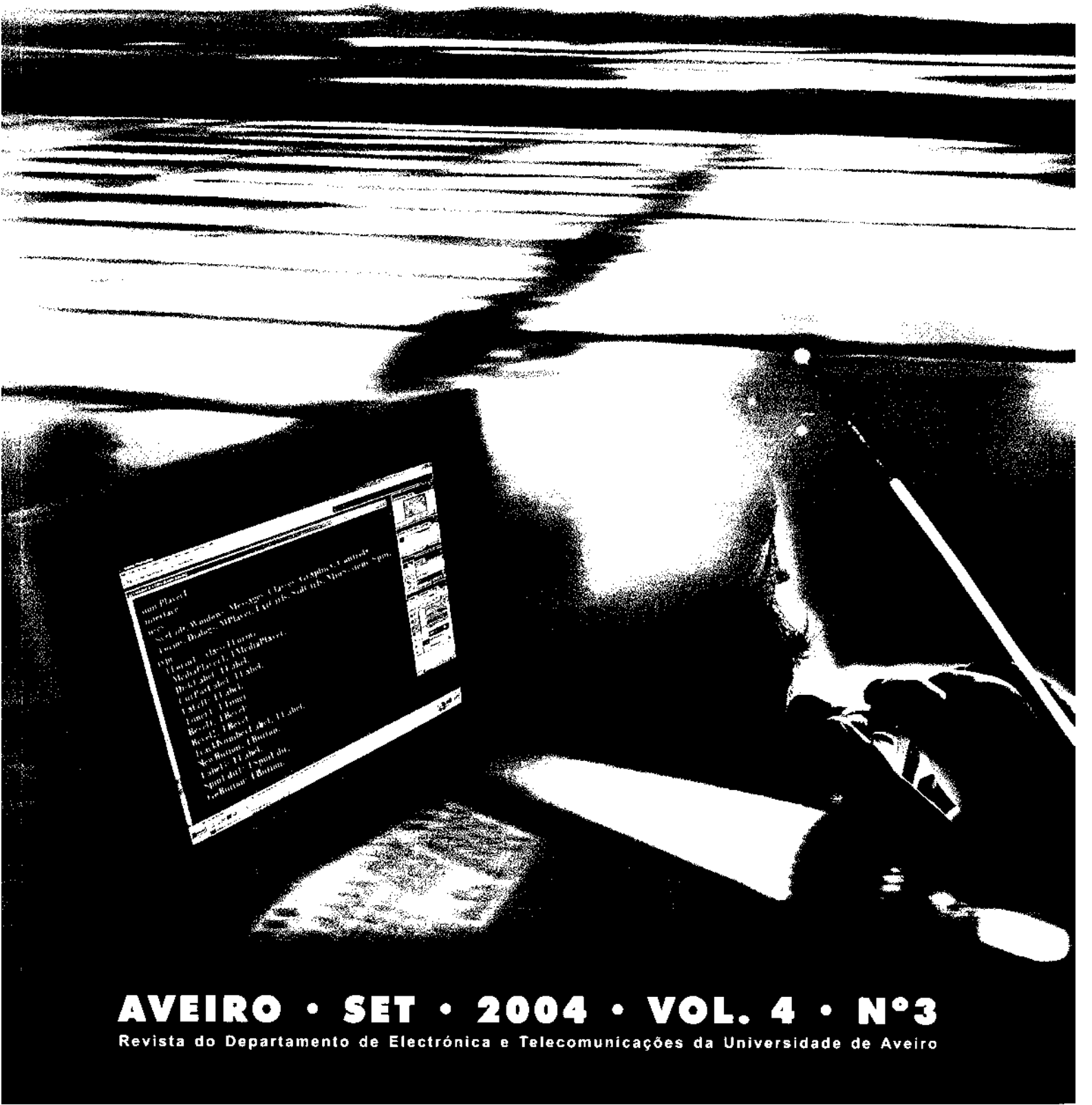
Electrónica e Telecomunicações



AVEIRO • SET • 2004 • VOL. 4 • Nº3

Revista do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro

Electrónica e Telecomunicações

Revista do Departamento de
Electrónica e Telecomunicações
da Universidade de Aveiro

Morada e Secretariado:

Departamento de Electrónica
e Telecomunicações
Universidade de Aveiro
Campo Universitário
3810 AVEIRO
Portugal

Artes Gráficas:

Sérgio Cabaço

Produção: DESIGNEED, Lda
Tiragem : 300 exemplares
Depósito Legal Nº 115607
ISSN: 1645-04

universidade de aveiro

Editorial

Este número da Revista do DET continua a evidenciar a habitual diversidade de interesses científicos e pedagógicos que determinam a nossa actividade como membros do DET. Basta uma rápida análise do índice para identificar temas das áreas das telecomunicações, sistemas de informação, processamento de sinal e imagem e sistemas computacionais, controlo e ainda um interessante artigo sobre detecção em *run-time* de problemas de *stack overflow* em programas escritos em linguagem C.

Esta Revista sempre dependeu significativamente de contribuições decorrentes de trabalhos de alunos desenvolvidos nas disciplinas de Projecto e/ou de Opção. Para esta vertente editorial considerada, desde início, como fundamental continue a evidenciar-se é necessário adaptar o calendário de publicação da Revista ao actual calendário escolar que, como se sabe, terminou com a época de avaliação em Setembro. Tal facto significa que se torna mais difícil apelar aos alunos para que se envolvam em esforços de publicação para a tradicional edição de Setembro, quando de facto a maioria deles já não se encontra no Departamento e/ou têm outras preocupações de índole profissional. Neste contexto passaremos então a partir do próximo ano a ter as duas edições anuais localizadas em Janeiro e Julho.

Aguardando desde já contribuições para uma próxima edição, apresento o meu agradecimento aos autores e a todas as entidades que de uma ou de outra forma viabilizaram a concretização de mais uma edição desta Revista.

Augusto Silva

A edição desta revista é subsidiada pela
Fundação para a Ciência e Tecnologia

Índice

Transponder for Coarse Wavelength Division Multiplexing (CWDM) Optical Communication Systems

V. Barbosa, A. N. Pinto, C. Lemos, R. M. Diz, N. J. Carvalho

Resumo **– Neste artigo é feita a descrição de um transponder óptico bidireccional implementado no âmbito do projecto SIRAC para um sistema óptimo com múltiplos comprimentos de onda espaçados (CWDM). As várias opções tomadas são detalhamento discutidas. É ainda analisada a possível extensão do transponder para sistemas densos com múltiplos comprimentos de onda (DWDM).**

Abstract **– In this paper a CWDM bi-directional optical transponder is described. This transponder was built in the framework of the project SIRAC. The design options followed are detailed discussed. This paper also discusses the possible extension of the transponder to Dense Wavelength Division Multiplexing (DWDM) systems.**

Index Terms **— Optical Transponder, CWDM Systems, DWDM Systems, FEC.**

I. INTRODUCTION

The optical communication systems had an enormous evolution in the last decade. The velocity of the transmitters has grown up to the gigabit rates. The reach of the systems was largely increased by the use of optical amplification. The systems went to single channel to multi-channels, allowing an aggregate capacity of the order of terabit rates.

The very large spectral window of optical fibers makes possible the multiplexing of several wavelengths channels in one single fiber (WDM-Wavelength Division Multiplex), each channel supporting transparently different protocols. This technology was rapidly improved and resulted in two different spectrum dispositions accordingly to the capabilities of the equipment used for transmission and detection, namely: CWDM - Coarse WDM and DWDM – Dense WDM. These two technologies were standardized by the ITU-T in 2002 [1, 2].

DWDM channel allocation is very dense with the possibility of having 150 channels in the spectral window ranging from the 1490nm to 1610nm, taking advantage of the optical transparency window provided by the erbium doped based amplifiers. However, this extremely dense channel allocation puts huge demands in several optical components, mainly in the spectral purity and stability of lasers sources and in the stability and suppression range of optical filters.

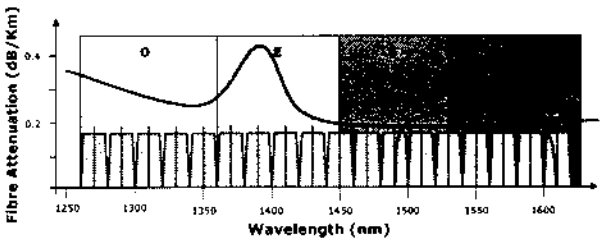


Fig .1 – Comparison of the CWDM and DWDM channel allocation. The CWDM channels are allocated between 1260nm and 1620nm. The DWDM channels are placed around 1550nm. A typical spectrum of fiber intrinsic attenuation is superposed in the graphic.

Due to advances in the fiber manufacturing process, mainly in the purification process regarding the hydroxide ion, the peak of absorption visible in the E band, see figure 1, is almost suppressed in recent fibers. Besides that other optical amplification mechanisms, mainly Raman amplification, have been developed in order to provide gain in all optical bands. Furthermore, fibers have reached the metro market, where distances are small and optical amplification is not needed. Additionally, there is still a great deal of dark fibers in the field that the operators want to make profitable. Having this in mind the CWDM standard was created to take advantageous of a higher separation between optical channels. The CWDM channel allocation has channel spacing of 20nm, whereas DWDM has channel spacing of 100GHz (~0.8nm). This less complex variation of WDM promises a cost-effective way to deliver bandwidth. Continuing technology advances combined with recent strides made in standards organizations, have led to a growing interest in CWDM.

This paper describes the implementation of an optical transponder for CWDM systems.

II. THE TRANSPONDER

The transponder is a module that does the interface between the electrical equipment, typically a SDH or Gigabit Ethernet card, and the optical line system.

According with the etymology **Transmitter-Responder,** a transponder is a device that transmits a specific signal in response to a given input signal that may or may not be of the same physical nature.

In this application the device implements the interface between the client and the optical line, see figure 2. It converts a signal intended to be transmitted in short-distance paths (SR - Short Reach), typically between local equipments, and a signal intended to be transmitted in long-distance paths (LR - Long Reach), typically between remote equipments [3]. Therefore, the projected transponder has two bi-directional optical interfaces, a short reach and a long reach optical interface. The SR interface connects with the SR interface of the electrical equipment. The LR interface connects through the optical line with the counterpart transponder in the other side of the line. The LR interface meets the CWDM standard requirements. The major differences between the SR and LR optical signal, besides the central wavelength, is the spectral purity. Typically the SR signal is generated using an inexpensive multimode laser, and the LR signal is generated by a single-mode laser.

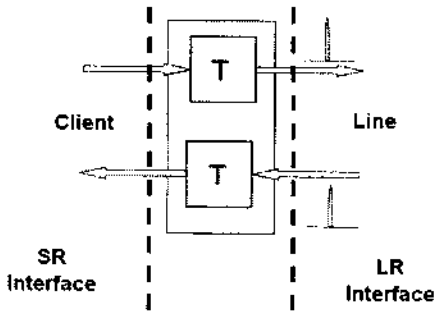


Fig. 2 – A bi-directional transponder.

New Generation Transponders

Taking advantage of recent advances in high speed digital electronic processing the new generation of optical transponders incorporates a module of inclusion and exclusion of extra digital information for forward error correction (FEC). With these FEC modules is possible to increase the bit-rate and/or extend the optical span without any BER penalty. This is achieve increasing slightly the bit-rate in order to incorporate the redundant information introduce by the FEC coding. For example a Reed Solomon code with 255/238 can improve de BER from $1x10^{-4}$ to $5x10^{-15}$, see table 1 [4].

BER_{input}	BER_{output}
10^{-4}	5×10^{-15}
10^{-5}	$6.3x\ 10^{-24}$
10^{-6}	$6.4x\ 10^{-33}$

Table 1 - Theoretical output versus input BER for a Reed Solomon coding with 255/238.

This BER improvement can be swapped by the increase of optical line spans.

III. SYSTEM SPECIFICATIONS

The transponder was specified to meet the standardized data rates specified for Synchronous Digital Hierarchy (SDH).

The information to and from the client (short distance connection) is STM-16 frame format. The CWDM (long distance connection) has information in the STM-16 with FEC encoded frame format.

Each client transmits and receives the information in the 1300nm and 1500nm wavelengths respectively.

The CWDM interface has optical wavelengths defined by the CWDM grid.

IV. TRANSPONDER ARCHITECTURE

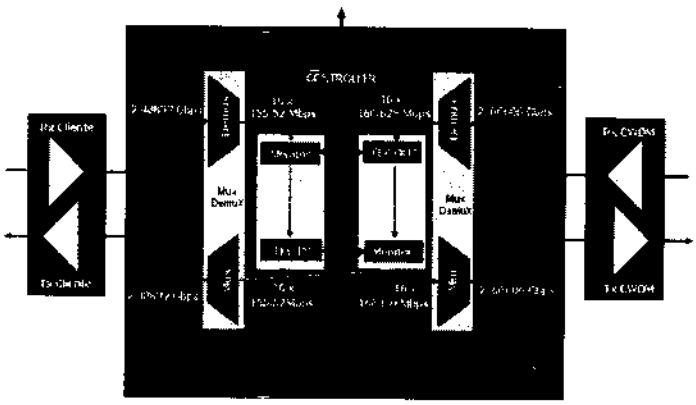


Fig. 3 - Transponder block diagram.

The present technology commercially available and the physical dimensions of the devices had influenced in the final architecture decision, see figure 3.

In this application the Client frames are STM-16 without FEC, with bit-rate equal to 2.48832 Gbps. The CWDM frames are STM-16 with FEC using the Red-Solomon code RS(239,255), with bit-rate equal to 2.66606 Gbps.

The main blocks and corresponding bit-rates are:

- Client transceiver (2.48832 Gbps);
- Line transceiver (2.66606 Gbps);
- Multiplexing-Demultiplexing stages (2.48832 Gbps to 155.52 Mbps, and 2.66606 Gbps to 166.628 Mbps, and vice-versa);
- Monitoring stages (155.52 Mbps and 166.628 Mbps);
- Controller;

The Client and Line interfaces have transceivers to make the electro-optic and optic-electric conversions.

From the Client to the Transponder the received optical data in the client receiver (Rx Client), after being converted to electrical format, is demultiplexed to 16 STM-4 frames in the Mux/Demux block. The bit-rate is equal to 155.52 Mbps. The Encoder/Decoder block introduces FEC in these frames and then they are multiplexed to STM-16 frames with FEC, at 166.628 Mbps bit-rate. The data is then converted to a CWDM optical signal and inject in the line (Tx CWDM) at 2.66606 Gbps.

In the LR interface the optical data is received at 2.66606 Gbps and it is demultiplexed to STM-4 frames with FEC having a bit-rate equal to 166.628 Mbps. After the demultiplexing stage the frames are monitored and the FEC is removed, allowing the multiplexing to STM-16 frames without FEC. These STM-16 frames are converted to optical non-return-to-zero (NRZ) signals and injected in the Client interface (Tx Client) at 2.48832 Gbps.

Client Transceiver

This device has a module that implements the electro-optic conversion at 2.48832 Gbps, being capable to control the average optical power.

The module that implements the optic-electric conversion has a PIN photodiode as the photodetector element being capable to recover the clock from the regenerated data at 2.48832 Gbps bit-rate.

CWDM Transceiver

This block is more complex than the client transceiver and required a separated project for the receiver and transmitter elements.

The optical fiber links in the CWDM network can be very long requiring high sensitivity photodetectors. An avalanche photodetector (APD) was chosen having a sensibility of -33dBm (0.5μW). One problem raised at this stage, the APD requires a high-voltage source power supply. With careful electronic design it was implemented a stable high-voltage source able to generate a voltage range between 30V and 90V [5].

The transmitter was implemented with a CWDM laser and a laser driver.

Multiplexing / Demultiplexing Blocks

This block was implemented with a Mux/Demux device that is able to work with different bit-rates including the previously stated ones. These devices could work with the following signals STM-1, STM-4, STM-16 and GBE (Gigabit Ethernet) with or without FEC.

Monitoring Block

This block is one of the most important in this project, being responsible for the removal of FEC and improvement of the BER. It is also able to work with different bit-rates and different FEC schemes. Table 2 shows the possible FEC modes of the block monitoring device.

Table. 2 – FEC possible modes for the block monitoring.

Error Correcting Capability	RS Code	Bit-rate Multiplying factor
8 bytes per 255-byte block	255/238	15/14
7 bytes per 255-byte block	255/240	17/16
6 bytes per 255-byte block	255/242	255/242
5 bytes per 255-byte block	255/244	255/244
4 bytes per 255-byte block	255/246	85/82
3 bytes per 255-byte block	255/248	255/248

The chosen mode was the error correcting capability of 8 bytes per 255-byte block, the highest available in this chipset.

The device is also able to access specific bytes of the SDH header frame, monitor them and give information to user about their state. It has also the capability of changing the headers when operating in conjugation with a microprocessor or a Field Programmable Gate Array (FPGA).

Controller Block

This block is responsible for the control and monitoring of all the other devices of the transponder.

In this implementation a FPGA was used as the controller of the devices.

It controls:

- The laser driver
- The APD source;
- Selects the Mux/Demux and clock and data recovery bit-rates;
- Selects the FEC mode;
- Monitor registers from the FEC monitoring chipset;
- Optionally changes specific bytes of the SDH frames;
- Initializes the devices;
- Output to the user the state of the two chipsets present in the transponder.

V. IMPLEMENTATION

The layout of the printed circuit board (PCB) had in consideration several factors:

- High-speed transmission lines are needed requiring microwave techniques to implement the impedance adaptation;
- Separated grounds for the different devices are crucial to minimize noise, cross-talk and current rings;
- Electromagnetic interference minimizing required isolation through the introduction of plane layers and not mixing the different logic family devices in the same area;
- Physical dimensions of the components have taken an important role in their respective placement.

VI. ADAPTATION TO DWDM

The adaptation of the presented transponder to DWDM systems deserve some attention and needs to take into account some important aspects.

A - Optical Spectrum

It can be seen from Fig. 1 that the optical spacing of CWDM channels is much broader when compared with DWDM channels. This requires a better spectral purity and stability of the laser source.

B - Power Supply

Power supply requirements for a DWDM laser are much higher compared to a CWDM laser. This difference of power consumption is manly because a DWDM requires a very thigh temperature control. Temperature control in these devices is made through the aid of a temperature sensor and an integrated cooler in the same package. In the DWDM grid the 100GHz spacing between channels is equivalent to 0.8nm and it is required a temperature drift less than 0.08nm/ºC. The CWDM 20nm spacing between channels is large enough to allow the central wavelength of the laser to drift without causing any interference in neighboring channels, even under modulation. It is important to have in mind that a DFB (distributed feedback) laser with a cooling circuit requires 5W of power supply and without cooling only demands 0.5W.

C - Hardware Complexity

The laser drive circuit becomes simpler in the CWDM version because it is not necessary an electronic circuit to control the temperature.

D - Physical Dimensions

The physical dimensions of a CWDM laser are smaller compared to a DWDM laser as they do not include the cooler element (a Peltier device). Eventually it can include a thermistor to obtain quantitative information about the temperature and a photodetector for average optical power monitoring.

It is important to refer that optical power monitoring allows the control of the laser bias point and counteract semiconductor aging.

VII. CONCLUSIONS AND DISCUSSION

The main goals of developing a transponder for CWDM systems were achieved. Expertise in different areas as optical communications, high-speed digital and analog electronic was required.

One of the design requirements was the implementation of the transponder in only one PCB card making it attractive for commercial purposes.

To finalize a DWDM transponder has a more complex, costly and bigger implementation compared to a CWDM one.

VIII. ACKNOWLEDGMENTS

This work was partially supported by the Portuguese Minister of Economic, through the POE project, SIRAC, and by the Portuguese Scientific Foundation, FCT, through the DOPTNET project (POSI/CPS/42073/2001), FEDER and POSI.

REFERENCES

[1] ITU-T G.694.2 (2002) "Spectral grids for WDM applications: CWDM wavelength grid"

[2] ITU-T G.694.1 (2002) "Spectral grids for WDM applications: DWDM frequency grid"

[3] *Optical Networks: A Practical Perspective*, Rajiv Ramaswami & Kumar Sivarajan, Morgan Kaufmann Publishers, 1998

[4] ITU-T G.975 (2000) "Forward error correction for submarine systems"

[5] MAXIM, APP 1085, "Choosing The MAX1932 External Inductor, Diode, Current Sense Resistor, and Output Filter Capacitors for Worst Case Conditions", 2002 http://www.maxim-ic.com/appnotes.cfm/appnote_number/1805/ln/en

V. Barbosa is with Universidade de Aveiro and Instituto de Telecomunicações – pólo Aveiro. Licenciatura Degree at FEUP, 2001, in Engenharia Electrotécnica e de Computadores – branch Telecomunicações, Electrónica e Computadores.

A. N. Pinto is with Universidade de Aveiro and Instituto de Telecomunicações – pólo de Aveiro. He is Professor Assistant in Departamento de Engenharia de Telecomunicações at Universidade de Aveiro and Senior Researcher. PhD at Universidade de Aveiro in 1999.

C. Lemos is with PT Inovação.

R. M. Diz is with PT Inovação.

Nefi J. Carvalho is with Universidade de Aveiro and Instituto de Telecomunicações – pólo Aveiro. Licenciatura Degree at University of Aveiro, 2001, in Engenharia Electrónica e de Telecomunicações.

Bragg Gratings with Enhanced Temperature Sustainability Written in a High-Germanium-Doped Fiber

M. J. N. Lima, R. N. Nogueira, J. C. C. Silva, A. L. J. Teixeira, P. S. B. André, J. R. F. da Rocha, H. J. Kalinowski, J. L. Pinto

Resumo - **Nesta contribuição apresentam-se redes de Bragg com elevada estabilidade às variações de temperatura, resultantes de regimes de exposição do tipo IIA e de um novo tipo especial, comparando-se os resultados de estabilidade com os de outras redes, obtidas a partir de diferentes mecanismos de crescimento, nomeadamente de tipos I e IA. Espera-se que as redes de tipo IIA, e a referida como especial, conduzam a uma melhoria da fiabilidade de aplicações sensoriais a temperaturas elevadas, bem como de dispositivos para comunicações ópticas baseados em redes de Bragg.**

Abstract - **In this contribution we report the achievement of fiber Bragg gratings exhibiting high temperature sustainability, resultant from type IIA and a new abnormal exposition regimes, and compare their thermal stability with other gratings obtained from different growth mechanisms, namely type I and IA. It is expected that type IIA and the designated as abnormal gratings improve the reliability of high-temperature grating-sensing applications and grating-based components for optical communication systems.**

I. INTRODUCTION

Fiber Bragg gratings (FBGs) are widely used for numerous applications in optical communications, and as sensing elements of various physical quantities. For a number of such applications, FBGs should withstand heating to high temperatures, and therefore the optimization of gratings' thermal stability has received considerable attention during the last few years [1]-[3].

The thermal stability of gratings depends on numerous factors such as the doping material, the pre- or pos-fabrication treatment used, and the exposition regime considered. Until recently, FBGs have been found to exhibit three distinct structural types according to their different growth mechanisms, the most commonly observed type I, and two other regimes, type II - arising from physical damage - and type IIA - most often observed in non-hydrogenated high Germanium (Ge) doped optical fibers [4]. Recently, Liu *et al.* have reported a new regime they termed type IA [5], obtained in hydrogenated germanosilicate fibers, which has been further studied [6].

In this letter we report a new FBG with high temperature sustainability, obtained from an abnormal growth mechanism (different from the presented), and compare its thermal stability with other FBGs, resultant from the three referred regimes.

II. FBGS FABRICATION

The FBGs used in this investigation were written in a commercial high Ge doped optical fibre (Fibercore SM1500 4.2/125), hydrogenated or not, by exposing it to an interference pattern originated from a UV beam through a phase-mask.

The type IIA FBG_1 was written using KrF excimer laser (248 nm), with a fluence of 300 mJ/cm^2 per pulse and a pulse repetition rate of 30 Hz, after 100 minutes irradiation. Type I FBG_2 was obtained by stopping a potential type IIA regime, after only 7.5 minutes (reflectivity ~0.7). To obtain the type IA FBG_3 first we have hydrogenated the fiber, keeping it under high-pressure hydrogen atmosphere (100 bars) during four weeks. After the UV exposure - continuous wave from a Ar^+ laser (244 nm) - the hydrogenated grating without coating was annealed for 4 h at 80 °C, to remove unreacted hydrogen. FBG_4 was written considering the same UV source conditions as FBG_3, but using the non-hydrogenated fiber instead.

In Fig. 1 [7] we present the measured maximum reflectivity (R_{max}) and mean refractive index perturbation (Δn_{mean}) - obtained from the Bragg wavelength shift - as a function of the irradiation time, for the referred FBGs.

Observing Fig. 1 we notice that, as regards the evolution of FBG_4 perturbation (mean value and amplitude - related to R_{max}), it cannot be associated to any of the indicated regimes. Clearly it is not a type I regime. The R_{max} variation could suggest the growth of a type IIA grating, with UV exposure resulting in the grating erasure, followed by a new spectral formation. However, we do not observe the typical Bragg resonance blue shifting (Δn_{mean} decrease), associated to that new formation (instead, Δn_{mean} continuously increases with UV exposure) [4]. On the other hand, it is not observed the characteristically large red shift of the Bragg wavelength that accompanies type IA grating formation [5][6]. In order to conclude about this abnormal regime, in the next section we will study its thermal behaviour, comparing it with the other mechanisms.

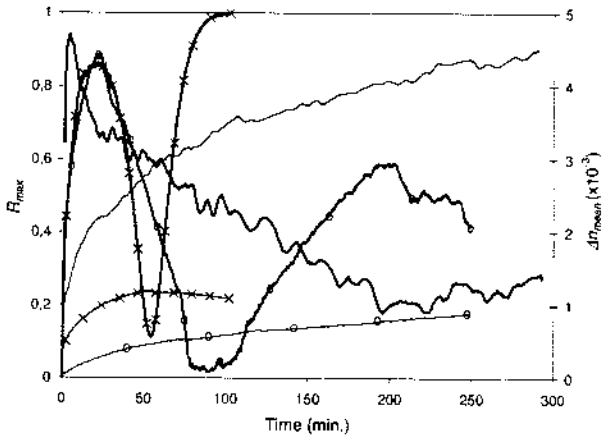


Fig. 1. Measured parameters during the gratings ($FBG_{1,2}$ -x-, FBG_3 —, FBG_4 -o-) formation: R_{max} (heavy lines) and Δn_{mean} (hair lines) [7].

III. EXPERIMENTAL RESULTS

Following gratings fabrication, we now evaluate the temperature performance (stability) of the four described FBGs.

To observe their thermal decay characteristics, the gratings were placed in a tubular stove and the temperature was increased till 800 °C, with step of 100 °C. The measured reflectivities are presented in Fig. 2 [7]. Considering as a limit a 10% decrease on the reflectivity (relatively to the initial one), the acceptable temperatures are approximately 550 °C, 150 °C, 200 °C and 350 °C, respectively for FBG_1 (type IIA), FBG_2 (type I), FBG_3 (type IA) and FBG_4 (abnormal).

The highest/lowest temperatures are sustained considering respectively type IIA/type I FBGs, following the results presented in the literature [4]. Type IA FBG_3 presents a slightly higher temperature limit than type I FBG_2. On the other hand, the grating obtained from the abnormal regime (FBG_4) has been shown to sustain considerably higher temperatures than both of them (type I and IA), although lower than the type IIA one. Additionally, when performing the reflectivity measurements, we observed that the variations on the reflection spectrum bandwidth as the temperature increased were much more evident for type I and IA FBGs. For the other FBGs (abnormal and type IIA), those variations are negligible.

The observed temperature dependencies for the different FBGs, may indicate several potential applications for them, within optical communications and sensing.

Indeed, type IIA FBGs can sustain high temperatures, and therefore may find application in sensing heads for industrial environments where large temperature variations occur [8]. FBGs like the obtained from the regime we considered abnormal, may also be used in such applications, although for a lower temperature range.

Also, this observed high stability (for both type IIA and abnormal FBGs) may result in optimized thermo-tunable

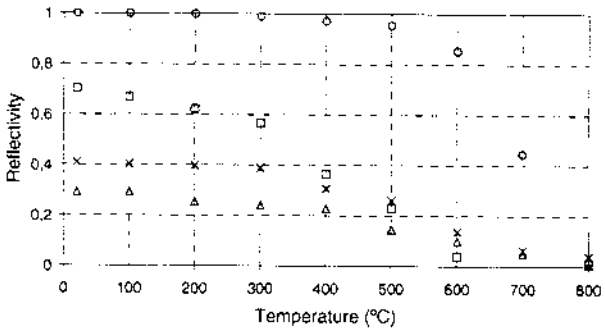


Fig. 2. The evolution of the reflectivity for the different gratings, FBG_1 (o), FBG_2 (€), FBG_3 (Δ), FBG_4 (×) as the temperature is increased [7].

(central wavelength and/or dispersion) optical filters for lightwave systems [9].

IV. CONCLUSIONS

We have compared the thermal stability of FBGs obtained from distinct exposition regimes, namely type I, IA, IIA and a new regime we designated as abnormal, written in a high-Ge-doped fiber. The highest temperature stability is achieved for the type IIA FBG. The grating obtained from the abnormal regime presents a lower temperature limit, although higher than the achieved for type I and IA FBGs. Therefore, it is envisaged that type IIA and the referred as abnormal FBGs improve the reliability of high-temperature grating-sensing applications and FBG components for optical communication systems.

ACKNOWLEDGEMENTS

We are grateful to A. G. Simpson from the Photonics Research Group, Aston University, Birmingham, UK, for helpful discussions. Funding through FCT (Project SFRH/BD/7043/2001), GRICES from Portugal and Fundação Auracária and CAPES from Brazil are also acknowledged.

REFERENCES

[1] O. V. Butov, K. M. Golant, I. V. Nikolin, "Ultra-thermo-resistant Bragg gratings written in nitrogen-doped silica fibres", Electronics Letters, vol. 38, pp. 523-525, 2002.

[2] G. Brambilla, H. Rutt, "Fiber Bragg gratings with ultra-high temperature-stability", in Optical Fiber Communication Conf. 2002, paper ThGG35, pp. 660-662.

[3] Y. Shen, J. Xia, T. Sun, K. T. V. Grattan, "Photosensitive Indium-doped Germano-silica fiber for strong FBGs with high temperature sustainability", IEEE Photonics Technology Letters, vol. 16, pp. 1319-1321, 2004.

[4] A. Othonos, K. Kalli, Fiber Bragg gratings – fundamentals and applications in telecommunications and Sensing, Artech House, 1999.

[5] Y. Liu, J. A. R. Williams, L. Zhang, I. Bennion, "Abnormal spectral evolution of fiber Bragg gratings in hydrogenated fibers", Optics Letters, vol. 27, pp. 586-588, 2002.

[6] A. G. Simpson, K. Kalli, K. Zhou, L. Zhang, I. Bennion, "Formation of type IA fibre Bragg gratings in germanosilicate optical fibre", Electronics Letters, vol. 40, pp. 163-164, 2004.

[7] M. J. N. Lima, R. N. Nogueira, J. C. C. Silva, A. L. J. Teixeira, P. S. B. André, J. R. F. da Rocha, H. J. Kalinowski, "Comparison of the temperature dependence of different types of Bragg gratings", submitted to Optical and Quantum Electronics.

[8] O. Frazão, M. J. N. Lima, J. L. Santos, "Simultaneous measurement of strain and temperature using type I and type IIA fibre Bragg gratings", Journal of Optics A: Pure and Applied Optics, vol. 5, pp. 183-185, 2003.

[9] B. Dabarsyah, C. S. Goh, S. K. Khijwania, S. Y. Set, K. Katoh, K. Kikuchi, "Adjustable dispersion-compensation devives with wavelength tenability based on enhanced thermal chirping of fiber Bragg gratings", IEEE Photonics Technology Letters, vol. 15, pp. 416-418, 2003.

Caracterização de um Amplificador a Guia de onda Dopada com Érbio

S. L. Stevan Jr., F. Couto, P. André, P. Monteiro, A. L. J. Teixeira

Resumo **- Neste trabalho reportamos a caracterização e teste de um amplificador de guia de onda dopado com Érbio (EDWA). As curvas de ganho e figura de ruído em função da potência de entrada e em função do comprimento de onda foram obtidas por um método por nós desenvolvido. As características de largura de banda de ganho saturado e figura de ruído foram analisadas e caracterizadas.**

I INTRODUÇÃO

Os últimos avanços na óptica integrada, sobretudo na área de dispositivos activos onde tem sido aprimoradas as questões do tamanho físico e custo, fazem do Amplificador de Guia de Onda Dopada com Érbio (*Erbium Doped Waveguide Amplifier* – EDWA) um dos focos de grande investimento actual na indústria de Telecomunicações [1-3].

Como principais vantagens, o EDWA apresenta uma baixa figura de ruído (*Noise Figure* - NF), independência do ganho em relação à polarização, reduzida interferência entre canais em transmissões WDM (*Wavelength Division Multiplexing*) e possibilidades de integração com outros dispositivos passivos e/ou activos no mesmo circuito integrado, além do reduzido tamanho e do baixo custo. No entanto, o principal problema de desenvolvimento e produção dos EDWA, relaciona-se com a necessidade de altas concentrações de iões de Érbio.

Os maiores desafios em transformar um guia de onda dopado com Érbio num amplificador óptico são diminuir as perdas na propagação e controlar os efeitos devido a alta concentração de iões de Érbio [4].

II. CARACTERÍSTICAS TÉCNICAS

O quadro da figura 1 apresenta um comparativo entre as características dos EDWAs, dos amplificadores de fibra dopada com Érbio (EDFA) e dos amplificadores ópticos de semicondutor (SOA) [1].

Através dos dados da figura 1, verifica-se que o EDWA apresenta um ganho tipicamente menor que os outros dois tipos de amplificador, no entanto, sobressaem-se os efeitos desvantajosos, como a dependência com a polarização do sinal e a interferência entre canais. Com as vantagens de terem um baixo custo de fabricação e um tamanho reduzido.

Actualmente, é esperado que um EDWA apresente um ganho de 2 a 3 dB/cm [1].

	EDFA	EDWA	SOA
Ganho de pequeno sinal	30dB	15dB	20dB
Figura de ruído	4 dB	4.5 dB	6-7 dB
Dependência com a polarização	0.5dB	<0.1dB	1.0dB
Diafonia entre canais	Não	Não	Alto
Controlo intrínseco da polarização	Não	Sim	Sim
Tamanho físico médio	dezenas de metros	milímetros	centímetros
Custo	Alto	Baixo	Baixo

Fig. 1 – Tabela comparativa de amplificadores ópticos

III. SETUP E VERIFICAÇÃO EXPERIMENTAL

Foi analisado experimentalmente um EDWA comercial disponível no laboratório, com aproximadamente 7cm de comprimento, potência de bombagem de 200mW a 980nm

Para essa caracterização utilizou-se a configuração básica apresentada na figura 2, baseada numa fonte de sinal (laser), variável em comprimento de onda e amplitude e num analisador de espectros ópticos (OSA).

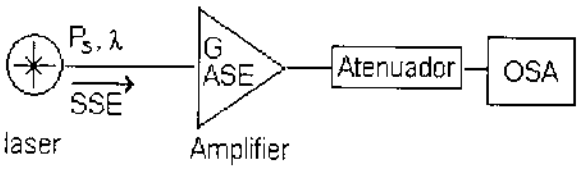


Fig. 2. Configuração básica para a caracterização de um amplificador.

A caracterização do ganho do amplificador óptico é feita com base na relação directa entre as potências do sinal de saída, P_{S_out}, e do sinal de entrada, P_{S_in}, conforme indicado na equação 1.

$$G = \frac{P_{S\_out}}{P_{S\_in}} \qquad (1)$$

A figura de ruído (NF) é calculada através da relação entre a relação sinal-ruído (SNR) à entrada e à saída do amplificador. Pode-se quantificar a NF através da expressão indicada na equação (2), onde se consideraram os seguintes parâmetros [5]:

ν : frequência óptica do sinal;

P_{SSE} : densidade espectral de potência da emissão espontânea do laser;

P_{ASE} : densidade espectral de potência do ruído à saída do amplificador de fibra dopada, incluindo o ruído do laser e contabilizando ambas as polarizações

$$NF = \frac{P_{ASE}}{G.h.\nu.B_o} + \frac{1}{G} - \frac{P_{SSE}}{h.\nu.B_o} \qquad (2)$$

O último termo da equação anterior refere-se à contribuição da potência de ruído do laser amplificado.

IV. RESULTADOS AND DISCUSSÕES

A figura 3 mostra a curva do ganho e da figura de ruído do EDWA em função do comprimento de onda, para uma potência do sinal de entrada de –35dBm. Verifica-se que este amplificador apresenta uma curva de ganho coerente com a curva característica dos iões de Érbio, com um pico a 1535 nm e um largo espectro quase plano até 1560 nm.

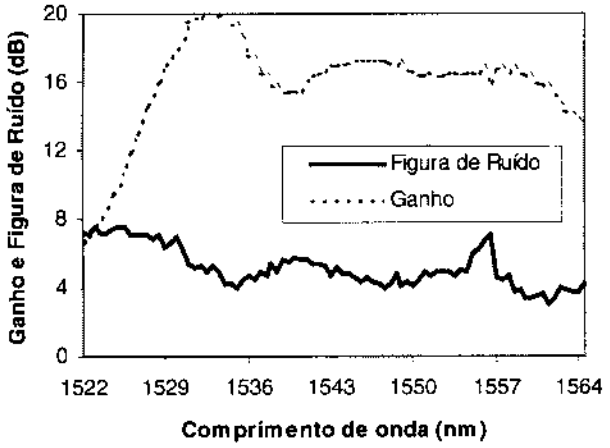


Fig. 3 – Espectros de ganho e de figura de ruído do EDWA, para uma potência de entrada de –30dBm.

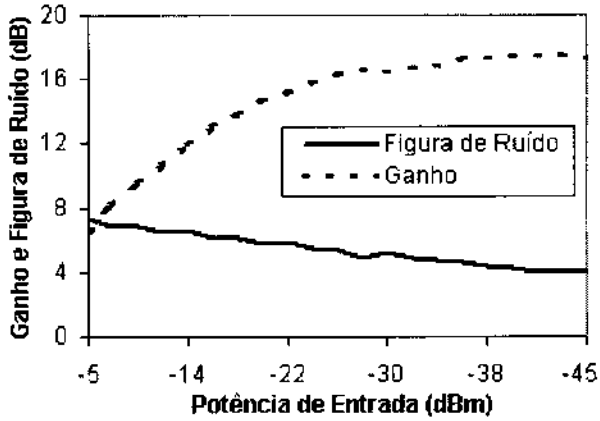


Fig. 4 - Evolução do ganho e da figura de ruído em função da potência de entrada para um comprimento de onda de sinal de 1545 nm.

Outra análise realizada foi a evolução do ganho em função da potência óptica de entrada, para um comprimento de onda de 1545nm. Esta análise tem como objectivo a determinação da potência de saturação e os resultados encontram-se na figura 4. Onde se verifica que o ganho deste amplificador satura quando o sinal de entrada tem uma potência de –19,5 dBm, considerando-se uma potência de bomba de 200 mW. Nesta figura também é apresentado o comportamento da figura de ruído em função da potência de entrada, onde se verifica um NF maior para a região de ganho saturado.

V. CONCLUSÃO

Caracterizou-se um EDWA nas suas características mais relevantes. Verificou-se que apresenta na sua região mais plana da curva de ganho um valor de 15 dB e uma figura de ruído de 4.5 dB

Este tipo de amplificador apresenta dimensões muito reduzidas e, como tal, apresenta-se como um grande candidato a integrar os sistemas futuros de regeneração e processamento óptico mais complexo, como *routers* .

AGRADECIMENTOS

Os autores gostariam de agradecer aos projectos WidCom, ALBAN, SIRAC e à SIEMENS AG pela suas comparticipação na obtenção dos resultados.

REFERÊNCIAS

[1] D. Barbier, "Erbium-doped Waveguide Amplifiers promote optical-networking evolution", Lightwave, 2001

[2] P.G. Kik and A. Polman, "Erbium doped optical waveguide amplifiers on silicon", FOM Institute for Atomic and Molecular Physics, April, 1998

[3] G. C. Righini, S. Pelli, M. Ferrari, M. Brenci, "ERBIUM-ACTIVATED SILICATE WAVEGUIDES AND AMPLIFIERS", 8th Microoptics Conference (MOC '01), Osaka, Japan, October 24-26, 2001

[4] J. L. Philipsen et al., "Erbium-Doped Waveguide Amplifier (EDWA): Technology and Components", 2003

[5] D. Derickson, *"Fiber Optic Test and Measurement"*, Prentice Hall, Upper Saddle River, NJ, 1998.

Asynchronous Capacities of the Sequential Symbol Synchronizer

António D. Reis[1], José F. Rocha, Atilio S. Gameiro, José P. Carvalho[1]

[1]Dep. de Fisica, Universidade da Beira Interior Covilhã, 6200 Covilhã, Portugal

Resumo **- Neste artigo apresentamos quatro tipos de sincronizadores de símbolo normais, nomeadamente o analógico, o híbrido, o combinacional e o sequencial.**

O sincronizador sequencial síncrono tem capacidades que permitem obter a correspondente versão assíncrona. Assim o flip flop é substituido por lógica com realimentação. Então todos os sincronizadores usam somente portas lógicas.

O principal objectivo é desenvolver o sincronizador sequencial assíncrono e depois estudar e comparar a performance das duas vesrsões.

Palavras chave: **Sincronismo em Comunicações Digitais**

Abstract **- In this paper we present four types of normal symbol synchronizers, namely the analog, the hybrid, the combinational and the sequential.**

The synchronous sequential synchronizer has capacities that permits to obtain the correspondent asynchronous version. So the flip flop is substituted by logic with feedback. Then all the synchronizers use only logic gates.

The main objective is to develop the synchronous sequential synchronizer and then to study and compare the performance of the two versions.

Key words: **Synchronism in Digital Communications**

I. INTRODUCTION

Before, we studied four synchronizer types, namely the analog, the hybrid, the combinational and the sequential.

The synchronizers analog, hybrid, and combinational are implemented with logic gates but the sequential one is implemented with logic gates and flip flops.

As, at low frequencies, the synchronous sequential synchronizer seems disadvantageous relatively to the others, then we go develop the correspondent asynchronous version that are implemented only with logic gates with feedback. So all the synchronizers are implemented only with logic gates.

Fig.1 shows the general block diagram of the closed loop synchronizers.

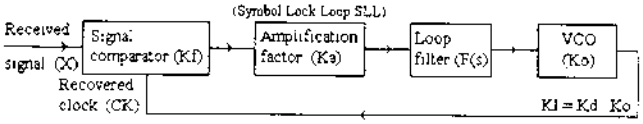


Fig.1 General closed loop synchronizer

The closed loop synchronizer is composed by the signal comparator, amplification factor, filter and VCO.

The great difference between the various synchronizers is inside the signal comparator. The others blocks are identical.

Firstly, we present the four types of synchronizers (analog, hybrid, combinational and sequential).

Next, we go present the correspondent asynchronous sequential version.

After, we show the design, tests and results with some comparisons [1, 6, 7, 8].

Finally, we present the main conclusions.

II. CLOSED SYMBOL SYNCHRONIZER TYPES

The four types of closed loop symbol synchronizers are got inserting the appropriate signal comparator [3, 4].

A. Analog synchronizer

Fig.2 shows the analog synchronizer, in which, the signal comparator is composed by three multipliers, two T/2 delays and inverter.

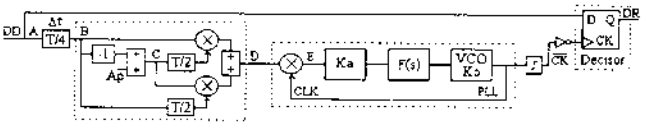


Fig.2 Analog symbol synchronizer

The main input signal and the VCO output, at the signal comparator inputs, are both analog.

B. Hybrid synchronizer

Fig.3 shows the hybrid synchronizer, in which, the signal comparator is composed by a triple switch, an exor and two delays.

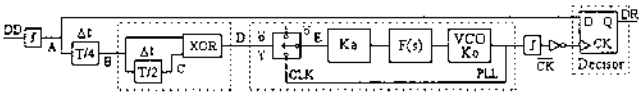


Fig.3 Hybrid symbol synchronizer

The main input signal is already digital but the VCO output is still analog.

C. Combinational synchronizer

Fig.4 shows the combinational synchronizer, in which, the signal comparator is composed by a Mux (two ANDs) with adder, an exor and two delays (T/4, T/2).

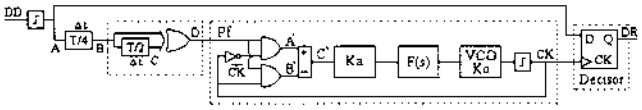


Fig.4 Combinational symbol synchronizer

The two inputs (main input and VCO output) of the signal comparator are both digital. The output is only function of the entries.

D. Sequential synchronizer

Fig.5 shows the sequential synchronizer, in which, the signal comparator is composed by one flip flop with exor and a delay with exor, followed of an adder [2].

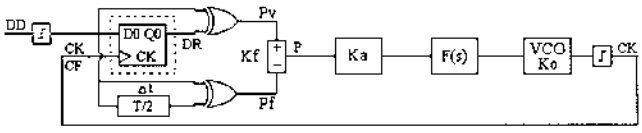


Fig.5 Synchronous sequential symbol synchronizer

The two inputs (main input and VCO output) of the signal comparator are both digital. However now, the output is simultaneously function of the two entries and also of the signal comparator state (memory).

III. ASYNCHRONOUS SEQUENTIAL VERSION

The difference between the synchronous and asynchronous versions can be evidenced by the CS dush outlined rectangle of Fig.6. The CF circuit determines the version of the symbol synchronizer that can be synchronous or asynchronous.

Fig.6 shows the synchronous version when the CF dush outlined is a flip flop as shown in Fig.5 [4].

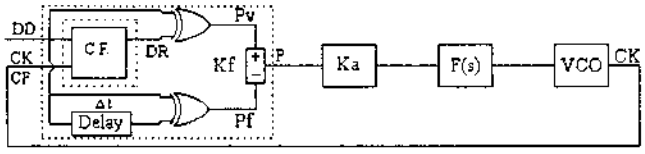


Fig.6 Synchronous version of the sequential synchronizer

The synchronous version is easier to implement, anyway we must obtain the asynchronous version, to make the performance comparisons.

Fig.7 shows the CF circuit implemented with feedback logic. So if this circuit corresponds to the CF circuit of Fig.6 we have the asynchronous sequential synchronizer [4].

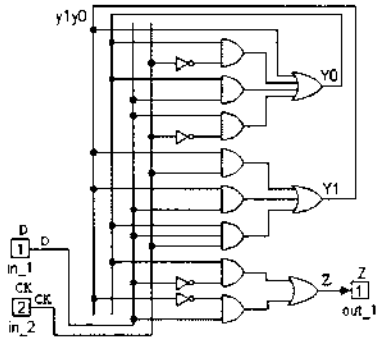


Fig.7 Asynchronous version of the sequential synchronizer

This two versions will be implemented and tested to see its jitter-noise performance.

IV. DESIGN, TESTS AND RESULTS

A. Test setup

Fig.8 shows the setup that we used to get the jitter-noise curves of the various synchronizers [5].

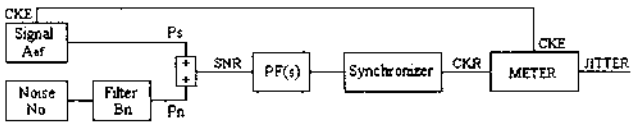


Fig.8 Block diagram of the test setup

The signal to noise ratio SNR is given by Ps/Pn, where Ps is the signal power and Pn is the noise power. They are defined as $Ps=A_{ef}^2$ and $Pn=No.Bn=2\sigma_n^2\Delta\tau$ Bn. A_{ef} is the RMS amplitude, Bn is the external noise bandwidth, No is the noise power spectral density, σ_n is the noise standard deviation and $\Delta\tau$ is the sampling period (inverse of samples per unit time).

The prefilter is not used here, but can be useful in system with high noise quantities (PF(s)=1).

B. Jitter measurer

Fig.9 shows the jitter measurer (METTER) that consists of a RS flip-flop which detects the variable phase of the recovered clock (VCO) relatively to the fixed phase of the emitter clock.

This relative phase variation is the recovered clock jitter.

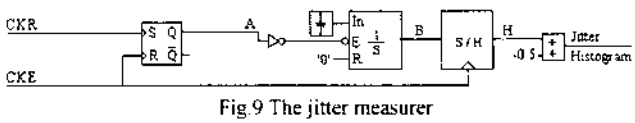


Fig.9 The jitter measurer

The others blocks convert this phase variation into an amplitude variation, which is the jitter histogram.

Fig.10 shows the waveforms that illustrate the operation mode of the jitter measurer.

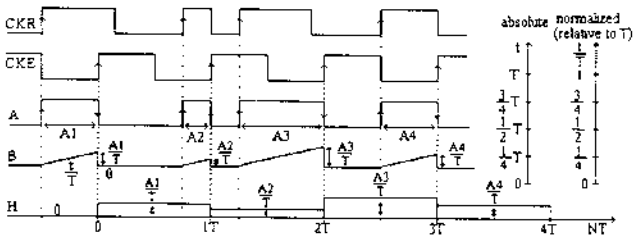


Fig.10 Waveforms at the jitter measurer

The jitter histogram is then sampled and processed by an appropriated program giving the average m, jitter variance in squared radians σ_n^2, jitter standard deviation in unit intervals root mean squared UIRMS and jitter standard deviation in unit intervals peak to peak UIPP.

We have used also others jitter measurers with similar results.

C. Loop parameters design

To establish guaranteed comparisons it is necessary to test all the synchronizers in equal conditions.

We use a normalized transmission rate tx=1baud (fo=1Hz) what facilitates the analyses and allows one more easy extrapolation for other rhythms of transmission. We use an equivalent external noise bandwidth Bn=5Hz for all SLL. For the closed loop symbol synchronizers SLL, we use a loop noise bandwidth Bl=0.02Hz.

For analog SLL, the relation between signal to noise ratio SNR and jitter variance σn is $SNR=Aef^2/No.Bn=A_{ef}^2/(2\sigma n^2.\Delta\tau.Bn)=(0.5)^2/(2\sigma n^2*10^{-3}*5)=25/\sigma n^2$. This relation is more complicated for the others symbol synchronizers.

We will now present the loop parameters design for the various PLLs considering the first (1st) and the second order loop (2nd).

- 1st order loop:

In the 1st order loop, the filter F(s)=0.5Hz eliminates only the high frequency, but maintain the loop characteristics. This cutoff frequency F(s)=0.5Hz is 25 times higher than Bl=0.02Hz. Then the transfer function of the 1st order is

$$H(s)=\frac{G(s)}{1+G(s)}=\frac{KdKo}{s+KdKo} \qquad (1)$$

the loop noise bandwidth for the SLLs is

$$Bl=\frac{KdKo}{4}=Ka\frac{KfKo}{4}=0.02\text{Hz} \qquad (2)$$

so for the analog SLL with Km=1, A=1/2, B=0.45 we have

$$Ka\frac{KmABKo}{4}=0.02Hz \Rightarrow Ka=0.08\frac{2.2}{\pi} \qquad (3)$$

for the hybrid SLL, with Km=1, A=1/2 and B=0.45 we have

$$Ka\frac{KmABKo}{4}=0.02Hz \Rightarrow Ka=0.08\frac{2.2}{\pi} \qquad (4)$$

for the combinational SLL (Kf=1/π) we have

$$Ka\frac{KfKo}{4}=Ka\frac{(1/\pi)2\pi}{4}=0.02Hz \Rightarrow Ka=0.04 \qquad (5)$$

and for the sequential SLL (Kf=1/2π) we have

$$Ka\frac{KfKo}{4}=Ka\frac{(1/2\pi)2\pi}{4}=0.02Hz \Rightarrow Ka=0.08 \qquad (6)$$

This formulas are useful in synchronizers design

- 2nd order loop:

The transfer function with F(s) = $F(s)=\frac{1+sT2}{sT1}$ is

$$H(s)=\frac{sKdKo(T2/T1)+KdKo/T1}{s+sKdKo(T2/T1)+KdKo/T1} \qquad (7)$$

$$=\frac{sA+B}{s^2+s2\xi Wn+Wn^2} \qquad (8)$$

and the loop noise bandwidth is

$$Bl=\frac{\xi Wn}{2}\left(1+\frac{1}{4\xi^2}\right) \qquad (9)$$

Taking (ξ=1 and Bl=0.02) and solving the above equations we obtain for F(s)

$$F(s)=\frac{1+s63}{s977} \qquad (10)$$

so for the analog SLL we have

$$Kd=KaKf=Ka(1)(1/2)(1/2)=\frac{1}{2\pi}\Rightarrow Ka=\frac{2.2}{\pi} \qquad (11)$$

for the hybrid SLL we have

$$Kd=KaKf=Ka(1)(1/2)(1/0.45)=\frac{1}{2\pi}\Rightarrow Ka=\frac{2.2}{\pi} \qquad (12)$$

for the combinational SLL we have

$$Kd=KaKf=\frac{Ka}{\pi}=\frac{1}{2\pi}\Rightarrow Ka=0.5 \qquad (13)$$

and for the sequential SLL we have

$$Kd=KaKf=\frac{Ka}{2\pi}=\frac{1}{2\pi}\Rightarrow Ka=1 \qquad (14)$$

This formulas can be used in others synchronizers.

D. Results

We studied four synchronizers types, namely the analog (an), the hybrid (hib), the combinational (cmb) and the sequential (seq). Then we taked this synchronous sequential synchronizer (sinc) and we create the correspondent asynchronous sequential synchronizer (asinc)

Fig.11 shows the jitter-noise curves of the four synchronizer types (ana, hib, cmb, seq).

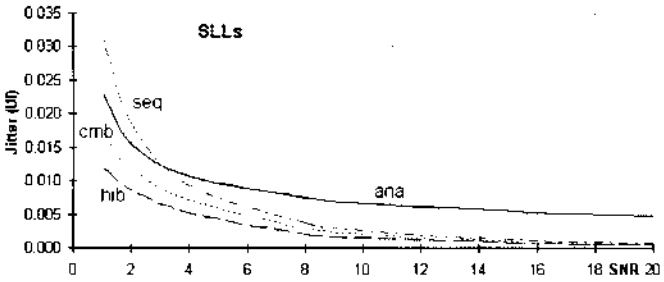


Fig.11 Jitter curves of synchronizers (ana, hib, cmb, seq)

We verify that generically the jitter UI diminishes when the signal to noise ratio SNR increases.

For low SNR (SNR<4), the synchronizer without input limiter (analog) is advantageous over the others with input limiter (hybrid, combinational, sequential)

However for high SNR (SNR>4) the synchronizer without input limiter is disadvantageous over the others with input limiter.

Fig.12 shows the jitter-noise curves of the two synchronizer versions (sinc, asinc).

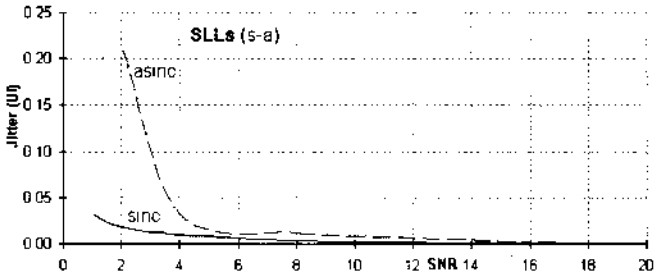


Fig.12 Jitter-noise curves of synchronizers (sinc, asinc)

We verify that, for low SNR (SNR<5) the synchronous synchronizer is advantageous over the asynchronous one.

However, for high SNR (SNR>5), the synchronous and asynchronous synchronizers have identical jitter-noise performance.

V. CONCLUSIONS

We studied four symbol synchronizers, namely the analog (ana), the hybrid (hib), the combinational (cmb) and the sequential (seq).

Then we taked the capacities of the synchronous sequential synchronizer (sinc) and we developed the asynchronous version (asinc).

In the synchronizers (ana, hib, cmb, seq) we verify that for low SNR the synchronizers without input limiter (ana) is advantageous over the others with input limiter (hib, cmb, seq). This is due to the noise spikes that provokes random gate commutations. The worst case is the sequential synchronizer but it can be minimized with a prefilter. However, for high SNR (SNR>4), the synchronizer without input limiter is disadvantageous over the others due to the noise margin of the limiter gates.

In the synchronizers (sinc, asinc), we verify that for low SNR (SNR<5), the synchronous synchronizer is advantageous over the asynchronous one due to its inter states critical runs, that conducts to the error state. However, for high SNR (SNR>5), in which the synchronizers normally operates, the two synchronizers have identical jitter-noise performance. The jitter is only due to the input noise.

Globally, we noted that the jitter UI decreases when the signal to noise rate SNR increases.

ACKNOWLEDGMENTS

The authors are thankful to the program FCT (Foundation for sCience and Technology).

REFERENCES

[1] Werner Rosenkranz, "Phase Locked Loops with limiter phase detectors in the presence of noise", IEEE Transactions on Communications com-30. October 1982.

[2] Charles R. Hogge, "A Self Correcting Clock Recovery Circuit", Journal of Lightwave Technology, Dec. 1985.

[3] A. D. Reis, J. F. Rocha, A. S. Gameiro, J. P. Carvalho, "Manual and Automatic Data Synchronizers", 7th International Conference on Telecommunications p.1076, Acapulco-MX 22-25 May 2000.

[4] A. D. Reis, J. F. Rocha, A. S. Gameiro, J. P. Carvalho, "High Data Rate Synchronizers Operating at Low Speed", The 8th IEEE International Conference on Electronics, Circuits and Systems p.1127, Malta-MT 2-5 September 2001.

[5] A. D. Reis, J. F. Rocha, A. S. Gameiro, J. P. Carvalho "A New Technique to Measure the Jitter", III Conference on Telecommunications p.64, F. Foz-PT 23-24 April 2001.

[6] A. D. Reis, J. F. Rocha, A. S. Gameiro, J. P. Carvalho, "Open Loop Symbol Synchronizers", 9th International Conference on Telecommunications p.286, Beijing-CN 23-26 June 2002.

[7] A. D. Reis, J. F. Rocha, A. S. Gameiro, J. P. Carvalho, "Mixed Loop Symbol Synchronizers", 3th International Symp. on Communication Systems Networks and Digital Signal Processing p.324, Stafford-UK 15-17 July 2002.

[8] A. D. Reis, J. F. Rocha, A. S. Gameiro, J. P. Carvalho "Closed Loop Symbol Synchronizers with Prefilter", IV Conference on Telecommunications p.433, Aveiro-PT 18-20 June 2003.

Desafios Tecnológicos em Comutação Óptica de Pacotes

João Paulo Madaleno, António Teixeira e Susana Sargento

Resumo **- As redes ópticas de comutação de pacotes (OPS – *Optical Packet Switching*) são cada vez mais uma alternativa credível às outras redes ópticas. O OPS promete trazer a flexibilidade e eficiência da Internet para as redes ópticas com taxas de transferência além das disponíveis actualmente. Neste artigo descreve-se o estado da arte das tecnologias inerentes às redes OPS e os aspectos que ainda se encontram em estudo e desenvolvimento.**

Index Terms **— Optical packet switching, optical label swapping, contention resolution**

I INTRODUÇÃO

O tráfego da Internet está a crescer de forma exponencial: o aparecimento de novos serviços e aplicações com características e requisitos de transmissão cada vez mais exigentes, assim como a crescente afluência de novos utilizadores às tecnologias da informação, tem aumentado em grande escala a quantidade de tráfego a fluir na rede Internet. O crescente congestionamento da rede em alturas de maior tráfego e o aumento dos atrasos na transmissão da informação são indicadores de que a infra-estrutura de rede actual não está preparada para suportar tais exigências de crescimento. Deste modo, é de grande necessidade que a próxima geração de redes IP (*Internet Protocol*) apresentem maiores capacidades das suas ligações, melhor desempenho e maiores taxas de transmissão da informação.

Do ponto de vista conceptual, as redes de comutação óptica de pacotes (OPS) são muito semelhantes às redes electrónicas de comutação de pacotes. Estas são caracterizadas pela elevada capacidade de transmissão (*throughput*), fácil adaptação a situações de congestionamento ou de falhas da rede. Actualmente, a grande limitação na implementação de redes OPS é o facto de não haver unidades de armazenamento ópticas (*buffers*) e equipamento óptico capaz de extrair e adicionar com precisão cabeçalhos de controlo ópticos (*Label Swapping*) [1]. Nas redes OPS, os dados são transmitidos em pacotes ópticos (de tamanho fixo) juntamente com a informação de controlo (cabeçalho - *label*), que é extraída e processada em cada nó.

II COMUTAÇÃO ÓPTICA DE PACOTES

Actualmente, há grandes expectativas depositadas sobre as redes OPS relativamente ao futuro das telecomunicações ópticas. Nos últimos anos têm-se realizado alguns projectos nesta área, nomeadamente, os projectos pioneiros OPERA e KEOPS [2]-[3].

Na figura 1 encontra-se um diagrama de blocos de um nó OPS genérico, que consiste de uma unidade de controlo de comutação, uma série de desmultiplexadores e multiplexadores, interfaces de entrada e saída, um comutador, uma unidade de armazenamento (*buffer* óptico que pode ser implementado de diversas formas, por exemplo linha de atraso em fibra – FDLs – *Fiber Delay Lines*) e/ou conversores de comprimento de onda. Os pacotes ópticos que chegam ao nó OPS são desmultiplexados e enviados para o interface de entrada. Os pacotes ópticos de OPS são constituídos pelos dados (incluindo o cabeçalho IP) e por um cabeçalho óptico, que contém a informação necessária para efectuar o encaminhamento destes no domínio óptico. A interface de entrada tem, entre outras funções, as de extrair o cabeçalho óptico e encaminhá-lo para a unidade de controlo de comutação. A unidade de controlo de comutação processa a informação desse cabeçalho, determinando a porta de saída e o comprimento de onda a atribuir ao respectivo pacote, dando instruções ao comutador para efectuar o seu encaminhamento. Para que os dados não sejam perdidos devido à existência de colisões, é necessário recorrer a estratégias que permitem evitar a colisão de pacotes. Algumas das técnicas mais utilizadas são a retenção dos dados por algum tempo no nó OPS ou o efectuar de uma conversão de comprimento de onda. Finalmente, o pacote é direccionado para o interface de saída, onde é adicionado um novo cabeçalho e encaminhado para a porta de saída predeterminada.

