DEPARTAMENTO DE

ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Testes de Conformidade e Interoperabilidade em Ambiente SMDS

FACULDADE DE ENGENHARIA

UNIVERSIDADE DO PORTO

Rua dos Bragas, 4099 Porto Codex – PORTUGAL

JORGE BOTELHO DA COSTA MAMEDE

Testes de Conformidade e Interoperabilidade em Ambiente SMDS

681.3(043)/MAM&/TES
UNIVERSIDADE DO PORTO
Faculdade de Engenharia
BIBLIOTECA
N.º 2149
COTA
DATA 5 6 96
N.N.º 2149
cat

Dissertação submetida para satisfação parcial dos requisitos do programa do curso de Mestrado em Engenharia Electrotécnica e de Computadores
(Área de especialização em Telecomunicações)

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Dissertação realizada sob a supervisão de

Prof. Doutor Eurico Manuel Elias de Morais Carrapatoso

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

À Isilda

Aos meus pais
e à minha irmã

Resumo

A evolução das redes de telecomunicações e a definição de uma arquitectura para redes de banda larga que permite a integração de serviços (RDIS-BL) têm proporcionado o desenvolvimento da indústria dos serviços e equipamentos. Este desenvolvimento está patente na definição de novos serviços de forma a tirar partido da tecnologia *ATM* em que se baseia a RDIS-BL.

Um serviço que ocupa uma posição bem definida nessa área é o *SMDS* (*Switched Multi-megabit Data System*), cujas características permitem que seja facilmente adaptado e utilizado de uma forma simples e transparente como suporte para outros serviços e aplicações.

Face à adopção do *SMDS* como meio para aceder às redes de banda larga por parte de um grande número de operadoras de telecomunicações, é previsível que serviços e aplicações sobre o *SMDS* venham a ser uma área de investimento dos fabricantes, a curto e a médio prazo.

O sucesso de um investimento no desenvolvimento desses produtos é muitas vezes condicionado pelo tipo de testes a que são submetidos durante a fase de fabrico. É nesta altura que os testes de conformidade e de interoperabilidade dos protocolos implementados desempenham um papel relevante ou mesmo imprescindível. Considerando este facto, pensou-se em adaptar uma ferramenta definida para efectuar testes de desempenho e de avaliação da qualidade de serviço em redes de comunicação, o SGDT - Sistema de Geração e Detecção de Tráfego, de forma a permitir-lhe executar também testes de conformidade e de interoperabilidade de implementações de protocolos que operem sobre o serviço *SMDS*.

A alteração do SGDT passou por desenvolver uma plataforma de interface a uma ferramenta de edição e definição de testes em *TTCN* (*Tree and Tabular Combined Notation*), o *ITEX*, e por desenvolver uma interface que garanta o acesso ao *SMDS*. O resultado foi um equipamento que mantem a capacidade de testar mas é muito mais poderoso e flexível porque possibilita a definição e execução de testes descritos em *TTCN* por qualquer operador de testes e porque dispõe de uma interface para aceder ao serviço *SMDS*.

Nesta dissertação são apresentados os conceitos de conformidade e de interoperabilidade de implementações de protocolos e os requisitos e metodologias de teste de acordo com as Recomendações *ITU* existentes. O *TTCN* e o *ITEX* são também apresentados, bem como uma descrição sumária da arquitectura e do funcionamento do SGDT antes das alterações a que foi submetido. As características e potencialidades do *SMDS* e da interface desenvolvida para o SGDT aceder a esse serviço são também descritas neste trabalho. Por fim, são comentadas e ilustradas algumas configurações de teste em que o novo SGDT pode ser usado para efectuar testes sobre as implementações de protocolos.

Palavras Chave: Testes de Protocolos, TTCN, SGDT, SMDS, Testes de Conformidade, Testes de Interoperabilidade.

Abstract

The telecommunication networks evolution and the definition of an architecture for broadband etworks which allows service integration (B-ISDN) have the fostered development of the services and quipments industry. This development is evident in the definition of new services as a way of taking dvantage from the ATM technology that supports B-ISDN.

One of the services that rank high in this area is SMDS (*Switched Multi-megabit Data System*), whose haracteristics allow it to be easily adapted and used in a simple and transparent way as a support for hers applications and services.

Considering the adoption of SMDS as the means to access the broadband networks by a large number telecommunication operators, services and applications over SMDS are expected to be areas of anufacturers investments in the short medium term.

The success of an investment in the development of those products is sometimes determined by the nd of tests made during the development phase. At that time, the conformance and interoperability tests protocol implementations play a relevant or even vital role. Considering this fact, we decided to modify evaluation tool for communications network performace and quality of service, the TGDS - Traffic enerator and Detector System, in order to allow it to execute conformance and interoperability tests of otocol implementations that work over SMDS.

The changes on the TGDS include the development of a platform to interface it with a qualified tool ed to edit and define tests in TTCN (Tree and Tabular Combined Notation), the ITEX, and also the velopment of an interface to access SMDS. The result was an equipment that kept the previous testing cilities, but that is more powerful and flexible because it enables the defintion and the execution of tests scribed in TTCN by any operator and also features an interface to SMDS that allows it to execute tests er wider areas.

The concepts of conformance and interoperability of protocol implementations and the test quirements as well as test methodologies acording the ITU recommendations are presented in this work. ne TTCN and ITEX are also presented, as well as a summary description of the old TGDS architecture d operation. Features and potencials of SMDS and the SMDS interface board developed for TGDS are so described in this work. At last, some of the possible test configurations with the new TGDS are ustrated and commented.

Keywords: Protocol Testing, TTCN, TGDS, SMDS, Conformance Testing, Interoperability Testing.

Agradecimentos

A todos os que desde o princípio me acompanharam e acreditaram em mim.

Ao Prof. Doutor Eurico Carrapatoso, pela oportunidade que me concedeu, e por todo o apoio e orientação que nas fases mais decisivas foram determinantes para a realização do trabalho descrito nesta Dissertação.

À Isilda pela paciência, compreensão e ajuda que incansavelmente me dispensou ao longo de todo o trabalho.

Aos meus pais e irmã pela incessante força e motivação que me transmitiram durante esta fase da minha vida.

Ao Fernando pela sua colaboração preciosa na implementação da placa de interface *SMDS*.

Ao Ricardo pela disponibilidade e pelas sessões esclarecedoras que muito contribuíram para esta obra.

Special thanks to Mr. Gert Vermeer for his helpful collaboration.

Ao INESC pelas condições e pelo óptimo ambiente de trabalho que me ofereceu.

À JNICT pela bolsa que me concedeu.

Índice

ÍNDICE i

ÍNDICE DE FIGURAS v

ABREVIATURAS vii

1. INTRODUÇÃO 1

2. TESTE DE PROTOCOLOS 5

2.1 TESTES DE CONFORMIDADE 6

2.1.1. Tipos de Testes 8

2.1.1.1. Testes Básicos de Interligação 9

2.1.1.2. Testes de Capacidade 9

2.1.1.3. Testes de Comportamento 9

2.1.1.4. Testes de Determinação de Conformidade 10

2.1.2. Informação Extra para o Teste da Implementação do Protocolo 10

2.1.3. Fases do Processo de Avaliação de Conformidade 11

2.1.4. Conjuntos de Testes 12

2.1.5. Análise dos Resultados 13

2.1.6. Métodos de Teste 14

2.1.6.1. Método de Teste Local 19

2.1.6.2. Método de Teste Distribuído 20

2.1.6.3. Método de Teste Coordenado 21

2.1.6.4. Método de Teste Remoto 22

2.2. TESTES DE INTEROPERABILIDADE 24

3. ESPECIFICAÇÃO DE TESTES EM TTCN 27

3.1. *TTCN - TREE AND TABULAR COMBINED NOTATION* 28

3.1.1. Secção de Descrição do Conjunto de Testes 29

3.1.2. Secção de Declarações 31
3.1.3. Secção de Restrições 33
3.1.4. Secção de Comportamento Dinâmico 34
3.2. - O *ITEX* **39**

4. SGDT - SISTEMA DE GERAÇÃO E DETECÇÃO DE TRÁFEGO **43**

4.1. ARQUITECTURA E PRINCÍPIO DE FUNCIONAMENTO **44**
4.1.1. Estação de Trabalho 45
4.1.2. Unidade de Processamento de Tráfego 46
4.2. DEFINIÇÃO DE UMA SESSÃO **48**
4.2.1. Ligações de Pacotes 49
4.2.2. Ligações de Sinalização 49

5. *SMDS* - UM SERVIÇO DE SUPORTE **51**

5.1. *SMDS - SWITCHED MULTI-MEGABIT DATA SERVICE* **52**
5.2. ARQUITECTURA DE SUPORTE DO *SMDS* **53**
5.2.1. Interface Assinante-Rede 54
5.2.2 Protocolo de Interface *SMDS* 55
5.3. FILTRAGEM DE ENDEREÇOS **61**
5.4. CLASSES DE ACESSO **62**
5.5. ACESSO DE *LANS* AO *SMDS* **63**

6. TESTES DE CONFORMIDADE E INTEROPERABILIDADE COM O SGDT **67**

6.1. ADAPTAÇÃO DO SGDT **68**
6.2. INTERFACE SGDT/*SMDS* **71**
6.2.1. Interface com o Bloco Servidor da UPT 72
6.2.2. Bloco de Transmissão 74
6.2.3. Bloco de Recepção 78
6.2.4. Bloco de Acesso à Linha 81
6.2.5. Registo de Estado 82
6.2.6. Considerações sobre o *hardware* 82
6.3. *SOFTWARE* DE SUPORTE **84**
6.3.1. *Software* do Bloco Servidor 85
6.3.2. *Software* do Bloco Controlador 88
6.4. *SOFTWARE* DE ADAPTAÇÃO AO *ITEX* **90**

6.5. POSSÍVEIS AMBIENTES DE TESTE 93

7. CONCLUSÃO 99

REFERÊNCIAS R - 1

APÊNDICE A A - 1

APÊNDICE B B - 1

Índice de Figuras

Fig. 2.1 - Modelo do processo de avaliação de conformidade 12

Fig. 2.2 - Estrutura de um Conjunto de Teste 13

Fig. 2.3 - Identificação da estrutura dos equipamentos 14

Fig. 2.4 - Primitivas de Serviço e Unidades de Dados 16

Fig. 2.5 - Modelo da arquitectura de teste 16

Fig. 2.6 - Serviço de nível N-1 17

Fig. 2.7 - Método de Teste de Retorno 18

Fig. 2.8 - Método de Teste Transversal 19

Fig. 2.9 - Método de Teste Local 20

Fig. 2.10 - Método de Teste Distribuído 21

Fig. 2.11 -Método de Teste Coordenado 22

Fig. 2.12 - Método de teste Remoto 23

Fig. 2.13 - Arquitectura para testes de interoperabilidade 26

Fig. 3.1 - Descrição de um conjunto de testes 31

Fig. 3.2 - Declaração de diferentes entidades e tipos 32

Fig. 3.3 - Tabelas das restrições impostas na transmissão 34

Fig. 3.4 - Comportamento dinâmico de um caso de teste 35

Fig. 3.5 - Evolução da estrutura em árvore 36

Fig. 3.6 - Invocação de outra árvore 36

Fig. 3.7 - Comportamento por omissão 37

Fig. 3.8 - Exemplo do comportamento dinâmico de um caso de teste 38

Fig. 4.1 - Arquitectura do SGDT 45

Fig. 4.2 - Estrutura funcional da *WS* 46

Fig. 4.3 - Arquitectura de uma UPT 47

Fig. 5.1 - Arquitectura de suporte da rede *SMDS* 54

Fig. 5.2 - Configurações de acesso ao *SMDS* 55

Fig. 5.3 - Interligação de equipamentos terminais pelo *SMDS* 55

Fig. 5.4 - Estrutura do *SIP* 56

Fig. 5.5 - Formato de uma *L3_PDU* 57

Fig. 5.6 - Formato de uma *L2_PDU* 58

Fig. 5.7 - Formato de uma trama *PLCP E1* 60

Fig. 5.8 - Formato das tramas *E1* 61

Fig. 5.9 - Ligação de uma *LAN* a um acesso *DQDB* 63

Fig. 5.10 - Acesso de cliente *ATM* ao *SMDS* 64

Fig. 6.1 - Métodos de execução de teste com o código gerado 69

Fig. 6.2 - Estrutura global da Interface *SMDS* 71

Fig. 6.3 - Estrutura do acesso às *fifos* 73

Fig. 6.4 - Construção da trama *PLCP* 75

Fig. 6.5 - Informação dos *bits Si* dos *slots 0* 77

Fig. 6.6 - Contrução da trama de transmissão *E1* 78

Fig. 6.7 - Determinação de *B1* e *G1* 80

Fig. 6.8 - Octeto *VALID* 81

Fig. 6.9 - Registo de Estado 82

Fig. 6.10 - Recepção das *L2_PDUs* 86

Fig. 6.11 - Reconstrução das *L3_PDUs* 87

Fig. 6.12 - Sinal do *driver* 89

Fig. 6.13 - Sinalização para recepção 90

Fig. 6.14 - Relação dos blocos funcionais 91

Fig. 6.15 - Método de Teste de Retorno com o SGDT 93

Fig. 6.16 - Método de Teste Coordenado com o SGDT 94

Fig. 6.17 - Método de Teste Remoto com o SGDT 95

Fig. 6.18 - Método de Teste Local com o SGDT 95

Fig. 6.19 - Método de Teste Transversal com o SGDT 96

Fig. 6.20 - Monitorização de Tráfego com o SGDT 97

Abreviaturas

AAL	*ATM Adaptation Layer*
AFAP	*As Fast As Possible*
AHDL	*ALTERA Hardware Description Language*
ANSI	*American National Standards Institute*
ASN.1	*Abstract Syntax Notation One*
ASP	*Abstract Service Premitive*
ATC	*Abstract Test Case*
ATM	*Asynchronous Transfer Mode*
ATS	*Abstract Test Suite*
BIT	*Basic Interconnection Test*
BNF	*Backus-Naur Form*
BSS	*Broadband Switching System*
CBDS	*Connectionless Broadband Data Service*
CPE	*Customer Premises Equipment*
CRC	*Cyclic Redundancy Check*
CSU	*Channel Service Unit*
DAMS	*Dynamically Adaptable Multi-service System*
DCN	*Data Communication Network*
DQDB	*Distributed Queue Dual Bus*
DSP	*Digital Signal Processor*
DSU	*Data Service Unit*
DXI	*Data Exchange Interface*
EPLD	*Erasable Programmable Logic Device*
ESPRIT	*European Specific Program for Research in Information and Technology*
ETSI	*European Telecommunications Standards Institute*
EWOS	*European Workshop on Open Systems*
ICI	*Interexchange Carrier Interface*
IEC	*International Electrotechnical Commission*
ISO	*International Standards Organization*
ISSI	*Inter-Switching System Interface*

ITU	*International Telecommunication Union*
IUT	*Implementation Under Test*
L2_PDU	*Level 2 Protocol Data Unit*
L3_PDU	*Level 3 Protocol Data Unit*
LAN	*Local Area Network*
LLC	*Logic Link Control*
LT	*Lower Test*
MAC	*Medium Access Control*
MAN	*Metropolitan Area Network*
MANTIS	*Metropolitan Area Network for Integrated Services*
MMI	*Man-Machine Interface*
NIST	*National Institute of Standards and Technology*
NP	*Network Performance*
NPL	*National Physical Laboratory*
NRZ	*Non Return to Zero*
OS	*Operations System*
OSI	*Open System Interconnection*
PAL	*Programmable Array Logic*
PCO	*Point of Control and Observation*
PCTR	*Protocol Conformance Test Report*
PDU	*Protocol Data Unit*
PICS	*Protocol Implementation Conformance Statment*
PIXIT	*Protocol Implementation Extra Information for Testing*
PLCP	*Physical Layer Convergence Procedures*
PLL	*Phase Locked Loop*
QoS	*Quality of Service*
RDIS-BL	Rede Digital com Integração de Serviços - Banda Larga
RPC	*Remote Procedure Call*
SAP	*Service Access Point*
SCS	*System Conformance Statment*
SCTR	*System Conformance Test Report*
SGDT	*Sistema de Geração e Detecção de Tráfego*
SIP	*SMDS Interface Protocol*
SMDS	*Switched Multi-megabit Data Service*
SNI	*Subscriber Network Interface*
SS	*Switching System*

SUT	*System Under Test*
TCP	*Test Coodenation Procedure*
TMP	*Test Management Protocol*
TTCN	*Tree and Tabular Combined Notation*
UCOL	*Ultrawideband Coherent Optical LAN*
UNI	*User Network Interface*
UPT	Unidade de Processamento de Tráfego
UT	*Upper Tester*
VME	*Versa Module Europe*
WAN	*Wide Area Network*
WS	*WorkStation*

Capítulo 1

Introdução

O progresso das sociedades está cada vez mais dependente das tecnologias de comunicação. O aparecimento de novos serviços e aplicações que necessitam de transferir grandes quantidades de informação a elevados débitos vem exigir cada vez mais largura de banda que as redes tradicionais não podem oferecer. Surgem então as redes de banda larga que garantem a transferência da informação a elevados débitos e permitem integrar um conjunto de novos serviços, para além e possibilitarem interligar redes locais localizadas em zonas geograficamente distantes.

Um dos serviços desenvolvidos para permitirem o acesso de clientes às facilidades das redes de banda larga foi o *SMDS - Switched Multi-megabit Data Service.* Este serviço foi particularmente desenvolvido para usufruir das potencialidades da tecnologia *DQDB - Distributed Queue Dual Bus* - e da semelhança do mecanismo de transporte com o da tecnologia *ATM* (*Asynchronous Transfer Mode*) na qual se baseia a RDIS-BL (Rede Digital com Integração de Serviços - Banda Larga).

A adopção do *SMDS* como serviço para transferência de informação entre *LANs* tem ganho cada vez mais adeptos entre as operadoras de telecomunicações, não só por ter sido o primeiro serviço de banda larga a ser normalizado, mas também por garantir a transferência dessa informação de uma forma transparente. É por esse facto que o *SMDS* pode ser utilizado como plataforma para suporte de um conjunto de outros serviços e aplicações.

A definição de uma arquitectura para redes de comunicação que permita a integração de serviços (RDIS e RDIS-BL) foi um passo marcante para o desenvolvimento a que actualmente se assiste na indústria de serviços e aplicações de telecomunicações. É neste cenário que surge a necessidade de se dispor de um conjunto de ferramentas que permita a essa indústria evoluir de uma forma coerente e eficaz. Esta evolução é dependente de inúmeros factores entre os quais se salientam a qualidade dos produtos oferecidos e, cada vez mais, o tempo que a própria indústria demora a disponibilzar produtos que satisfaçam as exigências dos clientes. Ferramentas que desempenham um papel relevante na fase de desenvolvimento desses serviços e equipamentos são as que dão ao fabricante a segurança de que os

produtos em que investiram vão operar correctamente, quando ligados a outros sistemas teoricamente compatíveis. Este é o principal objectivo de ferramentas que executam testes de conformidade e de interoperabilidade sobre implementações de protocolos.

As ferramentas para executar estes tipos de testes devem dispor de mecanismos que permitam ao operador definir o conjunto de testes através de notações ou linguagens dedicadas. Uma dessas notações é o *TTCN - Tree and Tabular Combined Notation*, que por ter sido normalizada aparece como uma das mais utilizadas na descrição de testes de conformidade. Associadas a esta notação várias aplicações apareceram de forma a facilitar a edição e a conversão dos testes descritos em *TTCN* em testes executáveis.

O Sistema de Geração e Detecção de Tráfego (SGDT) é uma ferramenta que foi desenvolvida com o propósito de efectuar testes de desempenho e de avaliação da qualidade de serviço em redes exprimentais construídas no âmbito dos projectos *ESPRIT UCOL* (*Ultra Wideband Coherent Optical Lan*) e *DAMS* (*Dynamically Adaptable Multi-Service System*). Este sistema, embora tendo uma arquitectura distribuída e aberta, apresenta algumas limitações sobretudo em termos de flexibilidade e de compatibilidade ao nível da definição do conjunto de testes a efectuar. Exigia também uma interface específica de acesso à rede que se pretendia avaliar e as interfaces que existiam não eram normalizadas.

A construção de uma ferramenta, com base no Sistema de Geração e Detecção de Tráfego, que permitisse executar testes de conformidade e de interoperabilidade, mantendo todavia a possibilidade de avaliar o desempenho, era o principal objectivo do trabalho que deu origem a esta dissertação.

Outros objectivos eram a definição e desenvolvimento de uma interface de acesso ao serviço *SMDS*, que possibilitasse ao SGDT e executar testes de implementações de protocolos desenvolvidos para operar sobre esse serviço. Um último objectivo era dotar a ferramenta com mecanismos que possibilitassem utilizar uma interface normalizada de forma a poder interpretar e executar testes descritos em *TTCN* por qualquer operador de testes e analisar como poderia ser usada num ambiente de teste.

Esta dissertação está organizada em 7 capítulos.

No capítulo 2 são apresentados os conceitos de conformidade e de interoperabilidade sendo evidenciada a sua importância. O conjunto de entidades e de requisitos envolvidos na definição e execução de testes de conformidade são apresentados de acordo com as recomendações *ITU X.290* a *X.294*, bem como uma série de metodologias de execução desses testes. Neste capítulo é também abordado o conceito de interoperabilidade segundo vários autores e apresentada uma metodologia para monitorização da interoperabilidade de dois sistemas.

No capítulo 3 é apresentada uma notação utilizada para descrever testes de conformidade, o *TTCN - Tree and Tabular Combined Notation*. São identificadas as diversas secções que constituem cada conjunto de testes e descritas a estrutura e a dinâmica da notação. É ainda apresentado o *ITEX*, que não é mais do que a ferramenta seleccionada para efectuar a edição e o processamento dos conjuntos de teste

em *TTCN*, e para a qual foram desenvolvidas uma série de funções de adaptação de forma a permitir a execução dos testes com o SGDT.

No capítulo 4 são descritos a arquitectura e o funcionamento do SGDT - Sistema de Geração e Detecção de Tráfego. São identificados e descritos os vários blocos que constituem esta ferramenta, para se fazer seguidamente uma abordagem descritiva da definição dos testes nesta ferramenta e das metodologias de teste possíveis.

No capítulo 5 é apresentado aquele que é o primeiro serviço normalizado para redes de banda larga, o *SMDS*. Este serviço foi o escolhido para garantir a comunicação entre o equipamento de teste que se pretende desenvolver e os sistemas sobre os quais os testes de conformidade e de interoperabilidade serão executados. As características destes serviço bem como a arquitectura que a rede de suporte deve possuir são descritas neste capítulo. De igual modo são inumeradas algumas das vantagens que a utilização deste serviço proporciona aos seus assinantes, nomeadamente a fácil adaptação às redes com tecnologia *ATM*. São também focados processos de conectar clientes de outras redes à rede com o *SMDS*.

No capítulo 6 são descritas a arquitectura e funcionalidades do SGDT. Numa fase inicial é apresentada a placa de interface do SGDT ao *SMDS*, desenvolvida com o intuito de transferir transparentemente tráfego entre o SGDT e o acesso ao *SMDS*. Uma abordagem descritiva da arquitectura e das funcionalidades dessa placa, bem como do formato da informação transferida entre o SGDT e a placa de interface *SMDS*, foi adoptada neste capítulo. Esta placa de interface *SMDS* foi elaborada no âmbito do programa *ESPRIT* no projecto *MANTIS (Metropolitan Area NeTwork for Integrated Services)*, e permite ao SGDT executar tanto testes de desempenho e de avaliação da qualidade de serviço, como testes de conformidade e de interoperabilidade de implementações que operem sobre o *SMDS*.

Nesse capítulo é, também, apresentada a estrutura do *software* desenvolvido para o SGDT de forma a completar as funcionalidades do *SMDS* não existentes na placa de interface. Além disso, descrevem-se ainda as implementações das diversas entidades envolvidas nos testes que são imprescindíveis para adaptar o código *ANSI C* gerado com o *ITEX* ao *SGDT*.

Este capítulo termina com a apresentação de várias configurações de teste em que a ferramenta desenvolvida pode executar testes de conformidade e de interoperabilidade

Por fim, no capítulo 7, são tiradas algumas conclusões, relativas ao trabalho realizado e às dificuldades que surgiram, e apontam-se novas metas para uma possível evolução do SGDT como ferramenta de teste.

Capítulo 2

Teste de Protocolos

O progresso tecnológico na área das redes de comunicações tem permitido a definição de novos serviços que exigem o desenvolvimento de novos protocolos ou de novas versões de protocolos já existentes. É também notório que esta evolução tem sido acompanhada por um crescimento da quantidade de produtos que implementam normas *OSI* (*Open System Interconnection*) [Cairns92][Vermee93]. No entanto, para garantir o perfeito interfuncionamento dos sistemas não basta que os fabricantes sigam as normas: as implementações dos protocolos devem ser testadas. Os testes executados sobre protocolos podem ser de diferentes tipos, sendo a metodologia usada definida em função do objectivo do teste. Alguns exemplos de tipos de testes incluem [RACE92]:

- testes de conformidade, para verificar se a implementação de um determinado protocolo obedece às especificações das normas;
- testes de interoperabilidade, para determinar se duas implementações estão a interagir correctamente;
- testes de desempenho, para avaliar as características e o desempenho da implementação sob teste;
- testes de robustez, para avaliar a capacidade de determinada implementação recuperar de situações de erro quando estas ocorrem.

Todos estes tipos de testes são imprescindíveis ao funcionamento perfeito de um sistema aberto; na prática, quando um operador se prepara para instalar produtos *OSI*, ele quer ter a garantia de que esses produtos irão de facto comunicar correctamente quando ligados entre si. Esta garantia só existe se esses produtos tiverem sido previamente testados, em relação à especificação *OSI* dos protocolos que implementam e em relação ao outro sistema com o qual se pretende que comuniquem.

A verificação dos protocolos face à especificação das normas é conseguida por meio dos testes de conformidade, enquanto que o funcionamento conjunto de sistemas distintos é garantido com base nos testes de interoperabilidade. Com estes dois tipos de testes (conformidade e interoperabilidade) é

possível determinar se, num sistema que engloba dois ou mais equipamentos *OSI*, estes vão operar correctamente quando ligados entre si. Grandes esforços têm sido desenvolvidos pela *ISO* (*International Standards Organisation*) e pela *ITU* (*International Telecommunication Union*), no sentido de normalizar a estrutura e a metodologia de testes de conformidade, e pelos *ETSI* (*European Telecommunications Standards Institute*) e *EWOS* (*European Workshop on Open Systems*) com objectivos idênticos relativamente aos testes de interoperabilidade [ETSI92][EGCT92].

A definição de uma recomendação para a metodologia de testes de conformidade foi conseguida (*ITU X.290* a *X.294*); no entanto, quanto à norma que estruture e defina uma metodologia para testes de interoperabilidade, não se chegou ainda a um consenso entre as várias organizações envolvidas.

2.1 Testes de Conformidade

Com o objectivo de fomentar mercados abertos e de permitir a competição entre os construtores de equipamentos é necessário garantir uma certa coerência ou conformidade protocolar por parte desses equipamentos. A normalização dos diversos protocolos tornou-se indispensável para o correcto funcionamento de uma rede. Surgiu também a necessidade de se estabelecer, com alguma prioridade, um acordo internacional de métodos e procedimentos de teste. Os métodos acordados devem fornecer, simultaneamente, uma boa base técnica, de modo a garantir uma elevada eficiência dos processos de teste, e serem provenientes de um consenso de forma a que os resultados possam ser reconhecidos em todo o mundo. É nesta área que a *ISO* e a *ITU* têm tido um papel relevante, já que definiram um conjunto de normas e recomendações, que especificam uma metodologia e uma linguagem para descrição de testes de conformidade - o *TTCN (Tree and Tabular Combined Notation)* -, aplicáveis a implementações desenvolvidas segundo o Modelo *OSI*.

As ferramentas de avaliação da conformidade de determinado protocolo verificam se as suas implementações obedecem à especificação.

Os testes de conformidade são descritos nas normas *ISO/IEC* 9646 e nas recomendações X.290 a X.294 da *ITU*. As normas *ISO/IEC* e as recomendações *ITU* constituem um grupo de regras que, em conjunto, definem o comportamento da comunicação de sistemas abertos.

Os requisitos de conformidade com as normas *OSI* e recomendações *ITU* podem ser caracterizados sob várias prespectivas: imprescindibilidade, positividade e operabilidade.

Quanto à imprescindibilidade, os requisitos poderão ser obrigatórios, condicionais ou opcionais;

- obrigatórios - se têm que ser observados em todos os casos;
- condicionais - se têm que ser observados apenas quando é aplicado um conjunto de condições à especificação;

- opcionais - se podem ser seleccionados de modo a satisfazer uma implementação, contando que quaisquer requisitos aplicáveis a essa opção sejam observados.

Quanto à positividade, o conjunto de condições impostas pelas normas pode ser declarado positivamente, se menciona o que deve ser feito, ou negativamente, se determina o que não deve ser feito.

Por fim, quanto à operabilidade, os requisitos das normas *ISO/ITU* podem ser divididos em dois grupos: os estáticos e os dinâmicos.

Os requisitos estáticos de conformidade são os que definem quais as diferentes combinações das funcionalidades implementadas que são permitidas no sistema que se pretende em conformidade. Por outras palavras, restringem o conjunto de funcionalidades implementadas de modo a facilitar o teste.

Os requisitos dinâmicos de conformidade são os que especificam quais os comportamentos observáveis que são consentidos pelas normas *ISO* e recomendações *ITU*, durante as comunicações. Estes requisitos constituem o núcleo da especificação de cada protocolo *OSI*, ao definir o maior conjunto de funcionalidades que uma implementação em conformidade pode ter. Pode dizer-se que um sistema apresenta uma conformidade dinâmica, em determinado instante da comunicação, se o seu comportamento pertence ao conjunto de todas as funcionalidades permitidas pela especificação do protocolo *OSI,* de um modo consistente com os requisitos estáticos [X290_92].

Para avaliar a conformidade de uma implementação em particular (*IUT - Implementation Under Test*), é necessário conhecer quais as funcionalidades e opções que foram implementadas, para que o teste seja elaborado de acordo com as capacidades do sistema. Essa informação imprescindível para o teste é denominada por Declaração de Conformidade da Implementação do Protocolo (*PICS - Protocol Implementation Conformance Statement*). Numa *PICS* deve haver distinção entre as diferentes categorias de informação que ela pode conter:

– informação relacionada com os requisitos obrigatórios, opcionais e condicionais estáticos do protocolo em si;

– informação sobre os mesmos requisitos para as dependências entre os níveis adjacentes.

Se um conjunto de protocolos *OSI*, relacionados entre si, é implementado num sistema é necessária uma *PICS* para cada protocolo. Nesta situação, será também necessária uma Declaração de Conformidade do Sistema (*SCS - System Conformance Statment*) para identificar todos os protocolos para os quais foram definidas Declarações de Conformidade das Implementações.

De um modo mais objectivo, pode dizer-se que uma implementação ou sistema está em conformidade se satisfaz os requisitos estáticos e dinâmicos consistentes com as funcionalidades definidas nas Declarações de Conformidade das Implementações, para cada protocolo declarado na Declaração de Conformidade do Sistema.

O objectivo principal dos testes de conformidade é, como já foi referido, aumentar a portabilidade dos sistemas, assegurando que, em sistemas reais distintos, diferentes implementações de determinado

protocolo funcionem correctamente quando ligadas entre si. Este tipo de testes apresenta, no entanto, algumas limitações.

Mesmo que duas implementações estejam em conformidade com a mesma especificação do protocolo *OSI*, podem não conseguir operar quando ligadas uma à outra. O sucesso do interfuncionamento de dois ou mais sistemas reais é, no entanto, mais provável se as diferentes implementações estiverem em conformidade com o mesmo subconjunto de recomendações e normas do que no caso contrário. Por isso, durante a preparação dos diferentes sistemas para o funcionamento conjunto, deve ser feita uma comparação entre as *SCSs* e as *PICSs* desses sistemas. Se as *PICSs* indicarem que diferentes versões ou subgrupos das normas foram implementadas, as diferenças e as suas implicações no funcionamento devem ser determinadas.

2.1.1. Tipos de Testes

No sentido de avaliar a conformidade de um dado sistema ou implementação, a norma [X290_92] define 4 tipos de testes:

- testes básicos de interligação (*BIT - Basic Interconnection Test*), que mostram à primeira vista que a implementação sob teste está conforme;
- testes de capacidade, que verificam se os comportamentos observáveis de uma *IUT* estão de acordo com os requisitos de conformidade estática e com as funcionalidades definidas nas *PICSs*;
- testes de comportamento, que permitem verificar, dentro das capacidades da *IUT*, os requisitos de conformidade dinâmica especificados pelas recomendações;
- testes de determinação da conformidade, que provam exaustivamente a conformidade da implementação sob teste com determinados requisitos de modo a obter um veredicto final para o teste.

Estes tipos de testes, como é evidente, não têm todos o mesmo peso na avaliação da conformidade de um sistema. De um modo geral, todos os tipos de teste são executados, podendo, no entanto, não ser tão exaustivos como lhes seria exigido.

Destes tipos de testes alguns foram normalizados, o que significa que foram definidos para avaliar os requisitos das implementações e, como tal, são os seleccionados para constituirem conjuntos de testes sobre essas implementações. A cada um desses conjuntos de testes normalizados, definidos pelos próprios implementadores e por outras organizações, é chamado conjunto abstracto de testes (*ATS - Abstract Test Suite*).

2.1.1.1. Testes Básicos de Interligação

Este tipo de testes é bastante limitado, pois tem apenas por objectivo avaliar se existe conformidade suficiente para que a interligação seja possível.

Os Testes Básicos de Interligação são apropriados para a detecção de casos severos de não conformidade e são usados como factor de decisão para que os testes seguintes, de capacidade e de comportamento, sejam ou não executados. É com os *BITs* que são verificados os endereços e outros pormenores relacionados com o ambiente de teste. Os utilizadores das implementações utilizam-nos para verificar se uma dada implementação poderá ser usada para comunicar com outras em conformidade.

Por outro lado, os Testes Básicos de Interligação não podem, por si próprios, garantir a conformidade de uma implementação, nem determinar as causas de falhas de comunicação.

2.1.1.2. Testes de Capacidade

Os testes de capacidade são testes restritos de cada requisito estático de um protocolo e/ou especificação. Servem para apurar quais das funcionalidades referidadas na Declaração de Conformidade da Implementação podem ser observadas e para verificar se essas funcionalidades são válidas no que respeita aos requisitos estáticos de conformidade.

Este tipo de testes é apropriado não só para verificar se as funcionalidades da implementação sob teste são consistentes com os requisitos de conformidade estática, e para verificar a consistência da Declaração de Conformidade da Implementação do Protocolo com a *IUT*, mas também para, em conjunto com os teste de comportamento, ser a base para as exigências da conformidade.

No entanto, os testes de capacidade, sozinhos, são insuficientes para satisfazer as exigências de conformidade de um fornecedor de implementações. Este tipo de testes mostra-se também inapropriado para testar, em pormenor, o comportamento associado a cada funcionalidade, quer tenha sido implementada ou não, e para determinar problemas ocorridos durante o tempo de uso.

2.1.1.3. Testes de Comportamento

Os testes de comportamento analisam a implementação tão minuciosamente como se ela estivesse em funcionamento. Estes testes são realizados sobre todo o domínio dos requisitos dinâmicos de conformidade especificados no protocolo.

Este tipo de testes é apropriado para verificar a sequência de reacções da implementação sob teste. Quando executados em conjunto com os testes de capacidade, constituem parte da estrutura de avaliação do processo de conformidade. São, no entanto, inadequados para a resolução de problemas relacionados

com a durabilidade do sistema, ou quando outro teste indica que, embora os testes de comportamento tenham sido satisfeitos, o sistema continua sem conformidade.

2.1.1.4. Testes de Determinação de Conformidade

Este tipo de testes, por não estar normalizado, pode ser eventualmente utilizado para complementar os testes normalizados do processo de avaliação de conformidade, permitindo investigar o comportamento de uma implementação sob teste no que respeita a determinados requisitos de conformidade.

Tanto a arquitectura como o método de teste serão seleccionados especificamente atendendo aos requisitos que se pertendem testar. Essa selecção poderá, no entanto, não ser a mais adequada para o teste da generalidade dos outros requisitos, para além de que poderá até não ser permitida por um conjunto abstracto de testes normalizados definidos para a avaliação de determinados requisitos da implementação.

Os testes de determinação de conformidade podem incluir meios de controlo da ocorrência de eventos e de estados internos do Sistema sob Teste (*SUT - System Under Test*). Deste modo é possível testar aspectos do protocolo que não são testáveis com os conjuntos abstractos de testes normalizados.

Os testes de determinação de conformidade são apropriados para completar os objectivos do teste que, sendo identificados nos propósitos dos testes normalizados, não permitem a inclusão de novos casos nos Conjuntos Abstractos de Testes por causa de limitações do método de teste escolhido ou da não testabilidade do requisito de conformidade.

De igual modo, são um método apropriado para investigar problemas encontrados na execução de *ATSs* normalizados. Poderão fornecer um resultado positivo ou negativo em situações estritamente limitadas e previamente identificadas. No entanto, este tipo de testes não é suficiente para avaliar a conformidade total de uma dada implementação.

2.1.2. Informação Extra para o Teste da Implementação do Protocolo

Com o objectivo de testar a implementação de um dado protocolo, o laboratório de teste vai necessitar de informação relacionada com essa implementação e com o ambiente de teste, bem como daquela que é fornecida pelas Declarações de Conformidade da Implementação do Protocolo. A Informação Extra para o Teste da Implementação do Protocolo (*PIXIT - Protocol Implementation Extra Information for Testing*) é fornecida pelo implementador do protocolo a testar, ao preencher um formulário que lhe é dado pelo laboratório.

Para cada conjunto abstracto de testes (*ATS*) existe uma Informação Extra associada à implementação do protocolo a testar. São o especificador do *ATS*, o executor do teste e o laboratório de

teste que contribuem para o desenvolvimento da proforma da Informação Extra para o Teste da Implementação do Protocolo.

A *PIXIT* contém dados que aumentam a precisão da informação disponibilizada nas Decarações de Conformidade das Implementações (*PICS*). Por exemplo, se essas Declarações de Conformidade definem os domínios dos valores possíveis para determinado parâmetro ou *timer*, a Informação Extra contém os valores específicos que serão usados em todos os casos de teste, para além dos definidos para testar os diferentes valores para um dado parâmetro ou *timer*. Outro tipo de informação, que é fornecida nessa Informação Extra, diz respeito ao Sistema sob Teste, como por exemplo, a informação sobre os endereços e a realização de analisadores do nível superior da implementação a testar. Na *PIXIT* existe também informação para ajudar a determinar quais das funcionalidades definidas nas Declarações de Conformidade da Implementação são suportadas, testáveis e não testáveis; contudo, a Informção Extra não pode contradizer essas declarações.

2.1.3. Fases do Processo de Avaliação de Conformidade

O processo de avaliação da conformidade é o processo completo de acompanhamento de todas as actividades de teste necessárias para que se considere que uma implementação ou sistema está em conformidade com as normas [X290_92][X293_92][X294_92].

Este processo decorre em três fases: preparação para o teste, operações de teste e produção de relatórios dos testes.

A fase de preparação para os testes consiste em: produzir as Declarações de Conformidade do Sistema (*SCSs*), as Declarações de Conformidade da Implementação do Protocolo (*PICSs*) e a Informação Extra para o Teste da Implementação do Protocolo (*PIXIT*); escolher o Método Abstrato de Teste e o Conjunto Abstracto de Testes, com base nesses documentos; e por fim preparar o Sistema a testar para o teste.

Na fase das operações de teste são feitas: uma revisão da conformidade estática, conduzida por uma análise das Declarações de Conformidade da Implementação no que respeita aos requisitos estáticos de conformidade; a selecção e parametrização dos testes a executar; por fim, a execução de um ou mais conjuntos de testes parametrizados, produzidos como resultado das fases de selecção e parametrização, e a gravação das sequências observadas nas diferentes situações de teste, bem como outras informações relevantes para a avaliação da conformidade.

A execução destes conjuntos de teste parametrizados envolve a configuração do equipamento, permitindo que ocorram variações no protocolo entre o Sistema sob Teste e o equipamento de teste, sem que este perca o controlo dessas variações. Os testes executados nesta fase são os Testes Básicos de Interligação, os Testes de Capacidade e os Testes de Comportamento. Esta fase termina com a análise dos resultados e segue-se, posteriormente, a fase de realização do relatório de teste.

Um esquema representativo das fases do processo de avaliação da conformidade é apresentado na Fig.2.1.

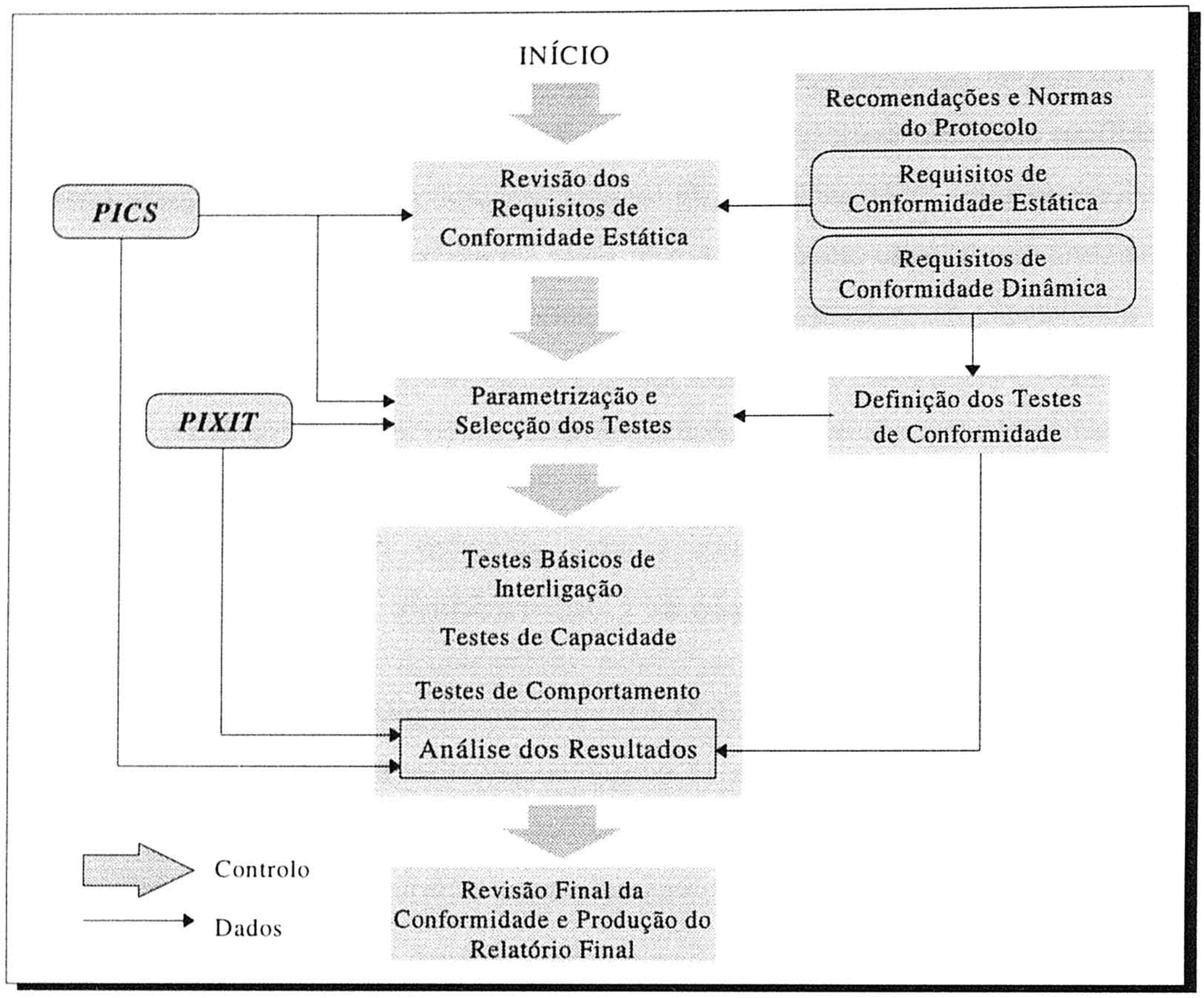


Fig. 2.1 - Modelo do processo de avaliação de conformidade

2.1.4. Conjuntos de Testes

Os conjuntos de testes têm uma estrutura hierárquica na qual os casos de teste ocupam um lugar de destaque. Cada um destes casos de teste tem um propósito, tal como verificar se uma implementação sob teste dispõe de determinada funcionalidade, ou se se comporta como é esperado (Fig.2.2).

Dentro de cada conjunto de testes existem grupos distintos, que são utilizados para estabelecer uma relação lógica entre os diferentes casos de teste, e podem, de um modo geral, ajudar no planeamento, desenvolvimento ou execução do conjunto de testes.

Os casos de teste, por sua vez, podem ser modularizados usando subdivisões: os passos de teste.

Os acontecimentos ou eventos do teste são as unidades indivisíveis dentro dos passos de teste; por exemplo, a transferência, recepção ou transmissão, de uma unidade de dados através de uma primitiva de serviço de ou para uma implementação em teste.

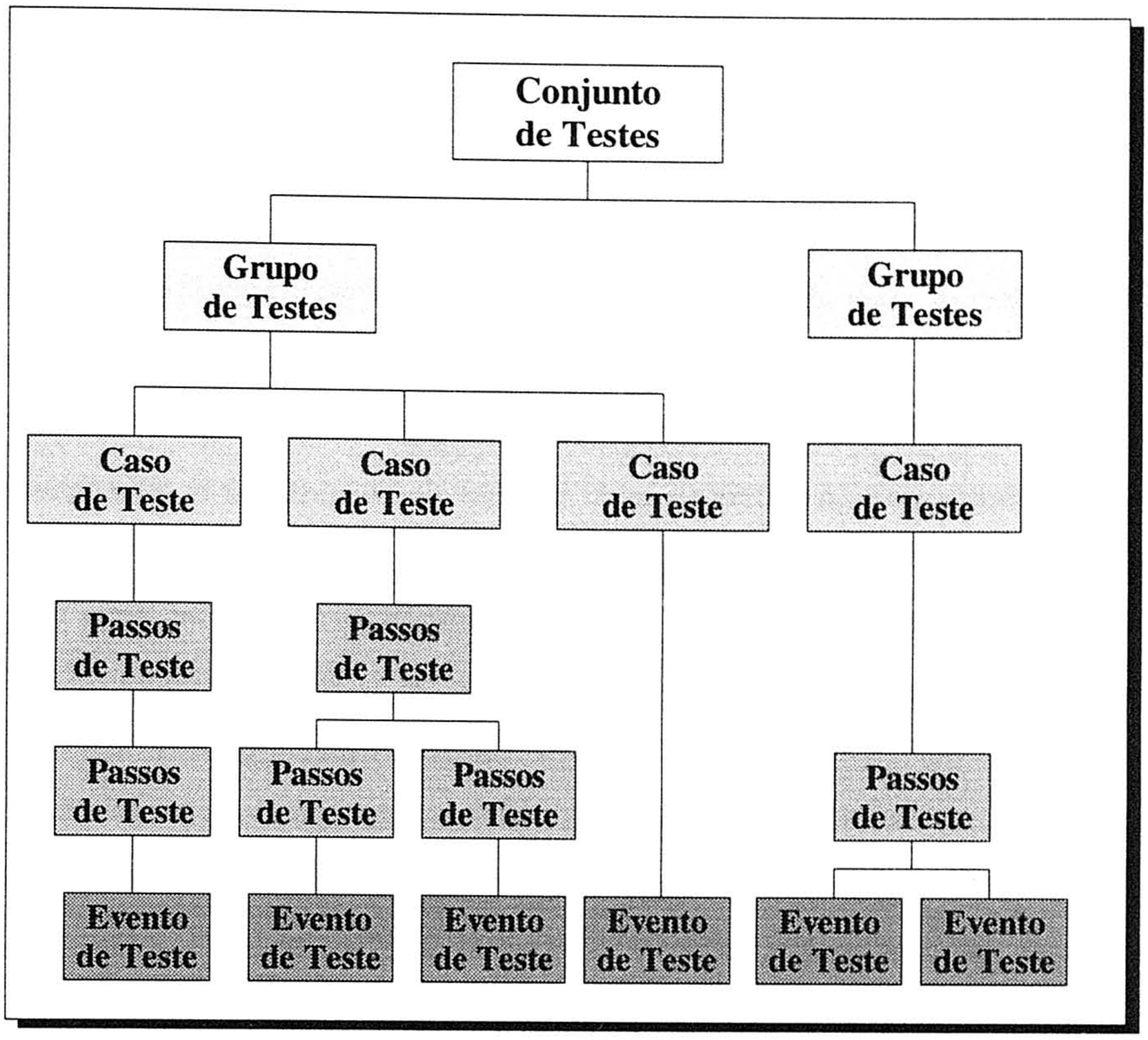


Fig. 2.2 - Estrutura de um Conjunto de Teste

2.1.5. Análise dos Resultados

Os resultados do processo de avaliação da conformidade de um sistema são documentados em Relatórios do Teste de Conformidade do Protocolo (*PCTR - Protocol Conformance Test Report*), em Relatórios do Teste de Conformidade do Sistema (*SCTR - System Conformance Test Report*), e num conjunto de mensagens da evolução do teste [X290_92][X294_92]. Estes documentos podem ser fornecidos ao cliente do laboratório de teste, podendo a informação ser mantida confidencial ou não, de acordo com o tipo de contrato estabelecido entre as duas partes.

Os resultados dos testes são séries de eventos ocorridos durante a execução de casos de teste, incluindo todas as entradas e saídas da implementação sob teste, nos diferentes pontos de controlo e observação.

Um resultado previsto é aquele que foi definido no caso de teste, isto é, as situações que ocorrem durante a execução de determinado teste coincidem com a sequência das situações de teste definidas no caso de teste. Os resultados previstos podem incluir situações não identificadas, resultando, no entanto, sempre na atribuição de um veredicto a esse caso de teste. Os veredictos de um teste podem ser: *passou*, *falhou* ou *inconclusivo*.

O veredicto *passou* significa que o resultado do teste está de acordo com os requisitos de conformidade em que os propósitos do teste se basearam e que é válido no que respeita às normas *ISO* e recomendações da *ITU*.

O veredicto *falhou* significa que o resultado do teste demonstra a não-conformidade com, pelo menos, um dos requisitos de conformidade em que os propósitos do teste assentaram. Ou então que foi detectado, pelo menos, um evento de teste inválido no que respeita às normas *ISO* e recomendações da *ITU*.

O veredicto *inconclusivo* é atribuído ao teste quando os eventos, ou a falta deles, não permitem classificar o teste com os veredictos *passou* ou *falhou*.

2.1.6. Métodos de Teste

Os sistemas reais, que contêm implementações de protocolos segundo o modelo *OSI*, aparecem com um leque muito vasto de configurações, o que exige que os modos de observação e de controlo do comportamento desses sistemas também sejam variados.

Antes de se optar por determinado método de teste, deve ser feito um estudo sobre as características do sistema a testar, bem como da sua implementação. Nesta fase, é muito importante identificar se o Sistema sob Teste é um equipamento terminal ou de retransmissão, e quais os níveis desse Sistema que usam protocolos *OSI*. Segundo este critério a recomendação [X290_92] distingue quatro tipos de configurações diferentes (Fig.2.3).

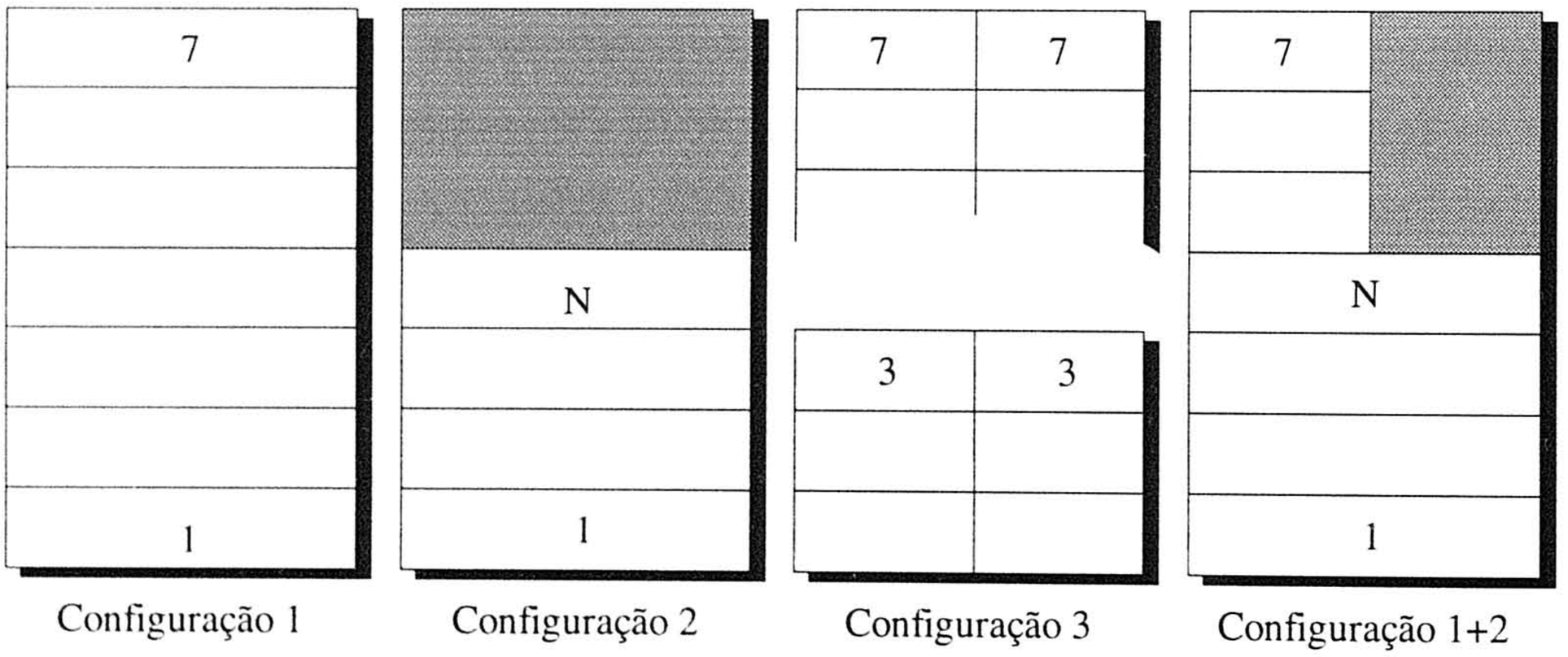


Fig. 2.3 - Identificação da estrutura dos equipamentos

Na configuração 1, todos os 7 níveis do sistema são implementados de acordo com protocolos *OSI*; na configuração 2, apenas nos níveis 1 até N se usam protocolos *OSI*; na configuração 3, os sistemas ou usam protocolos *OSI* nos níveis 1 a 3 (sistema de retransmissão de rede), ou nos níveis 1 a 7 (sistema de

retransmissão de aplicação); e, por fim, na configuração 1+2, os níveis 1 até N são implementados segundo protocolos *OSI*, mas a partir do nível N a implementação poderá ser feita, ou não, segundo esses protocolos.

A implementação sob teste (*IUT*) é a parte do sistema aberto que queremos estudar, com os testes de conformidade, e deve ser a implementação de um ou mais protocolos *OSI*, num dado nível ou em níveis adjacentes.

Em sistemas implementados segundo as configurações 1 e 2, as implementações são chamadas de protocolo único, se só um dos protocolos da totalidade do Sistema é testado, ou de protocolos múltiplos, caso sejam testados, em simultâneo, um grupo de protocolos adjacentes. Numa dada implementação sob teste o seu nível mais elevado é denominado por N_s e o mais baixo por N_i. No caso de uma implementação de protocolo único $N_s = N_i$; esta notação será, também, usada para referir níveis quer no Sistema sob Teste quer no equipamento de teste.

Os Métodos Abstractos de Teste são descritos em função das saídas observáveis e das entradas que podem ser controladas numa implementação sob teste. Mais concretamente, a descrição de um Método Abstracto de Teste inclui a identificação dos pontos mais próximos da implementação sob teste através dos quais são efectuados o controlo e a observação.

As normas que especificam os protocolos segundo o modelo *OSI* definem os eventos permitidos num protocolo em função de unidades de dados (*PDU - Protocol Data Unit*) e de primitivas de serviço (*ASP - Abstract Service Primitive*), ambas em níveis acima e abaixo da implementação sob teste.

