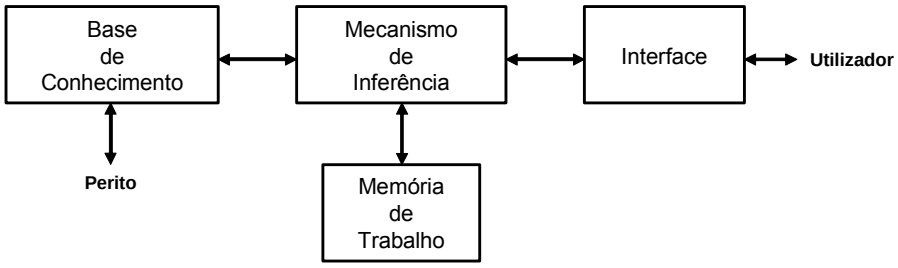


Figura 2-1: Diagrama de blocos de um sistema pericial.

O módulo de conhecimento contém informação e estratégias geralmente obtidas de especialistas, por forma a poderem tomar-se decisões complexas num domínio específico.

Como exemplos de aplicações de sistemas periciais à agricultura podem referir-se os trabalhos de análise e decisão dos nutrientes a fornecer às plantas com base na radiação solar e na resposta destas [Fyn94], de determinação dos valores ambientais diários desejados para a produção de tomate no Inverno, balanceando os objectivos de evitar doenças, manter o desenvolvimento da cultura e minimizar o consumo energético [Mor00], e de diagnóstico de doenças em abelhas [McC93].

Os exemplos apresentados atrás caracterizam-se, em geral, por recolherem informação em diversos locais, onde ocorrem diversos fenómenos que são traduzidos em grandezas mensuráveis (e.g. tensão e corrente eléctrica). A informação é convertida para formato digital de forma a ser processada e enviada através de uma rede de comunicação. A Figura 2-2 ilustra um cenário genérico onde os exemplos anteriores se podem incluir.

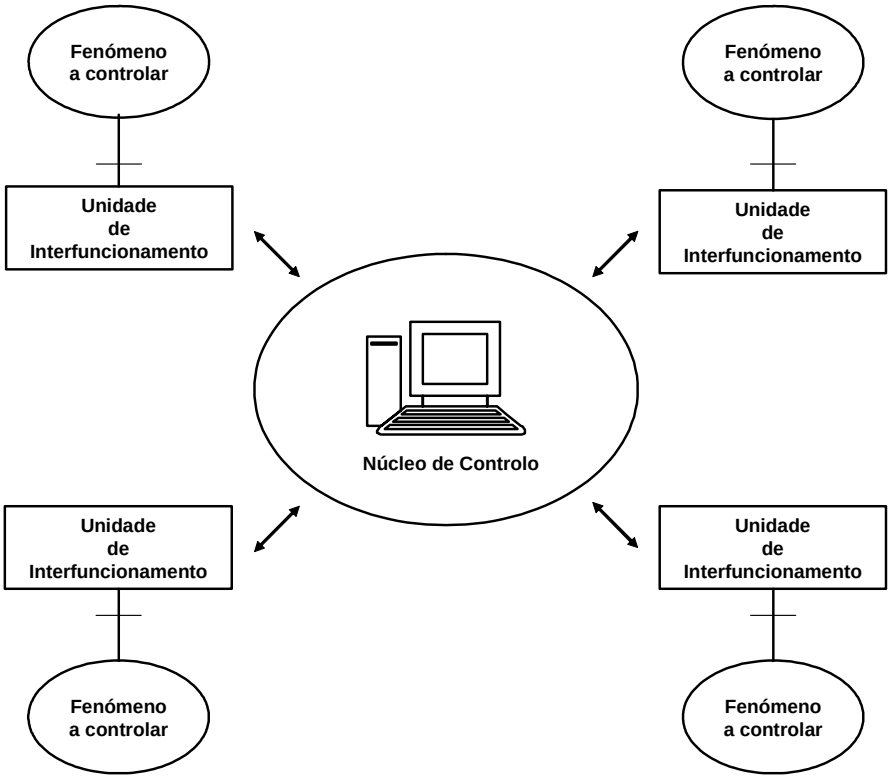


Figura 2-2: Cenário genérico de suporte a aplicações de controlo.

O cenário genérico apresentado na Figura 2-2 caracteriza-se pela existência de um núcleo central de controlo que necessita de recolher e enviar informação de e para os dispositivos associados ao fenómeno envolvido numa dada aplicação.

A informação é recolhida numa unidade central que se encarregará de processar a informação recebida. Uma aplicação de controlo poderá limitar-se a informar o observador acerca do fenómeno observado, fornecendo informação (i.e. medidas, estatísticas, amostras) acerca deste; ou poderá interagir com o próprio fenómeno, ou qualquer elemento relacionado com este, como reacção aos dados recebidos.

- **Núcleo de controlo**

Poderá ser constituído por uma estação de trabalho onde existirão os programas necessários ao controlo de toda a aplicação.

- **Unidade de Interfuncionamento**

Este módulo permitirá efectuar a conversão da informação recolhida, pelos dispositivos em formatos de dados passíveis de poderem ser transportados de forma eficiente para a aplicação e em sentido inverso.

- **Fenómeno a controlar**

Em função das características associadas a um determinado fenómeno, poderão existir dispositivos de diversos tipos e com diferentes características físicas que permitirão recolher a informação acerca das grandezas a medir. Por outro lado, em resposta à informação recolhida pela aplicação, poderá ser necessário intervir no fenómeno. Neste caso a informação será enviada, em sentido inverso, para dispositivos actuadores que desencadeiam acções susceptíveis de alterar o fenómeno a controlar.

Este tipo de aplicações poderá envolver sistemas, dispersos por diferentes áreas geográficas, que podem variar desde uma pequena propriedade rural até áreas que podem envolver um conjunto de países. Deste modo, será conveniente que a infra-estrutura de transporte da informação seja flexível de forma a adaptar-se às necessidades da aplicação, quer em termos de custos, quer em termos de operação e manutenção.

2.1.2 Tipos de dispositivos a interligar

A secção anterior apresentou alguns exemplos de aplicações de controlo industrial em diversas áreas de intervenção. Importa agora identificar os dispositivos terminais e suas principais características de forma a poder sistematizar a análise do problema e assim poder definir a arquitectura do sistema e os critérios de QoS necessários para o seu dimensionamento, selecção da tecnologia de suporte e análise de desempenho.

Eis alguns exemplos de sistemas concretos, ou seus componentes, que podem ser integrados numa aplicação de controlo.

- **Sensores de temperatura**

Estes sensores estão normalmente ligados em placas de aquisição de dados, ou em módulos que produzem informação sob a forma digital. No caso de aplicações relacionadas com a monitorização da temperatura ambiente, a grandeza a medir varia lentamente, enquanto que em processos onde existem fontes de calor artificiais a variação pode ocorrer mais rapidamente. Os períodos de aquisição de sinais desta natureza podem variar desde alguns décimos de segundo, por exemplo em máquinas de secagem de produtos têxteis [Ari02] até alguns minutos, por exemplo em estações meteorológicas [Aud05].

- **Sensores de humidade**

Aplicação idêntica à anterior. Apenas é usado um sensor que mede uma grandeza física diferente.

- **Sensores de luminosidade**

Aplicação idêntica às anteriores. Neste caso a grandeza a medir é a quantidade de luz absorvida pelo sensor. Os processos em que são usados este dispositivos são também caracterizados por constantes de tempo elevadas. Tal como as 2 anteriores, existem aplicações em que a luminosidade deve ser medida com intervalos de nano-segundos [Man03], de alguns minutos em estações meteorológicas e algumas horas em observações de astros.

- **Sensores de estado**

São dispositivos muito simples que apenas indicam o valor de um estado binário ("*On* / *Off*). Podem indicar-se como exemplos o simples detector de estado de um interruptor, detectores de estados de paragem de máquinas industriais e detectores de "fim de curso". Caracterizam-se por produzirem, esporadicamente, quantidades de informação muito baixas (i.e. um bit). A informação gerada por estes sensores pode necessitar de atrasos de transmissão muito baixos.

- **Dispositivos de actuação múltipla**

Um painel de comando é normalmente constituído por um conjunto de interruptores. Consoante o interruptor actuado, é enviado o endereço correspondente. Neste caso, é enviado um octeto sempre que é accionado um interruptor. O atraso de propagação, entre o painel de comando e o actuador correspondente, dependerá da aplicação em causa. Por exemplo, se a acção de comando for ligar a iluminação de uma sala, o atraso não necessita de ser baixo, enquanto que se o comando for no sentido de interromper um processo de fabrico, devido à ocorrência de uma falha, esse atraso deverá ser o mais reduzido possível no sentido de reduzir os danos associados a essa falha.

- **Dispositivos de Voz**

Esta aplicação consiste num serviço de voz de qualidade equivalente à do serviço telefónico tradicional, com débitos de 16kbit/s ou de 8kbit/s de acordo, respectivamente, com as especificações G.728 [ITU92] ou G.729 [ITU96] do ITU-T. Um exemplo de aplicação será a existência de um canal bidireccional de comunicação voz entre o local onde está situado o processo de controlo e o local onde se situa a aplicação de controlo. Nesta aplicação, o atraso

máximo extremo-a-extremo da informação não deverá exceder os 150ms [Iye97]. No caso de transmissão unidireccional, o atraso poderá ser bastante maior e será função do tamanho dos *buffers* de recepção da aplicação em causa.

- **Dispositivos de aquisição de imagens e de vídeo**

Existem vários exemplos de aplicação de dispositivos de aquisição de imagens em processos de controlo industrial. Um exemplo de aplicação simples é o de sistemas de captação de imagens de baixa qualidade, destinadas a aplicações de vigilância, em que se consideram débitos médios na ordem dos 80kbit/s [Pix02]. Em relação ao atraso, o valor máximo dependerá do tamanho do *buffer* de recepção que acomodará as suas variações no sentido de as imagens serem visualizadas sem perturbações. Um outro tipo de aplicações são os chamados "sistemas de visão" em que, através da análise de imagens por computador, se obtém informação acerca de um determinado processo [Cru02], [Fro04], [Jia04]. Neste casos, as imagens recolhidas pelas várias câmaras são enviadas para uma unidade central, onde reside a aplicação, que efectuará todo o processamento no sentido de obter a informação desejada. Os débitos produzidos nestes dispositivos variam em função da resolução das imagens, da quantidade de imagens enviadas por unidade de tempo e do número de câmaras associadas a um determinado processo.

Além dos dispositivos referidos existem outros tipos de sensores (e.g. sensores de pressão, detecção de velocidade, detectores de movimento) que, vistos do ponto de vista da adaptação do tráfego à rede, exibem um comportamento idêntico a um dos atrás mencionados.

2.1.3 Referenciais de QoS suportados

Em função dos requisitos dos diversos dispositivos e aplicações, será necessário definir formas de agregar tráfego, quer ao nível do transporte, quer ao nível da aquisição. No primeiro caso, parece evidente que a agregação de tráfego permitirá transportar os diversos fluxos de informação duma forma mais eficiente (i.e. menores custos de transmissão). No segundo caso, a agregação de tráfego ao nível da aquisição permitirá essencialmente resolver situações em que existem dispositivos que geram tráfego de muito baixo débito (e.g. sensores de estado, alarmes) e de forma ocasional, o que introduz a necessidade de juntar, em unidades de dados elementares, a informação gerada por vários dispositivos e, assim, aumentar a eficiência de transmissão.

O tráfego gerado pelos diversos dispositivos requer, ao nível do transporte da informação, diversos níveis de QoS tais como o atraso de transmissão e a taxa de perdas de informação. Deste modo, neste tipo de aplicações será necessário definir mecanismos que permitam identificar diferentes tipos de requisitos em função dos objectivos a atingir.

Classes de tráfego

Importa agora caracterizar, duma forma clara, a forma como os diferentes fluxos, associados ao tráfego gerado pelos diferentes dispositivos e sistemas apresentados, vão ser tratados pelo sistema de agregação de tráfego. Neste sentido, são definidas três classes de tráfego associadas a diferentes níveis de desempenho suportado pelo sistema de controlo:

- **Atraso Máximo (AM)**

Os fluxos associados a esta classe necessitam da garantia de um atraso máximo, bem definido, entre o dispositivo e a Aplicação de controlo. Esta classe é vocacionada para serviços de tempo-real. Como exemplos, podem ser referidos o serviço de voz interactivo e aplicações em que a leitura de um sensor desencadeia o envio de informação para um actuador dentro de um limite temporal bem definido.

- **Dados (D)**

Nesta classe, os fluxos associados necessitam apenas que lhes seja garantida a entrega da informação sem requisitos temporais. Esta classe é vocacionada para serviços do tipo *unicast*, em que a informação é enviada de um extremo para o outro da rede e em que o processamento da informação não requer o envio de informação em sentido inverso com restrições temporais.

- **Esforço Mínimo (EM)**

Quando a perda ocasional ou o atraso na entrega de informação não afectam determinado processo, podemos usar esta classe de tráfego. Por exemplo, quando estamos a monitorar a temperatura ambiente, a ocorrência de um erro numa leitura poderá não ser grave, uma vez que uma aplicação o poderá detectar facilmente.

2.1.4 Arquitectura de referência

De modo a sistematizar e organizar os conceitos até agora apresentados importa definir uma arquitectura de referência para aplicações de controlo centralizado, em que os diversos processos, associados a diferentes fenómenos, enviam a informação para uma entidade de controlo que poderá enviar informação, em sentido inverso, como resultado do processamento dos dados recebidos. Esse modelo é apresentado na Figura 2-3.

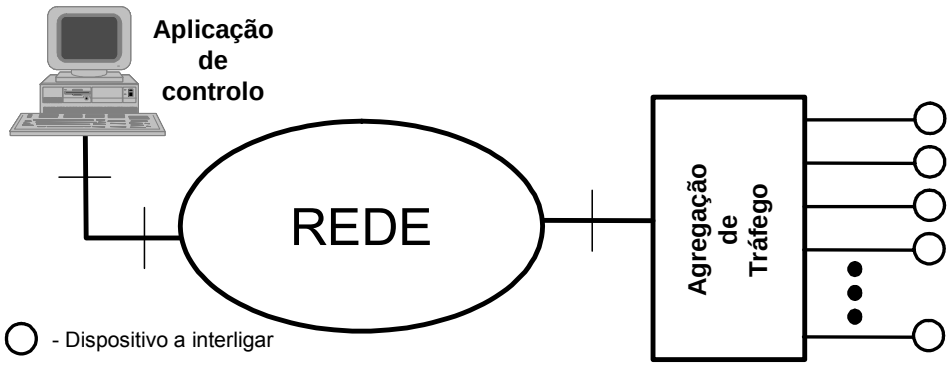


Figura 2-3: Arquitectura de referência para aplicações de controlo.

Esta arquitectura de referência é constituída por três componentes fundamentais:

- Rede de suporte,
- Sistema de agregação de tráfego,
- Aplicação de controlo.

Este modelo pretende ser genérico de forma a poder ser usado como suporte de grande parte das situações apresentadas e servir de referência para o dimensionamento dos diversos parâmetros de QoS a aplicar aos diversos componentes do sistema.

O tráfego gerado pelos dispositivos a interligar interage com a aplicação de controlo através da rede de suporte e do sistema de agregação de tráfego. Deste modo, cada um destes elementos é responsável por uma quota parte da qualidade com que a informação é transferida entre as extremidades do sistema.

Por outro lado, se a área de cobertura tiver uma dimensão que justifique o recurso a uma rede pública, será necessário reduzir ao máximo o débito contratado sem com isso degradar a QoS.

2.2 Características funcionais de desempenho

Quando os requisitos de QoS obrigam a garantir a entrega da informação com atrasos controlados, a rede terá de garantir um atraso máximo de forma ao resto do sistema poder quantificar os seus limites de funcionamento. Assim, o atraso total será composto por duas parcelas:

$$\delta = \delta_R + \delta_A \qquad (2\text{-}1)$$

em que δ_R representa o atraso introduzido pela rede e δ_A o atraso introduzido pelo sistema de agregação de tráfego. O valor da primeira parcela poderá ser garantido através da negociação dos valores dos parâmetros de tráfego e de QoS das ligações a estabelecer com a rede, enquanto que a segunda dependerá de aspectos relacionados com o grau de multiplexagem estatística das fontes e o algoritmo de escalonamento dos fluxos.

No caso em que há interactividade, ou seja, em que a informação processada pela Aplicação de controlo desencadeia uma acção em sentido inverso, é necessário adicionar o atraso da informação no sentido da Aplicação de controlo para o dispositivo final. Neste caso o atraso será dado por:

$$\delta = 2.\delta_R + \delta_A + \delta_D \qquad (2\text{-}2)$$

O valor δ_D corresponde ao atraso na desmultiplexagem dos canais no sistema de agregação de tráfego e é praticamente desprezável face às outras parcelas. Nos casos em que o atraso na rede não é igual nos dois sentidos a parcela $2.\delta_R$ terá de ser corrigida.

Na secção 2.2.1 será apresentado um estudo acerca dos atrasos de transmissão de algumas tecnologias de redes públicas e privadas. Na secção 2.2.2 são apresentados alguns parâmetros relacionados com algumas das tecnologias candidatas ao suporte do sistema, tais como largura de banda, área de cobertura, capacidade de endereçamento, débitos e atrasos suportados. Finalmente, na secção 2.2.3 são fundamentadas as opções de escolha da tecnologia de rede mais adequada, em função dos recursos disponíveis e dos requisitos de QoS, de modo a ser possível o dimensionamento dos vários componentes do sistema.

2.2.1 Atrasos de transmissão

O tempo necessário para a transmissão da informação, ao longo dos vários segmentos digitais da rede, é resultante de duas componentes: o atraso no processamento da informação nos

equipamentos (e.g. empacotamento, atrasos em filas de espera) e o atraso de propagação [ITU00]. Deverá ser dada especial atenção aos casos em que, nos equipamentos de telecomunicações, é efectuado processamento de sinal, empacotamento e desempacotamento de informação e codificação / descodificação de informação.

O valor de 400ms tem sido considerado como limite de referência para efeitos de planeamento, já que a transmissão de informação de voz era o factor que impunha as restrições mais apertadas em termos de desempenho global.

O atraso de transmissão é um parâmetro muito importante para qualquer aplicação cujo desempenho global é dependente da interactividade entre terminais ou utilizadores. Aplicações como a voz, a vídeo-telefonia ou a transmissão de dados podem envolver interacções com o utilizador ou com terminais cuja sensibilidade ao atraso pode variar substancialmente. Uma vez que nem a rede nem os operadores de telecomunicações podem alterar as características relacionadas com o atraso de propagação ou os meios de transmissão entre operadores, em resposta aos diferentes requisitos de utilizadores e aplicações, algumas tarefas que exijam uma maior interactividade poderão sofrer alguma degradação no seu desempenho mesmo para valores do atraso da ordem dos 100ms. Deste modo, o atraso de transmissão deverá ser "consumido" com precaução e só apenas quando este traz benefícios ao desempenho do serviço. Isto aplica-se especialmente no caso dos atrasos introduzidos devido ao processamento de sinal.

O serviço de voz é bastante sensível ao atraso e menos sensível a variações desse atraso (*jitter*). É muito importante garantir que o atraso extremo-a-extremo seja mantido entre valores bem definidos de modo a garantir uma comunicação interactiva com desempenho adequado. O atraso pode produzir dois tipos de efeito no desempenho de uma comunicação [Iye97]: pode interferir na qualidade de uma comunicação de voz na ausência de eco, enquanto que na sua presença, o aumento do atraso, provoca um aumento do seu efeito. Quando o atraso ultrapassa os 30ms, é necessário introduzir circuitos canceladores de eco. Com circuitos canceladores de eco, os atrasos na rede podem ser tolerados até valores da ordem dos 150ms sem degradação da QoS.

A Tabela 2-1 apresenta os limites temporais considerados aceitáveis, no caso da transmissão unidireccional com controlo de eco adequado, de acordo com a recomendação G.114 do ITU-T [ITU00].

Tabela 2-1: Limites temporais para a transmissão unidireccional, retirados da recomendação G.114 do ITU-T.

Atraso	Aceitabilidade
0 - 150 ms	Aceitável na maior parte das aplicações
150 - 400 ms	Aceitável quando o impacto na aplicação o permite
> 400 ms	Inaceitável

Nem sempre é possível prever os atrasos associados a algumas aplicações ou configurações de rede o que pode conduzir, em alguns casos, a que os tempos de processamento combinados com os tempos de propagação conduzam a um tempo de transmissão total superior a 400ms.

Na Tabela 2-2 são apresentados alguns valores retirados da Tabela A.1 da Recomendação G.114 do ITU-T.

Tabela 2-2: Alguns valores retirados da Tabela A.1 da Recomendação G.114 do ITU-T.

Sistema / Meio de Transmissão	Contribuição para o tempo total de transmissão
Cabo coaxial terrestre ou sistema transmissão rádio: FDM e transmissão digital	4μs/km
Cabo de fibra óptica submarino (transmissão digital)	5μs/km
Cabo coaxial submarino	6μs/km

Por exemplo, a contribuição para o tempo total de transmissão de um sistema por cabo coaxial terrestre entre duas localidades afastadas de 600km é de 2,4ms.

Os sinais propagam-se nos meios de transmissão com uma velocidade (**v**) finita. Numa transmissão por rádio a velocidade de propagação é próxima da velocidade da luz (300.000km/s). Quando se utilizam cabos as velocidades de propagação são inferiores. A fibra óptica monomodo permite atingir 70% da velocidade da luz, ou seja, cerca de 210.000km/s. A velocidade de propagação nos cabos de cobre situa-se entre 40% a 60% da velocidade da luz (i.e. entre 120.000km/s a 180.000km/s).

Ao percorrer uma distancia **d** o atraso de propagação será de:

$$T_p = d / v \qquad (2\text{-}3)$$

Esta expressão é válida para dois nós que se encontrem directamente ligados ao mesmo meio físico, o que na prática nem sempre acontece. Quando o sinal passa por diversos meios físicos será necessário contabilizar os atrasos em cada um deles. Além dos atrasos de propagação nos meios físicos têm de ser também considerados os atrasos em outros equipamentos activos da rede (e.g. *bridges*, comutadores, *routers*).

O tempo de transmissão de um pacote T_t depende do seu comprimento (**L** em bits) e da taxa de transmissão nominal (**R** em bit/s):

$$T_t = L / R \qquad (2\text{-}4)$$

Define-se atraso de propagação normalizado como sendo a razão entre o atraso de propagação e o tempo de transmissão:

$$\mathbf{a} = T_p / T_t \qquad (2\text{-}5)$$

As LANs (LAN, *Local Area Network*) são um exemplo típico da importância deste parâmetro sendo neste caso obrigatório que:

$$\mathbf{a} < 0{,}5 \qquad (2\text{-}6)$$

Numa LAN os tempos de atraso são pequenos porque as distâncias são curtas. No entanto, é preciso notar que é nas LANs que as taxas de transmissão tendem a ser mais elevadas; logo os valores do atraso de propagação normalizado poderão não respeitar a equação 2-6. Considere-se o seguinte exemplo:

- Rede Ethernet (IEEE 802.3),
- cabo de cobre com velocidade de propagação de 150.000km/s,
- tamanho mínimo de trama: 64 octetos (imposição CSMA/CD),
- valor máximo de **a**: 0,5 (imposição CSMA/CD).

Uma rede Ethernet a 10Mbit/s permite um comprimento máximo de 2500m (10base5). Para calcular o valor de **a** tem-se que:

$$T_p = 2{,}5 / 150.000 = 16{,}66\mu s \qquad (2\text{-}7)$$

Sendo o débito nominal de 10Mbit/s, e se se considerar um pacote de tamanho mínimo de 64octetos (512bit), obtém-se:

$$T_t = 512 / 10.000.000 = 51{,}2\mu s \qquad (2\text{-}8)$$

O valor obtido para **a** é de cerca de 0,33 o que representa uma folga razoável relativamente ao valor máximo de 0,5. Esta folga justifica-se pela necessidade de entrar em consideração com atrasos na sincronização e atrasos na passagem em dispositivos repetidores.

Supondo agora uma implementação Fast Ethernet a 100Mbit/s:

$$T_t = 512 / 100.000.000 = 5{,}12\mu s \quad (2\text{-}9)$$

O valor obtido para **a** é agora 3,3 não suportado pelo CSMA/CD. Esta é uma das razões pela qual as redes Fast Ethernet têm um comprimento máximo de cerca de 210m. Considerando 210m, obtém-se:

$$T_p = 0{,}21 / 150.000 = 1{,}4\mu s \quad (2\text{-}10)$$

resultando um valor de **a** de 0,27.

No caso de uma rede ATM (ATM, *Asynchronous Transfer Mode*), o atraso pode ser provocado pelos seguintes factores:

- **Tempo de Empacotamento (ou tempo de construção de uma célula):**

Tempo necessário ao preenchimento completo de uma célula antes de ser transmitida. Por exemplo, no caso de canais de voz de 64kbit/s, codificados em PCM (PCM, *Pulse Code Modulation*), o tempo necessário ao preenchimento do campo de informação de uma célula ATM (48 octetos) é de 6ms.

- **Tempo de desempacotamento (*buffering delay*)**

Por vezes, devido aos atrasos ao longo dos vários nós da rede, algumas células chegam ao destino para lá de um tempo determinado (*deadline*). Quando isso acontece a subcamada SAR (SAR, *Segmentation And Reassembling*) da camada AAL (AAL, *ATM Adaptation Layer*) tem de lidar com a falta de informação para processar o que resulta em falhas na conversação, no caso de serviços de voz. Como forma de evitar este fenómeno, a subcamada SAR pode implementar um mecanismo que consiste em acumular informação numa memória antes de iniciar o processo de reconstrução das amostras de voz. Para evitar a ocorrência do esvaziamento dessa memória, o seu tamanho poderá ser tal que provoque um atraso superior ao adequado. Neste caso o tamanho da memória tem implicações directas no atraso o que implica um controlo do parâmetro CDV (CDV, *Cell Delay Variation*) numa rede ATM de forma a reduzir ao máximo este factor.

- **Tempo de codificação**

Tempo gasto no processamento de algoritmos de compressão na codificação de um sinal analógico em formato digital.

A qualidade de um serviço de vídeo é obtida através de parâmetros relacionados com a taxa de tramas, resolução / tamanho da imagem, algoritmo de compressão e o método de codificação [Lap95]. As medidas principais de desempenho na transmissão deste serviço são:

- Latência: atraso verificado entre a fonte de vídeo e o dispositivo de visualização final,
- *Jitter*: variabilidade do parâmetro anterior entre transmissões sucessivas,
- Largura de Banda: recursos de rede necessários ao suporte da transmissão.

Os limites fixados para estes parâmetros variam conforme a aplicação. Por exemplo, a Latência é um parâmetro importante no caso da transmissão em tempo real de aplicações de vídeo, uma vez que atrasos superiores a 300ms podem ser facilmente perceptíveis durante comunicações interactivas. No entanto, em aplicações de *streaming* o *Jitter* é o principal problema.

- **Variação do atraso entre pacotes - *Packet Delay Variation* (PDV)**

A variação do atraso entre pacotes – PDV (PDV, *Packet Delay Variation*), numa rede ATM quando se usa o protocolo AAL-2, é medida entre a submissão de um pacote CPS na unidade de interfuncionamento – IWF (IWF, *Interworking Function*) e a recepção desse mesmo pacote pela sua semelhante na recepção. As causas da variação do PDV são:

- Disciplina de escalonamento da informação proveniente das fontes de entrada,
- Relógio de uso combinado da camada AAL-2 - Timer_CU,
- Filas de espera usadas pelas camadas AAL e ATM, respectivamente em concentradores ou dispositivos de controlo de tráfego,
- Variação do atraso entre células numa rede - CDV.

De acordo com [ATM99a], uma IWF deverá ter a capacidade de acomodar um valor máximo de PDV igual a 20ms, no transporte de pacotes através de uma ligação AAL-2 (incluindo o efeito do CDV). No caso de a ligação não ter sido estabelecida através de sinalização ou através de um outro acordo extremo-a-extremo, o mesmo valor de 20ms deverá ser usado como valor máximo por omissão.

2.2.2 Capacidades de transmissão e parâmetros de QoS das tecnologias estudadas

A escolha da tecnologia de suporte de aplicações de controlo terá de ser baseada em critérios objectivos, baseados na análise dos diversos parâmetros de QoS, que permitam avaliar quantitativamente as diversas opções em causa. Apresentando diferentes capacidades e características, as tecnologias de comunicação, merecem uma breve analise comparativa.

Na Tabela 2-3 são apresentados alguns valores que permitem comparar as capacidades de transmissão e alguns parâmetros de QoS, suportados pelas Redes de Área Local, Barramentos de Campo e pela rede pública digital, através da tecnologia ATM.

