Electrónica e Telecomunicações

universidade de aveiro

AVEIRO • JAN. • 94 • VOL. 1 • Nº 1

Revista do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro

Electrónica e Telecomunicações
Volume 1, nº 1
Abril de 1994

Revista do Departamento de
Electrónica e Telecomunicações
da Universidade de Aveiro

Editores:
Francisco Vaz
José Luís Oliveira

Comissão Editorial:
Alexandre Mota
Ana Maria Tomé
Aníbal Oliveira Duarte
António Ferrari de Almeida
António Nunes da Cruz
António Sousa Pereira
Atílio Gameiro
Dinis Magalhães dos Santos
Fernando Ramos
Joaquim Arnaldo Martins
João Pedro Estima de Oliveira
José Alberto Fonseca
José Alberto Rafael
José Carlos Neves
José Carlos Pedro
José Ferreira da Rocha
José Rocha Pereira
Nelson Pacheco da Rocha
Maria Beatriz Sousa Santos
Paulo Jorge Ferreira

Morada e Secretariado:
Departamento de Electrónica e
Telecomunicações
Universidade de Aveiro
Campus Universitário
3800 Aveiro
Portugal

Tiragem: 600 exemplares
Impressão: Sociedade Gráfica do Vouga

Editorial

A necessidade de editar uma revista técnica onde se divulgassem as actividades do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro, DETUA, foi sentida desde longa data. Quando em Junho de 1993 a Comissão Científica do DETUA decidiu proceder ao seu lançamento, foi com prazer e algum receio das dificuldades advinhadas, que aceitei a incumbência de o realizar. Passados alguns meses e alguns problemas, é com alegria que apresento o primeiro número, ciente das limitações que apresenta, mas certo de que um importante passo foi dado.

O objectivo principal de Electrónica e Telecomunicações é a publicação dos trabalhos de docentes, investigadores e alunos sobre temas pedagógicos, científicos ou tecnológicos, realizados no âmbito das actividades do DETUA. No entanto, considera-se desejável que num futuro próximo, a presença nestas páginas de autores externos enriqueça esta publicação.

Pretende-se uma periodicidade semestral para a edição, objectivo que será possível atingir desde que todos os responsáveis pela actividade do DETUA aceitem que esta revista seja indispensável para a realização completa do seu trabalho como docentes e investigadores, que modifiquem hábitos estabelecidos e tornem rotina a publicação dos resultados da sua actividade na Universidade. Só assim se poderá atingir um outro objectivo desta publicação que é a de ser mais um instrumento aberto à comunidade científica e universitária para a avaliação das actividades do DETUA.

Francisco Vaz

O aparecimento desta publicação periódica do DETUA é por si só um facto indicador da vitalidade do departamento e, logo, uma iniciativa que merece todo o apoio do Conselho Directivo. Acresce que os objectivos inerentes a esta publicação, onde se privilegia a actividade do departamento na qual a componente lectiva tem um lugar determinante, conferem a esta iniciativa um duplo papel como elemento de aprendizagem e elemento de estudo.

Na primeira perspectiva porque, ao estruturar os trabalhos no formato de artigo científico conferindo-lhes um rigor científico acrescido ao que habitualmente é exigido para fins meramente académicos, os grupos de alunos de projecto de licenciatura e mestrado participam activamente no seu processo de aprendizagem; como elemento de estudo pois servirá de referência, para futuras gerações de alunos, não só retrospectiva mas também de experiência e conhecimento acumulados.

Para além da função de informação e referência da sua própria actividade para a comunidade não-universitária, esta publicação cumpre ainda a função do departamento como escola .

José Alberto Rafael

Índice

Desenvolvimento de um Agente de Monitoração para Redes TCP/IP 5
Fernando Manuel Pipa, José Luís Oliveira, Joaquim Arnaldo Martins

Reconhecimento do Orador com Redes Neuronais 16
António Teixeira, Francisco Vaz

Plataforma SNMP para o Desenvolvimento de Objectos de Gestão 23
Abílio Pacheco de Carvalho, Nelson Pacheco da Rocha

Monitorização e Visualização de Recursos num Ambiente Distribuido 33
Pedro Miguel Costa, Maurício Frade Fernandes, Joaquim Arnaldo Martins
José Luís Oliveira

Bridge para Interligação de uma Rede *Ethernet* e de uma Rede Sem Fios 41
Alberto Teixeira de Queirós, Manuel Gomes de Oliveira, Rui Tomaz Valadas,
A. Manuel de Oliveira Duarte

Terminal Multi-Utilizador 47
Alexandre Sarmento, Carlos Carvalhal, Nelson Pacheco da Rocha

Treino assistido por computador 55
Fermando Sérgio Barbosa, Nelson Pacheco da Rocha

A System for Computer Assisted Analysis of Ventriculograms 59
Beatriz de Sousa Santos, Orlando de Sousa, Delfina Coelho,
Pedro Magalhães, Luís Batista

Sistema Integrado de Aquisição de Dados para Potenciometria 62
Rui Manuel Gonçalves, Sérgio Santiago e Santiago, António Rui Borges
Artur Carneiro Pereira, Armando Duarte, Isabel Boal

Projecto, construção e teste de agregados de antenas "microstrip" 69
Gulherme F.M. da Silva,Carlos A.T. Campos, José Rocha Pereira

Placa Quadriprocessador Baseada em Transputers 77
José C. Fonseca, Paulo J. Gonçalves, António Rui Borges,
Artur Carneiro Pereira,Joaquim Arnaldo Martins, Joaquim Sousa Pinto

Simulação de Algoritmos para Equalização de Sinias Áudio 86
António Ricardo Silva, Filipe Bento, José Vieira

Desenvolvimento de um Agente de Monitoração para Redes TCP/IP

Fernando Manuel Pipa, José Luís Oliveira, Joaquim Arnaldo Martins

Resumo- **Neste trabalho é apresentado o desenvolvimento de um agente de monitoração. O agente é construído sobre uma plataforma SNMP que permite o desenvolvimento de aplicações de gestão que utilizam o *Simple Network Management Protocol* (SNMP) como protocolo de comunicação. No agente é implementada a *Remote Network Monitoring Management Information Base* (RMON MIB) que reúne um conjunto de objectos de gestão relacionados com a informação de monitoração.**

No intuito de permitir ao agente um modo de funcionamento local e remoto, este é dotado de algumas capacidades de visualização e manipulação da informação de gestão, de modo a tornar possível a monitoração quer localmente no agente, quer remotamente no gestor. Para isso é apresentado um módulo de visualização gráfica, no qual o acesso à informação é feita através de uma *interface* genérica por forma a permitir a sua portabilidade para qualquer aplicação de gestão.

Abstract- **This work presents the development of a monitoring agent. This agent is built over a SNMP platform which allows the development of management applications that use the Simple Network Management Protocol (SNMP) as the communication protocol. The agent implements the Remote Network Monitoring Management Information Base (RMON MIB) which comprises a set of objects related with monitoring information.**

In order to allow local or remote monitoring operations, the agent is enhanced with some visualisation and operation capabilities. For that, it is supplied with a graphical module which collects object's values through an interface, in such a way that it can be portable to different management applications.

I. INTRODUÇÃO*

Actualmente, os utilizadores de serviços de informação necessitam de um acesso, cada vez mais rápido, a uma grande variedade de informação e recursos. Esses recursos podem estar distribuídos por diversas redes e utilizar diferentes sistemas de operação. No sentido de facilitar a partilha de informação e recursos através das redes, surgiram algumas comunidades que têm desenvolvido um conjunto de normas para os protocolos de comunicação.

Uma vez criadas as condições que tornaram possível as comunicações ao longo das diversas redes, o utilizador é confrontado com o problema de gestão desta colecção de redes e sistemas, como uma única rede de comunicações. O utilizador tem de ser capaz de monitorar a rede por forma a obter informação em tempo-real sobre as características de desempenho e tráfego, diagnosticar problemas de comunicação e reconfigurar a rede de acordo com as condições exigidas pela mesma. É neste sentido que surge aquilo que se designa por Gestão de Redes. Normalmente, a gestão abrange recursos que são constituídos por equipamentos de diversos fabricantes levando à necessidade de um sistema de gestão integrado.

Uma gestão integrada permite que um gestor desempenhe a sua função de um modo mais eficiente, através de um melhor acesso à informação (que pode ser obtida de uma maneira coerente a partir de todos os elementos do sistema), resultando num poder de decisão mais correcto, numa aplicação de controlos mais acertada e consequentemente num aumento da qualidade de serviço disponível ao utilizador. Para esta integração, contribui a produção de normas de gestão.

Assim, as normas representam a possibilidade de tornar real a integração de um sistema de gestão, permitindo que os fabricantes produzam equipamento de acordo com uma norma comum e que outras entidades desenvolvam serviços de gestão capazes de abranger os equipamentos dos diversos fabricantes.

As actividades de gestão requerem uma base de informação e um conjunto de ferramentas de manipulação de dados. A base de dados tem por finalidade armazenar a informação relativa ao estado da rede e à configuração do sistema, o desempenho actual e passado, registos de problemas, os parâmetros de segurança e a informação de contabilização.

A identificação dos elementos envolvidos num sistema distribuído, foi algo de importante na definição e normalização da informação de gestão, de modo a permitir que tais elementos possam ser geridos. Estes, tomam a designação de objectos de gestão, representando uma abstracção dos recursos existentes na rede. De um modo geral, os elementos de um sistema que necessitam de ser geridos, podem ser representados por objectos de gestão.

O objectivo deste trabalho consiste em definir e implementar objectos de gestão na arquitectura Internet (TCP/IP), com o desenvolvimento de um sistema integrado de monitoração de uma rede local. Este sistema baseia-se num agente de monitoração remota que implementa a RMON MIB.

* A realização deste trabalho enquadrou-se no projecto PMCT/C/TIT/458.90 tendo sido financiado pela JNICT.

II. SNMP

O *Simple Network Management Protocol* (SNMP) é um protocolo de comunicação utilizado na troca de informação entre as entidades de gestão na arquitectura Internet. Este protocolo é constituído por três elementos que estão definidos nos *Request for Comments* (RFCs) emitidos pela *Internet Activity Board* (IAB). Assim, a estrutura da informação de gestão (SMI - *Structure of Management Information*) está especificada no RFC 1155 [1]. Este, apresenta o modelo administrativo e a organização relativa à definição dos objectos de gestão. A SMI define uma sintaxe a baixo nível para cada objecto de gestão, baseada no ASN.1[1] [2], bem como a informação relativa ao estado do objecto, os direitos de acesso e directivas para a construção do nome do objecto. A SMI assegura uma informação com um formato comum quando o gestor está a controlar equipamento de diferentes fabricantes, desde que os seus produtos implementem variáveis SNMP.

O segundo elemento refere-se à base de informação de gestão (MIB - *Management Information Base*) especificado no RFC 1213. Esta base dispõe de uma lista dos recursos da rede (objectos de gestão) que podem ser interrogados a qualquer agente, bem como define a informação que deve ser devolvida por este, em consequência de uma acção de pedido relativa a um objecto. A MIB é implementada no agente.

Os utilizadores do SNMP podem definir objectos que, por serem específicos, são referidos como variáveis estendidas. Estas, podem ser colocadas no domínio público, permitindo que alguém possa desenvolver um gestor capaz de as controlar.

A SMI define uma estrutura da MIB em árvore, tal como se demonstra na fig. 1.

A cada nó da árvore é atribuído um número e um nome simbólico. Deste modo, a representação de um objecto da MIB pode ser feito tanto na forma numérica, como na forma do nome. A representação numérica de um objecto é feita concatenando todos os números dos nós que se encontram no caminho, desde a raiz até ao objecto em causa.

O terceiro elemento do SNMP é definido pelo RFC 1157 que descreve o próprio SNMP. Enquanto que os documentos da SMI e da MIB definem os objectos de gestão, o SNMP define o modo como a estação de gestão adquire a informação dos objectos geridos. O SNMP utiliza como protocolo de transporte o *User Datagram Protocol over IP* (UDP/IP), o qual assume uma transmissão fiável não fazendo qualquer tipo de verificação de erros. As mensagens SNMP podem ser transmitidas sobre qualquer mecanismo de transporte, desde que este permita uma comunicação bi-direccional

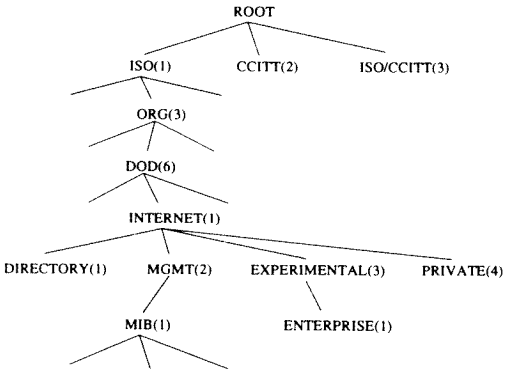


Fig. 1 - Estrutura da MIB em árvore.

feita com base num endereçamento. Na fig. 2 encontra-se representada a arquitectura SNMP.

As mensagens SNMP são representadas na notação ASN.1.

O SNMP utiliza um processo de *polling* no acesso à informação de gestão, não existindo uma ligação lógica entre o gestor e os objectos.

O SNMP é basicamente um protocolo do tipo pergunta-resposta, tendo 5 mensagens: GET-REQUEST, GET-NEXT-REQUEST, GET-RESPONSE, SET-REQUEST, e TRAP. A mensagem GET-REQUEST é utilizada pelo gestor para obter informação relativa a um ou mais objectos de uma MIB implementada num agente. O GET-NEXT-REQUEST tem uma função idêntica à do GET-REQUEST, permitindo percorrer a árvore da MIB pedindo o objecto hierarquicamente a seguir ao indicado no pedido. O GET-RESPONSE é usado pelo agente para retornar a informação de gestão pedida pelo gestor. O SET-REQUEST é utilizado pelo gestor para alterar o valor de um objecto da MIB. O TRAP desvia-se um pouco da filosofia das primitivas anteriores. Esta, é uma mensagem não solicitada, assíncrona, do agente para o gestor indicando que foi ultrapassado um limiar de decisão.

Rapidamente, o SNMP adquiriu uma grande popularidade, tendo sido implementado em diversos dispositivos, tais como PCs, Macs, *bridges*, *routers*, *workstations*. Apesar disso, este possui algumas limitações

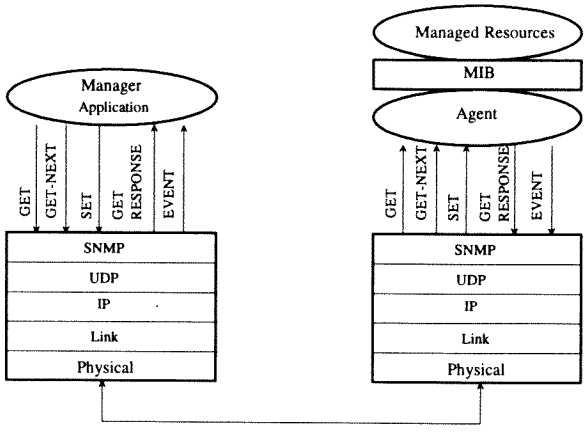


Fig. 2 - Comunicação SNMP.

[1] ASN.1 é uma linguagem OSI utilizada para descrever a sintaxe abstracta. Permite representar a informação num formato que é percetível para o utilizador.

e problemas. Uma das principais lacunas do SNMP refere-se à segurança, pelo que a *Internet Engineering Task Force* (IETF) aprovou em meados do ano de 1992 o *Secure SNMP* [3]. Esta foi a primeira alteração mais significativa que se efectuou no protocolo. No sentido de resolver outros problemas, foi proposta uma revisão ao SNMP, que se designou por *Simple Network Management Protocol version 2* (SNMPv2) [4].

O desenvolvimento do SNMPv2, tal como o SNMP, teve em conta a minimização do custo e da complexidade dos agentes, pondo o processamento de gestão nas estações centrais. Uma vez que, numa rede, existem mais agentes do que gestores, é fundamental que os primeiros consumam o mínimo dos recursos da rede. Um dos princípios do SNMP e do SNMPv2 é colocar a maior parte do processamento nas estações gestoras, ficando o agente com o menor processamento possível. Esta regra permite que a máquina que suporta o processo agente fique com tempo disponível para executar as suas próprias tarefas.

O SNMPv2 melhora o funcionamento das funções de controlo, através do comando SET. Por um lado, introduz um modo de *lock*, que permite à estação gestora configurar um recurso, sem que o seu acesso seja interrompido. Por outro, o SNMPv2 define os procedimentos para criar, modificar e remover linhas de tabelas.

O SNMPv2 foi dotado de uma nova capacidade no sentido de aumentar o seu desempenho relativamente ao SNMP. Com o auxílio do novo comando GETBULK, a estação gestora pode receber simultaneamente informação de várias variáveis. Este mecanismo produz uma baixa taxa de utilização dos recursos da rede e é fácil de implementar.

O SNMPv2 define novos tipos de dados incluindo contadores de 64 bits. Os contadores do SNMP são de 32 bits, mas com o crescimento das redes e o consequente aumento da complexidade, estes tornaram-se pequenos em determinados ambientes.

O SNMPv2 introduz um novo tipo de dados que se refere aos endereços OSI, permitindo assim um alargamento da gestão às redes OSI.

Se um agente SNMP não dispõe de informação relativa a uma parte do pedido feito pela estação gestora, porque a MIB que contém tais objectos não está implementada, esse pedido será completamente rejeitado. Este problema foi resolvido no SNMPv2 com a introdução de condições de excepção. Um agente SNMPv2 pode responder apenas a uma parte do pedido da estação de gestão, devido à falta de informação na MIB, pelo que o agente assinalará a falha com uma "excepção", retornando o resto da informação.

O SNMPv2 contém um vasto conjunto de códigos de erros, que podem ser utilizados pelo agente para justificar a sua incapacidade de resposta a certos pedidos da estação gestora. Tais mensagens de erro fornecem às aplicações de gestão mais informação relativa às falhas dos agentes, de modo que podem determinar se a falha é permanente ou temporária e tomar decisões para corrigir essas falhas.

Ao contrário do SNMP, o SNMPv2 permite a comunicação entre as estações gestoras. Em primeiro lugar, o SNMPv2 dispõe de transferência de informação confirmada entre os gestores, incluindo o envio de notificações. Isto é conseguido com o comando INFORM, no qual um gestor envia informação a outro, com pedido de confirmação da mensagem recebida. Por outro lado, foi definida uma SNMPv2 MIB [5] que controla a geração e a transmissão de eventos entre as estações gestoras.

A capacidade de comunicação entre gestores permite a construção de um sistema de gestão hierárquico. Num sistema deste tipo, um gestor de uma LAN, pertencente a um nível intermédio na estrutura hierárquica, é responsável pelo *polling* aos recursos pertencentes a essa LAN. Numa situação de detecção de uma falha, a estação gestora intermédia envia uma notificação à estação gestora do nível hierárquico superior. Este mecanismo reduz o tráfego de gestão global mantendo um bom nível de fiabilidade de gestão, que se baseia no *polling* dos agentes.

Do ponto de vista da construção e desenvolvimento de MIBs, o SNMPv2 permite adicionar anotações às MIBs, sob a forma de *macros*, de modo a serem interpretadas pela máquina.

O SNMPv2 pode utilizar diversos protocolos de transporte, tais como, o TCP/IP, o IPX, o AppleTalk e o CLNP (*Connection-less Network Protocol*) do OSI. Para tal, foram estabelecidas especificações que descrevem o modo de colocar mensagens SNMPv2 nessas pilhas [6].

As estações de gestão podem, com o auxílio de agentes *proxy*, dialogar com outros protocolos de rede que não sejam TCP/IP.

Um dos principais objectivos do desenvolvimento do SNMPv2 foi a compatibilidade com o SNMP, sempre que possível. Relativamente à transição do SNMP para o SNMPv2 são sugeridos dois caminhos: a utilização de gestores com duas linguagens e o uso de agentes *proxy*. Uma estação gestora com dupla linguagem implementa o SNMP e o SNMPv2, sendo a opção do protocolo feita de acordo com o tipo de agente (SNMP ou SNMPv2) envolvido no processo de gestão. Por outro lado, pode-se utilizar um agente *proxy* para converter o formato de uma mensagem noutro formato. Por exemplo, um gestor SNMP pode converter mensagens SNMPv2 através de um agente *proxy*.

Além disto, o SNMPv2 utiliza o conceito de *Party MIB*, introduzido pelo *Secure* SNMP, que suporta múltiplos agentes numa única plataforma (esta foi uma das principais inovações do SNMPv2) [7].

Uma estação gestora, com uma *Party* MIB, pode comunicar com vários agentes através de um endereço IP, desde que esses agentes estejam registados como *parties* na estação gestora. Por outras palavras, o SNMPv2 permite mais do que um agente numa plataforma computacional. Resumindo, a *Party* MIB permite que todos os elementos constituintes de um sistema computacional (sistema operativo, dispositivos de

armazenamento, ou mesmo aplicações) estejam equipados com os seus próprios agentes.

III. OBJECTOS DE GESTÃO

Esta secção pretende dar uma introdução teórica sobre os objectos de gestão na arquitectura TCP/IP. Começa-se por apresentar a sintaxe abstracta que é utilizada na descrição dos dados de gestão, passando depois à definição da estrutura da informação.

A. Sintaxe Abstracta

Ao nível da camada de aplicação, as estruturas de dados que são trocadas entre as entidades protocolares são bastante complexas. Por essa razão, é necessário introduzir um formalismo na descrição dessas estruturas. Este formalismo é designado por sintaxe abstracta, permitindo definir dados sem atender às estruturas e restrições que são impostas pela máquina, na qual se pretende efectuar a implementação. Deste modo, é possível definir informação de uma forma aberta, isto é, independentemente da arquitectura a que se destina. Para este efeito, a Comunidade Internet optou por uma linguagem da ISO, nomeadamente a *Abstract Syntax Notation One* (ASN.1).

O emprego da ASN.1 tem dois objectivos distintos, um consiste na definição dos formatos dos PDUs (*Protocol Data Units*) trocados pelo protocolo de gestão, o outro na definição dos objectos geridos. Isto significa que a descrição das estruturas de dados (utilizadas na comunicação ao nível dos protocolos) e da informação de gestão nelas contidas, é feita com base na sintaxe abstracta.

A codificação das estruturas de dados especificadas pela ASN.1, é estabelecida pelas *Basic Encoding Rules* (BER) [8].

A implementação da ASN.1 na sua totalidade resulta numa complexidade tal, que a Internet optou por adoptar apenas um subconjunto das suas capacidades.

B. Módulos

Um módulo é um conjunto de descrições ASN.1 relativas a um tema comum. A sua sintaxe é a seguinte:

```
<<module>> DEFINITIONS ::= BEGIN
<<linkage>>
<<declarations>>
END
```

O termo <<module>> indica o nome do módulo. O <<linkage>> permite relacionar este módulo com outros, fazendo o EXPORT/IMPORT de definições para/de outros módulos. As definições ASN.1 encontram-se nas <<declarations>>.

A ASN.1 define três tipos de objectos:

- *tipos*, que definem novas estruturas de dados (o nome começa com letra maiúscula, p. e. Gauge);
- *valores*, que representam as instâncias de um tipo (o nome começa com letra minúscula, p. e. internet);
- *macros*, que são utilizadas para alterar a actual gramática da linguagem ASN.1 (o nome consiste em letras maiúsculas, p. e. OBJECT-TYPE).

C. Estrutura da Informação de Gestão

A estrutura da informação de gestão (SMI - *Structure of Management Information*) [1] descreve as estruturas e o esquema de identificação para a definição da informação de gestão. Dela, fazem parte as descrições do modelo de informação de um objecto com os tipos que lhe estão associados. As descrições formais da estrutura da informação são feitas com base linguagem ASN.1.

Os objectos de gestão são acessíveis através de uma base de informação virtual, designada por *Management Information Base* ou MIB, sendo definidos através da *Abstract Sintax Notation One* (ASN.1).

Cada tipo de objecto tem um nome, uma sintaxe e uma codificação. O nome é representado unicamente como sendo um *OBJECT IDENTIFIER*, o qual é atribuído de uma forma administrativa. A sintaxe define a estrutura de dados abstracta, relativa ao tipo de objecto em causa. A codificação indica como são representadas as instâncias daquele tipo de objecto, aplicando as *Basic Encoding Rules* à respectiva sintaxe.

Cada objecto de gestão é descrito através de uma macro ASN.1 definida na SMI. A macro é identificada como OBJECT-TYPE . Esta permite representar os principais aspectos do tipo de objecto, de uma maneira formal.

No sentido de tornar as descrições da MIB mais concisas e informativas, foi feito um melhoramento à SMI com a edição do RFC1212 [9].

D. Objectos Tabelados

O SNMP permite executar operações sobre objectos escalares. No entanto, para os programadores de aplicações de gestão é conveniente ter uma visão, em termos conceptuais, dos objectos da MIB estruturados sob a forma de tabelas. Cada uma dessas tabelas conceptuais, possui zero ou mais linhas, tendo cada linha um ou mais objectos que se designam por objectos tabelados. Esta conceptualização é formalizada pela macro OBJECT-TYPE utilizada na definição de objectos que correspondem a uma tabela.

E. Definição de Objectos

A definição da MIB consiste em duas partes: uma parte textual, na qual os objectos estão divididos por grupos e um módulo da MIB, no qual os objectos são descritos somente em termos da macro ASN.1 designada por OBJECT-TYPE, que está definida na SMI:

```
IMPORTS
```

```
 ObjectName
  FROM RFC1155-SMI
 DisplayString
  FROM RFC1158-MIB;

OBJECT-TYPE MACRO::=
BEGIN
 TYPE NOTATION ::=
  "SYNTAX" type(ObjectSyntax)
  "ACESS" Access
  "STATUS" Status
  DescrPart
  ReferPart
  IndexPart
  DefValPart
 VALUE NOTATION ::= value (VALUE
ObjectName)

Access ::= "read-only"
   | "read-write"
   | "write-only"
   | "not-accessible"
Status ::= "mandatory"
   | "optional"
   | "obsolete"
   | "deprecated"
DescrPart ::= "DESCRIPTION" value
(description Display String)
   | empty
ReferPart ::= "REFERENCE" value
(reference DisplayString)
   | empty
IndexPart ::= "INDEX" "{" IndexTypes "}"
   | empty
IndexTypes ::= IndexType
   | IndexType "," IndexType
IndexType ::= value (indexobject
ObjectName)
   | type (indextype)
DefValPart ::= "DEFVAL" "{" value
(defvalue ObjectSyntax) "}"
   | empty
END

IndexSyntax ::=
 CHOICE {
  number
   INTEGER (0 .. MAX),
  string
   OCTET STRING,
  object
   OBJECT IDENTIFIER,
  address
   NetworkAddress,
  ipAddress
   IpAddress
 }
```

O significado de cada clausula é apresentado em seguida.

SYNTAX: a sintaxe do objecto define o tipo de dados que modela o objecto.

ACCESS: o nível de acesso a um objecto é um dos seguintes:

- read-only, as instâncias do objecto podem ser lidas, mas não podem ser alteradas;
- read-write, as instâncias do objecto podem ser lidas e alteradas;
- write-only, as instâncias do objecto podem ser alteradas, mas não se podem ler;
- not-accessible, as instâncias do objecto não podem ser lidas nem alteradas.

STATUS: os requisitos de implementação de um objecto podem ser um dos seguintes:

- mandatory, os agentes de gestão devem implementar o objecto;
- optional, os agentes podem implementar o objecto;
- obsolete, os agentes não necessitam de implementar o objecto.

DESCRIPTION: é opcional, contendo uma definição textual do tipo de objecto.

REFERENCE: é opcional e contém uma referência textual para um objecto definido noutro módulo da MIB.

INDEX: pode ou não estar presente. Tem como finalidade definir a informação de identificação da instância, para um tipo de objecto. Se a cláusula INDEX não está presente e o tipo de objecto corresponde a um objecto não tabelado, então as instâncias desse objecto são identificadas adicionando um sub-identificador de zero ao nome desse objecto. No caso dos objectos tabelados, esta cláusula tem de estar presente, no sentido de indicar o modo como deve ser feita a identificação das instâncias desses objectos.

DEFVAL: O emprego desta cláusula é opcional. Esta define um valor que é atribuído por defeito quando é criada uma instância de um objecto.

Nome: o valor (nome) do objecto é, segundo a macro, do tipo ObjectName. Deste modo, os nomes dos objectos são atribuídos de acordo com os OBJECT IDENTIFIERs

IV. MIB PARA MONITORAÇÃO REMOTA

Em finais dos anos oitenta, foi introduzida uma nova classe de recursos, no âmbito da gestão de redes, identificada como monitores distribuídos. Estes consistem numa plataforma de *hardware* dedicada, que permite uma monitoração da rede sem a necessidade de apoio de mais nenhum recurso. Os monitores distribuídos fornecem uma variedade de ferramentas para detecção de falhas e aumento do desempenho, através da "escuta" de todos os pacotes que circulam na rede, contabilizando as estatísticas das actividades de envio e recepção, efectuadas sobre a rede.

A *Remote Network Monitoring MIB* (RMON MIB) [10] é constituída por um conjunto de objectos que representam os diversos aspectos da informação de monitoração.

A RMON MIB dispõe de uma grande quantidade de objectos que podem ser aplicados à gestão de qualquer tipo de rede, havendo contudo, alguns que são específicos à gestão de redes Ethernet. No entanto, a estrutura desta MIB permite também definir objectos específicos a outros tipos de rede, nomeadamente Token Ring [11], prevendo-se que em futuras versões da RMON estejam definidas extensões para outras redes, tais como a FDDI.

Com a monitoração remota pretende-se suportar os seguintes casos:

- Esta MIB permite configurar o monitor de modo a realizar diagnósticos e recolher estatísticas, mesmo em situações em que a comunicação com o gestor não é possível ou não está eficiente. O agente de monitoração pode notificar a estação gestora da ocorrência de condições excepcionais. Assim, as falhas, o desempenho e a informação de configuração podem ser continuamente armazenadas e enviadas ao gestor de uma forma conveniente e eficiente.
- Com base nos recursos disponíveis, o monitor pode diagnosticar continuamente a rede e armazenar a informação de monitoração, de modo a permitir ao gestor a identificação das causas de falhas que tenham ocorrido.
- O monitor pode ser configurado para reconhecer certas condições, normalmente condições de erro, analisando-as continuamente. Quando uma dessas condições é verificada, o evento pode ser registado e as estações gestoras são notificadas.
- O dispositivo de monitoração remota pode valorizar a informação recolhida. Por exemplo, o monitor pode dar ênfase às máquinas que geram mais tráfego ou erros.

A. A Estrutura da MIB

Os objectos da RMON MIB estão distribuídos pelos seguintes grupos:

statistics: contém as estatísticas recolhidas pelo monitor, para cada tipo de *interface*. As estatísticas são armazenadas numa tabela de contadores com informação sobre o número de pacotes, bytes, *broadcasts*, erros, *multicasts* e colisões num segmento da rede.

history: consiste no armazenamento de um número limitado de amostras estatísticas, recolhidas periodicamente da rede.

alarm: recolhe periodicamente amostras estatísticas das variáveis do monitor e compara-as com limiares de decisão previamente configurados. Se o valor de uma variável monitorada ultrapassar o limiar, então é gerado um evento. Para limitar a geração repetida de alarmes, este grupo dispõe de um mecanismo de histerese.

host: contém estatísticas associadas a cada uma das máquinas detectadas na rede. A detecção das máquinas na rede, consiste na análise de uma lista de endereços físicos (fonte e destino) construída a partir de pacotes sem erros.

hostTopN: é utilizado para preparar relatórios que descrevem as máquinas que constam no topo de uma lista, ordenada por uma das estatísticas. Estas são amostras das estatísticas base, durante um intervalo especificado pela estação gestora. O gestor selecciona também o número de máquinas que irão constar no relatório.

matrix: armazena as estatísticas relativas ao diálogo entre duas máquinas. Sempre que é detectada uma nova conversação, o monitor cria uma nova entrada nas suas tabelas.

filter: permite capturar pacotes segundo uma expressão de filtragem arbitrária. Os pacotes filtrados formam uma sequência de informação, também designada por "canal", que pode ser posteriormente capturada ou utilizada para gerar eventos.

capture: permite capturar pacotes que tenham passado através do "canal", isto é, filtrados.

event: controla a geração e a notificação de eventos a partir do monitor.

A implementação destes grupos num agente, é opcional, devendo, no entanto, a implementação de todos os objectos de um determinado grupo ser obrigatória. Pode-se ter vários agentes RMON no mesmo segmento da rede, implementando cada qual uma parte da RMON, no sentido de executarem funções de gestão específicas com melhor desempenho.

V. COMPILADORES DE MIBS

Um compilador de MIBs SNMP é uma ferramenta extremamente útil não só para os autores das MIBs, mas também para os programadores de agentes SNMP e utilizadores de aplicações de gestão SNMP. Para além de verificar a sintaxe da MIB, um compilador pode gerar automaticamente estruturas de dados e código necessário ao agente para implementar uma determinada MIB.

Um compilador é uma ferramenta que traduz um programa escrito numa linguagem (linguagem fonte) para outro equivalente, noutra linguagem (linguagem destino). Tipicamente a linguagem fonte é uma linguagem de alto nível, tal como o Pascal ou C, enquanto que a linguagem destino é uma linguagem de programação a baixo nível, como por exemplo linguagem máquina.

A linguagem fonte para um compilador de MIBs é a linguagem ASN.1 (*Abstract Syntax Notation One*). A ASN.1 não é uma linguagem de programação, mas sim uma linguagem para descrever informação estruturada. A ASN.1 assemelha-se à declaração dos dados de uma linguagem de alto nível. A entrada de um compilador de MIBs é uma colecção de módulos MIB escritos com base num subconjunto da linguagem ASN.1. Estes módulos da MIB contêm definições dos objectos de gestão, que

correspondem à informação disponível nos recursos da rede que pode ser manipulada através do SNMP.

Os compiladores de MIBs geram vários tipos de representação dos objectos de gestão. Algumas destas representações são declarações de estruturas de dados numa linguagem de programação de alto nível, tal como o C, e que podem ser compiladas e ligadas a uma aplicação de gestão (gestor ou agente). Outras, são ficheiros de dados que contêm definições de objectos de gestão, que são lidos para a memória por uma aplicação de gestão, durante a execução do programa. Nalguns casos, os compiladores geram código para auxiliar a implementação das MIBs nos agentes.

Algumas implementações de agentes não possuem informação relativa à árvore da MIB, durante a fase de compilação do agente. Em vez disso, estes agentes constroem a estrutura da árvore, durante a execução, a partir de um ficheiro de dados que possui a definição da MIB.

Como já foi referido, os compiladores de MIBs geram ficheiros de saída com vários formatos. Um formato comum a muitos compiladores, é o formato *mosy*, cujo nome deriva de um compilador muito popular, o MOSY [12]. Um ficheiro com este formato possui uma representação da árvore da MIB, que é utilizada pelas aplicações de gestão para fazer uma correspondência entre os descritores e os respectivos OBJECT IDENTIFIERS. Este ficheiro, especifica também a sintaxe, o acesso e o estado de cada objecto. O exemplo de um extracto de um ficheiro (gerado a partir do RFC1213) com o formato *mosy* pode ser o seguinte:

```
mib-2 mgmt.1
system mib-2.1
. . .
sysDescr system.1 OctetString read-only
mandatory
sysObjectID system.2 ObjectID read-only
mandatory
. . .
```

Alguns compiladores produzem ficheiros com formatos de versões estendidas do *mosy*.

O MOSY é um compilador de MIBs, do domínio público e do tipo *standalone* (o compilador é um único programa, apenas com essa função), cujo nome corresponde à designação de *Managed Object Syntax-compiler (Yacc-based)*. Este compilador faz parte do pacote de *software* do ISODE (*ISO Development Environment*) e é utilizado na definição de novos objectos. Basicamente, o MOSY lê a descrição de um módulo de objectos de gestão, verificando a sua sintaxe, e produz um ficheiro ASCII contendo as definições equivalentes. Posteriormente, o ficheiro é lido como informação de configuração pelo programa de gestão.

VI. PLATAFORMA SNMP DE DESENVOLVIMENTO E IMPLEMENTAÇÃO DE OBJECTOS DE GESTÃO.

Realizou-se, na Universidade de Aveiro, o estudo e projecto de uma plataforma SNMP de desenvolvimento e implementação de objectos de gestão [13]. A plataforma define um ambiente de gestão SNMPv1 numa rede TCP/IP, disponibilizando primitivas SNMP para as aplicações de gestão (estação gestora e agente). Esta plataforma possibilita a extensão da MIB no agente, isto é, permite a definição e implementação de novos objectos, de modo a serem enquadrados na MIB que está associada ao agente. Os sistemas operativos suportados são o DOS e o UNIX, sendo a linguagem de desenvolvimento o C.

Em termos gerais, o ambiente definido pela plataforma SNMP engloba uma rede de comunicações à qual estão ligadas diversas entidades de gestão (gestores e agentes). Cada uma destas entidades dispõe de um controlo das sessões que lhes permite estabelecer comunicações com aplicações remotas. A aplicação tem igualmente acesso a um conjunto de primitivas SNMP, através das quais pode exercer a sua função de gestão.

Um possível cenário para a plataforma SNMP seria o representado na fig. 3, no qual a estação gestora efectua operações de gestão sobre cada um dos agentes, utilizando primitivas de gestão SNMP.

Encontra-se representada na fig. 4, a arquitectura de um agente de gestão. Neste está implementada uma estrutura de dados que representa a organização da MIB. O agente recorre a esta estrutura sempre que tem necessidade de obter ou alterar o valor de um dado objecto de gestão. Deste modo, a MIB funciona como uma *interface* entre o agente e os recursos, dando ao agente e ao gestor uma visão organizada da informação de gestão.

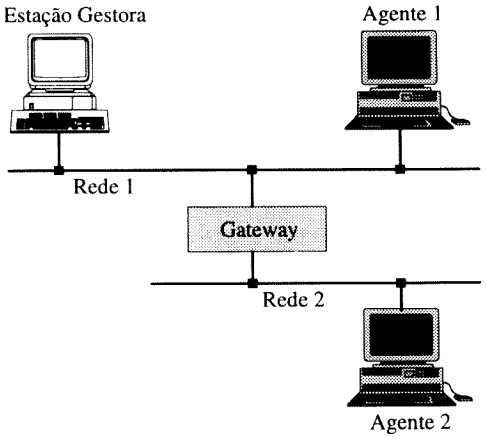


Fig. 3 - Ambiente de Gestão SNMP.

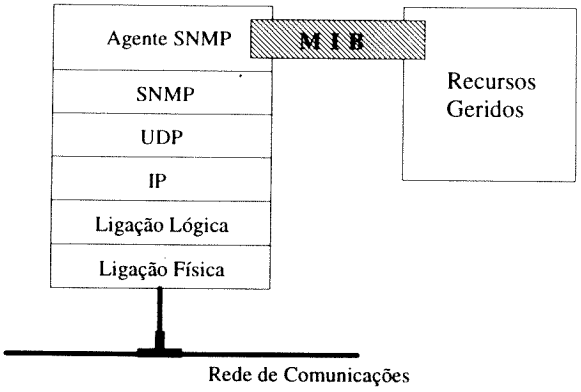


Fig. 4 - O agente SNMP, a MIB e os recursos geridos.

A. *Implementação de um Módulo da MIB no Agente SNMP*

Nesta secção apresenta-se o procedimento para a implementação de um módulo da MIB na plataforma SNMP.

Em primeiro lugar é necessário definir os objectos de gestão que irão ser implementados. Isto consegue-se através da construção de um módulo da MIB. A sintaxe do módulo é definida no RFC 1212 (*Concise MIB Definitions*).

O próximo passo consiste em compilar o módulo da MIB para um formato que seja capaz de ser interpretado pelo programa que implementa o agente. Para tal, é utilizado o MOSY que gera um ficheiro com as especificações dos objectos num formato específico utilizado na construção das estruturas de dados que representam a MIB.

Seguidamente definem-se as funções que correspondem às primitivas de SET, GET e GET-NEXT para cada um dos objectos.

As estruturas de dados, que representam a organização da MIB são criadas durante a execução do agente (resultado da leitura do módulo da MIB compilado) e inicializadas através de uma função definida pelo autor do módulo. Nessas estruturas de dados existe, associado a cada objecto, um campo com a indicação das funções que correspondem ao GET/GET-NEXT e SET para esse objecto. É então necessário construir uma ou mais funções de inicialização, capazes de preencher essas estruturas de dados com informação relativa às funções que correspondem às primitivas SNMP, para cada um dos objectos definidos. No agente, estas inicializações são efectuadas logo após a criação das estruturas de dados internas que representam a MIB.

B. *Implementação de um Agente de Monitoração Remota*

A arquitectura do agente de monitoração está representada na fig. 5.

Esta arquitectura tem como suporte um PC equipado com duas placas de rede, uma para a comunicação SNMP e a

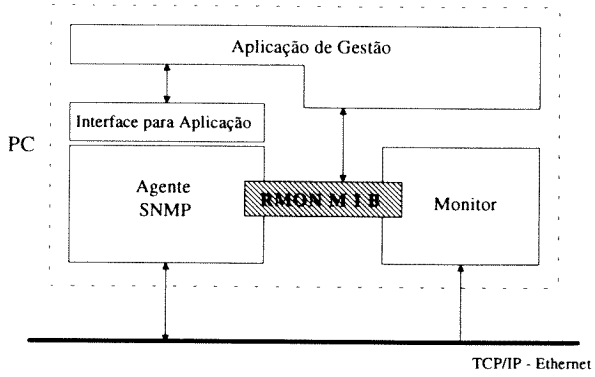


Fig. 5 - Arquitectura de um agente de monitoração remota.

outra para a monitoração da rede. Dela fazem parte um agente SNMP construído com base na plataforma SNMP e um monitor da rede. A informação de monitoração é representada pelos objectos definidos na RMON MIB (RFC 1271). Estas duas entidades, o agente e o monitor, utilizam o mesmo suporte computacional em ambiente DOS, funcionando este, na sua globalidade, como agente de monitoração.

O agente de monitoração realiza, essencialmente, três tarefas: uma consiste em fazer a monitoração da rede através de execução de um programa específico que recolhe toda a informação para tabelas que correspondem às definidas na RMON MIB; a outra consiste em responder aos pedidos da estação gestora, de acordo com o protocolo de comunicação SNMP; a terceira está associada às aplicações de gestão locais com visualização gráfica.

O agente SNMP, na sua fase de inicial, começa por executar uma função que faz a leitura das definições contidas no módulo da MIB compilado, tendo como resultado a criação das estruturas internas que representam a organização da MIB. Essas estruturas são posteriormente inicializadas por funções que indicam, para cada objecto, as funções de GET/GET-NEXT e SET a serem utilizadas pelo agente aquando dos pedidos efectuados pelo gestor. Realizadas estas operações, o agente aguarda pelos pedidos de gestão. Quando é recebida uma mensagem, esta é processada no sentido de identificar o tipo de operação a ser executada e os objectos a que se referem essa operação de gestão.

A informação de monitoração para a RMON MIB é fornecida pelo monitor que consiste basicamente em duas rotinas: uma faz o processamento do pacote que é capturado da rede, incrementando os contadores estatísticos relativos à informação de monitoração; a outra rotina tem como finalidade fazer uma amostragem aos contadores em intervalos de tempo definidos pelo utilizador. As amostras recolhidas são armazenadas em tabelas que correspondem às definidas para os grupos da RMON MIB.

No sentido de dotar o agente com capacidades de gestão locais, foi criada uma *interface* SNMP que permite às aplicações de gestão realizar, de uma forma normalizada,

operações sobre os objectos da MIB. Deste modo, uma aplicação construída sobre essa *interface* pode ser facilmente implementada em qualquer entidade que disponha dos serviços de gestão SNMP. Daqui resulta a possibilidade do desenvolvimento de aplicações de gestão de uma forma genérica, independentemente da entidade gestora a que se destinam (gestor ou agente).

Para aumentar o desempenho de uma aplicação, esta pode ter acesso directo às estruturas da MIB, permitindo assim que as operações de gestão sejam realizadas de uma maneira mais eficiente.

VII. NOTAS DE IMPLEMENTAÇÃO

Pretende-se, nesta secção, apresentar a forma como foram abordados certos pontos que não estão especificados no documento da RMON MIB ou são tratados de uma forma pouco clara.

etherHistoryUtilization: Este objecto pertence ao grupo history e representa uma estimativa da utilização do meio físico (rede Ethernet), em centenas de percentagem, durante o intervalo de tempo de amostragem. O cálculo do valor deste objecto teve por base o tamanho do pacote Ethernet, bem como o espaçamento entre pacotes que é designando por *inter-frame gap*.

A fig. 6 representa o formato de um pacote Ethernet [14]. Para além do número de bytes total, calculados com base no tamanho do pacote fornecido pela placa de rede (inclui os bytes do FCS - *Frame Check Sequence*), é preciso ter em conta o número de bytes relativos ao preâmbulo, ao SFD (*Start of Frame Delimiter*) e ao *inter-frame gap* para o cálculo da utilização de um segmento da rede Ethernet (em centenas de percentagem):

$$\text{Utilização} = \frac{(\text{Octetos} + \text{Pacotes} * 20) * 8}{C_{\text{Ethernet}} * T_{\text{amostragem}}} * 10000 \qquad (1)$$

onde:

Octetos - nº de bytes recolhidos num intervalo de amostragem (inclui os bytes do FCS);

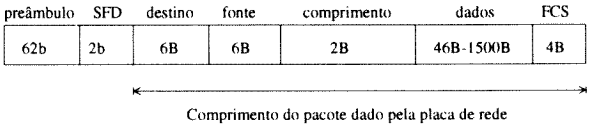
Pacotes - nº de pacotes capturados no intervalo de amostragem;

C_{Ethernet} - Capacidade da rede Ethernet = 10Mbits/s;

$T_{\text{amostragem}}$ - Período de amostragem em segundos.

***Overflow* dos contadores**: Uma vez que os contadores são implementados em variáveis de 4 bytes, o *overflow* destes será inevitável ao fim de um determinado tempo. O contador mais crítico é o que contabiliza o número de bytes total, pelo que o tempo de ocorrência de *overflow* depende da taxa de utilização da rede. Para se ter uma ideia deste tempo, na fig. 7 está representado o tempo de ocorrência de *overflow* em função da taxa de utilização da rede.

Nesta representação, assumiu-se que a taxa de utilização da rede era devida apenas aos bytes contabilizados no contador. Na realidade, para além destes bytes é necessário ter em conta o número de pacotes no cálculo da taxa de utilização (1), pelo que o gráfico apresenta uma estimativa do valor do tempo por defeito, contando que a taxa de utilização é real.



inter-frame gap = 9.6 us = 12 bytes

SFD - *Start of Frame Delimiter*

FCS - *Frame Check Sequence*

b - bits

B - bytes

Fig. 6 - Formato de um pacote Ethernet.

A RMON dispõe de meios para informar o gestor da ocorrência de um *overflow*. Isto consegue-se através da utilização do grupo alarm, vigiando periodicamente um objecto deste tipo e gerando um alarme aquando da detecção do *overflow*.

Variáveis com Valores por Defeito: Existem algumas variáveis (objectos) definidas no RFC1271 que não possuem valores por defeito. Perante uma situação destas, a questão que se põe é a seguinte: o que deve devolver o agente como resposta a uma operação de GET desses objectos? Existem duas possibilidades [15]:

- Definir um valor por defeito e retorná-lo até que seja alterado.

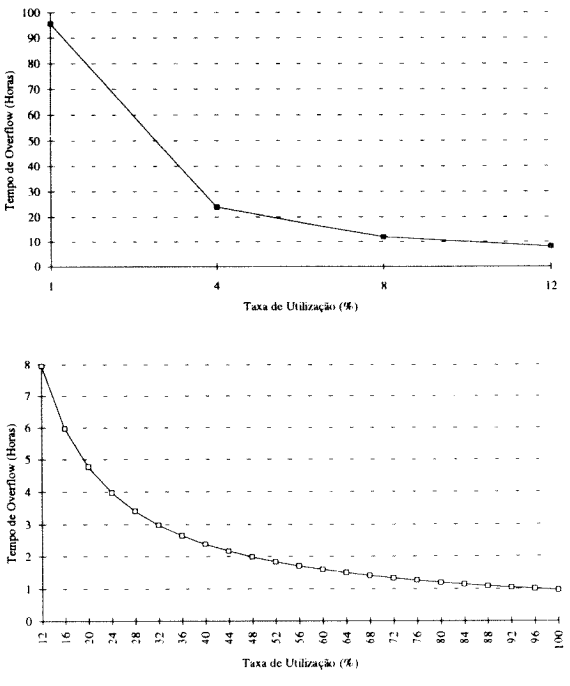


Fig. 7 - Representação do tempo de ocorrência de *overflow* em função da taxa de utilização da rede.

- Devolver NOSUCHNAME até que a variável seja inicializada pelo gestor.

No caso de ser possível encontrar valores por defeito satisfatórios para as referidas variáveis, então a primeira escolha é preferível. Isto porque a maior parte dos inteiros podem tomar o valor zero, as variáveis lógicas podem ser verdadeiras (1) ou falsas (0), etc. Esta opção só faz sentido se não gerar situações ambíguas quanto ao valor do objecto. Se isso acontecer, a escolha recai na segunda hipótese. O retorno de NOSUCHNAME torna essas variáveis "invisíveis", eliminando a possibilidade de interpretações erradas quanto aos seus valores. No entanto, esta alternativa obriga a que o gestor tenha um pleno conhecimento da MIB implementada no agente, de modo a poder inicializar essas variáveis.

Implementação de Objectos: Viu-se, em secções anteriores, que a implementação dos grupos da RMON é opcional, pelo que um agente poderá implementar apenas alguns deles. Quanto aos objectos de um grupo, as suas implementações são obrigatórias. Pode acontecer um agente não possuir informação relativa a um determinado objecto, por exemplo, devido às capacidades limitadas dos recursos que utiliza. Numa situação destas, que valor deve devolver o agente face a uma operação de GET para esse objecto?

Existem duas hipóteses [16]: uma consiste em retornar um valor adequado (zero no caso de ser uma variável inteira), a outra é retornar NOSUCHNAME. A primeira situação pode induzir o gestor em erro, levando-o a pensar que aquele é o valor efectivo do objecto. O retorno de NOSUCHNAME é o mais correcto, tendo contudo o inconveniente de interromper os algoritmos de recolha de dados de uma tabela, com base na operação GET, uma vez que recebendo um valor deste tipo abortam o pedido de informação. Se a informação é adquirida com a primitiva GET-NEXT, ao chegar o objecto cujo valor de retorno é NOSUCHNAME, a função GET-NEXT avança para o próximo objecto (em termos da árvore da MIB), não havendo qualquer interrupção no processo de aquisição de informação. Deste modo, o gestor concluirá que tal objecto não está implementado.

Analisando as duas hipóteses em termos de conformidade com as normas das MIBs, verifica-se que no caso em que é retornado um valor para um objecto nessas condições, o grupo, ao qual pertence esse objecto, respeita a respectiva norma porque implementa esse objecto, mas não implementa a sua semântica. No caso de retorno de NOSUCHNAME, o grupo implementado não fica de acordo com a norma porque não implementa aquele objecto, sendo no entanto, em termos de valor, a situação mais correcta.

VIII. DEFINIÇÃO DE UM AMBIENTE DE GESTÃO COM MONITORAÇÃO REMOTA

Com base no agente de monitoração remota que foi desenvolvido, pode-se definir um ambiente de gestão de acordo com a fig. 8.

Este cenário é constituído por um agente de monitoração e dois gestores. Estas entidades de gestão foram construídas sobre a plataforma SNMP que permite a implementação dos gestores quer em PCs quer em máquinas UNIX.

Um cenário deste tipo permite obter uma perspectiva global da utilização da rede, através da monitoração do tráfego e da actividade das máquinas, utilizando para isso a RMON MIB. Um agente, para obter um melhor desempenho numa actividade específica de monitoração, pode implementar apenas subconjuntos da RMON MIB.

O agente de monitoração, para além de comunicar com os gestores SNMP, pode funcionar também como um sistema de gestão independente. Neste caso, o agente dispõe de algumas funcionalidades de gestão com capacidades de visualização gráfica, de modo a permitir uma representação local da informação de monitoração e notificar a ocorrência de alguma falha.

A visualização da informação é suportada por um módulo constituído por rotinas gráficas. Esse módulo possui uma *interface* SNMP, que permite o acesso à informação de monitoração através das primitivas SNMP. Isto significa que o módulo opera sobre dos objectos utilizando as primitivas GET, GET-NEXT e SET, de modo a obter a informação necessária para a visualização gráfica e permitir que o utilizador possa também controlar os parâmetros de configuração do agente de monitoração. Com esta *interface*, o módulo gráfico é dotado de uma portabilidade que torna possível a sua implementação em diferentes gestores e agentes SNMP, sem necessidade de alteração do *software*.

Os gestores comunicam com os agentes de monitoração remota através do protocolo SNMP, realizando diversas

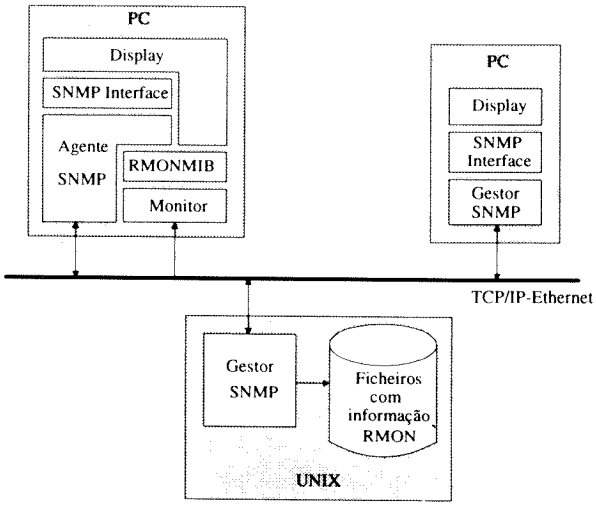


Fig. 8 - Exemplo de um ambiente de gestão baseado num agente de monitoração

operações de gestão tais como a aquisição da informação da MIB ou a configuração de alguns dos seus parâmetros. O gestor pode também armazenar a informação da MIB num ficheiro para uma análise posterior, quer para efeitos de estudos de tráfego na rede, quer para diagnosticar possíveis causas de falhas que tenham ocorrido.