III COMUTAÇÃO/ ENCAMINHAMENTO DE PACOTES

O controlo de comutação de pacotes, conversão de comprimento de onda e encaminhamento para a porta de saída, pode ser efectuado electrónica [2]-[3] ou opticamente [4]-[10]. No entanto, o facto de o controlo de comutação de pacotes ser efectuado electronicamente tem limitações devido à disparidade de ritmos entre a transmissão de sinais WDM (*Wavelength Division*

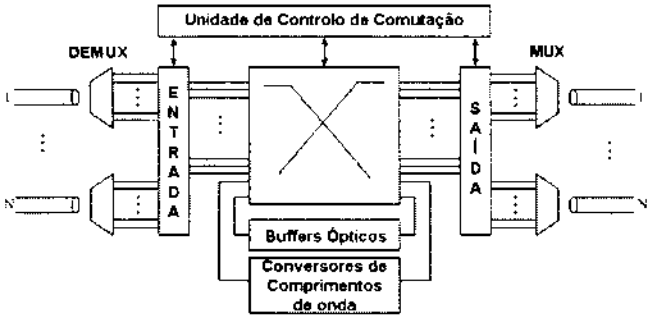


Fig. 1. Diagrama de blocos de um nó OPS genérico [8].

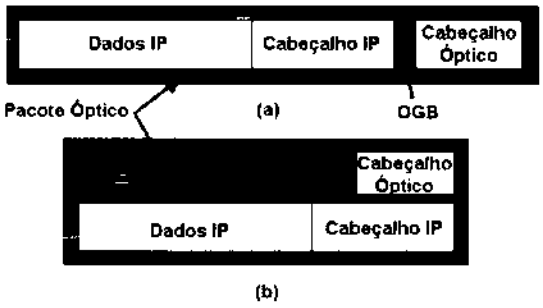


Fig. 2. Formatos de Cabeçalhos Ópticos. a) Em série. b) Multiplexado em subportadora [10].

Multiplexing - mutiplexagem no comprimento de onda) e a capacidade de efectuar o encaminhamento (processamento electrónico), principalmente quando as taxas de transferência aumentam (eg. >40Gbit/s) [10].

Seguidamente, descrevem-se tecnologias e funcionalidades necessárias em nós de redes totalmente ópticas para possibilitar a comutação óptica de pacotes:

Troca de Cabeçalho Óptico (Optical Label Swapping)

Este processo é complexo, pois pressupõe a existência de elementos de processamento integralmente ópticos, elementos estes que actualmente ainda estão em estado embrionário. No entanto, recentemente têm sido amplamente estudadas abordagens do tipo AOLS (*All-Optical Label Swapping*) [4]-[10], onde o cabeçalho de controlo é processado totalmente no domínio óptico. AOLS é uma abordagem promissora para resolver o encaminhamento em redes ópticas. Em cada nó da rede, o cabeçalho é extraído e processado e os pacotes são encaminhados em função da informação existente nestes. De seguida, o cabeçalho é apagado, rescrito e anexado novamente ao respectivo pacote. Note-se que o AOLS deve considerar a utilização de cabeçalhos ópticos para identificar os pacotes IP na camada óptica, ou seja, deve interagir de forma semelhante com sistemas WDM e TDM (*Time Division Multiplexing* - multiplexagem temporal).

Os cabeçalhos ópticos podem ser anexados aos dados de diversas formas: em série (bit-serial) [3] e [16], multiplexados numa sub-portadora (SCM - *SubCarrier Multiplexing*) [4], [9]-[12], utilizando métodos de modulação ortogonal (por exemplo, IM/FSK) [5] ou em paralelo noutro comprimento de onda (*out of band*) [13]. Nos dois primeiros casos a taxa de transferência do cabeçalho é independente e inferior à do pacote, tipicamente 2,5Gbit/s e 10Gbit/s, respectivamente. No caso do cabeçalho óptico em série, este é colocado no mesmo comprimento de onda do pacote à frente do respectivo pacote, separados por uma banda de guarda (OGB – *Optical Band Guard*) que permite remover e reinserir o cabeçalho sem necessidade de recorrer a unidades de armazenamento [10] (figura 2.a). No sistema paralelo, o cabeçalho é multiplexado numa sub-portadora eléctrica ou óptica do pacote IP mantendo por isso o mesmo comprimento de onda (figura 2.b). Uma outra técnica apresentada em [13] é baseada em cabeçalhos OCDMA (*Optical Code Division Multiple Access*) colocados em bandas de comprimentos de onda diferentes do comprimento de onda onde circulam os pacotes.

A tecnologia que permite realizar o processamento do cabeçalho totalmente no domínio óptico está ainda numa fase muito experimental [7],[13], pelo que é necessário encontrar soluções tecnológicas aplicáveis em comutadores ópticos de pacotes. Este último sistema de anexação do cabeçalho tem vantagens sobre o primeiro pelo facto de o cabeçalho poder ser removido e substituido de forma assíncrona relativamente ao pacote. Por outro lado, uma das suas desvantagens tem origem nos seus problemas de susceptibilidade à dispersão na fibra.

Demux/Mux Ópticos e OADMs (Optical Add/Drop Multiplexers)

A Desmultiplexagem e a multiplexagem (Demux/Mux) óptica são tecnologias fundamentais no WDM. Apesar de o WDM ser uma tecnologia bastante madura e robusta, o desenvolvimento de sistemas de Demux/Mux a taxas de transferência elevadas ainda tem alguma margem de evolução. Por outro lado, as tecnologias ópticas de multiplexagem e desmultiplexagem temporal (OTDM - *Optical Time-Division Multiplexing*) estão ainda a amadurecer. Actualmente, existem diversas tecnologias baseadas: em elementos não lineares (TOAD – *Terahertz Optical Assymetric Demultiplexer*[1] ou SESHG – *Surface-Emitted Second-Harmonic Generation*) [35], [36], em portas ópticas (UNI – *Ultrafast Nonlinear Interferometer*) [37], ou em MEMS (*Micro-Electro-mechanical System*) [38]. Apesar de algumas destas propostas permitirem elevadas taxas de transferência, são ainda pouco flexíveis, pouco transparentes à taxa de transferência, dependendo demasiado da variação do atraso dos pacotes, ou necessitam de funções de gestão centralizadas de recursos e de recuperação precisa de sinal de relógio.

Os OADMs (*Optical Add/Drop Multiplexers*) têm como funções inserir (*add*) ou extrair (*drop*) canais ópticos (comprimentos de onda) em ou *bits* de uma sequência de sinais ópticos. Desta forma, usando OADMs, os canais de

[1] Elemento não linear colocado assimetricamente num *loop* de uma fibra.

um sinal de múltiplos comprimentos de onda podem ser adicionados ou descartados sem recorrer a processamento electrónico [39].

As tecnologias mais utilizadas para implementar OADMs no comprimento de onda são as redes de difracção de *Bragg* (FGB – *Fiber Gratings Bragg*) e matrizes de difracção de guias de onda (AWGMs – *Arrayed Wavelength Gratings Multiplexers*) [40], [41], por serem as mais divulgadas e maduras. No entanto, são utilizadas também outras opções tecnológicas, nomeadamente, redes de difracção ajustáveis, matrizes de micro-espelhos, filtros ajustáveis acustico-óptico e anéis ressonantes [40].

Encaminhamento / Comutação

As tecnologias de comutação são muito importantes na implementação de redes ópticas. O aspecto mais atractivo dos comutadores ópticos é o facto de permitirem efectuar encaminhamento de sinais ópticos, sem recorrer a conversões dos sinais para o domínio da electrónica. As soluções totalmente ópticas para comutadores são ainda objecto de estudo e desenvolvimento. Num comutador existem vários parâmetros importantes: o tempo de comutação, as perdas de inserção, intermodulação, razão de potência na saída na comutação *ON-OFF* (*extinction ratio*) e perdas de dependência de polarização (PDL – *Polarization-Dependent Loss*) [39].

Actualmente, as principais tecnologias de comutadores ópticos existentes apresentam-se de seguida [39].

Comutadores opto-mecânicos consistiram na primeira tecnologia de comutadores ópticos a ser comercializada, onde a comutação é efectuada por meios mecânicos, prismas, espelhos ou acopladores direccionais. Este tipo de comutadores apresentam baixas perdas de inserção, baixas perdas dependentes da polarização, baixa intermodulação e são relativamente baratos. No entanto, são também limitados no número de portas e, consequentemente, pouco adaptáveis ao aumento das exigências de capacidade e de complexidade em sistemas escaláveis. O tempo de comutação é da ordem de alguns milisegundos, por isso bastante limitados para redes de ritmos elevados.

Os MEMS (*Micro-Eelectrical Mechanical Switch*) incluem todas as suas vantagens da tecnologia anterior, mas permitem uma maior agregação e menores tempos comutação. Dadas as suas dimensões e propriedades, podem ser colocados em PLCs (*Plannar Lightwave Circuits*), resultando em comutadores muito compactos. Há duas sub-categorias deste tipo de comutadores: MEMS de duas dimensões ou digitais, MEMS 2-D [43], [43], e MEMS de três dimensões ou analógicos, MEMS 3-D [44]. Nos MEMS 2-D o posicionamento do espelhos é biestável (ON/OFF). O limite prático do número de portas neste tipo de comutador é de 32x32 [39]. Devido à sua simplicidade têm perdas ópticas elevadas. Nos MEMS 3-D encontra-se um espelho ajustável funcionando analogicamente, permitindo inclinação variável segundo os dois eixos. Por estas razões os MEM 3-D podem ser considerados como uma tecnologia promissora para a implementação de comutadores OXC (*Optical Cross-Connect*) com um número elevado de portas (>1000). No entanto, apresentam um custo elevado. Os MEMSs podem ser utilizados em multiplexadores de adição/remoção de comprimentos de onda, serviço de monitorização óptica e protecção óptica. Os desafios futuros encontram-se no desenvolvimento dos espelhos, no algoritmo de controlo dos mesmos e na sua implementação.

Os comutadores eléctrico-ópticos utilizam um acoplador direccional e a sua relação de acoplamento é alterada, variando o índice de refracção do material do acoplador. Apesar da sua fiabilidade, os seus tempos de comutação destes comutadores são elevados, dependentes da capacidade dos eléctrodos, apresentando ainda elevadas perdas de inserção e PDL; são também mais caros do que os MEMSs [39].

Os comutadores térmico-ópticos consistem na variação do índice de refracção do material dieléctrico por variação da sua temperatura. Estes comutadores são normalmente pequenos, mas têm problemas com a dissipação de potência, limitações na densidade de integração e intermodulação e perdas de inserção inaceitáveis para determinadas aplicações. A maior parte dos comutadores térmico-ópticos existentes no mercado necessitam de refrigeração. Uma vantagem destes comutadores é o facto de poderem ser integrados com atenuadores e AWGMs [39].

Os comutadores Bolha (*bubble*) podem ser considerados uma sub-categoria dos comutadores térmico-ópticos: têm o mesmo princípio de funcionamento das impressoras de jactos de tinta e apresentando alguma escalabilidade. No entanto, em sistemas de telecomunicações não há certezas quanto à sua fiabilidade e perdas de inserção [39].

Os comutadores de cristal líquido são baseados na variação da polarização de luz incidente, por variação de um campo eléctrico aplicado ao cristal líquido. Estes comutadores não têm partes móveis, são muito fiáveis e têm um desempenho satisfatório. Apresentam limitações no seu projecto, podendo ser afectados por temperaturas extremas [39].

Os comutadores acústico-opticos baseiam-se na interacção entre som e a luz. Neste caso, são viáveis comutações de múltiplos comprimento de onda, dado que é possível ter várias ondas acústicas no material (várias frequências ao mesmo tempo). No entanto, o tempo de comutação é limitado à velocidade do som, na ordem dos microsegundos.

Os comutadores SOA (*Semiconductor Optical Amplifiers*), são dispositivos muito versáteis, podem ser usados em vários módulos de redes ópticas, por exemplo, como simples comutadores ON-OFF, através de variação da tensão de polarização. São dispositivos caros, mas que permitem a implementação de comutadores de grande capacidade (integrando os SOAs e acopladores passivos).

Conversão de Comprimento de Onda

Uma das possíveis funções da conversão de comprimento de onda é resolver problemas de disputa de recursos (que será descrito no ponto 0), permitindo melhorar a eficiência de utilização dos recursos de uma rede óptica, principalmente em sistemas com tráfego tão dinâmico como o das redes baseadas em OPS. Sendo assim, os conversores de comprimento de onda são uma peça essencial e indispensável em comutadores em redes OPS. A conversão do comprimento de onda pode ser obtida através de uma conversão OEO - Optica-Electrica-Optica. No entanto, o desejável é, obviamente, realizá-la totalmente no domínio óptico, devido à sua maior largura de banda e flexibilidade, tempos curtos de configuração, elevada relação sinal-ruído e transparência da taxa de transferência [14].

Os conversores totalmente ópticos são tipicamente baseados em dispositivos altamente não lineares. Exemplo de conversores que se encontram na literatura são: SOAs, que tiram partido dos efeitos não lineares de intermodulação de ganho (XGM – *Cross Gain Modulation*) [17], [18], de intermodulação de fase (XPM – *Cross Phase Modulation*) [17] e também (mas menos frequente) de modulação de mistura de quarta onda (FWM – *Four Wave Mixing*) [19], [20]; configurações de interferométrico que exploram o efeito XPM, tais como MZI [17], TOAD [21], UNI [22], DI (*Differencial Delay Interferometer*) [23] e Interferométrico *Sagnac* Integrado Monolítico (*Monolithically Integrated Sagnac Interferometer*) [24]; guias de onda LiNbO3 (MgO-doped LiNbO3 QPM, quasi-phase-matched) [25], [26]. As configurações interferométricas tiram proveito da XPM, requerendo baixa potência de entrada. De notar que com a configuração interferométrica DI conversão a ritmos superiores aos obtidos com SOAs [27].

Em [28], através da realização de uma nova configuração de interferométrico DI, conseguiram-se taxas de transferência da ordem de 100 Gbit/s e transmissão com cascata de regeneradores 3R bem sucedida a um milhão de quilómetros. Este método consiste em aproveitar ambos os efeitos de intermodulação XPM e XGM, permitindo converter a aproximadamente o dobro da velocidade dos SOAs convencionais. Outro tipo de configuração é baseado num elemento não linear e num filtro óptico RSOF (*Red-Shifted Optical Filter*) com um interferométrico DI. Finalmente, os mesmo autores apresentam uma nova configuração com um SOA seguido de um filtro PROF (*Pulse Reformatting Optical Filter*) [27]. O modo de funcionamento é muito semelhante ao do RSOF, mas com resultados ainda melhores. Os autores afirmam que com esta configuração baseada em SOAs é possível obter a melhor eficiência na conversão de comprimento de onda e na regeneração.

Optical 3R Regeneration

A Regeneração 3R óptica (O3R) consiste num processamento altamente não linear do sinal óptico, de modo a conseguir-se reamplificação, retemporização e reformatação do sinal degradado. É de extrema importância que essa funcionalidade seja realizada totalmente no domínio óptico, pois a utilização de soluções com processamento electrónico tendem a ser mais problemáticas, limitativas e mais caras. O3R é uma funcionalidade essencial em redes totalmente ópticas, principalmente em redes de longa distância e/ou em redes com um número elevado de nós, de modo a evitar efeitos acumulados devido a distorção de sinal e processamento de sinal realizado nos nós da rede.

Um grande número de implementações de O3R propostas na literatura são baseadas no comutador interferométrico semicondutor Simétrico *Mach Zenhder* (SMZ) [46]. No entanto, nestes casos, a O3R resulta numa perda de potência de sinal de 3 a 4 dB. Os autores de [46] conseguiram reduzir essa perda para aproximadamente 0dB, usando uma estrutura PD-SMZ (*Polarization-Discrimination*-SMZ) a uma taxa de 84 Gbit/s. Utilizando um interferométrico não linear ultra-rápido (UNI) conseguiram-se O3R a taxas de transferência acima de 80 Gbit/s [45]. Em [47] propõe-se uma configuração mais simples baseada no conversor de comprimento de onda SOA-DI (*Delayed Interference*), com perdas de inserção inferiores a 1 dB, uma taxa de 40 Gbit/s e uma relação sinal-ruído muito boa.

Outra abordagem diferente surge a 40 Gbit/s, utilizando moduladores de electro-absorção [48]. *C. Schubert et. al.* [49] conseguiram implementar o O3R mais rápido até esta data, a uma taxa de 160 Gbit/s, utilizando uma HNL-DSF (*Highly NonLinear Dispersion-Shifted Fiber*). Em [50], os autores recorrem a uma configuração de Interferométrico de *Sagnac* com SOA (SOA-*Assisted fiber Sagnac interferometer*) para 10 Gbit/s.

IV DISPUTA DE RECURSOS (CONTENTION RESOLUTION)

Outra questão que merece alguma atenção são as estratégias de resolução de disputa de recursos: se dois pacotes provenientes de duas entradas diferentes (portas e/ou comprimentos de onda) forem encaminhados para a mesma porta de saída (porta e comprimento de onda), o comutador deverá tomar uma decisão que evite a perda de um dos pacotes.

Há três estratégias possíveis para a resolução deste problema: um dos pacotes deve ser convertido para outro comprimento de onda disponível, deve ser retido em unidades de armazenamento ópticos; ou deve ser encaminhado para um trajecto alternativo [8]. A tecnologia de conversão de comprimento de onda foi discutida na alínea 0; as duas últimas estratégias serão abordadas de seguida.

Por outro lado, existem alguns estudos que abordam este problema de uma forma global, ou seja, apresentam configurações experimentais que permitem combinar as três estratégias [29]-[31]. Num destes estudos concluiu-se que o esquema mais eficiente de resolução de disputa de recursos passa por, primeiro, efectuar conversão de comprimento de onda, a seguir, reter os pacotes em

unidades de armazenamento e, em último lugar, recorrer ao encaminhamento alternativo [31].

Unidades de armazenamento Ópticas

A maior dificuldade no projecto de nós OPS é a ausência de uma tecnologia expedita para armazenamento e acesso a dados no domínio óptico. A investigação tem-se direccionado no sentido de obter dispositivos, do ponto de vista conceptual, semelhantes às RAMs (*Random Access Memory*) electrónicas, não havendo, no entanto, RAMs ópticas. Actualmente a alternativa óptica mais usual para realizar esta funcionalidade são as FDLs. As FDLs são fibras ópticas de comprimento fixo que permitem guardar um pacote durante o tempo que este leva percorrê-la à velocidade da luz; sem recorrer a técnicas alternativas, o pacote não pode ficar armazenado por tempo indefinido. Por outro lado, as FDLs são extremamente caras, ocupam bastante espaço e têm pouca flexibilidade. Portanto, esta é uma área que requer muita investigação.

Os *buffers* ópticos podem ter um único nível ou vários níveis, com um ou mais elementos em paralelo. As arquitecturas de *buffers* podem ter ou não realimentação, *feed-foward* ou *feedback*, respectivamente. O número de FDLs pode ser minimizado se forem utilizados algoritmos eficientes para projectar os *buffers* ou se utilizarem conversores de comprimento de onda. Há várias configurações de FDLs diferentes que tentam resolver uma determinada questão, nomeadamente, o armazenamento de vários pacotes em simultâneo [32], com atrasos variáveis [33] ou considerando modelo híbrido de memória (optico-eléctrico) [34].

Encaminhamento Deflexivo

Este método explora a dimensão espacial da rede para resolver o problema de disputa de recursos. Um pacote que perca a disputa de recursos, com outro(s) pacote(s), é encaminhado por um trajecto alternativo (normalmente, mais longo) até ao seu destino. A utilização deste método pode causar atrasos, o que resultará na chegada ao destino de pacotes fora de ordem. Por outro lado, a eficiência deste método depende directamente do padrão de tráfego e da topologia da rede [14].

A grande vantagem desta estratégia é o facto de a sua utilização não implicar a necessidade de grande esforço na implementação, nem de algoritmos de controlo, nem de dispositivos de *hardware* suplementares [39].

No entanto, não se pode efectuar encaminhamento alternativo em nós sucessivos de uma forma aleatória, pois o pacote pode regressar ao mesmo nó onde iniciou a resolução do problema de disputa de recursos, podendo desta forma ficar indefinidamente na rede. Portanto, terão que ser impostas regras que permitam utilizar esta estratégia eficientemente e aumentar, consequentemente, a eficiência da própria rede (por exemplo, o número de saltos - *hops* deve ser limitado e quando excedido o pacote deve ser descartado [31]).

V CONCLUSÃO E TRABALHO FUTURO

Neste artigo foram apresentados alguns aspectos gerais do funcionamento das redes OPS. As redes OPS têm limitações tecnológicas que impedem a sua implementação efectiva e eficiente. O tratamento e processamento de cabeçalhos de controlo, o encaminhamento e a questão da disputa de recursos são questões que merecem um estudo mais aprofundado. Uma das tecnologias mais limitativas das redes OPS são o armazenamento óptico de dados, pois, actualmente, as soluções amadurecidas têm claras limitações do ponto de vista de utilização e funcionalidade.

Um outro aspecto de grande importância para trabalho futuro é o estudo dos parâmetros de desempenho da rede ao nível físico (óptico) que têm impacto no desempenho da rede IP e de camadas superiores, tendo em conta os requisitos dos serviços e aplicações emergentes.

REFERÊNCIAS

[1] T. Battestilli. "Optical Burst Switching: a survey". Technical Report TR-2002-10, NC State Univ., Comp. Sci. Dept., July 2002.

[2] A. Carena, D. Vaughn, R. Guadino, M. Shell and D. J. Blumenthal. "OPERA: an Optical Packet Experimental Routing Architecture with Label Swapping Capability". IEEE Journal of Lightwave Technology, Vol. 16, Issue 12, pp. 2135-2145, December 1998.

[3] C. Guillemot et al., "Transparent Optical Packet Switching: The European ACTS KEOPS project approach". IEEE Journal of Lightwave Technology, Vol. 16, Issue 12, pp. 2117-2133, December 1998.

[4] M. Yong Jeon et al., "Demonstration of All-Optical Packet Switching routers with optical label swapping and 2R regeneration for scalable Optical label switching networks applications". IEEE Journal of Lightwave Technology, Vol. 21, Issue 11, pp. 2723-2733, November 2003.

[5] K. Vlachos et al., "An Optical IM/FSK Coding Technique for the Implementation of a Label-Controlled Arrayed Waveguide Packet Router". IEEE Journal of Lightwave Technology, Vol. 21, Issue 11, pp. 2617-2628, November 2003.

[6] X. C. Yuan V. O. K. Li, C. Y. Li and P. K. A.Wai. "A novel self-routing address scheme for all-optical packet-switched networks with arbitrary topologies". IEEE Journal of Lightwave Technology, Vol. 21, Issue 2, pp. 329-339, November 2003.

[7] H. J. S. Dorren et al., "Optical packet switching and buffering by using all-optical signal processing methods". IEEE Journal of Lightwave Technology, Vol. 21, Issue 1, pp. 2-12, January 2003.

[8] J. Yu, G. Chang and Q. Yang. "Optical Label Swapping in a Packet-Switched Optical Network Using Optical Carrier Suppression, Separation, and Wavelength Conversion". IEEE Photonics Technology Letters, Vol. 16, Issue 9, pp. 2156-2158, September 2004.

[9] Y. M. Lin, W. I. Way and G. K. Chang. "A Novel Optical Label Swapping Technique using erasable optical single-sideband subcarrier label". IEEE Photonics Technology Letters, Vol. 12, Issue 8, pp. 1088-1090, August 2000.

[10] D. J. Blumenthal et al., "All-Optical Label Swapping Networks and Technologies". IEEE Journal of Lightwave Technology, Vol. 18, Issue 12, pp. 2058-2074, December 2000.

[11] D. J. Blumenthal, A. Carena, L. Rau, V. Curri and S. Humphries. "All-Optical Label Swapping wavelength conversion for WDM-IP Networks with subcarrier multiplexed addressing". IEEE Photonics Technology Letters, Vol. 11, Issue 11, pp. 1497-1499, November 1999.

[12] Z. Zhu, V. J. Hernandez, M. Y. Jeon, J. Cao, Z. Pan and S. J. Ben Yoo. "RF photonics signal processing in subcarrier multiplexed Optical Label switching communication sytems". IEEE Journal of Lightwave Technology, Vol. 21, Issue 12, pp. 3155-3166, December 2003.

[13] Teixeira, A.; André, P.; Monteiro, P.; Lima, M.; Nogueira, R.; Da Rocha, J.; "All-optical routing based on OCDMA Headers", LEOS04, Tucson, USA, pp. 1046-1047

[14] G. N. Rouskas and L. Xu. "Optical Packet-Switching". Optical WDM Networks: Past Lessons and Path Ahead, (Krishna Sivalingam and Suresh Subramaniam, Editors), Kluwer, Norwell, Massachusetts, 2004

[15] W. Hung, C. Chan, L. Chen and F. Tong. "A bit-serial optical packet label-swapping scheme using DPSK encoded labels". IEEE Photonics Technology Letters, Vol. 12, Issue 8, pp. 1088-1090, August 2000.

[16] S. Xiao, Q. Zeng, J. Wang, J. Xu and Y. Wang. "Realization of a multiwavelength label optical packet switching". IEEE Photonics Technology Letters, Vol. 15, Issue 4, pp. 605-607, April 2003.

[17] T. Durhuus, B. Mikkelsen, C. Joergensen, S. L. Danielsen and K. E. Stubkjaer. "All-Optical Wavelength Conversion by Semiconductor Optical Amplifiers". IEEE Journal of Lightwave Technology, Vol. 14, Issue 6, pp. 942-954, June 1996.

[18] S. Rangarajan, Z. Hu, L. Rau and D. J. Blumenthal. "All-optical contention resolution with wavelength conversion for asynchronous variable-length 40Gb/s optical packets". IEEE Photonics Technology Letters, Vol. 16, Issue 2, pp. 689-691, February 2004.

[19] Z. Li, Y. Dong, J. Mo, Y. Wang and C. Lu. "Cascaded All-optical Wavelength conversion for RZ-DPSK signal based on four-wave mixing in Semiconductor Optical Amplifiers". IEEE Photonics Technology Letters, Vol. 16, Issue 7, pp. 1685-1687, July 2004.

[20] J. H. Lee, W. Berlardi, K. Furusawa, P. Petropoulos, Z. Yusoff, T. M. Monro and D. J. Richardson. "Four-wave mixing based 10Gb/s tunable wavelength conversion using a holey fiber with a high SBS threshold". IEEE Photonics Technology Letters, Vol. 15, Issue 3, pp. 440-442, March 2003.

[21] J.P. Sokoloff, P.R. Prucnal, I. Glesk and M. Kane. "A terahertz optical asymmetric demultiplexer (TOAD)". IEEE Photonics Technology Letters, Vol. 5, Issue 7, pp. 787-790, July 1993.

[22] C. Bintjas, K. Vlachos, N. Pleros and H. Avramopoulos. "Ultrafast Nonlinear Interferometer (UNI)-Based Digital Optical Circuits and Their Use in Packet Switching". IEEE Journal of Lightwave Technology, Vol. 20, Issue 11, pp. 2629-2637, November 2002.

[23] Y. Ueno, S. Nakamura, K. Tajima, S. Kitamura. "3.8-THz wavelength conversion of picosecond pulses using a semiconductor delayed-interference signal-wavelength converter (DISC)". IEEE Photonics Technology Letters, Vol. 10, Issue 3, pp. 346-348, March 1998.

[24] V. M. Menon, W. Tong, C. Li, F. Xia, I. Glesh, P. R. Prucnal and S. R. Forrest. "All-optical wavelength conversion using a regrowth-free monolithically integrated Sagnac interferometer". IEEE Photonics Technology Letters, Vol. 15, Issue 2, pp. 254-256, February 2003.

[25] J. A. Braken and C. Q. Xu. "All-optical wavelength conversion based on MgO-doped $LiNbO_3$ QPM waveguides using an EDFA as a Pump Source". IEEE Photonics Technology Letters, Vol. 15, Issue 7, pp. 954-956, July 2003.

[26] B. Olsson, P. Ohlen, L. Rau and D. J. Blumenthal. "A simple and robust 40-Gb/p Wavelength Converter using fiber cross-phase modulation and Optical filtering". IEEE Journal of Lightwave Technology, Vol. 12, Issue 7, pp. 846-848, July 2000.

[27] J. Leuthold, D. M. Marom, S. Cabot, J. J. Jaques, R. Ryf and C. Randy Giles. "All-optical wavelength conversion using a pulse reformatting optical filter". IEEE Journal of Lightwave Technology, Vol. 22, Issue 1, pp. 186-192, January 2004.

[28] J. Leuthold, B. Mikkelsen, R. E. Behringer, G. Raybon, C. H. Joyner and P. A. Besse. "Novel 3R Regenerator Based on Semiconductor Optical Amplifier Delayed-Interference Configuration". IEEE Photonics Technology Letters, Vol. 13, Issue 8, pp. 860-862, August 2001.

[29] F. Xue, Z. Pan, Y. Bansal, J. Cao, M. Jeon, K. Okamoto, S. Kamei, V. Akella and S. J. B. Yoo. "End-to-end contention resolution schemes for an optical packet switching network with enhanced edge routers". IEEE Journal of Lightwave Technology, Vol. 21, Issue 11, pp. 2595-2604, November 2003.

[30] Z. Pan, H. Yang, Z. Zhu, J. Cao, V. Akella, S. Butt and S. J. B. Yoo. "Demonstration of variable-length packet contention resolution and packet forwarding in an optical-label switching router". IEEE Photonics Technology Letters, Vol. 16, Issue 7, pp. 1772-1774, July 2004.

[31] S. Yao; B. Mukherjee, S.J.B. Yoo and S. Dixit. "A unified study of contention-resolution schemes in optical packet-switched networks". IEEE Journal of Lightwave Technology, Vol. 21, Issue 3, pp. 672- 683, March 2003.

[32] Y. Liu, M. T. Hill, G. D. Khoe and H. J. S. Dorren. "All-Optical buffering using laser neural networks". IEEE Photonics Technology Letters, Vol. 15, Issue 4, pp. 596-598, April 2003.

[33] Y. Liu, M. T. Hill, N. Calabretta, H. de Waardt, G. D. Khoe and H. J. S. Dorren. "Demonstration of a Variable Optical Delay for a Recirculating Buffer by Using All-Optical Signal Processing". IEEE Photonics Technology Letters, Vol. 16, Issue 7, pp. 1748-1750, July 2004.

[34] R. Takahashi, T. Nakahara, K. Takahata, H. Takenouchi, T. Yasui, N. Kondo and H. Suzuki. "Photonic Random Access Memory for 40-Gb/s 16-b Burst Optical Packets". IEEE Photonics Technology Letters, Vol. 16, Issue 4, pp. 1185-1187, April 2004.

[35] K. Deng, R. J. Runser, P. Toliver, I. Glesk, and P. R. Prucnal. "A highly-scalable, rapidly-reconfigurable, multicasting-capable, 100-Gb/s photonic switched interconnect based upon OTDM technology", IEEE Journal of Lightwave Technology, Vol. 18, Issue: 12, pp. 1892-1904, December 2000.

[36] T. G. Ulmer, M. C. Gross, K. M. Patel, J. T. Simmons, P. W. Juodawlkis, B. R. Wasbburn, W. S. Astar, A. J. Spring Thorpe, R. P. Kenan, C. M. Verber, and S. E. Ralph. "160-Gb/s optically time-division multiplexed link with all-optical demultiplexing", IEEE Journal of Lightwave Technology,Vol. 18, Issue 12, pp.1964-1977, December 2000

[37] S. A. Hamilton, B. S. Robinson, T. E. Murphy, S. J. Savage and E. P. Ippen. "100 Gb/s optical time-division multiplexed networks", IEEE Journal of Lightwave Technology, Vol. 20, Issue 12, pp. 2086 – 2100, December 2002.

[38] S. Pau, M. De Angelis and B. Holland. "Time-multiplexed signals and parallel signal analysis/switch optimization for MEMS-based optical cross-connect", IEEE Journal of Lightwave Technology ,Vol. 21, Issue 3, pp. 609-613, March 2003.

[39] G. I. Papadimitriou, C. Papazoglou and A. S. Pomportsis. "Optical Switching: Switch Fabrics, Techniques, and Architectures". IEEE Journal of Lightwave Technology, Vol. 21, Issue 2, pp. 384- 405, February 2003.

[40] S.D. Dods and R. S. Tucker. "A comparison of the homodyne crosstalk characteristics of optical add-drop multiplexers". IEEE

Journal of Lightwave Technology, Vol. 19, Issue 12, pp. 1829 - 1838, December 2001.

[41] D. Mechin, P. Grosso, D. Bose. "Add-drop multiplexer with UV-written Bragg gratings and directional coupler in SiO2-Si integrated waveguides", IEEE Journal of Lightwave Technology, Vol.: 19, Issue: 9, pp. 1282 – 1286, September 2001.

[42] G. Shen, T. H. Cheng, S. K. Bose, C. Lu, and T. Y. Chai. "Architectural Design for Multistage 2-D MEMS Optical Switches". IEEE Journal of Lightwave Technology, Vol. 20, Isseu 2, pp. 178-187, February 2002

[43] T. Y. Chai, T. H. Cheng, S. K. Bose, C. Lu, and G. Shen. "Array Interconnection for Rearrangeable 2-D MEMS Optical Switch" IEEE Journal of Lightwave Technology, Vol. 21, Isseu 5, pp. 1134-1140, May 2003

[44] V. Kaman, X. Zheng, R. J. Helkey, C. Pusarla and J. E. Bowers. "A 32-element 8-bit photonic true-time-delay system based on a 288 /spl times/ 288 3-D MEMS optical switch". IEEE Photonics Technology Letters, Vol. 15, Issue: 6, pp. 849-851, June 2003.

[45] A. E. Kelly, I. D. Phillips, R. J. Manning, A. D. Ellis, D. Nesset, D. G. Moodie and R. Kahsyap. "80 Gbit/s all-optical regenerative wavelength conversion using semiconductor optical amplifier based interferometer". Electroninc. Letters, Vol. 35, Issue 17, pp. 1477-1478, August 1999.

[46] Y. Ueno, S. Nakamura and K. Tajima. "Penalty-Free Error-Free All-Optical Data Pulse Regeneration at 84 Gb/s by Using a Symmetric-Mach–Zehnder-Type Semiconductor Regenerator", IEEE Photonics Technology Letters, Vol. 13, Issue 5, pp. 469-471, May 2001.

[47] J. Leuthold, B. Mikkelsen, R. E. Behringer, G. Raybon, C. H. Joyner and P. A. Besse. "Novel 3R Regenerator Based on Semiconductor Optical Amplifier Delayed-Interference Configuration", IEEE Photonics Technology Letters, Vol. 13, Issue 8, pp. 860-862, August 2001.

[48] T. Otani, T. Miyazaki and S. Yamamoto. "40-Gb/s Optical 3R Regenerator Using Electroabsorption Modulators for Optical Networks". IEEE Journal of Lightwave Technology, Vol. 20, Issue 2, pp. 195-200, February 2002.

[49] C. Schubert, R. Ludwig, S. Watanabe, F. Futami, C. Schmidt, J. Berger, C. Boerner, S. Ferber, and H. G. Weber. "160 Gbit/s wavelength converter with 3R regenerating capability", Electronics Letters, Vol. 38, Issue 16, pp. 903-904, August 2002.

[50] G. Gavioli and Polina Bayvel. "Novel 3R Regenerator Based on Polarization Switching in a Semiconductor Optical Amplifier-Assisted Fiber Sagnac Interferometer". IEEE Photonics Technology Letters, Vol. 15, Issue 9, pp. 1261-1262, September 2003.

Soluções de Banda Larga para Zonas Periféricas e Rurais (País do Lot, França)

Joana Tavares, José Carlos Couto, João Rocha, A. Manuel de Oliveira Duarte

Resumo **– Este artigo discute uma metodologia para uma avaliação socio-económica de oferta de serviços de telecomunicações avançados em zonas rurais e periféricas, neste caso específico, região do Lot em França.**
Na primeira parte, todos os passos necessários para produzir uma avaliação técnico-económica são identificados, tomando em consideração tanto o lado da procura como o da oferta. Seguidamente, são obtidos resultados económicos relativamente a projectos de investimento relacionados com a tecnologia satélite em dois cenários distintos, utilizando para isso uma ferramenta técnico-económica desenvolvida para este fim.

Abstract **- This paper discusses a methodological framework for socio-economic evaluation of providing advanced telecommunication services in rural and peripheral areas, in this specify case, region of Lot in France.**
In the initial part all the steps necessary to produce a techno-economic evaluation of advanced telecommunication services are identify considering both demand and supply aspects. Afterwards, economics outputs concerning telecommunication project investments are presented, for satellite technologies in two distinct scenes, these results are achieved with the support of a custom-designed techno-economic tool.

I. INTRODUÇÃO

Como temos vindo a assistir nestes últimos tempos as telecomunicações têm sofrido uma grande evolução, permitindo partilhar informação entre quase todos os pontos do globo. Contudo existem ainda regiões onde as infra-estruturas são escassas ou mesmo inexistentes e uma vez que as novas tecnologias da informação e da comunicação são um potencial dinamizador do desenvolvimento económico-social, torna-se importante criar soluções à medida das necessidades de modo a que tanto os utilizadores como os operadores se sintam atraídos.

Este trabalho enquadra-se no projecto *Cyberal*, o qual tem por objectivo minorar as dificuldades com que as regiões periféricas e rurais de alguns países do Sudoeste Europeu se confrontam relativamente ao acesso dos instrumentos da Sociedade da Informação, em particular, à Internet.
Em termos concretos, o projecto *CYBERAL* propõe-se disponibilizar e colocar em serviço as plataformas telemáticas de banda larga - à escala real - que permitem o acesso à Internet e aos serviços que lhe estão associados num conjunto de localidades
Trata-se de situações onde os operadores de telecomunicações das respectivas regiões não contemplam fazê-lo no momento actual, nem num futuro previsivelmente próximo, por considerarem tais operações desprovidas de rentabilidade comercial.

Para atingir os anteriores objectivos o projecto *CYBERAL* propõe-se levar a cabo dois grandes tipos de intervenções:

- Correcção de insuficiências infra-estruturais;
- Dinamização dos mercados, através da intervenção ao nível do estímulo da oferta de conteúdos locais e de serviços para a Internet, e também no encorajamento da utilização destes serviços por parte de diversas comunidades de usuários.

Desta forma, torna-se importante fomentar o equilíbrio entre a oferta e a procura, ou seja, criar as condições adequadas para qualificar a procura, comparar as formas de exploração e identificar as melhores políticas de promoção dos serviços telemáticos de banda larga nas regiões periféricas e rurais.
A Figura 1 mostra o compromisso que é necessário estabelecer entre a oferta e a procura de modo a estimular o equilíbrio.

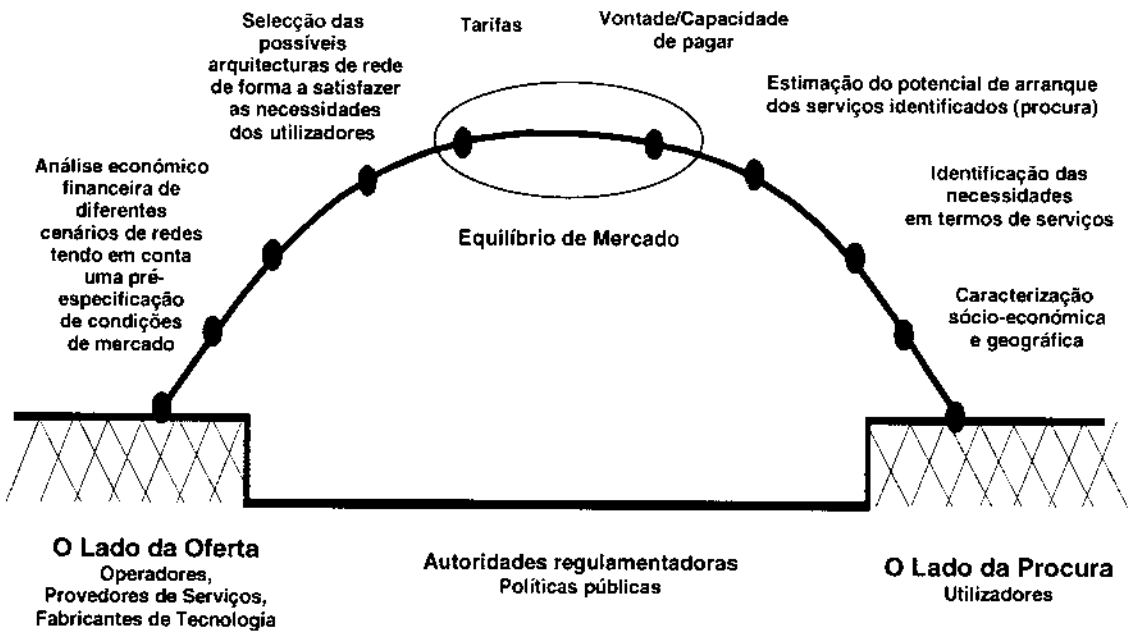


Figura 1: Relação entre a oferta e a procura [1]

Este documento começa por desenvolver uma plataforma metodológica para avaliação técnico-económica de serviços de telecomunicações, em função da tipologia da região rurais. Esta metodologia começou a ser desenvolvida no projecto IST 2000-25172 TONIC – TechnO-econNomICs of IP networks and services – e foi adaptada para as condições específicas desta realidade.
Os passos mais significativos da abordagem utilizada são os seguintes:

1. Definição dos locais com necessidades de intervenção;
2. Caracterização dos locais definidos;
3. Identificação dos possíveis cenários de oferta de infra-estruturas de acesso e de interligação;
4. Identificação de soluções de rede candidatas;
5. Identificação do enquadramento regulamentar aplicável;
6. Possíveis cenários de oferta de serviços de telecomunicações;
7. Negociação com operadores e fornecedores de serviços;
8. Análise técnico-económica para avaliação das soluções de rede e tomada de decisão;
9. Instalação e execução do projecto;
10. Indicadores e elementos de avaliação dos resultados do projecto;

11. Conclusões e recomendações.

Embora o estudo dos serviços de telecomunicações do lado da procura e da oferta sejam igualmente importantes, usualmente muito maior ênfase é dado ao lado da oferta, geralmente definido pelo operador incumbente em associação com a entidade de regulação. Este documento analisa a viabilidade económica do ponto de vista do operador. No entanto todo o estudo foi feito de modo a que o operador não tivesse prejuízo, mas que o período de recuperação do investimento fosse atingido até ao final do tempo de projecto (10 anos).
Este artigo apresenta na secção IIa descrição da tecnologia utilizada. Na secção III é apresentada a região em estudo (região do Lot). Por fim, na secção IV apresenta a análise técnico-económica realizada, e na secção V as conclusões deste estudo.

II. SOLUÇÕES DE REDE

As redes de acesso de banda larga têm de ser suficientemente flexíveis para suportar eficientemente uma série de serviços, quer de banda estreita, quer de banda larga, existindo para o efeito diversas tecnologias de acesso de banda larga.
A situação actual das telecomunicações é caracterizada pela presença de inúmeros factores que vão contribuir decisivamente para a imprevisibilidade dos cenários futuros quer para o mercado de serviços quer para a implementação das redes. Entre estes factores destaca-se o processo de liberalização que está a ocorrer em diversos países europeus, a inovação tecnológica e a introdução de novos serviços. Esta evolução está a provocar uma grande

mudança nas estratégias das empresas: a liberalização do mercado e o fornecimento de redes abertas aumenta assim a competição na área até agora controlada pelo operador público histórico.
Assim, vai surgir um grande requisito por tecnologias que sejam capazes de:

- Fornecer um grande número de serviços (existentes e novos), as chamadas arquitecturas abertas.
- Garantir uma alta rentabilidade, ou seja, grandes receitas comparadas com os investimentos a fazer.

Nesta secção é descrita apenas a solução satélite, uma fez que foi esta a solução implementada.

Satélite

O satélite é o meio utilizado para fazer chegar Internet, dados, LAN, conectando um grande número de sítios dispersos geograficamente, especialmente sítios onde não há possibilidade de implementar outras soluções.
Partindo do standard DVB atribuído à televisão digital, algumas companhias europeias avançaram com projectos para transmitir conteúdos multimédia para os milhares de lares que, em toda a Europa, possuam uma tradicional parabólica.
Os satélites artificiais são largamente empregues em telecomunicações, estes podem ser classificados em geostacionários ou não geostacionários de acordo com sua órbita e podem prover meios de comunicação da seguinte categoria: ponto-a-ponto, ponto-multiponto, multiponto-a-ponto e multiponto-multiponto.
Os satélites são ditos geostacionários quando estes são colocados em orbita circular em torno da terra tal que a sua velocidade de rotação seja a mesma da terra, ou seja, para um observador na terra o satélite comporta-se como se estivesse estacionário num determinado local do céu. Para a comunicação com este tipo de satélite as estações terrestres podem utilizar antenas fixas, antenas estas que apresentam um pequeno custo de operação e manutenção em relação às móveis.
Os satélites são ditos não geostacionários quando estes são colocados numa órbita circular em torno da terra tal que a sua velocidade de rotação não é a mesma da terra, ou seja, para um observador na terra o satélite comporta-se como se estivesse não estacionário num determinado local do céu. A implementação deste tipo de satélite pelas estações da terra é mais dispendioso, pois é necessária a utilização de antenas móveis para acompanhar o movimento do satélite, estas antenas apresentam um custo de operação e manutenção mais elevado.
A plataforma pode fornecer à rede o *hub* da central com múltiplas localizações espalhadas geograficamente. possível fornecer velocidades de *downstream* acima de 52.5 Mbps e *upstream* acima de 307.2 Kbps. [2]

Uma vez pedidos os conteúdos, a rede de satélites é accionada de forma a enviar o conteúdo especificado com um elevado débito para milhares ou milhões de utilizadores em simultâneo. Para receber e descodificar esta informação é necessário dispor de um sistema de recepção de satélite tradicional (parabólica) e de uma *slot PC Card*, através da qual a informação enviada será 'descodificada' para uma forma que o PC possa entender.
Existem algumas vantagens na utilização dos satélites, pois estes têm uma grande largura de banda e conseguem cobrir grandes áreas dando assim a todos os utilizadores as mesmas possibilidades de acesso. Mas em contrapartida também existem algumas desvantagens tais como o alto investimento, a pequena vida útil e as dificuldades relativas a manutenção e lançamento do satélite.
Hoje em dia os satélites permitem difundir sinais de televisão, áudio e dados. È possível a difusão de dados multimédia comprimidos utilizando transmissão satélite com larguras de banda de 512 Kbps, 1 Mbps e 2 Mbps. Os sistemas actuais de compressão de áudio e vídeo utilizando MP3 e MPEG4 permitem a difusão de dados multimédia com larguras de banda reduzidos e custos proporcionais a velocidade de transmissão.
Os dados são transmitidos desde a estação terrestre até ao satélite. O repetidor do satélite muda a frequência do sinal, amplifica e volta a emiti-la para a zona de cobertura.
Uma possível ligação satélite esta esquematizada na figura seguinte.

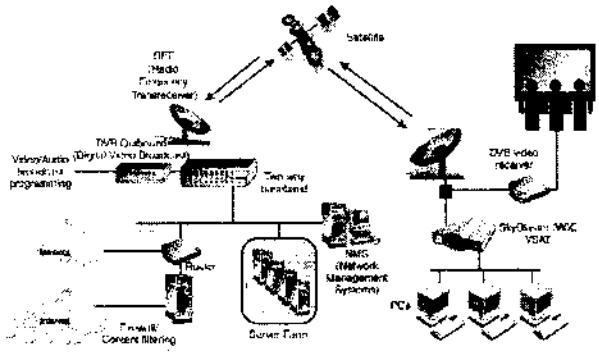


Figura 2: Possível ligação satélite [2]

Os sistemas receptores podem ter a capacidade de interligação com o sistema utilizando uma ligação terrestre de baixa velocidade e um fornecedor de serviços de Internet (ISP).
O desenvolvimento inicial de serviços sobre as plataformas IP via satélite foram o acesso à Internet de alta velocidade com suporte típico de 2 a 4 Mbps até 40 Mbps, o verdadeiro 'motor' de serviços está a ser o suporte do protocolo IP multicast, que permite a recepção simultânea da mesma informação a um número indefinido de receptores sem a necessidade de repetir o envio, porque

a largura de banda necessária para a transmissão, por exemplo de uma sessão de videoconferência, é independente do número de utilizadores que recebem a sessão e somente dependerá da capacidade (em Kbps) que foi solicitada. Desta forma, o IP multicast é o substituto do formato Broadcast ao nível dos dados, obtendo todo o rendimento de uma ampla cobertura geográfica dos sistemas satélites e convertendo-os na parte fundamental dos serviços relacionados com a emissão de vídeo e áudio sobre IP assim como os serviços de distribuição massiva de ficheiros de informação.

Existem duas bandas: a banda C e a banda KU. Estas faixas de frequências são utilizadas nas comunicações com satélites que têm a seguintes características.

A banda C: Espectro de frequência segundo o IEEE: 3.9 GHz até 6.2 GHz.

Espectro de frequência comercial utilizado: 3.7 GHz até 6.425 GHz.

É utilizado um sinal de frequência 6 GHz para comunicação no sentido da terra.

A banda KU: Espectro de frequência segundo o IEEE: 15.35 GHz até 17.25 GHz.

Espectro de frequência comercial utilizada: 10.7 GHz até 18 GHz.

É utilizado um sinal de frequência 14 GHz para comunicação no sentido terra-satélite e 12 GHz no sentido satélite-terra.

III. CARACTERIZAÇÃO DAS REGIÕES

Neste trabalho será alvo de estudo a região de *Lot*, em França. Na figura seguinte pode ver-se a localização geográfica da referida região.

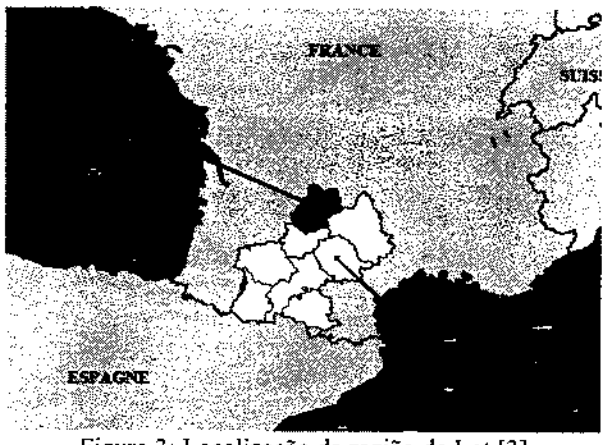


Figura 3: Localização da região de Lot [3]

Dentro da região de *Lot* serão efectuados dois estudos que correspondem a vertentes diferentes:

- Estudo a nível de comunidades, em que a distribuição e colocação de equipamentos é feita tendo em conta as necessidades das várias entidades de cada comunidade
- Estudo a nível empresarial em que a distribuição e colocação do equipamento no terreno é feita à medida das necessidades das empresas.

A. Caracterização das Comunidades

No mapa abaixo podemos ver a localização das comunidades dentro da região de *Lot*.

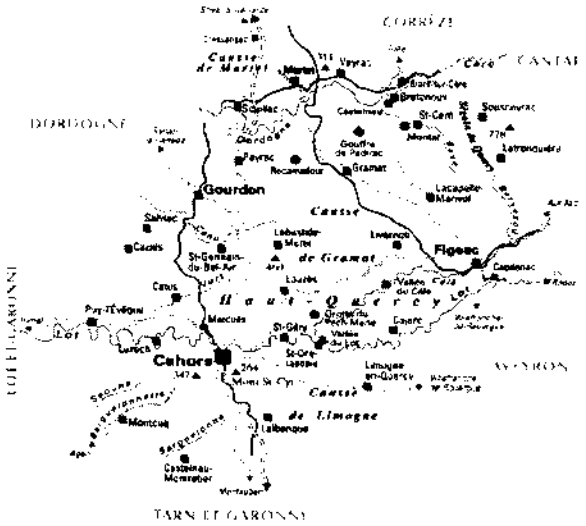


Figura 4: Localização das comunidades na região de *Lot*

Comunidade de Martel

- Cerca de 1500 habitantes;
- Aproximadamente 35 Km^2 de área;
- Colégio com 200 alunos, com necessidade de ligação a outros colégios para troca de informações e viabilização de aplicações pedagógicas inovadoras;
- Os serviços administrativos, assim como algumas empresas no centro da vila justificam a existência de uma rede local;
- Algumas entidades públicas têm a intenção de se ligarem, como por exemplo a *Mairie*, a *Gerdarmerie*, e o *Office du tourisme*.

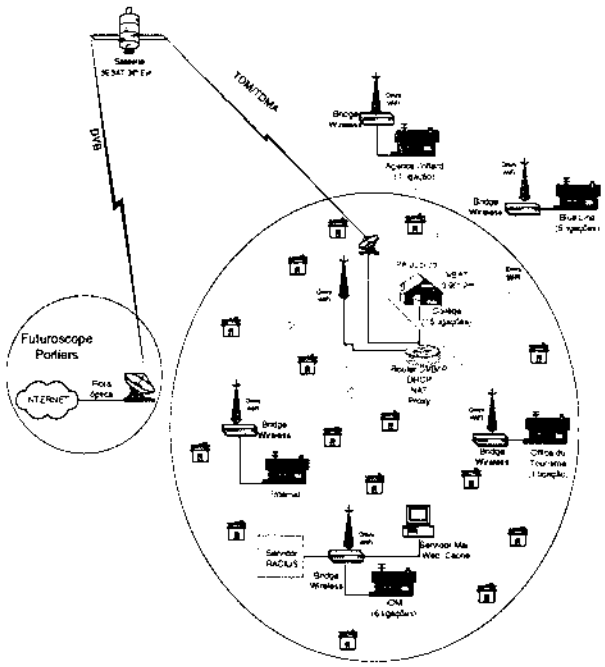


Figura 5: Configuração da comunidade de Martel

Comunidade de Montcuq

- Aproximadamente 1400 habitantes;
- Cerca de 32 Km^2 de área;
- A *Bibliothèque\Médiathèque* tem por missão a sensibilização do público para as TIC e de os iniciar em aplicações inovadoras, tais finalidades exigem um acesso em banda larga para permitirem a troca de conteúdos interactivos;
- O colégio possui cerca de 130 alunos, e necessita de uma ligação a outros colégios para troca de informações e para sessões de videoconferência;

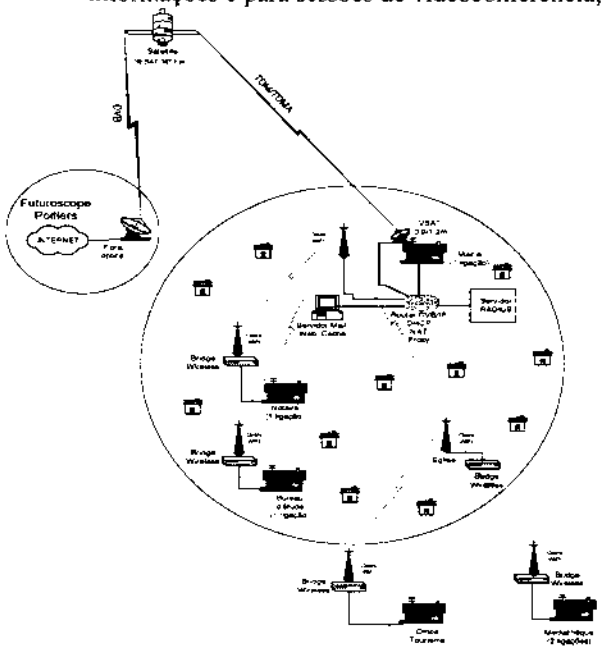


Figura 6: Configuração da comunidade de Montcuq

Comunidade de Leyme

- Cerca de 1000 habitantes;
- Área aproximadamente de 5 Km^2;
- As principais entidades com necessidade de ligação são a *Mairie*, a escola primária, e o centro hospitalar especializado.

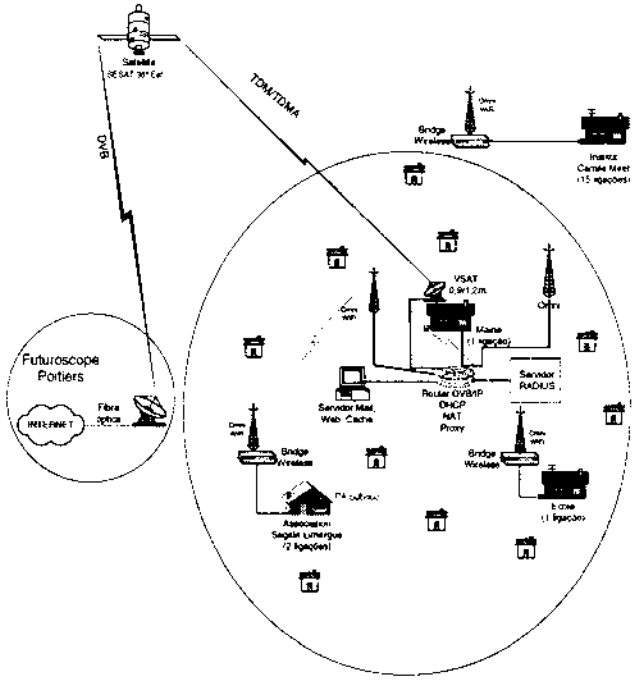


Figura 7: Configuração da comunidade de Salviac

Comunidade de Salviac

- Aproximadamente 1200 habitantes;
- Cerca de 30 Km^2;
- Colégio com 200 alunos, com necessidade de ligação a outros colégios para troca de informações e viabilização de aplicações pedagógicas inovadoras. Têm por hábito a realização de videoconferência com um colégio Espanhol.
- As aplicações multimédia desenvolvidas em *Salviac* focam a iniciação há Internet, acompanhamento ao uso das TIC, promoção turística de *Salviac* (através de postais electrónicos) e apoio à criação de sites;
- Possibilidade de ligação em rede da *Mairie*, do ponto multimédia e da *Gendarmerie*

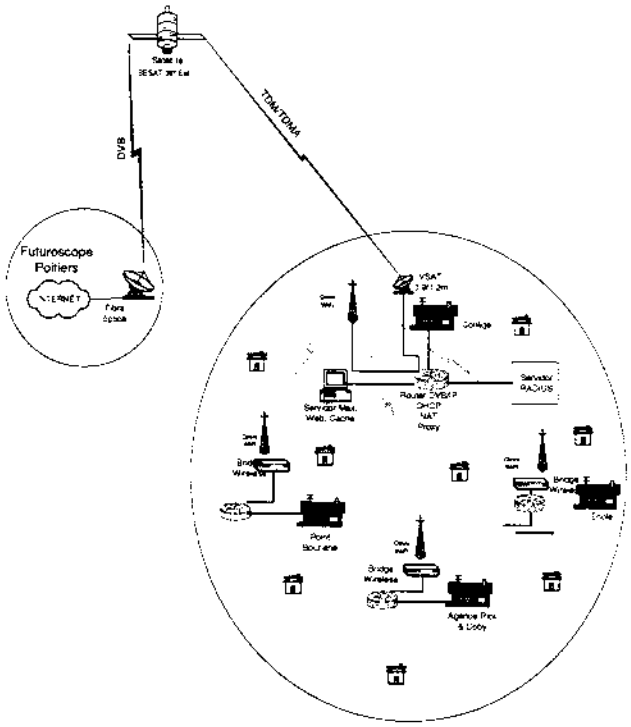


Figura 8: Configuração da comunidade de Salviac

B. *Caracterização dos pólos industriais*

- *Pépinière* (incubadora de empresas)
- *Parc des Expositons*, com um número máximo de 24 stands ligados à Internet
- *Zone Artisanal* (zona industrial)

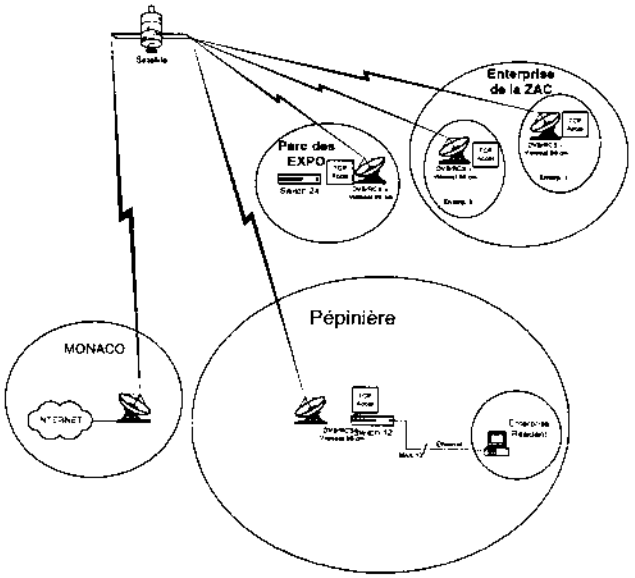


Figura 9: Situação geral proposta para os pólos industriais

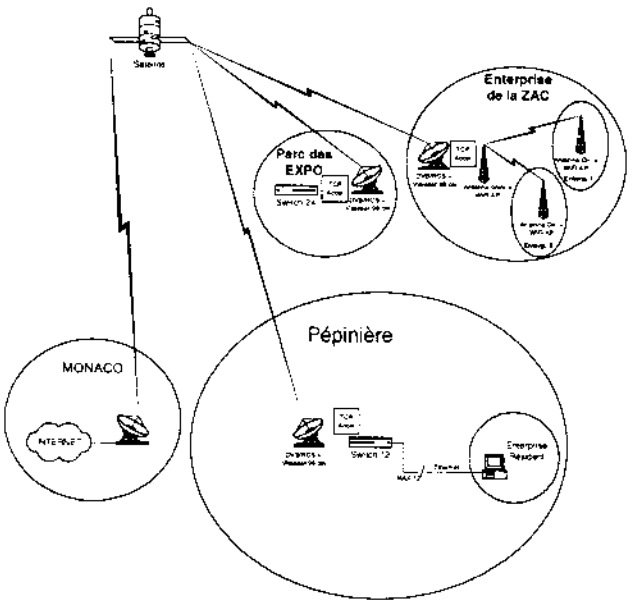


Figura 10: Situação alternativa para os pólos industriais

IV. AVALIAÇÃO TÉCNICO-ECONÓMICA

Com base no conhecimento da região e com alguns pressupostos foi feita uma avaliação técnico-económica para testar a viabilidade da implementação da tecnologia satélite nessas regiões.