Tabela 2-3: Capacidades de transmissão e parâmetros de QoS, suportados pelas Redes de Área Local, Barramentos de Campo e pela rede pública digital através da tecnologia ATM.

Tecnologias		**Débito Binário**	**Atraso**	**Capacidade de endereçamento**	**Cobertura**
IEEE 802.3	**Ethernet**	10Mbit/s	< 200µs	> 16 X 10^6	1500m (Cabo) 25km (Fibra óptica)
	Fast Ethernet	100Mbit/s	< 120µs	> 16 X 10^6	210m (Cabo) 20km (Fibra óptica)
	Gigabit Ethernet	1Gbit/s	< 30µs	> 16 X 10^6	25m (Cabo) 5km (Fibra óptica)
Barramentos de Campo	**CAN**	< 1Mbit/s (40m) < 50kbit/s (1km)	1ms	< 2048	1km
	PROFIBUS	500kbit/s	1ms	127	4,8km
	P-NET	300Trans/s (2)	1ms	300	1,2km
	World-FIP	2,5 Mbit/s	1ms	60 nós (3)	1km
ATM	**PDH**	< 139 Mbit/s	variável (1)	> 16 X 10^6	ilimitada
	SDH	> 139 Mbit/s	variável (1)	> 16 X 10^6	ilimitada

(1) Variável com a distância
(2) Transacções por segundo (na prática corresponde a um valor de 76,8kbit/s [Tov99])
(3) Nós por repetidor

Através da observação da Tabela 2-3, é possível verificar a grande capacidade de transmissão e de endereçamento bem como a área de cobertura propiciadas pela tecnologia ATM relativamente aos Barramentos de Campo.

Estas características permitirão superar as limitações dos Barramentos de Campo, sobretudo nas aplicações de aquisição de dados e de controlo remoto que não exijam grandes restrições relativamente aos atrasos.

Em relação às LANs, o ATM tem a vantagem de permitir áreas de cobertura superiores e maior flexibilidade no endereçamento dos dispositivos associados às aplicações de controlo. Com efeito, apesar de na Tabela 2-3 as capacidades de endereçamento serem idênticas, no caso das redes IEEE 802.3 o endereçamento é feito através do endereço (*MAC Address*) da respectiva carta de rede. Este endereço, sendo único, tem significado físico e universal. Pelo contrário, no caso das redes ATM o endereçamento é feito através de um mecanismo, baseado em caminhos e canais virtuais (VPI/VCI), que não tendo significado físico e tendo validade apenas a nível local, podem ser reutilizados, o que conduz a uma muito maior flexibilidade na gestão dessa capacidade de endereçamento.

Ao contrário das LANs e dos Barramentos de Campo, que podem apresentar tempos de transmissão reduzidos, os atrasos extremo-a-extremo esperados nas redes públicas de telecomunicações, variam em função da distância entre os terminais interligados. No entanto, esta aparente vantagem, relativamente ao ATM, resultante da área de cobertura suportada por aquelas tecnologias ser limitada, pode ser irrelevante para a grande maioria das aplicações.

2.2.3 Escolha da tecnologia de rede

Como foi referido anteriormente, o tráfego produzido pelos dispositivos (e.g. sensores, actuadores) é de muito baixo débito o que pode conduzir a situações de baixa eficiência de transmissão na rede de comunicação. Por outro lado, alguns destes dispositivos requerem a entrega da informação com atrasos controlados, o que exige da rede a capacidade de garantir a transmissão da informação com uma determinada QoS definida no estabelecimento da ligação.

Os valores apresentados na Tabela 2-3 permitem concluir que, usando a tecnologia ATM, é possível resolver algumas limitações, impostas pelos Barramentos de Campo, a determinados requisitos de Aplicações de controlo, nomeadamente ao nível da capacidade de endereçamento, débito binário e área coberta.

Em relação às LANs, o ATM permite também uma muito maior flexibilidade, quer em termos de endereçamento, quer ao nível da garantia de QoS negociada para a ligação, para além de garantir uma maior área coberta.

A tecnologia ATM, através da sua camada de Adaptação AAL, permite implementar uma estrutura de multiplexagem eficiente para agregar o tráfego dos fluxos de baixo débito garantindo, por um lado, a identificação de cada dispositivo e, por outro, a definição de diferentes níveis de QoS em função da especificidade do serviço a tranportar.

Como a tecnologia ATM não especifica o meio físico de transmissão, o transporte de células ATM poderá ser efectuado por um qualquer sistema que garanta a QoS necessária da ligação, entre a Aplicação de controlo e os dispositivos interligados.

Deste modo, ao seleccionar o ATM como suporte da infra-estrura de comunicação do sistema de adaptação de tráfego de baixo débito, será possível garantir, à partida, um conjunto de vantagens enumeradas a seguir:

- Elevada capacidade de endereçamento de dispositivos associados à Aplicação de controlo,
- Estrutura de multiplexagem flexível e eficiente para o transporte da informação dos diversos serviços,
- Garantia de QoS da ligação estabelecida.

Nas secções seguintes será apresentada a solução para o problema, tendo como base a tecnologia ATM para o suporte da infra-estrutura de comunicação do sistema proposto.

2.3 Solução do problema

A grande quantidade de sensores e / ou actuadores que poderão integrar alguns sistemas de aquisição de dados e controlo obriga à definição de uma arquitectura que suporte a sua interligação, maximize a eficiência de transmissão da rede e garanta a QoS requerida pelos serviços a suportar. Por outro lado, é desejável do ponto de vista económico que os sistemas de suporte destas aplicações possam integrar outros serviços de telecomunicações. Estes quatro requisitos apontam para a resolução de quatro problemas fundamentais:

- Capacidade de endereçamento para identificar um elevado número de dispositivos,
- Estrutura da agregação e multiplexagem, para suportar aplicações em áreas de grande dispersão geográfica,
- Algoritmos de escalonamento, para garantir a satisfação da QoS associada a cada serviço,
- Integração de outros serviços para permitir utilização de recursos eventualmente existentes, bem como a expansão de funcionalidades do sistema.

O primeiro, segundo e o quarto problemas são resolvidos directamente através do uso da tecnologia ATM. O problema fica resolvido uma vez que o endereçamento dos diferentes terminais ou fontes de tráfego pode ser efectuado através de um mecanismo baseado nos identificadores de caminho e de canal virtual (VPI/VCI), da camada ATM, bem como pelos identificadores dos canais eventualmente multiplexados pela camada AAL.

A resolução do segundo problema consegue-se através da escolha do protocolo AAL mais adequado à estrutura de multiplexagem pretendida. Existem já inúmeras normas definidas para o suporte de várias aplicações e serviços sobre ATM, principalmente para a adaptação de tráfego de débito constante, como, por exemplo, emulação de circuitos [Cab95], em que se utiliza o AAL-1 [ITU96b], e para emulação de outras redes e protocolos de transferência de dados, em que se utiliza sobretudo o AAL-5 [ITU96d]. No entanto, ainda estão por definir formas de interligar eficientemente alguns sistemas de transferência de dados que foram pensados para operarem isolados e, simultaneamente, adaptar os respectivos serviços de muito baixo débito e com requisitos temporais exigentes, tais como o tráfego gerado em sistemas de controlo distribuído, ou por sistemas de aquisição de dados, onde poderão ser usados protocolos do tipo AAL-2 [ITU97b] ou AAL-3/4 [ITU96c].

A adaptação ineficiente de serviços de muito baixo débito em redes ATM relaciona-se com o atraso no preenchimento dos pacotes ATM (células ATM) que pode degradar substancialmente a Qualidade de Serviço (atraso). A resolução deste problema passa pela criação de uma estrutura modular que permita multiplexar vários serviços de muito baixo débito em células ATM, reduzindo o atraso de empacotamento e aumentando a eficiência da rede.

Finalmente, para resolver o terceiro problema, é necessário estudar algoritmos de escalonamento temporal de forma a cumprir os requisitos, por vezes exigentes, de alguns tipos de aplicações industriais.

2.3.1 Sistema proposto

A Figura 2-4 apresenta o diagrama de blocos para o sistema de adaptação de tráfego de baixo débito em aplicações de controlo. Sem perda de generalidade, a topologia lógica do sistema de concentração é do tipo N para 1, uma vez que é suposto que uma Aplicação de controlo, destinada a recolher informação de vários pontos, esteja situada num único local físico. No entanto, os módulos de adaptação e de concentração poderão ter mais do que uma porta de saída, para interligação a outros módulos constituintes do sistema, por ligações de nível físico, que asseguram o transporte bidireccional de células ATM.

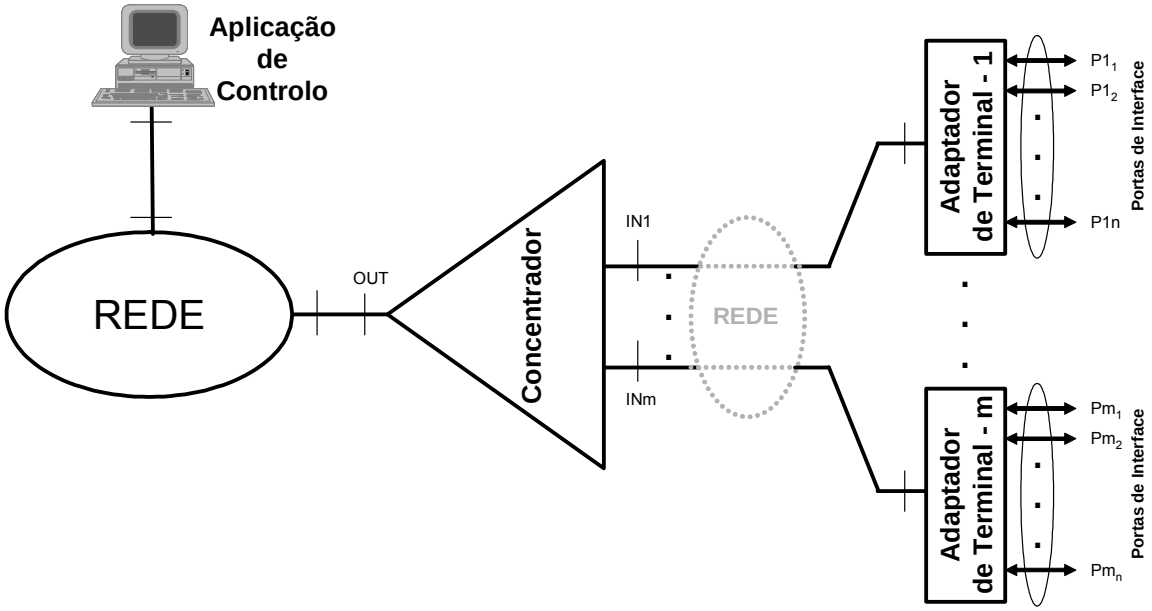


Figura 2-4: Diagrama de blocos do sistema de adaptação de tráfego de baixo débito em aplicações de controlo.

O sistema proposto consiste numa estrutura modular, suportada numa rede, por dois blocos fundamentais:

- Adaptador de terminal,
- Concentrador.

O Adaptador de terminal é o módulo que permite a interligação dos dispositivos de baixo débito com a rede, efectuando a multiplexagem e desmultiplexagem das várias fontes de tráfego, tendo em conta os requisitos de atraso (QoS) de cada uma.

O Concentrador permite não só agregar o tráfego proveniente dos vários Adaptadores de terminal como também efectuar uma re-multiplexagem de todo o tráfego. O seu objectivo principal é o de aumentar a taxa de preenchimento das células ATM e assim aumentar a eficiência de transmissão da rede. De referir que este módulo não é necessário, numa estrutura com apenas um Adaptador de terminal. Neste caso, a Aplicação de controlo e o Adaptador de terminal têm apenas a rede como elemento de interligação.

A Aplicação de controlo é o módulo que permite efectuar a gestão de todos os processos relacionados com o fenómeno a controlar. É implementada numa estação de trabalho onde um conjunto de programas efectuam um conjunto de tarefas específicas destinadas ao controlo da ligação, ao processamento de dados relacionados com a aplicação em causa e ao interface com o operador. Em função da especificidade da aplicação de controlo será necessário

desenvolver funções particulares, necessárias à resolução dos requisitos de cada aplicação. No entanto, existem um conjunto de funções comuns, independentes da aplicação, que terão sempre de existir. Essas funções devem permitir um elevado nível de abstracção no desenvolvimento das funções relacionadas com uma dada aplicação específica.

A interligação entre os diversos Adaptadores de terminal e o Concentrador, ou entre o Concentrador e a rede poderá ser efectuada usando outras tecnologias. No entanto, uma vez que se assume que essas ligações fornecem um canal transparente, não é dada relevância, nesta tese, ao estudo desse problema.

A especificação detalhada do sistema proposto, assim como dos seus módulos constituintes será apresentada no capítulo 3.

2.3.2 Escalonamento de tráfego

Para disciplinar a concentração dos fluxos de informação, provenientes das diferentes fontes de tráfego que concorrem pelo acesso às capacidades de transmissão disponíveis, será necessário desenvolver algoritmos de escalonamento temporal, de modo a determinar a ordem em que esses fluxos são servidos e assim satisfazer os diferentes requisitos de QoS de cada fonte de informação.

O algoritmo de escalonamento mais simples consiste em colocar os pacotes de informação dos diferentes fluxos numa mesma estrutura de memória do tipo FIFO (FIFO, *First In First Out*) pela ordem de chegada, sendo servidos pela mesma ordem. Este algoritmo, denominado por FCFS (FCFS, *First Come First Served*), embora simples de implementar não garante equidade no escalonamento, protecção entre os diversos fluxos de informação e garantia de QoS.

De modo a superar as limitações do algoritmo anterior, foram propostos na literatura [Kes97] e usados outro tipo de algoritmos para efectuar o escalonamento de tráfego assíncrono baseados, na sua maioria, no algoritmo ideal denominado por GPS (GPS, *Generalized Processor Sharing*), também conhecido pela designação FFQ (FFQ, *Fluid Fair Queuing*). Em termos lógicos, este algoritmo atribui uma fila de espera a cada fluxo de informação, visitando cada uma sequencialmente e servindo uma quantidade infinitesimal de informação de cada fila. Deste modo, num intervalo de tempo finito, cada fila é visitada pelo menos uma vez. A cada fluxo pode ser associado um peso diferente, permitindo que a quantidade de dados servida de um dado fluxo seja proporcional ao respectivo peso.

Este algoritmo pode ser implementado, duma forma simples, através de uma mecanismo de rotação (*round-robin*), em que as diversas filas de espera são servidas sequencialmente e da mesma forma que no GPS. No entanto, este mecanismo, serve um pacote de informação de cada vez em vez de uma quantidade infinitesimal. Este algoritmo constitui uma boa aproximação do GPS, nos casos em que os fluxos têm o mesmo peso e os pacotes possuem o mesmo comprimento. O algoritmo WRR (WRR, *Weighted Round-Robin*) é uma variação do mecanismo de rotação em que os fluxos são servidos na proporção dos seus pesos [Kes97].

O WFQ (WFQ, *Weighted Fair Queuing*) é um algoritmo que implementa melhor os critérios definidos pelo GPS, nomeadamente quando os pacotes são de tamanho variável [Dem89]. No entanto, este algoritmo apresenta uma maior complexidade de implementação que o WRR. O algoritmo WFQ atribui a cada pacote um identificador correspondente ao instante em que cada pacote deveria ser servido pelo algoritmo GPS, e serve cada pacote pela ordem dos seus identificadores. O algoritmo WFQ parece, à partida, ser adequado ao tráfego de tempo-real. No entanto, a largura de banda atribuída varia inversamente com o atraso pretendido para a ligação, tornando este algoritmo pouco eficiente quando se pretendem atrasos baixos e taxas de ocupação de banda elevadas [Par94].

Para o escalonamento de tráfego de tempo-real existem descritos outros algoritmos tais como o EDD (EDD, *Earliest Due Date*), também conhecido como EDF (EDF, *Earliest Deadline First*), que consiste em atribuir a cada pacote um prazo de validade (*deadline*), usado pelo escalonador para definir a ordem em que os pacotes deverão ser servidos. 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 lhe foram atribuídos.

Algoritmos de escalonamento

Como foi referido anteriormente, será necessário implementar mecanismos de escalonamento de tráfego de modo a permitir garantir que os fluxos associados a serviços que imponham requisitos de atraso na ligação possam ser suportados pelo sistema com a QoS pretendida.

Os algoritmos desenvolvidos para o Adaptador de terminal e para o Concentrador serão objecto de estudo detalhado no capítulo 4, onde serão especificados detalhadamente e efectuada a sua análise de desempenho. Nesta secção, apenas serão abordados os aspectos mais gerais, enquadrados nas classes de tráfego e outros conceitos apresentados anteriormente.

- **Algoritmo de escalonamento do Adaptador de terminal**

O algoritmo desenvolvido para o Adaptador de terminal caracteriza-se por implementar um mecanismo de prioridade híbrido baseado no algoritmo do tipo EDF e no critério de dar prioridade à fila que, num dado instante, tiver o tamanho maior. Cada fluxo de informação é introduzido numa memória do tipo FIFO, onde lhe é atribuída uma classe de tráfego em função dos requisitos de QoS a suportar pelo sistema. No caso da classe de tráfego ser do tipo AM é também introduzido, através do módulo de configuração, um parâmetro que especifica o atraso máximo (AM) que a informação deverá sofrer no Adaptador de terminal.

Em função dos parâmetros anteriores, é calculado o instante de atendimento (*deadline*) de cada fila do tipo AM, baseado no parâmetro AM (Atraso Máximo). Enquanto não for atingida a *deadline* das filas AM, o algoritmo opta por atender a maior das filas de classe AM ou D. Caso não exista informação destas classes nas filas, atende as filas de classe EF, de forma sequencial, da maior para a menor

- **Algoritmo de escalonamento do Concentrador**

O algoritmo de escalonamento do Concentrador é bastante mais simples que o anterior, uma vez que a informação do parâmetro de atraso máximo (AM) dos fluxos de classe AM, só é usada pelo Adaptador de terminal. Neste caso não é necessário calcular as *deadlines* associadas aos fluxos de classe AM.

Os pacotes que vão sendo recebidos, à entrada do Concentrador, são colocados em 3 filas específicas, em função da classe de tráfego que lhes está associada. O escalonador atende as filas pela seguinte ordem: AM, D e EM. Enquanto houver pacotes numa dada fila, não são atendidos os pacotes da fila menos prioritária.

Devido ao efeito de concentração, poderá ocorrer o fenómeno de *overflow* no FIFO de pacotes da classe EM, uma vez que estes pacotes têm a prioridade mínima de atendimento. Nestas situações, o algoritmo limita-se apenas a rejeitar os pacotes que excedam a capacidade do FIFO.

3

Especificação do sistema proposto

No capítulo anterior foi apresentada uma solução que permite superar algumas das limitações dos principais sistemas que constituem as redes industriais. No entanto, para atingir os objectivos propostos, torna-se necessário especificar, em concreto, a arquitectura geral do sistema, composta por um ou vários Adaptadores de terminal e um Concentrador ligados, através de uma Rede de comunicação, a uma Aplicação de controlo.

A Rede de comunicação poderá ser uma rede ATM (ATM, *Asynchronous Transfer Mode*) ou, então, terá de ter a capacidade de transportar, de uma forma transparente, células ATM. A tecnologia ATM foi desenvolvida baseada no pressuposto de ser independente das tecnologias do meio físico de transmissão e, como tal, se adaptar a qualquer tecnologia.

O Adaptador de terminal e o Concentrador são módulos que permitem optimizar a ocupação da largura de banda disponível para a agregação dos fluxos de baixo débito proveniente dos serviços a suportar. Essa agregação poderá ser integrada com outro tipo de tráfego, proveniente de outro tipo de serviços, que permitirá optimizar, ainda mais, a eficiência de transmissão. Estes módulos permitem também garantir o transporte de serviços com requisitos de atraso apertados através do dimensionamento de algoritmos de escalonamento apropriados.

Assim, neste capítulo são apresentadas as especificações funcionais do Adaptador de terminal e do Concentrador, assim como os requisitos fundamentais para o desenvolvimento da Aplicação de controlo.

3.1 Arquitectura geral do sistema proposto

No capítulo 2 foram apresentados os pressupostos que deram origem à especificação desta configuração e foi introduzida a sua arquitectura. Como foi ilustrado na Figura 2-4, o sistema proposto consiste numa estrutura modular suportada por dois blocos fundamentais:

- Adaptador de terminal,
- Concentrador.

O Adaptador de terminal permite a interligação de todos os dispositivos envolvidos na aplicação de controlo à rede. Assim, cada porta do Adaptador de terminal poderá suportar a ligação de um dispositivo único (e.g. sensor, actuador, sistema de aquisição de dados) ou de um dispositivo que poderá já agregar tráfego (e.g. painel de actuadores, bateria de sensores integrados).

Em função dos requisitos de QoS (QoS, *Quality of Service*) de cada serviço (i.e. porta de interface do Adaptador de terminal) será efectuada a agregação do tráfego de forma a optimizar a ocupação da largura de banda sem comprometer a garantia do atraso especificado para cada serviço.

A agregação de tráfego (multiplexagem) será suportada através do uso dos pacotes CPS (CPS, *Convergence Packet Sublayer*) do protocolo AAL-2 (AAL-2, *ATM Adaptation Layer – Type 2*) e de um mecanismo de escalonamento de tráfego, baseado num Algoritmo especificado e analisado no capítulo 4.

O endereçamento das fontes de tráfego poderá ser efectuada directamente através do campo CID (CID, *Channel Identifier*) do cabeçalho dos pacotes CPS ou, no caso da capacidade fornecida por este não ser suficiente, recorrer ao mecanismo fornecido pela camada ATM, através dos canais virtuais (VCI, *Virtual Channel Identifier*) e dos caminhos virtuais (VPI, *Virtual Channel Identifier*). Apesar desta possibilidade, a especificação do Adaptador de terminal efectuada para o estudo da adaptação de tráfego de baixo débito em aplicações de controlo, apenas prevê o endereçamento baseado no campo CID, uma vez que a complexidade acrescida, pela introdução deste mecanismo, não afecta os pressupostos que estiveram na base do desenvolvimento dos algoritmos de escalonamento.

O Concentrador permite não só agregar o tráfego proveniente dos vários Adaptadores de terminal como também efectuar uma espécie de multiplexagem de segunda ordem. O seu objectivo principal é o de aumentar a eficiência de transmissão da rede e, ao mesmo tempo,

implementar um mecanismo de prioridade de forma a permitir escalonar tráfego com diferentes requisitos de atraso.

O Concentrador utiliza igualmente os pacotes CPS como estrutura elementar de multiplexagem. Deste modo, este módulo efectua o desencapsulamento dos pacotes recebidos de cada Adaptador de terminal e, em seguida, efectua um novo escalonamento de forma a produzir um fluxo único, em que os pacotes com requisitos de atraso mais apertados são atendidos com maior prioridade. O algortimo de escalonamento desenvolvido para o Concentrador será igualmente especificado e analisado no capítulo 4.

De referir que este módulo não é necessário numa estrutura com apenas um Adaptador de terminal. Neste caso, a Aplicação de controlo e o Adaptador de terminal têm apenas a rede como elemento de interligação.

A Aplicação de controlo é o módulo que permite gerir o processo de comunicação entre todos os elementos integrantes do sistema de aquisição de dados e controlo. Consiste numa estação de trabalho, ligada à rede de comunicação, em que vários programas efectuam um conjunto de tarefas específicas. Toda a informação proveniente dos dispositivos interligados é recolhida e processada. Em função da especificidade da aplicação, poderá ser necessário enviar informação, em sentido inverso, para outros dispositivos igualmente ligados a Adaptadores de terminal. Embora exista um número elevado de situações possíveis para cenários deste tipo de aplicações, o que importa neste momento assegurar é a capacidade da solução proposta garantir a adequada transferência de informação, não apenas entre os elementos integrantes do sistema de aquisição de dados e controlo, mas também a de outros tipos de serviços, suportados numa rede de forma integrada e com garantia de QoS.

3.2 Adaptador de Terminal

No contexto da RDIS (RDIS, Rede Digital com Integração de Serviços), um Adaptador de terminal é um dispositivo que permite que equipamentos não RDIS possam operar nestas redes. Nos modelos propostos, a especificação do Adaptador de terminal pode ser simplificada, tendo em consideração apenas as funções necessárias para adaptar os fluxos de baixo débito às características das interfaces da rede de comunicação, maximizando a eficiência da rede e garantindo a QoS (atraso) requerida pelo serviço. No entanto, terão igualmente de ser referidas as restantes funcionalidades de gestão e sinalização necessárias ao interfuncionamento com equipamentos normalizados.

O Adaptador de terminal pode ser considerado como um conjunto de máquinas de estado de emissão e recepção que funcionam duma forma independente. Na emissão é necessário multiplexar o tráfego proveniente de várias fontes (e.g sensores), com diversos requisitos temporais, num único fluxo, garantindo por um lado a QoS exigida por cada terminal e, por outro, maximizando a eficiência de transmissão. Na recepção é necessário efectuar a desmultiplexagem dos canais, suportados pelo fluxo de informação destinado aos actuadores integrantes do sistema a controlar.

A solução proposta para o Adaptador de terminal, sendo baseada na tecnologia ATM, para além de estabelecer a interface de rede com os sensores/actuadores, permite resolver os problemas de identificação destes dispositivos pela aplicação de controlo, através da identificação dos circuitos e dos trajectos virtuais (VCI/VPI), utilizados nessa interface, bem como com a estrutura de multiplexagem da camada de adaptação (CID / AAL2). A Figura 3-1 ilustra a estrutura funcional em camadas do Adaptador de terminal.

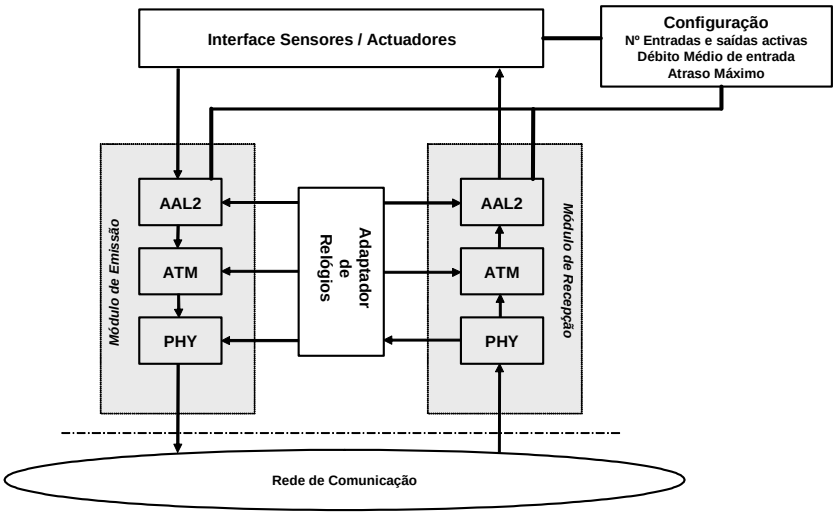


Figura 3-1: Estrutura Funcional do Adaptador de terminal.

- **Módulo de Configuração**

O módulo de configuração permite ao utilizador efectuar a definição das características de tráfego e de QoS dos dispositivos a ligar ao Adaptador de terminal. Os parâmetros de entrada são os seguintes:

- Número de canais de entrada e saída activos,
- Débito médio de entrada / saída,
- Atraso Máximo.

O primeiro indica, de todos os canais disponíveis, quantos vão estar activos. O segundo especifica qual o débito médio; no caso de o canal ser de débito constante, este valor é igual ao valor médio. Finalmente, o terceiro especifica o atraso máximo (quando requerido) de cada canal.

3.2.1 Módulo de Emissão

O módulo de emissão implementa as funções de multiplexagem necessárias à agregação dos diversos fluxos de entrada num único fluxo (ATM-SDU, *ATM-Service Data Unit*). O escalonamento dos fluxos de entrada é fundamental para se respeitarem os atrasos máximos e assim garantir a QoS requerida. Este escalonamento é matéria de discussão do capítulo 4, onde serão abordadas as questões do dimensionamento dos vários módulos funcionais do sistema.

As funções de multiplexagem usam como suporte os pacotes CPS do protocolo AAL-2. Deste modo, a subcamada SSCS (SSCS, *Service Specific Convergence Sublayer*) deste protocolo é suprimida. A Figura 3-2 ilustra o diagrama funcional do Módulo de Emissão do Adaptador de terminal.