IX. CONCLUSÕES

Actualmente, o processo de integração dos sistemas de gestão está a ser dificultado pelos fabricantes com a apresentação de *interfaces* gráficas sofisticadas, juntando características particulares aos seus próprios sistemas. Torna-se, por isso, necessário melhorar a gestão de redes do ponto de vista de funcionalidade, facilidade de utilização e integração, através da elaboração de normas que devem ser respeitadas pelos fabricantes no desenvolvimento dos sistemas de gestão.

De facto, o sucesso da arquitectura *Internet* (TCP/IP) implicou o desenvolvimento de uma estrutura de gestão para este tipo de redes, do qual surgiu o protocolo SNMP que teve como primeiro objectivo, a resolução do problema de gestão a curto prazo.

Os problemas resultantes das limitações do SNMP, levaram a uma revisão do próprio SNMP, que resultou num protocolo designado por SNMPv2.

Uma das principais actividades de gestão consiste na monitoração de redes, no sentido de diagnosticar falhas, planear e melhorar o desempenho destas. Para isto contribuiu a criação da RMON MIB, que representando uma nova geração de ferramentas de gestão complementa algumas das deficiências da gestão SNMP. Os agentes que implementam esta MIB designam-se normalmente por agentes de monitoração, ou simplesmente agentes RMON.

Para o desenvolvimento de um agente SNMP são necessárias ferramentas que permitam o diálogo com o gestor (implementação do protocolo SNMP) e que implementem a estrutura de dados que representa a organização da MIB, bem como a forma de aceder e processar (codificação e descodificação) essa informação. Essas ferramentas possibilitaram a definição e implementação de um sistema integrado de monitoração de redes locais, baseado num agente de monitoração que suporta a RMON MIB.

O agente de monitoração remota foi desenvolvido com base na plataforma SNMP, tendo-lhe sido adicionadas algumas capacidades de gestão por forma a permitir um modo de funcionamento local. Para tal foi implementado um módulo gráfico para a representação dos vários parâmetros relacionados com a rede (tráfego, erros, protocolos, etc). Este módulo assenta sobre uma *interface* SNMP que lhe permite manipular a informação de gestão de uma forma normalizada, através das primitivas GET, GET-NEXT e SET. O módulo assim desenvolvido pode ser implementado sobre qualquer entidade de gestão (gestor ou agente) que possua capacidades de manipulação dos valores dos objectos.

REFERÊNCIAS

[1] M. T. Rose, K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, DDN Network Information Center, SRI International, May 1990.

[2] "Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization / International Electrotechnical Commission, 1987, International Standard 8824.

[3] J. Galvin, K. McCloghrie, J. Davin, "SNMP Security Protocols", RFC 1352, Trusted Information Systems, Inc., Hughes LAN Systems, Inc., MIT Laboratory for Computer Science, July 1992.

[4] J. Case, K. McCloghrie, M. T. Rose, S. Waldbusser, "Introduction to version 2 of the Internet-standard Network Management Framework", RFC 1441, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.

[5] J. Case, K. McCloghrie, M. T. Rose, S. Waldbusser, "Manager-to-Manager Management Information Base", RFC 1451, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.

[6] J. Case, K. McCloghrie, M. T. Rose, S. Waldbusser, "Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1449, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.

[7] K. McCloghrie, J. Galvin, "Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1447, Hughes LAN Systems, Inc., Trusted Information Systems, Inc., April 1993.

[8] "Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, December 1987, International Standard 8825.

[9] M. T. Rose, K. McCloghrie, "Concise MIB Definitions", RFC 1212, DDN Network Information Center, SRI International, March 1991.

[10] S. Waldbusser, "Remote Network Monitoring Management Information Base", RFC 1271, DDN Network Information Center, SRI International, November 1991.

[11] S. Waldbusser, "Token Ring Extensions to the Remote Network Monitoring MIB", RFC1513, Carnegie Mellon University, September 1993.

[12] Marshall T. Rose, Julian P. Onions, Colin J. Robbins, "The ISO Development Environment: User's Manual", Performance Systems International, Inc., July 1991.

[13] Abílio Carvalho, Nelson Rocha, "Plataforma SNMP para o Desenvolvimento de Objectos de Gestão", Revista do DETUA, Aveiro, Dezembro 1993.

[14] ANSI/IEEE, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", ANSI/IEEE Std. 802.3-1985, The Institute of Electrical and Electronics Engineers, 1986.

[15] Kooijman, D. M. Wisse, "The Annotated RMON Developer's Guide", June 1993.

[16] Robin Iddon, "RmonMIB Implementation Problems", RMON MIB Mailing List, AXON Networks Inc., April 1993.

Reconhecimento do Orador com Redes Neuronais

António Teixeira, Francisco Vaz

Resumo - **Neste trabalho apresenta-se uma experiência no uso de uma rede neuronal dinâmica, a Rede Gama Concentrada, na área do Reconhecimento do Orador, nas suas duas variantes de Verificação e Identificação.**

A abordagem utilizada é baseada no modelo clássico em Reconhecimento de Voz. Primeiro são extraidos atributos relevantes usando um banco de filtros. As propriedades são comparadas com referências previamente construidas usando a rede neuronal. A decisão é feita usando as saídas da rede neuronal. Foi utilizada uma rede dinâmica devido á natureza variável no tempo dos padrões.

Foi usada uma rede por cada orador cliente e vários oradores por rede. A primeira abordagem deu os melhores resultados, e por ser modular, torna possível a construção de sistemas com muitos clientes.

Os resultados obtidos são promissores e mostram que o método é robusto.

Abstract - **We present an experience in using a dynamic neural network , the focused gamma net, in the area of speaker recognition, in the two variants of verification and identification. The approach used is based on the classical model for speech recognition. First we extract relevant features with a filter bank. Features are compared with previously constructed references using a neural network. Decision is done using neural network outputs. A dynamic neural network, the gamma net, was used because the patterns are time varying.**

We tried the use of a network per speaker and several speakers per network. The first approach gave better results and , being modular, make possible the construction of large systems.

Results obtained are promising and show that the method is robust.

I. INTRODUÇÃO

A. Assunto

As previsões apontam para um uso maciço de processamento de voz nos sistemas de informação num futuro próximo.

Sendo a voz o meio de comunicação por excelência do ser humano todos estes novos serviços dependem da existência de capacidades de reconhecimento de palavras, reconhecimento do orador, tradução automática e identificação da linguagem utilizada.

O reconhecimento do orador será uma peça chave nestes sistemas devido à necessidade de segurança.

Como sabemos vozes de pessoas diferentes soam de uma forma diferente. Esta importante propriedade da voz, de ser dependente da pessoa, é o que torna possível, por exemplo, reconhecermos um amigo ao telefone. A faculdade de reconhecer uma pessoa apenas pela sua voz é conhecida como **Reconhecimento do Orador**.

O reconhecimento por ouvintes humanos é uma experiência comum conhecida desde sempre. Recentemente, com a disponibilidade de computadores digitais, cientistas interessados na área da voz resolveram explorar a possibilidade de reconhecer automaticamente pessoas utilizando apenas a sua voz o, que se designa por Reconhecimento Automático do Orador.

Apesar de em muitas áreas ser muito difícil igualar o desempenho humano, nesta área (Reconhecimento do Orador) os dados experimentais sugerem que o desempenho das máquinas em muitos casos execede ados seres humanos .

O objectivo final de todos os estudos em Reconhecimento do Orador é chegar a um sistema automático, independente do tempo, que replique a capacidade humana de rapidamente, com exactidão e independentemente do texto pronunciado efectuar o reconhecimento de uma pessoa apenas pela sua voz [1].

B. Aplicações

O Reconhecimento do Orador tem aplicação em **sistemas de segurança** como controlo de acesso (sistemas de fechadura actuados por voz para portas de casa e de automóveis); controlo de acesso a dados em computadores (*password* por voz); controlo de transacções através das linhas telefónicas (reserva de passagens aéreas ou movimentos bancários pelo telefone) [2]. Muitas instituições financeiras, assim como companhias fornecendo acesso limitado a bases de dados em computadores gostariam de poder oferecer serviços automáticos pelo telefone. Como números de código podem ser perdidos, roubados, ou esquecidos o reconhecimento através da voz (se suficientemente de confiança) pode ser uma alternativa [3]. Existe outro tipo de aplicações com muito interesse, num domínio diferente, como sejam a **investigação criminal** (comparação da voz desuspeitos de um crime) e **vigilância de canais de comunicação** [2]. No domínio da tecnologia **militar,** especialmente em sistemas de alta segurança, existe a possibiliade de aplicação destes sistemas. Seria útil para por exemplo aceitar ou rejeitar relatórios da linha da frente [4]. Uma aplicação completamente diferente é o uso destas técnicas para **adaptar os sistemas de Reconhecimento de Voz ás características do orador** para melhorar o desempenho destes sistemas [4].

C. Resumo Histórico da Investigação na Área

A investigação na área de Reconhecimento do Orador iniciou-se na década de 1960 com o estudo do reconhecimento por ouvintes humanos. Nesta altura também era comum o uso de espectogramas para reconhecimento visual [2].

Rosemberg e Atal [5][6] fazem um ponto da situação do desenvolvimento até 1976. Era habitual usarem-se bancos de filtros para a extracção do *short-time spectrum* sendo usados o tom intensidade formantes e ceficientes de

predição linear (LPC) em alguns poucos sistemas. As técnicas de decisão eram muito rudimentares usando-se a distância Euclideana em muitos casos. Nesta altura já se utilizavam algoritmos de alinhamento dinâmicos para tratar as variações temporais da voz. Devido à sua maior simplicidade os sistemas eram dependentes do texto.

Tendo passado quase uma década [2][3] aumentou o interesse por sistemas independentes do texto o que levou ao uso de estatísticas recolhidas ao longo do tempo para representar o sinal de voz Os parâmetros utilizados evoluiram sendo usadas por exemplo análise cepstral e parâmetros LPC ortogonais. No capítulo da classificação é já vulgar a utilização do *Dynamic Time Warping* (DTW) e aparece a Quantificação Vectorial .

Em [7] são analisados alguns dos últimos progressos na área. No capítulo dos parâmetros é usual o uso de parâmetros LPC-cepstrais. Aparecem tambem os *Line Spectrum Pair* (LSP) e Δ–cepstrais. As técnicas actuais apresentadas incluem os Modelos de Markov não Observáveis (HMM) e várias variantes de *source coding* como Quantificação Vectorial, Quantificação Matricial, *Filler Template* e *Acoustic Segment Quantization*. Tem se tentado também o uso de redes neuronais nesta área.

Como é notório tem havido uma evolução quer no tipo de parâmetros usados quer nas técnicas usadas para efectuar a classificação.

D. Uso de Redes Neuronais

Algumas experiências foram relatadas na literatura sobre a utilização de modelos conexionistas para reconhecimento do orador. As primeiras referências são [8] usando Perceptrões multicamada com uma rede por orador, e [9][10] onde um orador é reconhecido por um *codebook* usando *Learning Vector Quantization* (LVQ). Ambas as abordagens são dependentes do texto.

Trabalhos mais recentes em sistemas independentes do texto usaram *Time Delay Neural Networks* (TDNN) [11] e *Predictive Neural Networks* [12]. A última usa parâmetros estáticos e dinâmicos, o que é necessário para atingir desempenhos superiores.

No nosso trabalho usamos uma forma diferente de utilizar a informação espectral dinâmica da voz, a rede gama [13].

II. CONCEITOS

A. Modelo

A abordagem habitual é extrair alguns atributos acústicos da voz e compará-los com um conjunto de referência previamente armazenado. Se existir uma semelhança grande, segundo um determinado critério, entre as propriedades de teste e da referência o orador é reconhecido [2].

De uma forma geral todos os sistemas de Reconhecimento do Orador são constituidos pelos seguintes blocos: Processamento acústico (extracção de parâmetros); Comparação de padrões; Referências e processo de Decisão (Figura 1).

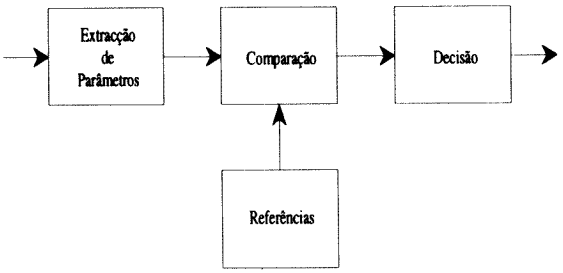


Fig 1 - Modelo do Sistema

Uma pronunciação de um orador desconhecido constitui a entrada do sistema. Esta entrada é analisada para dela se extrair as propriedades que caracterizam o orador. Na fase seguinte estas propriedades são comparadas com propriedades protótipo armazenadas na referência. Usando os resultados da comparação é finalmente efectuada uma decisão [7].

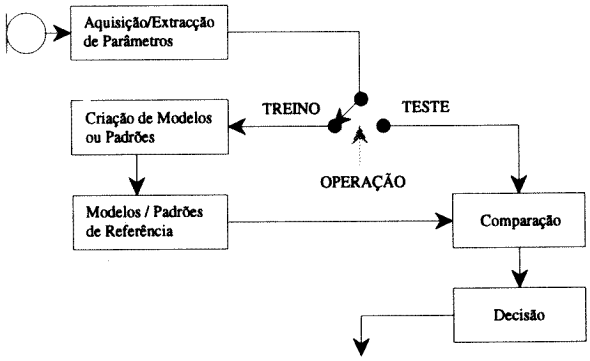


Fig 2 - Diagrama Operacional de um Sistema de Reconhecimento

B. Fases

Todas as tarefas de reconhecimento de padrões incluindo o Reconhecimento do Orador utilizam duas fases: treino e reconhecimento (Figura 2). Efectuado *off-line* e habitualmente combinando métodos manuais e automáticos a fase de treino establece as referências. A fase de reconhecimento é automática e usualmente em tempo real e tenta atribuir a uma entrada desconhecida uma identificação [3].

C. Verificação e Identificação

O Reconhecimento do Orador pode ser dividido em duas grandes classes: Identificação do Orador e Verificação do Orador. O problema mais geral Identificação do Orador pode ser definido da seguinte forma. Numa população de N oradores descobrir aquele a que corresponde o padrão de referência com mais semelhanças ao padrão do orador desconhecido, que constitui a entrada do sistema. A probabiliade total de uma decisão incorrecta é uma função monótona crescente do número de oradores.

O problema de Verificação do Orador pode definir-se da seguinte forma: Dado um padrão de um orador desconhecido conjuntamente com a identidade reclamada, que constituem a entrada do sistema, determinar se o padrão é suficientemente semelhante ao padrão de referência associado à identidade reclamada para se

aceitar a pretensão. Neste caso apenas é efectuada uma comparação independentemente do tamanho da população. Portanto a decisão é geralmente independente do tamanho da população. Uma premissa básica para muitos sistemas de verificação é a de que os utilizadores habituais são cooperantes, isto é, não tentam mudar o seu comportamento conscientemente de tentativa para tentativa.

D. Texto Utilizado

Uma dicotomia para sistemas de Reconhecimento do Orador é se existe um texto obrigatório para os oradores pronunciarem ou não. Em muitas aplicações o texto é fixo chamando-se estes sistemas **dependentes do texto**. Nos sistemas **independentes do texto** existem habitualmente restrições, tais como o tamanho, [5] e outros constrangimentos linguísticos, sendo muito raro o uso de texto completamente livre [14]. Os sistemas dependentes do texto precisam de um elevado grau de colaboração dos utilizadores tendo no entanto a vantagem de facilitarem bastante a comparação. Nos sistemas independentes do texto utilizam-se habitualmente estatísticas recolhidas ao longo de períodos alargados, como por exemplo a média e a variância. Esta abordagem tem o inconveniente de retirar a dimensão tempo. Mais recentemente nestes sistemas começaram a surgir técnicas baseadas em eventos de curta duração [7]. No que respeita às aplicações enquanto os sistemas com texto fixo são usados em sistemas de controlo de acesso (devido a haver cooperação por parte dos utilizadores) tem de usar-se texto livre em sistemas de vigilância (onde não há cooperação consciente).

E. Termos Habituais na Área

Apresentam-se de seguida alguns termos utilizados na área de Reconhecimento do Orador e que se irão aplicar ao longo deste texto.

Clientes - São os oradores conhecidos do sistema que reclamam a sua identidade verdadeira.

Impostores - São não clientes. Os impostores eventuais não imitam a voz de nenhum dos clientes apenas fazendo uso da sua voz natural

Imitadores - Classe de impostores que tentam entrar no sistema disfarçando a sua voz por forma a parecer-se de alguma forma com clientes do sistema.

Goats - Oradores que causam mais problemas ao sistema de reconhecimento.

Sheeps - Oradores que são facilmente reconhecidos.

F. Medição do Desempenho

As identidades usadas quer por clientes quer por impostores para tentar aceder ao sistema são geralmente apenas as pertencentes ao conjunto de clientes do sistema. É fácil rejeitar-se outras identidades usando por exemplo a consulta da tabela de clientes. No processo de verificação podem surgir 3 casos. O caso do uso da identidade de um não cliente como vimos é facilmente evitado distintos no tocante ao utilizador. Orador cliente do sistema que pretende verificar a sua identidade correcta Orador cliente que pretende fazer-se passar por outro Orador não cliente a tentar passar por cliente. Se defenirmos impostor como a pessoa que pretende assumir outra identidade os casos e são apenas variantes.

Podemos ter na entrada do sistema oradores Clientes ou Impostores e na saída pode obter-se cliente aceite ou não aceite.

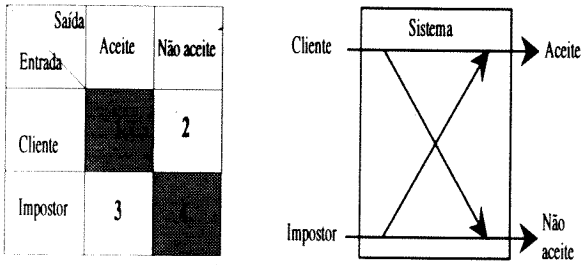


Fig 4 - Verificação - Casos

As situações 1 e 4 correspondem ao Acerto do sistema. A situação 2 corresponde a Falsos Negativos e a situação 3 a Falsos Positivos, tudo situações de erro. Os Falsos Positivos são designados geralmente na literatura por Falsas Aceitações (FA) e os Falsos Negativos por Falsas Rejeições (FR). A entrada Cliente pode ser decomposta em cada um dos clientes individuais para uma análise do comportamento de cada um deles.

Como para aceder ao sistema um impostor tem de usar a identidade de um cliente, nos testes, é necessário simular a escolha do cliente. Em geral não é fácil usar dados reais, ou seja, impostores reais a escolherem um cliente para ludibriar o sistema. O uso de distribuições estatísticas é difícil e duvidoso. Pode evitar-se a questão de saber qual a identidade assumida pelo impostor, usando o pior caso, isto é, a identidade correspondente à saída maior e compará-la com o limiar. Temos assim o erro no caso de o impostor fazer-se passar por esse cliente. No caso de apenas existir um único cliente, não existe qualquer dúvida sobre a identidade que os impostores assumem, o que torna os testes fáceis de realizar.

No caso dos clientes aí sim já é relevante a identidade. Pode determinar-se quais os oradores mais fáceis e mais difíceis para o sistema (*sheeps* e *goats*).

Para indicar o desempenho de um sistema de verificação é comum apresentar-se a sua taxa de erro. O erro do sistema é representado habitualmente pelas taxas FA e FR ou por combinações destas. Além da média aritmética de FA e FR é muito usada a média geométrica. Na caracterização do desempenho neste trabalho usaremos esta última taxa.

Para a identificação, na entrada do bloco de decisão temos clientes e não clientes que são identificados como clientes e impostores, correcta e incorrectamente. As várias hipóteses encontram-se representadas na Figura 5.

Saída / Entrada	Impostor	Cliente 1	Cliente 2	...	Cliente N
Impostor	1	2	2		2
Cliente 1	3	1	4		4
Cliente 2	3	4	1		4
...					
Cliente N	3	4	4		1

Fig 5- Identificação - Casos Possiveis

A identificação correcta corresponde aos casos assinalados com 1. Os casos 2 e 3 são os Falsos Positivos e Falsos Negativos respectivamente. O caso 4 corresponde também a um erro, mas agora devido à troca de um cliente por outro. Em aplicações em que não é feita qualquer distinção em termos de privilégios entre os clientes este último caso não é considerado erro.

A taxa de acerto na identificação pode ser obtida dividindo os padrões classificados correctamente pelo total de padrões de teste. De forma análoga se obtêm as taxas de trocas, falsos negativos e falsos positivos. É no entanto usual utilizar percentagens separadas para cada orador, usando como medida global de desempenho a média aritmética destas.

III. SISTEMA IMPLEMENTADO

O sistema desenvolvido tem a arquitectura clássica na área, já apresentada. Devido à natureza da rede neuronal os blocos de comparação e referências são ambos implementados por esta. Os parâmetros adptáveis da rede constituem as referências, que são adaptadas na fase de treino. Na fase de teste a rede comporta-se como um comparador.

A. Extracção de Parâmetros

Num determinado sistema de processamento da voz apenas parte da informação é usada dependendo da finalidade do sistema. Tradicionalmente a informação é extrraída quer do sinal quer da sua transformada de Fourier. Mas devido à natureza única da voz e a sua ligação ao sistema auditivo podemos explorar formas alternativas que utilizem conhecimentos dos mecanismos auditivos para extracção de informação relevante para a nossa aplicação. O sucesso da utilização desta abordagem depende da quantidade de conhecimento que temos sobre o sistema auditivo, conhecimento este que é conseguido pela combinação de dados recolhidos por estudos psicológicos e fisiológicos.

As propriedades são extraídas usando um banco de filtros baseado num modelo que incorpora conhecimentos acerca do ouvido humano.

O banco cobre a banda de frequências entre os 200 e os 3300 Hz sendo composto por 18 Filtros passa-banda *GammaTone* [15] calculados segundo o trabalho de Moore e Glasberg [16].

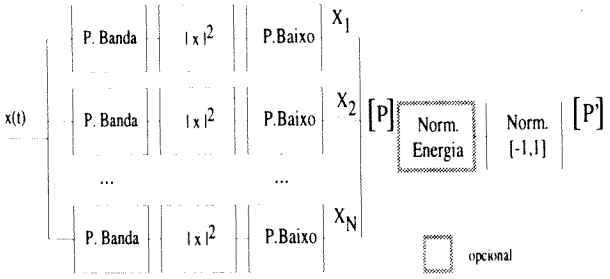


Fig 6- Extracção de Parâmetros - Banco de Filtros

Os sinais na saída do banco de filtros foram amostrados a 30 Hz.

A saída do Banco de Filtros para a frase escolhida consiste numa matriz, em que uma dimensão é a ordem do filtro e a outra pelo tempo. Esta matriz pode ser representada a 2 dimensões usando uma escala de cinzentos. Obtem-se assim uma aproximação ao Espectrograma (Figura 7).

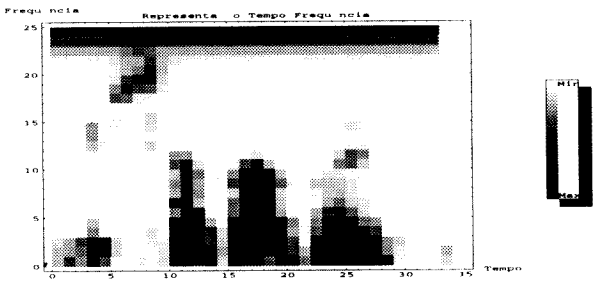


Fig 7 - Saída do Banco de Filtros para a frase "Universidade de Aveiro"

B. Comparação e Referências

Derivado do facto dos padrões extraidos, pelo banco de filtros, serem variante no tempo a rede neuronal utilizada, a Rede Gama Concentrada, possui uma camada de entrada dinâmica. Esta rede neuronal é uma simplificação do modelo de convolução que utiliza atrasos para introdução de dinâmica no modelo [13].

A rede gama é também uma simplificação do modelo *AutoRegressive Moving Average* (ARMA) [13].

Das simplificações obteve-se um modelo neuronal aditivo com estabilidade trivial. Por ser aditivo poderia usar-se o *back-propagation-thru-time* (BPTT) ou o *real-time-recurrent-learning* (RTRL). Mas, o primeiro tem o problema do desdobramento no tempo, criando redes muito grandes no caso de sequências no tempo. O segundo apenas se pode aplicar a redes pequenas, devido às necessidades de processamento.

O uso de arquitecturas *prewired* [17], onde apenas se usa estruturas dinâmicas na camada de entrada, torna possível o uso de uma regra de aprendizagem simples, derivada utilizando a técnica de retropropagação do erro (*back propagation*) [18].

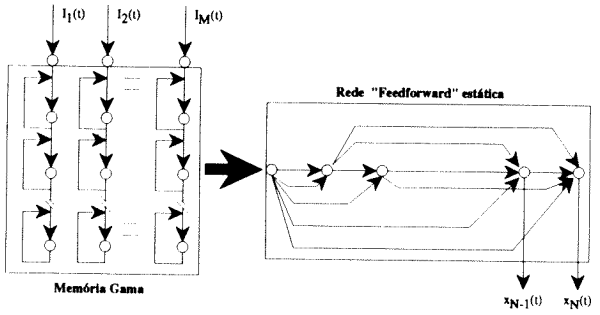


Fig 8 - A Rede Gama Concentrada

Para o treino precisamos de definir o sinal desejado. Como temos uma sequência no tempo, a tarefa é mais complicada que no caso estático habitual.

Habitualmente é fácil escolher o sinal desejado (d(t)) para o instante final. Na aprendizagem baseada em aproximação a escolha natural é ,

$$d_i(t_f) = \begin{cases} 1, i = k \\ 0, i \neq k \end{cases}$$

onde t_f é a duração da sequência e k a unidade correspondente à classe apresentada na entrada. No caso de treino discriminativo queremos maximizar a distância entre as saídas correctas e as incorrectas, isto é algo como,

$$\varepsilon(-y_k(t_f)) + \sum_{i \neq k} y_i(t_f)$$

onde $\varepsilon(.)$ é uma função de saturação monótona crescente. É difícil dizer qual destes métodos dá melhores resultados na prática. Optamos pelo uso do primeiro pela sua simplicidade.

A escolha das trajectórias do sinal desejado para para as unidades de saída é muito menos clara. Foi proposto usar *don't cares*, erro nulo qualquer que seja saída, assim como funções crescentes, como rampas e exponenciais par $t<t_f$, algumas vezes combinadas com funções decrescentes, para os outros nodos de saída [19] [20]. O sinal desejado correcto é função das probabilidades condicionais *a posteriori*, assim como da arquitectura da rede. Geralmente não será aproximado correctamente usando funções escolhidas arbitráriamente como rampas e exponenciais. Por outro lado o uso de *don't cares* conduz a um treino lento, pois só é possível fazer a realimentação do erro no instante final da sequência. Foi proposto o uso de *constraints* nas trajectórias num esforço para tornar mais rápida a aprendizagem e possivelmente aumentar o desempenho [21]. No entanto esta abordagem ainda não foi comprovada em aplicações práticas. Embora pagando bastante em termos de velocidade de aprendizagem usamos *don't cares* nos testes especificar contruir e simular redes neuronais requer um esforço considerável. O uso de ferramentas pode reduzir significativamente o esforço necessário. E fornecendo uma estrutura conceptual torna possível a simulação de redes complexas.

A implementação foi efectuada usando o *Stuttgart Neural Networks Simulator* (SNNS) [22]. O simulador de Redes Gama e o treino foram introduzidos no simulador, extendendo a aplicação deste simulador a novas áreas [23].

C. Decisão

Os valores de saída da rede neuronal são usados pelo bloco de decisão para aceitar ou rejeitar um pedido de verificação, ou para determinar a identidade do orador, no caso de indentificação.

O processo utilizado consiste em determinar a unidade de saída com o maior valor e usar um limiar, no caso de identificação (Figura 9).

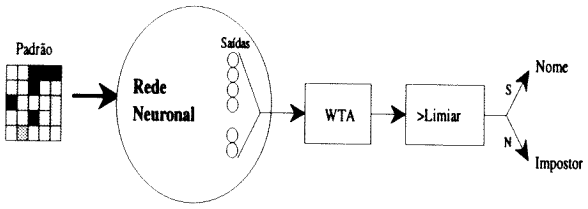


Fig 9 - Decisão -Identificação

No caso da Verificação o procedimento normal é apenas utilizar a informação da saída que corresponde à identidade pretendida. Se esta saída fôr superior ao limiar o pedido é aceite, caso contrário é rejeitado.

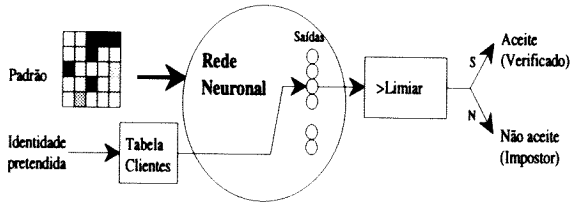


Fig 10 - Decisão -Verificação

Quando o bloco de decisão está inserido num sistema de reconhecimento apenas lhe é exigido que decida. No entanto em sistemas experimentais é necessário obter informações sobre o desempenho do sistema. Englobamos por isso na implementação deste bloco a elaboração das tabelas de confusão e das estatísticas do desempenho.

A decisão no nosso sistema consiste num programa desenvolvido em C que utiliza como entrada o ficheiro de resultados da rede neuronal e fornece á saída as estatísticas (acertos, falsos negativos, falsos positivos e trocas) e ainda as tabelas de confusão. O único parâmetro do programa é o valor do limiar.

IV. DADOS

Para a recolha dos dados para este trabalho foi desenvolvida uma aplicação, para facilitar a gestão dos dados e o trabalho do operador responsável pela recolha ,tornando a tarefa o menos entediante possível ao orador. O desenvolvimento da aplicação foi feito num computador NeXT.

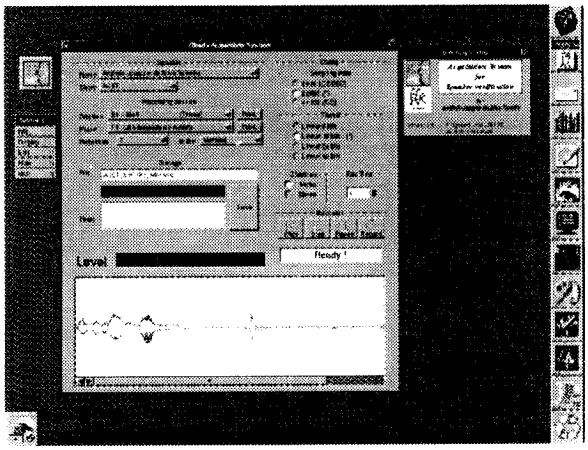


Fig 11 - Ecran da aplicação de recolha

A frase usada ("Universidade de Aveiro") foi escolhida tendo como objectivo a representação a variedade fonética da lingua Portuguesa [24], começando e acabando em vogais para facilitar a segmentação [25].

Os dados foram recolhidos num Laboratório com ruído moderado (e por vezes intenso !), e a frase foi segmentada de uma forma primitiva, utilizando um editor do sinal.

Recolhemos dados de 15 indíviduos do sexo masculino (5 clientes e 10 impostores) em várias sessões de 10 pronunciações cada. Os impostores gravaram 2 sessões, 2 dos clientes 10 sessões, e os 3 outros 7 sessões. A duração média das frases foi de aproximadamente 1 segundo.

V. RESULTADOS

A. Definição dos Parâmetros do Sistema

Efectuamos diversos testes preliminares para definição da topologia da rede a utilizar.

Apenas foi utilizada uma camada escondida com um número variável de unidades. Os melhores resultados foram obtidos com 14 e 16 unidades. Para a camada de saída foi usado um número de unidades igual ao número de clientes, existindo uma correspondência directa entre as unidades e os clientes, sem qualquer codificação. A activação de uma unidade, dependente do processo de decisão, indica um cliente, e não activação de todas as unidades representa um impostor.

Os testes mostraram que é necessário parar o treino com um erro razoavelmente alto, para evitar o excesso de treino e melhorar a generalização. Escolhemos parar o treino para um taxa de falsos positivos igual à taxa de falsos negativos, e usamos esse valor do erro nos testes subsequentes.

É importante que a rede generalize bem o que é um cliente, mas é também essencial que a rede generalize o que é um impostor. Os testes mostraram que a rede necessita de um número superior de padrões de treino de impostores em relação ao número de padrões de treino de clientes.

B. Verificação

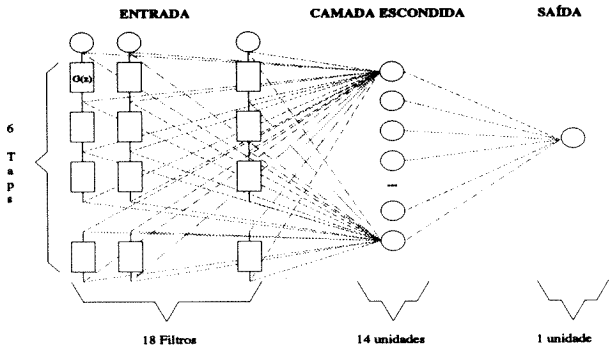


Fig 12 - Rede com apenas 1 saída

Para a verificação foi testada a utilização de uma rede para cada um dos clientes. A rede possuía apenas uma unidade de saída (Figura 12). Através de treino, a rede é especializada no reconhecimento de um orador em particular. Para adicionar outro cliente ao sistema não existe a necessidade de retreinar as redes já existentes, o sistema é modular. Este tipo de abordagem conduz, também, a redes menores que podem ser treinadas mais rápidamente e que generalizam melhor.

Os melhores resultados obtidos foram 8 % de falsa rejeição de clientes e 6.15 % de falsas aceitações de impostores. O pior resultado (usando a voz do primeiro autor) foi de 26 % para a falsa rejeição e de 22.31 % para a falsa aceitação.

Alguns dos resultados são apresentados na Tabela I (em %).

TABELA I

ALGUNS RESULTADOS DA VERIFICAÇÃO

Orador	FR	FA	$(FAxFR)^{1/2}$
PJG	8	6.15	7
AJST	26	22.31	24
JLVR	0	10	---

C. Identificação

Foi tambem efectuado um teste de identificação, usando uma rede com 5 unidades de saída, correspondentes ao mesmo número de clientes.

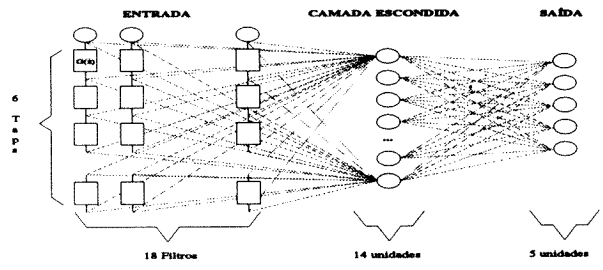


Fig 13 - Rede com 5 saídas

A média de decisões correctas para os 5 clientes foi de 72 %. O pior orador, o mesmo qu causou problemas na identificação, apenas atingiu 40 % e os outros 4 atingiram os 60, 85, 90 e 95 %. Os impostores foram rejeitados correctamente 62 % dos casos. Os impostores com

padrões de treino foram rejeitados, correctamente, 82 % dos testes, mas os outros impostores, sem treino, apenas foram rejeitados correctamente 52 % dos testes.

CONCLUSÕES

Apesar das más condições na recolha do sinal e da pequena base de dados utilizada, dos testes efectuados podem tirar-se as seguintes conclusões:

- Os resultados na verificação podem ser considerados bons, atendendo às limitações (ruído, pequena base de dados, segmentação pouco precisa, utilização de apenas uma frase). No entanto existem problemas para a utilização prática, tais como a melhor topologia ser dependente do orador; o desempenho também variar com o orador; e o treino ser bastante demorado.
- Os resultados da identificação são inferiores aos da verificação. A deterioração dos resultados pode estar relacionada com o reduzido número de padrões de treino (em especial de impostores).
- O método é robusto, pois sem grandes cuidados na recolha e segmentação obtiveram-se resultados aceitáveis.
- Os parâmetros extraídos pelo banco de filtros possuem informação sobre a identidade do orador, além de informação sobre as palavras proferidas.
- A rede gama concentrada tem capacidade para extraír dos parâmetros diferênças entre os oradores.

AGRADECIMENTOS

Este trabalho foi desenvolvido graças à Bolsa de Mestrado Nº BM-1763/91- IA da JNICT.

Agradecemos ao grupo da Universidade de Estugarda que desenvolveu o simulador de redes neuronais SNNS e ao Computational Neuroengineering Laboratory da Universidade da Florida.

REFERÊNCIAS

[1] Ronald S. Cheung e Bruce A. Eisenstein, "Feature Selection via Dynamic Programming for Text-Independent Speaker Identification", IEEE Trans. on Acoustics, *Speech and Signal Processing*, ASSP-26, n.5, pp. 397-403, Outubro 1978

[2] George R. Doddington, "Speaker Recognition - Identifying People by their Voices", *Proc. IEEE*, 73 (11), pp. 1651-1664, Novembro 1985

[3] Douglas O' Shaughnessy, "Speaker Recognition", *IEEE ASSP Magazine*, pp. 4-17, Outubro 1986

[4] Clifford J. Weinstein, "Opportunities for Advanced Speech Processing in Military Computer-Based Systems", *Proc. IEEE*, 79 (11), pp. 1626-1641, Novembro 1991

[5] Aaron E. Rosemberg, "Automatic Speaker Verification: A Review", *Proc. IEEE*, 64(4), pp. 475-487, Abril 1976

[6] Bishnu Atal, "Automatic Recognition of Speakers from their Voices", *Proc. IEEE*, 64(4), pp. 460-474, Abril 1976

[7] Aaron E. Rosemberg and Frank K. Soong, "Recent Research in Automatic Speaker Recognition", In *Advances in Speech Signal Processing*, pp. 701-738, 1992

[8] J. Oglesby and J. S. Mason, "Optimization of Neural Models for Speaker Identification",In *Proceedings ICASSP*, pp. S5.1, Abril 1990

[9] Y. Bennani and F. Fogelman and P. Gallinari, "A Connectionist Approach to Speaker Identifiaction", In *Proceedings ICASSP*, pp. S5.1 Abril 1990

[10] Y. Bennani and F. Fogelman and P. Gallinari, "Text-Dependent Speaker Identification Using Learning Vector Quantization", In *Proceedings INNC*, Paris, France, Julho 1990

[11] Younés Bennani and Patrick Gallinari, "A Modular Connectionist Architecture For Text-Independent Talker Identification", In *Proceedings IJCNN*, Julho 1991

[12] Hiroaki Hattori, "Text-Independent Speaker Recognition Using Neural Networks",In *Proceedings ICASSP*, pp. II-153-157,1992

[13] Bert de Vries e José C. Principe , "The Gamma Model - A New Neural Model for Temporal Processing", *Neural Networks*, 5, pp. 565-576, 1992

[14] John D . Markel e Steven B. Davis, "Text-Independent Speaker Recognition from a Large Linguistically Unconstrained Time-Spaced Data Base", *IEEE Trans. on Acoustics, Speech and Signal Processing*, ASSP-27, n.1, pp. 74-82, Fev. 1979

[15] John Holdsworth, Ian Nimmo-Smith, Roy Patterson e Peter Rice, "Implementing a GammaTone Filter Bank", *Annex C of the SVOS Final Report (Part A: The Auditory Filter Bank)*, Fev. 1988

[16] B. C. J. Moore e B. R. Glasberg, "Cochlear Modeling", *J. Acoust. Soc. Am.*, 74, pp. 750-753, 1983

[17] M. C. Mozer, "A focused back propagation algorithm for temporal pattern recognition", *Complex Systems*, 3, pp. 349-381, 1989

[18] Tomás Oliveira e Silva, "Generalized Feed-forward Filters: Some Theorectical Results", In *Proceedings ICASSP*, pp. 109-112, vol. III,1993

[19] K. P. Unnikrishnan, J. J. Hopfield e D. W. Tank, "Connected-digit speaker-dependent speech recognition using a neural network with time-delayed connections", *IEEE Trans. on Signal Processing*, vol. 39, n. 3, pp. 698-713, 1991

[20] R. L. Watrous, B. Ladendorf e G. Kuhn, "Complete gradient optimization of recurrent network applied to /b/,/d/,/g/ discrimination", *J. Acoust. Soc. Am.*, 87(3), pp. 1301-1309, Mar. 1990

[21] Bert de Vries, Leslie Dias e John Pearson, "Learning with target trajectory constraints for Sequence Classification tasks", In *Proceedings ICASSP*, pp. 525-528, vol. II,1993

[22] Andreas Zell, Niels Mache, Ralf Hübner, Michael Schmalzl, Tilman Sommer, Günter Mamier e Michael Vogt, "SNNS - Stuttgart Neural Network Simulator", Universität Stuttgart, Institute for Parallel and Distributed High Performance Systems (IPVR), User Manual Number 8/92, Fed. Rep. of Germany, 1992

[23] António J. S. Teixeira e Francisco A. C. Vaz, "Using SNNS in Speech Recognition", In *Proceedings of the Workshop on Simulation of Neural Networks with SNNS (SNNS'93)*, Universitaet Stuttgart, IPVR, Federal Republic of Germany, Setembro 1993

[24] M. R. D. Martins, *Ouvir Falar - Introdução à Fonética do Português*. CAMINHO, 1988.

[25] L. R. Rabiner e R. W. Schafer, *Digital Processing os Speech Signals*. Prentice-Hall, Englewood Cliffs, NJ, 1978

Plataforma SNMP para o Desenvolvimento de Objectos de Gestão

Abílio Pacheco de Carvalho, Nelson Pacheco da Rocha

Resumo - **O trabalho aqui apresentado consiste numa plataforma SNMP para o desenvolvimento de objectos de gestão, constituída por um gestor SNMP e um agente SNMP com possibilidade de definição de novos objectos de gestão. Esta plataforma foi desenvolvida de forma genérica, podendo ser instalada em qualquer sistema de comunicações e/ou sistema operativo.**

Abstract - **The article presents a SNMP platform to develop managed objects. This platform consists of a SNMP manager and a SNMP agent with capabilities for the development of new managed objects. It is a general purpose platform able to work in different communication systems and/or operating systems.**

I. INTRODUÇÃO

Os produtos de gestão de redes comercialmente disponíveis possuem particularidades próprias e operam com componentes de rede específicos, colocando dificuldades na sua utilização em cenários heterogéneos como aqueles das actuais redes de comunicação. Em redes de múltiplos segmentos é muito difícil obter informação acerca do nível de utilização da rede, bem como saber, em cada instante, quais os componentes que estão a funcionar com dificuldades. Assim, é necessária uma ferramenta capaz de monitorizar e controlar, de uma forma integrada, todos os elementos de rede, usando quer agentes de monitorização dedicados, quer as ferramentas de gestão que alguns desses elementos já possuem.

Neste contexto, a informação de gestão, é organizada num conjunto de objectos de gestão que constituem entidades capazes de representar os recursos da rede.

II. SISTEMA INTEGRADO DE GESTÃO

Em meados de 1990, começou-se a desenvolver no Departamento de Electrónica e Telecomunicações da Universidade de Aveiro (DETUA) um sistema integrado de gestão para redes locais, baseado em agentes de monitorização e numa Estação Central de Gestão (ECG) na qual se efectuam as operações de gestão [1,2] (Fig. 1). Os agentes de monitorização são ligados aos vários segmentos e permitem saber, em cada instante, o estado de funcionamento do canal de comunicações e dos diversos elementos de rede, em especial daqueles que não possuem qualquer mecanismo de gestão associado. A rede a gerir por este sistema caracteriza-se por ser heterogénea e constituída por múltiplos segmentos, nos quais operam várias famílias protocolares como, por exemplo, TCP/IP e OSI.

A. Estação Central de Gestão

A Estação Central de Gestão (ECG) têm a seu cargo a integração das operações dos agentes de monitorização e das operações de outros agentes. A comunicação com os agentes de gestão, é feita usando diferentes protocolos de gestão que podem ser o Commom Management Information Protocol (CMIP) [3], o Simple Management Information Protocol (SNMP), ou outros protocolos privados. É da responsabilidade da ECG integrar todos estes protocolos e toda a informação disponibilizada pelos diferentes agentes.

A ECG mantém uma base de dados de informação relevante sobre o funcionamento da rede coleccionada através dos agentes, e pode usar capacidades de sistemas periciais para tratamento e resolução de condições anómalas conhecidas. Possui também um sistema de visualização baseado no sistema X-Windows para a apresentação, ao gestor da rede, de forma amigável de um conjunto de mecanismos que lhe permite ter acesso a todos os aspectos, informações e detalhes da rede, e que suportam a representação da rede, a selecção de elementos específicos e a apresentação de resultados [4] (fig.2).

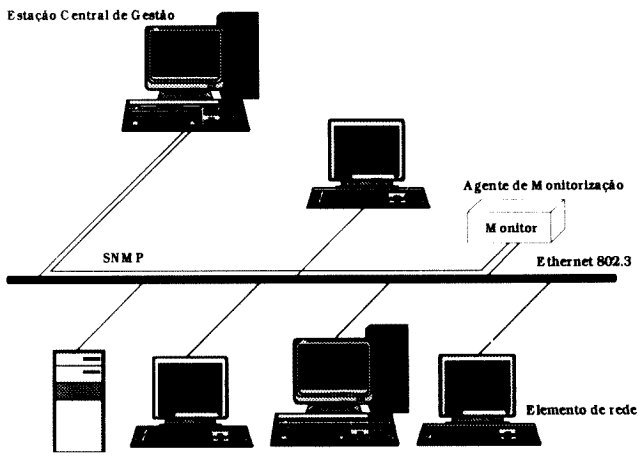


Figura 1. Sistema Integrado de gestão

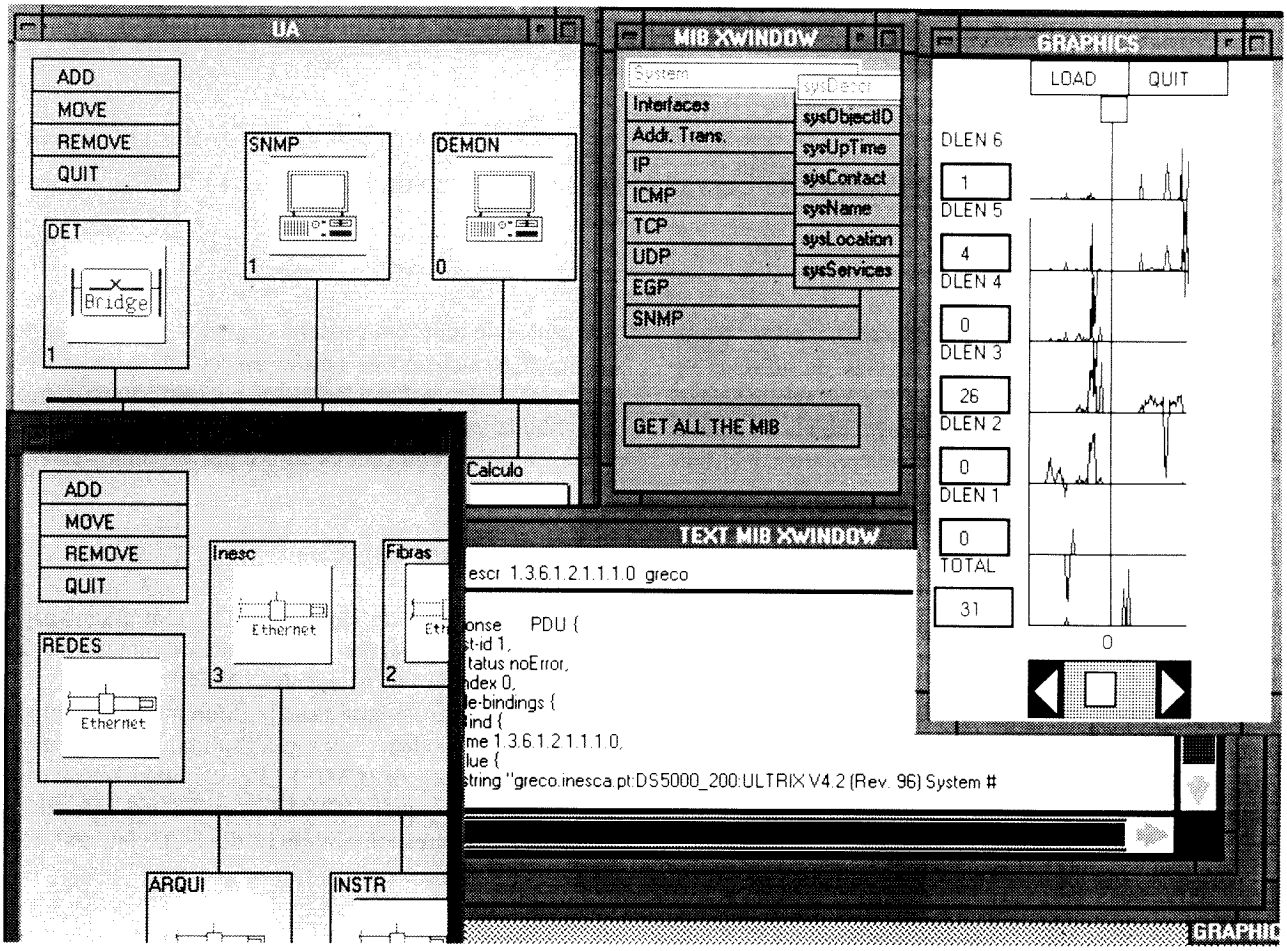


Figura 2. Interface gráfica do Sistema Integrado de Gestão

A ECG possui duas partes fundamentais: uma é o Processo de Comunicação (PC) que colecciona informação de gestão das Management Information Bases (MIBs) existentes nos agentes e envia comandos através do protocolo de gestão; a outra parte é a Aplicação de Gestão (AG), a qual tem a seu cargo o processamento da informação de gestão e a apresentação dos resultados. A interface entre estas duas partes é a Base de Dados de Informação de Rede (BDIR), que contém toda a informação coleccionada e permite a independência entre os dois módulos (Fig. 3).

A BDIR é uma representação abstracta da rede construída pela aplicação de gestão e é independente dos detalhes de comunicação. Por sua vez o PC permite à AG colocar na BDIR a informação obtida nos agentes e enviar comandos de gestão de acordo com as suas determinações.

B. Agentes de monitorização

Os agentes de monitorização são agentes especiais cuja finalidade é coleccionar informação relativa ao comportamento da rede e dos sistemas a ela ligados sem facilidades de gestão. Podem existir vários agentes de monitorização, desempenhando tipos diferentes de análise ou ligados a diferentes tipos de redes (Ethernet, Token-Ring, FDDI, ...). Estes agentes, têm somente funções de gestão e são de grande importância, uma vez que sem eles não seria possível gerir num cenário transitório, sistemas com e sem facilidades de gestão.

As facilidades de gestão disponibilizadas por estes agentes prendem-se com os seguintes tópicos [1]:

- Segurança - Os agentes de monitorização podem analisar os pacotes em trânsito na rede e tirar conclusões acerca dos sistemas a elas ligados, sendo capazes de detectar sistema novos, ou sistemas sem permissão de acesso a essa rede.
- Optimização - Os agentes de monitorização coleccionam dados directamente a partir do meio de transmissão podendo obter informação acerca das condições gerais de funcionamento da rede através de funções de processamento. Estas funções de processamento permitem libertar a ECG de tarefas como, por exemplo, o cálculo do tráfego total, poupando tempo e não sobrecarregando a rede com tráfego de gestão.
- Informação de gestão em tempo real - Os agentes de monitorização obtêm informação de gestão em tempo real, sendo possível a sua parametrização para fazer a aquisição com uma determinada cadência.
- Funcionamento autónomo - Para além das facilidades de gestão disponibilizadas à ECG, os agentes de monitorização podem também fazer a visualização de

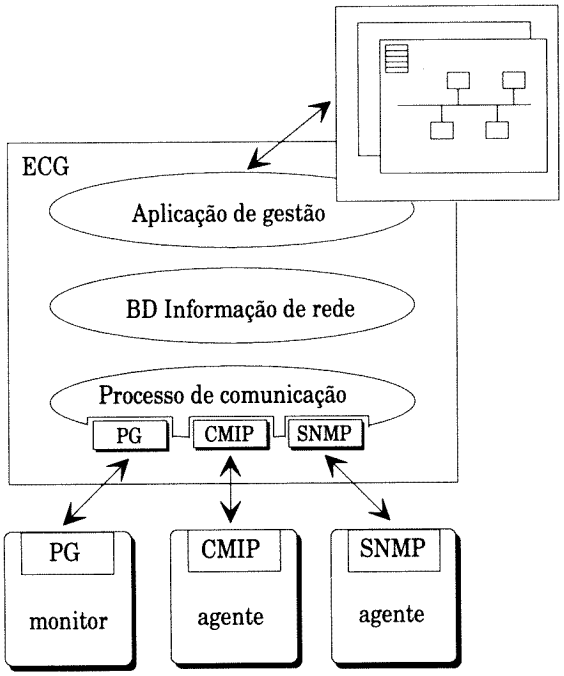


Figura 3. Arquitectura do Sistema Integrado de Gestão

aspectos de funcionamento da rede considerados relevantes.