Temos como pressupostos o número de clientes, residenciais e empresariais, que vão aderir ao serviço, bem como o preço a pagar por cada cliente.

De notar que nesta secção, por uma questão de síntese, serão apenas apresentados gráficos e tabelas necessários para a compreensão dos resultados obtidos, e não todos os resultados em que se baseiam as conclusões apresentadas.

Pressupostos para as comunidades

- A duração do projecto é 10 anos (2004-2013);
- Os impostos sobre os lucros (*TaxRate)* e a taxa de actualização (*DiscountRate)* são assumidas como de 5% e 30% respectivamente;
- Os clientes empresariais pagam uma tarifa de adesão de 800€ e uma tarifa anual de 2500€, enquanto que os clientes residenciais pagam de tarifa de adesão 100€ e de tarifa anual 350€.
- As tarifas anual e de adesão têm uma erosão de 5% e 2% respectivamente.
- A taxa de penetração dos clientes empresariais varia de 20% a 85%, e nos clientes residenciais de 1% a 15%.
- O custo para manutenção e administração da rede é de 2700€.

Os valores obtidos para a TIR (Taxa Interna de Rendibilidade) com estes pressupostos aproximam-se dos 10%, isto porque, este era uma dos nossos objectivos iniciais. Este estudo foi feito com a base de que não

poderia dar prejuízo, e que o PayBack Period (período de recuperação do investimento) fosse abaixo dos 10anos, ou seja, antes do término do projecto.
Este valor para o TIR foi em quase todos os cenários cumpridos, no entanto as tarifas a pagar por cada cliente podem disparar de uns cenários para os outros, uma vez que para obter um TIR de 10%, tivemos que 'manipular' essas mesmas tarifas.

- *Estudo por comunidade*

Na figura seguinte estão representados os resultados económicos mais relevantes para a avaliação do projecto: TIR, VAL e período de recuperação do investimento.

Nome	Valor
VAL	30.099
TIR	10,1 %
Período de recuperação do investimento	10

Figura 11: Resultados económicos

De seguida podemos observar os investimentos feitos por segmento de rede e por tipo de equipamento, respectivamente.

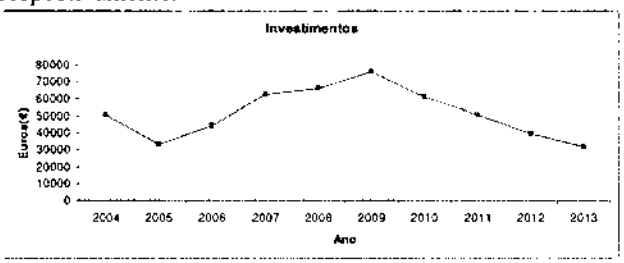


Figura 12: Investimentos realizados

Como podemos observar, no início do projecto temos um grande investimento, o que é normal, devido à necessidade de equipamento. Em 2009 temos um pico nos investimentos, uma vez que à medida que novos clientes aderem ao serviço vai sendo necessário comprar novo equipamento, neste caso é necessário um novo terminal de satélite, daí o pico nos investimentos.
A partir de 2009 os investimentos vão descendo uma vez que não é necessário novo equipamento.

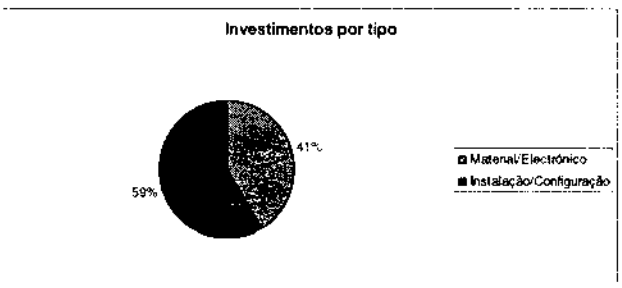


Figura 13: Investimentos por tipo

Como se pode verificar a maior parte do investimento é imputado aos custos de instalação, representando o investimento em material cerca de 40% do total investido. Isto acontece porque a tecnologia é barata sendo cara a sua instalação e configuração, não pela sua extrema dificuldade, mas porque normalmente é feita por uma mão-de-obra especializada.

O estudo da sensibilidade é muito importante num projecto deste tipo, neste caso foi estudado a sensibilidade do TIR a vários tipos de *inputs*.
Com este estudo da sensibilidade pretendemos saber qual o impacto que uma dada variação num dado input provoca na TIR.

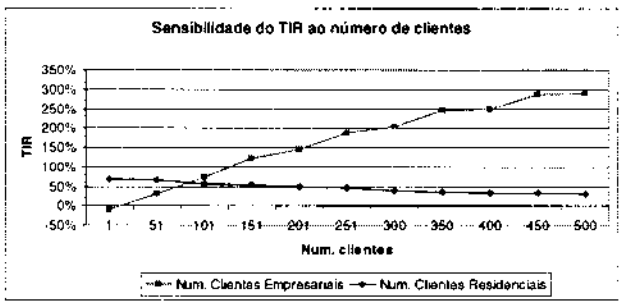


Figura 14: Sensibilidade da TIR ao número de clientes

Como se pode ver, aqui temos diferentes realidades para o caso em que o número de clientes residenciais ou empresariais aumenta.
Quando aumenta o número de clientes residenciais a TIR diminui ligeiramente, isto porque as tarifas mensais consideradas para estes são baixas. Logo, quando o número de clientes residenciais aumenta, o aumento de investimento é superior ao aumento das receitas geradas, pelo que estas não serão suficientes para suportar o investimento efectuado.

Ao aumentar o número de clientes empresariais podemos ver que tem uma maior influência sobre a TIR, e que contrariamente à influência dos clientes residenciais, a TIR aumenta.
Esta diferença deve-se a um simples facto, é que o material usado para os clientes residenciais e para os clientes empresariais é o mesmo. O que muda é a largura de banda disponibilizada (mais cara para os empresariais) e as tarifas que cada um paga (menores para os residenciais). Esta abordagem privilegia os clientes residenciais, isto é, as tarifas pagas pelos clientes empresariais irão, de certo modo, subsidiar parte do custo associado aos clientes residenciais. Assim deste modo podemos concluir que neste estudo, os clientes empresariais têm um papel vital para a sustentabilidade deste projecto.

Pólos industriais

➔ *Pépinière*

A *Pépinière* dispõe de antena emissora/receptora de satélite (DVB/RCS) de um acelerador TCP/IP e ainda de um *switch* de 12 portas, o que permite que as empresas residentes na *Pépinière* tenham acesso à Internet.
O operador fornece um serviço de Internet, com largura de banda de 512 Kbps de upload e 1024 Kbps de download. Será fornecido á *Pépinière* uma aplicação para 'controlar' o consumo de tráfego de cada utilizador, o que dará a *esta* a possibilidade de facturar os utilizadores mediante o seu consumo.

Pressupostos

- Duração do projecto: 10 anos (2004-2013)
- Taxa de actualização: 5% (*Discount Rate*)
- Taxa de impostos sobre lucros: 30% (*TaxRate*)
- O modelo de tarifário na *Pépinière* é composto por uma tarifa anual e uma tarifa de adesão.
  A tarifa de adesão a ser paga por cada cliente é de 150 €, e a tarifa anual é de 740 €. (Posteriormente será instalado um software de controlo de utilização, obrigando assim os clientes que utilizam mais tempo o serviço a pagar mais.)
- A erosão sofrida na tarifa de adesão e na tarifa anual é de 1%.
- No caso da *Pépinière* a penetração ao serviço é de 100%, uma vez que cada empresa que faça parte desta incubadora de empresas tem automaticamente ligação à Internet.
- O custo de manutenção e administração da rede é 700€.

Resultados económicos mais relevantes

Na figura seguinte estão representados os resultados económicos mais relevantes para a avaliação do projecto: TIR, VAL e período de recuperação do investimento.

Name	Value
VAL	778
TIR	10,5 %
Período de recuperaçao do investimento	6

Figura 15: Resultados económicos

Neste estudo um dos objectivos a atingir era que a Taxa Interna de Rentabilidade fosse de aproximadamente 10%. Para que tal acontecesse as tarifas foram ajustadas até conseguirmos este valor.
Com uma Taxa Interna de Rentabilidade de 10% obtivemos um Valor Actual Líquido de 778 € e temos um período de recuperação do investimento de 6 anos.

De seguida são apresentados os valores dos investimentos globais e por tipo.

Figura 16: Investimentos

Como podemos observar, no primeiro ano do projecto temos um grande investimento, o que é normal, devido à necessidade de equipamento. Contudo no ano de 2005, e até ao fim do projecto, os investimentos mantêm-se constantes uma vez que não há necessidade de mais equipamento, isto porque, a taxa de penetração é de 100%, ou seja, não vai haver adesão por parte de novos clientes. Durante estes anos o investimento é somente o aluguer da ligação satélite.

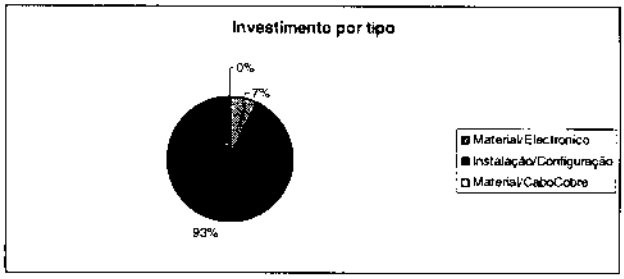


Figura 17: Investimento por tipo

Como se pode verificar quase todo o investimento é atribuído aos custos de instalação e configuração do equipamento, representando 93% do total investido. Uma pequena percentagem (7%) do investimento é em material electrónico. O investimento do cabo de cobre é desprezável, sendo aproximadamente 0%.

➔ *PARC DES EXPOSITIONS*

No *Parc des Expositions*, é proposta a instalação de um equipamento terminal de satélite próprio de modo a haver autonomia e uma rentabilização rápida do investimento.
É disposto para além da antena receptora (VISIOSAT) de satélite um acelerador de TCP/IP e de um *switch* de 24

portas que vai permitir que os vários expositores tenham acesso a Internet. É necessário ter em conta que o limite máximo de stands é 24.
O operador fornece um serviço de Internet, com largura de banda de 512 Kbps de upload e 1024 Kbps de download, a qual é taxada diariamente.
Na Figura 18 são apresentados os preços dos componentes necessários à instalação do serviço.

Componentes	PARK EXPO
Estação terminal de satélite	2.125 €
Acelerador TCP	1.150 €
Switch 12 portas	841 €
TCP Inst. & Config.	600 €

Figura 18: Preço dos componentes

Pressupostos

- Duração do projecto: 10 anos (2004-2013)
- Taxa de actualização: 5% (*Discount Rate*)
- Taxa de impostos sobre lucros: 30% (*TaxRate*)
- O modelo tarifário adoptado para este caso, é o de uma tarifa horária, ou seja, cada cliente pagará uma tarifa por cada hora de utilização do serviço de acesso à Internet.

  O valor proposto é de 2,5 €/h sem limite de consumo de tráfego. Esta tarifa foi considerada como constante ao longo dos anos de vida do estudo, ou seja, consideramos que esta tarifa não sofreria erosão. É também de referenciar o baixo valor proposto para a tarifa, isto acontece porque este estudo não tem fins meramente lucrativos, mas tem acima de tudo fins promocionais quer do serviço quer do desenvolvimento socio-económico da região.
- Neste caso a penetração ao serviço é praticamente impossível de determinar, pois depende do número de feiras, a sua duração, o seu tema, etc...
  Assim sendo consideramos que por feira haveria um total de 12 utilizadores (dos 24 possíveis) que usufruiriam do serviço durante 8 horas por cada dia de exposição, e em cada mês considerou-se que haveriam 5 dias de exposição. Assim deste modo foi possível determinar um número de horas anual de utilização do serviço, e tornou-se deste modo possível aferir resultados para as receitas.
- Custos de administração e manutenção da rede são de 2700€.

Resultados económicos mais relevantes

De seguida, na Figura 19, são apresentados os resultados mais importantes para a avaliação do projecto: TIR, VAL e período de recuperação do investimento.

Name	Value
VAL	22.088
TIR	222,0 %
Período de recuperaçao do investimento	1

Figura 19: Resultados económicos

Como se pode facilmente inferir, para as tarifas propostas (2,5 €/h) os resultados económicos são bastante satisfatórios. Sendo a totalidade do investimento recuperada em apenas 1 ano. Este valor da tarifa é proposto, como já foi dito atrás, com o intuito de dinamizar o serviço. Para uma melhor referência actual, este valor é cerca de metade do valor que é praticado em situações análogas (parques de exposição) em Portugal.

São de seguida apresentados os valores dos investimentos globais e por tipo.

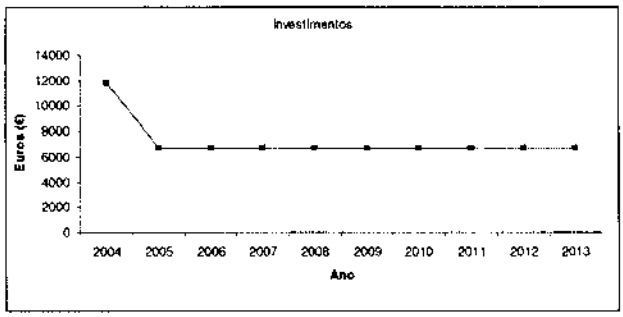


Figura 20: Investimentos

Como se pode observar, no início do projecto há um grande investimento, o que é perfeitamente normal devido à instalação do equipamento. Após este investimento inicial, os investimentos estabilizam. O valor em que estabilizam corresponde ao valor que é gasto no aluguer do satélite (TW3). Ou seja, não é necessário aquisição de mais equipamento além do inicial.

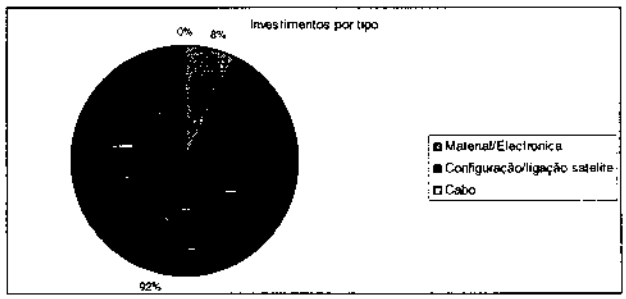


Figura 21: Investimento por tipo

Como se pode verificar no gráfico acima, a quase totalidade dos investimentos é direccionada para a instalação e configuração do material e também para o aluguer da ligação satélite. O investimento feito no material corresponde ao investimento que é feito inicialmente na compra do material. O investimento no cabo é tão baixo e irrisório que é considerado como nulo, face aos outros investimentos.

→ *ENTREPRISE DE LA ZAC*

Para a *Zone Artisanale* foram propostas duas soluções. A primeira é a que está representada na Figura 9 em que cada empresa tem uma estação terminal de satélite e um acelerador TCP/IP.
A Figura 22 apresenta o custo dos componentes para esta solução.

Componentes	ZAC (Opção 1)
Estação terminal de satélite	2.125 €
Acelerador TCP	1.150 €
TCP Inst. & Config.	600 €

Figura 22: Preço dos componentes para a primeira solução proposta para a *Zone Artisanale*

A alternativa a esta arquitectura é a que está representada na figura seguinte.

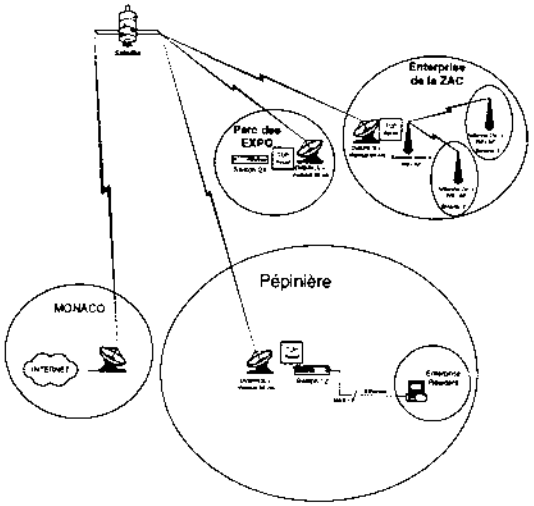


Figura 23: Proposta alternativa para a Zone Artisanale

Os custos do investimento em cada empresa são suportados pelas mesmas, somente o equipamento comum de *WiFi* é suportado pelo gestor da *Zone Artisanale (ZAC)*.
Para a segunda proposta o valor dos componentes necessários são apresentados na Figura 24.

Componentes	ZAC (Opção 2)
Estação terminal de satélite	2.125 €
Acelerador TCP	1.150 €
TCP Inst. & Config.	600 €
Emissor *WiFi*	1.237 €
Antena direccional	234 €
Instalação	660 €
Opção de 'tunneling' seguro	1.325 €

Figura 24: Preço dos componentes para a segunda solução proposta para a *Zone Artisanale*

Pressupostos

- Duração do projecto: 10 anos (2004-2013)
- Taxa de actualização: 5% (*Discount Rate*)
- Taxa de impostos sobre lucros: 30% (*TaxRate*)
- Os custos de manutenção e administração da rede são de 2000 €.
- Modelo tarifário: Para o caso da opção 1, o tipo de serviço de Internet é definido mediante as necessidades e o número de utilizadores de cada empresa. O preço do serviço é deste modo o mesmo para cada empresa, independentemente dos consumos individuais. Assim cada empresa terá as seguintes tarifas: Tarifa de adesão: 300€ e Tarifa anual: 4700€. No que respeita à segunda hipótese o tipo de serviço Internet é decidido mediante o número de utilizadores e das necessidades das empresas. No entanto inicialmente vai ser cobrada uma taxa fixa de 100 € por mês (1200 € anuais). Posteriormente e recorrendo a um software de gestão de consumos, a taxação será feita segundo os consumos individuais de cada empresa. Cada empresa terá ainda um custo de adesão de 200 €.
- Erosão: Os valores previstos para a erosão da tarifa anual são os seguintes:
  Opção 1: 3% ao ano
  Opção 2: 1% ao ano.
  Para a tarifa de adesão os valores da erosão previstos são os seguintes:
  Opção 1: 2% ao ano
  Opção 2: 2% ao ano
- As taxas de penetração ao serviço foram consideradas como diferentes, essencialmente devido à diferença de custos para as empresas que há entre as duas arquitecturas de rede.
  Assim temos para a opção 1:
  Penetração inicial: 2 %
  Penetração final: 79 %
  Enquanto que para a opção 2 temos:
  Penetração inicial: 21%
  Penetração final: 85 %

Resultados económicos mais relevantes

Na Figura 25 e Figura 26, temos os resultados económicos para as duas opções:

Nome	Value
VAL	123.266,85
TIR	10%
Periodo de recuperação do investimento	N.D

Figura 25: Resultados económicos Opção1

Nome	Value
VAL	9.284
TIR	10,1 %
Periodo de recuperação do investimento	8

Figura 26: Resultados económicos Opçao2

No caso da opção 1 o período de recuperação do investimento é maior que o tempo em que o estudo decorre. Esta razão será explicada no ponto seguinte, nos investimentos. As tarifas foram planeadas de modo a que a TIR tivesse um valor de 10%.
Na opção 2 temos um período de recuperação do investimento de 8 anos, aqui à semelhança de casos anteriores, as tarifas foram manipuladas de modo a que o valor obtido para a TIR seja de 10%.
Chama-se no entanto a atenção para o facto de na opção 1 o VAL ser muito maior do que na opção 2. Tal deve-se ao facto de na opção 1 haver um enorme investimento em material bastante dispendioso.

Veremos de seguida alguns valores para os investimentos.

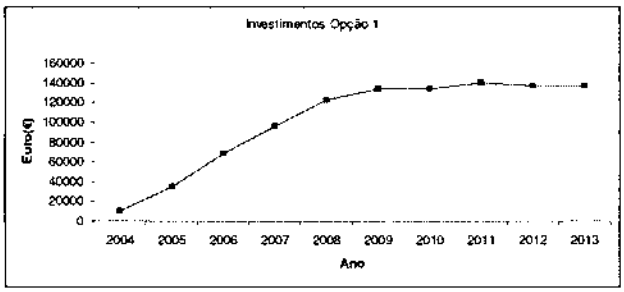


Figura 27: Investimentos opção1

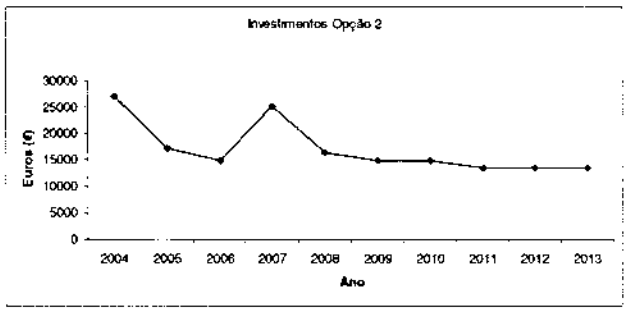


Figura 28: Investimentos opção2

Na opção 1 o investimento tem uma evolução muito parecida com a da penetração. Ora isto acontece porque cada cliente terá um receptor satélite nas suas instalações, e sendo este um material muito caro depressa os investimentos atingem quantias avultadas. Um outro factor que influencia este enorme valor é o facto de que cada cliente gastará uma ligação satélite. O preço dos investimentos estabiliza quando a penetração atinge o ponto de saturação, sendo que a partir deste ponto os investimentos referentes apenas ao aluguer da ligação satélite.
No que respeita à opção 2 inicialmente temos um investimento avultado, pois corresponde ao investimento inicial que engloba a ligação satélite e material de *WiFi*. Depois há um decréscimo natural pois os únicos investimentos que vão sendo feitos são para o material *WiFi* que tem de ser comprado cada vez que uma empresa adere ao serviço, e para o aluguer da ligação satélite. Em 2007 há de novo um pico nos investimentos, isto corresponde à situação em que é adquirido mais equipamento satélite. Esta situação (compra de mais equipamento para outra ligação satélite) acontece porque um equipamento satélite tem largura de banda limitada. Dado que cada cliente consome uma dada largura de banda, então sempre que o número de clientes a usufruir deste equipamento ultrapassa esta largura de banda terá que se adquirir outro equipamento.

De seguida analisaremos a sensibilidade da TIR ao número de clientes.

Figura 29: Sensibilidade da TIR ao número de clientes

Como se pode verificar na Figura 29 na opção 2 há uma maior variação da TIR quando se varia o número de clientes. E contrariamente ao que seria espectável quando se aumenta o número de clientes a TIR diminui. Isto acontece porque um maior número de clientes vai acarretar um aumento nos custos, aumento este que não será suplantado pelas receitas causando uma diminuição na TIR na ordem dos 50%. Para a opção 1 também se verifica que a TIR diminui com o aumento do número dos clientes. Contudo neste caso era espectável que o comportamento fosse este, pois como foi visto anteriormente, esta opção acarreta grandes custos por cada novo cliente.

V. CONCLUSÕES

Ao longo deste estudo fomos confrontados com as várias realidades que os operadores de telecomunicações enfrentam hoje em dia quando estudam a possibilidade de fornecer serviços a zonas remotas. Há casos em que a passagem de um simples cabo de cobre é das primeiras hipóteses a serem excluídas, pois quando se fala em levar este cabo para zonas montanhosas, os custos de aberturas de valas para a sua passagem colocam este tipo de tecnologia fora de questão,

Neste estudo foi dado principal ênfase a soluções *WiFi*, devido a especificações do projecto (*Cyberal*) em si. Foram estudados vários cenários em que o meio de acesso é via satélite e que o meio de distribuição é via *WiFi*.

À excepção de um caso (Zac – opção 1), os projectos são viáveis e oferecem rentabilidade. É de chamar novamente a atenção que a meta deste estudo é que o operador tenha uma TIR na ordem dos 10%. Com este objectivo em mente, as tarifas foram manipuladas de modo a que a TIR tivesse o valor desejado. As tarifas anuais e de adesão obtidas situam-se dentro dos valores praticados no mercado actual das telecomunicações em zonas mais atractivas para os operadores (zonas de uma maior densidade urbana).

No caso da Zac – opção 1, caso em que cada cliente é possuidor de uma ligação satélite, o projecto torna-se desinteressante para um operador. Esta solução tem contudo algumas vantagens face à outra solução (existe um receptor satélite e depois há uma distribuição do sinal via *WiFi* para os clientes). Uma delas é a fiabilidade da ligação, pois o *WiFi* pode apresentar alguns problemas em condições atmosféricas muito adversas e apresenta também alguns problemas quando há obstáculos entre os pontos de ligação.

Ao terminar este estudo podemos inferir que o mercado de banda larga das zonas rurais e periféricas pode funcionar como impulsionador para operadores emergentes, pois os investimentos são reduzidos e os proveitos podem ser elevados.

No que respeita às tecnologias, o *WiFi* é uma tecnologia relativamente recente, e que tem sido bastante utilizada com um vasto leque de aplicações, sendo a mais actual o fornecimento de serviços banda larga em lugares públicos vulgo: *hotspots*, tais como aeroportos, estações ferroviárias, parques de exposições, hotéis, etc...

É também de salientar a grande vantagem das soluções *Wireless* face às soluções mais tradicionais, nomeadamente o impacto social, pois nas tecnologias sem fios não há a necessidade de abrir ruas para instalação de cabos.

Uma tecnologia actualmente em testes, é a *powerline*, que consiste em aproveitar as linhas de eléctricas para transporte de dados. Esta tecnologia tem como principal entrave aspectos legais, mas pensa-se que num futuro próximo poderá ocupar um lugar cimeiro no que respeita ao acesso em banda larga.

REFERÊNCIAS

[1] Http://gsbl.det.ua.pt/gsbl/cyberal.

[2] http://www.networkmagazineindia.com/200206/case1.shtml

[3] Agence Lotoise de Développement, Le Lot Économique et Social 2001/2002.

Outros sites consultados:

http://www.telc.sounu.go.jp/e/system/trunk/fwa/

http://www.phsmou.or.jp/newsletter/issue4/WLLsystem.html

http://www.distribution-europe.com/

http://www.mainnet-plc.com/plus_architecture.htm

http://www.spower.com.sg/spt/faq.htm

http://www.plca.net/whatisplc.asp

http://www.phsmou.or.jp/newsletter/issue4/WLLsystem.html

Desenvolvimento de um projecto em VHDL para ordenação de vectores ternários

Filipe Cunha

Resumo **– Este artigo descreve os passos percorridos no desenvolvimento de um circuito desenvolvido numa FPGA que ordenasse uma matriz ternária 16x16 e que a imprimisse num monitor VGA.**

Ao iniciar o programa, a matriz original irá ser preenchida com 0, 1 e '-' (don't care) e em seguida será ordenada através do algoritmo de ordenação *bubblesort* sendo depois ambas imprimidas no ecrã.

Abstract – This paper describes the steps to be executed for the design of an FPGA-based circuit that sorts ternary vectors of a matrix (with the size 16x16) and visualizes the result on a VGA monitor screen. The initial matrix is arbitrary filled with values 0, 1 and '-' (don't care) and then it is sorted using the *bubblesort* algorithm. Finally the initial and the sorted matrices will be displayed on the monitor screen.

I. INTRODUÇÃO

Este trabalho aqui descrito, foi o projecto de avaliação da cadeira Computação Reconfigurável, leccionada pelo Professor Valery Sklyarov. Na realização deste projecto recorreu-se aos conhecimentos e material [1] fornecidos pelo docente da cadeira. A especificação do projecto foi realizada em VHDL e, para a sua implementação, fez-se uso do ambiente Xilinx ISE 5.2 [2].

A.. Descrição do projecto

O objectivo deste projecto é o de desenvolver um circuito, utilizando a linguagem VHDL, que ordene matrizes de tamanho 16x16 em função do número de elementos (0, 1 ou '-') de cada linha. Por exemplo, se o elemento de ordenação fosse o 0 (zero), depois da matriz ser ordenada, na primeira linha fica a linha que tenha menos zeros. Por outro lado na última linha ficaria a linha que tenha o maior número de zeros. Antes da matriz ser ordenada é preenchida dinamicamente por um algoritmo aleatório, sendo depois copiada para outra matriz de igual tamanho. Esta cópia é que vai ser ordenada dando a possibilidade de no fim da execução do programa tanto a matriz original como a matriz ordenada serem imprimidas no monitor VGA, tal como o elemento de ordenação.

II ORGANIZAÇÃO ESTRUTURAL DO PROJECTO

Na figura 1 podem ser vistos todos os blocos constituintes deste projecto.

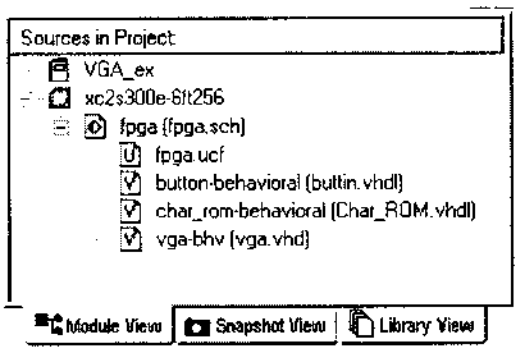


Figura 1 – Blocos constituintes do projecto.

Como se vê na figura acima, o projecto é formado por três blocos em linguagem VHDL e um ficheiro com extensão UCF. UCF significa *User Constraint File* e indica quais os pinos atribuídos e definições do sinal de relógio. Neste projecto usou-se um relógio de 25 MHz

No bloco "button" é controlado o uso dos botões usados (botão que modifica o elemento de ordenação). No bloco "char_rom" está especificada uma ROM que contém os três caracteres 16x16 usados na impressão das matrizes. O bloco "vga" é o bloco principal do projecto. É lá que estão as operações relacionadas com as matrizes e com a impressão destas no monitor VGA.

Na figura 2 seguinte são mostradas as ligações existentes entre os três blocos.

III IMPLEMENTAÇÃO

Para a realização deste projecto foi utilizado o projecto *VGA_ex* disponibilizado pelo Professor Valery Sklyarov. No projecto *VGA_ex* era implementado um contador de 0 a 9 accionado por um botão específico da placa Trenz Spartan IIE [3], contador que era imprimido no monitor VGA.

Ao começar este projecto foi implementada a matriz como um array de *std_logic_vector*. Posteriormente deparei-me com um problema de comparações pois a linguagem VHDL aceitava como verdade a comparação '-'='0' ou '-'='1'. Neste caso sempre que um dos valores da matriz fosse *don't care* esse valor seria contabilizado

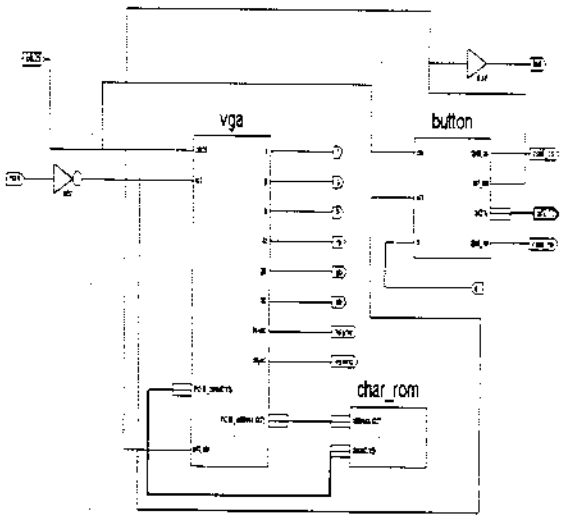


Figura 2 – Ligações entre os blocos

como 0 ou 1 consoante o valor de VAL (valor que vai definir a ordenação da matriz), não permitindo assim a ocorrência de *don't cares*.

Foi alterada então a matriz para um array de array de inteiros de 0 a 2, onde o 2 correspondia ao *don't care*, e assim já não existia o problema de comparações tornando o projecto viável.

Antes de ser construído o código para que a matriz fosse imprimida no monitor VGA, foi decidido implementá-lo de modo a imprimir uma pequena matriz (4x4) no LCD da placa Trenz Spartan IIE, para que testasse o funcionamento do processo de ordenação e da função contar.

Depois de verificado o bom funcionamento do programa começou-se a implementar a parte de impressão da matriz no monitor VGA.

Implementei então um processo de ordenação baseado no algoritmo *bubblesort*, mas com a particularidade de em vez de comparar o valor de um array, compara o número de 0, 1 ou – em cada linha da matriz e troca linhas adjacentes caso seja necessário. O número de 0,1 ou – em cada linha é devolvido pela função **contar**.

```
function contar ( linha : tipo_linha) return
std_logic_vector is
variable contador: std_logic_vector(0 to 4);
 begin
       contador:="00000";
       for i in 0 to N-1 loop
         if linha(i)=VAL then
           contador:=contador+1;
         end if;
       end loop;
 return contador;
 end contar;
```

O processo de ordenação começa com a verificação se o sinal de reset (rst) foi activado ou se está no início da execução do programa. Se sim então as variáveis necessárias ao funcionamento do projecto são inicializadas.

```
 if rst= '1' then
        -- inicialização de valores
        matrix_completa:=false;
        i:=0;
             TROCAS:=false;
             FIM:=false;
             VAL<=0;
                col:=0;
                row:=0;
```

Caso contrário começaria o preenchimento da matriz.

Para que a matriz original não seja sempre a mesma (não seja estática mas sim dinâmica) foi criado um algoritmo pseudo-aleatório (com recurso a um signal std_logic_vector que era incrementado em cada ciclo de relógio e uma operação XOR) que preenche a matriz.

```
 elsif rising_edge(clk25) then
   -- incremento do vector que vai ser usado
   -- no algoritmo de inserção de valores na
matriz
   if vector="11111111" then
     vector:="00000000";
     else vector:=vector+1;
   end if;

   -- preenchimento da matriz pelo algoritmo
aleatório
   if matrix_completa=false then
        temporario:=((vector(7) & vector(0) &
vector(3)) xor (vector(1) & vector(6) &
vector(4)));
        if temporario="000" or temporario="010"
or temporario="101" then MATRIX(row)(col)<=0;
MATRIX_ORD(row)(col)<=0;
        elsif temporario="001" or
temporario="100" or temporario="110" then
MATRIX(row)(col)<=1; MATRIX_ORD(row)(col)<=1;
        else MATRIX(row)(col)<=2;
MATRIX_ORD(row)(col)<=2;
        end if;

        if((row=N-1) and (col=N-1)) then
          -- activação do sinal para que se possa
iniciar a ordenação
          matrix_completa:=true;
        elsif(col=N-1) then
          col:=0;
          row:=row+1;
        else col:=col+1;
        end if;
```

Quando a matriz estiver completa a variável matriz_completa passa a ter o valor true o que permite o início da ordenação da matriz.

Como disse anteriormente implementei um processo de ordenação baseado no algoritmo *bubblesort*, com as necessárias alterações:

```
-- ordenação da matriz
else
if FIM=false then
  if
contar(MATRIX_ORD(i))>contar(MATRIX_ORD(i+1))
then
       -- troca de linhas
            MATRIX_ORD(i)<=MATRIX_ORD(i+1);
            MATRIX_ORD(i+1)<=MATRIX_ORD(i);
            TROCAS:=true;
         end if;

         i:=i+1;
         -- se não houve trocas no fim de um
ciclo de iterações
         -- a matriz está ordenada, caso
contrário volta à posição inicial
         if i=N-1 then
            if TROCAS=false then
                  FIM:=true;
            else
                  TROCAS:=false;
                  i:=0;
            end if;
         end if;
       end if;
end if;
```

De notar que na linguagem VHDL (ao contrário das outras linguagens, como C, C++, entre outras) não é necessária uma variável *"temp"* para que a troca de linhas se processe.

Caso fosse necessário testar a ordenação da matriz com os 3 elementos (0, 1 e -) não seria necessário alterar no código e voltar a compilar o projecto, pois foi criado um sinal que muda dinamicamente quando se carrega num determinado botão da placa Trenz. Ao carregar neste botão o sinal VAL é alterado para um dos elementos de ordenação da matriz na seguinte ordem: 0, 1, -.

```
-- alteração do valor de VAL
elsif but_val='1' then
        if matrix_completa=true then
            if VAL=2 then VAL<=0;
                  else VAL<=VAL+1;
                  end if;
                  FIM:=false;
                  i:=0;
                  TROCAS:=false;
                  MATRIX_ORD<=MATRIX;
            end if;
```

De cada vez que o valor da ordenação é alterado e para que o processo de ordenação decorra normalmente é necessário inicializar algumas variáveis e também é preciso reescrever na matriz que vai ser ordenada com os valores da matriz original para que não se percam valores.

Se for necessário testar novas matrizes apenas é necessário carregar no botão de reset (rst) e então todas as variáveis são inicializadas e a matriz é preenchida então por esse algoritmo aleatório sendo sempre diferente.

Foi desenvolvido outro processo que imprime no monitor VGA. No canto superior esquerdo é imprimida a matriz original, nas coordenadas (1,17) é imprimido o valor em que a matriz é ordenada (0,1 ou -) e no canto superior direito é imprimida a matriz ordenada consoante o valor de ordenação.

```
--impressão da MATRIX (matrix original)
if (text_row<N and text_col<N) then
  if
MATRIX(to_integer(text_row))(to_integer(text_col)
)=1 then
      ROM_address      <=      "0001"      &
std_logic_vector(char_row);
      pixel                                <=
ROM_data(conv_integer(std_logic_vector(char_col))
);
  elsif
MATRIX(to_integer(text_row))(to_integer(text_col)
)=0 then
      ROM_address      <=      "0000"      &
std_logic_vector(char_row);
      pixel<=ROM_data(conv_integer(std_logic_ve
ctor(char_col)));
  else
      ROM_address      <=      "0010"      &
std_logic_vector(char_row);
      pixel<=ROM_data(conv_integer(std_logic_ve
ctor(char_col)));
  end if;
```

O código indicado acima é o código necessário para a impressão da matriz original no canto esquerdo do monitor VGA.

Para se imprimir um carácter no monitor é preciso "desenhá-lo" primeiro, isto é, como o monitor VGA só apresenta pixéis é necessário transformar o carácter num conjunto de pixéis através de uma matriz (neste caso também de tamanho 16x16). O bloco Char_ROM recebe um endereço e devolve os dados correspondentes a esse endereço. No código indicado em baixo, mostra-se a maneira de imprimir um 0 (zero) da maneira acima indicada:

```
my_ROM(0)  <= "0111111111100000";
my_ROM(1)  <= "1111111111110000";
my_ROM(2)  <= "1100000000110000";
my_ROM(3)  <= "1100000000110000";
my_ROM(4)  <= "1100000000110000";
my_ROM(5)  <= "1100000000110000";
my_ROM(6)  <= "1100000000110000";
my_ROM(7)  <= "1100000000110000";
my_ROM(8)  <= "1100000000110000";
my_ROM(9)  <= "1100000000110000";
my_ROM(10) <= "1111111111110000";
my_ROM(11) <= "0111111111100000";
my_ROM(12) <= "0000000000000000";
my_ROM(13) <= "0000000000000000";
my_ROM(14) <= "0000000000000000";
my_ROM(15) <= "0000000000000000";
```

Neste projecto apenas foi necessário "desenhar" 3 caracteres (o 0, o ! e o '-'), mas o array está definido para se poderem usar até 16 caracteres.

Há também de neste projecto se poder escolher a cor do fundo e dos caracteres. Há seis tipos de cores (R, RB, G, GB, B, BB), e pode-se escolher cores isoladas, ou combinações de cores. Neste projecto escolhi o preto como cor de fundo e verde como cor dos caracteres através do seguinte código:

```
-------------------------------------
-- output signals
blank  <= h_blank or v_blank;

-- fundo preto e letras a verde
r   <= '0' when blank= '1' else '0';
rb  <= '0' when blank= '1' else '0';
g   <= '0' when blank= '1' else pixel;
gb  <= '0' when blank= '1' else pixel;
b   <= '0' when blank= '1' else '0';
bb  <= '0' when blank= '1' else '0';
```

IV EXECUÇÃO DO PROJECTO

Depois de o projecto ser compilado, o ficheiro fpga.bit é criado. Através do programa TEProg.exe este ficheiro é enviado para a FPGA da placa Trenz, podendo aí ser executado.

Como se pode ver na figura 3, naquela execução o elemento de ordenação é o 0 (zero). Podemos ver a matriz original (a da esquerda) e à direita a matriz ordenada, onde ser verifica que a ordenação está correcta, pois a primeira linha corresponde à linha com menor número de zeros (0 zeros) e a última linha corresponde à linha com maior número de zeros (16 zeros).

V CONCLUSÃO

O tamanho da matriz usada neste projecto faz com que sejam ocupadas mais de 100% das slices da FPGA, o que pode provocar situações (embora muito raras) onde existam problemas de sincronismo. Caso a matriz seja mais pequena, por exemplo 6x6 ou 10x10, este problema

```
00-00111-10110-0                1111111111111111
--0110-11----010                ----------------
1-0-000--010-00-        0       --0110-11----010
0-01000-000-1010                -0-1-11-0-10-01-
1-0-01-100-10-10                -000--1-1101-1-0
-000--1-1101-1-0                00-01100---1111-
1111111111111111                1-01001-1111--00
11-00000010110-0                1-0-01-100-10-10
00-01100---1111-                1-11-1-0000-100-
1-01001-1111--00                010-1-00101-1-10
1-11-1-0000-100-                00-00111-10110-0
0000000000000000                1-0-000--010-00-
-0-1-11-0-10-01-                11-00000010110-0
010-1-00101-1-10                -----0000-0000-0
----------------                0-01000-000-1010
|----0000-0000-0                0000000000000000
```

Fig3–Possivel resultado da execução do projecto.

não acontece.

Apesar destes problemas, todos os objectivos propostos pelo professor foram cumpridos sendo introduzidos outros objectivos para que a execução do programa fosse mais atractiva: a matriz é preenchida dinamicamente, é ordenada correctamente, são mostradas no ecrã as matrizes original e a ordenada e é possível alteração o valor da ordenação visualizando a matriz ordenada segundo esse valor.

Mesmo assim há sempre lugar para futuros melhoramentos e optimizações, nomeadamente para tentar diminuir o número de slices ocupadas e para diminuir também o tempo de compilação.

Este projecto serviu para melhorar os conhecimentos da cadeira Computação Reconfigurável e interacção com a placa Spartan IIE da Trenz.

AGRADECIMENTOS

Agradeço ao Professor Valery Sklyarov e aos meus colegas de Computação Reconfigurável pela ajuda fornecida.

REFERENCES

[1] V. Sklyarov, material da disciplina Computação Reconfigurável, 2º semestre, WebCT.

[2] http://www.xilinx.com, ISE 5.2, Xilinx series FPGA.

[3] http://www.trenz-electronic.com

Desenvolvimento de uma biblioteca em VHDL para manuseamento de dados a partir de uma porta de série

André Monteiro

Resumo **- Este artigo descreve uma biblioteca desenvolvida em VHDL para uma FPGA (*Field Programmable Gate Array*) que permite recepcionar dados a partir da sua porta de série RS232, guardá-los na sua memória interna e visualizá-los simultaneamente num monitor VGA ligado à placa FPGA e no LCD da própria placa. Para enviar os dados pela porta de série é utilizado o PC, tendo o desenvolvimento da biblioteca sido feito em VHDL. Este artigo ilustra o funcionamento básico das entradas e saídas da placa Spartan IIE XC2S300E, assim como a comunicação de dados em série e a sua difusão num periférico de saída, neste caso o monitor VGA.**

Abstract **– This paper describes a VHDL-based library which allows to input data from serial port RS232, to save in internal memory and to visualize simultaneously data on a VGA monitor connected and LCD. The paper ilustrates the basic functionality of the input/output ports of the FPGA Spartan IIE XC2S300E, serial data comunication and data exchange with periphal devices.**

I. INTRODUÇÃO

A reconfiguração de *hardware* é uma área de particular interesse nos dias de hoje, não só pelo seu potencial mas pela variedade de diferentes soluções para os seus problemas. Deste modo, o seu enquadramento numa disciplina do curso revela-se bastante importante para uma formação inicial e suporte para novos desenvolvimentos.

Este projecto foi realizado na sequência da disciplina Computação Reconfigurável leccionada em Licenciatura em Engenharia de Computadores e Telemática, no 4.º ano, 2.º semestre, pelo Professor Valery Sklyarov [1].

O material da disciplina forneceu as bases necessárias para a linguagem de descrição de hardware VHDL, assim como a utilização do ambiente ISE 5.2 da Xilinx [2].

A. Objectivos

Apesar de subentendidos no título do artigo, os objectivos pretendidos com a execução deste trabalho podem-se resumir nos seguintes aspectos:

- adquirir experiência na elaboração de soluções práticas;
- conjugar a programação em VHDL com as limitações do hardware;
- lidar com o funcionamento das entradas e saídas da placa Trenz [3], mais especificamente a entrada RS232 e a saída VGA;
- utilizar e manusear o LCD incluído na placa Trenz [3];
- integrar a realidade do *hardware* com a criação e utilização de uma RAM na memória interna da FPGA.

B. Descrição global do trabalho

O projecto consiste no desenvolvimento na linguagem VHDL de um circuito que permite, num a primeira fase, a recepção de dados a partir da porta de série da placa Trenz. Após a recepção de dados, o caracter recebido é directamente enviado para o LCD, sob a forma de caracter de 8 bit. Simultaneamente, o caracter é guardado na memória Single Port Block RAM de 32 bit, da qual é retirado com um contador sucessivamente o valor. Este valor é convertido em caracter gráfico para o VGA através de uma ROM estática de caracteres. A ROM define números de 0 a 9 e letras de A a F. Caso não seja efectuada nenhuma recepção, é exibido o número 0. A saída VGA está estabelecida para uma resolução de 640 x 480 pixéis, e a recepção de dados é efectuada a uma taxa de 115200 baud.

O utilizador tem que usar um software adicional num PC para enviar dados para a placa, sendo esta a única intervenção que tem que realizar.

A disposição estrutural do hardware é elucidada pela figura 1.

II. ORGANIZAÇÃO ESTRUTURAL DO CÓDIGO EM VHDL

A. Estrutura básica do código

O código é constituído por seis módulos principais, descritos em VHDL, um ficheiro UCF (*User Constraints File*), um esquemático específico e outro global.

O UCF fpga.ucf indica a atribuição dos pinos existentes, assim como as definições a nível de sinais de relógio. Estão definidos dois relógios: um de 48MHz e outro de 25MHz específico para o monitor VGA.

O esquemático clk_res.sch efectua no relógio inicial as divisões necessárias para este poder ser utilizado nos vários componentes do programa. Estas divisões são

indispensáveis devido à menor velocidade dos componentes utilizados, de modo a serem visíveis os resultados.

O char_rom.vhdl é o módulo no qual está definida uma pequena ROM de demonstração que contém apenas 15 caracteres gráficos, cada um definido em 16 x 16 bit.

O lcd.vhd é o módulo que permite visualizar no LCD o caracter introduzido.

O RS_divider é o módulo onde se faz a divisão do relógio de modo a proporcionar uma taxa de recepção de 115200 baud na porta de série.

O RS_control é o módulo onde se efectua a recepção de dados, armazenando-os de 8 em 8 bit em caracteres.

O spblockram.vhd é um módulo que define uma RAM de 32 byte localizada na memória interna e que armazena os caracteres.

O vga.vhd é o módulo que controla o monitor, recebendo o caracter da RAM e enviando para o monitor o respectivo gráfico.

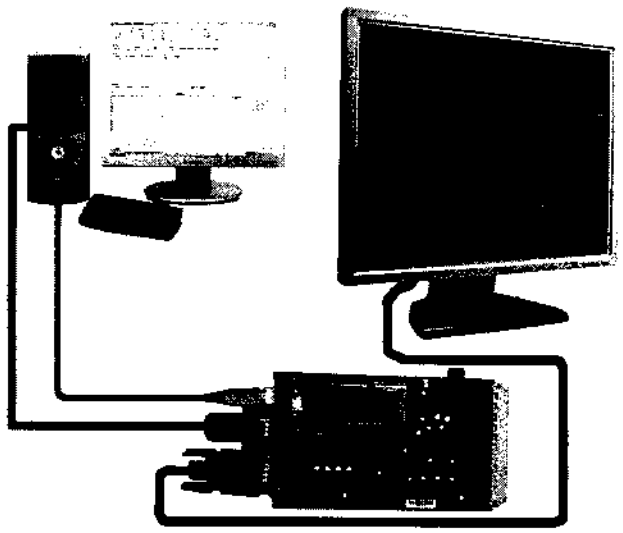


Fig. 1 – Disposição dos periféricos e ligações realizadas

III. ESTRATÉGIA DE SUPORTE

A. Recepção de dados através da porta série

Para a recepção de dados através da porta de série RS232, é necessário um cabo de série directo. O cruzamento dos pinos é necessário para se estabelecerem dois canais de transmissão unidireccionais (placa RxD – PC TxD; placa TxD – PC RxD). Como este cruzamente é feito na FPGA, não é necessário o uso de cabo cruzado.

A comunicação entre o PC e a placa de desenvolvimento é efectuada através de uma adaptação do protocolo standard RS232, já que apenas são utilizados dois pinos, RxD e TxD.

Para corrigir erros de transmissão em comunicações série é utilizado o método da comunicação assíncrona, em que são inseridas marcas no *stream* de bits a enviar. Deste modo, a transmissão tem que ser efectuada à mesma velocidade. É iniciada a sequência com o envio de um bit designado por *start bit* seguido de uma *stream* de 8 bits de dados. O bit menos significativo é enviado no início e o mais significativo no fim da *stream*. Por fim, segue-se o *stop bit* para indicar o fim da sequência. A figura 2 [4] ilustra esse aspecto.

Fig. 2 – Formato dos dados na comunicação rs232

Desta forma, a recepção de dados é efectuada sequencialmente, com se pode constatar no excerto transcrito.

```
process(clk, rst)

variable tmp, ind: integer;

begin
  if rst= '1' then
     tmp:=0; ind :=0;
  elsif falling_edge(clk) then
     if rs232in = '0' then ind := 1;
     end if;
   if (tmp >= 1) then
      if (tmp <= 8) then symbol(tmp-1) <= rs232in;
       end if;
    end if;
    if ind = 1 then tmp := tmp + 1;
    end if;
    if (tmp >= 9) and (rs232in = '1') then
        tmp := 0; ind := 0;
    end if;
  end if;
end process;
```

B. Controlo do LCD da placa

Para o controlo do LCD, existe um ficheiro VHDL que especifica a recepção de dados. Este ficheiro contém todos os procedimentos a efectuar sobre o CPLD e especifica também a saída do caracter para o LCD de 2 x 16 caracteres, no qual as duas linhas são preenchidas por constantes, exceptuando o caracter, que varia conforme a entrada. O código VHDL desenvolvido aqui é bastante similar ao [4] e [5].

C. Registo dos dados na RAM

A RAM utilizada é uma Single Port Block RAM, que inclui um relógio, um Write Enable, um endereço, uma entrada de dados e uma saída de dados. Quando o relógio se encontra numa transição ascendente e com o Write Enable activado, é armazenado o caracter recebido. O funcionamento está implícito no código apresentado:

```
architecture syn of spblockram is

 type ram_type is array (31 downto 0) of
std_logic_vector (7 downto 0);
 signal RAM : ram_type;
 signal read_a : std_logic_vector(4 downto 0);
```

```
begin
process (clk)
begin
  if (clk'event and clk = '1') then
       if (we = '1') then
          RAM(conv_integer(a)) <= di;
       end if;
       read_a <= a;
  end if;
end process;

do <= RAM(conv_integer(read_a));

end syn;
```

Existe um contador para os endereços da RAM no módulo do monitor VGA, que permite retirar da memória o caracter correcto. Esta memória permite armazenar 32 caracteres antes de ser reescrita pois uma vez chegado a 32 valores, o contador é reiniciado a zero. O código pode ser visto de seguida:

```
-- RAM address
process(clk25, rst)
  begin
       if rst= '1' then
        RAM_count <= "00000";
       elsif rising_edge(clk25)
         then   if   RAM_count   ="11111"   then
RAM_count <="00000";
       else RAM_count <= RAM_count + 1;
           end if;
       end if;
    RAM_address <= RAM_count;
  end process;

-- RAM data to respective rom address
process(result, rst)
  begin

    if rst = '1' then count <= "0000";
    else
     case result is
      when "00110000" = >count <= "0000";   -- 0
      when "00110001" => count <= "0001";   -- 1
      when "00110010" => count <= "0010";   -- 2
      when "00110011" => count <= "0011";   -- 3
      when "00110100" => count <= "0100";   -- 4
      when "00110101" => count <= "0101";   -- 5
      when "00110110" => count <= "0110";   -- 6
      when "00110111" => count <= "0111";   -- 7
      when "00111000" => count <= "1000";   -- 8
      when "00111001" => count <= "1001";   -- 9
      when "01000001" => count <= "1010";   -- A
      when "01000010" => count <= "1011";   -- B
      when "01000011" => count <= "1100";   -- C
      when "01000100" => count <= "1101";   -- D
      when "01000101" => count <= "1110";   -- E
      when "01000110" => count <= "1111";   -- F
      when others => count <= "0000";
     end case;
    end if;
end process;
```

D. Envio de caracteres para o monitor VGA

Como já foi referido, resolução usada foi de 640x480 pixéis, que serve para ilustrar o funcionamento do projecto. Também é possível utilizar resoluções mais elevadas, com uma taxa de varrimento de 60 Hz, mediante algumas alterações no módulo respeitante ao monitor.

A apresentação de caracteres no monitor requer uma transformação, já que o monitor apenas apresenta pixéis. Desta forma, é necessário converter cada caracter num conjunto de pixéis que simbolizem o caracter pretendido. Esta conversão é feita pelo módulo char_ROM, para o qual é enviado o caracter e devolvido o gráfico correspondente. A impressão no monitor é efectuada posteriormente e na posição escolhida. A posição inicial do caracter é escolhida com o referencial de linhas e colunas. Neste caso e nesta resolução a coluna 12 e a linha 14 indicam aproximadamente o centro do ecrã.

Também é possível definir uma cor para o caracter. Dentro das seis saídas (R, G, B, RB, GB, BB) do módulo basta definir uma combinação de cores. O excerto de código em baixo refere essa mesma potencialidade:

```
process(clk25)
 begin
       blank  <= h_blank or v_blank;

       if blank ='1' then
              r  <= '0';
              rb <= '0';
              g  <= '0';
              gb <= '0';
              b  <= '0';
              bb <= '0';
       else
              if (text_col < 15) then
                     r  <= pixel;
                     rb <= pixel;
                     g  <= '1';
                     gb <= '1';
                     b  <= '0';
                     bb <= '0';
              else
                     r  <= '1';
                     rb <= '1';
                     g  <= pixel;
                     gb <= pixel;
                     b  <= '0';
                     bb <= '0';
              end if;
       end if;
end process;
```

IV. INTERFACE AO NÍVEL DO SOFTWARE

A. Envio de dados através do software Terminal v1.9b

O programa Terminal [6] tem um funcionamento simples. Uma vez ligados os cabos, é necessário escolher a taxa de 115200 baud, 8 bit de dados, sem paridade, 1 stop bit, sem *handshake* e efectuar o "Connect". De seguida basta escrever o caracter pretendido, dentro da gama disponível, e carregar em "Send", como ilustra a figura 3. O caracter é enviado em 8 bit e exibido no LCD e monitor VGA simultaneamente.

Quando se efectua a ligação, é recebido o caracter "0", pois a saída rs232out foi definida estaticamente para esse valor. O projecto apenas está concebido para receber um caracter de cada vez, pelo que é desnecessário enviar mais que um, pois apenas o último é recepcionado. Qualquer

caracter fora da gama referida irá proporcionar um 0 no monitor VGA.

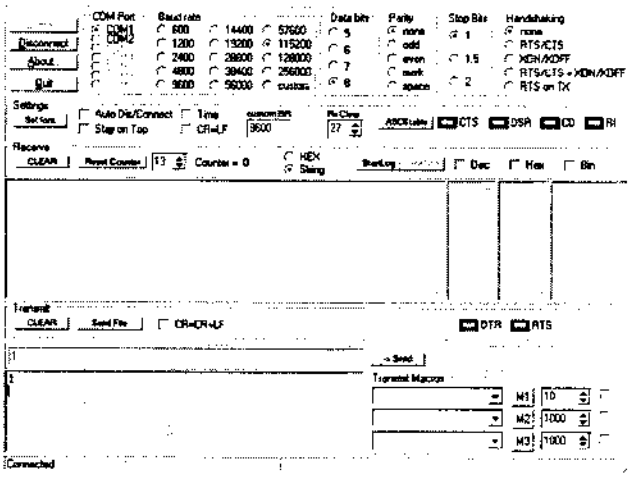


Fig. 3 – *Software* Terminal v1.9b: envio de dados pela porta série

B. Compilação VHDL e geração de bitstream

O código escrito que foi desenvolvido em VHDL tem que ser processado e sintetizado no Xilinx ISE [2] antes para poder ser utilizado na placa Trenz. Assim, é criado o *bitstream*, que através do programa TEProg.exe fornecido pela Trenz Electronics [3] é enviado directamente para a FPGA. O esquemático completo do circuito que apresenta a estrutura final do projecto pode ser visto na figura 4.

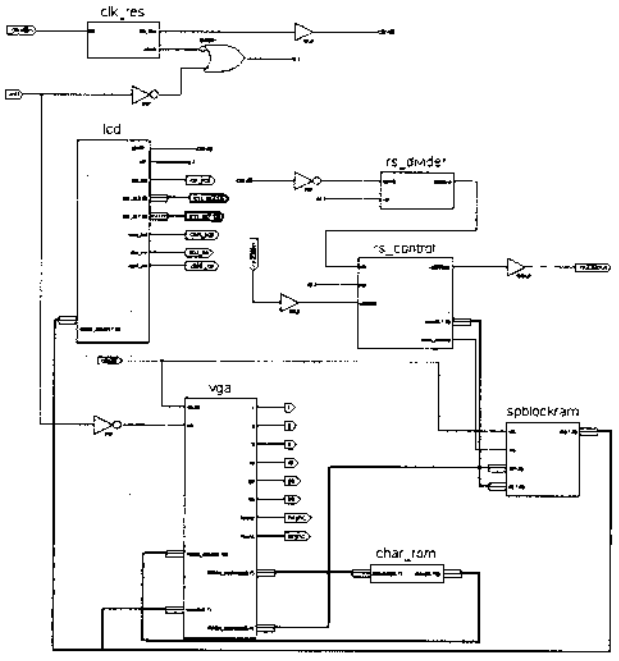


Fig. 4 – Esquemático completo do projecto

V. CONCLUSÕES

A realização deste trabalho prático tinha como ponto chave a aquisição de conhecimentos e princípios de funcionamento de um sistema de hardware reconfigurável.

Mais especificamente, foi estudada a interacção entre os vários componentes existentes e as potencialidades da placa Trenz [3], aliada ao desenvolvimento de código na linguagem VHDL.

As estratégias utilizadas no desenvolvimento deste projecto foram abordadas e clarificadas.

Uma das prioridades foi a optimização do funcionamento do programa, evitando operações demasiado complexas e reutilizando expressões para a minimização da utilização de recursos da FPGA.

O processo de comunicação RS232 foi o mais complicado, não pela dificuldade subjacente mas pelo facto de se difícil encontrar *software* ideal à realização do projecto, uma vez que nos dias de hoje a norma não é muito usada.

Um factor de sucesso foi a organização do projecto em 3 fases. A primeira fase consistiu em receber os dados da porta de série e mostrá-los no LCD da placa. A fase seguinte teve como objectivo visualizar esses mesmos dados e exibi-los no monitor VGA. Por fim, a última fase foi a integração de uma RAM no circuito já construído e funcional.

Ainda que os conhecimentos tivessem que ser aprofundados relativamente aos adquiridos na disciplina do âmbito do projecto, a aprendizagem foi progressiva e sustentada nas referências indicadas, revelando-se proveitosa. Todas as especificações do projecto foram cumpridas e de uma forma optimizada, pelo que o projecto foi bastante bom para aumentar o conhecimento e experiência em sistemas reconfiguráveis e interagir com a placa Spartan IIE da Trenz [3].

O projecto que inclui todas as partes apresentadas acima está disponível na WebCT [1].

AGRADECIMENTOS

O autor agradece ao Professor Valery Sklyarov e a Iouliia Skliarova a orientação e a ajuda prestada na realização deste artigo.

REFERÊNCIAS

[1] http://webct.ua.pt, 2º Semestre, disciplina Computação Reconfigurável.

[2] http://www.xilinx.com, ISE 5.2, Xilinx series FPGA.

[3] http://www.trenz-electronic.com.

[4] R. Costa, J. Limas, I. Oliveira, "Desenvolvimento de uma calculadora baseada numa FPGA e num *touch panel*", Electrónica e Telecomunicações, vol. 3, n. 9, Setembro 2003.

[5] B. Monteiro, "Desenvolvimento de um circuito aritmético em VHDL", Electrónica e Telecomunicações, Janeiro 2004.

[6] http://bray.velenje.cx/avr/terminal, Terminal v1.9b

Desenvolvimento de um circuito em *Handel-C* para representação de grafos

Gustavo Corrente

Resumo **– Este artigo visa a descrição do circuito criado para a representação de grafos com 25 vértices, baseado numa *FPGA Spartan-II XC2S200*, *que faz parte da placa* de desenvolvimento *RC100*, produzida pela *Celoxica*. Para a interacção com o circuito foram usados o rato e o monitor. A especificação do circuito foi na linguagem *Handel-C* num ambiente *DK2*, ambos desenvolvidos pela *Celoxica* [1].**

Abstract **– This paper describes a circuit to be designed for processing graphs with up to 25 vertices. The circuit has been implemented in a *FPGA Spartan-II XC2S200* (the development board *RC100* of *Celoxica*). A mouse and a monitor have been used for interactions with the circuit. The design has been performed in Celoxica [1] *DK2* environment from a specification in *Handel-C*.**

I. INTRODUÇÃO

O projecto aqui descrito, foi desenvolvido no âmbito da disciplina Computação Reconfigurável [2], do 4º ano do Curso de Engenheria de Computadores e Telemática, leccionada pelo professor Valery Sklyarov.