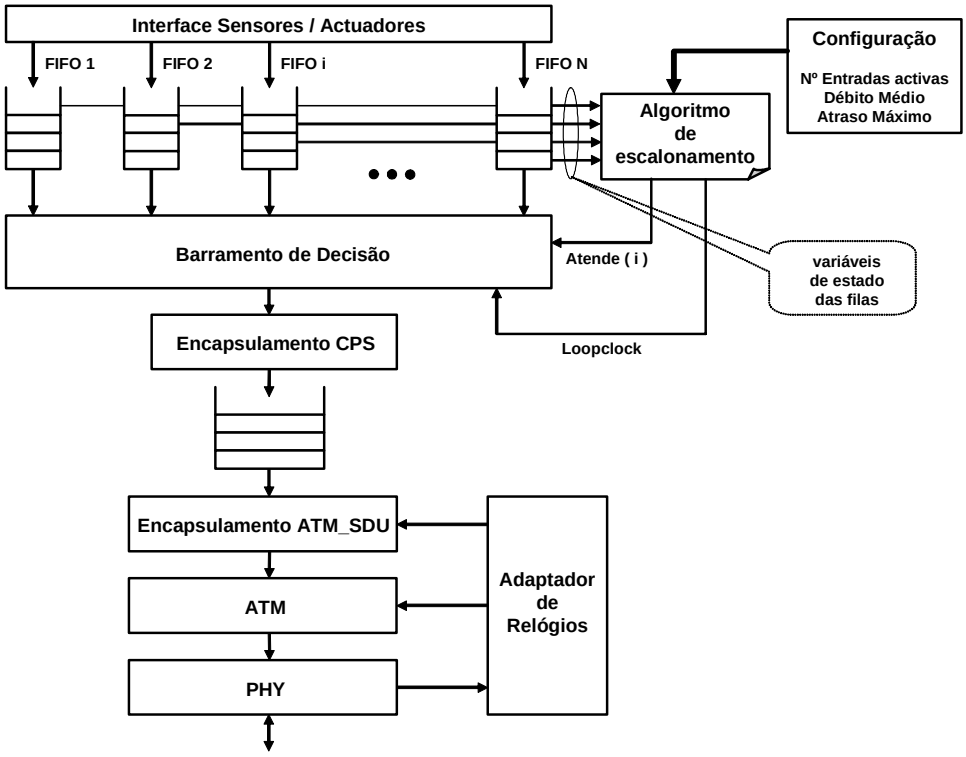


Figura 3-2: Estrutura Funcional do Adaptador de terminal (Módulo de Emissão).

- **Algoritmo de escalonamento**

Este é o módulo fundamental do Adaptador de terminal. As entradas são os parâmetros fornecidos pelo módulo de configuração e a indicação das variáveis de estado das filas de espera associadas a cada canal de entrada. Em função destes parâmetros, este módulo "decide" qual a fila (FIFO, *First In First Out*) a atender e a velocidade de atendimento (*LoopClock*). A descrição do Algoritmo de escalonamento, o seu funcionamento e os testes efectuados ao seu desempenho estão descritos com detalhe no capítulo 4 desta tese.

- **Encapsulamento CPS**

Este módulo efectua a construção dos pacotes CPS em função do FIFO atendido. Ao receber do Algoritmo de escalonamento a indicação de qual o FIFO a atender, é criado um pacote CPS com um comprimento igual ao número de octetos que estiver no FIFO. No caso de haver mais octetos no FIFO do que o comprimento máximo definido para os pacotes CPS, o pacote é construído com um comprimento igual a este valor. Os octetos restantes farão parte do pacote seguinte correspondente a este canal.

Os detalhes relacionados com o formato de dados deste protocolo são descritos na Recomendação I.363.2 do ITU-T [ITU97b]. No entanto, a Figura 3-3 mostra, de uma outra forma, a estrutura de um pacote CPS.

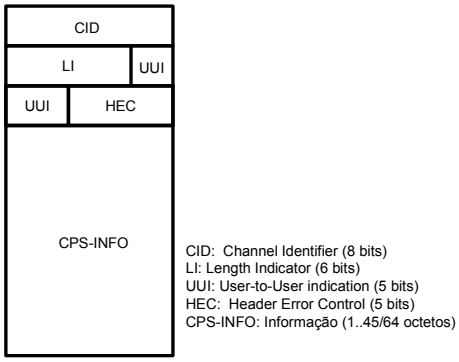


Figura 3-3: Estrutura de um pacote CPS.

O Campo CID identifica o canal utilizado. Os canais são numerados de 8 a 255, uma vez que a recomendação reserva os valores entre 0 e 7 para outras funções.

O campo LI indica o comprimento do campo de informação (CPS-INFO) em octetos. Para cada canal, o valor de LI indica quantos octetos foram lidos do FIFO da entrada correspondente. Este

valor pode variar, para cada canal, em função do número de octetos disponíveis no FIFO e em cada processo de leitura.

Como é referido na recomendação I.363.2 [ITU97b], o campo UUI pode implementar duas funções: transportar informação específica, de forma transparente, através da subcamada CPS e distinguir entre uma entidade SSCS (caso os valores estejam entre 0 e 27) e a camada de gestão. No contexto deste trabalho foi usada a primeira opção, onde o campo UUI serve para endereçar o Adaptador de terminal e implementar um mecanismo de identificação da classe de tráfego, associada ao fluxo transportado pelo pacote CPS, de acordo com o especificado na secção 2.1.3. A Figura 3-4 ilustra este mecanismo.

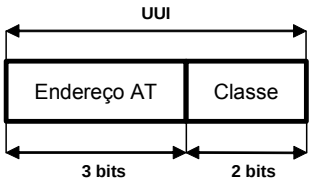


Figura 3-4: Estrutura do campo UUI dos pacotes CPS.

Os primeiros 3 bits permitem endereçar 8 Adaptadores de terminal, enquanto que os dois últimos identificam a classe de tráfego associada ao fluxo transportado pelo pacote CPS. No caso da aplicação necessitar de uma estrutura com mais Adaptadores de terminal e menos fontes de entrada / saída, podem ser usados bits do campo CID para aumentar a capacidade de endereçamento e assim suportar uma arquitectura com mais Adaptadores de terminal.

O campo HEC destina-se a proteger contra erros a totalidade dos campos de controlo do pacote CPS, ou seja, os campos CID, LI, UUI e o próprio HEC. É usado o código cíclico cujo polinómio gerador é p^5+p^2+1. Este campo é calculado através da divisão (módulo 2) pelo polinómio gerador, do produto de p^5 pelo polinómio construído através dos coeficientes dos primeiros 19 bits do cabeçalho do pacote CPS. Os coeficientes do resto da divisão são inseridos no campo HEC, em que o coeficiente do termo p^4 é do bit mais significativo.

Os pacotes CPS, à medida que vão sendo criados, são colocados num FIFO intermédio. Neste FIFO já foi feito o escalonamento dos fluxos de informação de entrada. Estes pacotes têm já a identificação do Canal (CID) a que correspondem, a identificação do Adaptador de terminal a que estão ligados (UUI) e da prioridade, associada à classe de tráfego que lhes é atribuída à entrada (UUI). Os pacotes CPS serão agora segmentados / agrupados em blocos de 48 octetos (ATM_SDUs) de forma a serem encapsulados em células ATM.

- **Encapsulamento ATM_SDU**

O formato dos pacotes ATM_SDU é recordado novamente na Figura 3-5

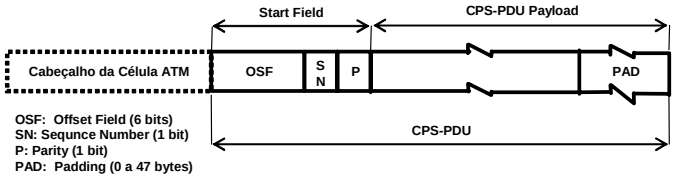


Figura 3-5: Formato dos pacotes ATM_SDU.

Esta estrutura de multiplexagem destina-se a juntar os diversos pacotes CPS, de diversos tamanhos, num único pacote (ATM_SDU). Estes pacotes necessitam de um temporizador de forma a organizar o seu escalonamento para transmissão. A natureza do tráfego de entrada pode ter exigências temporais restritas, o que faz com que um pacote não possa esperar mais do que um valor temporal bem definido. Assim, se o valor deste temporizador for baixo as células ATM poderão ficar parcialmente preenchidas, uma vez que o débito de chegada de pacotes CPS é baixo relativamente ao relógio de multiplexagem. No caso contrário, as células serão totalmente preenchidas mas o atraso de empacotamento aumentará.

Existe um relógio de uso combinado Timer_CU (CU, *Combined Use*), definido na recomendação I363.2 [ITU97b], que assegura que os pacotes CPS esperem no máximo o tempo indicado pelo Timer_CU antes de serem escalonados para transmissão. O bom dimensionamento deste temporizador permitirá obter uma taxa de ocupação de célula máxima sem degradar significativamente o atraso global. Este relógio é obtido através do relógio derivado da rede.

A seguir descreve-se o procedimento simplificado da criação dos ATM_SDUs, de acordo com os diagramas SDL (SDL, *Specification and Description Language*) da recomendação I363.2:

1. Espera a chegada de um pacote CPS à fila de pacotes CPS. Quando chega um pacote inicializa o Timer_CU,
2. Constrói novo pacote ATM_SDU. A seguir ao campo STF introduz os octetos, por ordem de saída da fila de pacotes CPS,
3. Se o ATM_SDU tiver sido preenchido antes do Timer_CU expirar e um pacote CPS não tiver sido totalmente colocado dentro do ATM_SDU, memoriza o número de octetos que sobraram para introduzir esse valor no campo OSF do próximo ATM_SDU. Reinicializa o Timer_CU. Salta para o ponto 2,

4. Se não existirem mais pacotes CPS na fila e o Timer_CU expirar, preenche os restantes octetos do ATM_SDU com *padding*. Salta para o ponto 1,

5. Se o Timer_CU expirar e o ATM_SDU não incluir nenhum início de pacote CPS, calcula o valor do OSF em função do início do *padding*. Salta para o ponto 1

3.2.2 Módulo de Recepção

O módulo de recepção do Adaptador de terminal é bastante mais simples uma vez que o escalonamento temporal dos diversos fluxos de entrada é efectuado em sentido inverso. Deste modo, a desmultiplexagem é efectuada directamente à medida que as células ATM vão sendo recebidas. No entanto, é necessário implementar um mecanismo que evite que um pacote ATM_SDU com erro não seja descartado completamente. Com efeito, é necessário efectuar primeiro a desmultiplexagem e analisar os pacotes CPS individualmente no sentido de verificar quais são os que verificam o campo HEC. A Figura 3-6 apresenta o diagrama funcional do módulo de recepção do Adaptador de terminal.

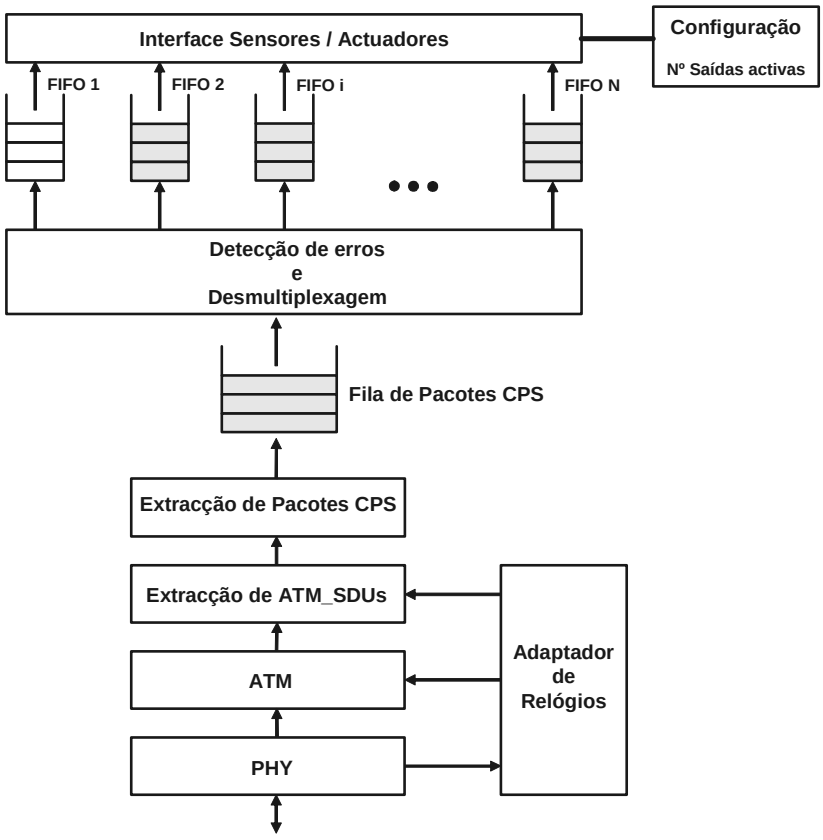


Figura 3-6: Estrutura Funcional do Adaptador de terminal (Módulo de Recepção).

- **Extração de ATM_SDUs / Extração de pacotes CPS**

Basicamente, este bloco retira o campo de informação de uma célula ATM, armazenando-o num registo, para ser processado pelo módulo acima. Para a detecção de erros é usado o cabeçalho do ATM_SDU ilustrado na Figura 3-7.

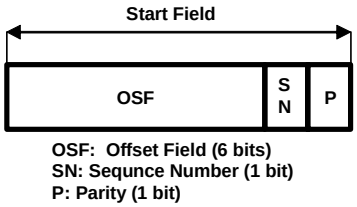


Figura 3-7: Estrutura do cabeçalho de um ATM_SDU (STF, *Start Field*).

O cabeçalho é composto por 3 campos. A seguir é recordado o significado de cada um deles, de acordo com a recomendação I363.2 [ITU97b]:

OSF: Representa o valor binário do *offset* (medido em octetos) desde o fim do STF até ao início de um pacote CPS ou, na ausência deste, ao início do campo PAD. O valor 47 indica que não existe nenhum início de pacote CPS dentro do ATM_SDU.

SN: Implementa um mecanismo de numeração (módulo 2) da sequência de ATM_SDUs.

P: Bit de paridade. Para detectar erros no STF verifica se a paridade deste é ímpar.

Seguidamente, é descrito o procedimento simplificado para a extracção dos pacotes CPS do fluxo de células ATM (ATM_SDUs), de acordo com os diagramas SDL da recomendação I363.2 [ITU97b]:

1. Espera a chegada de um ATM_SDU,
2. Na chegada de um pacote ATM_SDU é extraído o cabeçalho. Guarda os valores dos campos OSF, SN e P,
3. Se P e SN não forem verificados todo o pacote é descartado assim como os octetos pendentes do pacote CPS anterior recebido,
4. Se o valor do *offset* for diferente de 0 agrupa os octetos até ao valor do *offset*, aos octetos do pacote CPS pendente. Coloca o pacote CPS na Fila de pacotes CPS,
5. Se ainda sobrarem octetos, é verificado o valor do CID. Se for 0, então os octetos correspondem a *padding* e são descartados. O algoritmo salta para o ponto 1. Se não existir mais informação, salta igualmente para o ponto 1,

6. Se o número de octetos restantes for maior que o valor de LI são guardados os octetos do pacote CPS até ao valor de LI. Coloca o pacote CPS na Fila de pacotes CPS. Salta para o ponto 5,
7. Se o número de octetos restantes for inferior a LI significa que o resto do pacote CPS acaba no ATM_SDU seguinte. Guarda os octetos restantes e salta para o ponto 1.

- **Detecção de erros e Desmultiplexagem**

Neste módulo é verificada a existência de erros no cabeçalho dos pacotes CPS e efectuada a desmultiplexagem dos vários canais.

Para efectuar a detecção de erros são seguidos os procedimentos normais associados aos códigos cíclicos [Car86]. Neste caso estamos em presença de um código Hamming do tipo:

$$(n,k) = (24,19)$$

Significa que, para cada 19 bits de informação, temos 24 bits de código, ou seja:

$$q = n - k = 5$$

em que *q* representa o número de bits de paridade. De acordo com [Car86], este código consegue corrigir erros simples e detectar alguns erros duplos. Para efectuar a correcção de erros é necessário efectuar o cálculo do Síndroma:

$$S(p) = \text{rem}\,[Y(p) / G(p)]$$

S(p) representa o polinómio do Síndroma, Y(p) o polinómio da palavra recebida (cabeçalho do CPS) e G(p) o polinómio gerador do código (p^5+p^2+1). O operador "rem" corresponde ao resto da divisão (módulo 2). A Figura 3-8 ilustra o diagrama funcional do processo de controlo de erros do cabeçalho dos pacotes CPS.

Os pacotes que vão sendo processados são aceites ou rejeitados em função do cálculo do Síndroma. Se o número de erros for igual ou inferior a 1 os pacotes são aceites e, no caso contrário, serão rejeitados.

Após o controlo de erros do cabeçalho dos pacotes CPS, são extraídos os campos CID, UUI e LI. Para cada pacote CPS recebido são escritos LI octetos na fila correspondente ao canal indicado pelo campo CID.

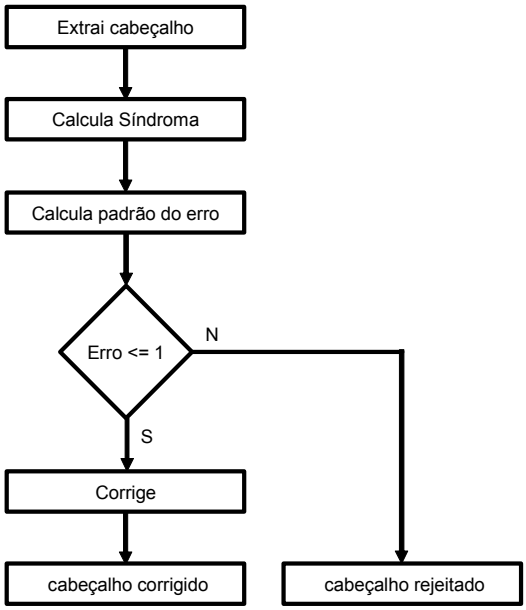


Figura 3-8: Controlo de erros no cabeçalho dos pacotes CPS.

3.3 Concentrador

O Concentrador, à semelhança do Adaptador de terminal, pode ser considerado como um conjunto de máquinas de estado de concentração e expansão que funcionam de uma forma independente. O módulo de concentração efectua a multiplexagem dos pacotes CPS provenientes de várias fontes e de vários Adaptadores de terminais num fluxo agregado, maximizando a eficiência de transmissão. O módulo de expansão efectua a desmultiplexagem dos pacotes CPS, em fluxos separados de acordo com o Adaptador de terminal de destino.

O módulo de concentração necessita de recuperar a estrutura dos pacotes CPS dos vários fluxos de entrada para efectuar nova multiplexagem, optimizando assim a taxa de preenchimento das células ATM e o consequente aumento da eficiência de transmissão na rede. À semelhança do Adaptador de terminal, a multiplexagem dos vários fluxos, provenientes das várias portas de entrada, é efectuada através de escalonamento temporal baseado no atraso requerido pelos serviços associados a cada fluxo.

As funcionalidades de gestão e sinalização, necessárias ao funcionamento em ambientes com equipamentos normalizados, constituindo assuntos laterais ao tema deste trabalho, deverão ser igualmente incluídas no Concentrador. A Figura 3-9 ilustra o princípio de funcionamento do Concentrador.

A principal vantagem do uso do Concentrador é o de aumentar a eficiência de transmissão na rede, principalmente quando um ou mais Adaptadores de terminal funcionam com uma reduzida ocupação da largura de banda. Este equipamento, efectua o desencapsulamento dos pacotes CPS, provenientes das células ATM das várias portas de entrada (Adaptadores de terminal), para depois efectuar novo encapsulamento destinado à porta de saída. No sentido contrário é feita a operação inversa.

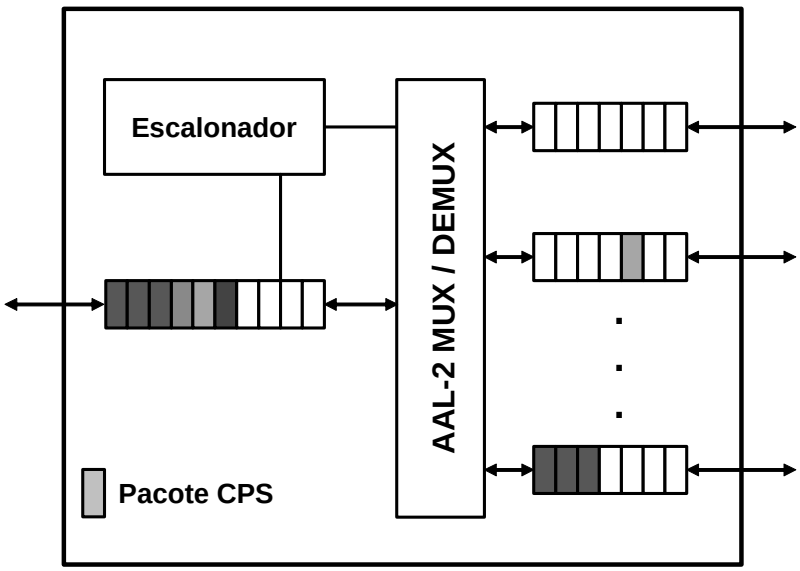


Figura 3-9: Princípio de funcionamento do Concentrador.

Como foi referido anteriormente, a multiplexagem é baseada em escalonamento temporal, em que o atendimento das filas, correspondentes aos canais de entrada, é controlado pelo Algoritmo de escalonamento do Concentrador.

Se as linhas de entrada tiverem todas o mesmo débito:

$$f_1 = f_2 = ... = f_n = f$$

o débito máximo de saída terá de ser:

$$f_s = n \times f$$

No entanto, como se trata de um concentrador, o débito de saída será bastante mais reduzido, uma vez que a taxa de ocupação de células, proveniente dos Adaptadores de terminal, será previsivelmente inferior a 100%. Quanto menor for o débito de saída maior será a eficiência ou o ganho na multiplexagem. Por outro lado, quanto maior for o débito de saída menor será o

atraso temporal introduzido pelo Concentrador. É através destes pressupostos que será dimensionado o Algoritmo de escalonamento do Concentrador.

A escolha da capacidade de transmissão será um dos recursos a negociar com a rede, juntamente com outros parâmetros de QoS (e.g. Categoria de Serviço e parâmetros de tráfego). Deste modo, a eficiência de transmissão é também dependente da capacidade de transmissão atribuída à ligação.

3.3.1 Módulo de concentração

É o módulo de concentração que efectua as funções mais importantes – aumentar a eficiência de multiplexagem, minimizando o atraso dos serviços que possuam requisitos temporais apertados. Para tal, é necessário recuperar a estrutura dos pacotes CPS a partir das células ATM. Deste modo, a técnica de multiplexagem usada é idêntica à efectuada no Adaptador de terminal assim como grande parte das funcionalidades, conseguindo-se economizar no desenvolvimento da maioria dos módulos. A estrutura funcional do Módulo de concentração está ilustrada na Figura 3-10.

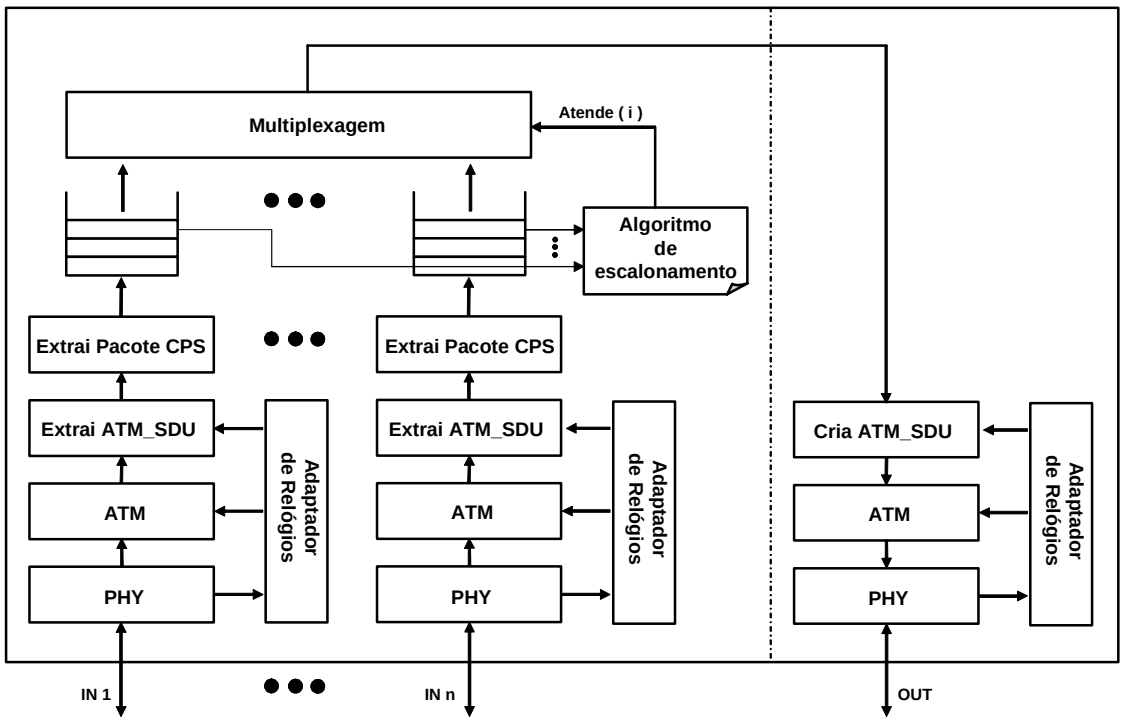


Figura 3-10: Estrutura funcional do Concentrador – módulo de concentração.

Todos os blocos foram já descritos com detalhe nas secções 3.2.1 e 3.2.2. De referir que não é efectuado o controlo de erros ao nível do cabeçalho dos pacotes CPS, uma vez que se supõe que as distâncias entre o Concentrador e os Adaptadores de terminal são curtas e que as linhas de transmissão apresentam taxas de erro baixas.

Existem *n* portas de entrada a concorrer para uma porta de saída. Deste modo, a capacidade da porta de saída deve ser calculada de forma a ser ligeiramente superior à soma dos débitos médios úteis dos pacotes CPS das portas de entrada.

3.3.2 Módulo de expansão

O módulo de expansão efectua a função inversa do módulo de concentração. Extrai os pacotes CPS das células ATM para os encaminhar, em vários fluxos separados de células ATM, para os Adaptadores de terminal de destino. A Figura 3-11 ilustra o diagrama funcional do módulo de expansão do Concentrador.

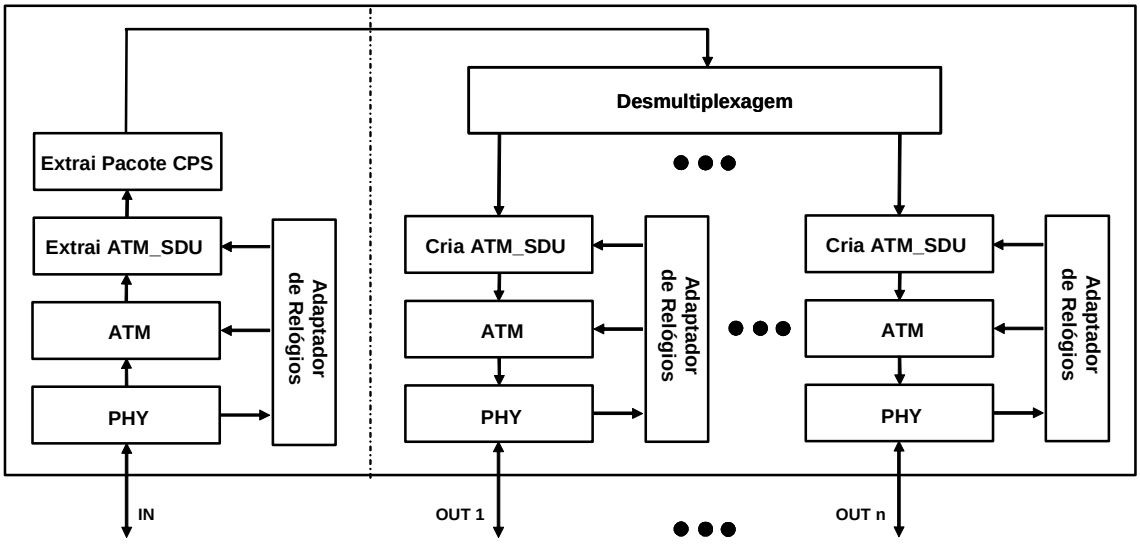


Figura 3-11: Estrutura funcional do Concentrador – módulo de expansão.

À semelhança do módulo anterior, deve também salientar-se que os módulos desenvolvidos são idênticos aos desenvolvidos para o Adaptador de terminal, sendo praticamente aproveitados na íntegra.

A informação relativa ao endereço do Adaptador de terminal de destino é veiculada no campo UUI, da mesma forma que é efectuada no sentido inverso e descrita na secção 3.2.1.