Os protocolos de um dado sistema comunicam entre si através dos Pontos de Acesso aos Serviços (*SAP - Service Access Point*) oferecidos pelos níveis adjacentes, e por meio de primitivas abstractas de acesso a esses serviços (as *ASPs*). Assim, se determinado protocolo de nível N pretende enviar dados para um dos protocolos de níveis vizinhos (N-1 ou N+1), ele utiliza as primitivas de acesso que esses níveis fornecem. As *ASPs* não são mais do que representações, abstractas e independentes da implementação, da interacção entre o utilizador de um serviço *OSI* e o disponibilizador desse serviço. Podem envolver a troca bidireccional de informação mas, neste caso, dispõem de um sistema de identificação do sentido do fluxo dessa informação.

As *PDUs* são as unidades de dados que são enviadas, por determinado protocolo de nível N de um sistema, para o protocolo, do mesmo nível, de outro sistema. Sempre que se pretende transmitir uma dada *PDU* de nível N para outro sistema, envia-se essa *PDU* pelos níveis inferiores, através das *ASPs* disponibilizadas por esses níveis, até ao nível físico. O processo de recepção no sistema destino é, em tudo, semelhante ao de transmissão até que a *PDU* alcance o nível N.

De um modo mais genérico, as *ASPs* estão associadas ao deslocamento vertical da informação enquanto que as *PDUs* estão associadas a uma movimentação horizontal dessa informação (Fig.2.4) [*Bernd 93*].

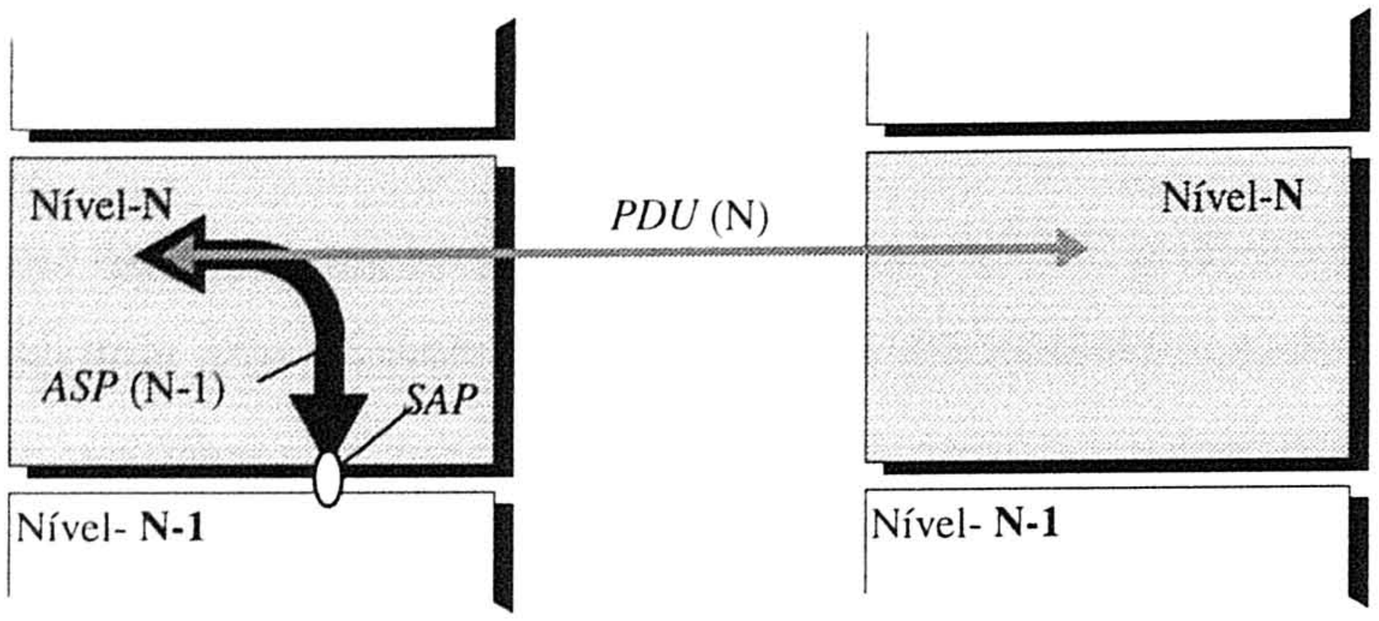


Fig. 2.4 - Primitivas de Serviço e Unidades de Dados

Assim, o comportamento da implementação de um protocolo de nível N é definido em termos de primitivas de serviço dos níveis N e N+1, incluindo as últimas as unidades de dados de nível N. No caso da implementação englobar mais do que um protocolo, o comportamento pode ser definido em função das primitivas de serviço acima e abaixo da implementação, incluindo as unidades de dados dos protocolos envolvidos nessa implementação.

O princípio de funcionamento de um sistema avaliador da conformidade de determinado protocolo consiste em isolar a implementação a testar, excitá-la com o equipamento de teste e analisar o seu comportamento (Fig.2.5). Este modelo é o ponto de partida para o desenvolvimento de Métodos Abstractos de Teste.

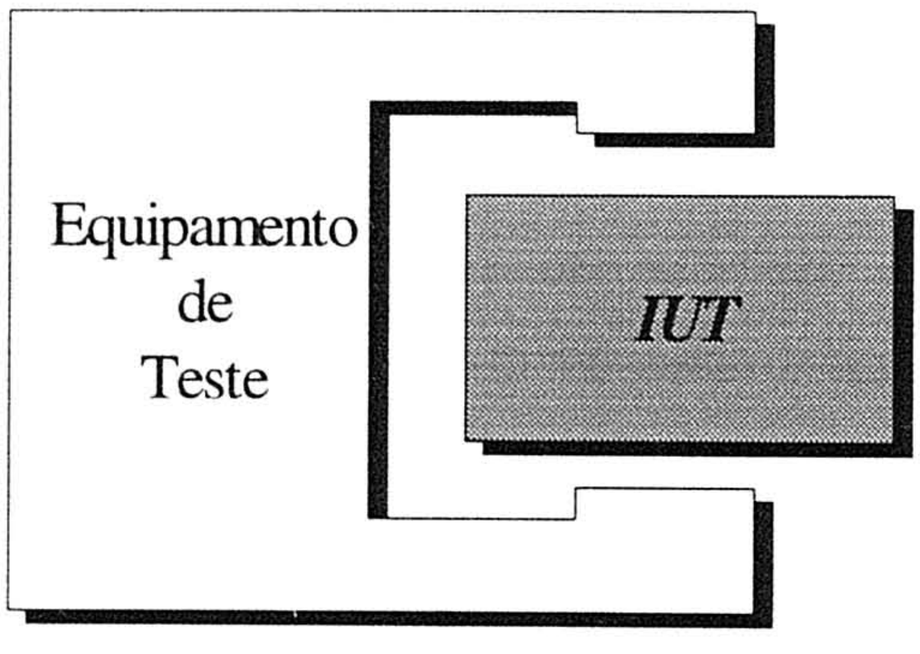


Fig. 2.5 - Modelo da arquitectura de teste

A acção deste analisador conceptual envolve dois grupos de interacções com a *IUT*: por cima e por baixo. Estas podem, teoricamente, ser controladas e observadas a partir de vários pontos distintos. Cada um destes Pontos de Controlo e Observação *(PCO - Point of Control and Observation)* é indentificado por três factores: os limites do serviço, o conjunto de primitivas abstractas de serviço e unidades de dados protocolares de teste e, por fim, se os *PCOs* estão localizados no Sistema sob Teste ou no equipamento de teste.

Os Pontos de Controlo e Observação podem ser modelizados por duas filas: uma de saída, para controlar os eventos de teste associados à transmissão de informação para a implementação sob teste, e uma de entrada, para observar os eventos associados à recepção de informação proveniente da implementação sob teste. As actividades das primitivas abstractas de serviço abaixo da implementação sob teste podem ser observadas e controladas por actividades paralelas no equipamento de teste, ligadas a essas primitivas, através de um serviço de nível inferior (Fig.2.6). Neste caso, é portanto assumido que o serviço fornecido, que garante os níveis inferiores, é suficientemente fiável para que o controlo e a observação possam ser feitos remotamente. Este serviço pode garantir vários níveis ou somente o meio físico.

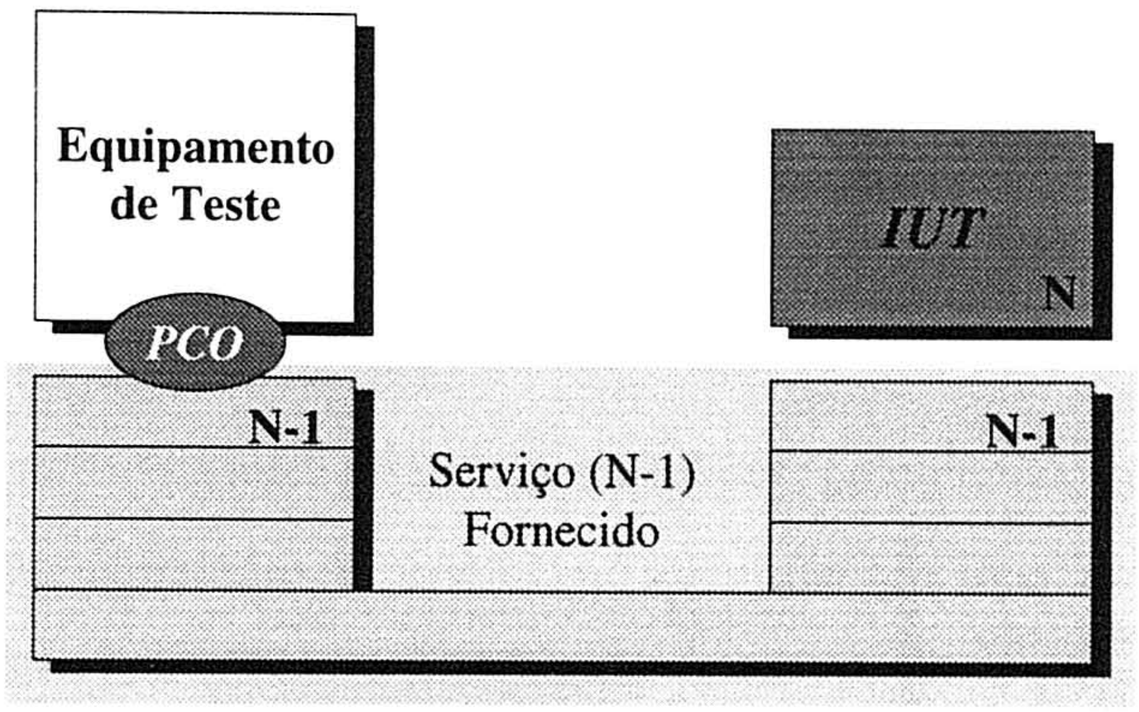


Fig. 2.6 - Serviço de nível N-1

Poderá, no entanto, acontecer que a actividade da primitiva de serviço acima da implementação sob teste não seja controlável ou observável; neste caso a actividade é chamada invisível.

A definição de um Método Abstracto de Teste utiliza dois analisadores de teste, um analisador superior (*UT - Upper Tester*) e outro inferior *(LT - Lower Tester*). O *UT* é o analisador que, durante a execução do teste, controla e observa a interface superior da implementação sob teste, de acordo com o método definido. O *LT* é o analisador que, durante a execução do teste, controla e observa, indirectamente, o limite inferior da implementação através dos serviços de nível inferior.

A cooperação entre os analisadores inferior e superior é orientada por Procedimentos de Coordenação de Teste (*TCP - Test Coordenation Procedure*).

Os conjuntos abstractos de teste para os diferentes métodos variam no modo como são especificados os requisitos de teste nos Procedimentos de Coordenação de Teste. Em alguns casos é possível definir um protocolo de gestão do teste que fornece a coordenação entre os analisadores superior e inferior. Noutros casos, não é possível especificar os mecanismos a usar para os Procedimentos de Coordenação de Teste, sendo apenas possível descrever os requisitos a atingir.

Pode então definir-se o Método Abstracto e Teste como sendo um método que descreve uma arquitectura abstracta de teste que consiste num analisador superior, num analisador inferior e em

procedimentos de coordenação de teste, e nas suas relações com o sistema de teste e com o Sistema sob Teste. Cada Método Abstracto de Teste determina os pontos de controlo e de observação bem como as primitivas abstractas e as unidades de dados que devem ser usadas num determinado caso de teste.

Com base na identicação das várias entidades envolvidas num Método Abstracto de Teste é possível definir um conjunto de métodos para testar implementações de protocolos, de acordo com o tipo de sistemas a que pertencem.

No caso do teste de um sistema de retransmissão (de rede ou de aplicação) a norma [X290_92] define somente dois métodos: o de retorno e o transversal.

O primeiro, o Método de Teste de Retorno, inclui dois Pontos de Controlo e Observação numa das sub-redes, em Pontos de Acesso ao Serviço externos ao nível de retransmissão que se pretende testar (Fig.2.7). Para protocolos orientados às ligações, as duas ligações usadas para o teste são ligadas entre si no outro lado do nível de retransmissão, podendo esta ligação ser feita nesse nível ou na outra sub-rede. Se os protocolos não forem orientados às ligações, as unidades de dados enviadas são retornadas pela segunda sub-rede e contêm o endereço correspondente ao segundo Ponto de Controlo e Observação.

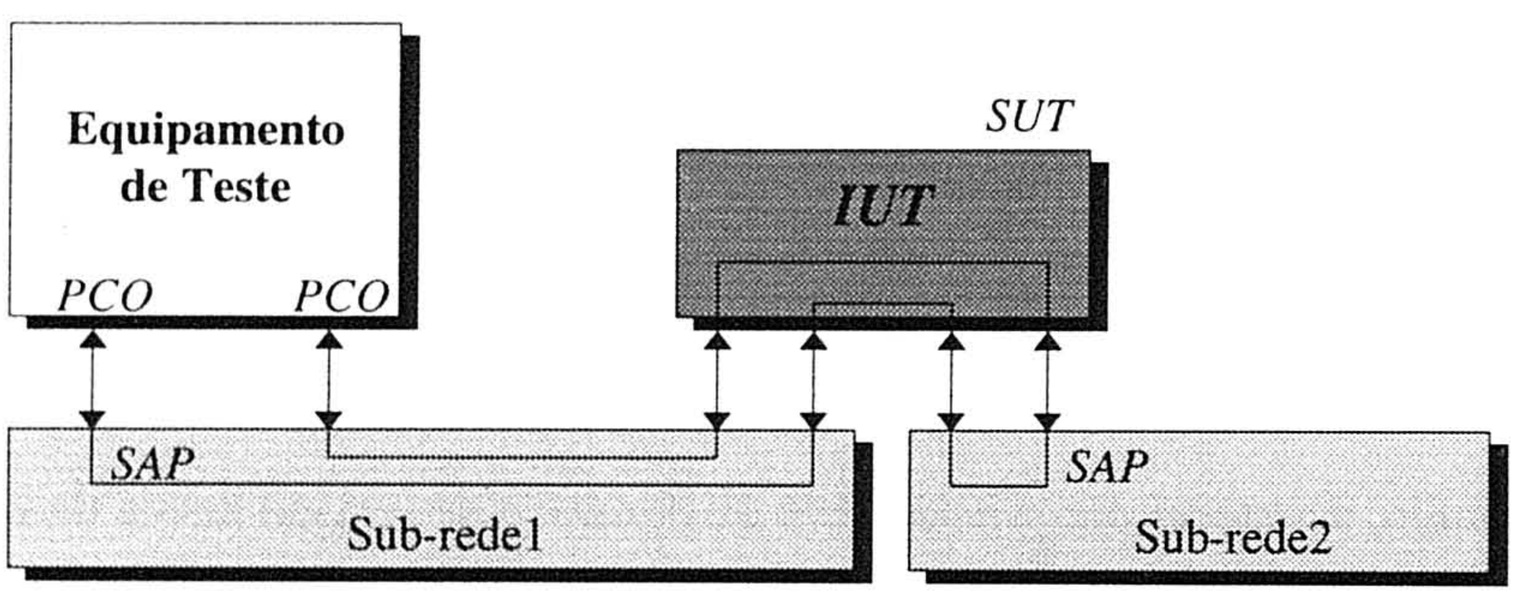


Fig. 2.7 - Método de Teste de Retorno

O Método de Teste Transversal é utilizado para testar sistemas de retransmissão a partir das duas sub-redes (Fig.2.8). Este método utiliza dois equipamentos de teste, cada um com um Ponto de Controlo e Observação, ligados à implementação do sistema de retransmissão através dos Pontos de Acesso em Serviço de cada sub-rede. Deste modo é possível observar o comportamento da implementação nas duas sub-redes, o que não era possível com o Teste de Retorno.

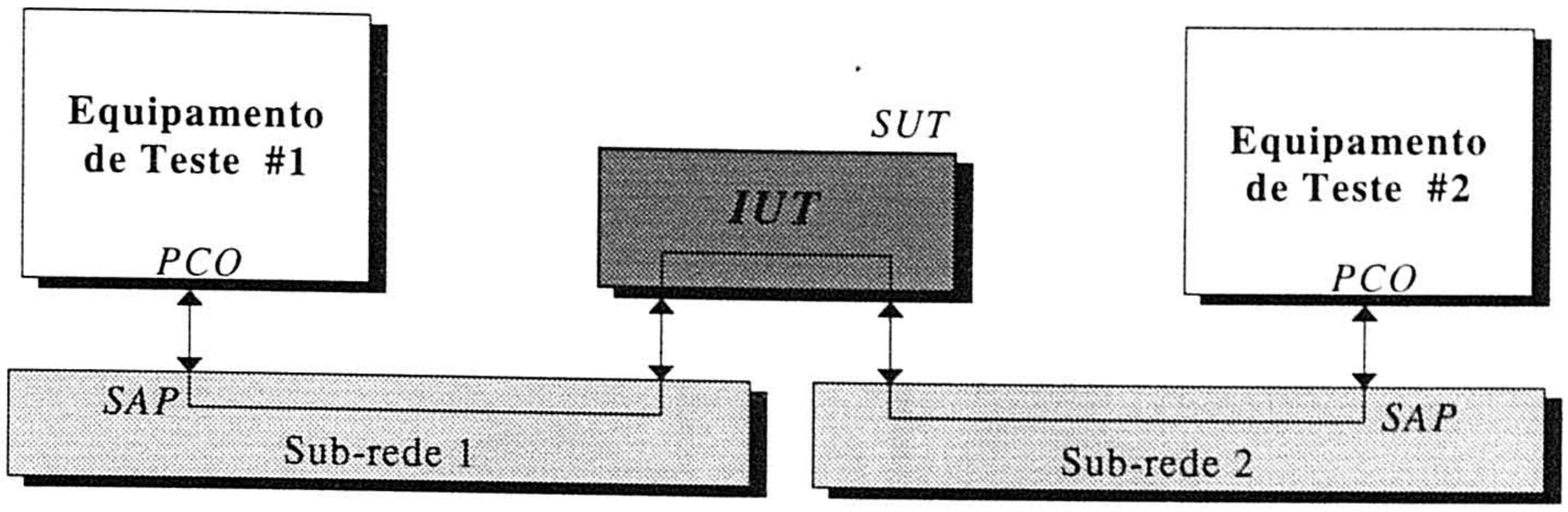


Fig. 2.8 - Método de Teste Transversal

Para as implementações sob teste existentes em equipamentos terminais existem quatro tipos de teste: dois que usam o Ponto de Controlo e Observação entre o *UT* e a implementação a testar, os Métodos de Teste Local e Distribuído, e dois que usam o Ponto de Controlo e Observação abaixo do *LT*, os Métodos de Teste Coordenado e Remoto. Todos eles efectuam o controlo e a observação das primitivas de serviço abaixo da implementação, e das unidades de dados de e para a implementação, por meio do *LT* localizado no equipamento de teste.

Uma ou mais destas quatro variantes podem ser escolhidas pelo cliente implementador do protocolo a testar, de acordo com as possibilidades do sistema, no que respeita a interfaces acessíveis, e com o grau de exposição pretendida para a implementação em teste.

2.1.6.1. Método de Teste Local

Neste tipo de testes existem dois Pontos de Controlo e Observação: um abaixo do *LT* e outro acima do limite superior da implementação. As situações de teste no Ponto de Controlo e Observação do *LT* são especificadas em função somente das primitivas de serviço (*ASPs*) do nível N_{i-1}, e/ou das unidade de dados (*PDU*s) dos níveis compreendidos entre N_i e N_s (sendo N_i o nível inferior da implementação e N_s o seu nível superior). Quanto às situações de teste no Ponto de Controlo e Observação do *UT*, são especificadas apenas em função das *ASPs* de nível N_s.

Este método de teste é ilustrado na Fig.2.9, onde se pode ver que o *UT* está localizado no sistema de teste. Este método só é viável se no limite superior da *IUT* existir uma interface de *hardware* onde será ligado o equipamento de teste. Os requisitos para os procedimentos de coordenação do teste são especificados no conjunto abstracto de testes mas são executados localmente no equipamento de teste.

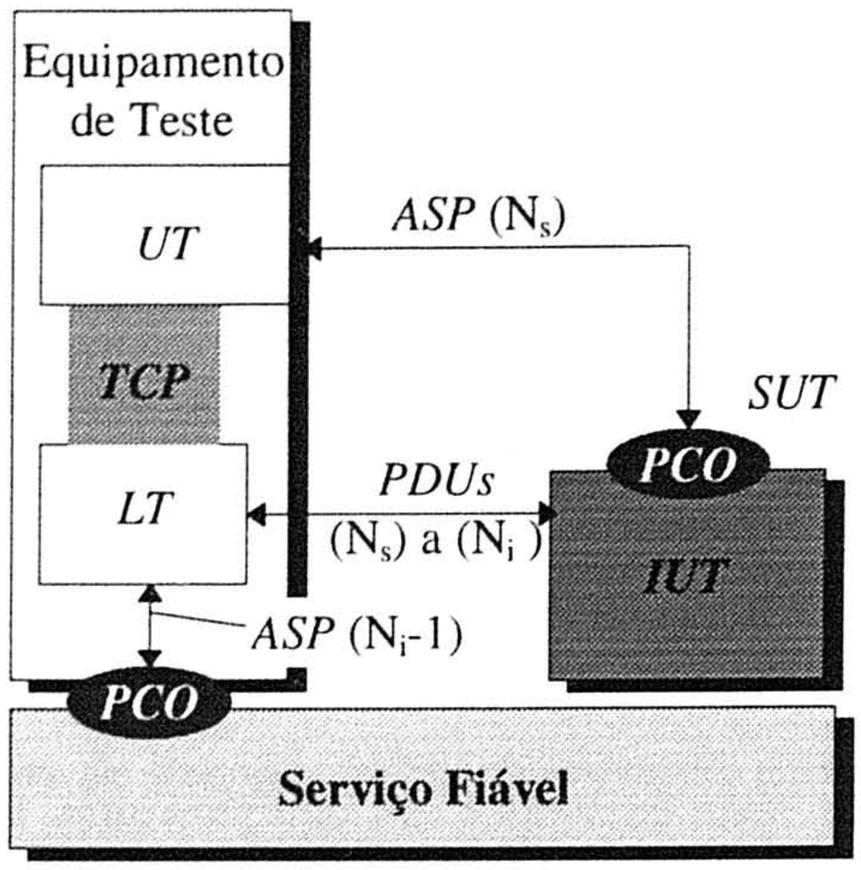


Fig. 2.9 - Método de Teste Local

No caso da implementação ser de protocolos múltiplos e para o teste de um protocolo de nível N_x, situado entre os níveis N_i e N_s dessa implementação, as diferentes situações de teste são especificadas em função de primitivas de serviço do nível N_x-1 e de unidades de dados dos níveis N_x até N_s, acima do serviço de nível N_x-1 do Sistema sob Teste.

O Método de Teste Local é aplicado só em sistemas que disponham de duas interfaces de *hardware* (por exemplo em *transceivers*).

2.1.6.2. Método de Teste Distribuído

Tal como no Método de Teste Local, o Método de Teste Distribuído também utiliza dois Pontos de Controlo e de Observação: um abaixo do *LT* e o outro acima do limite superior da implementação sob teste. Neste método as situações de teste são em tudo semelhantes ao anterior. No entanto, nesta configuração de teste, a interface superior da implementação tanto pode ser uma interface homem/máquina como a interface com uma linguagem de programação, que ser usada com os propósitos de teste.

No Método de Teste Distribuído o *UT* está localizado no Sistema Sob Teste (Fig.2.10).

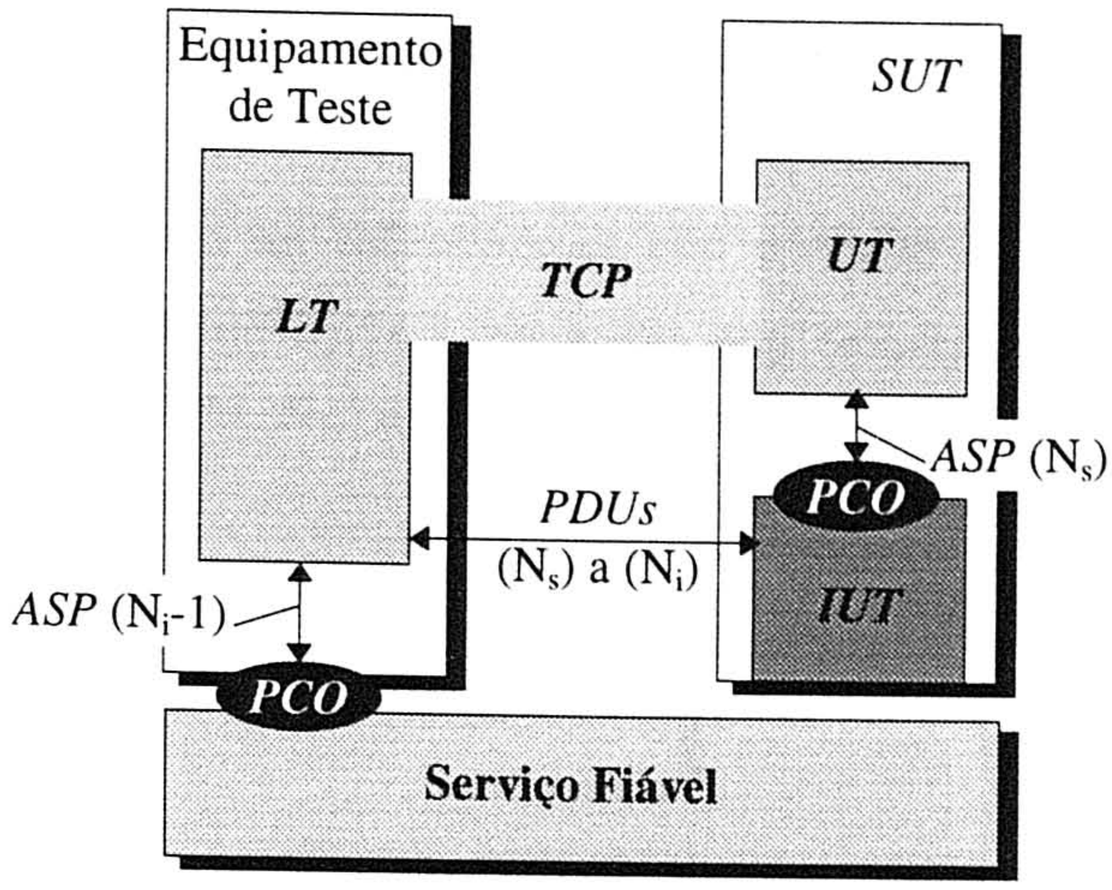


Fig. 2.10 - Método de Teste Distribuído

No caso da interface superior da implementação ser uma interface para um operador humano, esse deve seguir as indicações dos Procedimentos de Coordenação do Teste especificados no conjunto abstracto de testes. Caso seja uma interface com uma linguagem de programação, o *UT* é realizado em *software* e os requisitos dos Procedimentos de Coordenação do Teste são satisfeitos por ambos os analisadores, o *LT* e o *UT*.

Também com este método podemos testar implementações de protocolos múltiplos. Para se testar um protocolo do nível N_x, entre os níveis N_i e N_s da *IUT*, as situações de teste são especificadas em função das primitivas de serviço de nível N_s, acima da implementação, das primitivas de serviço dos níveis compreendidos entre o nível N_x-1 e o N_i na implementação sob teste, e das unidades de dados dos níveis N_x a N_s, acima do serviço de nível N_x-1 do Sistema sob Teste.

2.1.6.3. Método de Teste Coordenado

O Método de Teste Coordenado utiliza apenas um Ponto de Controlo e de Observação, abaixo do *LT*. Neste método de teste, os Procedimentos de Coordenação do Teste são realizados por meio de Protocolos de Gestão de Testes (*TMP - Test Management Protocols*). Neste caso a *UT* está localizada no Sistema sob Teste e não é mais do que a implementação de um determinado Protocolo de Gestão de Testes.

Um Protocolo de Gestão de Testes é aplicável só a um determinado conjunto abstracto de testes (*ATS*), e pode não ser aplicável a outros *ATSs* para este método de teste. Os Protocolos de Gestão são implementados no Sistema sob Teste imediatamente acima da *IUT*.

Num teste deste tipo, as situações de teste no Ponto de Controlo e de Observação são especificadas em função das primitivas de serviço do nível N_s-1 e/ou das unidades de dados dos níveis N_i a N_s, em

conjunto com as *PDUs* de gestão (*TMPDUs*). As unidades dos dados de gestão são enviadas, transparentemente, através da implementação sob teste do e para o *UT*.

As primitivas de serviço do nível N_s não são referidas na especificação do teste no conjunto abstracto de testes, pois não se assume que exista um serviço no limite superior da implementação (Fig.2.11).

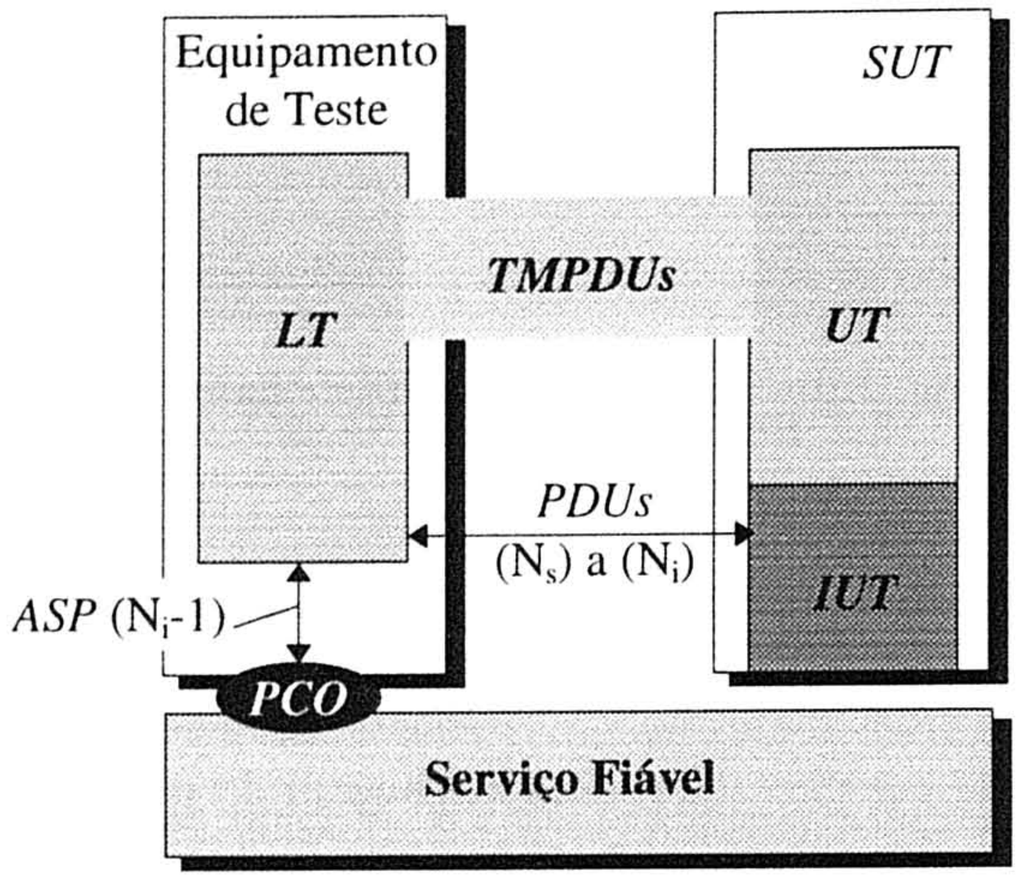


Fig. 2.11 -Método de Teste Coordenado

No Método de Teste Coordenado são adicionados casos de teste ao Conjunto Abstracto de Testes com o propósito de testar se o *UT* está em conformidade com os requisitos das especificações dos protocolos de gestão; no entanto, tais casos não contribuem para a avaliação da conformidade da implementação sob teste.

Se a *IUT* é de protocolos mútiplos, o teste de um protocolo de nível N_x é efectuado a partir de casos de teste especificados em função das primitivas de serviço do nível N_x-1, das unidades de dados dos níveis N_x até ao N_s e das unidades e dados de gestão. O Protocolo de Gestão de Testes opera, neste caso, sobre o serviço do nível N_s.

2.1.6.4. Método de Teste Remoto

Este tipo de teste, bem como o anterior, só contém um Ponto de Observação e de Controlo: abaixo do *LT*. É utilizado, na maioria dos casos, em situações em que não é possível controlar nem observar o limite superior da implementação sob teste.

No Método de Teste Remoto não existe um *UT*, mas as funções deste podem ser desempenhadas pelo Sistema sob Teste (*SUT*), através de procedimentos de coordenação do teste (*TCP*). Neste método, os requisitos desses procedimentos de coordenação são também expressos no conjunto abstracto de

testes, mas não é garantido que o *SUT* consiga corresponder a todas as exigências dos requisitos. Assim sendo, é possível que apareçam requisitos no conjunto de testes que não sejam realizáveis.

Os casos de teste no Ponto de Controlo e de Observação são especificados em função das primitivas de serviço do nível N_i-1 e/ou das unidades de dados dos níveis N_i a N_s. Também neste método de teste não são usadas as *ASPs* de nível N_s, uma vez que não existe o acesso ao limite superior da implementação.

De modo a superar a falta de informação acerca do comportamento acima da implementação em teste, os requisitos do comportamento do Sistema sob Teste são, sempre que necessário, especificados em função das primitivas de serviço do nível N_i-1 ou das unidades de dados dos níveis N_i até ao N_s, que podem ser observadas e controladas pelo *LT*. Através do *LT*, são aplicadas todas as combinações e sequências de casos de teste necessárias para controlar o Sistema sob Teste, de modo a colocá-lo nos estados, a partir dos quais se pretende testar o seu comportamento [X291_21]. Mas mesmo deste modo é possível que alguns dos requisitos definidos no conjunto abstracto de testes não sejam realizáveis.

A principal diferença entre este método e os restantes é o facto ser possível controlar mas não observar os eventos de teste acima da implementação sob teste.

Para o caso do teste de um protocolo de nível N_x de uma implementação de protocolos múltiplos, as situações de teste são especificadas em função das *ASPs* de nível N_x-1 e das *PDUs* de níveis N_x+1 a N_s (Fig.2.12).

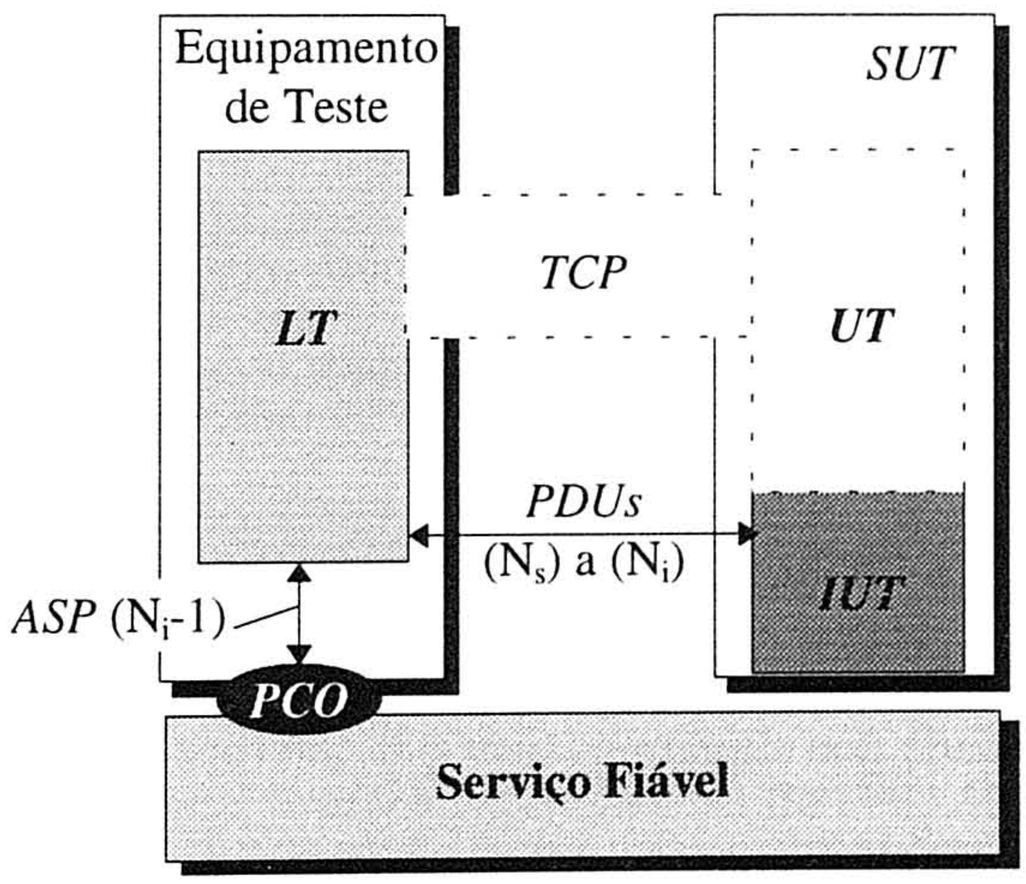


Fig. 2.12 - Método de teste Remoto

2.2. Testes de Interoperabilidade

Com o aparecimento de cada vez mais produtos desenvolvidos e implementados segundo o modelo *OSI*, as metodologias de teste para verificação da conformidade desses produtos têm evoluído cada vez mais. Por este facto, os produtos *OSI* são cada vez mais os seleccionados pelos utilizadores [Cairns92]. No entanto, como já foi referido, os testes de conformidade, por si sós, não são suficientes para garantir o sucesso da operabilidade conjunta de várias implementações. Numa perspectiva mais prática, podem até ser identificados vários factores passíveis de impedirem a interoperabilidade de implementações conformes [ETSI92][Haven92][Lenzin92][Vermee93].

O primeiro desses factores provém das próprias especificações e normas que nem sempre são suficientemente exactas. Assim, é possível haver, entre os vários implementadores, mais do que uma interpretação dessas normas, de que podem resultar problemas de funcionamento conjunto.

Por outro lado, as mesmas especificações e normas podem apresentar opções. De um modo geral, as implementações garantem as funções básicas para o funcionamento do sistema, mas pode também existir um conjunto opcional de funções que, podendo ser implementadas ou não, podem conduzir a problemas de falta de interoperabilidade.

Ainda no que respeita aos testes de conformidade, é referido em [X290_92] que a complexidade dos protocolos *OSI* torna, muitas vezes, os testes exaustivos pouco práticos e muito caros. Logo, são apenas efectuadas partes desses testes seleccionadas por ordem de importância. Assim, a conformidade de uma dada implementação pode não ser total, o que possivelmente se traduzirá na falta de operabilidade do sistema.

E, por fim, o facto dos testes de conformidade não avaliarem o desempenho nem a robustez de uma implementação. As implementações podem estar em conformidade mas, na prática, podem não dispor das características de desempenho suficientes.

Existe, portanto, um conjunto de situações que os testes de conformidade não conseguem detectar, e que poderão, eventualmente, condicionar o funcionamento de um sistema aberto. É com o objectivo de superar estas lacunas da conformidade que surgem os testes de interoperabilidade [Haven92] [SPAG88] [Vermee93].

Dado que ainda não foi acordada uma definição de interoperabilidade, são várias as propostas formuladas para este conceito. De um modo geral há sempre que considerar duas visões possíveis, a do operador e a do cliente. É possível que de cada ponto de vista existam ainda diferentes opiniões, mas já foi possível identificar um conceito para cada uma delas. Assim, para o operador, a interoperabilidade envolve dois ou mais sistemas abertos independentes, em conformidade com as suas especificações *OSI*, e o sistema de comunicação. Nesta perspectiva, existe interoperabilidade se os sistemas conseguem comunicar e funcionar em conjunto quando ligados entre si.

Segundo os clientes, a interoperabilidade existe se dois ou mais sistemas, ligados entre si, realizam todas as funções cooperativas definidas numa especificação comum [Dwyer93][Vermee91].

É de notar que dois ou mais sistemas não conformes podem interoperar.

Os testes de interoperabilidade permitem observar a capacidade de implementações funcionarem conjuntamente, num ambiente o mais próximo possível do real.

A execução deste tipo de testes requer que as implementações a testar tenham sido previamente testadas quanto à conformidade com as suas especificações. Os testes de interoperabilidade são sempre executados como complemento dos testes de conformidade, pois é com base nos resultados dos últimos que são elaborados documentos que indicam quais as partes das *PICSs* que foram ou não testadas, bem como quais os locais onde foram ou não detectados erros ou falhas. É a partir desta informação que são efectuados os testes de interoperabilidade.

Ao contrário dos testes de conformidade, os testes de interoperabilidade não testam uma implementação face a uma recomendação ou norma *OSI*, mas duas ou mais implementações ligadas entre si.

Embora não exista ainda uma recomendação *ITU* ou norma *ISO* que defina e especifique os testes de interoperabilidade entre sistemas, os fabricantes de equipamentos de teste são de um modo geral unânimes quanto à arquitectura a usar para este tipo de teste [Takaha92][Vermee93][Vermee93a].

A configuração básica para testes de interoperabilidade engloba as implementações a testar e a rede de interligação dessas implementações. As implementações devem ser estimuladas pelas aplicações de funcionamento normal ou, caso estas não estejam disponíveis, por uma aplicação que consiga desempenhar esse tipo de funções, possivelmente um sistema de teste.

Podem ser identificados três modos de teste de interoperabilidade [Vermee93a]. Por ordem decrescente de proximidade com a configuração real, e por ordem crescente de observação e/ou controlo, as arquitecturas de teste podem ser: sem monitorização, com monitorização passiva e com monitorização activa. Na configuração sem monitorização não existem meios de controlar ou observar a comunicação de dados fora das *IUTs*. Na configuração com monitorização passiva, é instalado um sistema de monitorização em todas as ligações entre duas implementações sob teste. Este sistema apenas permite a observação dos dados trocados entre essas duas *IUTs*. E por fim, na configuração com monitorização activa, é possível não só observar mas também controlar os dados da comunicação. Assim é possível gerar erros e analisar as respostas de cada uma das implementações (Fig.2.13).

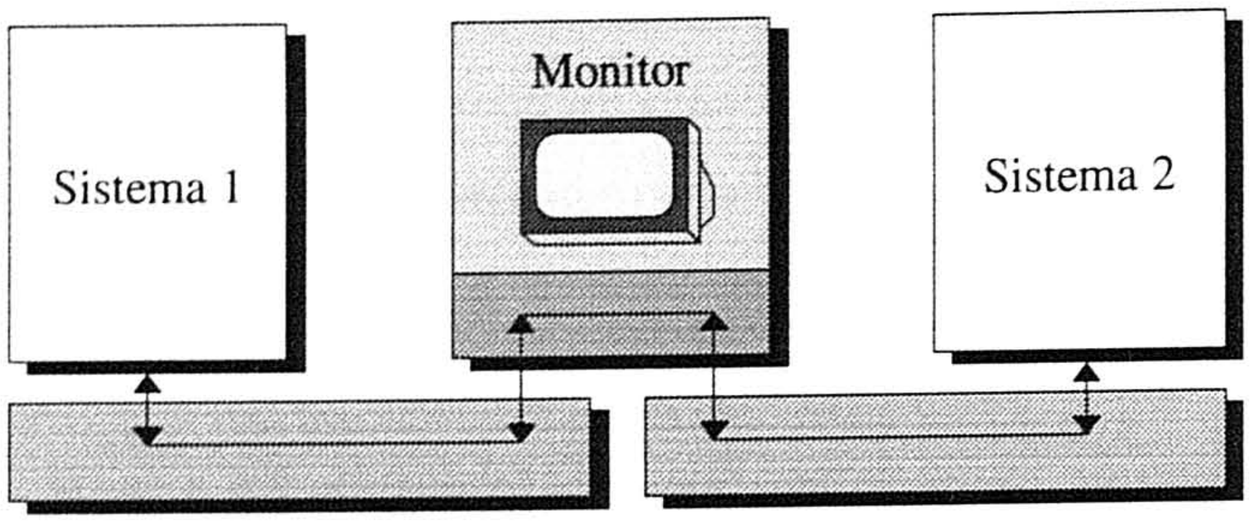


Fig. 2.13 - Arquitectura para testes de interoperabilidade

Com base nesta arquitectura, e para cada um dos modos de testes, podem ser ainda definidos dois métodos de teste, um em laboratório e outro na instalação definitiva (ambiente de funcionamento real). Enquanto que no primeiro, por ser num ambiente isolado, é mais fácil identificar um conjunto de testes e de o normalizar, num ambiente real, os testes executados abrangem um número maior de situações distintas.

Capítulo 3

Especificação de Testes em TTCN

Com a evolução dos testes de protocolos, principalmente os de conformidade e de interoperabilidade, surgiu a necessidade de se especificar, de um modo exacto, as unidades de dados protocolares (*PDUs*) e as primitivas abstractas de serviço (*ASPs*) que se pretendem observar e controlar nos pontos de controlo e observação (*PCOs*).

Várias propostas apareceram para a definição de uma notação de teste, baseadas em diagramas de sequências temporais, em liguagens de programação e/ou em técnicas de descrição formal, como por exemplo o *SDL*, o *ESTELLE* ou o *LOTOS*, e ainda notações em árvore e com tabelas. Todas estas estruturas se mostraram fortes candidatas; no entanto, a perferência dos operadores e dos especificadores de testes recaíu sobre as notações em árvore e com tabelas, por estes métodos se mostrarem mais compreensíveis. Foi então definido o *TTCN - Tree and Tabular Combined Notation,* uma notação que combina estes dois tipos de notação [X292_92][Prober92].

O *TTCN* apresentou uma série de vantagens sobre os outros métodos, por ser fácil de compreender, de manusear e, principalmente, de aprender. Por outro lado, enquanto que as linguagens de programação eram demasiado dependentes dos sistemas onde operavam, o *TTCN,* por ser uma notação de teste abstracta, podia descrever os testes de um modo completamente independente do sistema. Quanto aos diagramas de sequências temporais, embora fossem também de fácil compreensão, não eram suficientemente precisos. As técnicas de descrição formal, embora fossem adequadas para a especificação de tipos, eram superadas pelo *TTCN* tanto na objectividade do teste como na atribuição de valores concretos.

O *TTCN* foi utilizado na normalização do conjunto abstracto de testes dos protocolos *X.25*. Este trabalho foi desenvolvido pela *ISO/IEC,* e demonstrou que o *TTCN* pode de facto ser usado para especificar conjuntos de testes de conformidade, de acordo com os requisitos da metodologia deste tipo de testes e com as especificações das normas [Schnei92][Prober92].

A utilização do *TTCN* na especificação do conjunto de testes do *X.25* foi, de facto, um factor marcante na história desta notação. A partir dessa altura, várias instituições ligadas à área de teste de protocolos e alguns fabricantes de *software* investiram fortemente no desenvolvimento de ferramentas que facultassem todas as vantagens do *TTCN* aos operadores de teste.

Algumas das instituições mais marcantes neste campo são a *TeleLogic*, na Suécia, que está a desenvolver o *ITEX*, o *NIST* (*National Institute of Standards and Technology*), nos E.U.A., e o *NPL* (*National Physical Laboratory*), no Reino Unido.

3.1. *TTCN - Tree and Tabular Combined Notation*

O *TTCN* é uma notação de especificação de testes de conformidade, que foi definida pela *ISO* e aparece descrita nas normas *ISO 9646* parte 3 e nas recomendações *X.292* da *ITU*.

Esta linguagem surgiu quando foi necessário normalizar a descrição dos testes para a verificação da conformidade de um protocolo com as normas *OSI*. Tem como objectivo principal a especificação precisa dos conjuntos de testes, de modo a assegurar o desenvolvimento dos testes executáveis correspondentes, permitindo ainda verificar que essas versões executáveis dos testes são fiáveis face às suas especificações.

O *TTCN*, ao permitir a especificação dos testes de conformidade de uma forma objectiva sem atender às características do equipamento de teste, vai tornar essa especificação bastante portável e flexível.

A existência de uma notação comum, para a descrição dos testes de conformidade de qualquer protocolo, incentiva o desenvolvimento de ferramentas que facultem, aos descritores de testes e aos implementadores de protocolos, um conjunto de facilidades de teste. E, como consequência directa do aumento da oferta deste tipo de produtos, os custos do desenvolvimento de equipamentos descem.

Um outro objectivo do *TTCN* é aumentar o grau de interoperabilidade entre os protocolos de sistemas abertos ao facilitar a normalização dos testes de conformidade.

Por fim, o facto de ser uma notação simples de descrever e de compreender, permite uma fácil aprendizagem das especificações dos testes.

Para conseguir estes objectivos, uma descrição de um teste em *TTCN* consiste em duas partes: uma de declarações estáticas e outra de declarações dinâmicas. A primeira descreve os formatos e os valores de parâmetros particulares, as *PDUs* e as *ASPs*, bem como as características dos *PCOs* e *timers* que, segundo a metodologia dos testes de conformidade, foram utilizados na especificação desse teste. A outra parte descreve o comportamento dinâmico do teste através de um conjunto de operações e operadores associados à ocorrência da transmissão e da recepção de mensagens.

Como o próprio nome sugere, o *TTCN* usa dois tipos de notação: uma baseada em tabelas, que é utilizada para simplificar a representação de todos os elementos estáticos, como por exemplo os tipos de dados, o formato das primitivas abstarctas de serviço e das unidades de dados protocolares, os veredictos associados a determinados eventos do teste, etc; e outra em árvore, que é utilizada na descrição do comportamento dinâmico, para traçar a sequência de eventos que ocorrem como respostas alternativas a outros eventos.

O *TTCN* pode ser descrito de duas formas: uma gráfica, o *TTCN.GR*, e uma *BNF (Backus-Naur Form)*, o *TTCN.MP*. A forma gráfica do *TTCN* é a mais agradável para o utilizador, por ser mais acessível e de leitura mais compreensível. De um modo geral um ambiente *TTCN.GR* é constituído por um conjunto de tabelas, relacionadas entre si, que são facilmente preenchidas pelo programador do teste e onde este descreve tanto as entidades estáticas como o comportamento do teste. Normalmente, um editor de *TTCN.GR* tem a faculdade de, após completada e verificada a descrição do teste, permitir ao utilizador fazer a conversão do *TTCN.GR* para o *TTCN.MP*.

O *TTCN.MP*, embora seja uma forma menos explícita para a edição e descrição do teste, é bastante importante quando se pretende gerar uma versão executável desse teste. Esta forma é, como o próprio nome indica (*.MP - Machine Processable*), bastante versátil para o processamento, o armazenamento e para o transporte de conjuntos abstractos de testes entre diferentes equipamentos.

De acordo com a recomendação [X292_92], um conjunto abstracto de testes, em *TTCN*, é constituído pelas quatro secções seguintes:

– secção de descrição do conjunto de testes (*Test Suite Overview*);

– secção de declarações (*Declarations*);

– secção de restrições (*Constraints*);

– secção de comportamento dinâmico (*Dynamic Behaviour*).

3.1.1. Secção de Descrição do Conjunto de Testes

Esta secção identifica o Conjunto Abstracto de Testes (*ATS*), através de um nome, e define o seu contexto no que respeita às especificações normalizadas dos protocolos da implementação a testar, às Declarações de Conformidade (*PICSs)* e Informações Extra (*PIXITs*), e aos métodos de teste. É nesta secção que é definida a relação entre os vários grupos de teste, que devem ser executados segundo a ordem pela qual os seus casos de teste são invocados no conjunto abstracto de testes. Podem também ser definidas, opcionalmente, expressões condicionais que, como resultado de serem verdadeiras ou falsas, determinam a execução dos casos de teste de um grupo. Só no caso da expressão ser verdadeira é que os casos de teste desse grupo serão executados.

Nesta secção é também guardada informação acerca de todos os casos de teste, desde os seus nomes aos grupos de teste onde estão inseridos, passando por expressões que, como nos grupos de teste, definem se esse casos de testes podem ou não ser seleccionado para execução.

Os passos de teste são também declarados nesta secção, de um modo análogo aos casos de teste.

Todos os procedimentos executados por omissão, num dado conjunto abstracto de testes, têm, igualmente, uma secção reservada neste espaço. Assim, estes são identificados através de um nome, que será invocado nas declarações do comportamento dinâmico, sendo declarados segundo a ordem pela qual aparecem no conjunto abstracto de testes.

Todas estas declarações são feitas, em *TTCN.GR,* preenchendo um conjunto de quatro tabelas com os seguintes nomes: *Test Suite Structure*, *Test Case Index*, *Test Step Index* e *Default Index* (Fig.3.1).

Test Suite Structure			
Suite Name : DEMO Standard Ref : PICS Ref : PIXIT Ref : Test Method : Comments : Exemplo da declaração de um Conjunto de Teste			
Test Group Reference	Selection Ref	Test Group Objective	Page No.
Grupo1 Grupo2	Glower Gupper	*Test cases* do LT *Test cases* do UT	
Detailed Comments:			

i - Estrutura do conjunto de teste

Test Case Index				
Test Group Reference	Test Case Id	Selection Ref	Description	Page No.
Grupo1 Grupo1	TCase0 TCase1	ExecT0 CorrerT1	Um *test case* ... ...outro exemplo	
Detailed Comments:				

ii - Índice dos casos de teste

Test Step Index			
Test Step Group Reference	Test Case Id	Description	Page No.
GrpSteps1	ST0	Um *Step* do test case TCase0	
Detailed Comments:			

iii - Índice dos passos de teste

Default Index			
Default Group Reference	Default Id	Description	Page No.
GrpDefs1	Def0	As omissões do caso TCase0	
Detailed Comments:			

iv - Índice de omissões

Fig. 3.1 - Descrição de um conjunto de testes

Na Fig 3.1. é ilustrado, com um exemplo, o preenchimento das tabelas atrás referidas. Assim, na tabela ***i*** é descrito o conjunto de testes cujo nome é *DEMO*. Este conjunto é constituído por dois grupos de testes (*Grupo1* e *Grupo2*) referenciados na coluna dos grupos de testes. Na coluna das selecções são colocadas, opcionalmente, as expressões que irão condicionar a execução dos grupos refenciados: o *Grupo1* e o *Grupo2* são executados se as expressões *Glower* e *Gupper*, respectivamente, forem verdadeiras. Nos campos dos objectivos pode ser feita uma breve descrição das funções de cada grupo, de modo a auxiliar a descrição do conjunto de testes. Uma descrição mais pormenorizada pode ainda ser feita na área de comentários.

A tabela ***ii*** contém uma identificação de todos os casos de teste envolvidos no conjunto de testes. No exemplo são identificados dois casos de teste *TCase0* e *TCase1* pertencentes ao grupo de testes com a referência *Grupo1*. Estes casos, *TCase0* e *TCase1*, serão executados se as expressões referenciadas por *ExecT0* e *CorrerT1*, respectivamente, tiverem valores verdadeiros. Para cada um destes casos de teste pode ser dada uma explicação abreviada da sua função na coluna das descrições.

O conjunto de testes pode estar organizado em grupos de passos de teste e de omissões, em que cada um não é mais do que uma biblioteca de estruturas de passos de teste e de omissões, respectivamente.

As tabela ***iii*** e ***iv*** têm estruturas idênticas. Na coluna das referências aos grupos são colocados, opcionalmente, os nomes das bibliotecas de passos de testes em ***iii*** (*GrpSteps1*), e de omissões em ***iv*** (*GrpDefs1*), onde o passo de teste *ST0*, e a omissão *Def0*, respectivamente, estão definidos.

3.1.2. Secção de Declarações

Nesta secção são feitas as declarações de todas as entidades intervenientes num conjunto de testes. Todas as componentes do conjunto abstracto de testes, utilizadas em todas as outras secções, devem ser definidas na secção das declarações.

Existem vários tipos pré-definidos em *TTCN*, nomeadamente inteiros, *booleans*, *bitstrings*, *hexstrings*, *octetstrings* e *characterstrings*, mas outros podem ser gerados com base nestes. A definição de tipos, para as entidades de dados, unidades de dados protocolares e primitivas abstractas de serviço, pode ser feita utilizando o formato tabelar e/ou a notação *ASN.1* (*Abstract Syntax Notation-One*). Os

novos tipos declarados podem ser simples ou estruturados, mas não podem referenciar outros de uma forma recursiva.