A. Objectivos

O presente projecto visa, fundamentalmente, os seguintes objectivos:

- Aprender a linguagem *Handel-C* [3]
- Aprender a criar um modelo adequado à solução pretendida e ao *hardware*;
- Responder com sucesso à especificidade dos dispositivos periféricos tais como: rato e monitor VGA;
- Interiorizar técnicas de optimização e de reutilização dos recursos da *FPGA*;

B. Descrição global do projecto

O circuito desenvolvido permite a representação de grafos com 25 vértices, de uma forma muito particular. Cada vértice tem uma cor, mas vértices ligados entre si, não podem ter a mesma cor. A atribuição das cores é de uma forma optimizada, sendo o objectivo de colorir os vértices com o menor número de cores possível de modo a satisfizer a regra atrás descrita. Também foi desenvolvido um programa para o *PC*, em C++/QT [4], para facilitar a geração de ficheiros para teste.

C. Características principais da placa RC100

A placa *RC100* [5] tem como componente reconfigurável principal a *FPGA* da família *Spartan-II* da *Xilinx*, com 200.000 portas de sistema.
Esta *FPGA* tem ligação directa com:

- Oscilador de cristal de 80MHz;
- Dois blocos de memória *RAM* síncrona estática (*SSRAM*) com 256K palavras de 36bits cada;
- *Flash RAM* de tamanho 64Mbits;
- Vídeo *DAC* (VGA 24Bits);
- Descodificador da entrada de vídeo;
- Porta *PS/2;*
- *LED*s;
- 2 *Displays* de 8 segmentos;
- Um barramento de expansão.

II. ORGANIZAÇÃO BÁSICA DO PROJECTO

Este projecto dividiu-se em duas partes: o circuito desenvolvido em *handel-C* (*FPGA side*) e um programa devolvido em C++/QT (*PC side*).

Através do programa do *PC* podemos criar vários ficheiros de teste, que podem ser transferidos para a memória *flash* da *FPGA*. Essa transferência pode ser efectuada através do programa *FTU*. Esta arquitectura está ilustrada na *Figura 1*.

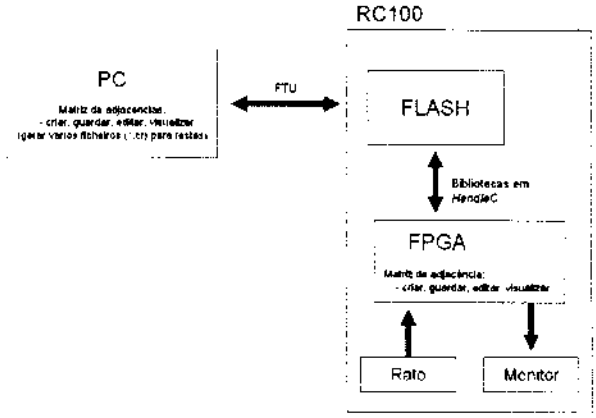


Figura 1 – Arquitectura global do projecto

Já na *Figura 2* podemos ver os dispositivos usados que possibilitaram a visualização e edição do grafo sob a forma da sua matriz de adjacência.

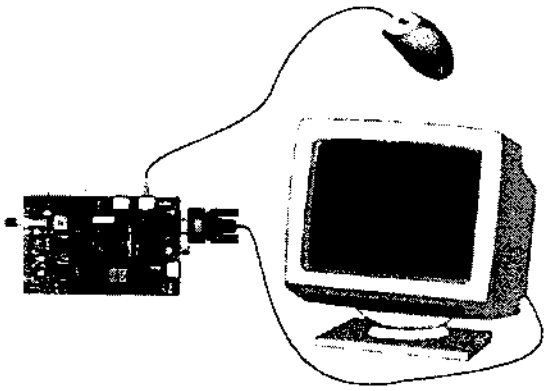


Figura 2 – Placa RC100 e dispositivos usados para a interacção

A *Figura 3* demonstra o funcionamento do *software* quer a nível da visualização, quer a nível de interacção com o utilizador.

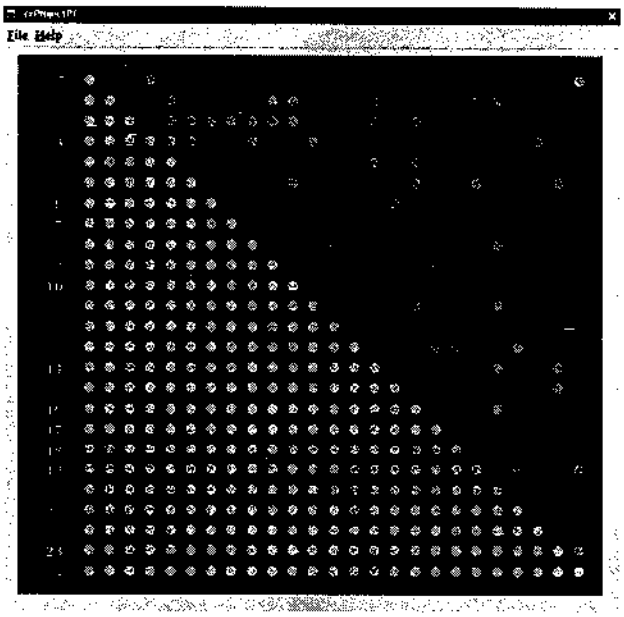


Figura 3 – Programa desenvolvido para o PC.

III. ESTRATÉGIAS ADOPTADAS

Para tirar partido de uma das principais características de *FPGA*, de se poder executar mais de uma instrução em simultâneo, foi adoptada a seguinte estratégia. Partilhar as variáveis (edgeMatrix, colorVector) entre os vários processos que estão a correr em paralelo. O esquema de funcionamento e os principais blocos de funcionamento estão resumidos na **Figura 4**.

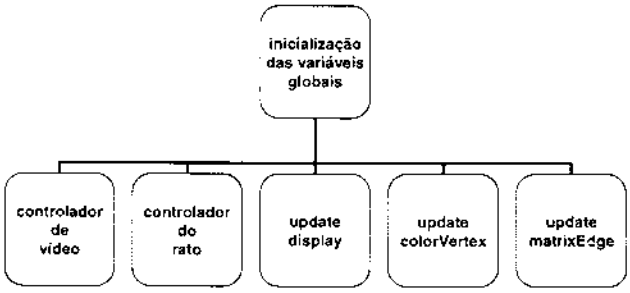


Figura 4 – Principais blocos de funcionamento

No *main* temos a correr em paralelo, cinco procedimentos expansíveis *in-line*: controlo do vídeo e do rato, ambas são bibliotecas [6] da *RC100*, controlo do *display*, controlo do vector onde está armazenada a informação sobre a cor de cada vértice e controlo da matriz de adjacências. Este funcionamento encontra-se ilustrado no bloco de código da **Figura 5**.

```
par
{
    RC100VideoDriver(&Video);
    RC100PS2MouseDriver(&Mouse, RC100_MOUSE_PORT);
    Display( &Video , &Mouse );
    UpdateColorVector();
    UpdateMatrixEdge( &Mouse );
}
```

Figura 5 – Bloco de código

No bloco de código descrito na **Figura 6** podemos ver esboço do algoritmo usado para seleccionar a cor de cada vértice.

```
for( int y = 0 ; y < 25 ; y++ )
    for( int x = y+1 ; x < 25 ; x++ )
        if( edgeMatrix[x][y] == '1' )
            if( colorVector[y] == colorVector[x] )
                colorVector[x]++;
```

Figura 6 – Bloco de código

Para o armazenamento da informação relativa à cor de cada vértice for usado um array *25* elementos (porque os grafos tem 25 vértices) de *unsigned int 24* porque cada *pixel* tem uma profundidade de 24 *bits*. A representação da matriz de adjacências do grafo é implementada através de um *array* de 300 elementos de *unsigned int 1*. Os 300 elementos são suficientes visto que a diagonal principal e o triângulo inferior não são usados, visto que representam arestas repetidas e arestas de um vértice para ele mesmo. A actualização da cor dos vértices é feita através do procedimento updateColorVertex que com base na variável edgeMatrix (implementação da matriz de adjacências). Já a edgeMatrix é alterada com base no *click* do rato, com um simples *click* podemos alternar entre 0 e 1, ou seja, alternamos entre "*há aresta*" e "*não há aresta*".

IV. CONCLUSÕES

A linguagem *Handel-C* revelou-se poderosa, bem como fácil de aprender. Apesar de uma grande optimização nas estruturas de dados, pôde-se comprovar que a implementação de memórias e estruturas de dados grandes devem ser postas nos blocos de *RAM* externos à *FPGA*, porque quando estão implementados na *FPGA* ocupam de uma forma exponencial as *slices* disponíveis. O uso da memória *RAM* iria contribuir para uma menor utilização dos recursos da *FPGA*, que de 89% passaria para 17% de utilização de *slices*.

AGRADECIMENTOS

O autor agradece ao Professor Valery Sklyarov e pela ajuda prestada na elaboração deste artigo.

REFERÊNCIAS

[1] Site: http://www.celoxica.com, on-line Agosto de 2004

[2] Site: http://webct.ua.pt, 2º Semestre, disciplina Computação Reconfigurável, on-line Agosto de 2004

[3] "Handel-C Language Reference Manual", Celoxica, 2003

[4] Site: http://www.trolltech.com, on-line Agosto de 2004

[5] "RC100 Hardware Manual", Celoxica, 2003

[6] "RC100 Function Library Reference", Celoxica, 2003

Plataforma de Gestão de Informação

Ana Rita Santos, Carlos André Sousa,

Joaquim Sousa Pinto, Lídia Oliveira Silva*, A. Manuel de Oliveira Duarte

*Departamento de Comunicação e Arte da Universidade de Aveiro

Resumo **– Este artigo, tem como propósito apresentar o sistema de Gestão de Projectos desenvolvido para o Programa PRAI- Programa Regional de Acções Inovadoras promovido pela Comissão de Coordenação e Desenvolvimento da Região Centro. Este sistema, integrado na Plataforma SITE, tem como principal objectivo a disseminação de informação dos projectos associados às várias entidades parceiras do Programa PRAI, através de um sistema de divulgação e disponibilização de informação de fácil interactividade e compreensão.**

Para além da contextualização do sistema no Programa PRAI, é feita uma descrição das funcionalidades e das áreas de acesso que se encontram disponíveis no sistema, tanto na área de BackOffice como na área de FrontOffice).

Por último, é feita uma validação do sistema onde é dado maior ênfase ao papel desta Plataforma no contexto da comunidade académico-científica e no âmbito do projecto SITE.

Abstract **- This paper presents a Project developed for Prai -*Programa Regional de Acções Inovadoras*, a Platform of Project Management. This Platform has as main objective the creation of a system that enables the diffusion and accessibility of information in an easy, interactive and understandable way.**

The initial part of this article concludes the presentation and describes the context of this Platform in the project SITE - *Sistema Interactivo para a Inovação e Transferência de Tecnologia.*

A description of the functionalities and areas of access that we can find available in this system, like the BackOffice and FrontOffice areas, is made in the following topics .

Finally, an overall analysis of the system is performed giving more emphasis to the purpose of this Platform in the context of the academic-scientific community as well as in the scope of project SITE.

I. INTRODUÇÃO

O objectivo desta plataforma, constituída por um conjunto de ferramentas em ambiente Web, é proporcionar a divulgação de informação e a interacção entre os utilizadores deste sistema.

Esta plataforma foi desenvolvida no âmbito do Programa Regional de Acções Inovadoras - **PRAI** e encontra-se incorporada no **SITE** - Sistema Interactivo para a Inovação e Transferência de Tecnologia.

O aparecimento desta Ferramenta surge como a realização de um dos principais objectivos a que o SITE se propõe: apresentação de uma plataforma telemática capaz de integrar sistemas que têm como função catalogar, armazenar e disponibilizar a informação e a produção científica gerada pelo sistema científico.

A plataforma desenvolvida pretende assim fazer com que o utilizador deste sistema se depare com um mecanismo de manipulação de informação dos projectos associados ao PRAI de fácil interactividade e compreensão.

Como se pode ver na figura 1, para aceder à Área dos projectos do PRAI disponíveis no SITE temos que clicar sobre o menu PRAI disponível no endereço http://gsbl.det.ua.pt/site/.

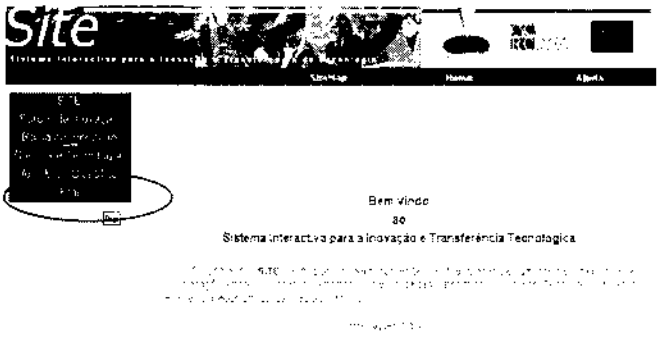


Fig. 1 - Página inicial do SITE

II. ARQUITECTURA DA PLATAFORMA DE GESTÃO DE PROJECTOS

Esta plataforma foi desenvolvida tendo em vista a criação de um sistema de informação que as entidades pudessem utilizar como um espaço comum e que permitisse uma fácil gestão da informação aí existente.

Neste espaço está disponível um conjunto de informações e ferramentas que não só permitem publicitar, mas também gerir informação dos projectos em curso, permitindo assim a pesquisa da produção cientifica gerada no seio das Instituições de Investigação(II).

O desenvolvimento da Plataforma de Gestão de Projectos foi efectuado numa arquitectura de 3 camadas:

- Interface - Cliente (HTML, JavaScript, CSS);
- Processamento – Servidor WEB (ASP.NET, XML);
- Dados – Servidor SQL (Stored Procedures, Views e Base de dados);

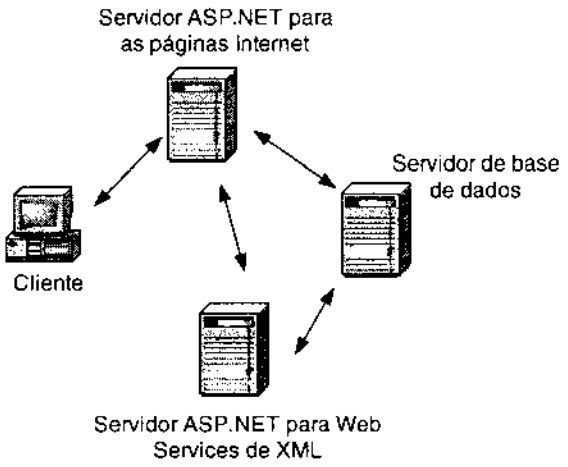


Fig. 2 - Descrição da Arquitectura do Sistema

Para plataforma WEB, utilizamos a aplicação do Internet Information Server (IIS) da Microsoft, dado o seu suporte nativo à tecnologia dinâmica de páginas WEB (ASP.NET) e ser da utilização do servidor do Grupo de Sistemas de Banda Larga (GSBL).

Na camada cliente foi desenvolvida uma interface com os utilizadores cujo objectivo principal foi obter um *design* simples, consistente e de fácil interactividade.

A camada de processamento, encontra-se no servidor WEB. Nesta camada é processada a autenticação dos utilizadores, o acesso à base de dados, e o processamento de toda a informação da Plataforma.

A camada Dados encontra-se no servidor SQL. Esta é responsável pelo armazenamento de todos os dados existentes no PRAI. O servidor utilizado foi o MS SQL Server 2000.

Para o desenvolvimento desta Plataforma de divulgação de Informação foram desenvolvidos 2 sistemas distintos:

- **Sistema de "FrontOffice"** (Área do utilizador comum) onde é possível a visualização da informação relativa aos projectos do PRAI.

- **Sistema de "BackOffice"**- Neste sistema está incluída uma área para a administração (que consiste na possível introdução de novos projectos, em apagar ou modificar projectos) e outra área destinada aos Responsáveis dos projectos (onde é dada ao responsável a possibilidade de alterar os dados dos projectos que lhe são atribuídos).

III. APRESENTAÇÃO DA PLATAFORMA DE GESTÃO DE PROJECTOS

Uma das Áreas incorporadas na Plataforma de Gestão de Informação é o Sistema de FrontOffice, área que é destinada ao utilizador comum

Nesta camada, o utilizador comum não tem qualquer poder de interacção na criação, modificação ou eliminação de projectos, pode apenas visualizar informação. Sendo assim, este utilizador comum situa-se no nível mais baixo da Hierarquia da plataforma desenvolvida para o PRAI (ver figura 3).

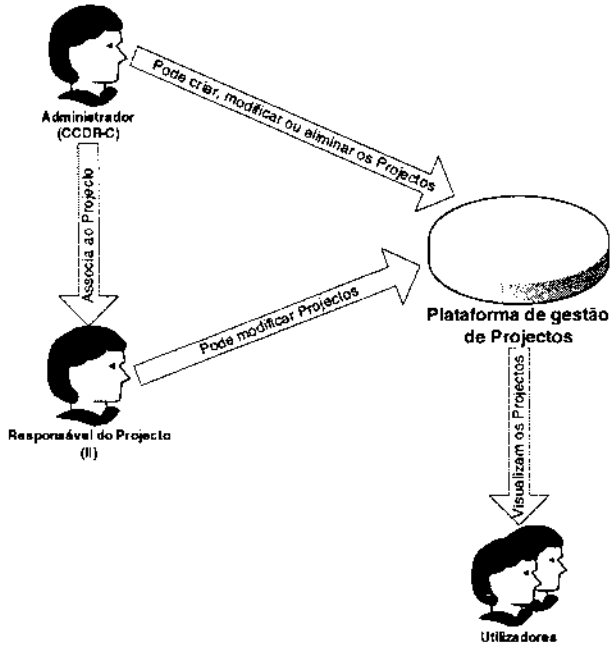


Fig. 3 - Esquema da estrutura hierárquica da Plataforma desenvolvida

O Sistema de BackOffice, que constitui a parte não visível para o utilizador, é composto por 2 áreas distintas: a área do Administrador e a do Responsável do Projecto. Para o Administrador ou o Responsável acederem ao

mecanismo de autenticação que permite aceder às respectivas áreas devem clicar sobre o Link 'Login'.

Como Entidade responsável pela gestão dos projectos contidos nesta Plataforma temos o Instituto de Investigação (ii), unidade responsável pelos projectos de investigação que decorrem na Universidade de Aveiro e pela gestão dos projectos PRAI nesta Universidade.

Cada Responsável do Projecto pode ter um (ou mais) projecto (s) associado (s). O Responsável do Projecto depois de autenticar o seu projecto, poderá proceder à sua modificação. Sendo assim, o Responsável situa-se no nível intermédio da Hierarquia da plataforma desenvolvida para o PRAI, visto que não tem o poder de criar ou apagar projectos (ver figura 3).

Como Administrador desta Plataforma temos a Comissão de Coordenação e Desenvolvimento Regional do Centro (CCDR-C).O Administrador é o utilizador que, depois de se autenticar, pode criar, modificar ou apagar informação relativa aos Projectos. O Administrador tem assim a seu cargo a gestão da Plataforma, o que faz com que este se situe no nível mais alto da Hierarquia da plataforma desenvolvida para o PRAI (ver figura 3).

A. ÁREA DO UTILIZADOR COMUM

Neste tópico é descrito o Sistema de FrontOffice, isto é, a área do utilizador comum.

Para um primeiro contacto com o sistema, recomenda-se a utilização desta área.

O utilizador comum pode apenas aceder à Área dos projectos do PRAI disponíveis no SITE. Para isso tem apenas que clicar sobre o menu PRAI disponível no endereço http://gsbl.det.ua.pt/site/.

Seguidamente surge a lista dos projectos que estão associados ao PRAI com a informação relativa ao Nome do projecto, Responsável e Instituição (ver figura 4).

Para obter uma descrição mais pormenorizada de cada projecto podemos clicar no botão de informação ( ) ou no link de cada projecto.

B. ÁREA DE BACKOFFICE

B.1 - ÁREA DO ADMINISTRADOR

Nesta secção é descrita uma das áreas do Sistema de BackOffice, a Área do Administrador.

Como Administrador do Sistema temos, como atrás referido, a CCDR-C, que tem associada um 'Login' e uma 'Password'.

Depois da comprovação de que a pessoa em causa é o Administrador surge um menu constituído por três opções(ver figura 5):

- Criar novo projecto;
- Modificar projecto
- Apagar projecto.

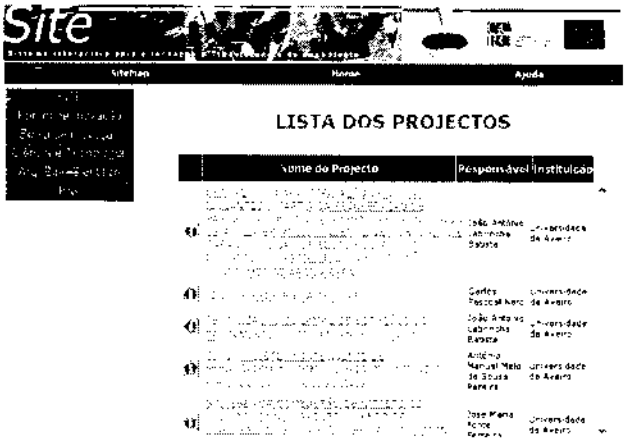


Fig. 4 - Página inicial da Área do utilizador comum

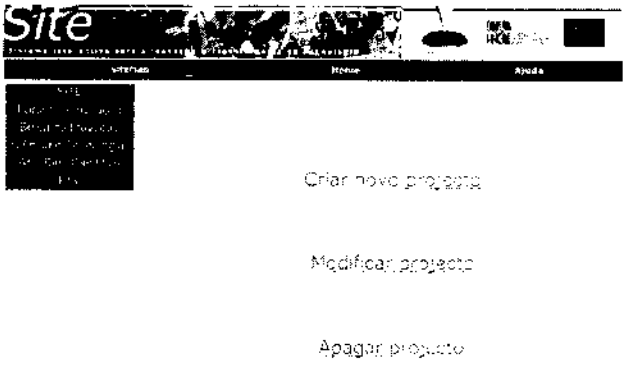


Fig. 5 - Menu da Área do Administrador

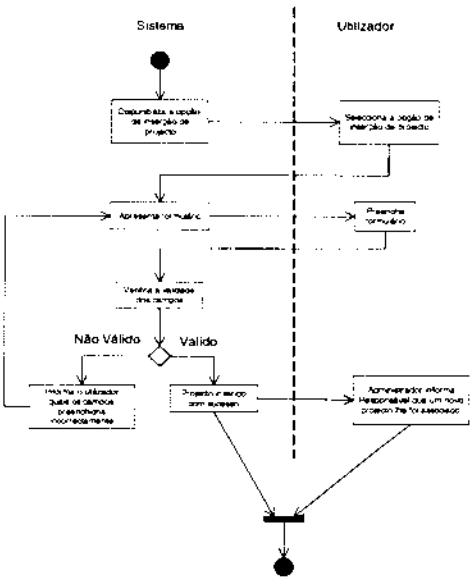


Fig. 6 - Diagrama de Actividades para a Criação de um Projecto

Na opção ***Criar novo projecto*** o Administrador cria um novo projecto, atribui um Responsável ao mesmo e cria um Login e Password que futuramente permitem ao Responsável o acesso à Área do Projecto que lhe foi associado.

O Administrador pode também preencher neste formulário outros dados referentes ao projecto criado (como por exemplo os Parceiros Envolvidos, Sumário do Projecto,...)

O Administrador depara-se no mecanismo de criação de um novo projecto com a opção de associar um novo Responsável ao projecto ou associar um Responsável que já tenha outro(s) projecto(s) associado(s)

No passo final da criação de um projecto o Administrador deve avisar o Responsável do Projecto que um novo projecto lhe foi associado. O Responsável deve de seguida preencher um formulário onde se faz a descrição pormenorizada do projecto e assim completar o processo iniciado pelo Administrador.

Na figura 6 está representado o diagrama de actividades para a criação de um Projecto.

A 2ª opção que existe no menu do Administrador é a de ***Modificar Projecto***.

Nesta opção é dada ao Administrador a possibilidade de modificar o Nome do Projecto, o seu Responsável, o e-mail e o Login associados ao projecto.

Caso o Login seja modificado o Administrador deve avisar o Responsável do Projecto, caso contrário este não consegue aceder ao projecto que lhe está associado.

Foi também criado um mecanismo para a modificação do Responsável onde este pode seleccionar um Responsável que já tenha projectos associados ou introduzir um novo Responsável..

A 3ª opção que surge no Menu do Administrador é a de ***Apagar Projecto***. Escolhendo essa opção surge uma lista de todos os projectos existentes (ver figura 7). Seleccionando o projecto que se pretende eliminar o Administrador é confrontado com a confirmação da eliminação do projecto.

O Administrador é vital para o bom funcionamento e expansão desta plataforma pois este é o responsável pela introdução de novos Projectos, actualização de informação e eliminação de Projectos no Sistema de divulgação de Informação, visto que é este quem tem permissões para efectuar tais modificações nos Projectos do Sistema de divulgação de Informação.

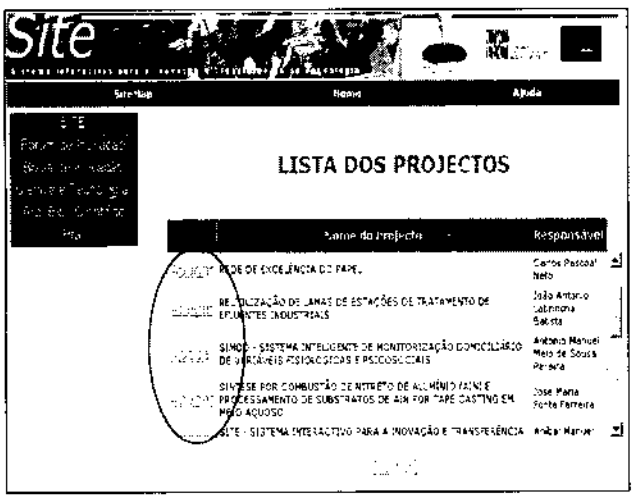


Fig. 7 - Mecanismo de eliminação de projectos

B.2 - ÁREA DO RESPONSÁVEL DO PROJECTO

A 2ª área que constitui o sistema de BackOffice é a área do Responsável do Projecto.

Depois de entrar na Área de Autenticação, o Responsável deve introduzir o Login e a Password do projecto que pretende efectuar alterações. É de relembrar que este Login e Password foram fornecidos ao Responsável pelo Administrador.

Depois da comprovação de que se acedeu a um Projecto (através da introdução do Login e Password correctos) o Responsável do Projecto visualiza uma página onde está descrito o projecto a que este acedeu.

Seguidamente o Responsável do Projecto pode aceder a uma página onde pode modificar todos os dados do projecto que lhe é associado excepto o Nome do Projecto, Investigador Responsável e Login do Projecto (estes são modificados apenas pelo Administrador).

Após a modificação do Projecto este fica visível na página cujo endereço é http://gsbl.det.ua.pt/site/Projectos_prai/projectos_prai.aspx .

Na figura 8 está representado o diagrama de actividades referente à modificação de um projecto para os actores Administrador e Responsável do Projecto,pois ambos têm como funcionalidade das áreas respectivas a possível modificação da informação relativa um projecto.

Pelo que foi descrito anteriormente verificamos que o Responsável do Projecto é vital para o bom funcionamento, nomeadamente para a actualização da informação nesta plataforma, visto que é este quem tem permissões para efectuar modificações nos Projectos do Sistema de divulgação de Informação.

IV. VALIDAÇÃO DO PROJECTO

A CCDR-C dispõe de um sistema de informação

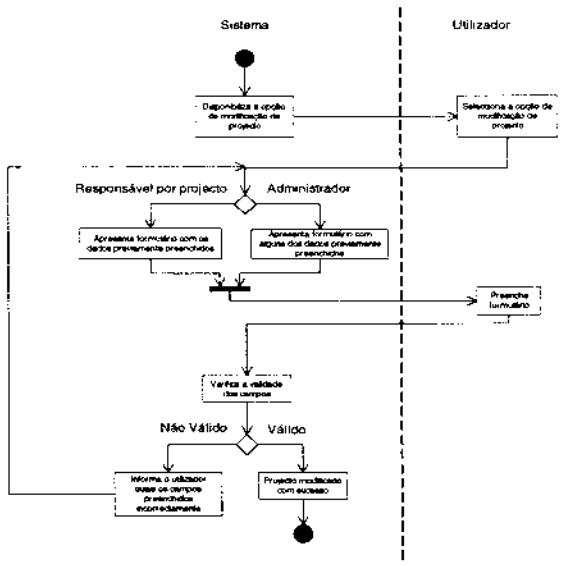


Fig. 9 - Diagrama de Actividades para a Modificação de um Projecto

autónomo que disponibiliza um conjunto de serviços on-line através do seu *site*. Neste sistema, apresenta-se uma "Bolsa de Inovação Regional", uma ferramenta especificamente criada com o objectivo de promover a inovação ao nível regional e encorajar a cooperação entre o tecido empresarial e os Institutos de Investigação promovendo a transferência tecnológica, e acima de tudo facilitar a difusão do conhecimento entre regiões .

A Plataforma de Gestão de Informação do PRAI pretende apresentar-se como um Sistema que mantém um conjunto de informações actualizadas e disponíveis para serem consultadas por todas as empresas, Associações Empresariais, CCR-C, e todas as outras Entidades que necessitem, de consultar informação relativa aos projectos do PRAI, revelando-se assim fundamentalmente como um centro de contactos de extrema importância para o dinamismo da *interface* entre a comunidade Académica e Empresarial da Região Centro.

V. CONCLUSÕES

Foram atingidos os objectivos de tornar o protótipo actual desta Plataforma num mecanismo de manipulação de informação dos projectos associados ao PRAI de fácil interactividade e compreensão. Para isso, tivemos o cuidado de não sobrecarregar as páginas desta plataforma com aspectos acessórios, não fundamentais para o sucesso das operações a realizar.

Assim, um utilizador com pouca "literacia informática" consegue perfeitamente entender e utilizar o mecanismo desenvolvido. É também de notar que este mecanismo foi pensado para alojar não só os projectos relativos ao PRAI mas futuramente também englobar outros projectos.

Verificamos assim a importância da inclusão desta ferramenta de administração e de divulgação de informação (que tem como função catalogar, armazenar e disponibilizar a informação e a produção científica gerada pelo sistema científico) no SITE: através desta o projecto SITE incrementa substancialmente o seu contributo para o desenvolvimento de um sistema de informação com características adequadas aos requisitos da comunidade académico-científica e para o fomento da transferência de conhecimento e cooperação entre o sistema científico (Universidades, Institutos, Centros Tecnológicos, etc.) e o tecido económico.

Pela análise atenta das vantagens, desenvolvimentos possíveis e da realidade existente no SITE, concluímos que a continuidade deste projecto se torna quase uma "obrigação" perante a precária situação do panorama actual.

AGRADECIMENTOS

Os autores deste artigo agradecem a todos aqueles que contribuiram para o desenvolvimento do SITE, mais especificamente da Plataforma de Gestão de Projectos, em especial ao Prof. Dr. Manuel de Oliveira Duarte, Prof. Dr. Joaquim Sousa Pinto, Prof. Dra. Lídia Oliveira, Miguel Oliveira, Pedro Beça, Nuno Carvalho e a toda a equipa do Grupo de Sistemas de Banda Larga .

REFERÊNCIAS

[1] W3schools Homepage, http://www.w3schools.com, Julho de 2003

[2] Msdn, HomePage, http://msdn.microsoft.com

[3] SQLTeam, HomePage, http://www.sqlteam.com

[4] Vieira, João, *Programação com ASP.NET*, volume I, FCA

[5] Mauro Nunes, Henrique O'Neill ,*Fundamental de UML*, FCA

[6] John Sharp, Jon Jagger, Microsoft Visual C# .NET Step by Step,Microsoft,2003

Aprend.e – Sistema integrado de formação e aprendizagem

Pedro Beça*, Miguel Oliveira*, A. Manuel de Oliveira Duarte

* Escola Superior Aveiro Norte

Resumo **– Este artigo pretende demonstrar o impacto e relevância da Plataforma Aprend.e como instrumento de dinamização do Programa Aveiro Norte e da Escola Superior Aveiro Norte.**

Esta Plataforma surge inicialmente como uma ferramenta de gestão e administração do Programa Aveiro Norte, cujo objectivo é simplificar e automatizar os processos relacionados com todas as actividades formativas do programa. Assim, com o decorrer de actividades formativas, as tarefas de gestão orçamental inerentes ao processo de formação tornaram-se mais simples e funcionais, tendo como impacto imediato a redução de processos burocráticos e a diminuição significativa do tempo dispensado nestas actividades . Esta plataforma além do apoio administrativo dado pretende dar apoio às actividades formativas, tanto aos formadores como formandos[R2]. Este apoio reflecte-se na disponibilização aos formandos, por parte dos formadores, da planificação das disciplinas, dos sumários, bem como os materiais de suporte às aulas. Como os graus de literacia informática dos formadores são variados, o sistema de apoio aos formadores foi estruturado de modo à não existência de pré-requisitos especiais na sua utilização, deste modo a própria disponibilização dos materiais de apoio às aulas é feita com um simples sistema de upload de ficheiros.

I. INTRODUÇÃO

A plataforma Aprend.e pretende ser um sistema integrado de formação e aprendizagem que suprima as necessidades dos formadores, alunos e pessoal administrativo, e que potencie o uso das tecnologias de informação por parte de todos os elementos que constituem o Programa Aveiro Norte.

No âmbito deste projecto foram desenvolvidas um conjunto de ferramentas em ambiente Web para proporcionar o máximo de informação e interactividade a todos os elementos envolvidos no programa. Assim, além de todo um leque de informações relativas aos vários cursos, os interessados podem ainda recorrer a uma secretaria virtual, onde têm desde logo a possibilidade de realizar algumas acções que até então só seriam possíveis junto da instituição em causa.

A existência deste serviço on-line tem enumeras vantagens para os seus utilizadores, na medida em que permite uma maior flexibilidade temporal e espacial, uma vez que está disponível 24 horas por dia e em qualquer ponto de acesso à Web.

Assim, procedeu-se, à informatização e optimização de processos relacionados com as actividades do Programa Aveiro Norte, através do desenvolvimento de ferramentas de BackOffice para o Aprend.e.

O sistema encontra-se disponível em :

http://www.aveiro-norte.ua.pt/secretaria virtual

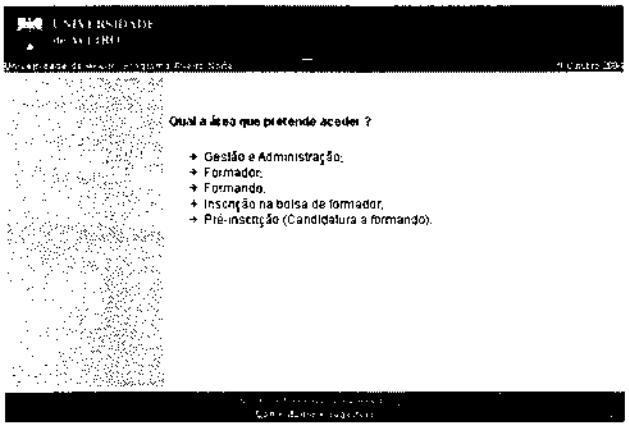


Fig. 1 – Página inicial do Aprend.e

II. ACESSIBILIDADE E APRESENTAÇÃO

O desenvolvimento de aplicações para Web, pressupõe que o uso desta tecnologia seja do conhecimento geral dos utilizadores a quem esta se destina. Assim, é expectável que as pessoas sejam minimamente capazes de interagir com o computador e seus periféricos, pelo que é apenas exigido o mínimo de literacia informática para o uso desta aplicação. Para aceder a esta aplicação basta ter acesso à Internet e dirigir-se ao sítio http://www.aveiro-norte.ua.pt

e aí clicar sobre a opção "Secretaria virtual" que se encontra no menu (Figura 2).

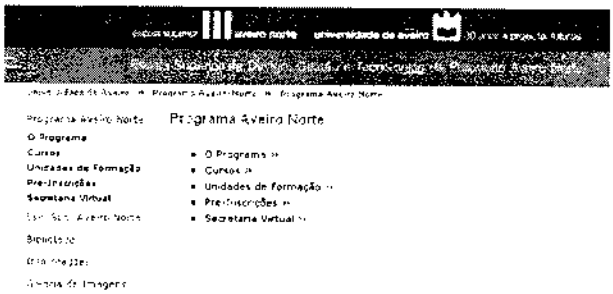


Fig. 2 – Acesso ao site do Programa Aveiro Norte e à Secretaria Virtual

Tendo em conta que o serviço é uma secretaria virtual, cada pessoa terá de dirigir-se à área que mais lhe convém. Para tal, existem diferentes acessos:

- Gestão e Administração
- Formador
- Formando
- Inscrição na bolsa de formador
- Pré-inscrição (candidatura a formando)

III. SISTEMA DE BASE DE DADOS

Todo sistema tem como suporte informático uma base de dados relacional, onde toda a informação disponibilizada é armazenada. Esta base de dados foi concebida e estruturada a pensar na sua escalabilidade, flexibilidade, e respeitando as 3 normas normalização designadas por Boyce-Codd Normal Form, ou seja, na Base de Dados os atributos numa relação são atómicos (First Normal Form) ou, simplesmente, que cada coluna contem apenas um valor e que cada linha deverá conter o mesmo número de colunas. Adicionalmente, os atributos que não são chaves devem depender funcionalmente da chave primária (Second Normal Form). Esta regra elimina as dependências parciais. Por fim, os atributos que não dependem da chave primária devem ser eliminados ou deve ser criada uma nova tabela para os mesmos (Third Normal Form).

O modelo conceptual da base de dados é bastante extenso e não sendo relevante estar a mostrar todo o modelo, apresentamos apenas o modelo conceptual simplificado de forma a ilustrar a base de dados (Figura 3).

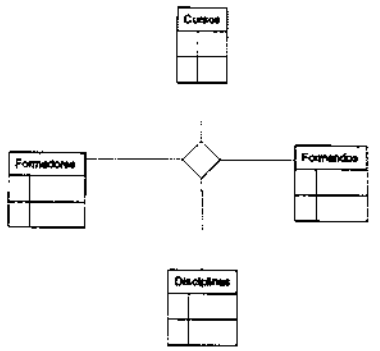


Fig. 3 – Modelo Conceptual (Simplificado)

A estrutura da base de dados, é formada por aproximadamente 100 entidades, o que ilustra a sua dimensão. Na Figura 4 está representada uma parte da estrutura da base de dados que é usada pela interface de formadores que iremos descrever em seguida.

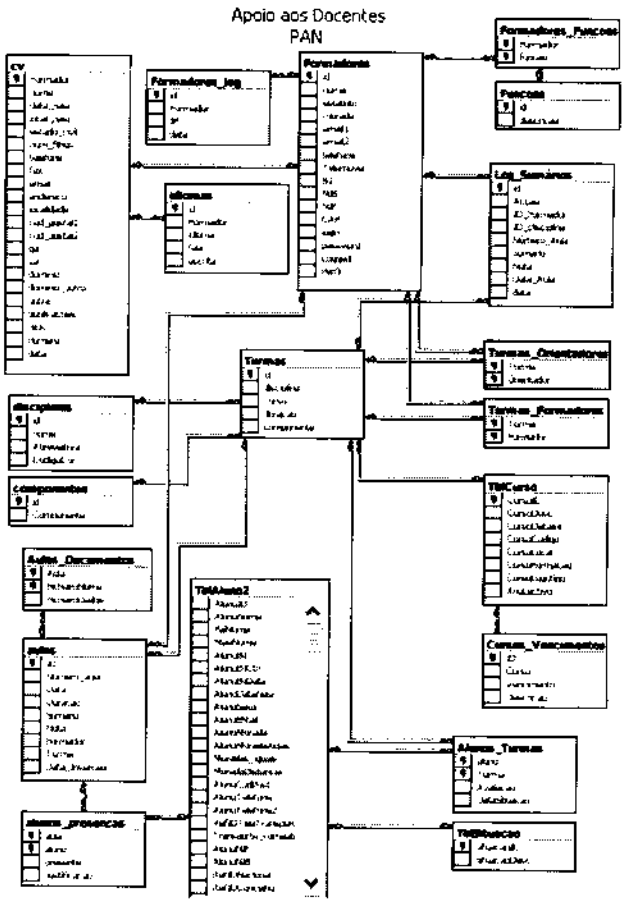


Fig. 4 – Modelo lógico da base de dados usada pela interface de formadores

IV. FUNCIONALIDADES DO MÓDULO DE GESTÃO E ADMINISTRAÇÃO

Este módulo permite aceder a toda a informação relativa aos administradores, formadores, formandos, cursos e outros dados de importância para o Programa Aveiro Norte. Quem acede a este módulo tem ao seu dispor uma lista de funcionalidades como se pode constatar pela Figura 5.

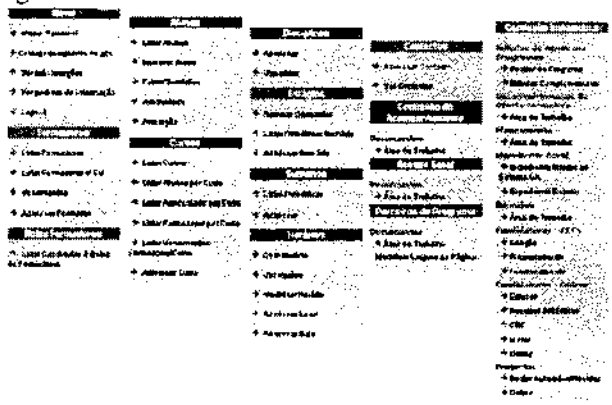


Fig. 5 – Funcionalidades disponíveis no módulo de Gestão e Administração

Este módulo disponibiliza aos utilizadores funcionalidades que vão desde uma simples consulta dos formandos inscritos a um curso até ao armazenamento de ofícios.

V. FUNCIONALIDADES DO MÓDULO FORMADORES

Para aceder ao módulo do formador é necessária a autenticação por login e password. Após a autenticação ter sido efectuada com sucesso, o formador fica com acesso à lista de disciplinas que lecciona, como se pode ver pela Figura 6.

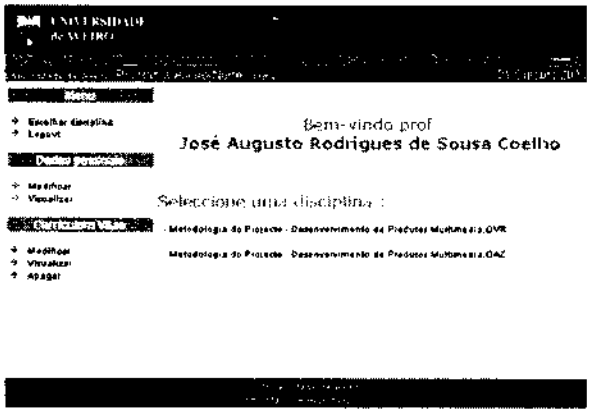


Fig. 6 – Página inicial do módulo de formadores

No menu que tem ao seu dispor nesta página, o formador pode ainda modificar ou visualizar os seus dados pessoais e/ou modificar, visualizar ou apagar o Curriculum Vitae.

Após a selecção de uma disciplina são disponibilizadas ao formador um novo conjunto de funcionalidades relacionadas com a disciplina que escolheu e aparece na parte do cabeçalho da página o nome da disciplina escolhida, permitindo assim o formador saber a que disciplina está a aceder.

O formador poderá listar, adicionar, modificar ou apagar sumários e ainda ver uma listagem dos alunos inscritos nessa disciplina. Na área de sumários o formador tem ao seu dispor um espaço que lhe permite sumariar as aulas e registar as presenças dos seus alunos, obter informação sobre o número de horas leccionadas e a duração total de horas a leccionar. Esta interface está representada na Figura 7.

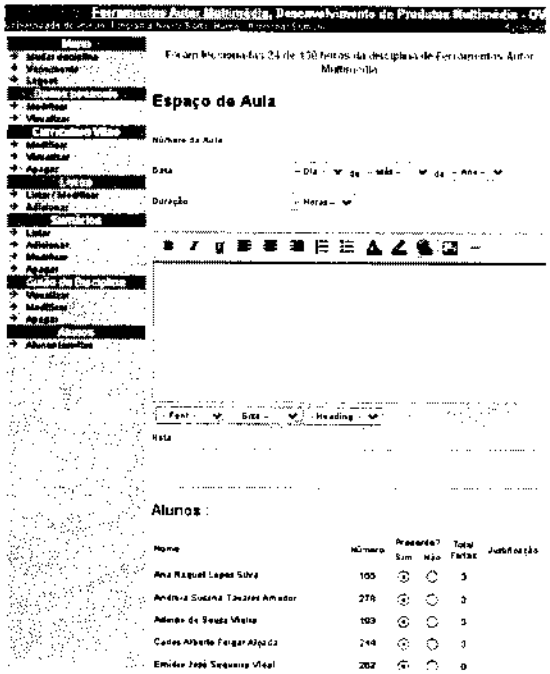


Fig. 7 – Página de registo do sumário e presenças em aula

Após o preenchimentos do sumário, bem como registo de presenças, o formador irá ser questionado se deseja agregar materiais de apoio à aula leccionada (Figura 8), este mecanismo de agregação é um simples mecanismo de upload de ficheiros (Figura 9), com a particularidade de estes serem armazenadas numa base de dados, por questões de segurança.

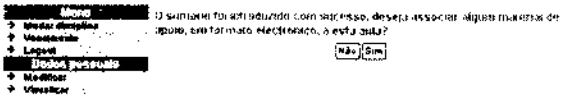


Fig. 8 – Agregação de materiais de apoio às aulas

Fig. 9 – Mecanismo de upload de ficheiros

O formador pode ainda visualizar uma listagem dos seus alunos inscritos a essa disciplina. Deste modo, o formador tem acesso a todo o histórico de faltas do aluno após a selecção do mês referente ao campo frequência (Figura 10).

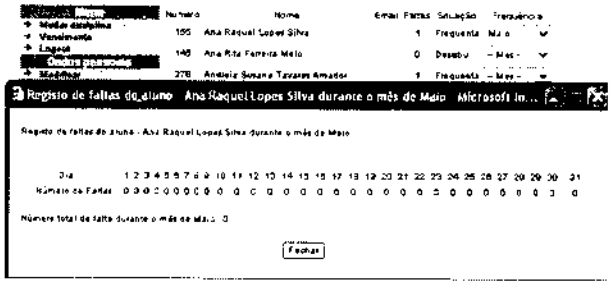


Fig. 10 – Assiduidade dos alunos

Tal como já foi referido, para além do espaço de aula o formador tem sempre a possibilidade de alterar ou consultar os seus dados pessoais e aceder ao seu Curriculum Vitae o qual pode ser alterado sempre que se justifique.

VI. FUNCIONALIDADES DO MÓDULO DE FORMANDO

Este módulo tem uma particularidade em relação aos módulos anteriores, é um módulo de acesso livre, ou seja, não é pedido qualquer tipo de autenticação ao utilizador, tal não acontece quando se tenta aceder aos módulos anteriores. Este módulo foi concebido de modo a disponibilizar o máximo de informação, aos formandos, relativa às actividades formativas (Figura 11).

Este módulo permite-nos obter informação relativa aos cursos, tanto dos CETs – Cursos de Especialização Tecnológica como dos CAEs – Cursos de Actualização e Especialização, onde podemos saber modo de funcionamento de cada curso, os locais de funcionamento, o número de vagas, etc., o utilizador pode ainda consultar um arquivo de documentos relativos a cada curso. Além de ser disponibilizado o plano curricular de cada curso e disponibilizada informação de cada disciplina do curso, isto é, podemos consultar os formandos inscritos a cada disciplina e consultar os sumários de cada aula bem como o registo de presenças dos formandos na mesma.

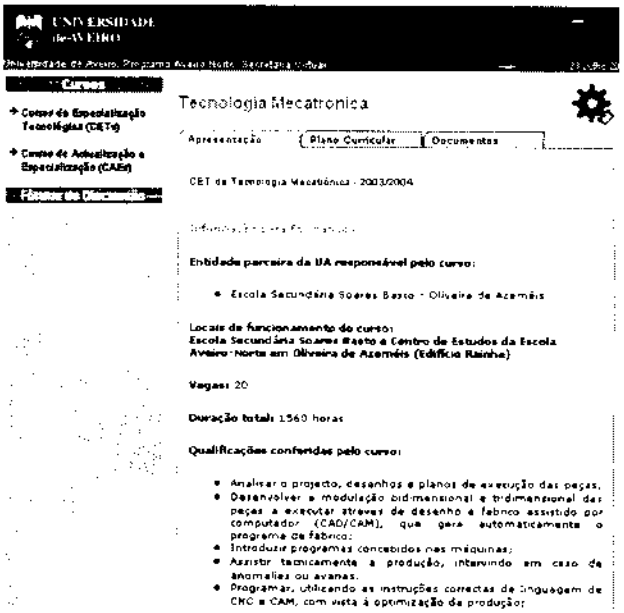


Fig. 11 – Informação para os formandos

O formandos e formadores têm ainda ao seu dispor um fórum de discussão (Figura 121). Este fórum está dividido por cursos, ou seja, cada utilizador pode consultar os comentários por cada um dos cursos existente. Neste fórum cada utilizador é livre e incentivado a colocar as suas dúvidas, questões, as quais poderão a vir ser comentadas pelos outros utilizadores.

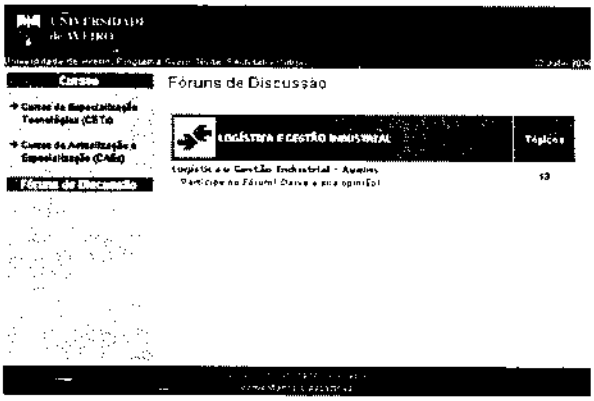


Fig. 12 – Fórum de discussão

VII. CONCLUSÕES

Com o uso do sistema pelo grupo de utilizadores do Programa Aveiro Norte constatou-se que houve efectivamente uma maior eficácia e uma redução do tempo nos processos burocráticos associados ao programa. Através da recolha de opiniões junto aos formadores e formandos, através do preenchimento de inquéritos anónimos, pudemos constatar que o sistema veio dar um apoio real nas actividades formativas, permitindo um maior acompanhamento, por parte dos formandos, das aulas.

Pode-se considerar que o sistema está, neste momento, mais próximo do sistema idealizado no início da sua concepção.

REFERÊNCIAS

[1] Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management, Addison-Wesley, 3ª Edição, 2002.

[2] Date C., An Introduction to Database Systems, VI Edição, Addison-Wesley Systems Programming Series, 1996.

[2] Vieira, João, *Programação web com Active Server Pages*, Centro Atlântico, 2000

[3] Devguru, Homepage, http://www.devguru.com , (on-line, Outubro de 2004)

[4] DevX, Homepage, http://www.devx.com , (on-line, Outubro de 2004)

[5] Nunes, Mauro , O'Neill, Henrique, *Fundamental de UML*, FCA Editora, 2001

A Talking Face for Portuguese

Raquel de Castro Lisboa[1], António J. S. Teixeira, Artur Pimenta Alves[1,2]

Resumo – **A combinação de voz sintetizada com agentes animados, completos ou representados pela cara apenas, tem muitas aplicações, inlusive nas Telecomunicações. Tendo por objectivo contribuir para a existência desta tecnologia para a nossa língua, este artigo apresenta o desenvolvimento dos blocos necessários para uma "cara falante" para o Português. O trabalho incidiu na adaptação de um sistema, o RUTH, o desenvolvimento de uma voz rudimentar para para o Português utilizável no sistema de síntese de voz Festival e o desenvolvimento de uma forma simples de controlar o comportamento da cara pela utilização de anotações XML. Como demonstrador, foi desenvolvida uma aplicação capaz de controlar em tempo real várias "caras falantes", podendo cada uma utilizar uma voz diferente, tendo por entrada texto com anotações XML**

Abstract – **Combination of speech output with animated agents, complete or represented by face alone, has many applications, including in telecommunications. Aiming at contributing to the availability of such technologies for our language, this paper presents the development of the necessary building blocks to have a talking face for Portuguese. Work focused on the adaptation of an available system, named RUTH, the development of a basic Portuguese voice for the Festival speech synthesis system, and development of an easier way of controlling the talking face behaviour by using XML annotations. As a demonstration of the developed talking face, an application capable of controlling in real time several faces, each speaking with a different voice, based on XML annotated text input was implemented.**

Keywords: Talking face, visual speech, computer animation, face animation, audiovisual speech synthesis, Text-to-Speech synthesis.

I. INTRODUÇÃO

Assim como a integração da voz se apresenta muitas vezes como uma mais valia em relação ao texto escrito, a imagem de uma cara animada em 3D surge, também ela, como um complemento importante para a transmissão e compreensão de uma determinada mensagem [3]. É indiscutível que a possibilidade de observar a cara (ou mesmo apenas os lábios) do falante aumenta a inteligibilidade da fala, bastando para isso reportarmo-nos à técnica de leitura labial, usada por pessoas com deficiências auditivas. Não é, portanto, surpreendente que o mesmo se passe com pessoas que não possuam esse tipo de deficiência, quando na presença de ambientes nos quais a comunicação se apresenta de alguma forma degradada (por exemplo, no caso de ambientes ruidosos) [6]. Existe, sem sombra de dúvida, informação preciosa transportada na cara do falante que não deve ser desprezada.

De facto, muitas vezes usamos diversos sinais, verbais e não verbais, para reforçar a ideia chave num discurso. Coisas simples que utilizamos instintivamente, como a entoação, expressões faciais e movimentos da cabeça não são necessariamente redundantes, podendo, muitas vezes, acrescentar algo de novo à interpretação de uma mensagem, transformando-se numa contribuição eficaz para a conversação.

Na área concreta dos agentes de conversação, existem já algumas aplicações direccionadas para o mercado do consumidor, como o fornecimento de serviços de informação (notícias, meteorologia, etc.) em que a própria personagem computorizada surge como complemento do texto e da fala. É ainda de destacar, a título de exemplo, o projecto SYNFACE [7] que tem como objectivo principal facilitar a utilização de um simples telefone a pessoas com deficiências auditivas, conseguido através da conexão de uma cara falante ao telefone.

II. OBJECTIVOS E ABORDAGEM

O objectivo principal deste projecto consistiu no desenvolvimento de uma cara sintética para utilização conjunta com Português falado. Como objectivo complementar, pretendia-se desenvolver uma aplicação de demonstração dos componentes desenvolvidos.

Adicionalmente, no decorrer do projecto, e uma vez que não existe nenhuma voz de Língua portuguesa que podesse ser usada (freeware), começou a fazer sentido tentar construir uma base de desenvolvimento.

O esquema da Figura 1 ilustra, de uma forma geral, os diferentes módulos e programas intervenientes, e sua interacção. O módulo designado por Interface gráfica para além de lidar com a visualização e interacção com o utilizador controla os outros componentes: sistema de síntese Festival e a cara.

[1] FEUP, [2] INESC Porto

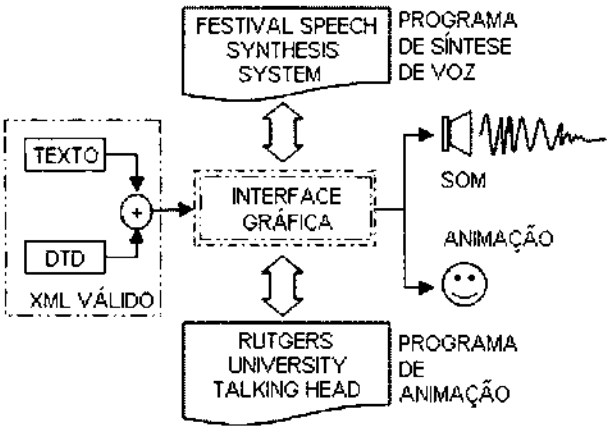


Fig. 1 - Esquema dos módulos, programas e sua interacção

III. SOFTWARE

O primeiro passo na execução deste projecto, foi a pesquisa, análise e teste de diversos programas, a fim de perceber qual o mais adequado à finalidade pretendida. Foram avaliados o *POSER 5*, o *GALATEA*, o *CUANIMATE* [11], o *BALDI* [9,10], LUCIA [5] e o *RUTH* [2].

Considerando factores como a programabilidade, o custo, a plataforma usada e a compatibilidade com o *Festival Speech Synthesis System*, o programa RUTH revelou ser a melhor opção.

Como consequência directa da escolha adequada do programa de animação, a tarefa de desenvolvimento das facilidades de sincronização do sinal de voz com a cara revelou-se desnecessária.

Não pretendendo descrevê-lo exaustivamente, nas subsecções seguintes é feita uma breve caracterização do software utilizado.

A. Rutgers University Talking Head

O RUTH [2] é um sistema de animação facial, em tempo real, para a Língua inglesa que, em sincronização com a fala e movimentos dos lábios, combina entoação, movimentos e expressões faciais. O programa consiste num conjunto de três threads independentes que usam filas para a sua coordenação e comunicação. O diagrama da Figura 2 é ilustrativo dessa arquitectura.

A animação é conseguida fornecendo ao programa texto anotado com informação não só sobre a fala, mas também sobre os movimentos da cara. Tudo isto é possível recorrendo a comandos de alto nível permitidos pelo programa de animação. Estes comandos permitem controlar:

- Na voz
  - a acentuação do pitch (accent)
  - a gama do pitch (register)
  - o tom (tone)

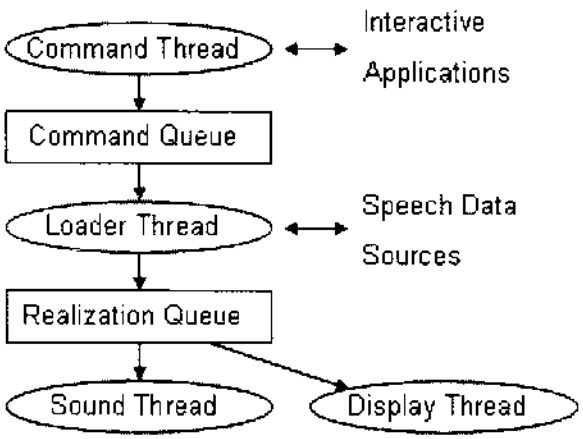


Fig. 2 - Arquitectura do RUTH

- Movimentos
  - o piscar dos olhos (blink)
  - o movimento das sobrancelhas (brow)
  - o movimento da cabeça (jog)
  - o sorriso (smile)

No que diz respeito ao modelo, os seus autores adoptaram a ilustração, por oposição ao foto-realismo, a fim de obterem um resultado atractivo dentro de limites computacionais razoáveis. Além disso, é propositada a ambiguidade em relação ao sexo, raça e idade do modelo.

Ao longo deste ponto, tentou dar-se uma ideia do funcionamento do programa de animação. No entanto, note-se que este programa é composto por mais de 30 ficheiros de código C e outros tantos que mapeiam os visemas [8] e fonemas utilizados para a correcta movimentação da boca, bem como código de programação gráfica que controla a movimentação da face de uma forma geral. Embora não tenham sido todos exaustivamente estudados, foi necessário perceber a sua estrutura e adquirir um grau de confiança sólido para poder manipulá-los de forma correcta, sem prejuízo para as funcionalidades existentes. Como se deve depreender, este é um programa com alguma complexidade e embora haja documentação disponível, esta centra-se essencialmente na sua utilização e concepção, de uma forma geral, não tendo o intuito de o modificar e adaptar, conforme era pretendido neste projecto.

B. Festival Speech Synthesis System

O programa Festival [1] é um sintetizador Text-to-Speech, sendo ele que permite ao RUTH realizar a síntese do texto escrito, conseguido pelo seu funcionamento em modo de servidor. Para os sistemas funcionarem conjuntamente, a comunicação é feita recorrendo a sockets.

Considerando os objectivos inicialmente propostos, apenas seria necessário conhecer este programa ao nível do utilizador, mas a partir do momento que se decidiu construir a nova voz, foi inevitável compreender de uma forma mais aprofundada o seu funcionamento.

Foi realizada a instalação do programa de síntese Festival Speech Synthesis System (versão 1.4.2), em Linux, sendo, para o seu correcto funcionamento, necessário instalar as "speech tools". Foi ainda necessário instalar o OGIresLPC (Residual LPC synthesizer), requerido pela voz brasileira instalada em seguida, tendo-se também procedido à instalação das vozes disponíveis.

IV. ADIÇÃO DE NOVAS FUNCIONALIDADES

Nesta secção serão descritas as funcionalidades acrescentadas às já permitidas pelo programa de animação. Tentando fazer uma abstracção ao código propriamente dito, pretende-se explicar, de uma forma simples, as soluções encontradas para a sua implementação.

Para contemplar todas as alterações que serão referidas, foi necessário alterar o código do programa de animação da cara e construir a interface, de forma conveniente.

No que diz respeito à síntese de voz, as funcionalidades acrescentadas prenderam-se com a instalação e disponibilização de uma voz brasileira, "aga_diphone", e de um sintetizador de voz baseado na concatenação de fonemas, o MBROLA [14], e as respectivas bases de dados disponíveis para as vozes da Língua inglesa, e portuguesa.