3.4 Aplicação de Controlo

Para controlar eficazmente a transferência da informação entre os vários elementos integrantes do sistema de aquisição de dados e controlo é indispensável desenvolver, para cada implementação específica, uma aplicação que, numa primeira abordagem e duma forma resumida, proceda à identificação da fonte e ao encaminhamento da informação para o seu destino.

A identificação de cada elemento do sistema, sensor, actuador ou adaptador de terminal, será efectuada, recorrendo quer aos identificadores de caminho e de canal virtual (VPI/VCI) da camada ATM, quer aos identificadores dos canais multiplexados na camada de Adaptação, de acordo com as especificações do protocolo AAL-2.

Especificação de Aplicações de controlo

Embora não seja o objectivo deste trabalho especificar os requisitos para o desenvolvimento da aplicação de controlo, é possível inventariar um conjunto de metodologias genéricas que poderão servir de base ao seu projecto e especificação.

- **Modelos de transferência de Informação**

As aplicações de controlo, que envolvem dispositivos de baixo débito, podem ser classificadas em relação à forma como é efectuada a transferência de informação, entre o fenómeno e o observador. Estas podem ser classificadas como: contínua, orientada ao evento, iniciada pelo observador e híbrida. No modelo contínuo, os sensores transmitem a informação continuamente com um débito pré-determinado. No modelo orientado a eventos os sensores enviam informação apenas quando ocorre um evento de interesse. Neste caso, o observador está interessado na ocorrência de um fenómeno específico ou num conjunto de fenómenos. No modelo iniciado pelo observador os sensores só enviam informação em resposta a um pedido explícito do observador. Finalmente, no modelo híbrido as três estratégias coexistem na mesma aplicação.

Por exemplo, numa aplicação construída para a detecção de intrusos, o modelo de transferência de informação é orientado ao evento, que corresponde à entrada de um intruso na área monitorada. Se o modelo de encaminhamento da informação na rede é baseado em *flooding*, os sensores que estiverem fisicamente próximos detectarão o intruso ao mesmo tempo e vão enviar, simultaneamente, informação para o observador. Estas comunicações

concorrentes na vizinhança de um fenómeno podem gerar contenção no meio de comunicação, aumentando a probabilidade de perda de informação crítica e a latência em reportar o evento.

- **Modelos dinâmicos de rede**

Uma rede de sensores forma um caminho entre o fenómeno e o observador. O objectivo do protocolo na rede de sensores é criar e manter este caminho ou múltiplos caminhos sob condições dinâmicas. Sem perda de generalidade esta discussão assume um único observador. Múltiplos observadores podem ser suportados com instâncias múltiplas de um único observador.

- **Tolerância a falhas**

Os sensores podem falhar devido a más condições ambientais ou por falta de energia. Uma vez que nem sempre é possível a troca dos dispositivos em tempo útil, a aplicação deve ser tolerante a falhas, ou seja, é desejável que as falhas não catastróficas sejam transparentes para a aplicação.

A aplicação de controlo executará igualmente as funções de controlo de tráfego, de acordo com os requisitos de cada aplicação concreta, bem como as funções de sinalização e de gestão, indispensáveis à interligação de equipamentos normalizados.

4

Análise de desempenho do sistema

Para avaliar as qualidades e limitações dos modelos propostos, foram realizadas simulações com tráfego característico das redes industriais, onde os sistemas de controlo se inserem, bem como com outro tipo de tráfego típico das redes de comunicação usadas para interligar os módulos propostos.

O estudo apresentado no capítulo 2 acerca da adaptação de tráfego de baixo débito em redes de comunicação conduziu à especificação de uma arquitectura composta por dois módulos fundamentais para o suporte de aplicações de controlo. Esses módulos, o Adaptador de terminal e o Concentrador, apresentados com detalhe no capítulo 3, visam resolver dois problemas fundamentais:

- Integrar o maior número de dispositivos e sistemas,
- Garantir uma alta eficiência na agregação dos diversos fluxos de informação garantindo a Qualidade de Serviço (QoS, *Quality of Service*) negociada.

O primeiro problema foi resolvido com a escolha da tecnologia ATM (ATM, *Asynchronous Transfer Mode*) como suporte de infra-estrutura de rede e da escolha do protocolo AAL-2 (AAL-2, *ATM Adaptation Layer – type 2*) para o suporte da estrutura de multiplexagem das diversas fontes de tráfego.

Para a resolução do segundo problema será necessário o estudo de algoritmos de escalonamento de tráfego de modo a definir mecanismos que garantam o atendimento prioritário a fluxos que possuam requisitos de atraso mais apertados. O escalonamento de tráfego é feito em duas fases. A primeira é efectuada no Adaptador de terminal onde os diversos fluxos concorrem para uma dada capacidade de transmissão. Nesta fase, é necessário implementar um algoritmo que garanta a satisfação dos requisitos de atraso dos diversos fluxos que se apresentam à entrada maximizando, ao mesmo tempo, a ocupação da largura de banda disponível. A segunda fase é efectuada no Concentrador, onde será necessário implementar um mecanismo de escalonamento que permita optimizar a agregação de tráfego proveniente de diversos Adaptadores de terminal, garantindo também a satisfação dos requisitos temporais impostos pelos diversos fluxos individuais.

No sentido de avaliar o desempenho do sistema, será necessário implementar mecanismos que permitam testar o comportamento dos vários algoritmos de escalonamento propostos. Para tal será necessário desenvolver um sistema que permita testar e avaliar, duma forma quantitativa, as diversas soluções apresentadas. O sistema de testes apresentado neste capítulo consiste num conjunto de modelos simplificados dos módulos apresentados no capítulo 3 mas que, no entanto, possuem as características necessárias à obtenção dos resultados essenciais para a avaliação dos diversos algoritmos.

Para a avaliação dos algoritmos, é definida uma configuração de referência que permite identificar as diversas componentes que contribuem para o atraso total da informação ao longo dos vários troços do sistema. Deste modo, será possível atribuir a cada módulo uma parcela desse atraso que permitirá avaliar individualmente o desempenho dos algoritmos de escalonamento do Adaptador de terminal e do Concentrador.

Os testes efectuados ao Adaptador de terminal caracterizaram-se pela análise da resposta dos vários módulos, em que são usados diversos cenários de tráfego para carregar o sistema. Concretamente, são usados dois cenários: o primeiro permite avaliar a capacidade de integração de fluxos de informação associados a serviços com requisitos de QoS diferentes e o segundo avalia a capacidade do sistema suportar um elevado número de fluxos de baixo débito duma forma eficiente.

Para a avaliação do Concentrador, utilizou-se um cenário em que são gerados pacotes CPS de tamanho variável, em cada porta de entrada, e são medidos os atrasos máximos dos fluxos das diversas classes assim como os tamanhos das filas correspondentes. Estas classes de tráfego, definidas no capítulo 2, são recordadas a seguir:

- Atraso Máximo (AM),

- Dados (D),
- Esforço Mínimo (EM).

A descrição das fontes geradoras de tráfego e a sua modelização é efectuada na secção 4.3, assim como a descrição das funções geradoras de números aleatórios. Seguidamente, nas secções 4.4 e 4.5 são apresentados e analisados os resultados da avaliação efectuada aos Algoritmos de escalonamento do Adaptador de terminal e do Concentrador.

4.1 Caracterização do sistema de testes

Tendo em vista a avaliação de desempenho do sistema proposto para adaptação de tráfego de baixo débito foram implementados, com as convenientes simplificações, os módulos do Adaptador de terminal e do Concentrador, apresentados no capítulo anterior, bem como modelos de simulação das fontes geradoras de tráfego.

De forma a poder optimizar o trabalho de especificação, simulação e desenvolvimento, optou-se por usar como suporte um PC (PC, *Personal Computer*) baseado numa arquitectura com processador Pentium. A especificação dos diversos módulos do sistema de testes foi efectuada em linguagem C sobre o sistema operativo Linux. Em Anexo poderão ser encontradas as listagens dos programas desenvolvidos. Em relação ao protocolo AAL-2, toda a programação foi efectuada respeitando o mais possível o algoritmo real, ou seja, o que é apresentado na recomendação I363.2 do ITU-T [ITU97b], seguindo de perto os diagramas SDL (SDL, *Specification and Description Language*) da própria recomendação.

Os módulos desenvolvidos para o sistema de testes poderão ser reutilizados na fase de implementação sem necessidade de efectuar grandes modificações. Seguidamente, é feita a descrição dos sistemas de testes para avaliação do Adaptador de terminal e do Concentrador.

4.1.1 Sistema de testes para avaliação do Adaptador de terminal

Como foi referido anteriormente, o Adaptador de terminal destina-se a efectuar o interface entre os dispositivos, envolvidos numa Aplicação de controlo, e a rede de suporte. Para avaliar o seu desempenho especificou-se um sistema simplificado, do ponto de vista de uma possível

implementação final, que permitirá quantificar os parâmetros principais relacionados com o seu comportamento perante diversos tipos de tráfego. A Figura 4-1 mostra a sua estrutura funcional.

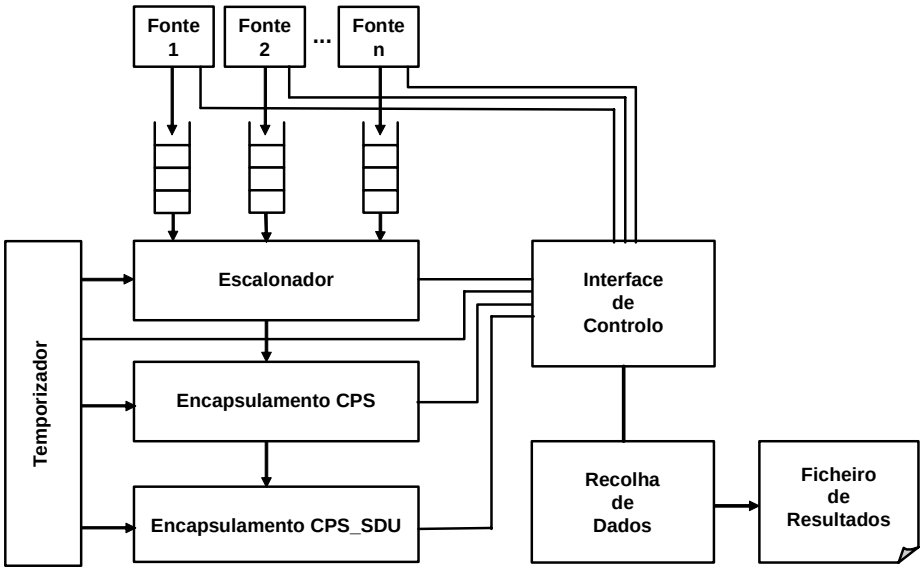


Figura 4-1: Estrutura do sistema de testes para avaliação do Adaptador de terminal.

O sistema, implementado por um conjunto de programas em C, é constituído por diversos componentes que permitem efectuar a simulação do comportamento dos módulos de emissão do Adaptador de terminal relacionados com o escalonamento de informação. Em seguida serão descritos cada um dos blocos funcionais:

- **Interface de controlo:**

Este módulo permite efectuar o controlo de toda a simulação. Efectua a gestão dos diversos módulos sincronizado pelo Temporizador. É neste módulo que se define o tempo de simulação.

- **Fontes de informação:**

Para cada fonte de informação é necessário definir as características de cada fluxo. Existe para cada fonte um módulo em que são especificados os seguintes parâmetros:

- Débito - Constante ou variável,
- Valor médio do débito,
- Classe de tráfego (de acordo com o especificado na secção 2.1.3),
- Distribuição estatística usada – no caso do fluxo não ser de débito constante.

Quando se inicia a simulação, a informação gerada por cada fonte é armazenada num FIFO (FIFO, *First In First Out*), implementado no sistema operativo através de um *named pipe* que permite a comunicação entre o processo associado à fonte e o processo do escalonador.

Mais à frente, na secção 4.3, serão descritas, com mais detalhe, as características principais das fontes de tráfego usadas assim como os modelos escolhidos para a sua representação.

- **Escalonador:**

Este módulo é responsável pelo escalonamento de informação de acordo com o algoritmo usado. Para efectuar o escalonamento da informação, ou seja, decidir qual a fila a atender, cada algoritmo tem acesso aos seguintes elementos:

- o instante de chegada dos octetos “à cabeça” de cada FIFO,
- o tamanho actual de cada FIFO,
- a classe de tráfego.

Em função dos elementos anteriores, o algoritmo de escalonamento irá escolher a fila a atender. Uma vez escolhida a fila a atender o escalonador irá ler, do FIFO correspondente, o maior número de octetos possível de modo a preencher um pacote CPS (CPS, *Convergence Packet Sublayer*). Nesta fase podem acontecer três situações. A primeira corresponde ao número de octetos lido ser inferior ao valor máximo do tamanho de pacotes CPS. Neste caso o pacote terá um comprimento correspondente ao número de octetos lido. A segunda situação corresponde ao preenchimento do pacote CPS com um número de octetos igual ao valor máximo. Finalmente, a terceira situação ocorre quando, no FIFO, existem mais octetos do que aqueles que o escalonador consegue ler. Neste caso, os octetos que não foram lidos ficarão em espera até o algoritmo decidir atender novamente essa fila.

A leitura dos octetos dos FIFOs é sincronizada por um relógio designado por *Loopclock*, que é derivado do relógio do sistema operativo e fornecido pelo bloco Temporizador. A frequência do *Loopclock* é um parâmetro muito importante que afecta directamente a eficiência de multiplexagem e, ao mesmo tempo, garante o cumprimento dos atrasos máximos dos fluxos que atravessam o Adaptador de terminal.

Deste modo, o valor da frequência do *Loopclock* será um dos parâmetros a avaliar na simulação dos diversos algoritmos apresentados. Este valor será limitado inferiormente pelo princípio de que o débito da soma dos fluxos médios, que entram no Adaptador de terminal,

tem de ser inferior ao débito do fluxo agregado, uma vez que são adicionados cabeçalhos aos pacotes resultantes e o débito instantâneo pode ser superior ao débito médio (*burstiness*):

$$\sum_i \overline{f_i} < f_{Loopclock} \tag{4-1}$$

Por outro lado, se a frequência do relógio tiver um valor muito elevado, relativamente à soma do valor médio dos débitos de entrada, a eficiência de multiplexagem será baixa. Assim, a escolha da frequência do *Loopclock* será feita através de um compromisso entre estes dois factores.

- **Encapsulamento CPS:**

Os octetos resultantes da leitura do FIFO escolhido pelo algoritmo de escalonamento são passados para este bloco que vai adicionar o cabeçalho de acordo com o explicado na secção 3.2.1. São calculados os campos CID, LI, UUI e HEC, num total de 3 octetos. O pacote de informação é, em seguida, enviado para o bloco seguinte.

- **Encapsulamento CPS_SDU:**

Este módulo tem como função principal a segmentação da informação, recebida do bloco anterior, em blocos de 48 octetos de forma a poderem ser mapeados no campo de informação das células ATM. Este procedimento foi também já explicado na secção 3.2.1. São calculados os campos OSF, SN e P e é introduzido *padding* quando não houver informação útil para preencher a célula.

- **Temporizador:**

No sistema de simulação, todos os módulos são sincronizados pelo relógio do sistema operativo que é função do relógio do CPU. Num sistema real este seria o relógio de rede.

- **Recolha de dados:**

Neste módulo serão recolhidos todos os elementos necessários à avaliação do desempenho dos algoritmos a simular. Este módulo recolhe informação de todos os restantes módulos ao longo de toda a simulação. Os principais valores recolhidos são os seguintes:

- Atraso máximo de cada fluxo,
- Tamanho máximo de cada FIFO,
- Número de octetos gerado por cada fonte,
- Número de octetos gerados nos módulos de encapsulamento.

O primeiro parâmetro permite avaliar se o atraso que cada fluxo sofreu ao atravessar o Adaptador de terminal é inferior ao máximo especificado nos fluxos pertencentes às classes de Atraso Máximo. O atraso é calculado através da medição da diferença entre o registo do instante temporal de escrita de cada octeto no FIFO do módulo de encapsulamento CPS_SDU (ver Figura 4-1) e o registo do instante de escrita no FIFO de entrada. Para cada fila são mantidas variáveis que guardam os valores máximos associados.

O segundo parâmetro permite monitorar o tamanho de cada FIFO de modo a prevenir *overflows* e, num sistema real, permitirá dimensionar a capacidade máxima de cada FIFO. Finalmente, os dois últimos indicam a quantidade de informação que é gerada à entrada e à saída do Adaptador de terminal. A partir destes valores são calculados os seguintes parâmetros:

- Quantidade de pacotes CPS produzidos,
- Valor médio de octetos por CPS,
- Valor médio do *overhead* dos pacotes CPS.

Estes últimos parâmetros permitem avaliar a eficiência de multiplexagem. Os pacotes CPS têm tamanho variável, apesar do número de octetos do cabeçalho ser fixo (3 octetos). Deste modo, o *overhead* é função do tamanho do pacote CPS, de acordo com a relação 6-2:

$$Overhead = \frac{TamanhoCPS - NumeroOctetosInfo}{TamanhoCPS} \; x \; 100\% \qquad (4\text{-}2)$$

No 2.º termo da relação anterior, o valor do numerador é fixo – 3 octetos. A Figura 4-2 apresenta um gráfico da variação do *overhead* em função do tamanho do pacote.

O valor máximo do *overhead* é de 75%, correspondente a pacotes com 1 octeto de informação útil. Os valores mínimos são de 6,25% e 4,69%, respectivamente para pacotes de tamanho 48 e 64 octetos.

A recomendação I.363.2 do ITU-T [ITU97b] especifica um valor por omissão máximo de 45 octetos. Apesar da recomendação suportar o uso de pacotes com tamanho máximo de 64 octetos, este valor não foi usado na avaliação dos algoritmos uma vez que valores entre 45 e 64 não são permitidos e, desse modo, ser difícil uma análise uniforme de desempenho em função do tamanho do pacote CPS.

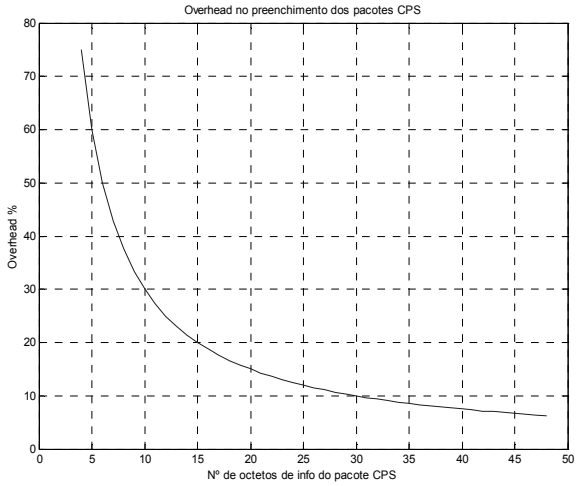


Figura 4-2: Variação do *overhead* em função do tamanho o pacote CPS.

- **Ficheiros de resultados**

Para mais fácil manipulação da informação recolhida na simulação e também para produzir os ficheiros de entrada que serão usados na avaliação do Concentrador, são criados dois ficheiros. Um dos ficheiros armazena todos os pacotes CPS_SDU gerados à saída do Adaptador de terminal. O segundo guarda todos os valores obtidos pelo módulo de Recolha de dados. Num sistema real, o primeiro ficheiro corresponde ao fluxo de dados enviado para preencher o campo de informação de uma célula ATM, ou seja, o seu tamanho será sempre um múltiplo de 48 octetos.

Cenários de Simulação

A avaliação dos algoritmos de escalonamento do Adaptador de terminal e do Concentrador terá de ser efectuada mediante a utilização de cenários de simulação que imponham situações limite, de forma a ser possível dimensionar os diversos parâmetros, associados a cada a algoritmo, e assim se conseguir definir uma solução genérica para a adaptação de tráfego de baixo débito em aplicações de controlo.

Como já foi várias vezes mencionado, os objectivos principais a atingir no escalonamento dos fluxos de informação são:

1. Permitir a integração de fontes de tráfego de vários tipos, garantindo os requisitos de atraso de cada fonte,
2. Maximizar a ocupação da largura de banda de forma a reduzir os custos de transmissão,
3. Integrar um número elevado de fontes de baixo débito.

Para tal, foram definidos dois cenários que permitirão avaliar a capacidade dos algoritmos de escalonamento atingirem os objectivos atrás mencionados. O primeiro permitirá avaliar a capacidade de cada algoritmo atingir os dois primeiros objectivos enquanto que o segundo permitirá avaliar a capacidade de se atingirem os dois últimos.

Cenário 1

Este cenário de simulação permite avaliar a capacidade dos algoritmos de escalonamento de suportarem a integração de fontes com requisitos de tráfego e atraso diferentes assim como a capacidade de optimizar a ocupação da largura de banda de transmissão.

Para tal, será necessário avaliar o comportamento dos algoritmos perante um cenário em que as fontes usadas possam produzir resultados em que seja possível retirar informação acerca da eficiência de multiplexagem, dos atrasos máximos dos diferentes fluxos, do tamanho das filas de espera associadas a cada fluxo e do valor do *Loopclock* máximo que satisfaz os requisitos de todos os serviços.

Deste modo, são usadas como entradas um conjunto de fontes com as características apresentadas na Tabela 4-1.

Tabela 4-1: Caracterização das fontes do cenário 1.

Fonte	Débito médio	Tipo	Peso (%)
Voz	16kbit/s	constante	14,17
Sensor	800bit/s	constante	0,71
Dados	16kbit/s	variável	14,17
Interruptores	80bit/s	aleatório	0,071
Vídeo	80kbit/s	variável	70,87
Total	**112 880bit/s**	---------	**100**

Às fontes do tipo Voz, Sensor e Interruptores será atribuída a classe do tipo AM, à fonte do tipo Dados a classe D e à fonte de Vídeo é atribuída a classe EF. Cada serviço da classe do tipo AM terá um atraso máximo associado em função dos requisitos definidos para cada serviço. Neste cenário, o serviço que impõe mais restrições, em termos de atraso, será o serviço de Voz. Em termos de débito, o serviço de Vídeo é aquele que carrega o sistema com a maior quantidade de informação.

A última coluna da Tabela 4-1 indica os pesos de cada serviço em função do débito médio total. Este valor dá uma ideia da percentagem de informação média que cada serviço produz relativamente à média total.

Como foi referido na secção 4.1.1, em cada intervalo de tempo dado pelo *Loopclock*, o escalonador tentará ler o maior número de octetos do FIFO escolhido pelo algoritmo de escalonamento. Com este cenário, o valor máximo do período do *Loopclock* será:

$$Loopclock_{max} = \frac{45 \times 8}{112880} = 3{,}19\text{ms} \qquad (4\text{-}3)$$

O valor indicado na equação 4-3 corresponde à situação em que o escalonador consegue encher pacotes CPS com 45 octetos e supondo que todos os fluxos de entrada têm débito constante (o que na prática não se verifica). Deste modo, o valor dado pela equação 4-3, permite indicar um valor máximo, abaixo do qual se deve procurar o valor adequado que satisfaça os objectivos pretendidos. Quanto mais baixo for o período do *Loopclock* mais baixa será a eficiência de multiplexagem, o que conduz à procura de um valor do *Loopclock* o mais próximo possível do indicado na equação 4-3.

Cenário 2

Com este cenário de simulação pretende-se avaliar a capacidade de suporte de um grande número de fontes de baixo débito assim como maximizar a eficiência de multiplexagem de forma a conseguir economizar a largura de banda de transmissão.

Deste modo, este cenário de simulação serve para analisar o comportamento do escalonador, à medida que se vão introduzindo cada vez mais fontes na entrada. Neste cenário de simulação serão usadas apenas fontes do tipo sensor e dados, uma vez que se prevê que sejam estas as que existam com maior predominância neste tipo de aplicações. Na Tabela 4-2 estão indicadas as características das fontes usadas no cenário 2.

Tabela 4-2: Caracterização das fontes do cenário 2.

Fonte	Débito médio	Tipo
Sensor	800bit/s	constante
Dados	16kbit/s	variável

O comportamento do algoritmo será analisado em duas variantes. A primeira consiste em partir de um número fixo de fontes do tipo Sensor e ir acrescentando fontes do tipo Dados, enquanto

que, na segunda, será feito o inverso, ou seja, a partir de um número fixo de fontes de Dados irão sendo acrescentadas fontes do tipo sensor.

Os resultados apresentados serão apenas aqueles que maximizem a ocupação da largura de banda, ou seja, possuam o valor do *Loopclock* máximo que satisfaça os requisitos de atraso dos serviços da classe AM. Deste modo, será necessário calcular, para cada simulação, o valor indicativo do *Loopclock* máximo:

$$Loopclock_{max} = \frac{45 \times 8}{DébitoMédioTotal} \qquad (4\text{-}4)$$

em que o valor do Débito Médio Total será o somatório dos débitos médios individuais de todos os serviços. À semelhança do valor dado pela equação 4-3, este será um majorante e nunca será atingido na prática.

4.1.2 Sistema de testes para avaliação do Concentrador

Para a avaliação do Concentrador, optou-se por um sistema de testes que aproveita grande parte da estrutura desenvolvida para o Adaptador de terminal. Este modelo simplificado permite avaliar os Algoritmos de escalonamento do Concentrador tendo como entradas um conjunto de portas que recebem o tráfego proveniente de diversos Adaptadores de terminal. A Figura 4-3 mostra a sua estrutura funcional.

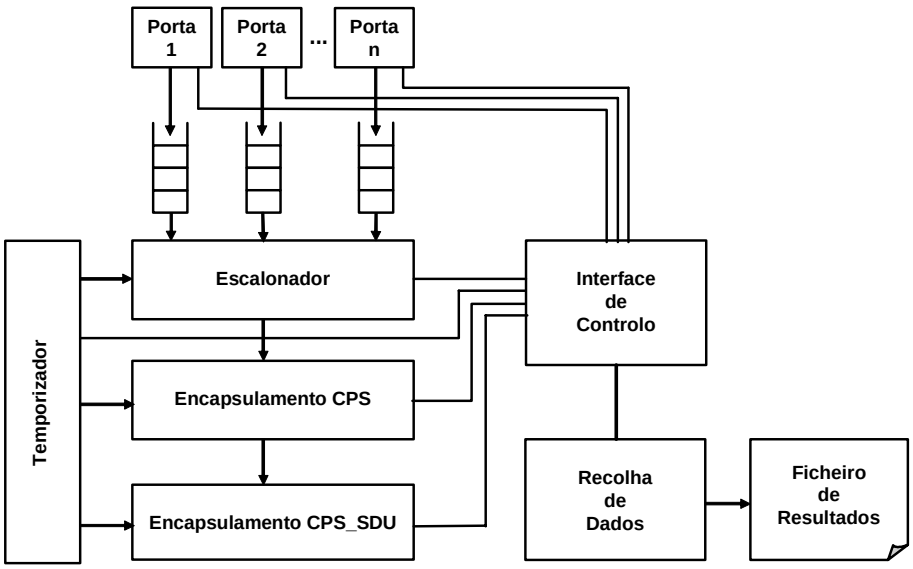


Figura 4-3: Estrutura do sistema de testes para avaliação do Concentrador.

Todos estes módulos foram já descritos na secção anterior, em que apenas se substituíram as fontes geradoras de tráfego por portas de entrada que servirão para gerar tráfego já formatado em pacotes CPS. Com efeito, não é implementado, neste sistema, o desencapsulamento prévio dos pacotes CPS_SDU, provenientes de cada Adaptador de terminal, que terá de existir num sistema real. O tráfego produzido pelos geradores de pacotes CPS é introduzido directamente nas diversas portas de entrada deste sistema de testes.

O escalonador deste módulo será mais simples que o desenvolvido para o Adaptador de terminal, uma vez que o algoritmo usado terá menos parâmetros de entrada como consequência da informação relativa ao Atraso máximo, associada aos fluxos de classe AM, não ser transportada nos pacotes CPS. Deste modo, não será possível implementar um mecanismo de prioridade, baseado no parâmetro de Atraso Máximo do género do implementado no Adaptador de terminal. Por outro lado, será mantida a estrutura de pacotes CPS como unidade elementar de multiplexagem. A definição do Algoritmo de escalonamento do Concentrador e a sua avaliação serão apresentadas com detalhe na secção 4.5.

O módulo de Recolha de dados efectua o cálculo dos seguintes parâmetros:

- Atraso máximo dos fluxos associados a cada classe de tráfego,
- Tamanho máximo dos FIFOs,
- Quantidade de octetos rejeitados (*drop out*) dos pacotes de classe EM.

Devido às características inerentes ao funcionamento do Concentrador, será apenas possível implementar mecanismos de prioridade que permitam que o tráfego de classe AM seja atendido duma forma prioritária em relação ao das outras classes. Deste modo, o desempenho dos diversos algoritmos será avaliado baseado na quantificação do atraso máximo que este módulo introduz a serviços de classe AM.