Quanto às operações, o *TTCN* define um conjunto de operadores aritméticos e lógicos e ainda um conjunto de funções de conversão de tipos e de análise de objectos. Se outras operações forem definidas no conjunto abstracto de testes, elas também podem ser descritas em *TTCN* através de um nome, que as identifica, e de uma lista de parâmetros com os seus tipos.

É, também, nesta secção que são declaradas as constantes derivadas das *PICS* e das *PIXITs*, que serão usadas para parametrizar o conjunto abstracto de testes.

As expressões de selecção dos casos de teste a executar, referidas opcionalmente na secção de descrição do conjunto de testes, são também referenciadas na secção de declarações.

A declaração de todas as variáveis, quer dos casos de teste quer do conjunto global de teste, bem como das constantes neles usadas (*PCOs*, *timers*, *ASPs*, *PDUs*, etc) é feita nesta secção, preenchendo o conjunto das respectivas tabelas ou então utilizando a notação *ASN.1*.

A utilização de ambas as notações, *TTCN* e *ASN.1*, exige que utilizadores e ferramentas tenham a possibilidade de operar com estes dois tipos de notação em simultâneo. De um modo geral, *ASN.1*, que é um formato mais estruturado, é utilizado sobretudo na especificação de mensagens (*PDUs* e *ASPs*) de protocolos de níveis mais elevados, pelo facto de estas serem mais complexas, enquanto que o *TTCN* é usado para especificar essas mensagens de níveis mais baixos (*X.25*, nível de transporte) [Prober92].

Na Fig.3.2 são apresentadas várias tabelas de declaração de entidades de diferentes tipos que poderiam ser utilizados num conjunto abstracto de testes.

PCO Declarations			
PCO Name	PCO Type	Role	Comments
PCOL PCOU	LowerSAP UpperSAP	LT UT	PCO inferior PCO superior
Detailed Comments:			

i - Declaração de *PCOs*

Timer Declarations			
Timer Name	Duration	Unit	Comments
T0 T1	100 10	ms s	
Detailed Comments:			

ii - Declaração de *Timers*

PDU Type Definition		
PDU Name :LxPDU PCO Type :LowerPCO		
Field Name	Field Type	Comments
Header Body	HEXSTRING OCTETSTRING	
Detailed Comments:		

iii - Definição do tipo de uma *PDU*

ASP Type Definition		
ASP Name : LxASP PCO Type : LowerPCO Comments :		
Parameter Name	Parameter Type	Comments
PDUsize PDU	INTEGER LxPDU	
Detailed Comments:		

iv - Definição do tipo de uma *ASP*

Fig. 3.2 - Declaração de diferentes entidades e tipos

Na Fig. 3.2 são declarados, como exemplos, dois Pontos de Controlo e de Observação (*PCOs*) e dois *timers* e são ainda definidos os tipos de uma *PDU* e de uma *ASP*. Na tabela ***i*** são declarados dois *PCOs*, com os nomes *PCOL* e *PCOU*, dos tipos *LowerSAP* e *UpperSAP*, respectivamente. Nesta tabela existe uma coluna, *Role*, onde é descrito que tipo de analisador, *UT* ou *LT*, está colocado nesse *PCO*.

Na tabela ***ii*** são declarados dois timers (*T0* e *T1*) que, caso não lhes sejam atribuídos outros valores ao longo da descrição do teste, irão contar *100 ms* e *10 s*, respectivamente.

Na tabela ***iii*** é definido o tipo *LxPDU* de *PDUs*, transferidas através de *PCOs* do tipo *LowerPCO*. As *PDUs* do teste tipo são constituídas por dois campos, o *Header* e o *Body*, cujos tipos respectivos, *HEXSTRING* e *OCTETSTRING*, são referidos na coluna seguinte.

Para concluir, na tabela ***iv*** é definido o tipo *LxASP* de *ASPs* utilizadas em *PCOs* do tipo *LowerPCO*. As *ASPs* do tipo *LxASP* têm como parâmetros *PDUsize* e *PDU*, que são respectivamente um inteiro (*INTEGER*) e uma *PDU* do tipo *LxPDU* declarado na tabela ***iii***.

3.1.3. Secção de Restrições

Na secção de Restrições de um conjunto abstracto de testes, são especificados valores particulares para os campos das unidades de dados e para os parâmetros das primitivas de serviço, que são usados nesse conjunto. A descrição do comportamento dinâmico do teste é feita em função destes valores. Assim, na transmissão de uma *PDU* e/ou uma *ASP*, os valores contidos nos campos e/ou os parâmetros a transmitir são os definidos nesta secção. Também na recepção, e de um modo análogo, é com os valores aqui definidos que as entidades recebidas (os campos das *PDUs* e/ou os parâmetros das *ASPs*) vão ser comparado, e é em função da validade do valor recebido que são atribuídos os veredictos.

O grau de precisão do teste de conformidade de um protocolo pode variar de acordo com o número de restrições feitas aquando da recepção das mensagens (*PDUs* e *ASPs*). O *TTCN* dispõe de uma gama de símbolos ("* - *AnyValue*", "*?* - *AnyOrOmit*", "*( )* - *ValueList*", ".. - *List*", etc.) que podem ser usados em vez dos valores. Deste modo, é possível definir uma gama vasta de valores permitidos, desde campos de tamanhos variáveis até campos que podem ser omitidos, passando por aqueles cujos valores são quaisquer ou pertencem a determinado conjunto. Estes símbolos só podem ser utilizados na recepção na fase de comparação do valor recebido com o(s) valor(es) permitido(s). Na transmissão, a utilização dos símbolos "*" e "*?*" não é permitida a menos que sejam atribuídos valores concretos a esses campos ou parâmetros ao longo da descrição do comportamento do teste.

Um exemplo de restrições impostas a uma *PDU* e a uma *ASP* dos tipos do exemplo da Fig.3.2 é apresentado na Fig.3.3.

PDU Constraint Declaration		
Constraint Name : pdu1 PDU Type : LxPDU Derivation Path : Comments :		
Field Name	Field Value	Comments
Header Body	'004532'H '014EDF09876'O	Um exemplo ...
Detailed Comments:		

i - Restrição a uma *PDU*

ASP Constraint Declaration		
Constraint Name :asp1 ASP Type : LxASP Derivation Path : Comments :		
Parameter Name	Parameter Value	Comments
PDUsize PDU	9 pdu1	
Detailed Comments:		

ii - Restrição a uma *ASP*

Fig. 3.3 - Tabelas das restrições impostas na transmissão

Nesta figura são ilustradas duas tabelas de definição de uma restrição a uma *PDU* e a uma *ASP*. A tabela ***i*** é um exemplo de uma restrição possível, de nome *pdu1*, a uma *PDU* do tipo *LxPDU*. Nesta, valores concretos, *004532H* e *014EDF09876O*, são atribuídos ou esperados, conforme se trate de uma transmissão ou de uma recepção, nos campos *Header* e *Body*, respectivamente.

Na tabela ***ii*** é definida uma restrição denominada *asp1*, a uma *ASP* do tipo *LxASP* (definido na tabela ***iv*** no exemplo da Fig. 3.2). Aqui o parâmetro *PDUsize* é definido com o valor *9*, enquanto que a *PDU* é definida pela *pdu1* definida na tabela ***I***. Os valores definidos para os parâmetros são os que serão transmitidos ou esperados através do *PCO*.

3.1.4. Secção de Comportamento Dinâmico

A secção do Comportamento Dinâmico de um conjunto de testes engloba as bibliotecas dos casos de teste, dos passos de teste e dos comportamentos por omissão. As funções de cada uma destas bibliotecas são, também, definidas através de tabelas e a descrição da sequência de acções é feita de um modo semelhante ao de uma linguagem de programação. Uma tabela destas (Fig.3.4) inclui uma parte identificativa que tem o nome do caso de teste, o grupo de testes a que pertence, informação relativa a esse caso de teste e ainda, se necessário, uma referência a tabelas de acções alternativas para esse caso. Contém ainda uma outra área onde é feita a descrição da sequência de eventos do caso de teste, zona esta que é seccionada em colunas. Podendo algumas ser ou não preenchidas, outras são imprescindíveis para a descrição do teste.

Test Case Dynamic Behaviour						
Test Case Name : TCase0 Group :Grupo1 Purpose : Default : Comments :						
No	Label	Behaviour Description	Label	Constraints Reference	Veredict	Comments
1		PCOL!LxPDU		pdu1		
2		PCOL?LxPDU		pdu2		
3		PCOL!LxASP		asp1		
4		PCOL?LxPDU		pdu1	PASS	
5		PCOL?LxPDU		pdu2	FAIL	
Detailed Comments:						

Fig. 3.4 - Comportamento dinâmico de um caso de teste

Na coluna de descrição do comportamento (*"Behaviour Description"*) são definidas as funções dos analisadores superior e inferior, nos respectivos pontos de controlo e obsevação (*PCOs*). Na coluna de referência às restrições (*"Constraints Reference"*) são colocados os valores que irão ser usados, transmitidos ou recebidos, através desses *PCOs*. A coluna dos veredictos (*"Veredict"*) é preenchida com os resultados dos eventos ocorridos nessa linha. Esses resultados só podem ser *passou* (*"PASS"*), *falhou* (*"FAIL"*) ou *inconclusivo* (*"INCONC"*), de acordo com a metodologia dos testes de conformidade.

As tabelas de comportamento dinâmico dos passos de teste e dos comportamentos por omissão são análogas a esta.

Se num caso de teste houver repetições das várias sequências de eventos, então devem ser declaradas e preenchidas tabelas de passos de teste com essas sequências, tabelas essas que podem ser invocadas por qualquer outro caso de teste. Este tipo de descrição do comportamento do caso de teste é denominado estruturado [X292_92].

O controlo e a observação de eventos nos *PCOs* são, como já foi referido em 2.1.5., especificados por uma série de declarações na coluna de descrição do comportamento (*"Behaviour Description"*). As instruções são escritas em linhas sucessivas, começando na mesma coluna todas as declarações alternativas, isto é, que podem ocorrer em determinada situação. Para representar a evolução sequencial do sistema, quando um evento previsto ocorre, admite-se que o sistema passa para o nível da coluna seguinte da linha que se segue. Este procedimento repete-se até que seja atribuído um veredicto ao caso de teste e deste modo é possível descrever sequências de instruções em forma de árvore. Um exemplo da estrutura descrita é apresentado na Fig.3.5.

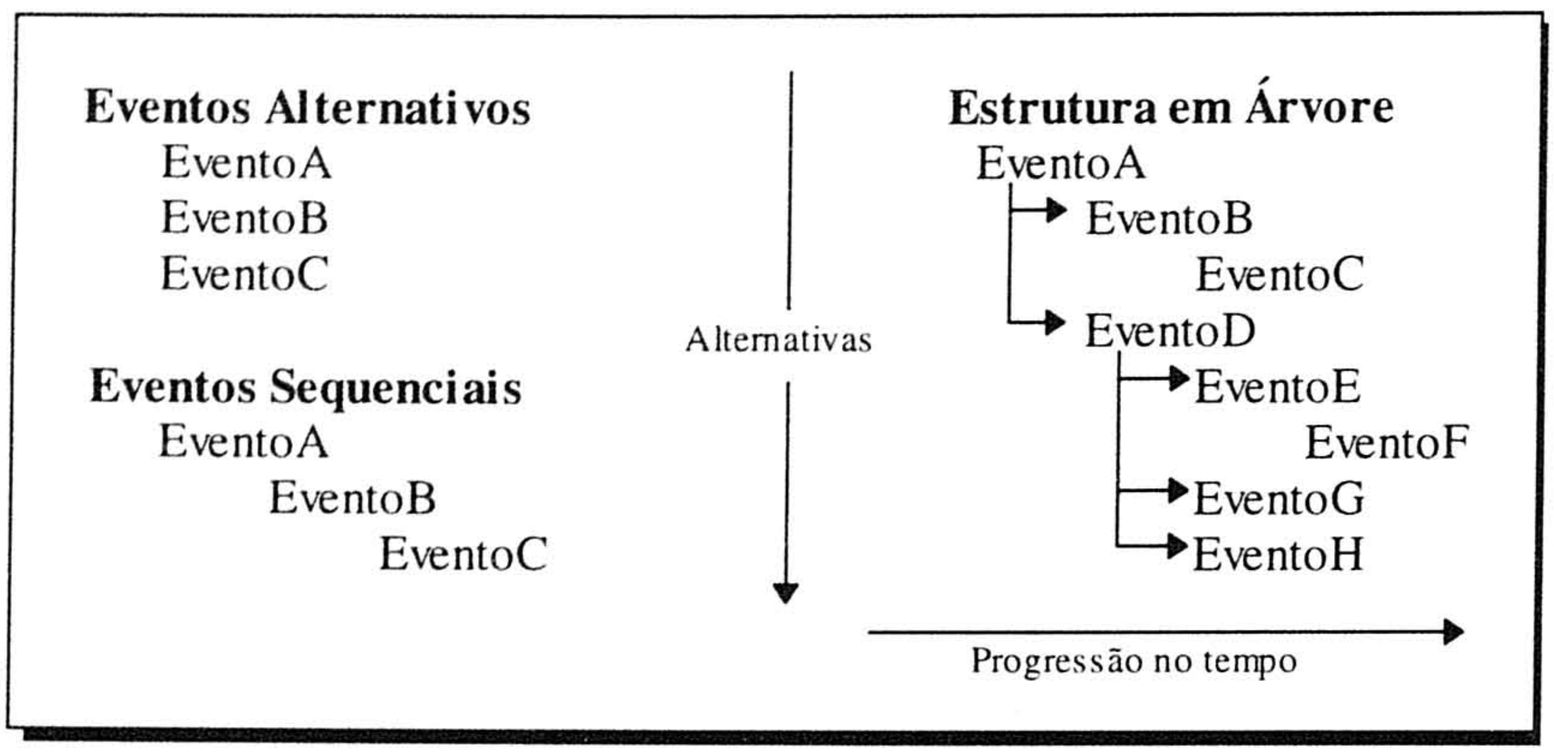


Fig. 3.5 - Evolução da estrutura em árvore

No exemplo da estrutura em árvore, se o "EventoA" ocorrer então o equipamento de teste evolui para o estado em que o "EventoB" ou o "EventoD" são esperados. O processo aguarda neste estado até que uma destas situações ocorra. Se ocorrer o "EventoD" passa para o nível seguinte onde vai esperar pelo "EventoE", pelo "EventoG" ou ainda pelo "EventoH". Este processo desenrola-se sucessivamente até que seja atribuído um veredicto ao caso de teste. Pode ocorrer que ambos os eventos tenham lugar em simultâneo, situação em que é considerado o que foi declarado em primeiro lugar.

Um evento de transmissão é considerado com sucesso sempre que é executado. O mesmo não acontece no caso das recepções: uma recepção só é considerada bem sucedida se os valores recebidos corresponderem aos que são esperados. Os valores esperados são os que aparecem referenciados na coluna das restrições da tabela.

Cada tabela de descrição do comportamento do teste pode conter uma ou mais árvores. A primeira que aparece é a árvore raíz e o seu nome coincide com o do caso ou passo de teste. A partir desta árvore outras podem ser invocadas, as chamadas árvores locais, que também são identificadas por um nome e podem ter ou não parâmetros (Fig.3.6).

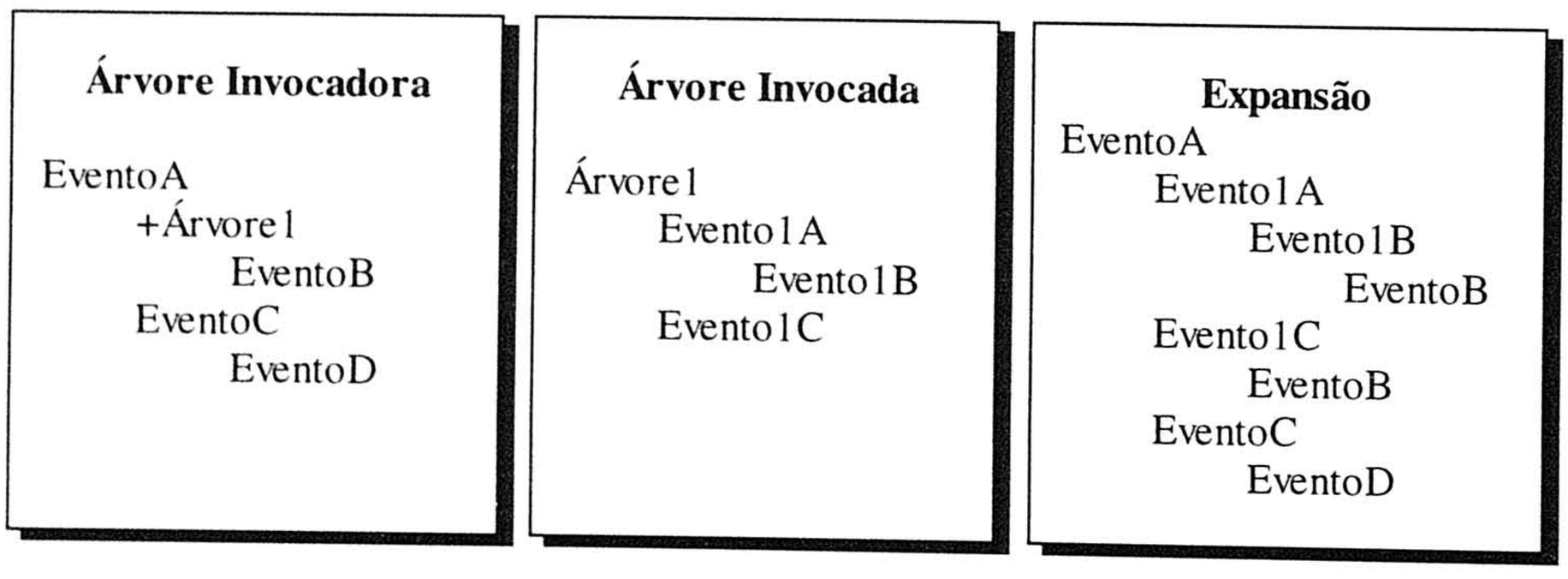


Fig. 3.6 - Invocação de outra árvore

Existe ainda um outro tipo de árvores, que só são usadas no fim da execução de cada um dos ramos, caso nenhum dos eventos esperados ocorra. Estas árvores são declaradas nas tabelas do comportamento por omissão (*"Default Behaviour"*). Para facilitar a leitura dos eventos mais relevantes de um caso de teste deve utilizar-se, sempre que possível, este tipo de árvores, reduzindo-se deste modo a complexidade aparente desse caso de teste. A utilização dos comportamentos por omissão é também válida para os passos de teste. Na Fig.3.7 é ilustrado esse conceito.

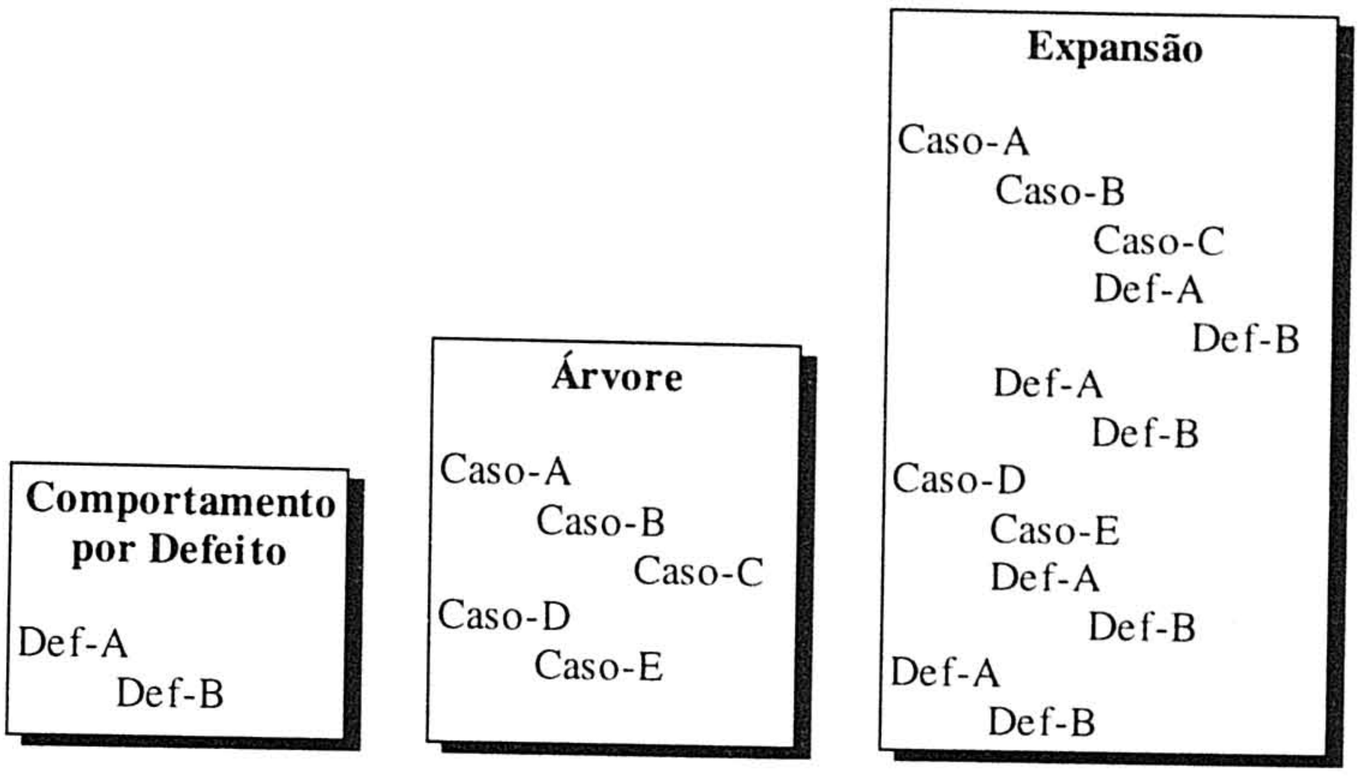


Fig. 3.7 - Comportamento por omissão

As instruções usadas em *TTCN* permitem definir os eventos que ocorrem nos analisadores, *Upper Tester (UT)* e *Lower Tester (LT)*, identificados pelos *PCOs* neles localizados. De um modo geral, definem os eventos iniciados pelos analisadores (*"SEND"* e *"IMPLICITSEND"*), os eventos de teste recebidos ou esperados (*"RECEIVE"*, *"OTHERWISE"* e *"TIMEOUT"*) em ambos os analisadores (*UT* e *LT*), as instruções de construção e repetição de sub-árvores (*"GOTO"*, *"ATTACH"* e *"REPEAT"*) e as instruções de operação dos relógios (*"START"*, *"READTIMER" e "CANCEL"*).

O *SEND* é representado pelo símbolo "*!*" e é, geralmente, seguido do nome do tipo de mensagem (*ASP* ou *PDU*) que vai ser enviada. O *RECEIVE* é representado pelo símbolo "*?*" e é, também, seguido pelo nome do tipo de mensagem que é esperada nessa recepção. Ambos os eventos podem ser precedidos pelo nome do *PCO* onde vão ocorrer, caso exista mais do que um ponto de controlo e observação no conjunto de teste. Estes eventos ficam completos com a coluna das restrições, na qual é colocada uma referência ao valor da mensagem a transmitir ou a receber.

Na Fig.3.8 é apresentado um exemplo representativo da secção dinâmica de um caso de teste.

Test Case Dynamic Behaviour						
Test Case Name : TCase1 Group :Grupo1 Purpose : Default :Def0 Comments :						
No	Label	Behaviour Description	Label	Constraints Reference	Veredict	Comments
1		START T1				
2		PCOL!LxPDU START		pdu1		
3		+ST0 (var1)				
4		PCOL?LxPDU		pdu1		
5		READTIMER T1				
6		CANCEL			PASS	
7		?TIMEOUT T0			INCON	
8		PCOL?OTHERWIS			FAIL	
Detailed Comments:						

i - Comportamento dinâmico do caso de teste

Test Step Dynamic Behaviour						
Test Case Name : ST0 (var:INTEGER) Group :GrpSteps1 Purpose : Default :Def0 Comments :						
No	Label	Behaviour Description	Label	Constraints Reference	Veredict	Comments
1		PCOU?LyPDU		pdu2		
2		READTIMER T0 (var)				
3		PCOU!LyPDU		pdu2		
4		PCOU?OTHERWISE			INCONC	
5						
Detailed Comments:						

ii - Comportamento dinâmico de um passo de teste

Default Dynamic Behaviour						
Test Case Name : Def0 Group :GrpDefs1 Purpose : Comments :						
No	Label	Behaviour Description	Label	Constraints Reference	Veredict	Comments
1		CANCEL			FAIL	
Detailed Comments:						

iii - Comportamento dinâmico de uma omissão

Fig. 3.8 - Exemplo do comportamento dinâmico de um caso de teste

No exemplo da Fig.3.8 são apresentadas três tabelas que descrevem o comportamento dinâmico de um conjunto de testes. Na tabela ***i*** é descrito um caso de teste, com o nome *TCase1*, do grupo de testes *Grupo1*, que a dada altura vai invocar o passo de teste *ST0,* cuja estrutura está na biblioteca *GrpSteps1,* definido na tabela ***ii***. No caso de eventos alternativos, do caso ou do passo de teste, não ocorrerem a

evolução do caso de teste desenrola-se de acordo com a tabela ***iii*** que descreve o comportamento dinâmico da omissão *Def0*.

Neste exemplo, o passo de teste *ST0* tem um parâmetro do tipo inteiro (*INTEGER*), que deve ser passado de ou para uma variável do mesmo tipo previamente definida e declarada nas tabelas destinadas para o efeito. Este caso de teste utiliza duas variaveis, *var1* e *var2*, para onde irão ser lidos os valores dos *timers*.

Na tabela ***i*** descreve-se o caso de teste *TCase1* que se inicia activando o *timer T1*. Seguidamente, pelo *PCO* com o nome *PCOL* é enviado uma *PDU* do tipo *LxPDU*, em que os campos contêm os valores da referência *pdu1*, e em simultâneo é activado o *timer T0*. De seguida é invocada (a invocação é representada nas tabelas pelo símbolo "+") a sub-árvore que descreve o passo de teste *ST0*. A execução do caso de teste passa então a evoluir segundo a tabela ***ii***, em que os eventos possíveis são: a recepção de uma *PDU* de tipo *LyPDU*, no *PCO* com nome *PCOU*, e cujos campos contenham os valores da referência *pdu2*, ou a recepção, nesse *PCO*, de uma outra entidade. Na última situação o veredicto atribuído é *FAIL* e o caso de teste termina aqui. Na primeira situação o teste continua, sendo enviada uma *PDU* de tipo *LyPDU*, através do *PCO* com nome *PCOU*, cujos campos contêm os valores da referência *pdu2*, e de seguida é lido para a variável *var1* o valor da contagem do *timer T0*.

A evolução do caso de teste, depois de executado *ST0* sem que seja atribuído um veredicto, passa de novo a efectuar-se de acordo com o descrito na tabela ***ii***. Assim, nesta fase é prevista a ocorrência de três eventos: a chegada de uma *pdu1* no *PCOL*, o fim da contagem do *timer T0* ou a chegada de uma outra entidade no *PCOL*; se uma *pdu1* for recebido no *PCOL*, é lido o valor da contagem do *timer T0* para a variável *var2*, os *timers T0* e *T1* são parados e o caso de teste termina com o veredicto *PASS*; no caso de ser o *timer T0* a terminar a contagem, o veredicto atribuído ao caso de teste é *INCONC*; e, por fim, caso seja recebido em *PCOL* algo diferente da *pdu1*, o caso de teste termina com o veredicto *FAIL*.

Durante a descrição do caso de teste não foi feito qualquer tipo de referência à omissão *Def0*; no entanto, esta poderia ser invocada no início de *ST0* se, enquanto o sistema aguardasse pela *pdu2*, um dos *timers*, *T0* ou *T1*, chegasse ao fim da contagem antes do *PCOU* receber qualquer entidade. Nesta situação ambos os *timers* seriam parados e o veredicto do caso de teste seria *FAIL*, como é descrito na tabela ***iii***.

3.2. - O *ITEX*

O *ITEX* é uma ferramenta que permite especificar conjuntos abstractos de testes em *TTCN* e gerar o correspondente código executável. Esta ferramenta está ainda em desenvolvimento, mas uma versão já diponibilizada permite gerar, de uma forma automática, o código *ANSI C* correspondente a ao teste *TTCN*.

Esta ferramenta, desenvolvida pela *TeleLogic*, integra um conjunto de utilitários para gerar, a partir da descrição em *TTCN.GR* de um conjunto abstracto de testes, testes executáveis.

O editor do *ITEX*, bastante prático, é usado para editar as várias tabelas do *TTCN*. Com grande facilidade são geradas novas tabelas que o *ITEX* se encarrega de relacionar em função do seu tipo. O editor de tabelas dispõe de um modo de geração semi-automática das instruções de transmissão e recepção, ou seja, apresenta ao utilizador listas com os pontos de controlo e observação, com os tipos e as restrições usadas. Os componentes numa instrução de transmissão ou recepção são seleccionados pelos utilizadores destas listas. O *ITEX* acompanha esta escolha permitindo que apenas as opções válidas sejam seleccionadas [ITEX94][ITEX94a].

O *ITEX* permite a descrição de testes em *TTCN* e em *ASN.1*, como é definido na norma [X292_92]. Após a edição das várias tabelas do teste, é feita a análise e a verificação da sintaxe nelas usadas. Para além desta verificação da sintaxe, o analisador verifica, também, a parte semântica estática, nomeadamente a existência das diferentes entidades e a sua unicidade.

Um outro utilitário é o exportador que permite gerar o código *TTCN.MP* de um conjunto de testes completo, ou apenas de algumas partes pré-seleccionadas. A geração do *TTCN.MP* pode ser feita sem uma análise prévia do *TTCN.GR*, usado como fonte, mas há que ter em atenção que o equipamento, para onde esse código vai ser transportado, pode não o conseguir identificar como sendo um ficheiro *TTCN.MP*, por não estar correctamente descrito e não permitir a sua importação. Assim sendo, antes de proceder a uma operação de exportação, o código *TTCN.GR* fonte deve ser correctamente verificado.

No *ITEX* existem utilitários para importação de conjuntos de testes, ou apenas partes, descritos em *TTCN.MP*, mesmo que estes não tenham sido previamente verificados. É, no entanto, condição obrigatória para o sucesso desta aplicação que o ficheiro *TTCN.MP* a importar seja descrito segundo as normas *ISO*.

Infelizmente, uma sequência de teste expressa em *TTCN* não pode ser executada, isto porque a notação não é executável e muito poucos sistemas conseguem ler um ficheiro em *TTCN*.

É importante transformar a descrição em *TTCN* do teste numa linguagem executável. Para tal, o *ITEX* dispõe de um gerador de código *ANSI-C*, o *ITEX C Code Generator*. Este utilitário gera o código *ANSI-C* para qualquer caso de teste expresso em *TTCN*. Existe, no entanto, alguma falta de funcionalidade no código gerado, isto porque é necessário desenvolver um grupo de funções de adaptação deste código ao equipamento que irá executar o teste. Esta adaptação tem por objectivo completar faltas de informação nas seguintes áreas: codificação e descodificação, comunicações e ambiente de teste.

A área da codificação e descodificação está fora do âmbito desta ferramenta, pois um conjunto de teste não contém informação acerca das regras de codificação e de descodificação dos protocolos utilizados. Assim, embora o *ITEX CCG* defina a representação das várias mensagens, isto é, *PDUs* e *ASPs*, tem de ser o implementador do equipamento de teste a definir as regras de transmissão e de

recepção dessas mensagens de e para esse equipamento. Essas regras, que podem ser implementadas por funções, definem uma sintaxe de transferência comum entre o equipamento de teste e o sistema onde o teste é executado.

Mas mesmo assegurando a codificação e a descodificação das mensagens, não é possível transformar, directamente, um conjunto abstracto de testes em testes executáveis.

De um ponto de vista abstracto, a transmissão e a recepção são feitas através dos pontos de controlo e observação. A representação física destes *PCOs* tem também de ser definida pelo implementador do equipamento de teste. A parte da comunicação, no lado de controlo do *PCO*, poderá ser implementada de diversas formas, como por exemplo memória partilhada, comunicação série, *sockets UNIX*, etc. Mas a outra parte da comunicação, no lado da observação do *PCO*, tem que estar de algum modo inserida na interface do equipamento de teste. Este facto traduz-se numa falta parcial de informação pois só é possível caracterizar correctamente a comunicação do lado de controlo do *PCO*, enquanto que, a comunicação do seu lado de observação, por ser dependente da máquina onde é implementado o equipamento de teste, não pode ser caracterizada.

Quanto ao ambiente de teste, a falta que é necessário superar revela-se a dois níveis. Num primeiro nível, é necessário saber se o conjunto de testes executáveis irá operar no equipamento de teste, isto é, se a comunicação entre o teste executável e a *IUT* é feita dentro desse equipamento. Se for este o caso, o teste executável deve então ser colocado nesse equipamento. Num segundo nível, os *timers* utilizados num conjunto abstracto de teste são representados de modos diferentes nas diversas máquinas (*UNIX*, *PC*, *OS9*, etc). Por isso, é atendendo à máquina onde o teste será executado que o implementador do teste deve desenvolver as funções dos *timers* definidos no conjunto abstracto de testes.

Por fim é de referir que o *ITEX* dispõe também de um simulador que facilita ao implementador o desenvolvimento dos conjuntos de testes. Com este utilitário é possível:

– executar um conjunto de testes simulado e analisar os resultados;

– desenvolver um conjunto de testes passo a passo;

– inspeccionar os valores das variáveis e constantes de um teste;

– analisar as mensagens a transmitir ou recém-chegadas;

– enviar mensagens novas ou alteradas.

Como é fácil de constatar, o *ITEX* é uma ferramenta que ocupa um lugar previlegiado no mundo do desenvolvimento de novos serviços, por permitir descrever, simular e testar qualquer protocolo. Por outro lado, permite que laboratórios de teste de protocolos não necessitem de fazer grandes investimentos sempre que pretendam executar um determinado teste em ambientes diferentes.

Capítulo 4

SGDT - Sistema de Geração e Detecção de Tráfego

A evolução dos sistemas de comunicação é cada vez mais um factor crítico para o sucesso das operadoras de telecomunicações. Esta evolução não é só provocada pelo facto da crescente adesão de clientes a um determinado serviço, mas também pelo facto de surgirem novos serviços que exigem cada vez mais das actuais redes de comunicação. Com o aparecimento das aplicações muti-utilizador (por exemplo a video-conferência) e multimédia em tempo real era necessário garantir a transferência de grandes quantidades de informação em curtos intervalos de tempo, havendo, portanto, a necessidade de melhorar as características dos sistemas de comunicação existentes. Estes e outros tipos de aplicações exigem que durante breves períodos a rede disponibilize grande largura de banda; no entanto esta é limitada, e na maioria dos casos tem que satisfazer centenas de utilizadores. É por isso necessário avaliar as características e as possibilidades de uma rede comunicação de modo a garantir a qualidade exigida por todos os serviços que dela dependem.

A qualidade dos serviços (*QoS - Quality of Service*) que operam numa dada rede não é uma grandeza directamente mensurável e, de certo modo, é até subjectiva. Este facto não facilita de modo algum o processo de avaliação da qualidade dos serviços. Há então necessidade de se identificarem parâmetros, directa ou indirectamente mensuráveis e quantificáveis, dos quais depende a qualidade dos serviços. Dos parâmetros de uma rede os que estão mais relacionados com a qualidade dos serviços são os de desempenho da rede (*NP - Network Performance*).

O desempenho de uma rede pode ser definido como sendo a capacidade dessa rede executar todas as suas funções, sob determinadas condições, num determindado intervalo de tempo [RACE92]. Este conceito permite a definição, a avaliação e o controlo das características da rede de modo a garantir todas as funções de comunicação entre os utilizadores e a permitir a obtenção de níveis satisfatórios na qualidade dos serviços. Os parâmetros que medem o desempenho de uma rede são vários e podem

exprimir uma quantidade, uma probabilidade ou uma média. Todos eles são obtidos a partir de medições efectuadas directamente sobre a rede, como por exemplo contagens de eventos em determinados estados, a duração desses estados, a medição dos intervalos de tempo entre eventos, etc. Deste modo são determinados os valores dos atrasos no estabelecimento e libertação de chamadas entre sistemas, dos atrasos da transmissão de informação na rede, das probabilidades de haver perdas ou atrasos na informação, das probabilidades do estabelecimento ou da libertação de uma chamada falharem, e tantos outros que em conjunto caracterizam o desempenho de uma rede.

Neste capítulo é apresentada uma ferramenta, o Sistema de Geração e Detecção de Tráfego (SGDT), desenvolvida no âmbito de dois projectos *ESPRIT*, o *DAMS* (*Dynamically Adaptable Multi-Service System*) e o *UCOL* (*Ultra Wideband Coherent Optical Lan*) [Carrap90][Ricard90][Assis93][Mamede93] [Mamede93a]. Nestes projectos era requerida uma ferramenta de teste que, acedendo a diversos pontos de uma rede de transmissão, conseguisse avaliar, de forma estatística, os parâmetros de desempenho dessa rede.

O SGDT é um equipamento de teste que permite determinar vários parâmetros associados ao desempenho de uma rede. Para tal, são utilizados mecanismos de geração de tráfego, com características bem definidas que é transmitido através da rede que se pretende testar, bem como mecanismos para a detecção e a realização de medidas sobre o tráfego gerado [Ricard91][Proenç92].

4.1. Arquitectura e Princípio de Funcionamento

A arquitectura do SGDT é baseada em várias entidades distintas, com funcionamentos independentes mas relacionados. Globalmente este equipamento de teste é coordenado a partir de uma estação de trabalho (*WS - WorkStation*). Aqui são gerados os comandos para as unidades de processamento de tráfego (UPTs) que se encarregam de executar os testes de acordo com esses comandos [Ricard91][Assis95](Fig.4.1).

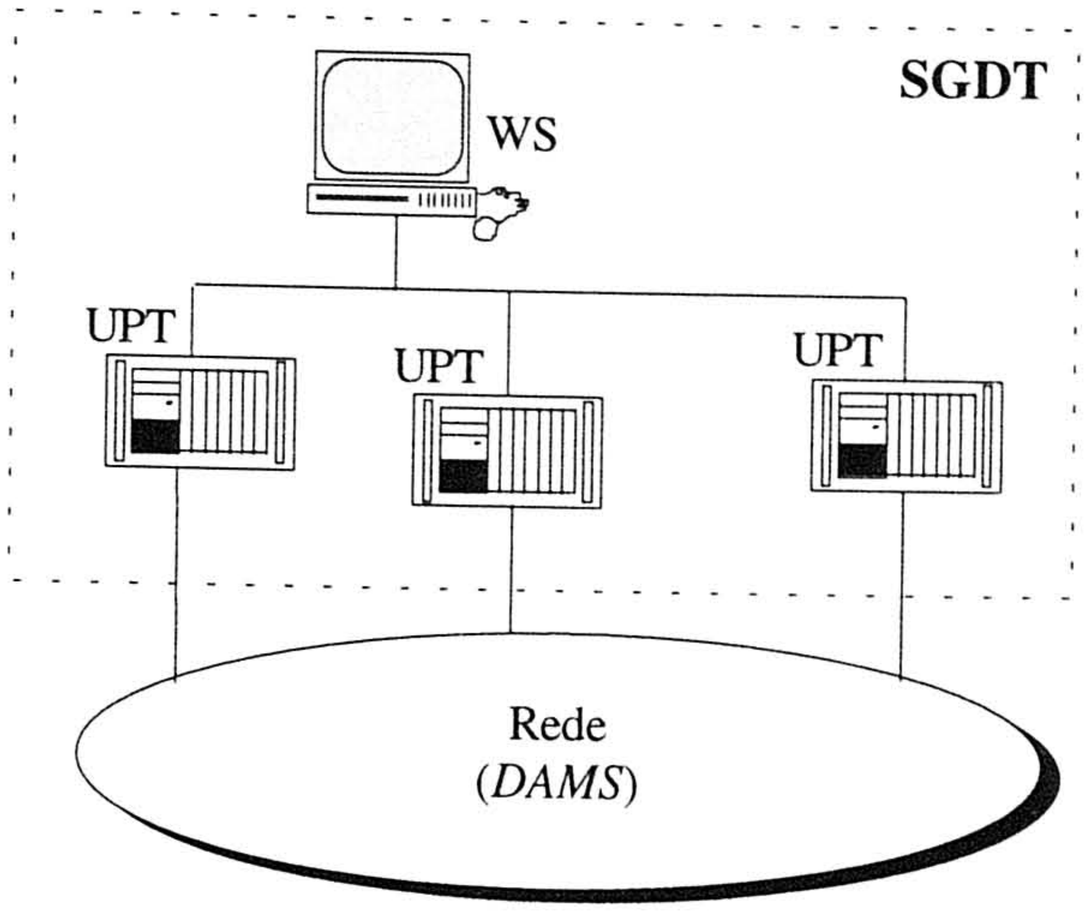


Fig. 4.1 - Arquitectura do SGDT

4.1.1. Estação de Trabalho

A estação de trabalho (*WS*) desempenha vária funções de relevo entre as quais a configuração de todas as unidades de processamento de tráfego (UPTs), a definição dos testes e a apresentação dos resultados. É através da *WS* que o operador dos testes coordena todo o sistema por meio de uma interface gráfica desenvolvida especialmente para o efeito, a interface homem-máquina (*MMI - Man-Machine Interface*).

O processo de avaliação das características da rede assenta no estabelecimento de ligações ponto-a-ponto entre um elemento gerador e um detector de uma ou mais UPTs, ou entre uma UPT e um elemento da rede [Assis95][Ricar91].

Na *MMI* são descritas as sessões de teste, que não são mais do que sequências de experiências a efectuar por cada uma das unidades de processamento de tráfego intervenientes no teste.

A descrição de cada sessão consiste na caracterização de cada uma das ligações utilizadas no teste. Estas são identificadas pela origem, pelo destino, pelo perfil de tráfego nessa ligação e pelo tipo de resultados a obter. Essa informação é guardada em ficheiros e, assim que o operador dá ordem para que essa sessão seja executada, uma entidade de gestão, o gestor de sessões, acede a esse ficheiro, codifica a informação nele contida e envia-a para as UPTs envolvidas nessa sessão.

A *MMI* possibilita também ao operador do teste analisar os resultados das várias sessões e a visualização dos histogramas respectivos. Isto é conseguido à custa dos resultados enviados pelas UPTs que o gestor de sessões se encarregou de descodificar e armazenar num ficheiro de resultados.

É ainda através da *MMI* que o operador dá os comandos de controlo e de coordenação dos testes, comandos estes que, uma vez mais, são recebidos pelo gestor de sessões que se encarrega de os executar.

Um esquema representativo da estrutura funcional da *WS* é apresentado na Fig.4.2.

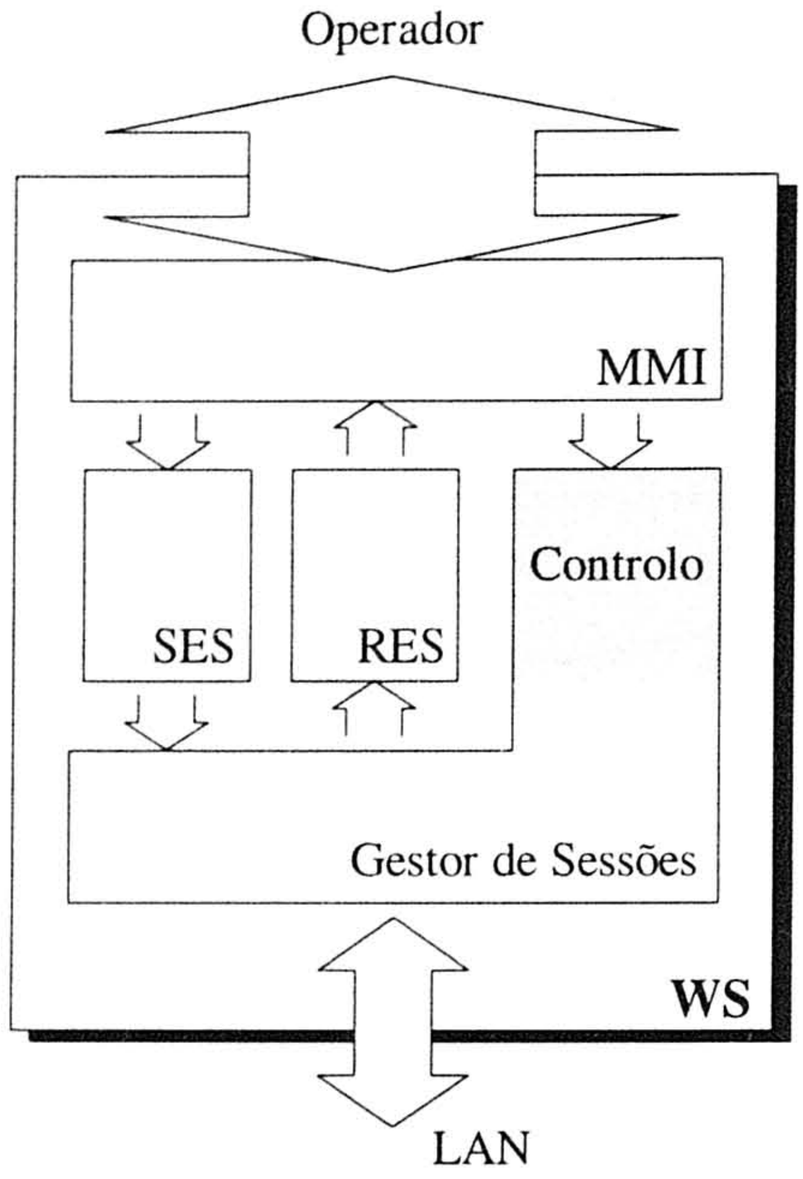


Fig. 4.2 - Estrutura funcional da *WS*

A comunicação entre a *WS* e as várias unidades de processamento de tráfego é feita por meio de uma *LAN* (*Ethernet*), isto porque a informação transmitida são apenas os comandos de gestão ou os resultados das sessões, não interferindo estes no funcionamento de uma sessão. Também no caso de serem seleccionadas para uma sessão várias UPTs, elas vão comunicar entre si através dessa *LAN*.

4.1.2. Unidade de Processamento de Tráfego

As unidades de processamento de tráfego são as componentes mais activas do SGDT. São unidades auto-inicializadas que, de acordo com os comandos e as configurações impostas pelo gestor de sessões, vão estabelecer as ligações, gerar o tráfego com as características exigidas e enviá-lo pela rede, e ainda, no caso da detecção desse tráfego, são responsáveis por efectuar as medidas, organizar e guardar os resultados e, por fim, enviá-los para o gestor de sessões [Assis95].

Cada UPT é constituída por três ou mais blocos: um de acesso à *Ethernet,* um Bloco Controlador responsável pela gestão da UPT e pelo menos um Bloco Sevidor com a respectiva interface à rede de comunicação (Fig.4.3).

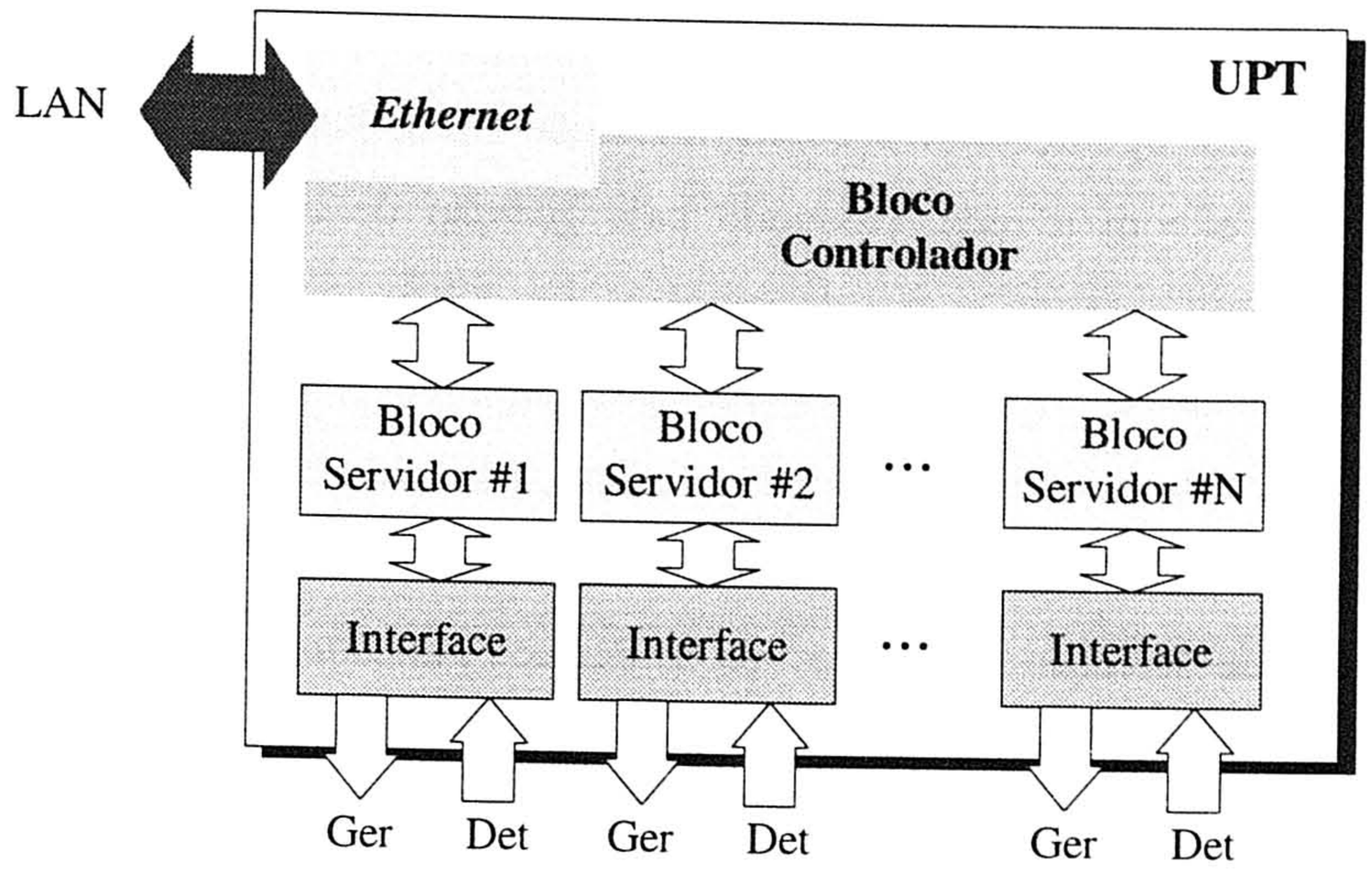


Fig. 4.3 - Arquitectura de uma UPT

O bloco de acesso à *Ethernet* é o meio utilizado para se estabelecer a comunicação quer com a *WS* quer com as outras UPTs. A comunicação com a *WS* é feita por meio de chamadas de procedimentos remotos (*RPC - Remote Procedure Call*). Se o teste, definido na sessão, utilizar mais do que uma UPT é necessário seleccionar uma UPT mestre de entre as activas que coordene a execução dessa sessão de teste. A selecção pode ser feita por pré-definição ou por difusão de mensagens a todas as UPTs. Assim, em função das respostas (mensagens que descrevem a actual configuração da UPT) de cada uma é escolhida a que reunir o maior conjunto de funcionalidades usadas nessa sessão.

O Bloco Controlador tem por função coordenar toda a unidade de processamento de tráfego. É ele que determina o funcionamento de cada Bloco Servidor: inicializa os registos e carrega para a memória o programa com as funções a desempenhar. Durante a fase de inicialização o Bloco Controlador executa alguns testes, de modo a detectar quantos Blocos Servidores tem que gerir. Após este processo de identificação dos servidores, é muito importante sincronizá-los, no caso de se utilizar mais do que um, pois só deste modo é possível efectuar algumas das medidas (por exemplo os atrasos) necessárias para avaliar a qualidade dos serviços. Por esta razão, e pelo facto de não terem sido desenvolvidos mecanismos de sincronização entre as várias unidades de processamento de tráfego, as ligações *gerador-detector* de uma sessão destinada a medir atrasos devem ser feitas entre Blocos Servidores da mesma UPT.

A implementação deste bloco assenta num processador multi-tarefa, o *68030* da Motorola, para que o atendimento dos vários Blocos Servidores fosse rápido. Pela mesma razão, esse processador foi dotado de um sistema operativo que, além de ser multi-tarefa, opera em tempo real, o que significa que os atrasos no atendimento das interrupções é mínimo. A ligação dos diferentes Blocos Servidores ao Bloco Controlador é feita através de um barramento *VME* (*Versa Module Europe*).

Os Blocos Servidores são, em cada unidade de processamento de tráfego, as entidades que têm de realizar mais funções, desde operações aritméticas (incluindo multiplicações e divisões) até às funções de atendimento da interface com a rede, passando pela geração de resultados e pela construção de histogramas. Por este motivo, optou-se por desenvolver este bloco com base num *DSP* (*Digital Signal Processor*), o *TMS320C25* da Texas Instruments. Este tipo de processadores é caracterizado pela possibilidade de realizar uma quantidade elevada de cálculos aritméticos em poucos ciclos de relógio. A selecção de um *DSP* foi decisiva face à necessidade de se realizarem cálculos matemáticos entre os períodos de geração e de detecção de pacotes: se os débitos utilizados no teste forem elevados, esses intervalos de tempo são muito reduzidos, exigindo um processamento rápido da informação.

É ao nível do Bloco Servidor que são executadas as diversas funções de uma sessão. Neste bloco são construídos os pacotes a enviar, são inicializados os relógios para a medição de atrasos, são estabelecidas as ligações e são ainda analisados os pacotes recebidos e, em função deles, gerados os resultados, de acordo com os requisitos da sessão.

Finalmente há que referir que cada Bloco Servidor acede à rede de comunicação através de uma interface. Esta dispõe de mecanismos de armazenamento para adaptação de débitos e implementa o protocolo de acesso à rede de comunicações.

No caso do projecto *DAMS,* este protocolo define um modo de transferência bidireccional de informação, entre a rede e cada um dos elementos externos a ela ligados. São transferidos oito *bits* em paralelo a uma frequência de 8.192 MHz, o que perfaz um débito de 64 Mbps.

Numa versão posterior deste equipamento, no âmbito de um outro projecto *ESPRIT*, o *MANTIS* (*Metropolitan Area Network for Integrated Services*), foi desenvolvida uma interface para aceder a um serviço *SMDS* (*Switched Multi-megabit Data Service*) a 2 Mbps.

4.2. Definição de uma sessão

A descrição de um teste no SGDT é feita a partir da definição de uma sessão, que não é mais do que um ficheiro que contém informação acerca da duração dessa sessão e de caracterização das ligações estabelecidas. Estas ligações podem ser de dois tipos: de pacotes ou de sinalização. As ligações de pacotes são caracterizadas por serem unidireccionais e serem estabelecidas entre uma ou mais UPTs, enquanto que as ligações de sinalização são bidireccionais e de um modo geral ligam uma UPT a uma unidade de gestão de chamadas da rede [Ricard91].

4.2.1. Ligações de Pacotes

As ligações de pacotes envolvem a geração destes e podem ter vários objectivos: a medição de atrasos e taxas de erro, se o destino desses pacotes for um Bloco Servidor (da mesma UPT que inclui o gerador), ou então a carga da rede com um perfil de tráfego bem definido, no caso dessa ligação não se destinar a nenhum Bloco Servidor.

A caracterização de uma ligação é, como já foi referido, feita durante a descrição da sessão e consiste na definição de uma origem, um destino, um perfil de tráfego e dos tipos de resultados a obter.

A origem é a identificação do Bloco Servidor que gera os pacotes, enquanto que o destino tanto pode ser a identificação de um Bloco Servidor, como de outra entidade da rede, existente ou não.

O perfil de tráfego corresponde à caracterização do processo de geração de pacotes, e nele são definidos os comprimentos dos pacotes, os intervalo entre pacotes consecutivos e, possivelmente, o número de pacotes gerados ou a duração do período de geração de pacotes com essas características. Cada um destes parâmetros pode ser constante ou descrito por uma função distribuição de probabilidade, com uma dada média e, se necessário, uma variância. As funções distribuição de probabilidade utilizadas são a de Poisson, a Gaussiana e a Uniforme. Embora seja possível ao operador definir o perfil de tráfego, o sistema já tem algumas funções de geração pré-definidas: a determinística, a markoviana, a gaussiana e *AFAP* (*As Fast As Possible*). As três primeiras são caracterizadas por um comprimento de pacote constante e um intervalo entre pacotes constante, no caso da função determinística, com ditribuição exponencial, para a função markoviana, e com distribuição gaussiana, para a função do mesmo nome. A função *AFAP* é caracterizada pela geração da maior quantidade possível de tráfego nessa ligação.