O objectivo da adição destas funcionalidades foi, tal como referido, a tentativa de criar um sistema para o português. O método de construção da voz é explicado de uma forma superficial na secção seguinte.

Como consequência directa da introdução de novas Línguas, tornou-se evidente a necessidade de acrescentar os fonemas em falta ao ficheiro de mapeamento existente (para o Inglês), adaptando-o assim às vozes portuguesas usadas. Esta alteração é fundamental, pois os fonemas variam consoante a Língua, e quando aparece um fonema que o RUTH não conhece, não sabe como o há-de traduzir no movimento adequado da boca.

No que diz respeito aos comandos, as funcionalidades acrescentadas relacionam-se com a adição ao código fonte do RUTH, código que permite a comutação para qualquer voz existente no Festival. A implementação desta funcionalidade teve como objectivo, não só possibilitar a comutação interactiva da voz (pelo utilizador), mas também a utilização de mais de uma cara com vozes distintas.

Até este ponto, era possível correr em simultâneo várias caras, mas como a definição da voz era feita no ficheiro de configuração, lido no início da execução do programa Festival, em modo servidor, a voz usada tinha obrigatoriamente de ser a mesma.

Para ultrapassar a limitação referida, foi adicionado ao programa de animação um novo comando, *voice*, através do qual é possível, no decorrer da aplicação, mudar para qualquer voz actualmente instalada no Festival. Para evitar uma sobrecarga desnecessária da interface, e uma vez que se pretende demonstrar essencialmente o uso do Português, apenas é permitida pela interface a escolha entre as vozes da Língua portuguesa (Português Europeu e Português Brasileiro) e inglesa.

Foi criada uma nova opção da linha de comandos, - "position"- que deve ser acompanhada de 2 valores: as coordenadas (x, y) que indicam a posição onde a janela com a cara deve aparecer. Estes valores variam consoante o número da cara, identificado pelo pipe, sendo geridos pela interface. Este comando foi criado para evitar sobreposições das janelas com as caras, evitando, assim ao utilizador o trabalho de ter de as reposicionar no ecrã.

V. CONSTRUÇÃO DA VOZ PARA O PORTUGUÊS

O processo de construção da voz portuguesa passou pela compreensão do modo de construção e funcionamento das vozes sintetizadas que recorrem à técnica de concatenação de difones. Deve ter-se em mente que o MBROLA não é um sintetizador Text-to-Speech, já que não recebe texto simples, mas, em vez disso, aceita informação prosódica e uma lista de fonemas, produzindo amostras de voz. Quando usado juntamente com o Festival, é possível obter uma saída áudio.

A utilização de um programa de conversão de grafemas para fonemas, *String2phon*, disponibilizado pelo grupo de investigação interdisciplinar do IEETA e Centro de Línguas e Culturas da Universidade de Aveiro, traduziu-se num melhoramento substancial da qualidade da voz.

Seguindo o manual do Festival, o método para definir uma nova voz passa pela definição de parâmetros para as várias partes que constituem a voz. Recorreu-se aos exemplos das vozes espanhola e inglesa disponíveis. A sua programação é feita em código Scheme, tendo sido necessário aprender a interpretá-lo para o conseguir utilizar convenientemente.

De seguida, apresenta-se uma breve descrição das partes constituintes da voz, que serviram de guias para a construção de um protótipo de uma voz portuguesa, que usa a base de dados de uma voz feminina (disponível no site do projecto MBROLA [14]).

- **Fonemas** - O bloco básico para a construção de uma nova voz é a definição dos seus fonemas. Esta é uma parte fundamental, já que outros blocos serão construídos à custa deste. É neste bloco que, para cada fonema, se definem os parâmetros característicos que os distinguem. Faz-se a distinção entre vogal e consoante, o tamanho e tonalidade das vogais, o arredondamento dos lábios, o tipo de consoante, e sítio de articulação.
- **Léxico** - Neste bloco são definidas as regras de conversão grafema – fone, conhecidas por regras LTS (Letter To Sound), bem como o dicionário para o Português. Basicamente, consiste em determinar como cada letra ou conjunto de letras se pronuncia, recorrendo ao contexto fonético em que se situa. Para as excepções, palavras difíceis, ou palavras que simplesmente soam mal segundo estas regras, existe um dicionário, onde é definida, manualmente, a sua transcrição. No caso da voz construída, esta tarefa

foi delegada num programa externo, já referido. Optou-se por esta alternativa, já que a construção destas regras não é uma tarefa trivial, ficando fora do âmbito do trabalho apresentado.

- **Fraseamento** - Com este bloco deveria ser possível prever as quebras de frase. Na versão simples adoptada as pausas são determinadas com base apenas na pontuação.
- **Entoação** - Neste bloco deveriam ser definidas as regras de entoação para o Português, pela alteração do pitch do falante. Mais uma vez, a previsão da acentuação das sílabas não é uma tarefa fácil, e o *String2phon* resolve, parcialmente, o problema, de uma forma bastante razoável, para a maioria das palavras.
- **Duração** - Este bloco é também uma parte com substancial importância, já que é aqui que são estabelecidas as durações de cada fonema. Estes valores advêm de médias calculadas a partir de valores medidos. Quanto mais adequados forem estes valores, mais natural resultará a voz.

É perfeitamente plausível assumir que foi criada uma base, sendo de salientar que, embora esteja utilizável, esta voz se encontra num estado primário de desenvolvimento, necessitando ainda de inúmeros ajustes.

Uma vez que a sua criação não era objectivo principal deste projecto, não foi afectado tempo à tarefa de melhoramento da voz, nem tal seria possível.

Os testes efectuados englobam exemplos variados, recorrendo a notícias retiradas de jornais on-line, excertos de textos sobre museus, disponíveis em várias línguas, nos sites oficiais, etc.

VI. ANOTAÇÃO DO TEXTO

Como foi dito anteriormente, o objectivo de possuir texto anotado é o de reforçar as ideias chave, contribuindo de alguma forma para a inteligibilidade da conversação. Analogamente, é vantajoso ter uma forma simples de efectuar a sua anotação, que permita a utilização desta facilidade, sem ter de perceber os detalhes técnicos do RUTH.

Ao longo desta secção, serão explicadas as anotações em causa e o processo para obter um texto correctamente anotado. Para o efeito, aplicou-se uma tecnologia actual – a *eXtensible Markup Language (XML)* [16-19].

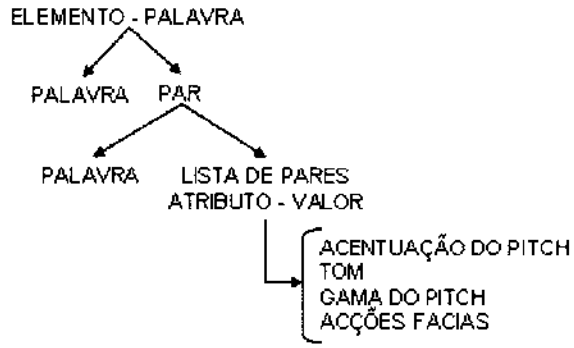
A interface construída evoluiu no sentido da utilização de documentos XML como texto de entrada ao programa de animação. Esta abordagem permite um controlo sobre a introdução dos comandos de alto nível disponibilizados, operando sobre a voz e a cara, com o objectivo de facilitar a sua correcta utilização.

Tendo em vista este objectivo, foi construída uma *Document Type Definition (DTD)* [12] que contém as regras pretendidas para o XML, onde se define o tipo de transformações que a voz e a cara podem sofrer (é de referir que os atributos das marcas não estão tão explícitos como pretendido, limitação resultante da impossibilidade de definir caracteres especiais como elementos enumerados de uma DTD).

Para cada tipo de anotação, existe uma marca de abertura, que possui as propriedades (valor), e uma marca de fecho. No caso das marcas vazias, estas não possuem qualquer atributo. É apresentado de seguida um exemplo ilustrativo de um texto anotado com algumas das marcas XML referidas:

```
<?xml version="1.0" ?>
<!DOCTYPE doc (View Source for full doctype...)>
<doc>
   <SPEAKER nr="1">
      Considerada a mais notável colecção do
      <ACCENT type="L">mundo</ACCENT>
      <BLINK/>
      permite ao visitante compreender não só a
      evolução técnica dos transportes de tracção
      <REGISTER type="HL">
         <BROW type="raise">animal</BROW>
      </REGISTER>
      como acompanhar as mudanças tão bem
      <JOG type="forward">expressas</JOG>
      na ornamentação das viaturas.
   </SPEAKER>
</doc>
```

O parsing das marcas XML é feito pela interface, que as transforma em código Scheme, interpretável pelos programas de animação e síntese. Para melhor se entender o esquema de anotação utilizado, deve considerar-se a seguinte hierarquia:



Analisando o esquema conjuntamente com o exemplo apresentados, verifica-se que as anotações do texto são processadas tendo como base um "elemento-palavra" que pode ser apenas a própria palavra ou uma lista constituída por um par (palavra, (atributo, valor)).

Para as anotações terem o comportamento pretendido, deve ter-se em consideração um conjunto de regras e restrições de uso. Algumas destas limitações são consequência do próprio programa de animação, outras do parsing feito pela interface. Note-se que este parsing não traz apenas limitações, mas também vantagens, uma vez que é mais robusto em diversos aspectos.

Uma das regras utilizadas define que é apenas permitida uma marca de cada tipo, para cada palavra, o que faz todo o sentido. Tome-se como exemplo a marca "ACCENT": não faz sentido marcar a mesma palavra com um pitch alto e baixo.

No que diz respeito às restrições, estas não limitam substancialmente a marcação do texto, mas devem ser tidos em atenção os seguintes aspectos: as marcas não podem englobar mais de uma palavra; não são permitidas marcas em palavras seguidas; as marcas vazias devem ser colocadas depois de alguma palavra e nunca no início de uma frase.

Marcas mal formadas (atributo ou valor inválido) são eliminadas, não permitindo que um pequeno erro impeça todo o texto de ser sintetizado.

Outra regra extremamente importante é a que determina a necessidade da existência das marcas de abertura e fecho "SPEAKER", uma vez que, num universo multi-falante, é esta a marca que define qual a cara a que o texto se destina. Faz sentido que todo o texto fora destas marcas seja ignorado.

Como conselho de utilização, deve recorrer-se frequentemente ao uso de pontuação final (".", "!" ou "?"), uma vez que as regras de fraseamento apenas recorrem à pontuação para produzir pausas no texto. A restante pontuação (",", ";", ":", "(" e ")") é retirada do texto, porque não é aceite pelo RUTH ou, ainda, porque interfere com as marcas, como é o caso dos parênteses.

A síntese é feita de uma forma transparente para o utilizador, sem o conhecimento dos formalismos requeridos anteriormente, quer pelo programa de animação, quer pelo próprio sintetizador, tais como o uso de comandos específicos, parênteses e linhas em branco. No caso de se pretender anotar o texto, é apenas necessário ter em conta as observações feitas anteriormente.

VII. APLICAÇÃO

Embora se pretenda que a aplicação construída seja uma demostração dos componentes desenvolvidos, o principal objectivo da sua elaboração foi o de tornar o programa de animação utilizável por qualquer pessoa, isto é, por utilizadores que não possuam qualquer conhecimento sobre o seu funcionamento. Anteriormente à implementação da interface, a utilização do RUTH era feita extensivamente através da linha de comandos e, de uma forma muito limitada, através do menu que apenas permitia o acesso a demonstrações da sua aplicabilidade.

Esta aplicação não pode ser vista simplesmente como uma interface gráfica, já que não tem como função apenas facilitar a execução de instruções de uma forma intuitiva através de menus e botões, mas também é ela quem controla todos os processos e realiza algumas funções mais específicas, como o parsing das marcas XML.

Tendo em vista os objectivos mencionados, o primeiro passo foi a instalação do Qt C++ toolkit, para Linux. A escolha de desenvolver a interface gráfica, recorrendo a essa biblioteca de classes, deveu-se a esta ser multiplataforma e de fácil uso e integração com OpenGL.

Por uma questão de simplicidade, a abordagem foi a de criar a interface de uma forma independente do programa de animação, recorrendo ao uso de *pipes* para a comunicação entre eles. Para tornar isso possível, foi necessário uma reimplementação da comunicação entre os vários processos, redireccionando a forma como o RUTH recebe a informação: anteriormente pela linha de comandos, agora pelo unnamed pipe.

Ao iniciar a aplicação, é solicitado ao utilizador que escolha o número de falantes pretendidos. Foi estipulado um número máximo de 4 falantes, que pareceu um valor razoável, tendo em vista os objectivos pretendidos.

Existe um menu de opções, apresentado na Figura 3, que permite a qualquer altura, para cada falante, comutar entre as várias vozes existentes. Pelos motivos já mencionados, o sistema apenas está configurado para permitir a comutação entre vozes da Língua Portuguesa (Europeu e Brasileiro) e da Língua Inglesa.

Devido à sua arquitectura, o programa de animação permite determinar facilmente se existe texto a ser sintetizado e, através da troca de mensagens com a interface, é possível evitar por completo a sobreposição das vozes, no caso de existir mais de um falante. Para além disso, esta troca de mensagens serve também para facilitar o acesso a mais informação sobre a eventual ocorrência de erros no decorrer do programa, permitindo fornecer mensagens de erro mais específicas.

Na Figura 4 apresenta-se o ecrã exemplo da interface, acompanhado de duas janelas com as caras. Além de ser mais funcional, uma apresentação gráfica é sempre mais apelativa do que escrever texto numa consola.

Para além das funcionalidades descritas, continua a ser possível guardar e carregar ficheiros com informação

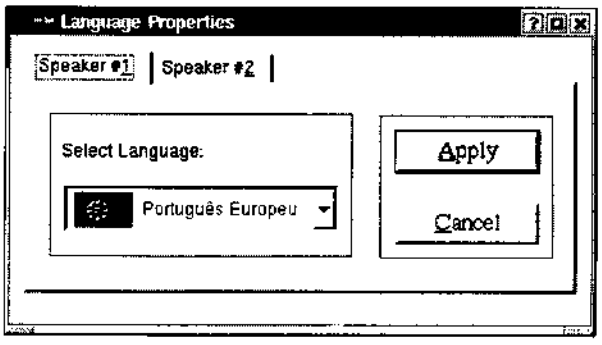


Fig. 3 - Menu de opções

sobre animações previamente realizadas, estando estas funcionalidades acessíveis no menu *Options*.

VIII. CONCLUSÕES

Após um estudo mais ou menos extensivo dos programas intervenientes e tecnologias envolvidas, como resultados relevantes conseguidos salientam-se: o uso de novas vozes pela cara falante; a construção de um protótipo de uma voz de Língua portuguesa; o uso de XML para anotação

do texto de uma forma simples e directa, não só respeitante à voz, mas também à animação da cara em si e devido uso das mesmas; a inserção de novos comandos

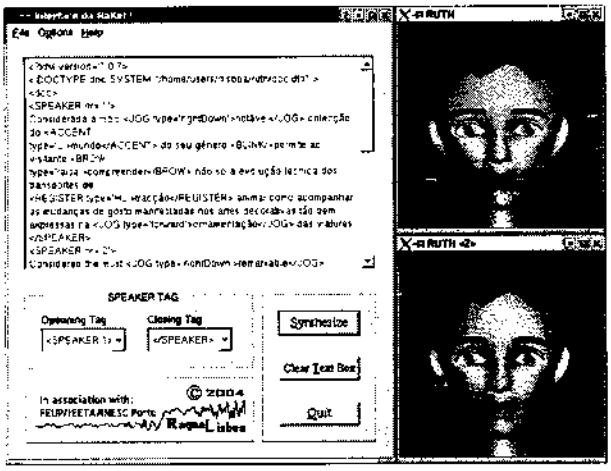


Fig. 4 - Exemplo da interface, com 2 caras

que permitem não só a comutação interactiva da voz pelo utilizador, mas também a utilização de mais de uma cara com vozes distintas.

A evolução natural do projecto levou à construção de uma API simplificada, que permite uma utilização intuitiva por parte do utilizador. Ao manter as funcionalidades existentes e adicionando outras, foi possível aumentar largamente as capacidades e aplicações do programa de animação.

Mais uma vez se reforça a ideia de que os programas envolvidos são complexos e que, por essa razão, o tempo empregue no seu estudo e compreensão não é desprezável.

Como já foi sendo referido ao longo deste artigo, existem diversas áreas onde há ainda lugar para evolução, nomeadamente, no que diz respeito à construção da voz de Língua portuguesa e à anotação automática do texto. Apesar de não estar previsto um envolvimento tão aprofundado na área da síntese de voz, é importante referir que todo o trabalho desenvolvido nessa área se apresenta como um contributo imprescindível para este projecto. No que diz respeito à anotação do texto, seria interessante poder automatizar o processo, quer baseado na pontuação, quer na própria pré-análise do texto a sintetizar. A título de exemplo, pode referir-se a mudança de tom na presença de um ponto de interrogação ou o movimento da cabeça na presença incontestável de afirmações ou negações.

Consideramos que os objectivos inicialmente propostos foram atingidos e, alguns, inclusivamente superados.

Tratou-se de um projecto interessante, educativo e bastante apelativo, numa área em constante movimento e desenvolvimento. A utilização de animação facial em conjunto com voz, em tempo real, desempenhará um papel importante no futuro das comunicações em geral.

Para informações mais detalhadas sobre este Projecto, pode consultar-se a página da internet elaborada para o efeito, disponível em:

http://pwp.netcabo.pt/formatodecores/raquelisboa/projecto.html

AGRADECIMENTOS

Agradecemos todo o apoio prestado durante o desenvolvimento do projecto pelo Eng.º Luís Gustavo Martins (INESC Porto) e pela Professora Lurdes de Castro Moutinho (DLC/Universidade de Aveiro).

REFERÊNCIAS

[1] A. Black, P. Taylor e R. Caley, "The Festival Speech Synthesis System" (http://www.cstr.ed.ac.uk/projects/festival)

[2] D. DeCarlo e M. Stone, "The Rutgers University Talking Head: RUTH" (http://www.cs.rutgers.edu/~village/ruth)

[3] C. Benoit e B. Le Goff, "Audio-visual speech synthesis from French text: Eight years of models, designs and evaluation at the ICP", Speech Communication, Vol 26 , n 1-2, Special issue on auditory-visual speech processing, p 117 - 129 , 1998.

[4] J. Hulstijn et al., "Dialogues with a Talking face for Web-based Services and Transactions", In Proceedings Face to Face Workshop, CWI, Amsterdam. Available as CTIT Technical report 99-07.

[5] P. Cosi, A. Fusaro e G. Tisato, "LUCIA a New Italian Talking-Head Based on a Modified Cohen-Massaro's Labial Coarticulation Model", Proc. Eurospeech 2003.

[6] A. Newell, "Spoken Language and e-inclusion", Proc. Eurospeech 2003.

[7] I. Karlsson, A. Faulkner e G. Salvi, "SYNFACE – a talking face telephone", The Eurospeech Special Event on "Spoken Language Technology in E-inclusion", Proc of EuroSpeech 2003.

[8] J. De Martino e L. Magalhães, "Um Conjunto de Visemas para uma Cabeça Falante do Português do Brasil",Congresso Iberdiscap, Tecnologías de Apoyo para la Discapacidad, Vol. 1, pp.198-203, San José, CR, 2004.

[9] BALDI (http://itakura.kes.vslib.cz/kes/baldie.html)

[10] CSLU toolkit (http://www.cslu.ogi.edu/tts/)

[11] CUANIMATE (http://cslr.colorado.edu/beginweb/cuanimate/cuanimate_paper.html)

[12] DTD Tutorial (http://www.w3schools.com/dtd/default.asp)

[13] INOVANI (http://www.inovani.no/TechSpeech.htm)

[14] MBROLA Project (http://tcts.fpms.ac.be/synthesis/mbrola.html)

[15] TrollTech – Qt C++ toolkit (http://www.trolltech.com)

[16] XED: An XML Document Instance Editor (http://www.cogsci.ed.ac.uk/~ht/xed.html)

[17] XML 1.0 – Extensible Markup Language (W3C) (http://www.w3.org/TR/1998/REC-xml-19980210)

[18] XML Developer Center (http://www.msdn.microsoft.com/xml/)

[19] XML FAQ (http://www.ucc.ie/xml/)

Modelos Deformáveis na Segmentação de Imagens Médicas: uma introdução

José Silvestre Silva*, Beatriz Sousa Santos, Augusto Silva, Joaquim Madeira

*Departamento de Física, Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Resumo **– Neste artigo introduzem-se os fundamentos matemáticos dos modelos deformáveis de forma muito breve, descrevem-se as principais características de cada e faz-se uma revisão da sua utilização em cenários de segmentação de imagens médicas.**

Abstract **– In this paper we briefly introduce the mathematical foundations of deformable models, their properties and their usage in medical image segmentation.**

I. INTRODUÇÃO

A extracção de informação útil sobre estruturas anatómicas a partir de imagens de CT (Computed Tomography), MR (Magnetic Ressonance), PET (Positron Emission Tomography) e outras modalidades, é actualmente uma área de investigação activa. Neste contexto, a segmentação automática e obtenção de uma representação geométrica compacta têm vindo a ter um papel cada vez mais importante na imagem médica, sendo utilizadas em numerosas aplicações. No entanto, continuam a constituir um problema difícil devido, entre outras causas, à quantidade de dados, à variabilidade das formas que as estruturas a detectar podem assumir e a questões de qualidade da imagem. Em particular o ruído e os problemas de amostragem podem degradar os resultados obtidos através das técnicas tradicionais de segmentação de imagem que consideram apenas informação local, implicando a necessidade de intervenção humana frequente. Outra desvantagem destas técnicas é o facto de gerarem, em regra, representações baseadas em *pixels* ou *voxels* que dificultam a posterior análise e interpretação dos objectos segmentados.

Para ultrapassar as referidas dificuldades, os métodos de análise de imagem baseados em modelos utilizam informação adicional baseada em conhecimento, sendo, em geral, mais sofisticados que as abordagens que não usam modelos [5].

Os modelos deformáveis têm sido extensivamente aplicados na segmentação de imagens médicas (2D e 3D), com resultados promissores. As potencialidades que geralmente lhes são reconhecidas, resultam da sua capacidade de segmentar (bem como emparelhar e seguir) estruturas em imagens explorando simultaneamente restrições derivadas da imagem (abordagem *bottom-up*) e conhecimento *a priori* sobre a localização, tamanho e forma das estruturas (abordagem *top-down*). Desta forma conseguem acomodar a grande variabilidade que as estruturas biológicas podem apresentar ao longo do tempo e entre indivíduos e proporcionar mecanismos de interacção com o utilizador muito intuitivos. Têm ainda a vantagem de permitir, em geral, uma precisão sub-*pixel*, característica muito interessante nas aplicações de imagem médica [8].

A designação modelo deformável deve-se a Terzopoulos e seus colaboradores [10], embora a ideia de deformar um modelo para extrair estruturas de imagens seja anterior [8].

A segmentação de contornos baseada em modelos deformáveis tem sido considerada frequentemente como um dos maiores sucessos da Visão por Computador nas últimas décadas. Na imagem médica tem sido um dos campos em que a sua aplicação se revelou mais fértil.

Neste artigo faremos uma breve introdução aos modelos deformáveis, apresentando as principais características, vantagens e desvantagens dos mais usados. Faremos ainda uma revisão de trabalhos em que estes modelos têm sido usados para segmentação de estruturas em imagens médicas, não sendo nosso objectivo apresentar trabalhos em que tenham sido usados para outros fins, como co-registo (*matching*) ou seguimento de movimento. Revisões extensas sobre estes modelos e sua utilização podem ser encontradas em [14] [8] [16] [17].

II. FUNDAMENTOS MATEMÁTICOS

A designação "modelo deformável" abrange muitos métodos. A maioria das estratégias usadas por este tipo de modelos passa pela optimização de funções objectivo, procurando encontrar um compromisso entre um termo de energia baseado na imagem e outro termo relacionado com uma energia interna ou modelo de forma (tipicamente a suavidade de pontos adjacentes). Uma alternativa aos modelos baseados na optimização de uma função objectivo, consiste em formular a deformação de um contorno como uma frente de onda que se propaga, que pode ser considerada como uma iso-linha de uma função envolvente. Os fundamentos matemáticos deste tipo de modelos podem ser encontrados em [21] e [14]. Neste trabalho limitamo-nos a apresentar a formulação matemática original correspondente aos contornos activos ou *snakes* [10].

As *snakes* podem ser imaginadas como curvas definidas no domínio da imagem, que se podem deslocar sob a influência de forças internas (definidas na própria curva) e forças externas calculadas a partir da imagem ou de processos de alto nível [24]. As primeiras mantêm o modelo suave durante a deformação, enquanto que as

últimas fazem o modelo mover-se em direcção às fronteiras do objecto ou outras características de interesse da imagem.

Os fundamentos matemáticos das *snakes* resultam da confluência da geometria, física e teoria da aproximação. A geometria permite a representação da forma do objecto, a física impõe restrições à variação da forma ao longo do espaço e do tempo e a teoria da aproximação permite formalizar os mecanismos que possibilitam o ajuste do modelo aos dados [8] [17].

O funcional de energia a minimizar é uma combinação pesada das forças referidas, sendo uma *snake* definida parametricamente como $v(s) = [x(s), y(s)]$, em que $x(s)$ e $y(s)$ são as coordenadas x, y ao longo do contorno e o parâmetro $s \in [0,1]$. O funcional de energia a minimizar pode ser escrito como:

$$E^*_{snake} = \int_0^1 E_{snake}(v(s))ds = \int_0^1 \{[E_{int}(v(s))] + [E_{imag}(v(s))] + [E_{con}(v(s))]\}ds \quad (1)$$

Em que E_{int}, E_{imag} e E_{con} representam respectivamente a energia interna da *snake*, as forças da imagem e as forças externas de restrição.

A energia interna da *snake* pode ser expressa como:

$$E_{int} = \alpha(s)\left|\frac{dv}{ds}\right|^2 + \beta(s)\left|\frac{d^2v}{ds^2}\right| \quad (2)$$

em que $\alpha(s)$ e $\beta(s)$ especificam a elasticidade e rigidez da *snake*. Enquanto que o primeiro termo faz o modelo comportar-se como um elástico, o segundo fá-lo comportar-se como um corpo rígido.

O segundo termo de (1) é obtido a partir da imagem e pode ser uma combinação pesada de funcionais que atraiam a *snake* para características de interesse da imagem como por exemplo:

$$E_{imag} = w_{linha}E_{linha} + w_{front}E_{front} + w_{term}E_{term} \quad (3)$$

O ajuste dos pesos w permite controlar o comportamento da *snake*. Por exemplo, se $f(x)$ for o nível de cinzento da imagem em (x,y), o funcional correspondente a uma *snake* que seja atraída por contornos com gradientes de imagem elevados pode ser:

$$E_{front} = -\left|\nabla f(x,y)\right|^2 \quad (4)$$

O terceiro termo de (1) exprime as restrições externas impostas quer por um utilizador quer por um processo de alto nível que atraia ou afaste a *snake* de propriedades específicas da imagem. Por exemplo, se a *snake* estiver na proximidade de alguma propriedade desejável da imagem, será atraída através do processo de minimização de energia; no entanto, se se detiver numa zona correspondente a um mínimo local de energia que um processo de mais alto nível considere como uma localização incorrecta, pode ser forçada, através deste último termo, a deslocar-se para outro mínimo local [10].

De acordo com a condição de Euler-Lagrange, a *snake* $v(s)$ que minimiza E^*_{snake}, deve satisfazer:

$$\frac{d}{ds}E_{v_s} - E_v = 0 \quad (5)$$

em que E_{vs} é a derivada parcial de E em relação a dv/ds e E_v é a derivada parcial de E em relação a v.

A minimização da energia de uma *snake* é um problema complexo. Isto deve-se não só ao facto de a resolução da equação de Euler Lagrange ter problemas de instabilidade numérica, como também à necessidade de proceder ao ajuste de numerosos parâmetros (como factores de ponderação, número de iterações) e a uma inicialização suficientemente próxima.

Originalmente foi proposta uma solução com recurso ao Método das Diferenças Finitas [10], no entanto esta solução apresenta problemas de instabilidade numérica, tendo sido propostas mais tarde outras alternativas, como a utilização do Método dos Elementos Finitos [3, 6, 38-40].

Esta formulação corresponde a considerar o problema como estático. Por vezes é mais conveniente formular o problema como dinâmico, usando directamente forças, o que permite a utilização de forças externas mais gerais. Em [14] pode encontrar-se esta formulação.

III. MODELOS DEFORMÁVEIS TRADICIONAIS

Nesta secção fazemos uma revisão das principais características dos tipos de modelos deformáveis mais utilizados.

A- Contornos Activos ou Snakes

Os modelos deformáveis paramétricos podem ser vistos como um caso especial de uma técnica mais geral de ajustar um modelo deformável a uma imagem por minimização de energia. Estes modelos necessitam de uma inicialização próxima do contorno desejado. A *snake* é então impelida em direcção a uma solução apropriada. Ao contrário da maioria dos outros modelos usados em análise de imagem, as *snakes* são activas no sentido em que se minimiza o funcional de energia pelo que exibem um comportamento dinâmico; a designação contornos activos provem desta característica [10] [24].

Outra característica interessante das *snakes* é o facto de permitirem um tratamento unificado para um conjunto de problemas de visão que tradicionalmente eram tratados independentemente. Essencialmente o mesmo mecanismo permite a detecção de contornos e contornos subjectivos, bem como o seguimento do seu movimento e o co-registo em situações de visão estereoscópica.

É ainda de referir que, de acordo com os seus autores [10], as *snakes* provavelmente incorporam, mais do que o próprio *sketch* 2.5D, a noção de "menor comprometimento", proposta em [49].

A pesar de serem muito utilizadas em imagens médicas, as *snakes* apresentam limitações, sendo as mais importantes o problema da inicialização, já referido, a falta de flexibilidade e a impossibilidade de alteração da topologia. De facto, a inicialização das *snakes* é crítica, frequentemente a curva inicial tem que ser relativamente

próxima do contorno a segmentar para que a segmentação seja bem sucedida devido à presença de características na imagem que, não pertencendo ao objecto podem forçar a *snake* a convergir para um mínimo local de energia. Também a parametrização fixa das *snakes* torna difícil a sua deformação assumindo formas tubulares ou com concavidades e protuberâncias acentuadas. Finalmente, a topologia do objecto de interesse tem que ser conhecida à partida pois as *snakes* são incapazes de transformações topológicas sem recurso a mecanismos adicionais.

Com o objectivo de desenvolver um *framework* unificado que permitisse ultrapassar as limitações das *snakes*, mantendo no entanto as suas as capacidades, McInerney e Terzopoulos [15] propuseram as *T-snakes* (*topology adaptable snakes*). Esta variante dos modelos paramétricos inclui a capacidade de re-parametrização que permite modificar a sua topologia. Utiliza uma grelha que os autores denominam de ACID (*Affine Cell Image Decomposition*), que introduz um mecanismo eficiente de reparametrização, permitindo aos contornos evoluírem de acordo com geometrias complexas, podendo mesmo alterar a sua topologia. À medida que uma *T-snake* se deforma de acordo com a influência de forças externas e internas, é periodicamente reparametrizada com um novo conjunto elementos através do cálculo dos pontos de intersecção do modelo com a grelha de células sobreposta. Durante a reparametrização o interior da *T-snake* é também seguido. A conversão para uma *snake* convencional envolve apenas a não utilização da grelha e pode ocorrer em qualquer momento. Quando uma *T-snake* colide com ela própria ou com outra, se divide em duas ou mais partes, ou desaparece, a sua topologia altera-se com a ajuda da grelha.

B - Level Set

Os modelos geométricos ou *level set* constituem uma alternativa aos modelos baseados na optimização de uma função objectivo; a deformação do contorno é formulada como uma frente de onda que se propaga e que pode ser considerada como *level set* de valor zero de uma função envolvente. Esta função envolvente pode ser expressa na forma de uma equação diferencial parcial em que um termo de velocidade força a paragem da propagação de acordo com informação obtida a partir da imagem.

Os esforços nesta área têm as suas raízes em [21] e têm sido aplicados a imagem médica por vários autores. Este método fornece a base para o processo numérico usado pelos métodos designados em [14] por modelos geométricos.

O processo de segmentação incorpora uma curva inicial que é o *level set* de nível zero de uma superfície de dimensão superior e faz evoluir esta superfície por forma a que o *level set* de nível zero convirja para o objecto a segmentar.

A perspectiva usada corresponde a uma formulação Euleriana do movimento e não Lagrangiana como é o caso dos modelos paramétricos [14] [16]. Uma propriedade útil desta abordagem é o facto da função *level set* se manter válida mesmo quando a curva altera a sua topologia, uma vantagem em relação aos modelos paramétricos.

Uma das características mais notáveis desta aproximação é a facilidade de generalização a dimensões superiores [20].

No entanto, a facilidade de adaptação da topologia, útil em muitas aplicações, pode por vezes, conduzir a resultados indesejáveis, produzindo formas com topologia não consistente com a do objecto a detectar. Outra desvantagem destes modelos é o facto da formulação implícita, apesar constituir um sistema matemático válido, ser muito menos conveniente que a explícita quanto à facilidade de incorporação de mecanismos de controlo tais como forças externas adaptadas às imagens ou à interacção do utilizador.

Caselles et al., utilizando uma formulação de minimização de energia, demonstraram, primeiro para 2D [58] e depois para 3D [19], a relação existente entre modelos que utilizam funções velocidade induzidas por forças potencias (o que acontece na maior parte das variantes) e modelos paramétricos que não incluem o termo de rigidez. Mais tarde, Xu [14] obtiveram uma relação matemática explícita entre uma formulação de força dinâmica para modelos deformáveis paramétricos e uma formulação para modelos geométricos, permitindo a utilização de funções velocidade derivadas de forças não potenciais (i.e. forças que não podem ser expressas como o negativo do gradiente de funções de energia potencial).

C – ASM AAM

Os *Active Shape Models* (ASM) propostos por Cootes [33] usam para incorporar conhecimento *a priori*, uma abordagem não baseada em parametrização mas em várias características relevantes da imagem (*landmarks*) num conjunto de treino.

Cada objecto ou estrutura a segmentar é representado por um conjunto de *landmarks* colocados manualmente em cada uma das imagens do conjunto de treino. Depois os conjuntos de pontos são alinhados automaticamente por forma a minimizar a distância entre pontos correspondentes. Analisando estatisticamente a variância da distância entre estes pontos obtém-se o chamado *Point Distribution Model* (PDM) [24] que é utilizado para restringir a variação da forma, ao longo da deformação de cada instância, à variância conhecida. É também criado um modelo de aparência dos níveis de cinzento limitado aos contornos do objecto, que consiste na 1ª derivada normalizada dos perfis centrados em cada *landmark*. A função de energia a ser minimizada é a distância de Mahalanobis destes perfis. O ajuste utiliza uma abordagem multiresolução.

Ginneken [37] propôs outro método de segmentação do tipo ASM que utiliza características locais óptimas em vez dos perfis normalizados da derivada de 1ª ordem e um classificador kNN em vez da distância de Mahalanobis para determinar as deslocações óptimas dos *landmarks*.

De acordo com os autores este método deverá ser especialmente útil na segmentação de objectos texturados em fundos texturados.

O paradigma dos ASM foi posteriormente alargado [59] por forma a incorporar informação *a priori* sobre os níveis de cinzento no interior do objecto, nos *Active Appearence Models* (AAM). Neste caso é feita uma Análise em Componentes Principais das *landmarks* e dos valores dos níveis de cinzento no interior do objecto, o que permite gerar instâncias plausíveis não só da geometria, mas também da textura. O processo de optimização da segmentação é guiado pela diferença entre os valores reais dos níveis de cinzento e os valores modelados.

Sclaroff [60] propôs um método comparável em que o objecto é modelado em termos de elementos finitos. Embora haja diferenças, estes métodos apresentam as seguintes características em comum: i) um modelo de forma assegura que a segmentação apenas pode produzir formas plausíveis; ii) um modelo de aparência de níveis de cinzento assegura a colocação do objecto numa localização onde a estrutura da imagem, à volta do contorno e dentro do objecto, é semelhante à esperada a partir das imagens de treino; iii) um algoritmo para ajustar o modelo através da minimização de uma função de custo, em geral, usando uma abordagem multiresolução.

O paradigma multiresolução é utilizado nos ASM e AAM para permitir, por um lado uma boa localização inicial aproximada baseada na estrutura global da imagem e, por outro lado, um refinamento da segmentação (a resoluções maiores), podendo produzir benefícios tanto na qualidade do ajuste final como no custo computacional.

Como estes métodos se baseiam em protótipos, são facilmente adaptáveis a novas aplicações, bastando a substituição do protótipo. A grande limitação à utilização destes modelos é a necessidade de colocar *landmarks* nas imagens de treino, o que se torna demasiado trabalhoso em algumas aplicações. O desenvolvimento de melhores métodos automáticos de anotação de imagens poderá minorar este problema.

D - Deformable Templates

Os *deformable templates* (também conhecidos por modelos *handcrafted*) permitem a utilização de formas complexas, através da utilização de um modelo de forma paramétrico com poucos graus de liberdade. Este modelo é ajustado à imagem de forma semelhante às *snakes*, procurando o valor do vector de parâmetros que minimize a energia externa. A energia interna pode ser usada como "regularizadora" tal que favoreça certas formas. Uma das primeiras abordagens deste tipo, e um exemplo muito citado [61] [62] [63] [16], foi a utilizada por Yuille [63] para extracção de características em faces. Este método é constituído por três partes fundamentais: um modelo parametrizado da forma do objecto a detectar (que contem as relações entre as várias características do objecto); um modelo da imagem e um algoritmo que controla iterativamente os parâmetros para minimizar uma energia total. O modelo utilizado para os olhos é não linear (usa uma circunferência e duas parábolas) e inclui vários parâmetros geométricos. O círculo, por exemplo, poderia ser extraído através da transformada de Hough, no entanto isto não é possível para a combinação das parábolas com a circunferência. A utilização de um *deformable template* possibilita a combinação das duas formas, permitindo a sua variação em tamanho e orientação, mantendo simultaneamente a sua relação espacial (a circunferência entre as parábolas).

Os métodos deste tipo têm a desvantagem de serem demasiado dependentes da aplicação e de nada garantir que o modelo (projectado por alguém) e a função de custo utilizados correspondam à melhor alternativa para a estrutura a extrair.

E – Modelos Probabilísticos

Os modelos deformáveis podem ser concebidos como um processo de ajuste num enquadramento probabilístico, que incorpora um modelo *à priori* com características em termos de distribuição de probabilidade [45].

Uma função de probabilidade *a priori* é inicialmente definida, manualmente ou semi-automaticamente, delimitando estruturas da mesma classe da estrutura a ser extraída [45]. Depois, essas estruturas são parametrizadas de acordo com coeficientes de Fourier, ou usando um conjunto de parâmetros baseados em elipsoides, sendo determinada a média e a variância para cada um desses parâmetros. Assumindo independência entre parâmetros, é determinada a função Gaussiana de probabilidade *à priori*.

Por último é definida uma função de probabilidade *à posterior*, que pesa o modelo de probabilidade *à priori* e o modelo de dados, determinado a discrepância entre as características da fronteira e o contorno deformável [64] [65] [66] [14].

F - Modelos Deformáveis com Modelação da Forma Global

Diferentes tipos de modelos deformáveis usam uma decomposição modal do modelo. A base da decomposição é um conjunto de harmónicos de frequência diferente. Esta representação é equivalente a um conjunto de curvas ou superfícies paramétricas cujos parâmetros são os pesos dos diversos modos. A soma dos primeiros modos fornece uma aproximação grosseira da sua forma, que é refinada quando se adicionam modos de frequência mais elevada.

Na prática é desejável reduzir, tanto quanto possível, o número de modos usados para obter uma representação compacta de formas relativamente complexas. A escolha deste número resulta de um compromisso entre a precisão, a concisão e a suavidade desejadas.

É possível fazer o *mapping* dos coeficientes de Fourier para um conjunto de parâmetros que descreve a forma dum objecto. Estes parâmetros seguem uma ordenação tal como os coeficientes de Fourier, pelo que os de menor índice descrevem propriedades globais e os de maior

índice descrevem deformações mais locais. Staib e Duncan [45] propuseram um modelo deformável utilizando uma decomposição modal de Fourier. Esta abordagem pode ser alargada a superfícies com funções harmónicas esféricas, como proposto, por exemplo, em [34], em que as representações das superfícies são expandidas numa série de funções harmónicas esféricas cujos coeficientes fornecem uma descrição paramétrica das formas dos objectos.

Os modelos deformáveis com superquádricas [67] são outra extensão aos modelos deformáveis que utilizam modelos globais de forma, incorporam informação global além de informação local. Uma superfície superquadrática (que pode ser definida a partir de um número reduzido de parâmetros) deforma-se localmente para reconstruir a forma do objecto. Embora o ajuste local e global seja feito simultaneamente, a deformação global é forçada a ter em conta, tanto quanto possível, a forma do objecto. A superfície superquadrática estimada incorpora as características globais da forma do objecto, enquanto que as deformações locais incorporam os detalhes.

IV. APLICAÇÕES PARA SEGMENTAÇÃO

Os modelos deformáveis mais usados são os modelos deformáveis paramétricos, geométricos e ainda os A.S.M./A.A.M, conforme se pode observar na tabela 1. Alguns investigadores têm desenvolvido novos modelos deformáveis, aplicados em imagens médicas, desde os mais simples, usando um *threshold* apropriado seguido de operações morfológicas para detectar as transições e por fim ajustando a forma com uma superquádrica [55], até aos modelos mais complexos (sob o ponto de vista matemático) recorrendo a um misto de um modelo deformável paramétrico com superfícies definidas por harmónicos esféricos [50] ou com re-parametrização das superfícies [15].

Na tabela 1 indicam-se os órgãos segmentados pelos modelos deformáveis referidos ao longo deste trabalho.

Os modelos deformáveis paramétricos na sua forma mais simples ou com superfícies definidas por objectos geométricos (desde esferas, passando por superelipsoides, até harmónicos esféricos) tem sido o modelo mais referido nos artigos dos investigadores, não só pela variedade de órgãos segmentados mas também por servir de base a variantes deste modelo deformável, com a utilização dos conceitos de minimização de energia interna e externa usando a equação de Euler-Lagrande seguido de uma discretização pelos Métodos das Diferenças Finitas ou dos Elementos Finitos ou parametrização usando uma equação dinâmica com recurso aos conceitos de deslocação, velocidade e aceleração.

Tabela 1 –Modelos deformáveis aplicados em órgãos ou estruturas de órgãos

Modelo Deformável:	Autor	Ano	Órgãos ou estruturas segmentadas
Paramétrico	Amini	1990	Células [1]
	Cohen	1992	Ventrículos cardíacos [2] [3]
	Leymarie	1993	Células isoladas [4]
	McInerney	1995	Ventrículos cardíacos [6]
	Lobregt	1995	Canais sanguíneos, tumores no cérebro [7]
	Yezzi	1997	Ventrículos cardíacos [9]
	Atkins	1998	Cérebro [11]
	Xu	1998	Ventrículos cardíacos [12]
	Honea	1999	Nódulos linfáticos (lymph nodules) [13]
	Xu	1999	Ventrículos cardíacos, cérebro [14]
	McInerney	1999	Cérebro, vértebras, arvore vascular cerebral [15]
	McInerney	2000	Fantoma de vértebras, Ventrículos cardíacos [8]
Geométrico	Malladi	1995	Estômago, arvore sanguínea [18]
	Caselles	1997	Osso em MRI, tumores em MRI [19]
	Sethian	1997	Fígado, baço, Ventrículos [20]
	Siddiqi	1998	Cérebro, Ventrículos cardíacos [22]
	Kawata	1998	Nódulos pulmonares [23]
	Xu	1999	Ultra-som do peito, Cérebro [14]
	Kovacevic	1999	Canais sanguíneos [25]
	Zeng	1999	Cérebro [26]
	Wang	2000	Cérebro, ventrículos cardíacos [27]
	Ye	2001	Fantoma Cérebro [28]
	Suri	2002	Cérebro [29]
Active Shape Models / *Active Apprearance Models*	Cootes	1994	Ventrículos cerebrais, Ecocardiograma, [30]
	Smyth	1997	Coluna (vértebras) [31]
	Cootes	1998	Joelho (em imagens de raios X) [32]
	Cootes	1999	Ventrículos cardíacos [33]
	Kelemen	1999	Cérebro [34]
	Stegmann	2000	Mão em imagens de raios-x, endocardiograma [35]
	Frangi	2002	Ventrículos cardíacos [36]
	Ginneken	2002	Cerebrelo (Cérebro) [37]
	Mitchell	2002	Ventrículos cardíacos, ecocardiograma [41]
Deformable templates	Escolano	1997	Intra-vascular [42]
	Rueckert	1997	Ventrículos cardíacos [43]
	Wang	1998	Ventrículos cerebrais, Endocardio do cardíacos [44]
Probabilísticos	Staib	1992	Coração, Ventrículos cerebrais [45]
	Lundervold	1995	Ventrículos cerebrais, [46]
	Vincken	1995	Ventrículos cerebrais, [47]
	Vincken	1997	Cérebro [48]
	Wang	1998	Ventrículos cerebrais, Endocardio (do coração) [44]
com harmónicos esféricos	Szekely	1996	Cérebro [50]
	Haigron	1998	Vértebras lombares, ventrículo cardíacos [51]
	Kelemen	1999	Cérebro [34]
	Quicken	2000	Bexiga, próstata, cérebro 3d [52]
	Gerig	2001	Ventrículos cerebrais [53]
	Weistrand	2001	Cérebro [Weistrand,01
com Superquádricas ou Hiperquádricas	Bardinet	1994	Miocardio [54] [55]
	Cohen	1994	Ventrículo cardíacos [56]
	Kumar	1995	Ventrículo cardíacos [57]

V. CONCLUSÕES

Os modelos deformáveis têm-se revelado uma abordagem muito útil ao problema da segmentação de estruturas em imagens médicas. Este problema é considerado difícil devido ao tamanho dos conjuntos de dados e à complexidade e variabilidade das formas de interesse, bem como às limitações típicas dos dados amostrados que podem tornar indistintos e fragmentados os contornos das estruturas a segmentar.

Numerosas variantes destes modelos têm vindo a ser propostas com o objectivo de ultrapassar as limitações dos modelos previamente existentes. Estas limitações estão relacionadas com questões como autonomia, facilidade de controlo pelo utilizador, generalidade e especificidade, flexibilidade da topologia, concisão e abrangência da representação geométrica, precisão e robustez ao ruído e à inicialização.

Neste trabalho procuramos dar uma perspectiva de quais os principais tipos de modelos deformáveis existentes bem como da sua utilização em cenários de segmentação em imagens médicas.

Apesar de todo o sucesso que os modelos deformáveis têm tido nas últimas décadas, ainda recentemente Duncan e Ayache [68] consideraram não existir nenhum algoritmo que conseguisse segmentar de forma robusta uma grande variedade de estruturas relevantes em imagem médica, numa gama apreciável de dados [68]. Os autores deste trabalho consideram que a situação na corrente data ainda se mantém e provavelmente manterá durante mais algum tempo, já que os algoritmos de segmentação de contornos ou superfícies baseados em modelos deformáveis são em geral sensíveis aos parâmetros de aquisição da imagem e à sua própria posição inicial, entre outros. A investigação nesta área continua activa, procurando-se obter algoritmos cada vez mais robustos ao ruído e à inicialização, bem como a outras características da imagem que dificultam a segmentação.

REFERÊNCIAS

[1] A. Amini, T. E. Weymouth, and R. C. Jain, "Using Dynamic Programming for Solving Variational Problems in Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 855-867, 1990.

[2] I. Cohen, L. D. Cohen, and N. Ayache, "Using Deformable Surfaces to Segment 3D Images and Infer Differential Structures," *CVGIP: Image Understanding*, vol. 56, no. 2, pp. 242-263, 1992.

[3] L. D. Cohen and I. Cohen, "Finite-Element Methods for active Contour Models and balloons for 2D and 3D Images.," *IEEE Transactions on Pattern Analisys and Machine Intelligence*, vol. 15, no. 11, pp. 1131-1147, 1993.

[4] F. Leymarie and M. D. Levine, "Tracking deformable Objects in the Plane Using an Active Contour Model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 617-634, 1993.

[5] W. J. Niessen and M. A. Viergever, "Guest Editorial," *IEEE Transactions on Image Processing*, vol. 18, no. 10, pp. 825-827, 1999.

[6] T. McInerney and D. Terzopoulos, "A Dynamic Finite Element Surface Model for Segmentation and Tracking in Multidimensional Medical Images with Application to Cardiac 4D Image Analysis," *Journal of Computerized Medical Imaging and Graphics*, vol. 19, no. 1, pp. 69-83, 1995.

[7] S. Lobregt and M. A. Viergever, "A Discrete Dynamic Contour Model," *IEEE Transactions on Medical Imaging*, vol. 14, no. 1, pp. 12-24, 1995.

[8] T. McInerney and D. Terzopoulos, "Deformable Models," in *Handbook of Medical Imaging Processing and Analysing*, I. N. Bankman, Ed. San Diego - USA: Academic Press, 2000, pp. 127-145.

[9] A. Yezzi, S. Kichenassamy, A. Kumar, P. Olver, and A. Tannenbaum, "A Geometric Snake Model for Segmentation of Medical Imagery," *IEEE Transactions on Medical Imaging*, vol. 16, no. 2, pp. 199-209, 1997.

[10] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *International Journal of Computer Vision*, pp. 321-331, 1988.

[11] M. S. Atkins and B. T. Mackiewich, "Fully Automatic Segmentation of the Brain in MRI," *IEEE Transactions on Medical Imaging*, vol. 17, no. 1, pp. 98-107, 1998.

[12] C. Xu and J. L. Prince, "Snakes, Shapes and Gradient Vector Flow," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 359-369, 1998.

[13] D. M. Honea and W. E. Snyder, "3D Active surface approach to Lymph Node Segmentation", *SPIE - Medical Imaging 1999: Image Processing*, vol. 3661, pp. 1003-1011, U.S.A., 1999.

[14] C. Xu, D. L. Pham, and J. L. Prince, *Image Segmentation Using Deformable Models (cap. III)*, vol. 2: SPIE - The International Society for Optical Engineering, 1999.

[15] T. McInerney and D. Terzopoulos, "Topology Adaptive Deformable Surfaces for Medical Image Volume Segmentation," *IEEE Transactions on Medical Imaging*, vol. 18, no. 10, pp. 840-850, 1999.

[16] J. Montagnat, H. Delingette, and N. Ayache, "A Review of Deformable Surfaces: Topology, Geometry and Deformation," *Image and Vision Computing*, vol. 19, no. 14, pp. 1023-1040, 2001.

[17] J. Tavares, J. Barbosa, and A. Padilha, "Modelos Deformáveis em Imagens Médicas," Faculdade de Engenharia da Universidade do Porto, Porto - Portugal (Relatório Interno), Janeiro 2003.

[18] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape Modeling with Front Propagation. A Level Set Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, pp. 158-175, 1995.

[19] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert, "Minimal Surfaces Based Object Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 394-398, 1997.

[20] J. A. Sethian, "Level Set Methods: An Act of Violence," *American Scientist*, vol. 85, no. 3, 1997.

[21] S. Osher and J. A. Sethian, "Fronts Propagation with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations," *Journal of Computational Physics*, vol. 79, pp. 12-49, 1988.

[22] K. Siddiqi, Y. B. Lauziere, A. Tannenbaum, and S. W. Zucker, "Area and Length Minimizing Flows for Shape Segmentation," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 433-443, 1998.

[23] Y. Kawata, N. Niki, H. Ohmatsu, R. Kakinuma, K. Eguchi, R. Kaneko, and N. Moriyama, "Quantitative surface characterization of pulmonary nodules based on thin-section CT images," *IEEE*

Transactions on Nuclear Science, vol. 45, no. 4, pp. 1218-1222, 1998.

[24]R. Uppaluri, E. A. Hoffman, M. Sonka, P. G. Hartley, G. W. Huuninghake, and G. McLennan, "Computer Recognition of Regional Lung Disease Patterns," *American Journal of Respiratory and Critical Care Medicine*, vol. 160, pp. 648-653, 1999.

[25]D. Kovacevic, S. Loncaric, and E. Sorantin, "Deformable Contour Based Method for Medical Image Segmentation", *21st International Conference on Information Technology Interfaces ITI'99*, Pula - Coatia, 1999.

[26]X. Zeng, L. H. Staib, R. T. Schultz, and J. S. Duncan, "Segmentation and Measurement of the Cortex from 3D MR Images Using Coupled-Surfaces Propagation," *IEEE Transactions on Medical Imaging*, vol. 18, no. 10, pp. 927-937, 1999.

[27]H. Wang and B. Ghosh, "Geometric Active Deformable Models in Shape Modeling," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 302-308, 2000.

[28]J. C. Ye, Y. Bresler, and P. Moulin, "A Self-Referencing Level-Set Method for Image Reconstruction From Spare Fourier Samples", *IEEE Workshop on Variational and Level-Set Methods (VLSM'01)*, pp. 171, Vancouver-Canada, 2001.

[29]J. S. Suri, K. Liu, S. Singh, S. N. Laxminarayan, X. Zeng, and L. Reden, "Shape Recover Algorithms Using Levels Set in 2D/3D Medical Imagery: A State of the Art Review," *IEEE Transactions on Information Technology in BioMedicine*, vol. 6, no. 1, pp. 8-28, 2002.

[30]T. F. Cootes, A. Hill, C. J. Taylor, and J. Haslam, "The Use of Active Shape Models For Locating Structures in Medical Images," *Image and Vision Computing*, vol. 12, no. 6, pp. 355-366, 1994.

[31]P. P. Smyth, C. J. Taylor, and J. E. Adams, "Automatic measurement of vertebral shape using active shape models," *Image and Vision Computing*, vol. 15, pp. 575-581, 1997.

[32]T. F. Cootes, G. F. Edwards, and C. J. Taylor, "Active Appearance Models", *Proceedings of 5th European Conference on Computer Vision*, vol. 2, pp. 484-498, Springer-Berlin, 1998.

[33]T. F. Cootes, G. Edwards, and C. J. Taylor, "Comparing Active Shape Models with Active Appearance Models", *Proceedings of British Machine Vision Conference*, pp. 173-182, 1999.

[34]A. Kelemen, G. Szekely, and G. Gerig, "Elastic Model Based Segmentation of 3D Neuroradiological Data Sets," *IEEE Transactions on Medical Imaging*, vol. 18, no. 10, pp. 828-838, 1999.

[35]M. B. Stegmann, R. Fisker, B. K. Ersboll, H. H. Thodberg, and L. Hyldstrup, "Active Appearance Models: Theory and Cases", *Proceedings of 9th Danish Conference on Pattern Recognition and Image Analysis*, vol. 1, pp. 49-57, 2000.

[36]A. F. Frangi, D. Rueckert, J. A. Schnabel, and W. J. Niessen, "Automatic Construction of Multiple Object Three Dimensional Statistical Shape Models: Application to Cardiac Modeling," *IEEE Transactions on Medical Imaging*, vol. 21, no. 9, pp. 1151-1166, 2002.

[37]B. v. Ginneken, A. F. Frangi, J. J. Staal, B. M. t. H. Romeny, and M. A. Viergever, "Active Shape Model Segmentation with Optimal Features," *IEEE Transactions on Medical Imaging*, vol. 21, no. 8, pp. 924-933, 2002.

[38]D. Braess, *Finite Elements: Theory, Fast Solvers and Applications in Solid Mechanics*: Cambridge University Press, 2001.

[39]T. McInerney and D. Terzopoulos, "Finite Element Techniques for Fitting a Deformable Model to 3D Data", *Vision Interface'93*, pp. 70-76, Toronto - Canada, 1993.

[40]J. W. Thomas, *Numerical Partial Differential Equations: Finite Difference Methods*. New York: Springer-Verlag, 1995.

[41]S. C. Mitchell, J. G. Bosch, B. P. F. Lelieveldt, R. J. Geest, J. H. C. Reiber, and M. Sonka, "3D Active Appearance Models: Segmentation of Cardiac MR and Ultrasound Images," *IEEE Transactions on Medical Imaging*, vol. 21, no. 9, pp. 1167-1178, 2002.

[42]F. Escolano, M. Cazorla, D. Gallardo, and R. Rizo, "Deformable Templates for Tracking and Analysis of Intravascular Ultrasound Sequences", *Proceedings of First International Workshop of Energy Minimization Methods in Computer Vision and Pattern Recognition*, vol. 1223, pp. 521-534, Venice-Italy, 1997.

[43]D. Rueckert and P. Burger, "Deformable Templates for Tracking and Analysis of Intravascular Ultrasound Sequences", *Proceedings of First International Workshop of Energy Minimization Methods in Computer Vision and Pattern Recognition*, vol. 1223, pp. 83-98, Venice-Italy, 1997.

[44]Y. Wang and L. H. Staib, "Boundary Finding with Correspondence Using Statistical Shape Models", *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 338-345, 1998.

[45]L. H. Staib and J. S. Duncan, "Boundary Finding with Parametrically Deformable Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 11, pp. 1061-1075, 1992.

[46]A. Lundervold and G. Storvik, "Segmentation of Brain Parenchyma and Cerebrospinal Fluid in Multispectral Magnetic Resonance Images," *IEEE Transactions on Medical Imaging*, vol. 14, no. 2, pp. 339-349, 1995.

[47]K. L. Vincken, *Probabilistic Multiscale Image Segmentation by the Hyperstack*: Universitei Utrecht - German, 1995.

[48]K. L. Vincken, A. S. E. Koster, and M. A. Viergever, "Probabilistic Multiscale Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 109-120, 1997.

[49]D. Marr, *Vision - A Computational Investigation into the Human Representation and Processing of Visual Information*, 3rd ed. New York - USA: W.H.Freeman and Company, 1996.

[50]G. Szekely, A. Kelemen, C. Brechbuhler, and G. Gerig, "Segmentation of 2-D and 3-D objects from MRI volume data using constrained elastic deformations of flexible Fourier contour and surface models," *Medical Image Analysis*, vol. 1, no. 1, pp. 19-34, 1996.

[51]P. Haigron, G. Lefaix, X. Riot, and R. Collorec, "Application of Spherical Harmonics to the modeling of Anatomical Shapes," *Journal of Computing and Information Technology*, vol. 4, pp. 449-461, 1998.

[52]M. Quicken, C. Brechbuhler, J. Hug, H. Blattman, and G. Szekely, "Parameterization of Closed Surfaces for Parametric Surface Description", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2000*, Hilton Head Island - USA, 2000.

[53]G. Gerig, M. Styner, D. Jones, D. Weinberger, and J. Lieberman, "Shape Analysis of Brain Ventricles Using SPHARM", *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA2001)*, 2001.

[54]E. Bardinet, L. D. Cohen, and N. Ayache, "Fitting of Iso-Surfaces Using Superquadrics and Free-Form deformations", *Proceedings of IEEE Workshop on Biomedical Image Analysis (WBIA'94)*, Washington, 1994.

[55]E. Bardinet, L. D. Cohen, and N. Ayache, "Analyzing the deformation of the left ventricle of the heart with a parametric deformable model," *INRIA Research Report 2797, Sophia-Antipolis, France, February*, 1996.

[56]I. Cohen and L. D. Cohen, "A Hybrid Hyperquadric Model for 2D and 3D data Fitting," *INRIA Research Report #2188, France, January*, 1994.

[57]S. Kumar, s. Han, D. Goldgof, and K. Bowyer, "On Recovering Hyperquadrics from Range Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 11, pp. 1079-1083, 1995.

[58]V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic Active Contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61-79, 1997.

[59]T. F. Cootes, G. Edwards, and C. J. Taylor, "Active Appearance Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681-685, 2001.

[60]S. Sclaroff and A. Pentland, "Modal Matching for Correspondence and Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 6, pp. 545-561, 1995.

[61]A. Blake and M. Isard, *Active Contours*: Springer Verlag London Limited, 1998.

[62]A. Watt and F. Policarpo, *The Computer Image*: Addison Wesley, 1998.

[63]M. Nixon and A. Aguado, *Feature Extration & Image Processing*. Oxford - UK: Reed Education and Professional Publishing Ltd, 2002.

[64]T. McInerney and D. Terzopoulos, "Deformable Models in Medical Image Analysis: A Survey," *Medical Image Analysis*, vol. 1, no. 2, pp. 91-108, 1996.

[65]D. MacKay, "Bayesian Methods for Adaptive Models," phD thesis, California Institute of Technology, 1991.

[66]M. Werman and D. Karen, "A Bayesian Method for Fitting Parametric and Nonparametric Models to Noisy Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 5, pp. 528-534, 2001.

[67]D. Terzopoulos and D. Metaxas, "Dynamic 3D Models with Local and Global Deformations. Deformable Superquadrics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 7, pp. 703-714, 1991.

[68]J. S. Duncan and N. Ayache, "Medical Image Analysis: Progress over Two Decades and the Challenges Ahead," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 85-106, 2000.

Localização de Células em Imagens Histológicas usando Modelos baseados em Formas Activas

Luís Coelho*, Augusto Silva

Resumo - **A análise digital de imagem tem vindo a assumir um papel importante no desenvolvimento de diversas áreas. O presente artigo apresenta um estudo das técnicas de análise digital de imagens baseadas nos conceitos de forma activa e respectiva aplicação na localização de células em imagens histológicas.**

O artigo começa por estabelecer qual o classe de objectos a modelar e quais as variações que estes pode ter. Segue-se a descrição da técnica de modelação estatística e do algoritmo de actualização usado para localizar os objectos em imagens alvo . Para finalizar são expostos os resultados obtidos, com a aplicação dos modelos construídos na segmentação de núcleo e citoplasma em várias imagens alvo.

Abstract - **Digital image analysis is being increasingly used in many application areas. This paper presents a study of the Active Shape Models and their application for locating cells in histological images.**

The paper begins with the choice of the class of objects to be modelled including their variations. It follows with the description of the statistical modelling technique and the iterative updating algorithm used to locate corresponding objects within the target image. The paper concludes with a results section where segmented nuclei and cytoplasm are presented within a set of different target images.