No caso do nível de concentração ser elevado, ou seja, quando a soma dos fluxos médios de entrada é superior à capacidade de saída, o escalonador terá de rejeitar pacotes de classe EM de forma a poder satisfazer os restantes requisitos.

Finalmente, é de referir que, ao contrário do Adaptador de terminal, o Concentrador não introduz *overhead* à informação recebida, uma vez que os pacotes CPS de entrada, provenientes de cada Adaptador de terminal, já contêm toda a informação nos respectivos cabeçalhos acerca da fonte e do Adaptador de terminal de onde são provenientes e, como tal, não é necessário adicionar nem manipular nenhuma desta informação.

- **Geradores de pacotes CPS**

Para simular o comportamento do Concentrador recorreu-se a um mecanismo artificial de geração de pacotes CPS aplicado a cada uma das portas de entrada. No sistema proposto, o número máximo de portas é de 8 (i.e. são usados os 3 primeiros bits do campo UUI) que corresponde a tráfego gerado por 8 Adaptadores de terminal. A Figura 4-4 ilustra o mecanismo de geração de pacotes CPS.

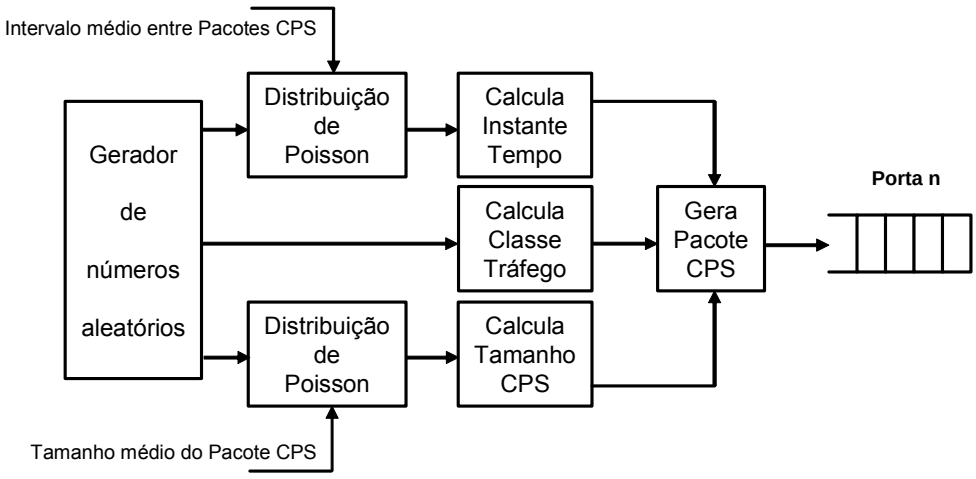


Figura 4-4: Diagrama funcional do mecanismo de geração de pacotes CPS.

São calculados aleatoriamente dois parâmetros, seguindo distribuição de Poisson: o instante de geração e o tamanho do pacote CPS. Para atribuir a classe de tráfego é gerado aleatoriamente, com igual probabilidade, um valor que pode ser 0, 1 ou 2. Estes valores estão associados, respectivamente, às classes de tráfego AM, D e EM. O módulo Gera Pacote CPS vai produzir um pacote CPS, em que o instante de tempo, o tamanho e a classe de tráfego são indicados pelos módulos a montante.

4.2 Configuração de referência

Com o objectivo de se poder conhecer os valores parciais e totais dos atrasos de transmissão da informação ao longo da rede é necessário a definição de uma configuração de referência que possa servir de base para a análise dos vários módulos do sistema. Essa configuração é baseada na estrutura apresentada na Figura 2-4. Para tal é necessário obter os valores máximos para os atrasos de transmissão nos diversos segmentos ao longo da rede. A Figura

4-5 apresenta a configuração de referência onde são evidenciados os diversos atrasos ao longo dos vários segmentos de rede, enquanto que na Tabela 4-3 é feita a sua descrição.

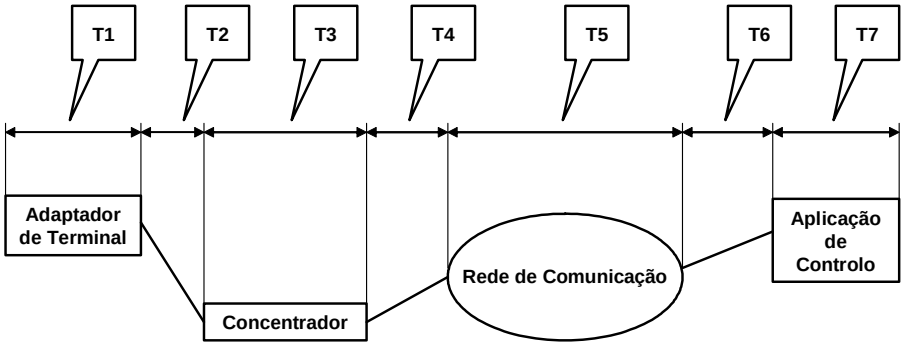


Figura 4-5: Configuração de referência para determinação dos tempos de transmissão do sistema.

Tabela 4-3: Definição dos atrasos de transmissão parciais.

T_1	Atraso da informação provocado pelo escalonamento efectuado no Adaptador de terminal
T_2	Tempo de propagação no troço de ligação entre o Adaptador de Terminal e o Concentrador
T_3	Atraso da informação provocado pelo escalonamento efectuado no Concentrador
T_4	Tempo de propagação no troço de ligação entre o Concentrador e a Rede de Comunicação
T_5	Atraso no interior da Rede de Comunicação
T_6	Tempo de propagação no troço de ligação entre a Rede de Comunicação e a Aplicação de Controlo
T_7	Tempo de processamento da informação na Aplicação de Controlo

Dos atrasos apresentados na Tabela 4-3, podemos considerar dois grupos, de acordo com a sua ordem de grandeza. O primeiro grupo, composto pelas componentes T_2, T_4 e T_6, caracteriza-se por o atraso ser, aproximadamente, função da distância. No segundo grupo, constituído pelas componentes T_1, T_3, T_5 e T_7, o atraso depende dos seguintes factores: escalonamento, filas de espera (e.g. comutadores), empacotamento e processamento de informação (e.g. Aplicação de controlo). Este último grupo é composto por componentes cuja ordem de grandeza é bastante superior às do primeiro.

Para efectuar a análise de desempenho será necessário quantificar o atraso da informação no Adaptador de terminal e no Concentrador (T_1 e T_3) uma vez que todas as outras componentes são determinadas, à partida, pelo comprimento das ligações, pela negociação dos parâmetros

de tráfego e pelo tempo de processamento da Aplicação de controlo. Deste modo, para melhor analisar o atraso global, este é decomposto por três parcelas:

$$T = T_1 + T_3 + T_{garantido} \quad (4\text{-}5)$$

Em que o valor de $T_{garantido}$ é a soma de todas as componentes à excepção de T_1 e T_3. Deste modo, para avaliar o desempenho do Adaptador de terminal e do Concentrador, em termos de atraso introduzido, bastará verificar se o atraso máximo que é imposto a cada um é cumprido. Assim, na simulação dos vários algoritmos de escalonamento, a avaliar em ambos os módulos, é verificado apenas o cumprimento do atraso parcial e não do atraso global.

No projecto de um sistema real será necessário quantificar o atraso correspondente à componente $T_{garantido}$ para se poder atribuir limites a T_1 e T_3.

4.3 Fontes geradoras de tráfego

A especificação completa do sistema a desenvolver necessita de uma caracterização das várias fontes geradoras de tráfego não só em termos de débito mas também em função da QoS (atraso) requerida por cada serviço. Em função destes parâmetros, o sistema deverá efectuar os procedimentos necessários para garantir o suporte do serviço com o atraso pretendido. Essa garantia terá de ser assegurada por duas entidades: a rede e os módulos do sistema onde é efectuado escalonamento temporal. No primeiro caso os parâmetros de QoS são negociados e atribuídos quando é estabelecida a ligação, enquanto que no segundo a garantia da satisfação do atraso é assegurada pelo Adaptador de terminal e pelo Concentrador.

Deste modo, para avaliar e dimensionar os algoritmos de escalonamento, será necessário simular o comportamento destes, tendo como entradas fontes geradoras de tráfego apropriadas. Embora o objectivo deste estudo seja o de integrar o maior número possível de fontes de baixo débito, será desejável que o modelo proposto possa também integrar outros tipos de tráfego, característicos de aplicações de controlo, que possuem requisitos de débito e atraso diferentes. Neste sentido, será necessário avaliar se o sistema apresenta uma boa resposta a serviços com requisitos de atraso temporal baixo suportados, duma forma integrada, com outros em que os requisitos de atraso temporal não são importantes mas que produzem um fluxo elevado de informação. Por exemplo, consideremos o cenário da Figura 4-6, em que se apresenta um exemplo simples onde são suportados 3 tipos de aplicações:

- sistema de aquisição de imagens (débito variável),
- serviço de voz (débito constante),
- sistema de actuação múltipla de dispositivos (baixo débito, aleatório e esporádico).

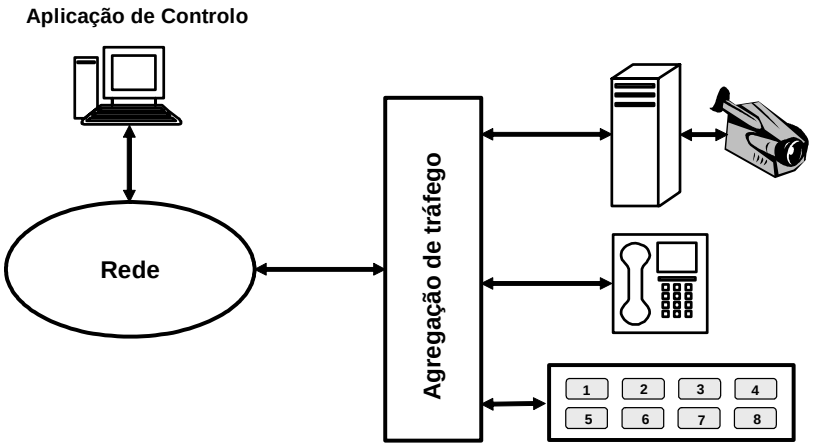


Figura 4-6: Cenário básico de um sistema de suporte integrado de três aplicações com diferentes requisitos temporais.

Neste caso, é importante verificar se o fluxo mais elevado de informação, produzido pelo sistema de aquisição de imagens, não afecta o transporte das outras duas aplicações, sem serem excedidos os tempos de atraso máximos exigidos por cada uma delas, ou seja, é preciso verificar se o escalonamento dos diversos fluxos é efectuado de forma eficiente.

A escolha das diversas fontes teve em conta, por um lado o cuidado de usar as mais genéricas e de uso mais vulgar em aplicações de baixo débito e, por outro, a necessidade de gerar tráfego com características especiais e de débito mais elevado de forma a avaliar não só a capacidade de agregação mas também a capacidade de integração de fluxos de características diversas e assim conseguir efectuar o correcto dimensionamento de um sistema genérico para o suporte de aplicações de baixo débito. Para testar o desempenho dos vários algoritmos, foram usados quatro tipos de fontes de tráfego a saber:

- Voz – 16kbit/s – débito constante,
- Sensor de temperatura, pressão ou humidade – 800bit/s – débito constante,
- Dados – 16kbit/s – débito variável,
- Painel de interruptores – 80bit/s – aleatório,
- Vídeo – 80kbit/s – débito variável.

O enquadramento das fontes descritas acima, em aplicações de controlo, foi já abordado na secção 2.1.2. O que importa aqui referir é que, através da definição de cenários concretos em que se usam um conjunto de fontes do tipo das mencionadas acima, será possível carregar o sistema com tráfego de características diversificadas de forma a permitir obter informação acerca da resposta, de cada um dos algoritmos, em termos de atraso e eficiência de multiplexagem.

4.3.1 Geradores de números aleatórios

Normalmente, os processos de simulação requerem a obtenção de amostras provenientes de diferentes distribuições de probabilidade. Este processo é efectuado em duas etapas. Primeiro, obtém-se uma sequência de números aleatórios uniformemente distribuídos num dado intervalo. Depois, a sequência é transformada de modo a produzir valores aleatórios associados à distribuição desejada.

Como geradores de números aleatórios foram testados dois algoritmos do tipo LCG (LCG, *Linear Congruential Generator*) [Ent97] cuja base é:

$$y(n + 1) = [a \cdot y(n) + b] \ (mod\ m)^1$$

O primeiro algoritmo é o usado na função rand() do ANSI C [Ker88] e tem os seguintes parâmetros:

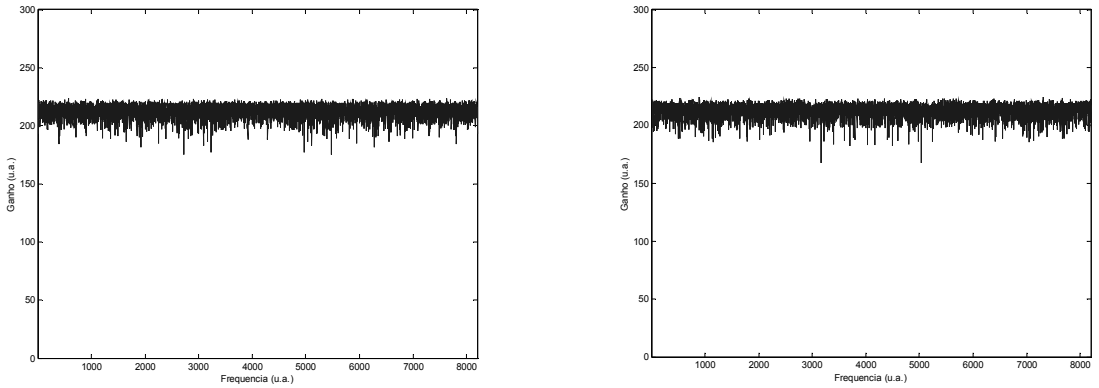
$m=2^{31}$ $a=1103515245$ $b=12345$ $y_o=12345$

O segundo algoritmo é conhecido por *minimal standard* e foi sugerido por [Lew69] em 1969 e tem a particularidade de ser um gerador de números aleatórios de período completo, ou seja, ao fim de um número muito elevado de números são gerados todos os valores da sequência. Apesar de ser um algoritmo com alguns anos, tem passado em todos os novos testes de desempenho e demonstrado a sua utilidade em muitos casos. Os seus parâmetros são:

$m=2^{31}-1$ $a=7^5$ $b=0$ $y_o=1$

[1] O operador *mod* representa o resto da divisão inteira entre dois números

A Figura 4-7 mostra as Transformadas de Fourier de sequências de 8192 números aleatórios gerados com cada um dos algoritmos. Os gráficos de ambas as transformadas aproximam-se do espectro contínuo do ruído branco, o que permite que qualquer um deles possa ser usado na modelização das fontes de tráfego. No entanto, devido às vantagens e desvantagens referenciadas em [Ent97], optou-se por usar a versão *minimal standard*.



(a) (b)

Figura 4-7: Transformada de Fourier de sequências de 8192 números aleatórios gerados por 2 algoritmos: (a) – Função rand() do ANSI C. (b) - *Minimal Standard*. Os valores do ganho e da frequência têm unidades arbitrárias.

4.3.2 Modelização das fontes de tráfego

Em vez de se usarem fontes reais, foram criados modelos de simulação para geração de tráfego de forma a poder criar as situações mais favoráveis ao melhor conhecimento do sistema e assim se poder efectuar o seu dimensionamento duma forma mais exacta.

Para todos os algoritmos geradores de tráfego foi usada a interrupção do relógio de tempo real do sistema operativo que, teoricamente, pode ser programada para funcionar em múltiplos inteiros de 1μs [Set93].

- **Voz**

Gerador de tráfego de débito constante de 16kbit/s. Este gerador coloca na fila de entrada 100 octetos a cada interrupção de 50ms em 50ms.

- **Sensor**

Gerador de tráfego de débito constante a 800bit/s. Para tal, são colocados na fila respectiva 10 octetos a cada interrupção de 100ms em 100ms.

- **Dados**

Gerador de tráfego aleatório com média de 16kbit/s. Para implementar este gerador de tráfego, o número de octetos gerado é fixo e igual a 20 e o instante de tempo é aleatório, seguindo a distribuição de Poisson com média igual a 10ms. Algumas simulações não apresentadas mostraram que o resultado seria idêntico se fosse usado um intervalo de tempo fixo de 100ms e um número de octetos gerado a cada interrupção aleatório, seguindo a distribuição de Poisson com média igual a 20 octetos.

- **Interruptores**

Gerador de trafego aleatório com média igual a 80bit/s. Tal como o anterior, o número de octetos gerado é fixo e igual a 10, enquanto que o instante em que ocorre a interrupção é aleatório, seguindo a distribuição de Poisson com média igual a 1s.

- **Vídeo**

Gerador de tráfego aleatório com média igual a 80kbit/s. O número de octetos gerado a cada interrupção é fixo e igual a 50, enquanto que o instante em que ocorre a interrupção é aleatório, seguindo a distribuição de Poisson com média igual a 5ms.

Para implementar o gerador de tráfego aleatório foi usado um gerador não uniforme que segue a distribuição de Poisson. O algoritmo implementado é descrito em [Pre92]. A Figura 4-8 mostra o histograma do resultado de um gerador de números aleatórios com média 100, seguindo a distribuição de Poisson.

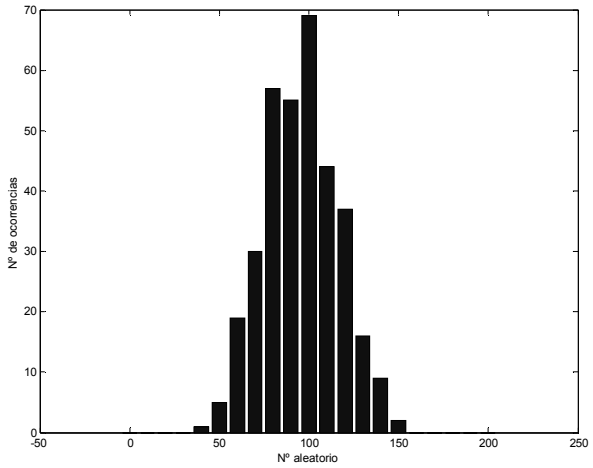


Figura 4-8: Histograma do resultado de um gerador de números aleatórios com média 100 com distribuição de Poisson. Os dados foram agrupados em intervalos de 10 unidades.

Os valores reais da média e da variância são respectivamente 100.01 e 99.433, ou seja, aproximam-se do valor teórico de 100 da distribuição de Poisson.

4.3.3 Critério de paragem da simulação

Para definir o critério de paragem das simulações efectuadas foram observados os instantes de escrita no ficheiro de resultados durante uma simulação com a duração de 60m, usando um algoritmo arbitrário para cada cenário de simulação. Através da observação desses instantes, verificou-se que o número de acessos de escrita ao registo de valores máximos do módulo de Recolha de dados era tanto maior quanto mais próximo se estava do início da simulação. A partir de um dado instante, verificou-se que esse acesso deixava de ser efectuado. Além disso, o intervalo entre instantes consecutivos de escrita aumentava normalmente à medida que se avançava na simulação. No entanto, verificou-se que um dado intervalo entre duas escritas consecutivas era sempre inferior ao dobro do anterior. Deste modo, definiu-se o seguinte critério de paragem:

1. Em cada instante de escrita no registo de valores máximos, calcula-se o valor do instante T_{Fim} correspondente ao dobro do intervalo anterior,
2. Se até ser atingido esse instante (T_{Fim}) houver escritas nesse registo calcula-se um novo valor para T_{Fim} e repete-se o processo,
3. Caso contrário a simulação é terminada.

Foram efectuadas várias simulações de 60m com alguns dos algoritmos descritos em que se comprovou que este critério era suficiente para definir o fim de uma simulação.

4.4 Avaliação dos Algoritmos de escalonamento do Adaptador de terminal

Nesta secção são apresentados os resultados do estudo efectuado para a especificação e avaliação de desempenho dos algoritmos de escalonamento do Adaptador de terminal. Este bloco funcional, pertencente ao módulo de emissão do Adaptador de terminal, é responsável pelo escalonamento dos fluxos de informação recebidos, tendo em conta os requisitos temporais de cada serviço.

4.4.1 Algoritmos básicos

Estes algoritmos destinaram-se avaliar o funcionamento global dos diversos blocos funcionais de modo a ganhar sensibilidade para a definição de outros algoritmos capazes de satisfazer os requisitos de QoS (i.e. atraso) de cada serviço. Qualquer um destes algoritmos não entra em conta com os requisitos de atraso de cada serviço nem com nenhum outro parâmetro relacionado com a QoS.

Foram definidos 4 algoritmos básicos de forma a efectuar o escalonamento da informação proveniente de cada um dos FIFOs associados a cada fonte de tráfego. A descrição desses algoritmos é apresentada na Tabela 4-4.

Tabela 4-4: Descrição dos Algoritmos básicos usados na avaliação do Adaptador de terminal.

Algoritmo 1: Critério do FIFO mais cheio	Analisa o tamanho dos diversos FIFOs em cada instante dado pelo *Loopclock*. Atende o FIFO que tem o maior número de octetos.
Algoritmo 2: Critério do octeto mais antigo	Lê a amostra temporal do primeiro octeto de cada FIFO ao ritmo do *Loopclock*. Atende o que, em cada instante, tiver o octeto mais antigo.
Algoritmo 3: Leitura Sequencial	Atende os FIFOs sequencialmente.
Algoritmo 4: Leitura Aleatória	Atende os FIFOs aleatoriamente.

Para efectuar o teste destes algoritmos foi apenas usado o cenário de simulação 1, uma vez que o que interessa avaliar, nestes casos, é o funcionamento básico das diversas funções envolvidas no escalonamento da informação. Deste modo, o cenário 1 permitirá carregar o sistema de forma a retirar informações mais importantes do que o cenário 2 uma vez que é constituído por um conjunto de fontes de tráfego com características mais diversificadas em termos de débito e requisitos de atraso.

Para cada algoritmo foram efectuadas simulações para diferentes valores do *Loopclock*. Como foi referido e se verificará a seguir, o valor deste parâmetro influencia directamente a eficiência de multiplexagem e inversamente o atraso no escalonamento da informação.

Nos parágrafos seguintes são analisados os resultados obtidos para cada um dos algoritmos descritos acima. As quatro primeiras figuras representam a variação do atraso de escalonamento em função do *Loopclock*.

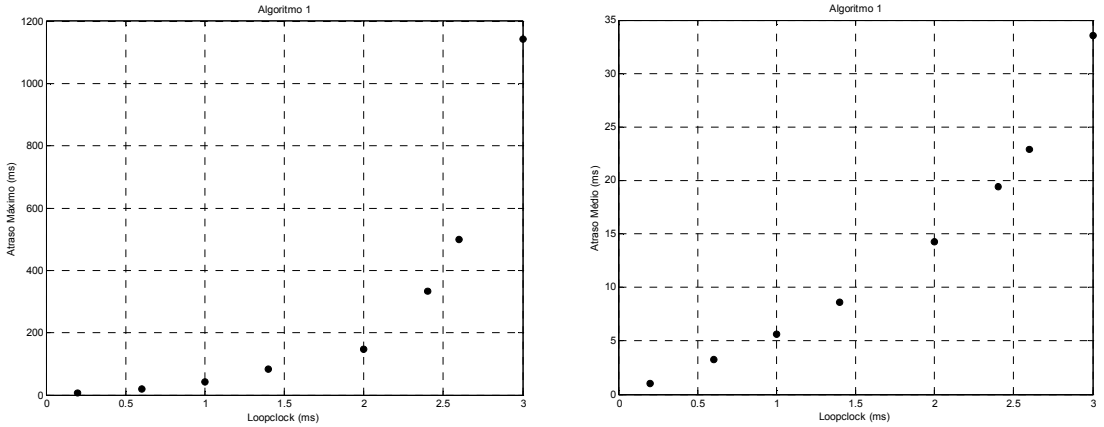


Figura 4-9: Algoritmo 1: Atraso Máximo e Médio em função do *Loopclock*.

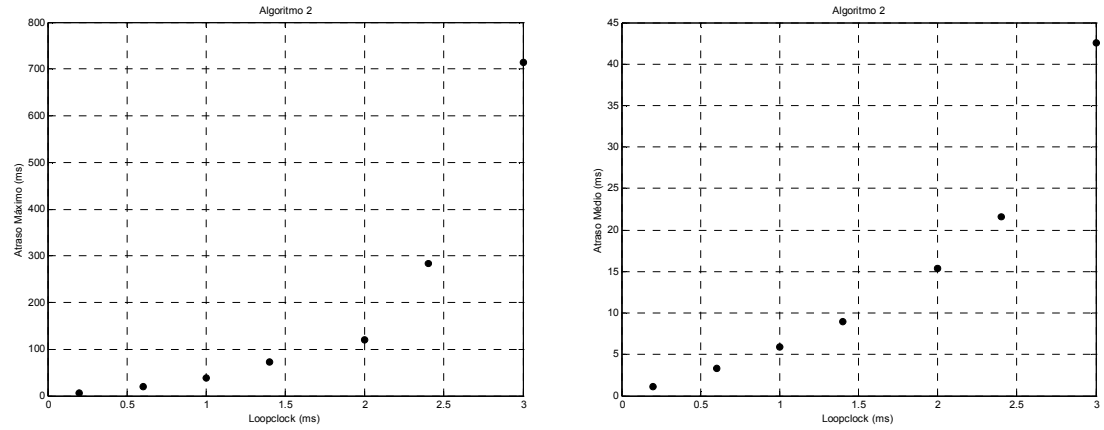


Figura 4-10: Algoritmo 2: Atraso Máximo e Médio em função do *Loopclock*.

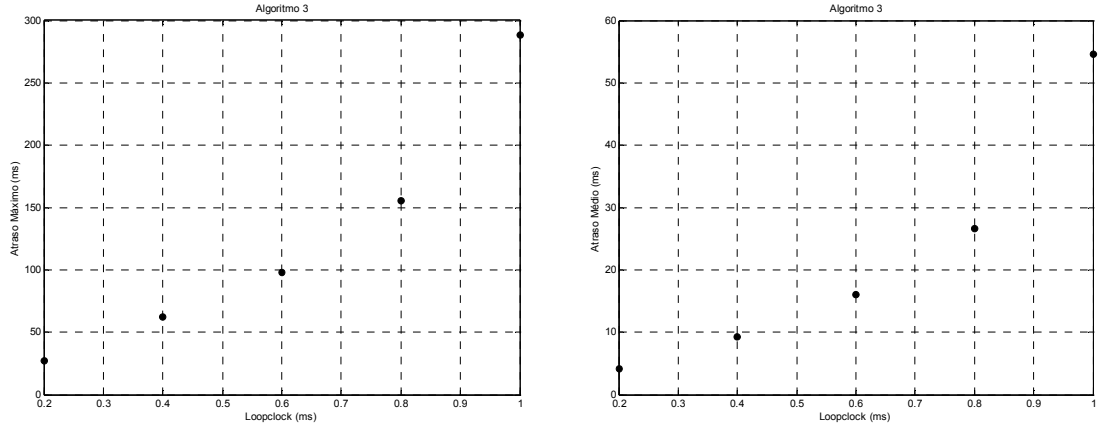


Figura 4-11: Algoritmo 3: Atraso Máximo e Médio em função do *Loopclock*.

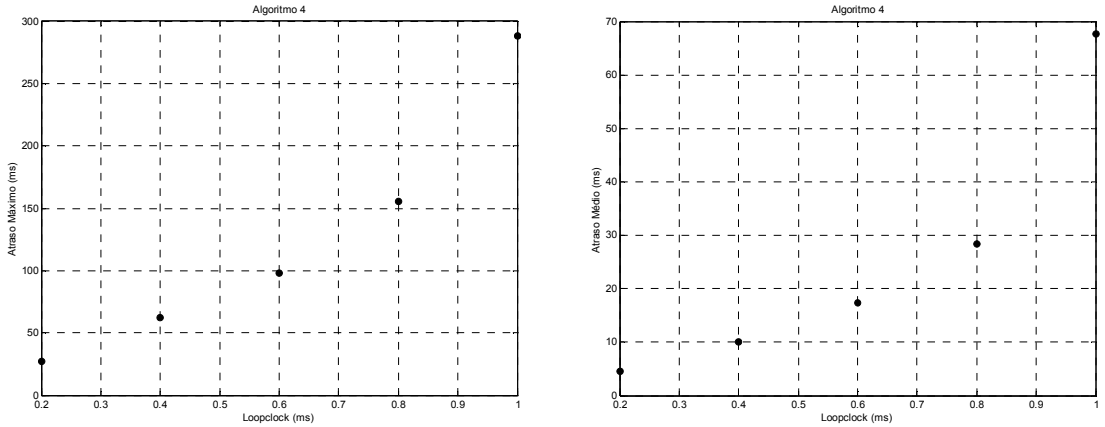


Figura 4-12: Algoritmo 4: Atraso Máximo e Médio em função do *Loopclock*.