- Planeamento - dado que o agente de monitorização colecciona informação acerca do meio de transmissão, ele é capaz de concluir sobre as condições de funcionamento da rede, nomeadamente saber quem são os maiores geradores de tráfego em determinado segmento.

A implementação destes agentes de monitorização baseia-se num computador do tipo IBM compatível com dois controladores Ethernet. Este sistema pode possuir vários módulos de monitorização, um módulo de protocolos, e para o funcionamento em modo autónomo possui também uma interface gráfica [5] (Fig. 4). O módulo de monitorização interage com o controlador Ethernet e colecciona todas as tramas do segmento a que está ligado. A informação coleccionada é processada e organizada em formatos adequados, que constituem as sintaxes locais, e que permitem facilmente a conversão para as sintaxes dos objectos de gestão definidos para suportarem essa informação. O módulo de protocolos desempenha as tarefas de interligação com a estação central de gestão para a transferência da informação de gestão. Por último, a interface gráfica permite fazer a visualização local da informação adquirida.

III. GESTÃO SNMP

O SNMP foi desenvolvido no sentido de constituir um protocolo de gestão para a internet, sendo o seu principal objectivo ser um protocolo simples e de fácil implementação[7]. Hoje em dia, ele é um padrão para

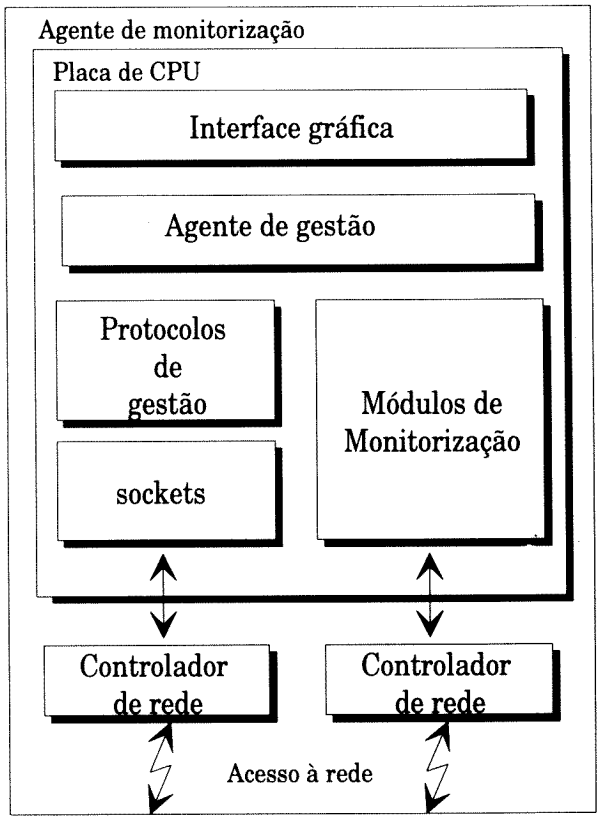


Figura 4. Agente de monitorização

gestão de redes locais, tendo sido adoptado por um número bastante significativo de fabricantes, sendo utilizado em muitos equipamentos, tais como: portões, encaminhadores, estações de trabalho, servidores de terminais, etc[7].

O protocolo SNMP é um protocolo comando-resposta assíncrono (Fig. 5), e suporta vários Protocol Data Units (PDUs), que permitem a execução de diferentes operações (Fig. 6):

- O **get-request**, é usado para requerer aos agentes informação de gestão específica, isto é, o objecto é exactamente especificado;
- O **get-next-request**, é usado para requerer informação de gestão, sem que o objecto pretendido seja especificado exactamente. O objecto retornado é aquele que se encontra lexicograficamente a seguir ao objecto especificado na operação, sendo o resultado da operação dependente do agente. É portanto uma operação de varrimento.
- O **set-request**, é usado para modificar o valor dos objectos com acesso de escrita;
- O **get-response**, é usado pelos agentes para responder às operações de get e set;
- O **trap**, é usado pelos agentes para assinalarem a ocorrência de situações anómalas através de mensagens não solicitadas.

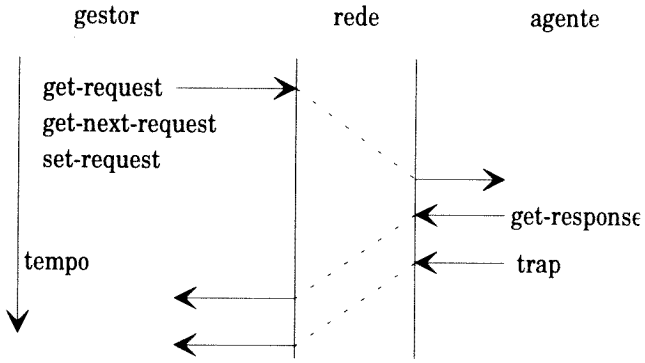


Figura 5. Protocolo comando-resposta

As funções de gestão, são especialmente necessárias quando existem problemas no funcionamento da rede [6], pelo que deverá ser a aplicação de gestão a assegurar o nível de segurança desejado para o tráfego de gestão. O serviço de transporte datagrama é o mais adequado para dar essa liberdade à aplicação de gestão, e também o mais rápido, uma vez que não é necessário o estabelecimento de ligação.

Por outro lado, a transmissão de mensagens SNMP pela rede, é feita através de um processo de seriação, do qual resulta uma sequência de octetos. Esta sequência de octetos pode ser transmitida por qualquer serviço de transporte, sendo apenas necessário que a entidade SNMP receptora seja capaz de aceitar mensagens que quando seriadas ocupam até 484 octetos[8].

Assim, o SNMP não é muito exigente a nível do protocolo de transporte requerido, sendo o protocolo de transporte UDP o preferido.

O SNMP tem-se manifestado como sendo o protocolo dominante para a troca de informação de gestão entre

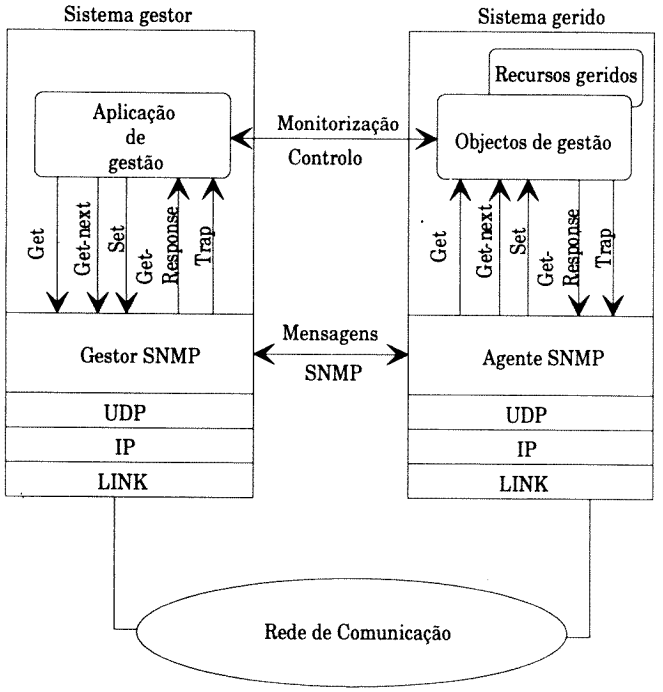


Figura 6. Arquitectura SNMP

estações de gestão e os elementos de rede. O SNMP é constituído por três componentes: o protocolo [8], a Management Information Base (MIB) [9,10], e a estrutura de informação de gestão (Structure of Management Information - SMI) [11] através da qual os objectos de gestão são definidos como variáveis escalares[12].

Os objectos de gestão são definidos usando a macro OBJECT-TYPE, através da qual é estabelecida uma relação de nomes entre os objectos de gestão, assim como a sua sintaxe e o nível de acesso[11]. Neste sentido é usado um subconjunto da linguagem Abstract Syntax Notation One (ASN.1) [11,13], como uma notação que fornece informação concisa para a implementação. O exemplo seguinte é uma definição ASN.1 de um objecto de gestão:

```
sysDescr OBJECT-TYPE
 SYNTAX OCTET STRING
 ACCESS read-only
 STATUS mandatory
 ::= {system 1}
```

A. Agente de monitorização SNMP

A comunidade Internet normalizou à partida uma colecção de objectos que constitui a MIB II [9,10] e que serve de base à gestão SNMP. No entanto, este conjunto de objectos, abrange somente uma área muito restrita dos aspectos de gestão a ter em consideração. Para superar esta limitação inicial, foram criados na comunidade Internet vários grupos de trabalho, com a tarefa de especificação de novas MIBs, visando cobrir novas áreas da gestão. Assim para a informação de monitorização foi definida e normalizada pela comunidade Internet a Remote Monitoring MIB (RMON-MIB) [14,15].

A RMON MIB endereça várias áreas da gestão como diagnóstico de falhas, análise de desempenho e planeamento. Para atingir estes propósitos, a RMON MIB possui algumas características, das quais, as mais relevantes são [15]:

- Operações "offline": A ECG pode não estar em permanente contacto com os dispositivos de monitorização, por razões de economia de comunicações ou por avaria na rede. Assim, a RMON MIB permite configurar os dispositivos de monitorização para fazerem continuamente diagnósticos e estatísticas mesmo quando não se processa qualquer comunicação.
- Historial da informação de monitorização: A RMON MIB define objectos que permitem ao dispositivo de monitorização reservar recursos e armazenar informação de monitorização correspondente a um determinado período. Em caso de avaria, a estação central de gestão pode proceder à análise deste historial e diagnosticar as causas do problema.
- Detecção e notificação de problemas: O monitor pode ser configurado para detectar condições de erro e notificar a estação central de gestão.

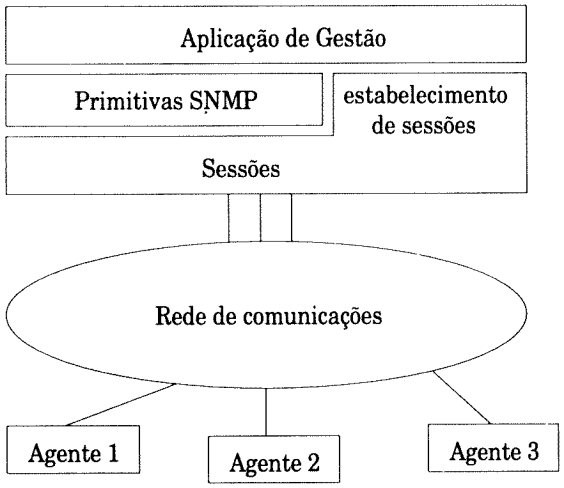


Figura 7. Gestor SNMP

- Informação acrescentada: Os dispositivos de monitorização são exclusivos para as funções de gestão. Assim, este dispositivos tem oportunidade para fazer algum processamento na informação adquirida. Por exemplo assinalar quem são os elementos da rede com maior geração de tráfego ou erros.
- Aquisição em tempo real: Os dispositivos onde a RMON MIB é instalada possuem hardware dedicado que permite a aquisição e a disponibilização de informação de monitorização em tempo real.
- Extensibilidade a outras redes: Embora grande parte dos objectos definidos na RMON MIB tenha interesse na gestão de qualquer tipo de rede, existem alguns que são específicos para a rede Ethernet. No entanto, o desenho da MIB permite a definição de objectos com características similares para outras redes, por exemplo, as redes FDDI e Token Ring.

IV. GESTOR SNMP

O gestor SNMP é constituído pelo conjunto de primitivas do protocolo SNMP que são utilizadas pela aplicação de gestão, e por um mecanismo de estabelecimento de ligações de gestão baseado no conceito de sessão. O conceito de sessão permite a inicialização e manutenção de todos os parâmetros necessários para as operações SNMP entre o gestor e cada agente de gestão, como, por exemplo, o endereço do agente e o descritor da socket para o envio dos datagramas (fig. 7).

As primitivas disponíveis no gestor são cinco e são implementadas pelas seguintes funções:

- send_get_request
- send_get_next_request
- send_set_request
- recv_get_response
- recv_trap

As primitivas do SNMP permitem à aplicação de gestão a manipulação de variáveis de gestão, as quais, do ponto de vista do protocolo SNMP, são pares (nome, valor) que podem ser transportados em listas. Assim, os parâmetros passados a cada primitiva são listas de pares (nome, valor) e também o descritor de sessão para identificar o destinatário da operação SNMP. Com estes parâmetros, cada primitiva controi uma mensagem SNMP, faz a sua codificação usando o código gerado pelos compiladores ASN.1 e em seguida procede ao seu envio para a rede.

Pelo facto da estrutura das mensagens SNMP ser igual para todas as operações (com excepção dos traps) e também para a reutilização do código, foi construída uma primitiva genérica à qual é passado um parâmetro adicional, o parâmetro tag, para a distinção entre primitivas.

```
primitive(session, names, values, nelem, tag)
{
msg = message_construct(session, names, values,
nelem, tag);
encode(msg, buffer);
sendto(session->sd, buffer, len, session->remote, ...);
}
```

Esta primitiva genérica permite o envio de primitivas get_request, get_next_request e set_request. Para a recepção de primitivas get_response e de traps o gestor SNMP contém a função recv_primitive que devolve à aplicação de gestão a mensagem SNMP recebida:

```
Message_SNMP *recv_primitive(session)
{
recvfrom(session->sd, data, ..., &remote,...);
decode(&msg, ..., &data,...);
return(msg);
}
```

A aplicação de gestão pode usar estas primitivas, competindo-lhe decidir quando as utilizar e quais os objectos a gerir. O código a seguir ilustra a simplicidade dos procedimentos necessários para uma interacção gestor-agente.

```
session = init_session(hostname, comunidade);
primitiva(session, names, valueToSend,1, get_request);
msg=recv_primitiva(session);/*recebe um get_response
*/
extract_variables(msg);
```

V. AGENTE SNMP

Em virtude de um agente SNMP poder conter diferentes conjuntos de objectos de gestão conforme a informação de gestão que dele se pretenda extrair, a presente implementação do agente constituí uma plataforma de desenvolvimento de objectos de gestão, à qual podem ser facilmente adicionados novos objectos de gestão (definição de novas MIBs) sendo apenas necessária a construção das respectivas rotinas de acesso e de um procedimento simples de inicialização.

O agente SNMP é mais complexo do que o gestor, e é composto por três módulos: a unidade de controlo SNMP, a representação da MIB e a unidade de comunicação com os recursos geridos (Fig. 8, Fig. 10).

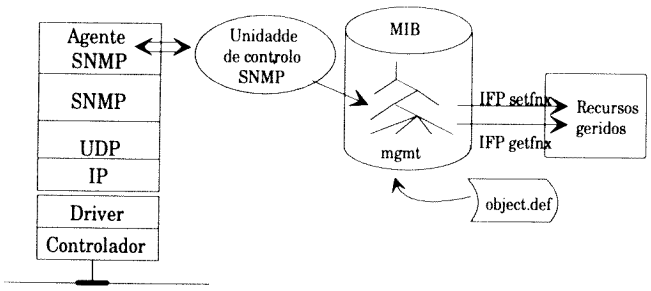


Figura 8. Agente SNMP

Para o caso particular de implementação da RMON-MIB a informação dos recursos geridos é fornecida pelo monitor de rede que é constituído pela interface de rede e pelos módulos de monitorização (Fig. 9).

A interface de rede do monitor é constituída por uma placa controladora Western Digital e por um driver Western Digital, que permite aos módulos de monitorização ler toda a informação que circula na rede. Esta escolha deve-se ao facto de possuirmos já desenvolvido um driver de software para esta placa podendo, no entanto, ser utilizada qualquer outra interface desde que ela permita aceder a toda a informação da rede.

Os módulos de monitorização têm a seu cargo a colecção de informação de monitorização e a sua organização de forma adequada aos requisitos de informação da RMON-MIB.

Por razões de modularidade, a interface e os diversos módulos de monitorização estão integrados num único programa designado por monitor, que pelo facto de constituir um programa distinto do agente, necessita do estabelecimento de um mecanismo de comunicação.

No sistema operativo DOS para simular a existência desses dois processos, o monitor após a reserva de recursos é deixado residente em memória sendo colocado a correr através das interrupções do driver Western Digital, e o programa agente é colocado a correr

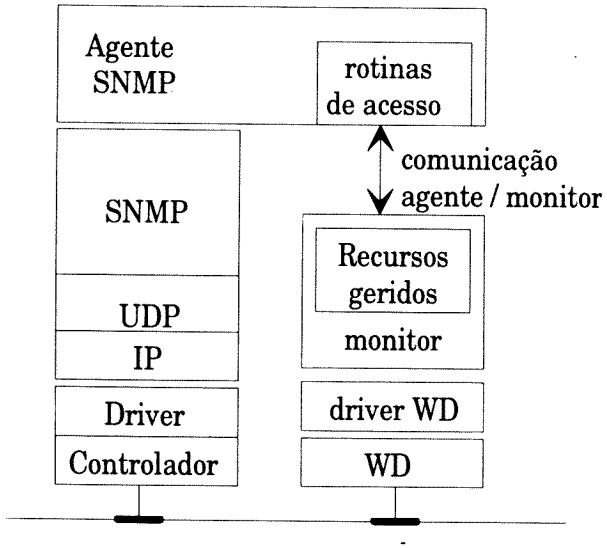


Figura 9. Agente SNMP com um monitor de rede associado

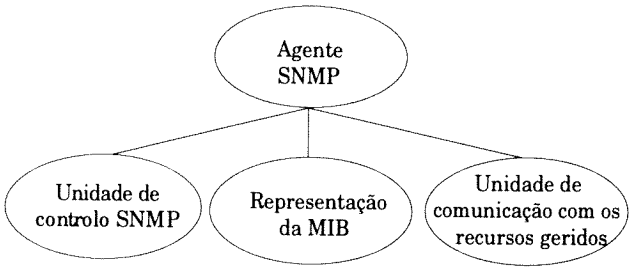


Figura 10. Diagrama de estrutura do agente SNMP

normalmente.

Como mecanismo de comunicação, é utilizada a partilha da área de memória onde é armazenada a informação de monitorização coleccionada pelo monitor. Para esta partilha é necessário que o agente tenha conhecimento da localização da área de memória a partilhar. Assim é-lhe passado através da tabela de interrupções do DOS o endereço inicial dessa área de memória. A escolha da tabela de interrupções prende-se com o facto dela conter um conjunto de posições de memória que não são alteradas quer pelo sistema operativo quer por outros programas. Para a igual interpretação dos dados são usadas as mesmas estruturas quer pelo monitor quer pelo agente.

A. Unidade de controlo SNMP

Na unidade de controlo do protocolo SNMP, o agente procede à recepção das mensagens, ao seu processamento e ao consequente envio da resposta, conforme se ilustra no seguinte fragmento de código:

```
while(1) {
 msg = recv_request(sessiond);
 process(&msg);
 send_get_response(sessiond, msg);
 }
```

O agente lê o datagrama UDP recebido e guarda o endereço do gestor, que é depois utilizado para o envio da resposta. É feita também a descodificação do datagrama para uma estrutura interna, a estrutura mensagem, sendo em seguida passada à função de processamento.

No processamento, para cada variável na lista de variáveis, é pesquisado o respectivo protótipo sendo em seguida invocada a correspondente rotina de acesso ao recurso.

Após o processamento de todas as variáveis da lista, o agente envia de volta a mensagem, que foi devidamente modificada pela função de processamento e pelas rotinas de acesso.

Durante o processamento, é na própria mensagem recebida que são actualizados os respectivos campos, isto é, o campo valor é preenchido com a informação requerida ou em caso de erro é identificado o respectivo erro.

A utilização da mesma mensagem na recepção e na emissão, é importante porque permite que um único bloco

de memória seja reservado quando a mensagem é recebida, e seja libertado logo após o seu processamento e envio. Este procedimento permite utilizar o mínimo de memória necessária e também uma gestão simples dessa memória.

B. Representação da MIB

A representação da MIB é construída a partir de um ficheiro em formato mosy (Managed Object Syntax Compiler), resultado da compilação da MIB, cuja informação é utilizada para preencher uma estrutura, a estrutura object_type, que implementa a árvore de objectos (OIT).

Os objectos de gestão são colocados numa tabela "hash", em que cada elemento da tabela é uma lista ligada com múltiplas ligações entre estruturas object_type (ot), representando cada ot um objecto de gestão, tal como ilustra a Fig. 11.

```
typedef struct object_type {
 char *ot_text;  /* descritor */
 char *ot_id; /*identificador - string */
 OID ot_name;/* identificador */
 OS ot_syntax;/* sintaxe SMI do objecto */
 int ot_access;/* acesso */
 int ot_status; /* status */
 caddr_t ot_info; /* informação do objecto */
 IFP ot_getfnx; /* metodo get / get_next */
 IFP ot_setfnx;/* metodo set */
 struct object_type *ot_chain;
 struct object_type *ot_sibling;
 struct object_type *ot_children;
 struct object_type *ot_next;
 } object_type, *OT;
```

Por razões de busca mais rápida, os objectos são colocados em diversas listas ligadas, cada uma com origem numa posição da tabela "hash". O seu posicionamento é calculado pela respectiva função de "hashing", a qual foi desenvolvida de forma a gerar um valor entre 0 e o máximo da tabela, utilizando como chave o nome dos objectos. Assim, para procurar um objecto na MIB, é necessário calcular em primeiro lugar em que lista ligada é que o objecto se encontra, sendo este um processo simples e rápido, uma vez que consiste somente na execução da função de "hashing".

Depois de localizada a lista ligada, ela deve ser percorrida até se encontrar o objecto pretendido. Por exemplo, para procurar um objecto de nome "nome_objecto" pode usar-se a seguinte linha de código:

```
OT ot;
for(ot = Hash[ FHASH( nome_objecto)];
 ot &&strcmp(ot->ot_text, nome_objecto); ot = ot->ot_chain)
continue;
/* neste ponto, ot é o protótipo do objecto procurado */
```

Os restantes ponteiros existentes permitem outras alternativas de pesquisa conhecendo por exemplo o objecto através do seu OBJECT IDENTIFIER, ou para localizar o objecto lexicograficamente a seguir para as operações de get-next-request.

Como se pode observar, a procura dos objectos é mais demorada ao percorrer-se a lista ligada, sendo portanto conveniente que as listas sejam pequenas. Assim o tamanho da tabela "hash" deve ter um comprimento adequado ao número de objectos que a MIB contém.

Depois de encontrado o objecto, pode-se aceder ao recurso associado através dos ponteiros parar função ot_getfnx e ot_setfnx.

Para o processamento das mensagens SNMP, e porque os objectos são identificados pelo seu OBJECT IDENTIFIER, usa-se também a estrutura object-instance que combina o nome do objecto e o seu protótipo:

```
typedef struct object_instance {
 OID oi_name;/* nome da instância */
 OT  oi_type; /* protótipo do objecto */
 }object_instance, *OI;
```

A representação da MIB, que se baseia nestas estruturas, assume um elevado grau de complexidade, sendo portanto necessário dotar o programador de mecanismos que auxiliem a sua manipulação. Neste sentido, existe um conjunto de funções que permitem o acesso à MIB, quer para a pesquisa de objectos quer para a conversão de nomes. Desta forma é feito o encapsulamento dos mecanismos internos à representação da MIB, sendo fornecido ao programador o seguinte conjunto de funções que constituem os métodos de acesso à MIB:

- OID text2oid(name) - converte o descritor de um objecto no respectivo OBJECT IDENTIFIER.
- OT name2obj(oid) - procura um objecto na árvore através do OBJECT IDENTIFIER e retorna o seu protótipo.
- OT text2obj(text) - procura um objecto na árvore através do descritor e retorna o seu protótipo.
- OI name2inst(oid) - procura um objecto na árvore através do OBJECT IDENTIFIER e retorna a estrutura object_instance correspondente.

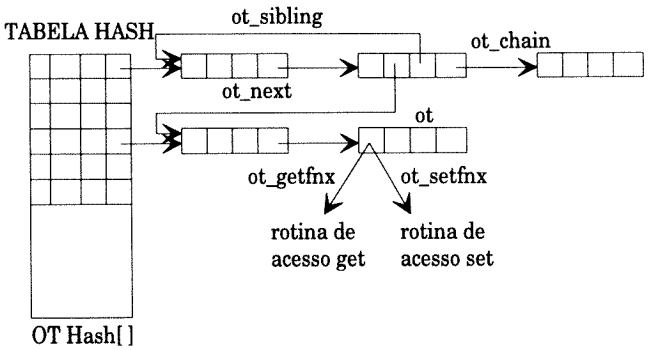


Figura 11. Disposição dos objectos na representação da MIB

- OI next2inst(oid) - procura o objecto lexicograficamente a seguir ao objecto identificado pelo seu OBJECT IDENTIFIER e retorna a estrutura object_instance correspondente.
- OI text2inst(text) - procura um objecto na árvore através do descritor e retorna a estrutura object_instance correspondente.

C. Inicialização do agente

O submódulo Inicialização procede à construção da representação da MIB recorrendo a uma função apropriada, e em seguida para cada objecto suportado, procura o correspondente protótipo object_type, e os respectivos campos ot_getfnx e ot_setfnx são inicializados com os endereços das correspondente rotinas de acesso. Normalmente estas rotinas desempenham as operações get-request e set-request para vários objectos, sendo para tal associada uma constante a cada objecto e colocado o seu valor no campo ot_info, permitindo assim às rotinas de acesso distinguir os vários objectos de que são responsáveis. Por exemplo, se as rotinas o_matrix e s_matrix forem as rotinas de acesso para os objectos de gestão do grupo matrix da RMON-MIB, a inicialização deste grupo de objectos de gestão é feita como ilustra o seguinte fragmento de código:

```
#define matrixControlDataSource 2
...
OT ot;
if ( ot = text2obj ("matrixControlDataSource")) {
ot -> ot_setfnx = s_matrix,
ot -> ot_getfnx = o_matrix,
ot -> ot_info = (caddr_t) matrixControlDataSource;
...
```

Assim, para a inclusão de novos objectos de gestão é necessário somente escrever as rotinas de acesso get e set, e efectuar as respectivas inicializações.

Para além destas inicializações, o agente lê algumas informações adicionais de configuração como:

- O nome das comunidades suportadas, e o endereço IP dos gestores;
- O nível de acesso para cada comunidade;
- A vista da MIB associadas a cada comunidade.

O nome da comunidade é uma string ASCII, e a ela devem ser associados os endereços IP dos gestores que usam essa comunidade. O endereço 0.0.0.0 permite que a comunidade seja usada por qualquer gestor.

O nível de acesso especifica para os gestores associados à comunidade se têm permissão de leitura ou escrita.

Por sua vez, a vista da MIB define a colecção de objectos de gestão que podem ser acedidos, sendo identificada por um nome, um OBJECT IDENTIFIER, e pelos nomes das subárvores que a compõem como, por exemplo, system, interfaces, matrix e statistics.

Depois de todas as inicializações o agente entra num ciclo de espera para o atendimento às mensagens SNMP que chegam ao endereço de transporte, que é normalmente a porta 161 UDP.

D. Unidade de comunicação com os recursos geridos

A unidade de comunicação com os recursos geridos é constituída pelas rotinas de acesso aos recursos. Cada objecto de gestão ou grupo de objectos de gestão possui duas rotinas de acesso: uma rotina de acesso para as operações get-request e get-next-request e outra para as operações set-request. Estas rotinas são apontadas pelos campos ot_getfnx e ot_setfnx, seguem um padrão no que se refere à sua estrutura e aos seus parâmetros, e fazem parte do trabalho de definição de novos objectos. A sua estrutura é constituída por um corpo comum que facilita a sua definição para os novos objectos a criar. Por forma a uniformizar a comunicação com os recursos geridos, as rotinas de acesso são invocadas sempre da mesma forma:

```
int result = metodo (oi, v, offset)
OI oi;
VarBind *v;
int offset;
```

Os parâmetros têm o seguinte significado:

- O parâmetro oi é do tipo object_instance, contém o nome da variável e um ponteiro para a estrutura object_type;
- O parâmetro v contém também o nome da variável, e um campo para armazenar o seu valor. A estrutura deste parâmetro é a estrutura VarBind usada na mensagem SNMP:

```
typedef struct {
  ObjectName name;
  ObjectSyntax value;
}VarBind ;
```

- O parâmetro offset, identifica a operação a ser desempenhada, pode ser get, get-next ou set.

Para as operações get-request e get-next-request o endereço de metodo é igualado ao endereço apontado por ot_getfnx, e para as operações set-request é igualado ao endereço apontado por ot_setfnx.

VI. CODIFICAÇÃO E DESCODIFICAÇÃO DE MENSAGENS SNMP

A codificação e descodificação de mensagens protocolares está directamente relacionada com a especificação dos protocolos e com os mecanismos associados ao método de especificação. No caso do protocolo SNMP, a especificação é feita em linguagem ASN.1, sendo portanto a codificação e descodificação dependente da linguagem ASN.1.

A codificação e descodificação fazem parte de um processo de transformação de sintaxes, no qual estão envolvidas por um lado a sintaxe abstracta usada na especificação e a correspondente sintaxe local e, por outro lado, a sintaxe de transferência usada para as mensagem enquanto em trânsito na rede de comunicações [16] (Fig. 12).

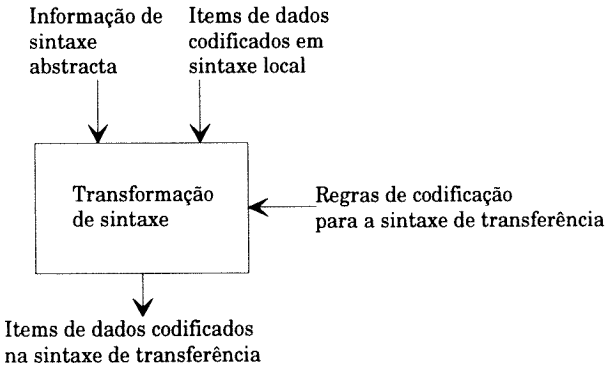


Figura 12. Transformação de sintaxe

Os procedimentos do protocolo são feitos com base na manipulação de estruturas C que constituem a sintaxe local. A partir destas estruturas a informação é codificada-descodificada utilizando-se as funções de codificação/descodificação geradas pelos compiladores de ASN.1. Estes compiladores permitem a geração automática das funções de codificação e descodificação as quais são posteriormente ligadas ao código do protocolo conforme ilustra a Fig. 13.

Sendo o processamento protocolar efectuado com base nas estruturas C, é possível tornar a máquina protocolar independente do processo de codificação/descodificação, isto é, independente das particularidades dos compiladores de ASN.1. Assim, para a plataforma SNMP de desenvolvimento de objectos de gestão foram usados os compiladores de ASN.1 do ISODE [17] e o compilador de ASN.1 desenvolvido pelo INESC[18], permitindo a sua instalação em vários sistemas operativos.

VII. CONCLUSÃO

Este trabalho enquadra-se no vasto esforço de investigação e de desenvolvimento nas áreas tecnológicas associadas à comunicação de dados em particular nos sistemas de gestão de redes.

Foi apresentado um Sistema Integrado de Gestão que integra facilidades de gestão já instaladas e as facilidades de gestão fornecidas pelos agentes de monitorização. Para este sistema foi desenvolvido uma plataforma SNMP para o desenvolvimento de objectos de gestão constituída por um gestor SNMP e por um agente de gestão SNMP. A plataforma SNMP é independente do sistema operativo e dos compiladores de ASN.1 e permite facilmente a sua evolução para a nova versão do SNMP (SNMPv2) [19].

O Sistema Integrado de Gestão e a plataforma SNMP permitem o estudo de ferramentas de gestão de redes, nomeadamente a correcta definição e implementação de objectos de gestão relevantes.

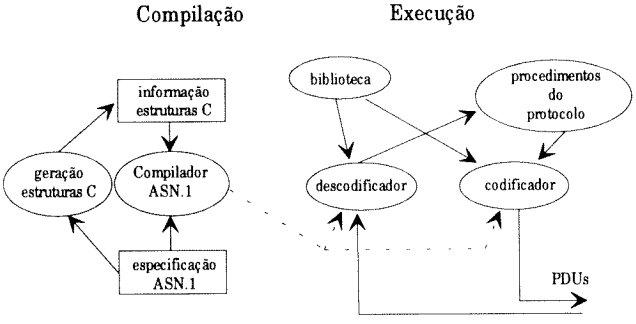


Figura 13. Relação compilação / execução ASN.1

AGRADECIMENTOS

O trabalho de Mestrado que serviu de base para este artigo, foi o projecto Greco, suportado pela JNICT através do contrato PMCT/C/TIT/458/90 e por uma bolsa de Mestrado.

Por outro, o desenvolvimento do trabalho não seria possível sem o apoio quer material quer humano do INESC, particularmente do grupo Telemática do INESC Aveiro.

REFERÊNCIAS

[1] J. Arnaldo Martins, Nelson Pacheco da Rocha, José Luis Oliveira, César Toscano "Integrated Management System for Interconnected Local Area Networks" - Mediterranean Electrotechnical Conference, Melecon'91, Ljubljana, Juguslávia, Maio 1991

[2] Abílio Pacheco de Carvalho, Nuno Eduardo da Silva, Nelson Pacheco da Rocha, Joaquim Sousa Pinto, "Sistema Distribuído de Monitorização de Redes Locais Ethernet", "Comunicações" (III Congreso Português das Comunicações), APDC, Fevereiro 1991.

[3] "ISO IS9595-1: Information Processing - Open Systems Interconnection - Management Information Service Definition - Part 1: Overview", ISO, 1990.

[4] Abílio Pacheco de Carvalho, Nelson Pacheco da Rocha, "Greco- Gestão de Redes de Comunicação, Sistema de Display Local e Remoto" INESC, Julho 1992.

[5] J. Sousa Pinto, José L. Oliveira, J. Arnaldo Martins, "PC Workstation for Ethernet Monitoring and Management", SBT/IEEE, International Telecommunications Symposium, Brazil, 3-6 Sep. 1990.

[6] Marshall T. Rose, "The Simple Book, An Introduction to Management of TCP/IP - based Internets", Prentice Hall, 1991.

[7] Ben-Artzi Amatzia, Chandna A., Warrier U., "Network Management of TCP/IP Networks: Present and Future", IEEE Network Magazine, Julho de 1990.

[8] Jeffrey D. Case, Mark S. Fedor, Martin L. Schoffstall, and James R. Davin, "A Simple Network Management Protocol". Request for Comments 1157, DDN Network Information Center, SRI International, Maio 1990.

[9] Marshal T. Rose, Keith McCloghrie, "Concise MIB Definitions". Request for Comments 1212, Performance Systema International, Hughes LAN Systems, Março 1991.

[10] Marshall T. Rose, "Management Information Base for Network Management of TCP/IP-based Interntes: MIB-II, Request for Comments 1213, Performance Systems International, Março 1991.

[11] Marshal T. Rose, Keith McCloghrie, "Struture and Identification of Management Information for TCP/IP-based Internets", Request for Comments 1155, DDN Network Information Center, SRI International, Maio 1990.

[12] Nelson Pacheco da Rocha, Abílio Pacheco de Carvalho, José Luis Oliveira, "Defining Managed Objects for an Integrated Management System with Monitoring Agents", ERCIM Workshop on Network Management, Heraklion, Creta, Grécia, Outubro 1992.

[13] Douglas Steedman, "ASN.1 - Abstract Syntaxe Notation One, The Tutorial and Reference", Tecnology Appraisals, 1990.

[14] Mark Hoerth, "The RMON MIB - a New Standard for Remote Monitoring of Networks", Telecommunications, March 1992.

[15] Steven Waldbusser, "Remote Network Monitoring MIB", Request for Comments 1271, Carnegie Mellon University, November 1991.

[16] Marshall T. Rose, "The Open Book, A Pratical Perspective on OSI", Prentice Hall, 1990.

[17] Marshall T. Rose, "The ISO Development Environment - ISODE - User`s Manual".

[18] João Filipe Serra da Silva, Vitor Manuel Paulino Vargas "Compilador de ASN.1 baseado em informação em linguagem C", Manual do programador (versão 3.0), INESC, Março 1991.

[19] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose and Stenven Waldbusser, "Coming Soon to a Network Near You", Data Communication, Novembro 1992.

Monitorização e Visualização de Recursos num Ambiente Distribuído

Pedro Miguel Costa, Maurício Frade Domingues,
Joaquim Arnaldo Martins, José Luís Oliveira

Resumo- **Esta apresentação reporta um conjunto de estudos efectuados sobre as funcionalidades de determinadas interfaces, ao nível físico e lógico para receber ou enviar informação de gestão sobre um meio de comunicação. Com esse intuito serão discutidas as potencialidades do *Packet Filter* e *Data Link Interface* (DLI).**

Apresenta-se igualmente um trabalho, com base no *Simple Network Management Protocol* (SNMP), relacionado com a *Remote Monitoring* MIB, no qual se irá descrever a implementaçao de dois dos seus grupos, nomeadamente os grupos *alarm* e *event*.

Abstract- **This presentation reports a number of studies about the functionalities of some commercial interfaces, at physical and logical levels, in order to receive or send management information through a communication medium. With this purpose we will be discuss the potentialities of both de Packet Filter and Data Link Interface (DLI).**

We will also present the Simple Network Management Protocol (SNMP) and a work related with the Remote Monitoring MIB, namely the implementation of two of its groups, the "alarm" amd "event" groups.

I. INTRODUÇÃO*

Começamos por fazer um estudo das potencialidades de duas interfaces existentes no sistema operativo ULTRIX, Packet Filter e Data Link Interface (DLI) tendo em vista a apresentação de um sistema capaz de recepção e transmissão de pacotes que possa dar resposta às necessidades impostas pelos vários grupos que compõem a RMON-MIB - *Remote Monitoring Management Information Base*.

O protocolo de comunicação Simple Network Management Protocol já deu provas de ser uma solução real para a troca de informação de gestão entre sistemas heterogéneos, isto é, de fabricantes diferentes. O protocolo SNMP, não só por estar disponível, mas também por ser muito simples, tal como o próprio nome indica, é de momento o que mais se aplica à criação de sistemas de gestão e monitorização de redes. Aqui irá ser feito um estudo sobre este protocolo, e em seguida descrever-se-á uma implementação de dois grupos da RMON-MIB: *alarm* e *event*, responsáveis, respectivamente, pela monitorização de variáveis e geração de eventos aquando da ocorrência de determinadas condições pré-definidas.

Finalmente tecer-se-ão algumas considerações respeitantes à implementação destes dois grupos.

* Trabalho realizado no âmbito da disciplina de projecto.

II. PACKET FILTER

O Packet Filter [1] é um pseudo-device driver, suportado pelo sistema operativo ULTRIX, para criação de canais de acesso a redes do tipo Ethernet e semelhantes.

A estrutura de software do *Packet Filter* é ilustrada na Fig. 1. Observando esta estrutura verificamos que as os componentes principais são o *device driver* e o *Packet Filter* que fazer parte do kernel do sistema operativo.

O protoloco RARP (`rarpd`) está aqui ilustrado como exemplo de utilização deste mecanismo. Utiliza a interface para ler e escrever pedidos RARP trabalhando directamente ao nível de rede. Um determinado filtro ao detectar pacotes RARP, passa-os para o *daemon* `rarpd` para posterior processamento.

A *Monitor aplication* diz respeito a outras aplicações definidas pelo utilizador, como por exemplo aplicações de monitorização do tráfego da rede.

A. *Configuração do Packet Filter*

Para que se possa criar uma aplicação com base no packet filter é necessário reconfigurar o kernel, da seguinte forma:

1. Adicionar as seguintes entradas na file de configuração do sistema, existente no directório `/sys/conf/{mips,vax}/HOSTNAME`:

```
optionsPACKETFILTER
.
.
.
pseudo-device packetfilter
```

2. Reconstituir o kernel com o comando `/etc/`

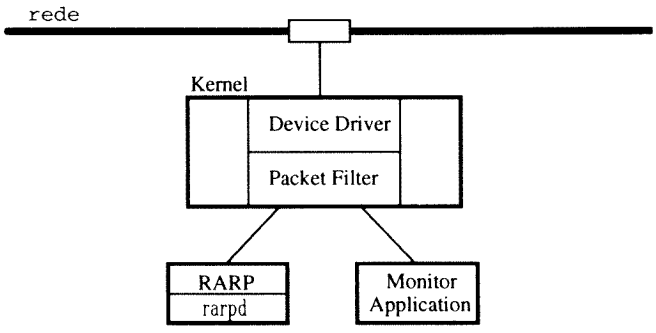


Fig.1 - Estrutura de software do Packet Filter

```
doconfig -c.
```

3. Verificar se `/dev` inclui o directório `pf`. Caso não contenha, terá de se executar o comando `MAKEDEV` com a opção `pfilt`. Este comando cria 64 *packet filter character special files* em `/dev/pf`, com os nomes `pfilt`**n**, em que **n** representa o número da file.

B. Programação do Packet Filter

Cada processo de aplicação do packet filter faz uso de uma ou mais *character special files* para obter acesso à rede.

A abertura de cada uma destas *special files* é feita usando a rotina `pfopen`. A este procedimento está associado um programa implementado pelo utilizador, que permite configurar o modo de recepção e leitura dos pacotes, bem como seleccionar o tipo de pacotes que se desejam receber, através de um mecanismo de filtragem.

Os pacotes capturados pela interface associada à aplicação em causa, permanecem numa fila de entrada de tamanho limitado, até que sejam lidos. Se esta fila de entrada se encontrar completa, então os pacotes são removidos.

A leitura de pacotes é feita por meio da *system call* `read` que lê os pacotes pela ordem em que se encontram na fila de entrada.

O envio de pacotes para a rede, por meio das *special files*, é feito utilizando a *system call* `write`, o qual envia exactamente um pacote, cada vez que é chamado. A construção do pacote é da responsabilidade da aplicação.

C. Filtragem de pacotes

A filtragem de pacotes é feita através de uma linguagem específica deste sistema. Cada lista de comandos do filtro especifica uma sequência de acções, que colocam um valor lógico no topo de um stack interno. Cada palavra da lista de comandos especifica uma acção e um operador binário.

As acções são: PUSHLIT, coloca a próxima palavra no stack; PUSHZERO, coloca um zero no stack; PUSWORD+N coloca a palavra N do pacote de entrada no stack. Os operadores põem o resultado da operação de dois elementos no topo do stack e são: EQ, LT, LE, GT, GE, AND, OR, XOR.

A associação de um dado filtro a uma *character special file* previamente aberta é feita por intermédio da *system call* `ioctl` da seguinte forma:

```
ioctl(fdes,EIOCSETF,filtro)
struct enfilter *filtro

struct enfilter
{
 u_char enf_Priority;
        // prioridade do filtro
 u_char enf_Filterlen;
        // comprimento do filtro
        // em shortwords
 u_short enf_Filter[ENMAXFILTERS];
        // lista de comandos do filtro
}
```

Um exemplo de um filtro destinado a aceitar pacotes do tipo IP provenientes da rede Ethernet 10Mb (Fig. 2) é apresentado de seguida.

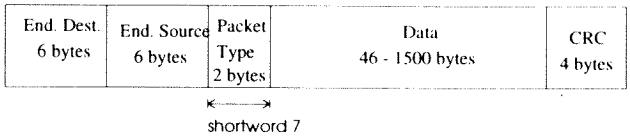


Fig. 2 - Pacote Ethernet 10Mb

```
struct enfilter filtro =
{
 10,1,// prioridade, número de
      // swortwords
      // utilizados na filtragem
 ENF_PUSHORD + 6,
      // Coloca a shortword 6 do
      // pacote de entrada no stack
 ENF_PUSHLIT, 0x0008,
      // Coloca a shortword 7 no
      // stack e compara-a
      // com o tipo IP (0x0800)
 ENF_EQ
      // Se for igual aceita
      // o pacote
}
```

D. Recepção em modo promíscuo

A recepção de pacotes em modo promíscuo possibilita ao utilizador a leitura de todo o tipo de pacotes circulantes na rede. Para isso é necessário realizar os seguintes passos:

1. Configurar o sistema com o comando `pfconfig` da seguinte forma:
   `#pfconfig +promisc [interface-name]`
2. No programa de aplicação utilizar o *system call* `ioctl` com os seguintes parâmetros:
   `ioctl(fdes,EIOCMBIS,ENPROMISC);`

Desta forma, se a *packet filter descriptor* (fdes) estiver aberta, a interface associada ficará programada para recepção em modo promíscuo.

E. Campos de informação

O packet filter possui ferramentas que nos permitem obter vários tipos de informação.

Através do pedido `ioctl(fdes,EIOCMBIS, ENTSTAMP)` cada pacote recebido vem precedido por um *header*, que está armazenado numa estrutura `enstamp` que contém os seguintes campos de informação:

- **`ens_stamplen`** - indica o comprimento da estrutura.
- **`ens_flags`** - contém informação relativa ao modo de recepção de cada pacote: modo promíscuo, *broadcast, multicast ou trailer encapsulation.*

- **ens_count** - Contém o comprimento do pacote em bytes, não incluindo o *header*.
- **ens_dropped** - Número de pacotes perdidos devido à fila de entrada se encontrar completa. Esta variável é um contador acumulativo desde que a `enstamp` anterior tenha sido lida da mesma *packet filter special file*.
- **ens_ifoverflows** - Número total de *overflows* de entrada desde que o sistema foi inicializado.
- **ens_tstamp** - Hora aproximada em que o pacote foi recebido.

Outros tipos de informação encontram-se nas seguintes estruturas:

- **endevp** - contém dados relativos á interface, que está ligada a *packet filter special file*, os quais podem ser acedidos através do comando `pfsat(8)`.
- **eniocb** - fornece-nos parâmetros sobre a *character special file* que está a ser utilizada.

F. Conclusões

O acesso à informação, aliado à possibilidade de recepção em modo promíscuo, faz com que o Packet Filter seja uma ferramenta bastante útil, no que diz respeito a análise global de tráfego da rede.

O facto do seu código residir no kernel confere-lhe um desempenho bastante aceitável mas simultaneamente insuficiente se pretendermos construir aplicações de monitorização em modo promíscuo. Com uma taxa de recepção de pacotes máxima de, aproximadamente, de 112 Kbytes/s (~1Mbit/seg) o *Packet Filter* revela-se com pouca capacidade para acompanhar taxas de utilização acima dos 10%. Contudo, recentes versões do *Packet Filter* têm surgido sendo de referir a *Network Interface Tap* (NIT) [2] e o *BSD Packet Filter* (BPF) [3] com desempenhos superiores à versão estudada mas mesmo assim incapazes de captar toda a informação se a taxa de utilização rondar os valores máximos admissiveis.

III. DATA LINK INTERFACE (DLI)

A DLI [4] é uma interface de nível lógico que permite o desenvolvimento de protocolos de comunicação entre diferentes sistemas.

A. Serviços suportados pela DLI

Ao nível da camada dois a DLI suporta os seguintes serviços:

- Tipo de transmissão: datagrama.
- Formato da trama: Ethernet ou IEEE 802-3.
- ISO 8802-2: Ao nível da subcamada superior do nível 2, LLC, permite serviços Classe I, Tipo I *(connectionless)*.
- Endereços: Permite o uso de endereços *multicast*.

B. Serviços a suportar pela aplicação

A DLI, sendo uma interface que acede directamente à camada 2, não fornece serviços próprios de camadas mais elevadas. Deste modo, as aplicações desenvolvidas com base na DLI devem assegurar os seguintes serviços:

- Encaminhamento de pacotes
- Controlo do fluxo dos pacotes
- Recuperação de erros
- Segmentação da informação a transmitir

C. Desenvolvimento de aplicações

O desenvolvimento de aplicações exige um conhecimento profundo da system call *socket*, e de todas as que lhe estão associadas : `socket`, `bind` e `close`, relacionadas com o canal de transmissão; `write`, `send` e `sendto`, relacionadas com a transmissão de pacotes; `read`, `recv` e `recvfrom`, relacionadas com a recepção de pacotes; `setsockopt`, relacionada com as opções da socket.

A DLI fornece uma estrutura de dados através da qual é possível configurar os serviços necessários à comunicação ao nível da camada dois. A estrutura `sockaddr_dl` é usada para transferir informação para a DLI quando quando uma aplicação executa a primitiva `bind`, ou quando transmite um pacote para a rede. A DLI usa também esta estrutura para transferir informação para a aplicação quando esta recebe um pacote da rede. No programa do utilizador esta estrutura irá conter todos os dados necessários à execução do `bind()`, nomeadamente:

- Familia de endereços: AF_DLI.
- Device: Identificação do device utilizado (por exemplo ln0).
- Formato da trama: Ethernet ou 802.
  De acordo com o especificado neste campo, será utilizada a estrutura correspondente: `sockaddr_edl` (que será aqui considerada) ou `sockaddr_802`.
  Na estructura `sockaddr_edl` há a considerar os seguintes campos :

  - A Flag `(dli_ioctlflg)` que especificará o modo como os pacotes serão filtrados. Os modos possíveis são :
    NORMAL - Permite ao programa a troca de pacotes com apenas um nó Ethernet. Com esta *flag* activa o endereço da máquina destino deve ser especificado à partida, de modo a ser utilizado pela *system call* `bind()` na conecção à socket. A aplicação receberá todos os pacotes com o seu endereço, desde que o endereço da fonte e o protocolo sejam os especificados na `bind()`. Duas aplicações com esta *flag* activa comunicarão exclusivamente uma com a outra .
    Neste modo de operaçao a aplicação poderá utilizar todas as *system calls* para transferência de informação referidas acima.
    EXCLUSIVE - Dá ao programa uso exclusivo do protocolo especificado, permitindo-lhe comunicar

com qualquer outro programa que utilize o mesmo protocolo. Ao contrário do modo NORMAL, o envio e recepção de mensagens só é possível através da utilização das *system calls* `recvfrom()` e `sendto()`. Como na chamada da `bind()` o endereço destino não é especificado (sendo preenchido com um NULL), dado que vários destinos e várias fontes são permitidos, o programa terá que usar `recvfrom()` para saber o endereço da fonte de cada pacote, usando depois esse endereço para enviar uma resposta através da *system call* `sendto()`.

DEFAULT - Quanto ao uso das *system calls* é similar ao caso anterior. Também como no caso anterior, só o protocolo é verificado. A diferença fundamental é que o programa só recebe o pacote se mais nenhum outro o aceitar, ou seja, se o protocolo não for exclusivo de outro programa (que tenha esta flag a EXCLUSIVE), ou o par endereço/protocolo não coincida com o de outra aplicação (*flag* a NORMAL).

- Tipo de protocolo utilizado.
- Endereço físico da máquina destino.

Quando não especificado, (`ioctlflg` EXCLUSIVE ou DEFAULT) o programa utiliza `recvfrom()` para determinar o endereço fonte de cada pacote, utilizando esse mesmo endereço para, através de `sendto()`, enviar uma resposta.

Para o desenvolvimento de aplicações é necessário que o utilizador tenha permissões de administrador do sistema.

Esta interface não é a mais adequada à análise de tráfego da rede por não permitir a recepção de pacotes em modo promíscuo.

IV. SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

O SNMP [5] é um protocolo de gestão de redes TCP\IP. Foi desenvolvido pela comunidade Internet e cedo revelou uma grande aceitação por parte dos fabricantes. A sua relativa simplicidade levou ao recente aparecimento de uma segunda versão conhecida por SNMPv2 que se espera vir colmatar algumas das lacunas da sua antecessora versão, nomeadamente ao nível da segurança.

Numa gestão baseada no protocolo SNMP existem basicamente quatro componentes:

- Os agentes que representam dispositivos ou equipamentos geridos.
- Os gestores - NMS (Network Management Stations) os quais são regra geral computadores centrais que fazem a gestão da rede.
- O protocolo que descreve a forma como os gestores e os agentes comunicam entre si.
- O tipo e o formato da informação trocada (MIB [6] e SMI [7]).

O SNMP faz uso do protocolo de transporte UDP sobre IP -*User Datagram Protocol/Internet Protocol* - o qual não requer o estabelecimento de uma associação lógica entre sistemas. Como consequência, é necessária a identificação (endereço IP + porta UDP) do emissor e do destinatário em cada pacote enviado. Embora apresente inconvenientes, como por exemplo ausência de confirmação de recepção, é utilizado na implementação de diferentes redes dada a sua simplicidade.

Para que uma NMS interactue com vários agentes, o protocolo SNMP suporta vários PDUS (Protocols Data Units) os quais permitem ao gestor, não só obter valores de variáveis (Get) mas também, alterar esses valores (Set). Por outro lado, os agentes além de responderem aos pedidos de informação vindas do gestor, informam-no da ocorrência de situações anómalas através de mensagens não solicitadas (traps).

A Fig. 3 representa a arquitectura de um agente e um gestor SNMP, na qual se pode visualizar o conjunto de serviços disponível [8].

O standard SNMP especifica como as mensagens SNMP são transportadas sobre UDP/IP. Contudo as mensagens SNMP têm a possibilidade de serem transportadas sobre qualquer mecanismo de transporte que suporte trocas bidireccionais e apresente capacidades de endereçamento. Como prova disso encontram-se implementações SNMP sobre TCP, Ethernet, LLC, Appletalk, etc.

Sob o ponto de vista de organização administrativa, o SNMP recorre ao conceito de comunidade para obter autenticação de acesso entre o gestor e o agente.