I. INTRODUÇÃO

As ferramentas computacionais constituem uma ajuda importante na análise digital de imagem, sendo muitas as áreas que beneficiam destes métodos de análise. A investigação em Biologia é uma das áreas que mais recorre aos meios de imagem constituindo um desafio importante aos métodos de processamento digital.

A determinação de parâmetros e a identificação de estruturas que permitam caracterizar as imagens histológicas que se obtêm por microscopia, são exemplos de aplicação da análise digital de imagem. A caracterização rigorosa de populações de microorganismos é outro caso onde a aplicação das técnicas de análise digital, em conjugação com métodos de reconhecimento de padrões, também tem um contributo importante.

Na verdade, o processamento digital de imagens tem vindo a resolver uma quantidade cada vez maior de problemas, aumentando a precisão na detecção de estruturas de interesse em imagens cito e histológicas. Também as principais áreas da pesquisa em Medicina, tais como a pesquisa de tumores, são fortemente dependentes do desenvolvimento das técnicas de processamento de imagem, uma vez que necessitam de rapidez, rigor e objectividade nos seus diagnósticos [1].

Os modelos baseados em superfícies deformáveis permitem descrever uma grande variedade de superfícies lisas, tanto com contornos abertos como fechados, com um pequeno número de parâmetros e podem ser usados na detecção de objectos 3-D nas imagens médicas, tais como o cérebro e o coração, entre outros órgãos de interesse [2,3]. Os modelos activos de contorno (ACM), que são conhecidos por *"Snakes"* porque são atraídos às características da imagem, também têm aplicações nessa área [4].

Interessa para este artigo estudar os modelos baseados nos conceitos de forma activa *(Active Shape Models – ASM)*, também pertencentes à "classe" dos Modelos Deformáveis. Estes modelos têm aplicações conhecidas na análise de imagens médicas e de outras estruturas biológicas e também na análise e reconstrução facial, [5,6,7,8]. Será feita a avaliação às potencialidades destes métodos quando usados na identificação de estruturas histológicas, nomeadamente na localização de células em imagens histológicas.

Estes modelos têm a característica de serem bastante versáteis e usarem como base o modelo de distribuição de pontos (PDM). A sua implementação passa por várias fases de execução: construção de um conjunto de treino; alinhamento do conjunto de treino; tratamento estatístico das nuvens de pontos encontradas; e aplicação do modelo construído na identificação de um objecto numa dada imagem.

Este artigo começa por estabelecer qual o objecto a modelar e quais as variações que este pode ter. Segue-se a descrição da modelação do objecto; o uso desta modelação para localizar o objecto numa imagem, assim como a apresentação dos resultados da aplicação dos modelos construídos. Finalmente, são apresentadas algumas conclusões e propostas para trabalhos futuros.

* Financiado pela Unidade de Investigação 127/94 IEETA da Universidade de Aveiro.

II. VARIAÇÕES NA FORMA DOS OBJECTOS

O algoritmo *Active Shape Models* (ASM) tem algumas aplicações a nível da detecção de estruturas de interesse em imagens médicas, no reconhecimento facial, entre outras de menor relevância.

Pela variedade das aplicações pode-se constatar que estes modelos possuem um elevado grau de adaptação às várias formas que cada objecto pode tomar. Veja-se: O coração humano não tem exactamente a mesma forma em todas as imagens. A sua forma varia durante o batimento cardíaco e com a idade, para além de outras variações possíveis. Também a forma facial é alvo de muitas variações. Varia de pessoa para pessoa, e num mesmo indivíduo varia conforme a sua expressão facial e idade.

Apesar das variações descritas, a forma dos objectos de cada caso obedece a um certo padrão que se mantém em todas as imagens, de outra forma não seria possível usar, com sucesso, o algoritmo ASM para localizar estes objectos.

Também no caso das células pode encontrar-se um padrão, apesar da variação da sua forma ser muito superior à dos objectos apresentados anteriormente. Contudo, todas têm núcleo e citoplasma, sendo que o núcleo apresenta uma forma relativamente circular. Assim, pode-se usar algoritmo ASM para modelar um determinado tipo de células, de modo a localizar células desse tipo em diversas imagens histológicas.

Para se conseguir modelar as variações da forma de um determinado objecto é necessário possuir uma série de imagens, onde o objecto possua ligeiras variações na sua forma. A este conjunto de imagens dá-se o nome de conjunto de treino *("Training Set")*. É necessário fazer uma escolha cuidadosa do conjunto de treino para construir um bom modelo do objecto.

A forma de cada objecto do conjunto de treino é representada por um conjunto de pontos do seu contorno, designados por *landmarks*. O número de *landmarks* tem de ser escolhido de forma a representar, não só a variação geral da forma do objecto, mas também alguns detalhes que se revelem necessários. Cada objecto terá de ser representado por um número predefinido de pontos, tendo em conta a sua complexidade e o grau de pormenor com que se pretende representar.

Como os objectos são representados por pontos, a modelação da forma dos objectos é feita de segundo o Modelo de Distribuição de Pontos (PDM). No entanto, não basta só fazer a modelação da forma dos objectos, é igualmente importante fazer a modelação da variação dos níveis de cinzento em torno de cada ponto do objecto [8,9]. Esta modelação contribui para a criação de um sistema robusto de localização dos objectos numa imagem.

Descrevem-se, de seguida, as modelações feitas aos objectos: a modelação da forma e a modelação dos níveis de cinzento.

III. MODELAÇÃO DA VARIAÇÃO DA FORMA DOS OBJECTOS

A obtenção do PDM para um objecto obedece a três etapas de execução: a etiquetagem, o alinhamento e a captura das estatísticas do conjunto de treino. Segue-se a descrição detalhada de cada etapa.

Etiquetagem do Conjunto de Treino

A marcação dos pontos representativos de cada objecto obedece a determinadas regras. Uma das mais importantes tem a ver com o número de pontos que define cada objecto do conjunto de treino, que tem de ser o mesmo para todos os objectos. O número de pontos é escolhido tendo em conta a precisão com que se pretende representar o objecto.

Na marcação dos pontos é necessário estabelecer critérios de modo a que cada ponto marque características semelhantes em todos os objectos. Existem três tipos de *landmarks* que podem ser usados [5,6]:

- *Application-dependent landmarks*;
- *Application-independent landmark*;
- *Landmarks* interpolados a partir dos anteriores.

Os *landmarks* do primeiro tipo são usados para marcar aspectos que dependem de outros, por exemplo, o centro do olho num modelo da face ou um vértice de um objecto de cantos afiados. Os do segundo tipo são usados para marcar aspectos que não dependem de outros, por exemplo, o ponto mais elevado de um determinado objecto. E os pontos do terceiro tipo são igualmente espaçados de uma determinada distância entre dois pontos do tipo 1 ou do tipo 2. Na maior parte dos casos são os pontos do tipo 3 os responsáveis por caracterizar a maior parte do contorno dos objectos.

Denotando-se o número de pontos de cada objecto por n e o número de imagens do conjunto de treino por N, então o i^{esimo} objecto pode ser representado por:

$$X_i = [x_{i0}, y_{i0}, x_{i1}, y_{i1}, \ldots, x_{in-1}, y_{in-1}] \text{ com } 1 \le i \le N \quad (1)$$

No final da marcação de todos os pontos em todos os objectos do conjunto de treino obtém-se uma matriz **X** de dimensões $N \times 2n$.

$$X = \begin{bmatrix} x_{10} & y_{10} & x_{11} & y_{11} & \cdots & x_{1n-1} & y_{1n-1} \\ x_{20} & y_{20} & x_{21} & y_{21} & \cdots & x_{2n-1} & y_{2n-1} \\ & & & \cdots & & & \\ x_{N0} & y_{N0} & x_{N1} & y_{N1} & \cdots & x_{Nn-1} & y_{Nn-1} \end{bmatrix} \quad (2)$$

Alinhamento do Conjunto de Treino

Para que se estudem as variações da posição de cada *landmark*, em todos os objectos do conjunto de treino, é

necessário fazer o alinhamento de cada um dos objectos do conjunto. O alinhamento consiste em escalar, rodar e fazer a translação de cada um dos objectos, de modo a que todos os objectos fiquem numa posição semelhante, isto é, de modo a que os pontos de um objecto estejam tão próximo quanto possível dos pontos do outro objecto. O objectivo é minimizar a soma dos quadrados das distâncias entre pontos equivalentes em objectos diferentes [5].

A primeira tarefa a fazer, antes de se alinhar o conjunto, é determinar a matriz dos pesos de cada ponto, pois esta será importante para o alinhamento do conjunto. O cálculo da matriz dos pesos de cada ponto é efectuado de modo a

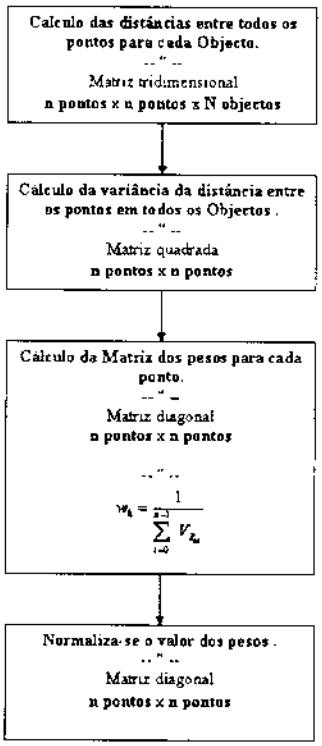


Fig. 1 - Fluxograma do algoritmo de escolha dos pesos de cada ponto.

atribuir maior peso aos pontos que tendem a ser mais estáveis, e um peso mais reduzido aos que tendem a variar dentro do conjunto [5,6]. O valor do peso de cada ponto é inversamente proporcional à variância desse ponto dentro do conjunto de treino. Na figura 1 está representado o fluxograma do algoritmo da escolha dos pesos.

O vector dos pesos pode ser determinado pela seguinte expressão:

$$W = diag\,(w_k) \qquad (3)$$

Alinhar dois objectos, $\mathbf{X}_i$ e $\mathbf{X}_j$, consiste em determinar o valor do factor de escala **S**, do ângulo de rotação θ, e da translação em ambas as dimensões (t_x, t_y) a aplicar a $\mathbf{X}_j$ de modo a que este fique alinhado com $\mathbf{X}_i$. A matriz dos pesos W é aplicada de forma a dar mais significado aos pontos mais estáveis [5]. A estabilidade de um ponto é a medida da variação desse ponto em relação aos outros, de objecto para objecto.

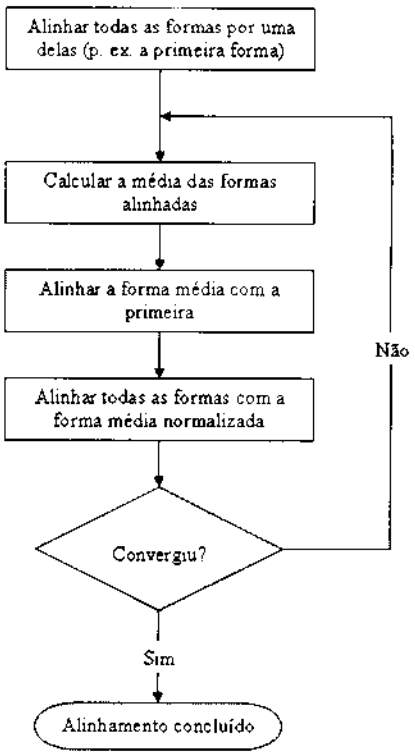


Fig. 2 - Fluxograma do algoritmo de alinhamento do conjunto de treino.

Generalizando, o alinhamento dos N objectos do conjunto de treino obedece ao seguinte algoritmo [9]:

O critério de convergência escolhido foi, S_{min} = 0.001; θ_{min} = 0.001°; $t_{x\,min}$ = $t_{y\,min}$ = 0.01, de modo a que o valor do erro

$$E = (X_i - M(S,\theta)[X_j] - t)^T W (X_i - M(S,\theta)[X_j] - t),$$

seja mínimo e em que $M(S,\theta)$ uma matriz de transformação que depende apenas dos factores de escala e de rotação [5].

Captura das Estatísticas do Conjunto de Treino

Após o alinhamento do conjunto de treino verifica-se a existência de uma nuvem difusa de pontos em torno de cada *landmark*, o que equivale a dizer que o objecto representado apresenta alguma variabilidade. O PDM procura modelar a variação destas nuvens através de um processamento estatístico do conjunto de treino.

A captura das estatísticas de um conjunto de objectos alinhados permite obter a forma média dos objectos, assim como os *vectores próprios* e respectivos *valores próprios*, capazes de explicar uma percentagem razoável da variação total da forma dos objectos do conjunto.

A forma média dos objectos é determinada usando:

$$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i \quad (4)$$

O desvio de cada objecto em relação à média é dado pela seguinte expressão:

$$dX_i = X_i - \overline{X} \quad (5)$$

Assim, é possível determinar a matriz de covariância usando:

$$C_x = \frac{1}{N}\sum_{i=1}^{N} dX_i dX_i^T \quad (6)$$

Os métodos de Análise em Componentes Principais (PCA) permitem determinar um conjunto de variáveis, chamadas de Componentes Principais, a partir da decomposição em *vectores/valores próprios* da matriz de covariância [9]. Assim, a diferença dX_i pode ser representada como uma combinação linear dos componentes principais.

$$dX_i = b_{i0}p_0 + b_{i1}p_1 + \ldots + b_{i2n-1}p_{2n-1} = Pb_i \quad (7)$$

Com, $b_i = [b_{i0}\ b_{i1}\ \ldots\ b_{i2n-1}]^T$

e $P = [p_0\ p_1\ \ldots\ p_{2n-1}]$

A partir da expressão (5) e (7), pode-se escrever

$$X_i = \overline{X} + dX_i = \overline{X} + Pb_i \quad (8)$$

Portanto, um objecto também pode ser representado como uma combinação linear dos componentes principais.

A vantagem da utilização destes métodos é que os *valores próprios* e os correspondentes *vectores próprios* são determinados por ordem decrescente de importância, o que quer dizer que os primeiros valores determinados são os responsáveis pela maior parte da variação total dos objectos. Assim, assumindo uma percentagem relativamente elevada da variância total, pode-se reduzir muito a dimensão dos dados.

Considerando que os primeiros t componentes principais explicam essa percentagem da variação total dos dados originais, obtém-se uma nova representação para um objecto,

$$X = \overline{X} + Pb \quad (9)$$

Onde $b = [b_0\ b_1\ \ldots\ b_{t-1}]^T$ e $P = [p_0\ p_1\ \ldots\ p_{t-1}]$.

Note-se no entanto que cada componente do vector $\boldsymbol{b}$ tem de ser restringido a um intervalo, cujos limites adequados são:

$$-3\sqrt{\lambda_i} \le b_i \le 3\sqrt{\lambda_i}\ , \quad \text{para } 0 \le i \le t-1 \quad (10)$$

Com esta aproximação obtém-se uma representação fiel da variação da forma do objecto, com um volume de dados consideravelmente menor. O volume de dados reduz de $2n \times 2n$ para $2n \times t$, e $t << 2n$.

IV. MODELAÇÃO DOS NÍVEIS DE CINZENTO EM TORNO DO OBJECTO

Para além do armazenamento do valor das coordenadas dos pontos (informação acerca da forma de cada objecto), é necessário também o armazenamento da informação da variação dos níveis de cinzento em torno de cada ponto. Esta informação tornar-se-á útil no momento da aplicação do algoritmo na procura de um objecto numa imagem, em particular no cálculo do movimento desejável para cada ponto. Esse movimento tem o objectivo de aproximar o modelo ao objecto que se pretende detectar.

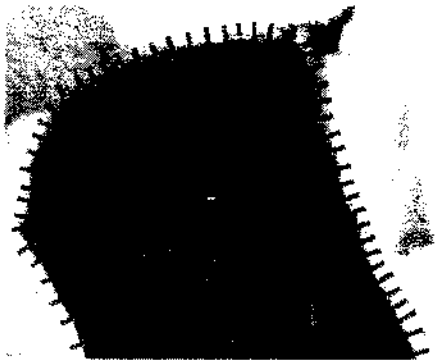


Fig. 3 - Segmentos de recta normais ao contorno.

Em termos técnicos, será guardada a informação da variação dos níveis de cinzento ao longo de um segmento de recta normal ao contorno do objecto, em todos os seus pontos. O tamanho do segmento de recta normal é escolhido pelo utilizador do sistema no momento em que guardar o primeiro objecto do modelo. Na figura 3 está representado um exemplo das normais a cada ponto para o caso de uma célula.

Assumindo que o comprimento do segmento de recta normal é n_p pixels. Para o *landmark* j da imagem i, obtêm-se os seguintes níveis de cinzento ao longo da recta (perfil),

$$g_{ij} = \left[g_{ij0}\ \ g_{ij1}\ \ \cdots\ \ g_{ijn_p-1}\right] \quad (11)$$

Aproximando a derivando da expressão anterior tem-se,

$$dg_{ij} = \left[g_{ij1} - g_{ij0}\quad g_{ij2} - g_{ij1}\ \cdots\ g_{ijn_p-1} - g_{ijn_p-2}\right] \quad (12)$$

Normalizando a derivada do perfil,

$$y_{ij} = \frac{dg_{ij}}{\sum_{k=0}^{n_p-2} |dg_{ijk}|} \quad (13)$$

Saliente-se que, para imagens ruidosas, pode-se efectuar uma média ortogonal aos níveis de cinzento, de forma a reduzir os efeitos do ruído [11]. (Ver figura 4).

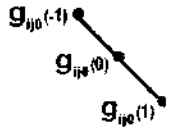


Fig. 4 - Esquema da filtragem ortogonal.

Neste caso, os níveis de cinzento ao logo do perfil são obtidos pela seguinte expressão:

$$g_{ij} = \left[g_{ij0} \;\; g_{ij1} \;\; \cdots \;\; g_{ijn_p-1} \right] \quad (14)$$

Onde g_{ij0}, por exemplo, vai tomar o seguinte valor:

$$g_{ij0} = 0,25g_{ij0}(-1) + 0,5g_{ij0}(0) + 0,25g_{ij0}(1) \quad (15)$$

A partir dos valores da derivada normalizada (13) será efectuada a modelação da variação dos níveis de cinzento, de forma a caracterizar as variações desses níveis em torno dos contornos do objecto.

A média da derivada do perfil para cada *landmark*, em todo o conjunto de treino é dada por,

$$\overline{y_j} = \frac{1}{N}\sum_{i=1}^{N} y_{ij} \quad (16)$$

Agora determina-se a matriz de covariância da derivada do perfil normalizada,

$$C_{yj} = \frac{1}{N}\sum_{i=1}^{N} (y_{ij} - \overline{y_j})(y_{ij} - \overline{y_j})^{\mathrm{T}} \quad (17)$$

Com estes dados detém-se a informação necessária e suficiente sobre a variação do aspecto do objecto em torno dos seus contornos.

V. LOCALIZAÇÃO DE UM OBJECTO NUMA IMAGEM

Já foi referido que se pode representar a forma de qualquer objecto como a soma entre a forma média dos objectos do conjunto de treino e uma combinação linear dos primeiros t componentes principais.

Assim, e usando o modelo de distribuição de pontos, é possível identificar o objecto modelado numa determinada imagem. Para uma melhor compreensão, dividiu-se o processo de localização em quatro fases a descrever:

Estimativa Inicial

O algoritmo de localização de um objecto numa determinada imagem inicia-se com a sobreposição da forma média sobre a área onde se encontra esse objecto. Depois é necessário fazer uma estimativa das principais características do objecto (Factor de escala e ângulo de rotação iniciais), que facilitará o arranque do algoritmo.

Pode-se então escrever a expressão da estimativa inicial,

$$X_{init} = M(S_i, \theta_i)\left[\overline{X}\right] + t_i \quad (18)$$

Onde S_i é o factor de escala inicial, θ_i é o ângulo de rotação inicial e t_i são as coordenadas do ponto onde se descarregou a forma média ($\overline{X}$). Note-se que, t_i pode ser expresso por um vector de comprimento $2n$, sendo n o número de *landmarks*, $t_i = \left[t_{x1} \;\; t_{y1} \;\; t_{x2} \;\; t_{y2} \;\; \ldots \;\; t_{xn} \;\; t_{yi} \right]^T$.

Determinação dos movimentos de cada "landmark"

A determinação dos movimentos desejados para cada *landmark*, pode ser feita de várias formas. Uma das formas consiste em percorrer o segmento de recta normal ao contorno do objecto em cada ponto e determinar a variação máxima do nível de cinzento. Este método poderá ser eficaz para imagens sem ruído e com boa resolução, mas para imagens com ruído e com baixa resolução este método poderá falhar. Um método alternativo, consiste em usar a modelação da variação dos níveis de cinzento em torno de cada *landmark*, para determinar esses movimentos desejados [9,10].

Usando a modelação da variação dos níveis de cinzento em torno da cada *landmark*, é possível determinar a deslocação necessária a cada *landmark* (dX_i), de modo a ajustar o modelo ao contorno do objecto.

Assim, será retirada a informação da variação dos níveis

Fig. 5 - Segmentos de recta normais ao contorno.

de cinzento ao longo da normal ao contorno do objecto, em todos os pontos. No entanto, agora o tamanho do segmento de recta normal tem de ser maior que o tamanho usado para a modelação. Na figura 5 está representado um exemplo dos segmentos de recta usados para a procura de um objecto.

Assumindo que a recta normal terá n_s pixels, com $n_s > n_p$. Para o *landmark* ***i***, obtêm-se os seguintes níveis de cinzento do perfil,

$$s_i = \begin{bmatrix} s_{i0} & s_{i1} & \cdots & s_{in_s-1} \end{bmatrix} \quad (19)$$

Aproximando a derivada da expressão anterior como,

$$ds_i = \begin{bmatrix} s_{i1}-s_{i0} & s_{i2}-s_{i1} & \cdots & s_{in_s-1}-s_{in_s-2} \end{bmatrix} \quad (20)$$

Normalizando a derivada do perfil,

$$y_{si} = \frac{ds_i}{\sum_{k=0}^{n_s-2} |ds_{ik}|} \quad (21)$$

Dividindo a derivada do perfil normalizada (21), y_{si}, em vários subintervalos com o mesmo tamanho do vector $\overline{y_i}$ determinado na expressão (16), e assumindo que a cada um desses subintervalos se representa por **h(*d*)**. Pode-se determinar o valor de ***d*** correspondente ao subintervalo **h(*d*)** mais similar ao vector $\overline{y_i}$.

Esse valor é determinado através da expressão do erro quadrático, que será mínimo quando o modelo se ajustar ao contorno do objecto,

$$f(d) = (h(d) - \overline{y_i})^T \, C_{yi}^{-1} \, (h(d) - \overline{y_i}) \quad (22)$$

Onde C_{yi}^{-1} representa o inverso da matriz de covariância de y_i, determinada na expressão (17).

Assim, determinou-se o ponto para o qual o *landmark* ***i*** deve ser deslocado. Seguindo o mesmo procedimento para os restantes, obtém-se o vector dos movimentos desejados

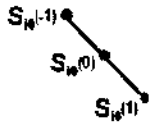


Fig. 6 - Esquema da filtragem ortogonal.

para cada *landmark* (***dXi***).

Note-se que também se pode efectuar uma média ortogonal aos níveis de cinzento de forma a reduzir os efeitos do ruído. Neste caso cada elemento da expressão (19) será dado pela seguinte expressão:

$$S_{i0} = 0,25S_{i0}(-1) + 0,5S_{i0}(0) + 0,25S_{i0}(1) \quad (23)$$

Determinação dos novos parâmetros do objecto

A partir do vector dos movimentos para cada *landmark* (***dXi***), pode-se ajustar a posição (rodar, fazer a translação e mudar a escala) do objecto do modelo, de forma a ajustar a forma estimada X_{init} tanto quanto possível de $X_{init} + dX_i$. De forma simbólica, pode-se escrever

$$X_{init} = M(S_i, \theta_i)\left[\overline{X}\right] + t_i \xrightarrow{(1+ds),\, d\theta,\, dt} X_{init} + dX_i \quad (24)$$

Ou,

$$M(S_i(1+ds), \theta_i + d\theta)\left[\overline{X}\right] + t_i + dt \longrightarrow X_{init} + dX_i \quad (25)$$

Sabendo os valores dos parâmetros **1** + ***ds***, ***dθ*** e ***dt***, é necessário resolver a seguinte equação em ordem a ***dX***:

$$M(S_i(1+ds), \theta_i + d\theta)\left[\overline{X} + dX\right] = X_{init} + dX_i - (t_i + dt) \quad (26)$$

Se $X_{init} = M(S_i, \theta_i)\left[X_m\right] + t_i$, obtém-se,

$$M(S_i(1+ds), \theta_i + d\theta)\left[\overline{X} + dX\right] = M(S_i, \theta_i)\left[\overline{X}\right] + dX_i - dt \quad (27)$$

Como $M^{-1}(s, \theta)[\ldots] = M(s^{-1}, -\theta)[\ldots]$, da expressão (27) obtém-se,

$$\overline{X} + dX = M((S_i(1+ds))^{-1}, -(\theta_i + d\theta))\left[M(S_i, \theta_i)\left[\overline{X}\right] + dX_i - dt\right] \quad (28)$$

Assim,

$$dX = M((S_i(1+ds))^{-1}, -(\theta_i + d\theta))\left[M(S_i, \theta_i)\left[\overline{X}\right] + dX_i - dt\right] - \overline{X} \quad (29)$$

Se se escrever $dX = Pdb$, e sabendo que os vectores de P são ortogonais entre si ($P^T = P^{-1}$), obtém-se,

$$db = P^T dX \quad (30)$$

Actualização do objecto

Com a informação determinada no ponto anterior, pode-se actualizar os parâmetros da forma do objecto e respectiva posição para a estimativa inicial, obtendo assim uma nova estimativa $X_{init}^{(1)}$, em que,

$$X_{init}^{(1)} = M(S_i(1+ds), \theta_i + d\theta)\left[X + Pdb^T\right] + t_i + dt \quad (31)$$

Ou dito de outra maneira,

$$\begin{aligned} t_{xi} &\rightarrow t_{xi} + dt_x \\ t_{yi} &\rightarrow t_{yi} + dt_y \\ S_i &\rightarrow S_i(1+ds) \\ \theta_i &\rightarrow \theta_i + d\theta \\ b &\rightarrow b + db \end{aligned} \quad (32)$$

Analogamente, também se pode obter outra nova estimativa $X_{mir}^{(2)}$ a partir de X_{mir}, e assim sucessivamente, até que não se registem variações significativas.

VI. RESULTADOS EXPERIMENTAIS

Para testar o desempenho dos métodos descritos, implementou-se uma interface que contempla todas as funcionalidades inerentes ao ASM. Foram construídos vários modelos de células afim de testar o desempenho destes métodos na localização de células em imagens histológicas.

De seguida são apresentados alguns resultados obtidos para dois dos modelos construídos.

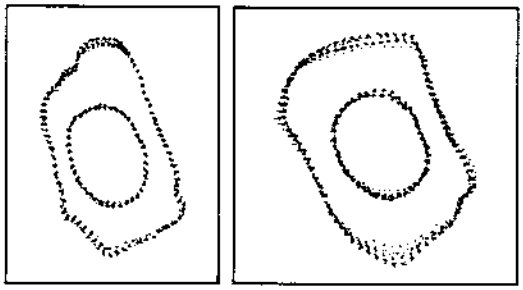


Fig. 7 - Conjunto de Treino alinhado de dois modelos.

A. Conjunto de Treino alinhado

A partir de um conjunto de 12 imagens construiu-se os conjuntos de treino que testaram o desempenho destes métodos de análise de imagem. Cada célula do conjunto de treino foi definida por 124 *landmarks* (43 para o núcleo e 81 para o citoplasma). Na figura 7 estão representados dois conjuntos de treino alinhados.

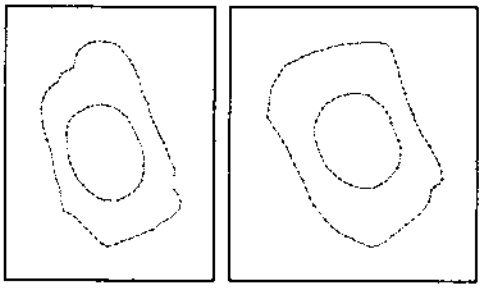


Fig. 8 - Forma média de dois modelos.

B. Forma média do conjunto de treino

Da análise estatística ao conjunto de treino resultou, para além de outros dados, a forma média dos objectos. Na figura 8 estão representadas as formas médias dos dois conjuntos de treino.

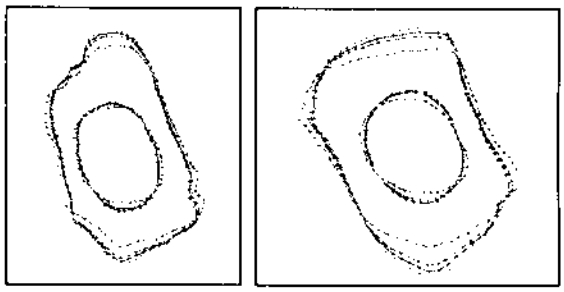


Fig. 9 - Variação do primeiro modo.

C. Variabilidade do Modelo

Para além da forma média, da análise estatística resultaram também os modos de variação do modelo. Na figura 9 está representado o efeito do primeiro modo na variação do objecto, para os dois modelos.

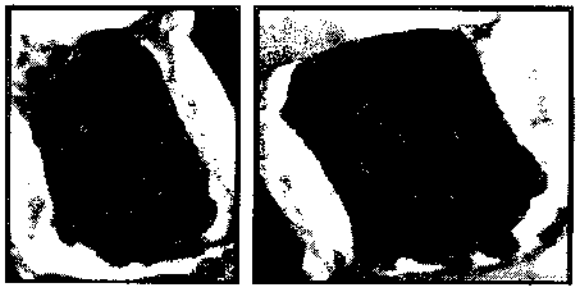


Fig. 10 - Resultado da procura de um Objecto.

D. Resultado da Procura de um Objecto

Em cumprimento de um dos objectivos do estudo, foi efectuada uma avaliação ao desempenho destes métodos quando aplicados na localização de uma determinada célula numa imagem histológica. Na figura 10 estão representados os resultados desta localização para duas imagens distintas.

O desempenho dos modelos construídos é bastante bom, alcançando a convergência de cerca de 92% dos seus pontos e num pequeno número de iterações.

O tempo despendido por cada iteração é dependente do número de pontos de cada objecto. Para os modelos construídos (124 pontos) foi cerca de 0,4 segundos por iteração.

VII. CONCLUSÕES E TRABALHO FUTURO

Com os resultados obtidos neste trabalho pode-se confirmar a possibilidade de identificar qualquer objecto numa determinada imagem, usando ASM.

Confirma-se que a aplicação estes métodos conduzem a resultados muito satisfatórios. No entanto, estes métodos apresentam algumas limitações. Os modelos criados não são universais para todas as células, uma vez que é necessário haver informação prévia acerca da forma dos objectos. Logo, para cada tipo de células tem de ser criado um modelo próprio.

Para resolver esta situação, é legítimo pensar-se que se forem introduzidas mais imagens no conjunto de treino, misturando vários tipos de células, se obtêm mais modos de variação, e portanto melhores resultados. Esta afirmação não é de todo verdade porque se não se tiver cuidado na escolha das imagens, o elevado número de modos de variação pode degradar (toldar) o modelo, conduzido o algoritmo a piores resultados.

Assim, são necessários alguns cuidados na implementação destes modelos, nomeadamente na marcação dos pontos para a construção do conjunto de treino, na escolha das imagens para o conjunto de treino e posteriormente na inicialização da forma média no momento da localização de objectos numa imagem.

De uma maneira geral, em termos de velocidade estes métodos revelam-se bastante rápidos, o que constitui uma vantagem. No entanto, não se verifica grande versatilidade nestes métodos quando aplicados na análise de células, pois o citoplasma apresenta grandes variações na sua forma. Apesar disso, estes métodos serão muito vantajosos quando aplicados a imagens cuja variação não é muito significativa, como por exemplo reconhecimento facial e detecção de estruturas em ecocardiogramas.

Como trabalho futuro, poder-se-ia combinar estes métodos com outros já conhecidos, por exemplo *"Snakes"*, e usar o melhor de cada um para construir um sistema mais versátil e viável.

Mais tarde poder-se-á avançar para plataformas de desenvolvimento de alto nível como por exemplo *Delphi*, *Visual Basic*, ou *Visual C++*, que permitem desenvolver boas interfaces com o utilizador; aproveitando no entanto as livrarias de funções construídas neste trabalho.

AGRADECIMENTOS

Agradece-se à Unidade de Investigação 127/94 IEETA da Universidade de Aveiro, o financiamento do projecto.

REFERÊNCIAS

[1] G. Parmigianni, E. S. Garret, R. Anbazhagan and E. Gabrielson, "A Statistical Framework for Expression-Based Molecular Classification in Cancer", J Royal Statistical Society, 2002.

[2] L. H. Staib and J. S. Duncan, "Parametrically Deformable Contour Models", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, pp. 427 - 430, 1989.

[3] L. H. Staib and J. S. Duncan, "Model-based deformable surface finding for medical images", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, n. º 5, 1996.

[4] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active Contour Models", First International Conference on Computer Vision, IEEE Computer Society Press, pp. 259 – 268, 1987.

[5] T. F. Cootes, C. J. Taylor, D. H. Cooper and J. Graham, "Active Shape Models – Their Training and Application", Computer Vision and Image Understanding, Vol. 61, n.º 1, pp. 38 - 59, 1995.

[6] T. F. Cootes, C. J. Taylor, D. H. Cooper and J. Graham, "Training models of shape from sets of examples", Proceedings of the British Machine Vision Conference, Springer-Verlag, pp. 9 - 18, 1992.

[7] T. F. Cootes, C. J. Taylor and A. Lanitis, "Active Shape Models: Evaluation of a Multi-Resolution Method for Improving Image Search", Proceedings of the British Machine Vision Conference, pp.327 - 336, 1994.

[8] T. F. Cootes, A. Hill, C. J. Taylor and J. Haslam, "The Use of Active Shape Models for Locating Structures in Medical Images", Image and Vision Computing, Vol. 12, n.º 6, pp. 355 - 366, 1994.

[9] G. Hamarneh, R. Abu-Gharbieh, and T. Gustavsson, "Review - Active Shape Models - Part I: Modelling Shape and Gray Level Variation", Swedish Symposium on Image Analysis, pp. 125 - 128, 1998.

[10] R. Abu-Gharbieh, G. Hamarneh and T. Gustavsson, "Review - Active Shape Models - Part II: Image Search and Classification", Swedish Symposium on Image Analysis, pp. 129 - 132, 1998.

[11] Active Shape Models (ASM), http://www.isbe.man.ac.uk/courses/Computer_Vision/downloads/L15_ASMs.pdf, (Jun 2004)

[12] A. Hill, T. F. Cootes and C. J. Taylor, "Active Shape Models and the shape approximation problem", Image and Vision Computing, pp. 601 - 608, 1996.

[13] M. Rogers and J. Graham, "Robust active shape model search", In Proceedings of the European Conference on Computer Vision, 2002.

Sistema de Monitorização para Elevadores

Ana Margarida B. M. Sargento, Bernardo Cunha e Osvaldo da Rocha Pacheco

Resumo – **Neste artigo é descrito o desenvolvimento de um Sistema de Monitorização para Elevadores, efectuado no âmbito da disciplina de Projecto do 5º ano do curso de Engenharia Electrónica e Telecomunicações da Universidade de Aveiro. Este projecto enquadra-se num conjunto de três projectos baseados em propostas de desenvolvimento provenientes de uma empresa exterior à Universidade, de nome LifTech, a qual desenvolve, fabrica e comercializa Sistemas de Comando para Elevadores.**

O sistema desenvolvido é adaptável a qualquer tipo de elevador, tem capacidade para monitorizar um conjunto de sinais e para disponibilizar comandos actuáveis externamente. O sistema permite ainda fazer o registo de assistências técnicas. A monitorização propriamente dita é efectuada a partir de um Centro de Supervisão Remoto que está em contacto com vários sistemas de elevadores.

Abstract – **In this paper we present the final result of the development of a general purpose Lift Monitoring System, which has been held in the context of a Project discipline in the last year of the graduated course in Engenharia Electrónica e Telecomunicações da Universidade de Aveiro. This project is part of a three project proposals block brought to us by a company called LifTech. This company is responsible for the production and commercialisation of Lift Command Systems.**

The developed system can be adapted to any kind of lift, is able to monitor a significant amount of signals and provides a group of command signals that can be actuated on request. Technical assistance by authorized personnel can also be monitored. Remote monitoring is provided by a Remote Supervising Centre, which can be in contact with several distinct lift systems.

I. INTRODUÇÃO

O sistema desenvolvido tem como objectivo a supervisão remota de elevadores, permitindo não só a monitorização e registo de eventos, como também a actuação remota de comandos e o registo de assistências técnicas.

Com o objectivo de desenvolver uma solução actual e competitiva foi efectuado um levantamento dos sistemas de monitorização de elevadores que existem no mercado. De entre os sistemas encontrados destacam-se os sistemas da OTIS [1], da ADVANTECH [2], da ELEVATOR WOLD [3] e da MITSUBISHI [4]. Alguns destes sistema foram desenvolvidos exclusivamente para integração nos elevadores da própria empresa, como o da OTIS, enquanto outros de destinam a ser integrados na generalidade dos elevadores já existentes. Alguns destes sistemas apresentam algumas funcionalidades de destaque como o facto de permitirem monitorizar a deterioração de componentes, prever anomalias, detectar ocorrências com duração parametrizável, gerar relatórios de ocorrências, permitirem a configuração remota do sistema, permitirem a comunicação por voz com passageiros presos dentro do elevador, possuirem uma bateria que assegura o funcionamento do sistema (por um curto período de tempo) em caso de falha de energia ou ainda o facto de assentarem numa arquitectura modular, facilitando a manutenção e a substituição de módulos danificados e permitindo uma fácil expansão do sistema.

O sistema desenvolvido visa reunir as funcionalidades encontradas nos demais sistemas existentes.

A arquitectura da solução adoptada baseia-se numa rede de Unidades Locais de Monitorização (ULMs) inteligentes, responsáveis pela monitorização do elevador ou conjunto de elevadores pertencentes a um mesmo edifício, e um Centro de Supervisão Remoto (CSR) constituído por um PC remoto. O *software* do PC localizado no CSR permite receber alarmes provenientes de múltiplas unidades e manter toda a informação relacionada com as mesmas. A comunicação entre ambos é efectuada através de rede fixa comutada ou rede GSM.

Ao longo do artigo, são descritas as várias fases de desenvolvimento do projecto. Inicialmente é apresentada uma secção inteiramente dedicada à descrição da Unidade Local de Monitorização (secção II), que engloba a solução adoptada em termos de arquitectura e funcionalidade, o protocolo de comunicação entre os módulos da ULM (Unidade Local de Monitorização) e a estrutura do *software* desenvolvido. De seguida é apresentada uma secção relativa ao Centro de Supervisão Remoto (CSR) (secção III), onde se descreve o protocolo de comunicação entre PC e ULM e a estrutura da base de dados que deve constar no PC do CSR. Na secção IV é então apresentada a aplicação de demonstração desenvolvida. Finalmente, na secção V são apresentadas as conclusões do trabalho.

II. UNIDADE LOCAL DE MONITORIZAÇÃO (ULM)

A. Arquitectura da Solução Encontrada

Cada ULM é responsável pela monitorização e registo local de eventos e tem capacidade de reportar alarmes para o Centro de Supervisão Remoto, actuar em saídas a pedido

do mesmo e fazer o registo de assistências técnicas. O registo de eventos e de assistências técnicas constitui um histórico que, a pedido do Centro de Supervisão Remoto, é enviado para o mesmo. Cada evento é acompanhado da data e hora da ocorrência, com resolução temporal de 250ms.

A sua implementação é baseada numa arquitectura distribuída que adopta um modelo funcional do tipo *mater/slave*, podendo o sistema local (ULM) integrar:

- 1 módulo principal;
- 1 módulo de comunicações;
- Até 30 módulos I/O (entradas/saídas digitais) (figura 1),

interligados por uma *Rack* que suporta até 8 módulos.

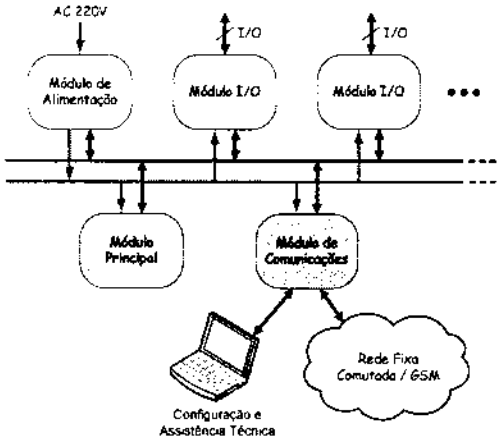


Figura 1 – Diagrama de blocos da Unidade Local de Monitorização

Os módulos comunicam através de um barramento CAN e possuem inteligência local baseada em microcontroladores da *Microchip* [6]. No Módulo de Comunicações é utilizado o PIC18F458 enquanto os restantes módulos se baseiam no PIC18F258.

A *Rack* fornece aos módulos uma identificação *hard-wired*, dependente da posição onde cada um se encontra, as alimentações desacopladas de 5V e 24V e as linhas de comunicação CAN [5].

O Módulo Principal é responsável por coordenar toda a ULM, assegurando a gestão de eventos, o armazenamento da configuração da ULM e do histórico de eventos e ainda o registo de assistências técnicas (Figura 2).

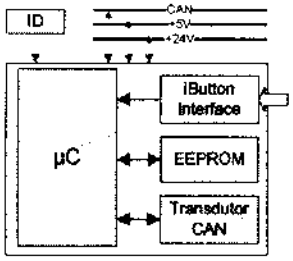


Figura 2 – Esquema do Módulo Principal

Os eventos a detectar podem ser:

- O início/fim de um período de manutenção local por parte de um técnico especializado;
- A ocorrência de erros na ULM;
- A ocupação da memória de histórico acima de um *threshold* predefinido;
- A detecção de ocorrências associadas a cada porto de entrada.

Cada evento pode conduzir ao registo de ocorrência do mesmo no histórico, ao envio de um alarme para o CSR, à actuação numa saída de um módulo I/O ou a qualquer combinação destas acções.

O armazenamento de configurações e histórico de eventos é efectuado numa memória externa não volátil, de tecnologia EEPROM.

O sistema de identificação do técnico especializado, usado para efectuar a detecção de início e fim das assistências técnicas em cada ULM é do tipo *iButton* [7] – dispositivo de armazenamento de dados, robusto, que actua como um registo de identificação electrónica, possuindo um número único associado.

O Módulo de Comunicações permite a comunicação com o exterior, ou seja, a comunicação entre a Unidade Local de Monitorização e o Centro de Supervisão e entre a Unidade Local de Monitorização e o PC de Assistência. A referida comunicação é efectuada a partir de uma interface USB para ligação a um PC de assistência e através de uma interface RS232 para ligação a um *modem* (Figura 3).

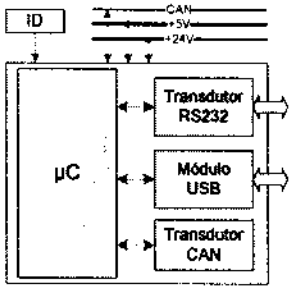


Figura 3 – Esquema do Módulo de Comunicações

O Módulo de Comunicações tem como função tornar o sistema independente do tipo de modem (GSM ou PSTN).

Finalmente, cada Módulo I/O é responsável pela interface física dos sinais a monitorizar, através de 8 entradas, isoladas opticamente para interface com contactos livres de tensão, e 2 saídas por relés (Figura 4).

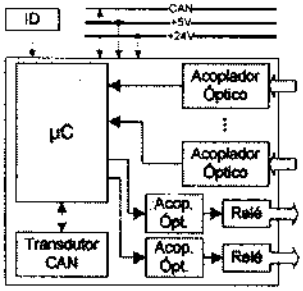


Figura 4 – Esquema do Módulo I/O

Cada um destes módulos é ainda responsável pela detecção de ocorrências com duração parametrizável de 250ms a 1 minuto e resolução temporal de 250ms.

B. Protocolo de Comunicação entre os Módulos

O Módulo Principal actua como *master* da comunicação com os restantes módulos da ULM, havendo apenas troca de mensagens quando este toma a iniciativa de iniciar a mesma. Todas as mensagens trocadas implicam um *handshaking*, ou seja, o Módulo Principal (*master*), após ter enviado uma mensagem, espera por uma resposta ou uma confirmação (*acknowledge*) da recepção da mesma por parte do outro módulo interveniente na comunicação.

Cada um dos módulos I/O pode receber, em qualquer altura do seu funcionamento, qualquer mensagem com um dos comandos que lhe pode ser dirigido. Relativamente ao Módulo de Comunicações acontece o mesmo, com a excepção das mensagens de configuração, as quais devem ser precedidas de uma mensagem de pedido para iniciar a configuração.

C. Software do Módulo Principal

C.1. Algoritmo

A execução das várias tarefas a cargo deste módulo obedece à máquina de estados apresentada na figura 5.

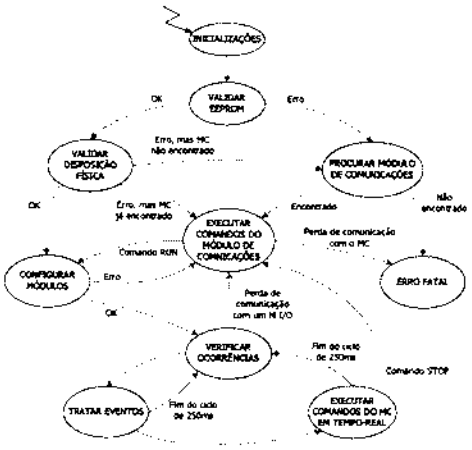


Figura 5 – Diagrama de estados do *software* do Módulo Principal

Na fase de arranque do Módulo Principal são efectuadas as inicializações dos vários periféricos, tanto internos como externos ao microcontrolador (inicialização dos portos, módulo da USART, módulo CAN, leitor de iButtons, EEPROM, tick do sistema e variáveis globais).

Seguidamente é efectuada uma validação do conteúdo da EEPROM. No caso de se verificar alguma incoerência da informação contida na EEPROM, a ULM terá de aguardar a intervenção de um técnico, pelo que tem necessidade de determinar a posição do Módulo de Comunicações para poder comunicar com o exterior.

Caso a informação contida na EEPROM seja válida, é afectuado um reconhecimento da disposição física dos restantes módulos. Se se verificar haver alteração, relativamente à disposição física anterior, é necessária a intervenção de um técnico, pelo que o Módulo Principal fica a aguardar comandos do Módulo de Comunicações, passando para o estado correspondente.

Considerando a disposição física validada, é efectuada a (re)configuração dos módulos. Caso ocorra algum erro durante este procedimento o módulo passa para o estado *EXECUTAR COMANDOS DO MÓDULO DE COMUNICAÇÕES*, aguardando intervenção de um técnico.

Após o reconhecimento da disposição física é efectuada a detecção de ocorrências relacionadas com o estado dos portos de entrada, detecção de iButton, percentagem de ocupação do histórico e ocorrência de erros (que não impeçam o funcionamento da ULM mas que sejam susceptíveis de gerar eventos). Os eventos disparados são então colocados numa fila de espera para posterior tratamento.

No estado *TRATAR EVENTOS* são processados os eventos que estão na lista de espera, ou seja, é levada a cabo a acção associada ao evento (actuar numa saída, registar evento, enviar alarme), tendo o cuidado de não ultrapassar o tempo de ciclo de amostragem do sistema. Durante a execução deste estado, caso o tempo de ciclo chegue ao fim, retorna-se ao estado anterior de modo a verificar novas ocorrências. Caso o tempo de ciclo não se esgote, passa-se para o estado *EXECUTAR COMANDOS DO MÓDULO DE COMUNICAÇÕES EM TEMPO-REAL*.

Na execução deste último estado são atendidos comandos, que não implicam a paragem da amostragem do sistema, vindos do Módulo de Comunicações.

O estado *EXECUTAR COMANDOS DO MÓDULO DE COMUNICAÇÕES* é em tudo semelhante ao estado descrito anteriormente, com a diferença de ser um estado bloqueante.

O estado *PROCURAR MÓDULO DE COMUNICAÇÕES* tem o objectivo de informar o Módulo Principal do ID do Módulo de Comuinicações (e *vice-versa*) para que a comunicação entre a ULM e o exterior se possa processar, apesar da impossibilidade de funcionamento normal da mesma.

O estado *ERRO FATAL* é atingido quando existe uma falha de comunicação entre o Módulo Principal e o Módulo de Comunicações. Este estado é sinalizado com o LED multicor a piscar vermelho à frequência de 2Hz.

C.2. Detalhes de Implementação

DEVICE DRIVER PARA INTERFACE COM A EEPROM

Foi criada uma cache de escrita para a EEPROM: a informação a gravar na EEPROM é guardada numa primeira fase em *buffers*, e uma rotina de interrupção activada por um *timer* com período de 5ms é responsável pelo desencadear do processo de escrita efectiva, na EEPROM, da informação presente nos *buffers*.

DEVICE DRIVER PARA INTERFACE COM O MÓDULO CAN

Foi desenvolvida uma camada de *software* capaz de gerir tanto a recepção como a transmissão de tramas CAN

usando interrupções. Deste modo o código de transmissão e recepção de mensagens não é bloqueante [8].

DEVICE DRIVER PARA INTERFACE COM O IBUTTON

A interface com o *iButton* foi feita implementado em *software* o protocolo de comunicação descrito no *datasheet* do próprio componente [9].

D. Software do Módulo de Comunicações

D.1. Algoritmo

O Módulo de Comunicações actua como *gateway* entre o Módulo Principal e o CSR ou um PC local. O funcionamento deste módulo tem que obedecer a duas caracteristicas difíceis de conciliar: a comunicação a ritmo elevado com o Módulo Principal, simultaneamente com a comunicação, a ritmo mais lento, com o *modem*. A solução adoptada passa por tratar os comandos de mais alta prioridade (mensagens CAN), vindos do Módulo Principal, numa rotina de interrupção.

O algoritmo de controlo associado ao Módulo de Comunicações engloba a execução de 3 processos: processamento de comandos vindos do CSR ou PC local quando está estabelecida uma ligação com um desses sistemas; envio de alarmes ao CSR e execução de comandos do CSR recebidos por SMS.

D.2. Detalhes de Implementação

COMUNICAÇÃO VIA USB

A comunicação via USB com o exterior é feita através de um módulo externo ao microcontrolador. Este módulo encapsula todo o protocolo USB, dispensando a implementação de qualquer tipo de *protocol stack* associado ao USB. O *software* de interface com este dispositivo baseia-se no protocolo de comunicação descrito no respectivo *datasheet* [10].

E. Software de cada Módulo I/O

E.1. Algoritmo

Os módulos I/O são responsáveis pela validação das ocorrências associadas aos respectivos portos de entrada e pela actuação nas saídas (sob a forma de comando). Os processos relacionados com a amostragem e processamento dos sinais de entrada são executados com um período de 10ms. O tempo que resta é usado para executar comandos do Módulo Principal, caso tenham sido recebidos.

Os sinais de entrada podem estar ligados a relés ou a dispositivos de comutação com actuação mecânica que têm associado ruído de bouncing derivado aos seus contactos. Para eliminar o ruído decorrente deste facto as entradas são amostradas a uma frequência de 100Hz, sendo a validação de cada estado feita apenas quando 25 amostras consecutivas assumem o mesmo valor. Deste modo só existe mudança de estado quando o sinal amostrado se mantém estável durante 250ms.

III. CENTRO DE SUPERVISÃO REMOTO (CSR)

O *software* do PC do Centro de Supervisão deve, em primeiro lugar, permitir efectuar a monitorização de um conjunto de ULMs. Adicionalmente deve permitir actuar na ULM, o que compreende, entre outras acções, o pedido de informações relativas ao estado de funcionamento da ULM, actuação em saídas, (re)configuração e *download* do histórico.

Do ponto de vista funcional, uma (re)configuração envolve não só a criação de uma identificação de cada ULM registada em Base de Dados, como a especificação de todos os aspectos funcionais e de interligação eléctrica dessa mesma ULM.

A. Protocolo de Comunicação entre PC e ULM

Toda a comunicação entre o PC e cada ULM implica *handshaking*. O envio de cada mensagem requer, portanto, a posterior recepção de uma mensagem de *Acknowledge* e/ou uma mensagem de resposta se se tratar de um pedido de informação. O *handshaking* praticado envolve, no entanto, a recepção de uma dupla confirmação. Cada pedido do CSR é seguido de uma mensagem de *Acknowledge* e de uma resposta ao pedido. Isto acontece porque a maior parte da informação susceptível de ser pedida tem que ser fornecida pelo Módulo Principal, o que requer comunicação entre os dois módulos e portanto implica algum *overhead*. Assim, o *Acknowledge* confirma a recepção da mensagem por parte do Módulo de Comunicações e, após a obtenção da informação, o Módulo de Comunicações envia a resposta correspondente para o CSR (Figura 6).

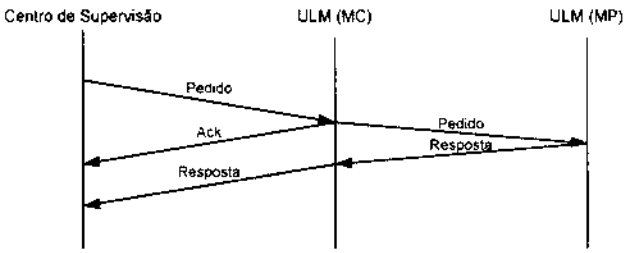


Figura 6 – Sequência de mensagens trocadas quando é efectuado um pedido de informação

As mensagens oriundas do CSR, que não constituem um pedido de informação, não recebem uma resposta mas apenas uma confirmação de comando transmitido com sucesso.

Cada mensagem trocada entre a ULM e o PC (quer seja do Centro de Supervisão quer seja de Assistência) é constituída por um caracter de início de trama, um comando (um byte), dados de tamanho variável (eventualmente nulo), o *checksum* da mensagem e um caracter de fim de trama. O *checksum* é aplicado ao conjunto (comando + dados) e é utilizado para detecção de erros na transmissão. Para garantir que os caracteres de início e fim de trama só surgem nessas situações (como

delimitadores da trama), toda a trama, excluindo esses caracteres, é codificada em ASCII.

A.1. Comunicação por GSM ou Rede Fixa Comutada

No modelo de comunicação utilizado a transmissão de qualquer tipo de informação implica *handshaking*, pelo que, conforme a situação em concreto, uma das entidades, CSR ou ULM, se mantém, durante algum tempo, à espera de receber uma resposta por parte da outra entidade. Por outro lado, a transmissão de determinados fluxos de informação, como o *download* do histórico ou o envio de configuração, implica a transmissão de uma sequência longa de tramas, implicando *handshaking* para cada trama. Utilizando o envio de mensagens SMS sobre GSM, o tempo que medeia o envio de cada mensagem e a recepção da respectiva resposta é da ordem de alguns segundos levando a que a transmissão de sequências longas de pacotes seja muito lenta. O tratamento de mensagens, tanto ao nível da ULM como ao nível do CSR, está sujeito a *timeouts* para permitir que o processamento do Módulo de Comunicações possua garantias temporais, não bloqueando indefinidamente à espera de uma mensagem. O dimensionamento dos *timeouts* para valores razoáveis em termos de SMSs inviabilizaria todo o restante processamento do Módulo de Comunicações e, consequentemente, do Módulo Principal.

A utilização de SMS será assim apenas viável para uma ULM cuja configuração e *download* do histórico sejam efectuados localmente e que utilize a ligação GSM apenas para o envio de alarmes e recepção de comandos de actuação nas saídas. Estas comunicações envolvem apenas o envio da mensagem pretendida, deixando a responsabilidade da entrega da mesma ao operador da rede GSM. O atraso inerente à comunicação, bem como a possibilidade de haver erros na transmissão devem ser assumidos como limitações do sistema. Uma alternativa a esta tecnologia será a utilização GPRS, alternativa esta que não chegou a ser estudada no âmbito do projecto.

A utilização de rede fixa comutada permite o estabelecimento de uma chamada de dados que fica disponível aos 2 intervenientes até ser terminada por um deles. Assim é possível efectuar a transmissão de fluxos de informação com diferentes características, incluindo sequências longas de tramas.

B. Estrutura da Base de Dados

A base de dados armazenada no CSR deve ter capacidade para armazenar a informação recolhida da monitorização e, simultaneamente, a informação necessária para controlar um conjunto de ULMs. Assim, cada entrada da base de dados representa uma ULM e contém os seguintes campos:

- O número de telefone do elevador (modem) ⇒ Campo obrigatório;
- O número de telefone do Centro de Supervisão Remoto ⇒ Campo obrigatório;
- A morada do elevador de onde provém o alarme;
- O número do contrato;
- O número de telefone da empresa de manutenção;
- O número de telefone do porteiro do prédio;
- O tipo de modem da ULM ⇒ Campo obrigatório;
- O número de tentativas para estabelecer chamada entre o CSR e a ULM ⇒ Campo obrigatório;
- A percentagem de ocupação do histórico à qual deve ser enviado um alerta para a central;
- Uma cópia, datada, da configuração actual do sistema ⇒ Campo obrigatório;
- Uma cópia, datada, de cada uma das configurações que o sistema já teve ⇒ Campo obrigatório;
- Uma cópia do estado actual das saídas.

IV. APLICAÇÃO DE DEMONSTRAÇÃO

Todo o projecto foi centrado, desde o início, na concepção da ULM, procurando desenvolver um sistema o mais completo possível, ao qual pudesse ser facilmente associado, mais tarde, o *software* de monitorização e configuração. Embora não tendo havido possibilidade de desenvolver o *software* do CSR, surgiu a necessidade de ter um *software* de demonstração que permitisse por um lado fazer um *debug* mais profundo ao *software* da ULM, e por outro permitisse demonstrar as funcionalidades da ULM concebida. O *software* desenvolvido não explora profundamente as potencialidades da ULM, permitindo apenas ao utilizador um conjunto de actuações simples que emulam as transferências de informação que fazem parte do protocolo de comunicação implementado.

Este *software* não utiliza a base de dados descrita, mas sim uma estrutura baseada nessa base de dados e que contém toda a informação de uma ULM. A estrutura referente a cada ULM é guardada num ficheiro de modo a poder ser recuperada cada vez que se utilizar a aplicação.

O *software* de demonstração permite ao utilizador configurar uma ULM e provocar as ocorrências previamente configuradas de modo a fazer despoletar os eventos (Figura 7). Finalmente, permite efectuar o *download* do histórico e analisar o seu conteúdo de forma a determinar se tudo se processou como era esperado.

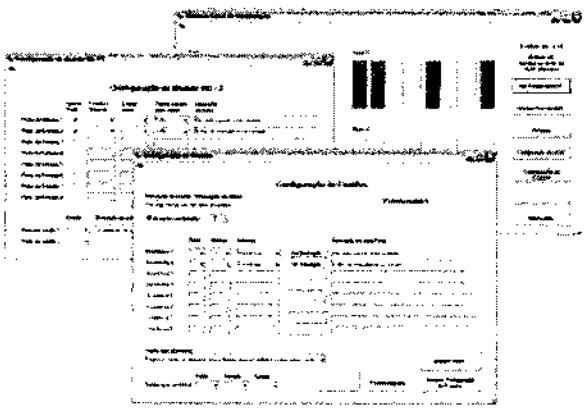


Figura 7 – Apresentação da Aplicação de Demonstração

V. CONCLUSÕES

Neste artigo são descritas as várias fases de desenvolvimento de um sistema de monitorização remota para elevadores bem como a sua arquitectura final. Este sistema permite a monitorização e registo de eventos, a actuação remota de comandos e o registo de assistências técnicas.

O sistema projectado baseia-se na existência de unidades de monitorização locais (ULMs), que fazem a monitorização directa dos sinais e a gestão dos eventos, e na existência de um Centro de Supervisão Remoto, o qual possui *software* que permite receber alarmes provenientes das ULMs e manter, numa base de dados, toda a informação relacionada com as mesmas. A comunicação entre ambos pode ser efectuada através de rede fixa comutada ou rede GSM. Foi no entanto possível concluir que a escolha da rede GSM acarreta algumas limitações relativamente às características da informação que pode ser transmitida, uma vez que não permite a transmissão de sequências longas de tramas.

O *software* desenvolvido para a ULM possui um elevada redundância ao nível das comunicações entre os módulos o que confere ao sistema uma elevada tolerância a falhas e esse nível. As funcionalidades disponibilizadas pela ULM permitem ainda uma elevada flexibilidade no desenvolvimento do *software* do CSR, uma vez que cada ULM pode responder aos comandos provenientes do CSR sem que a sua principal função de monitorização seja afectada.

O sistema desenvolvido é totalmente parametrizável pelo que pode ser utilizado noutras aplicações que não os elevadores. Para esta possível aplicação prática contribui o facto da resolução temporal do sistema ser de 250ms, permitindo uma maior reactividade do que a que seria estritamente necessária para um sistema de elevadores.

REFERÊNCIAS

[1] http://www.otis.com/maintenancedetail/0,1422,CLI1_MID7828_RES1,00.html, OTIS – Remote Elevator Monitoring

[2] http://www.advantech.com.tw/efms/appstories/12.pdf, ADAM-5510 - Centralized Elevator Monitoring System

[3] http://www.elevator-world.com/magazine/archive01/0009-001.html-ssi, W.L. Chan, Albert T.P. So and S.K. Liu, A Cost-effective Remote Monitoring and Communication System

[4] http://global.mitsubishielectric.com/pdf/advance/vol88/vol88_tr8.pdf, Kiyoji Kawai and Hideki Shiozaki, A Remote Inspection System for Elevators

[5] http://www.can-cia.org, CAN in Automation homepage

[6] MicroChip PIC18FXX8 Datasheet

[7] http://www.ibutton.com/ibuttons/

[8] Alexandre Santos, Ana Sargento, Pedro Leite, Ricardo Marau - Developing APIs and Device Drivers for CAN Based Embedded System

[9] http://pdfserv.maxim-ic.com/en/ds/DS1990A-F3-DS1990A-F5.pdf

[10] http://www.ftdichip.com/Documents/ds245b14.pdf

Aspectos de Medição de Qualidade de Serviço em Redes Móveis baseadas em IP

Nuno Inácio, Ricardo Azevedo, Rui L. Aguiar, Francisco Fontes

Resumo - Os aspectos de Qualidade de Serviço (Quality of Service – QoS) em redes avançadas aparecem como uma área de trabalho interessante, dada a complexidade de que se reveste o planeamento e controlo de redes IP com QoS, associadas a ambientes móveis.