Os resultados apresentados nas figuras anteriores representam a variação do atraso máximo e médio em função do *Loopclock*, calculado para todas as fontes de tráfego.

- **Análise dos resultados**

  1. Todos os resultados obtidos mostram uma evolução exponencial do atraso em função do valor do *Loopclock*,

  2. Os dois primeiros algoritmos revelam um comportamento bastante superior relativamente aos dois últimos. Este facto já era esperado, uma vez que só os dois primeiros se baseiam em parâmetros que influenciam o atraso,

  3. Pela mesma razão, os dois últimos algoritmos não conseguem funcionar (ocorre *overflow* nos FIFOs) para valores do *Loopclock* acima de 1ms relativamente aos 3ms dos dois primeiros. Esta limitação implica uma perda na eficiência de multiplexagem dos dois últimos algoritmos relativamente aos dois primeiros,

  4. Para valores do *Loopclock* abaixo de 3ms, os dois primeiros algoritmos produzem valores baixos de atraso, sendo o algoritmo 1 ligeiramente melhor,

  5. A relação entre o atraso máximo e médio é praticamente constante para valores baixos do *Loopclock* como pode ser observado na Figura 4-13,

  6. Através da análise da Figura 4-14 pode ser observado que a eficiência de multiplexagem aumenta directamente com o valor do *Loopclock*. Este facto também era esperado assim como o melhor desempenho do algoritmo 1 relativamente ao 2 para valores acima dos 2ms.

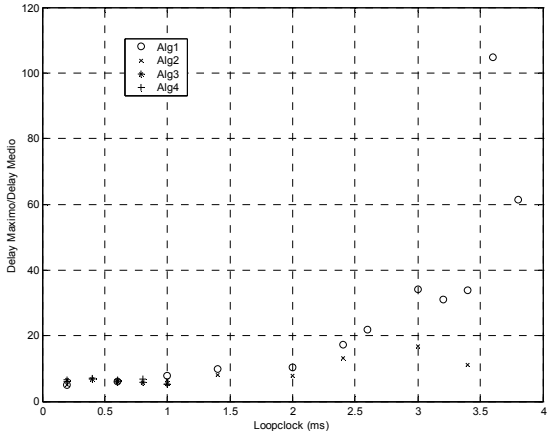


Figura 4-13: Variação do quociente entre o Atraso Máximo e Médio em função do *Loopclock*.

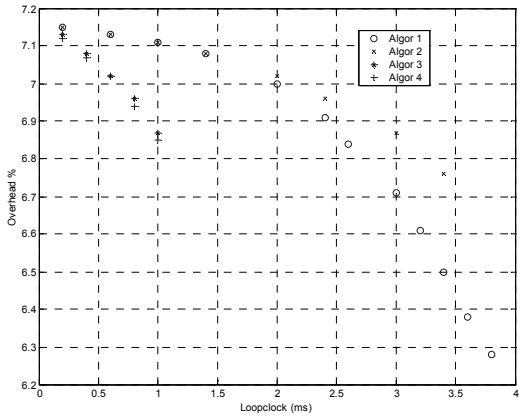


Figura 4-14: Variação do *Overhead* em função do valor do *Loopclock*, para os 4 algoritmos.

- **Conclusões:**

Após a análise atenta dos resultados obtidos é possível retirar as seguintes conclusões:

1. O algoritmo 1 tem um melhor desempenho em termos de eficiência de multiplexagem e atraso. Deste modo, o algoritmo a dimensionar deverá usar o valor do tamanho dos FIFOs como um dos parâmetros principais,

2. A garantia do atraso para um dado serviço é conseguida à custa de um dado valor de *Loopclock*. No entanto, para valores baixos do *Loopclock* a eficiência de multiplexagem diminui devido aos pacotes CPS terem um comprimento menor. Deste modo, o algoritmo a dimensionar deverá permitir estimar o valor do *Loopclock* em função dos parâmetros de tráfego das fontes que garanta a satisfação do atraso máximo de cada fonte e maximize, ao mesmo tempo, a eficiência de multiplexagem.

A partir destas conclusões será apresentado o algoritmo orientado às classes de tráfego, descrito na secção seguinte.

4.4.2 Algoritmo orientado às classes de tráfego

Na secção anterior foram descritos os resultados das simulações de alguns algoritmos básicos que permitiram obter algum conhecimento sobre o comportamento deste módulo em termos de atraso e eficiência de multiplexagem. Através dos resultados obtidos e dos conhecimentos adquiridos foi possível definir um algoritmo que permitirá efectuar o escalonamento da informação proveniente das diversas fontes em função do atraso requerido por cada uma delas.

Os parâmetros associados às classes de tráfego, definidas no capítulo 2, em função das características do serviço a suportar, permitem implementar um mecanismo eficiente de prioridade no atendimento dos FIFOs, associados a cada fonte de tráfego. De acordo com a classe de tráfego, associada a um dado fluxo de informação, um dado algoritmo de escalonamento poderá implementar mecanismos de prioridade no sentido de satisfazer os requisitos de atraso para cada serviço maximizando, simultaneamente, a eficiência no preenchimento dos pacotes CPS.

Assim, para cada fluxo de informação é atribuída uma fila do tipo FIFO e, em cada uma delas, é mantida uma tabela de estado que indicará os seguintes parâmetros:

- Ocupação da fila em octetos,
- Classe de tráfego do fluxo associado,
- Instante temporal do octeto mais antigo na fila (octeto que está “à cabeça” do FIFO),
- Atraso Máximo a garantir – no caso dos fluxos pertencerem às classes AM.

Em função dos parâmetros anteriores, o escalonador usará o seguinte algoritmo:

1. Calcula os instantes de atendimento das filas (*deadlines*), de classe AM, em função dos parâmetros de “Atraso Máximo” respectivos,
2. Enquanto nenhum desses instantes for atingido (*deadline*), atende sequencialmente a fila que tiver maior tamanho (podem ser filas de classes AM ou D),

3. Quando é atingida a *deadline* de uma das filas de classe AM atende a fila correspondente,

4. Se não houver informação nas filas de classe AM ou D, transmite os pacotes das filas das classes EF, de forma sequencial, da maior para a menor.

Deste modo, e de acordo com os resultados obtidos na simulação com os algoritmos básicos, enquanto não são atingidos os instantes de envio dos pacotes da classe AM, o critério de prioridade é baseado na selecção da fila que possuir, num determinado instante, o maior número de octetos. As filas de classe EM só ganham prioridade quando, num dado instante, não existir informação nas filas das restantes classes.

A Figura 4-15 ilustra o mecanismo de escalonamento de tráfego em que é usada a estrutura de multiplexagem dos pacotes CPS da camada AAL-2.

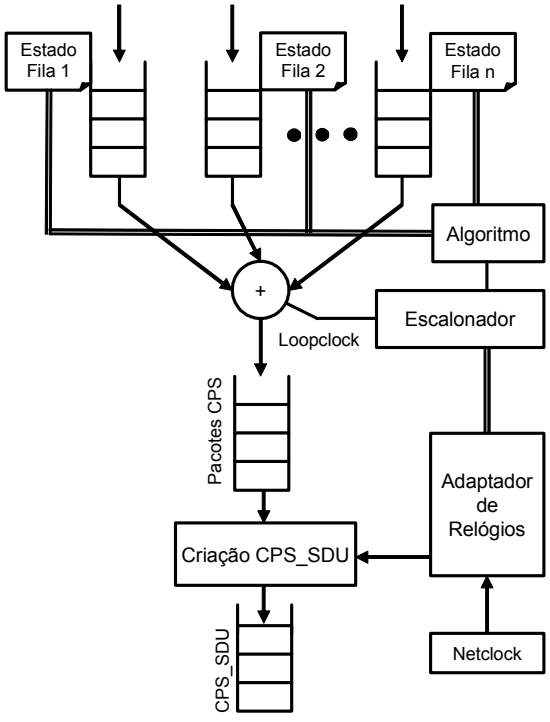


Figura 4-15: Diagrama funcional do mecanismo de escalonamento de tráfego do Adaptador de terminal, em que é usada a estrutura de multiplexagem dos pacotes CPS da camada AAL-2.

Em cada instante, sincronizado pelo *Loopclock*, o escalonador vai tentar ler do FIFO, seleccionado pelo algoritmo de escalonamento, o maior número de octetos possível (45) de modo a preencher um pacote CPS com o número máximo de octetos e assim conseguir a

eficiência máxima. No caso de não existirem octetos em nenhuma fila, o escalonador não formará nenhum pacote CPS e o algoritmo retorna ao ponto de partida.

Uma vez que os fluxos de baixo débito e de classe AM podem produzir pouca informação relativamente aos fluxos de outras classes, poderá acontecer que, devido à necessidade de atender estes serviços com alguma prioridade, não seja possível preencher os pacotes com o tamanho máximo. Nestes casos a perda de eficiência de multiplexagem, associada à criação de pacotes mais pequenos, é derivada da necessidade de satisfazer os requisitos de atraso destes serviços.

Avaliação do comportamento do algoritmo com o cenário 1

Para efectuar a avaliação do algoritmo com o cenário 1 será necessário atribuir uma classe de tráfego a cada fluxo de acordo com as características do serviço a suportar. A Tabela 4-5 apresenta uma possível atribuição das classes de tráfego, definidas no capítulo 2, a cada um dos serviços definidos para o cenário 1.

Tabela 4-5: Características de tráfego dos serviços de acordo com o cenário 1. Especificação da classe de tráfego e do atraso máximo associado.

Serviço	Características	Débito médio	Classe de tráfego	Atraso Máximo (ms)
S1	Voz – débito constante	16kbit/s	AM	15
S2	Sensor – débito constante	800bit/s	AM	200
S3	Dados – débito variável	16kbit/s	D	---
S4	Interruptores – débito constante	80bit/s	AM	50
S5	Vídeo – débito variável	80kbit/s	EM	---

Aos serviços de débito constante foram atribuídas classes do tipo AM, uma vez que estes fluxos necessitam que o atraso extremo-a-extremo seja controlado de forma a respeitar a QoS pretendida para o serviço. No caso do fluxo de dados optou-se por se atribuir à classe D, uma vez que esta foi especialmente definida para o suporte deste tipo de tráfego. Finalmente, foi atribuída ao serviço de Vídeo de débito variável a classe EM uma vez que foi definido, para este serviço, que a perda ocasional de informação não seria prejudicial para o desempenho da aplicação (e.g. vigilância).

A última coluna da Tabela 4-5 especifica o atraso máximo que um dado fluxo de informação poderá ter ao “atravessar” o Adaptador de terminal. Este valor corresponde à parcela designada por T1 no modelo de referência da Figura 4-5. Como foi definido, só os serviços da classe AM é

que têm associado este parâmetro. Para as outras classes, como não existem limitações no atraso, o escalonador não impõe nenhum tipo de prioridade relacionado com este parâmetro.

Os valores atribuídos para o Atraso Máximo aos serviços S1, S2 e S4 tiveram por base as características apresentadas na secção 2.1.2, considerando que uma parcela do atraso total seria devida ao Adaptador de terminal. Deste modo, os valores apresentados na Tabela 4-5 representam a parte do atraso total que é imposto ao Adaptador de terminal de acordo com o que foi explicado na secção 4.2. Além disso, estes valores são apenas indicativos para efectuar o teste do algoritmo, o que pode significar que, na prática, estes valores possam ser menos restritivos.

Os resultados dos testes efectuados com este cenário de simulação são apresentados na Tabela 4-6. Os valores representados na tabela correspondem aos atrasos máximos de cada serviço assim como os valores máximos dos níveis dos FIFOs associados a cada fluxo de informação. Como valor indicativo da eficiência de multiplexagem é indicado, na última coluna, o tamanho médio dos pacotes CPS.

Tabela 4-6: Resultados da avaliação do algoritmo de escalonamento, orientado às classes de tráfego, usando o cenário 1.

	Atraso máximo (ms)					**Tamanho máximo do FIFO (nº de octetos)**					
Loopclock (ms)	**D1**	**D2**	**D3**	**D4**	**D5**	**F1**	**F2**	**F3**	**F4**	**F5**	**CPS**
0,2	0,8	0,2	1,6	0,2	3,4	100	10	200	10	500	39,01
0,6	2,4	0,6	5,4	0,6	12,6	100	10	200	10	640	39,01
1,0	4,2	1,0	9,0	1,0	21,9	100	10	200	10	820	39,10
1,4	6,9	1,4	12,5	1,4	33,6	100	10	200	10	955	39,11
1,8	8,9	1,8	16,2	1,8	56,1	100	10	265	10	1005	39,11
2,2	9,4	2,2	19,8	2,2	105,4	100	10	310	10	1580	39,13
2,4	9,8	2,4	21,4	2,4	151,7	100	10	310	10	2120	39,15
3,0	14,8	3,0	29,9	3,0	632,4	100	10	355	10	5365	39,17

Cada linha da tabela corresponde a uma simulação para um determinado valor do *Loopclock*. Foram apenas apresentados os valores abaixo do valor indicativo da equação 4-3 (i.e. 3,19ms) Acima deste, os valores obtidos do atraso e tamanho dos FIFOs não satisfazem os objectivos pretendidos.

- **Análise dos resultados**

Através da análise da tabela de resultados podemos observar o seguinte:

1. O algoritmo tem uma boa resposta aos serviços da classe AM, conforme se pretendia aquando o seu dimensionamento,

2. A satisfação dos requisitos de atraso para serviços de muito baixo débito, quando multiplexados com outros, de débitos de ordem de grandeza superior, é garantida à partida (o atraso é praticamente igual ao valor do *Loopclock*),

3. Os pacotes CPS correspondentes aos serviços S2 e S4 têm um tamanho bastante pequeno (i.e. entre 1 a 10 octetos). Este facto deve-se à necessidade de escalonar as filas correspondentes de modo a poder cumprir o valor do parâmetro AM,

4. O serviço que impõe mais restrições ao sistema é o S1 porque é o serviço de classe AM com maior peso (14,17%),

5. Como era pretendido, verifica-se que o serviço S1 é, também, o que impõe mais prioridade ao escalonador de forma a tentar cumprir o atraso máximo pretendido. Isso é observado através do tamanho de F1 que, dentro da gama de valores apresentada na tabela, é constante (100 octetos). Ao não haver acumulação de informação no FIFO significa que, para esta gama de valores, o valor do atraso está relacionado directamente com o valor do *Loopclock*, como ilustra a Figura 4-16,

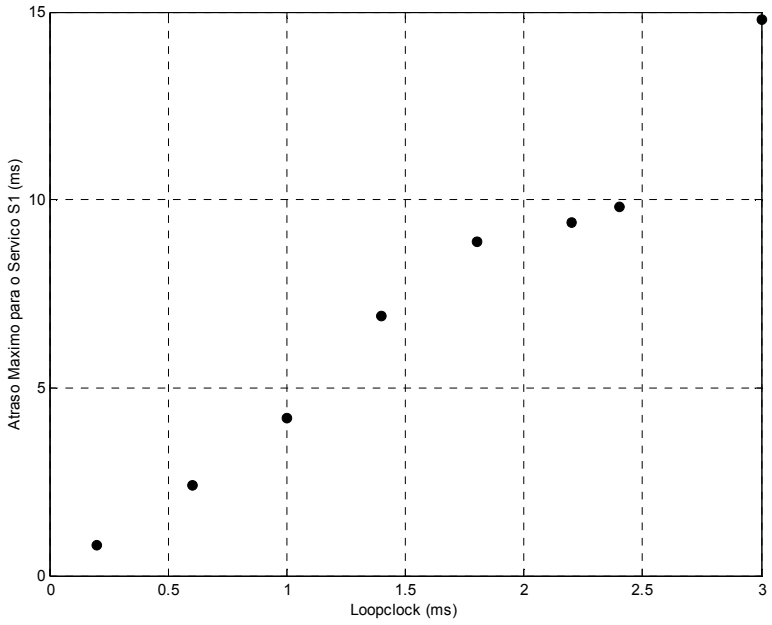


Figura 4-16: Variação do Atraso máximo em função do *Loopclock* para o serviço S1.

6. Os resultados obtidos mostram que os requisitos de atraso são satisfeitos para todos os serviços,

7. O serviço S5, embora tenha um peso elevado (70,87%) e seja do tipo EM, é multiplexado duma forma eficiente pelo escalonador, contribuindo fortemente para uma boa taxa de ocupação média de pacotes CPS. O tamanho médio dos pacotes obtido foi de 39 octetos o que corresponde a cerca de 7,7% de *overhead*,

8. O serviço S5, por ter a mais baixa prioridade, apresenta valores do nível do FIFO relacionados directamente com o valor do *Loopclock*, como ilustra a Figura 4-17.

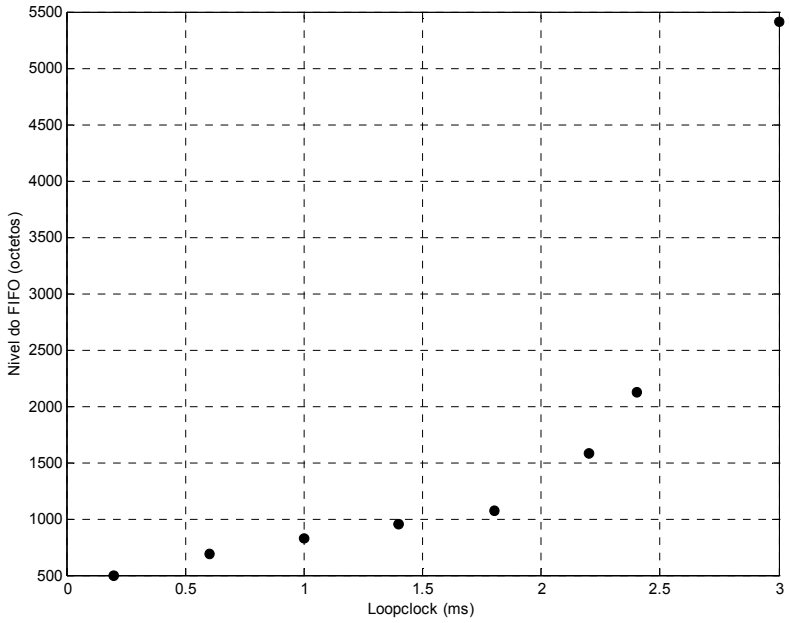


Figura 4-17: Variação do nível do FIFO em função do *Loopclock* para o serviço S5.

- **Conclusões**

1. Este cenário de simulação permite concluir que o escalonador consegue ter um bom desempenho. Consegue satisfazer os requisitos de atraso dos serviços de classe AM assim como garantir uma boa eficiência na multiplexagem de serviços com características diferentes em termos de débito e requisitos temporais,

2. No caso em que uma das fontes geradoras de tráfego de classe AM imponha ao sistema um valor para o seu atraso máximo duma ordem de grandeza próxima do valor indicativo da equação 4-3 (e.g. S1 – ver Figura 4-16), haverá uma degradação da eficiência de multiplexagem, uma vez que o valor da frequência do *Loopclock* terá de ser mais elevada para conseguir cumprir esse requisito,

3. A desvantagem apontada no parágrafo anterior será minimizada quando o sistema suporta um número elevado de fontes e em que o valor indicativo da equação 4-3 é sempre de ordem de grandeza bastante inferior ao do atraso máximo de qualquer uma das fontes de entrada.

Avaliação do comportamento do algoritmo com o cenário 2

Como foi referido na secção 4.1.1, este cenário de simulação destina-se a avaliar o comportamento dos algoritmos de escalonamento em função do número de fontes de tráfego.

Em aplicações de baixo débito as fontes que existirão com mais predominância são as do tipo Sensor e Dados. Deste modo, será conveniente analisar o comportamento dos algoritmos em função do número de fontes de cada um destes tipos. Assim, foram simuladas duas variantes deste cenário, em que a primeira se destina a avaliar o efeito do aumento do número de fontes do tipo Dados, enquanto que a segunda avalia o efeito do aumento do número de fontes do tipo Sensor. A Tabela 4-7 apresenta a caracterização das fontes de tráfego usadas no cenário 2.

Tabela 4-7: Características de tráfego das fontes de acordo com o cenário 2. Especificação do atraso máximo e classe de tráfego associada.

Serviço	Características	Débito médio	Classe de tráfego	Atraso Máximo (ms)
S2	Sensor – débito constante	800bit/s	AM	200
S3	Dados – débito variável	16kbit/s	D	---

As características de tráfego apresentadas na Tabela 4-7 mantêm a coerência com os pressupostos apresentados na secção 4.3, em que o parâmetro de Atraso Máximo associado à fonte do tipo Sensor é o mesmo usado para o cenário 1.

- **Análise de resultados**

A primeira variante deste cenário pretende avaliar o comportamento do algoritmo à medida que se vão introduzindo fontes do tipo Dados. Assim, partiu-se de uma situação com o mesmo número de fontes do tipo Sensor e Dados (i.e. 5 de cada) e, posteriormente, foram-se acrescentando, em cada simulação, mais 5 fontes de cada vez. Na Tabela 4-8 (Variante A) estão registados os valores obtidos para os atrasos máximos de cada tipo de fluxo, o tamanho máximo dos FIFOs, associados a cada fluxo, o tamanho médio dos pacotes CPS, assim como os valores indicativos ($Loopclock_{max}$) e simulados ($Loopclock_{sim}$) do *Loopclock*.

Tabela 4-8: Resultados da avaliação do algoritmo orientado às classes de tráfego usando o cenário 2 (Variante A).

Nº S3	$Loopclock_{Max}$ (ms)	$Loopclock_{Sim}$ (ms)	D2 (ms)	D3 (ms)	F2 (octetos)	F3 (octetos)	CPS (octetos)
5	4,286	4,1	20,5	2054	10	3575	35,65
10	2,195	2	10,5	1047	10	1985	41,11
15	1,475	1,4	7,0	691	10	1365	43,67
20	1,111	1,1	5,5	964	10	1805	45,17

A segunda variante consiste em efectuar um procedimento semelhante. Começa-se por simular o algoritmo com o mesmo número de fontes de cada tipo (i.e. 5 de cada), em que agora se acrescentam, de cada vez, 5 fontes do tipo Sensor. Os resultados obtidos (Variante B) são apresentados na Tabela 4-9, seguindo a mesma lógica dos apresentados na Variante A.

Tabela 4-9: Resultados da avaliação do algoritmo orientado às classes de tráfego usando o cenário 2 (Variante B).

Nº S2	$Loopclock_{Max}$ (ms)	$Loopclock_{Sim}$ (ms)	D2 (ms)	D3 (ms)	F2 (octetos)	F3 (octetos)	CPS (octetos)
5	4,286	4,1	20,5	2054	10	3575	35,65
10	4,091	3,9	39	5251	10	7375	32,21
15	3,913	2,4	36	311	10	740	21,32
20	3,750	1,6	32	122	10	510	19,53

1. No caso da Variante A verifica-se que o aumento do número de fontes do tipo Dados melhora o desempenho do algoritmo, uma vez que aumenta o tamanho médio dos pacotes CPS (i.e. diminui o *overhead*) assim como diminui o atraso associado a cada fonte,

2. No caso da Variante B verifica-se um fenómeno inverso, ou seja, a introdução de fontes do tipo Sensor diminui o desempenho do algoritmo, em termos de eficiência de multiplexagem, uma vez que é necessário processar pacotes com tamanhos menores de modo a cumprir os requisitos de atraso destas fontes,

3. Em qualquer uma das variantes, o algoritmo consegue cumprir os atrasos estabelecidos para as fontes do tipo Sensor (200ms). Na prática, verifica-se que este valor é menos restritivo, uma vez que poderão existir fontes deste tipo com valores de Atraso Máximo bastante superior. Para fundamentar este pressuposto efectuou-se a

simulação com as características indicadas na Tabela 4-7, em que apenas se alterou o valor do Atraso Máximo, associado às fontes do tipo Sensor, para 2,5s. Os resultados são apresentados na Tabela 4-10.

Tabela 4-10: Resultados da avaliação do algoritmo orientado às classes de tráfego usando o cenário 2 (Variante B - modificado).

Nº S2	**$Loopclock_{Max}$ (ms)**	**$Loopclock_{Sim}$ (ms)**	**D2 (ms)**	**D3 (ms)**	**F2 (octetos)**	**F3 (octetos)**	**CPS (octetos)**
5	4,286	4,1	221	534	30	1145	37,12
10	4,091	3,9	236	644	30	1320	29,63
15	3,913	2,4	235	145	30	465	23,78
20	3,750	1,6	177	86	20	375	21,17

4. Ao aumentar o valor do Atraso Máximo para as fontes do tipo Sensor, verificou-se um ligeiro aumento da eficiência de multiplexagem, uma vez que o escalonador consegue reter octetos do tipo AM nos FIFOs durante mais tempo e assim conseguir criar pacotes com tamanhos maiores,

5. É também possível prever que, se os débitos das fontes do tipo Sensor forem da mesma ordem de grandeza das do tipo Dados, a eficiência também iria aumentar, pelas mesmas razões apontadas no parágrafo anterior.

- **Conclusões**

1. Este cenário de simulação permite concluir que o escalonador consegue ter um bom desempenho em termos de cumprimento do atraso dos serviços de classe AM,

2. A eficiência de multiplexagem é sempre dependente das características de tráfego das fontes que se apresentam à entrada do Adaptador de terminal,

3. Quanto menor for a quantidade de fontes do tipo AM e os requisitos de atraso destas melhor será o desempenho do algoritmo em termos de eficiência de multiplexagem,

4. Em qualquer dos cenários verificou-se que o algoritmo proposto para o escalonador consegue cumprir os requisitos de atraso imposto à entrada pelas fontes de classe AM.

4.5 Avaliação dos Algoritmos de escalonamento do Concentrador

Nesta secção são apresentados os resultados do estudo efectuado para a especificação e avaliação de desempenho dos algoritmos de escalonamento do Concentrador. Este bloco funcional, pertencente ao módulo de concentração do Concentrador, é responsável pelo escalonamento dos fluxos de pacotes CPS recebidos tendo como objectivo aumentar a eficiência de multiplexagem final sem degradar significativamente a QoS dos diversos fluxos de informação transportados.

Como foi referido na secção 4.1.2, este módulo recebe, em cada porta de entrada, um fluxo de pacotes CPS, associado aos fluxos de entrada de cada Adaptador de terminal. Cada pacote CPS, além da informação útil que transporta, tem associado um cabeçalho que identifica, entre outros, a classe de tráfego (UUI) e o tamanho do pacote (LI). Uma vez que se torna complexo implementar um mecanismo que entre em conta com o Atraso Máximo associado aos fluxos de classe AM, o critério de prioridade, implementado pelo Algoritmo de escalonamento, terá de minimizar o tempo de atendimento dos pacotes desta classe além de promover uma maior eficiência de multiplexagem e assim economizar os custos de transmissão relacionados com a largura de banda utilizada. Deste modo, podem ser implementados algoritmos de escalonamento utilizando diversas abordagens:

- **1ª abordagem**

Os pacotes que chegam a cada porta são colocados em FIFOs. Seguidamente, é calculada uma média deslizante de chegada de pacotes por FIFO. O Algoritmo atende os FIFOs com uma prioridade baseada na média deslizante calculada anteriormente.

- **2ª abordagem**

Os pacotes que chegam a cada porta são colocados numa memória de acesso aleatório (RAM, *Random Access Memory*). O algoritmo atende, por ordem de chegada, os pacotes de classe AM, em seguida os da classe D e, finalmente, os da classe EM.

- **3ª abordagem**

Os pacotes que chegam a cada porta são encaminhados para 3 filas, de acordo com a sua classe de tráfego. O algoritmo atende, em primeiro lugar as filas de pacotes de classe AM, em seguida os da classe D e, finalmente, os da classe EM.

Qualquer uma destas abordagens maximiza a ocupação da largura de banda, uma vez que o princípio de funcionamento do Concentrador o garante à partida.

A 1ª abordagem caracteriza-se por implementar um mecanismo de prioridade baseada no tamanho dos FIFOs. Embora este critério tivesse demonstrado um desempenho razoável nas simulações efectuadas com os algoritmos básicos do Adaptador de terminal (secção 4.4.1), não dá prioridade ao tráfego de classe AM e, desse modo, não garante a optimização do atraso para estes serviços. Por outro lado, o cálculo da média deslizante introduz complexidade na implementação deste algoritmo. A 2ª abordagem resolve o problema da prioridade relacionada com o tráfego de classe AM à custa da introdução de mecanismos de gestão de memória mais complexos. Na 3ª abordagem é implementada uma solução semelhante à anterior em que cada pacote, ao chegar a uma das portas do Concentrador, é encaminhado para uma fila específica em função da classe de tráfego que o pacote transporta. Deste modo, à custa da introdução de um pequeno atraso, consegue-se implementar um mecanismo que atende prioritariamente os pacotes de classe AM e assim minimize o atraso dos serviços associados a estas classes. A Figura 4-18 ilustra o mecanismo usado para o escalonamento de tráfego no Concentrador.