Por fim, a caracterização de uma ligação de pacotes fica completa com a definição dos resultados a obter, que podem ser histogramas e contagens. Os histogramas podem incluir por exemplo intervalos entre pacotes, comprimentos de pacotes e atrasos extremo a extremo, enquanto que as contagens poderão ser de pacotes gerados, de pacotes recebidos, de pacotes com erro e de pacotes incompletos.

4.2.2. Ligações de Sinalização

Este tipo de ligações tem por objectivo avaliar a facilidade com que são estabelecidas chamadas, entre uma unidade de processamento de tráfego e a entidade gestora de chamadas da rede, bem como a qualidade dessas chamadas.

A caracterização deste tipo de ligações é feita com a identificação do Bloco Servidor que origina as mensagens, o perfil de tráfego e os resultados a obter.

O perfil de tráfego define a taxa de estabelecimento, isto é, o intervalo entre chamadas consecutivas e a duração das chamadas. Cada um destes parâmetros é descrito através das funções de distribuição já descritas para as ligações de pacotes.

Quanto aos resultados, são obtidos no Bloco Servidor que originou as chamadas e, também neste caso, podem ser histogramas ou contagens. Os histogramas incluem intervalos de tempo entre pedidos consecutivos de estabelecimento de chamadas, duração de chamadas, atrasos de estabelecimento e da terminação de chamadas. As contagens, por sua vez, podem ser de chamadas estabelecidas com sucesso, rejeitadas e de chamadas incompletas.

Capítulo 5

SMDS - Um Serviço de Suporte

As redes de comunicação utilizam cada vez mais tecnologias de banda larga de modo a poderem suportar novos serviços que exigem maior largura de banda que os serviços tradicionais. Um serviço de banda larga é o *SMDS* (*Switched Multi-megabit Data Service*), serviço cuja disponibilização é já uma realidade nos Estados Unidos e tem vindo a ganhar adeptos entre as várias operadoras europeias [Heywoo94]. Através dele é possível interligar várias *LANs* dispersas de modo que, para os utilizadores, a resposta seja a que se obteria se se tratasse de uma única *LAN*.

O *SMDS* oferece, actualmente, acessos de 1.5 Mbps e 45 Mbps, nos Estados Unidos, e de 2 Mbps e 34 Mbps, na Europa, mas estão previstos acessos de 155 Mbps e de 622 Mbps [Hemric91].

É natural que, atendendo a este conjunto de factores, haja por parte das entidades envolvidas no desenvolvimento de equipamentos e serviços uma certa tendência para investir em produtos que utilizem este serviço como meio de comunicação.

Começaram também a surgir projectos para o desenvolvimento de novas arquitecturas de banda larga que utilizam o *SMDS* como meio de acesso a redes de área metropolitana (*MANs*) e áreas ainda mais vastas (*WANs*). Foi este o caso concreto de um projecto *ESPRIT II*, o *MANTIS* (*Metropolitan Area Network for Integrated Services*), cujo objectivo era desenvolver uma arquitectura para *MANs* sobre *ATM* (*Asynchronous Transfer Mode*), capaz de integrar uma grande variedade de serviços. No âmbito deste projecto foi desenvolvida uma interface *SMDS* (2 Mbps) para o SGDT, através da qual seria avaliada a qualidade deste serviço sobre a rede *ATM*.

Surgiu então a ideia de alargar o horizonte das funcionalidades do SGDT e, utilizando parte dos recursos existentes, e desenvolvendo uma plataforma que permitisse também executar testes de conformidade e de interoperabilidade de serviços e aplicações que operassem sobre o *SMDS*. Para o efeito foi desenvolvido software para a unidade de processamento de tráfego do SGDT, que permitisse implementar as funcionalidades dos níveis SMDS e oferecer primitivas para a execução de testes de conformidade e de interoperabilidade em implementações que operem sobre esse serviço.

5.1. *SMDS - Switched Multi-megabit Data Service*

O *SMDS* (*Switched Multi-megabit Data Service*) foi o primeiro serviço de banda larga normalizado a atingir a fase de comercialização. Trata-se de um serviço de comutação de informação a elevados débitos não orientado às ligações, que permite aos utilizadores usufruir das características de uma rede local (*LAN - Local Area Network*) em áreas metropolitanas (*MANs - Metropolitan Area Networks*) ou ainda mais vastas (*WANs - Wide Area Networks*) [Clapp91][Wrigle90][Hemric91][Wright93] .

Este serviço foi definido e especificado pela *Bellcore* para explorar as vantagens da tecnologia *DQDB (Distributed Queue Dual Bus)*, motivo pelo qual o *SMDS* é baseado na norma *IEEE 802.6* que rege as redes para área metropolitana [Hemric90][Wrigle90][Hemric91][Wright93].

O *DQDB* é uma tecnologia distribuída que baseia o mecanismo de acesso à rede em filas de espera, que foi definida de modo a ser compatível, tanto quanto possível, com a tecnologia *ATM* (*Asynchronous Transfer Mode*) usada na RDIS-BL (Rede Digital com Integração de Serviços - Banda Larga). Essa compatibilidade é evidenciada ao nível das células e cabeçalhos que, por serem semelhantes, apenas requerem pequenas adaptações para serem convertidos de uma tecnologia para a outra [802.6_91] [Hemric91] [Rodeck93] [Wright93].

Esta compatibilidade, que permite desde já prever o sucesso do serviço *SMDS* em redes *ATM,* isto é, com base nestas tecnologias e através do *SMDS* é possível aos assinantes operar numa *WAN* como se de uma *LAN* se tratasse.

Na Europa, um serviço com características idênticas ao *SMDS*, o *CBDS* (*Connectionless Broadband Data Service*), encontra-se em fase de normalização pelo *ETSI* (*European Telecommunications Standards Institute*) e pela *ITU* [Antone93][Wrigtht93][Heywoo94].

O *SMDS* tem por função oferecer aos assinantes tanto as características de uma *LAN* como, em simultâneo, as de uma *WAN*. Assim, por um lado, como é característico das *LANs,* é possível ao assinante: comunicar com vários destinos sem estar sujeito aos atrasos do estabelecimento de chamadas; utilizar interactivamente computadores remotos, bem como sistemas de armazenamento e outros equipamentos; transmitir pacotes com vários milhares de octetos de tamanho; dispor de uma taxa de erros baixa e enviar picos de tráfego. Por outro lado, e como é característico das *WANs,* o *SMDS* deve garantir aos assinantes uma segurança elevada, a possibilidade de seleccionar grupos de utilizadores, através da especificação dos destinos com os quais se pretende comunicar, e a possibilidade multiplexar o tráfego dos vários utilizadores numa só linha.

O *SMDS* é acedido através das linhas de débitos *E1* (2.048 Mbps) e *E3* (34 Mbps) na Europa e de linhas de débitos *DS1* (1.544 Mbps) e *DS3* (45 Mbps) nos E.U.A..

O ambiente típico de uma rede com *SMDS* geralmente envolve várias *LANs*, que poderão estar ou não geograficamente próximas uma das outras, em que cada uma delas constitui uma subrede com

determinadas características tecnológicas, do ponto de vista do operador da rede global. A rede que suporta o serviço *SMDS* é normalmente denominada rede *SMDS*.

O *SMDS* garante a transferência de unidades de dados do utilizador, que podem conter até 9188 octetos de informação. A dimensão destas unidades permite ao *SMDS* encapsular pacotes completos da maioria das *LANs*, incluindo *IEEE 802.3 CSMA/CD*, *IEEE 802.4 Token Bus*, *IEEE 802.5 Token Ring* e *ANSI X3T9 FDDI* [Henric90].

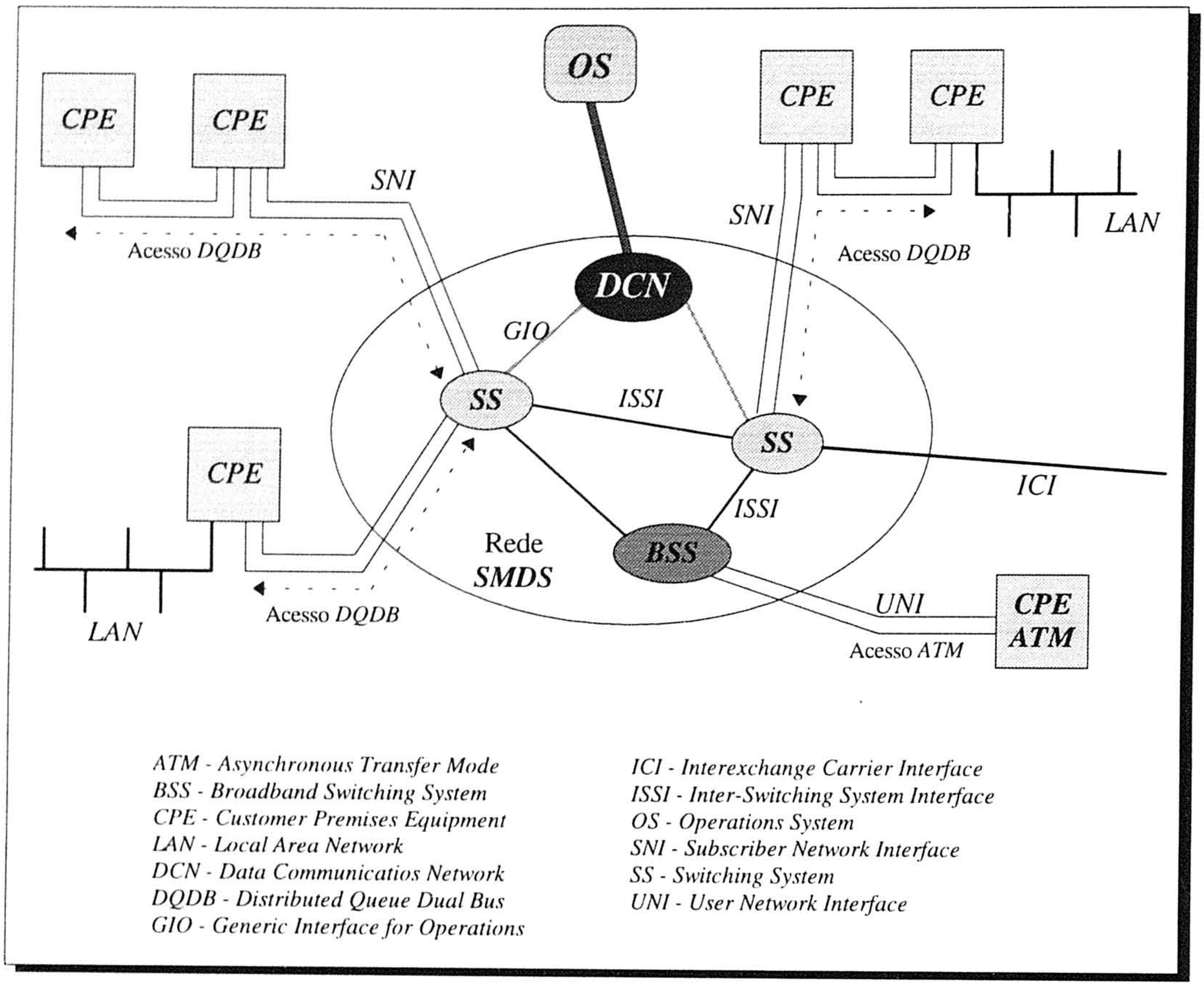
5.2. Arquitectura de Suporte do *SMDS*

De um modo geral uma arquitectura para o suporte deste serviço envolve sempre os equipamentos dos clientes (*CPE - Customer Premises Equipment*) e a rede através da qual vão comunicar. O acesso dos clientes à rede é feito por intermédio de um sistema de comutação (*SS - Switching System*), através de uma interface dedicada para cada assinante (*SNI - Subscriber Network Interface*) e segundo um protocolo de interface (*SIP - SMDS Interface Protocol*). Para o caso de assinantes *ATM*, o acesso é feito por intermédio de sistemas de comutação de banda larga (*BSS - Broadband Switching System*), através de interfaces dedicadas dos assinantes *ATM* (*UNI - User Network Interface*)

O *SIP* é a implementação do nível de controlo do acesso ao meio (*MAC - Medium Access Control*) do protocolo *DQDB*, descrito na norma *IEEE 802.6*. Este protocolo garante a comunicação entre os equipamentos dos vários assinantes e o sistema de comutação por meio de dois barramentos (*Bus A* e *Bus B*) unidireccionais. Este mecanismo de acesso ao sistema de comutação é denominado *acesso DQDB* e não é mais do que uma rede deste tipo, em que o *SS* desempenha as funções de *Head* do barramento *A* e o *CPE* terminal desempenha as funções de *Head* do barramento *B* da rede *DQDB*.

Os sistemas de comutação existentes na rede estão ligados entre si através de uma interface específica, a *ISSI* (*Inter-SS Interface*), e são geridos por um sistema de operação (*OS - Operations System*) que lhes acede por intermédio de redes de comunicação de dados (*DCNs - Data Communication Networks*). As comunicações entre redes a grandes distâncias são feitas pela interface *ICI* (*Interexchange Carrier Interface*).

Uma ilustração representativa da arquitectura para suporte do *SMDS* é apresentada na figura 5.1.



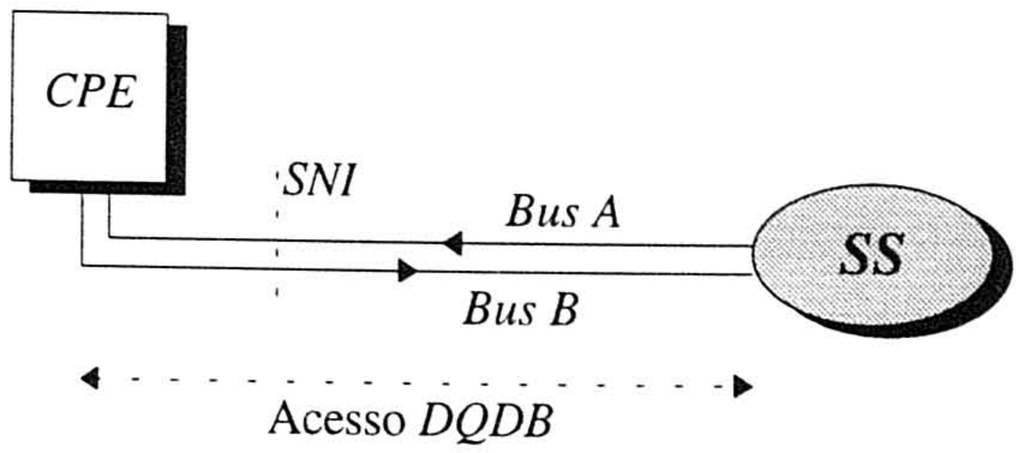
ATM - Asynchronous Transfer Mode
BSS - Broadband Switching System
CPE - Customer Premises Equipment
LAN - Local Area Network
DCN - Data Communicatios Network
DQDB - Distributed Queue Dual Bus
GIO - Generic Interface for Operations
ICI - Interexchange Carrier Interface
ISSI - Inter-Switching System Interface
OS - Operations System
SNI - Subscriber Network Interface
SS - Switching System
UNI - User Network Interface

Fig. 5.1 - Arquitectura de suporte da rede *SMDS*

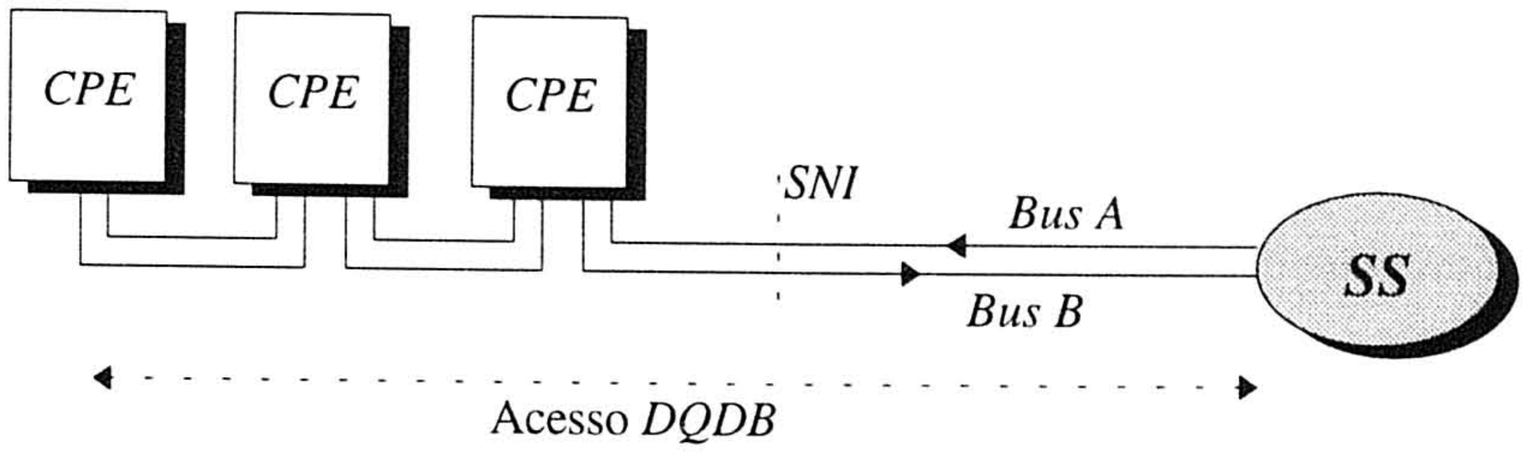
5.2.1. Interface Assinante-Rede

Esta interface define o ponto onde um determinado assinante se liga à rede que oferece o *SMDS*. É no *SNI* que o *CPE* se liga fisicamente ao acesso do sistema de comutação. Esta interface permite duas configurações de acesso distintas: o acesso de um *CPE* simples e o acesso de múltiplos *CPEs* (multi-*CPE*). Em qualquer uma destas configurações de acesso os dados trocados através de uma dada *SNI* de um sistema de comutação não são visíveis aos *CPEs* de outras *SNIs* desse sistema de comutação (Fig. 5.2). Podemos também ter casos de *CPEs* que desempenham funções de *routers* e que servem para fazer a interligação de *LANs* a esse acesso *DQDB*.

O modo como a comunicação é feita entre o(s) *CPE*(s) e o *SS*, através da *SNI*, é definido pelo protocolo de interface *SMDS*.



i - Acesso *CPE* simples



ii - Acesso multi-*CPE*

Fig. 5.2 - Configurações de acesso ao *SMDS*

5.2.2 Protocolo de Interface *SMDS*

A ligação dos equipamentos dos assinantes ao *SMDS* é estabelecida através de um protocolo de interface dedicado, o *SIP*, que executa as funções do nível de controlo de acesso ao meio e do nível físico (Fig. 5.3). Sobre o *SIP* são colocados os protocolos dos restantes níveis, necessários para a interligação dos equipamentos terminais.

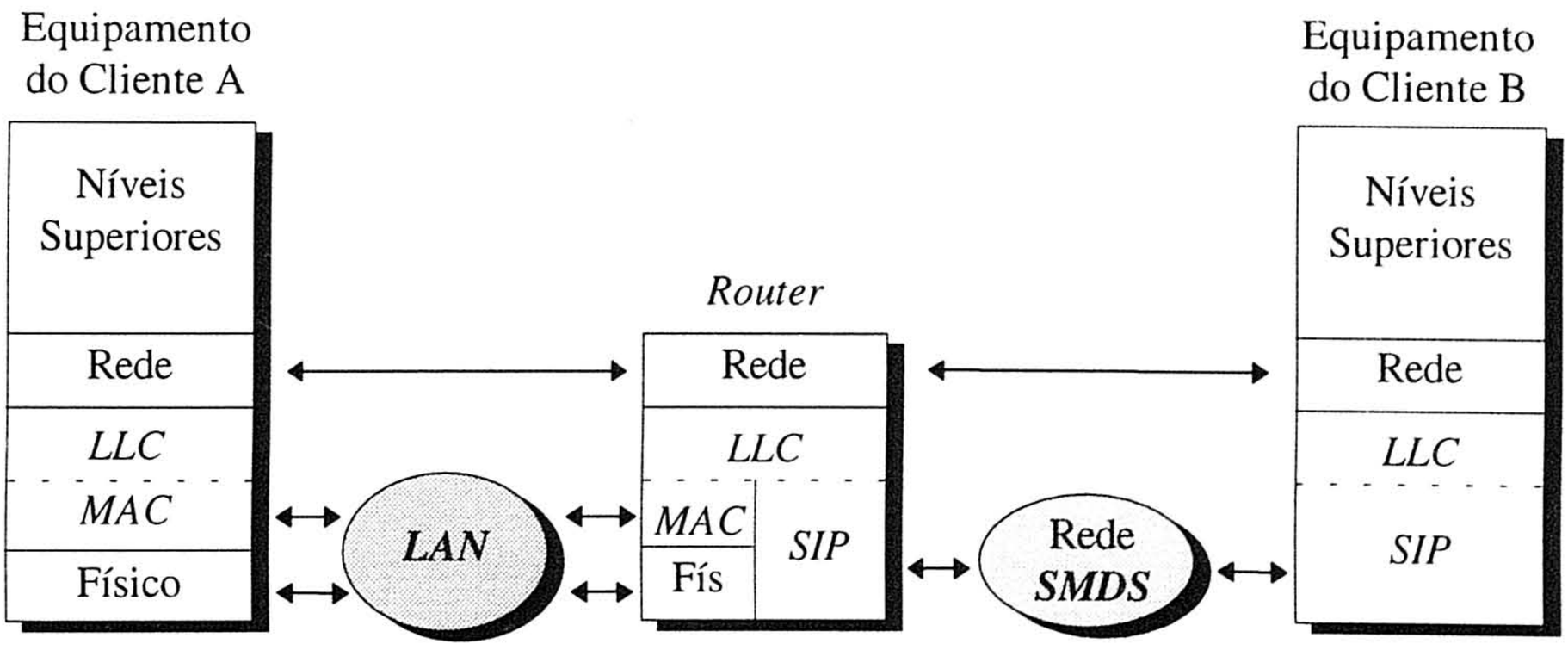
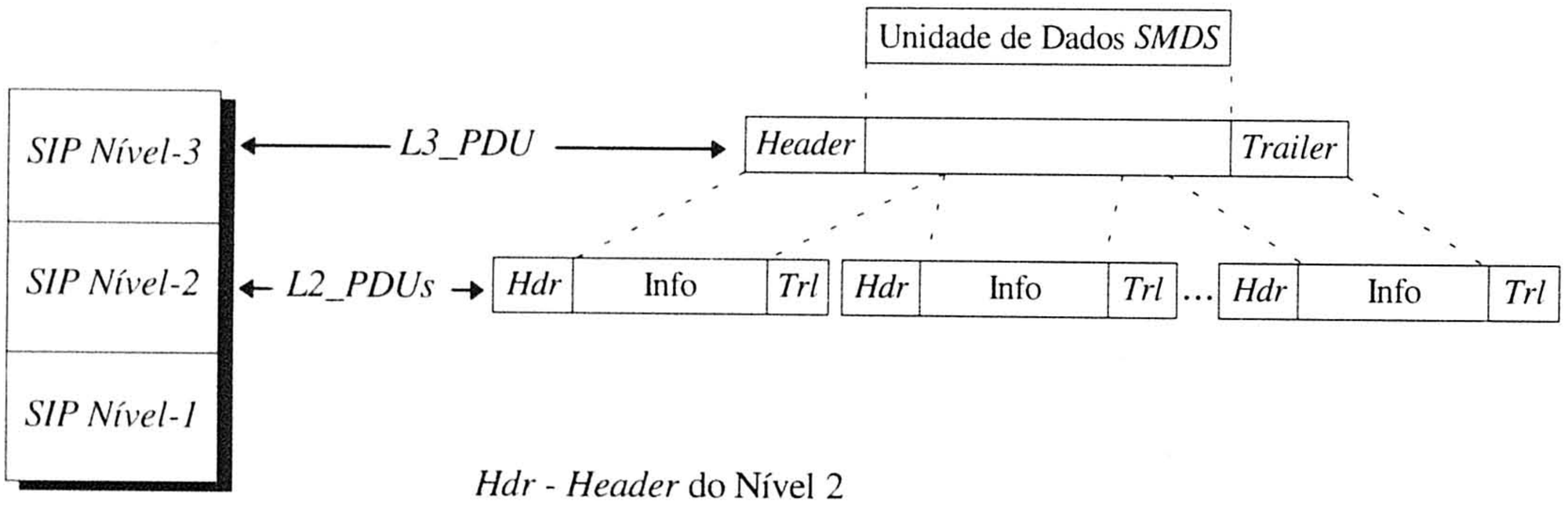


Fig. 5.3 - Interligação de equipamentos terminais pelo *SMDS*

Os níveis *SMDS* não desempenham qualquer tipo de funções de controlo de fluxo ou de recuperação de erros mas dispõem de mecanismos eficazes de detecção de erros com base em códigos *CRC* (*Cyclic Redundancy Check*) [Hemric91].

O *SIP* pode ser dividido em três níveis protocolares distintos (*SIP Nível-1*, *SIP Nível-2* e *SIP Nível-3*), cujas funções incluem o endereçamento, o empacotamento, a detecção de erros e o transporte físico. A individualização dos níveis deste protocolo, que não tem outro objectivo senão o de simplificar a descrição do protocolo, não tem nenhuma relação com os níveis do modelo *OSI*.

Os dois níveis superiores desempenham funções de encapsulamento e desencapsulamento da informação assim que esta passa de uns níveis para os outros (Fig. 5.4). O *SIP Nível-3* contém a informação de endereçamento necessária para as unidades de dados passadas pelo utilizador, bem como mecanismos para detectar a perda de unidades de dados do *SIP Nível-2* (as *L2_PDUs*). O *SIP Nível-2* dispõe de mecanismos de detecção de *bits* errados e de divisão das unidades protocolares do *SIP Nível-3* (as *L3_PDUs*), de tamanhos variáveis, em unidades de tamanho fixo que depois de encapsuladas serão passadas ao *SIP Nível-1*. O *SIP Nível-2* é o responsável pela segmentação e pelo reagrupamento das *L3_PDUs*. O *SIP Nível-1* desempenha funções de nível físico como por exemplo a geração de relógios, o alinhamento de tramas, a geração, codificação e descodificação dos *bits*, etc.



Hdr - Header do Nível 2
Trl - Trailer do Nível 2
L2_PDU -Unidades de Dados do Nível 2
L3_PDU -Unidades de Dados do Nível 3

Fig. 5.4 - Estrutura do *SIP*

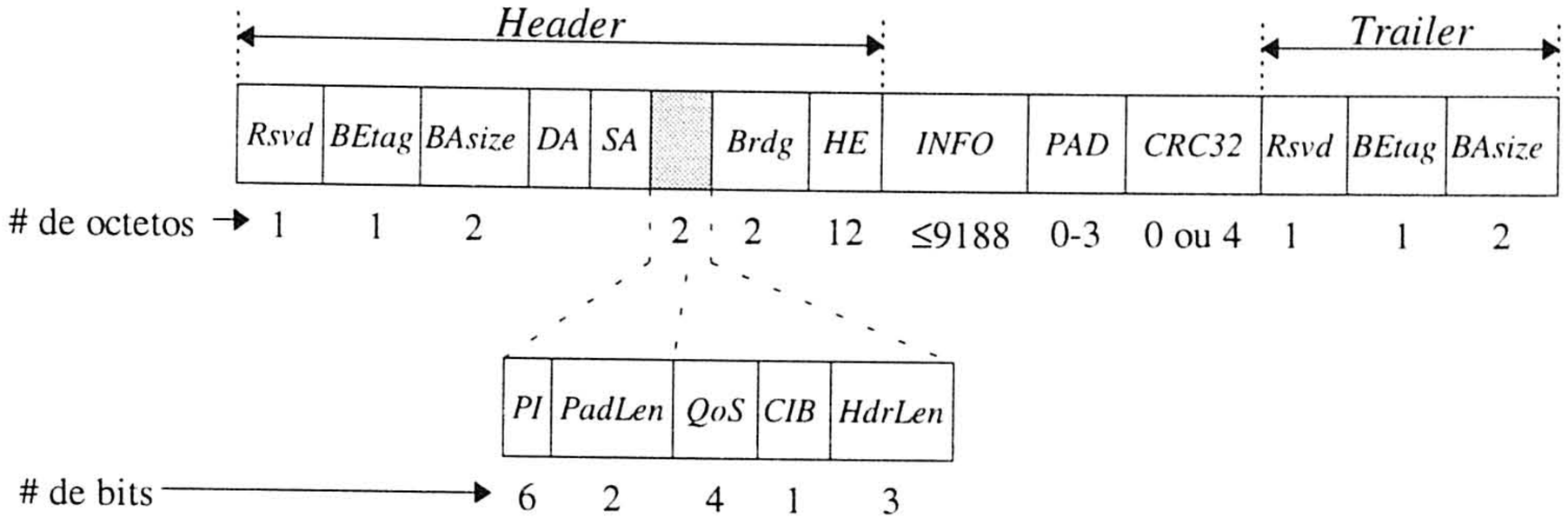
- **Funções do *SIP Nível-3***

O *SIP* Nível-3 associa informação de endereçamento a cada unidade de dados *SMDS*, que no máximo pode conter 9188 octetos de informação.

A comunicação entre os *SIPs Nível-3* de dois sistemas é bastante simples. Assim que um dos sistemas transmite uma *L3_PDU*, o sistema receptor desencadeia um processo de validação dessa

L3_PDU e, de acordo com o resultado do processo, envia-a ou não para os níveis superiores (no caso de aplicações nos *CPEs*) ou retransmitida (no caso dos *routers* existentes nas *LANs*).

Na Fig. 5.5 é ilustrado o formato de uma unidade protocolar do *SIP Nível-3*. Uma descrição mais pormenorizada das funções de cada um dos diferentes campos pode ser obtida em [Bellco91] e [ESIG92].



Rsvd - Reserved
BEtag - Beginnind-End Tag
DA - Destination Address
SA - Source Address
PI - (Higher Layer) Protocol Identifier
PadLen - PAD Lenght
QoS - Quality of Service
CIB - CRC32 Indication Bit
HdrLen - Header Extension Lenght
Brdg - Bridging
HE - Header Extension
INFO - Information
CRC32 - 32bits Cyclic Redundance Check

Fig. 5.5 - Formato de uma *L3_PDU*

O processo de validação consiste na verificação dos vários campos do *header* e do *trailer*, e pode ser diferente nos *SSs* e nos *CPEs*.

É neste nível que são realizadas as funções de endereçamento, que podem ser individuais ou de grupos. Cada endereço individual corresponde unicamente a uma *SNI*; no entanto, a uma *SNI* podem ser atribuídos vários endereços individuais (16 no máximo) caso a configuração envolva vários *CPEs* ligados ao mesmo acesso *DQDB*. No caso deste tipo de configuração (multi-*CPE*), a associação dos vários endereços atribuídos a uma *SNI* aos *CPEs* é da responsabilidade do assinante do acesso e não do *SS*.

Quanto ao endereçamento de grupos, pode dizer-se que é o mecanismo utilizado por um determinado *CPE* para enviar uma unidade de dados *SMDS* para vários receptores de uma só vez. Estas unidades são enviadas para os *SSs* que depois de descodificar os endereços, enviam uma só cópia para cada um dos *SNIs* a onde estão ligados os *CPEs* destino. É aberta uma excepção no caso dos *CPEs* destino se encontrarem no mesmo acesso *DQDB* do emissor; em tal situação os *SSs* não geram uma cópia para enviar de volta para essa *SNI*, pois os destinatários recebem os dados directamente da rede *DQDB*.

O facto de este nível não conter qualquer mecanismo de controlo de fluxo, de sequenciamento, ou de retransmissão é adequado para complementar a comunicação não orientada às ligações comum na maioria das *LANs*.

- **Funções do *SIP Nível-2***

As funções oferecidas por este nível são um pouco mais complexas, uma vez que a transmissão de *L2_PDUs* tem que ser contínua, isto é, no caso de não haver nenhuma *L3_PDU* com informação para transmitir, o *SIP Nível-2* envia para o nível correspondente do outro sistema uma *L2_PDU* vazia.

Se houver uma *L3_PDU* para ser transmitida, o *SIP Nível-2* vai dividi-la em unidades de 44 octetos que identifica por meio de um *header* e um *trailer* que lhes adiciona. As unidades protocolares do *SIP Nível-2* têm sempre um tamanho fixo e igual a 53 octetos (de formato igual ao das unidades de dados do nível 2 do protocolo *DQDB*), cujos campos são ilustrados na figura 5.6. Uma descrição mais pormenorizada desta unidade de dados e da funcionalidade de cada um dos diferentes campos pode ser encontrada em [Bellco91] e [ESIG92].

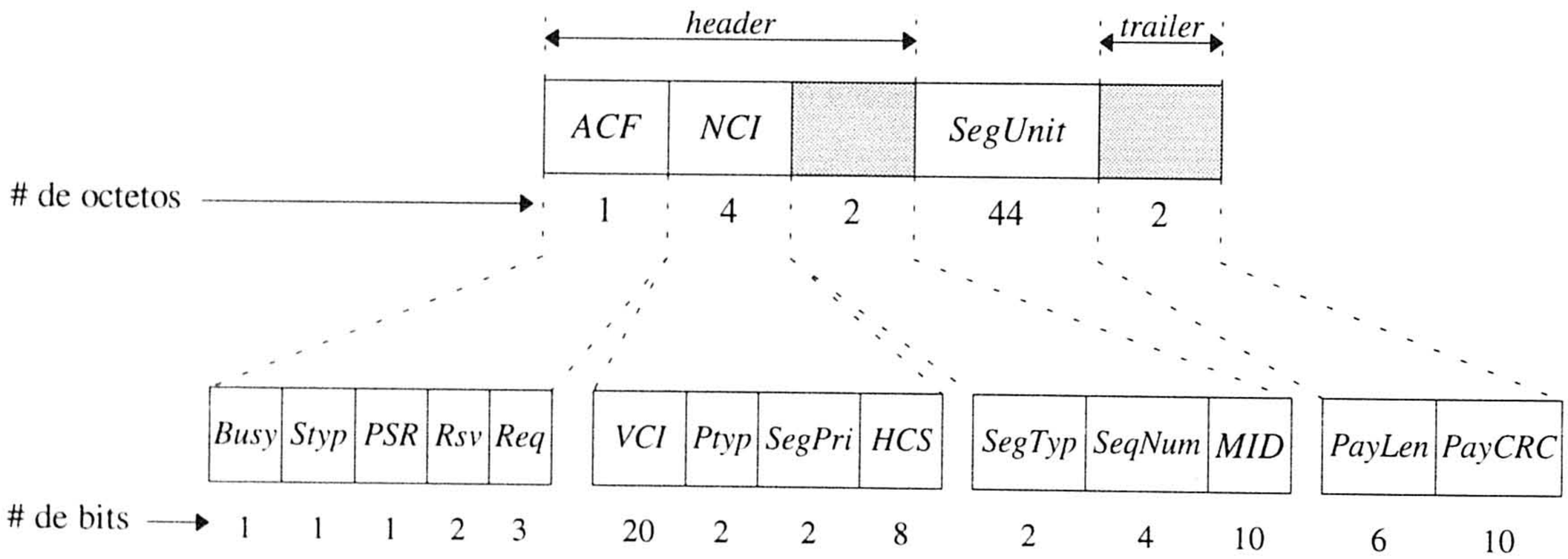


Fig. 5.6 - Formato de uma *L2_PDU*

Existem quatro tipos de *L2_PDUs*, que se distinguem pela parte da *L3_PDU* que contêm no seu campo de informação: o início da mensagem (*BOM - Beginning of Message*), a continuação da mensagem (*COM - Continuation of Message*), o fim da mensagem (*EOM - End of Message*) e a mensagem isolada (*SSM - Single Segment Message*). No caso das *EOMs* e das *SSMs*, se o tamanho da *L3_PDU* não for suficiente para preencher na totalidade os 44 octetos de informação das *L2_PDUs* são

acrescentados zeros para perfazer esse total. Todas as *L2_PDUs* envolvidas no transporte da mesma *L3_PDU* têm igual valor de identificação da mensagem (*MID*). Este identificador de mensagem é muito importante porque permite ao *SMDS* transmitir diferentes unidades de dados em simultâneo para o mesmo acesso, o que é conseguido alternando, no barramento, as mensagens de nível 2 de cada uma das unidades de dados. Deste modo, um utilizador que pretenda enviar para outro um pacote pequeno não necessita de esperar que um terceiro utilizador finalize a transmissão de um pacote de 9188 octetos de comprimento.

O *SIP Nível-2* dispõe de mecanismos para identificar cada uma das *L2_PDUs* que tem de processar, quer seja na recepção ou na transmissão, de modo a poder reconstruir, na recepção, a *L3_PDU*, a partir da reunião da informação das várias *L2_PDUs* recebidas. A sequência da informação contida em diferentes *L2_PDUs* é estabelecida a partir de um contador módulo 16 (*RxSN*), cujo valor é guardado num campo do *header* da *L2_PDU* durante a transmissão. Sempre que é enviada uma destas unidades de dados o contador é incrementado. Assim, na recepção, logo que é recebido um início de mensagem (*BOM*) é extraído do *header* dessa *L2_PDU* o valor inícial do contador e todas as outras *L2_PDUs* dessa *L3_PDU* são aceites ou regeitadas conforme estejam na ordem correcta ou não.

A detecção de erros é feita com base num código *CRC* de 10 *bits*.

Este nível garante a integridade das *L3_PDUs* reconstruídas; no entanto, não dispõe de qualquer tipo de mecanismo de recuperação caso seja detectada uma *L2_PDU* com erros, fora da sequência ou para além de um determinado período de tempo. Se tal acontecer, toda a informação já recebida para a reconstrução de uma *L3_PDU* é desprezada e o processo volta à fase inicial de espera por um novo *BOM*.

- **Funções do *SIP Nível-1***

Este nível do *SIP* tem como função a transmissão e a recepção das *L2_PDUs* através dos barramentos *B* e *A*, respectivamente. Para tal o *SIP Nível-1* pode ser dividido em três subníveis: um de procedimentos de convergência do nível físico (*PLCP - Physical Layer Convergence Procedures*), um de formatação da trama de transmissão segundo a norma [G704_84] e outro de conversão dessas tramas de e para sinais eléctricos de acordo com a norma [G703_72].

Os formatos das tramas do *SIP Nível-1* descritas nesta secção são referentes à interface digital *E1* (2 Mbps) de acesso à rede.

Os procedimentos de convergência do nível físico (*PLCP*) visam mapear informação de controlo do *SIP Nível-1* e as *L2_PDUs* numa trama com um formato apropriado para o subnível de transmissão, a trama *PLCP*.

Na Fig.5.7 é ilustrado o formato global de uma trama *PLCP* de um acesso *E1*. É nesta trama que os 53 octetos de cada *L2_PDU* são encaixados. Os restantes octetos da trama desempenham várias funções de controlo como por exemplo:

- definir o alinhamento das tramas *PLCP* (octetos *A1*, *A2* e *P0* a *P9*);

- detectar erros nos *bits* dos octetos da coluna 4 à 57 da trama *PLCP* anterior (octeto *B1*);
- contar ocorrências de erros detectados (octeto *G1*);
- avisar o sistema remoto de que detectou perda de alinhamento das tramas *PLCP* (octeto *G1*);
- avisar o sistema remoto do estado das ligações (octeto *G1*).

primeiro *bit* →	1 octeto	1 octeto	1 octeto	1 octeto	53 octetos
A1	*A2*	*P9*	*Z4*	*L2_PDU*	
A1	*A2*	*P8*	*Z3*	*L2_PDU*	
A1	*A2*	*P7*	*Z2*	*L2_PDU*	
A1	*A2*	*P6*	*Z1*	*L2_PDU*	
A1	*A2*	*P5*	*F1*	*L2_PDU*	
A1	*A2*	*P4*	*B1*	*L2_PDU*	
A1	*A2*	*P3*	*G1*	*L2_PDU*	
A1	*A2*	*P2*	*M2*	*L2_PDU*	
A1	*A2*	*P1*	*M1*	*L2_PDU*	
A1	*A2*	*P0*	*C1*	*L2_PDU*	← último *bit*

Fig. 5.7 - Formato de uma trama *PLCP E1*

Uma descrição pormenorizada dos valores e funções de cada um dos octetos pode ser encontrada em [Bellco91] e em [ESIG93].

No subnível de construção da trama de transmissão *E1*, as tramas *PLCP* são mapeadas nas tramas de transmissão *E1*, na recepção ocorre a conversão inversa. Estas tramas consistem numa sequência de 32 octetos, denominados *slots*, dos quais dois são usados para alinhamento e detecção de erros (*slot 0*) e para funções de gestão (*slot 16*), e os restantes são octetos de informação. Estas tramas, que podem ser de alinhamento ou não de acordo com *slot 0*, são associadas alternadamente para perfazerem uma multitrama. O conceito de sub-multitrama surge associado a cada uma das duas metades de uma multitrama.

Uma ilustração representativa do formato de uma trama de transmissão *E1* bem como o de uma multitrama é apresentada na Fig. 5.8.

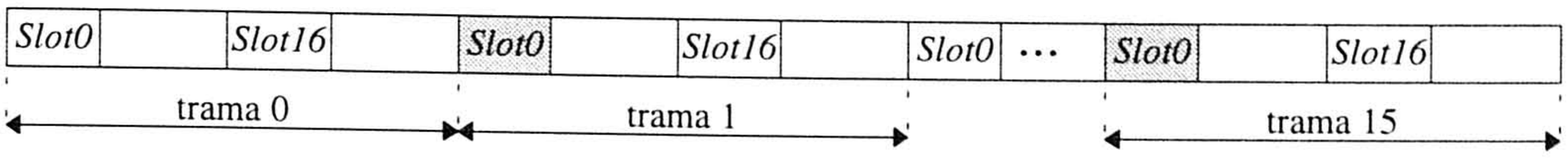
Slot 0	*Slots 1* a *15*	*Slot 16*	*Slots 17* a *31*

Valores dos bits do *Slot 0*

Trama de alinhamento	*Si*	0	0	1	1	0	1	1
Trama de não-alinhamento	*Si*	1	*A*	*Sa4*	*Sa5*	*Sa6*	*Sa7*	*Sa8*

Si - Bits de alinhamento de multitrama e de *CRC-4*
A - *Remote Alarm Indication*
Sa4..Sa8 - Bits de monitoração e gestão

i - Trama de transmissão *E1*



ii - multitrama de transmissão *E1*

Fig. 5.8 - Formato das tramas *E1*

O mapeamento das tramas *PLCP* é feito directamente nos octetos de informação da trama de transmissão de modo a que o *header* da trama *PLCP* fique imediatamente a seguir ao *slot 0* da trama de transmissão *E1*.

Este subnível dispõe de mecanismos para determinar e monitorar o alinhamento das tramas e multitramas de transmissão que estão definidas na norma [G706_88].

No último subnível cada um dos *bits* da trama de transmissão é codificado e descodificado do código de linha *HDB3* de acordo com a norma [G703_72].

A ligação física entre o *CPE* e o *SS* pode ser feita por uma linha de *120 Ohm* balanceada e conectores de 8 contactos, ou então por uma linha não-balanceada de *75 Ohm* e conectores de tipo coaxial.

5.3. Filtragem de Endereços

O *SMDS* dispõe de um mecanismo que permite ao sistema de comutação (*SS*) restringir a transmissão de unidades de dados a partir de determinados endereços e para derminados destinos. Este mecanismo baseia-se na utilização de dois tipos de listas de filtragem de endereços, uma de endereços individuais e outra de endereços de grupo. A primeira é usada para seleccionar os endereços de destino das unidades de dados enviadas por um *CPE* e os endereços de origem das unidades por ele recebidas. Esta lista consiste num conjunto de endereços individuais, que poderão ser todos os endereços de destino permitidos a um assinante ou todos os que esse assinante não pode utilizar durante as comunicações. As listas de endereços de grupos desempenham funções idênticas ao definir os vários endereços de grupos que podem ser utilizados nas transferências de dados para um determinado *CPE*. As considerações são simétricas caso se opte por um ou por outro tipo de lista e, por isso, serão descritos apenas os comportamentos para o caso das listas conterem os endereços permitidos.

A definição das listas de filtragem de endereços é acordada entre o assinante e o fornecedor do acesso à rede *SMDS* na fase de instalação. Se é estipulada a utilização das listas de filtragem de

endereços, individuais ou de grupos, numa determinada *SNI*, todas as unidades de dados *SMDS* que passarem através dela serão submetidas a essa filtragem.

Um assinante pode utilizar mais do que uma lista de endereços individuais ou de grupos, desde que especifique quais as listas estão associadas a determinados endereços. Na filtragem dos endereços de destino, a utilização de mais do que uma lista significa que qualquer uma delas pode ser usada na filtragem das unidades de dados, dependendo apenas do endereço fonte (*Source Address*) dessas unidades. Esta filtragem executada pelos *SSs* consiste em receber a unidade de dados proveniente do *CPE* e verificar se o endereço de destino nela contido pertence à lista de filtragem de endereços, individuais ou de grupo, associada ao endereço do *CPE* que enviou essa unidade de dados. Só após esta verificação é que os dados poderão ser enviados pela rede *SMDS*.

Na filtragem dos endereços fonte, a utilização de mais do que uma lista de filtragem tem rigorosamente o mesmo significado que na filtragem dos endereços destino mas, desta vez, o que acontece depende do endereço destino (*Destination Address*) de cada unidade de dados *SMDS* enviada para a *SNI* do assinante. Assim, os *SSs* só enviam cada unidade de dados ao respectivo *CPE* destino depois de verificarem se o endereço destino dessa unidade pertence à lista de filtragem de endereços, individuais ou de grupos, associada ao endereço fonte dessa unidade. No caso concreto de utilização de endereços de grupo, poderão existir *SNIs* desse grupo cujas listas de filtragem de endereços não permitam que essa unidade de dados lhes seja entregue; por isso, cópias dessa unidade de dados só são enviadas para as *SNIs* desse grupo que permitirem a sua entrega.

5.4. Classes de Acesso

Picos de tráfego surgem quando uma ou várias unidades de dados *SMDS* são transferidas sucessivamente através da *SNI* sem intervalos entre elas. Para controlar estes picos o *SMDS* utiliza um mecanismo de restrição da quantidade de tráfego injectado na rede durante determinados períodos.

Para o acessos à rede de tipo *E3* (a 34 Mbps), o *SMDS* prevê uma gama de classes de acesso que não são mais do que um processo de limitar as taxas de transferência de informação entre o *CPE* e o *SS* e o número médio de picos de tráfego existentes durante essa transferência de informação. Cada classe pode ser vista como um escalão de acesso à rede com um determinado débito. Este débitos são definidos só em termos médios, isto é, se um *CPE* quiser transmitir tráfego este é transmitido à taxa máxima permitida pelo serviço (25 Mbps se não se considerar o *overhead*), desde que a média do tráfego gerado durante determinado período não ultrapasse o valor estipulado para essa classe de acesso. Assim, o assinante e o operador da rede devem definir, de acordo com as necessidades do primeiro, a classe de acesso mais favorável de modo a permitir uma alocação mais eficiente dos recursos da rede.

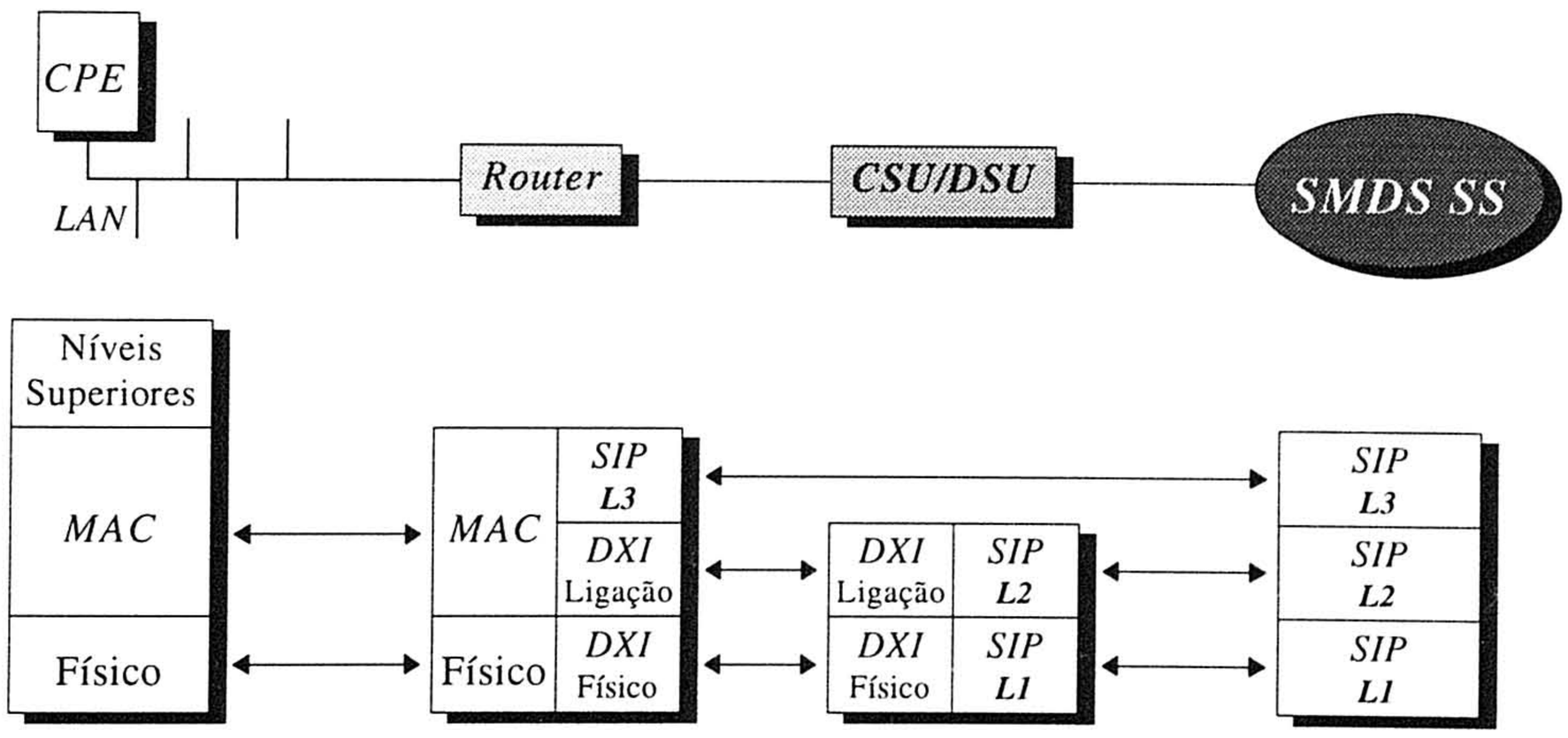
As classes de acesso definidas para um acesso *E3* são 4: classe 1 (4 Mbps), classe 2 (10 Mbps), classe 3 (16 Mbps) e classe 4 (25 Mbps). As classes 1, 2 e 3 definem um conjunto de débitos bastante úteis para a interligação de diferentes *LANs*. Assim, o tráfego gerado numa rede *Token Ring* (*IEEE 802.5*) a 4 Mbps e 16 Mbps poderia aceder ao *SMDS* através de acessos das classes 1 e 3, respectivamente, enquanto que o gerado por uma rede *Ethernet* (*CSMA/CD - IEEE 802.3*) a 10 Mbps acederia ao *SMDS* recorrendo a um acesso de classe 2.

Na classe 4 não existe qualquer tipo de restrição no acesso, isto é, sempre que um *CPE* disponha de informação para transmitir, esta é transmitida assim que conseguir o acesso ao meio, mesmo que isso provoque picos de tráfego.

5.5. Acesso de *LANs* ao *SMDS*

A interligação de *LANs* situadas em locais geograficamente distantes é, como já foi referido, uma das grandes utilidades do *SMDS*. Para tal é necessário garantir que o acesso dessas *LANs* ao *SMDS* seja feito correctamente e de forma eficaz. Nesta secção são abordadas duas configurações possíveis para acesso de uma *LAN* ao *SMDS*: na primeira, a *LAN* é ligada directamente ao sistema de comutação *SMDS*; na segunda, essa *LAN* é já cliente de uma rede *ATM* através da qual se pretende ligar ao *SMDS*.

De um modo geral, um cliente necessita de um *router* e de uma unidade de serviços de canal e de dados (*CSU/DSU - Channel Service Unit/Data Service Unit*) para ligar a sua *LAN* ao sistema de comutação [Wright93] (Fig.5.9).



CPE - Customer Premises Equipment
LAN - Local Area Network
CSU - Channel Service Unit
DSU - Data Service Unit
SMDS SS - SMDS Switching System

MAC - Nível *MAC* da *LAN*
Físico - Nível físico da *LAN*
DXI - Data Exchange Interface
SIP - SMDS Interface Protocol

Fig. 5.9 - Ligação de uma *LAN* a um acesso *DQDB*

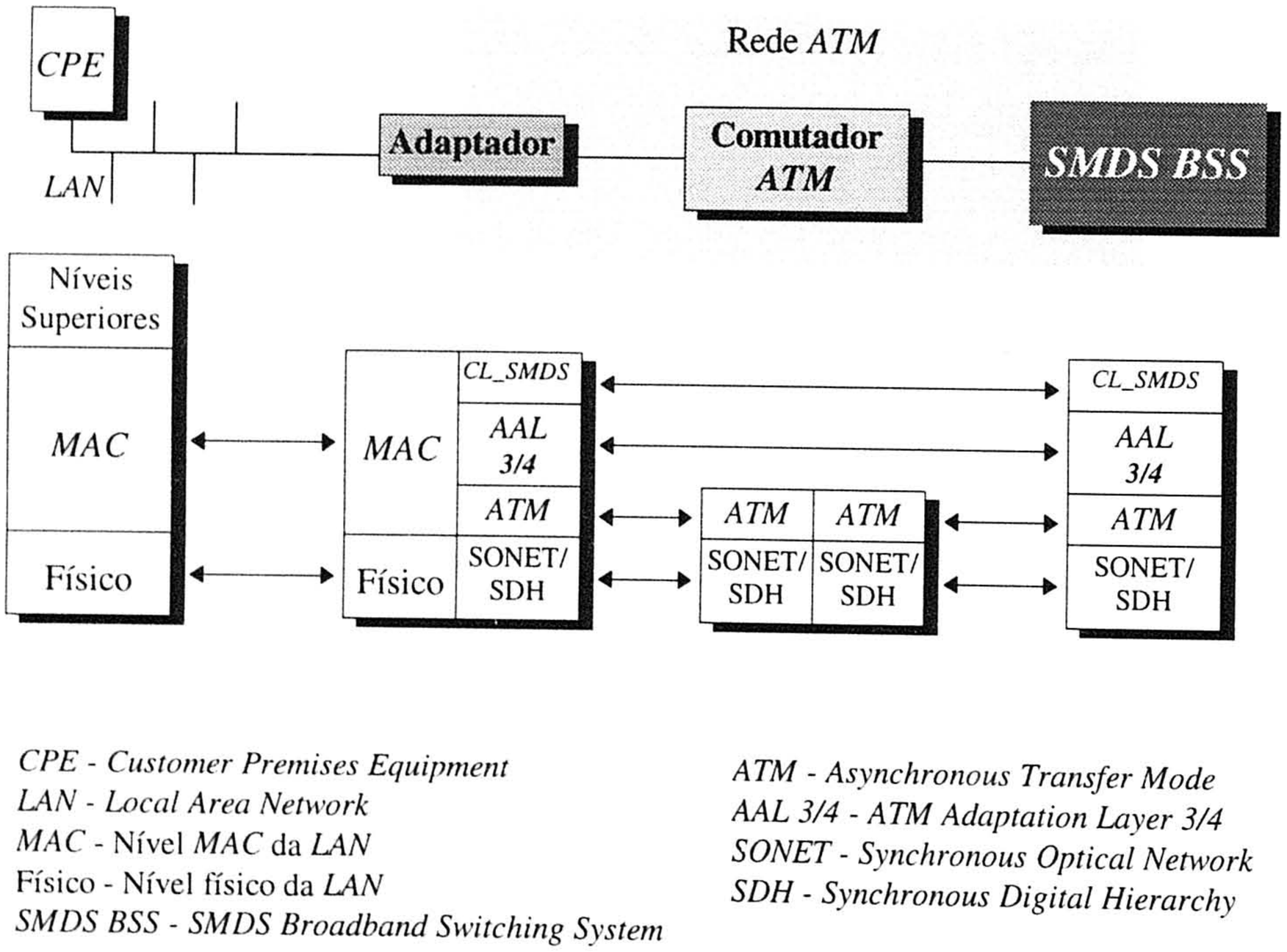
O *router* desempenha as funções do *SIP Nível-3*, ou melhor, recebe um pacote de dados da *LAN* e converte-o para uma unidade protocolar de dados de nível 3 (*L3_PDU*) do *SMDS*, adicionando-lhe os *header* e *trailer* respectivos.