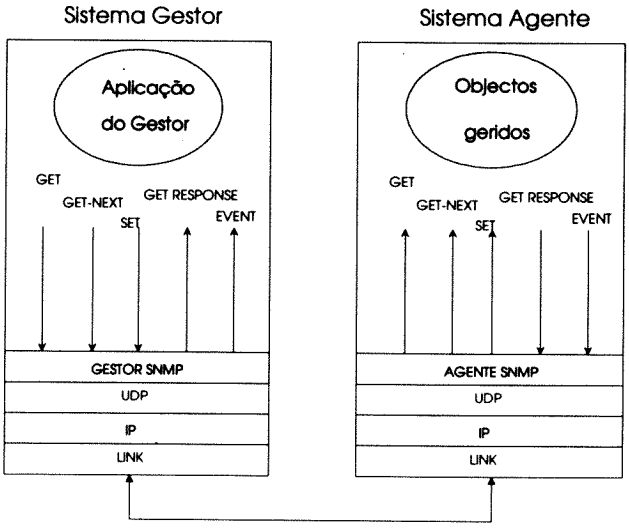


Fig. 3 - Comunicação Agente-Gestor SNMP

A. Management Information Base (MIB)

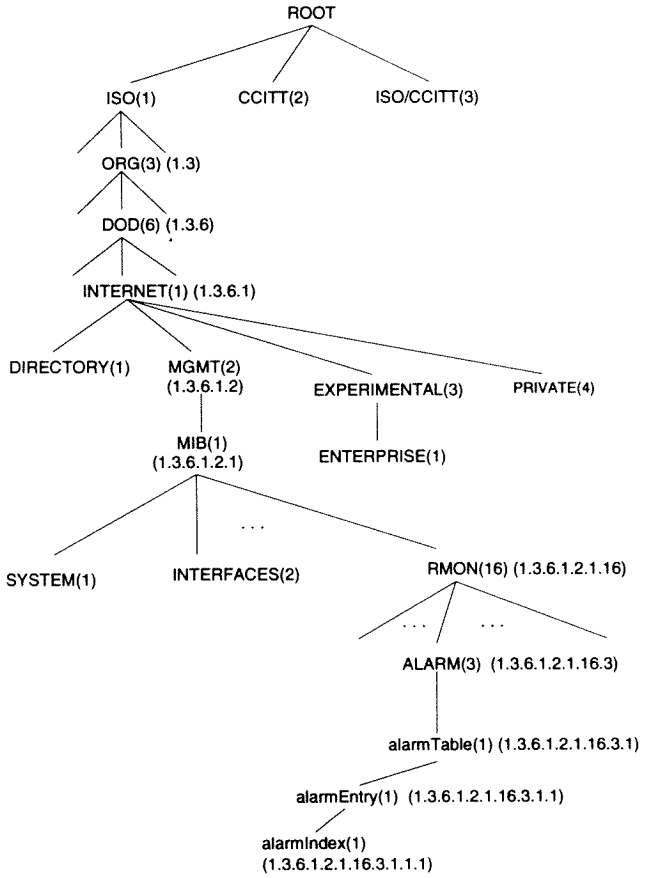
A MIB associada ao protocolo SNMP está organizada como uma base de dados hierárquica, surgindo assim, como uma árvore n-ária em que cada folha representa um objecto.

Cada agente contém uma estrutura de informação normalmente designada por MIB que, não é mais do que uma lista de variáveis.

A informação de gestão é estruturada em objectos, os quais apresentam determinados atributos. Cada objecto terá de possuir:

- Um nome que o identifica inequivocamente.
- Uma sintaxe, a qual é descrita pelo recurso à linguagem ASN.1.
- Uma codificação, isto é, o modo como o objecto é identificado quando da sua transmissão pela rede - identificador .

Como se pode ver na Fig. 4 a partir da raiz (root) da árvore há três nós administrativos distribuídos respectivamente pelà ISO, pelo CCITT e um terceiro em simultâneo pela ISO e CCITT. Toda a subdivisão da árvore a partir daí, é da inteira responsabilidade do organismo que a originou. Como exemplo, é ilustrada a localização do objecto *alarmIndex* pertencente ao grupo *alarm* da RMON-MIB.



V. REMOTE MONITORING MANAGEMENT INFORMATION BASE (RMON MIB)

A RMON MIB [9] foi a primeira tentativa da comunidade Internet para normalizar os monitores de redes baseados em TCP/IP. Permite simplificar técnicas de diagnóstico de falhas na rede e de planeamento, assim como um apuramento do desempenho global do processo de monitorização.

A RMON MIB está organizada em nove grupos :

- **statistics** - armazena dados estatísticos com base no número de pacotes, bytes, *broadcasts*, *multicasts*, erros e colisões ocorridos num segmento local da rede Ethernet.

  Este grupo está organizado numa única tabela: *etherStatsTable*.
- **history** - este grupo faz periodicamente uma análise da rede. Os dados obtidos são armazenados para posterior utilização.

  Este grupo está organizado em duas tabelas: *historyControlTable* e *etherHistoryTable*.
- **alarm** - obtém amostras do valor de variáveis e compara-os com limiares pré-definidos, gerando eventos quando estes são ultrapassados. Requer a implementação do *event group*.

  Este grupo está organizado numa única tabela: *alarmTable*.
- **host** - contém dados relativos à todas as máquinas da rede (por exemplo pacotes enviados e recebidos, *multicasts*, *broadcasts* ou pacotes com erros enviados). Este grupo mantém actualizada uma lista das máquinas que fazem parte da rede através da leitura dos campos de endereços dos pacotes que circulam na rede (só os pacotes sem erros são considerados).

  O grupo está organizado em três tabelas: *hostControlTable*, *hostTable* e *hostTimeTable*.
- **hostTopN** - utilizando os dados estatísticos armazenados pelo grupo Host cria listas ordenadas de máquinas pertencentes à rede de acordo com parâmetros a definir pelo gestor, nomeadamente : interface, momentos inicial e final do período de amostragem, número de máquinas e o dado estatístico que funcionará como chave para a ordenação.

  Este grupo está organizado em duas tabelas: *hostTopNControlTable* e *hostTopNTable*. Requer a implementação do grupo Host.
- **matrix** - armazena informação relativa a comunicações entre pares de endereços. Sempre que uma nova comunicação é detectada, é criada uma nova entrada na respectiva tabela. A informação pode ser ordenada de acordo com os endereços fonte ou destino.

  Este grupo está organizado em três tabelas: *matrixControlTable*, *matrixSDTable* e *matrixDSTable*.
- **filter** - permite filtrar pacotes por meio dos seus bits constituintes e/ou do seu *status* . Um pacote que passe o teste de filtragem pode ser capturado ou gerar um evento.

  Este grupo está organizado em duas tabelas: *filterTable* e *channelTabel*.
- **packet capture** - permite capturar pacotes que tenham sido previamente aceites pelo grupo Filter e direccionados para um canal. A função deste grupo é controlar a saída de vários desses canais.

  Está organizado em duas tabelas: *bufferControlTable* e *captureBufferTable*. Requer a implementação do grupo Filter.
- **event** - controla a geração e notificação de eventos.

  Está organizado em duas tabelas: *eventTable* e *LogTable*.

A adaptação da RMON MIB para outras redes, que não só a Ethernet, tem sido objecto de estudo. Recentemente surgiu um documento [10] que define já a TR RMON

MIB, ou seja uma RMON para redes Token Ring esperando-se uma futura extensão a redes FDDI.

Dos nove grupos acima referidos, dois deles irão ser tratados em pormenor, nomeadamente os grupos *alarm* e *event*.

A. Grupo alarm

Este grupo destina-se à monitorização de variáveis de grupos pertencentes, ou não, à RMON MIB.

Para um conhecimento dos objectos que compõem este grupo apresenta-se de seguida, a tabela I que refere o nome, sintaxe e tipo de acesso correspondente a cada objecto.

TABELA I
OBJECTOS CONSTITUINTES DO GRUPO *ALARM*

OBJECT-TYPE	SYNTAX	ACCESS
alarmTable	SEQUENCE OF AlarmEntry	not-accessible
alarmEntry	AlarmEntry	not-accessible
alarmIndex	INTEGER (1..65535)	read-only
alarmInterval	INTEGER	read-write
alarmVariable	OBJECT IDENTIFIER	read-write
alarmSampleType	INTEGER	read-write
alarmValue	INTEGER	read-only
alarmStartupAlarm	INTEGER	read-write
alarmRisingThreshold	INTEGER	read-write
alarmFallingThreshold	INTEGER	read-write
alarmRisingEventIndex	INTEGER (1..65535)	read-write
alarmFallingEventIndex	INTEGER (1..65535)	read-write
alarmOwner	OwnerString	read-write
alarmStatus	EntryStatus	read-write

Durante a monitorização, o valor do objecto que está a ser monitorizado, definido em *alarmVariable*, é lido periodicamente de acordo com o valor do objecto *alarmInterval* e em seguida comparado com os limiares previamente definidos em *alarmRisingThrehold* e *alarmFallingThreshold*. Quando um desses limiares for ultrapassado desencadeia-se uma acção específica, cuja execução é da responsabilidade do grupo *event*, sendo por isso requerida a sua implementação.

É de referir que apenas poderão ser monitorizadas variáveis existentes na implementação do agente em causa e do tipo INTEGER, Gauge, Counter ou TimeTicks.

B. Grupo event

Os objectos pertencentes a este grupo encontram-se na tabela II, onde se indica a sintaxe e o tipo de acesso correspondentes a cada objecto. Note-se que este grupo é constituído por duas tabelas, nomeadamente *eventTable* e *logTable*.

TABELA II
OBJECTOS CONSTITUINTES DO GRUPO *EVENT*

OBJECT-TYPE	SYNTAX	ACCESS
eventTable	SEQUENCE OF EventEntry	not-accessible
eventEntry	EventEntry	not-accessible
eventIndex	INTEGER (1..65535)	read-only
eventDescription	DisplayString (Size (0..127))	read-write
eventType	INTEGER	read-write
eventCommunity	OCTET STRING (Size (0..127))	read-write
eventLastTimeSent	TimeTicks	read-only
eventOwner	OwnerString	read-write
eventStatus	EntryStatus	read-write
logTable	SEQUENCE OF LogEntry	not-accessible
logEntry	LogEntry	not-accessible
logEventIndex	INTEGER (1..65535)	read-only
logIndex	INTEGER	read-only
logTime	TimeTicks	read-only
logDescription	DisplayString (Size (0..255))	read-only

Este grupo controla a geração e notificação de eventos. A notificação pode ser efectuada de duas formas : logs e/ou SNMP traps. O registo (*log*) de ocorrências de um determinado evento é efectuado pelo agente criando uma entrada na tabela de logs a ele associada. Se o gestor pretender fazer o registo dos eventos ocorridos, à medida que eles forem sucedendo serão criadas novas entradas na tabela de logs correspondentes aos eventos. O número de entradas na tabela de logs relativas ao mesmo evento deverá ser controlado de acordo com a quantidade de memória disponível. Nesta implementação apenas as últimas cinquenta entradas correspondentes a um dado evento estarão disponíveis.

Este grupo está intimamente relacionado com os grupos *alarm* e *filter*. Sem a implementação de pelo menos um deles, a implementação do Event group não tem qualquer utilidade.

O envio de notificações é efectuado quando o objecto *eventType* possui o valor 3 ou 4.

VI. NOTAS REFERENTES À IMPLEMENTAÇÃO DOS GRUPOS *ALARM* E *EVENT*

A. Descrição do funcionamento da monitorização

O diagrama de blocos apresentado na Fig. 5 esquematiza as operações básicas executadas no processo de monitorização.

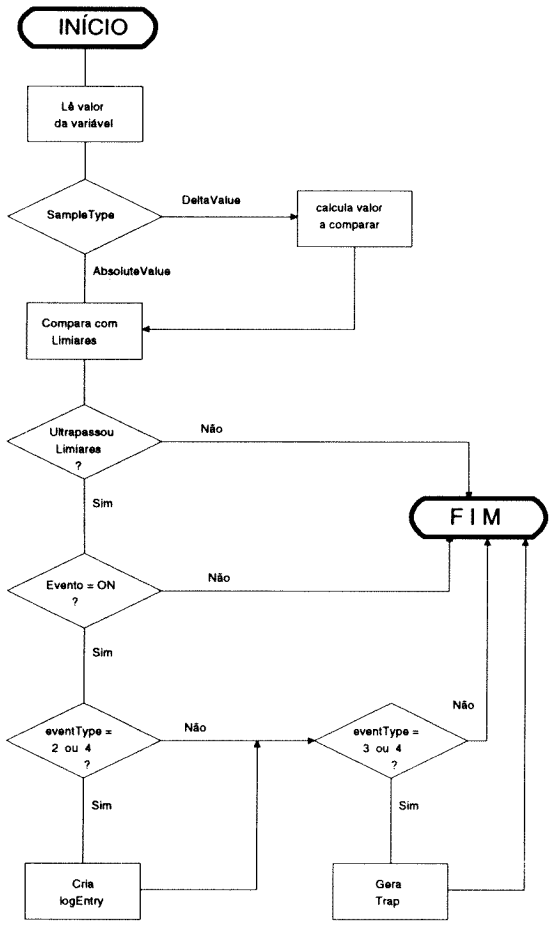


Fig. 5 - Fluxograma do processo de monitorização de uma variável

A rotina que implementa a monitorização é responsável pela realização das seguintes operações:

- Obter o valor da variável, através de um get interno, cujo identificador (OID) está contido em *alarmVariable.*
- Verificar se o mecanismo de histerese, regulador da geração de eventos, já foi inicializado. Este mecanismo consiste em não permitir que dois eventos consecutivos do mesmo tipo sejam gerados. A inicialização deste mecanismo é feito através do objecto *alarmStartUpAlarm*, cujo valor determina qual o tipo de evento que deverá ser gerado após uma entrada ter sido validada.
- Calcular o valor a comparar com os limiares. Para isso testa o valor de *alarmSampleType:* se for igual a *AbsoluteValue(1)* o valor lido é o valor a comparar, se for *DeltaValue(2)* serão feitas duas amostragens em cada período definido por *alarmInterval.* O valor a comparar será o módulo da diferença entre o valor actual e o valor lido na amostragem anterior.
- Comparar o valor obtido com os limiares pré-definidos. Se o valor a comparar for maior/menor ou igual a *alarmRisingThreshold/alarmFallingThreshold* e houver permissão para gerar eventos será gerado um evento cujo índice corresponde ao valor de *alarmRisingEventIndex/ alarmFallingEventIndex.*

B. Criação de entradas nas tabelas

É de todo o interesse que o gestor possa criar novas entradas nas tabelas relativas a alguns grupos. Um número fixo de entradas criadas pelo agente, que poderão ser oportunamente preenchidas pelo gestor, revela-se um processo pouco eficiente, pois é difícil prever as necessidades futuras. O esquema definido para a implementação de grupos na RMON MIB [11] possibilita ao gestor criar dinamicamente novas entradas em tabelas. Este esquema utiliza o valor 2 (under create) do objecto de status da respectiva entrada, o qual permite ao gestor requerer a criação de uma nova entrada.

Por razões de gestão de memória, o agente deverá estabelecer um limite máximo para o número de entradas. Se esse limite for atingido e, em seguida, for feito um pedido de criação de uma nova entrada, fazendo um *set* do status a dois, o agente deverá responder com noSuchName.

C. Eliminação de entradas nas tabelas

Uma entrada pode ser eliminada da tabela atribuindo ao objecto status respectivo o valor 4, através de uma operação de set.

Sempre que o status de uma dada entrada permanecer com o valor 3 (under create) durante um tempo que exceda um período previamente definido por quem implementou o agente, a entrada correspondente será apagada. Este procedimento evita que entradas estejam activas sem que delas se tire qualquer proveito, libertando assim recursos do sistema.

D. Inicialização dos objectos

O documento que especifica a RMON MIB define valores iniciais para apenas um número muito reduzido de objectos. Por esta razão, cabe ao implementador decidir qual o valor de certos objectos no momento em que são criados. Essa escolha tem como base apenas o respeito pela sintaxe dos objectos, já que a diversidade de comportamentos das variáveis que podem ser monitorizadas, afasta a hipótese da utilização de valores típicos para todos os objectos. A estratégia seguida nesta implementação consistiu em, por um lado atribuir um conjunto de valores que permitisse ao gestor efectuar o mínimo de operações de set, e por outro não lhe permitir validar a entrada sem antes configurar um determinado número de objectos. Por esta razão, serão feitos alguns comentários aos valores utilizados.

- **alarmInterval** - valor inicial: 2 (segundos)
  Como o número de entradas activas permitidas nesta implementação é de 50, um período de amostragem menor poderia provocar uma sobrecarga do agente.
- **alarmVariable** - valor inicial: 0.0

Esta variável contém o *Object Identifier* da variável a ser monitorizada. Como é óbvio, este valor não corresponde a nenhum objecto da RMON MIB. Este valor apenas pretende elucidar quanto ao tipo de sintaxe do objecto. Por não ser um valor válido, o agente só permite que o status desta entrada seja posto a *valid* depois deste objecto ser correctamente preenchido. O mesmo não é aplicável aos outros elementos da mesma entrada, já que todos eles são inicializados com valores válidos.

- **alarmSampleType** - valor inicial: 1 (absoluteValue)
  Este valor é utilizável para variáveis do tipo INTEGER e Gauge.
- **alarmValue** - valor inicial: 0 (read only)
- **alarmStartupAlarm** - valor inicial: 3
  Permite uma mais rápida inicialização do mecanismo de histerese na geração de eventos.
- **alarmRisingThreshold** e **alarmFallingThreshold** - valor inicial: 0
  Pela sua depêndencia da variável a monitorizar não é possível estimar um valor para a sua inicialização.
- **alarmRisingEventIndex** e **alarmFallingEventIndex** - valor inicial: 0
  Este valor implica a ausência de eventos associados.
- **alarmOwner** - valor inicial: string nula
  Deve ser preenchido pelo gestor.

E. Partilha de recursos disponíveis

Com a utilização de um objecto *owner*, existente em todos os grupos da RMON MIB, poder-se-ão evitar conflitos na partilha de recursos. O preenchimento deste objecto permite a outras estações gestoras reconhecer facilmente os recursos que estão na sua posse, bem como obter a informação necessária para contactar outras estações gestoras que possuam entradas activas, podendo assim negociar com elas a libertação de recursos que estejam nas sua posse e que sejam necessários para outras actividades.

VII. RESULTADOS

A implementação destes dois grupos permite-nos obter informação de variáveis implementadas no agente em questão. Desta forma, podemos verificar se estas ultrapassam valores não desejados, possibilitando-nos assim, estar a par da situação actual da rede.

O grupo *alarm* está implementado para suportar a monitorização de 50 variáveis ao mesmo tempo.

Nesta implementação o número de entradas na tabela de *logs* relativas ao mesmo evento é controlado de modo a que apenas as últimas 50 entradas sejam visualizadas, evitando assim uma sobrecarga da memória do sistema.

Contudo, na utilização máxima do sistema, ou seja com 50 entradas na tabela de alarmes activas, em que cada uma delas poderá ter a si associada 2 entradas na tabela de eventos e 100 entradas na tabela de logs activas, poderemos ter 100 entradas de eventos e 5000 entradas de logs activas, respeitantes ao grupo *alarm,* o que é perfeitamente possível passadas algumas horas de funcionamento do sistema. Nesta situação o sistema ficaria muito sobrecarregado, surgindo atrasos tanto na resposta a pedidos do gestor como na execução das funções de monitorização, pondo em questão a fiabilidade do sistema.

VIII. CONCLUSÕES

Estudamos e apresentamos algumas conclusões sobre a viabilidade de construção de aplicações de gestão, particularmente de monitorização de redes, com base em interfaces de baixo nível. Tanto a interface fornecida pelo *Packet Filter* como a fornecida pela DLI mostraram algumas limitações, em particular esta última pela impossibilidade de captar pacotes em modo promíscuo.

Apresentamos igualmente um estudo e uma implementação da *Remote Monitoring MIB*, uma interface standard, para monitores de redes, normalizada pela comunidade Internet para um modelo global de gestão baseado no protocolo SNMP.

AGRADECIMENTOS

Agradecemos ao Eng. Fernando Pipa pelo apoio na realização deste trabalho.

REFERÊNCIAS

[1] Digital Equipment Corporation, "The Packet Filter: An Efficient Mechanism for User-level Network Code", *Ultrix V4.1 Manual.*

[2] Sun Microsystems Inc. *NIT(4P); SunOS 4.1.1 Reference Manual.* Mountain View, CA, October 1990.

[3] S. McCanne and Van Jacobson, "The BSD Packet Filter: A New Architecture for User-level Packet Capture", *Winter USENIX Conference*, San Diego, CA, January 1993.

[4] Digital Equipment Corporation, "Guide to the Data Link Interface",*Ultrix V4.1 Manual.*

[5] J. Case, M. Fedor, M. Schffstall, J. Davin, "A Simple Network Management Protocol (SNMP)", *RFC 1157*, FAB, May 1990.

[6] K. McCloghrie and M. Rose (eds), "Management Information Base for Network Management of TCP/IP-based Internets: MIB-II", *RFC 1213*, Hughes LAN System, Performance Systems Internation, March 1991.

[7] M. Rose and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", *RFC 1155*, Performance Systems Internation, Hughes LAN System, March 1991

[8] A. P. Carvalho and N. P. da Rocha, "Plataforma SNMP para o Desenvolvimento de Objectos de Gestão", *Revista do DETUA, 1993.*

[9] S. Waldbusser, "Remote Network Monitoring Management Information Base", *RFC 1271*, Carnegie Mellon University, November 1991.

[10] S. Waldbusser, "Token Ring Extentions to the Remote Network Monitoring MIB", *RFC 1513*, Carnegie Mellon University, Setember 1993.

[11] M. Rose and K. McCloghrie (eds), "Concise MIB definitions", *RFC 1212*, Performance Systems Internation, Hughes LAN System, March 1991.

Bridge para Interligação de uma Rede *Ethernet* e de uma Rede Sem Fios

Alberto Teixeira Queirós, Manuel Gomes de Oliveira
Rui Tomaz Valadas, A. Manuel de Oliveira Duarte

Resumo- **Este artigo descreve o projecto de fim-de-curso designado por "*Bridge* para interligação de uma Rede *Ethernet* e de uma Rede sem Fios".**
O trabalho decorreu em duas fases. Na primeira fase desenvolveu-se o *software* da *bridge* tendo como suporte um PC. Este software foi testado e optimizado utilizando uma package comercial para análise de tráfego em redes. Na segunda fase projectou-se, desenvolveu-se e testou-se o hardware da *bridge*.

Abstract- **This paper describes the work carried out as part of the final year project designated by "Bridge for the interconnection of Ethernet and Wireless Local Area Networks".**
The work was developed in two distinct phases. In the first phase the software of the Bridge was developed assuming a PC based platform. This software was tested and optimised using a commercially available software package for LAN traffic analysis. In the second phase the hardware of the Bridge was designed, developed and tested.

I. INTRODUÇÃO*

O trabalho de investigação no qual se inseriu este projecto tem como objectivo o desenvolvimento de uma *bridge* para interligar uma rede *Ethernet* a uma rede sem fios a 1Mbps compatível com o *standard* IEEE802.3 [2], como representado na Fig.1.

No sentido de facilitar as fases de desenvolvimento e teste optou-se por desenvolver inicialmente uma *bridge* entre duas redes por cabo *Ethernet*. O trabalho efectuado neste ano lectivo dividiu-se em duas partes. Na primeira implementou-se um sistema tendo como suporte um PC IBM compatível e duas placas de rede, de modo a interligar duas redes *Ethernet* a 10 Mbps. Este sistema permitiu projectar, desenvolver e testar o algoritmo de funcionamento da *bridge* e o respectivo *software*. Na segunda parte do projecto desenvolveu-se e testou-se um sistema *stand-alone* em *hardware* que permitirá a implementação de todas as funções da *bridge* desenvolvida na primeira parte, para além de outras, nomeadamente de gestão de redes.

O trabalho aqui apresentado decorreu no âmbito do projecto Europeu ESPRIT.6892 POWER ("*Portable Workstation for Education in Europe*").

Na secção II descreve-se a *bridge* desenvolvida bem como os seus princípios de funcionamento. Na secção III apresenta-se a *bridge* desenvolvida por *software*. Na secção IV apresenta-se o sistema *stand-alone* desenvolvido em *hardware*. Finalmente na secção V apresentam-se as conclusões do projecto.

* Trabalho realizado no âmbito da disciplina de projecto

II. DESCRIÇÃO GERAL DO FUNCIONAMENTO DA *BRIDGE*

Uma *bridge* é um sistema que permite interligar LANs (*Local Area Networks*). Opera no nível 2 do modelo OSI (camada lógica), ao contrário dos *routers* que operam no nível 3 (camada lógica) e dos repetidores que operam ao nível 1 (camada física). A *bridge* desenvolvida interliga duas LANs do tipo IEEE 802.3 (redes *Ethernet*).

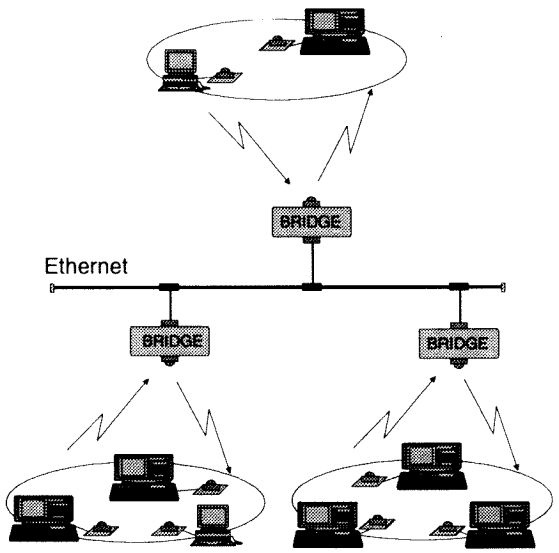


Fig. 1: Diagrama geral da rede.

A. A rede Ethernet (IEEE802.3)

O *standard* IEEE 802.3 define uma família de sistemas baseados no protocolo de acesso ao meio CSMA/CD para velocidades de transmissão entre 1Mbps e 10Mbps. O modo de operação do protocolo de acesso ao meio, CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) pode descrever-se da seguinte forma: quando uma estação deseja transmitir, ela "observa" primeiro o canal de transmissão para verificar se há actividade neste. Se o canal estiver livre a estação transmite de imediato. Caso contrário espera até este ficar desocupado e tenta

imediatamente transmitir. No caso de duas ou mais estações começarem a transmitir simultaneamente ocorrerá uma colisão. Nesta situação as estações deverão interromper as suas transmissões e esperar um intervalo de tempo aleatório antes de voltar a repetir todo o processo.

Na implementação em banda base é usado o código de linha *Manchester* o qual facilita a extracção do sinal de relógio por parte do receptor. A interface entre o cabo e a estação é constituída típicamente por um *transceiver* e um controlador de rede. O *transceiver* faz a interface entre o controlador e o cabo. É ainda responsável pela detecção da portadora (*carrier sense*) e pela detecção de colisões (*collision detection*). Por seu lado, o controlador é responsável pela implementação do protocolo de acesso ao meio e pela interface com o CPU (*Central Processor Unit*).

É possível estender um troço de cabo acima do valor especificado pela topologia usada, fazendo uso de repetidores. No entanto uma trama enviada por uma dada estação nunca poderá passar por mais de quatro repetidores.

B. A trama Ethernet

Designa-se por trama, ao conjunto de bits enviados por uma dada estação. Ao nível da subcamada MAC (*Medium Access Control*), a trama é vista como um conjunto de campos. O protocolo IEEE 802.3 especifica o formato da trama como mostra a Fig.2.

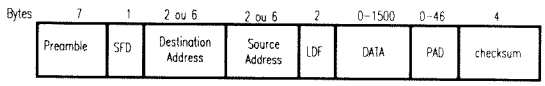


Fig. 2: Formato da trama *Ethernet*.

Seguidamente descreve-se o significado de cada um dos campos:

➪*Preamble*: sequência de uns e zeros consecutivos que permitem ao relógio do receptor sincronizar com o do emissor.

➪SFD (*Start of Frame Delemiter*): indica o início da trama.

➪Endereços: identificam o destinatário (*Destination Address*) e remetente (*Source Address*) da trama. Esses campos têm um comprimento de 2 ou 6 *bytes*.

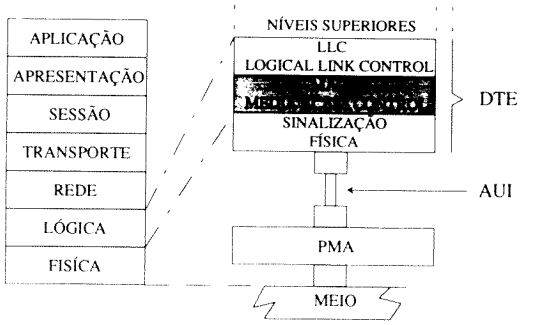
➪LDF (*Lengh of Data Field*): indica o número de *bytes* que ocupa o campo de dados (DATA).

➪DATA: campo de dados, o seu comprimento pode variar entre 0 e 1500 *bytes*.

➪PAD: campo de comprimento variável que só é inserido nas tramas mais pequenas de maneira a terem um comprimento mínimo de 64 *bytes* (comprimento mínimo da trama *Ethernet*). Assim, qualquer trama com comprimento inferior a 64 *bytes* terá um campo PAD.

➪*Checksum*: campo que permite a detecção de erros.

A Fig. 3 relaciona o protocolo IEEE 802.3 com o modelo de referência OSI. Por definição uma *bridge* implementa apenas as camadas física e lógica.



AUI - ATTACHMENT UNIT INTERFACE
PMA - PHYSICAL MEDIUM ATTACHMENT
DTE - DATA TERMINAL EQUIPMENT

Fig. 3: Relação entre o protocolo CSMA/CD (Std IEEE 802.3) com o modelo de referência OSI.

C. A rede sem fios

A rede sem fios proposta [5] difere de uma rede *Ethernet* apenas no nível físico, utilizando o mesmo protocolo de acesso ao meio (CSMA/CD) desta. Esse meio consiste no espaço livre de uma sala, não num cabo, a transmissão no meio passa a ser sinais ópticos e não de sinais eléctricos. Esta solução permite facilitar o *interworking* com o backbone Ethernet uma vez que *backbone* e células operam com o mesmo protocolo de acesso ao meio [5].

D. A bridge

A *bridge* é constituída por dois portos sendo cada um deles sendo constituído pelos elementos representados na Fig. 4.

Um dos portos está ligado à rede *Ethernet* - *Backbone* (Port_B) e o outro à rede sem fios - Célula (Port_C).

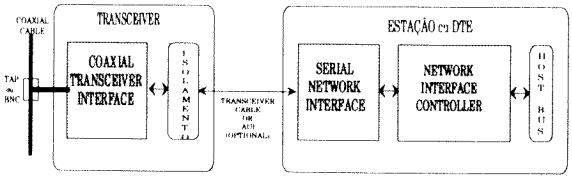


Fig. 4: Diagrama de blocos de um porto da *bridge*.

A rede *Backbone* pode ser encarada como a espinha dorsal do sistema de comunicação. As redes Célula podem ser encaradas como ramificações dessa espinha dorsal. Assim, a *bridge* pode ser vista como um dispositivo que estende a rede principal.

Uma característica importante da rede Célula em relação à rede *Backbone* é que estas somente diferem relativamente

à velocidade de transmissão. Assim o *software* a desenvolver para os *drivers* da *bridge* associados a cada um dos portos é praticamente o mesmo.

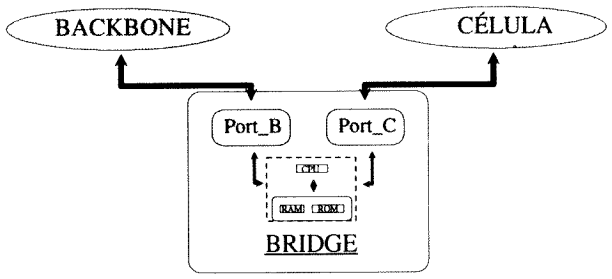


Fig. 5: Esquema básico da *bridge*

Em termos físicos, além dos dois Portos, a *bridge* contém ainda o *hardware* necessário para armazenar e processar a informação (processador, memória RAM, memória ROM, etc.).

A Fig. 5 mostra o esquema básico da *bridge*.

E. Funcionamento da bridge

Como já se referiu acima, a *bridge* é um dispositivo que interliga duas LANs e difere dum simples repetidor nos seguintes aspectos:

⇨ Filtragem:

A *bridge* processa toda a informação que viaja nas redes que interliga de modo a seleccionar as tramas que têm seguimento para a rede oposta àquela em que foram transmitidas e as que não têm seguimento. Esta operação de filtragem é executada de acordo com os dados armazenados em duas tabelas de endereços (uma associada a cada porto) e é uma das principais funções da *bridge*. Esta particularidade permite evitar uma sobrecarga de tráfego nas redes que são interligadas pela *bridge*. A *bridge* inicializa e actualiza as tabelas de endereços automaticamente e também adaptativamente através de um processo de um processo designado por *bridge learning*.

⇨ *Buffering*:

No eventual seguimento das tramas de uma rede para a outra, a *bridge* não poderá retransmiti-las ao mesmo ritmo com que chegam, devido à diferença de velocidades de transmissão, ao tempo de processamento e ao eventual "congestionamento" das redes. Assim ela terá que as armazenar num buffer para posterior retransmissão.

A *bridge* projectada é uma *bridge* transparente - *bridge* inteligente que não necessita de inicialização externa e "aprende" ao longo do tempo de funcionamento a localização das máquinas ligadas a cada uma das redes que interliga (*bridge learning*).

A *bridge* funciona com os seus controladores em modo promíscuo. Neste modo a *bridge* analisa todas as tramas que viajam nas redes a ela ligadas.

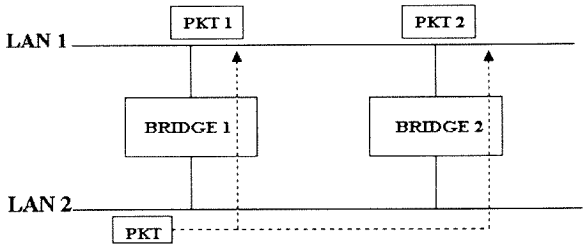


Fig. 6: Exemplo de uma duplicação de tramas.

No caso da topologia considerada, o acesso a uma rede Célula será feito por uma única *bridge*. Esse aspecto permite que não se gerem eventuais duplicações de tramas que circulam na rede, além de eliminar possíveis situações de *dead lock*- existência de loops fechados que podem provocar a circulação permanente de um pacote entre a rede Célula e a rede *Backbone* [1]. Esta situação acontece quando uma rede pode ser acedida por várias *bridges* que não têm conhecimento do destinatário desse pacote. A Fig. 6 exemplifica esta situação. Sendo assim não é necessário neste caso implementar os algoritmos de *spanning tree* que visam, precisamente, evitar as situações de *dead lock* [1,2].

III. DESENVOLVIMENTO DO *SOFTWARE*

A. Objectivos

Nesta parte do trabalho desenvolveu-se uma *bridge* suportada por um PC IBM compatível para interligar duas redes *Ethernet* funcionando a 10 Mbps.

Este *software* servirá de ponto de partida para o desenvolvimento de uma *bridge* que interligará uma rede *Ethernet* a 10 Mbps tendo um cabo coaxial como meio, com uma rede sem fios e que terá uma velocidade de transmissão de, aproximadamente, 1Mbps.

B. Constituição

As placas de rede que fazem a interface entre o PC, que suporta a *bridge*, e as redes são placas WD8003 da SMC (*Standard Microsystems Corporation*) [3]. Os drivers para essas placas foram criados tendo como base funções cedidas pela SMC o que permitiu uma fácil optimização do *software* para os drivers.

O software desenvolvido é constituído por três grandes partes:

⇨ os drivers para as placas WD8003;

⇨ as funções que implementam o processo de bridging;

⇨ as funções que suportam o interface com o utilizador.

As funções que constituem estas três partes foram desenvolvidas em Microsoft C600, com excepção das

rotinas enviadas pela SMC que tinham sido desenvolvidas em Microsoft C500, segundo informações da própria SMC.

C. Algoritmo da bridge

O funcionamento da *bridge* pode-se resumir nos seguintes passos:

1 - Inicialização das tabelas de endereços para os dois portos;
2 - Recepção das tramas de um porto;
4 - Processamento das tramas;
4.1 Verificar se uma determinada trama tem que seguir para o porto oposto, se sim então armazená-la em memória.
4.2- Actualização das tabelas de endereços.
5 - Transmissão das tramas que tenham como destino uma máquina do porto oposto.
6 - Repetir os passos 2, 3, 4 e 5 para o outro porto.

Os passos 2, 3, 4, 5 e 6 são repetidos até que uma tecla seja primida.

Interrompendo o processo de *bridging* pode-se alterar a configuração das placas de rede, introduzir ou remover os endereços permanentes nas tabelas de endereços e consultar as estatísticas referentes ao último bridging.

D. Os testes da bridge

Depois de desenvolvido o software era necessário testá-lo para verificar o seu funcionamento e tentar optimizar o desempenho da *bridge*, principalmente no que se referia à sua velocidade de processamento. Era ainda necessário fazer as medidas relacionadas com o desempenho.

Para os testes da *bridge* usou-se o analisador de tráfego *NetSight Analyst v.1.1* da *Intel* [4]. Este analisador tem a facilidade de gerar tráfego, permitindo controlar o número de pacotes a enviar por segundo e o seu tamanho.

A base de trabalho usada para testar a *bridge* e medir as *performances* é a representada na Fig. 7. A referida base de trabalho é constituída por:

- Um PC + 2 placas de rede (WD8003 da SMC). Este PC suporta o *software* desenvolvido.
- Um PC + 1 placa de rede para simular a rede Célula.
- Um PC + 1 placa de rede para simular a rede Backbone.

Nestes testes instalou-se no PC que simulava uma das redes o NetSight Analyst a gerar tráfego, e no PC que simulava a outra rede o NetSight Analyst a capturar esses pacotes (que tinham como endereço de destino o endereço físico da placa instalada nesse PC). Pode-se desta forma verificar a velocidade a partir da qual a *bridge* não conseguia passar todos os pacotes transmitidos para o porto oposto e a percentagem de pacotes perdidos.

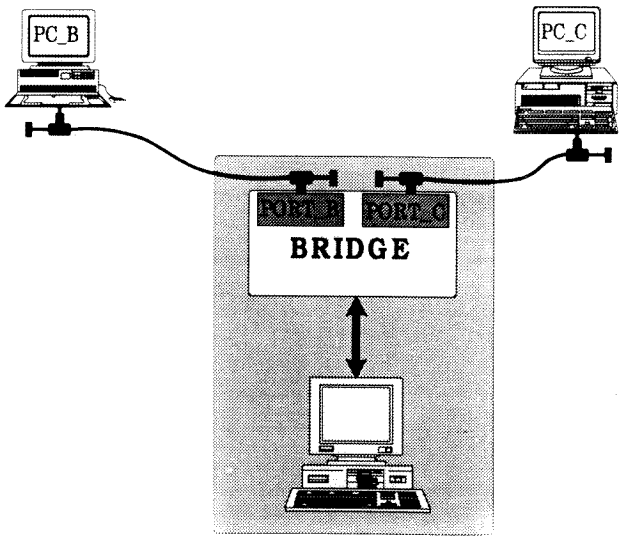


Fig. 7: Base de trabalho usada para testar a *bridge* e medir as suas *performances*.

Fizeram-se testes para pacotes *Ethernet* com comprimento de 60, 1520 e 650 *bytes*. Usaram-se estes comprimentos porque na rede *Ethernet*, 60 *bytes* é o tamanho mímino de uma pacote, 1520 *bytes* é o tamanho máximo e 650 *bytes* é um valor intermédio.

Como exemplo, pode-se dizer que nas primeiras versões da *bridge* uma transferência de 190 ficheiros num total de 2.312.894 *bytes* de informação, usando o ftp, entre uma máquina *Unix* e um PC situado na Célula, demorou cerca de 410 segundos, em horas de pouca carga na rede, o que dá uma média de 45462 bps (5682 *bytes* por segundo). Na última versão uma transferência de 218 ficheiros num total de 2416789 *bytes* de informação demorou cerca de 123 segundos (157189 bps - 19648 *bytes* por segundo), em condições idênticas. Esta última transferência realizada logo de seguida mas sem a *bridge* a interligar a fonte e o destino (estavam na mesma rede) demorou cerca de 68 segundos (284328 bps - 35541 *bytes* por segundo), cerca de metade do tempo gasto com a *bridge* a interligar os dois pontos extremos (fonte e destino da informação). Pode-se ver que a *performance* mais do que triplicou o que mostra uma evolução significativa no refinamento do software. Este teste foi realizado com 50 entradas na tabela de endereços associada à rede Backbone e 20 entradas na tabela de endereços associada à rede Célula.

Verificou-se que o número de entradas nas tabelas de endereços têm uma grande influência no desempenho da *bridge*. Esse número não deve ser muito pequeno para garantir que só atravessam a *bridge* as tramas que realmente tenham como destino uma máquina do porto oposto, ou as tramas cujo endereço destino seja do tipo *broadcast*. Também não deve ser exageradamente grande para não se perder muito tempo com o processamento (*bridging*), o que implica velocidades baixas para a *bridge*. E, o número de entradas na tabela de endereços associadas à rede *Backbone* deve ser bastante maior que o

número de entradas na tabela associada à rede Célula, visto as dimensões físicas serem bastante diferentes.

Estes testes permitiram no entanto verificar que existem algumas limitações de performance devidas ao facto da *bridge* ser baseada num PC, como aliás era esperado.

III. DESENVOLVIMENTO DO *HARDWARE*

Nesta segunda parte do trabalho desenvolveu-se um sistema *stand-alone* em *hardware* que permitirá implementar todas as funções da *bridge* desenvolvida na primeira parte, para além de outras, nomeadamente de gestão [2].

Esta segunda parte decorreu em duas fases:

- ⇨ projecto do hardware da *bridge*.
- ⇨ montagem e teste da placa por etapas, de maneira a mostrar o bom funcionamento do sistema projectado.

A *bridge* que se pretende implementar é basicamente constituida por:

- dois portos, cada um ligado a uma rede. Esses portos fazem a interface entre a rede a qual estão ligados e o *bus* do sistema, sendo o seu principal componente um controlador de rede 82586 da Intel.

- Um microprocessador que faz a gestão dos recursos do sistema e processa a informação envolvida durante o funcionamento da *bridge*. A utilização dum CPU da família do 80186 simplificou muito o desenho da placa, permitindo um elevado desempenho do sistema numa configuração com um mínimo de hardware.

- Memória ROM onde é armazenado o programa que implementa as várias funções da *bridge* e ainda, informação para a inicialização dos controladores.

- Memória RAM para armazenar a restante informação (tabelas de endereços, tramas, comandos e registos).

- *Latches* de endereços que fazem a captura dos endereços lançados pelo microprocessador, permitindo aos controladores e ao CPU o acesso as memórias.

- *Data trasceivers* que servem de buffer e de driver ao varios componentes ligados ao *bus* do sistema.

- Lógica para descodificação dos endereços lançados pelo microprocessador que permite activar os sinais de *chip select* das memórias e eliminar o conflito na inicialização dos controladores que acedem, por defeito, a uma posição predeterminada da memória ROM.

- Lógica suplementar que garanta a exclusividade de posse do *bus* por parte dos controladores quando estes lhe querem aceder.

- Gerador de *wait state* que permite programar o número de *wait state* no acesso dos controladores às memórias.

A Fig. 8 mostra o diagrama de blocos do sistema que se implementou.

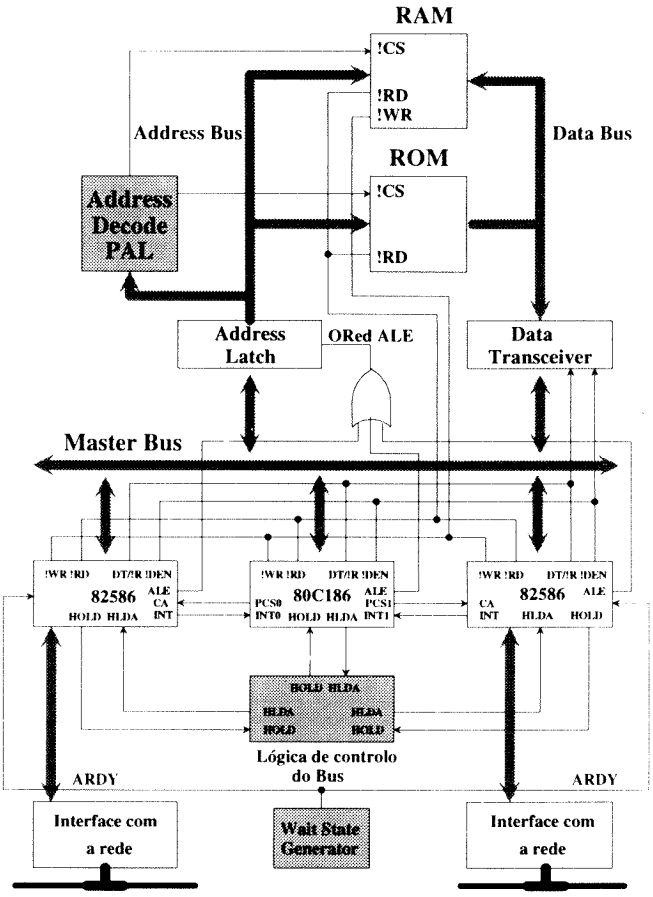


Fig. 8: Diagrama de blocos da *bridge*

A montagem e os testes da placa efectuaram-se em três etapas, tendo como objectivos a verificação do funcionamento das seguintes interfaces:

- entre o CPU e as memórias, verificando as operações de leitura nas memórias ROM e RAM, escrita na RAM, inicialização do processador e activação dos sinais de *chip select* programáveis;

- entre o CPU e um controlador, conferindo os vários passos na inicialização do controlador e o correcto acesso as memórias;

- entre o CPU e os dois controladores, confirmando os endereços e os dados de inicialização do sistema e a resolução do conflito no acesso ao *bus* por parte dos controladores.

Todos os testes foram efectuados com sucesso.

CONCLUSÕES

No que se refere à primeira parte do trabalho (secção II), o objectivo principal era desenvolver uma *bridge* transparente que servisse de base ao desenvolvimento de uma *bridge* que assentará sobre o *hardware* apresentado na segunda parte (secção III). Esse objectivo foi atingido, pois o algoritmo é eficaz no que se refere ao processo de *bridging*. Como comprovação desse facto temos que as velocidades da *bridge* são bastante boas tendo em

consideração que é uma *bridge* suportada por um PC. Outras melhorias em termos de velocidade de processamento poderiam ainda ser feitas se algumas das rotinas responsáveis pelo *bridging* fossem desenvolvidas em *assembly*.

O trabalho desenvolvido na segunda parte consistiu no projecto, na montagem e no teste da placa sôbre a qual se pretende desenvolver a *bridge*. Os resultados obtidos são muito animadores, tendo-se conseguido uma montagem simples (com um mínimo de *hardware*) e funcional, nomeadamente no que respeita a resolução do conflito no acesso ao *bus* (pela lógica de controlo do *bus*) e na inicialização dos controladores de rede. A maneira como foi concluído o trabalho permitirá o prosseguimento do projecto sem necessidade do conhecimento específico das várias interfaces, isto é a plataforma *hardware* básica para a *bridge* ficou concluída. Torna-se agora necessário aumentar a velocidade de relógio para a frequência prevista, montar as interfaces séries de rede e montar a interface de comunicação série com o computador. Note que esquemas detalhados destas interfaces já existem, não devendo portanto criar nenhum problema grave.

AGRADECIMENTOS

Agradecemos à SMC as rotinas enviadas. Aos docentes do GSBL pelo apoio dispensado.

REFERÊNCIAS

[1] Tanenbaum, Andrew S, *Computer Network*, Prentice Hall Software.
[2] *Std IEEE 802.1D 1990*, IEEE.
[3] *WD8003E Hardware Installation Guide*, Western Digital.
[4] *Intel NetSight Analyst^{TM} Reference Guide*, Intel.
[5] *Rui T. Valadas, Adriano C. Moreira, A.M. de Oliveira Duarte "Hybrid (Coaxial/Infrared) Ethernet Local Area Networks" Proceedings IEEE Conference on Wireless LAN Implementation, Dayton, Ohio, USA, 17-18 September 1992.*

Terminal Multi-Utilizador

Alexandre Sarmento, Carlos Carvalhal, Nelson Pacheco da Rocha

Resumo- **Nesta comunicação apresentaremos a arquitectura de um sistema distribuído, que designámos por *Terminal Multi-Utilizador* e que se pretende adequado à preparação de documentos multimédia por um grupo de pessoas cooperando entre si.**

O protótipo desenvolvido é suportado em estações de trabalho correndo os sistemas *UNIX* (sobre o ambiente de desenvolvimento *X11* e *OSF/Motif*) e *MS-DOS* (sobre o ambiente de desenvolvimento *Microsoft Windows*) ligadas por uma rede *Ethernet* e permite que vários utilizadores trabalhem em conjunto na edição de documentos que poderão conter texto, gráficos e imagens.

No protótipo desenvolvido foi dado um ênfase às facilidades disponibilizadas aos diferentes intervenientes na sessão de cooperação, o que inclui interfaces adequadas à cooperação e edição de documentos, identificação de *Co-autores* e possibilidades de realizar comentários públicos e privados.

Abstract- **In this paper we will present the architecture of a distributed system, the *Terminal Multi-Utilizador*. The function of this system is to prepare multimedia documents by a group of persons co-operating between each other.**

The developed prototype is supported in workstations running *UNIX* (under the *X11, OSF/Motif* development environment) and *MS-DOS* (under the *Microsoft Windows* development environment) connected by an *Ethernet* network and allows various users to work together to edit documents that may contain text, graphics and images.

In this prototype emphasis was given to the available facilities to the different participants in the co-operative session, which includes adequate interfaces to the co-operation and document editing, identification of *Co-authors* and possibilities to make public and private comments.

I. INTRODUÇÃO*

A tecnologia actualmente disponível possibilita o desenvolvimento de sistemas que suportem a realização de tarefas que requeiram a colaboração simultânea de vários utilizadores [1], como, por exemplo, ensino remoto, estudo assistido, resolução de problemas e preparação de documentos. A ideia base do *Terminal Multi-Utilizador* (*TMU*) consiste em apresentar, sobre um sistema distribuído composto por várias estações de trabalho ligadas por uma rede local, um ambiente de trabalho cooperativo.

* Trabalho realizado no âmbito da disciplina de projecto

II. ALGUNS CONCEITOS

Antes de iniciar a descrição do trabalho desenvolvido é conveniente definir alguns termos que vão ser usados ao longo da presente comunicação:

- ⇨ *Co-autor* - qualquer um dos intervenientes na sessão de trabalho cooperativo.
- ⇨ *Coordenador* - utilizador que inicia a sessão de trabalho cooperativo, convidando todos os restantes *Co-autores*. É o utilizador que dirige a sessão de trabalho, possuindo ferramentas inacessíveis aos restantes cooperantes.
- ⇨ *Co-autor Convidado* - um *Co-autor* que não é *Coordenador* e que foi convidado por este a participar na sessão de trabalho cooperativo.
- ⇨ *Sessão de Trabalho Cooperativo* - tempo durante o qual dois ou mais *Co-autores* estão a cooperar na edição de um documento partilhado. Inicia-se logo que houver um dos *Co-autores* a aceitar o convite do *Coordenador* e conclui-se quando todos os *Co-autores Convidados* se tiverem retirado.
- ⇨ *Situação de Gestão* - toda a informação referente à gestão da *Sessão de Trabalho Cooperativo* indicando quem é o *Coordenador*, quem são os *Co-autores Convidados*, quem possui a palavra, quem pediu a palavra e quem pediu autorização para abandonar a sessão de trabalho.
- ⇨ *Documento* - ficheiro multimédia constituído por uma sequência de páginas que poderão conter texto, gráficos e imagens.
- ⇨ *Objecto* - qualquer um dos elementos de um *Documento*. A cada objecto está associada uma posição espacial dentro da página, determinada por três parâmetros. Dois destes parâmetros dizem respeito à posição do objecto no plano da página (abcissa e ordenada) e o outro refere a posição ocupada pelo objecto no plano perpendicular ao plano da página (cota).

III. ARQUITECTURA DO SISTEMA

A arquitectura do *TMU* assume um aspecto centralizado *(Fig.1)*. Compete a um dos sistemas, aquele onde está o utilizador *Coordenador*, a gestão das interacções entre os diferentes participantes, de acordo com um protocolo de gestão de trabalho cooperativo, para o que existe uma aplicação designada por *GestorCooperação*. Compete à aplicação *GestorCooperação* manter o sincronismo entre os diferentes sistemas de forma a todos os *Co-autores* "verem a mesma coisa" (filosofia *WISIWYS - What I See Is*

What You See). Adicionalmente o *GestorCooperação* mostra a *Situação de Gestão e* disponibiliza comandos de gestão que permitem ao *Coordenador* gerir a *Sessão de Trabalho Cooperativo*. Para além da especificação e desenvolvimento do protocolo de gestão de trabalho cooperativo, e do *GestorCooperação*, foram desenvolvidas diversas aplicações:

- ⇨ *Cooperação* - permite a um *Co-autor Convidado* participar na *Sessão de Trabalho Cooperativo*. Tal como a aplicação *GestorCooperação* disponibiliza os comandos de gestão a que o utilizador tem acesso em cada instante e mostra a *Situação de Gestão*.
- ⇨ *RecepçãoConvites* - recebe os convites para participar nas *Sessões de Trabalho Cooperativo*, seguidamente interroga o utilizador sobre o que deseja fazer com o convite. Se o utilizador aceitar o convite a aplicação *RecepçãoConvites* lança as aplicações *Cooperação*, *Editor* e *Mensagem*.
- ⇨ *Editor* - permite visualizar e editar um documento multimédia.
- ⇨ *Mensagem* - disponibiliza canais de comunicação bidireccionais que possibilitam a comunicação escrita entre *Co-autores*.

As aplicações *Cooperação e RecepçãoConvites* só existem do lado dos *Co-autores Convidados*, a aplicação *GestorCooperação* só existe do lado do *Coordenador* e as restantes existem em todos os *Co-autores*.