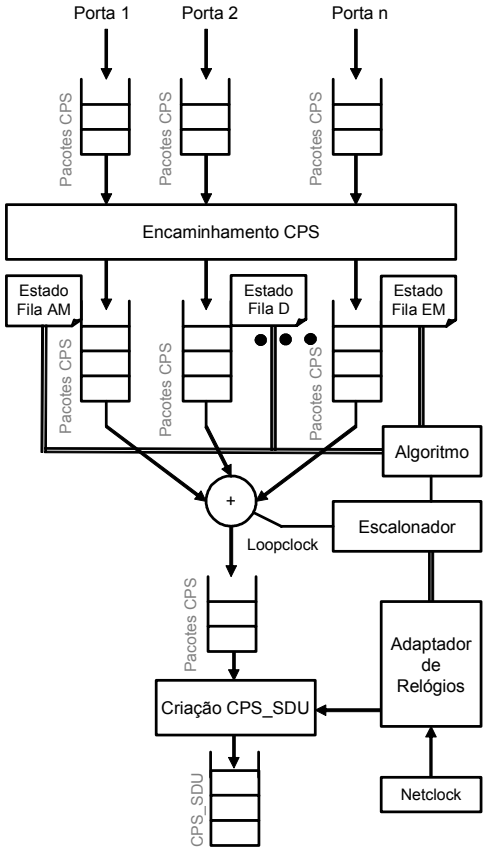


Figura 4-18: Diagrama funcional do mecanismo de escalonamento de tráfego do Concentrador, em que é usada a estrutura de multiplexagem dos pacotes CPS da camada AAL-2.

A estrutura funcional deste módulo é basicamente idêntica à da usada para o Adaptador de terminal. Este algoritmo implementa um mecanismo de prioridade mais simples uma vez que não tem de calcular os tempos de envio dos pacotes de classe AM. Por outro lado, neste algoritmo é necessário processar os campos LI, de cada pacote, de forma a determinar o seu comprimento. Assim, o escalonador usará o seguinte algoritmo:

1. Em cada instante dado pelo *Loopclock*, processa os pacotes das filas de classe AM,
2. Caso não existam pacotes nas filas AM, processa os pacotes das filas D,
3. Caso não existam pacotes nas filas D, processa os pacotes das filas EM.

Devido ao efeito de concentração, poderá ocorrer o fenómeno de *overflow* no FIFO de pacotes da classe EM, uma vez que estes pacotes têm a prioridade mínima de atendimento. Nestas situações, o algoritmo limita-se apenas a rejeitar os pacotes que excedam a capacidade do FIFO. De referir que esta classe de serviço foi definida (secção 2.1.3) para serviços que podem sofrer perdas ocasionais de informação.

Avaliação do algoritmo do Concentrador

Para efectuar a avaliação de desempenho do algoritmo proposto anteriormente, efectuou-se a simulação baseada no sistema de testes descrito na secção 4.1.2. Para cada gerador de pacotes será necessário definir as seguintes entradas:

- Intervalo médio entre pacotes CPS,
- Tamanho médio do pacote CPS.

A partir destes valores é possível relacionar o débito médio produzido por cada fonte, ou seja, por cada Adaptador de terminal:

$$DébitoMédio = \frac{Tamanho\ médio\ de\ pacote\ CPS}{Intervalo\ médio\ entre\ pacotes\ CPS} \quad (4\text{-}6)$$

Usando para o Tamanho médio de pacote CPS o valor de 39, obtido a partir dos resultados da simulação com o cenário 1 de avaliação do Adaptador de terminal (Tabela 4-6), efectuaram-se simulações, caracterizadas por se considerar, à entrada de cada porta do Concentrador, um débito médio aproximadamente igual ao produzido pelas fontes descritas no mesmo cenário, ou seja, 113kbit/s.

O valor do *Loopclock* é calculado a partir da equação 4-4. Como existem 8 portas, então o valor do *Loopclock* será de:

$$Loopclock = \frac{39 \times 8}{8 \times 113000} \approx 0{,}345ms \qquad (4\text{-}7)$$

No caso do Concentrador, um valor baixo do *Loopclock* não conduz a perda de eficiência, uma vez que o tamanho dos pacotes CPS não é alterado, ao contrário do Adaptador de terminal, em que a escolha do valor do *Loopclock* é baseada numa solução de compromisso entre a minimização do atraso e a optimização da eficiência de multiplexagem.

Quanto mais baixo é o valor do *Loopclock* menor será o atraso global sofrido por cada pacote, uma vez que a velocidade de processamento aumenta. Por outro lado, quanto menor for a quantidade de pacotes AM, que entram no Concentrador, relativamente aos pacotes das outras classes, menor será o seu tempo de espera no escalonador.

Assim, interessa avaliar o comportamento do Concentrador em função da percentagem total de pacotes AM que entram no Concentrador. A Tabela 4-11 apresenta os resultados obtidos na simulação do Algoritmo de escalonamento do Concentrador para diferentes valores da percentagem de pacotes AM relativamente às outras duas classes.

Tabela 4-11: Resultados da avaliação do Algoritmo de escalonamento do Concentrador.

Fontes AM (%)	**$Atraso_{AM}$ (ms)**	**$Atraso_{D}$ (ms)**	**$Atraso_{EM}$ (ms)**	**$FIFO_{AM}$ (octetos)**	**$FIFO_{D}$ (octetos)**	**$FIFO_{EM}$ (octetos)**
20	0,34	0,69	1,95	48	48	135
40	0,34	0,71	2,30	48	48	131
60	0,34	1,17	3,35	48	48	90
80	0,46	3,04	9,36	93	48	129
100	27,7	0	0	351	0	0

As simulações efectuadas usaram um *Loopclock* cujo valor é aproximadamente igual à soma dos valores médios dos débitos de entrada das portas do concentrador. Deste modo, não haverá perda de pacotes devido ao efeito de concentração. A partir de um valor do *Loopclock* acima do calculado na equação 4-7 a fila de pacotes de classe EM começará a crescer, até ocorrer *overflow*, com a consequente perda de pacotes desta classe. A perda de pacotes EM estará relacionada com a diferença entre a soma dos débitos médios de entrada e o débito de saída.

- **Análise de resultados**

  1. Verifica-se, através da observação da Tabela 4-11, que o atraso introduzido pelo Concentrador é bastante reduzido para as fontes de classe AM, excepto quando não existem fontes de outro tipo,

  2. O atraso introduzido depende de dois factores: da disciplina de escalonamento, implementada pelo algoritmo, e pelo valor do *Loopclock*.

  3. Os pressupostos que conduziram à especificação do Algoritmo de escalonamento do Concentrador foram totalmente satisfeitos na prática, verificando-se uma boa resposta à satisfação dos requisitos de atraso das diversas fontes de tráfego.

- **Conclusões**

  1. Como era de prever, verificou-se que o Algoritmo de escalonamento do Concentrador cumpre, com relativa facilidade, o objectivo de satisfazer os requisitos de atraso dos fluxos associados às fontes de classe AM,

  2. A prioridade dada aos fluxos de classe AM reflecte-se na introdução de um maior atraso aos fluxos das outras duas classes. Esse atraso é sempre maior nos fluxos de classe EM e depende essencialmente do nível de concentração e da largura de banda do canal de saída do concentrador,

  3. Em termos de eficiência de multiplexagem será sempre conveniente a utilização de um Concentrador uma vez que a largura de banda, associada aos fluxos agregados dos Adaptadores de terminal, pode não estar totalmente ocupada. Esta particularidade foi já ilustrada na Figura 3-9,

  4. Os valores absolutos dos atrasos dos vários fluxos estão directamente relacionados com o débito do tráfego agregado de saída. Deste modo, o desempenho do Concentrador, em termos de atraso, é condicionado pelo número de fontes e seus débitos médios e quantidade de Adaptadores de terminal utilizados,

  5. No caso do nível de concentração ser tal que implique a perda de pacotes, os fluxos de classe EM serão afectados. Neste caso, a Aplicação de controlo terá de possuir mecanismos que permitam lidar com esta situação.

4.6 Avaliação global da solução proposta

As secções anteriores apresentaram o estudo efectuado para avaliação dos Algoritmos de escalonamento do Adaptador de terminal e do Concentrador. Os resultados obtidos foram baseados na simulação destes módulos, usando um sistema de testes que consiste num conjunto de modelos simplificados dos módulos apresentados no capítulo 3.

Na secção 4.2 foi apresentada uma configuração de referência que permitiu evidenciar os atrasos de transmissão parciais ao longo dos vários segmentos de rede, entre os quais os do Adaptador de terminal e do Concentrador. Através desta separação, foi possível testar os vários Algoritmos de escalonamento em função da parcela de atraso atribuída a cada um destes módulos.

Os testes efectuados ao Adaptador de terminal foram baseados em dois cenários concretos destinados a avaliar, respectivamente, a capacidade de integração de fluxos de informação associados a serviços com requisitos de atraso diferentes, e a capacidade do sistema suportar um elevado número de fluxos de baixo débito duma forma eficiente.

Os resultados obtidos evidenciaram um bom desempenho do Algoritmo de escalonamento proposto, quer em termos do cumprimento dos atrasos impostos pelas fontes de entrada, quer em termos de eficiência de multiplexagem. Através da análise dos resultados é possível definir um conjunto de regras que permitem garantir um desempenho elevado do Algoritmo de escalonamento do Adaptador de terminal:

1. A utilização de um número elevado de fontes que não imponham requisitos temporais melhora o desempenho em termos de eficiência de multiplexagem,

2. O desempenho do escalonador em termos de satisfação dos atrasos parciais das fontes de tráfego de classe AM está sempre condicionado às características das fontes que estão envolvidas numa dada aplicação. Com o cenário 1 verificou-se que, se as fontes de classe AM tiverem um peso baixo em termos de débito médio total, o Algoritmo consegue cumprir facilmente o atraso imposto à entrada,

3. O cenário 1 permitiu também constatar que o escalonador consegue suportar eficazmente a integração de serviços com características diversas o que permite concluir que, sempre que possível, é desejável integrar, numa aplicação de controlo, serviços de débitos mais elevados e sem requisitos temporais com outros, de débito mais baixo com requisitos de atraso mais apertados,

4. O cenário 2 permitiu constatar o contrário, ou seja, haverá uma degradação da eficiência de multiplexagem à medida que o número de fontes de classe AM aumenta relativamente ao número de fontes das outras classes.

A avaliação efectuada foi baseada no modelo simplificado de testes desenvolvido para o efeito. As simplificações assumidas na especificação do sistema de testes permitiram tornar mais fácil a obtenção de resultados em relação a um sistema real e, ao mesmo tempo, definir modelos de fontes artificiais que permitiram também a criação de cenários específicos de teste, difíceis de conseguir com fontes reais.

Deste modo, é possível concluir que o desempenho do Algoritmo de escalonamento do Adaptador de terminal satisfaz os requisitos para os quais foi dimensionado, permitindo validar a especificação deste módulo, apresentada no capítulo anterior.

Para a avaliação do Concentrador, utilizou-se um cenário em que são gerados pacotes CPS de tamanho variável, em cada porta de entrada. São medidos os atrasos máximos dos fluxos das diversas classes de tráfego, assim como os tamanhos das filas correspondentes.

Em termos de complexidade, este algoritmo, é bastante mais simples que o desenvolvido para o Adaptador de terminal, uma vez que não usa o parâmetro “Atraso Máximo” associado aos fluxos de classe AM. Deste modo, os testes efectuados basearam-se em avaliar a variação dos atrasos dos fluxos das diferentes classes em função da percentagem de pacotes da classe AM, gerados à entrada do Concentrador.

Os resultados obtidos permitem comprovar o que já se esperava, ou seja, a melhoria da eficiência global de multiplexagem.

Em relação ao atraso, verificou-se que ele depende do valor do débito de saída do Concentrador, desde que as fontes de tráfego não sejam maioritariamente de classe AM. Deste modo, o atraso introduzido pelo Concentrador a fluxos de classe AM, com o cenário descrito, é bastante reduzido (≈0,4ms) face ao Atraso Máximo especificado (200ms), para um débito de saída inferior a 1Mbit/s (i.e. 8 X 113kbit/s).

Perante os resultados obtidos é possível concluir que a introdução de um Concentrador, numa arquitectura de controlo com vários Adaptadores de terminal, é bastante vantajosa, uma vez que permite agregar ainda mais o tráfego, proveniente dos diversos Adaptadores de terminal, reduzindo os custos associados ao tráfego contratado a um operador de telecomunicações da rede de suporte, à custa da introdução de um atraso pouco significativo.

Por estes motivos, é possível igualmente constatar um bom desempenho do Algoritmo de escalonamento do Concentrador, permitindo assim validar a especificação deste módulo, apresentada no capítulo 3.

Bibliografia

[Alm96] Almesberger, W.; “Linux ATM device driver interface Draft, version 0.1”, Laboratoire de Réseaux de Communication (LRC), EPFL, CH-1015 Lausanne, Suiça, Fevereiro de 1996.

[Alm97] Almesberger, W.; “ATM on Linux - The 3rd year”; Proceedings of 4th International Linux Kongress, Laboratoire de Réseaux de Communication at EPFL (École Polytechnique Fédérale de Lausanne, Suíça, Março de 1997.

[Amo01] Amorim, D.; Rochol, J., “Avaliação de Desempenho do Protocolo de Adaptação ATM tipo 2 (AAL-2)”, Proceedeings da XXVII Conferência Latino Americana de Informática - CLEI2001, Universidad de Los Andes, Mérida, Venezuela, 2001.

[Ant00] Anthonis, J., Ramon, H., Bardemaeker, J.; “Implementation of an Active Horizontal Suspension on a Spray Boom”; Transactions of the ASAE, Vol. 43(2), pp:213-220; 2000.

[Ari02] Arioli machines for fabric finishing product catalog; http://www.arioli.biz; 2005.

[ATM94] ATM Forum, AF-UNI-0010.002, “User to Network Interface Specification - Version 3.1”, Setembro de 1994.

[ATM97] ATM Forum, AF-VTOA-0078.000, “ATM Circuit Emulation Service – Intereoperability Specification”, Version 2.0, Janeiro de 1997.

[ATM99a] ATM Forum, AF-VTOA-0113.000, ATM Trunking using AAL-2 for Narrowband Services, Fevereiro de 1999.

[ATM99b] ATM Forum, AF-TM-0121.000, “Traffic Management Specification Version 4.1”, Março de 1999.

[Aud05] Audon Electronics UK electronics and software design; http://www.audon.co.uk; 2005.

[Bea95] Beaulah, S.; Zaid, S.; “Intelligent real-time fault diagnosis of greenhouse sensors”; IFAC Artificial Intelligence in Agriculture, Wageningen, pp.245-250, 1995.

[Bla95] Black, Uyless D., "ATM: Foundation for Broadband Networks", Prentice Hall Series in Advanced Communications Technologies, Maio de 1995.

[Boe99] Boed, V., “Networking and Integration of Facilities Automation Systems”, Capítulo 8. CRC Press LLC, Setembro de 1999.

[Cab95] Cabral, J.M., “Emulação de Circuitos em Redes ATM, Sistema de Teste de Algoritmos de Recuperação de Relógio”, Tese de Mestrado, Faculdade de Engenharia da Universidade do Porto, Junho de 1995.

[Car86] Carlson, A.B.; “Communication Systems – An Introduction to Signals and Noise in Electrical Communication”, 3ª Edição; McGraw Hill, 1986.

[Cru02] Cruvinel, P.; Minatel, R. “Image Processing in Automated Pattern Classification of Oranges”; Proceedings of the World congress of Computers in Agriculture and Natural Resources, Iguaçu Falls, Brazil, pp:56-61, 2002.

[Cun04] Cunha, J.,B.; "As novas tecnologias da informação e comunicação na indústria Agro-Pecuária"; 1º Congresso Luso-Brasileiro de Tecnologias de Informação e Comunicação na Agro-Pecuária; Junho, 2004.

[Dem89] Demers, A.; Keshav, S., Shenker, S.; “Design and Analysis of a Fair Queuing Algorithm.”; Proceedings of ACM SIGCOMM’89, Austin; Setembro de 1989.

[Dur04] Durresi, A.; Dash, D.; Anderson, B.L.; Kannana, R.; Kota, S.; Jain, R.; "Routing of Real-time Traffic in a Transformational Communications Architecture," Proc. of 2004 IEEE Aerospace Conference, Big Sky, MT, 7 a 12 de Março, 2004.

[Ent97] Entacher, K."A collection of selected pseudorandom number generators with linear structures", Technical Report 97-1, ACPC -- Austrian Center for Parallel Computation, University of Vienna, Austria, 1997.

[Fyn94] Fynn, R.; William L.; Bauerle and Warren L; “Implementing a decision and expert system model for individual nutrient selection”; Agricultural Systems, Vol. 44 (2), pp: 125-142, 1994.

[Fro04] Frost, A.R., French, A.P., Tillett, R.D.; “A vision guided robot for tracking a live, loosely constrained pig”, Computers and Electronics in Agriculture 44; pp. 93–106, Elsevier, Março de 2004.

[Goy99] Goyal, R; “Traffic Management for TCP/IP over Asynchronous Transfer Mode (ATM) Networks”; Tese de Doutoramento; The Ohio State University; EUA, 1999.

[Gre94] Greer, J.; Sarah F.; Greer, J.; Murray, J.; “Explaining and Justifying Recommendations in an Agriculture Decision Support System”; Computers and Electronics in Agricultural (11), pp 195-214, 1994.

[Hän94] Händel, R.; Huber, M. N.; Schröder, S.; “ATM Networks - Concepts, Protocols, Applications”, Siemens AG, Munich, Addison-Wesley Publishing Company, 1994.

[ITU91a] ITU-T, Rec. G.703, “Physical/electrical characteristics of hierarchical digital interfaces”, Genebra, Abril, 1991.

[ITU91b] ITU-T, Rec. I.321 - “B-ISDN Protocol Reference Model and its Application”, Genebra, Abril de 1991.

[ITU92] ITU-T, Rec. G.728 – “Coding of Speech at 16 kbit/s using Low-Delay Code Exited Linear Prediction”, Setembro de 1992.

[ITU93a] ITU-T, Rec. I.320, “ISDN Protocol Reference Model”, Genebra, Novembro de 1993.

[ITU93b] ITU-T, Rec. I.327 - “B-ISDN Network Functional Architecture”, Rev. 1, Genebra, 1993.

[ITU93c] ITU-T, Rec. I.362 - “BISDN ATM Adaptation Layer (AAL) Functional Description”, Rev. 1, Genebra, 1993.

[ITU93d] ITU-T, Rec. I.363 - “BISDN ATM Adaptation Layer (AAL) Specification”, Helsínquia, Março de 1993.

[ITU93e] ITU-T, Rec. I.411, “ISDN User-Network Interfaces – Reference Configurations”, Rev. 1, Genebra, 1993.

[ITU94] ITU-T, Rec. Q.2110, “B-ISDN ATM Adaptation Layer – Service Specific Connection Oriented Protocol (SSCOP)”, Julho de 1994.

[ITU96] ITU-T, Rec. G.131, “Control of talker echo”, Agosto, 1996.

[ITU96a] ITU-T, Rec. G.729 – “Coding of Speech at 8kbit/s using Conjugate Structure Algebraic Coded Excited Linear Prediction (CS-ACELP)”, Março de 1996.

[ITU96b] ITU-T, Rec. I363.1 – “B-ISDN ATM Adaptation Layer Specification: Type 1 AAL”, Agosto de 1996.

[ITU96c] ITU-T, Rec. I363.3 – “B-ISDN ATM Adaptation Layer Specification: Type 3/4 AAL”, Agosto de 1996.

[ITU96d] ITU-T, Rec. I363.5 – “B-ISDN ATM Adaptation Layer Specification: Type 5 AAL”, Agosto de 1996.

[ITU96e] ITU-T, Rec. I371 – “Traffic control and congestion control in B-ISDN”, Agosto de 1996.

[ITU96f] ITU-T, Rec. I356 – “B-ISDN ATM layer cell transfer performance”, Outubro de 1996.

[ITU97a] ITU-T, Rec. I371.1 – “Traffic control and congestion control in B-ISDN: conformance definitions for ABT and ABR”, Junho de 1997.

[ITU97b] ITU-T, Rec. I363.2 – “B-ISDN ATM Adaptation Layer Specification: Type 2 AAL”, Setembro de 1997.

[ITU98] ITU-T, Rec. I.366.1, “Segmentation and Reassembly Service Specific Convergence Sublayer for the AAL type 2”, Junho de 1998.

[ITU00] ITU-T, Rec. G.114, “One-way transmission time”, 18 Maio, 2000.

[Iye97] Iyer, J.; Jain, R., Munir, S.; Dixit, S.; "Performance of Compressed Voice Sources over VBR", ATM Forum/97-0608, Julho de 1997.

[Jia04] Jiahua, W., Tillett, R., McFarlane, N.; “Extracting the three-dimensional shape of live pigs using stereo photogrammetry”, Computers and Electronics in Agriculture 44; pp. 203–222, Elsevier, Maio de 2004.

[Ker88] Kernighan, B.; Ritchie, D.; “The C Programming Language”, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[Lei02] Lei, T.; “Sensor-Based Precision Chemical Application Systems.”; Proceedings of the World congress of Computers in Agriculture and Natural Resources, Iguaçu Falls, Brazil, pp:279-289, 2002.

[Lew69] Lewis, P.; Goodman, A. and Miler, A.; “A pseudo-random number generator for the System /360. IBM Syst. J., pp. 136-146, 1969.

[Man03] Manfredi, P.F.; Ratti, L.; Speziali, V.; "The readout of the LHC beam luminosity monitor: Accurate shower energy measurements at a 40MHz repetition rate",

Lawrence Berkeley National Laboratory. Paper LBNL-52654, http://repositories.cdlib.org/lbnl/LBNL-52654; Maio, 2003.

[Iye97] Iyer, J.; Jain, R., Munir, S.; Dixit, S.; "Performance of Compressed Voice Sources over VBR", ATM Forum/97-0608, Julho de 1997.

[Jam04] James, J.H.; Chen, B.; Garrinson, L.; “Implementing VoIP: A Voice Transmission Performance Progress Report”, IEEE Communications Magazine, pp 36-41, Julho de 2004.

[Kes97] Keshav, S.; “An Engineering Approach to Computer Networking”; pp. 209-246 Addison-Wesley, Janeiro de 1997.

[Lap95] Lapid, Y.; “Video Communications in B-ISDN”; Computer Networks Lab, Department of Electrical Engineering Technion, Israel, Fevereiro de 1995, http://www2.rad.com/networks/1994/atm/videoatm.htm.

[Liu99] Liu, Chunlei; Munir, Sohail; Jain, Raj; Dixit, Sudhir, "Packing Density of Voice Trunking using AAL-2", Proceedings IEEE Global Telecommunications Conference (GlobeCom99), Vol. 1(B), pp. 611-615, 5-9 de Dezembro, 1999, Rio de Janeiro, Brasil.

[Lue98] Luetchford, J.C.; Schreinemachers, M.; Arai, N.M.; “Applications of ATM In Global Networks”, IEEE Communications Magazine, pp 104-109, Agosto de 1998.

[McC93] McClure, J.; Tomasko, M.; Collison, C.; “BEE AWARE, an expert system for honey bee diseases, parasites, pests and predators”; Computers and electronics in agriculture, vol.9, pp111-112, 1993.

[McD95] McDysan, D. E.; Spohn, D. L., “ATM Theory and Application”, McGraw-Hill Series on Computer Communications, 1995.

[Mcl97] Mcloughlin, Mike; O'Neil, John. A Management Briefing on Adapting Voice for ATM Networks, An AAL-2 Tutorial. General DataComm, 1997.

[Moo98] Moondhra, V.; “Implementation And Performance Analysis of ATM Adaptation Layer Type 2”; Tese de Mestrado, Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas, EUA, Maio, 1998.

[Mor00] Morimoto T.; Hashimoto, Y.; “AII approaches to identification and control of total plant production systems”; Control Eng. Practice, Vol. 8 (5), pp. 555-567, 2000.

[Par94] Parekh, A.; Gallager, R.; “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks – The Multiple Node Case.”; IEEE/ACM Transactions on Networking, pp. 137-150, Abril 1994.

[Pee96] Pee, V.; Berckmans, D.; “Quality of modeling plant responses for environment control purposes. Information and communication technology applications in agriculture: state of the art and future perspectives.”; Proceedings of the Congress on ICT applications in Agriculture, pp. 359-367, 1996.

[Per03] Pereira, M.R., Amorim, C.L., Castro, M.C.; “Tutorial sobre Redes de Sensores”; Cadernos do IME (Instituto de Matemática e Estatística), Universidade do Estado do Rio de Janeiro, Série Informática : 14, Junho de 2003.

[Pix02] Pixord 100/120 Network Camera User’s Manual, PIXORD Corporation, Rev. 1.0, Software Version 1.14, Janeiro de 2002.

[PRO94] PROFIBUS Nutzerorganisation e.V., Haid-und-Neu-Str. 7,. Implementation Guide to DIN 19245 Part 1, Karlsruhe, Alemanha, Agosto de 1994.

[Pre92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, “Numerical Recipes in C: The Art of Scientific Computing”, 2nd ed., Cambridge University Press, Cambridge, 1992.

[Pry95] Prycker, M., “Asynchronous Transfer Mode - Solution for Broadband ISDN”, Prentice Hall, 3ª Edição, 1995.

[Set93] Manual da função *Setitimer*, de acordo com os standards SVr4 e 4.4BSD, Linux 0.99.11, 1993.

[Shi96] Shimizu, H., Yamazaki, M.; “Generalized system for plant growth analysis using infrared LED. Plant production in closed ecosystems: automation, culture and environment.”; Acta Horticulturae 440, pp:446-451, 1996.

[Tov99] Tovar, E.M.M., “Supporting Real - Time Communications with Standard Factory - Floor Networks”, Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto, Julho de 1999.

[Win98] Winstanley, S.B.; “Quality of Service over ATM Networks”; Tese de Doutoramento; Department of Electronic Engineering, Queen Mary and Westfield College, University of London, 1998.