A *CSU/DSU* faz a adaptação dos níveis 1 e 2 do *router* aos níveis correspondentes do *SIP*, por meio de interfaces específicas para troca de dados (*DXI - Data Exchange Interface*). Estas interfaces de adaptação são necessárias quando o *router* e a *CSU/DSU* são desenvolvidos por construtores diferentes. A *CSU/DSU*, após receber os pacotes de dados do nível 2 do *router*, vai convertê-los em segmentos *DQDB* e envia-los para o sistema de comutação da rede *SMDS*.

Na outra situação, em que a *LAN* dispõe de um acesso a uma rede *ATM* e pretende usufruir do *SMDS*, poderá fazê-lo através de um sistema de comutação *SMDS* de banda larga (*BSS*).

Numa rede RDIS-BL um serviço não orientado às ligações, como o *SMDS*, é designado como sendo de classe D e é garantido pelo nível de adaptção *ATM 3/4* (*AAL 3/4*).

Os pacotes do cliente são enviados pela *LAN* até um adaptador, que os converte para o formato *ATM* de modo a poder transpor os vários comutadores dessa rede. Numa situação destas é apenas exigido que o adaptador acrescente a informação de endereçamento *SMDS* a esse pacote. Como o nível *AAL 3/4* não fornece essa informação *SMDS*, o adaptador deve dispor de um outro nível, o *CL_SMDS*, para a acrescentar aos dados (Fig.5.10).



CPE - Customer Premises Equipment
LAN - Local Area Network
MAC - Nível *MAC* da *LAN*
Físico - Nível físico da *LAN*
SMDS BSS - SMDS Broadband Switching System
ATM - Asynchronous Transfer Mode
AAL 3/4 - ATM Adaptation Layer 3/4
SONET - Synchronous Optical Network
SDH - Synchronous Digital Hierarchy

Fig. 5.10 - Acesso de cliente *ATM* ao *SMDS*

Assim que toda a informação *SMDS* necessária seja adicionada aos dados, é construída a trama do nível *AAL* 3/4 para tráfego da classe D. De seguida os pacotes são divididos em células *ATM* que, podendo transpor os comutadores dessa rede, são enviadas para o sistema de comutação de banda larga da rede *SMDS*. Este encarrega-se de interpretar a informação de endereçamento de cada um dos pacotes e, em função dela, encaminha-o pelas interfaces *ISSI* ou *ICI* para o(s) comutador(es) destino.

Um último método que parece ser bastante promissor devido ao seu baixo custo e facilidade de aplicação, recorre a placas *SMDS* para computadores pessoais, que implementam os três níveis do *SIP* e permitem ao cliente ligar-se de uma forma muito directa ao acesso *DQDB* [Guimar95][Guimar95a].

Capítulo 6

Testes de Conformidade e Interoperabilidade com o SGDT

É sabido que o fabrico de sistemas de comunicação obriga a que sejam respeitadas as especificações por parte dos implementadores e que ferramentas para avaliar essas implementações são instrumentos muito valiosos durante a fase de desenvolvimento. Nesta fase, o implementador preocupa-se apenas em seguir uma especificação e obter um conjunto de resultados nela previstos. Só após este período de desenvolvimento é que se inicia a fase de integração, em que as novas implementações são submetidas às condições de interoperabilidade através das redes de comunicação.

Face à importância que os testes, tanto de conformidade como de interoperabilidade, têm vindo a assumir durante a fase de desenvolvimento de novos produtos, havia todo o interesse em expandir o leque de potencialidades do SGDT, de modo a que este dispusesse de mecanismos que lhe permitissem definir e executar estes tipos de testes.

Como foi descrito no capítulo 4, o SGDT é uma ferramenta que foi desenvolvida para executar testes de avaliação do desempenho e da qualidade de serviço de uma rede. Esta ferramenta dispõe de uma interface onde o utilizador caracteriza as sessões de teste que pretende executar, isto é, sempre que se pretenda avaliar uma rede com este equipamento, é necessário descrever o conjunto de testes a efectuar em ficheiros de sessões. Embora a interface seja simples de operar, o formato dos ficheiros de sessão não é normalizado; como tal, as possibilidades de integração deste equipamento de teste com outros sistemas são bastante reduzidas. Este facto leva a que o operador deste equipamento se veja obrigado a converter outros ficheiros de configuração de teste para o formato das sessões, ou então a descrever o teste neste formato de acordo com a sua interpretação e o seu método. Por outro lado, os conteúdos dos ficheiros de sessões não são mais do que um conjunto de parâmetros que vão caracterizar um determinado tipo de tráfego e, embora o operador possa definir novos tipos de tráfego, não lhe é possível definir outro tipo de mensagens para além das utilizadas para gerar os vários tipos de tráfego.

Atendendo a estas limitações e ao interesse em dotar o SGDT com a possibilidade de executar testes de conformidade e de interoperabilidade, e já que existe uma linguagem normalizada para descrever testes de conformidade, o *TTCN*, foi julgado importante acrescentar ao SGDT um mecanismo que lhe permitisse executar testes descritos nesta notação. Deste modo, o SGDT passaria a dispor de uma interface normalizada que poderia ser utilizada para interpretar testes descritos por qualquer operador desde que estes obedecessem às regras do *TTCN*. De igual modo este equipamento de teste permitiria ao operador definir qualquer tipo de pacotes de informação e enviá-los ou recebê-los sobre o serviço de comunicação utilizado.

Para alargar ainda mais as potencialidades do SGDT, optou-se por utilizar este equipamento conjuntamente com o *ITEX*. Esta aplicação, já referida na secção 3.2., permite editar, por meio de um sistema de tabelas, um conjunto de testes que posteriormente serão executados pelo SGDT. Com o *ITEX* é possível gerar, com base nas tabelas de descrição do conjunto de testes, código *ANSI C* que, por sua vez, é adaptado para interagir com o SGDT e dar origem a um conjunto de testes executáveis. A utilização do *ITEX* como meio para comunicar com o operador acrescenta uma série de vantagens ao SGDT, nomeadamente: dispor de meios para importar ficheiros de testes descritos em *TTCN.MP*, converter correctamente estes ficheiros para o formato tabelar (*TTCN.GR*), mais simples de analisar e alterar, e todas as restantes funcionalidades do *ITEX*.

Numa análise mais objectiva, pode dizer-se que o SGDT, que era uma ferramenta que só permitia executar testes de avaliação da qualidade de serviço e do desempenho de uma rede, passa a dispor de mecanismos que lhe permitem executar também testes de conformidade e de interoperabilidade. Além disso, e talvez o mais importante, esta ferramenta, que inicialmente operava a partir da definição de ficheiros de sessão com um formato próprio, passa agora a fornecer um acesso normalizado que possibilita a execução de qualquer conjunto de testes descritos em *TTCN* por qualquer operador ou programador.

A interacção entre o *ITEX* e o SGDT é feita através de um conjunto de funções de adaptação definidas pelo *ITEX* para esse efeito. Estas funções são invocadas no código *C* gerado pelo *ITEX* e referente a cada conjunto de testes, mas são desenvolvidas de acordo com as características físicas do SGDT.

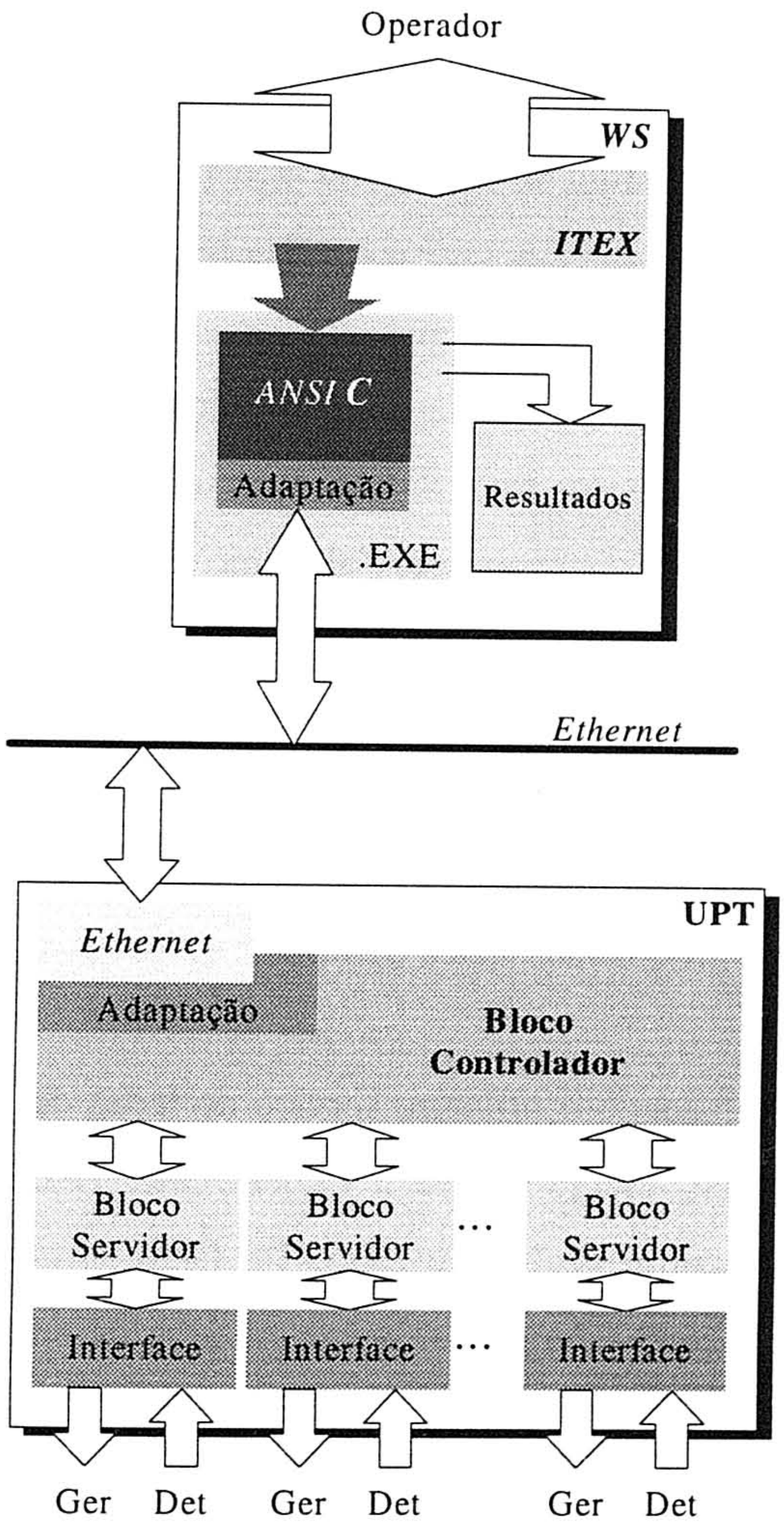
6.1. Adaptação do SGDT

A arquitectura do novo SGDT para a execução de testes de conformidade e de interoperabilidade não é diferente da previamente utilizada, sendo no entanto alteradas as funcionalidades de cada uma das entidades constituintes. Assim, a estação de trabalho (*WS*) passa a ser o suporte do *ITEX*, onde são descritos (editados, alterados e corrigidos) os conjuntos de testes, e onde é gerado o código *ANSI C* que,

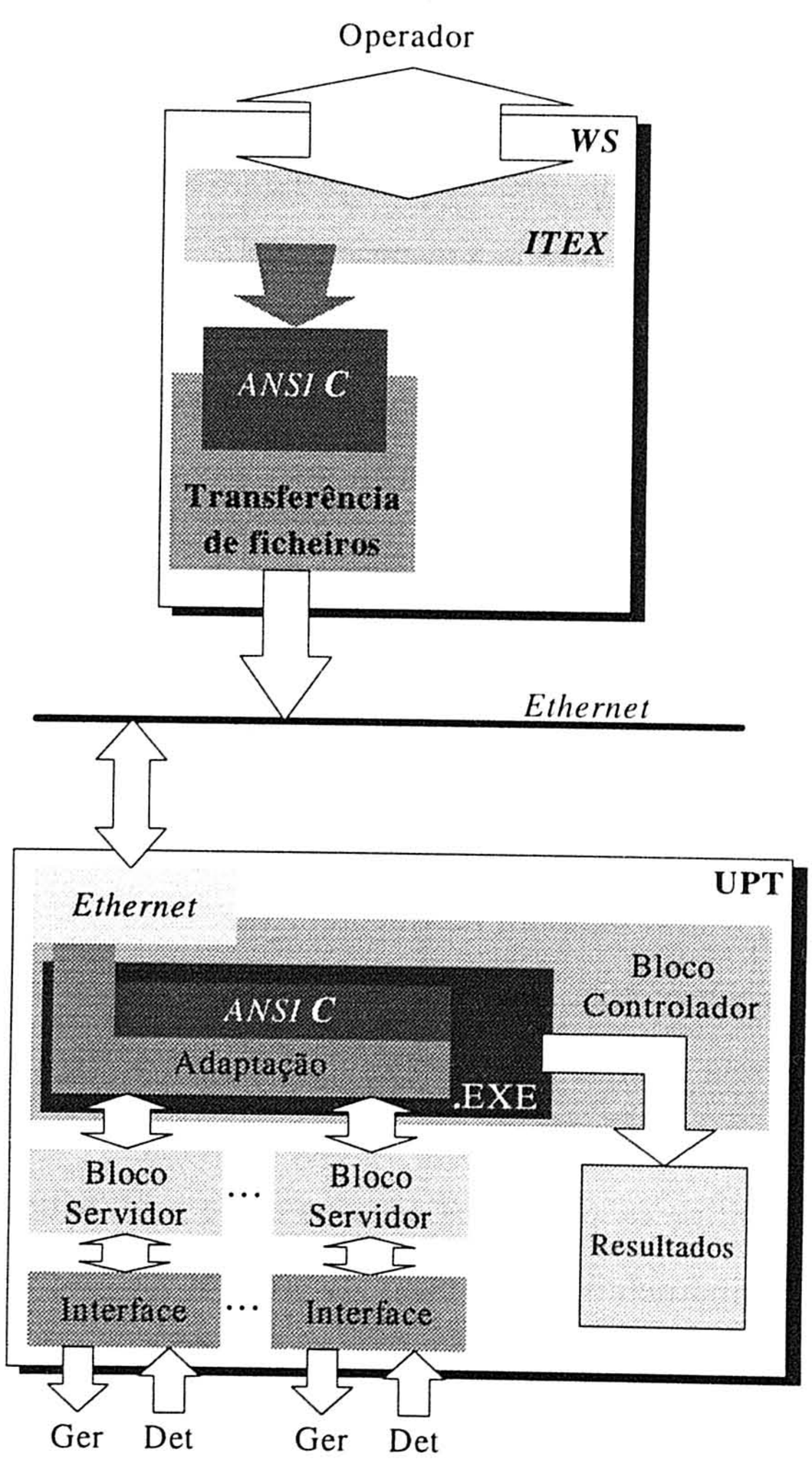
posteriormente, será ligado com as funções de adaptação ao SGDT. De um modo análogo à anterior estrutura do SGDT, a interface do sistema com o operador reside na *WS* durante o perído de definição dos conjuntos de teste, e pode residir nas unidades de processamento de tráfego (UPTs) nos períodos de execução desses testes.

Os métodos possíveis para execução dos testes são dois: no primeiro, o código *C* é gerado e tornado executável (.EXE) na *WS* e, no segundo, esse código é apenas gerado na *WS* e é tornado executável nas unidades de processamento de tráfego. As funções de adaptação têm objectivos distintos de acordo com o método utilizado para executar os testes.

A selecção do primeiro método de teste obriga a que as funções de adaptação garantam a comunicação entre a *WS* e as várias UPTs envolvidas no teste, uma vez que a versão executável do código gerado é posta em funcionamento a partir da *WS* (Fig. 6.1). Os resultados dos testes são também apresentados na *WS* de acordo com os dados transmitidos e recebidos por cada uma das UPTs.



i - Código a operar a partir da *WS*



ii - Código a operar a partir da UPT

Fig. 6.1 - Métodos de execução de teste com o código gerado

O segundo método, por sua vez, requer a utilização de *software* para a transferência do código *ANSI C* gerado na *WS* para cada uma das unidades de processamento de tráfego utilizadas no teste. Com este método as funções de adaptação não necessitam de garantir a comunicação com a *WS*, mas no caso de haver mais do que uma UPT envolvida no teste é preciso garantir a comunicação entre elas. A geração de ficheiros de resultados neste caso é feita nas UPTs. De acordo com a arquitectura do SGDT é necessário definir individualmente as funcionalidades de cada uma das unidades de processamento de tráfego, o que poderá ser conseguido por dois processos: por meio da selecção de uma unidade mestre, onde irá ser colocado em funcionamento o código executável, e que dispõe de funções de adaptação para as restantes UPTs, ou por meio da definição de vários conjuntos de teste dedicados a cada uma das UPTs, o que exige que estas disponham de mecanismos de sincronização mútua. Esta última hipótese pode levantar alguns problemas quanto à análise dos resultados, uma vez que é natural que cada UPT disponha do seu ficheiro, bem como quanto à atribuição de um veredicto final do teste.

A arquitectura das unidades de processamento de tráfego é idêntica à previamente utilizada no SGDT; apenas as funcionalidades dos Blocos Controlador e Servidores são alteradas. De acordo com o método escolhido, essas podem variar mas existe um conjunto de funções que, sendo comuns a todas as UPTs, são as que garantem a programação de cada um dos Blocos Servidores e a transferência da informação entre a rede a testar e a *WS*.

Para se executarem testes de conformidade de protocolos é necessário que seja fornecido um serviço que garanta a comunicação entre o sistema de teste e o equipamento com a implementação a testar (ver secção 2.1.6). O serviço seleccionado para desempenhar tais funções foi o *SMDS*. Esta selecção teve em consideração alguns factores importantes, tanto do ponto de vista operacional como comercial, que caracterizam o *SMDS*.

Numa perspectiva operacional, este serviço é adequado por garantir transparência e segurança na transferência da informação necessária para a execução de testes de conformidade e de interoperabilidade. A qualidade deste serviço [Ferrei95] e a possibilidade de abranger áreas muito vastas (*WANs*) foram também factores considerados por permitirem efectuar testes de implementações situadas em locais remotos.

Numa análise mais comercial, o *SMDS* aparece como sendo o primeiro serviço de redes de banda larga a ser comercializado pelas operadoras. Em Portugal, espera-se que a *PORTUGAL TELECOM* disponibilize os primeiros acessos a 2 e a 34 Mbps no decorrer de 1994 e 1995. Tal facto permite que este equipamento possa estar disponível na fase em que começarem a ser desenvolvidos os primeiros equipamentos para operarem sobre o *SMDS*.

Com base nesse serviço foram desenvolvidas primitivas para implementação das funções de transmissão e recepção de unidades de dados protoculares do *SIP Nível-3* (*L3_PDUs*). As funções do *SIP Nível-2* foram, na sua maioria, implementadas nos Blocos Servidores em *software* do *TMS320C25*, enquanto que as do *SIP Nível-1* foram integralmente implementadas na placa de interface.

6.2. Interface SGDT/*SMDS*

A interface SGDT/SMDS foi implementada numa placa dedicada, denominada placa *SMDS*. Esta implementação, para a qual o autor deste texto contribuiu significativamente, tem por função fazer a interligação da UPT ao *SMDS* permitindo, deste modo, ao SGDT executar testes de avaliação da qualidade e do desempenho desse serviço. Esta interface implementa todas as funcionalidades do *SIP Nível-1* para um acesso *E1*, algumas do *SIP Nível-2*, e suporta transmissão e recepção simultâneas.

A placa *SMDS* foi desenvolvida especificamente para adaptar o Bloco Servidor já existente a esse serviço; por isso, atendendo às características e aos recursos desse bloco, e tendo também em consideração as exigências dos *SIPs Níveis 1* e *2*, foram implementadas algumas das funções do *SIP Nível-2*. De um modo geral, pode dizer-se que a placa *SMDS* opera sobre alguns dos campos das unidades protocolares do *SIP Nível-2*, as *L2_PDUs*, e à custa destes gera os restantes campos necessários para a transmissão. Na recepção, os campos das *L2_PDUs* são filtrados, isto é, só alguns deles são enviados para o Bloco Servidor.

A estrutura global da placa *SMDS* envolve uma interface de comunicação entre esta e o Bloco Servidor da UPT, um bloco responsável pela transmissão e outro pela recepção, um registo do estado da placa e as interfaces à linha. Um diagrama representativo desta estrutura é ilustrado pela Fig. 6.2.

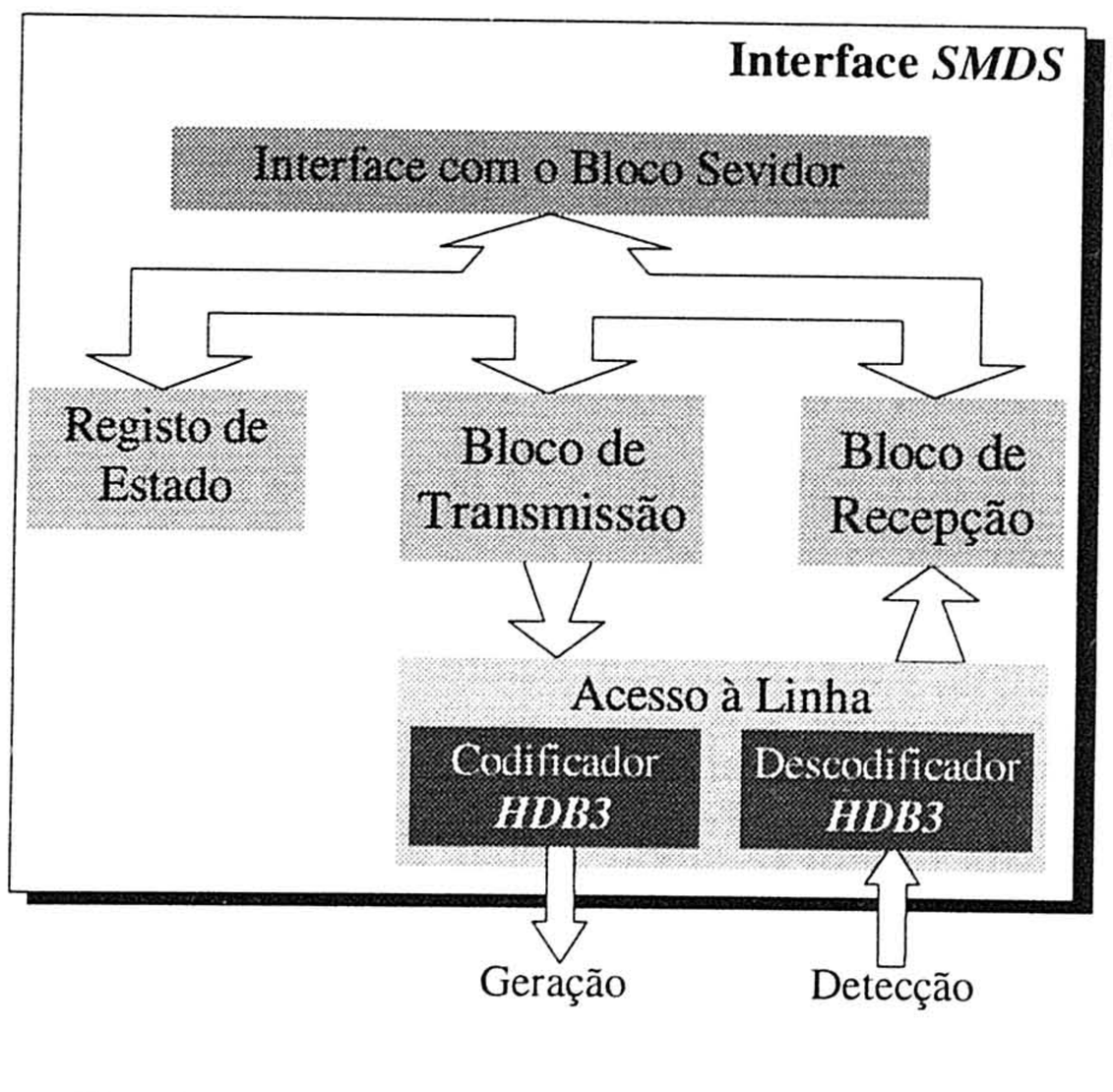


Fig. 6.2 - Estrutura global da Interface *SMDS*

Os blocos de transmissão e de recepção funcionam com base num relógio de 8.192 MHz que, no caso da recepção, é recuperado a partir do sinal *HDB3* recebido e, no caso da transmissão, pode ser também o recuperado na recepção ou um gerado externamente, com um oscilador. A hipótese de

utilização do oscilador externo só é colocada se existirem problemas de comunicação e o sinal recebido não permitir a recuperação do relógio, assegurando-se, deste modo, a qualidade da transmissão.

6.2.1. Interface com o Bloco Servidor da UPT

A ligação da placa *SMDS* à UPT é feita através de um conector. Assim é garantida uma certa flexibilidade na adaptação do SGDT a diferentes redes, sendo para tal apenas requerido o desenvolvimento de interfaces que implementem cada um dos protocolos de acesso a essas redes.

Esta ligação envolve vários tipos de sinais, em ambos os sentidos, que podem ser de dados, de endereçamento ou de controlo.

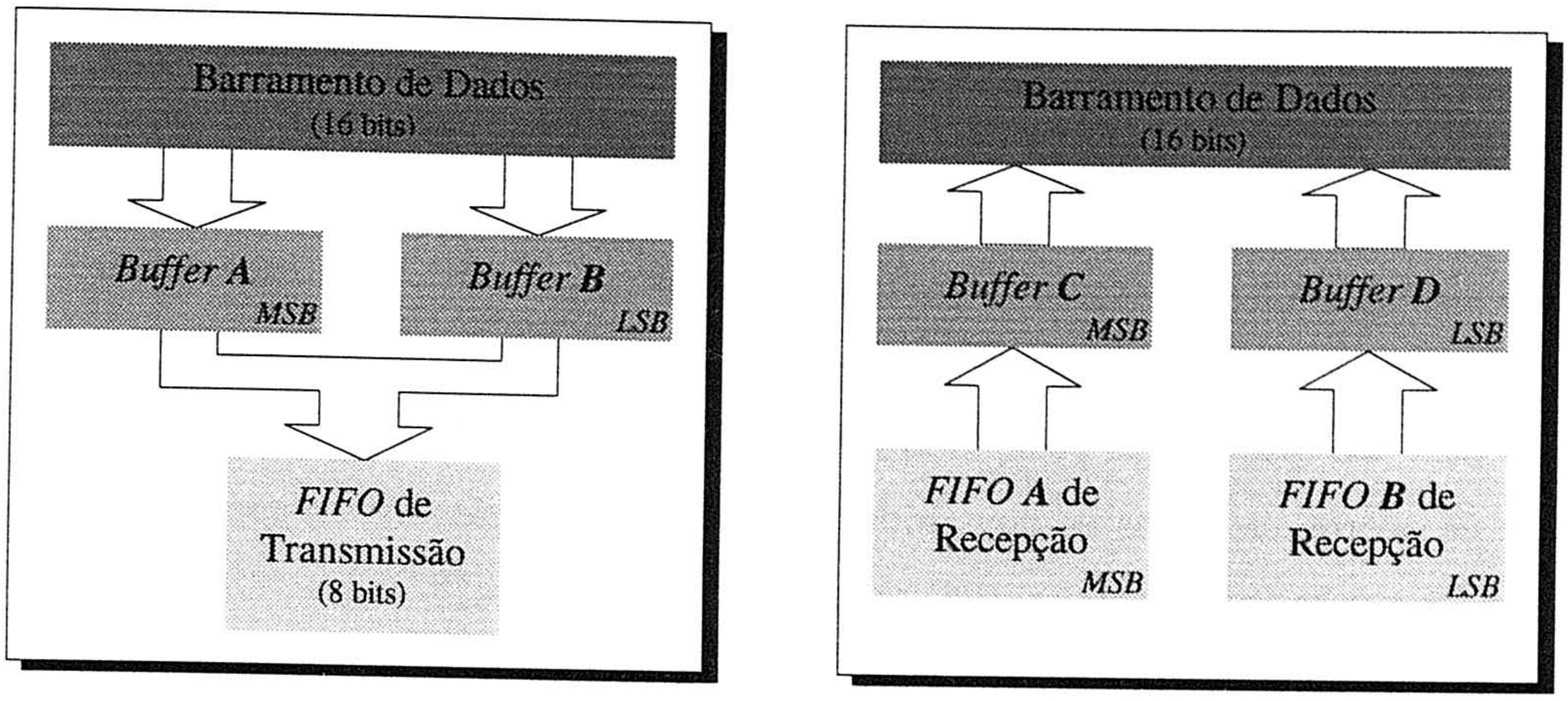
Esta interface assenta na utilização de memórias nela existentes através das quais são efectuadas as transferências de informação, sendo usadas memórias distintas para a transmissão e para a recepção. Na transmissão, a informação é enviada para a memória respectiva pelo Bloco Servidor, sendo activado um sinal que alerta para a sua existência os restantes mecanismos da placa de interface. Na recepção, o processo é simétrico, isto é, a informação recebida pela placa de interface é escrita nas memórias de recepção e, assim que estiver armazenada, é enviado um sinal de aviso ao Bloco Servidor para que este a venha ler.

Essas memórias, tanto de transmissão como de recepção, são do tipo *fifo* (*first in first out*), assíncronas e organizadas em 512 palavras de 9 *bits*, nas quais a informação é armazenada em octetos.

O acesso do Bloco Servidor tanto às *fifos* de transmissão como de recepção, é efectuado através de *buffers* de 8 *bits*.

Estes *buffers* são utilizados na dissociação das palavras de 16 *bits* em octetos de forma a serem armazenados na *fifo* de transmissão, e na associação dos octetos armazenados nas *fifos* de recepção de forma a construir palavras de 16 *bits*. Esta adaptação é necessária porque o Bloco Servidor, bem como o barramento só operam com palavras de 16 *bits*, enquanto que o armazenamento e o processamento dos dados na placa *SMDS*, a este nível, é realizado com octetos.

São utilizadas 3 *fifos*, uma para a transmissão e as restantes para a recepção. Na *fifo* de transmissão os octetos são armazenados pela ordem segundo a qual vão ser transmitidos (octeto mais significativo primeiro), enquanto que nas *fifos* de recepção os octetos são associados em palavras de 16 *bits*, sendo distribuídos alternadamente pelas duas *fifos*, uma com os octetos mais significativos e a outra com os menos significativos (Fig. 6.3).



i - Acesso à *fifo* de transmissão

ii - Acesso às *fifos* de recepção

Fig. 6.3 - Estrutura do acesso às *fifos*

Os sinais de controlo são responsáveis pela gestão dos processos de escrita e leitura dos *buffers* sempre que o Bloco Servidor necessita de lhes aceder. São também incluídos nos sinais de controlo os de interrupção, gerados pela placa *SMDS* e destinados ao Bloco Servidor, e os de *reset* enviados pelo Bloco Servidor para a placa de interface.

As interrupções que são geradas pela placa de interface são:

- *INT_L2PDU* - que anuncia a chegada de uma nova *L2_PDU* não vazia;
- *INT_PF* - que é gerado sempre que as *fifos* de recepção atingem determinada quantidade de informação ou sempre que termina uma trama *PLCP*.

A interrupção *INT_PF* é activada sempre que ocorra uma de duas condições possíveis, sendo a decisão relativa à operação a efectuar tomada em função do estado de outras linhas de controlo no instante dessa interrupção.

Os sinais de *reset* podem ser gerados no Bloco Servidor ou no barramento *VME* de interligação entre os Blocos Controlador e Servidor e sempre que são activados provocam o *reset* da placa *SMDS*.

Um outro sinal de controlo que pode ser destacado é o enviado pela interface *SMDS* ao Bloco Servidor que lhe dá ou não permissão para escrever nos *buffers* de transmissão. A escrita nestes *buffers* deve ser sempre precedida da verificação do estado desta linha.

Os sinais de endereçamento são utilizados para identificar cada uma das portas usadas pelo *TMS320C25*, no qual se baseia o Bloco Servidor. A placa de *SMDS* utiliza duas dessas portas, a 15 e a 14. À primeira correspondem os *buffers*, tanto de transmissão como de recepção, pelo que esta porta pode ser utilizada nos dois sentidos. À porta 14 corresponde o registo de estado da placa de interface, que só pode ser lido. Estes sinais são descodificados num dispositivo de lógica programável e, em função do seu valor, são gerados outros para aceder às *fifos* ou ao registo de estado.

A transferência de dados, entre o Bloco Servidor e a interface *SMDS,* é realizada em paralelo através de um barramento bidireccional de 16 *bits*. Quando o Bloco Servidor pretende enviar uma *L2_PDU* pela interface, envia-a pela porta 15, em palavras de 16 *bits*, sempre que para tal tenha permissão. Essas são armazenadas nos *buffers* e são posteriormente transferidas para a *fifo* de transmissão onde aguardam pelo mapeamento na trama *PLCP*. Na recepção, o processo é semelhante: os octetos da informação recebida nas tramas *PLCP* são distribuídos pelas duas *fifos,* onde ficam a aguardar a leitura do Bloco Servidor. Assim que esta ocorre, são lidos os dois primeiros octetos dos *buffers* de recepção em simultâneo (a palavra de 16 *bits*) e são lá colocados os dois seguintes.

As *L2_PDUs* enviadas pelo Bloco Servidor não estão completas, isto é, em vez dos 53 octetos que as constituem são apenas passados para a interface *SMDS* os 52 primeiros. O último octeto, correspondente aos 8 *bits* menos significativos do *CRC-10* (*10 bit Cyclic Redundancy Check*), não é transmitido porque estes 8 *bits*, bem como os 2 *bits* menos significativos do 52º octeto, são calculados e inseridos pela placa *SMDS* na *L2_PDU,* na fase de mapeamento desta na trama *PLCP*.

De um modo análogo, na recepção não é transferida para o Bloco Servidor a totalidade dos 53 octetos da *L2_PDU* recebida, mas apenas os que a permitem identificar e caracterizar, a informação nela contida, e um octeto de validação da mesma.

6.2.2. Bloco de Transmissão

Este bloco desempenha a maior parte das funções envolvidas na transmissão de uma *L2_PDU*, esteja ela ocupada ou vazia. É neste bloco que são geradas as tramas *PLCP* e de transmissão *E1*, e que as *L2_PDUs* são mapeadas nas tramas *PLCP*.

Uma das fases de construção da trama *PLCP* consiste no cálculo dos *bits* do *CRC-10* dos octetos que constituem cada uma das *L2_PDUs* que se encontram na *fifo* de transmissão. Este cálculo é feito para cada uma em simultâneo com o seu mapeamento na trama.

A construção desta trama evolui de acordo com uma máquina de 10 estados, em que cada um deles é caracterizado por um determinado valor nos octetos *P0* a *P9* (Fig. 5.7). De acordo com a sequência de transmissão de cada um dos octetos da trama, ao estado s_0 corresponde o valor de *P9*, ao estado s_1 corresponde o valor de *P8*, e assim, sucessivamente, até ao estado s_9 [ESIG93][G704_84]. A mudança de estado ocorre sempre que são transmitidos os 57 octetos associados a um determinado estado (pertencentes à mesma linha) e, assim que uma trama é transmitida na totalidade, a máquina retorna ao estado inicial para processar a trama seguinte. Um diagrama representativo da evolução da máquina de estados e da construção de uma trama *PLCP* é ilustrado na figura 6.4.

Os valores que os vários octetos *P(i)* podem tomar em cada um dos diferentes estados são referenciados na figura 6.4 por *P(i)*. Este octetos, em conjunto com *A1* e *A2*, servem para definir o alinhamento da trama *PLCP* na recepção. Os octetos *A1* e *A2* são apenas usados na recuperação e

monitoração do alinhamento das tramas *PLCP* e, como contêm valores fixos, não exigem grande processamento.

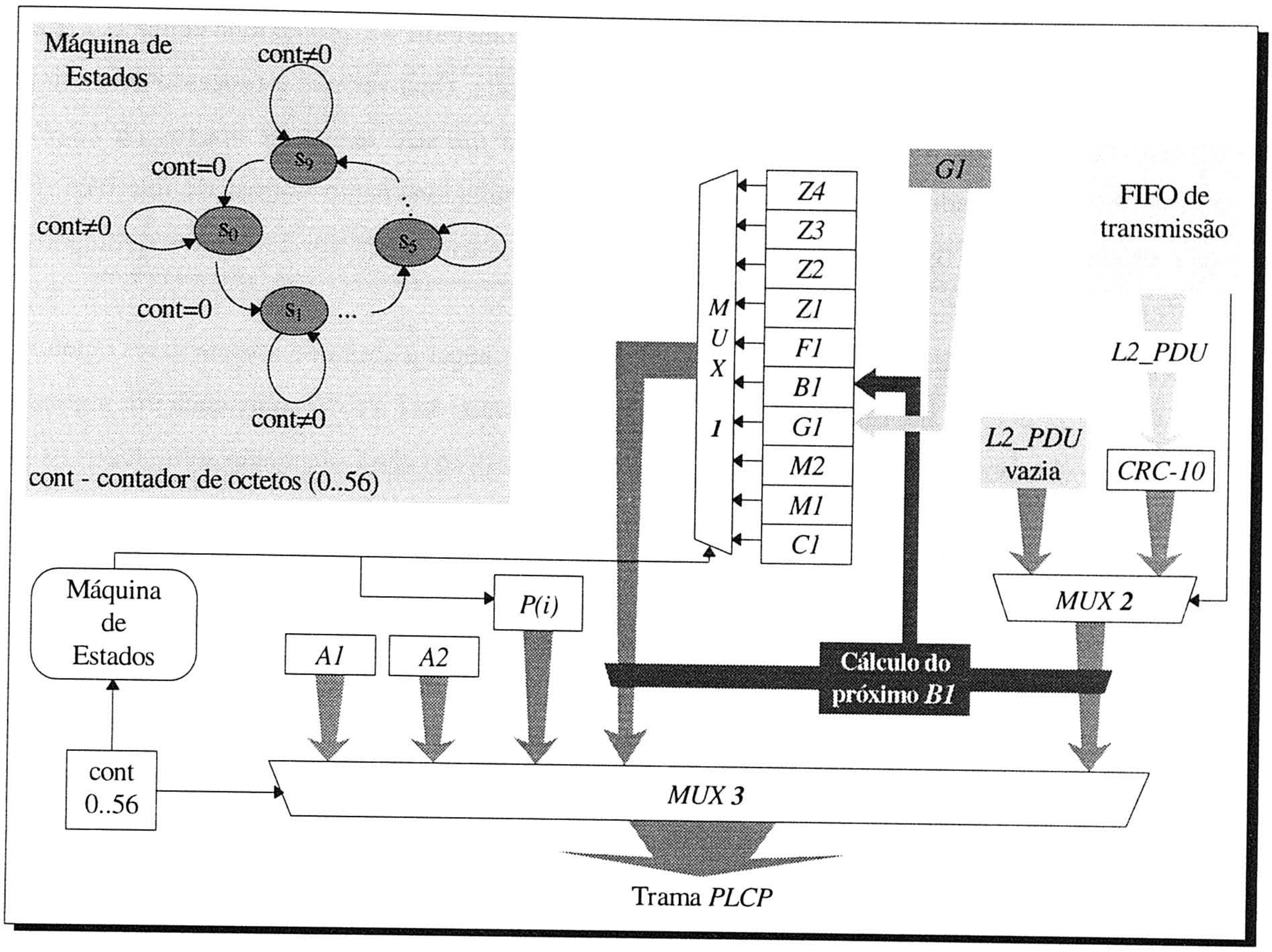


Fig. 6.4 - Construção da trama *PLCP*

Na figura 5.7 pode observar-se que os octetos da quarta coluna variam com a mudança de estado; a selecção de cada um desses valores é controlada pelo estado da máquina e é representada na figura 6.4 pelo *MUX 1*. Dos octetos seleccionados em *MUX 1* dois deles merecem especial atenção por exigirem cálculos: o *B1* e o *G1*. Os restantes octetos fixos da trama *PLCP* tomam valores constantes que estão definidos em [ESIG93] [G704_84].

O octeto *B1* (*Bit Interleaved Parity*) é utilizado para monitorar erros que possam ocorrer neste nível de transmissão: o seu valor é calculado a partir de todos os octetos das colunas 4 a 57 da trama *PLCP* e representa a paridade vertical de todos esses octetos. O octeto *B1*, calculado sobre uma dada trama, é guardado assim que esta chega ao fim e é enviado na trama seguinte.

O octeto *G1* é construído em função da qualidade da informação recebida e do estado das comunicações entre os subníveis *PLCP*. A construção deste octeto é um dos motivos pelos quais os

Blocos de Transmissão e de Recepção necessitam de comunicar. Uma abordagem mais descritiva da geração deste octeto será feita durante a análise do Bloco de Recepção.

A determinação de transmissão de uma célula vazia ou ocupada é feita em função do estado da *fifo* de transmissão. Se esta estiver vazia, não há informação para transmitir e é gerada uma célula (*L2_PDU*) vazia; caso contrário a *L2_PDU* que se encontra na *fifo* é enviada. Uma vez que o processo de escrita na *fifo* é mais rápido do que o de leitura, situações como a da *fifo* não ter os 52 octetos da *L2_PDU* disponíveis em determinado instante não ocorrem durante o funcionamento normal da interface. Esta selecção, representada na figura 6.15 pelo *MUX 2*, define quais os valores dos 53 octetos destinados às *L2_PDUs* na trama *PLCP*.

Se a *fifo* de transmissão dispuser de *L2_PDUs*, assim que chegar a altura de mapear esses octetos na trama *PLCP*, os 52 octetos são tirados da fifo e é calculado o valor do *CRC-10* sobre cada um segundo o polinómio $1+x+x^4+x^5+x^9+x^{10}$. De um modo mais preciso, este cálculo não é efectuado sobre todos os 416 *bits* dos 52 octetos, mas sobre os primeiros 414 *bits*, porque os últimos 2 *bits* correspondem ao espaço ocupado pelo campo *CRC-10* na trama. Assim que o cálculo termina, os dois *bits* mais significativos dos 10 calculados são guardados nos *bits* menos significativos do $52^{\underline{o}}$ octeto para que este possa ser normalmente inserido na trama. Os restantes 8 *bits* menos significativos constituem o $53^{\underline{o}}$ octeto da *L2_PDU* e são também adicionados à trama como qualquer outro octeto.

De entre todos os octetos passíveis de serem transmitidos em determinado estado, é de acordo com o valor do contador de 57 octetos (*cont 0..56* na figura) que cada um deles é seleccionado. Isto é ilustrado na figura 6.4 através do *MUX 3*. A selecção dos octetos por este contador garante a transmissão de todos os octetos da trama *PLCP* segundo a ordem estipulada na figura 5.7.

Depois de gerada a trama *PLCP*, os seus octetos são mapeados nas tramas de transmissão *E1* (Fig. 5.9). A construção desta trama envolve ainda uma série de passos intermédios para a geração dos valores de alguns *bits* do *slot 0*.

Os valores necessários para o cálculo de cada um dos 16 *slots 0* de uma multitrama são os *Sis* e os *As*. Os primeiros contêm a informação ilustrada na figura 6.5 para cada uma das 16 tramas de uma multitrama.

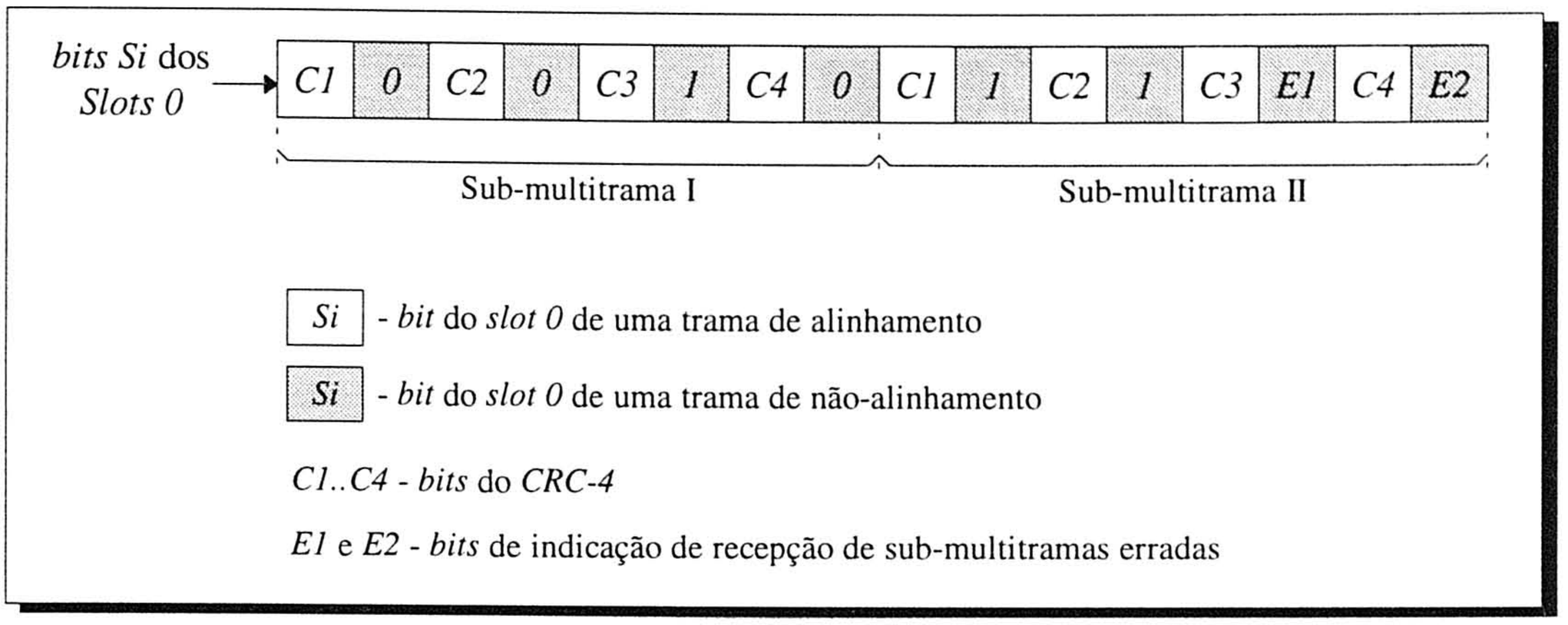


Fig. 6.5 - Informação dos *bits Si* dos *slots 0*

O *CRC-4* (*4 bit Cyclic Redundancy Check*) é calculado sobre todos os *bits* de uma dada sub-multitrama, a partir do polinómio $1+x+x^4$, e os 4 *bits* do resultado são enviados nos *bits C1* a *C4* da sub-multitrama seguinte. O cálculo e a inserção destes *bits* na trama de transmissão são feitos só após a conversão dos octetos de paralelo para série (Fig. 6.6). Foi tomada esta decisão para evitar duas conversões dos octetos, uma para série e outra novamente para paralelo, que seriam necessárias de modo a facilitar o cálculo. Assim, é possível efectuar o cálculo do *CRC-4* no último *bit* de uma sub-multitrama e colocar um dos *bits* do resultado, o *C1*, no primeiro *bit* da sub-multitrama seguinte.

Outros valores que os *bits Si* podem tomar, nas duas últimas tramas de não-alinhamento, são os *bits E1* e *E2*. Estes *bits* traduzem a qualidade das últimas sub-multitramas recebidas, sendo, por isso, sinais provenientes do Bloco de Recepção.

Por fim, os *bits As* são preenchidos de acordo com a qualidade do sinal recebido e são gerados e enviados para este bloco pelo Bloco de Recepção, bem como os *bits Es* e o octeto *G1*.

A selecção do tipo de *slot 0* a transmitir realiza-se em função de uma máquina de dois estados, um de alinhamento e outro de não-alinhamento, que alternam entre si sempre que são contados 57 octetos (Fig. 6.6). Os vários octetos são mapeados, nas respectivas posições, dentro da trama *E1*, coordenados por um contador de 32 octetos.

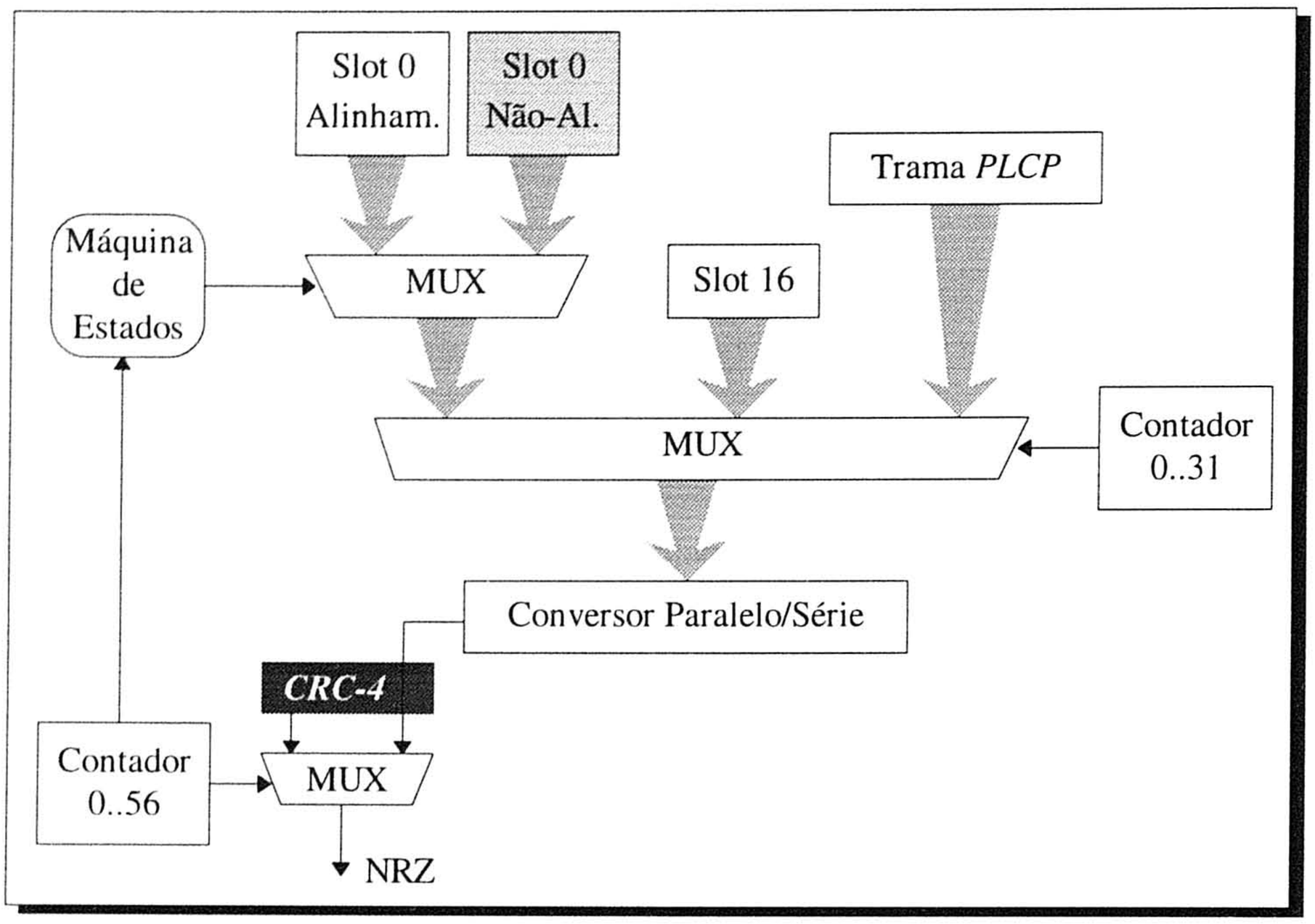


Fig. 6.6 - Contrução da trama de transmissão *E1*

Em simultâneo com a construção da trama transmissão *E1*, é gerado o código *NRZ* (*Non Return to Zero*) correspondente a cada um dos *bits* que é, de seguida, enviado para um conversor *NRZ/HDB3* [G703_72].

6.2.3. Bloco de Recepção

Este bloco recebe as *L2_PDUs* e coloca-as nas *fifos* de recepção. Todos as funções necessárias, desde a recepção do sinal *NRZ* (obtido a partir do sinal *HDB3* da linha), a detecção e a monitoração dos alinhamentos das diferentes tramas [G706_88], a detecção de erros de transmissão, até à selecção entre as *L2_PDUs* vazias e ocupadas, estão a cargo deste bloco.

Este bloco recebe dados, na forma *NRZ*, provenientes da interface *HDB3/NRZ*, que começa a agrupar os *bits* em octetos assim que é detectado o primeiro *slot 0* de alinhamento. A partir deste momento é desencadeado todo o processo de determinação de alinhamento de trama segundo a recomendação [G706_88]. Segundo esta norma, toda a informação recebida durante as fases de determinação dos alinhamentos de trama e de multitrama não deve ser considerada correcta. Como tal, optou-se na implementação desta interface, por só passar os octetos de informação das tramas de transmissão *E1* à fase de detecção do alinhamento da trama *PLCP* quando são declarados os alinhamento nas tramas e multitramas *E1*.

A monitoração do alinhamento das tramas de transmissão *E1* é realizada com base no armazenamento em registos do resultado do teste de validade de 3 *slots 0* consecutivos do mesmo tipo, de

alinhamento ou de não-alinhamento. Para se monitorar o alinhamento das multitramas, foi necessário implementar um relógio que declara a falta de alinhamento nestas se não forem detectadas duas correctamente alinhadas, num dado intervalo de tempo. Caso seja declarada a falta de alinhamento, por esse relógio ou por qualquer um dos outros mecanismos de monitoração, a placa de interface volta ao estado inicial de identificação do próximo *slot 0* e de detecção dos novos alinhamentos.

A par da monitoração do alinhamento de multitrama, e que pode também condicioná-la, é efectuado o cálculo do *CRC-4* de cada uma das sub-multitramas. O valor calculado é armazenado num registo de 4 *bits* e, posteriormente, é conferido com o recebido na sub-multitrama seguinte. Os *bits E1* e *E2* dos *slots 0* (nos *bits Sis*) das duas últimas tramas de não-alinhamento da multitrama a enviar pelo Bloco de Transmissão são definidos a partir dessa comparação (Fig. 6.5).

Garantidos os alinhamentos das tramas e multitramas de transmissão *E1*, inicia-se o processo de detecção de alinhamento das tramas *PLCP* [ESIG93]. Para tal, são conferidas as posições e as sequências dos octetos *A1*, *A2* e *P(i),* e os resultados são guardados em registos. A declaração de alinhamento resulta da confirmação das sequências *A1-A2-P(i)* e *A1-A2-P(i-1)* separadas pelos 57 octetos, enquanto que, para monitoração, apenas se verifica a ocorrência de um dos octetos *A1* ou *A2* e do *P(i).*

Assim que é detectado o alinhamento *PLCP* iniciam-se as restantes funções deste nível, de entre as quais se destacam a verificação do octeto de paridades *B1*, a geração do octeto *G1* e a geração da interrupção *INT_PF*. A verificação do octeto *B1* é feita com base num registo de cálculo (de 8 *bits*), onde são acumuladas as paridades verticais dos octetos das colunas 4 a 57 da trama *PLCP*, obtidas com a aplicação da dijunção exclusiva sobre todos esses octetos. O valor obtido é armazenado num outro registo para ser depois comparado com o octeto *B1* da trama *PLCP* seguinte. Essa comparação é feita de modo a poder detectar-se a ocorrência de erros em cada uma das colunas associadas a um *bit*. O número total de colunas com *bits* errados é um dos valores dos campos do octeto *G1*.

Este mecanismo de detecção de erros falha no caso de ocorrer um número par de erros em cada uma das colunas, mas, caso tal aconteça, esses erros serão detectados e localizados com a validação das *L2_PDUs*, sendo devidamente identificadas as que tiverem erros.

O processo de relatar ao sistema de comutação (*SS*) o estado em que a informação, vinda nas tramas *PLCP*, está a ser recebida na interface *SMDS*, é baseado no octeto *G1*. Da informação contida nele fazem parte o valor calculado a partir do *B1*, um sinal de alarme e uma série de mensagens protocolares que definem o estado da ligação [ESIG93]. Por fim, há apenas que referir que foi implementado um relógio, independente de todos os outros relógios do sistema, dedicado exclusivamente à medição de intervalos de tempo da ordem dos segundos necessários para a activação do sinal de alarme relativo à perda de alinhamento nas tramas *PLCP* recebidas.

Um esquema representativo da construção dos octetos *B1* e *G1* é ilustrado na figura 6.7.