Quando um determinado utilizador quer iniciar uma sessão de trabalho cooperativo activa a aplicação *GestorCooperação* a qual se encarregará de notificar os diferentes *Co-autores* e activar localmente as aplicações *Editor* e *Mensagem (Fig.2)*. Simultaneamente, nos sistemas onde se encontram os potenciais *Co-autores Convidados* para essa específica sessão de trabalho cooperativo, a aplicação *RecepçãoConvites* estabelece um diálogo com o *GestorCooperação*, ao fim do qual activa as aplicações *Cooperação*, *Editor* e *Mensagem*. Nesta altura, o ambiente de trabalho está completo e os diferentes utilizadores podem começar a cooperar na preparação de um documento.

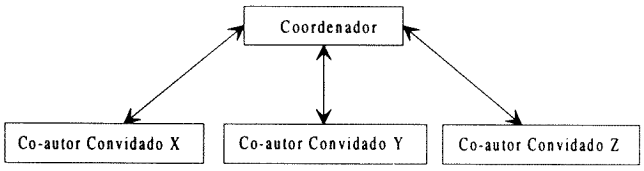


Fig.1 - Controlo centralizado do TMU.

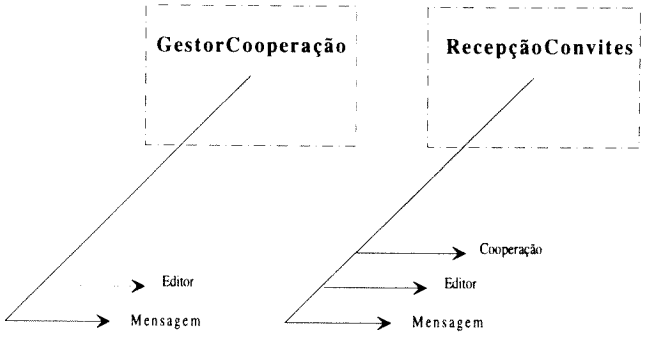


Fig.2 - Diálogo entre a aplicação *GestorCooperação* e a aplicação *RecepçãoConvites*, bem como activação das diferentes aplicações necessárias à *Sessão de Trabalho Cooperativo*.

Do que foi anteriormente referido, é de notar que a aplicação *GestorCooperação* está associada ao *Coordenador* da sessão cooperativa enquanto que a cada *Co-autor Convidado* está associado um processo *Cooperação*. As aplicações *GestorCooperação* e *Cooperação* são basicamente semelhantes, apenas diferindo no facto de o *GestorCooperação* permitir ao *Coordenador* o controlo da sessão cooperativa enquanto que a aplicação *Cooperação* apenas permite a participação de cada *Co-autor* na sessão cooperativa.

Ao *GestorCooperação* compete transmitir para todas as aplicações *Cooperação* associadas aos *Co-autores Convidados* as alterações efectuadas sobre o documento partilhado, os comandos de gestão originados pelo *Coordenador*, comandos de edição e de gestão enviados por cada processo *Cooperação*.

Por outro lado, também compete ao *GestorCooperação* receber os comandos de edição e gestão gerados pela aplicação *Cooperação* do *Co-autor Convidado*, que num dado momento, possui a palavra, e retransmiti-los para as aplicações *Cooperação* dos restantes *Co-autores Convidados*.

As aplicações *GestorCooperação* e *Cooperação* devem "sensibilizar" ou "insensibilizar" o *Editor* sempre que o *Co-autor* "ganha" ou "perde" a palavra, de forma a garantir que só o *Co-autor* que possui a palavra pode editar o documento partilhado.

A comunicação entre os processos GestorCooperação e Cooperação e entre os processos *GestorCooperação* e *RecepçãoConvites* é feita usando uma *socket* (protocolo *TCP/IP* numa *LAN Ethernet*) a funcionar em modo datagrama (protocolo *UDP - User Datagram Protocol*). Este facto permite que, através de uma única *socket*, seja possível enviar pacotes para muitos destinos e receber pacotes provenientes de vários destinos. Esta escolha justifica-se também pelo facto dos processos envolvidos serem simétricos, não existindo uma relação *client-server*.

A *Fig.3* representa a arquitectura de comunicação do sistema *TMU*. Nesta figura está representada em detalhe uma ligação do *Coordenador* com um *Co-autor Convidado* e uma ligação do *Coordenador* com um utilizador não convidado para "negociar" um convite. Notar que o processo *RecepçãoConvites* só existe até o convite ser aceite após o qual "desaparece".

IV. PROTOCOLO DE GESTÃO DO TRABALHO COOPERATIVO

Para gerir a sessão do trabalho cooperativo foi implementado um protocolo de gestão no qual toda a responsabilidade recai sobre o *Coordenador*, pois é ele quem convida os utilizadores a participar na sessão de trabalho, quem atribui a palavra, quem atribui um número de ordem aos pedidos de palavra, quem expulsa os *Co-autores* ou os autoriza a abandonar a sessão de trabalho e só ele pode dar a sessão por terminada.

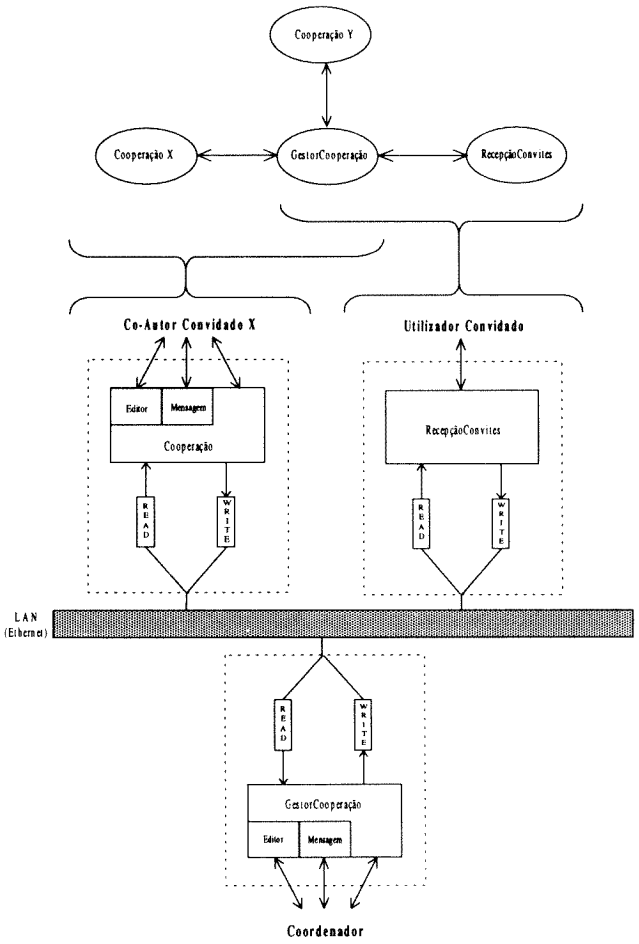


Fig.3 - A arquitectura de comunicação do sistema *TMU*.

Um *Coordenador* pode convidar até 15 utilizadores para participar na sessão de trabalho. Um utilizador após receber um convite, pode aceitá-lo ou rejeitá-lo. No caso de aceitação, a aplicação *GestorCooperação* e as aplicações *Cooperação* passam a exibir a fotografia e o nome do novo *Co-autor Convidado* e este recebe toda a informação sobre o documento partilhado e sobre a *Situação de Gestão*. As aplicações *Mensagem* de todos os *Co-autores* são actualizadas, passando a exibir mais um canal de comunicação bidireccional com o novo *Co-autor*.

Em qualquer momento o *Coordenador* pode dispensar um *Co-autor Convidado*, sendo as aplicações *Cooperação* de todos os *Co-autores Convidados* e a aplicação *GestorCooperação* actualizadas.

Um *Co-autor Convidado* quando deseja editar o documento partilhado tem de pedir primeiro a palavra. Os pedidos de palavra podem ser numerados pelo *Coordenador*, podendo, no entanto, este seguir ou não essa numeração. Um pedido de palavra pode ser satisfeito sem, no entanto, ter-lhe sido atribuído um número de ordem. Assim, o número de ordem dos pedidos de palavra é uma referência para que o *Coordenador*, que numa situação com muitos *Co-autores*, evita que alguém fique muito tempo à espera de autorização para editar o documento partilhado.

Só o *Coordenador* pode atribuir a palavra, e quando um *Co-autor* tem a palavra todos os restantes (incluindo o *Coordenador*) têm o *Editor* inactivo. Quando um *Co-autor* acaba a sua intervenção passa a palavra, a qual "passa para a mesa". No entanto, o *Coordenador* pode a qualquer momento retirar a palavra a um *Co-autor* atribuindo-a a um outro. O *Coordenador* para poder editar o documento partilhado tem de atribuir a palavra a si próprio. Quando um *Co-autor Convidado* deseja abandonar a sessão de trabalho pode assinalá-lo, tendo um comando para o fazer. Seguidamente compete ao *Coordenador* conceder a autorização de abandono.

O *Coordenador* pode ainda dispensar todos os restantes *Co-autores* ao mesmo tempo activando o comando de término de sessão, ao qual só ele tem acesso.

V. APLICAÇÕES

Como já tivemos a oportunidade de referir, o funcionamento do *TMU* baseia-se em cinco aplicações distintas (*GestorCooperação*, *Cooperação*, *RecepçãoConvites*, *Editor* e *Mensagem*) com interfaces gráficas amigáveis que informam o *Co-autor* da *Situação de Gestão* (aplicações *GestorCooperação* e *Cooperação*), do estado de edição do *Documento* (aplicação *Editor*), das mensagens enviadas pelos outros *Co-autores* para o utilizador (aplicação *Mensagem*) ou simplesmente sinalizam que a aplicação de recepção de convites está activa (aplicação *RecepçãoConvites*).

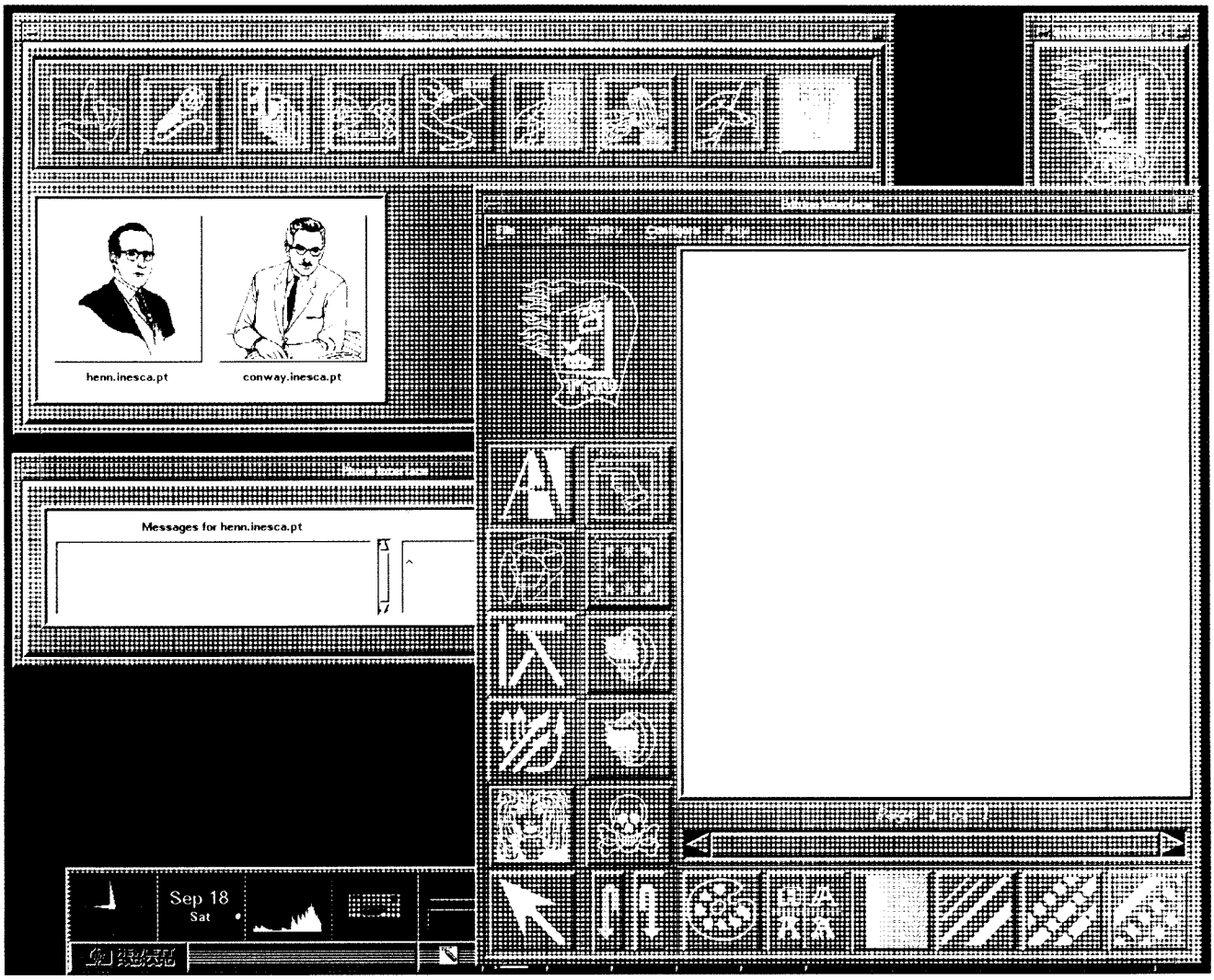


Fig.4 Aspecto geral das *interfaces* do protótipo *TMU* para uma *Sessão de Trabalho Cooperativo* na qual estão presentes dois *Co-autores* (o *Coordenador* é *henn.inesca.pt* e o único *Co-autor Convidado* é *conway.inesca.pt*) sob o ponto de vista da interface do Co-autor *conway.inesca.pt*.

As cinco aplicações atrás referidas também disponibilizam ao utilizador todas as capacidades a que este tem acesso em cada instante através de um conjunto de elementos gráficos (*PushButtons* com etiquetas literais e gráficas, *ScrollBars*, etc.). Estas aplicações serão, nesta secção, referidas mais pormenorizadamente.

A *Fig.4* apresenta uma situação possível durante uma *Sessão de Trabalho Cooperativo* com dois utilizadores. Nela podem distinguir-se as interfaces das aplicações *GestorCooperação, Cooperação, RecepçãoConvites, Editor* e *Mensagem*.

A. Editor

O *Editor* (cuja interface com o utilizador é apresentada na *Fig.5*) permite visualizar e editar um *Documento*. Permite ler um *Documento* do disco, gravar as suas alterações, inserir e apagar páginas, e ainda mudar a página actual de uma forma sequencial ou aleatória. Está provido de ferramentas que permitem criar e editar *Objectos*.

Esta aplicação apresenta um *MenuBar* e quatro zonas de comandos, que permitem criar e editar objectos (primeira coluna de *PushButtons*, da esquerda para a direita, na *Fig.5*), alterar as propriedades dos objectos (segunda coluna de *PushButtons*, da esquerda para a direita, na *Fig.5*), configurar as características dos objectos (linha de *PushButtons* situada na parte inferior da *Fig.5*), mudar de página (*ScrollBar* da *Fig.5*), e uma janela de edição (zona a branco na *Fig.5*) que mostra a página actual do documento e onde se efectuam todos os procedimentos de edição. O tipo de cursor da janela de edição reflecte a capacidade que está activa num dado instante (para o que existem oito cursores diferentes).

Quando o utilizador não possui a palavra todas as facilidades que alteram o documento estão inibidas (inactivas).

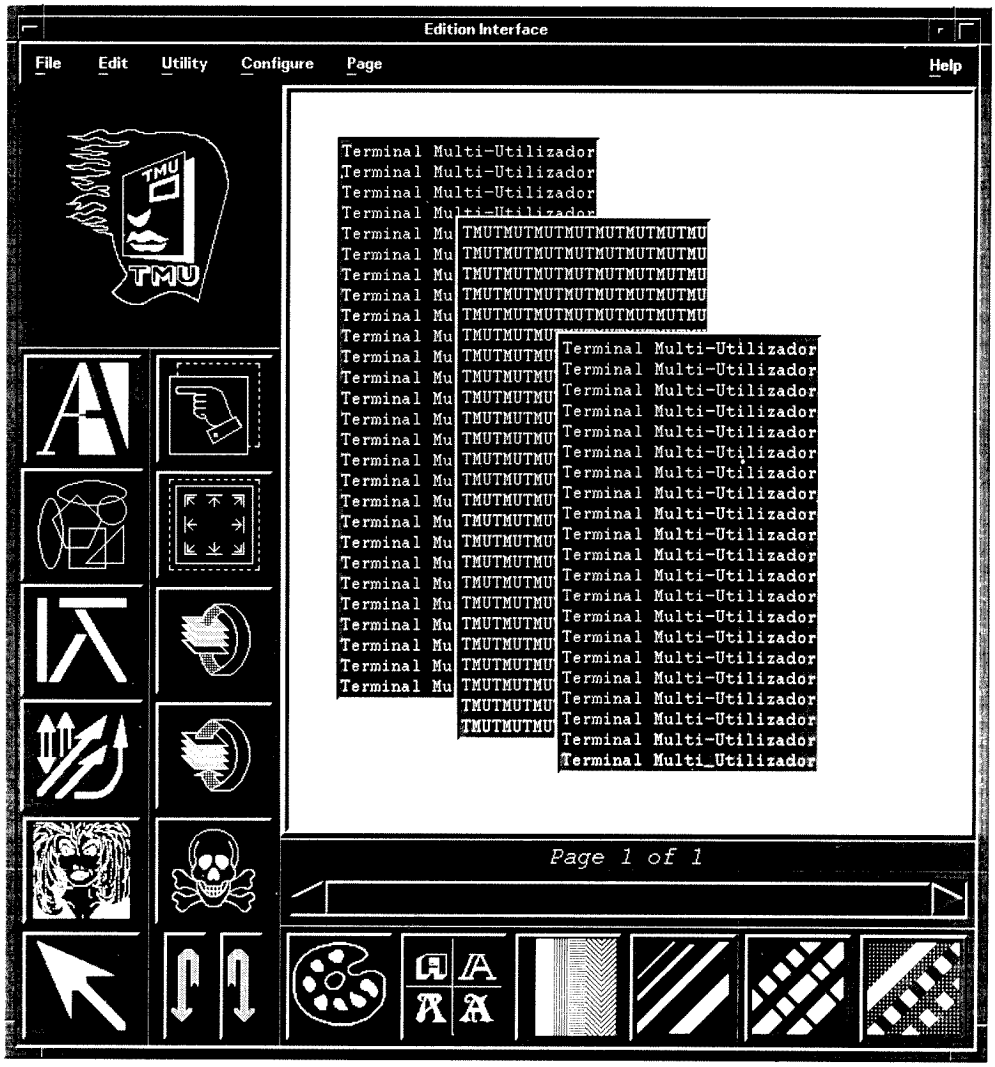


Fig.5 A interface da aplicação *Editor*.

B. GestorCooperação e Cooperação

É através das aplicações *GestorCooperação e Cooperação* (cuja interface com o utilizador é apresentada na *Fig.6*) que o *Coordenador* gere a *Sessão de Trabalho Cooperativo* (*GestorCooperação*), que os *Co-autores* intervêm na *Sessão de Trabalho Cooperativo* (*Cooperação*) e que os *Co-autores* têm conhecimento da *Situação de Gestão* (*GestorCooperação* e *Cooperação*) Qualquer uma destas aplicações apresenta uma zona de comandos e uma zona de visualização. A zona de comandos tem um conjunto de *PushButtons* (na *Fig.6*, da esquerda para a direita temos os *PushButtons* que activam os comandos que permitem: pedir a palavra, atribuir a palavra, atribuir um número de ordem a um pedido de palavra, passar a palavra, pedir autorização para sair, dispensar um *Co-autor Convidado*, terminar a *Sessão de Trabalho Cooperativo*, convidar um utilizador para participar na *Sessão de Trabalho Cooperativo* e o *Help*) que activam os diferentes comandos de gestão. Na zona de visualização estão a fotografia e a identificação de cada um dos *Co-autores* (a fotografia do *Coordenador* encontra-se sempre no canto superior esquerdo da janela de visualização e as fotografias dos restantes *Co-autores* encontram-se ordenadas por colunas numa sequência que corresponde à ordem com que os *Co-autores* "entraram" na *Sessão de Trabalho Cooperativo*).

Algumas das fotografias dos *Co-autores* podem apresentar etiquetas que sinalizam uma das situações:

- ⇨ o *Co-autor* possui a palavra (é o caso da *patt.inesca.pt* na *Fig.6*).
- ⇨ o *Co-autor* pediu a palavra (é o caso da *vlsina.inesca.pt* na *Fig.6*).

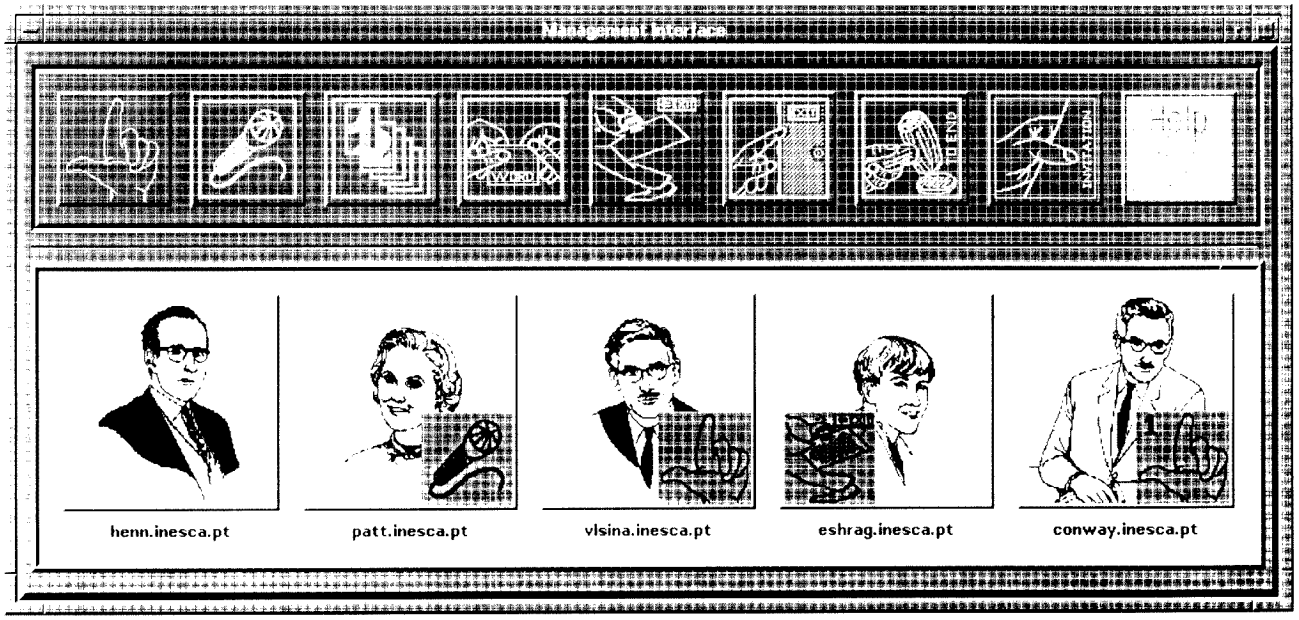


Fig.6 A interface da aplicação *GestorCooperação*.

- ⇨ o *Co-autor* pediu autorização para abandonar a sessão de trabalho (é o caso da *eshrag.inesca.pt* na *Fig.6*).
- ⇨ o *Co-autor* pediu a palavra e o seu pedido de palavra tem um número de ordem (na *Fig.6* o *conway.inesca.pt* pediu a palavra e, em princípio, será o primeiro pedido de palavra a ser satisfeito).

Qualquer uma destas aplicações só disponibiliza os comandos a que o utilizador tem acesso. Assim na interface da aplicação *GestorCooperação* estão inactivos os *PushButtons* referentes aos comandos de pedido de palavra e de pedido de autorização para abandonar a sessão de trabalho, e nas interfaces das aplicações *Cooperação* estão activos os *PushButtons* inactivos no *Coordenador* e o *PushButton* referente ao comando de passagem de palavra.

É importante referir que do lado do *Coordenador* o cursor reflecte a capacidade que está activa quando esta capacidade tem efeito sobre um *Co-autor* que o *Coordenador* tem de escolher (os comandos de atribuição de palavra, de atribuição de número de ordem de palavra e expulsão de um *Co-autor*). Assim, por exemplo, quando o *Coordenador* prime o *PushButton* referente ao comando de atribuição de palavra o seu cursor muda para um microfone, com o formato da etiqueta do *PushButton* atrás referido. Esta capacidade da aplicação *GestorCooperação* permite ao *Coordenador* ter conhecimento de qual a *Capacidade de Gestão* que está activa num dado instante.

C. Mensagem

A aplicação *Mensagem* (cuja interface com o utilizador é apresentada na *Fig.7*) disponibiliza ao utilizador um canal de comunicação bidireccional privado com cada um dos restantes *Co-autores*. A comunicação é escrita. Este serviço permite o diálogo entre dois *Co-autores* acerca da tarefa de edição (discutindo planos ou conceitos sobre o documento, formas de desenhar um diagrama de blocos ou expressar uma ideia numa frase), acerca do conteúdo do documento, ou ainda sobre assuntos fora do contexto da *Sessão de Trabalho Cooperativo* sem, no entanto, prejudicar o trabalho cooperativo.

Cada um dos canais bidireccionais apresenta duas zonas; uma de escrita e outra de leitura, sendo cada uma delas devidamente identificadas com uma etiqueta do tipo "*Messages for <Co_Autor_name>*" para o canal de escrita e "*Messages from <Co_Autor_name>*" para o canal de leitura. Adicionalmente a estas etiquetas, as zonas de escrita e de leitura distinguem-se por apresentarem *foreground* de cores diferentes.

Quer a zona de escrita quer a zona de leitura de qualquer um dos canais de comunicação apresentam uma *ScrollBar* que permite consultar, ou mesmo alterar, aquilo que o utilizador escreveu e ler aleatoriamente as mensagens recebidas.

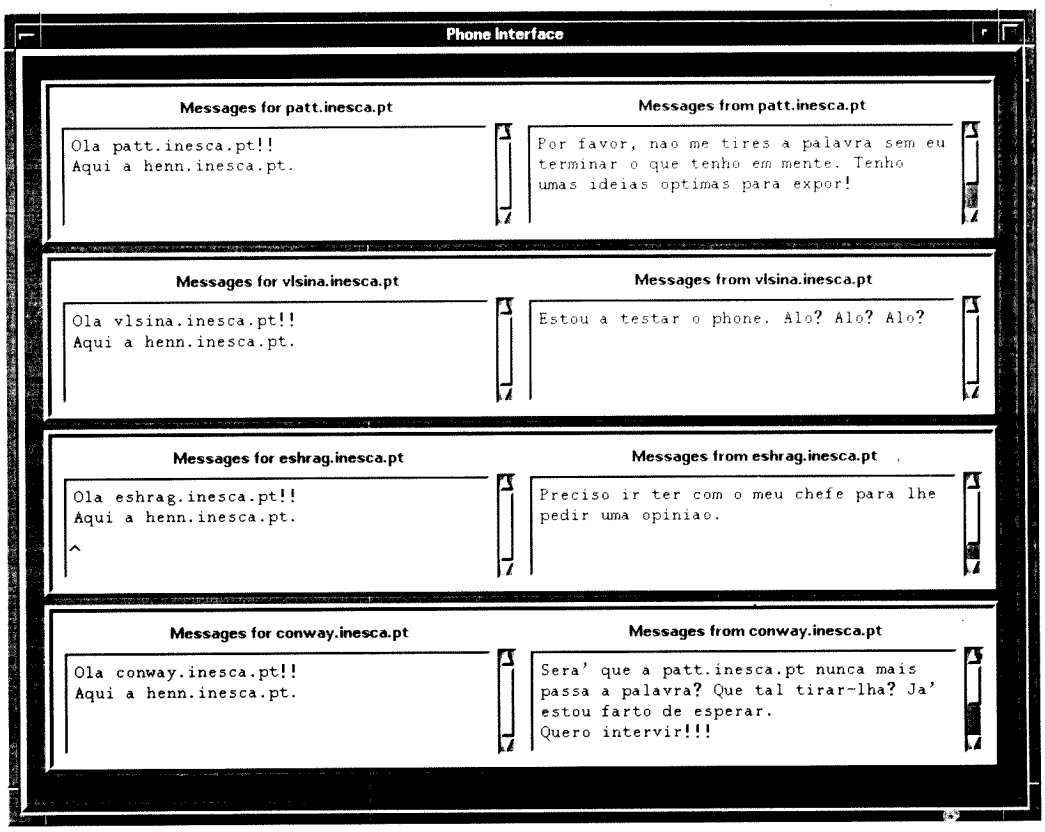


Fig.7 A interface da aplicação *Mensagem* do *Co-autor henn.inesca.pt*, mostrando as mensagens enviadas por este para os restantes *Co-autores* (coluna da esquerda) e as mensagens recebidas pelo utilizador dos restantes *Co-autores* (coluna da direita). Notar que o *Coordenador* desta *Sessão de Trabalho Cooperativo* é o *Co-autor henn.inesca.pt.*

D. *RecepçãoConvites*

A aplicação *RecepçãoConvites* (cuja interface com o utilizador é apresentada na *Fig.8*) consiste num *label* com o logotipo do sistema e sinaliza que a aplicação de recepção de convites está activa, assim se o utilizador não estiver a participar numa sessão de trabalho poderá ser convidado a participar numa sessão de trabalho por um *Coordenador*.

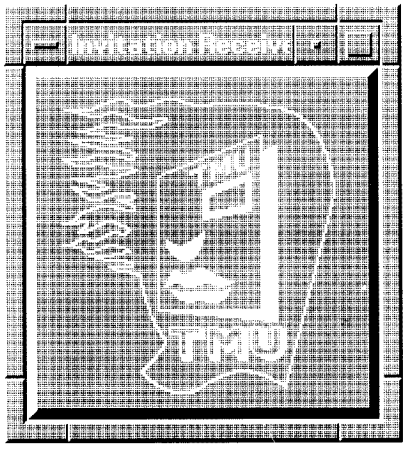


Fig.8 A interface da aplicação *RecepçãoConvites*

VI. PRÉ-AVALIAÇÃO

O trabalho cooperativo executado em terminais remotos pareceu denotar uma experiência um tanto ou quanto estranha para quem inicialmente a experimentava. A percepção humana sente talvez, nestes casos, a falta de algo mais. Uma aplicação como esta estaria mais completa, provavelmente, se existisse um serviço de voz. É que haviam determinadas operações que se tornavam aborrecidas de escrever... com voz seria outra coisa!

No que se refere ao uso de todos os recursos presentemente já em curso na aplicação *TMU* não deixa de, pelo que se pôde apurar, ser surpreendente. A apreciação geral daqueles que a experimentaram foi francamente positiva ao observarem um documento a ser editado por alguém que não viam e em seguida poderem reeditá-lo, permitindo o anterior interveniente assistir às suas acções sobre o mesmo documento. De igual forma, pareceu agradar a possibilidade de ter uma perspectiva geral do que se passava com todos os intervenientes (quem estava presente, quem tinha a palavra, quem a pretendia, quem queria sair, etc...). A aplicação Mensagem pôde ser inclusivamente interpretado como um meio de dar "dois dedos de conversa" à parte com um vizinho (que eventualmente até poderia estar fisicamente distante).

VII. PERSPECTIVAS FUTURAS

A comunicação verbal entre os utilizadores, na qual cada um dos comportamentos individuais é mais rapidamente afectado e influenciado pelos restantes, é necessária. A reforçar o argumento atrás referido está o facto da comunicação verbal ser a forma mais natural de comunicação e apresentar uma eficiência superior à escrita. A inclusão de canais de voz é, pois, essencial. Também seria conveniente desenvolver mais funções de edição gráfica nomeadamente funções de manipulação de imagens. Tais ferramentas gráficas permitiriam ao utilizador exprimir as suas ideias duma forma rápida, o que melhoraria a produtividade do grupo de trabalho.

Uma outra ferramenta a desenvolver seria um *clipboard*, o qual permitiria ao utilizador inserir no *Documento* texto, gráficos e imagens desenvolvidas noutras aplicações.

É importante referir que o código desenvolvido está estruturado e é modular o que facilita a implementação das capacidades ainda não desenvolvidas (mas para as quais a interface gráfica já se encontra preparada), a introdução de novas facilidades e a alteração do código existente.

O *Editor* é um editor de objectos (*widgets* no *OSF Motif*). Esta característica garante uma flexibilidade suplementar no desenvolvimento de novas capacidades para os objectos já existentes e na introdução de novos objectos, o que nos permite garantir que o código poderá ser continuado.

VIII. CONCLUSÕES

Este artigo descreve um sistema de edição cooperativa de documentos multimédia por um grupo de *Co-autores*, a trabalharem em estações de trabalho ligadas por uma rede local do tipo *Ethernet*. As interfaces, com os *Co-autores* são amigáveis ao utilizador e foram desenvolvidas tendo em conta a filosofia *WYSIWYG* (*What You See Is What You Get*) para uma aprendizagem mais rápida e uma utilização mais agradável.

Também é descrito o protocolo de gestão desenvolvido para coordenar as actividades dos diferentes *Co-autores*. Neste protocolo existe um *Coordenador* a quem compete convidar os utilizadores a participar na sessão de trabalho, autorizá-los a abandonar esta mesma sessão, assim como também gerir as intervenções dos *Co-autores* na edição do documento partilhado. Este protocolo de gestão garante também a cada momento o sincronismo entre as interfaces dos diferentes *Co-autores* de forma a implementar a filosofia *WISIWYS* (*What I See Is What You See*).

O sistema desenvolvido disponibiliza aos *Co-autores* canais de comunicação bidireccionais que permitem a troca rápida de ideias, planos, conceitos, informação de coordenação e conversas informais laterais.

AGRADECIMENTOS

Agradecimentos ao grupo de Telemática do INESC que disponibilizou o equipamento necessário à realização deste trabalho.

REFERÊNCIAS

[1] Santos A, Marcos A, "CoMEdiA - Uma Ferramenta para a Edição Cooperativa de Informação Multimédia", in Proc. 5º Encontro Português de Computação Gráfica, Aveiro, Fev. 1993.

Treino assistido por computador

Fernando Sérgio Barbosa, Nelson Pacheco da Rocha

Resumo - Pretende-se neste artigo dar a conhecer as dificuldades inerentes à concepção de um sistema de treino assistido por computador para ajudar pessoas em fase de reabilitação. De uma maneira geral, serão descritos os aspectos a ter em consideração para se desenvolver um sistema deste tipo e, de um modo particular, será apresentada a implementação de um sistema.

O sistema desenvolvido é composto por duas aplicações: uma destinada ao terapeuta onde este pode definir quais os aspectos que vão ser estudados e outra destinada à pessoa em reabilitação para que esta se possa exercitar de acordo com o objectivos definidos pelo terapeuta.

Abstract - This paper will describe the problems associated to the conception of a computer based training system to help disabled people during their rehabilitation process. It will describe general aspects related with the development of such systems and it will also present a particular implementation.

The developed system includes two applications: one is dedicated to the therapist for the preparation of the training sessions and the second one is dedicated to the disabled people for his rehabilitation process.

I - INTRODUÇÃO*

O ensino assistido por computador, com base em texto, gráficos e imagens estáticas, tem vindo a ser utilizado com sucesso em várias áreas. Particularmente no que se refere à reabilitação de pessoas com debilidade mental, o ensino assistido por computador pode propiciar estímulos à memória, estruturação de tarefas de aprendizagem e sequenciamento, identificação de novos conceitos e auxílio na resolução de problemas concretos. Por outro lado, pode também oferecer ritmos de aprendizagem controlados, repetição de processos e monitorização dos resultados da aprendizagem.

Estas características, para além de ajudar os sujeitos em processos de reabilitação, vem também em auxílio dos terapeutas, permitindo-lhes a eliminação de algumas tarefas, como, por exemplo, a necessidade de repetirem inúmeras vezes um determinado exercício.

As razões atrás evocadas, permitiram a introdução de treino assistido por computador em algumas áreas de reabilitação como é o caso do treino perceptivo [1]. No entanto, noutras situações, o treino assistido por computador tem-se mostrado ineficiente como acontece nos casos do treino vocacional ou de competências sociais.

Tal é devido à grande dificuldade de generalização por parte das pessoas com debilidade mental, pelo que, para lhes ensinar situações concretas como atravessar uma estrada ou manejar uma peça, os gráficos e as imagens estáticas são insuficientes. Este problema é, no entanto, ultrapassado com o aparecimento de sistemas multimédia que permitem o processamento e a apresentação de imagens em tempo real.

Um sistema multimédia pode processar, armazenar e transmitir imagens fixas e dinâmicas, audio, vídeo e texto sincronizados. Para a apresentação de vídeo, os sistemas devem ler os dados do disco, descomprimi-los e apresentá-los a uma elevada cadência conjuntamente com o som. Estes sistemas devem também gravar (por exemplo, o sinal vídeo proveniente de uma câmara e o som de um microfone), comprimir os dados e armazená-los em disco.

II - PERSPECTIVAS

Para poder aproveitar as características apresentadas pelas aplicações multimédia e compatibilizá-las com as necessidades de uma aplicação de treino/ensino optou-se por uma organização sequencial de imagens em que o utilizador escolhe, perante a situação descrita na imagem (fixa ou dinâmica), qual o rumo a dar à lição.

A interface com o utilizador tem de fornecer meios para possibilitar essa escolha. A solução entretanto encontrada, mas podendo ser alterada num futuro próximo, é a de diferenciar as zonas de imagem e a zona de interacção. Nesta última zona são criados botões, diferentes de imagem para imagem, para fornecer ao utilizador a tal hipótese de escolha. Assim, uma sessão de treino é composta por várias páginas estando cada página associada a uma imagem e a um conjunto de botões que definem o próximo estado da sessão (Fig. 1).

Não se pode esquecer que numa estrutura de treino para além da reprodução de sessões é também necessário, para que o sistema esteja completo, pensar na parte do terapeuta que deve ter ferramentas para poder criar novas sessões com material pedagógico alternativo.

O sistema é então subdividido em duas unidades: a

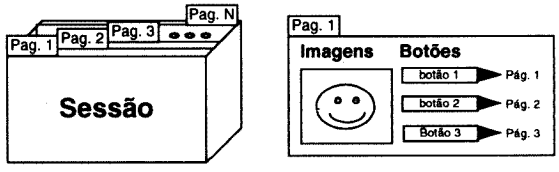


Fig. 1 - Composição de uma sessão

* Trabalho realizado no âmbito da disciplina de projecto

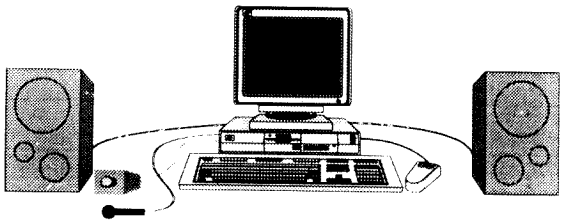


Fig. 2 - PC equipado para suportar o *Viewer* e o *Session Editor*

primeira designada por *Viewer* trata da reprodução das sessões e é pensada tendo em conta os utilizadores a que se destina, enquanto que a segunda, designada por *Session Editor,* trata da edição de sessões por parte do terapeuta.

III - ARQUITECTURA DO SISTEMA

Como se pretende criar uma aplicação que seja baseada num sistema multimédia é necessário que a máquina sobre a qual as aplicações desenvolvidas vão correr esteja equipada com o equipamento necessário para as suportar. Assim, enquanto que no *Viewer* são apenas necessários os sistemas de reprodução (como, por exemplo, a placa de reprodução de imagem e/ou som), já o *Session Editor* necessita um pouco mais de equipamento pois, além do equipamento reprodutor, esta aplicação precisa de equipamento de aquisição (como, por exemplo, microfones, câmaras, placa de aquisição de vídeo e/ou som, scanners, etc) para que o terapeuta possa editar as aulas (Fig. 2).

O sistema pode conter vários elementos de hardware pelo que se torna necessário fazer a interface destes com o software. Esta interface é feita, sob o ponto de vista das aplicações desenvolvidas, através do *Microsoft Windows* (Fig. 3) que dispõe para isso de "device drivers" associados a cada um dos elementos que compõem o hardware.

IV - VIEWER

Tentar-se-á descrever aqui a parte que será utilizada pelas pessoas em processo de reabilitação, ou seja, o reprodutor

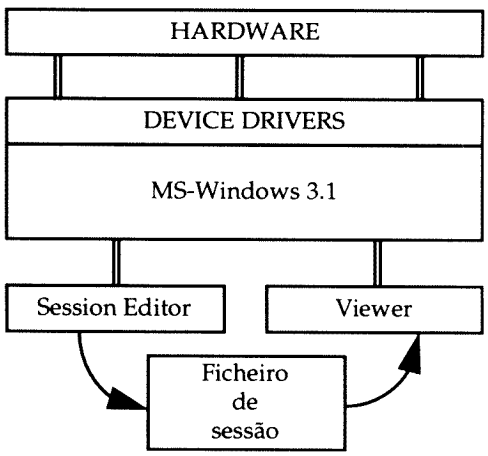


Fig. 3 - Arquitectura do software utilizado

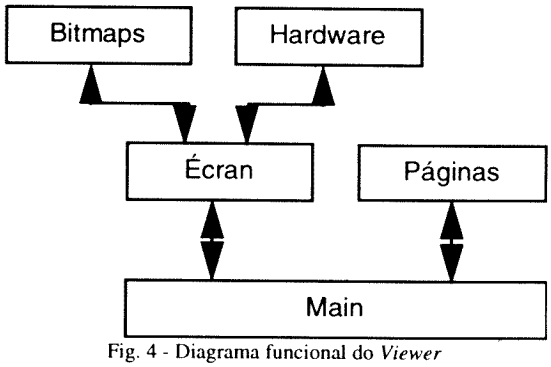


Fig. 4 - Diagrama funcional do *Viewer*

de sessões *Viewer*. Este terá de ter uma interface capaz de proporcionar a apresentação de imagens e, simultaneamente, os botões que permitem ao utilizador mudar de página.

Aspectos funcionais

Sob o ponto de vista funcional o sistema é subdividido em cinco módulos (Fig. 4):

Main - responsável pela coordenação dos restantes módulos;

Écran - responsável pela manutenção da aparência do écran, tal como criar e manter a interface, fazer a apresentação dos botões e tudo o que se relacione com as funções de desenhar no écran e controlar os dispositivos de entrada e saída de dados;

Bitmaps - responsável pela leitura e processamento dos bitmaps, imagens e ícons, passando ao módulo Écran apenas a informação do que este deve desenhar;

Páginas - responsável pela estrutura da sessão, definindo qual é a página activa, para qual das páginas se deve evoluir, quantos botões tem cada página, qual o ficheiro da imagem que está relacionada com a página e qual o ficheiro de ícon que está relacionado com cada botão, sendo ainda responsável pela leitura dos ficheiros de sessões;

Hardware - responsável pela reprodução das imagens dinâmicas e imagens estáticas que requeiram o uso de hardware específico.

Interface

Dado que o *Viewer* se destina a utilizadores com eventuais deficiências mentais, deverão existir, em princípio, cuidados especiais no desenho e implementação da interface com o utilizador. No entanto, nesta primeira fase, optámos por aplicações em que o sistema seja utilizado em simultâneo por uma pessoa em reabilitação e por um terapeuta, eliminando, portanto, as situações de auto-estudo, o que não requer tanto cuidado com a interface.

As interacções com o sistema são realizadas através de um rato e, eventualmente, através de um écran sensível ao toque. A configuração escolhida para o aspecto gráfico da interface está representado na Fig. 5.

Nesta figura pode-se ver a zona onde são reproduzidas as imagens e a zona onde são criados os botões relativos à página actual. Pode-se ainda verificar a existência de duas outras zonas que ainda não foram mencionadas. São as zonas de controlo de vídeo e do ajuste de imagem. A primeira possui um conjunto de botões semelhantes aos de um vídeo e cujas funções servem para controlar a exposição de imagens dinâmicas. A segunda zona contém 4 botões em forma de seta que são usados para ajustar a posição da imagem na zona de visualização quando as dimensões da imagem ultrapassam as dimensões desta. É de referir ainda a existência, tal como em todas as aplicações *Windows*, das barras do título e do menu. A barra do título apresenta o nome da aplicação, *Viewer*, ou, quando existe uma sessão aberta, o título da sessão, enquanto que a barro do menu apresenta os comandos que o utilizador pode realizar.

V - SESSION EDITOR

Como já foi dito anteriormente um sistema de treino não estaria completo se não fossem criados meios que possibilitem ao terapeuta preparar as lições de acordo com a evolução do aluno em reabilitação.

O primeiro passo dado neste sentido foi o de criar um conjunto de mnemónicas e uma pseudo-linguagem de programação para permitir ao terapeuta criar uma sessão editando um ficheiro com essa pseudo-linguagem. Este método, apesar de funcionar, requer que o professor possua conhecimentos da linguagem criada, para além de, em sessões com um grande número de páginas ou com páginas com um grande número de botões, o código escrito tornar-se muito confuso. Isto, associado ao facto de não existir um compilador de erros, apesar do *Viewer* fazer uma detecção simples antes de abrir a sessão, torna a tarefa de edição muito morosa e cansativa.

Surgiu, assim, a necessidade de desenvolver uma aplicação que fosse capaz de criar e editar as sessões de forma dinâmica, isto é, construir uma sessão sem escrever uma linha de código. A aplicação desenvolvida é uma espécie de ferramenta CASE em que o professor cria a sessão num ambiente semelhante aquele em que a sessão vai ser reproduzida.

A aplicação desenvolvida tem o nome de *Session Editor*, sendo a sua interface igual à do *Viewer* exceptuando a zona de controlo de vídeo que foi substituída por uma

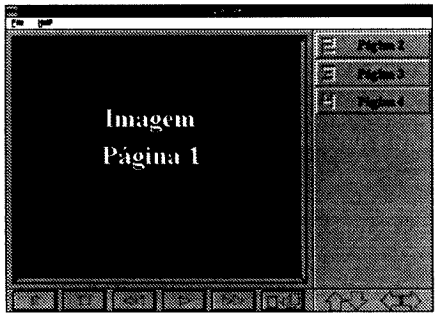


Fig. 5 - Interface do *Viewer*

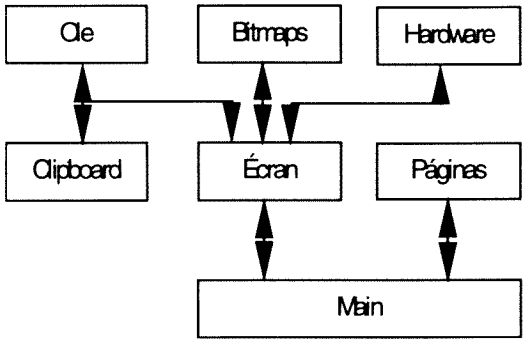


Fig. 6 - Diagrama funcional do *Session Editor*

toolbar.

Aspectos funcionais

Sob o ponto de vista funcional o sistema é subdividido em sete módulos (Fig. 6):

Os módulos Main, Páginas, Écran, e Bitmaps tem o mesmo funcionamento que tinham no *Viewer*, o bloco Hardware tem também o mesmo funcionamento mas é agora também responsável por adquirir imagens ou som além de os reproduzir;

Clipboard - responsável pela passagem de informação, nomeadamente imagens estáticas, entre aplicações de aquisição de informação e o *Session Editor*;

Ole - responsável pela troca de informação entre servidores "Object Linking and Embedding" (OLE) [2] e o *Session Editor*.

VI - FORMATO DOS FICHEIROS

Uma sessão completa é constituída por vários ficheiros. Um deles, que é o da própria sessão, com a extensão '.SON', que contém a descrição das páginas, introdução e outros parâmetros. Há outros ficheiros que contêm as imagens das páginas, filmes, sequências de som e os ícons associados aos botões. Para que a aplicação funcione sem problemas, os ficheiros terão de se encontrar no mesmo directório do ficheiro '.SON' pois é neste directório que a aplicação vai procurar os ficheiros necessários quando deles precisar.

Para se compreender completamente o formato dos ficheiros há vários conceitos que é preciso adquirir, alguns dos quais já mencionados mas não explicados. Os conceitos necessários são os seguintes:

Título da sessão - é o texto que aparece na barra do título, quando a sessão é aberta, e que deve ser uma descrição sumária, máximo de 32 caracteres, do conteúdo da sessão;

Introdução - A introdução, uma característica opcional de uma da sessão, é composta por uma imagem que será vista apenas quando a sessão é iniciada e que ocupará todo o écran com a excepção do título e do menu, servindo como uma capa da sessão e elucidando sobre o conteúdo ou tema da sessão;

```
BS                      P#2
Título da sessão        BEGINP
3                        NOIMG
INTRO                    LAST
introdu.bmp              NOSOUND
                         1
P#1                      BUTTONS
BEGINP                    Botao 1
 img_pag1.bmp             icon_1_2.bmp
 IMG                      GOTO#3
 sound_1.snd              PASS
 2                      ENDP
 BUTTONS
  botao 1               P#3
  icon_1_1.bmp          BEGINP
  GOTO#2                 movie_3.mov
  PASS                   MOV
                         sound_3.snd
  botao 2                1
  NOICON                 BUTTONS
  GOTO#3                  Botao 1
  BACK                    NOICON
ENDP                      GOTO#1
                          BACK
                        ENDP
```

Fig. 7 - Exemplo de um ficheiro de sessão

Modos de passagem de página - Há dois modos de passagem de página. O primeiro, designado por Pass, transfere o controlo da sessão para a página seguinte enquanto que no segundo, Back, o controlo permanece na página actual pelo que da página seguinte apenas será visualizada a imagem mas os botões não serão criados. Para retornar o controlo o utilizador deve seleccionar o botão Last Page. O uso deste botão não se limita a estes casos tendo um uso mais geral, isto é, sempre que se deseja voltar atrás na sequência de páginas carrega-se neste botão.

VII - CONCLUSÕES

Neste artigo foi descrita a implementação de um sistema de ensino assistido por computador baseado em sessões de treino, em que cada sessão é subdividida em páginas sendo cada página composta por um conjunto de botões e uma imagem (fixa ou dinâmica). O sistema, tal como foi descrito, é subdividido em duas partes: uma, *Session Editor*, que se destina à preparação das sessões e a outra, *Viewer*, que se destina à reprodução das sessões pela parte do indivíduo em reabilitação.

Refira-se que esta segunda parte foi desenvolvida tendo em vista não uma utilização autónoma por parte da pessoa em reabilitação, mas sim como uma ferramenta que o terapeuta poderia utilizar para ajudar este no seu processo, ou seja, é suposto o terapeuta acompanhar o desenvolvimento da sessão.

REFERÊNCIAS

[1] Nelson Pacheco da Rocha, Leonor Moniz Pereira, Clara Cidade, "Remote Learning and Training Services for Eldery and Disabled People", TeleTeaching 93, Trondheim, Noruega, Agosto de 1993 - IFIP Transaction, A-29, "TeleTeaching", page 715-720, IFIP, North-Holland.

[2] "Object Linkings and Embeddings, Programmers Reference", Microsoft Press, 1992.

A System for Computer Assisted Analysis of Ventriculograms

Beatriz Sousa Santos, Orlando de Sousa, Delfina Coelho, Pedro Magalhães, Luis Batista

Abstract - **Left Ventricle (LV) volume and contractile abnormalities can be important manifestations of coronary artery deseases. Cineangiography is used extensively to visualize the LV and may provide quantitative information about its performance.**

In routine clinical practice the cineangiograms are, in many laboratories, assessed visualy, which results in significant inter and intra observer variation. For this reason the quantification of the left ventricle function is desirable. We have developed a system for the semi-automated quantitative analysis of the LV to help cardiologists in routine clinical practice and research. It was developed on a MS-DOS platform, with image processing hardware and a graphical user interface. A report is produced for each patient including graphics of the results and patient data. This system is currently instaled at Centro Hospitar de Gaia to be tested.

Sumário - **As anormalidades de volume e contracção do ventrículo esquerdo (VE), podem ser manifestações importantes da doença coronária. A cineangiografia é usada extensivamente para visualizar o VE e pode fornecer informação quantitativa sobre o seu desempenho.**

Em prática clínica de rotina os cineangiogramas são, em muitos laboratórios, inspeccionados visualmente o que resulta em significativa variabilidade intra e inter-observador sendo desejável quantificar a função ventricular. Foi desenvolvido um sistema para a análise semi-automática quantitativa da função do VE que se pretende que seja uma ferramenta útil ao cardiologista na análise dos ventriculogramas na clínica de rotina e na investigação. Este sistema é baseado numa plataforma MS-DOS com hardware de processamento de imagem e uma interface gráfica com o utilizador. No fim da análise correspondente a cada paciente é produzido um relatório que inclui dados do paciente e resultados sobre a forma gráfica. Este sistema está instalado no Centro Hospitalar de Gaia com a finalidade de ser testado.

I. INTRODUCTION*

Left Ventricle (LV) volume and contractile abnormalities can be important manifestations of coronary artery deseases. Cineangiography is used extensively tovisualize the LV and may provide quantitative information about its performance.

LV cineangiography is a technique that allows the visualization of the LV silhouette along the cardiac cycle after the injection of a contrast product and using X-rays. These images contain a great deal of information which is difficult to apreend by visual inspection. In routine clinical practice, in many laboratories, the cineangiograms are still assessed visualy which results in significant inter and intra-variation, that is why the quantification of LV performance is desirable. A great number of methods of quantifying this performance have been proposed, but there is no consensus on the best method[1]. Because it is important to have the possibility to use more than one method and since the LV quantification, using any of these methods, is too time consuming to perform manually on a routine basis, some semi or fully automated systems for the quantitative analysis of the LV cineangiograms have been developed [1,2,3,4].