Esta comunicação aborda problemas de medição em redes heterogéneas, em particular medidas relevantes para QoS, essenciais tanto para avaliação de desempenho, como para mecanismos de controlo. São discutidas duas ferramentas distintas: uma ferramenta de medição do estado de ligações TCP e um elemento de controlo de um sistema de medição baseado em pontas de prova (probes) IP. As ferramentas foram desenvolvidas e testadas com recurso à rede Moby Dick, obtendo-se bons resultados.

Abstract – Quality of Service (QoS) aspects are one of the most interesting areas of work in new advanced networks, due to the complexity of planning and controlling IP networks with QoS, especially when associated with mobile environments.

This paper discusses measuring problems in heterogeneous networks, particularly relevant measures for QoS. It covers two distinct tools: one for measurements of TCP connections; and the other a control element for a probe based system. The tools were developed and tested with the Moby Dick network with good results.

I. INTRODUÇÃO

O desenvolvimento simultâneo de sistemas de comunicações de 3ª geração e de novos meios de acesso à Internet tem levado ao desenvolvimento de conceitos e sistemas que promovam a junção destas duas ideias chaves no mundo das telecomunicações actuais. Neste sentido, os aspectos de qualidade de Serviço (*Quality of Service – QoS*) na Internet aparecem como uma das áreas de trabalho mais interessantes, dada a complexidade de que se reveste o planeamento e controlo de redes IP com QoS, associadas a ambientes móveis – a própria mobilidade dos utilizadores pode criar facilmente pontos de congestionamento na rede. Além disso, a multiplicidade de serviços possíveis de disponibilizar nestes ambientes torna o controlo da rede uma tarefa complexa.

A implementação de QoS na Internet é uma área muito activa nos últimos anos e recentemente tem sido centrada em ambientes de mobilidade. Exemplo disso é o projecto Moby Dick [MOBYWEB, MOBY], que aborda, propondo e demonstrando soluções, os temas de qualidade de serviço e autorização e autenticação de acesso. Esta rede é baseada no modelo de referência de serviços diferenciados, associado a mecanismos eficientes de mobilidade e a mecanismos de AAA (autorização, autenticação e contabilização), baseados no protocolo DIAMETER.

Este documento apresenta os resultados de um trabalho final de curso que aborda esta área. O documento começa por uma introdução a aspectos de medidas para o protocolo TCP na secção II e descreve uma ferramenta dedicada a análise de comportamentos deste protocolo durante *handovers* em redes móveis. A secção III aborda os conceitos de medidas baseadas em *probes* e por fim, a secção IV discute a implementação de um elemento de controlo para um sistema de medição baseado em *probes*. A secção V apresenta as principais conclusões retiradas do trabalho.

II. MEDIDAS DE DESEMPENHO DO TCP EM AMBIENTES MÓVEIS

O TCP (*Transmission Control Protocol*) é um protocolo responsável pelo transporte de dados fim-a-fim. O TCP é um protocolo orientado à conexão, e tem mecanismos que protegem contra a perda de pacotes, a sua duplicação, a sua desordenação na entrega ou ainda a existência de dados corruptos. Este protocolo implementa ainda vários algoritmos para o controlo do tráfego na rede e para o reenvio de pacotes no caso destes serem perdidos em trânsito.

A Otenção de variáveis internas do TCP

Sendo o TCP um protocolo orientado à conexão, uma implementação do protocolo tem de guardar um grande conjunto de informação. Na implementação do protocolo em Linux acontece exactamente isso, sendo guardados vários valores por cada ligação TCP existente. A obtenção desses valores, geralmente internos ao *kernel*, pode ser de grande importância para se perceber o comportamento de determinada implementação num determinado cenário. A obtenção de valores para a janela de congestionamento poderá mostrar-nos como a performance do TCP se comporta durante determinados processos de rede – entre

os quais é de particular interesse o *handover*, a situação de mudança de ponto de ligação do terminal à rede.

Existem várias ferramentas para analisar o comportamento do TCP, sendo as mais interessantes o TBIT, o WEB100 e o Sockmon.

O TBIT [1] é uma ferramenta desenvolvida pelo "AT&T Center for Internet Research" que tem como objectivo caracterizar o comportamento do TCP num servidor remoto. Mais especificamente, esta ferramenta permite obter aproximações dos valores da cwnd (*Congestion Window*, um parâmetro importante para o desempenho do TCP), qual a versão do protocolo TCP utilizada, entre outras medidas. Com o uso desta ferramenta é ainda possível a detecção de erros de implementação do TCP no sistema operativo do servidor em questão.

WEB100 [2] é um projecto que tem como objectivo desenvolver uma plataforma onde possam ser executadas aplicações *web* que utilizem 100% da largura de banda disponível, seja ela qual for. De uma forma mais geral, o projecto tem com objectivo fazer com que qualquer máquina possa ter acesso a toda a largura de banda disponível sem a ajuda de peritos para as configurações. Com a utilização deste *software* o sistema operativo (no caso o Linux) conseguirá utilizar a *stack* TCP a tempo real, sem a intervenção por parte do utilizador, de forma a que a maior largura de banda possível seja utilizada na(s) ligação(ões). Neste momento existe apenas uma versão do projecto para Linux, mas é esperado que os seus promotores desenvolvam uma versão genérica que possa ser utilizada em qualquer sistema operativo. Como se trata de um projecto com uma duração de três anos, encontrando-se a meio na altura da realização do trabalho, estão ainda muitos dos objectivos por implementar.

O WEB100 é dividido em duas componentes distintas. A primeira componente é um *patch* ao *kernel* que é responsável por colocar instrumentos de medida, recolher todas as informações necessárias e colocá-las visíveis para posterior apresentação ou alteração. A segunda componente é uma biblioteca que tem com objectivo ler e escrever para as variáveis que o *patch* coloca visíveis. Com a utilização da referida biblioteca é possível a qualquer interessado desenvolver uma aplicação que leia e apresente as variáveis que o *patch* colocou visíveis. A comunicação entre o *kernel space* e *user space* é feita através do sistema de ficheiros /proc. É através da escrita, por parte do *kernel*, e da leitura, por parte da biblioteca, no /proc que ambos os componentes de *software* comunicam. Assim é possível a um utilizador alterar um determinado valor através da escrita no ficheiro certo do /proc.

Os tipos de instrumentos disponíveis vão desde a medição de estatísticas sobre o tráfego que está a sair referente a uma determinada ligação TCP, o estado da ligação, informação e controle sobre os vários *buffers* de saída, RTT, congestionamento do emissor, etc.

Esta ferramenta mostra-se de grande interesse para a obtenção/observação e "afinamento" (*tuning*) das várias variáveis envolvidas no protocolo TCP. No entanto não foi possível utilizá-la na rede Moby Dick, a rede de teste disponível, pois esta baseia-se no *kernel* Linux 2.4.16, para o qual o *patch* WEB100 não funciona correctamente.

O Sockmon [3] é um módulo para *kernel* Linux 2.2.x e 2.4.x que permite a monitorização em tempo real de informação respeitante a todas as ligações TCP existentes, isto é, todas as ligações que se encontrem no estado ESTABLISHED. Também esta ferramenta faz uso do /proc para mostrar as informações medidas no *kernel*. Uma vantagem deste método em relação ao utilizado no projecto Web100 tem a ver com o facto do Sockmon ser um módulo que é carregado e não um *patch* ao *kernel*. Para a utilização do Sockmon não é necessário a instalação que qualquer *patch*, sendo apenas necessário escolher, durante a configuração para a compilação do *kernel*, uma opção específica. Depois de recompilado o *kernel*, basta carregar o módulo e verificar em "/proc/net/sockmon" o valor das variáveis extraídas do *kernel*. Este módulo é, no entanto, incompatível com o *kernel* do Moby Dick (especificamente com o Swamp, que combina os *patches* de Mobile IPv6 e IPSec).

O facto do Sockmon não funcionar no *kernel* do Moby Dick levou-nos ao desenvolvimento de um *patch* específico para o *kernel* 2.4.16. Esse *patch* tem basicamente a mesma funcionalidade do Sockmon, permitindo reproduzir as suas facilidades dentro da rede de testes existentes no Instituto de Telecomunicações [8].

B Mdidas de handover na rede Moby Dick

O *patch* desenvolvido foi utilizado para executar medidas da *cwnd* na rede Moby Dick. Os testes consistiam na realização de *handovers* entre dois Access Routers (AR) da rede Moby Dick, um AR *wireless* e um AR *Ethernet*.

O primeiro teste realizado foi um *handover* de tecnologia *wireless* para *ethernet*. Foi iniciada uma ligação TCP com o *Mobile Node* (MN) a enviar os pacotes para um *router wireless* e posteriormente foi ligado um cabo *ethernet* ao MN passando este a enviar pacotes pela *ethernet*.

A imagem seguinte mostra um gráfico da janela de congestionamento da ligação TCP referida a cima.

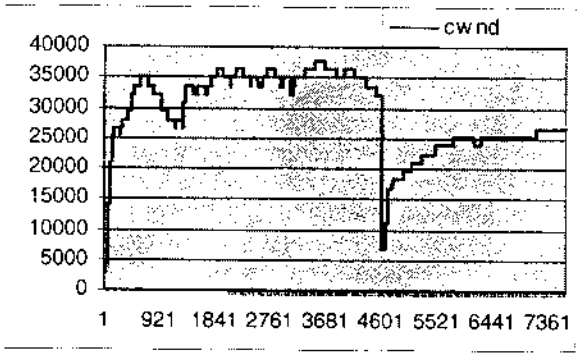


Figura 1– janela de congestionamento num handover wlan->eth

Como se pode verificar existe uma quebra no valor da janela, reflectindo a diminuição da velocidade de ligação. A imagem seguinte mostra em pormenor a quebra na janela. É importante verificar que embora a ligação TCP sofra uma redução abrupta da janela, vendo com mais atenção reparamos que existem valores intermédios, isto é, o valor não cai para um valor mínimo instantaneamente.

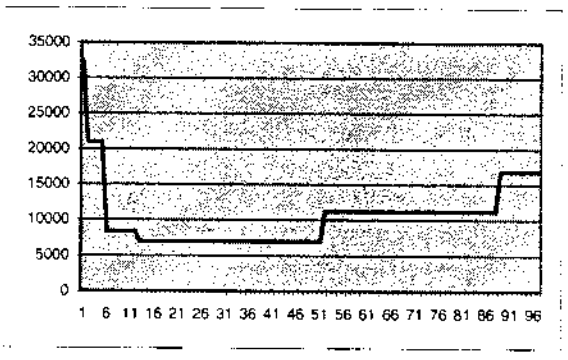


Figura 2 – pormenor da janela de congestionamento

Viemos a constatar que este comportamento é explicado pelo facto de o sistema operativo do MN começar a receber pacotes duplicados, uns vindos do AR *ethernet* outros vindos do AR *wireless*.

Num segundo teste realizou-se um *handover* da tecnologia *ethernet* para *wireless*. A Figura 3 apresenta um gráfico do valor da janela de congestionamento durante o *handover*.

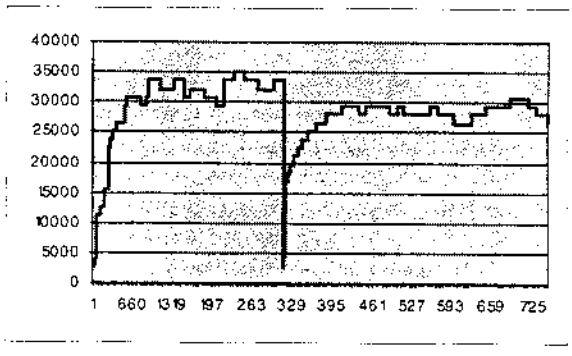


Figura 3 – janela de congestionamento num handover eth->wlan

É iniciada uma ligação TCP com o MN ligado à ethernet; durante a ligação é retirado o cabo da ethernet e o MN começa a enviar pacotes através da interface wireless. Como se pode verificar na imagem acima existe uma queda abrupta no valor da janela de congestionamento. Uma nova imagem, desta vez mais específica da queda abrupta do valor da janela, mostra-nos que ao contrário do que se passou no handover anterior, o valor da janela cai repentinamente para um valor mínimo, o que implica a existência de um *time-out*.

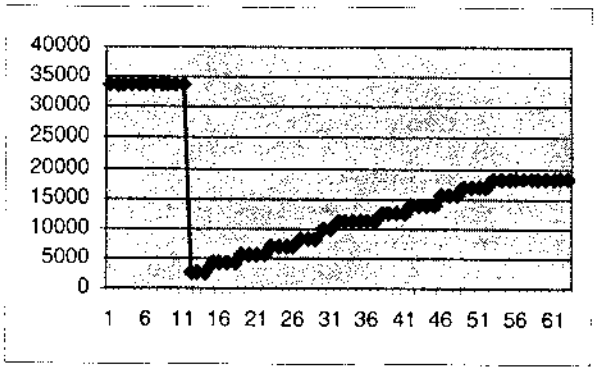


Figura 4 – pormenor da janela de congestionamento

Tal acontece pois o MN demora algum tempo a aperceber-se que o cabo *ethernet* foi retirado; esse intervalo de tempo é suficientemente grande para que aconteça um *time-out* na ligação TCP. De seguida, depois de se aperceber da inexistência de ligação física pela rede ethernet, o MN começa a utilizar a interface *wireless*.

III MEDIDAS BASEADAS EM PROBES

O IETF tem desenvolvido nos últimos anos, através do IPPM WG (*IP Performance Metrics Working Group*) [6], vários documentos onde são definidas, conceptualmente, as várias métricas e metodologias de medição. A lista de métricas especificada pelo IPPM WG é algo extensa, e é provável que seja ainda alterada, visto alguns documentos não estarem ainda encerrados.

O IPPM WG divide as métricas em 3 tipos:

singleton, são medidas atómicas, as quais correspondem a valores de uma medição, por exemplo o atraso de um pacote;

samples, referem-se a métricas que derivam das *singleton*. São calculadas utilizando um certo número de instâncias das *singletons*;

statistic, são estatísticas feitas sobre as métricas *samples*.

As mais importantes a referir são portanto as *singleton*, que se dividem também elas em dois tipos: *One-way*, que são medidas unidireccionais entre dois pontos, isto é, as que se compõem da ida de um pacote (ou conjunto de pacotes), de uma origem até um destino; e *Round-trip* que são medidas bidireccionais, isto é, as que se compõem da ida de um pacote (ou conjunto de pacotes) de uma origem até um destino e volta.

É importante salientar que no caso da utilização de medidas *Round-trip* não existe qualquer certeza de que o caminho de ida é o mesmo que o caminho de volta; não há sequer garantias que exista um caminho de volta estável.

O principal parâmetro na definição de métricas é o tipo de pacote com o qual se realizam as medidas; por exemplo a verificação de conectividade IP pode ter vários resultados consoante os pacotes de teste utilizados, especialmente se estivermos na presença de *firewalls*.

Devido às possíveis diferenças obtidas entre testes com a mesma finalidade, o IPPM WG definiu o conceito abstracto de "*Type-P packets*" (Pacotes de Tipo-P) [7].

Assim, os pacotes utilizados para efectuar as medidas, sejam elas de que tipo forem são identificados como *Type-P packets* que, depois de especificados, identificam inequivocamente uma determinada medida.

As métricas *singleton* englobam a possibilidade da verificação de conectividade IP, o cálculo de latências, perdas e débito. Com a aplicação de métodos estatísticos às métricas *singleton* é possível obter métricas derivadas: médias, *jitter*, lista de distâncias, mínimos, máximos, etc..

A realização das medidas acima referidas, pode ser feita através de medidas passivas ou activas.

A Monitorização Passiva

As medidas passivas utilizam o tráfego existente na rede. Como nenhum tráfego é artificialmente introduzido na rede, este tipo de monitorização só pode ser utilizado em situações onde os pacotes que se pretendem analisar já se encontram em trânsito. Um exemplo do uso deste tipo de aplicações é a negociação dos parâmetros de QoS na especificação de um serviço pedido por um cliente. Para a monitorização dos parâmetros de QoS é necessário a correlação de informação proveniente de diferentes pontos de medida (por exemplo para medidas *one-way*).

As medidas passivas não se remetem apenas à possibilidade de se capturarem pacotes, em tempo real, num determinado ponto de rede. A leitura efectuada a elementos de rede do operador e do cliente, colocando agentes nesses elementos (ou simplesmente efectuando leituras remotas da informação aí já existente, p.ex. com contadores em MIBs SNMP ou RMON) pode também ser considerado como uma medição passiva da rede. É de notar que dificilmente este tipo de leituras a objectos remotos nos dará uma visão da rede em tempo real. As informações assim obtidas são geralmente de âmbito mais alargado, e usadas para gestão de rede.

As medidas efectuadas de forma passiva devem ser aplicadas a dois tipos distintos de tráfego, rígido e elástico. Por tráfego rígido, entende-se aquele que é sensível a atrasos, onde não existe controlo de fluxo extremo a extremo. É tolerante a perdas e permite baixos valores de atrasos e de *jitter*. Geralmente trata-se de tráfego com requisitos de tempo real, seja *streaming* de vídeo, de áudio, ou mesmo telefonia *sobre* IP.

Pelo contrário, o tráfego elástico é pouco sensível a atrasos, permite ser adaptado às condições da rede mas é normalmente pouco tolerante a perdas. Como exemplo deste tipo de tráfego temos pacotes SNMP, HTTP, transferência de ficheiros, acesso a bases de dados, etc..

B Monitorização Activa

O funcionamento da monitorização activa baseia-se na introdução de tráfego na rede, por parte de um dos elementos de medida (origem), que será posteriormente recolhido da rede por outro elemento de medida (destino). A grande desvantagem deste método é o facto de se estar a introduzir pacotes na rede, pacotes esses que não contêm *dados reais*, podendo vir a aumentar o congestionamento da rede. No entanto, através da especificação de determinados parâmetros, como, por exemplo, a periodicidade de pacotes enviados, a quantidade, entre

outros, é possível minorar os efeitos da introdução de pacotes extra na rede.

Neste caso, as medidas efectuadas servem essencialmente para fazer verificações de SLA, detecção de falhas, engenharia de tráfego, e monitorização geral da rede. É ainda possível utilizá-las para fornecer informação a um *QoS Broker* para auxílio a decisões de forma a maximizar a utilização dos recursos disponíveis.

As medidas efectuadas de forma activa deverão, tal como as medidas passivas, levar em conta os dois tipos de tráfego, elástico e rígido.

As medidas podem ser divididas em medidas de nível de rede e medidas de nível de serviço. O primeiro tipo, nível de rede, não leva em consideração o tipo de tráfego que está em trânsito (que se está a medir) e tem apenas como objectivo avaliar atrasos, *jitter*, perdas e *throughput* em termos de pacotes. É importante referir que embora estas medidas possam ser alteradas com o tipo de tráfego que se encontra em trânsito devido às políticas de QoS existentes na rede, ao nível de rede essa informação não é importante. Se, por exemplo, desejar saber qual o atraso de um *stream* de áudio, essa informação não está directamente relacionada com a qualidade do *stream*, aquando da sua chegada ao destino.

As medidas a nível de serviço pretendem dar alguma informação sobre qual o estado do serviço em causa. O tipo de métricas associado a este tipo de medidas serão erros, tempos de *download*, tempos de acesso ou ainda a qualidade de um determinado *stream* áudio ou vídeo, através de algoritmos bem conhecidos como PESQ [10], entre outros. Por exemplo, através do uso do algoritmo PESQ é possível medir o grau de dificuldade que um ouvinte terá em entender um determinado *stream* de áudio.

É importante que a plataforma de medição activa seja capaz de gerar tráfego e interpretar resultados que possam satisfazer os dois tipos de tráfego acima indicados.

C *Arquitectura geral de uma plataforma de medidas*

Uma plataforma das medidas deve conter dois tipos de elementos: controlo e medida. Os elementos de controlo têm por objectivo definir qual o tipo de testes que se pretende fazer e enviar essa informação para os elementos de medida. É ainda da sua responsabilidade receber, guardar e retirar conclusões das medidas efectuadas pelos elementos de medida. Por exemplo, o elemento de controlo deve calcular dados estatísticos tendo por base os valores de medidas efectuadas pelos elementos de medida. Os elementos de medida têm por objectivo enviar e receber os pacotes de teste, com base nas informações de controlo recebidas do elemento de controlo. É ainda da sua responsabilidade extrair métricas directas dos pacotes de teste, bem como enviar os resultados para o elemento de controlo.

Os dados enviados dos elementos de medida para o elemento de controlo devem ser feitos através do protocolo IPFIX, já que se trata de um *standard* projectado exactamente com esse objectivo [4].

Para que os resultados obtidos sejam correctos é necessário que os vários elementos envolvidos nas medidas estejam de alguma forma sincronizados. Para a sincronização podem-se utilizar antenas GPS ou pode-se utilizar o protocolo NTP, tendo este último menor precisão.

IV SOFTWARE DESENVOLVIDO

A . *Descrição da plataforma IP Probes*

Como descrito na secção III-C, uma plataforma de medição tem geralmente dois tipos de elementos, o elemento de medida e o elemento de controlo. A plataforma IP-Probes segue esta definição e apresenta estes dois tipos de elementos. A designação para estes elementos foi de elemento de medição (EM) e elemento de operação (EO), respectivamente. Numa rede podem existir vários EMs e EOs, embora o mais normal seja que para um determinado número de EMs exista um EO de controlo. A figura 5 mostra de forma abstracta a localização dos diferentes elementos envolvidos na plataforma IP-Probes.

A definição de testes é feita tendo por base a definição de Sessão e de Sub-Sessão.

Uma sessão é um conjunto de N (N>=1) sub-sessões, que é identificada por um identificador único. As sub-sessões partilham as mesmas características da sessão no que diz respeito ao tipo de medição que se pretende. Uma sessão e as suas N sub-sessões partilham a seguinte informação:

O EM deve ou não enviar pacotes;

O EM deve reportar resultados;

Em qual das sentidos o EM deve reportar resultados (no caso de *two-way)*;

Tipo de métrica, isto é, se a sessão é *one-way*, *round-trip* ou *forward*;

Uma sub-sessão é identificada por um identificador único e por todas as características configuráveis que tiver. Essas características são as que existem para a construção de um pacote do tipo P.

A utilização de sub-sessões foi tomada em consideração pois pretendia-se que uma das funcionalidades da plataforma fosse executar sessões de teste de forma

recorrente (por exemplo, que todos os dias às X horas fosse executado um determinado teste). Assim, com a definição de sessão e sub-sessões associadas a uma sessão torna-se mais simples (organizada) a definição de uma sessão (ou sub-sessão) e o armazenamento dos resultados.

A imagem seguinte representa esquematicamente a plataforma IP Probe. Nela é visível a existência de dois elementos de medida e de um elemento de operação. Repare-se que o EO tem uma *interface* de comunicação, baseada em XML-RPC, que lhe dá a possibilidade de comunicar com entidades externas. É ainda visível na imagem a existência de antenas GPS, para a sincronização dos diversos elementos envolvidos nas medidas.

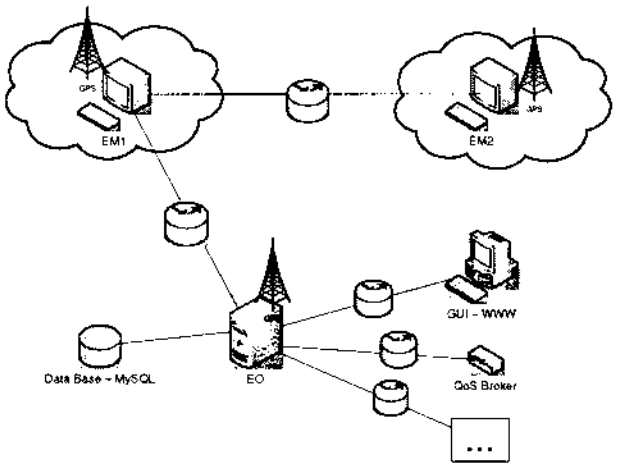


Figura 5 – Representação da plataforma IP Probe

B . Elemento de medida

O EM, tem como tarefa a realização das medidas e testes propriamente ditos, enviando e recebendo pacotes de teste (medição activa), com as características definidas, e guardando os resultados de uma forma não definitiva, isto é, não são escritos em disco. Outra responsabilidade do EM é o envio de resultados para o EO, periodicamente ou respondendo a pedidos explícitos.

O EM está dividido em dois blocos funcionais, o bloco de controlo e o bloco de medição.

O Bloco de Controlo deverá comunicar com o EO, recebendo os testes a realizar (sessões), e enviando resultados, quer totais, quer parciais, referentes à sessão que está a ser executada. Esta comunicação deverá ser feita através de uma *interface* de rede específica para esse efeito (a Interface de Gestão).

O Bloco de Medida deverá enviar e/ou receber pacotes de teste de/para outros EMs, fazendo de receptor, emissor ou de ambos. Um componente essencial nos EMs é o receptor GPS que faz com que todos os EMs estejam sincronizados. Isto é fundamental para obter bons resultados nas medidas *One-way Delay*, em que os pacotes saem de um EM e chegam a outro: os resultados temporais enviados para o EO devem ser consistentes daí a necessidade de as diferentes unidades estarem sincronizadas temporalmente. O protocolo NTP pode também ser utilizado para tal, ainda que não tenha a mesma precisão que a conseguida pelo GPS e necessária para redes com débitos elevados. Usando GPS, os EM podem também funcionar como servidores NTP. A comunicação entre os EMs é efectuada utilizando-se uma *interface* de rede diferente da utilizada para a comunicação com o EO (a *interface* de Teste).

Os EMs extraem métricas directas dos pacotes de teste obtidos, independentemente das estatísticas finais a produzir; são produzidas e enviadas para o EO, medidas auxiliares de apoio, como por exemplo, quantos pacotes enviados, volume total de bytes enviados, tamanhos pacotes enviados, etc.

É possível aos EMs funcionarem como *proxy*, isto é, um EM pode efectuar *forward* de tráfego de teste para outro EM, podendo ou não o elemento que está a servir de *proxy* efectuar medidas.

C . Elemento de operação (EO)

O EO, tem por responsabilidade receber os resultados de uma sessão de teste e enviar a configuração de novas sessões de teste para os EMs.

O EO, é um sistema dividido em três blocos funcionais, EO Low, EO High e interface XML-RPC, como mostra a Figura 6.

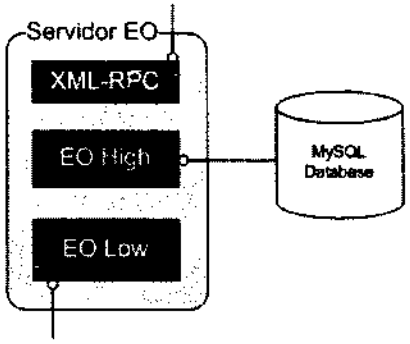


Figura 6 – Arquitectura do EO

O EO Low tem que lidar basicamente com duas acções: a configuração das Sessões e a obtenção dos resultados. O envio dos parâmetros relativo a uma sessão de teste entre o EO e os EMs é feito através do protocolo proprietário IP-Probe EO2EM Protocol (IPEP), assim como o envio de resultados dos EMs para o EO. Este bloco funcional é

responsável por toda a manipulação de rede entre os EMs e o EO.

O EO High é responsável pela comunicação com a base de dados e pela comunicação entre o EO LOW e a interface com o mundo exterior, através do bloco XML_RPC. Na base de dados são guardadas as configurações relativas às sessões, bem como os resultados obtidos através dos testes efectuados. É ainda guardada na base de dados toda a informação referente aos EMs e EOs que possam co-existir na rede.

A interface XML-RPC é a parte responsável pela comunicação entre o EO High e o mundo exterior. Este bloco funcional disponibiliza um conjunto de *interfaces* com o exterior que lhe permitem interagir com diferentes entidade, permitindo uma fácil integração com outras ferramentas, como por exemplo um *QoS Broker*. Com esta funcionalidade é possível que o EO esteja localizado numa outra máquina que não a máquina onde o utilizador está a definir os parâmetros da sessão de teste.

Existe neste momento uma *interface web* capaz de comunicar com o bloco funcional XML-RPC, para a definição de sessões e obtenção de resultados. É ainda possível a manipulação de EMs e de EOs, é possível por exemplo criar-se um novo EM dizendo quais os seus IPs, nome e o local onde se encontra.

V CONCLUSÕES

A utilização de uma plataforma de medição baseada em pontas de prova ou sondas é uma mais valia para os operadores de redes IP. A recolha de dados referentes às métricas normalizadas é a principal aplicação do sistema, permitindo que a definição de contratos de serviço prestados aos clientes ou a outros operadores sejam claramente definidos. A plataforma vem também permitir que os resultados sejam avaliados em tempo real permitindo que outras entidades, de controlo ou gestão, utilizem os resultados obtidos para se ajustar às condições da rede. Também o dimensionamento, o planeamento e a gestão podem ser facilitados, usando os dados obtidos na análise estatística. Em ambientes móveis esta plataforma tem especial interesse para analisar a variação do comportamento médio de desempenho do nós móveis, à medida que estes vão mudando de ponto de ligação à rede.

No entanto, este tipo de plataforma não está vocacionada para uma análise pormenorizada do comportamento do *handover*, algo que é importante para avaliar o comportamento protocolar da rede de comunicações. Neste sentido, a ferramenta TCP desenvolvida vem proporcionar um complemento interessante para um sistema de análise de desempenho para ambientes móveis da próxima geração.

O desenvolvimento do Elemento de Operação foi realizado com recurso à rede Moby Dick. Os resultados obtidos satisfizeram os requisitos iniciais do projecto, e em particular a análise feita ao comportamento do TCP permitiu clarificar alguns detalhes dos novos protocolos [8] desenvolvidos para esta rede.

AGRADECIMENTOS

Os autores do trabalho agradecem a utilização da rede Moby Dick no IT, e a disponibilização do software original de IP-Probes por parte da Portugal Telecom Inovação, que serviu de base a parte deste trabalho.

REFERÊNCIAS

[1] TCP Behavior Inference Tool - http://www.icir.org/tbit/

[2] Web100 project - http://www.web100.org/

[3] Sockmon Socket Monitoring Utility - http://sourceforge.net/projects/sockmon

[5] IP Flow Information Export (ipfix) - http://www.ietf.org/html.charters/ipfix-charter.html

[6] IP Performance Metrics (ippm) - http://www.ietf.org/html.charters/ippm-charter.html

[7] V. Paxson, G. Almes, J. Mahdavi, M. Mathis - Framework for IP Performance Metrics, http://www.ietf.org/rfc/rfc2330.txt

[8] D. Gomes, N. Duarte, N. Sénica, "Demonstrador de Mobilidade Heterogénea", Revista do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro - Vol. 4 - Nº1, Setembro de 2003, ISSN: 1645-0493

[9] Moby Dick Project - http://www.ist-mobydick.org/

[10] Perceptual evaluation of speech quality - http://www.itu.int /rec/recommendation.asp?type=folders&lang=e&parent=T-REC-P.862

The ARPA Project - Creating an Open-Source Real-Time System-on-Chip

Arnaldo S. R. Oliveira, Valery A. Sklyarov, António B. Ferrari

Abstract – This paper describes the ARPA project. The aim of this project is to develop an open-source System-on-Chip model for real-time applications. The main component of the SoC is a MIPS-based RISC processor. It is implemented using a pipelined Simultaneous Multithreading (SMT) structure, which allows exploring the Instruction and Task Level Parallelism, decrease the context switching time and avoid speculative execution. Another fundamental component of the SoC is the Operating System Coprocessor, which implements in hardware some of the operating system functions, such as task scheduling, switching, synchronization, communication and timing. The proposed approach allows building high performance and time predictable processors optimized for embedded real-time systems that consume less energy than currently available superscalar processors.

Resumo – Este artigo descreve o projecto ARPA. O objectivo deste projecto é a concepção de um modelo aberto de um sistema integrado para aplicações de tempo real. O componente principal do sistema é um processador RISC baseado na arquitectura MIPS e implementado usando uma estrutura pipelined com suporte para multitarefa simultânea. Esta implementação permite combinar a exploração do paralelismo entre instruções de uma e de várias tarefas, diminuir o tempo de comutação de tarefas e evitar a utilização de técnicas complexas de execução especulativa. Outro componente fundamental do sistema é o coprocessador de sistema operativo, que implementa em hardware algumas das funções de sistema, tais como temporização, escalonamento, comutação, sincronização e comunicação entre tarefas. A abordagem proposta permite construir processadores de elevado desempenho, previsíveis e optimizados para sistemas de tempo real e que consomem menos energia que os processadores superescalares actuais.

Keywords – Simultaneous multithreading, Time-predictable processors, Operating system coprocessors.

Palavras chave – Multitarefa simultânea, Processadores temporalmente previsíveis, Coprocessadores do sistema operativo.

I. Introduction

The number of transistors on a single chip has increased considerably over the last decades. The recent advances in integrated circuit (IC) technology has allowed the construction of complex Systems-on-Chip (SoC) and the manufacturing of large field programmable logic devices, such as FPGAs. FPGAs with the equivalent logic capacity of millions of gates are now available at reasonable price and are considered a strong alternative to custom ICs (ASICs), mainly due to its reconfigurability and low NRE costs. Assuming the continuous evolution of the IC technology, the next generation of processors will reach, by the year 2010, the 10 billions of transistors on a single chip. The mission of the engineers and computer architects is to find the best way to use efficiently this huge number of transistors, which obviously depends on the application domain.

The advances in computer architecture during the last decade led to the construction of very fast and extremely complex superscalar processors, employing advanced techniques to improve performance, such as superpipelining, branch prediction/speculation, out-of-order/predicated execution and sophisticated memory hierarchies. However, these techniques are also responsible for much of the power consumption and for the non deterministic performance of nowadays computers. Although, they were successfully applied to improve the performance of desktop computers, they are not appropriate for embedded real-time systems because predictable performance and in some cases low power consumption are considered important properties for this class of systems. In this paper we suggest an approach to explore the large integration capacity currently available, to build optimized, predictable and energy efficient SoCs for embedded real-time systems.

II. Motivation

Embedded hard real-time systems typically consist of a set of concurrent tasks executing periodically or aperiodically with a well defined deadline. The execution is usually done on a single processor, which is able to execute only one task at a time. Therefore, a Real-Time executive or Operating System (OS/RTOS) is normally used to manage the concurrent execution of the tasks, performing the following roles:

- Schedule task execution accordingly to the predefined parameters and the established scheduling policy;

- Switch, i.e. preempt and dispatch, tasks accordingly to the computed schedule;
- Synchronize task execution and provide the adequate primitives to ensure a correct sequential behavior;
- Provide efficient mechanisms for inter-task communication.

In systems with high temporal resolution and/or a large number of (computationally intensive) tasks with tight parameters (small periods and deadlines) a high performance processor is required and a large number of context switches between tasks and the OS are performed during runtime. Context switches take time and spend energy without doing any useful work. A possible approach to improve the performance of such systems would be the utilization of an optimized processor with the following capabilities:

- Simple pipelined execution of instructions from each task, avoiding the complex techniques used currently in superscalar processors to minimize pipeline stalls;
- Simultaneous execution of several tasks eventually time sharing the same functional units, i.e. the instruction fetch stage is able to feed the succeeding pipeline stages with instructions from multiples tasks;
- Efficient hardware implementation of the OS functions using an Operating System Coprocessor (OSC);
- Reduced overhead associated with context switches.

Such a processor would reduce the number of context switches, explore both types of parallelism (Instruction Level Parallelism - ILP and Task Level Parallelism - TLP) and improve the energy efficiency without require considerably hardware complexity. The OSC based implementation of OS activities such as, task scheduling, switching, communication and synchronization also improves the system performance because they are performed in parallel with the user tasks, reducing processor load and the number of context switches. A close interaction between the OSC and the CPU core would also improve the utilization of CPU resources and increase the ability to introduce optimizations (e.g. at the context switching level).

III. OBJECTIVES

The main objective of the ARPA project is to create, implement and test a synthesizable model of a SoC optimized for hard real-time systems, as well as develop or adapt the respective compilation tools. This ambitious goal can be divided into the following parts:

- Create a processor architecture optimized for hard real-time systems. It must provide high performance and execute the tasks accordingly to the timing constraints specified during system design. The architectural optimizations must not be responsible for a considerable increase in power consumption. The use of complex performance improvement techniques which consume considerable area and power and introduces non-determinism must be avoided. To save project time, this work could be based on an existing architecture with stable development tools widely available.
- Develop an optimized pipelined implementation of the architecture employing simple techniques to improve performance and avoid pipeline stalls caused by control and data hazards. The implementation must explore different types of parallelism (ILP and TLP) and exhibit low overhead associated with context switches.
- Using an hardware description language (e.g. VHDL) and/or a high-level language (e.g. Handel-C) elaborate a behavioral model of the processor, which must be used directly for hardware synthesis. The model must be portable, i.e. independent of the target technology.

IV. RELATED WORK

Intel Corporation launched in 2001 the Hyper-Threading (HT) technology [1]. A HT-enabled processor is seen by the OS as a single physical processor acting as multiple logical processors. HT improves the performance of multitasking applications executing in a processor by dynamically multiplexing instructions from multiple tasks into the same functional units. Current Pentium 4 HT processors can execute two tasks simultaneously by time sharing the same hardware resources. The Sweden company RealFast launched in 2003 the "UltraFast Micro Kernel" real-time kernel/executive, previously named "Sierra 16" [2]. It implements in hardware some of the operating systems functions, conventionally performed in software. Task scheduling is fully implemented in hardware and the interrupt service routines are scheduled as ordinary tasks accordingly to a predefined priority. The CPU has a single interrupt request line driven by the executive and activated whenever a task switching has to be performed. The executive also provides timing primitives and synchronization mechanisms. Data exchange between the CPU and the executive is performed through the system bus, acting the coprocessor as an ordinary peripheral. In [3] it is also presented a configurable scheduler for real-time systems. The principle of operation and the interface are similar to the "UltraFast Micro Kernel".

The ARPA project is on the same research line of the referred work, but follows a different approach:

- Integrating closely the CPU and the OSC, allowing a more efficient use of the functional units and an optimized context switching procedure.
- Providing the means to dynamically select the scheduling policy and the static choice, i.e. at compile time, of memory sizes, number of execution contexts, task synchronization and communications mechanisms.

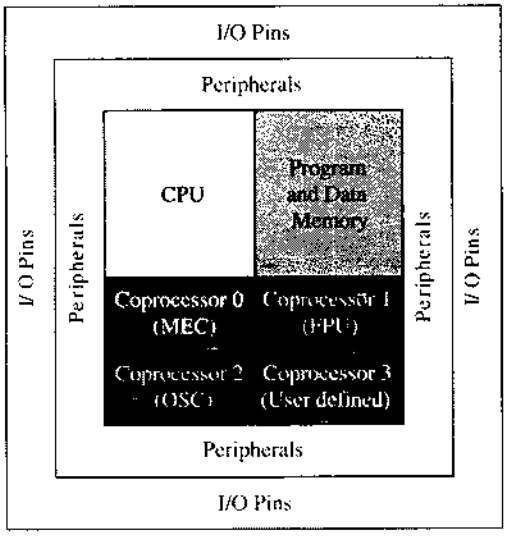


Figure 1 - Target structure for the ARPA SoC.

Comparing the ARPA project with the HT technology and the "UltraFast Micro Kernel", our approach has the advantage of simultaneously implement the OS in hardware and provide a closer interaction between the CPU and the OSC. This is possible because we are developing open models of both components.

V. ARCHITECTURE

The target structure for the ARPA SoC is depicted in fig. 1. Its main components are the following:

- The Central Processing Unit (CPU);
- Up to 4 coprocessors for exception handling, memory management, OS implementation, floating point calculations and user-defined operations;
- Program and data memory;
- Peripherals and I/O pins.

ARPA is based on the MIPS32 architecture because of its simplicity, ease of implementation and availability of stable compilation tools. In fig. 2 it is depicted the CPU pipeline. The main stages are named on the top of the figure. Depending on the implementation technology and required performance/area tradeoff, each of these stages can be split into sub-stages to bound the critical path. To support Simultaneous Multithreading (SMT), N execution contexts are provided, each supporting the execution of an individual task. Each context has its own Program Counter and bank of General Purposes Registers. The remaining pipeline components are shared by all contexts.

ARPA is an Harvard Architecture, i.e. contains separate memories for program and data. This memory organization eliminates structural hazards during memory accesses. The required duplication of the bus signals is not an issue because the CPU and memory are both integrated into the SoC. The need to predefine the memory sizes assigned to code and data is not a problem in embedded systems, because they are normally fixed at design time.

A distinctive feature of the ARPA project is the hardware implementation of the RTOS. The basic OS functions are implemented in hardware and performed in parallel with the user tasks. This approach has several advantages, such as better performance, improved predictability and requires less memory for the RTOS code and data. The main OSC blocks are shown on the bottom side of fig. 2. The *Task Table* is used to store the task parameters (entry point, state, timing information, stack size, etc.). All units have access to the *Task Table*. The *Context Switching Unit* has privileged access to the CPU pipeline. The interface and data exchange between the CPU and the OSC are based on the predefined MIPS coprocessor interface and instruction set.

VI. IMPLEMENTATION

The ARPA architecture can be implemented in different ways depending on the required performance and features. To obtain a reasonable performance/complexity compromise, the first implementation is based on a single issue, 5 stage multithreaded pipeline. To simplify and save hardware resources, all traditional superscalar techniques, such as branch prediction, and speculative/out-of-order execution were avoided. The adoption of a SMT implementation minimizes the negative impact of the proposed simplifications in the performance. Pipelined SMT allows the exploration both ILP and TLP. The ARPA SoC is being prototyped on a Trenz Electronic TE-XC2Se development system [4], which contains a XC2S300E Xilinx FPGA. The CPU core is being modeled at RTL level using VHDL because the operations performed are relatively simple and VHDL provides good control over hardware synthesis. For CPU synthesis the Xilinx XST engine is used. On the other hand, the OSC is being modeled in Handel-C because the operations and algorithms are more complex and Handel-C provides the adequate support for describing them. For OSC synthesis the Celoxica's DK2 suite is being employed. The netlists produced by both synthesis tools are integrated and used within Xilinx ISE tools for implementation on the target device.

VII. APPLICATION DESIGN FLOW

The design flow for FPGA based applications using the ARPA architecture is depicted in fig. 3. It consists of two sub-flows: the hardware design flow and the software design flow. The hardware sub-flow allows obtaining, with the aid of the synthesis and FPGA vendor implementation tools, the hardware configuration file from the behavioral SoC description. The output of the software sub-flow is an executable file generated by compilation of the software source code and used to program the system memory. If the FPGA internal memory is large enough to store the application code and data, the executable file can be merged with the hardware configuration file into a single file used to program the FPGA at system startup. The executable file

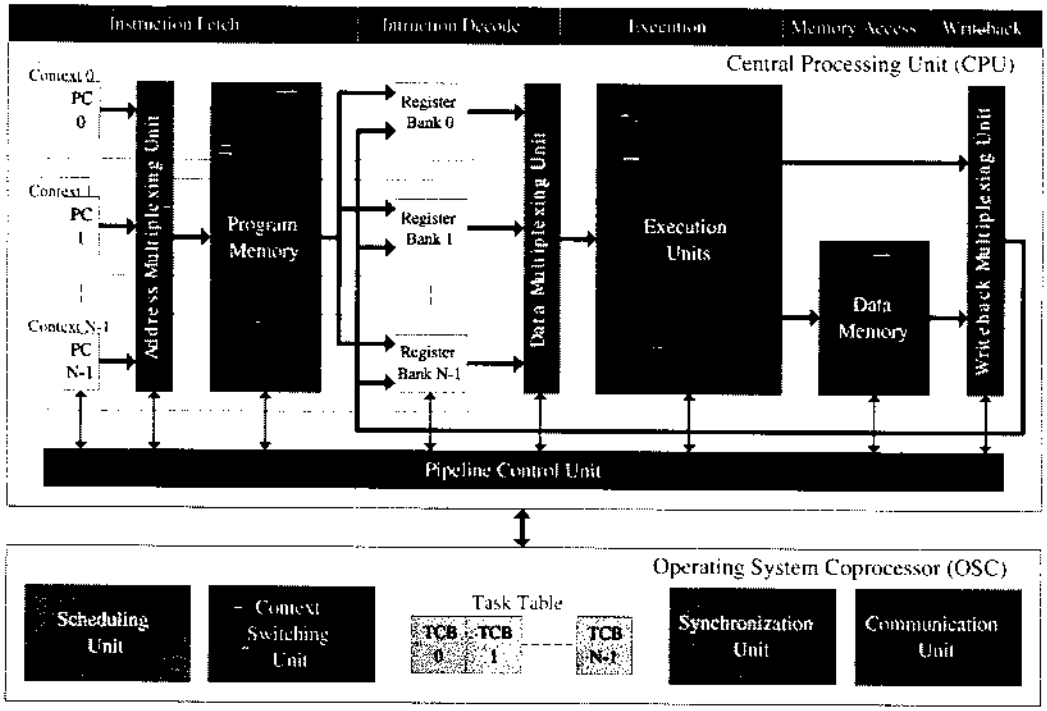


Figure 2 - ARPA CPU and OSC internal structure.

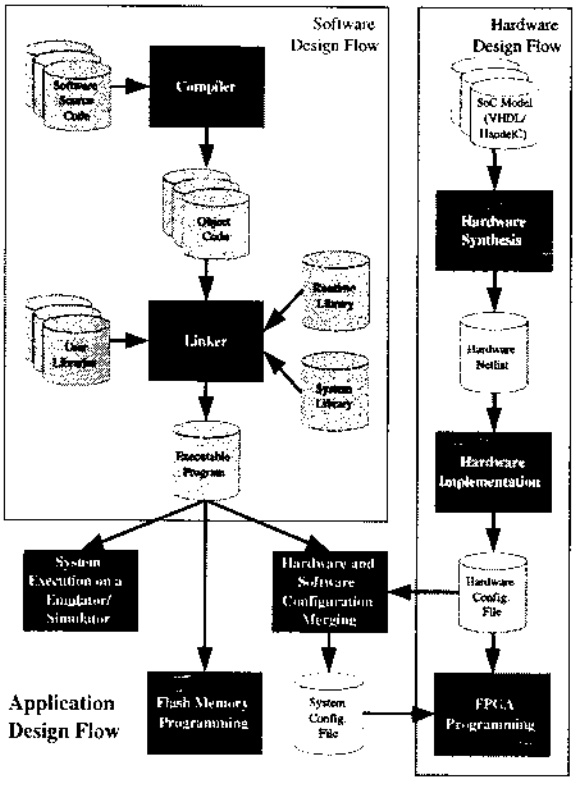


Figure 3 - Application design flow assuming an FPGA implementation.

resulting from software compilation can also be used for simulation purposes within an emulator.

VIII. CONCLUSIONS

The ARPA architecture provides a convenient platform for implementing hard real-time systems. It provides hardware support for traditional OS functions and allows exploiting both ILP and TLP. The less aggressive exploration of ILP is compensated by the TLP exploration and the hardware implementation of OS functions which allows improving performance without the complexity and power consumption of current superscalar processors. The web page for the ARPA project is at the following address: http://www.ieeta.pt/~arnaldo/projects/ARPA/.

REFERENCES

[1] D. Marr D. Koufaty. "Hyperthreading technology in the netburst microarchitecture". *IEEE Micro*, pp. 56–65, March-April 2003.

[2] RealFast. http://www.realfast.se. *UltraFast Micro Kernel*. 2003.

[3] V. Mooney III P. Kuacharoen, M. Shalan. "A configurable hardware scheduler for real-time systems". in *Engineering of Reconfigurable Systems and Algorithms*, Las Vegas - USA, 2003, pp. 96–101.

[4] Trenz Electronic. http://www.trenz-electronic.de. *Spartan-IIE Development Platform Overview*. 2004.

Escalonador de Tempo Real em SystemC

Hugo Manaia, Arnaldo Oliveira, Nuno Lau, Orlando Moreira

Abstract – **This paper discusses the implementation, features and utilization of a real-time scheduler simulator modeled with SystemC. Currently, this scheduler handles only periodic hard real-time tasks. Three scheduling policies were implemented: RM (Rate Monotonic), EDF (Earliest Deadline First) and LSF (Least Slack First). A set of task-related commands is available, which allows the user to create, destroy, stop, start, sleep, suspend, resume and change the task parameters. These commands can be specified in a configuration file that is read during initialization. In the configuration file, besides the specific parameters of each command, a time stamp is assigned to each command. In each simulation a VCD (Value Change Dump) file is generated, which allows the visualization of the evolution of some of the most important system signals.**

Resumo – **Este artigo descreve a implementação, as funcionalidades e a utilização de um simulador de um escalonador de tempo real desenvolvido em SystemC. Actualmente este escalonador aceita apenas tarefas críticas periódicas de tempo real. Três políticas de escalonamento foram implementadas: RM (*Rate Monotonic*), EDF (*Earliest Deadline First*) e LSF (*Least Slack First*). É disponibilizado um conjunto de comandos que permite manipular as tarefas, dando a possibilidade ao utilizador ou ao escalonador de criar, destruir, parar, arrancar, adormecer, suspender, retomar e alterar parâmetros das tarefas. Estes comandos podem ser especificados num ficheiro de configuração que é lido durante a fase de inicialização. Neste ficheiro de configuração especifica-se, para além dos parâmetros específicos de cada comando, o tempo no qual se pretende que este ocorra. Em cada simulação é gerado um ficheiro VCD (*Value Change Dump*) através do qual se pode acompanhar a evolução de alguns dos sinais mais importantes que caracterizam o estado do sistema.**

Keywords – **Real-time systems, Task scheduling, Rate Monotonic, Earliest Deadline First, Least Slack First, SystemC**

Palavras chave – **Sistemas de tempo real, escalonamento de tarefas, *Rate Monotonic*, *Earliest Deadline First*, *Least Slack First*, SystemC**

I. INTRODUÇÃO

Os sistemas computacionais uniprocessador lidam cada vez mais com tarefas de tempo real, em especial tarefas críticas de tempo real. A estes sistemas dá-se o nome de sistemas de tempo real.

As tarefas de tempo real têm características diferentes das usuais, com restricções temporais que terão que ser cumpridas. Aos sistemas de tempo real é pedido que executem as tarefas de forma a estas cumprirem as restricções. Para isso, os sistemas agendam as tarefas para execução mediante as suas necessidades, e não numa política *First Come First Served*.

O escalonador é o componente de um sistema de tempo real encarregue do agendamento para execução de tarefas. Este agendamento é feito, normalmente, com base em prioridades. Para a realização desta actividade, um escalonador tem em conta diversas características das próprias tarefas, em função da política de escalonamento a utilizar. Em cada instante, é agendada para execução a tarefa pronta a executar com prioridade mais elevada. As prioridades podem ser fixas ou dinâmicas. No caso de um sistema de tempo real que incorpore escalonamento de tarefas com atribuição dinâmica de prioridades é a política de escalonamento que define como devem as prioridades das tarefas variar.

Para além desta introdução, este artigo estende-se ao longo de mais 6 secções. Na secção seguinte resumem-se alguns conceitos relativos a sistemas de tempo real, úteis na compreensão deste artigo. Na terceira secção é apresentada a arquitectura do escalonador desenvolvido. Na quarta secção discutem-se alguns pormenores de implementação. Na secção V é descrito como criar um exemplo e são analisados três exemplos apresentados. Por fim são apresentados alguns tópicos referentes a possível trabalho futuro bem como algumas conclusões.

II. SISTEMAS DE TEMPO REAL

Ao contrário de outros tipos de sistemas nos quais o mais importante pode passar pelo desempenho médio ou pela sequência de operações, um sistema de tempo real é [1], [2]:

- Um sistema computacional capaz de responder a eventos dentro de restrições temporais precisas;
- Um sistema cujo funcionamento correcto depende do instante de produção das saídas e não apenas do valor destas;
- Um sistema cuja dinâmica deve estar sincronizada com a do ambiente em que opera.

Assim, um sistema de tempo real não necessita necessariamente de responder rapidamente, mas sim de responder no instante certo, de responder a tempo.

A. Tarefas

Os sistemas de tempo real são, em geral, constituídos por um conjunto de tarefas ou processos que executam concorrentemente. Uma tarefa é uma sequência de instruções que, na ausência de outras actividades, é executada ininterruptamente pelo processador até ser completada. Simplificando um pouco, uma tarefa pode encontrar-se num de 3 estados [1]:

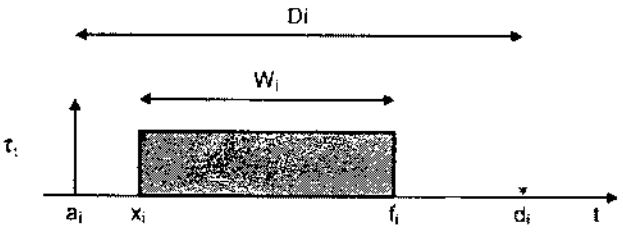


Figure 1 - Parâmetros temporais de uma tarefa.

- A executar
- Pronta a executar
- Bloqueada

As tarefas a executar ou prontas a executar, ou seja, as tarefas que num dado instante puderem ser executadas pelo processador, designam-se por activas.

Em termos de modo de activação, as tarefas podem ser de dois tipos distintos [1], [2]:

- Periódicas - a tarefa é automaticamente activada em intervalos de tempo regulares;
- Aperiódicas - a tarefa é activada assincronamente quando ocorrer um evento ou através da invocação explícita de uma primitiva de activação.

Em relação ao nível de criticalidade, as tarefas podem ser caracterizadas da seguinte forma [1], [2]:

- Ordinárias - se não possuírem quaisquer restrições temporais;
- *Soft real-time* - se o não cumprimento duma restrição temporal causar apenas uma degradação do desempenho do sistema;
- *Hard real-time* - se o não cumprimento duma restrição temporal causar efeitos catastróficos no sistema controlado.

A.1 Parâmetros

A execução de uma tarefa τ_i é caracterizada por diversos parâmetros, entre os quais (figura 1) [1], [2]:

- a_i - instante de activação;
- x_i - instante de execução;
- f_i - instante de terminação;
- d_i - *deadline* absoluta;
- D_i - *deadline* relativa;
- W_i - tempo máximo de execução (*Worst Case Execution Time* - WCET).

Uma tarefa periódica, além dos parâmetros acima, tem também um período T_i associado. Numa tarefa periódica caracterizada pelo terno $\tau_i(W_i, T_i, D_i)$ verificam-se as seguintes relações [1], [2]:

$$a_{i,k} = a_{i,k-1} + T_i$$

$$d_{i,k} = a_{i,k} + D_i$$

$$W_i \leq T_i$$

em que k é a k-ésima activação da tarefa.

B. Escalonamento

Um escalonamento é uma atribuição particular de tarefas ao processador. À estratégia usada para escolher uma tarefa entre as que se encontram activas e atribuir-lhe tempo de processador, chama-se algoritmo ou política de escalonamento.

O escalonamento em sistemas de tempo real pode ser efectuado de diversas maneiras. Uma das mais utilizadas é o escalonamento baseado em prioridades, que poderão ser fixas ou dinâmicas. As políticas disponíveis no escalonador implementado são:

- *Rate Monotonic* (RM);
- *Earliest Deadline First* (EDF);
- *Least Slack First* (LSF).

B.1 Prioridades fixas e dinâmicas

As políticas de escalonamento baseadas em prioridades podem usar prioridades fixas ou dinâmicas.

Utilizando uma política baseada em prioridades fixas, a política define a prioridade a atribuir a cada tarefa e esta não mais é alterada ao longo do escalonamento. As prioridades das tarefas são definidas com base em parâmetros ou características fixas das tarefas.

Com uma política que use prioridades dinâmicas, pelo contrário, as prioridades das tarefas podem ser alteradas durante o processo de escalonamento. As políticas de escalonamento baseadas neste tipo de prioridades têm em conta não só características fixas das tarefas mas também parâmetros só conhecidos em tempo de execução do escalonador.

De entre as políticas disponíveis no escalonador implementado, a RM baseia-se em prioridades fixas sendo que as restantes (EDF e LSF) utilizam prioridades dinâmicas.

B.2 Rate Monotonic

O algoritmo RM é um algoritmo de escalonamento baseado em prioridades fixas. A atribuição das prioridades é feita com base no período das tarefas, sendo inversamente proporcionais a este. É seleccionada para execução a tarefa com período mais pequeno. Uma tarefa pode ser interrompida por uma de prioridade mais elevada, pelo que se diz que o algoritmo é preemptivo.

O algoritmo RM é considerado óptimo entre os algoritmos de escalonamento de prioridades fixas, ou seja, se um conjunto de tarefas for escalonável utilizando uma qualquer política de prioridades fixas, então é escalonável utilizando RM.

A figura 2 mostra um exemplo de 3 tarefas usando a política de escalonamento RM.

Na figura cada linha temporal representa uma tarefa, identificadas por τ_i. As setas ascendentes representam activações das tarefas, ao passo que as descendentes representam as *deadlines* das mesmas. Os blocos representam os intervalos de tempo nos quais as tarefas executam realmente. De notar que a tarefa τ_3 é a tarefa com mais alta prioridade pois é aquela que executa mais frequentemente. Assim, cada vez que esta tarefa fica pronta a executar a tarefa que estiver na posse do processador sofre preempção, ou seja, o escalonador interrompe a sua execução. No lado

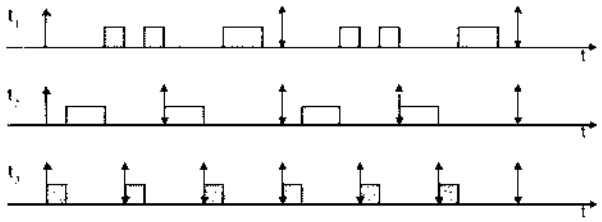


Figure 2 - Exemplo de um escalonamento de 3 tarefas usando o algoritmo RM.

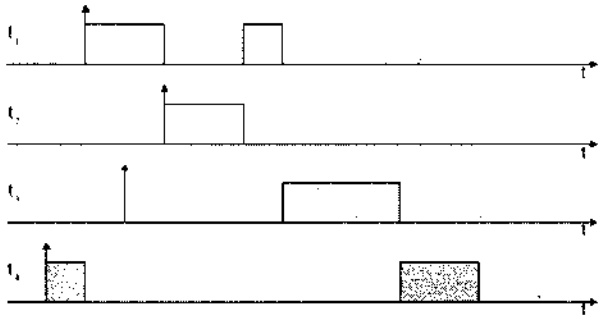


Figure 3 - Exemplo de um escalonamento de 4 tarefas usando o algoritmo EDF.

oposto encontra-se a tarefa τ_1 que, sendo a tarefa com prioridade mais baixa, é interrompida cada vez que uma outra tarefa se encontra pronta a executar.

B.3 Earliest Deadline First

O algoritmo EDF é um algoritmo de escalonamento baseado em prioridades dinâmicas, isto é, tem em conta um parâmetro dinâmico só conhecido em tempo de execução do escalonador. Neste algoritmo, as prioridades das tarefas são inversamente proporcionais ao tempo até à *deadline*. É seleccionada para execução a tarefa com *deadline* absoluta mais próxima. Tal como o algoritmo RM, este é um algoritmo preemptivo.

O algoritmo EDF é considerado óptimo entre todos os algoritmos de escalonamento, ou seja, se um conjunto de tarefas for escalonável, então é escalonável utilizando EDF, e permite alcançar uma taxa de ocupação do processador de 100%.

A figura 3 mostra um exemplo de 4 tarefas usando a política de escalonamento EDF.

B.4 Least Slack First

O algoritmo LSF é um algoritmo de escalonamento baseado em prioridades dinâmicas, isto é, tem em conta um parâmetro dinâmico só conhecido em tempo de execução do escalonador. Neste algoritmo, as prioridades das tarefas são inversamente proporcionais ao tempo livre das mesmas, isto é, são inversamente proporcionais à folga que a tarefa teria para a *deadline* caso executasse ao longo do seu tempo máximo de execução. É seleccionada para execução a tarefa com tempo livre mais pequeno. Tal como os algoritmos anteriores, este é um algoritmo preemptivo. Comparando com o algoritmo EDF, este provoca maior número de preempções.

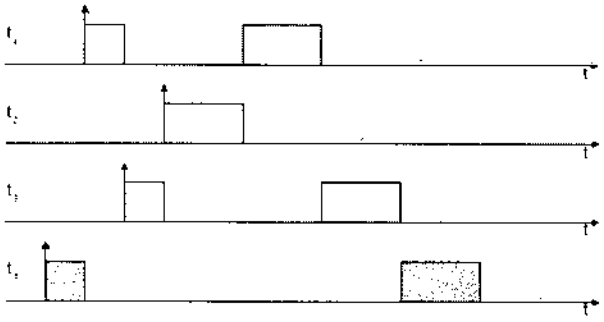


Figure 4 - Exemplo de um escalonamento de 4 tarefas usando o algoritmo LSF.

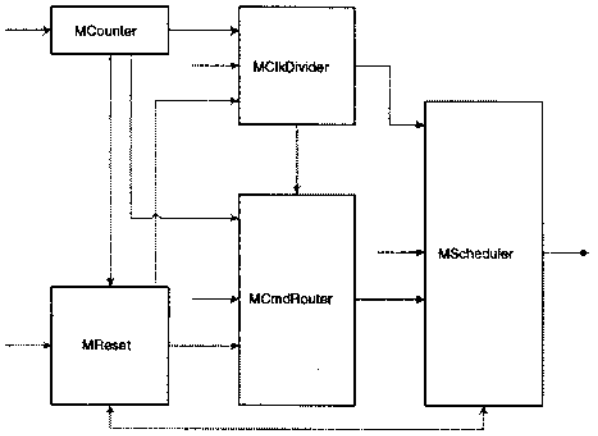


Figure 5 - Arquitectura do escalonador.