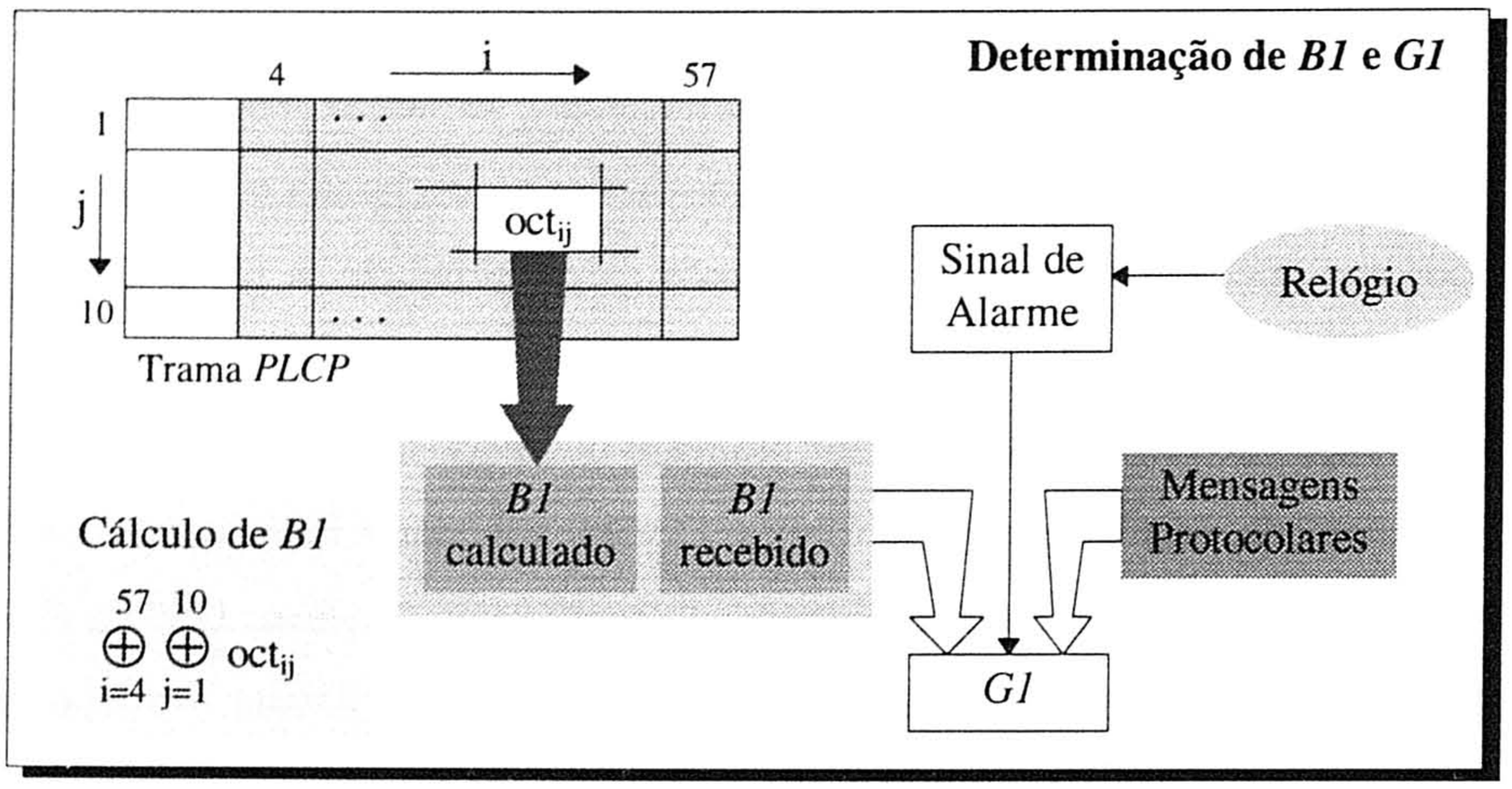


Fig. 6.7 - Determinação de *B1* e *G1*

As *L2_PDUs* são, em seguida, extraídas da tramas *PLCP* e, após identificação das células ocupadas, dá-se início à sua validação. A validação não é mais do que uma análise dos valores dos campos *ACF* e *NCI* e a detecção ou não de erros, com base no *CRC-10*. Depois dessa validação, é elaborado um octeto relatório dos erros detectados, que será acrescentado aos octetos a escrever nas *fifos* de recepção.

O octeto relatório é utilizado como mecanismo de validação da *L2_PDU* recebida; daí a sua denominação de octeto *VALID*. Os *bits* do campo *NCI* contêm valores constantes que são automaticamente testados e, se forem detectados erros em qualquer um deles, é activado (*bit*=1) um *bit* no octeto *VALID*. Um outro *bit* deste octeto relata o resultado do cálculo do *CRC-10* dessa *L2_PDU;* assim, se o valor recebido não coincidir com o calculado, esse *bit* será activado.

Para contemplar a situação de se estar a meio da recepção de uma *L2_PDU* e ser detectada uma perda de alinhamento nos níveis de transmissão, utiliza-se um mecanismo que consiste em escrever na *fifo* octetos a zero até perfazer os 47 esperados pelo Bloco Servidor. Sempre que uma situação deste tipo ocorre, é activado um terceiro *bit* no octeto *VALID* para avisar o Bloco Servidor de que essa *L2_PDU* não está completa.

Como já foi referido, os 53 octetos da *L2_PDU* não são todos enviados para o Bloco Servidor: apenas 47 são transferidos para as *fifos* de recepção. Desses 47 octetos fazem parte 2 de identificação dessa *L2_PDU* (com os campos *SegTyp*, *SeqNum* e *MID*), os 44 de possível informação (*SegUnit*) e 1 octeto em que os 6 *bits* mais significativos correpondem ao tamanho do campo de informação (*PayLen*) e os 2 *bits* menos significativos não têm significado. O octeto *VALID* é acrescentado a estes 47, ocupando aqueles que seriam os 8 *bits* menos significativos do campo do *CRC-10* da *L2_PDU*.

Assim que é recebido o último octeto de uma *L2_PDU*, o octeto *VALID* é escrito na *fifo* dos octetos menos significativos a par com o 47º octeto de informação e será lido pelo Bloco Servidor no mesmo ciclo de relógio que este, não introduzindo qualquer atraso (Fig. 6.8). Neste instante, é também activada a

linha da interrupção *INT_L2PDU* que avisa o Bloco Servidor de que uma nova *L2_PDU* aguarda leitura na *fifo*.

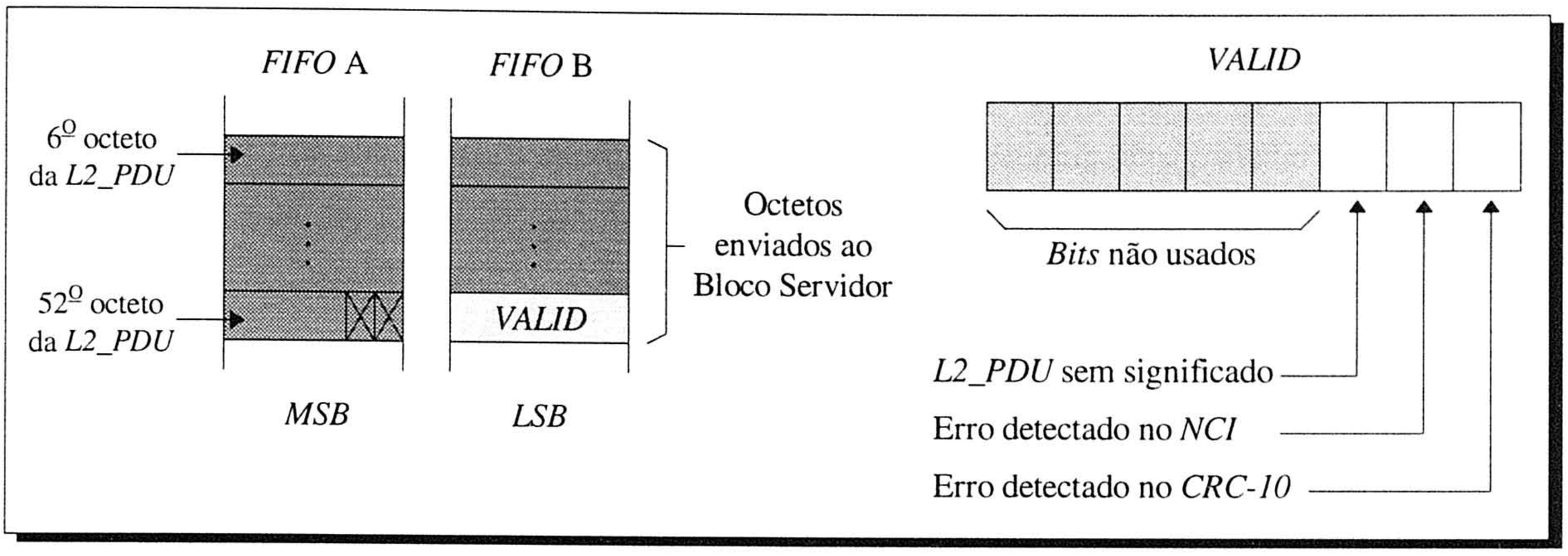


Fig. 6.8 - Octeto *VALID*

A implementação do mecanismo do octeto *VALID* teve por objectivo reduzir o tempo de processamento necessário para validar as *L2_PDUs*, sem que houvesse perda de informações que poderiam ser relevantes para determinadas medidas. Um exemplo concreto de ganho em tempo está relacionado com a quantidade de operações e ciclos de relógio que seriam necessários para calcular o *CRC-10*, por *software* e no Bloco Servidor, para cada *L2_PDU* recebida. Deste modo, se o Bloco Servidor pretender efectuar cálculos estatísticos com base nas faltas detectadas a este nível, apenas necessita de analisar o octeto *VALID* para identificar as falhas ocorridas.

O processo de recepção no Bloco Servidor deve assegurar a verificação do octeto *VALID* para todas as *L2_PDUs* recebidas, atendendo a que os campos das *L2_PDUs* associadas a octetos *VALID* que assinalem a existência de erros podem ter valores errados.

6.2.4. Bloco de Acesso à Linha

Este bloco é responsável pela codificação dos *bits* a transmitir, de *NRZ* para *HDB3*, e dos recebidos em *HDB3* para *NRZ* [G703_72] [G823_84]. Está também incluída neste bloco a tarefa de recuperação do relógio a partir do sinal recebido.

A recuperação do relógio de 8.192 MHz é realizada com base num mecanismo de controlo do desfasamento de um sinal obtido à custa da soma das alternâncias positivas e negativas do sinal *HDB3* da linha.

6.2.5. Registo de Estado

A placa *SMDS* dispõe de um registo destinado a relatar o seu estado em vários pontos da implementação, e que pode ser consultado pelo Bloco Servidor em qualquer altura. Deste registo de 16 *bits* fazem parte alguns dos sinais dos quais depende o funcionamento da placa. Na figura 6.9 é apresentada uma tabela que, resumidamente, descreve a indicação de cada um dos *bits*.

	Bit	Designação	Descrição
LSB →	0	*tx_E1*	Próximo bit *E1* a ser transmitido
	1	*tx_E2*	Próximo bit *E2* a ser transmitido
	2	*rx_E1*	Último bit *E1* recebido
	3	*rx_E2*	Último bit *E2* recebido
	4	*rx_A*	Último bit *A* recebido
	5	*tramalinhada*	Alinhamento das tramas de transmissão *E1*
	6	*almultitrama*	Alinhamento das multitramas de transmissão *E1*
	7	*LOS*	*Loss of Signal*
	8	*rx_AIS*	Situação de *Alarm Indication Signal*
	9	*LOIS*	*Loss Of Incoming Signal*
	10	*inframe_PLCP*	Alinhamento da trama *PLCP*
	11	*LOF_PLCP*	*Loss Of Frame PLCP*
	12	*Alarm10*	*Alarm Signal* em 10 tramas *PLCP* consecutivas
	13	*B1_error*	Detecção de erros no octeto *B1*
	14	*MSS_B1_error*	Detecção no *SS* de uma *PLCP* com erro em *B1*
MSB →	15	*EF_rx*	*FIFOs* de recepção vazias

Fig. 6.9 - Registo de Estado

Algumas das designações utilizadas correspondem directamente à implementação dessas funções de acordo com a norma [ESIG93].

6.2.6. Considerações sobre o *hardware*

Devido à complexidade do sistema, a interface *SMDS* foi implementada com uma placa de circuito impresso com quatro camadas, sendo uma para a alimentação, outra para a massa e as restantes para os sinais.

Embora a placa de interface *SMDS* não opere com frequências muito elevadas (no máximo 8.192 MHz), teve-se algum cuidado na minimização das linhas de relógio como forma de reduzir a radiação por elas provocada. Para diminuir as interferências electrostáticas nesta placa de interface, definiu-se uma rede de condensadores de acoplamento colocados em diversos pontos [Horowi89].

A interface com o Bloco Servidor engloba um descodificador de endereços e de outros sinais, que foi implementado numa matriz de lógica programável (*PAL - Programmable Array Logic*). As *fifos* de

transmissão e de recepção utilizadas são assíncronas por forma a tornar a placa de interface *SMDS*, tanto quanto possível, independente do Bloco Servidor. A ligação entre as duas placas, a do Bloco Servidor e a *SMDS*, é feita por um conector de 96 pinos.

A implementação dos Blocos de Transmissão e Recepção, descritos nas secções anteriores, foi realizada com base em dispositivos de lógica programável (*EPLD - Erasable Programmable Logic Device*) da *ALTERA* (família 7000) [Altera91]. Estes dispositivos são caracterizados por uma arquitectura baseada em macro-células, que lhes confere uma grande flexibilidade de programação e garante atrasos constantes, a diferentes frequências de funcionamento. A programação dos *EPLDs* é feita numa linguagem para descrição de hardware concebida pelo fabricante, a *AHDL* (*ALTERA Hardware Description Language*).

Foram necessários quatro *EPLDs*, dois *EPM7256ERC208-20* e dois *EPM7096LC68-15*, com 256 e 96 macro-células, respectivamente. As funções do Bloco de Transmissão utilizam um de cada, e, para se conseguir implementar a totalidade das funções do Bloco de Recepção, recorreu-se não só aos dois restantes, mas também a algumas macro-células utilizadas pelo outro bloco. Diagramas de blocos das funções executadas na *EPLDs* associadas aos Blocos de Transmissão e de Recepção são ilustrados no Apêndice A.

Associado aos processos de monitoração das tramas *PLCP* do Bloco de Recepção, foi desenvovido um relógio externo de 10 Hz, com base num *HEF4047B*, cuja função é contar tempos da ordem dos vários segundos. Esta opção foi tomada considerando que os diferentes relógios já existentes na placa, por operarem a frequências elevadas, iriam exigir contadores de muitos *bits*, que poderiam ser internos aos *EPLDs* ou implementados externamente. Quanto à implementação nos *EPLDs* estava fora de causa, uma vez que estes já não dispunham de macro-células suficientes para tal. Por outro lado, a implementação externa desse contador iria exigir a utilização de integrados mais caros e que necessitariam de mais área de circuito impresso.

O Bloco de Acesso à Linha, que envolve a transmissão e recepção dos *bits* codificados em *HDB3*, é, de certo modo, considerado a parte discreta da placa *SMDS*. Este bloco é responsável por converter de e para os níveis de tensão do *HDB3* os *bits* em *NRZ* gerados e recebidos pelos Blocos de Transmissão e Recepção, respectivamente. Essa conversão é feita em duas fases distintas: uma intergrada e outra discreta.

A primeira, incluída nas *EPLDs*, converte o *NRZ* de e para dois sinais, correspondendo cada um destes às alternâncias positiva e negativa do código *HDB3* [G703_72], que são directamente ligadas às partes discretas de transmissão e recepção.

A segunda fase, a discreta, tem por função fazer a adaptação de todo o equipamento à linha. Assim, e de acordo com cada uma das componentes obtidas, são gerados os níveis de tensão apropriados para a comunicação durante a transmissão. Na recepção, são separadas as duas componentes do sinal obtido a partir dos pulsos detectados na linha.

A interface com a linha de comunicação (de *75 Ohm*, não-balanceada) é concretizada por meio de conectores *BNC* para cabo coaxial.

A recuperação do relógio, incluída no Bloco de Acesso à Linha, foi implementada a partir de uma *PLL* (*Phase Locked Loop*), que foi configurada de modo a obter-se, em funcionamento normal, um relógio de 8.192 MHz com uma percisão de aproximadamente 30 p.p.m [G823_84].. Era importante conseguir uma boa precisão neste sinal pois é a partir dele que são gerados todos os restantes relógios da placa, isto se não se utilizar um oscilador externo no Bloco de Transmissão.

6.3. *Software* de Suporte

Face às diferenças no funcionamento das UPTs da nova ferramenta, houve necessidade de desenvolver *software* para cada um dos dois blocos nelas existentes.

O desenvolvimento deste *software* foi condicionado, desde o primeiro instante, pelos recursos de *hardware* existentes, mas de um modo geral, pode dizer-se que as funcionalidades básicas de acesso ao serviço *SMDS*, exigidas à UPT, foram implementadas com sucesso. A limitação de recursos fez-se sentir, principalmente, ao nível do Bloco Servidor das UPTs, que apenas dispõe de uma área de memória, para programa e dados, de 64 koctetos.

O *software* desenvolvido para o Bloco Servidor foi escrito em linguagem *assembly* do processador digital de sinal *TMS320C25*, no qual se baseia esse bloco. A versão executável do código foi obtida através de ferramentas da *Texas Instruments*, específicas para a programação deste processador: *assembler* e *linker*. O código final, já executável, deve ser transferido para as UPTs de modo a que o *software* de cada Bloco Controlador, nelas existente, lhe consiga aceder e copiá-lo para a memória de programa de cada Bloco Servidor. Assim que o *software* é carregado, o Bloco Servidor passa a um estado em que aguarda comandos vindos do Bloco Controlador, que condicionarão o seu funcionamento.

Quanto às funções desempenhadas pelo Bloco Servidor, consistem no conjunto de funcionalidades do *SIP nível-2*, exceptuando o cálculo do *CRC-10* que é implementado na placa *SMDS*.

O *software* para o Bloco Controlador foi desenvolvido em *C*, num ambiente *OS-9*, com o conjunto de facilidades nele disponíveis [OS9_87]. As funções que este desempenha são de coordenação dos vários Blocos Servidores (no máximo 4 em cada UPT), de implementação de algumas entidades utilizadas na descrição do teste em *TTCN* (por exemplo os *timers*) e de transferência de informação de e para a rede através do Bloco Servidor.

Para fazer a interligação entre os Blocos Servidor e Controlador foi desenvolvido um *driver* em *assembly* do processador 68030 [MOTO85].

O conjunto das funções implementadas no Bloco Controlador fazem parte de uma estrutura básica imperscindível para o funcionamento deste equipamento de teste. No entanto, foi também necessário

desenvolver um conjunto de funções que, para o método de teste local à UPT (Fig. 6.1), são encarregadas da interligação desta estrutura com o código *ANSI C* gerado no *ITEX* e correspondente ao teste descrito em *TTCN*.

6.3.1. *Software* do Bloco Servidor

Ao nível do Bloco Servidor é importante referir que o *software* desenvolvido teve como objectivo garantir o conjunto de funcionalidades associadas aos equipamentos dos clientes (*CPEs*), com acesso simples ao comutador *SMDS* (secção 5.2.1.). Na configuração utilizada na fase de desenvolvimento são usados dois Blocos Servidores e uma única UPT. O facto do acesso ser simples implica que os sistemas de comutação (*SSs*) dispõem de pelo menos dois acessos desse tipo onde cada um dos Blocos Servidores se irá ligar. Como o desenvolvimento do sistema de comutação estava fora do âmbito deste trabalho, optou-se por ligar, na fase de desenvolvimento, os Blocos Servidores um ao outro.

O Bloco Servidor, dada a posição que ocupa na UPT, dispõe de meios para comunicar tanto com a placa de interface *SMDS* como com o Bloco Controlador. Essa comunicação é normalmente sinalizada por interrupções, isto é, sempre que uma dessas entidades pretende avisar o Bloco Servidor da ocorrência de determinado evento, activa uma interrupção. Poderá acontecer, como no caso da placa *SMDS*, que uma dada interrupção possa ter mais do que um significado. Para essas situações, a arquitectura do Bloco Servidor dispõe de registos de controlo e de estado, que permitem analisar o estado de outros sinais quando surgem as interrupções, por forma a tornar possível a identificação da causa de cada uma delas.

Quanto à comunicação com o Bloco Controlador, esta é feita por intermédio de um registo dedicado, a *mailbox*: sempre que um dos blocos escreve nele provoca uma interrupção no outro. A transmissão e recepção de informação decorrem do atendimento das interrupções geradas pelos Blocos Controlador e Servidor, respectivamente.

O Bloco Servidor desempenha, como já foi referido, algumas das funções associadas ao *SIP nível-2* do *SMDS*, entre as quais as operações de segmentação e de reconstrução das unidades de dados do *SIP nível-3*.

Na transmissão o Bloco Servidor começa por controlar o acesso a uma área reservada, para onde o processo do Bloco Controlador transfere a *L3_PDU* que pretende transmitir. Depois divide-a em segmentos de 44 octetos para preencher os campos de informação (*SegUnits*) de cada uma das *L2_PDUs* que irá gerar para garantir a transmissão da totalidade da *L3_PDU*.

A geração de *L2_PDUs* passa então a uma fase em que cada uma delas é identificada pela atribuição de um *MID* e relacionada com as restantes através de um contador módulo 16 (ver secção 5.2.2.) associado às *L2_PDUs* com esse *MID*. Assim que a cada uma dessas unidades só falta o campo do *CRC-10*, ela é enviada para a *fifo* de transmissão da placa *SMDS*.

Para o processo de recepção é importante que seja definido um endereço do *SIP nível-3* de modo que haja uma selecção das *L3_PDUs* a recuperar. Com este processo tenta-se optimizar o uso da memória existente destinando-a apenas à reconstrução de unidades de dados que poderão ser aceites no *SIP nível-3*. Assim, sempre que chega uma *L2_PDU* do tipo *BOM,* o campo que corresponde ao endereço de destino (*DA* na Fig. 5.5) é comparado com o endereço atribuído a esse Bloco Servidor e só se coincidirem é que a *L2_PDU* é considerada. Com o mesmo objectivo, a informação do *SIP nível-2* recebida da placa *SMDS* só é processada se o octeto *VALID* garantir a sua qualidade (Fig. 6.10).

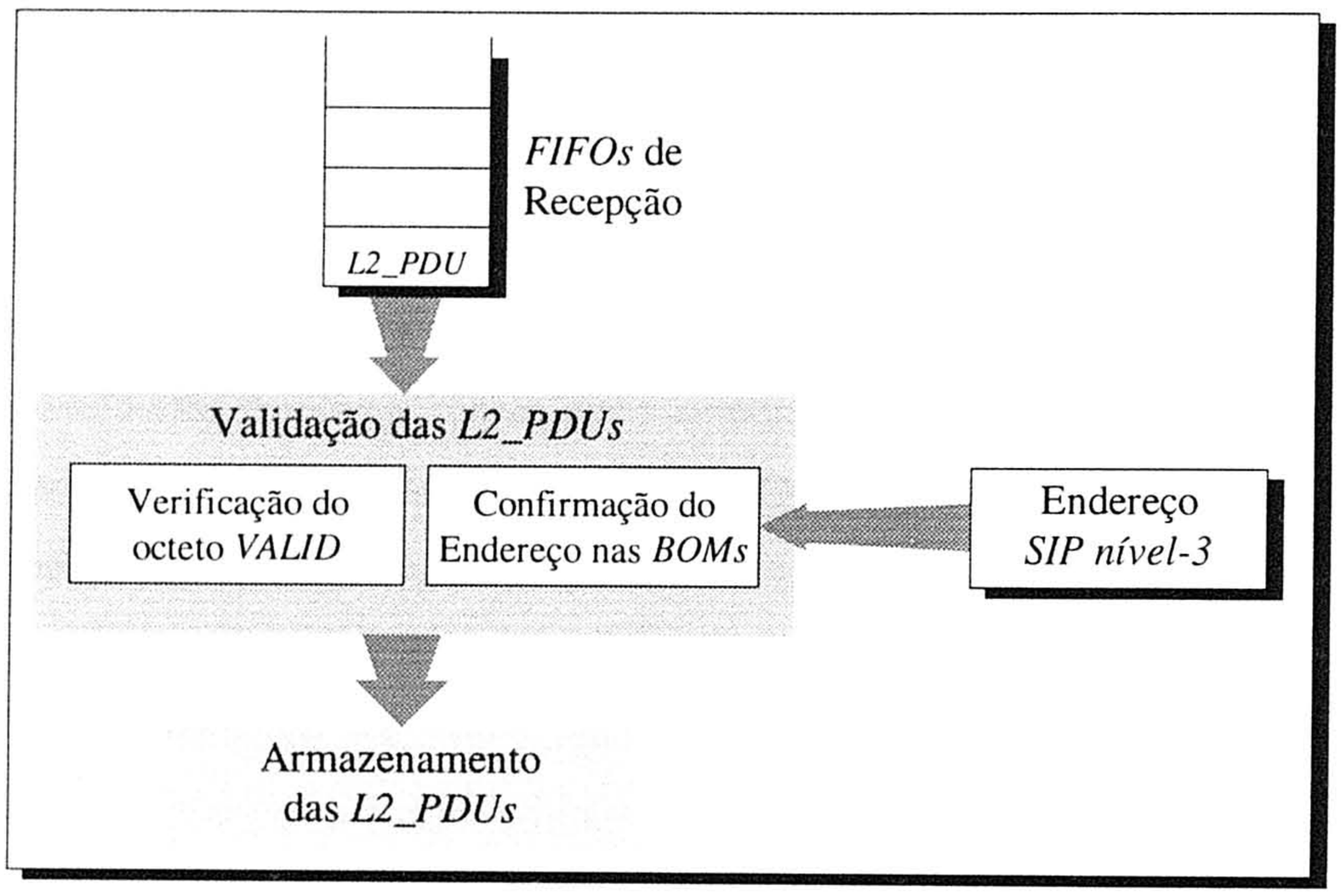


Fig. 6.10 - Recepção das *L2_PDUs*

Após a fase de validação é confirmada a possibilidade de armazenar a informação dessa *L2_PDU*, de acordo com os recursos existentes. A reconstrução de cada *L3_PDU* é parcialmente feita em áreas da memória que são associadas ao identificador dessa unidade de dados (*MID*). Esta parcialidade na reconstrução das *L3_PDUs* é provocada pelas limitações nas dimensões das áreas de armazenamento que, por isso, funcionam apenas como acumuladadores das várias partes de cada *L3_PDUs* que vão sendo recebidas nas *L2_PDUs*.

É a este nível que se notam as limitações existentes no que respeita à memória, pois não se dispõe de uma área que permita reconstruir, de uma só vez, 16 *L3_PDUs* diferentes, isto é, de *MIDs* diferentes, em simultâneo [Bellcore91]. Optou-se então por definir no Bloco Servidor 16 zonas de armazenamento de dimensões menores onde são acumuladas várias *L2_PDUs* que contêm partes de cada uma das 16 *L3_PDUs* distintas. Associado a essas zonas de armazenamento opera um mecanismo que, assim que estas ficam completas, avisa o Bloco Controlador de que as deve ler (Fig. 6.11).

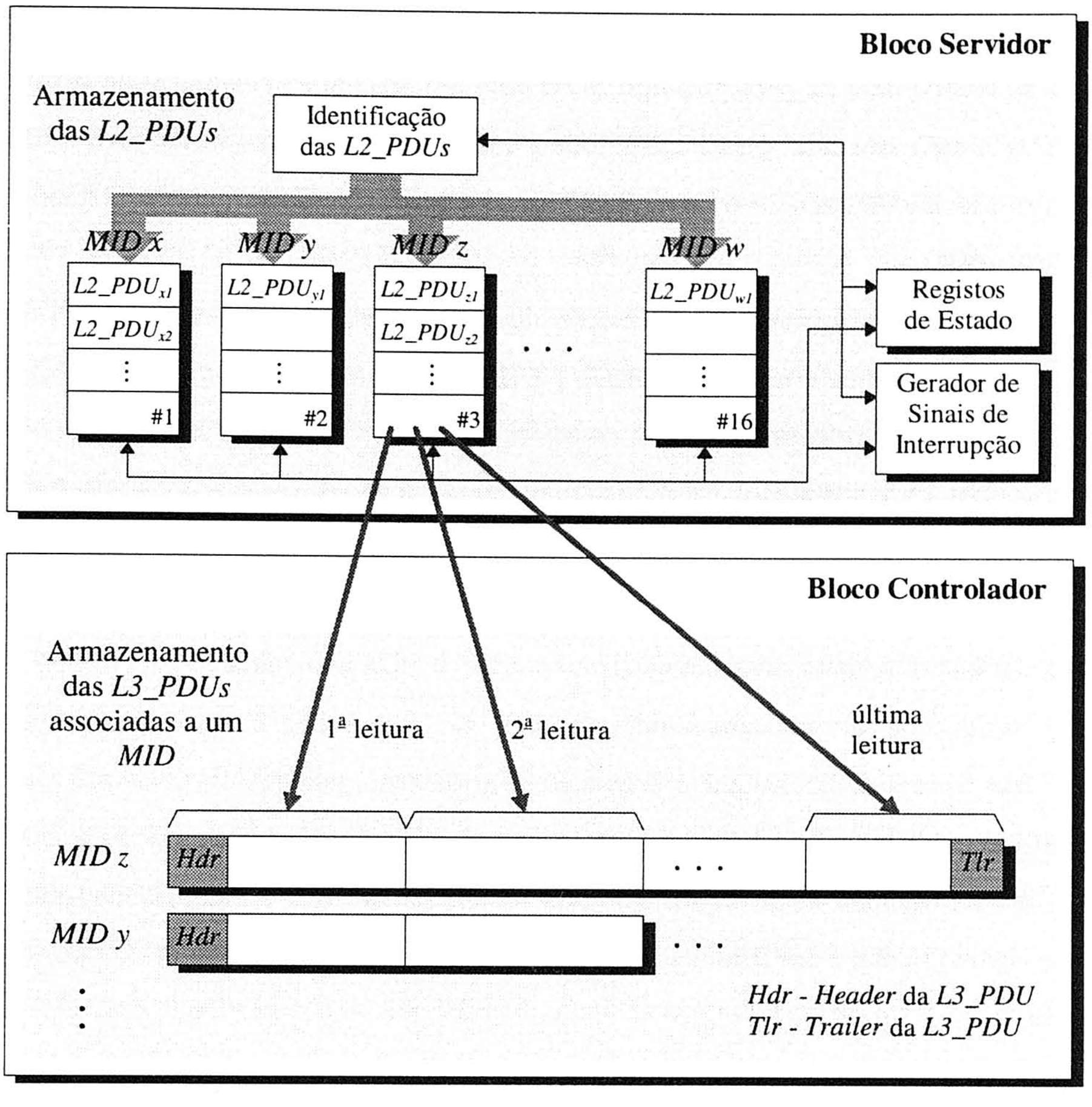


Fig. 6.11 - Reconstrução das *L3_PDUs*

Cada uma das áreas de armazenamento será lida tantas vezes quantas as necessárias para perfazer a recepção de uma *L3_PDU* na totalidade. Geralmente, é no Bloco Controlador que as várias *L3_PDUs* são integralmente reconstruídas com base na informação obtida nas zonas de memória do Bloco Servidor a elas associadas.

A identificação das *L2_PDUs* consiste em analisar o *MID* e determinar se já existe alguma área na qual unidades de dados com essa identificação estejam a ser colocadas e, no caso de ainda não existir, associar-lhe uma das áreas disponíveis. Se for localizada uma área onde já estejam a ser colocadas unidades de dados com esse *MID*, então é confirmada a ordem da informação dessa *L2_PDU* na *L3_PDU*, de acordo com o contador módulo 16 utilizado para ordenação das *L2_PDUs* dessa área. Só depois de todas as confirmações é que a informação é acrescentada à parte da *L3_PDU* anteriormente recebida.

Situações poderão ocorrer em que seja necessário avisar o Bloco Controlador de que a área associada a um *MID* teve que ser inicializada ou que deixou de ser utilizada antes de ser recebida a última *L2_PDU* dessa *L3_PDU*. Esses avisos são concretizados por meio de sinais de interrupção aos

quais é associada uma mensagem de duas poss. Num primeiro caso as mensagens de inicialização são geradas quando se recebe uma *L2_PDU* do tipo *BOM* com um *MID* que já estava a ser utilizado, antes de se receber o *EOM* a ele associado. Neste caso, toda a informação existente nessa área é desprezada, os registos de estado são inicializados e o novo *BOM* passa a ser a referência para as restantes mensagens recebidas com esse *MID*.

Num outro caso, as mensagens de inicialização são geradas quando determinada área deixa de estar reservada para um *MID* porque é recebido um *L2_PDU* fora da ordem prevista, isto é, o *SeqNum* dessa unidade de dados não coincide com o valor do contador módulo 16 associado a essa área. Nesta situação os registos de estado associados a essa área são actualizados e é gerada a interrupção para avisar o Bloco Controlador.

Em ambos os casos, o Bloco Controlador deve inicializar a zona para onde estava a concatenar a informação lida da área de armazenamento do Bloco Servidor a ela associada.

Para gerir cada uma dessas zonas de memória, foi necessário desenvolver mecanismos que permitissem ao Bloco Servidor controlar a ocupação de cada uma delas. Assim, sempre que estas ficam preenchidas são enviados sinais ao Bloco Controlador para que ele se encarregue de as ler e as liberte para permitir a leitura de nova informação. De igual modo, sempre que é identificada uma *L2_PDU* do tipo fim de mensagem (*EOM*) é enviado um sinal que notifica o Bloco Controlador de que deverá ler essa área de memória.

6.3.2. *Software* do Bloco Controlador

O conjunto de funções que este bloco tem de realizar é um pouco dependente do tipo de teste (Fig. 6.1) seleccionado. Tal dependência deve ser considerada caso a caso face aos distintos requisitos de *software* que cada um desses tipos de teste apresenta. Há, no entanto, um conjunto mínimo de funcionalidades que têm que ser garantidas independentemente do tipo de teste, por serem as responsáveis pela inicialização dos Blocos Servidores e por estabelecerem a comunicação com estes.

A inicialização dos Blocos Servidores é feita através da cópia, para a memória de programa de cada um deles, dos programas que vão definir os seus comportamentos. Assim que esses programas são carregados, é atribuído um endereço do *SIP nível-3* a cada um dos Blocos Servidores, sendo, de seguida, dadas instruções para que esses comecem a funcionar. As inicializações são implementadas por meio de uma função que é invocada na fase inicial do programa de execução do teste.

Para a transmissão e recepção das unidades de informação foram desenvolvidas primitivas que se encarregam de as mapear e extrair das *L3_PDUs* e de as transferir através dos Blocos Servidores seleccionados. São estas primitivas que facultam o acesso ao *SIP nível-3* do *SMDS*. Para um tipo de teste em que o código *ANSI C* gerado com o *ITEX* é transferido para a UPT (Fig. 6.1), onde é gerada a versão executável e posta em funcionamento, as primitivas desenvolvidas são acrescentadas ao código, bem

como as funções de adaptação necessárias [ITEX94b], por forma que a versão executável actue directamente sobre o acesso *SMDS*.

Para um teste executado a partir da Estação de Trabalho (*WS*), a UPT tem que dispor também de um conjunto de funções que consigam garantir a transferência da informação contida em cada *L3_PDU* para a *WS*. Este tipo de teste requer mais cuidados na implementação, pois a utilização de temporizadores (*timers*) nos testes exige que um protocolo para coordenação e análise destes seja também desenvolvido.

No trabalho descrito neste texto optou-se por efectuar o desenvolvimento do *software* necessário para executar os testes a partir da UPT. Assim, considera-se que ao código *C* transferido para a UPT são acrescentadas as funções de adaptação necessárias, bem como as primitivas de acesso ao *SMDS* e só então é gerado o ficheiro executável responsável pela evolução do teste previamente descrito em *TTCN* no *ITEX*.

Sempre que os Blocos Servidores pretendem que o Bloco Controlador vá ler determinada zona de memória geram, como já foi referido, uma interrupção. Essa é detectada por um *driver* que, de acordo com o Bloco Servidor que a originou e com a mensagem por ele transmitida, codifica uma palavra de 16 *bits* que identificará um sinal a enviar para o Bloco Controlador [Rochat??]. Com este sinal, o Bloco Controlador pode determinar o estado do Bloco Servidor que o gerou, porque existe um campo de informação que tanto pode conter o código de estado das zonas de armazenamento, como uma mensagem de erro nesse Bloco. A distinção é feita nos 4 *bits* de identificação do tipo de informação (Fig. 6.12).

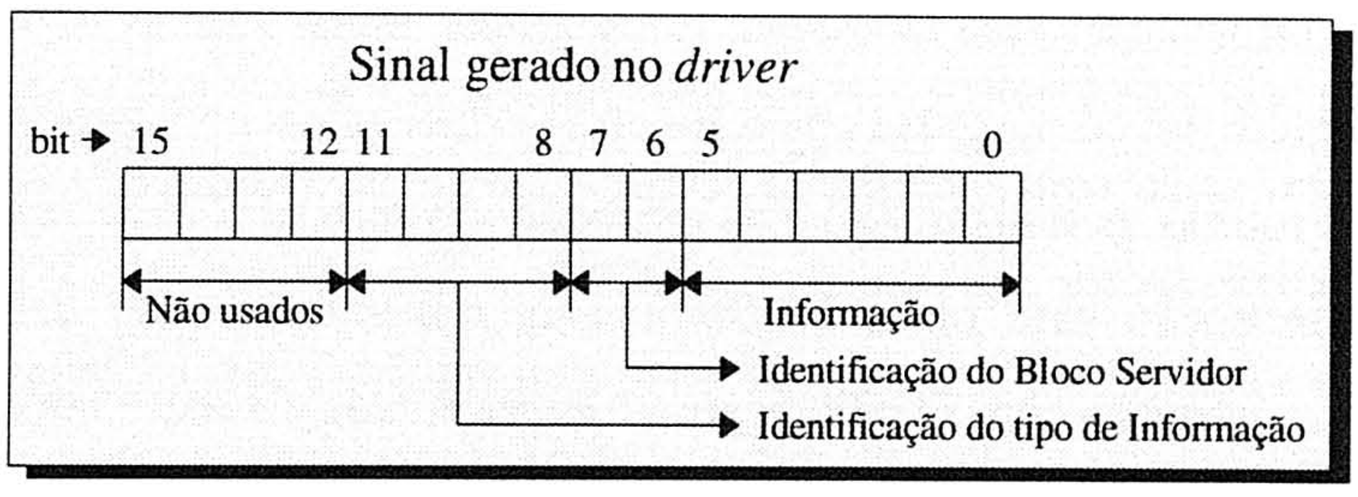


Fig. 6.12 - Sinal do *driver*

A transferência da informação da memória do Bloco Servidor é feita por leitura directa dessa zonas uma vez que o Bloco Controlador consegue endereçá-la. Esta operação é efectuada por meio de primitivas de baixo nível, disponibilizadas pelo sistema *OS-9* para efectuar cópias de áreas de memória a elevada velocidade.

A informação contida nas áreas de armazenamento do Bloco Servidor, correspondente a partes de *L3_PDUs*, é transferida para zonas de memória do Bloco Controlador reservadas para o efeito. Para garantir a integridade das *L3_PDUs* que vão resultar do reagrupamento das várias partes armazenadas em cada Bloco Servidor, são definidas 16 zonas possíveis (uma para cada um dos 16 *MIDs*) de reconstrução associadas a cada um dos Blocos Servidores. Essas áreas são caracterizadas por uma série de parâmetros

que estabelecem a correspondência com as áreas do Bloco Servidor de onde recebem a informação (Fig. 6.13).

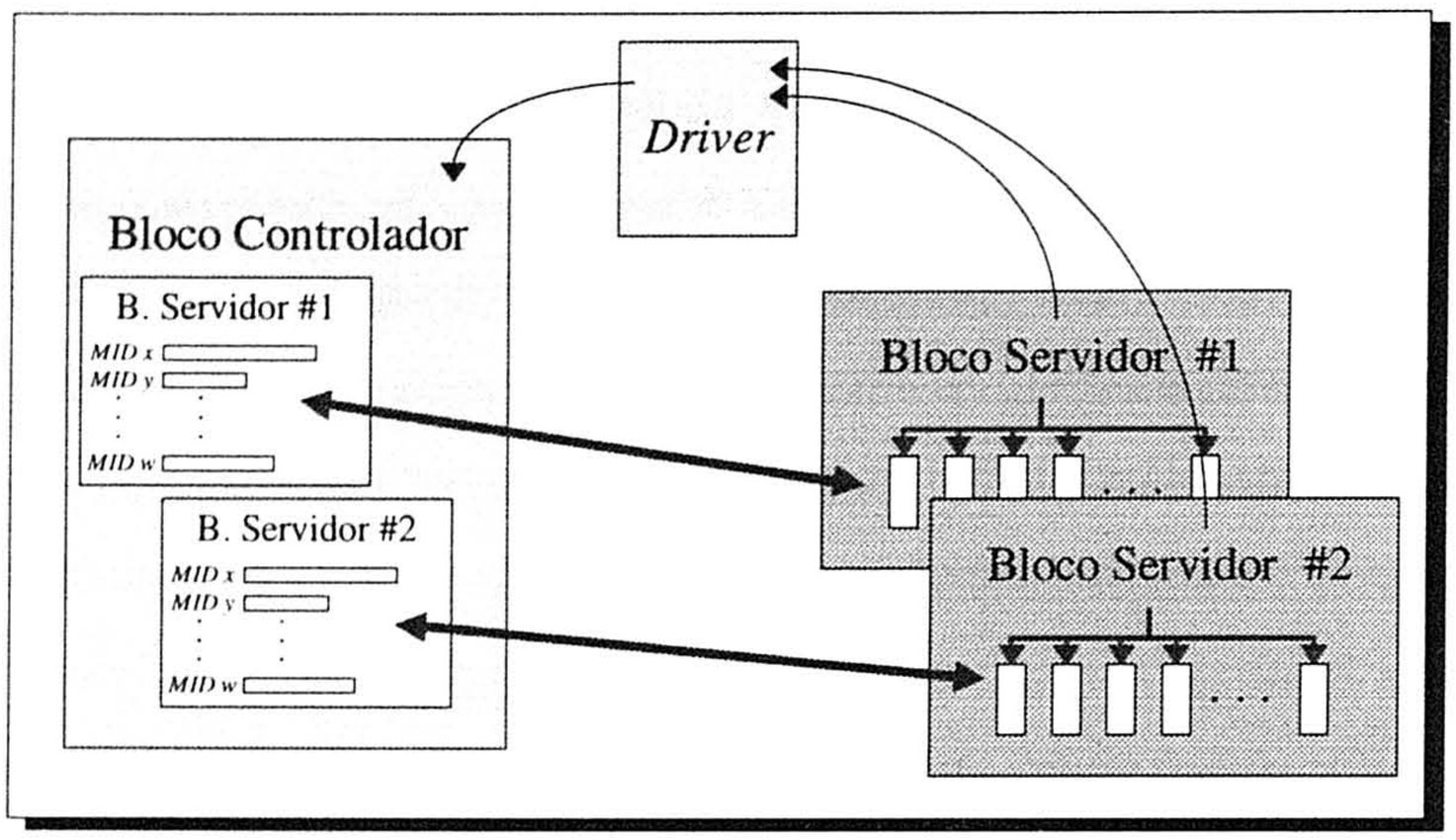


Fig. 6.13 - Sinalização para recepção

A transmissão de uma *L3_PDU* é, de um modo geral, uma operação mais simples: o Bloco Controlador averigua a disponibilidade da área de transmissão do Bloco Servidor que lhe interessa e se esta estiver a ser ocupada adormece durante um certo tempo e depois volta a tentar. Isto repete-se até que lhe seja dada permissão para aceder a essa área; assim que consegue a permissão reserva-a para si e efectua a cópia da sua informação para essa área. Em seguida, envia, através do *driver,* uma mensagem para a *mailbox*, que contém um código para que o Bloco Servidor inicie a transmissão.

É de referir que as rotinas de transmissão e de recepção são activadas, fisicamente, pela mesma linha de interrupção, a de recepção pela placa *SMDS* e a de transmissão pelo *driver,* ao escreverem a mensagem na *mailbox*, e portanto com idênticos graus de prioridade. Como é importante que a recepção seja prioritária relativamente à transmissão, para evitar congestionamentos nas *fifos* de recepção, houve a necessidade de se definir um mecanismo para estabelecer diferentes prioridades para as rotinas de atendimento de cada uma destas interrupções.

6.4. *Software* de Adaptação ao *ITEX*

A descrição de um teste em *TTCN* não permite que este seja executado directamente, isto é, é necessário converter o teste descrito nessa notação num ficheiro que, sendo directamente executável, possa realizar esse teste. Para se conseguir gerar esse ficheiro executável de teste é, imprescindível desenvolver um conjunto de funções de adaptação que têm por objectivo fazer a ligação entre o código *ANSI C* gerado com o *ITEX* e as UPTs.

Das funções de adaptação que devem completar as lacunas de informação existentes no código após a sua geração (ver secção 3.2), fazem parte as funções de codificação e descodificação das unidades de dados, as funções de gestão dos pontos de controlo e observação e as funções de acesso e controlo dos *timers* (Fig. 6.14) [ITEX94b]. Outras funções que são também incluídas no conjunto de funções de adaptação são as que se destinam a gerar ficheiros com os resultados do teste.

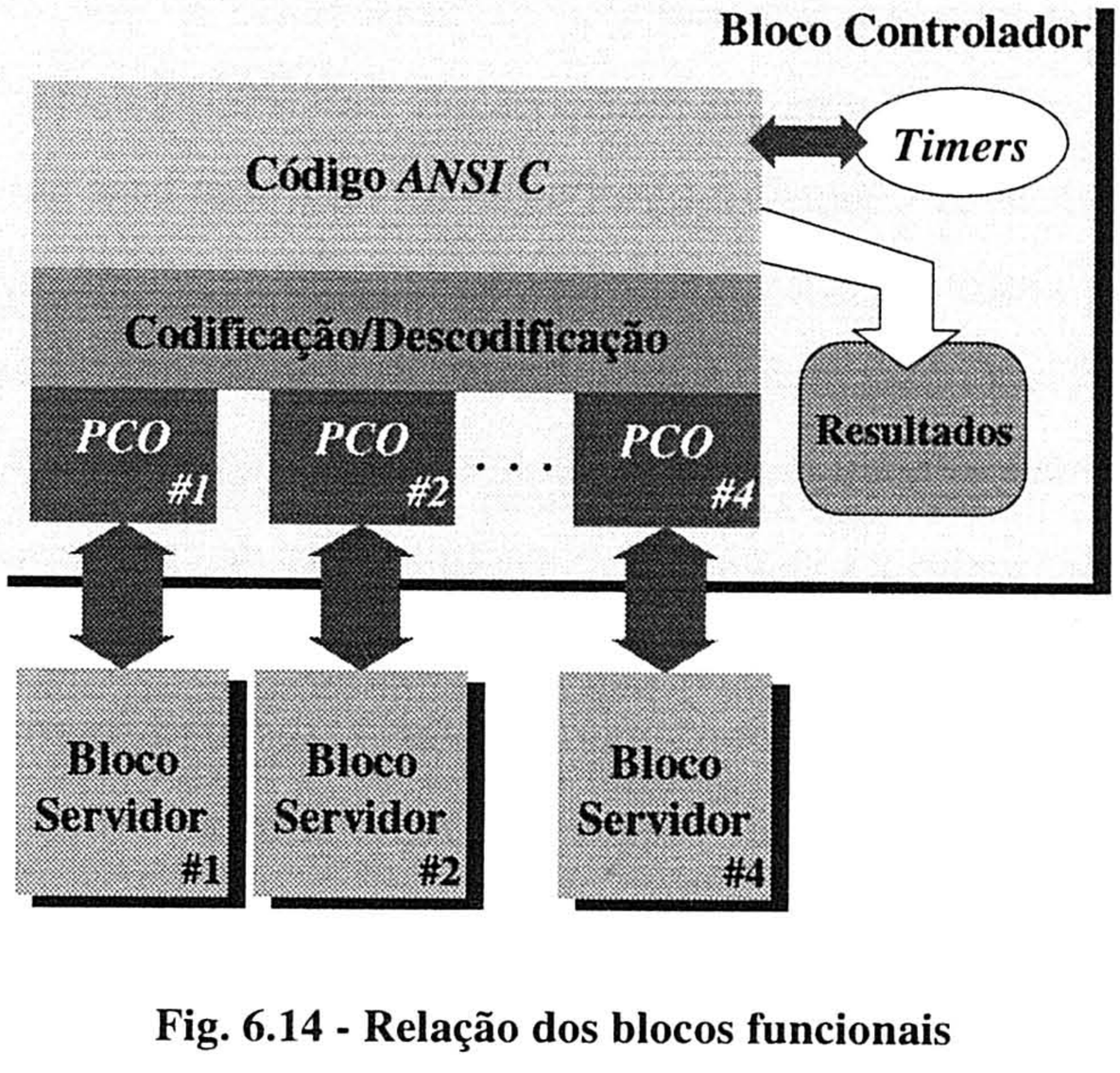


Fig. 6.14 - Relação dos blocos funcionais

De um modo geral, as funções desenvolvidas para o Bloco Controlador, que garantem a transmissão e a recepção de informação através do serviço *SMDS,* constituem uma parte integrante das funções de adaptação. Essas funções poderiam ser incluídas no grupo das encarregadas pelas operações de codificação e de descodifição das unidades de dados. No entanto, foram desenvolvidas outras funções para completar a codificação e a descodificação das unidades de dados.

As funções de codificação associadas à transmissão são directamente invocadas pelo código *ANSI C* gerado com o *ITEX.* Estas funções são responsáveis pela adaptação e pelo mapeamento dos formatos usados pelo *ITEX* em unidades de dados *SMDS,* que serão parte integrante das *L3_PDUs* transferidas. A variedade de formatos que é necessário codificar em cada conjunto de testes é função da quantidade de tipos diferentes utilizados para definir cada um dos campos das unidade de dados e das primitivas envolvidas nesse conjunto de testes.

O processo de codificação de cada uma das entidades que se pretende transmitir baseia-se na análise individual dos vários campos que constituem essas entidades. Os valores que são submetidos à codificação são definidos na secção das restrições (ver secção 3.1.3.) para cada tipo de entidade e os resultados dessa codificação não são mais do que a escrita numa zona de memória de transmissão associada ao *PCO* a partir do qual se pretende transmitir.

As funções de descodificação estão associadas à recepção das unidades de dados *SMDS* em cada um dos Blocos Servidores. Estas funções associam grupos de *bits* recebidos na unidade de dados aos vários tipos existentes na trama que era esperada.

Cada uma das unidades é submetida a uma avaliação tanto estrutural como de conteúdo, cujo resultado é função das restrições definidas no conjunto de testes e associadas a essa unidade de dados. Estas funções de adaptação começam por verificar se a sequência de *bits* recebida pode corresponder a uma trama com o formato que era esperado, e só nesse caso é que os campos são analisados individualmente de acordo com cada uma das restrições que lhes são impostas.

A identificação dos pontos de controlo e observação foi feita com base numa análise dos recursos do sistema aos quais fizesse sentido associar tais funções. Como resultado dessa análise optou-se por associar um ponto de controlo e observação a cada um dos Blocos Servidores existentes na UPT.

Esta solução é a que mais versatilidade acrescenta a esta ferramenta ao permitir a selecção de cada um dos métodos de teste abordados na secção 2.1.6. para os testes de conformidade. Assim, é possível ao operador definir e executar um teste em que sejam utilizados dois (no máximo quatro) pontos distintos de controlo e observação, um como analisador superior (*Upper Tester*) e outro como analisador inferior (*Lower Tester*) da implementação a testar, porque são fornecidos ao utilizador acessos a dois Blocos Servidores distintos. Para isso, é apenas necessário que, em conjunto, o Bloco Servidor e a interface que lhe está associada garantam o acesso a cada uma das interfaces disponibilizadas para testar essa implementação.

A implementação dos pontos de controlo e observação é feita com base em zonas de memória reservadas no Bloco Controlador, isto é, sempre que se pretende transmitir uma unidade de dados por um *PCO*, essa unidade é escrita numa zona de transmissão que é reservada para esse *PCO*. De igual modo, sempre que se pretende receber informação em determinado *PCO* essa é aguardada na zona de recepção a ele associada. Numa perspectiva mais prática, um *PCO* pode ser visto como duas *fifos*, uma de transmissão e outra de recepção, por onde são transferidas as unidade de dados *SMDS*.

A *fifo* de transmissão é um pouco fictícia uma vez que é implementada com capacidade para apenas uma unidade de dados. Esta dimensão é suficiente porque a evolução do programa de execução dos testes é sequencial, e como as operações de transmissão descritas em *TTCN* são sempre consideradas bem sucedidas, o programa só evolui após essa operação ter terminado.

A *fifo* de recepção de cada *PCO* é implementada dinamicamente a partir das 16 áreas para armazenamento de *L3_PDUs* associadas a cada Bloco Servidor.

As funções de adaptação necessárias para operar com os *PCOs* limitam-se apenas a invocar as primitivas de construção e transmissão de *L3_PDUs*, para enviar a unidade de dados existente na memória dos *PCOs*, e a relatar o estado das memórias de recepção dos *PCOs*, isto é, se receberam ou não *L3_PDUs*, sempre que tal lhes seja solicitado.

Outras funções de adaptação importantes durante a execução de um conjunto de testes são as que gerem os vários *timers*. Estas funções são também invocadas directamente pelo código gerado no *ITEX* e permitem pôr um *timer* em contagem, ler o tempo decorrido desde o início da contagem, verificar se o *timer* já atingiu o fim da contagem e parar um ou todos os *timers* activos nesse instante.

Os *timers* foram implementados com base em funções de geração e atendimento de sinais disponibilizadas pelo sistema *OS-9*. Embora apenas se consigam *timers* com precisões da ordem de alguns milisegundos, estas mostram-se suficientes se atendermos a que se pretende testar protocolos de níveis superiores ao *SIP Nível-3* do *SMDS*.

6.5. Possíveis Ambientes de Teste

As alterações a que foi submetido o SGDT permitem-lhe a executar uma variedade de conjuntos de testes utilizando os recursos de que dispõe. Na fase de adaptação desta ferramenta foram tidas em atenção as possíveis metodologias e configurações que podem ser utilizadas para executar testes de conformidade e de interoperabilidade, e, de acordo com os novos requisitos, foram desenvolvidas as novas funcionalidades do SGDT.

Nesta secção são descritas algumas das configurações de teste possíveis com o SGDT. Para estas configurações utilizam-se um Bloco Controlador e dois Blocos Servidores com interfaces *SMDS*.

Um tipo de testes que é possível executar com este equipamento são os testes de *IUTs* de sistemas de retransmissão. Um cenário de teste possível é apresentado na figura 6.15.

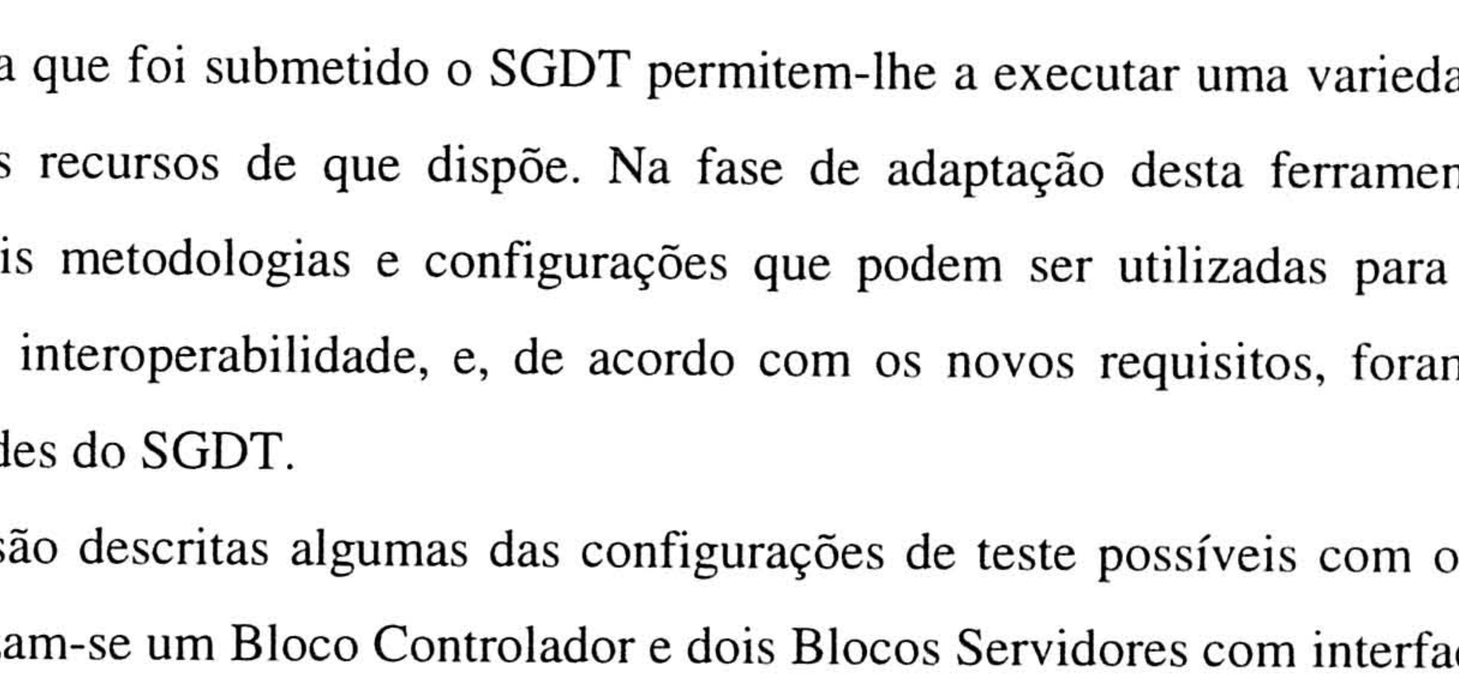


Fig. 6.15 - Método de Teste de Retorno com o SGDT

Esta figura ilustra a configuração usada para executar um teste de uma implementação utilizando o Método de Teste de Retorno. Esta configuração envolve o Bloco Controlador e os dois Blocos Servidores com as respectivas placas *SMDS* e é suposto que o sistema de retransmissão opere sobre o *SMDS* e se encarregue de converter a informação para os formatos do outro serviço.