* Trabalho realizado no âmbito da disciplina de projecto

II. SYSTEM OVERVIEW

Our system was developed to help cardiologists in routine clinical practice and research. It is a new version, more friendly and complete, of a previously developed system [4], based on a MS-DOS platform with image processing hardware and a graphical interface. We shall describe the system in further detail in the next sections.

A. System Configuration

The LV cineangiographic images are currently recorded at the catheterization Laboratory of the Centro Hospitalar de Gaia using a video tape recorder but, in the near future, we will have access to digital images directly from a DSA (Digital Subtraction Angiography) system.

The system is based on a MS-DOS platform with image processing hardware (a commercial frame grabber with A/D flash converter, a frame buffer of 512x512x12bits, input and output Look Up Tables that allow some real time image processing and three D/A converters for pseudo-colour display) and a video colour monitor (figure 1).

Software was developed mostly in C, however some routines to handle the frame grabber were written in

Assembly. It is modular and open, allowing the adition of any new functionality.

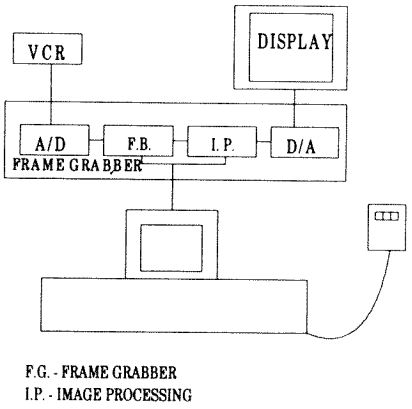


Figure 1- The system is based on a MS-DOS platform with image processing hardware, video tape recorder, a mouse and a video monitor

B. Functionality

This system offers the user the following functionality:

i)- Image acquisistion, calibration and interest zone definition;

ii)- Image enhancement (histogram modification and pseudo-colour);

iii)- Manual and semi-automated contour detection and edition;

iv)- contour processing according to several methods of quantitative analysis of LV performance;

v)- Image, contour and results archive;

vi)- Final report including patient data, images and analysis results presented as graphics.

C. User Interface

The main goal of this user interface is to assure that it can be used by clinicians who don't need to have previous experience with computer systems. Considering the users' profile and the nature of the task they will have to perform using this system, we chose to use menus as main dialog style. The interface is iconic and offers on line help to the user as well as protection against many possible wrong entries. Warnings and error messages are produced whenever necessary. Some features that must be available everywhere can be invoqued also using function keys.

D. Image acquisition and calibration

Images are digitized using a spatial resolution of 254x254, considered to be adequate for LV cineangiograms [1] and a tonal resolution of 256 grey levels (8 bits), which leaves the other 4 bits to graphic overlays.

Until now no distortion correction is done. A calibration factor is obtained from a grid of known step filmed at the begining of the catheterization procedure. Assuming that the existing distortion is zero near the centre of the images and using the four central nodes it is possible to generate an undistorted grid. Each set of nodes defining corresponding areas in both grids (distorted and undistorted) allow to compute a distortion transformation for each of these areas. Using these transformations and grey level interpolation it is possible to generate an undistorted image. These transformations are computed only once and can be used for all cineangiograms of each patient. Currently we are testing and evaluating several procedures and techniques.

E. Image preprocessing

The preprocessing techniques implemented are pseudo-colour and histogram operations. These techniques were used to produce images of enhanced 'quality' from the user's point of view, since in this system the user must validate (and correct if necessary) all detected contours. These histogram operations are performed only over an interest zone previously defined (by the user using the mouse) and have no effect on the contour detection algorithms.

E. Contour detection

After acquiring the images, defining the interest zone, and having some preprocessing if necessary, the user can detect the LV contours manually (using the mouse) or choose a contour detection algorithm. In the first version of this systrem [4] two types of algorithms were available: i)- 'one image algorithms' which detect the LV contour on the current image using the information contained in this image and information introduced interactively by the user

ii)- intermediate contour detection algorithms- which use the information contained in the current image along with information from the contours previously detected on images corresponding to the same patient.

All these algorithms needed an interest zone and now we are developing new algorithms which do not use an interest zone.

F. Contour processing: global and regional parameters

After detecting all the contours corresponding to a patient (the minimum number is 2: end-systolic and end-diastolic) it is possible to compute global and regional parameters that characterize the LV performance during contraction. The global parameters computed are volume, corresponding to each detected contour (using Chapman's model), and ejection fraction [4] defined as:

$$EF = \frac{Volume(end\text{-}diast) - Volume(end\text{-}syst)}{Volume(end\text{-}diast)}$$

The computation of regional parameters, regional ejection fraction or regional wall contractility is done using three different models for the left ventricle contraction:

i)- Standford Model [6]
ii)- Slager Model[1]
iii)- Centerline Model[7]

G. Report

After completing the analysis, the user can produce a printed report including patient data, images and graphical results of the models. This report is generated in POSTSCRIPT language.

III. DISCUSSION

This system is instaled at the Catatherization Laboratory of Centro Hospitalar de Gaia to be tested in clinical routine. The new contour detection algorithms have to be validated and we have to define which is the mostconvenient distortion correction procedure. In the near future it will be possible to obtain images directly from a DSA system, in which case the overall procedure is simpler since we have already digital images.

AKNOWLEDGEMENT

The authors wish to thank the staff of the Catheterization Laboratory at the Centro Hospitalar de Gaia for its important colaboration on this project.

REFERENCES

[1] Reiber, J.H.C., P.W. Serruys, C.J. Slager, *Quantitative Coronary and Left Ventricular Cineangiography*, Dordrecht, Martinus Nijhoff Plubishers, 1986

[2] Lerallut, J.F, P. Gambier, G. Fontenier, "Semi-automated Medical Image Processing", *Journal of Medical Eng. & Technology*, vol. 10, n.1, 1986, pp24-26

[3] Toraichi, K., K. Katagishi, R. Mori, "A Left Ventricular Function Analyser and its Applications", IEEE Transaction on Biomedical Eng., vol. BME-34, n.5, May 1987, pp317-328

[4] Sousa Santos, B., *Análise Assistida por Computador de Cineangiogramas do Ventrículo Esquerdo*, PhD thesis, University of Aveiro, 1989

[5] Gonzalez, R. , R. Woods, *Digital Image Processing*, Addison-Wesley, 1992

[6] Ingels, N., C. Mead, G. Daughters, E. Stinson, E. Alderman, A New Method for the Left Ventricular Wall Motion", *Computers in Cardiology*, IEEE Computer Society, 1978, pp 57-61

[7] Sheean, F., E. Bolson, H. Dodge, S. Mitten, "Centerline Method- Comparison with other methods for measuring regional Left Ventricular motion", *Ventricular Wall Motion*, ed. Sigwart, U. and P. Heintzen, Suttgard, New York, Thieme-Statton, 1984, 139-149

Sistema Integrado de Aquisição de Dados para Potenciometria

Rui Manuel Gonçalves, Sérgio Santiago e Santiago, António Rui Borges, Artur Carneiro Pereira
DETUA
Armando Duarte, Isabel Boal
DQUA

Resumo - **Este artigo descreve um sistema integrado de aquisição de dados para potenciometria desenvolvido sobre o sistema operativo Windows 3.1 da Microsoft. Este sistema é basicamente constituído por um computador ligado através das suas portas série a uma bureta e a um potenciómetro. A comunicação com estes periféricos é *interrupt-driven*.**

Uma vez que se destina a utilizadores com poucos conhecimentos na área da computação o sistema foi concebido por forma a ser o mais intuitivo e o mais iteractivo possível e, por forma a permitir um controlo o mais apertado possível sobre o processo de aquisição.

Abstract - **This paper describes an integrated data acquisition system for potentiometrie. It is intended to run under Microsoft Windows 3.1. The system assumes a hardware set up consisting of a PC connected (through is serial ports) to a potentiometer and to a burette. The communication with this two devices is interrupt driven.**

Since it aims to be used by non-computer experts, it creates a user friendly environment which enables a tight control over the acquisition process.

I. INTRODUÇÃO*

A.Potenciometria

A utilização de técnicas potenciométricas na área da instrumentação quimica permite estimar a concentração de iões específicos em soluções. Um procedimento comum consiste em adicionar volumes sucessivos (constantes ou variáveis) de um determinado produto a uma solução e medir, após cada adição, o potencial existente num eléctrodo aí mergulhado. As curvas, assim obtidas, são depois processadas para a extracção de parâmetros característicos da solução e/ou da reacção que teve lugar.

Este projecto visa automatizar o processo de medida e construir um ambiente integrado que possibilite aos utilizadores um controlo tão apertado quanto possível da aquisição de dados, quer ao nível da variação de parâmetros específicos (volumes do produto a adicionar, tempos de estabelecimento, etc.), quer a nível de pré-processamento *on-line* dos valores já adquiridos.

* Trabalho realizado no âmbito da disciplina dê Projecto.

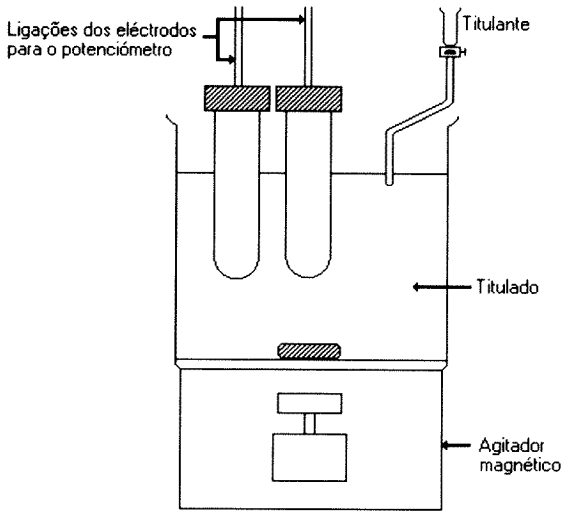


Fig. 1
Esquema de uma Titulação Potenciométrica.

B.Sistema Operativo Windows 3.1

O sistema operativo Windows 3.1 da Microsoft permite uma infindável variedade de possibilidades no projecto e concepção de aplicações. Quase todo o projecto das aplicações é feito sem o conhecimento arquitectónico da máquina, sem preocupações quanto ao interface com o monitor, impressoras, rato, etc...

O Windows 3.1 oferece ao programador toda uma nova filosofia de programação. Com o evoluir do software e hardware dos computadores, começa a deixar de ser necessário o projecto de aplicações diferentes para máquinas diferentes. No Windows 3.1 existe todo um novo modelo de programação, todo um novo conceito de ambiente de programação e um infindável número de novos conceitos e termos que se tornarão, de algum modo, de uso comum no futuro.

Em Windows 3.1 é permitido que diferentes aplicações corram concorrentemente, e é permitido às aplicações o compartilhar de dispositivos de entrada/saída, memória, video e do CPU. Há toda uma série de funções de utilização e acesso a estes dispositivos à disposição do programador. O uso destas funções, para além de (nalguns casos) facilitar o trabalho do programador, garante o não conflito entre aplicações no acesso a dispositivos comuns. O Windows 3.1 funciona como um árbitro sobre as aplicações, gerindo a ocupação de cada recurso e o tempo de ocupação por parte de cada aplicação. No caso da comunicação através das portas, o Windows 3.1 dispõe de

funções que permitem a utilização de *buffers* de tamanho variável para cada porta, e a comunicação é *interrupt-driven*.

Porém, apesar do Windows 3.1 ser um ambiente *multitasking*, é também *nonpreemptive* pelo que uma aplicação deve partilhar o CPU e não deve tentar tomar o seu controlo exclusivo. O Windows 3.1 não poderá retomar o controlo do CPU a não ser que a aplicação lho devolva.

O Windows 3.1 só pode receber ou enviar informação duma ou para uma aplicação através de eventos. O Windows 3.1, quando um evento é gerado, "lê-o" e envia uma mensagem à aplicação, à qual pertence o objecto sobre o qual ocorreu o evento. Todas estas mensagens vão para um *buffer* do tipo FIFO, denominado *message queue*.

Este tipo de *multitasking* é um método cooperativo que confia no facto das aplicações, elas próprias, partilharem alternadamente os recursos do sistema.

Em Windows 3.1 uma aplicação não deve aceder à memória ou a dispositivos entrada/saída directamente (sem o controlo do Windows).

No projecto de uma aplicação deve-se fazer com que o utilizador tenha a sensação que a controla, e não que é controlado por ela, ou seja, a aplicação deve ser moldada de modo a adaptar-se ao utilizador e não o contrário.

A aplicação deve ser o mais INTERACTIVA possível e deve ser facilmente ADAPTÁVEL às exigências do utilizador em aspectos como a sua aparência (cor) e funcionabilidade (menus).

O interface da aplicação deve ser totalmente TRANSPARENTE ao utilizador, de modo a permitir que este apenas faça uso dela sem que para isso seja necessário um conhecimento profundo do seu funcionamento.

O interface deve ser o mais INTUITIVO possível, e a aplicação deve actuar de modo a que o utilizador se aperceba de todas as suas acções e respectivas consequências sem ter que ler o manual da aplicação.

A CONSISTÊNCIA da aplicação deve ser vista de dois pontos: do ponto de vista do utilizador - CONSISTÊNCIA EXTERNA - e do ponto de vista do Windows - CONSISTÊNCIA INTERNA. A CONSISTÊNCIA EXTERNA da aplicação deve ser baseada na experiência do utilizador e no seu habitual ambiente de trabalho. A CONSISTÊNCIA INTERNA da aplicação refere-se a manter a filosofia de funcionamento e utilização das aplicações em Windows.

Todo o texto, elementos visuais e ideias devem ser organizados de modo a criar um écran visualmente agradável, sem por isso distrair o utilizador. As cores escolhidas devem contrastar entre si, mas não devem agredir visualmente o utilizador, e devem ser escolhidas em função do seu gosto e das condições de luz do ambiente de trabalho. Todos os objectos semelhantes devem ser mantidos juntos e sempre que possível devem criar-se comandos tridimensionais e de fácil utilização.

O utilizador deve ter *feedback* imediato das suas acções, ou seja, se seleccionar um objecto ou uma parte de texto (num editor), o écran deve revelar esta selecção. Embora este *feedback* gráfico seja primordial, o *feedback* textual ou áudio podem também ser usados.

Erros do utilizador não devem ser fatais, pois o utilizador normalmente usa tentativas e erros para aprender a usar uma aplicação. A possibilidade de escolhas erróneas deve ser minimizada, e sempre que um erro ocorra a sua correcção deve ser facilmente efectuada pelo utilizador. As mensagens de erro, que normalmente são transmitidas ao utilizador pela aplicação, devem, para além da informação de erro, especificar resumidamente o modo de o ultrapassar, sempre que tal seja possível.

Deve utilizar-se, sempre que possível, as potencialidades do sistema e da aplicação, em vez das do utilizador.

Todo o projecto da aplicação deve ser feito em função do utilizador e da facilidade de utilização, e não só da funcionalidade. Os testes de utilização são primordiais e críticos para o sucesso e aceitação de uma aplicação. Uma boa apresentação gráfica, e até áudio, da aplicação pode impressionar o utilizador numa primeira análise superficial, porém, se a aplicação não for cómoda e facilmente inteligível e utilizável, fará com que o seu sucesso seja limitado.

II. DESCRIÇÃO DO AMBIENTE

O ambiente de trabalho foi concebido de modo a ser o mais interactivo e intuitivo possível e de modo a reduzir ao mínimo a intervenção do utilizador no processo de aquisição.

Os parâmetros iniciais da aquisição podem ser modificados durante o decorrer da mesma, possibilitando ao utilizador um controlo tão apertado quanto possivel do decorrer da experiência. Os valores adquiridos são, além de visualizados no écran, armazenados (juntamente com outros valores relativos à experiência) num ficheiro. O utilizador tem também a possibilidade de armazenar, num outro ficheiro, toda a "história" do processo de aquisição efectuado.

Os valores de potencial dos eléctrodos são adquiridos por um potenciómetro em mV e são visualizados no écran quer em forma de texto, quer em forma gráfica, permitindo deste modo uma mais fácil monitorizacão da aquisicão.

A. Filosofia da Aquisição

Todo o processo de aquisição de valores e cálculo de incrementos é transparente ao utilizador. Este apenas tem de escolher a estratégia a seguir durante a experiência, quanto a, nomeadamente critérios de estabilidade a aplicar aos valores lidos, modo de cálculo dos incrementos a adicionar à solução, decisão de fim de experiência e selecção dos eléctrodos a utilizar.

Existem dois tipos de critérios de decisão quanto à estabilidade dos valores lidos. Estes critérios são mutualmente exclusivos.

Pode-se, por um lado, considerar um valor final estável se a sequência de valores médios obtida num dado intervalo de tempo (Tempo de Estabilização) estiver toda contida no intervalo delta - critério de intervalo fixo. Por outro lado, e este será outro critério de decisão, pode-se considerar um valor final estável quando se obtém uma sequência de valores médios (Nº de Valores Médios) todos iguais - critério de intervalo variável.

Porém, pode dar-se o caso de o tempo de espera para obtenção de um valor estável, segundo um dos critérios atrás definido, seja demasiado grande ou mesmo infinito (o sistema é intrinsecamente instável). Há, pois, toda a vantagem em estabelecer-se um tempo máximo de decisão (Tempo Máximo de Medida). Assim, se se atingir este tempo máximo de decisão sem se obter a estabilidade, o valor final é calculado também a partir da sequência de valores médios, e é considerado instável.

Para uma maior fiabilidade dos valores finais, os valores a partir dos quais se toma a decisão, são eles próprios médias de um número (Nº de Medicões (Média)) de valores lidos directamente do potenciómetro.

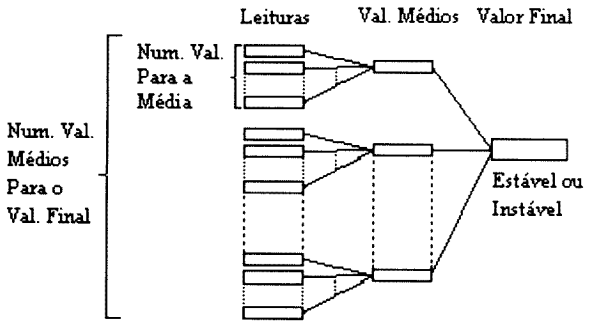


Fig. 2
Método para o Cálculo de Valores Finais.

O cálculo dos incrementos de volume de titulante a adicionar à solução pode seguir dois critérios, também mutualmente exclusivos.

O primeiro critério é o de impor um incremento constante. O outro critério é o de definir um valor máximo para a variação de potencial pretendido e um valor máximo para o incremento de volume a adicionar à solução, impondo assim os limites de definição da curva. Se a variação de potencial entre dois valores consecutivos não ultrapassar a definição imposta, o novo incremento a adicionar será o imposto como máximo. Se a variação de potencial ultrapassar a definição imposta, o novo incremento é calculado segundo a interpolação linear indicada na Fig. 3.

Estes valores de incremento de volume são adicionados à solução através de uma bureta ligada ao computador

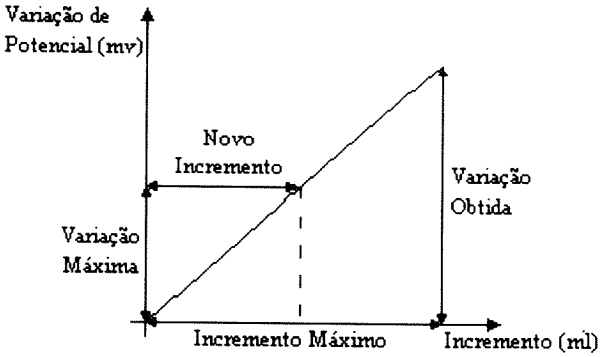


Fig. 3
Método de Cálculo de Incrementos Variáveis.

O fim da experiência é imposto quando pelo menos uma das três condições seguintes é verificada:

- ♦ Foi atingido o valor esperado como final da experiência.
- ♦ Foi atingido o número máximo de pontos de amostragem especificado.
- ♦ Não se obteve uma variação de potencial considerada mínima.

B.Reagentes e parâmetros da reacção

O utilizador introduz os dados referentes aos reagentes que vão intervir na reacção e os parâmetros que a controlam através de dois quadros normalmente designados por *Dialog Boxes*.

Dialog Boxes são janelas que são utilizadas para efectuar o "diálogo" entre uma aplicação e o utilizador. Os seus campos podem ser acedidos através da utilização do rato e/ou do teclado. Para além de campos de edição, as *Dialog Boxes* possuem dois botões: o botão Ok e o botão Cancelar. O botão Ok faz a validação de todos os dados introduzidos. O botão Cancelar apenas fecha a *Dialog Box*, cancelando qualquer alteração nos dados que tenha sido efectuada pelo utilizador.

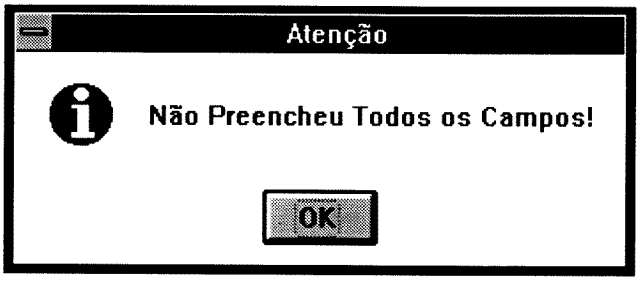


Fig. 4
Message Box Usada para Notificar o Utilizador
da Falta de Preenchimento dum Campo de Edição.

Todos os campos de edição duma *Dialog Box* têm que estar preenchidos para que os dados introduzidos sejam considerados válidos. Isto é, se o utilizador tentar sair da *Dialog Box* premindo o botão Ok, é confrontado com uma mensagem de aviso sendo o foco colocado num campo

não preenchido. Esta mensagem de aviso ao utilizador é feita através de uma *Message Box* representada na Fig. 4.

A *Dialog Box* que permite ao utilizador a introdução dos dados relativos aos reagentes da experiência intitula-se "Reagentes - Inicialização" e o seu aspecto é representado na Fig.5. O acesso a esta *Dialog Box* só é possível no início de cada experiência.

Todos os valores introduzidos nos campos desta *Dialog Box* são apenas de interesse informativo para o utilizador, e são colocados à cabeça do ficheiro de armazenamento de dados.

A *Dialog Box* intitulada "Parâmetros de Inicialização e Estabilização" permite ao utilizador a introdução dos parâmetros de controlo da experiência, e está representada na Fig. 6.

Na caixa "Estabilidade", pode ser escolhido o critério de estabilidade, introduzidos os valores dos parâmetros a ele referentes e introduzido o valor do tempo necessário ao estabelecimento da reacção.

Na caixa "Eléctrodos", estão os campos de selecção dos eléctrodos que se pretende utilizar no decorrer da aquisição. Estes campos só se encontram disponíveis na definição inicial das condições da aquisição. Após o arranque desta, o seu estado não pode mais ser modificado.

Na caixa "Tipo de Incrementos", o utilizador pode introduzir o valor do incremento inicial de volume a adicionar à solução e escolher o tipo de incrementos a fazer posteriormente.

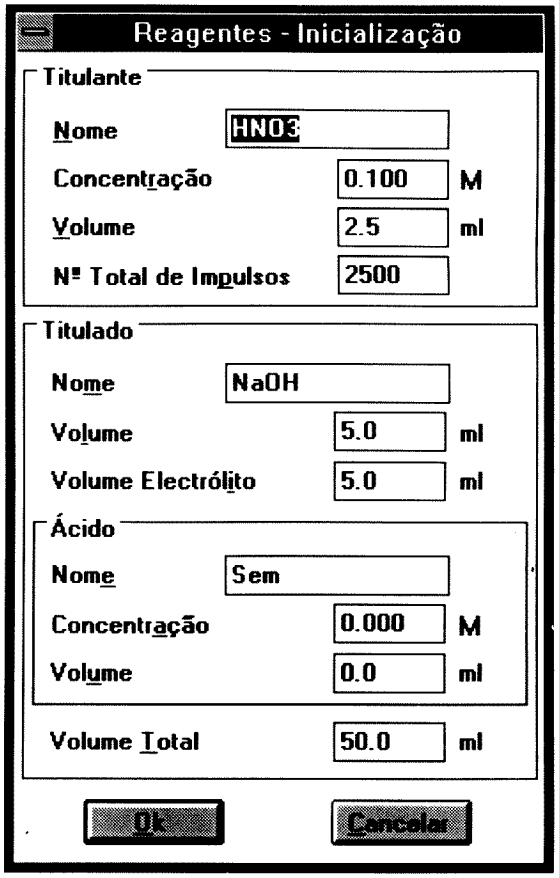


Fig. 5
Dialog Box "Reagentes - Inicialização".

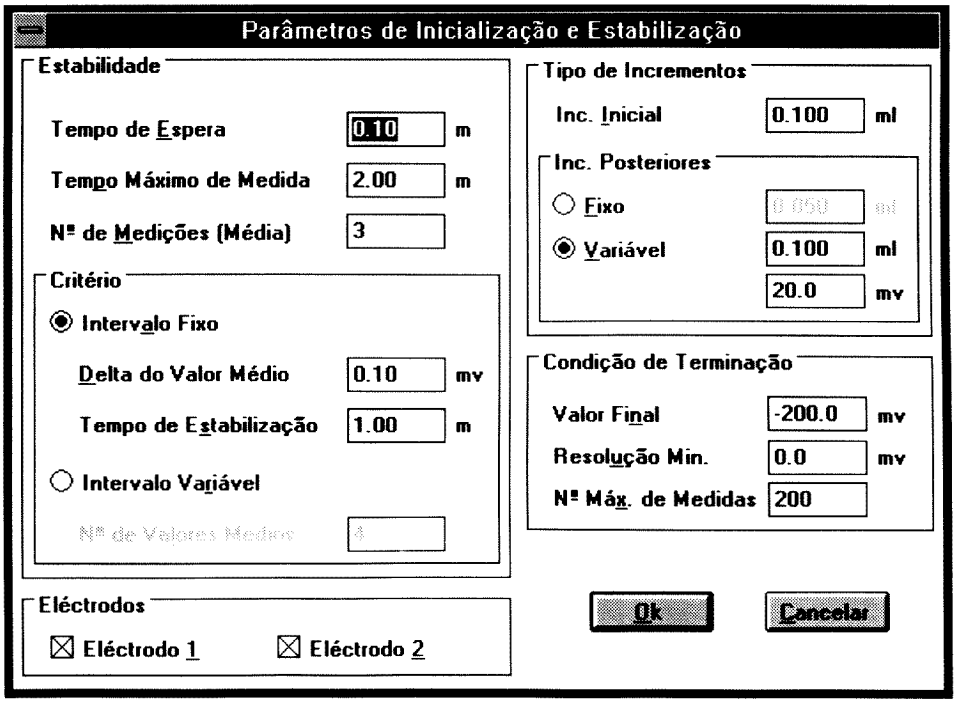


Fig. 6
Dialog Box "Parâmetros de Inicialização e Estabilização".

Na caixa "Condição de Terminação", o utilizador define as condições de decisão do fim de experiência.

Os valores introduzidos nos campos desta *Dialog Box* não podem ser "absurdos". Ou seja, não podem ser valores que façam com que a aquisição se torne impraticável (Ex: O tempo de estabilização tem de ser tal que permita pelo menos uma medição e, por outro lado, não pode ser maior que o tempo máximo de medida). Se o utilizador tentar introduzir um valor "absurdo", esse valor é corrigido e o foco direccionado para o campo de edição onde ocorreu o erro e respectiva correcção, dando assim hipótese ao utilizador de alterar o valor inserido pela aplicação.

C.Armazenamento de dados

Para efectuar o armazenamento dos dados referentes à reacção, de maneira a que se possa mais tarde processá-los, é permitida ao utilizador a introdução do nome do ficheiro para o armazenamento. Este procedimento é feito na *Dialog Box* "Armazenamento de Dados", onde o utilizador poderá também escolher o armazenamento, ou não, de toda a "história" da reacção. Para além dos reagentes, parâmetros e valores finais que são armazenados no primeiro ficheiro, o ficheiro da "história" da reacção contém também todos os valores médios correspondentes a cada valor final.

Fig. 7
Dialog Box Armazenamento de Dados.

D.Visualização dos resultados

O utilizador possui dois modos diferentes de monitorização do processo de aquisição no écran, um em forma de texto, e o outro em forma de gráfico. Assim, poderá acompanhar o desenrolar da experiência, e poderá controlá-lo através da mudança dos parâmetros.

Na janela "Gráfico", o utilizador poderá visualizar a curva da variação do potencial medido no(s) eléctrodo(s) mergulhado(s) na solução em função dos incrementos de volume de titulante adicionados. Esta janela de visualização possui no canto inferior direito um esquema de cores dos pontos e segmentos de recta de modo a que o utilizador possa diferenciar a(s) curva(s) referente(s) ao(s) eléctrodo(s) usado(s) e os respectivos pontos finais estáveis e instáveis.

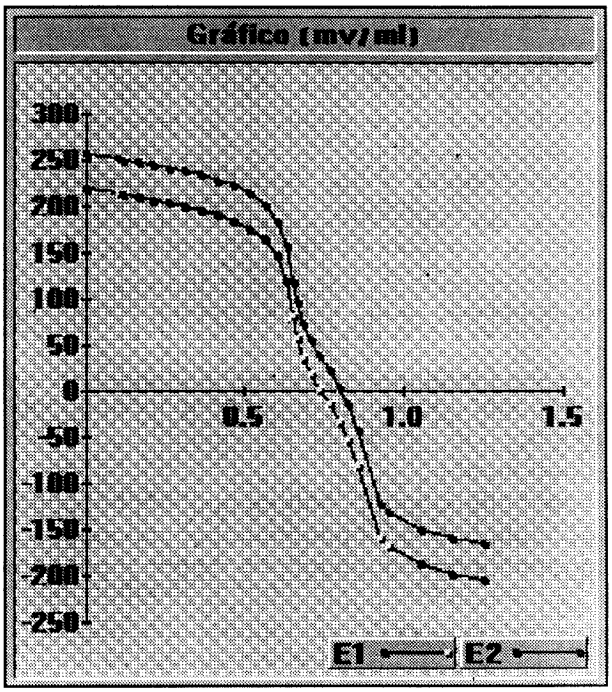


Fig. 8
Janela de Visualização "Gráfico".

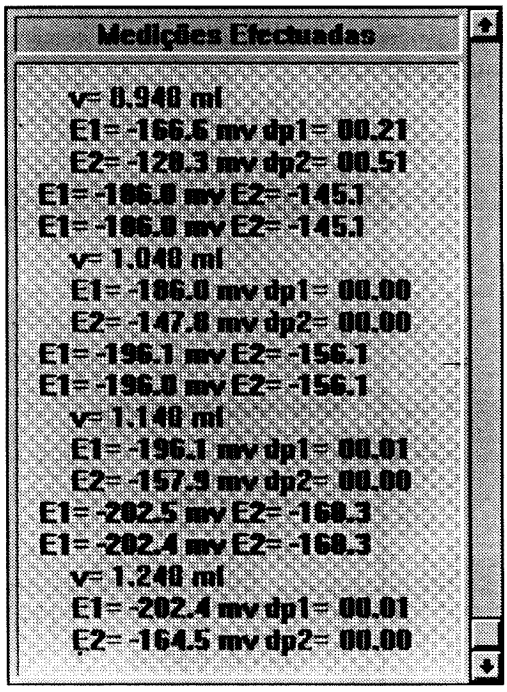


Fig. 9
Janela de Visualização "Medições Efectuadas".

Na janela "Medições Efectuadas" o utilizador poderá visualizar a listagem dos valores do potencial medido no(s) eléctrodo(s) mergulhado(s) na solução, para os correspondentes incrementos de volume de titulante adicionados. Para além dos valores finais, o utilizador poderá também visualizar os valores médios de potencial

correspondentes. Os valores são representados em cores diferentes de modo a se diferenciar valores médios de valores finais de potencial e valores finais estáveis de valores finais instáveis.

Esta janela possui uma barra de *scroll* vertical, permitindo ao utilizador um deslocamento dos valores na janela, de maneira a poder visualizar alguns valores passados.

III. FUNCIONAMENTO DO SISTEMA

Ao iniciar a aplicação é efectuada a abertura da comunicação com a bureta e o potenciómetro através das portas série do computador. Se ocorrer um erro na abertura da comunicação, o utilizador é avisado da sua ocorrência e a aplicação é abandonada e volta ao seu estado minimizado (*Icon*).

Depois de aberta a comunicação com a bureta e o potenciómetro, o utilizador tem à sua disposição quatro comandos na barra de menu da janela principal da aplicação: "Reagentes...", "Parâmetros...", "Começar" e "Sair".

Activando os comandos "Reagentes..." ou "Parâmetros...", o utilizador tem acesso às *Dialog Boxes* "Reagentes - Inicialização" ou "Parâmetros de Inicialização e Estabilização" respectivamente.

Ao activar o comando "Começar" é iniciado o processo de inicialização da aquisição. É testada a ligação do potenciómetro e da bureta, ao computador e à alimentação. Enquanto a aplicação detectar falhas nas ligações, avisa o utilizador para as confirmar.

Caso a bureta não esteja cheia aquando do início da aplicação, o utilizador tem a possibilidade de reenche-la pois a aplicação confirma a sua pretensão em fazê-lo ou não. Por outro lado se o utilizador não mudou os valores dos reagentes ou parâmetros antes de activar o comando "Começar", a aplicação permite-lhe ainda antes de começar a aquisição a possibilidade de o fazer, ou então se o desejar começa com os valores de defeito armazenados na última experiência.

Ainda antes do arranque propriamente dito da aquisição, é pedido o nome do ficheiro de armazenamento de dados. Dá-se então início à aquisição.

O potenciómetro é colocado no modo de leitura em mv e as janelas de visualização dos resultados são abertas.

Faz-se a leitura do potencial inicial da solução e se o utilizador optou por fazer um incremento inicial de volume de titulante na solução, procede-se ao seu adicionamento. É esperado um tempo de estabelecimento da solução (ou não, dependendo da opção do utilizador), é feita a leitura do potencial da solução, é decidida a estabilização dos valores e é calculado o novo incremento de volume a adicionar à solução, segundo os critério escolhidos. Repete-se todo este procedimento até se atingir uma das condições de decisão de fim de experiência.

Se o utilizador desejar fazer uma pausa na aquisição, activa o comando "Temporariamente" do sub-menu "Parar" do menu da janela da aplicação. Pode em seguida recomeçar no ponto em que tinha deixado a experiência, activando o comando "Recomeçar".

O utilizador pode forçar o fim de experiência em qualquer ponto da aquisição, activando o comando "Fim de Experiência" do sub-menu "Parar". Ao activar este comando a aquisição pára mas o utilizador continua a poder visualizar os dados da reacção agora terminada. Se desejar realizar uma nova experiência activa o comando "Nova Experiência" da barra de menu da janela da aplicação e o processo é reiniciado.

Se o utilizador desejar abandonar a aplicação em qualquer ponto da aquisição pode fazê-lo através do comando "Sair", e todos os valores adquiridos até esse momento ficam armazenados no ficheiro de armazenamento de dados.

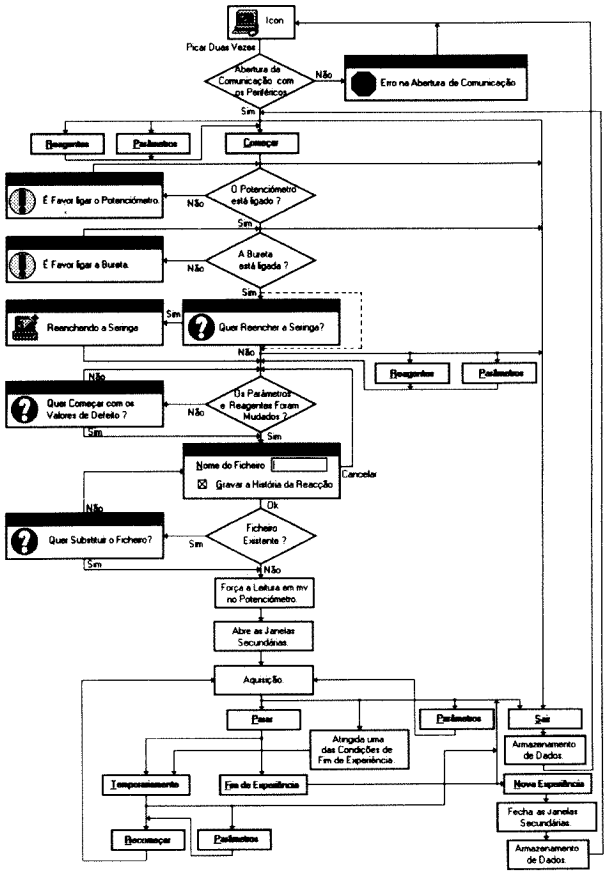


Fig. 10
Diagrama de Funcionamento.

IV. ESTRUTURA DO PROGRAMA

O programa está dividido basicamente em dez grandes módulos, de modo a permitir uma melhor estruturação, e um melhor acesso a zonas do programa onde se pretendam fazer alterações.

Assim, as rotinas de registo das janelas principal e secundárias da aplicação, de abertura e fecho da janela da aplicação, e de processamento das mensagens enviadas a estas janelas costituem o módulo principal. Entre as mensagens a gerir, a mais importante para esta aplicação é a do relógio do sistema, uma vez que as leituras e o seu processamento são feitos em ciclos de relógio definidos. A declaração das variáveis globais também é feita neste módulo.

As rotinas de processamento de mensagens das *Dialog Boxes* "Reagentes - Inicialização", "Parâmetros de Inicialização e Estabilização" e "Armazenamento de Dados" estão em três módulos separados. A leitura de todas as variáveis de entrada e o controlo sobre a introdução de valores absurdos é feito nestes módulos.

Todas as rotinas comuns às *Dialog Boxes*, como as de colocação destas no centro da janela da aplicação e as de controlo da inserção de caracteres não válidos nos campos de edição das mesmas foram colocadas num outro módulo independente.

Para cada janela de visualização de resultados ("Gráfico" e "Medições Efectuadas") existe um módulo com as rotinas de processamento das respectivas mensagens associadas. Nestes módulos estão também as rotinas de escrita nas respectivas janelas, que são usadas quando é enviado pela janela principal uma mensagem de aviso de escrita, depois do processamento dos valores estar feito.

Num outro módulo do programa, estão as rotinas de cálculo de incrementos a adicionar à solução, a sua correspondência em impulsos a transmitir à bureta e cálculo de médias e desvios padrão de potencial.

As rotinas de abertura e inicialização da comunicação com a bureta e com o potenciómetro e de leitura e escrita nas portas de comunicação e as rotinas que tem com função gerir o armazenamento dos valores lidos em disco e em memória constituêm outros dois módulos.

V. COMENTÁRIOS E CONCLUSÕES

O sistema operativo Windows 3.1 da Microsoft, não é o mais indicado para sistemas de tempo real, devido às caracteristicas do seu *multitasking nonpreemptive*. A mensagem do despertador do sistema (*timer*) é a mensagem de menor prioridade, e uma vez chegada ao *message queue* o seu envio pode ser atrasado pela chegada de outras mensagens. Assim um sistema de aquisição de dados que se queria periódico deixa de o ser. E, com o acumular de mensagens do despertador no *message queue* fazem-se por vezes medidas consecutivas, quando se desejava que houvesse um tempo de espera entre elas. Este problema de acumulação de mensagens no *message queue* pode ser evitado, desligando-se o despertador sempre que a mensagem por ele enviada é recebida pela função que faz o seu processamento e tornando a ligá-lo no fim desse mesmo processamento. Porém este método não resolve o problema da não periodicidade das medidas, podendo os atrasos no envio das mensagens do despertador serem minimizados pelo utilizador ao não fazer correr outra aplicação enquanto utiliza o Sistema Integrado de Aquisição de Dados para Potenciometria, no entanto este método vai contra a filosofia do sistema operativo em questão. O método de resolver estes problemas seria o de fazer com que o despertador envia-se a mensagem directamente para uma função encarregada de a processar, sem passar pelo *message queue*.

REFERÊNCIAS

[1] Tim Farrell, Runnoe Connally, "Programming in Windows 3.1", Second Edition, Programming Series, QUE.

[2] Rogerio Saran, "Programando em Windows 3.1", Livros Érica Editora Ltda.

[3] Brian W, Kernighan, Dennis M. Ritchie, "The C Programming Language", Second Edition, Prentice Hall Software Series.

[4] Peter A. Dornell, Philip E. Margolis, "C: A Software Engeneering Approach", Springer - Verlag.

[5] Mark Szamrej, "Custom Toolbars", Windows/DOS Developer's Journal, February 1993, vol. 4, Nº 2, pp. 47-57.

[6] John Caufield, "Dialog Boxes on the Fly", Windows/DOS Developer's Journal, February 1993, vol. 4, Nº 2, pp. 25-38.

[7] John Reynolds, "Real-World Device Control", Windows/DOS Developer's Journal, March 1993, vol. 4, Nº 3, pp. 6-22.

[8] Guy Eddon, "Monitoring Hardware Interrupts in Real Time", Windows/DOS Developer's Journal, February 1993, vol. 4, Nº 3, pp. 23-28.

[9] Microsoft Corporation, "Run-Time Library Reference", Microsoft C/C++ Version 7.0.

[10] Orion Research Incorporated, "Laboratory Products Group Bench Top pH/ISE Meter Instruction Manual".

Projecto, construção e teste de agregados de antenas "microstrip"

Guilherme F. M. da Silva, Carlos A. T. Campos, José Rocha Pereira

Resumo- **Este artigo descreve um estudo básico das antenas impressas do tipo "microstrip". É feita uma descrição breve dos métodos de análise usuais para este tipo de estruturas, sendo feita a análise e o projecto de "patches" rectangulares pelo método da cavidade ressonante.**

Foram projectadas, construidas e testadas "patches" rectangulares e quadradas com o fim de se obter as suas características principais. Dedicou-se especial atenção ao método de adaptação, destas estruturas, a linhas "microstrip" de 50Ω.

Estas "patches" foram depois usadas na construção de agregados a 1.5GHz e a 11.626GHz, apresentando-se os resultados obtidos.

Abstract- **This paper describes a first approach on the microstrip antennas design. Several theoretical analysis technics are summarized. The analysis and design of standard rectangular patches is done using the cavity model.**

Some square and rectangular patches have been designed and measured in order to evaluate their main characteristics. Special attention was paid to the problem of matching these patches to 50Ω microstrip lines.

Finally arrays of rectangular patches have been design and measured at 1.5GHz and 11.626GHz. Their results are presented.

I. INTRODUÇÃO*

A configuração básica de uma antena "microstrip" consiste numa "patch" metálica, de espessura muito fina, ($t<<\lambda$), impressa num substracto fino, com plano de massa. A espessura, do substracto é, apenas, uma pequena fracção do comprimento de onda, ($h<<\lambda$) [2].

A "patch", radiante, pode ser quadrada, rectangular, circular, elíptica, ou pode assumir qualquer outra forma. Contudo, as formas mais utilizadas são a quadrangular, rectangular e a circular, por serem as formas que permitem um estudo analítico mais simples e por serem mais fáceis de projectar [1].

A linha de alimentação é frequentemente, também, uma linha do tipo "strip", geralmente de menor largura. Os cabos coaxiais também podem ser utilizados, como linhas de alimentação e, neste caso, liga-se o condutor interior ao elemento radiante. Pode-se obter polarização linear ou circular com este tipo de antenas. Agregados de elementos "microstrip", com apenas uma ou mais alimentações, podem, também, ser utilizados para obter maiores directividades [1].

Devido à espessura da "microstrip" ser muito pequena, as ondas geradas no interior do substrato dielétrico (entre a "patch" e o plano de massa), sofrem reflexões consideráveis quando chegam às extremidades da "patch". Apenas uma pequena fração de energia incidente é radiada: a antena é pouco eficiente e comporta-se mais como uma cavidade [1].

Como as antenas "microstrip" têm um perfil muito baixo, podem ser fabricadas utilizando as técnicas de circuito impresso, podendo por isso ser fabricadas, facilmente, quer nas universidades quer noutros laboratórios de investigação e, potencialmente, a baixo custo. Outras vantagens incluem a fácil fabricação de agregados lineares ou planares e fácil integração com circuitos integrados de microondas [2].

Desde que a configuração original foi proposta, dezenas de formas da "patch", técnicas de alimentação e configurações de substrato bem como geometrias de agregados foram sendo desenvolvidas por investigadores de todo o mundo excedendo, portanto, as antenas "microstrip", em termos de projecto, qualquer outro tipo de antena [2].

Uma larga extensão do desenvolvimento foi conduzida por sistemas que impunham determinadas condições às antenas, tais como: baixo perfil, peso reduzido, fácil integração em agregados ou em circuitos integrados de microondas, ou diversidade de polarização e baixo preço [2].

TABELA I
ALGUMAS APLICAÇÕES DAS ANTENAS "MICROSTRIP"

PLATAFORMA	SISTEMAS
Aviação	Radar, comunicações, navegação, altímetro, sistemas de aterragem
Mísseis	Radar, Telemetria
Satélites	Comunicação, televisão, radiómetros e radares sensíveis controlados remotamente
Navios	Comunicações, radar e navegação
Veículos terrestres	Telemóvel via satélite, rádio móvel
Outros	Sistemas biomédicos, alarmes contra intrusos

As desvantagens da antena "microstrip" original incluem largura de banda estreita, radiação espúria pela alimentação, baixa pureza de polarização, capacidade limitada de potência e problemas de tolerância. Muito do trabalho foi desenvolvido no sentido de minimizar estes problemas, para satisfazer os requisitos, cada vez mais severos, dos sistemas. Este esforço envolveu o desenvolvimento de novas configurações e o desenvolvimento de modelos analíticos mais versáteis e precisos para a compreensão das limitações que estão

* Trabalho realizado no âmbito da disciplina de projecto

inerentes a este tipo de antenas bem como ao seu projecto e optimização [2].

II. MODELO DE RADIAÇÃO DE UMA "PATCH" DE FORMA RECTANGULAR

Na fig. 1(a) está representada uma "patch" rectangular alimentada por linha "microstrip". A estrutura do campo, no interior do substracto, entre a "patch" e o plano de massa é a indicada na fig. 1(b). O campo sofre uma inversão de fase, ao longo do comprimento, L, mas é, aproximadamente, uniforme ao longo da sua largura, W.

A "patch" pode ser considerada como sendo constituida por duas "slots", radiantes, cada uma de comprimento W e espessura, aproximadamente, igual à espessura do substracto, h, nas extremidades perpendiculares à linha de alimentação. As "slots" estão separadas por uma linha de transmissão, coplanar, de baixa impedância, de comprimento L, que actua como transformador. O comprimento desta linha de transmissão é, aproximadamente, metade do comprimento de onda, no substracto, para que as duas "slots" formem um agregado de dois elementos em fase. O campo eléctrico radiado é, portanto, máximo segundo a normal à "patch" [1].

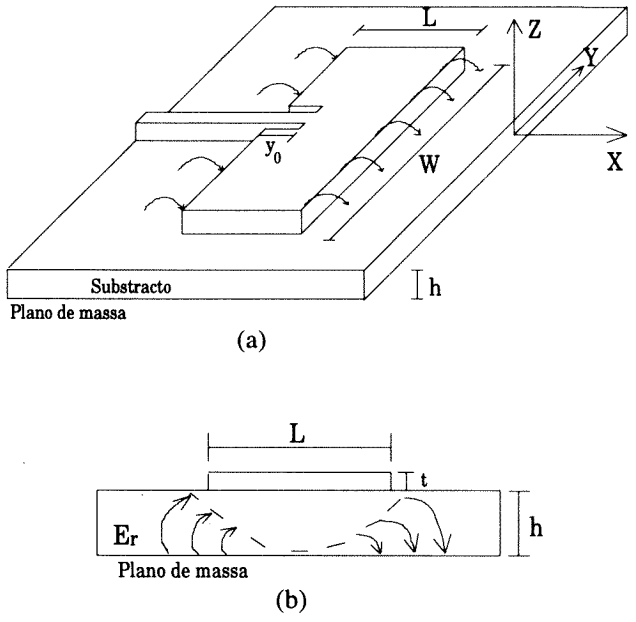


Fig. 1. (a) Antena rectangular alimentada por linha "microstrip". (b) Antena vista de perfil.

Na fig. 2 está representada a "slot" e o respectivo sistema de coordenadas.

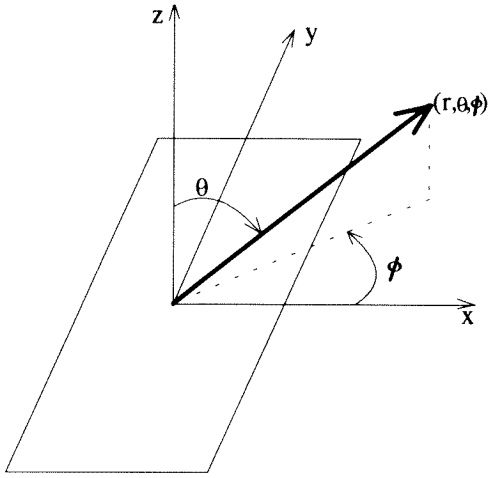


Fig. 2. Sistema de coordenadas, para a abertura.

Os campos radiados, pela "patch", podem ser determinados, se considerarmos a antena como sendo um agregado de duas aberturas (as "slots"). O campo eléctrico na abertura, de cada "slot", é orientado segundo o eixo dos xx e é constante ao longo da abertura e dado pela expressão [1],

$$\mathrm{Ea} = \hat{a}_x E_0 \begin{cases} -h/2 \le x' \le h/2 \\ -w/2 \le y' \le w/2 \end{cases} \quad (1)$$

Pode-se mostrar, utilizando o principio da equivalência, que cada "slot" irá radiar os seguintes campos:

$$E_r = 0$$

$$E_\theta = j\frac{ke^{-jkr}}{4\pi r}\cos(\phi)(1+\cos(\theta))E_0 hW \cdot \frac{\sin\left(\pi\frac{h}{\lambda_0}\sin(\theta)\cos(\phi)\right)}{\pi\frac{h}{\lambda_0}\sin(\theta)\cos(\phi)} \frac{\sin\left(\pi\frac{W}{\lambda_0}\sin(\theta)\sin(\phi)\right)}{\pi\frac{W}{\lambda_0}\sin(\theta)\sin(\phi)} \quad (2)$$

$$E_\phi = -j\frac{ke^{-jkr}}{4\pi r}\sin(\phi)(1+\cos(\theta))E_0 hW \cdot \frac{\sin\left(\pi\frac{h}{\lambda_0}\sin(\theta)\cos(\phi)\right)}{\pi\frac{h}{\lambda_0}\sin(\theta)\cos(\phi)} \frac{\sin\left(\pi\frac{W}{\lambda_0}\sin(\theta)\sin(\phi)\right)}{\pi\frac{W}{\lambda_0}\sin(\theta)\sin(\phi)} \quad (3)$$

Na "patch", estes campos vêm multiplicados por dois factores de agregado:

$$2\cos\left(\pi\frac{L}{\lambda_0}\sin(\theta)\cos(\phi)\right) \quad (4)$$

que é o factor de agregado para as duas "slots" e

$$2\sin\left(2\pi\frac{h}{\lambda_0}\cos(\theta)\right) \quad (5)$$

que é o factor de agregado da "patch" com a sua imagem.

III. ANÁLISE E PROJECTO DE UMA ANTENA RECTANGULAR

Os métodos de análise das antenas "microstrip" são numerosos e variados e podem-se dividir em duas classes; os métodos analíticos e os métodos numéricos.

Os métodos analíticos são métodos muito simples, aproximados, e fáceis de usar, que permitem uma boa compreensão dos mecanismos físicos do funcionamento das antenas. De entre esta classe destacam-se o modelo da linha de transmissão e o da cavidade ressonante [2].

Contrastando com a simplicidade dos modelos analíticos, existem os métodos numéricos, que recorrem ao cálculo numérico intensivo em computador. Estes métodos têm a vantagem de proporcionarem uma análise rigorosa das antenas. São, no entanto, baseados em formulações teóricas de grande complexidade, que não permitem a compreensão dos mecanismos de radiação proporcionada pelos modelos analíticos simples, e são, normalmente, muito exigentes em termos de computação.

Devido, essencialmente, à grande complexidade computacional dos métodos numéricos e à boa explicação, intuitiva, do modo de funcionamento, deste tipo de antenas, proporcionada pelos métodos analíticos, decidiu-se que nos iriamos apoiar no modelo da cavidade ressonante [3]. Neste modelo a antena é assimilada a uma cavidade formada por condutores eléctricos perfeitos (superfície superior e respectiva projecção no plano de massa) e paredes laterais (ao longo dos bordos da antena), consideradas como condutoras magnéticas perfeitas. Os campos no interior da cavidade são obtidos pela sobreposição dos modos próprios da superfície ressonante que constitui o condutor superior da antena. Não se consideram, portanto, variações ao longo da direcção perpendicular ao plano de massa, o que resulta na principal limitação deste modelo, que assim também só é utilizável em estruturas onde o substracto é muito fino quando comparado com as dimensões da "patch". A radiação é contabilizada através de de um aumento fictício da tangente do ângulo de perdas do substracto dieléctrico [3].