Anexo A

Listagens dos programas desenvolvidos

Funções que simulam determinadas fontes de tráfego

```
/* random-gauss.h
** Declaração das funções do gerador de números aleatórios.
*/

#ifndef RANDOM_GAUSS_H
#define RANDOM_GAUSS_H

extern double Raw();
extern void SetSeed(double s);
extern double Density(double x);
extern void Normal(int sym);
extern double RandomNext();
extern double Next();

#endif /* RANDOM_GAUSS_H */
```

```
/* random-gauss.c
** Funções do gerador de números aleatórios.
*/

#include <stdio.h>
#include <math.h>
#include "random-gauss.h"

double seed;
double xi,sx[60],sfx[60];
double Buffer[128];

/* Raw: Gera um número aleatório uniformemente distribuído */
double Raw()
{
      /* m = 2147483647 = 2^31 - 1; a = 16807;
      ** 127773 = m div a; 2836 = m mod a */

      long iseed,hi,lo;

      iseed=(long)seed;
      hi=iseed/127773L;
      lo=iseed-hi*127773L;
      iseed=16807*lo-2836*hi;
      if(iseed<=0)iseed+=2147483647L;
      seed=(double)iseed;
      return seed*4.656612875e-10;
}

/* Estabelece a semente do gerador uniforme de números aleatórios.
** s deve ter um valor entre 0 e 1.
*/
void SetSeed(double s)
{
      int i;

      if(s>=1.0 || s<=0.0){
            fprintf(stderr,"SetSeed: seed out of range. Must be
between 0 and 1\n");
            exit(1);
      }
      seed=(int)(s*2147483648.0);
      for(i=0;i<128;i++)
            Buffer[i]=Raw();
}

/* Density: Calcula a densidade da distribuição normal. */
double Density(double x)  // normal density
{
      return (fabs(x)>8.0)?0:0.398942280*exp(-x*x/2);
}

/* Normal: Inicializa os vectores para a distribuição normal. */
void Normal(int sym)
{
      int i;
      double sxi,inc;
      double f1;
```

```
	sxi=0.0;
	inc=sym?0.01:0.02;
	for(i=0;i<60;i++){
		sx[i]=sxi;
		f1=Density(sxi);
		sfx[i]=f1;
		if(f1<=0.0)break;
		sxi+=inc/f1;
	}
	if(i==60){
		fprintf(stderr,"Build: area too large\n");
		exit(1);
	}
	if(i<50){
		fprintf(stderr,"Build: area too small\n");
		exit(1);
	}
	xi=sym?2*i:i;
}

/* RandomNext: Função a ser chamada por Next(). */
double RandomNext()
{
	int i;
	double f;

	if(!seed){
		fprintf(stderr,"RandomNext: Random number generator not
initialised\n");
		exit(1);
	}
	i=(int)(Raw()*128);	/* 0 <= i < 128 */
	f=Buffer[i];
	Buffer[i]=Raw();
	return f;
}

/* Next: Gera o próximo número aleatório segundo a distribuição
normal. */
double Next()
{
	double s,ak,y,r1,sxi;
	int ir;

	do{
		s=1.0;
		r1=RandomNext();
		if(r1>0.5){
			s=-1.0;
			r1=1.0-r1;
		}
		ir=(int)(r1*xi);
		sxi=sx[ir];
		ak=sxi+(sx[ir+1]-sxi)*RandomNext();
		y=sfx[ir]*RandomNext();
	}
	while(y>=sfx[ir+1] && y>=Density(ak));
	return s*ak;
}
```

```
/* input1.c
** Gerador de tráfego a 2kBytes/s (áudio). */
** Em simulação funciona a 20 bytes/s.
*/

#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>
#include <time.h>
#include <unistd.h>

#define PERIOD 5000000              /* gera 100 bytes de 5 em 5
segundos */
#define NBYTES 100
#define SIMULATIONTIME 6000         /* tempo de simulação (seg) */

int fd;                             /* descritor do ficheiro de saída
*/
char line[BUFSIZ];
unsigned long long bytestotal;      /* número total de bytes escritos
*/

/* GenerateTrafic: Gera 100 bytea a cada interrupção do relógio. */
void GenerateTrafic(int s)
{
      write(fd,line,NBYTES);  /* escreve 100 bytes no dispositivo de
saída */
      bytestotal+=NBYTES;
}

main(int argc, char **argv)
{
      struct itimerval itime;
      time_t t1,t2;
      int i;

      bytestotal=0;
      if(argc!=3){
            fprintf(stderr,"%s usage: %s <outfile>
<char>\n",argv[0],argv[0]);
            exit(1);
      }

      fd=open(argv[1],O_WRONLY|O_CREAT,0644);
      if(!fd){
            perror(argv[1]);
            exit(1);
      }
      for(i=0;i<NBYTES;i++){
            line[i]=argv[2][0];
      }
      /* Programa a interrupção do timer. */
      itime.it_interval.tv_sec=0;
      itime.it_interval.tv_usec=PERIOD;
      itime.it_value.tv_sec=0;
      itime.it_value.tv_usec=PERIOD;
```

```
	setitimer(ITIMER_REAL,&itime,NULL);
	signal(SIGALRM,GenerateTrafic);

	t1=time(NULL);

	/* No ciclo principal, apenas é controlado o tempo de simulação.
*/
	do{
		sleep(1);
		t2=time(NULL);
	}while((t2-t1)<SIMULATIONTIME);
	close(fd);
	printf("Input1: %ld bytes\n",bytestotal);
	return 0;
}
```

```
/* input2.c
** Gerador de tráfego a 100Bytes/s (10 bytes de 100 em 100
milissegundos).
** Sensor de temperatura  humidade ou luminosidade.
** Em simulação funciona a 1 byte/s.
*/

#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>
#include <time.h>
#include <unistd.h>

#define PERIOD 10000000             /* gera 10 bytes de 10 em 10
segundos */
#define NBYTES 10
#define SIMULATIONTIME 6000         /* tempo de simulação (seg) */

int fd;                             /* descritor do ficheiro de saída
*/
char line[BUFSIZ];
unsigned long long bytestotal;      /* numero total de bytes escritos
*/

/* GenerateTrafic: Gera 10 bytea a cada interrupção do relógio. */
void GenerateTrafic(int s)
{
      write(fd,line,NBYTES);        /* escreve 10 bytes no ficheiro de
saída */
      bytestotal+=NBYTES;
}

main(int argc, char **argv)
{
      struct itimerval itime;
      time_t t1,t2;
      int i;

      bytestotal=0;
      if(argc!=3){
            fprintf(stderr,"%s usage: %s <outfile>
<char>\n",argv[0],argv[0]);
            exit(1);
      }

      fd=open(argv[1],O_WRONLY|O_CREAT,0644);
      if(!fd){
            perror(argv[1]);
            exit(1);
      }
      for(i=0;i<NBYTES;i++){
            line[i]=argv[2][0];
      }
      itime.it_interval.tv_sec=0;
      itime.it_interval.tv_usec=PERIOD;
      itime.it_value.tv_sec=0;
```

```
	itime.it_value.tv_usec=PERIOD;
	setitimer(ITIMER_REAL,&itime,NULL);
	signal(SIGALRM,GenerateTrafic);

	t1=time(NULL);
	do{
		sleep(1);
		t2=time(NULL);
	}while((t2-t1)<SIMULATIONTIME);
	close(fd);
	printf("Input2: %ld bytes\n",bytestotal);
	return 0;
}
```

```
/* input3.c
** Gerador de tráfego aleatório a 200Bytes/s.
** Média igual a 2kBytes em 10 segundos.
** Serviço de dados.
** Em simulação funciona a 20bytes/s.
*/

#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>
#include <time.h>
#include <unistd.h>
#include "random-gauss.h"

#define PERIODMED 10000000          /* período médio */
#define PERIODVAR 5000000           /* variância */
#define NBYTES 200
#define SIMULATIONTIME 6000         /* tempo de simulação (seg) */

int fd;                             /* descritor do ficheiro de saída
*/
char line[BUFSIZ];
unsigned long long bytestotal;      /* número total de bytes escritos
*/

/* GenerateTrafic: Gera 10 bytes a cada interrupção do relógio. */
void GenerateTrafic(int s)
{
      struct itimerval itime;
      int i;

      /* Programa a interrupção do relógio seguinte com um instante
      aleatório
      ** que segue uma distribuição normal. */
      i=(int)(Next()*PERIODVAR+PERIODMED);
      itime.it_interval.tv_sec=0;
      itime.it_interval.tv_usec=i;
      itime.it_value.tv_sec=0;
      itime.it_value.tv_usec=i;
      setitimer(ITIMER_REAL,&itime,NULL);
      write(fd,line,NBYTES);        /* escreve 10 Bytes no ficheiro de
saida */
      bytestotal+=NBYTES;
}

main(int argc, char **argv)
{
      struct itimerval itime;
      time_t t1,t2;
      int i;

      bytestotal=0;
      if(argc!=3){
            fprintf(stderr,"%s usage: %s <outfile>
<char>\n",argv[0],argv[0]);
```

```
            exit(1);
      }

      fd=open(argv[1],O_WRONLY|O_CREAT,0644);
      if(!fd){
            perror(argv[1]);
            exit(1);
      }
      for(i=0;i<NBYTES;i++){
            line[i]=argv[2][0];
      }

      SetSeed(0.5);
      Normal(1);

      line[i]=0;
      i=(int)(Next()*PERIODVAR+PERIODMED);
      itime.it_interval.tv_sec=0;
      itime.it_interval.tv_usec=i;
      itime.it_value.tv_sec=0;
      itime.it_value.tv_usec=i;
      setitimer(ITIMER_REAL,&itime,NULL);
      signal(SIGALRM,GenerateTrafic);

      t1=time(NULL);
      do{
            sleep(1);
            t2=time(NULL);
      }while((t2-t1)<SIMULATIONTIME);
      close(fd);
      printf("Input3: %ld bytes\n",bytestotal);
      return 0;
}
```

```
/* input4.c
** Gerador de tráfego a 10Byte/s.
** Trafego aleatório de tempo real (interruptores).
** Distribuição uniforme com média igual a 10 bytes em 1 segundos.
** Em simulação funciona a 10 bytes em 100 segundos (0.1B/seg).
*/

#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>
#include <time.h>
#include <unistd.h>

#define PERIODMIN 1000000           /* período mínimo 1s */
#define PERIODMAX 199000000.0       /* período máximo: 200s-1s=199s */
#define NBYTES 10
#define SIMULATIONTIME 6000         /* tempo de simulação (seg) */

int fd;                             /* descritor do ficheiro de saída
*/
char line[BUFSIZ];
unsigned long long bytestotal;      /* número total de bytes escritos
*/

void GenerateTrafic(int s)
{
      struct itimerval itime;
      int i;

      i=PERIODMIN+(int)(PERIODMAX*rand()/(RAND_MAX+1.0));
      itime.it_interval.tv_sec=0;
      itime.it_interval.tv_usec=i;
      itime.it_value.tv_sec=0;
      itime.it_value.tv_usec=i;
      setitimer(ITIMER_REAL,&itime,NULL);
      write(fd,line,NBYTES);  /* escreve 10 Bytes no ficheiro de saida
*/
      bytestotal+=NBYTES;
}

main(int argc, char **argv)
{
      struct itimerval itime;
      time_t t1,t2;
      int i;

      bytestotal=0;
      if(argc!=3){
            fprintf(stderr,"%s usage: %s <outfile>
<char>\n",argv[0],argv[0]);
            exit(1);
      }

      fd=open(argv[1],O_WRONLY|O_CREAT,0644);
      if(!fd){
```

```
            perror(argv[1]);
            exit(1);
      }
      for(i=0;i<NBYTES;i++){
            line[i]=argv[2][0];
      }
      line[i]=0;
      i=PERIODMIN+(int)(PERIODMAX*rand()/(RAND_MAX+1.0));
      itime.it_interval.tv_sec=0;
      itime.it_interval.tv_usec=i;
      itime.it_value.tv_sec=0;
      itime.it_value.tv_usec=i;
      setitimer(ITIMER_REAL,&itime,NULL);
      signal(SIGALRM,GenerateTrafic);

      t1=time(NULL);
      do{
            sleep(1);
            t2=time(NULL);
      }while((t2-t1)<SIMULATIONTIME);
      close(fd);
      printf("Input5: %ld bytes\n",bytestotal);
      return 0;
}
```

```
/* input5.c
** Gerador de tráfego a 10kBytes/s.
** Tráfego de vídeo aleatório.
** Distribuição normal com média igual a 100kBytes em 10 segundos.
** Em simulação funciona a 1kByte em 10 segundos (100B/seg).
*/

#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>
#include <time.h>
#include <unistd.h>
#include "random-gauss.h"

#define PERIODMED 5000000           /* período médio */
#define PERIODVAR 2500000           /* variância */
#define NBYTES 500
#define SIMULATIONTIME 6000         /* tempo de simulação (seg) */

int fd;                             /* descritor do ficheiro de saída
*/
char line[BUFSIZ];
unsigned long long bytestotal;      /* numero total de bytes escritos
*/

void GenerateTrafic(int s)
{
      struct itimerval itime;
      int i;

      i=(int)(Next()*PERIODVAR+PERIODMED);
      itime.it_interval.tv_sec=0;
      itime.it_interval.tv_usec=i;
      itime.it_value.tv_sec=0;
      itime.it_value.tv_usec=i;
      setitimer(ITIMER_REAL,&itime,NULL);
      write(fd,line,NBYTES);        /* escreve 500 Bytes no ficheiro
de saída */
      bytestotal+=NBYTES;
}

main(int argc, char **argv)
{
      struct itimerval itime;
      time_t t1,t2;
      int i;

      bytestotal=0;
      if(argc!=3){
            fprintf(stderr,"%s usage: %s <outfile>
<char>\n",argv[0],argv[0]);
            exit(1);
      }

      fd=open(argv[1],O_WRONLY|O_CREAT,0644);
```

```
        if(!fd){
                perror(argv[1]);
                exit(1);
        }
        for(i=0;i<NBYTES;i++){
                line[i]=argv[2][0];
        }
        line[i]=0;

        SetSeed(0.5);
        Normal(1);

        i=(int)(Next()*PERIODVAR+PERIODMED);
        itime.it_interval.tv_sec=0;
        itime.it_interval.tv_usec=i;
        itime.it_value.tv_sec=0;
        itime.it_value.tv_usec=i;
        setitimer(ITIMER_REAL,&itime,NULL);
        signal(SIGALRM,GenerateTrafic);

        t1=time(NULL);
        do{
                sleep(1);
                t2=time(NULL);
        }while((t2-t1)<SIMULATIONTIME);
        close(fd);
        printf("Input6: %ld bytes\n",bytestotal);
        return 0;
}
```

Código usado para criar os pacotes CPS a partir do tráfego de entrada

```
/* CPS.h
** Definição das estruturas de dados usadas para criar os CPSs
*/
#ifndef CPS_H_
#define CPS_H_

#include <stdio.h>
#define     NINPUTS     25    /* número de entradas */
#define     LIMAX       45    /* li máximo */

long long mytime;
unsigned long long nbyteswrited,totaldelay;

#define LOOPTIME 16
#define TIMEPERIOD 10000            /* 10 milissegundos */
#define SIMULATIONTIME  6120         /* 6120 segundos */

typedef enum{     /* Classes de tráfego */
AM, DT, EM        /* AM – Com atraso máximo. DT – Classe de dados */
}dataclass;       /* EM – Classe de esforço mínimo */

struct InputFifo{
      char name[BUFSIZ];
      char data[BUFSIZ];
      unsigned long long datatime[BUFSIZ];
      int fifofd;
      int r,w;
      int cid;
      int uui;
      long long atrasomaximo; /* tempo máximo permitido de espera da
fila. */
      dataclass classe;       /* classe de tráfego */
      unsigned int maxfifolen;      /* apenas para calculo de
estatísticas */
      unsigned long long delaymax;  /* apenas para calculo de
estatísticas */
};

struct OutputFifo{
      char name[BUFSIZ];
      int fifofd;
};

struct InputFifo finput[NINPUTS];
struct OutputFifo fout;

extern int decide(struct InputFifo *finput);
long long ReadInputFifoDate(struct InputFifo *f);

#endif /* CPS.h */
```

```
/* CPS.c
** Decide qual é a fila de entrada que deve ser atendida e cria as
células CPS
*/

#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/time.h>
#include <signal.h>
#include "CPS.h"

/* Calcula o comprimento da fila de entrada apontada por f */
unsigned int InputFifoLen(struct InputFifo *f)
{
      if(f->w>f->r)return(f->w-f->r-1);
      else return(BUFSIZ-f->r+f->w-1);
}

/* WriteInputFifo: Coloca o octeto c, proveniente de um dos
dispositivos
** de entrada, na fila f
*/
void WriteInputFifo(char c,struct InputFifo *f)
{
      unsigned int len;
      if(f->w==f->r){
            printf("Overflow...\n");
            return;
      }
      else{
            f->data[f->w]=c;
            f->datatime[f->w]=mytime;
            f->w++;
            f->w=f->w&(BUFSIZ-1);
            len=InputFifoLen(f);
            if(len>f->maxfifolen)f->maxfifolen=len;
      }
}

/* WriteOutputFifo: Escreve o octeto c na fila de saída f */
void WriteOutputFifo(char c,struct OutputFifo *f)
{
      int n;

      n=write(f->fifofd,&c,1);
      if(n==0){
            perror("escreve-fifo output:");
      }
}

/* ReadInputFifoDate: Lê o instante em que o octeto mais antigo chegou
à
** fila f */
long long ReadInputFifoDate(struct InputFifo *f)
{
      return f->datatime[((f->r)+1)&(BUFSIZ-1)];
}
```

```
/* ReadInputFifo: Retira um octeto da fila de entrada f */
char ReadInputFifo(struct InputFifo *f)
{
      char c;
      unsigned long long delay;

      f->r++;
      f->r=f->r&(BUFSIZ-1);
      if(f->r==f->w){
            printf("Underflow...\n");
            f->r--;
            return 0;
      }
      else{
            c=f->data[f->r];
            delay=mytime-f->datatime[f->r];
            totaldelay+=delay;
            if(delay>f->delaymax)f->delaymax=delay;
            nbyteswrited++;
            return c;
      }
}
/* CalcHec: Recebe o Cabeçalho sem o HEC e calcula-o.
** Recebe o valor Hexadecimal do CID + LI + UUI
*/
int CalcHec(int x)
{
      int y,i;
      int hec0=0,hec1=0,hec2=0,hec3=0,hec4=0,temp4=0;
      int cps=0;

      for (i=1;i<20;i++)
      {
            y=(x>>(19-i)&0x01); /* Extrai o bit da direita (um a um)
*/
            /* Efectua a divisão de polinómios */
            hec4=hec3;
            hec3=hec2;
            hec2=hec1^temp4^y;
            hec1=hec0;
            hec0=temp4^y;
            temp4=hec4;
            cps=cps+(y<<(24-i));
      }
      cps=cps+(hec0*1+hec1*2+hec2*4+hec3*8+hec4*16);
      return cps;
}

/* CPSPacketHeader: Cria a célula CPS a partir do cid, li, uui e dos
octetos
** de informação disponíveis.
*/
void CPSPacketHeader(char cid, char li,char uui,char *cps_info,
struct OutputFifo *f)
{
      int i;
      union{
            int i;
            char c[4];
```

```
      }u;

      u.i=((cid&0xff)<<11)|((li&0x3f)<<5)|(uui&0x1f);
      u.i=CalcHec(u.i);

      /* Escreve o cabeçalho na fila de saída */
      WriteOutputFifo(u.c[2],f);
      WriteOutputFifo(u.c[1],f);
      WriteOutputFifo(u.c[0],f);
      /* Escreve os octetos de informação útil na fila de saída */
      for(i=0;i<=li;i++){
            WriteOutputFifo(cps_info[i],f);
      }
      /* printf("Escreveu %d bytes\n",li); */
}

/* SetConfigFile: Estabelece a configuração do tipo de serviço e dos
seus
** requisitos para cada um dos serviços.
** Neste exemplo existem 20 serviços do tipo AM e 5 serviços do tipo
DT.
*/
void SetConfigFile(struct InputFifo *finput,struct OutputFifo *fout)
{
      int i;

      for(i=0;i<20;i++){
            finput[i].r=BUFSIZ-1;
            finput[i].w=0;
            finput[i].cid=2;
            finput[i].uui=0x80;
            sprintf(finput[i].name,"../fifos/fifoin%d",i+1);
            finput[i].atrasomaximo=200;   /* atraso máximo do serviço
*/
            finput[i].maxfifolen=0;
            finput[i].classe=AM;          /* Classe de tráfego */
            finput[i].delaymax=0;
      }
      for(i=20;i<25;i++){
            finput[i].r=BUFSIZ-1;
            finput[i].w=0;
            finput[i].cid=3;
            finput[i].uui=0x80;
            sprintf(finput[i].name,"../fifos/fifoin%d",i+1);
            finput[i].atrasomaximo=0;
            finput[i].maxfifolen=0;
            finput[i].classe=DT;
            finput[i].delaymax=0;
      }

      /* strcpy(fout->name,"../fifos/fifocps"); */
      strcpy(fout->name,"cps.out");
}

/* readinputs: Lê os caracteres provenientes das entradas e coloca-os
nas filas
** correspondentes.
*/
void readinputs(struct InputFifo *finput)
```

```
{
      int i,j,len;
      char text[BUFSIZ];

      for(i=0;i<NINPUTS;i++){
            len=read(finput[i].fifofd,text,BUFSIZ);
            for(j=0;j<len;j++)WriteInputFifo(text[j],&finput[i]);
      }
}

/* writeoutput: Cria a célula CPS e coloca-a na fila de saída. */
void writeoutput(struct InputFifo *finput, struct OutputFifo *fout)
{
      int i,j;
      char text[BUFSIZ];
      char uui=0;

      i=decide(finput);
      if(i<0)return;
      j=0;
      while(j<45 && InputFifoLen(&finput[i])>0){
            text[j]=ReadInputFifo(&finput[i]);
            j++;
      }
      CPSPacketHeader(finput[i].cid,j-1,uui,text,fout);
}

/* IncrTimer: A cada interrupção do relógio incrementa o tiler
interno.
** Quando tiler passado um tempo igual ao LOOPTIME, produz uma célula
CPS.
*/
void IncrTimer(int s)
{
      mytime++;
      if(!(mytime%LOOPTIME))
            writeoutput(finput,&fout);
}

main()
{
      struct itimerval timer_CU;
      time_t t1,t2;
      int i;

      /* Estabelece a configuração dos serviços a atender. */
      SetConfigFile(finput,&fout);
      mytime=0;
      nbyteswrited=0;
      totaldelay=0;

      /* Abre os dispositivos de entrada. */
      for(i=0;i<NINPUTS;i++){
            finput[i].fifofd=open(finput[i].name,O_RDONLY|O_NDELAY);
            if(finput[i].fifofd==-1){
                  perror("open input fifo");
                  exit(1);
            }
      }
```

```
	/* Abre o dispositivo de saída. */
	fout.fifofd=open(fout.name,O_WRONLY|O_CREAT,0644);
	if(fout.fifofd==-1){
		perror("open out fifo");
		exit(1);
	}

	/* Programa a interrupção do relógio para ser activada a cada
	** TIMEPERIOD e chamar a função IncrTimer. */
	signal(SIGALRM,IncrTimer);
	timer_CU.it_interval.tv_sec=0;
	timer_CU.it_interval.tv_usec=TIMEPERIOD;
	timer_CU.it_value.tv_sec=0;
	timer_CU.it_value.tv_usec=TIMEPERIOD;
	setitimer(ITIMER_REAL,&timer_CU,NULL);

	t1=time(NULL);

	/* Durante todo o período de simulação lê os octetos das
entradas para
	** as filas de espera */
	do{
		readinputs(finput);
		t2=time(NULL);
	}while((t2-t1)<SIMULATIONTIME);

	/* Apresenta as estatísticas. */
	for(i=0;i<NINPUTS;i++){
		printf("fifo %d: delaymax=%lld\n",i+1,finput[i].delaymax);
	}
	printf("totaldelay=%lld\n",totaldelay);
	printf("nbyteswrited=%lld
delaymed=%f\n",nbyteswrited,(double)totaldelay/(double)nbyteswrited);
	for(i=0;i<NINPUTS;i++)
		printf("fifo %d:
maxfifolen=%d\n",i+1,finput[i].maxfifolen);
	return 0;
}
```

```
/* decide.c
** Algoritmos que tomam a decisão sobre qual deve ser a fila de
entrada a
** ser atendida.
*/
#include <stdlib.h>
#include "CPS.h"

/* decidealgorithm1: Decide-se pela fila de maior comprimento.*/
int decidealgorithm1(struct InputFifo *finput)
{
      int lenmax,nmaior;
      int i;

      nmaior=-1;
      lenmax=InputFifoLen(&finput[0]);
      if(lenmax)nmaior=0;
      for(i=1;i<NINPUTS;i++){
            if(InputFifoLen(&finput[i])>lenmax){
                  lenmax=InputFifoLen(&finput[i]);
                  nmaior=i;
            }
      }
      return nmaior;
}

/* decidealgorithm2: Decide-se pela fila que tem os octetos em espera
há
** mais tempo.
*/
int decidealgorithm2(struct InputFifo *finput)
{
      unsigned long long datemax;
      int i,nmaior;

      nmaior=-1;
      datemax=ReadInputFifoDate(&finput[0]);
      if(InputFifoLen(&finput[0]))nmaior=0;
      for(i=1;i<NINPUTS;i++){
            if(ReadInputFifoDate(&finput[i])>datemax){
                  datemax=ReadInputFifoDate(&finput[i]);
                  if(InputFifoLen(&finput[i]))nmaior=i;
            }
      }
      return nmaior;
}

/* decidealgorithm3: Decisão sequencial */
int decidealgorithm3(struct InputFifo *finput)
{
      static int i=NINPUTS-1;

      if(i==NINPUTS-1)i=0;
      else i++;
      if(InputFifoLen(&finput[i]))return i;
      else return -1;
}

/* decidealgorithm4: toma a decisão aleatoriamente */
```

```
int decidealgorithm4(struct InputFifo *finput)
{
      int i;
      i=(int)(4.0*rand()/(RAND_MAX+1.0));
      if(InputFifoLen(&finput[i]))return i;
      else return -1;
}

/* decidealgorithm5: toma a decisão de acordo com a classe de serviço
e
** atraso máximo permitido.
*/
int decidealgorithm5(struct InputFifo *finput)
{
      int lenmax=0,nmaior=-1;
      int lenmaxem=0,nmaiorem=-1;
      long long maioratraso=0,mareg;
      int nmaioratraso=-1;
      int i;

      for(i=0;i<NINPUTS;i++){
            if(InputFifoLen(&finput[i])){
                  switch(finput[i].classe){
                  case AM:
                        mareg=mytime-
(ReadInputFifoDate(&finput[i])+finput[i].atrasomaximo);
                        if(mareg>maioratraso){
                              maioratraso=mareg;
                              nmaioratraso=i;
                        }
                  case DT:
                        if(InputFifoLen(&finput[i])>lenmax){
                              lenmax=InputFifoLen(&finput[i]);
                              nmaior=i;
                        }
                        break;
                  default:
                        if(InputFifoLen(&finput[i])>lenmaxem){
                              lenmaxem=InputFifoLen(&finput[i]);
                              nmaiorem=i;
                        }
                        break;
                  }
            }
      }
      if(nmaioratraso>=0)return nmaioratraso;
      if(nmaior>=0)return nmaior;
      return nmaiorem;
}

/* decide: escolhe a fila de entrada que deve ser atendida */
int decide(struct InputFifo *finput)
{
      long long mareg;
      int i;

      i=decidealgorithm5(finput);
      return i;
}
```

Código usado para criar os pacotes CPS_SDU a partir das estruturas CPS

```
/* osf.c
** Cria os pacotes CPS_SDU a partir das estruturas CPS
*/

#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/time.h>
#include <signal.h>

#define PDULEN 45
#define TIMEMAX 80
char buffer_saida[PDULEN+1];
char buffer_entrada[BUFSIZ];
int bytes_no_buffer;
int fifofdout;

void bufferdummy()
{
      int i;
      for(i=0;i<PDULEN+1;i++)
            buffer_saida[i]=0;
}

void printbuffer()
{
      static int n=0;
      int i;
      for(i=0;i<PDULEN+1;i++)
            printf("%d - [0x%x]\n",i,(int)buffer_saida[i]&0xff);
      if(!write(fifofdout,buffer_saida,PDULEN+1)){
            perror("Write out buffer");
            exit(1);
      }
      if(n++==10){
            close(fifofdout);
            exit(0);
      }
}

char calculaSTF(int osf,int sn)
{
      int result;
      int p=1;
      int i;

      result=osf<<1|sn;
      for(i=0;i<7;i++){
            p^=((result>>i)&0x01);
      }
      return(result<<1|p);
}

/* calculaCPS: A cada interrupção do relógio gera uma estrutura OSF.
*/
```

```
void calculaCPS_PDU(int s)
{
      static int sn=0;
      static int split=-1;
      static int part=0;
      static int i=0;
      char c;
      int n;
      char *line;

      line=buffer_entrada;
      n=bytes_no_buffer;
      if(n==0)part=0;
      if(i==0){
            bufferdummy();
            c=calculaSTF(part,sn);
            buffer_saida[i++]=c;
            sn=(sn+1)%2;
      }
      if(split==-1)split=3+(int)(line[1]>>2)&0x3f;
      while(n && i<=PDULEN){
            buffer_saida[i]=*line;
            if(split>0)split--;
            else{
                  c=*(line+2);
                  if(n>=2)split=3+(c>>2)&0x3f;
            }
            line++;
            n--;
            i++;
      }
      i=0;
      printbuffer();
      bytes_no_buffer=0;
      if(split>PDULEN){
            part=PDULEN;
      }
      else{
            part=(split)+1;
      }
      return;
}

main(int argc,char **argv)
{
      struct itimerval timer_CU;
      char *ptr;
      int fifofdin;

      if(argc!=3){
            fprintf(stderr,"%s usage: %s <infifo>
<outfifo>\n",argv[0],argv[0]);
            exit(1);
      }
      printf("Opening %s\n",argv[1]);
      fifofdin=open(argv[1],O_RDONLY);
      if(fifofdin==-1){
            perror(argv[1]);
            exit(1);
```

```
	}
	fifofdout=open(argv[2],O_WRONLY|O_CREAT,0644);
	if(fifofdout==-1){
		perror(argv[2]);
		exit(1);
	}
	printf("->\n");
	bytes_no_buffer=0;

	/* Programa a interrupção do relógio */
	signal(SIGALRM,calculaCPS_PDU);
	timer_CU.it_interval.tv_sec=0;
	timer_CU.it_interval.tv_usec=TIMEMAX;
	timer_CU.it_value.tv_sec=0;
	timer_CU.it_value.tv_usec=TIMEMAX;
	setitimer(ITIMER_REAL,&timer_CU,NULL);

	/* No ciclo principal apenas lê a entrada. */
	while(1){
		if(bytes_no_buffer<PDULEN){
			ptr=buffer_entrada+bytes_no_buffer;
			bytes_no_buffer+=read(fifofdin,ptr,PDULEN-
bytes_no_buffer);
		}
	}
	return 0;
}
```