Este método requer a utilização de dois *PCOs* pelos quais as unidades de dados usadas no teste são transferidas. Essas unidades podem ser enviadas por qualquer um dos *PCOs* mas são sempre recebidas no outro.

Para se executarem testes em implementações de equipamentos terminais podem ser utilizados dois métodos dos descritos na secção 2.1.6. de uma forma quase directa: o Método de Teste Coordenado e o Método de Teste Remoto. A utilização do primeiro desses métodos requer que o sistema a testar suporte um pacote de *software* que implemente um protocolo de gestão que coordene a interface de nível superior da *IUT*, e que exista um outro processo desse *software* para comunicar com o SGDT sem ser através da implementação a testar. Deste cenário de teste fazem parte um Bloco Controlador, um Bloco Servidores e a respectiva interface (Fig. 6.16).

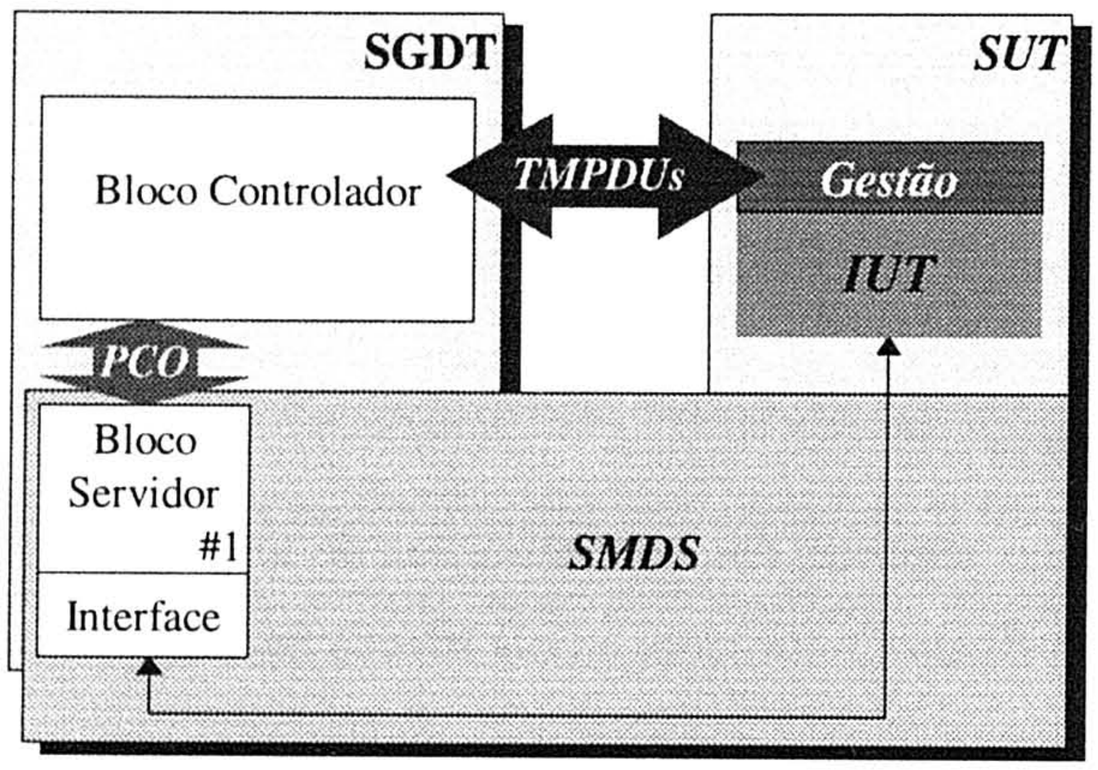


Fig. 6.16 - Método de Teste Coordenado com o SGDT

Neste método é somente utilizado um *PCO* que opera como analisador inferior da *IUT*. O *software* de gestão pode ser construído assim que se decida sobre a utilização de um meio de comunicação e que se conheçam as características da interface de nível superior da *IUT*. Do lado do Bloco Controlador é apenas necessário acrescentar algumas funções de adaptação ao teste que permitam enviar unidades de dados de gestão (*TMPDUs*) para o Sistema sob Teste.

Para o Método de Teste Remoto o cenário é muito parecido com o anterior, só que não é acrescentado qualquer *software* de gestão ao sistema que está sob teste (Fig. 6.17).

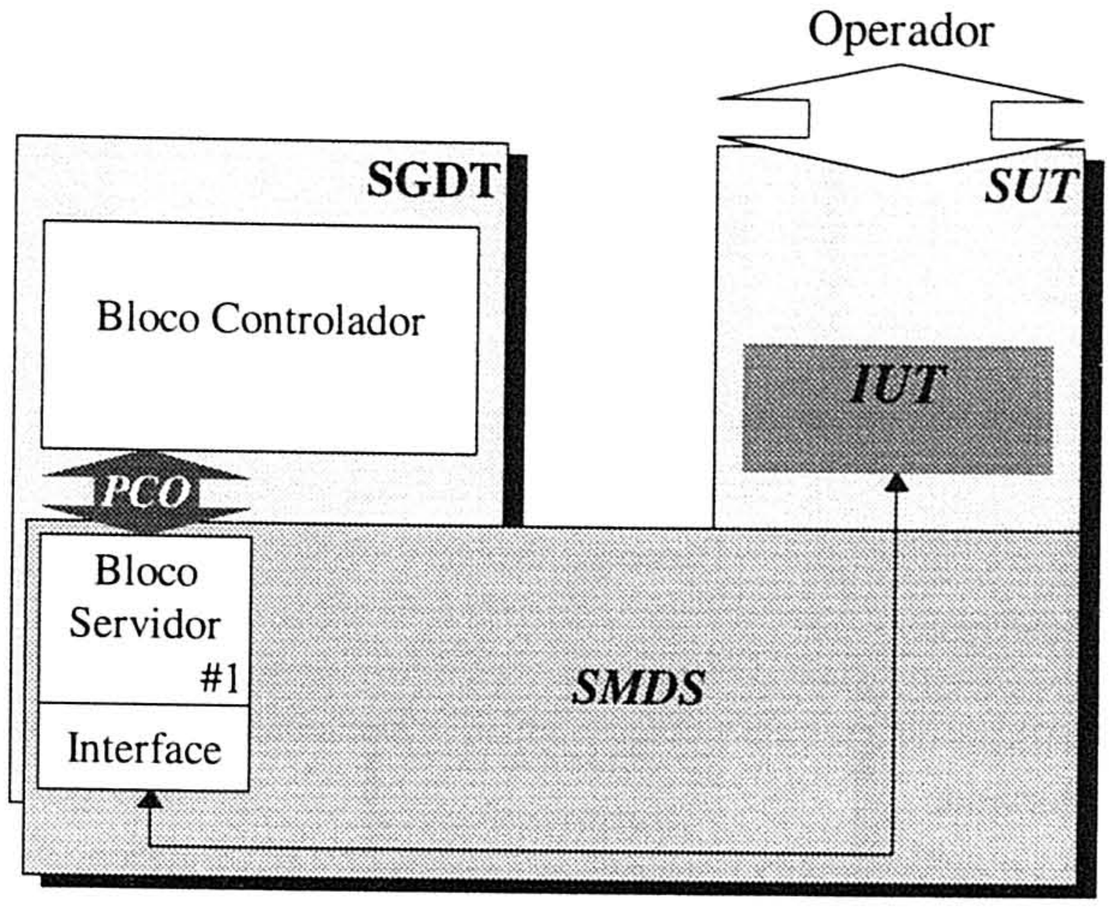


Fig. 6.17 - Método de Teste Remoto com o SGDT

Este método de teste é o que dá menos informação acerca da *IUT* ao operador e só deve ser usado quando é impossível aceder à *IUT* pela interface de nível superior a não ser através dos níveis localizados acima dela. A utilização de um outro operador que trabalhe com as aplicações do *SUT* segundo um conjunto de procedimentos de coordenação do teste, é importante por forma a colocar o Sistema sob Teste num estado conhecido.

A execução de testes utilizando o Método de Teste Local com o SGDT exige uma outra placa de interface que permita actuar directamente sobre o nível superior da *IUT* (Fig. 6.18).

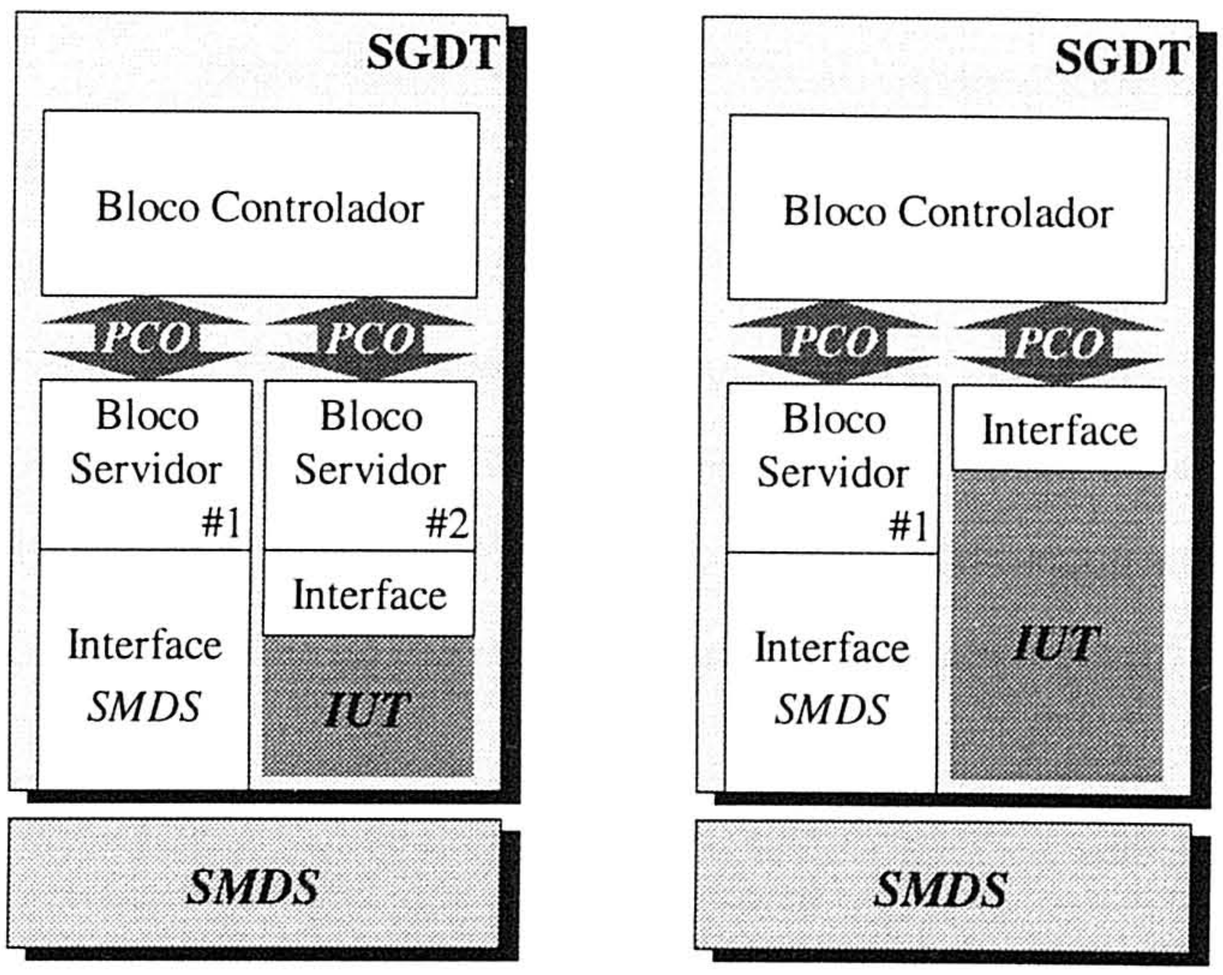


Fig. 6.18 - Método de Teste Local com o SGDT

Essa interface poderá estar ligada a um Bloco Servidor ou estar directamente ligada ao Bloco Controlador pelo barramento *VME*. No último caso seria necessário repensar as implementações dos *PCOs* associados às novas interfaces.

Um método semelhante ao Método de Teste Transversal pode ser utilizado se pretendermos testar a implementação de um sistema de retransmissão utilizado para ligar duas sub-redes com serviços diferentes. (Fig. 6.19).

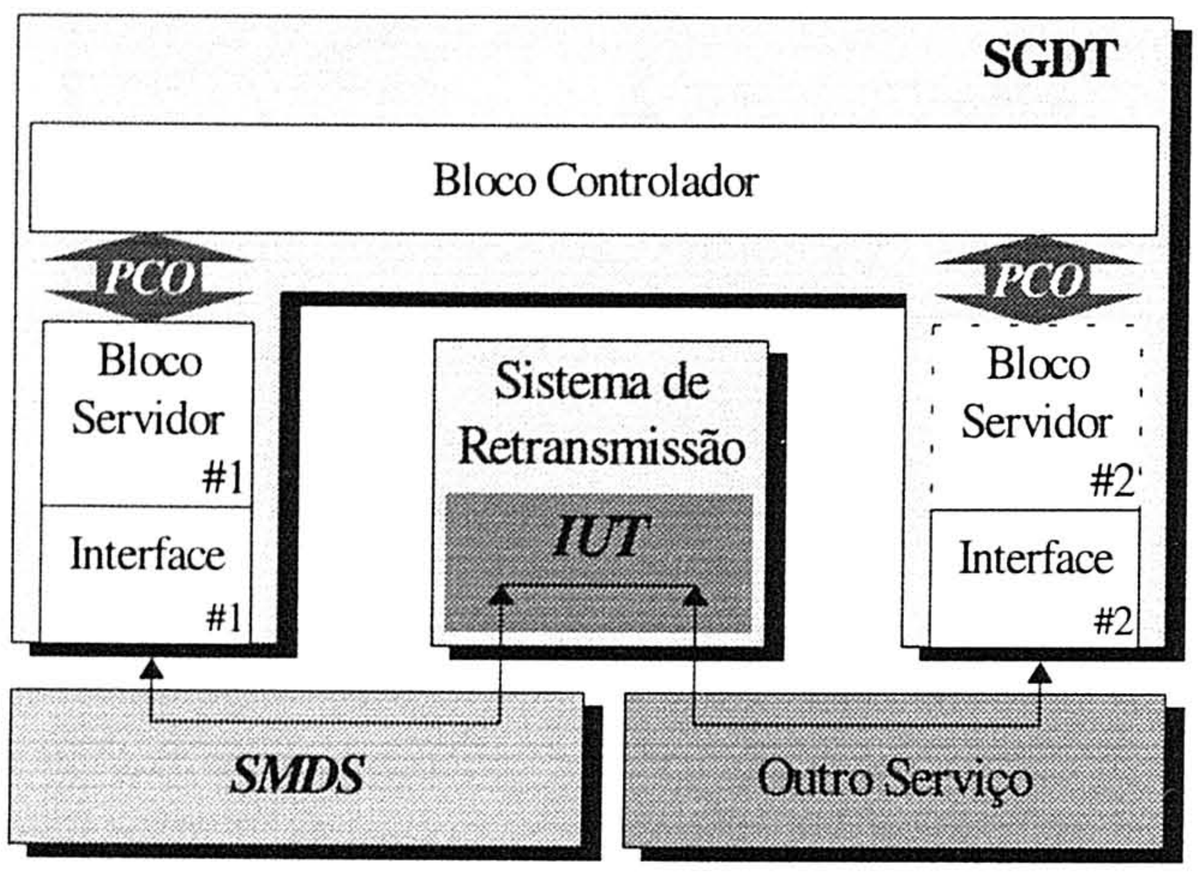


Fig. 6.19 - Método de Teste Transversal com o SGDT

Este cenário requer o desenvolvimento de interfaces para aceder a cada um dos serviços, que estariam ligadas ao Bloco Controlador, através de Blocos Servidores ou não. Na figura 6.19 é ilustrado um esquema representativo desta configuração de teste em que um dos serviços utilizados é o *SMDS*. Nesta configuração estão envolvidos dois *PCOs* que transferem unidades de dados alternadamente de modo a permitir testar o comportamento da *IUT* em ambas as sub-redes.

Por fim, uma outra funcionalidade em que o SGDT pode ser utilizado é na monitorização do tráfego trocado entre dois sistemas que se pretende que interoperem. A monitorizaação do tráfego requer que parte do *software* desenvolvido seja optimizado por forma a diminuir os atrasos associados à recepção e retransmissão da informação para o outro sistema. Nesta configuração, em que o serviço fornecido poderia ser o *SMDS*, seria necessário utilizar o Bloco Controlador e os dois Blocos Servidores com as respectivas interfaces por forma a garantir a monitorização em ambos sentidos (Fig. 6.20)

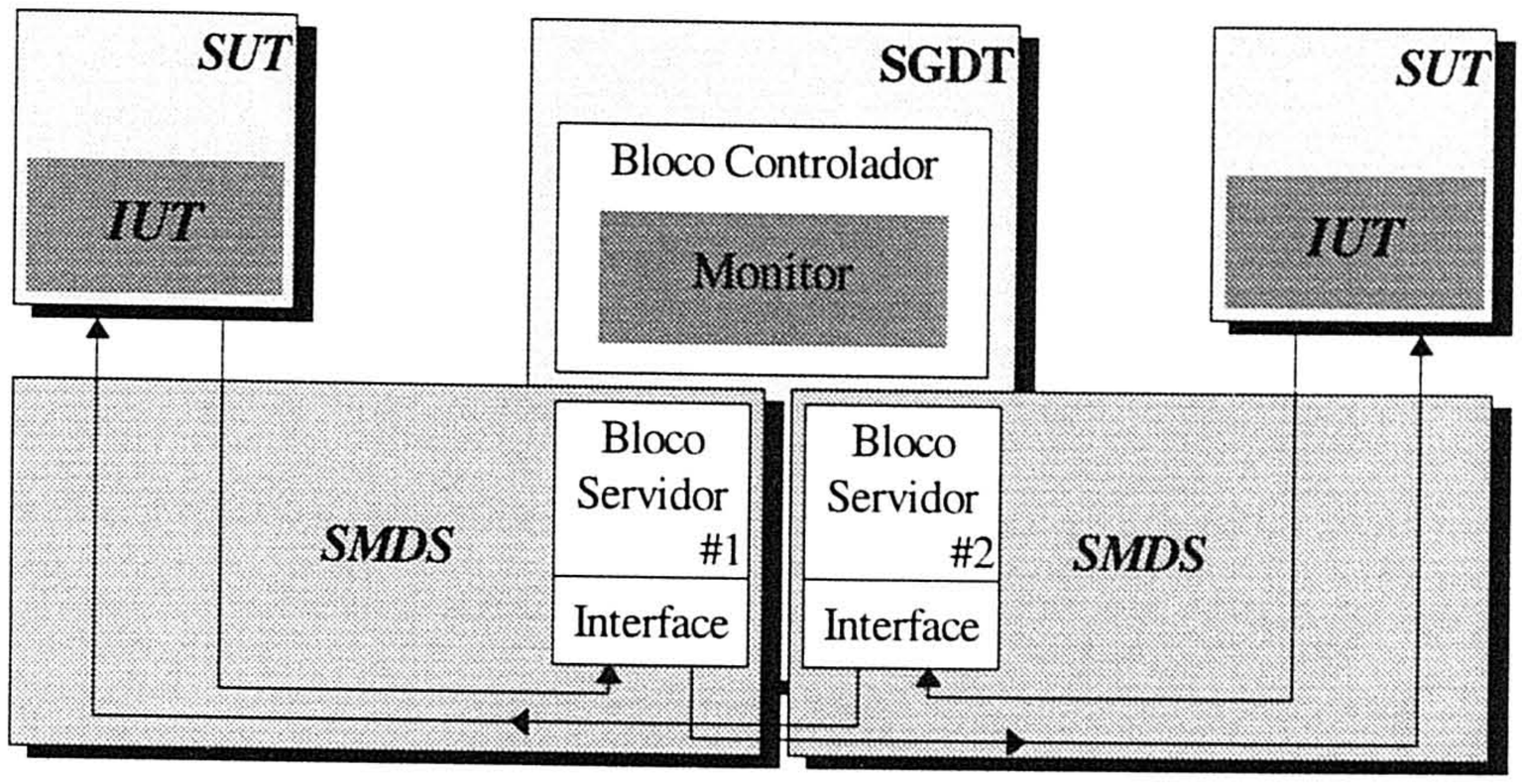


Fig. 6.20 - Monitorização de Tráfego com o SGDT

Uma monitorização activa poderia ser empreendida com base na definição de um conjunto de procedimentos de análise do tráfego transferido. Esses procedimentos poderiam ser descritos em *TTCN* não na prespectiva de obter um veredicto final mas no sentido de se gerarem um conjunto de mensagens que permitisse tirar conclusões à cerca da interoperabilidade dos Sistemas sob Teste.

Capítulo 7

Conclusão

O aparecimento de aplicações que requerem a transferência de grande quantidade de dados a elevadas velocidades, como por exemplo o vídeo, a multimédia e outras aplicações interactivas, ou não, levaram os operadoras e os fabricantes à definição de arquitecturas de banda larga para as redes de comunicação. Foi então definida a rede digital com integração de serviços de banda larga (RDIS-BL) e a escolha de uma tecnologia de suporte para a RDIS-BL foi condicionada por um conjunto de factores tão distintos como a flexibilidade, os custos dos equipamentos, do funcionamento e da manutenção, e a adaptabilidade a possíveis evoluções. A tecnologia que se adoptou foi o *ATM* por conseguir conciliar de forma razoável os vários requisitos.

A adopção do *ATM* como suporte da RDIS-BL despertou o interesse dos fabricantes de serviços e equipamentos por esta tecnologia. Serviços para abranger áreas metropolitanas têm vindo a ser desenvolvidos a pensar numa futura adaptação para as redes *ATM,* que certamente os irão levar mais longe. O primeiro desses serviços a ser normalizado foi o *Switched Multi-megabit Data Service* (*SMDS*) que se apresenta com sendo um serviço de comutação de pacotes para interligação de *LANs* geograficamente distantes. O *SMDS* garante a transferência de informação de forma transparente e, como tal, é possível adaptar protocolos de níveis superiores por forma a operarem sobre este serviço. O *SMDS* começa já a ser comercializado e por isso prevê-se que não demore muito a oferta por parte dos fabricantes de aplicações que operem sobre este serviço. O desenvolvimento dessas aplicações passa pela implementação de um conjunto de protocolos por forma a serem disponibilizadas aos utilizadores.

A implementação de protocolos requer certos cuidados, quando se pretende que os equipamentos desenvolvidos operem correctamente quando integrados em sistemas produzidos por outros fabricantes. A garantia do sucesso de cada integração está quase sempre associada à conformidade das várias implementações com determinada especificação ou norma, e à interoperabilidade entre os vários sistemas e protocolos em questão. A utilização de ferramentas para a avaliação da conformidade e da

interoperabilidade das implementações torna-se então praticamente obrigatória, e foi a opção tomada pelos grandes fabricantes de produtos e equipamentos.

Nesta dissertação descreveu-se a construção de um equipamento de teste, efectuada a partir da adição de novas funcionalidades a uma ferramenta de teste, o Sistema de Geração e Detecção de Tráfego (SGDT). O novo equipamento tem a possibilidade de executar testes de conformidade e de interoperabilidade em implementações de protocolos que operem sobre o serviço *SMDS*. O SGDT era uma ferramenta para executar exclusivamente testes de desempenho e de avaliação da qualidade de serviço, com uma interface para definição de testes própria e não normalizada, e que operava a partir de uma interface não normalizada numa rede protótipo. Das alterações efectuadas é de salientar a implementação da placa de interface *SMDS* que, graças à sua flexibilidade, pode ser utilizada para executar tanto testes de conformidade e de interoperabilidade, como testes de avaliação dos parâmetros de desempenho do *SMDS* na rede.

O desenvolvimento da interface *SMDS* vem, por si só, acrescentar ao SGDT, a vantagem de dispor de um acesso normalizado a uma rede com esse serviço; deste modo passou a ser possível executar testes com esta ferramenta a partir de qualquer ponto de acesso a esse serviço. Esta interface foi já utilizada na fase de integração do projecto *MANTIS*, com meio para avaliar as caracteristicas de desempenho e de qualidade do *SMDS* sobre uma rede *ATM*.

Novas funcionalidades foram desenvolvidas para cada uma das entidades do SGDT por forma a garantir as características exigidas a um equipamento cliente do serviço *SMDS*. Embora os recursos físicos disponíveis em cada Bloco Servidor não fossem os ideais, foi possível contornar o problema através da utilização conjunta dos recursos dos Blocos Controlador e Servidor de forma a serem desempenhadas com sucesso a totalidade das funções.

Outro objectivo de particular importância era definir um mecanismo que permitisse ao operador do SGDT definir testes com flexibilidade e compatibilidade superiores às proporcionadas pelas sessões de teste que tinham sido usadas anteriormente. A selecção do *TTCN* (*Tree and Tabular Combined Notation*), como notação utilizada para descrever testes de conformidade, provou ser uma boa aposta uma vez que se trata de uma notação normalizada e já com alguma aceitação por parte de outros fabricantes de equipamentos para executar este tipo de testes. A descrição dos testes em *TTCN* é realizada através de ferramentas como o *ITEX* que permitem o fácil manuseamento desta notação. O *ITEX* foi a ferramenta seleccionada por permitir gerar código *ANSI C* correspondente ao teste descrito em *TTCN* que pode depois ser adaptado ao SGDT. Assim, são reunidas as potencialidades dos dois sistemas usufruindo ambos dos recursos do outro.

Pode considerar-se que o objectivo da definição de uma interface com o operador do SGDT, normalizada e mais versátil, foi conseguido, ao ser desenvolvida uma plataforma de adaptação das funções desempenhadas pelo SGDT ao código *ANSI C* gerado com o *ITEX*. Deste modo o operador pode utilizar todo o conjunto de facilidades do *ITEX* para descrever os testes e executá-los através do SGDT.

O último objectivo consistia em analisar como é que o SGDT poderia ser usado para efectuar os teste de conformidade e de interoperabilidade num ambiente *SMDS*. Esse objectivo foi também realizado com a apresentação e discução de algumas configurações de teste que podem ser utilizadas para efectuar testes num ambiente *SMDS*.

Uma série de dificuldades surgiram na fase de integração, pois o *ITEX* é uma ferramenta que, como o SGDT, está ainda em fase de desenvolvimento e a versão do *ITEX* que nos foi disponibilizada, por não ser definitiva, apresentava ainda alguns defeitos no módulo de geração do código *ANSI C*. Tal facto obrigou a que fossem desenvolvidos pequenos programas para simulação das várias chamadas às funções de adptação que seriam invocadas pelo código *ANSI C* gerado com o *ITEX*.

Perspectivas Futuras

Um vasto leque de áreas de possível adaptação do SGDT são abertas face à grande versatilidade deste equipamento, versatilidade esta que existe principalmente ao nível dos Blocos Servidores e das respectivas interfaces. Os primeiros podem ser programados de forma dinâmica, de modo a adaptarem-se às interfaces às quais estão conectados, desempenhando assim uma infinidade de funções de interligação entre o Bloco Controlador e outros serviços que venham a ser normalizados. As interfaces por serem fisicamente independentes dos restantes Blocos podem ser desenvolvidas por forma a adaptar o sistema a diferentes protocolos e tecnologias.

Numa fase inicial de evolução seria interessante adaptar esta ferramenta para possibilitar a execução de testes descritos em *TTCN* concorrente [Walter93], implementando o conjunto de entidades e de funções de adaptação necessárias para a execução desses testes.

O desenvolvimento de interfaces para *ATM* poderá ser uma evolução possível no sentido de permitir testar serviços que, como o *SMDS*, permitam aos clientes utilizar essa tecnologia.

Acompanhar o aparecimento e o desenvolvimento de novos serviços de transferência de dados sobre redes de banda larga deverá ser uma prioridade, por forma a permitir o investimento atempado na adaptação do SGDT a esses serviços em fases anteriores à da criação das aplicações suportadas por eles.

A evolução das ferramentas de teste, como por exemplo, as de geração automática de testes a partir das especificações e as de validação desses testes, tem sido objecto da investigação levada a efeito por alguns fabricantes de ferramentas [Schnei92] [Boulli94]. Numa perspectiva futura, poderá ter interesse definir uma plataforma para adaptar o SGDT a essas ferramentas.



Referências

[802.6_91] ISO/IEC CD 8802-6
"Telecommunications and Information Exchange Between Systems"
ISO/IEC, 1991

[Altera91] ALTERA
Data Book, 1991

[Antone93] Ferruccio Antonelli e E. Perretti
"*SMDS service in Europe: a public Operator´s experience*"
SPIE Vol. 1975 Local and Metropolitan Area Networks, 1993

[Assis93a] Pedro Assis, A. Almeida, E. Ferreira, M. Henriques, M. Ricardo, P. Proença e E. Carrapatoso
"*TGDS - A new tool for HSLAN performance evaluation*"
Proced. of Local and Metropolitan Area Networks Berlin, 1993

[Assis94] Pedro Assis
"*Ferramentas e Métodos para Availação do Desempenho de Redes de Comunicação*"
Tese de Mestrado - FEUP 1994

[Bellco91] Bellcore
"*Generic System Requirements in Support of Switched Multi-megabit Data Service*" Bellcore, 1991

[Bernd93] Bernd Baumgarten
"*Open Issues in Conformance Testing Specifications*"
International Workshop on Protocol Test Systems VII, 1994

[Boulli94] Laurent Boullier, B. Kelly, M. Phallipou, A. Rouger e N. Webster
"*Evaluation of some test generation tools on a real protocol example*"
International Workshop on Protocol Test Systems VII, 1994

[C25_86] Texas Intruments
TMS320C25 User's Guide, 1986

[Cairns92] Alan Cairns
"*OSI Interoperability - Making It Work*"
Proced. of Interoperability Symposium, 1992

[Carrap90] Eurico Carrapatoso, M. Ricardo e C. Aguiar
"*Traffic Generator/Detector System Concept*"
EFOCLAN 90 - LAN Proceedings of The Seventh Annual European Fibre Optic Communication and Local Area Network Exposition, 1990

[Clapp91] George H. Clapp
"*LAN Interconnection Across SMDS*"
IEEE Network Magazine, 1991

[Dwyer93] D. J. Dwyer
"*The Silence of the LANs*"
Protocol Test Systems V, 1993

[EGCT93] ETSI-EWOS Ad-Hoc Group in Interoperability Testing
"*Classification of Interoperability*"
EGCT/92/030 R4, 1992

[ESIG92] European SMDS Interest Group
"*SMDS Subscriber Network Access Facility Service and Level 2 and 3 Subscriber Network Interface Specification*"
ESIG, 1992

[ESIG93] European SMDS Interest Group
"*SMDS Subscriber Network Interface Level 1 Specification*"
ESIG, 1993

[ETSI92] European Telecommunications Standards Institute
"*Goals and Criteria for Interoperability*"
Sixth Working Draft of DTR/ATM-2003, 1992

[Ferrei95] Eduarda Ferreira e E. Carrapatoso
"*Final SMDS Testing System*"
MANTIS Internal TDI INESC, 1995

[G703_72] ITU Recommendation G.703
"*Physicl/Electrical Characteristics of Hierarchical Digital Interface*"
ITU, 1972

[G704_84] ITU Recommendation G.704
"*Synchronous Frame Structures used at Primary and Secondary Hierarchical Levels*"
ITU, 1984

[G706_88] ITU Recommendation G.706
"*Frame Alignment and Cyclic Redundancy Check (CRC) procedures relating to basic frame structures defined in Recomendation G.704*"
ITU, 1988

[G823_80] ITU Recommendation G.823
"*Controlled Slip Rate Objectives on an International Digital Connection*"
ITU, 1980

[Guimar95] Fernando Guimarães e E. Carrapatoso
"*Final SMDS PC Module*"
MANTIS Internal TDI INESC, 1995

[Guimar95a] Fernando Guimarães
"Equipamento Terminal para Acesso ao Serviço SMDS"
Tese de Mestrado - FEUP 1995

[Haven92] M. E. van der Haven, M Kockelmans e E. Slotboom
"*Real Effects Testing of OSI Applications*"
Protocol Test Systems IV, 1992

[Hemric90] Christine Hemrick e L. Jang
"*Introduction to Switched Multi-megabit Data Service (SMDS), An Early Broadband Service*"
Proced. of XIII International Switching Symposium Vol. 2, 1990

[Hemric91] Christine Hemrick, T. Cox, F. Dix e J. McRoberts
"*SMDS: The Beginning of WAN Superhighways*"
Data Communications, 1991

[Heywoo94] Peter Heywood e E. Gronert
"*SMDS Is Here: Where Are the Customers?*"
Data Communications, 1994

[Horowi89] Paul Horowitz and W. Hill
The Art of Electronics
Cambridge University Press, 1989

[Lenzin92] L. Lenzini e F. Zoccolini
"*Interoperbility tests on OSI products in the framework of OSIRIDE - Interest iniciative*"
Computer Networks and ISDN Systems 24, 1992

[Mamede93] Jorge Mamede, P. Assis, A. Almeida, M. Henriques, M. Ricardo, E. Ferreira, F. Guimarães, P. Proença e E. Carrapatoso
"*TGDS - A system for performance evaluation of High Speed Networks*"
INESC, 1993

[Mamede93a] Jorge Mamede, M. Ricardo, E. Ferreira, F. Guimarães, M. Henriques, J. Silva e E. Carrapatoso
"*Functional and Non-Functional Testing of ATM Networks*"
INESC, 1994

[MOTO95] MOTOROLA
MC 68020 - 32 Microprocessor User´s Manual Motorola, 1985

[OS9_87] Microware
Advanced Systems Software, 1987

[Prober92] R. L. Probert e O. Monkewich "*TTCN: the international notation for specifying tests of communication systems*"
Computer Networks and ISDN Systems 23, 1992

[Proenç93] Paulo Proença, E. Ferreira e E. Carrapatoso
"*Testing Methodology Definition*"
MANTIS Internal TDI INESC, 1993

[RACE92] E. Papachristou e J.Burnmeister
"*Methods for QoS Verification and Protocol Verification in IBC - SoA Report*"
RACE Report, 1992

[Ricard90] Manuel Ricardo C. Aguiar e E. Carrapatoso
"*Traffic Generator/Detector System for High Speed Networks*"
Communication Control and Signal Processing - Proceedings of the 1990 Bilkent International Conference - Bilcon 90, 1990

[Ricard91] Manuel Ricardo
"*Especificação e Estratégia de Implementação de um Sistema de Geração e Detecção de Tráfego*"
Tese de Mestrado - FEUP 1991

[Rochat??] Cedric Rochat
"*How To Write a Driver Under OS-9*"

[Rodeck93] Arnim Rodeck
"*Evolution of protocol testers from SMDS to B-ISDN*"
SPIE Vol. 1975 Local and Metropolitan Area Networks, 1993

[Schnei92] U. Bär e J. M. Schneider
"*Automated Validation of TTCN Test Suites*"
Protocol Test Systems XII, 1992

[SPAG88] S R Dooley
"*Conformance and Interoperability Testing*"
SPAG Services Report 7.2, 1988

[Takaha94] Ken'ichi Takahashi, S. Suzuki, K. Sawai, M. Hatafuku, K. Gotoh e K. Kazama
"*Design and Implementation of an Interconnectability Testing System - AICTS*"
International Workshop on Protocol Test Systems VII, 1994

[Vermee91] Gert Vermeer
"*Interoperability : A confusion of tongues ?*"
PTT Research - NL, 1991

[Vermee93] Gert Vermeer e Hans Blik
"*The way towards INTEROPERABILITY*"
Protocol Test Systems V, 1993

[Vermee93a] Gert Vermeer e H. Blik
"*Interoperability Testing: Basis for Acceptance of Communication Systems*"
Protocol Test Systems V, 1993

[Walter93] Thomas Walter e B. Plattner
"*An Operational Semantics for Concurrent TTCN*"
Protocol Test Systems V, 1993

[Wright93] David Wright
Broadband: Business Sevices, Technologies and Strategic Impact
Artech House,1993

[Wrigle90] Grace A. Wrigle e K. Hanner
"*Future Operations for Switched Multi-megabit Data Service*"
GLOBECOM 90 Vol. 1, 1990

[X290_92] ITU Recommendation X.290
"*OSI Conformance Testing Methodology and Framework for Protocol Recommendations for CCITT Applications - General Concepts*"
ITU, 1992

[X291_92] ITU Recommendation X.291
"*OSI Conformance Testing Methodology and Framework for Protocol Recommendations for CCITT Applications - Abstract Test Suite Specification*"
ITU, 1992

[X292_92] ITU Recommendation X.292
"*OSI Conformance Testing Methodology and Framework for Protocol Recommendations for CCITT Applications - The Tree and Tabular Combined Notation (TTCN)*" ITU, 1992

[X293_92] ITU Recommendation X.293

"*OSI Conformance Testing Methodology and Framework for Protocol Recommendations for CCITT Applications - Test Realization*"

ITU, 1992

[X294_92] ITU Recommendation X.294

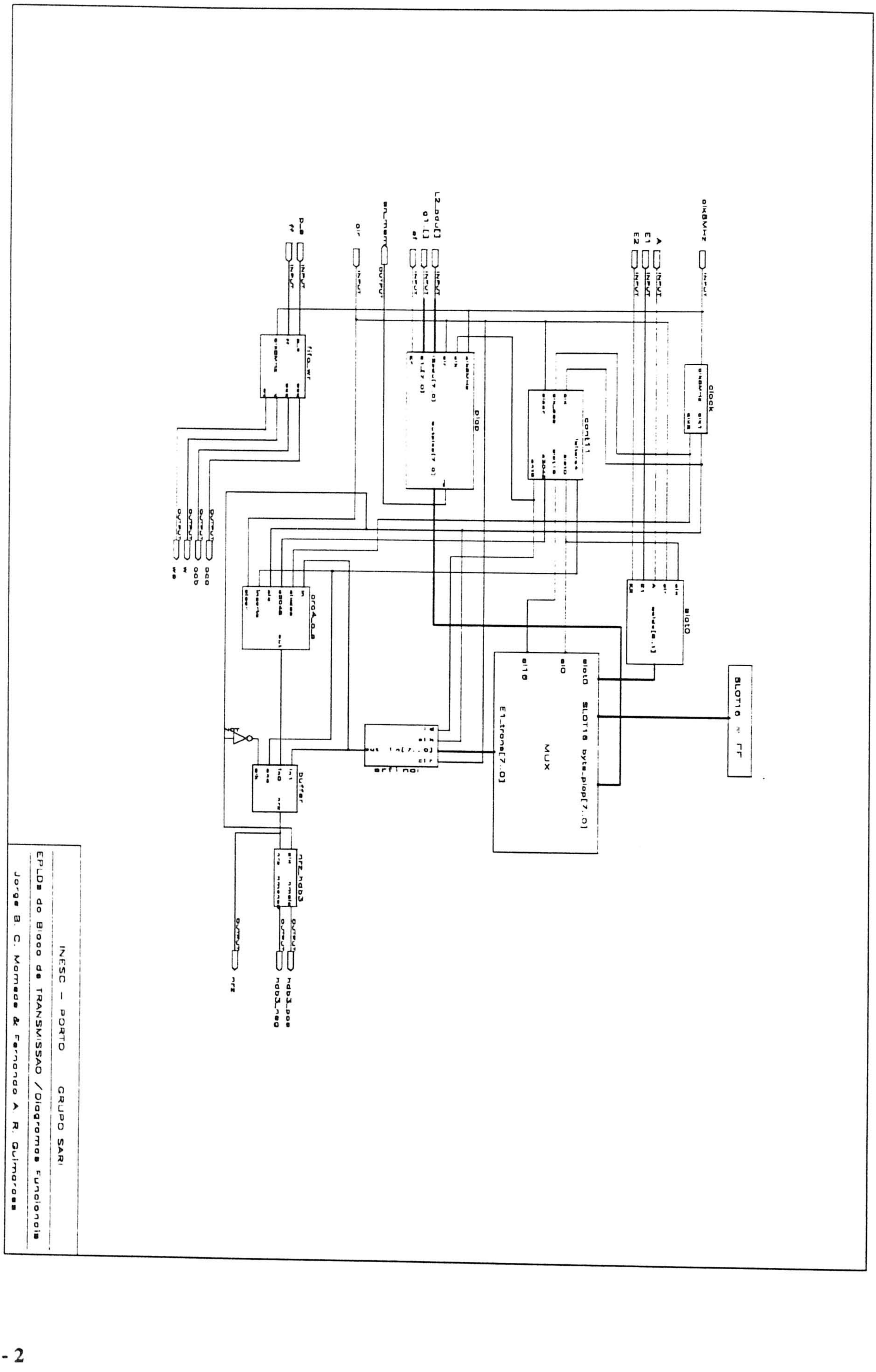
"*OSI Conformance Testing Methodology and Framework for Protocol Recommendations for CCITT Applications -Requirements on Test Laboratories and Clients for the Conformance Assessment Process*"

ITU, 1992

Apêndice A

Blocos Funcionais dos EPLDs

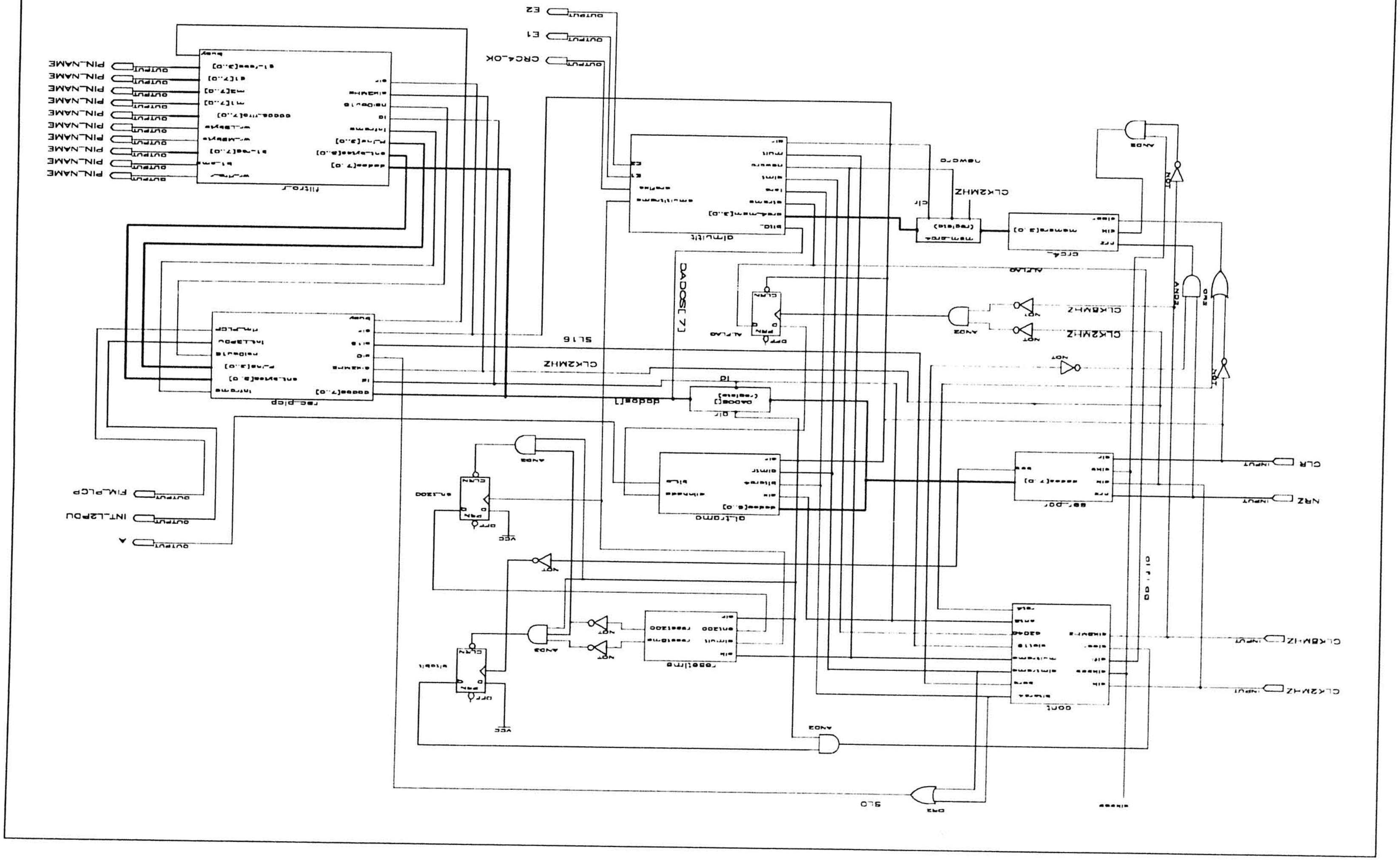
da Placa de Interface SMDS


clock
MUX
SLOT10 = FF
INESC – PORTO GRUPO SARI
EPLDs do Bloco de TRANSMISSAO / Diagramas Funcionais
Jorge B. C. Moreira & Fernando A. R. Guimarães

INESC – PORTO GRUPO SARI

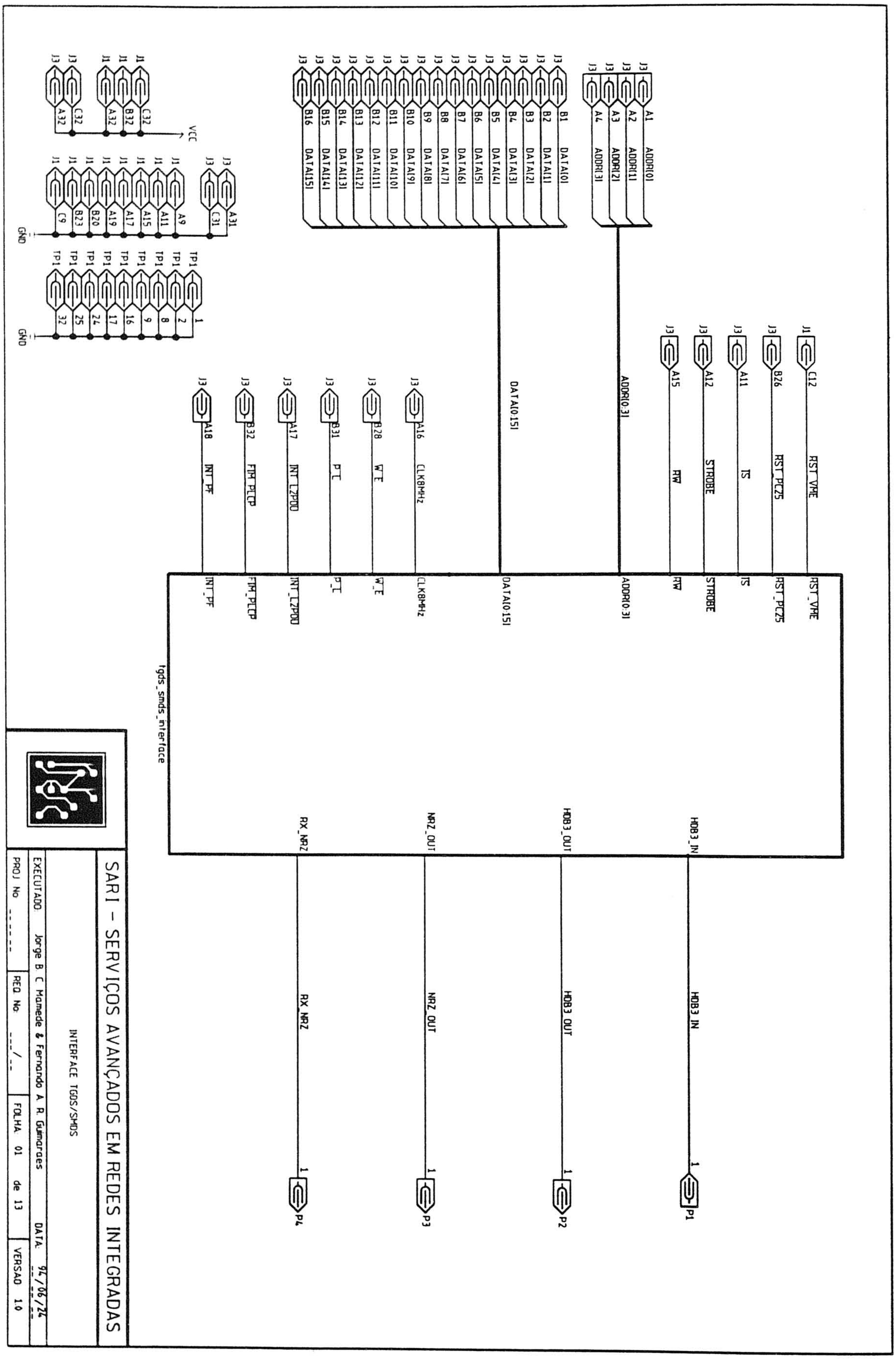
EPLDs do Bloco de RECEPCAO / Diagramas Funcionais

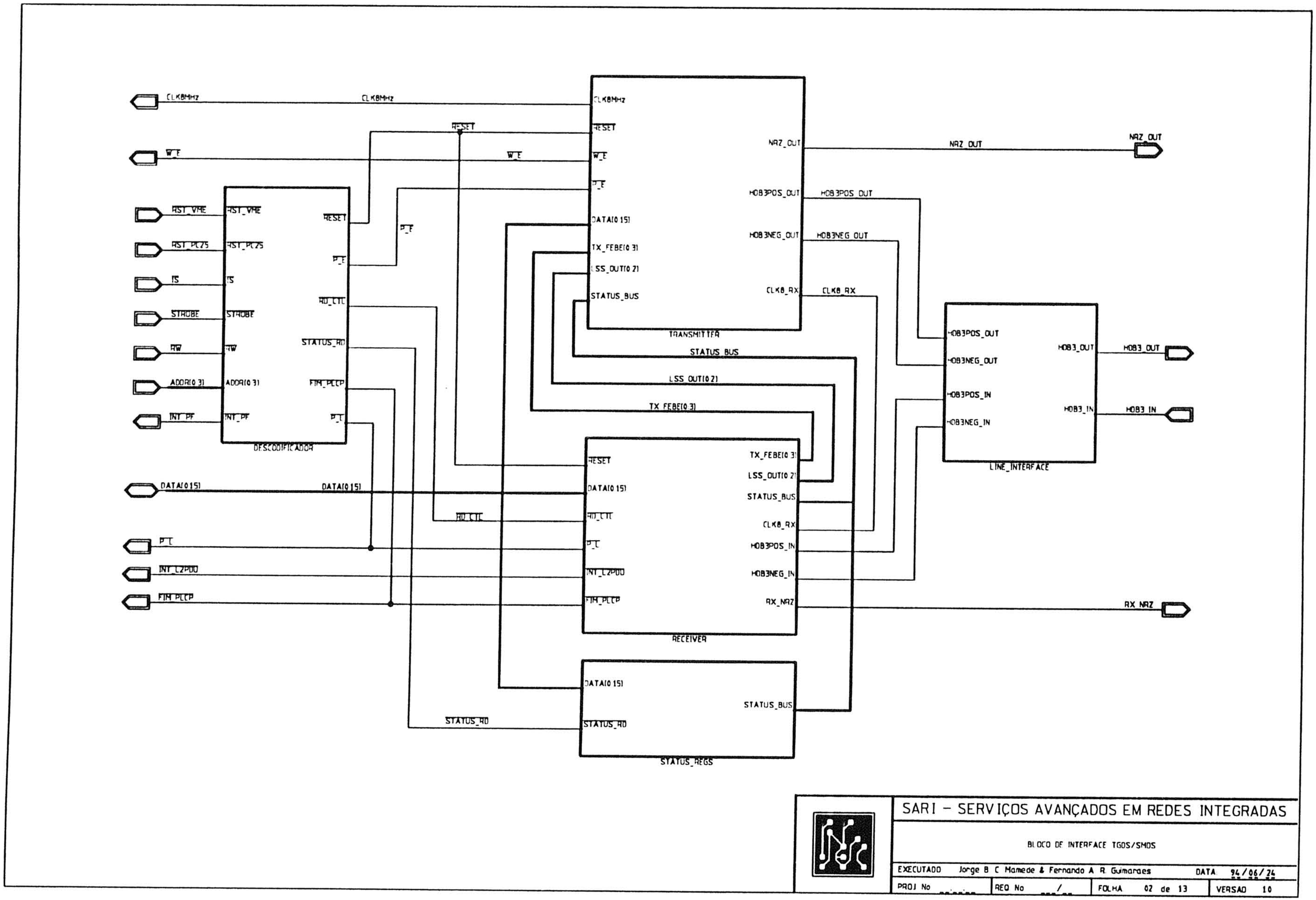
Jorge B. C. Momeda & Fernando A. R. Guimaraes