Segundo este modelo, o projecto de uma antena "microstrip", rectangular, inicia-se reconhecendo que o modo TM_{10} é excitado fazendo com que a distância entre "slots" seja ligeiramente menor que meio comprimento de onda, no substracto:

$$\lambda_s = \lambda_0 / \sqrt{\varepsilon_r} \quad (6)$$

portanto, fazendo com que as duas extremidades radiantes, paralelas, de comprimento W, se comportem como um agregado de dois elementos, em fase. O comprimento W é, tipicamente, $\lambda_0/2$. Se não ocorresse nenhum efeito, nas extremidades radiantes, a frequência de ressonância seria dada por:

$$f_{ro} = \frac{c}{2L\sqrt{\varepsilon_r}} \quad (7)$$

Contudo, na prática, o efeito da capacidade, associada ás extremidades radiantes, provoca um ligeiro aumento da distância entre slots, L, e portanto a frequência de ressonância é ligeiramente menor que f_{ro} por um factor de redução, q [3]. Portanto

$$f_r = q\frac{c}{L\sqrt{\varepsilon_r}} \quad (8)$$

Quanto maior for a espessura do substracto maior será a distância efectiva entre as "slots", devido ao incremento da capacidade, que lhes está associada, decrescendo, portanto, a frequência de ressonância de um modo aproximadamente linear com o incremento da espessura do substracto. A susceptância da parede radiante e, portanto, a capacidade nas extremidades radiantes, é aproximadamente proporcional ao comprimento das "slots", W [3]. Portanto, para uma dada espessura do substracto, um incremento no comprimento das "slots" provoca um decréscimo da frequência de ressonância.

A resistência de ressonância, pode ser obtida, em função do factor de qualidade e da capacidade da "patch" [3]. Esta análise revela que a resistência de ressonância também é função da espessura do substracto e do ponto de alimentação, y_0/L. Verifica-se que a resistência de ressonância, de uma "patch", com alimentação num ponto situado no seu interior, é dada por:

$$R_{rad} = R^e_{rad}\cos^2(\pi y_o/L) \quad (9)$$

isto é colocar o ponto de alimentação, no interior da "patch", provoca o decréscimo na resistência (R^e_{rad} é a resistência de radiação na extremidade da "patch"). A resistência de ressonância, também, pode ser diminuida pelo aumento do comprimento das "slots", permanecendo a distância entre "slots" aproximadamente igual a meio comprimento de onda no substracto. Contudo, não são aconselháveis razões W/L maiores que 2, porque a eficiência da abertura, de uma "patch", começa a diminuir, acentuadamente.

Com base neste formalismo elaborou-se um pequeno programa que permite dimensionar "patches" rectangulares bem como linhas de transmissão "microstrip".

IV. "PATCH" RECTANGULAR

Com base no software mensionado, dimencionou-se uma "patch" rectangular para a frequência de 1.5GHz, num substracto com 1.5mm de espessura e $\varepsilon_r = 4.47$.

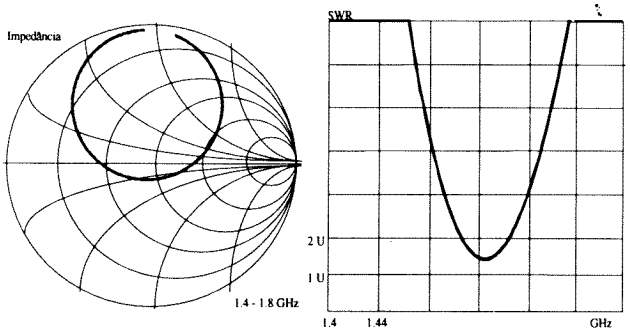
A. *Dimensões*

-Distância entre "slots": 47mm;
-Comprimento das "slots": 70mm;

-Impedância de entrada: 353Ω

B. Resultados do Network analyzer

Para verificar as características eléctricas da "patch", a esta frequência, utilizou-se um Network analyser sendo de realçar as seguintes características:



- Frequência de ressonância: 1.506GHz;
- Impedância de entrada: 107.02Ω

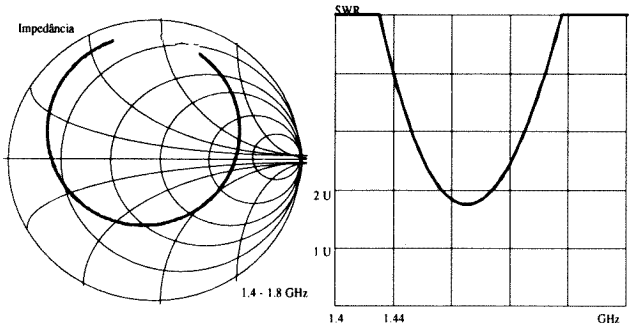
V. "PATCH" QUADRADA

Repetiu-se o procedimento para uma "patch" quadrada, com os seguintes resultados:

A. *Dimensões*

-Distância entre "slots": 47mm;
-Comprimento das "slots": 47mm;
-Impedância de entrada: 734Ω

B. Resultados do Network analyzer



- Frequência de ressonância: 1.465GHz;
- Impedância de entrada: 182.6Ω

C. Conclusões

Verificou-se que as "patches" são ressonantes a uma frequência muito próxima da frequência pretendida. Contudo, a impedância de entrada, de ambas as antenas, diferiu bastante do valor esperado.

Conhecida a impedância de entrada, de ambas as antenas, projectaram-se "patches" adaptadas a linhas de transmissão ,"microstrip", de 50 Ω por dois processos: reentrância e transformadores de $\lambda/4$.

VI. ADAPTAÇÃO POR REENTRÂNCIA NA "PATCH"

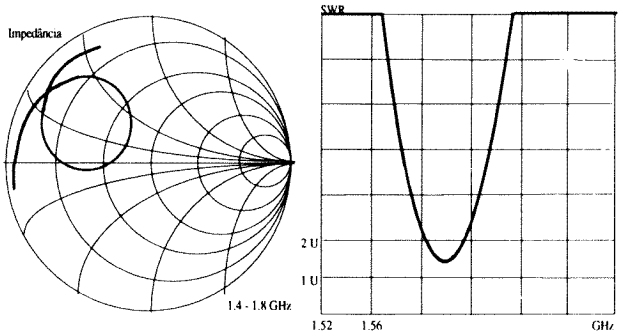
Usando o mesmo software, projectaram-se duas "patches", uma rectangular e uma quadrada, com reentrância, para as adaptar a uma linha de transmissão de 50Ω

A. "Patch" rectangular

A "patch" rectangular tinha as seguintes dimensões:

-Distância entre "slots": 43.8mm;
-Comprimento das "slots": 70mm;
-Reentrancia: 12mm;
-Largura da linha de alimentação: 2.83mm.

Esta "patch" apresentava as seguintes caracteristicas eléctricas:



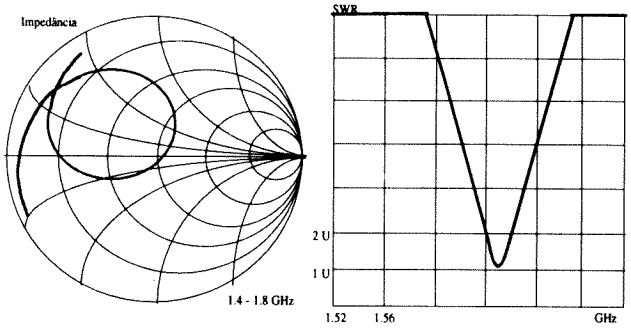
-Frequência de ressonância: 1.6256GHz;
-Erro relativo de frequência: 8.37%;
-Impedância de entrada: 31.76Ω;
-SWR: 1.57;
-Largura de banda:
i) Critério do dobro do SWRmin: 2.67%;
ii) Critério de metade da $P_{t_{max}}$: 3.1%.

B. "Patch" quadrada

A "patch" quadrada tinha as seguintes dimensões:

-Distância entre "slots": 43.8mm;
-Comprimento das "slots": 43.8mm;
-Reentrancia: 13.5mm;
-Largura da linha de alimentação: 2.83mm.

Esta "patch" apresentava as seguintes caracteristicas eléctricas:



-Frequência de ressonância: 1.649GHz;
-Erro relativo de frequência: 9.9%;
-Impedância de entrada: 54.46Ω;
-SWR: 1.09;
-Largura de banda:
i) Critério do dobro do SWRmin: 2.18%;
ii) Critério de metade da $P_{t_{max}}$: 2.68%.

C. *Conclusões*

Confirmou-se que, efectivamente, a largura de banda aumenta com o aumento do comprimento das "slots", embora este aumento não seja muito significativo e que a impedância de entrada diminui fazendo a linha de alimentação reentrar na "patch".

VII. ADAPTAÇÃO POR TRANSFORMADORES DE $\lambda/4$

Pretendeu-se dimensionar uma "patch" rectangular, que estivesse adaptada a uma linha de transmissão, "microstrip", de 50Ω utilizando para o efeito um transformador de $\lambda/4$, que minimizasse as reflexões e, portanto, maximizasse a energia radiada ou recebida.

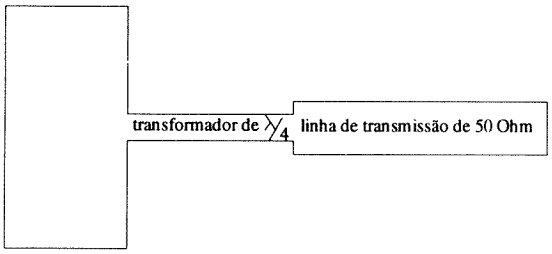


Fig. 3. "Patch" rectângular com adaptação, por transformador de λ/4, a uma linha de transmissão de 50Ω.

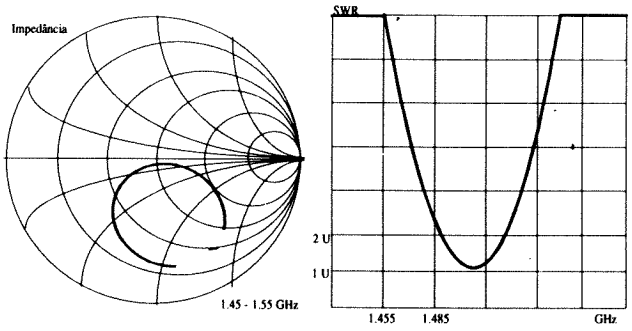
A. Dimensões

O software elaborado forneceu as seguintes dimensões:

-Distância entre "slots": 46.9mm;
-Comprimento das "slots": 70mm;
-Impedância de entrada: 107Ω;
-Transformador de λ/4: -Impedancia: 73Ω;
-Comprimento: 27.96mm;
-Largura: 1.40mm;
-Largura da linha de transmissão de 50Ω: 2.83mm.

B. Resultados do Network analyzer

Esta "patch" apresentava as seguintes características eléctricas:



-Frequência de ressonância: 1.509GHz;
-Erro relativo de frequência: 0.6%;
-Impedância de entrada: 50.79Ω;
-SWR: 1.09;
-Largura de banda:
i) Critério do dobro do SWRmin: 2.3%;
ii) Critério de metade da $P_{t_{max}}$: 3.3%.

C. Conclusões

Os resultados obtidos são bons pois quer a frequência de ressonância quer a impedância de entrada estão muito próximos dos valores pretendidos. Por isso, utilizou-se esta técnica, de adaptação, no projecto de agregados de antenas.

VIII. ESTUDO DE AGREGADOS DE ANTENAS A 1.5GHz

O passo seguinte foi arranjar uma maneira de agregar várias "patches", de modo a que estas constituissem um agregado de vários elementos, alimentados em fase e que a partir dele se pudesse extrapolar um agregado de maiores dimensões em que cada elemento, deste novo agregado, fosse o agregado anterior e assim sucessivamente. Após pesquisa bibliográfica [4] e [5] e algumas experiências chegamos ao agregado da fig. 4 no qual, como se pode verificar, os elementos são alimentados em fase. Este agregado foi alimentado por cabo coaxial tendo sido o conector colocado por detrás, no plano de massa, com o condutor central ligado ao centro do agregado.

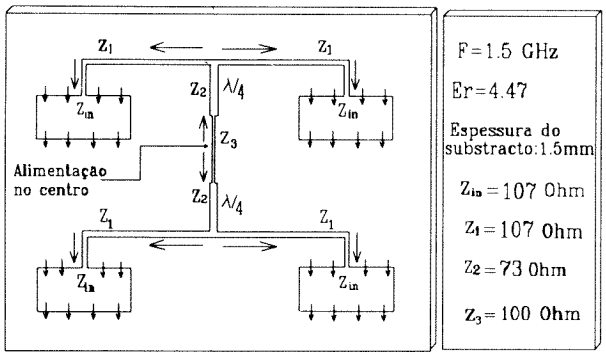


Fig. 4. Agregado de 4 elementos, alimentados em fase, com transformadores de λ/4 de 73Ω para a frequência de 1.5 GHz.

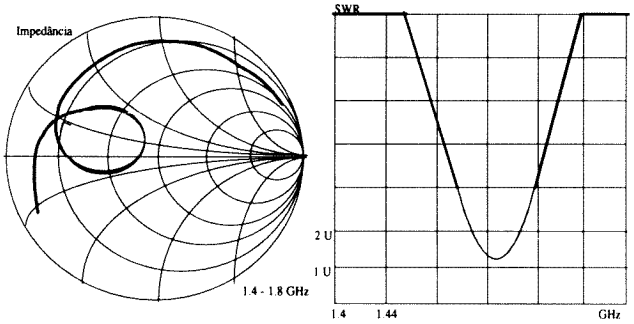
A. Dimensões

Mais uma vez, socorremo-nos do programa para dimencionar as "patches" e os transformadores de $\lambda/4$. Obteve-se os seguintes valores:

-Distância entre "slots": 46.9mm;
-Comprimento das "slots": 70mm;
-Transformador de $\lambda/4$, Z_2: -Impedância: 73Ω;
-Comprimento: 27.96mm;
-Largura: 1.40mm;
-Linha de transmissão, Z_1: -Impedância: 107Ω;
-Largura: 0.537mm;
-Linha de transmissão, Z_3: -Impedância: 100Ω;
-Largura: 0.65mm.

B. *Resultados do Network analyser*

O agregado apresentava as seguintes características eléctricas:



-Frequência de ressonância: 1.523GHz;
-Frequência que minimiza o SWR: 1.5226GHz;
-Erro relativo de frequência: 1.5%;
-$SWR_{F_{res}} = 1.109$;
-$SWR_{\min} = 1.107$;
-Largura de banda:
i) Critério do dobro do $SWR_{\min}$: 2.76%;
ii) Critério de metade da $P_{t_{\max}}$: 3.37%.
-Impedância de entrada: 55.25Ω.

C. Diagrama de radiação teórico

O módulo do campo eléctrico é dado por:

$$|E| = (F.E.)(F.A.)_x^1(F.A.)_x^2(F.A.)_Y(F.A.)_Z \quad (10)$$

sendo

$$(F.E.) = \sqrt{|E_\theta|^2 + |E_\phi|^2} \quad (11)$$

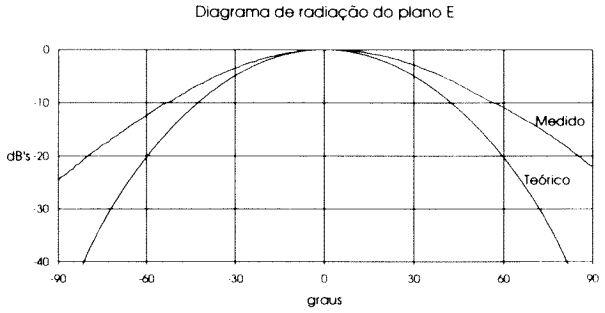
$$(F.A.)_x^{(1)} = 2\cos\left(\frac{\pi}{\lambda_0} L \sin(\theta)\cos(\phi)\right) \quad (12)$$

$$(F.A.)_x^{(2)} = 2\cos\left(\frac{\pi}{\lambda_0} D_2 \sin(\theta)\cos(\phi)\right) \quad (13)$$

$$(F.A.)_y = 2\cos\left(\frac{\pi}{\lambda_0} D_1 \sin(\theta)\sin(\phi)\right) \quad (14)$$

$$(F.A.)_z = 2\sin\left(\frac{2\pi}{\lambda} h\cos\theta\right) \quad (15)$$

No plano E $\phi = 0°$ e no plano H $\phi = 90°$



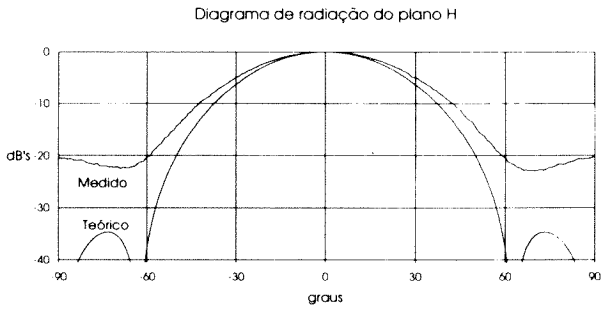


TABELA II
LARGURA DE FEIXE A MEIA POTÊNCIA

	Teórico	Medido
Plano E	47°	59°
Plano H	42°	48°

D. Ganho

Para se determinar o ganho, deste agregado, utilizou-se o método das duas antenas. Seguindo este método é necessário ter uma antena padrão, da qual se conheça o ganho, ou então ter duas antenas exactamente iguais.

Como não tinhamos nenhuma antena padrão, para esta frequência, fez-se um duplicado do agregado.

Este método baseia-se na fórmula de Friis, por isso, é necessário obedecer à condição de campo distante,

$$r \geq \frac{2D^2}{\lambda_0} \tag{16}$$

em que r é a distância entre as antenas e D é a maior dimensão das antenas. Temos, então que

$$P_r = P_{in}\left(\frac{\lambda}{4\pi r}\right)^2 G_A G_B \tag{17}$$

Como as antenas são iguais, $G_A = G_B = G$, pelo que

$$P_r = P_{in}\left(\frac{\lambda}{4\pi r}\right)^2 G^2 \tag{18}$$

$$G = \sqrt{\frac{P_r}{P_{in}}}\left(\frac{4\pi r}{\lambda}\right) \tag{19}$$

O ganho em dB's é $G_{dB} = 20\log(G)$. Portanto, para se determinar o ganho, do agregado, tem-se que conhecer a distância entre as antenas, saber qual a potência injectada na antena emissora, P_{in},, e medir a potência recebida pela antena receptora, P_r.

As medições podem ser feitas em espaço livre ou então simular essa condição utilizando, para o efeito, uma câmera anecoica. Utilizou-se a câmera anecoica.

$$G_{dB} \approx [11.5, 12.5] dB$$

E. Conclusões

Os diagramas de radiação medidos diferem, um pouco, dos teóricos. Esta diferença deve-se, essencialmente, ao facto da dedução, utilizada para obter o diagrama de radiação teórico, ser muito simplista, mas de fácil compreensão, que modela, apenas de forma grosseira, o mecanismo de radiação de uma "patch".

Os resultados obtidos são aceitáveis pois quer a frequência de ressonância quer a impedância de entrada e a largura de banda, estão dentro dos valores esperados. Por isso, utilizou-se esta geometria no estudo de agregados a uma frequência mais elevada (F=11.626GHz).

IX. ESTUDO DE AGREGADOS À FREQUÊNCIA DE 11.626 GHz

Projectou-se um agregado de 4 elementos para a frequência de 11.626GHz. O substracto utilizado tem 0.254mm de espessura e a sua constante dieléctrica é 2.33. Como a impedância de entrada, de cada "patch", era de, aproximadamente, 130Ω foi necessário aplicar um transformador de λ/4, à sua entrada, pois não é possível implementar linhas de transmissão com essa impedância característica, neste substracto. O conector não foi colocado no plano de massa porque o substracto utilizado é flexivel não permitindo um contacto perfeito entre o conector e o plano de massa, sendo necessário, para o efeito, utilizar uma placa de latão ou cobre o que dificultava, em muito, a operação. Por isso resolveu-se colocar o conector como mostra a fig. 6.

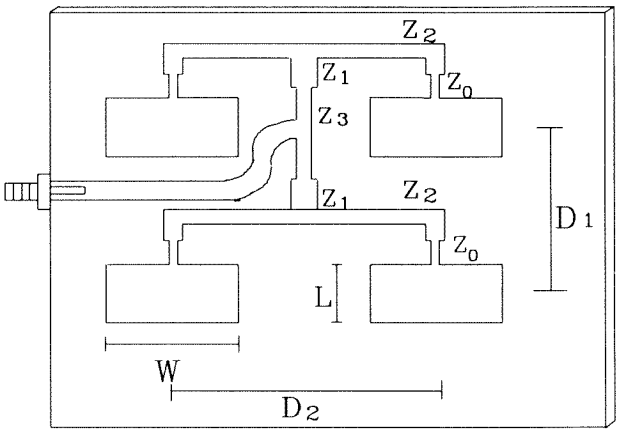


Fig. 6. Agregado de 4 elementos com 4 transformadores de λ/4 para a frequência de 11.626 GHz.

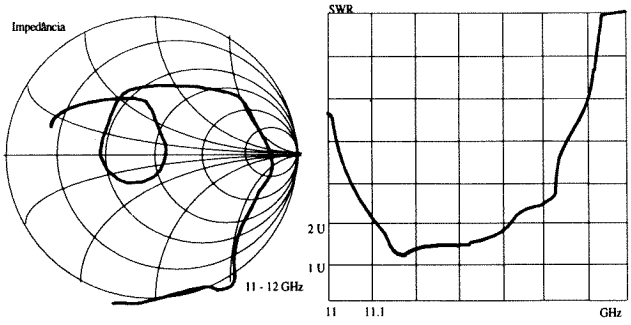
A. Dimensões

O software elaborado forneceu as seguintes dimensões:

-Distância entre "slots":8.2 mm;
-Comprimento das "slots": 15 mm;
-Transformador de λ/4, Z_0:-Impedância: 99Ω;
-Comprimento: 4.76 mm;
-Largura : 0.22 mm;
-Transformador de λ/4, Z_1:-Impedância: 61.4Ω;
-Comprimento: 4.65 mm;
-Largura: 0.546 mm;
-Linha de transmissão de 50 Ω: -Largura: 0.755 mm;
-Linha de transmissão Z_2:-Impedância:75.4 Ω;
-Largura: 0.38mm.
-Linha de transmissão Z_3: -Impedância:100 Ω;
-Largura: 0.217mm.

B.Resultados do Network analyzer

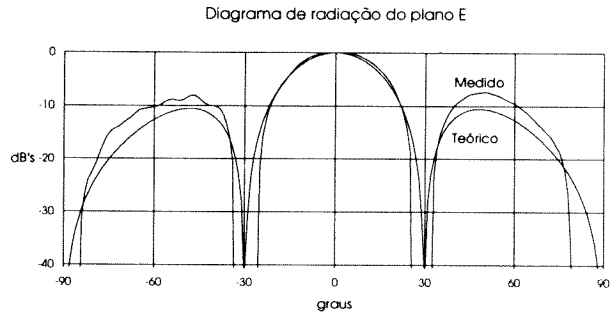
O agregado apresentava as seguintes características eléctricas:

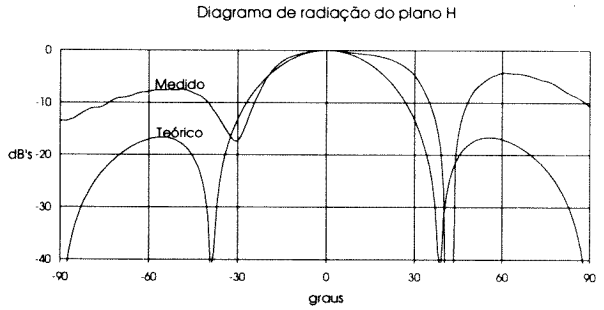


-Frequência: -Frequência de ressonância: 11.186 GHz;
-Erro relativo de frequência: 3.8%;
-SWR_{min}: 1.295;
-Largura de banda:

i) Critério do dobro do SWR_{min}: 3.66%;
-Impedância de entrada: 67.4 Ω.

C. Diagrama de radiação





As diferenças entre os diagramas de radiação teórico e medido devem-se, essencialmente, aos factos já mencionados na secção VIII. E

As assimetrias, no diagrama de radiação medido, devem-se, muito provavelmente, a imperfeições nas linhas de transmissão que alimentam as "patches".

D. Conclusões

O erro, na frequência de ressonância e na impedância de entrada, deve-se, fundamentalmente, a não termos conseguido testar apenas uma "patch", para determinarmos, com exactidão, o valor da impedância de entrada e do factor de redução e, à dificuldade em passar, para o substracto, o agregado com as dimensões desejadas. A esta frequência, um erro de uma ou duas décimas de milimetro provoca alterações significativas nos resultados.

REFERÊNCIAS

[1] Constantine A. Balanis, *Antenna theory: Analysis and design*, Harper & Row, 1982.

[2] David M. Pozar, "Microstrip Antennas", *Proceedings of the IEEE*, vol. 80, No. 1, January 1992.

[3] Keith R. Carver and James W. Mink, "Microstrip Antenna technology", *IEEE Transactions on Antennas and Propagation*, vol. AP-29, No.1, January 1981.

[4] Ely Levine, Gabi Malamud, Shmuel Shtrikman e David Treves, "A Study of Microstrip Array Antennas with the feed Network", *IEEE Transactions on Antennas and Propagation*, vol. AP-37, No.4, April 1989.

[5] J. Ashkenazy P. Perlmutter e David Treves, "A Modular approach for the Design of Microstrip Array Antennas", *IEEE Transactions on Antennas and Propagation*, vol. AP-31, No. 1, January 1983.

Placa Quadriprocessador Baseada em Transputers

José C. Fonseca , Paulo J. Gonçalves
António R. Borges , Artur C. Pereira , Joaquim A. Martins , Joaquim S. Pinto

Resumo - **Neste artigo, é descrito um módulo multiprocessador baseado em transputers. Tem quatro processadores, cada qual com memória local, memória partilhada e interfaces a *PC-AT* e *Ethernet*. A proposta principal é a de construir um bloco (nó) com uma arquitectura *message-passing* de modo a evitar a interligação em malha *2-D*. Foram tomados cuidados especiais para minimizar o número de chips envolvidos, bem como usadas as características especiais apresentadas pelo transputer no que diz respeito às comunicações interprocessador e concentrada toda a lógica de controlo num *ASIC* multifuncional. A especificação da placa, a análise racional do seu desenho e *pin-out* do *ASIC* estão apresentados e descritos em grande detalhe.**

Abstract - **In this paper, a transputer-based multiprocessing module is described. It has four processors, each with local memory, a shared memory and interfaces for PC-AT and Ethernet. The main purpose is to use it as a building block (node) for a message-passing architecture saving a 2-D mesh interconnection. Special care was taken to minimize chip count, both by using the special features presented by the transputer family for interprocessor communication and by concentrating all controlling logic in a multipurpose ASIC. The board specification, the rationale of its design and the ASIC pin-out are presented and described in full detail.**

I. INTRODUÇÃO*

Hoje em dia são cada vez mais comuns as aplicações para computadores e, ao mesmo tempo, mais ambiciosas devido aos progressos da ciência da computação, necessitando que se desenvolvam computadores cada vez mais poderosos. Algumas das aplicações que estão em franca expansão são as denominadas aplicações computacionais intensivas como sejam o processamento de imagem, síntese de imagens, realidade virtual, imagem médica, visão por computador, resolução de modelos dinâmicos, física da atmosfera, aerodinamismo, astronomia, controlo industrial, etc., nas quais a intensidade de processamento pode atingir valores muito elevados. Esta capacidade de processamento só pode ser conseguida usando processadores tecnologicamente muito avançados ou então arquitecturas que recorram ao processamento paralelo usando vários processadores, como se mostra na figura 1. Desta última forma obtêm-se resultados semelhantes aos da a primeira usando-se componentes mais standardizados tornando-se muito mais económica.

Para cobrir especialmente esta área de arquitectura, em 1985 o fabricante de *LSI InMos* (Reino Unido) colocou no mercado um *CPU* cuja arquitectura se pode considerar inovadora, à qual deu o nome *transputer*. O nome escolhido pretende realçar as suas características principais: *TRANSport* e *comPUTER*. O *transputer* é um circuito integrado *VLSI* com processador, memória e *links* de comunicação para ligações directas com outros *transputers* e preparado para multiprocessamento.

O presente trabalho descreve o projecto de uma placa multiprocessador baseada em *transputers*, mais precisamente em quatro *IMS T800* da *InMos* [1], placa essa que possui também *interfaces* para *bus PC* [2] e rede *ethernet*. Esta placa está dotada de um sistema de memória partilhada, pois o processo de transferência de dados entre processadores via *links* não é muito eficiente no que diz respeito às taxas de transferência.

No desenvolvimento de uma placa deste tipo deve-se ter em conta as suas dimensões e viabilidade económica. Desta forma e, de modo a reduzir o espaço, o projecto teve como base o desenho de *Aplication Specific Integrated Circuits* (*ASIC*) que englobam tanto quanto possível toda a lógica necessária. Como ferramenta de desenho assistido por computador foi usado o *Cadence* da *European Silicon Structures (ES2)* [3]. O *ASIC* foi desenhado em *semicustom* pois permite projectar sistemas de grande complexidade e dimensão com um tempo de projecto bastante menor do que o *fullcustom*.

A placa poderá vir a constituir um módulo de processamento que terá a capacidade de ligação a outras

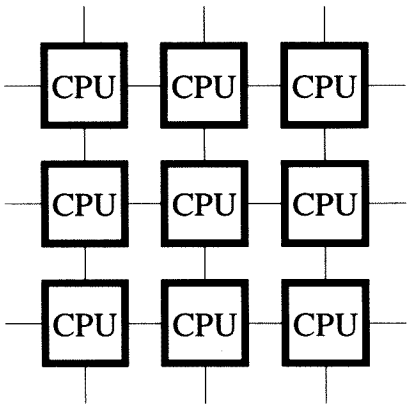


FIGURA 1
TOPOLOGIA CONVENCIONAL DUMA MALHA *2-D* DE PROCESSADORES.

* Trabalho efectuado no âmbito da disciplina de projecto

placas semelhantes pela utilização dos quatro *links* que estarão disponíveis para o exterior. Assim sendo será possível criar uma rede bidimensional de placas que desta forma terão uma capacidade muito superior à de um só elemento.

II. OBJECTIVOS.

Tal como o nome do projecto sugere, "*Placa quadriprocessador baseada em transputers*" ou simplesmente "*Macrotransputer*", o que se pretendeu desenvolver foi uma placa protótipo constituída internamente por quatro processadores do tipo *transputer*, memória e lógica de *interface*. É natural que se pretendam aproveitar as facilidades e potencialidades dos *transputers*, nomeadamente a facilidade de comunicações entre si e a elevada velocidade de processamento. A figura 2 representa a topologia *standard* para uma situação onde se encontram quatro *transputers* interligados, formando um bloco mais complexo e potente.

Como se pode ver a ligação é feita muito facilmente através dos canais série (*links*) que são ligados directamente de um *transputer* a outro, sem necessidade de lógica adicional, ou seja, para ligar um *transputer* a outro só é necessário ter os dois *transputers*.

A placa que se descreve tem funcionalmente os quatro *transputers* ligados da forma acima descrita, mas é diferente desta num aspecto essencial que é a inclusão de uma memória partilhada por todos os processadores em vez de só possuírem memória local a cada um. Esta diferença pode parecer, à primeira vista, pouco significativa, mas rapidamente se torna evidente a grande vantagem desta topologia que é a de tornar acessível, virtualmente ao mesmo tempo e aos quatro processadores, um mesmo bloco de dados aos quais os *transputers* vão poder aceder em paralelo, permitindo uma distribuição de tarefas entre eles e uma mais rápida conclusão das mesmas.

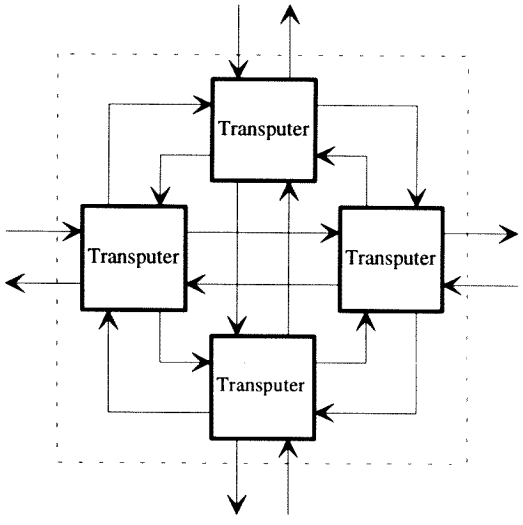


FIGURA 2

NÓ DE QUATRO *TRANSPUTERS* LIGADOS ATRAVÉS DOS SEUS CANAIS SÉRIE

Durante a execução das tarefas, os *transputers*, poderão comunicar entre si para se sincronizarem, trocarem informações e resultados intermédios, ou ainda executarem outras tarefas. Esta topologia, no entanto, tráz também consigo algumas desvantagens e limitações, pois, actualmente, as memórias que permitem acessos *multi-port*, têm capacidades de armazenamento muito reduzidas o que as torna impróprias para este caso. A solução para esta situação foi a de simular estas memórias com base em memórias convencionais e um controlador de acesso (figura 3). Neste caso torna-se imprescindível a existência de um mecanismo que coordene quem vai ser atendido num determinado instante, quem tem de esperar a sua vez, quem se segue na fila de espera, enfim, coordenar todas os problemas e decisões que possam aparecer. Designámos esse controlador de acesso por árbitro e, como é de prever, ele é uma das peças chaves do sistema e da qual depende o seu melhor ou pior desempenho.

Tendo em vista aumentar ainda mais a área de trabalho onde poderá ser usado uma placa deste tipo, a sua funcionalidade e tipo de utilizadores, aparecem ainda mais duas entidades que vão funcionar em paralelo com os *transputers*. A primeira é um *interface* para *bus PC*, de modo que, a partir de uma ferramenta tão divulgada como seja um *PC*, seja possível comunicar e transferir dados de informação ou controlo com a placa. A segunda, não sendo menos importante, é um *interface* para rede *ethernet* aumentando ainda mais a gama de acção que se conseguiria com os *links* e alargando a utilidade da placa.

O que pretendemos desta placa é que ela permita a transferência, manipulação, tratamento e processamento de grandes quantidades de informação de uma forma rápida e sincronizada. Além disso possibilita-se a recepção, transmissão e controlo de dados de diversos periféricos tais como o *bus PC*, a rede *ethernet* e outros *transputers*.

Um ponto interessante de salientar é o facto de que, tal como no caso dos *transputers*, deve ser possível ligar várias placas entre si ou a *transputers* formando uma rede com *performances* muito superiores à de um único elemento. Poder-se-á substituir e com vantagem, por exemplo, redes que tenham sido projectadas para *transputers* por redes baseadas nesta placa multiprocessador.

Pretendia-se que a lógica de controlo deste sistema fosse implementada sob a forma de um *ASIC* de forma que, tanto quanto possível, se evitassem componentes discretos. Este desenvolvimento deveria ser feito com base em captura de esquemáticos usando células de biblioteca (*standard cells* e *macro cells*). Como ferramenta de trabalho para captura destes esquemáticos e respectivas simulações usa-se o pacote de *software* de *CAD* da *Eurochip*, o *Cadence*, para a configuração da *ES2* e tecnologia de 1.0μ [4]. Trata-se de uma ferramenta que é geral no que diz respeito à tecnologia que usa e aos fabricantes e está em fase de divulgação e expansão no nosso país.

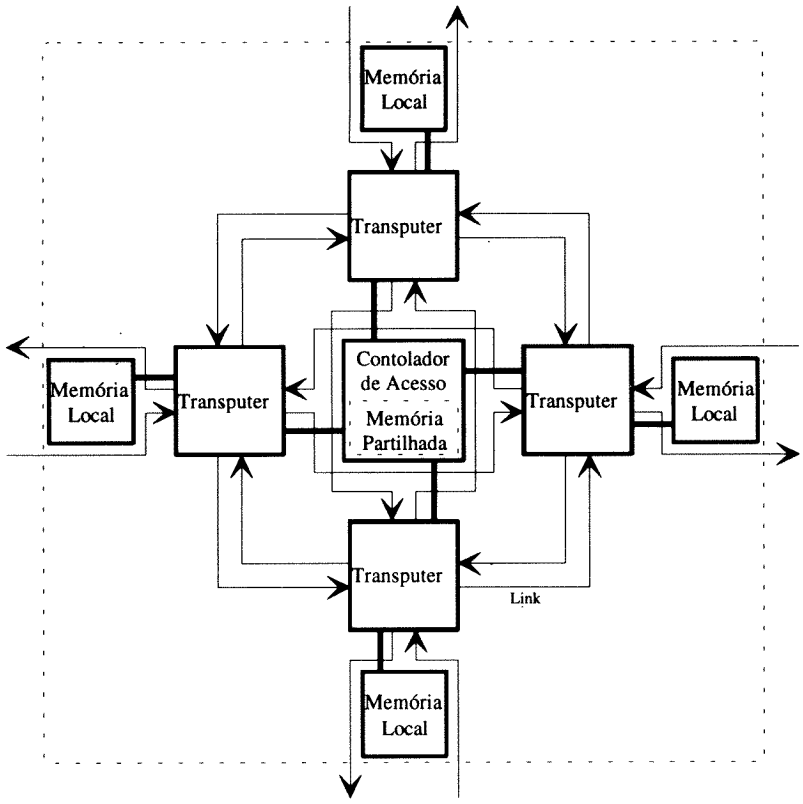


FIGURA 3
REPRESENTAÇÃO DAS ÁREAS DE MEMÓRIA, ÁRBITRO E *LINKS*.

III. ESPECIFICAÇÃO DO SISTEMA

A. Especificação funcional

À partida o sistema é constituído por quatro *transputers*, cada um deles com um *interface* para memória dinâmica local de 4Mbyte ou 16Mbyte e um outro *interface* para um bloco de memória dinâmica partilhada também de 4Mbyte ou 16 Mbyte [5]. O acesso a esta é controlado por um elemento designado por árbitro, tratando com igual prioridade os vários pedidos de acesso seguindo um mecanismo *round-robin*. Estas memórias necessitam de ciclos de *refresh*, pelo que no caso da memória local, é o próprio *transputer* o reponsável por estes, mas no caso da memória partilhada existe uma entidade específica para esse fim. O acesso desta entidade à memória dinâmica partilhada para execussão de ciclos de *refresh* é feito em concorrência com os restantes utilizadores desta memória, pelo que necessita também de ser controlado pelo árbitro. O árbitro, devido à natureza da entidade em questão, concede a permissão de acesso da forma mais rápida, tratando o pedido desta com prioridade sobre os restantes pedidos de acesso.

Este sistema está ainda dotado de um *interface* para *bus PC*, possibilitando o acesso transparente, sob o ponto de vista do utilizador, à memória dinâmica partilhada. Este acesso é feito por endereçamento directo do CPU do PC em questão a uma área de memória de 32Kbyte que corresponde a uma página da memória partilhada. Essa página é seleccionavel através de registos I/O do PC de forma que é possível aceder a qualquer posição da memória partilhada. Os acessos são feitos sempre a oito bits.

O *interface* para rede *ethernet* é feito através de um controlador de rede (*NIC - Network Interface Controller*). Neste caso em concreto, é usado o DP8390C da *National Semiconductor* [6]. Este interface pode ser visto como duas entidades sendo a primeira tomada como passiva. Esta entidade trata-se de um conjunto interno de registos para programação e *status* do *NIC*. Para maior flexibilidade do sistema, estes registos são vistos da mesma forma que a memória partilhada, sendo o acesso a estes possível de ser concretizado por qualquer *transputer* ou mesmo através do *bus PC* e de igual forma controlado pelo árbitro. A segunda entidade é activa e está ligada ao restante sistema através de uma memória *dual-port* [7]. Por parte do *NIC*, esta memória é usada como o seu buffer externo, possibilitando o armazenamento de pacotes. Por parte do restante sistema é vista como mais uma área de memória partilhada.

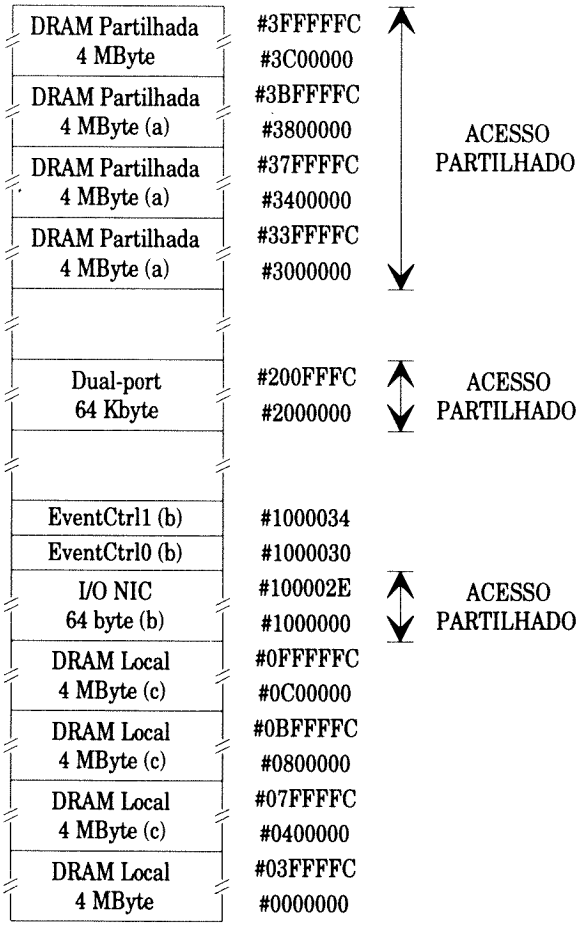
Os dois *interfaces* adicionais, tanto para *bus PC* como para rede *ethernet* através do *NIC*, vieram impor a existência de um processo de comunicação assíncrono entre estes e os *transputers*, já que os *transputers* têm entre si os *links* para esse efeito. Assim, no caso do *transputer* é usado o sinal de *Event* e no caso do *bus PC* um dos sinais *IRQ*. O sinal de *Event* pode ser activado pelo controlador de rede ou pelo *PC* e o sinal *IRQ* pode ser também activado por parte do controlador de rede ou ainda por qualquer um dos quatro *transputers*. Como os sinais de entrada para reconhecimento de interrupções são únicos, para serem reconhecidas individualmente cada uma das fontes, existe um registo (*EventCtrl0*), que através da sua leitura posterior se poderá determinar qual a origem da fonte de interrupção. O sinal de interrupção por parte do controlador de rede é gerado por meio de *hardware* através de um pino que este possui. No caso de sinais de interrupção entre *PC* e *transputers* (e *transputers* e *PC*), estes são gerados pela escrita num registo local (*EventCtrl0*) de uma palavra que deve conter o destino da mesma. Esse registo é individual, não sendo necessário portanto recorrer ao controlador de acesso para o seu uso. Está ainda implementado um outro registo (*EventCtrl1*) para tornar possível mascarar fontes de interrupção não desejadas.

Para aceder aos vários dispositivos existentes na placa, cada *Transputer* pode endereçar ao mapa de memória que

se encontra indicado na figura 4. No caso do *bus PC*, a área da memória local não tem significado, no entanto os restantes dispositivos encontram-se distribuidos da mesma forma. No caso dos *transputers*, o mapeamento é directo, correspondendo a posição #0000000 ao endereço #00000000 do *transputer*. Dado se desprezarem na descodificação os bits de endereço de 26 a 31, a memória dinâmica local pode ser vista no endereço #80000000 que é a base do *transputer*, assim como a memória dinâmica partilhada pode ser vista em #7FC00000 ou #7F000000 caso se trate respectivamente de 4Mbyte ou 16Mbyte indo assim até ao topo de endereçamento deste. Convém referir que quando a memória local é acedida a partir do endereço #80000000, o *transputer* não reconhece a área compreendida entre #80000000 e #80000FFF pois trata-se do espaço de endereçamento à sua memória interna e que é usada entre outras coisas para guardar os registos do processador. Para fazer uso da memória dinâmica local na totalidade, basta aceder por exemplo através do endereço #00000000. O *PC* faz o acesso através de uma área de 32Kbyte, que é reconhecida como memória local do mesmo, nos endereços de #000000 a #007FFF sendo os nove bits de maior peso configuráveis de forma que esta área não entre em conflito com zonas já ocupadas. Esta área equivale a uma página da área ocupada pelos dispositivos existentes na placa podendo ser definida através de um registo.

A nível funcional, o diagrama de blocos que se propõe é o indicado na figura 5. A memória dinâmica local está organizada em quatro bancos de 1Mbyte ou 4Mbyte, o que perfaz 4Mbyte ou 16Mbyte respectivamente. É a memória que cada transputer tem como sendo particular e o seu acesso é feito sem necessidade de inserção de ciclos de espera nem arbitragem. O *interface* para memória local contém a lógica necessária para a ligação da memória dinâmica local ao *transputer*. Contém uma unidade de registo e multiplexagem de endereços capaz de gerar linhas e colunas. De todos os restantes sinais destinados ao controlo da *DRAM*, apenas o sinal de *CAS* é tratado, com o fim de selecção da referida memória entre os restantes dispositivos. O *interface* para acesso partilhado possui duas funções. Uma delas é a de fazer o tratamento referente a pedidos de *Event* ou *IRQ*. Para isso contém registos que podem ser acedidos por parte do *transputer* ou *PC* a este *interface* associado, tanto para leitura como escrita. Estes registos contêm vários *bits*, estando cada um deles associado a um significado, caso se trate de uma escrita ou leitura respectivamente. No caso de um dos registos (*EventCtrl0*) o processo de escrita possibilita gerar remotamente um pedido de interrupção, enquanto a leitura é usada para verificar o *status* das interrupções chegadas. O outro registo (*EventCtrl1*) serve para mascarar individualmente cada entrada de pedido de interrupção.

A segunda função é a de servir de porto para acesso de escrita e leitura nos recursos partilhados, que são a memória dinâmica partilhada, memória dual-port partilhada e espaço de endereçamento *I/O* do *NIC*. Para isso existe lógica que permite guardar e gerar endereços de forma simples ou multiplexada (endereço normal ou por linhas e colunas) e gerar sinais de controlo de uma forma temporal e autónoma, de forma a tornar possível a compatibilidade dos vários tipos de dispositivos partilhados com o *transputer* ou o *bus PC*, sendo esta última tarefa realizada com ajuda de máquinas de estados. Estas máquinas existem principalmente pelo facto de ser necessário garantir uma determinada evolução temporal dos sinais de controlo quando o acesso é desencadeado. Incluído no porto para escrita e leitura, está o sistema que gera o pedido de acesso. O pedido é activado quando se detecta que o dispositivo (*transputer* ou *PC*) pretende realizar um acesso a um endereço referente a um recurso partilhado. Este pedido de acesso destina-se ao controlador, designado por árbitro, que por sua vez, enviará um sinal de confirmação no momento que for possível esse mesmo acesso. Este sistema está equipado



a) Só para memória dinâmica partilhada de 16Mbyte.
b) Apenas existem os 8 bits menos significativos.
c) Só para memória dinâmica local de 16Mbyte.

FIGURA 4
MAPA DE MEMÓRIA.

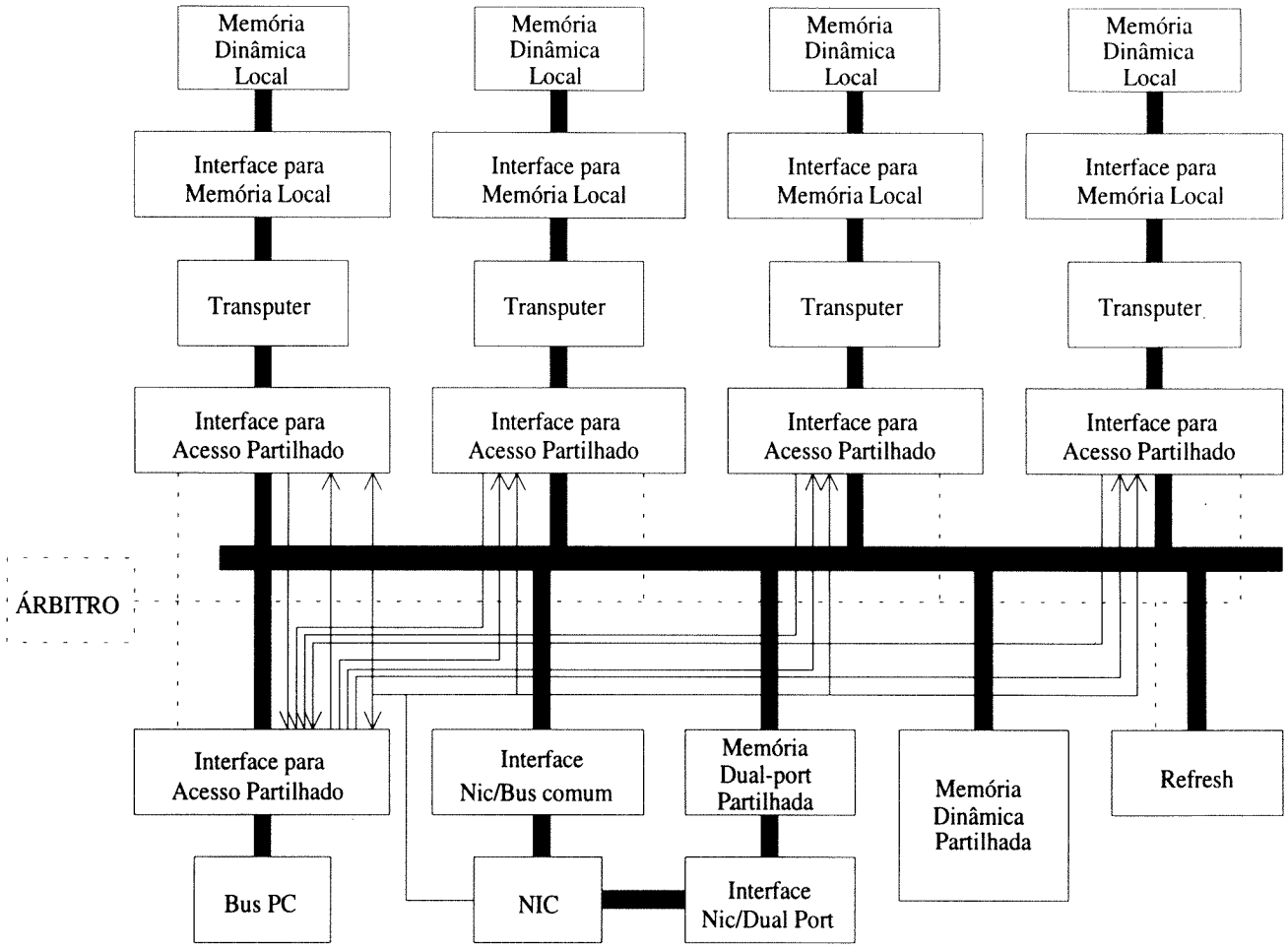


FIGURA 5
DIAGRAMA DE BLOCOS SEM OS *LINKS*.

com a capacidade de tratamento a nível de *time-out*, de forma que a não atribuição de confirmação de acesso a um recurso partilhado não provoca a falha do mesmo dispositivo que realizou o pedido. A falha neste caso está no facto dos *transputers* não poderem estar indeterminadamente em estado de *wait*, pois são responsáveis pelo refrescamento das suas memórias locais.O *Interface NIC / bus* comum permite isolar o *bus* do *NIC* do *bus* comum aos recursos partilhados. Isto deve-se ao facto do *bus* que o *NIC* usa para funcionamento autónomo através dos seus *DMA's* para transferências com a sua memória externa (*dual-port*) ser o mesmo por onde se pode aceder aos registos internos deste controlador. Desta forma, quando o *NIC* se encontra em funcionamento autónomo, não existe necessidade de recorrer à arbitragem para uso do *bus* comum.

A memória dual-port partilhada trata-se de um bloco de memória de 64Kbyte do tipo *dual-port*, implementado com base em quatro bancos de 16Kbyte. Esta memória está organizada a 32 *bits*, embora por parte do *NIC* o acesso se efectue apenas a 8 *bits*. O uso de quatro bancos de 8 *bits* em vez de uma memória de 32 *bits*, é devido à inflexibilidade da segunda opção no que diz respeito a ciclos de escrita nos bancos individualmente.

O *interface NIC / dual-port* serve apenas para realizar o *latch* dos endereços do *NIC* e descodificar o banco ao qual se destina o mesmo, quando este pretende efectuar um acesso ao seu *buffer* externo, ou seja, à memória *dual-port*. Os acessos do *NIC* devem ser feitos sempre a 8 *bits*, embora este tenha potencialidade para os fazer a 16 *bits*, não devendo ser esquecido este facto na inicialização do mesmo, sob pena de incorrecto funcionamento por perda de dados.

O *Refresh* é responsável por gerar os pedidos de acesso à memória dinâmica partilhada para execução de um ciclo de refresh sobre a mesma, bem como a própria execução. O primeiro ponto é executado por uma unidade de temporização que gera periodicamente um pedido de acesso. A execução fica a cargo de uma máquina de estados que realiza a correcta sequência temporal equivalente a um ciclo *CAS Before RAS Auto Refresh*. Este tipo de acesso dispensa uma unidade de contagem para indicação da linha de refrescamento.

O árbitro é responsável pela atribuição ordenada de permissões de acesso aos recursos partilhados. Nesta unidade todos os pedidos têm igual prioridade à excepção do pedido proveniente da unidade de *refresh* que é o mais prioritário. A atribuição nos restantes casos é tratada segundo uma ordem tipo *round-robin*.

B. Opções tomadas na implementação.

Sendo o projecto constituído por vários blocos bem definidos e com fins bastante específicos, inicialmente estaria prevista uma certa divisão das várias entidades que iriam constituir a placa em circuitos separados, ou seja, pretendia-se que os diferentes blocos constituíssem entidades fisicamente diferentes, muito embora comunicantes entre si. Com esta solução pretendia-se integrar as partes de arbitragem, de *interface* com o *transputer, bus PC* e ainda à rede *ethernet*. O *interface* para a memória local dos *transputers* seria feito com lógica discreta. Esta solução implicaria, possivelmente, a construção de quatro *ASIC's* diferentes o que seria inviável economicamente, pois ficaria demasiado dispendioso. Procuraram-se então soluções alternativas. A solução seguinte foi a de construir um único *ASIC* que englobaria a maior e mais complexa parte da lógica e dos circuitos necessários. Para complementar esse circuito integrado desenvolver-se-iam outros circuitos em lógica discreta que

TABELA I
MODOS FUNCIONAIS DO ASIC.