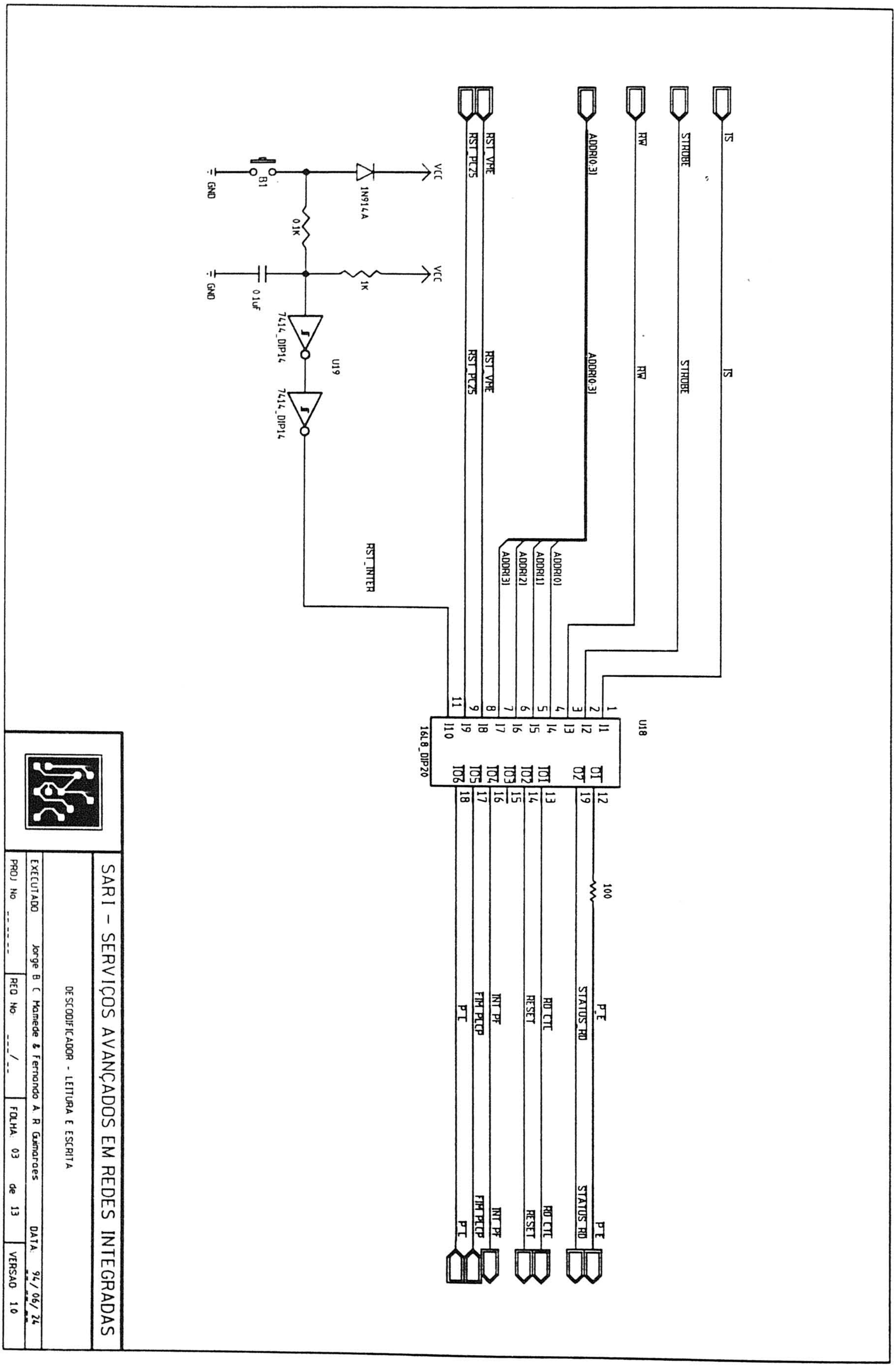
Apêndice B

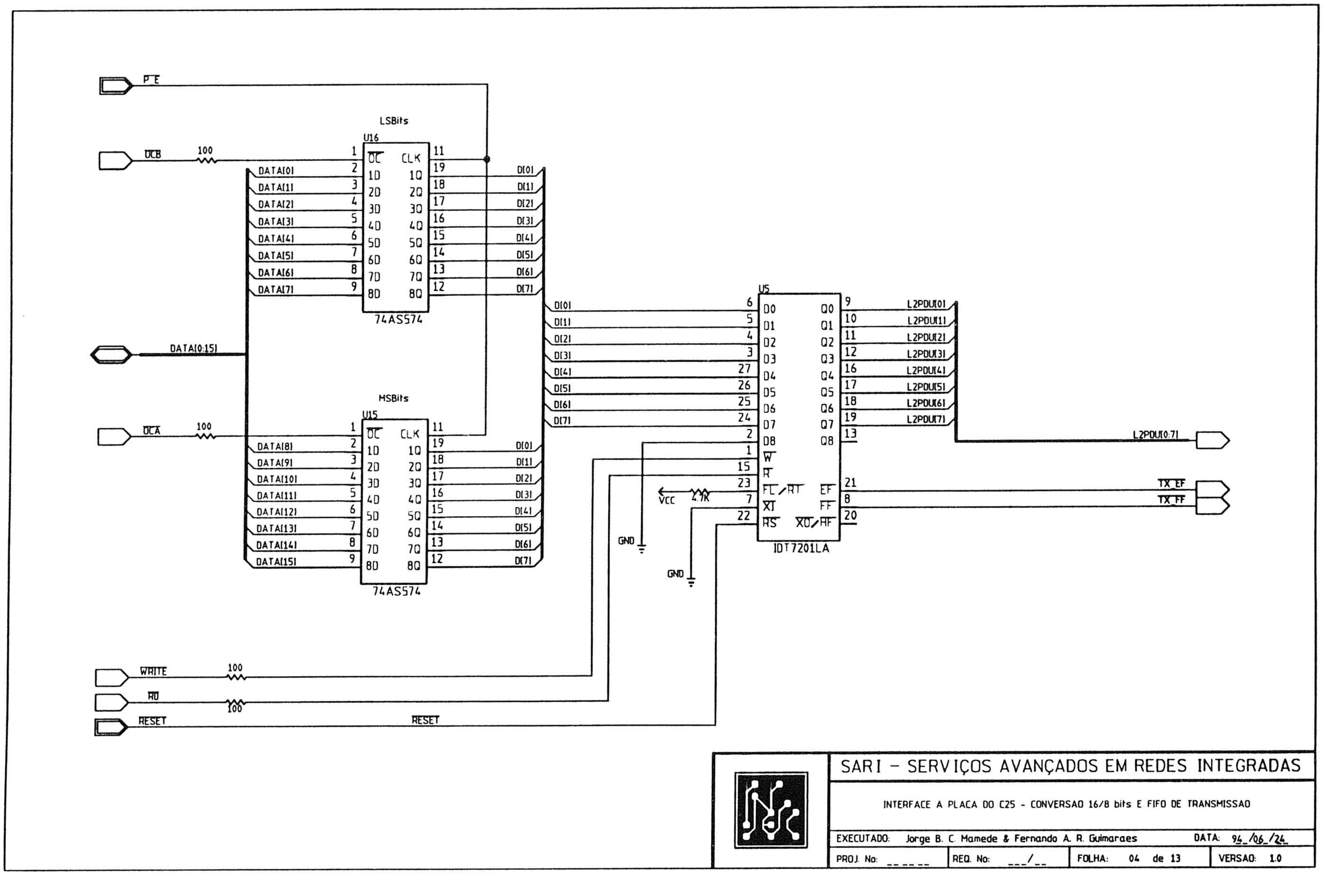
Esquemáticos da Placa de Interface SMDS


J3 A1 ADDR(0)
J3 A2 ADDR(1)
J3 A3 ADDR(2)
J3 A4 ADDR(3)
J3 B1 DATA(0)
J3 B2 DATA(1)
J3 B3 DATA(2)
J3 B4 DATA(3)
J3 B5 DATA(4)
J3 B6 DATA(5)
J3 B7 DATA(6)
J3 B8 DATA(7)
J3 B9 DATA(8)
J3 B10 DATA(9)
J3 B11 DATA(10)
J3 B12 DATA(11)
J3 B13 DATA(12)
J3 B14 DATA(13)
J3 B15 DATA(14)
J3 B16 DATA(15)
J1 C12 RST_VME
J3 B26 RST_PCI25
J3 A11 TS
J3 A12 STROBE
J3 A15 R/W
ADDR(0:3)
DATA(0:15)
RST_VME
RST_PCI25
TS
STROBE
R/W
ADDR(0:3)
DATA(0:15)
J3 A16 CLK8MHz
J3 B28 W/E
J3 B31 PTC
J3 A17 INT_LZP00
J3 B32 FIM_PLCP
J3 A18 INT_PF
CLK8MHz
W/E
PTC
INT_LZP00
FIM_PLCP
INT_PF
HDB3_IN
HDB3_OUT
NRZ_OUT
RX_NRZ
tgds_smds_interface
HDB3_IN
HDB3_OUT
NRZ_OUT
RX_NRZ
P1
P2
P3
P4
J1 C9 A9
J1 B23 A11
J1 B20 A15
J1 A19 A17
J3 C31 A31
GND
TP1 1
TP1 2
TP1 8
TP1 9
TP1 16
TP1 17
TP1 24
TP1 25
TP1 32
GND
J3 A32 C32
J1 A32 B32 C32
VCC
SARI – SERVIÇOS AVANÇADOS EM REDES INTEGRADAS
INTERFACE TGDS/SMDS
EXECUTADO Jorge B. C. Mamede & Fernando A. R. Guimarães
DATA: 94/06/24
PROJ No ------
FOLHA 01 de 13
REQ No ---/--
VERSAO 1.0



SARI – SERVIÇOS AVANÇADOS EM REDES INTEGRADAS				
BLOCO DE INTERFACE TGOS/SMOS				
EXECUTADO Jorge B C Mamede & Fernando A R Guimarães			DATA 94/06/24	
PROJ No	REQ No /	FOLHA 02 de 13	VERSÃO 1.0	

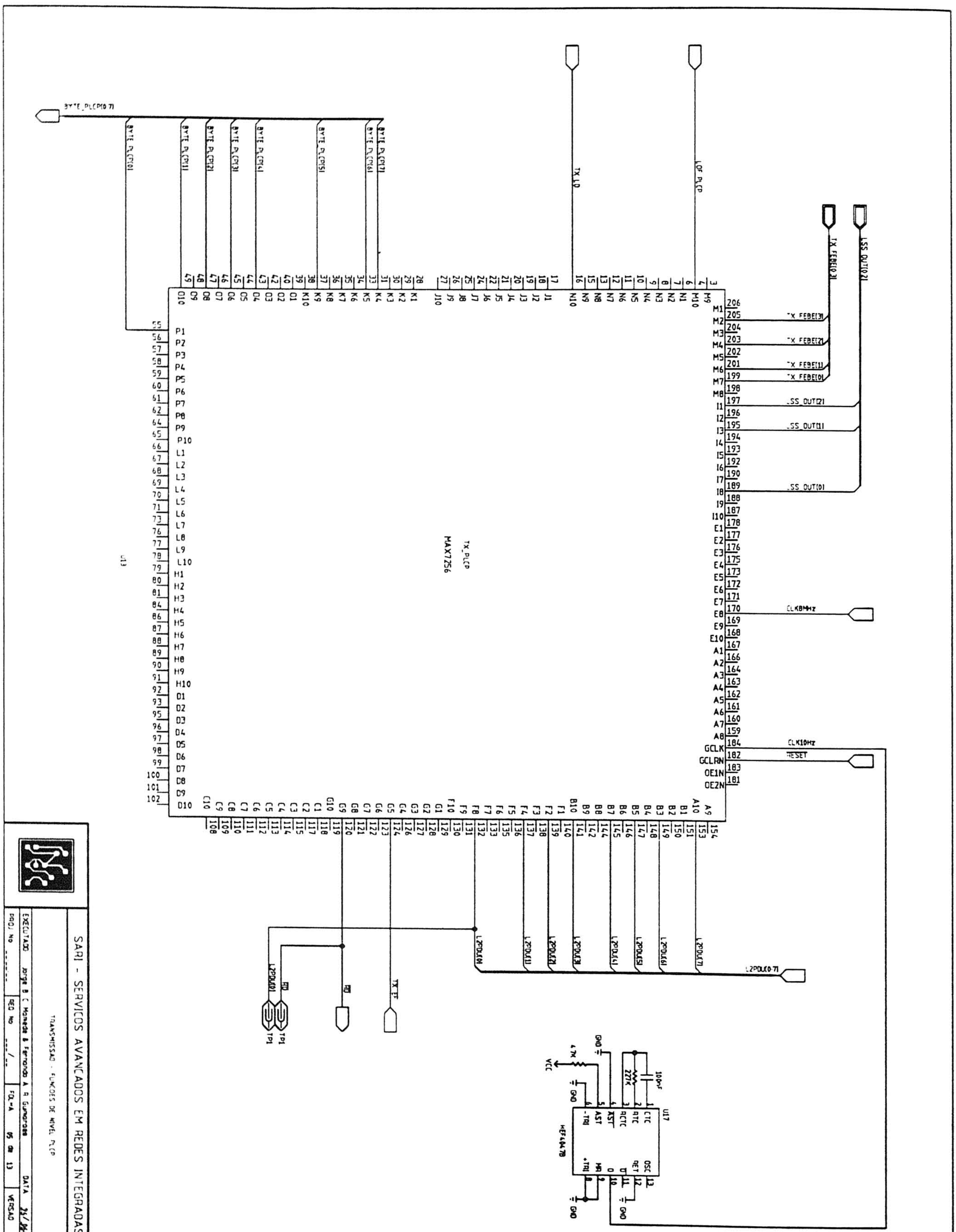

RST_VME
RST_PC25
ADDR(0:3)
RW
STROBE
IS
VCC
B1
GND
1N914A
0.1K
1K
0.1uF
7414_DIP14
U19
RST_INTER
ADDR0
ADDR1
ADDR2
ADDR3
U18
16L8_DIP20
100
STATUS_RD
PE
RD_CTL
RESET
INT_PF
FIM_PLCP
PL
SARI - SERVIÇOS AVANÇADOS EM REDES INTEGRADAS
DECODIFICADOR - LEITURA E ESCRITA
EXECUTADO Jorge B C Mamede & Fernando A R Guimaraes
DATA 94/06/24
PROJ No
REQ No
FOLHA 03 de 13
VERSAO 1.0



SARI – SERVIÇOS AVANÇADOS EM REDES INTEGRADAS

INTERFACE A PLACA DO C25 - CONVERSAO 16/8 bits E FIFO DE TRANSMISSAO

EXECUTADO: Jorge B. C. Mamede & Fernando A. R. Guimaraes			DATA: 94/06/24
PROJ. No: _____	REQ. No: ___/__	FOLHA: 04 de 13	VERSAO: 1.0



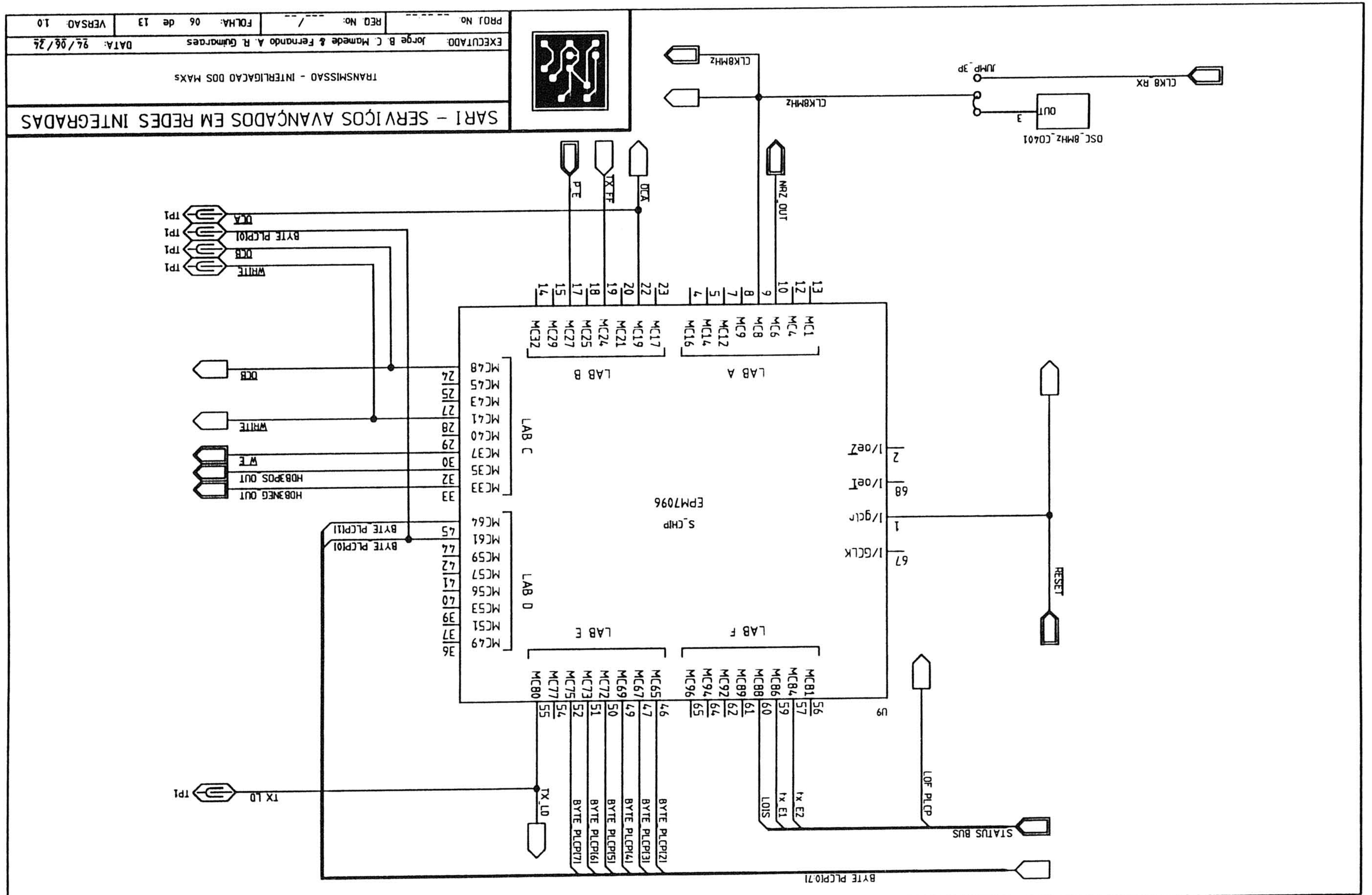
SARI - SERVIÇOS AVANÇADOS EM REDES INTEGRADAS	
EXECUTADO Jorge B C Monteiro & Fernando A B Gonçalves	DATA 25/04/94
TRANSMISSOR - FLUXOS DE NÍVEL PLCP	
PROJ No ------	FOLHA 05 de 13
DES No ---/--	VERSÃO 1.0

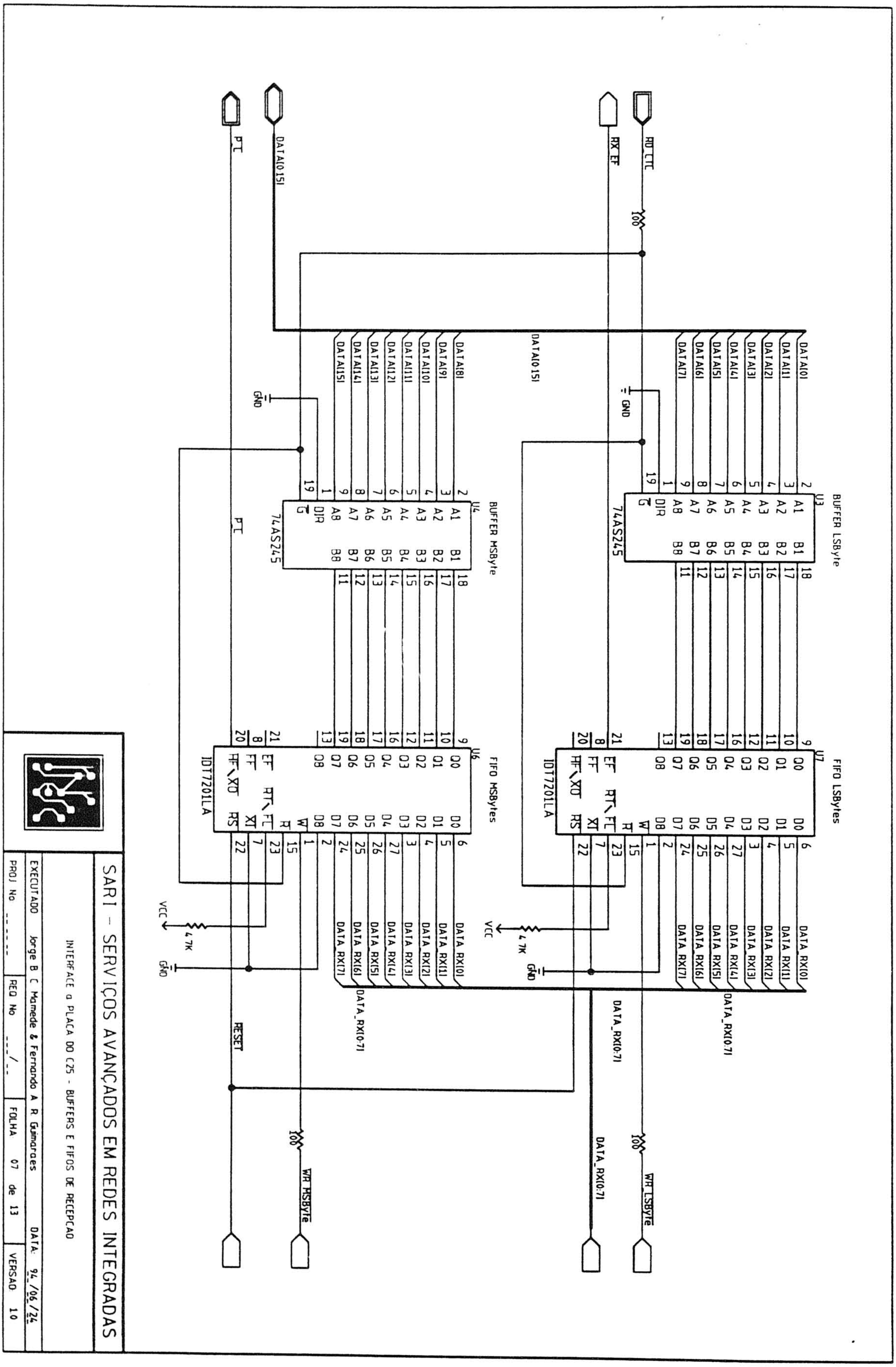
SARI – SERVIÇOS AVANÇADOS EM REDES INTEGRADAS

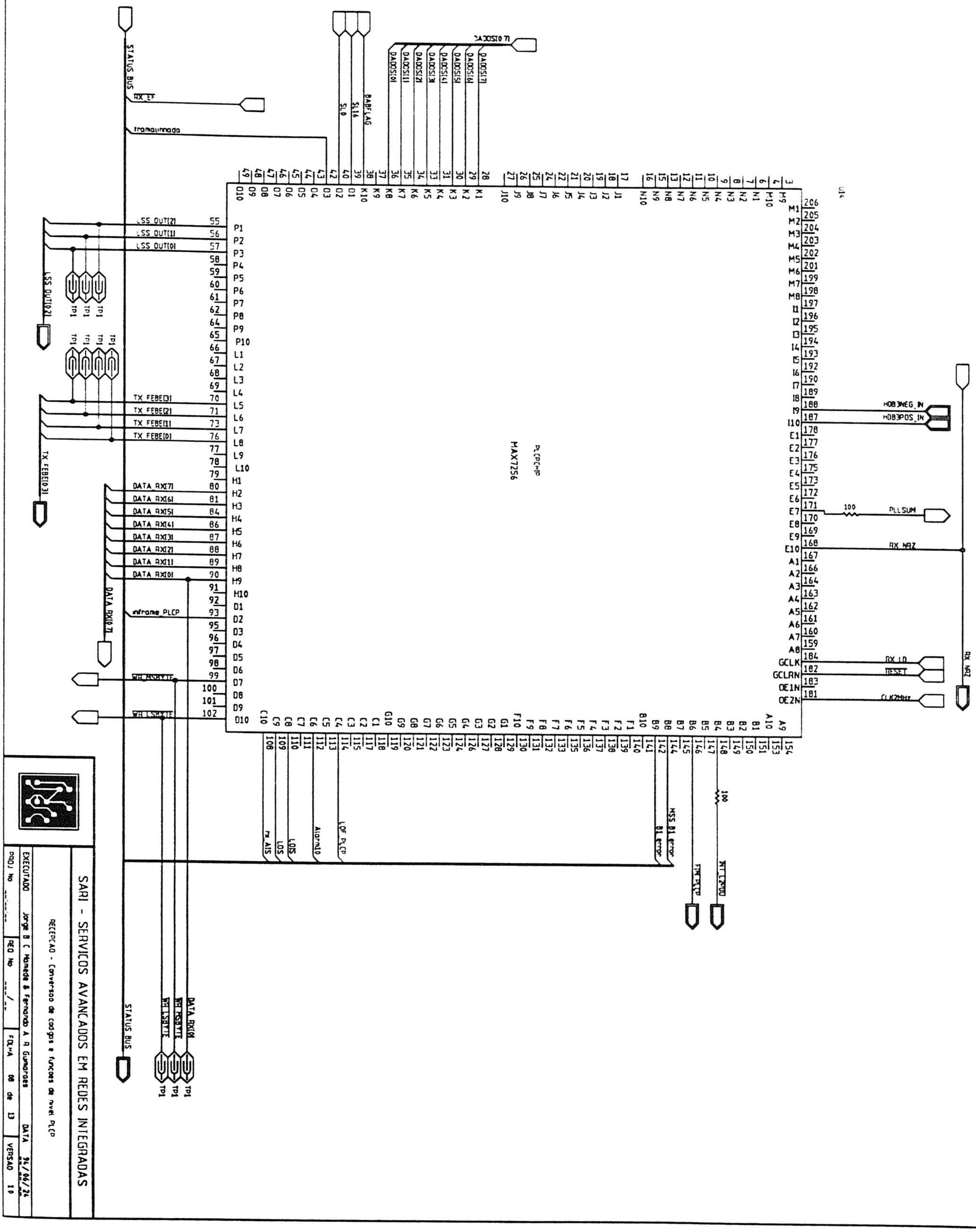
TRANSMISSAO - INTERLIGACAO DOS MAXS

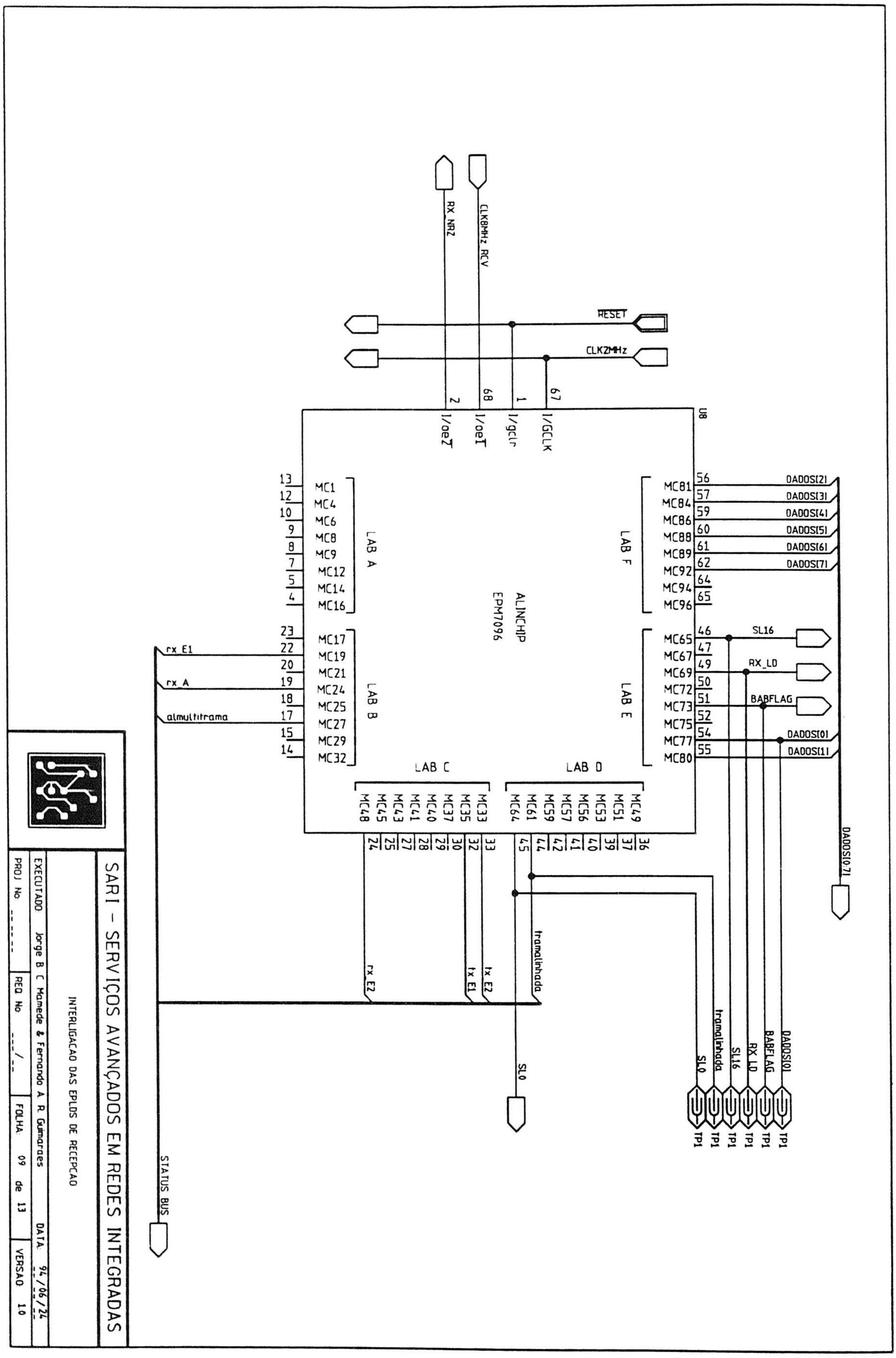
EXECUTADO: Jorge B. C. Mamede & Fernando A. R. Guimaraes | DATA: 24/06/24

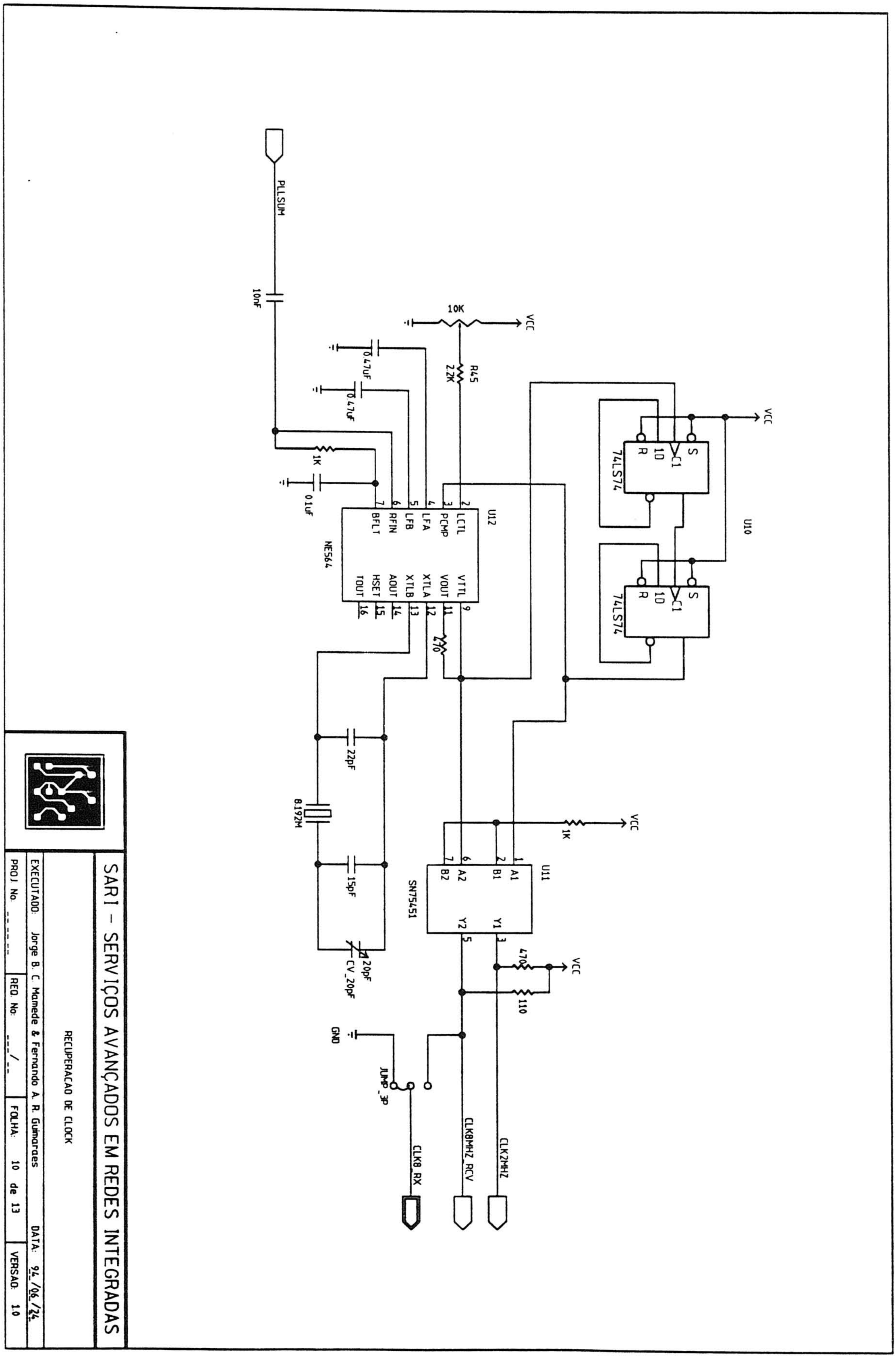
PROJ No: ------ | REQ. No: ---/-- | FOLHA: 06 de 13 | VERSAO: 1.0

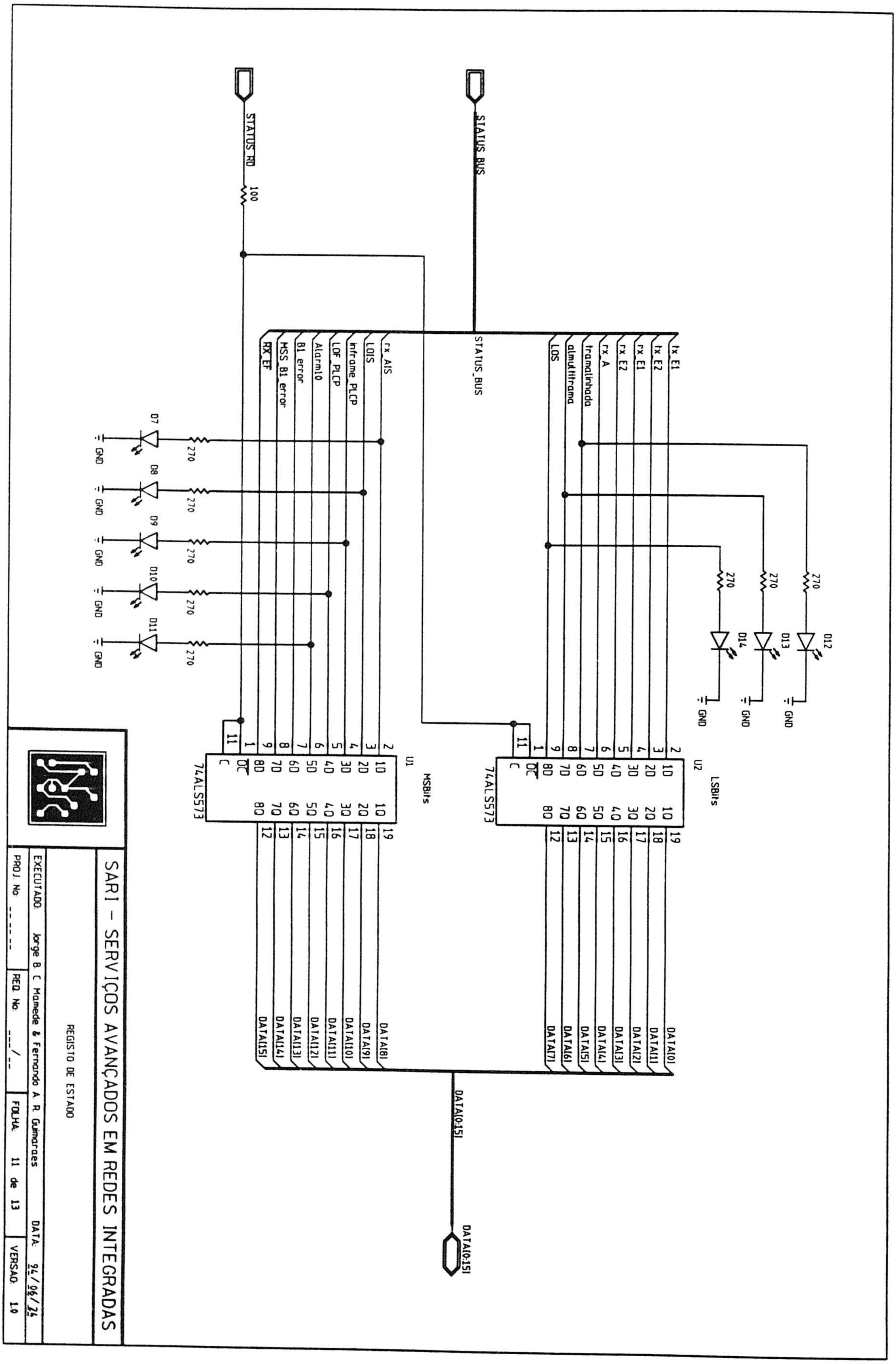


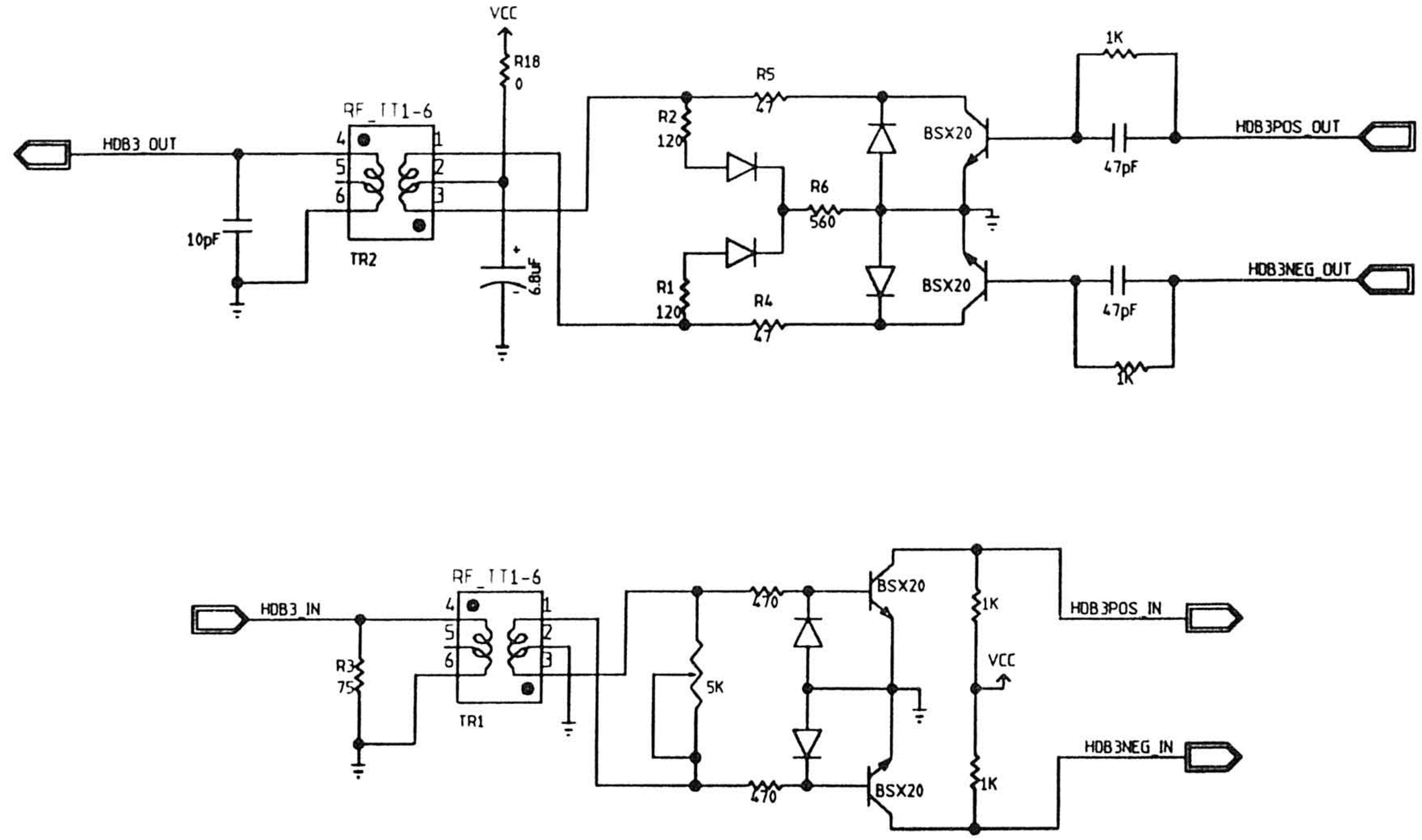

PTL
DATA[0:15]
RX_EF
RD_CTL
100
BUFFER LSByte
U3
DATA[0]
DATA[1]
DATA[2]
DATA[3]
DATA[4]
DATA[5]
DATA[6]
DATA[7]
A1
A2
A3
A4
A5
A6
A7
A8
B1
B2
B3
B4
B5
B6
B7
B8
DIR
G
74AS245
GND
DATA[0:15]
BUFFER MSByte
U4
DATA[8]
DATA[9]
DATA[10]
DATA[11]
DATA[12]
DATA[13]
DATA[14]
DATA[15]
GND
PTL
FIFO LSBytes
U7
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
EF
FF
HF\XO
RT\FL
XI
RS
W
R
IDT7201LA
4.7K
VCC
GND
DATA_RX[0]
DATA_RX[1]
DATA_RX[2]
DATA_RX[3]
DATA_RX[4]
DATA_RX[5]
DATA_RX[6]
DATA_RX[7]
DATA_RX[0:7]
FIFO MSBytes
U6
DATA_RX[8]
DATA_RX[9]
DATA_RX[10]
DATA_RX[11]
DATA_RX[12]
DATA_RX[13]
DATA_RX[14]
DATA_RX[15]
RESET
WR_LSByte
WR_MSByte
SARI - SERVIÇOS AVANÇADOS EM REDES INTEGRADAS
INTERFACE a PLACA DO C25 - BUFFERS E FIFOS DE RECEPCAO
EXECUTADO Jorge B C Mamede & Fernando A R Guimaraes
DATA: 94/06/24
PROJ No ------
REQ No ---/--
FOLHA 07 de 13
VERSAO 1.0


SARI - SERVICOS AVANCADOS EM REDES INTEGRADAS
EXECUTADO Jorge B C Moreira & Fernando A R Guimarães
DATA 94/06/24
FOLHA 00 de 13
VERSAO 10
RECEPCAO - Conversao de codigos e funcoes de nivel PLCP
MAX7256
PLCPCHIP
STATUS BUS
DATA RXIN
WR_MSBYTE
WR_LSBYTE
DATA RXIOUT
DATA RXI[7]
DATA RXI[6]
DATA RXI[5]
DATA RXI[4]
DATA RXI[3]
DATA RXI[2]
DATA RXI[1]
DATA RXI[0]
wframe_PLCP
TX FEBE[3]
TX FEBE[2]
TX FEBE[1]
TX FEBE[0]
LSS OUT[2]
LSS OUT[1]
LSS OUT[0]
LSS OUT[2]
TX FEBE[3]
RX MSZ
RX_LD
RESET
CLK2MHz
HOB3NEG_IN
HOB3POS_IN
PLLSUM
RX MSZ
100
10K
MSS B1 error
B1_error
LOF_PLCP
Alarm10
LOS
LOOS
rx_AIS
STATUS BUS


RX_NRZ
CLK8MHz RCV
RESET
CLK2MHz
2
68
1
67
1/oeT
1/oeT
1/gclr
1/GCLK
U8
LAB A
MC1
MC4
MC6
MC8
MC9
MC12
MC14
MC16
LAB B
MC17
MC19
MC21
MC24
MC25
MC27
MC29
MC32
LAB C
MC33
MC35
MC37
MC40
MC41
MC43
MC45
MC48
LAB D
MC49
MC51
MC53
MC56
MC57
MC59
MC61
MC64
LAB E
MC65
MC67
MC69
MC72
MC73
MC75
MC77
MC80
LAB F
MC81
MC84
MC86
MC88
MC89
MC92
MC94
MC96
ALINCHIP
EPM7096
rx_E1
rx_A
almultitrama
SL16
RX_LD
BABFLAG
DADOSI01
DADOSI11
DADOSI[0:7]
DADOSI[2]
DADOSI[3]
DADOSI[4]
DADOSI[5]
DADOSI[6]
DADOSI[7]
rx_E2
tx_E2
tx_E1
tramalinhada
SL0
STATUS BUS
DADOSI01
BABFLAG
RX_LD
SL16
tramalinhada
SL0
TP1
SARI – SERVIÇOS AVANÇADOS EM REDES INTEGRADAS
INTERLIGACAO DAS EPLDS DE RECEPCAO
EXECUTADO Jorge B C Mamede & Fernando A R Guimaraes
DATA 94/06/24
PROJ No ------
REQ No ---/--
FOLHA 09 de 13
VERSAO 1.0


PLLSUM
10nF
10K
VCC
R45
2.2K
0.47uF
0.47uF
1K
0.1uF
U12
NE564
LCTL
PCMP
LFA
LFB
RFIN
BFLT
VTTL
VOUT
XTLA
XTLB
AOUT
HSET
TOUT
270
U10
74LS74
74LS74
22pF
8.192M
15pF
20pF
CV_20pF
GND
U11
SN75451
A1
B1
A2
B2
Y1
Y2
1K
470
110
VCC
JUMP_JP
CLK8MHZ_RCV
CLK8_RX
CLK2MHZ
SARI – SERVIÇOS AVANÇADOS EM REDES INTEGRADAS
RECUPERACAO DE CLOCK
EXECUTADO: Jorge B. C. Mamede & Fernando A. R. Guimaraes
DATA: 94/06/24
PROJ No. ---/--
REQ. No: ---/--
FOLHA: 10 de 13
VERSAO: 1.0

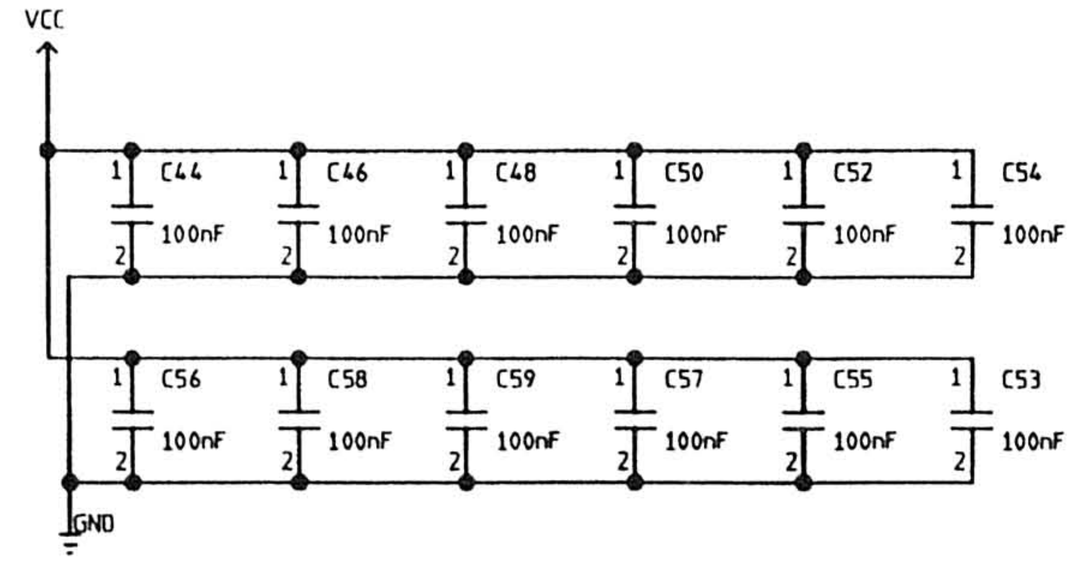

STATUS_RD
100
STATUS_BUS
tx_E1
tx_E2
rx_E1
rx_E2
rx_A
tramalinhada
almultitrama
LOS
rx_AIS
LOIS
inframe_PLCP
LOF_PLCP
Alarm10
B1_error
MSS_B1_error
RX_EF
270
D7
D8
D9
D10
D11
D12
D13
D14
GND
U2
U1
74ALS573
LSBits
MSBits
DATA[0]
DATA[1]
DATA[2]
DATA[3]
DATA[4]
DATA[5]
DATA[6]
DATA[7]
DATA[8]
DATA[9]
DATA[10]
DATA[11]
DATA[12]
DATA[13]
DATA[14]
DATA[15]
DATA[0:15]
SARI – SERVIÇOS AVANÇADOS EM REDES INTEGRADAS
REGISTO DE ESTADO
EXECUTADO: Jorge B. C. Mamede & Fernando A. R. Guimarães
DATA: 94/06/24
PROJ. No: ------
REQ. No: ___/__
FOLHA: 11 de 13
VERSÃO: 1.0

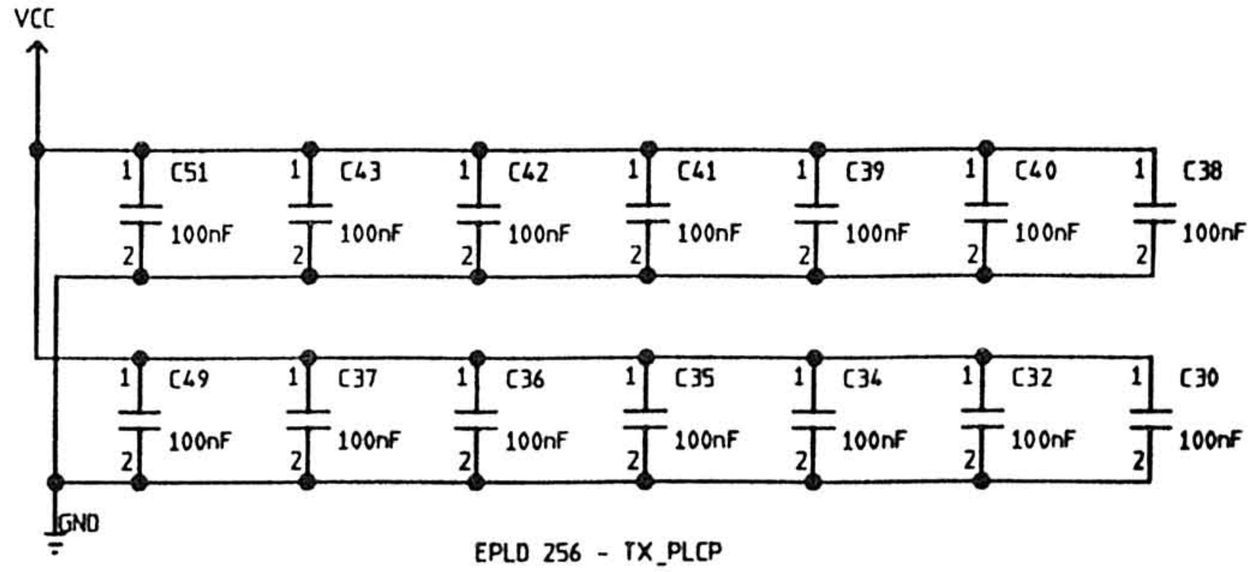


SARI – SERVIÇOS AVANÇADOS EM REDES INTEGRADAS

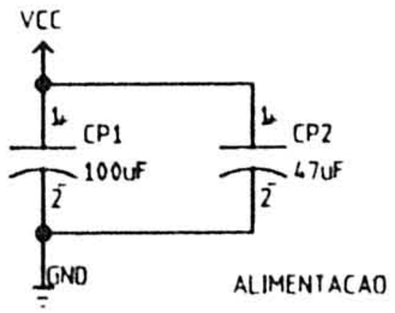
INTERFACE A LINHA DE TRANSMISSAO

EXECUTADO: Jorge B. C. Mamede & Fernando A. R. Guimaraes			DATA: 94/06/24
PROJ. No: _____	REQ. No: ___/__	FOLHA: 12 de 13	VERSAO: 1.0

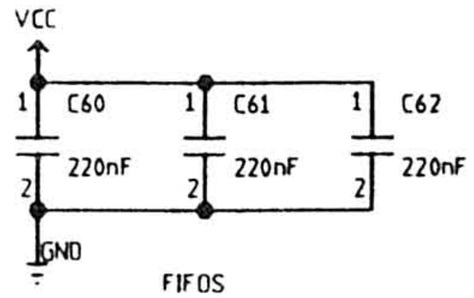




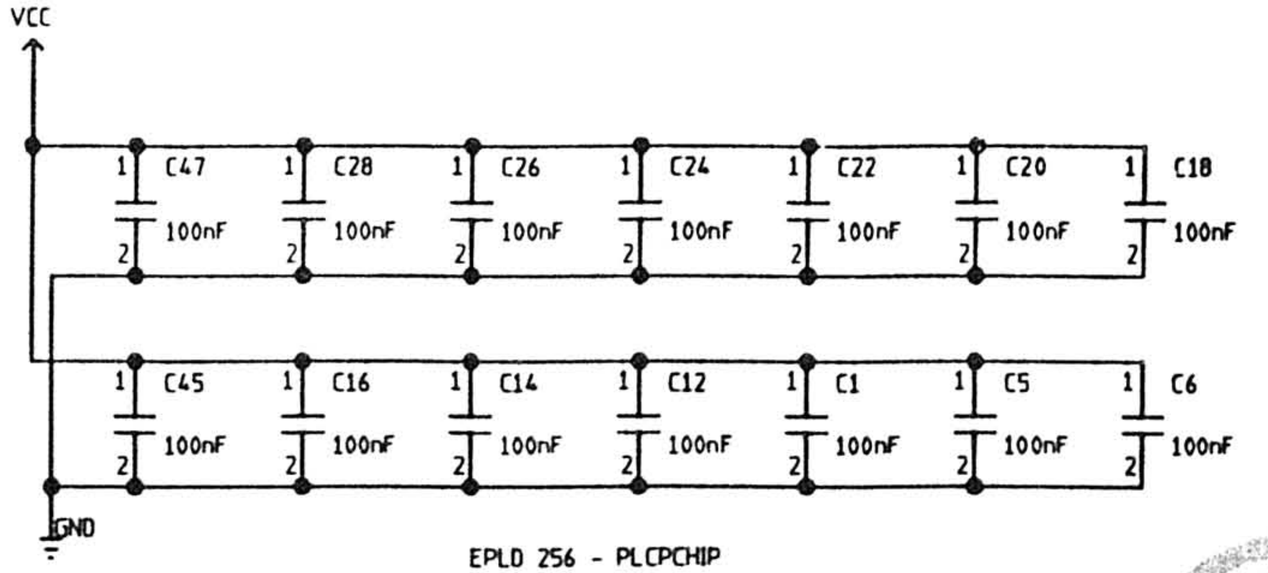
EPLD 256 - TX_PLCP



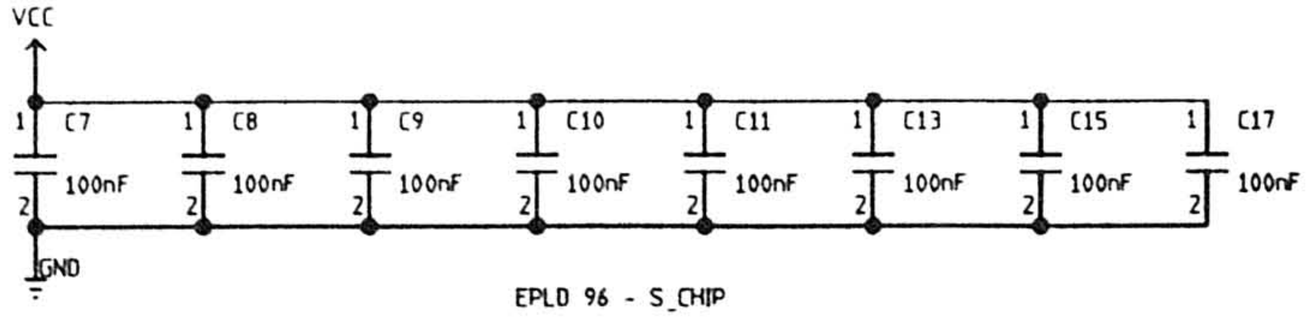
ALIMENTACAO



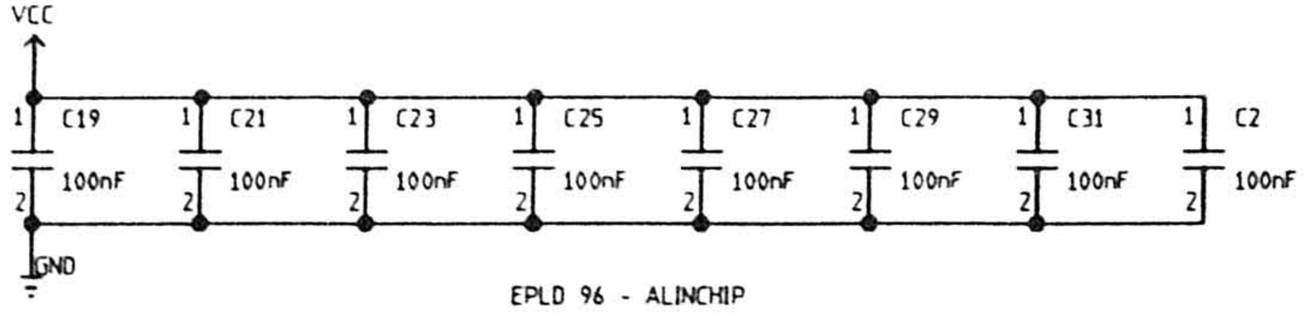
FIFOS



EPLD 256 - PLCPCHIP



EPLD 96 - S_CHIP



EPLD 96 - ALINCHIP



SARI – SERVIÇOS AVANÇADOS EM REDES INTEGRADAS			
REDES DE DESACOPLAMENTO			
EXECUTADO: Jorge B. C. Mamede & Fernando A. R. Guimarães			DATA: 94/06/24
PROJ. No: _____	REQ. No: ___/__	FOLHA: 13 de 13	VERSAO: 1.0

FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO
BIBLIOTECA
0000002149