Modo		
ArbE	Mode	Blocos funcionais activos
0	1	***Interface* para *transputer*.** • *Interface* para memória local. • *Interface* para acesso partilhado.
1	1	***Interface* para *transputer*.** • *Interface* para memória local. • *Interface* para acesso partilhado. • Árbitro. • *Refresh*.
0	2	***Interface* para *bus PC*.** • *Interface* para acesso partilhado.
1	2	***Interface* para *bus PC*.** • *Interface* para acesso partilhado. • Árbitro. • *Refresh*.
0	3	***Interface* para *NIC*.** • *Interface Nic / bus* comum. • *Interface Nic / dual-port*.
1	3	***Interface* para *NIC*.** • *Interface Nic / bus* comum. • *Interface Nic / dual-port*. • Árbitro. • *Refresh*.

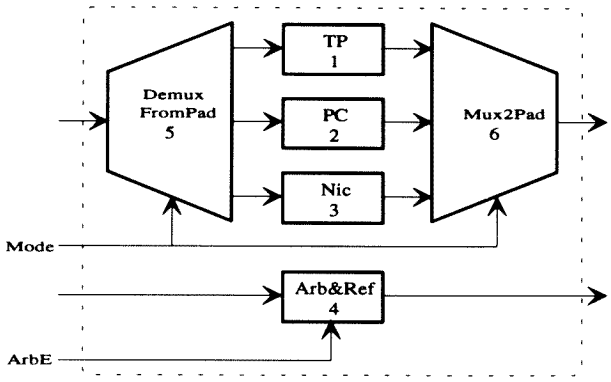


FIGURA 6
DIAGRAMA DE BLOCOS INTERNO DO *ASIC*.

preenchiriam as necessidades de cada *interface* em particular. Esta solução parecia, à primeira vista, que resolveria o problema, mas tal não se verificou pois o número de pinos que o circuito deveria ter, superaria o número máximo que pode ser usado actualmente. Isto deve-se principalmente ao elevado número de linhas de dados e endereços. Por fim surgiu a ideia de desenhar um único *ASIC*, mas com uma filosofia radicalmente diferente desta última. A ideia base desta atractiva solução foi a de construir um *ASIC* que estivesse internamente dividido em várias unidades, cada uma tendo por objectivo realizar uma função específica. Para isso existem no *ASIC* alguns pinos de configuração que permitem definir um modo de funcionamento entre os vários possíveis. Esta opção torna-se especialmente atractiva, pois é possível reduzir bastante o custo do projecto, já que este implica a construção de apenas um *ASIC*. Em termos gerais, o *ASIC* está implementado da forma que a figura 6 ilustra.

De acordo com o modo pretendido é possível efectuar um dos *interfaces* e ainda activar ou não a parte de arbitragem e a unidade de refresh da memória dinâmica partilhada. A tabela I ilustra os vários blocos funcionais que são permitidos obter em cada modo. Pelo facto do ASIC ter vários modos de funcionamento, muitos dos pinos deste tomam funções distintas como se pode ver na tabela II. Notar que os modos de teste apresentados nesta tabela foram implementados com o fim de mais facilmente detectar e isolar erros do *ASIC*.

A placa quadriprocessador tem vários *ASIC's* destes, um para cada *interface* necessário, funcionando de acordo com as respectivas configurações e onde, num e apenas num só, a parte de arbitragem e refrescamento da memória dinâmica partilhada deve estar activa, não interessando no entanto em qual deles. O diagrama de blocos da figura 7 ilustra como na realidade é construida a placa quadriprocessador.

Na introdução do *interface* para *bus PC*, foi de início adoptada a ideia deste ser visto como um sexto dispositivo a disputar a memória dinâmica partilhada. A opção de realizar o *interface* através de um dos *links* de um dos *transputers* pertencente ao sistema através de um *link adaptor* estaria fora de questão, devido à ocupação

TABELA II

DISTRIBUIÇÃO DOS PINOS DO *ASIC*.

Transputer					PC					Nic				
Normal	Teste	I	O	T	Normal	Teste	I	O	T	Normal	Teste	I	O	T
Addr<0:13>	Addr<0:13>		X	X	Addr<0:13>	Addr<0:13>		X	X	Ad32<0:13>	Ad32<0:13>		X	
Data<0:7>	Data<0:7>	X	X	X	Data<0:7>	Data<0:7>	X	X	X	Data<0:7>	Data<0:7>	X	X	X
Data<8:31>	Data<8:31>	X	X	X	Data<8:31>	Data<8:31>	X	X	X					
notNicE	notNicE		X	X	notNicE	notNicE		X	X					
notMemDPE	notMemDPE		X	X	notMemDPE	notMemDPE		X	X					
notRd	notRd		X	X	notRd	notRd		X	X					
notWr<0:3>	notWr<0:3>		X	X	notWr<0:3>	notWr<0:3>		X	X	notCEB<0:3>	notCEB<0:3>		X	
notRas	notRas		X	X	notRas	notRas		X	X	notCas	notCas		X	
notCas	notCas		X	X	notCas	notCas		X	X	notRas	notRas		X	
notResReq	ColnotRow		X		notResReq	ColnotRow		X			Bst1Out		X	
notResAck	notResAck	X			notResAck	notResAck	X				BstTest	X		
BusyNic	BusyNic	X			BusyNic	BusyNic	X			notNicE	notNicE	X		
Reset	Reset	X			Reset	Reset	X			Reset	Reset	X		
MClock	MClock	X			MClock	MClock	X			MClock	MClock	X		
MemAd<0:7>	MemAd<0:7>	X	X		SD<0:7>	SD<0:7>	X	X		Ad<0:7>	Ad<0:7>	X	X	
MemAd<8:15>	MemAd<8:15>	X	X		SA<0:7>	SA<0:7>	X	X		Ad<8:15>	Ad<8:15>	X	X	
MemAd<16:31>	MemAd<16:31>	X	X		SA<8:23>	SA<8:23>	X	X						
notMemRd	notMemRd	X			notMemSR	notMemSR	X			notRd	notRd	X		
notMemWrB0	notMemWrB0	X			notMemSW	notMemSW	X			notWr	notWr	X		
notMemWrB<1:3>	notMemWrB<1:3>	X			IntTp<1:3>	IntTp<1:3>	X							
notMemS0	TestMemSE	X			AEN	TestMemSE	X			ADS0	ADS0	X		
notMemS1	TestMemDPE	X			IOCfg0	TestMemDPE	X				BstClk	X		
notMemS2	TestNicE	X			IOCfg1	TestNicE	X			notAck	notAck	X		
notMemS3	notTestFinish	X			IOCfg2	notTestFinish	X				notTestReqAkRf	X		
notMemS4	notMemS4	X			IOCfg3	IOCfg3	X				notTestRfReq	X		
notMemWait	RWE		X		notIOChRdy	RWE		X			Bst2Out		X	
MemS	MemS	X			MemS	MemS	X							
MemL	MemL	X			notIOW	notIOW	X							
notCasL	End		X		EventTp1	End		X			notRfReq		X	
AddrL<0:8>	AddrL<0:8>		X		MemCfg<0:8>	MemCfg<0:8>	X							
AddrL<9:10>	AddrL<9:10>		X		EventTp<2:3>	EventTp<2:3>		X						
EventNic	EventNic	X			IntNic	IntNic	X							
EventPc	EventPc	X			IntTp0	IntTp0	X							
Event	ContTp		X		Int	ContPc		X			ReqAkRf		X	
IntPc	Idle		X		EventTp0	Idle		X						
					ALE	ALE	X							
Mode<0:1>	Mode<0:1>	X			Mode<0:1>	Mode<0:1>	X			Mode<0:1>	Mode<0:1>	X		
Test	Test	X			Test	Test	X							
notArbE	notArbE	X			notArbE	notArbE	X			notArbE	notArbE	X		
notRemAck<0:4>	notRemAck<0:4>		X		notRemAck<0:4>	notRemAck<0:4>		X		notRemAck<0:4>	notRemAck<0:4>		X	
notRemReq<0:4>	notRemReq<0:4>	X			notRemReq<0:4>	notRemReq<0:4>	X			notRemReq<0:4>	notRemReq<0:4>	X		

I - ENTRADA ; O - SAÍDA ; T - TRISTATE

relativa de um link necessário para outros fins e a troca de informação não ser suficientemente rápida.

Convém referir que, dos quatro *links* que cada *transputer* possui. três são utilizados para comunicação assíncrona entre os *transputers* do sistema. Os quatro *links* que ficam disponíveis servem para que, em aplicações futuras, se possa proceder à ligação deste sistema em malha com outros sistemas idênticos. Sob o ponto de vista do *PC*, este vê a memória partilhada como sendo a sua. Como a memória dinâmica partilhada pode ter uma dimensão até 16Mbyte, nunca seria possível mapea-la no espaço de endereçamento de um *PC*, até porque parte deste já está ocupado. Desta forma, o acesso é feito por paginação de 32Kbyte. Para selecção da página que se deseje aceder são usados registos mapeados no espaço de endereçamento *I/O* do *PC*. O acesso será sempre a oito bits para evitar incompatibilidades existentes entre barramentos de vários fabricantes de *PC* compatíveis que não respeitam rigorosamente o *standard*.

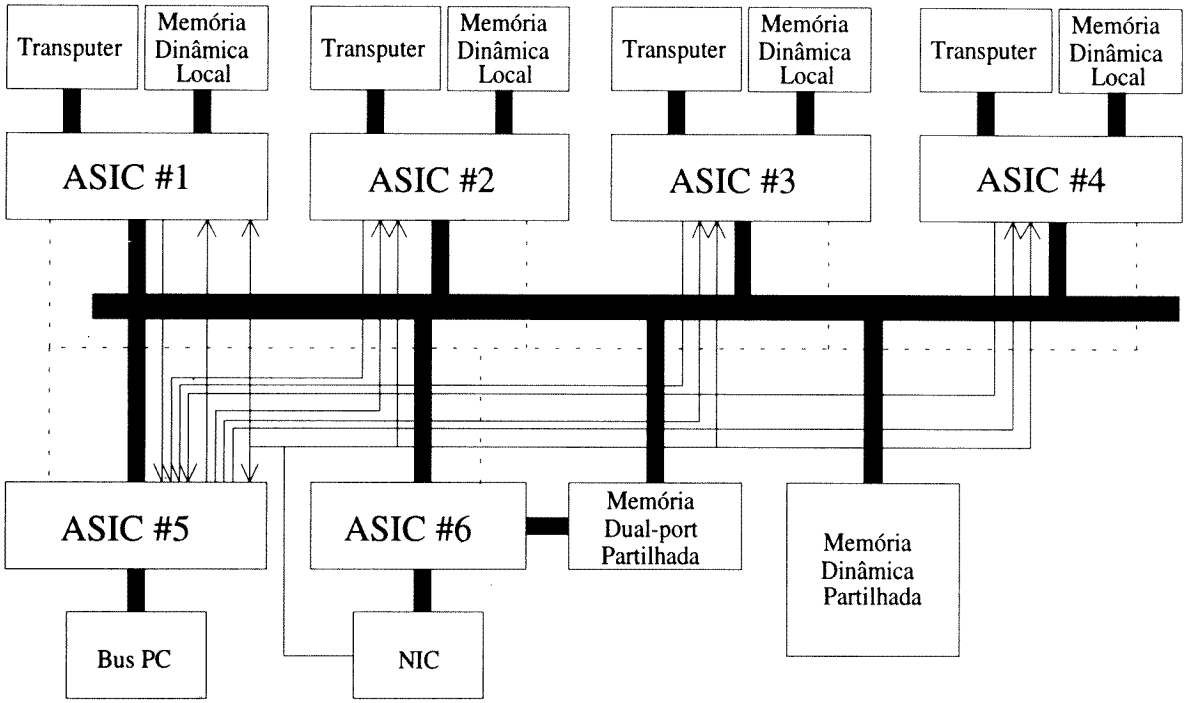


FIGURA 7
PLACA QUADRIPROCESSADOR BASEADA EM *TRANSPUTERS*.

Na especificação do *interface* para rede *ethernet* foi pensado inicialmente mapear a memória que o NIC necessitava para buffer externo na própria memória dinâmica partilhada, o que implicaria que o NIC teria de concorrer com os restantes transputers, ou mesmo com o bus PC, quando efectuasse um acesso ao seu buffer. Esta solução teve de ser abandonada por duas razões. A primeira tem a ver com os tempos de acesso à memória dinâmica partilhada que deveriam ser bastante curtos, pois o *NIC* possui um *buffer* interno muito pequeno.

A única solução seria tornar este utilizador da memória dinâmica partilhada também prioritário sobre os outros utilizadores. A segunda razão afecta cada um dos *transputers* e tem a ver com o tempo de ocupação pelo *NIC* da memória dinâmica partilhada em conjunto com os outros utilizadores, que poderá exceder o tempo máximo que um *transputer* pode ser mantido em lista de espera. Este tempo tem a ver com a necessidade que este tem de efectuar acessos periódicos à memória local para efectuar ciclos de refresh, já que esta também é dinâmica e ser o próprio *transputer* o responsável pela manutenção da mesma. A alternativa tomada para solucionar o primeiro problema relacionado com o tempo de acesso do *NIC* à memória dinâmica partilhada viria agravar o acesso por parte dos restantes utilizadores da memória, já que em certas ocasiões, o *NIC* tem uma actividade muito intensa sobre a sua memória externa. Optou-se então pelo uso de uma memória *dual-port*. Esta memória permite a utilização em simultâneo por duas entidades, tendo o seu próprio gestor de acesso. O acesso a esta memória só não é permitido se de ambos os lados for acedida no mesmo instante, no mesmo endereço e se pelo menos um dos acessos for de escrita.

CONCLUSÕES

Ainda a meio do projecto nos demos conta do quão volumoso ele era na realidade. Em parte devido à novidade dos assuntos que são focados, ao grande número de circuitos que era necessário desenvolver e ainda à complexidade da topologia que foi por nós seguida, nomeadamente a multiplexagem dos vários blocos. Mesmo assim tentámos realizar o maior número de itens que nos foi possível e, em certas ocasiões, desenvolvemos outros que não eram pedidos mas que nós julgarmos serem indispensáveis tendo em vista a maior funcionalidade da placa. Entre eles encontra-se a possibilidade de comunicação assíncrona entre o *NIC*, *PC* e *transputers* (*interrupts* e *events*).

Uma coisa que notámos foi que os prazos que nós previmos para a conclusão de certas tarefas foram sendo progressivamente aumentados devido a contratempos pontuais e imprevisíveis, o que nos leva a concluir que nestes casos convém dar sempre uma margem de variação para que não surjam surpresas desagradáveis, como seja a falta de tempo para acabar tarefas importantes. Daquilo que nos foi proposto concluímos o que se pode chamar primeira fase do projecto que era a síntese do circuito integrado *VLSI* que pudesse servir de *interface* e árbitro. O projecto ficou pelo desenvolvimento de um ASIC. Falta começar a segunda e última parte deste projecto que é o

desenho da placa de circuito impresso, da qual já fizemos um primeiro desenho completo, mostrando como devem ser interligados todos os elementos.

Em simultâneo pode-se ir pensando num, ou antes vários programas em *OCCAM* de teste do placa quadriprocessador. Esta fase é muito importante e os programas devem testar todo o tipo de situações, por mais insólitas que sejam. Deveria também ser desenvolvido um bom programa de interface com o utilizador para *PC* para tornar a utilidade futura deste projecto uma realidade e não só uma boa ferramenta de ensino para professores e alunos.A totalidade do trabalho que foi desenvolvido foi feito no *Cadence*, com a ajuda dos seus programas satélite e bibliotecas *ECPD10*. Depois de tantos meses, quase um ano, a trabalhar com ele descobrimos algumas particularidades que dependem da maneira particular como cada um prefira trabalhar.

Estamos convictos que o projecto que nos propusemos desenvolver é muito útil e pode vir a ser bastante usado em aplicações que necessitem de grande capacidade de processamento paralelo tais como tratamento e síntese de imagens, de som, controlo de grandes quantidades de informação em tempo real, etc. Se no futuro se juntarem duas ou mais placas quadriprocessador como esta de modo a formarem uma rede seria interessante fazê-las correr um programa do tipo de redes neurais, ou seja tentar desenvolver aplicações tendo em vista a inteligência artificial.

Apesar da pastilha não se encontrar pronta a ser produzida, já que seria necessário proceder a mais algumas simulações, a fim de estudar o seu comportamento em alguns pormenores, foi feito o *place* e *route*. A primeira operação permite posicionar geograficamente na pastilha, as diferentes células que compõem o circuito. Este operação pode ser feita em parte automaticamente. A segunda executa as ligação entre células de modo a implementar o circuito projectado.

Apesar desta fase também poder ser automática, por vezes é necessário efectuar algumas correcções. Na primeira fase é gerado um ficheiro com alguns dados em forma de previsões. Estas permitem ter uma ideia de algumas dimensões da pastilha de silício final. Assim, foi verificado que a pastilha é *pad limited*, ou seja, as suas dimensões são impostas pelo número de pinos e não pela área do circuito. O tamanho deste circuito é cerca de $6.6mm^2$ no entanto a área do ASIC incluindo os *pad's* de *I/O* é aproximadamente $34.9mm^2$. Um facto importante que se constatou, foi o de a área ocupada pelas macro células utilizadas no esquemático do árbitro, ser bastante grande.

AGRADECIMENTOS

Um agradecimento muito especial para o Engº Velez e o Engº Joaquim Ferreira que se prontificaram a elucidar-nos em todos os problemas referentes ao uso do software e das máquinas disponíveis desbloqueando-nos os meios e conhecimentos necessários de modo a facilitar o nosso trabalho.

REFERÊNCIAS

[1] *The transputer databook* - INMOS

[2] *IBM AT technical reference manual* - IBM

[3] *Cadence VLSI Design Notes* - A.P. Marriott

[4] *ES2 Asic Design Guidelines* - European Standard Structures

[5] *Data sheet MSC2312A -xxYS9/KS9 DRAM* - OKI semiconductor

[6] *Data sheet do DP8390C Network Interface Controller* - National Semiconductor

[7] *Data sheet da IDT7006S/L* - Integrated Device Tecnology, Inc.

Simulação de Algoritmos para Equalização de Sinais Áudio

António Ricardo Silva, Filipe Bento e José Vieira

Resumo- **Este trabalho pretende analisar a viabilidade de um equalizador totalmente digital para aplicação em sistemas áudio. São abordadas duas arquitecturas : uma utilizando filtros FIR e outra com filtros IIR. As vantagens e desvantagens de cada uma das arquitecturas serão discutidas. São também apresentados e comparados os resultados obtidos com cada uma das abordagens.**

Abstract- **This paper analises the possibility of designing an all digital audio equalizer. Two aproaches are discussed : one uses FIR filters and the other uses IIR filters. The advantages and disadvantages of each one of these architectures are discussed as well. Finally, we will present and compare the results.**

I. INTRODUÇÃO

A. Enquadramento do trabalho

Este trabalho insere-se no estudo de viabilidade de um equalizador totalmente digital para ser utilizado em sistemas áudio a colocar no mercado por uma conceituada marca da especialidade.

B. Filtros Digitais

Existem basicamente dois tipos de filtros digitais : os filtros de resposta impulsional finita (Finite Impulse Response - FIR) e os filtros de resposta impulsional infinita (Infinite Impulse Response - IIR). Os primeiros caracterizam-se por a transformada de Z da sua resposta impulsional só ter zeros, pelo que são sempre estáveis. Podem ter resposta em fase linear - e atraso de grupo constante - factor importante em aplicações áudio. A sua principal desvantagem consiste em necessitarem de um elevado número de coeficientes para garantirem bandas de transição muito estreitas.

A transformada de Z da resposta impulsional de um filtro IIR tem polos e zeros, pelo que podem ser instáveis em determinadas condições. A principal desvantagem destes filtros reside no facto da fase não ser linear. No entanto, necessitam de um número muito inferior de coeficientes para as mesmas especificações em frequência que os filtros FIR.

Ao longo deste estudo foram projectados, simulados e comparados dois equalizadores de características exteriores semelhantes, mas com estruturas internas diferentes. A principal diferença entre eles reside no facto de utilizarem tipos de filtros diferentes : um utiliza filtros FIR e o outro utiliza filtros IIR.

II. ARQUITECTURA

Os Equalizadores projectados têm arquitecturas diferentes. Um deles utiliza filtros FIR e decimação/interpolação, enquanto o outro utiliza filtros IIR e um processamento paralelo. Designaremos estes dois equalizadores por EFIR e EIIR. Segue-se uma pequena descrição de cada uma delas.

A. Arquitectura EFIR

O EFIR tem uma estrutura em árvore, sendo o sinal de entrada dividido em duas bandas. Destas duas, a banda superior é multiplicada pelo respectivo ganho e a inferior é decimada e dividida novamente em duas bandas. Esta operação será repetida até se obterem o número de bandas pretendidas.

Após ter sido efectuada a separação do sinal nas várias bandas de frequência e terem sido aplicados os vários ganhos, procede-se ao reagrupamento do sinal. É feita a compensação dos atrasos introduzidos pelos filtros pelo facto de se trabalhar a frequências de amostragem distintas em cada banda. Será ainda efectuada a operação de interpolação, e respectiva filtragem, para repor a frequência de amostragem original e permitir o reagrupamento do sinal já equalizado. A Fig. 1 mostra um equalizador de cinco bandas que utiliza esta arquitectura. São utilizados três tipos de filtros: filtro decimador (DC), filtro interpolador (IT) e filtro passa-alto (HP). A Fig. 2 mostra uma aproximação da resposta em frequência destes três filtros.

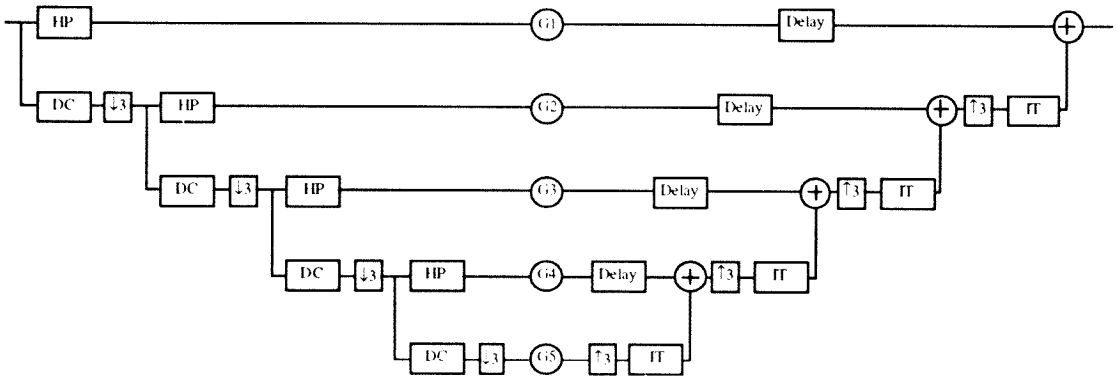


Figura 1 Arquitectura EFIR

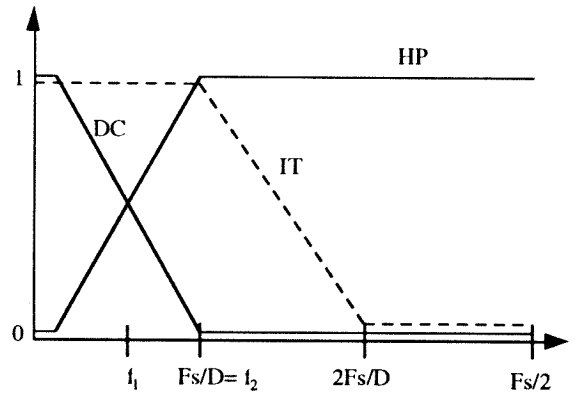


Figura 2 Filtros do EFIR

Os filtros HP e DC dividem as bandas simetricamente em torno da frequência f1, e o filtro IT elimina as réplicas geradas na operação de interpolação. Quando os ganhos em todas as bandas tomam o valor 1, o sinal à saída do equalizador será igual ao sinal de entrada, a menos de um atraso constante e de ruído.

B. Arquitectura EIIR

No que toca ao desenho do EIIR (Fig. 3), é importante realçar a utilização de um conceito que, embora não sendo muito inovador, é bastante prático. Este conceito visa eliminar o incómodo *ripple* resultante da não continuidade entre bandas e da sua soma, presente nos equalizadores convencionais. Nestes, conseguia-se uma diminuição do ripple "sacrificando" a ordem dos filtros, tornando todo o processo de computação bastante lento devido à complexidade introduzida pelo aumento da ordem dos filtros. Com este novo conceito, o sinal resultante à saída não é a soma dos sinais resultantes da filtragem pelas diferentes bandas, mas sim a soma ou subtracção destes em relação ao sinal de entrada. Isto é, o sinal de entrada está sempre presente na saída, sendo amplificado ou atenuado conforme a equalização desejada nas frequências relativas às bandas escolhidas. Conclusão: se todos os ganhos forem 0, então o sinal de entrada está presente à saída sem ter sido minimamente alterado. Se se desejar dar enfase ou atenuar uma dada banda ou bandas, isto poderá ser feito alterando o ganho dessa banda, continuando as outras frequências inalteradas (sem ripple).

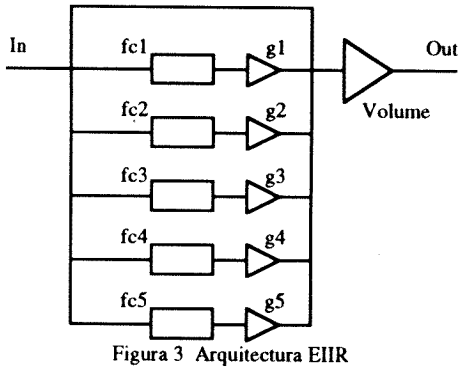


Figura 3 Arquitectura EIIR

III. CARACTERÍSTICAS EXTERNAS

As características externas (frequências centrais e nº de bandas) do equalizador, foram escolhidas após uma pesquiza de equalizadores da várias marcas e tendo em conta as limitações impostas pela arquitectura do EFIR. No estudo efectuado, procurámos saber qual o nº de bandas, bem como as frequências centrais mais utilizadas. A primeira decisão foi, como é natural, a escolha do número de bandas, de modo a dar ao equalizador o aspecto necessário para se tornar comercial (objectivo final do produto em investigação). Pelos dados adquiridos, a maioria dos equalizadores possuia um número de bandas menor do que 10.

Inicialmente, implementámos os equalizadores EFIR e EIIR com especificações diferentes, porque nesta fase do estudo procurávamos saber até que ponto poderiam ir as nossas espectativas. Assim, ao longo deste percurso, ficaram para trás várias etapas. Destas salientamos a simulação de equalizadores com 10, 7 e 5 bandas para o EIIR e 8,7 e 5 bandas para o EFIR, diferentes valores para as frequências centrais de cada banda, e de um modo geral diferentes características dos filtros a utilizar. Nesta altura, evidenciou-se uma grande desvantagem dos equalizalizadores de desenho FIR sobre os de desenho IIR: a escolha das frequências está restringinda a valores resultantes das operações de interpolação e decimação, enquanto nos de desenho IIR, qualquer frequência é possível. Um outro problema, reside no facto de no EFIR, quando aumentamos o número de bandas, os atrasos se

TABELA I
Alguns equalizadores existentes no mercado

Equalizador	Bandas	Frequências centrais (Hz)									
PHILIPS	3	250			4K			16K			
GRUNDIG	3	100			1K			10K			
ATLANTIS	4	150		500			3K			8K	
PIONEER	5	100		330		1K		3.3K		10K	
SONY	5	100		400		1K		4K		12K	
SANSUI	5	100		350		1K		3.5K		10K	
PHILIPS	5	63		250		1K		4K		10K	
SHARP	5	63		250		1K		4K		16K	
ATLANTIS	5	100		300		1K		3K		10K	
AUDIOSONIC	5	60		250		1K		4K		10K	
OMEGA	5	100		310		1K		3.1K		10K	
TECHNICS	7	63	160	400	1K		2.5K		6.3K		12.5K
SONTEC	10	100	180	310	550	1K	1.8K	3.1K	5.5K	10K	16K
PHILIPS	10	31	63	125	250	500	1K	2K	4K	8K	16K

TABELA II
Características dos equalizadores

BANDA	Frequências Centrais (Hz)
1	84
2	335
3	1004
4	3014
5	13285

tornarem incomportáveis para aplicações em tempo real. Com 7 bandas e decimação por 3, o atraso era da ordem dos 400 milisegundos! Entende-se como aplicação em tempo real, por exemplo, a situação em que se exige sincronismo entre imagem e som.

Depois de uma análise mais detalhada, tendo em atenção estas limitações, e para se poder proceder a um teste comparativo, decidimos optar por um equalizador de cinco bandas para versão "oficial" da nossa investigação, sendo este o modelo a utilizar em futuros trabalhos (como por exemplo a sua implementação no DSP 56004). Assim, decidimos simular um equalizador com cinco bandas e com as frequências centrais indicadas na Tabela I. Este modelo é utilizado tanto no EFIR como no EIIR.

IV. MODELO FINAL

A simulação do equalizador final foi independente para os dois tipos de equalizadores (EFIR e EIIR). Todas as experiências foram efectuadas separadamente, apesar de ser mantido algum diálogo entre os dois executantes, de modo a permitir que os resultados obtidos fossem comparados e actualizados. A simulação foi feita utilizando o programa *MatLab*©, tendo sido desenvolvido ao longo do trabalho um conjunto de funções que se prestassem à simulação dos equalizadores pretendidos.

A. Equalizador EFIR

Nesta arquitectura, enfrentámos alguns problemas que influenciaram decisivamente a versão final do equalizador. Em primeiro lugar, e como já referimos, utilizando esta arquitectura baseada em decimações e interpolações, não é muito viável a execução de um equalizador com mais de 5 bandas. Vejamos um pequeno exemplo. Utilizando decimações e interpolações por 2 e 7 bandas de frequência, teríamos as frequências centrais na ordem dos 15KHz, 5.8KHz, 2.9KHz, 1.5KHz, 720Hz, 360Hz e 120Hz (valores aproximados). A utilização destes valores de frequências centrais não é muito comercial, visto controlar pouco as baixas frequências. O atraso de grupo ronda os 50 ms, valor aceitável para áudio. Utilizando decimações por valores superiores vamos aumentar o atraso de grupo, entrando em valores não aconselháveis para este tipo de aplicações (utilização em tempo real).

Assim, depois de termos decidido realizar um equalizador de cinco bandas e decimação por 3, surgiu um novo problema : qual o valor de f1 (Fig. 2) a utilizar nos filtros DC e HP? Depois de uma fase de estudo, chegámos ao valor 0.1025*Fs, que dá origem aos valores mencionados anteriormente para as frequências centrais, além das frequências inferior e superior de cada banda apresentadas na Tabela II. Este equalizador tem ainda uma característica especial : controla todas as frequências a partir dos 0Hz.

O terceiro problema que encontrámos foi decidir onde e como calcular os filtros. Possuíamos algumas "ferramentas" para fazer esse cálculo, mas decidimos utilizar a função *Remez* do *MatLab* para o fazer. Esta escolha deveu-se a três factores : em primeiro lugar, os filtros são muito fáceis de obter; em segundo, quando precisamos de fazer uma pequena alteração a um filtro, isso pode ser feito dentro do programa, evitando o cálculo externo ao *MatLab* e consequentemente um novo *import* para este; a terceira deveu-se ao facto de se puderem controlar as características dos filtros de um modo muito simples. Apesar disto, o programa por nós desenvolvido comtempla a hipótese de carregar filtros externos ao *MatLab* se o utilizador assim o desejar.

A escolha da ordem dos filtros, da frequência f1 (Fig. 2) e da banda de transição dos filtros DC e HP foi feita tendo como objectivo conseguir que o equalizador tivesse um *ripple* inferior a 0.1dB, considerando ganhos unitários em todas as bandas e resposta a impulso.

Os melhores resultados foram conseguidos utilizando filtros com as características incluidas na Tabela IV (o valor 1 na coluna da Frequência normalizada corresponde a metade da frequência de amostragem).

Utilizando estes filtros, obtivemos um ripple máximo de 0.084 dB à saída do equalizador, considerando ganhos unitários e resposta a impulso, o que dá um valor inferior a 0.1 dB em toda a gama de frequências (0 ~ 22050 Hz), tal como pretendíamos.

TABELA III
Frequências do EFIR

Banda	Frequência inferior (Hz)	Frequência superior (Hz)	Frequência central (Hz)
1	0	167	84
2	167	502	335
3	502	1507	1004
4	1507	4520	3014
5	4520	22050	13285

TABELA IV
Características dos filtros HP, DC e IT

Filtro	Frequência normalizada (Fs=2)	Amplitude
Passa alto	0	0
	0.15	0
Ordem : 51	0.26	1
	1	1
Decimador	0	1
	0.15	1
Ordem : 51	0.26	0
	1	0
Interpolador	0	0
	0.26	0
Ordem : 45	0.4066	1
	1	1

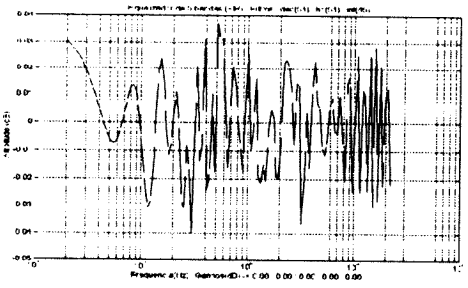


Figura 4 EFIR : Resposta em amplitude

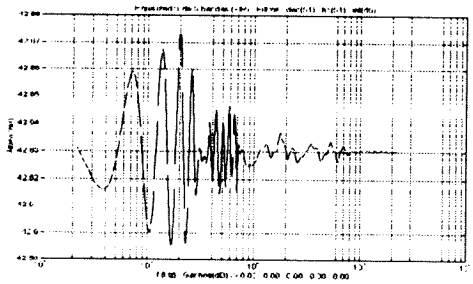


Figura 5 EFIR : Atraso de Grupo

Quanto ao atraso de grupo, este é praticamente constante em todas as frequências, tomando o valor de 42.63ms com um desvio máximo de 0.1 %. As figuras 4 e 5 mostram estes resultados.

Devemos referir que os resultados que se obtêm quando colocamos os ganhos iguais em todas as bandas são os mesmos que os obtidos para 0 dB, qualquer que seja o valor utilizado para o ganho. Por exemplo, se o ganho fosse de 10 dB em todas as bandas, o atraso de grupo seria de 42.63ms e o ripple seria inferior a 0.1 dB. Uma outra experiência que fizemos foi alterar o ganho nas várias bandas do equalizador. Assim, numa primeira simulação, utilizámos ganhos alternadamente a +12 dB e a -12 dB. Os resultados foram bastante bons, com o equalizador a responder muito bem a variações bruscas do ganho em bandas consecutivas. O atraso de grupo manteve-se em torno de 42.63 ms, aumentando o desvio máximo para 1%. Na segunda experência, fizemos os ganhos tomar vários valores entre -10 dB e +10 dB. A resposta continuou a ser extremamente satisfatória, com os resultados a seguirem os dados fornecidos à entrada. O atraso de grupo manteve-se dentro dos valores obtidos anteriormente.

Os resultados obtidos estão dentro das características que pretendíamos para o nosso equalizador, pois temos um erro máximo de 0.1 dB na amplitude e uma distorção em fase inferior a 1%, valores excelentes para a aplicação em vista.

B.Equalizador IIR

A ordem escolhida para as implementações foi 2 porque, depois de comparados os resultados obtidos para diferentes ordens, chegámos à conclusão de que aumentar a ordem só piora os resultados. Se bem que se obtêm filtros com um menor *ripple* na banda de passagem (de qualquer forma este *ripple* para ordem 2 é practicamente zero), obtém-se também uma maior atenuação na zona de corte associada a uma rápida transição entre estas duas zonas, e isto embora à primeira vista pareça benéfico, na realidade não o é! A razão para tal, prende-se com o facto que o fosso entre duas bandas consecutivas vai aumentar, fazendo portanto aumentar o *ripple* para ganhos iguais para todas as bandas (excepto para ganhos 1, em que o ripple se mantém, e é igual a zero), sendo este tanto maior quanto o forem os ganhos.

Por outro lado, embora um equalizador do tipo "degraus ascendentes e descendentes" para a transição entre bandas seja gráficamente mais agradável, do ponto de vista prático, transições bruscas entre frequências resultam num efeito algo artificial aquando da audição da equalização. Por isso a opção de utilizar apenas ordem 2, pelas suas transições mais suaves.

Os filtros IIR são muito versáteis, mas apresentam um atraso de grupo com valores muito diferentes entre frequências consecutivas. Pondo este factor em investigação, chegámos à esperada conclusão que, quanto maior a ordem utilizada, maior o atraso de grupo e maiores as suas transições. Mais uma vez, a ordem 2 é a melhor opção.

Agora, o nosso objectivo tinha dois fins: um era diminuir o atraso de grupo, o outro diminuir o *ripple* para ganhos iguais para todas as bandas, quando estes são diferentes de um. Aqui era preciso conciliar diferentes variáveis, como por exemplo: o *ripple* máximo permitido na banda-de-passagem para cada filtro no desenho Chebyshev tipo I, a atenuação na zona-de-corte para os filtros Chebyshev tipo II, e os dois factores anteriores para o desenho de filtros Elípticos.

Outro factor comum a todos é o dimensionamento das frequências de corte inferior e superior dos filtros. Esta assume uma vital importância, e a sua escolha poderá parecer apenas dependente do número de bandas e do espaçamento das frequências centrais de cada filtro mas, na práctica, esta tem de ser escolhida atendendo ao facto que terá de haver sobreposição entre bandas e se o seu valor for muito baixo (entre .5 e 1), a soma atrás mencionada tomará valores que tenderão a dar um ganho muito maior ou muito menor (conforme as bandas adjacentes tiverem ganho ou sejam atenuadas) em relação ao escolhido pelo utilizador. Por outro lado, quanto menor for esta relação (maior espaçamento entre frequência superior e inferior de corte dos filtros), menor será o *ripple* na banda de passagem do equalizador. Portanto, a sua escolha é o resultado de uma solução de compromisso

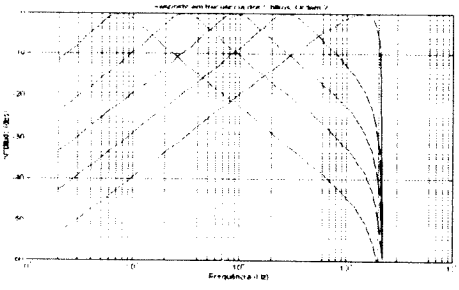


Figura 6 EIIR : Resposta em frequência dos filtros

entre os dois efeitos mencionados. Decidimos optar, para o equalizador de 5 bandas, pelo valor 1.15 e utilizá-lo em todos os testes.

Poderíamos simplificar o problema do *ripple* para ganhos iguais de todas as bandas, afirmando que este não existe porque o equalizador foi desenhado para trabalhar à volta de ganhos unitários (0dB), em que o *ripple* é igual a zero. O utilizador, se quizer só ganhos ou só atenuações poderá fazê-lo aumentando ou diminuindo o volume no amplificador, não utilizando o equalizador para tal fim (pondo este com equalização à volta dos zeros dB, e não apresentando *ripple*).

Os resultados obtidos são bastante elucidativos: o filtro de Butterworth, que não pede variáveis de entrada, apresenta melhores resultados que os outros filtros, com um processo computacional mais simples (e logo mais rápido), só se alcançando resultados iguais através do filtro de Chebyshev tipo II. Utilizando este desenho, o *ripple* de equalização, para ganhos iguais em todas as bandas (iguais a 1 dB), situa-se em .4dBs.

Quanto a esta última varíavel de saída, o atraso de grupo, e para as mesmas condições em que foi medido o valor do *ripple* de equalização, obtivemos valores praticamente idênticos para os diferentes métodos, decaindo o seu valor sempre para zero, sendo quase nulo para valores de variação angular superiores a .25rad/seg (1.75 KHz), o que significa que, mesmo para trabalhar em condições que se pretenda sincronismo entre som e imagem, este não constitui qualquer ameaça.

De seguida, mostramos alguns resultados práticos obtidos com a versão final do equalizador EIIR.

V. EXPERIÊNCIAS COM SONS

Após a conclusão do projecto dos dois equalizadores, previa-se a sua implementação em DSP, de modo a podermos testá-los em tempo real. Devido ao facto de não terem sido reunidas as condições necessárias para a execução desta etapa, optámos por fazer a equalização de pequenos trechos de música através da gravação em ficheiros.

Para esta fase do trabalho, deparámos com um novo conjunto de dificuldades, entre as quais se encontram a conversão de formatos de som e o enorme tamanho dos ficheiros (recorde-se que 1 segundo de som de um CD corresponde a 44100 amostras). Para tornear estes problemas, seguimos várias etapas :

1 - Gravar o trecho de som do CD para um ficheiro em formato *.aiff* da Silicon Graphics, utilizando uma aplicação da máquina;

2 - Converter estes ficheiros para formato *.snd* da NeXT com a função *soundfiler* da Silicon Graphics;

3 - Utilizando a função *tomono* desenvolvida na NeXT, convertemos o trecho stereo para mono.

4 - Com a função *toascii*, convertemos o ficheiro do formato *.snd* para *ascii*, de modo a podermos importá-lo para uma variável do *MatLab*. Esta função atribui valores entre -2exp15 e 2exp15 (32827), tendo que ser normalizado para valores entre -1 e 1, para evitar distorção;

5 - No *MatLab*, efectuamos as operações de equalização dos trechos e gravamos o resultado num novo ficheiro em formato *ascii*;

6 - De novo no NeXT, efectuamos a conversão de formato ascii para *.snd*, de modo a podermos ouvir os resultados. Esta conversão é feita utilizando uma função chamada *sndconvert* existente no sistema. A função deve ser utilizada com os seguintes parâmetros :

sndconvert -o "file".snd -f 6 -s 44100 -c 1 -a "file".ascii

Estes parâmetros fazem a conversão para float (-f 6), frequência de amostragem de 44100 Hz (-s 44100) e mono (-c 1) de um ficheiro em formato ascii (-a).

Um outro problema neste tipo de simulações, prende-se com o facto de um ficheiro de som ter um tamanho considerável (tipicamente, acima de 1 M, dependendo do tempo de gravação). Quando carregamos esse ficheiro para uma variável do *MatLab*, esta vai ocupar um espaço em memória que é proporcional ao tempo de duração do trecho musical, além do tempo de processamento também aumentar consideravelmente. Assim, tivemos de nos restringir a trechos com duração inferior a 2 ou 3 segundos.

Quanto a resultados, foram bastante satisfatórios, se atendermos ao facto de que é difícil analisar trechos musicais com 2 segundos de duração! Apesar deste facto, é notória a diferença entre o trecho original e o trecho equalizado, sendo perceptível, por exemplo, a ênfase dada aos agudos numa das equalizações que fizemos.

VI. CONCLUSÕES

Chega agora a altura de compararmos as duas implementações do equalizador e de fazermos uma análise aos resultados obtidos ao longo deste estudo.

No que se refere aos equalizadores implementados, estes têm características algo diferentes. Podemos comparar, essencialmente, duas delas: o módulo da resposta em frequência e o atraso de grupo. As figuras 7 e 8 mostram os resultados obtidos em simulações com as mesmas especificações para os dois equalizadores.

Quanto à primeira característica, o EFIR tem uma resposta em frequência mais plana e segue mais de perto as indicações dadas pelo utilizador. O IIR tem uma resposta mais curva, mas não consegue, por vezes, seguir as especificações de entrada, especialmente quando se verifica a sobreposição entre bandas adjacentes, sendo este problema ultrapassável, como foi referido em V. Em relação ao EFIR, o facto da resposta ser plana poderá não ser tão agradável para o ouvido como para a vista, embora, com os ficheiros de som que equalizámos este problema não tenha sido notado. No entanto, no caso de se verificar que as transições entre bandas adjacentes são muito bruscas, isto poderá ser ultrapassado se a banda de transição dos filtros DC e HP for mais larga.

O atraso de grupo, no EFIR, é praticamente constante, e está dentro de valores perfeitamente aceitáveis para aplicações áudio. No entanto, se aumentarmos o número de bandas, poderemos cair em valores incomportáveis para aplicações em tempo real. No EIIR, o atraso de grupo assume valores baixos, mas não é constante em toda a

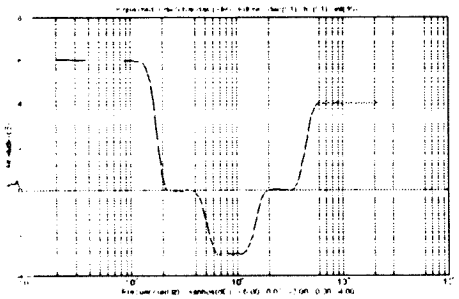


Figura 7 Simulação EFIR : Resposta em frequência

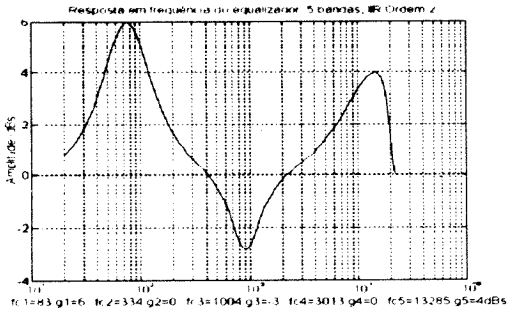


Figura 8 Simulação EIIR : Resposta em frequência

gama de frequências. Um melhoramento possível passa por introduzir filtros que tornem a fase linear na banda passante (e consequentemente melhoram o atraso de grupo).

Um exemplo típico deste tipo de filtros é o de desenho *Bessel*. Contudo, para obtermos tais resultados teríamos de sacrificar a ordem para conseguirmos atenuações dentro dos valores alcançados com ordem 2 nos filtros de desenho *Butterworth* (estando contudo outros tipos de desenho disponíveis e outras tantas opções no programa implementado).
Numa análise final aos resultados obtidos, consideramos que estes se enquadraram dentro daquilo que estava previsto inicialmente, ultrapassando as expectativas em alguns pontos. As equalizações que fizemos a ficheiros de som apresentaram resultados bastante promissores para uma futura implementação em DSP. Alguns problemas que possam surgir nessa implementação (em termos de arquitectura e projecto dos equalizadores) poderão, eventualmente, ser ultrapassados com as sugestões atrás referidas.

REFERÊNCIAS

[1] R. E. Crochiere and L. R. Rabiner, "Multirate Digital Signal Processing", Prentice Hall, 1983
[2] Dennis A. Bohn, "Accelerated Slope Tone Equalizers",Journal of the Audio Engineering Society, vol 40, n°12, December 1992
[3] José Vieira, "Controlo de Tonalidade para um Amplificador Áudio", Dep. Electrónica e Telecomunicações, Universidade de Aveiro, 1993
[4] A. Z. Adkins, "An All-Digital Audio Equalizer Design", University of Florida, 1989
[5] "Digital Stereo 10 band Graphic Equalizer using the DSP56001", Motorola Inc, 1988
[6] Childers and Durling, "Digital filtering and Signal Processing", West Publishing Co, 1985
[7] Fred Mintzer, "On half-band, third-band and Nth-band FIR filters and their design", IEEE Transactions on Accoustics, Speech and Signal Processing, vol ASSP30, n° 5
[8] H. J. Blinchikoff and A. I. Zverev, "Filtering in the time and frequency domain", John Wiley and Sons, 1976
[9] V. K. Jain and R. E. Crochiere, "A Novel Approach to the Design of Analysis/Synthesis Filter Banks", Proc. 1983, IEEE Int. Conf. Acoust. Speech Signal Processing, April 1983
[10] A. V. Oppenheim and R. W. Shafer, "Digital Signal Processing", Prentice Hall, 1975
[11] J. D. Johnston," A Filter Family Design for use in Quadrature Mirror Filter Banks ", Proc. 1980, IEEE Int. Conf. Acoust. Speech Signal Processing, April 1980
[12] Roman Kuc,"Introduction to Digital Signal Processing", McGraw Hill book Company, 1988
[13] J. Dattorro, "The Implementation of Digital Filters for High Fidelity Audio", AES 7th International Conference
[14] R. Hooke and T. A. Jeeves,"Direct Search Solution of Numerical and Statistical Problems", Westinghouse Research Laboratories, 1960
[15] V. K. Jain and R. E. Crochiere, "Quadrature Mirror Filter Design in The Time Domain", Transactions on Accoustics, Speech and Signal Processing, vol ASSP32, n°2, 1984

A Universidade não Acaba Aqui

Associa-te

ASSOCIAÇÃO DE
ANTIGOS ALUNOS
DA UNIVERSIDADE
DE AVEIRO

Campus Universitário 3800 AVEIRO

VANTEC

tecnologias de vanguarda
sistemas de informação

Rua da Preciosa, 306 - 1º
4100 Porto
PORTUGAL
Tel. 351.2.610 7420/21
Fax. 351.2.610 7419
E - mail. com@vantec. pt

ENAFAR ENAFAR ENAF

Distribuimos para todo o pais

- Modems
- Modems / Fax
- UPS
- Scanners A4 e Scanners de mão
- Hardware para redes Ethernet
- Multimédia

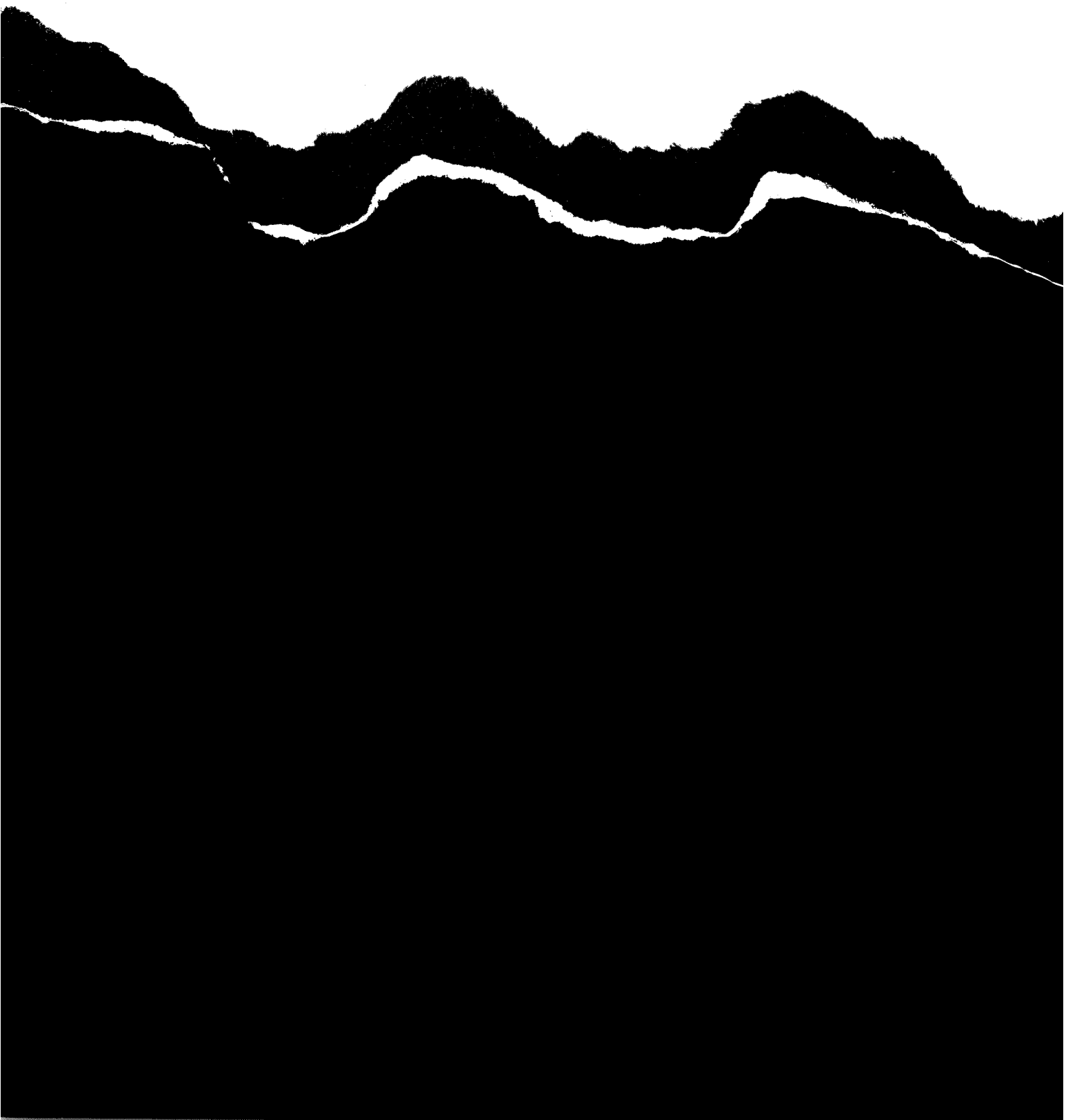
Somos especialistas em periféricos de informática

Sede: Quinta D. João Lt 1 R/c Esq. 3000 COIMBRA
Telf: 039 404561 (3 linhas) fAX: 404549
Filial: Tv Senhor de Matozinhos 75 4400 GAIA
Telf: 02 7124229/449 Fax: 7124528

NORMÁTICA, LDA

COMPUTADORES
IMPRESSORAS LASER
PLOTTERS
PERIFÉRICOS
CONSUMIVEIS

FORNECEDORES DA CENTRAL
DE COMPRAS DO ESTADO

HEWLETT PACKARD
REVENDEDOR
AUTORIZADO

MORADA: R. General Justiniano Padrel, 16-A
1100 LISBOA
TELEFONES: 814 65 69 - 814 07 28 - 814 75 25
FAX: 814 65 99