Tal como o algoritmo EDF, o algoritmo LSF é considerado óptimo entre todos os algoritmos de escalonamento, ou seja, se um conjunto de tarefas for escalonável, então é escalonável utilizando EDF, e permite alcançar uma taxa de ocupação do processador de 100%.

A figura 4 mostra um exemplo de 4 tarefas usando a política de escalonamento LSF.

III. ARQUITECTURA

A figura 5 ilustra a arquitectura do escalonador implementado, dando já uma ideia do fluxo de informação no sistema.

A figura 6 refere-se às características dinâmicas do escalonador, as entradas que este considera para sua configuração inicial bem como as saídas produzidas por este. Como entradas admite dois ficheiros de configuração, `scheduler.conf` e `reset.conf`, e alguns parâmetros passados na linha de comandos. Como saídas produz algum texto directamente no ecrã tal como um ficheiro VCD, que pode mais tarde ser visualizado com o programa `gtkwave` [3].

Mais à frente será dada mais informação acerca das entradas e saídas do escalonador, bem como uma descrição mais detalhada de cada módulo implementado.

A. *Tarefas*

Uma tarefa pode estar num dos seguintes estados:

- *Idle* (Inactiva)
- *Ready* (Pronta)

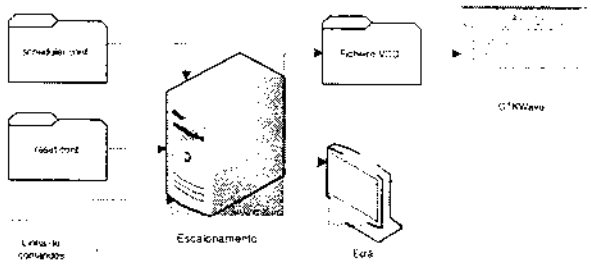


Figure 6 - Características dinâmicas do escalonador.

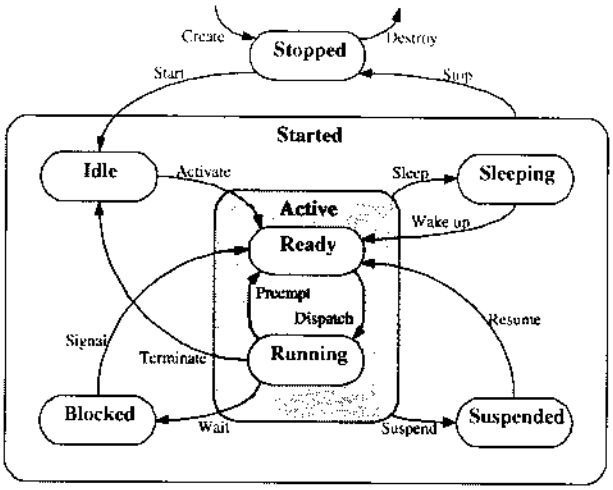


Figure 7 - Estados das tarefas e possíveis transições.

- *Running* (A executar)
- *Sleeping* (Adormecida)
- *Suspended* (Suspensa)
- *Stopped* (Parada)
- *Blocked* (Bloqueada)

Qualquer tarefa que se encontre num dos cinco primeiros estados (*Idle*, *Ready*, *Running*, *Sleeping* ou *Suspended*) diz-se iniciada (*Started*), caso contrário diz-se parada (*Stopped*). Uma tarefa que se encontre num dos estados *Ready* ou *Running* diz-se activa (*Active*). A figura 7 ilustra os vários estados em que uma tarefa se pode encontrar, bem como as transições permitidas.

Uma tarefa, quando é criada, é colocada no estado *Stopped*, permanecendo nesse estado até ser explicitamente iniciada. Depois de iniciada é colocada no estado *Idle* até ser alcançado o instante da primeira activação. Nessa altura transita para o estado *Ready*. Uma tarefa no estado *Ready* está pronta a executar. A passagem para o estado *Running* é feita pelo algoritmo de escalonamento através da atribuição de tempo de processador à tarefa. Uma tarefa que se encontre no estado *Running* pode ser interrompida pelo escalonador e passando para o estado *Ready*, ao que se chama preempção. Uma tarefa activa pode ser adormecida (passagem para o estado *Sleeping*) ou suspensa (passagem para o estado *Suspended*). O retorno ao estado *Ready* é automático no caso de uma tarefa adormecida e feito quando expirar o respectivo tempo. Uma tarefa suspensa deve ser reactivada explicitamente. Uma tarefa, depois de terminar, é colocada no estado *Idle* até à próxima activação. Uma tarefa activa é colocada no estado *Blocked* sempre que estiver à espera dum evento ou da libertação dum recurso partilhado. Este estado não se encontra implementado no escalonador. Em qualquer estado pode ser dada ordem de paragem a uma tarefa, transitando neste caso para o estado *Stopped*. Uma tarefa pode ser destruída, deixando de existir no sistema.

B. *Módulos*

A arquitectura do escalonador contempla diversos módulos sendo que cada um representa um possível componente físico, na perspectiva de simulação da implementação do escalonador em *hardware*.

Assim, os seguintes módulos estão presentes actualmente no escalonador:

- `MClkDivider` - Módulo responsável pela geração do relógio interno do escalonador de tempo real, que tem por base o *clock* interno do sistema;
- `MCounter` - Módulo encarregue da contagem do número de ciclos do *clock* interno do sistema já decorridos desde o início da simulação;
- `MReset` - Módulo responsável pelo botão de *reset* do sistema;
- `MScheduler` - Módulo no qual todo o escalonamento é feito, bem como todas as operações relativas aos blocos de controlo das tarefas (*Task Control Blocks* - TCB's);
- `MCmdRouter` - Módulo encarregue da interpretação dos comandos através dos quais se efectuam as operações relativas às tarefas.

IV. IMPLEMENTAÇÃO

A implementação do escalonador, feita na linguagem SystemC, assenta essencialmente em duas partes: a estrutura que define os blocos de controlo das tarefas e os diferentes módulos. Um outro aspecto é a possibilidade de configuração do sistema, discutida mais à frente nesta secção.

A. *O SystemC*

O escalonador desenvolvido foi implementado usando a linguagem de programação SystemC [4].

O SystemC é uma biblioteca da linguagem C++, que a dota de mecanismos que permitem simular a modelação de *hardware*. Assim, esta linguagem foi escolhida pois permite simular a implementação do escalonador em *hardware*, com todos os aspectos que lhe estão associados. O SystemC simula a execução concorrente dos módulos desenvolvidos e permite a utilização de *interfaces* tipo *hardware*, incluindo a possibilidade de tornar métodos sensíveis a determinados sinais, que despoletam a sua execução. Ao mesmo tempo, permite a composição de algoritmos de alto nível com algoritmos de baixo nível e algoritmos sequenciais com algoritmos paralelos, permitindo o refinamento do código até uma eventual implementação concreta em *hardware*.

B. A Estrutura sc_TCB

A estrutura sc_TCB define o bloco de controlo da tarefa (TCB - *Task Control Block*). Esta estrutura possui vários campos onde são armazenados os parâmetros de configuração e guardado o estado da respectiva tarefa. Do ponto de vista do escalonador implementado a execução das tarefas é modelada através da passagem do tempo de simulação, não existindo qualquer execução de código da tarefa. O tempo de execução de uma tarefa é determinado pelo simulador através dos parâmetros definidos pelo utilizador durante a criação da tarefa. Para cada tarefa presente no sistema existe apenas um TCB que contém a informação relativa aos seus parametros de execução.

A definição desta estrutura (ver figura 8) engloba diversos campos e um construtor, encarregue de assegurar que quando é criado um TCB este se encontra livre, não representando nenhuma tarefa.

Os campos que constam na estrutura são os seguintes:

- m_id - identificador atribuído ao TCB;
- m_state - estado da tarefa;
- m_period - período da tarefa;
- m_deadline - tempo limite de execução da tarefa;
- m_firstActivation - instante da primeira activação relativamente ao arranque da tarefa;
- m_activation - contador decrescente usado no escalonamento das tarefas. Possui o número de unidades temporais restantes para a (re)activação da tarefa;
- m_ponctuality - contador decrescente usado no escalonamento das tarefas. Armazena o número de unidades de tempo que a tarefa possui para terminar antes de ser atingida a sua *deadline*;
- m_bestCaseExecTime - número mínimo de unidades de tempo que a tarefa gastará com uma execução. Este parâmetro não está, normalmente, presente nos sistemas reais e é apenas utilizado na geração aleatória do tempo de execução da tarefa;
- m_worstCaseExecTime - número máximo de unidades de tempo que a tarefa gastará com uma execução. É utilizado na geração aleatória do tempo de execução da tarefa e também no escalonamento usando a política LSF;
- m_execCyclesCounter - contador decrescente que representa o número de unidades de tempo que faltam correr à tarefa nesta activação;
- m_cyclesExecuted - número de unidades de tempo que a tarefa já executou na última activação. É usado no algoritmo de escalonamento LSF para determinar o tempo livre da tarefa e assim a sua prioridade;
- m_sleepingCounter - contador decrescente que armazena o número de unidades de tempo restantes para a tarefa acordar. É utilizado para tarefas no estado *Sleeping*.

C. Módulos

Tal como já foi referido, a implementação do escalonador contempla diversos módulos. As subsecções seguintes descrevem sucintamente cada um dos módulos implementados.

```
typedef struct sc_TCB
{
   unsigned int m_id;

   unsigned int m_state;

   unsigned int m_period;
   unsigned int m_deadline;
   unsigned int m_firstActivation;

   int m_activation;
   int m_ponctuality;

   int m_bestCaseExecTime;
   int m_worstCaseExecTime;

   int m_execCyclesCounter;
   int m_cyclesExecuted;

   int m_sleepingCounter;

   sc_TCB()
   {
      m_state = FREE;
   }
}SC_TCB;
```

Figure 8 - Definição da estrutura sc_TCB.

C.1 O Módulo MClkDivider

O módulo MClkDivider encarrega-se da criação do relógio interno do escalonador. Para isso tem em conta um parâmetro de configuração do sistema, designado por factor de divisão, que vai especificar quantas vezes mais lento é o relógio interno do escalonador relativamente ao relógio do sistema. Este módulo compreende as seguintes entradas:

- Clk - relógio de simulação;
- Clk_Count - tempo absoluto de simulação. Número de ciclos de relógio passados desde o início da simulação;
- Factor - factor de divisão para criação do relógio interno do escalonador a partir do relógio do sistema;
- Rst - botão de reset.

O módulo produz ainda as seguintes saídas:

- Tick - relógio interno do escalonador;
- Tick_Count - tempo absoluto do escalonador. Número de ciclos de relógio do escalonador passados desde o início da simulação.

Este módulo está representado na figura 9.

C.2 O Módulo MCounter

O módulo MCounter encarrega-se apenas do incremento dum contador que representa o número de ciclos do relógio do sistema já ultrapassados desde o início da simulação.

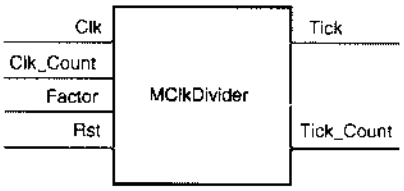


Figure 9 - Módulo `MClkDivider`.

Figure 10 - Módulo `MCounter`.

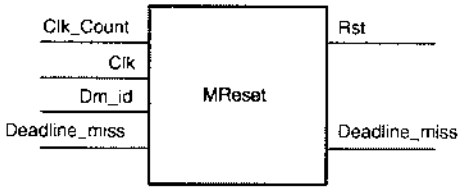


Figure 11 - Módulo `MReset`.

Este módulo compreende uma entrada:

- `Clk` - relógio de simulação.

O módulo produz ainda uma saída:

- `Clk_Count` - tempo absoluto de simulação. Número de ciclos de relógio passados desde o início da simulação.

Este módulo está representado na figura 10.

C.3 O Módulo `MReset`

O módulo `MReset` tem por função a manipulação do *reset* do escalonador. Principalmente, encarrega-se da leitura do ficheiro de configuração `reset.conf` e activação ou desactivação a tempo do *reset* como especificado neste.

Também aceita como entrada uma *flag* que sinaliza uma falha duma restrição temporal, situação na qual faz um *reset* do escalonador. Este módulo compreende as seguintes entradas:

- `Clk` - relógio de simulação;
- `Clk_Count` - tempo absoluto de simulação. Número de ciclos de relógio passados desde o início da simulação;
- `Deadline_miss` - *flag* que sinaliza uma falha duma restrição temporal;
- `Dm_id` - identificador da tarefa que violou a sua restrição temporal.

O módulo produz ainda as seguintes saídas:

- `Deadline_miss` - *flag* que sinaliza uma falha duma restrição temporal;
- `Rst` - sinal de *reset*.

Este módulo está representado na figura 11.

C.4 O Módulo `MScheduler`

O módulo `MScheduler` é o centro do escalonador implementado. É este módulo que se encarrega do escalonamento das tarefas propriamente dito, aplicando o algoritmo de escalonamento escolhido ao conjunto de tarefas presente no sistema. É também neste módulo que se processam as transições de estado das tarefas tal como os comandos relacionados com estas. Este módulo compreende as seguintes entradas:

- `Tick` - relógio interno do escalonador;
- `Tick_Count` - tempo absoluto do escalonador. Número de ciclos de relógio do escalonador passados desde o início da simulação;
- `Algorithm` - algoritmo a ser utilizado pelo escalonador. É determinado através de um parâmetro passado na linha de comandos;
- `Rst` - sinal de `reset`;
- `Activate` - sinal que indica que um novo comando deve ser executado. Os seguintes sinais caracterizam o comando:
  - `Cmd` - comando a ser executado;
  - `T_id` - identificador da tarefa à qual o comando se refere;
  - `Period` - período da tarefa;
  - `Deadline` - *deadline* da tarefa;
  - `FirstActivation` - tempo da primeira activação da tarefa;
  - `BestCaseExecTime` - tempo mínimo de execução da tarefa;
  - `WorstCaseExecTime` - tempo máximo de execução da tarefa.

O módulo produz ainda as seguintes saídas:

- `Handle` - identificador da tarefa agendada para execução em cada instante;
- `State[]` - vector com os estados actuais de cada tarefa. Apenas foi incluído para poder facultar esta informação ao utilizador;
- `Deadline_miss` - *flag* que sinaliza uma falha duma restrição temporal;
- `Dm_id` - identificador da tarefa que violou a sua restrição temporal. Apenas foi incluído para poder facultar esta informação ao utilizador.

Este módulo está representado na figura 12.

C.5 O Módulo `MCmdRouter`

O módulo `MCmdRouter` encarrega-se da leitura do ficheiro de configuração `scheduler.conf` e da passagem no instante de execução dum comando dos parâmetros deste ao módulo `MScheduler`. Este módulo compreende as seguintes entradas:

- `Clk` - relógio de simulação;
- `Clk_Count` - tempo absoluto de simulação. Número de ciclos de relógio passados desde o início da simulação;
- `Tick_Count` - tempo absoluto do escalonador. Número de ciclos de relógio do escalonador passados desde o início da simulação;

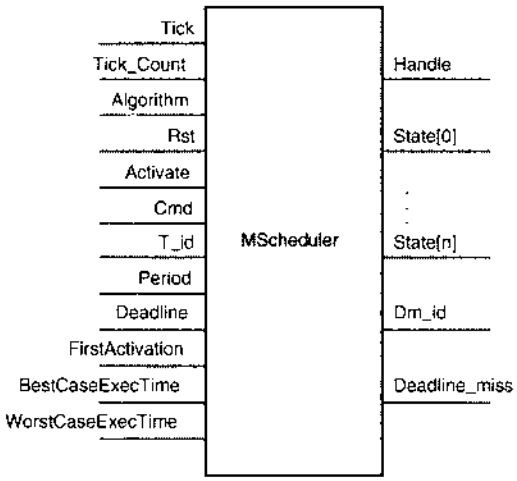


Figure 12 - Módulo `MScheduler`.

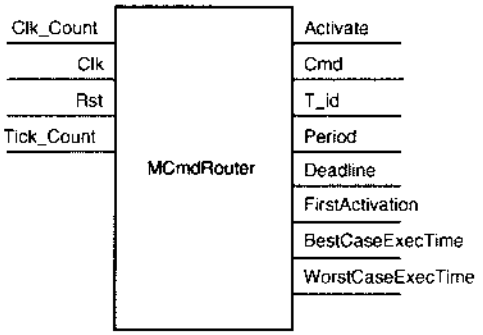


Figure 13 - Módulo `MCmdRouter`.

- `Rst` - sinal de `reset`.

O módulo produz ainda as seguintes saídas:

- `Activate` - sinal que indica que um novo comando deve ser executado. Os seguintes sinais caracterizam o comando:
  - Cmd - comando a ser executado;
  - `T_id` - identificador da tarefa à qual o comando se refere;
  - `Period` - período da tarefa;
  - `Deadline` - *deadline* da tarefa;
  - `FirstActivation` - tempo da primeira activação da tarefa;
  - `BestCaseExecTime` - tempo mínimo de execução da tarefa;
  - `WorstCaseExecTime` - tempo máximo de execução da tarefa.

Este módulo está representado na figura 13.

D. Configuração

O escalonador implementado é passível de ser configurado. Nas secções seguintes vão ser apresentadas as opções de configuração e modo de utilização destas.

D.1 O Ficheiro `scheduler.conf`

O ficheiro `scheduler.conf` é o meio através do qual se passam os comandos relativos a tarefas ao escalonador. O ficheiro pode ter no início texto que não será avaliado, devendo-se introduzir um ';' no fim a partir do qual deverá estar um comando por linha, eventualmente com linhas em branco a separá-los. Cada comando deverá ser precedido pelo instante no qual se pretende a sua execução, em ciclos de relógio do escalonador. Os comandos disponíveis são os seguintes:

- `create` - Sintaxe: `<t> create <t_id P D PA min max>`
  Comando que permite a criação duma tarefa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador pretendido para a tarefa a criar;
  - `P` - período a atribuir à tarefa;
  - `D` - *deadline* para a tarefa;
  - `PA` - instante da primeira activação da tarefa;
  - `min` - número mínimo de unidades de tempo que a tarefa gastará com uma execução;
  - `max` - número máximo de unidades de tempo que a tarefa gastará com uma execução.
- `destroy` - Sintaxe: `<t> destroy <t_id>`
  Comando que permite eliminar uma tarefa do sistema. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a destruir.
- `change` - Sintaxe: `<t> change <t_id P D min max>`
  Comando utilizado para alterar as características duma tarefa já presente no sistema. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a alterar;
  - `P` - período a atribuir à tarefa;
  - `D` - *deadline* para a tarefa;
  - `min` - número mínimo de unidades de tempo que a tarefa gastará com uma execução;
  - `max` - número máximo de unidades de tempo que a tarefa gastará com uma execução.
- `start` - Sintaxe: `<t> start <t_id>`
  Comando para arrancar uma tarefa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a arrancar.
- `stop` - Sintaxe: `<t> stop <t_id>`
  Comando que permite parar uma tarefa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a parar.
- `sleep` - Sintaxe: `<t> sleep <t_id C>`
  Comando utilizado para adormecer uma tarefa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a adormecer;
  - `C` - número de ciclos ao longo dos quais a tarefa estará adormecida.

- `suspend` - Sintaxe: `<t> suspend <t_id>`
  Comando que permite suspender uma tarefa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a suspender.
- `resume` - Sintaxe: `<t> resume <t_id>`
  Comando utilizado para retomar uma tarefa suspensa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a retomar.

D.2 O Ficheiro `reset.conf`

O ficheiro `reset.conf` disponibiliza a interacção com o botão de *reset* do escalonador. Tal como o ficheiro `scheduler.conf`, poderá ter um cabeçalho seguido de um ';', após o qual deverá estar um número por linha, eventualmente com linhas em branco entre eles, que representarão os instantes nos quais o *reset* deverá mudar de valor, começando activo, em ciclos do relógio de simulação.

D.3 Invocação

A linha de comandos permite ainda a configuração de mais alguns parâmetros. Na sequência do nome do programa poder-se-á especificar o algoritmo de escalonamento que se pretende utilizar (RM, EDF ou LSF) seguido do factor de divisão para a criação do relógio interno do escalonador e por fim o nome para o ficheiro VCD (sem a extensão `.vcd`). A execução escrevendo `default` a seguir ao nome do programa decorre com os parâmetros com os valores por defeito, sendo o algoritmo de escalonamento o EDF, o factor de divisão 4 e gerado um ficheiro VCD com o nome `Scheduler.vcd`.

V. EXEMPLOS DE APLICAÇÃO

Nesta secção irá ser descrita a sequência típica de passos para a execução de uma simulação de um escalonamento, e visualização dos resultados.

Primeiro é necessário editar o ficheiro `scheduler.conf` introduzindo os comandos pretendidos relacionados com o conjunto de tarefas que se pretende simular. Na figura 14 pode-se ver a configuração do ficheiro `scheduler.conf` usada para este exemplo.

O segundo passo consiste na edição do ficheiro `reset.conf` segundo as necessidades da simulação. Para este exemplo introduziu-se apenas um número neste ficheiro, que representou o instante a partir do qual o *reset* deixou de estar activo.

Feito isto, pode-se correr a simulação. Neste exemplo foi usado o algoritmo de escalonamento EDF, um factor de divisão do relógio para criação do tick do sistema de 4 e foi indicado que o ficheiro VCD a ser criado deveria ser nomeado `exemplo.vcd`, como ilustra o comando que foi executado:

```
scheduler.x EDF 4 exemploEDF
```

De seguida surgem no ecrã mensagens relativas ao escalonamento efectuado (figura 15).

```
0 create 1 100 100 0 4 5
0 create 2 27 27 0 7 8
0 create 3 48 48 0 4 5
0 create 4 59 59 0 9 9
0 create 5 14 14 0 5 5
100 sleep 3 20
200 suspend 2
500 stop 3
600 stop 4
700 change 4 50 50 4 9
900 start 4
950 resume 2
1000 destroy 3
1300 destroy 4
1500 change 1 120 120 2 5
```

Figure 14 - Ficheiro `scheduler.conf` usado no exemplo.

```
A tarefa nº 2 foi agendada para execução.
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME
A tarefa nº 5 foi agendada para execução.
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME

Tarefa nº 1 alterada.

A tarefa nº 4 foi agendada para execução.
A tarefa nº 2 foi agendada para execução.
A tarefa nº 5 foi agendada para execução.
A tarefa nº 2 foi agendada para execução.
A tarefa nº 4 foi agendada para execução.
A tarefa nº 5 foi agendada para execução.
A tarefa nº 1 foi agendada para execução.
```

Figure 15 - *Output* no ecrã gerado pela simulação do escalonamento.

Estes resultados servem para dar uma ideia de como a simulação decorreu. No entanto existe a possibilidade de analisar o que decorreu ao longo da simulação com mais detalhe. O ficheiro VCD produzido contém muito mais informação e pode ser consultado executando o seguinte comando:

```
gtkwave exemploEDF.vcd
```

Aí é possível encontrar a evolução ao longo do tempo de simulação de diversos sinais (figura 16). São estes:

- relógio e contador deste;
- *tick* do sistema e contador deste;
- *reset*;
- identificador da tarefa em execução;
- *flag* que assinala uma violação de uma *deadline*;
- identificador da tarefa que violou a sua *deadline*;
- estados das tarefas presentes no sistema.

A. *Análise dos resultados*

Para além do exemplo descrito acima foram corridos outros dois exemplos, utilizando os mesmo ficheiros de configuração, desta vez indicando ao escalonador para

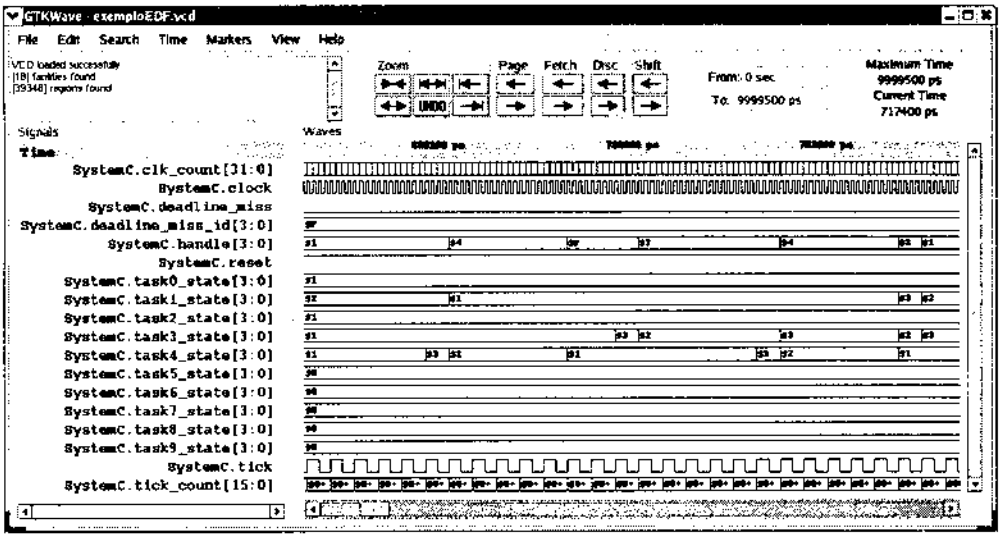


Figure 16 - Visualização do ficheiro VCD gerado na simulação utilizando a política EDF.

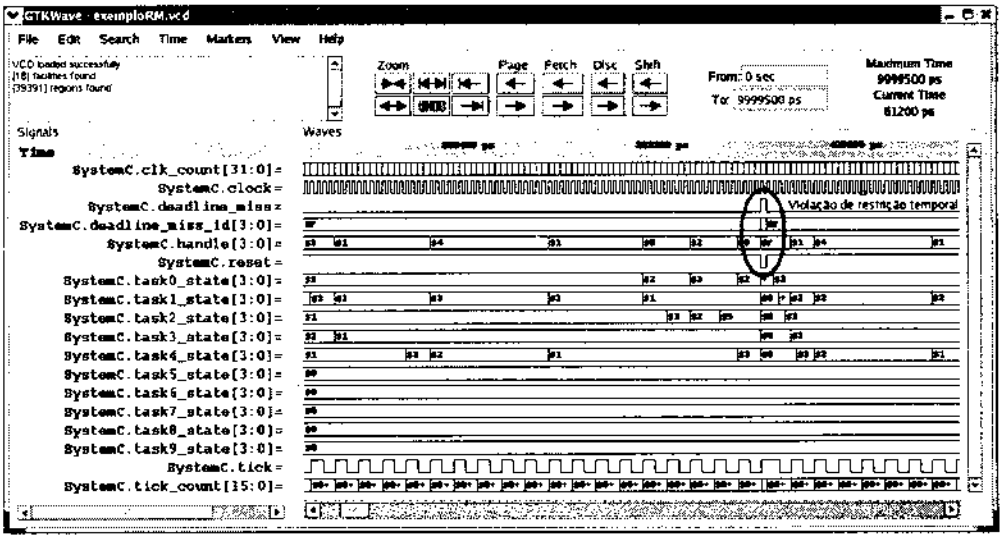


Figure 17 - Visualização do ficheiro VCD gerado na simulação utilizando a política RM.

agendar as tarefas segundo as outras duas políticas suportadas: RM e LSF. As figuras 17 e 18 mostram uma parte do ficheiro VCD criado em cada um dos casos.

O conjunto de tarefas utilizado nos exemplos poderia ser executado num processador com uma taxa de ocupação entre 89% e 96%. No entanto, utilizando a política de escalonamento RM ocorreram violações de restricções temporais. Isto acontece devido ao facto desta política, que utiliza prioridades fixas, não conseguir garantir que atinja uma taxa de ocupação do processador de 100% como as outras duas políticas implementadas, que se basciam em prioridades dinâmicas.

Comparando as políticas de prioridades dinâmicas pode-se ver que ambas conseguiram escalonar o conjunto de tarefas. Apenas de registar que usando a política LSF se verificam mais preempções do que com a EDF, sendo que este facto poderá ser relevante num sistema real uma vez que as comutações de contexto que acontecem nessas alturas levam algum tempo a serem efectuadas.

VI. CONCLUSÃO E TRABALHO FUTURO

A implementação do escalonador descrito neste artigo permite a simulação do escalonamento de tarefas concorrentes processado por um módulo de *hardware*. Neste momento a sua funcionalidade consiste apenas na simulação do escalonamento de conjuntos de tarefas *hard real-time*, que se podem configurar, usando um dos algoritmos de escalonamento incluídos, e visualização do resultado deste escalonamento quer através do output no ecrã mas principalmente através do ficheiro VCD gerado, que permite a visualização e análise de mais informação. No entanto este trabalho não se apresenta na sua versão final, sendo possível desenvolvê-lo na direcção de algumas ideias que já surgiram para trabalho futuro:

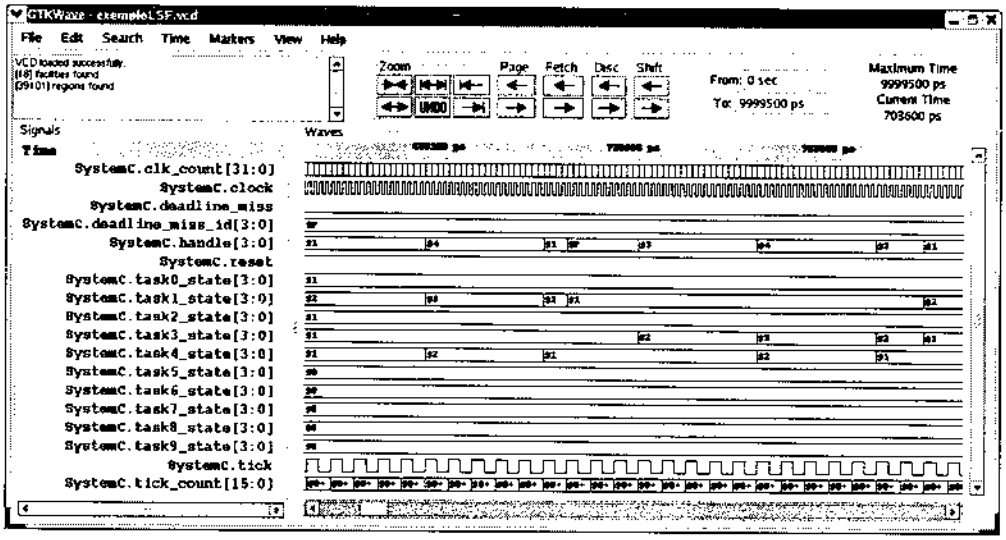


Figure 18 - Visualização do ficheiro VCD gerado na simulação utilizando a política LSF.

- Suporte para diferentes tipos de tarefas;
- Implementação de análise de escalonabilidade;
- Refinamento do código para possível implementação futura em *hardware*.

REFERENCES

[1] Arnaldo Oliveira e Luís Almeida, "Ensaio sobre os Sistemas de Tempo Real", Janeiro de 2004.

[2] Arnaldo Oliveira, Luís Almeida, e Valery Sklyarov, "OReK - Um Executivo de Tempo Real Orientado por Objectos", Janeiro de 2004.

[3] "GTKWave".
URL: *http://www.cs.man.ac.uk/apt/tools/gtkwave/*

[4] "SystemC".
URL: *http://www.systemc.org*

[5] Giorgio Buttazo, "Hard Real-Time Computing Systems: Predictadle Scheduling Algorithms and Applications", 1997.

StackFences: a run-time approach for detecting stack overflows

André Zúquete

Abstract – **This article describes StackFences, a run-time technique for detecting overflows in local variables in C programs. This technique is different from all others developed so far because it tries to detect explicit overflow occurrences, instead of detecting if a particular stack value, namely a return address, was corrupted because of a stack overflow. Thus, StackFences is useful not only for detecting intrusion attempts but also for checking the run-time robustness of applications. We also conceived different policies for deploying StackFences, allowing a proper balancing between detection accuracy and performance. For testing StackFences we developed a prototype for Linux systems using TCC (Tiny C Compiler). C modules compiled with StackFences are fully compatible with C modules compiled differently and standard libraries. Effectiveness tests confirmed that all overflows in local variables are detected before causing any severe damage. Performance tests ran with several tools and parameters showed an acceptable performance degradation.**

Resumo – **Este artigo descreve o StackFences, uma técnica para detectar em tempo de execução transbordamentos de memória em variáveis locais de programas em C. Esta técnica é diferente das demais desenvolvidas para lidar com este problema porque detecta directamente os transbordamentos de memória, em vez de detectar se valor específicos na pilha, como endereços de retorno, foram corrompidos devido a um transbordamento de memória. Assim, o StackFences é útil não só para detectar tentativas de intrusão mas também para monitorizar a correcção de execução das aplicações. Foram também concebidas duas políticas de exploração do StackFences que permitem um equilíbrio apropriado entre correcção e desempenho. Para testar o StackFences desenvolveu-se um protótipo para sistemas Linux usando o TCC (Tiny C Compiler). Os módulos C compilados com o StackFences são totalmente compatíveis com módulos C compilados diferentemente ou com bibliotecas padrão. Os testes de eficácia confirmaram que todos os transbordamentos em variáveis locais são detectados antes de causar um estrago significativo. Os testes de desempenho executados com diversas ferramentas e parâmetros revelaram uma degradação de desempenho aceitável.**

Keywords – **Buffer overflows, run-time detection, run-time correctness assessment, damage containment, dependability**

Palavras chave – **Transbordamentos de memória, detecção em tempo de execução, verificação de correcção em tempo de execução, minimização de estragos, confiança operacional**

This article is an extended version of another one published in the 1st International Conference on E-business and Telecommunication Networks (ICETE 2004).

I. INTRODUCTION

The exploitation of overflow vulnerabilities has been one of the most popular forms of computer attacks during the last 15 years. According to the ICAT Metabase vulnerability statistics[1], about 20% of the vulnerabilities published in the last 4 years are related with buffer overflows. Such vulnerabilities existed in compiled C or C++ code used for different purposes: operating system (OS) kernel; server and client applications. For some people the problem will continue to exist as long C is used, and can only be minimized or eliminated by using other languages instead of C [1]. However, C is still widely used, and probably will continue to be, because it generates fast compiled code and there is a large repository of legacy code written in C. Therefore, there is a real need for techniques and tools that help to minimize the number and the risk of overflows in old or new C programs.

In this paper we address only problems caused by *buffer overflows*, i.e., with write operations outside the proper memory area, invading neighboring memory areas. But there are other kinds of overflows, such as overflows in the values of variables. These overflows do not directly affect neighboring memory areas but do modify the execution flow of the application (e.g. the SSH daemon integer-overflow vulnerabilities CVE-2001-0144 and CVE-2002-0639 reported in the ICAT Metabase).

StackFences is a solution for detecting buffer overflows affecting local variables, i.e., variables allocated in stack frames. The key idea that we explored is an extension to the *canary mechanism*, introduced in StackGuard [2], complemented with the *XOR canary mechanism* introduced also in StackGuard. The original canary mechanisms were deployed for protecting only return values stored in the stack. We extended the protection scope and used canaries for controlling *all stack areas susceptible to overflow*. This way, we hope to detect each and every stack overflow, either affecting highly sensible values, like return addresses, or not. We also conceived different policies for balancing detection accuracy and performance. Namely, run-time validations can be performed in two different ways: (i) one more detailed, allowing a more accurate and timely detection of overflows, more suitable for development scenarios, and (ii) one lazier, checking only when absolutely necessary, more suitable for production environments.

For testing StackFences we developed a prototype for Linux systems using TCC (Tiny C Compiler), a very simple, though complete C compiler. The code generated by our modified version of TCC manages boundary canaries near local variables susceptible to overflow. C modules

[1] `http://icat.nist.gov`

compiled with StackFences are fully compatible with the standard C libraries and with modules compiled with other compilers or compilation options. The modifications introduced by StackFences affect only the code inside each function, and the external visibility of such modifications is required only for some StackFences' checking procedures.

This paper is structured as follows. Section II overviews the issues raised by stack buffer overflows. Section III presents the related work concerning run-time detection/prevention of stack buffer overflows. Section IV presents our contribution to detect stack buffer overflows. Section V presents some implementation details regarding the modified compiler and some extra modules with auxiliary variables and functions. Sections VI and VII evaluate the effectiveness of StackFences and the overheads introduced in the execution of microbenchmarks and real applications. Finally, Section VIII presents the conclusions and future work.

II. OVERVIEW

There are mainly two reasons for the buffer overflow problems in C programs. First, the language does not check for any boundaries around variables and allows programmers to manage memory areas at will, without any run-time control, using pointers, type casts and pointer arithmetic. Second, many standard C library functions are intrinsically unsafe concerning buffer overflows (e.g. the infamous `gets()` and several functions for manipulating strings [3]).

Buffer overflows are a problem because they can be used to modify data that controls the execution of a victim process or OS kernel. The exploitation of a buffer overflow vulnerability can expose a victim application to two different risks:

Denial of Service (DoS): an attacker may interfere randomly with the application's execution flow, eventually making it fail after some illegal operation. The damage caused by such an attack is difficult to assert, both for the attacker and the victim, because the attacked application may fail without any control.

Penetration: An attacker may take control of the application's execution flow by performing a crafty buffer overflow. In principle, the attacker knows very well the damage produced by the attack, or produced by conducting an intrusion thereafter.

Ideally, one would like to avoid both risks, but that may be difficult or even impossible to achieve with current hardware architectures and existing software. Comparing both risks, the risk of penetration is greater than the risk of DoS. Therefore, it seems useful to avoid penetration risks by transforming them into DoS risks. We followed this reasoning in the design of StackFences.

A. Detection/prevention of buffer overflows

Buffer overflows may be detected using static analysis, i.e., before actually compiling and deploying the code (e.g. [4], [5]). This approach, however, is not likely to be complete, even though it may be used to spot well-known vulnerabilities. Another possibility is to tackle buffer overflows dynamically, or at run-time, during the execution of the vulnerable applications. In this case, the vulnerable application, or the execution environment, must be instrumented to prevent or detect overflows or problems caused by overflows. This was the approach that we followed in StackFences.

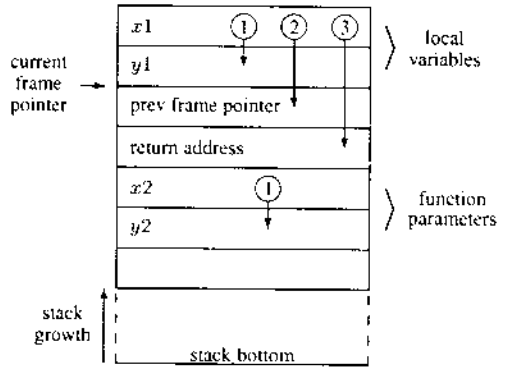


Figure 1 - Possible overflows of the stack memory reserved for a C variable x (either $x1$ or $x2$). Overruns may affect (1) neighboring variables $y1$ or $y2$, (2) a saved frame pointer, or (3) a return address.

Dealing with buffer overflows at run-time implies either prevention or detection. *Prevention* attempts to conceal overflow vulnerabilities or to make their occurrence partially or totally harmless. If the prevention is effective a vulnerable application may continue to execute normally even when under attack. Though prevention does not help finding problems out, it is highly desirable for avoiding both DoS and penetration risks. But total prevention is difficult to achieve and partial prevention should be avoided because it gives a false sense of security.

Detection attempts to detect the occurrence of buffer overflows or the occurrence of abnormal facts that may be consequence of buffer overflows. Detection only mitigates the problem, since the usual reaction is to raise some sort of alert and immediately terminate the affected application or OS kernel, eventually leading to a DoS situation. Therefore, detection helps to assess correctness on the execution of applications or OS kernels with overflow vulnerabilities, which is important for improving damage containment and reducing penetration risks. StackFences is a detection solution.

B. Anatomy of a stack overflow

The overflow of the stack memory reserved for a C variable x can corrupt the execution flow of an application in many ways. Assuming only overflows across memory with growing addresses and a x86 processor, we have at least the following scenarios (cf. Fig. 1):

1. By setting the value of a neighboring variable y. If y is a pointer to a function, one can reassign it to a different function. If y is a pointer to a memory area, one can cause a (probably fixed) memory modification somewhere else. If y is a `longjmp` buffer, one can control a future jump using it. If y is an integer, float or struct/union variable one can modify the flow of the program if the value of y is relevant for making flow

decisions or producing useful results.

2. By setting the value of a saved frame pointer of a previous stack frame to another address. When the modified frame pointer is recovered, the application will use, in that stack frame, different local variables and function arguments, since the frame pointer is the base address for addressing them.
3. By setting the return address for the calling function. When the function returns it will continue to execute at an address chosen by the attacker.

These attacks can be complemented by inserting auxiliary data, like microprocessor instructions or function parameters, into input buffers, overflowed or not. The auxiliary data can be used later for a penetration bootstrap if the attacker succeeds in using it.

Most overflow attacks are *stack-smashing attacks* [6], i.e., attacks of the third type referred above, that overwrite return addresses and jump into bootstrap penetration code. The famous Internet Worm of 1988 did it [7], as well as many other attacks thereafter. But the two other types are also risky. For instance, M. Rolf presents in [8] a difficult, though possible, exploit using a tampered frame pointer. Furthermore, the use of *canaries* and other protection mechanisms for detecting the modification of return addresses (cf. Section III) pushes attackers to work around them in order to exploit stack buffer overflows without triggering those protection mechanisms. Thus, by detecting as much as possible all stack overflows we should be able to defeat more overflow attacks, both already known or still unknown.

C. Protection paradigms

J. Wilander and M. Kambar pointed out, in [9], that a general weakness of the solutions presented so far for run-time detection or prevention is that they protect *known attack targets*, mainly return addresses in the stack, instead of protecting *all targets*. From a pragmatic point of view, we can understand why that happens: stack-smashing attacks are the simplest and most popular ones. But for dealing with future and more sophisticated attacks exploiting buffer overflows we need to change the protection paradigm.

In [9] it is also mentioned that "*changing the flow of control occurs by altering a code pointer*". This is true but not complete. Namely, we can change the flow of control by overflowing a simple integer variable. It may be argued that it should be difficult to succeed in a penetration attempt using such a simple overflow vulnerability, but the fact of not knowing any instance of such an attack doesn't mean that it could not appear in the future. And, in any case, such apparently useless overflows may lead to DoS situations, either by crashing the victim application or, even worse, by leading it to an abnormal behavior. Thus, for ensuring the run-time correctness of an application we should control each and every variable susceptible to be affected by an overflow; we tried to do so for local variables with StackFences.

III. RELATED WORK

In this section we describe the approach followed by several run-time overflow detection or prevention techniques. We do not address any static detection solutions.

Many run-time protection techniques were developed to protect the most common target of overflow attacks: the return pointer. StackGuard [2] uses *canaries*, which are specific values that are placed between the local variables and the return address in the same stack frame. The canary is installed in the function prologue and checked in the epilogue. If its value changed in the meanwhile then there was an overflow and the process is halted; otherwise the function returns normally. Two types of canaries where proposed for StackGuard — *terminator canary* for preventing overflows with character strings and *random canary* computed in run-time — but only the first is actually used [9].

`Propolice` [10] enhances the basic protection of StackGuard by rearranging local variables. The assumptions of `propolice` are that (i) only character arrays are vulnerable to overflows and (ii) function arguments do not contain character arrays. Thus, vulnerable local variables — character arrays or structures with character arrays — are packed together in a *vulnerable location* next to the canary (*guard*). All other variables are placed after the vulnerable location, above in the stack. This way, overflows can occur inside vulnerable locations but cannot affect non-vulnerable variables, because they are in lower stack addrtess, neither the return address because it is protected by the canary. However, `propolice` assumptions are not complete, because there are other kinds of vulnerable variables besides character arrays, and ignores overflows affecting only variables in a vulnerable area.

Another way of protecting return addresses is by hiding them with a *XOR canary*, or *cookie*. The XOR canary is XORed with the return address in a function's prologue and again in the epilogue. Attackers not knowing the value of the XOR canary are unable to modify return addresses in a useful manner. StackGuard was the first to use XOR canaries to frustrate attacks (to return addresses) circumventing the basic canary mechanism[2]. StackGhost [11] is kernel-level solution for Sparc architectures that protects return addresses this way. StackGhost uses either per-kernel or per-process XOR canaries, but the first is too weak for competent hackers. Overflow attacks affecting return values produce wrong return addresses. These can be automatically detected in 75% of the cases because Sparc instructions must be aligned on a 4-byte boundary; on the other 25% the program will run uncontrolled.

StackGuard's MemGuard [2] protection makes return addresses in the stack read-only during the normal execution of functions. This way, any attempts to overwrite them raise a memory exception. However, the performance penalty of this approach is huge.

Another way of protecting return addresses is to keep a separate copy of them in a *return-address stack*. Return addresses are stored in and fetched from the return-address stack in the function's prologue and epilogue, respectively.

[2]This mechanism was introduced in version 1.12.

Vendicator's StackShield[3] `Global Ret Stack` protection and Secure Return Address Stack [12] use only the copied values, thus preventing attacks affecting return addresses stored in the normal stack. These solution are very good in keeping return addresses correct but fail completely in protecting any other stack values from overflows.

The Return Address Defender [13] also uses a return-address stack but provides only detection because return addresses on the Return Address Repository are compared with the ones in the ordinary stack before being used and the process is halted if they are different. StackShield's `Ret Range Check` is similar but stores a copy of the current return address in a global variable. All these detection mechanisms are useless against overflow attacks overwriting the return address with its exact value, which is not difficult to guess for a competent hacker.

Libverify [14] is another protection mechanism that uses a return-address stack but, unlike the former, it can be transparently applied to existing binary code by means of a dynamic library. The drawback of Libverify is that all protected code must be copied into the heap to overwrite instructions in the prologue and epilogue of all functions. This means that processes are unable to share the code they effectively run in main memory and absolute jumps within the text area must be handled with traps.

Yet another way of protecting return addresses is to check memory limits within certain critical functions. For instance, Libsafe [14] is a dynamic library that replaces unsafe functions of the standard C library that are typically used for performing buffer overflows. All Libsafe functions compute the upper bounds of destination's buffers before actually transferring data into memory. The upper bounds are defined by the location of return addresses. But the protection is limited and misses unsafe functions compiled inline within existing applications or libraries.

PointGuard [15] is an extension of the original XOR canary mechanism for protecting pointer variables. Pointers are stored encrypted (XORed with the XOR canary) and are decrypted when loaded into CPU registers. However, this approach raises problems when integrating mix-mode code (some PointGuard, some not).

Practically all protection mechanisms look only at the effects of overflows within the current stack frame — exceptions are PointGuard and Libsafe. The solutions like StackGuard, that use a boundary canary, can also protect frame pointers stored next to return addresses; it is only necessary to place the canary between the frame pointer and the local variables. StackGuard 3 [16] and Libsafe protect frame pointers. And `propolice`, *under their assumptions*, further protects all variables that are not character arrays or structures with character arrays.

Most of the protection mechanisms described are added at compile time; exceptions are Libsafe, Libverify and StackGhost. StackFences is also added at compile time.

With StackFences we tried to further improve the detection of stack overflows. Namely, we tried to (i) detect overflows within all existing stack frames, not only the current one; and to (ii) detect overflows from all variables that could be overflowed. In this particular case we extended `propolice`'s and PointGuard's notions of vulnerable areas and we do not ignore overflows within variables belonging to a vulnerable area, as `propolice` does.

[3] `http://www.angelfire.com/sk/stackshield`

IV. OUR CONTRIBUTION: STACKFENCES

To detect overflows in stack variables we decided in favor of managing StackGuard-like boundary canaries between them. The management of canaries, which involves their allocation, setup and checking, depends on the kind of stack variables we are dealing with: local variables or function parameters. In this paper we handle only the management of boundary canaries for local variables. But a similar approach for function parameters is already scheduled for future work.

To protect the value of canaries we use a *XOR canary*. Like in StackGhost and PointGuard, this is a per-process random 32-bit value that is used to hide important values. It can be compared to a 32-bit long keystream that is used to encrypt and decrypt sensitive values. The canary is stored in a publicly known variable (`cXor` in this text) and should not be modified after being setup. Adaptive attacks trying to guess the correct value of a process' canary are infeasible, in theory, because attacked processes should terminate after an attack with an unsuitable, tentative canary value. We also assume, like for StackGhost and PointGuard, that it is impossible for an attacker to get dumps of stack areas containing any boundary canaries XORed with the XOR canary.

As for StackGuard, we assumed that overflows are caused by writing continuous amounts of data from a memory address pointing to any byte of the correct memory area. But, unlike StackGuard, we can also tackle certain overflows caused from wrong pointers, i.e., pointers referring to (semantically) wrong memory areas. For instance, such pointers appear when wrong indexes, both positive or negative, are used to access arrays.

A. Overview

Boundary canaries are located near local variables, on higher addresses. The questions now are (i) which variables we want to, or should, bound with canaries, (ii) how do we find the canaries and check their value; and (iii) when do we want to, or should, check the value of canaries.

A.1 Variables to bound with canaries

Which variables should be bounded with canaries? Following a high security policy, we should bound all stack variables. But this is a sort of brute force approach that has considerable impact in the performance of modified applications.

A more relaxed policy is to setup boundary canaries after all *potentially vulnerable variables*. Such variables are the ones *for which a pointer is taken and used in the current function or other function called upon it*. Note that with this definition all local arrays are frequently vulnerable (unless they are not used or used only with constant indexes) but a pointer is not vulnerable until getting its address (see Figure 2). This policy is more efficiency then the previous one and should tackle most stack overflow problems (since

they usually derive from deficient uses of legitimate stack addresses). Consequently, this was the policy we chose for installing our boundary canaries.

```
f ()
{
  int A, B;
  int * C;
  int D[10], E[10];
  ...
  C = &A;          // A becomes vulnerable
  f ( &B, C, D ); // B and D become vulnerable,
                   // but not C
  g ( &C );        // C becomes vulnerable
  E[A] = 3;        // E becomes vulnerable ...
}
```

Figure 2 - C function with 5 local variables, all of them potentially vulnerable to buffer overflows.

Additionally, all vulnerable variables are packed together to improve the detection mechanism. In fact, those variables and the canaries between them and at the end of the pack form a sort of "mined area", where most overflows, caused by dangling pointers and using either positive or negative offsets, are highly unlikely to occur without being noticed. The pack can either be placed close to the saved frame pointer or as far as possible; in principle, such decision should have no impact on the security provided by StackFences.

Because the size of C structures cannot be altered, canaries cannot be added between members of local `struct` variables. Therefore, overflows may still exist strictly inside structures, but not affecting external memory areas.

A.2 Looking for and checking canaries

How can we look for all or part of the canaries and how can we check the correctness of their value? One possibility is to generate and use per-function checking code, knowing the exact location of canaries and their values in the current stack frame. However, this approach would complicate a simple task of checking all existing canaries. Another possibility is to define a more function-independent way of locating and checking canaries.

We followed the last approach because it appeared to be more flexible and simple to implement and test. Consequently, the set of all boundary canaries form a linked list, as shown in Fig. 3. The location of the head and tail canaries of the list are stored in publicly known variables (`cHead` and `cTail` in this text). The virtual address (of another canary) stored in each canary is protected using the XOR canary previously referred. This way an attacker causing the overflow of a stack variable cannot easily guess valid values for boundary canaries between the overflowed variable and the neighboring variables to be tampered.

The full list of canaries, or particular sublists, can easily be checked by dereferencing canaries, XORing the obtained value with the XOR canary and testing whether the result is a valid address. Testing the validity of an address is straightforward: (i) it must be higher than the previous one, because the list goes strictly from the top to the bottom of the stack, and (ii) it cannot be higher than a target canary

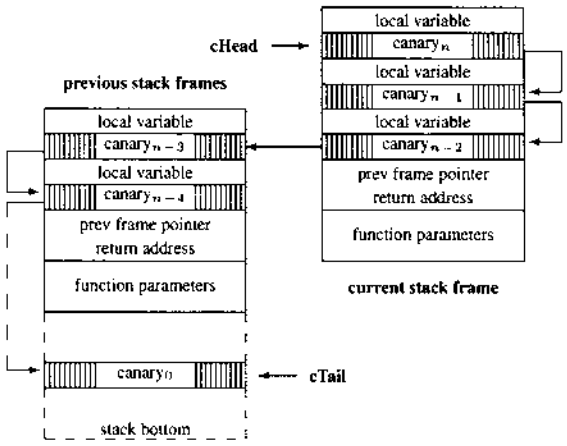


Figure 3 - Example of the list of canaries, starting in the current stack frame (`canary`$_n$) and until the first one inserted at the beginning of the process execution (`canary`$_0$). Variables `cHead` and `cTail` point the head and tail of the list. Shaded boxes represent canaries XORed with the process' XOR canary.

address that we want to reach. Any violation of these assertions is an overflow evidence.

A.3 Triggering canary checking

When should the application check the boundary canaries for looking for stack overflows? We think that there are at least two distinct situations that should trigger canary checking:

Before doing some operation related with the external perception of the application's behavior. In other words, the coherence of stack variables should be checked before each and every operation capable of reflecting a wrong behavior of the application to the outside world. Broadly, this means that checking should be done before any I/O attempt;

Just before the return of a function. In this case, we should check for overflows within the current stack frame, i.e., caused in local variables or parameters, that will disappear because the stack frame will be released. Note that a local overflow may have affected the past execution flow of the function, values returned by the function, variables modified by the function that are not bounded by overflow canaries (v.g. heap variables), or the frame pointer or the return address of the current stack frame.

In the first case, we should check the full list of canaries, because we are looking for any stack overflow. The full list of canaries, from the head `cHead` to the tail `cTail`, can be checked at any time during the execution of the program for finding out overflows affecting any of the canaries. The more times it is checked, the more timely we can find stack overflows, but with a significant impact on the performance of the application. Checking the full list of canaries is a straightforward iterative walk, starting in the canary pointed to by `cHead` and ending when the canary pointed to by `cTail` is reached. Since canaries along the list must have strict growing addresses and cannot be higher than `cTail`, any violation of these assertions is an overflow evidence.

In the second case, we should check only the list of canaries belonging to the local stack frame (`canary`$_n$ to `canary`$_{n-2}$ in Fig. 3) since we are looking for evidences of local stack overflows that are about to disappear. If no canaries exist in the current stack frame then no local checking is required. Checking a local list of canaries is also an iterative walk starting in `cHead` but ending in the head canary in the previous stack frames (`canary`$_{n-3}$ in Fig. 3). The location of this canary must be supplied by the compiler.

B. Management of the canary list

The canaries in each stack frame are similar to local variables. The difference is that (i) they are invisible to application code, being only known and managed by the compiler-generated code and (ii) they have a mandatory initial value that depends on the address of the next canary and the value of the XOR canary. Therefore, managing local canaries follows many of the usual procedures used for ordinary local variables.

The stack space for canaries around local variables can be allocated when the offsets of local variables (from the frame pointer) are defined by the compiler. This is a compile-time action that does not incur any run-time overhead. The setup of the canaries should occur after the normal C prologue[4].

The linked list is increased after the prologue of a function and decreased when the function returns. Increasing the list consists of adding all canaries next to local variables to the head of the list, referred to by `cHead`, and setting a new value for `cHead` with the address of the new top-most stack canary (the one with lowest address, `canary`$_n$ in Fig. 3). Decreasing the list consists simply of setting the value of `cHead` using the address of the first canary of the previous stack frames (the one with highest address, `canary`$_{n-3}$ in Fig. 3).

The list of canaries must also be increased when the program calls `alloca`. The space requested should be increased to accommodate a canary at the end of it and that canary should become the new list head. The list must also be decreased when the program calls `longjmp`; in this context it is similar to a *long return*. The value of `cHead` must be set with the address of the first canary (the one with lowest address) in the stack frame we are jumping into, or in some other stack frame below.

C. Policies for canary checking

Checking boundary canaries is a potentially expensive operation, thus it needs to be carefully managed in order to balance two requirements: (i) effective detection of stack overflows and (ii) efficient execution of the program. Furthermore, in terms of effectiveness, two natural approaches should be contemplated: (i) for development or testing purposes, the sooner the overflow is spotted the better, while (ii) for the execution of the program in production environments, preventing the application from "making damage" may be enough for most cases.

[4]In order to use a correct frame pointer, or stack pointer in compilations omitting stack frames, to setup their values.

Thus, considering the two execution environments mentioned above – development and production – we conceived two different policies for checking the correctness of boundary canaries. The two policies define when two lists of canaries are checked: (i) the list of canaries belonging to the current stack frame and (ii) the full list of canaries.

C.1 Checking local canaries

As previously explained, boundary canaries on the current stack frame should be checked when the stack frame is about to be released. Otherwise, we could fail to detect some local overflows. Note that an overflow in a stack frame may compromise memory areas all over the program if local memory pointers where affected (case 1 of Fig. 1). Consequently, the reduction of the canary list, both within a normal function return or within a call to `longjmp`, always checks the consistency of all released canaries.

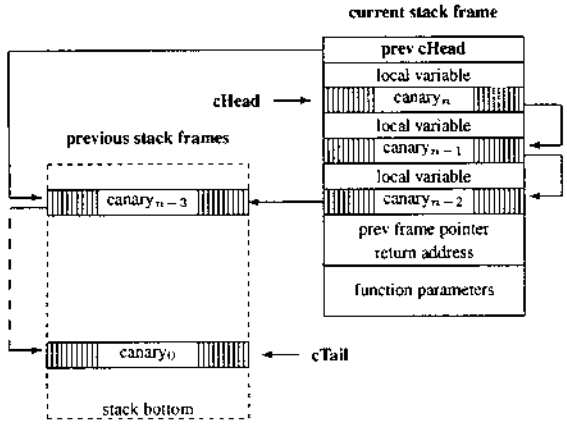
The extra code for checking local canaries and reducing the list of canaries was placed in a function (`canReduce`) that is called before the function's epilogue. All registers used to carry the return value of the ending function — EAX for 32-bit integer values or EAX+EDX for 64 bit integer values — are saved in the stack before calling `canReduce` and recovered afterward. `canReduce` gets, as a parameter, the address of the canary after the last one of the current stack frame (the address of `canary`$_{n-3}$ in Fig. 3).

For passing the correct value to `canReduce` we should not use the value stored in the last canary of the current stack frame because it may have been modified and, yet, be apparently valid. Being apparently valid means, in this context, that (i) it points to an address higher than its own and (ii) its value is not higher than the tail canary address (given by `cTail`). Thus, for a more accurate validation of canaries we store a clear copy of the previous head of the canary list at the top of each stack frame that adds canaries to the list (see Fig. 4). The function `canReduce` receives the previous head as parameter and follows the canaries from the current head until reaching the previous one; at the end the previous head will be stored in `cHead`. Overflow attacks cannot correctly corrupt both copies of the same value: one copy XORed with the XOR canary and another copy in clear, i.e., not XORed with the XOR canary (`canary`$_{n-2}$ and `prev cHead`, respectively, in Fig. 4).

For handling the reduction of the canary list after a `longjmp` call we use a function similar to `canReduce`. The function starts from `cHead` and walks along the canary list until finding a canary with an address higher than the stack pointer saved in the jump context. The address of that canary will be stored in `cHead`. Note that this approach does not require any modification of the jump context stored by `setjmp` and used by `longjmp`. It only requires an extension of the `longjmp` functionality.

C.2 Checking all canaries

We conceived two policies for checking the full list of canaries: one more suitable for development scenarios, another more suitable for production environments. In either case, we tried to prevent an attacked process from doing any




```
...
#--- Check/remove local canaries
push   %eax
mov    prev cHead off(%esp), %eax
push   (%eax)
call canReduce
pop    %eax
#--- end of canary processing
leave
ret
```

Figure 4 - Epilogue of function (returning a 32-bit integer value) with extra code to check local canaries and remove them from the global list.

I/O after a stack buffer overflow. The checking function is named `canWalk`.

Development policy: In a development scenario we want to catch an overflow as accurately as possible, in order to simplify the process of finding and fixing the vulnerability. It is, thus, natural to sacrifice execution efficiency in favor of debugging effectiveness. Our development policy consists of a `canWalk` call before each function call. This approach is computationally costly but has the advantage of detecting overflows not far from where they occurred.

The rationale for the development policy is the following. We want to check the list of canaries often but we need to define exactly how often, when and why. By checking before calling a function we can do it quite often and still prevent the program from doing any I/O after a stack overflow. We are assuming, of course, that for doing I/O it is necessary to call a function (e.g. a system call library function) but this is not completely true. In fact, I/O can happen without actually calling any functions. For instance, we can use memory-mapped I/O objects, like files, and do I/O using pointers and read/write memory accesses. But we believe that most I/O, even for memory-mapped objects, is done using functions of utility libraries, thus enabling our development approach.

Production policy: In a production scenario we want the applications to run efficiently and, yet, we want to detect overflows before they can interfere with their I/O. Our production policy consists of a watchdog process to catch all the system calls of the target process and to call `canWalk` before each I/O system call requested by the target process. We define an I/O system call as a system call interacting with I/O objects (files, pipes, sockets, etc.) or with other operating system resources (e.g. send signals to other processes, manage virtual memory attributes, change the ownership of the process, etc.).

This approach should be faster than the previous one, since it implies fewer calls to `canWalk`. Again, any I/O using pointers and memory-mapped objects will not trigger the checking of all boundary canaries. Thus, it may be possible to perform I/O operations after a stack overflow without notice. Nevertheless, the overflow would eventually be detected afterward.

V. IMPLEMENTATION

For implementing all the mechanisms and policies previously described for detecting stack buffer overflows we modified a C compiler and developed 3 auxiliary C modules for Linux systems that must be linked with the applications we want to protect. The applications don't need to be modified to use StackFences; they only need to be recompiled with the modified compiler and linked with some of the auxiliary modules. The C startup function `main` is transparently redefined for doing some setup actions in the auxiliary modules before actually starting the application.

For the C compiler we used the version 0.9.16 of TCC (Tiny C Compiler), a simple, complete and relatively easy to modify compiler developed by Fabrice Bellard [17]. We chose this compiler mainly because it allows a fast prototyping for getting a proof of concept. In the future we plan to modify `gcc`, a more popular but also more complex compiler, to support StackFences. For lack of space we will not describe here the modifications we did in TCC for implementing StackFences.

The first auxiliary module defines variables and functions that are common to all overflow checking facilities. Namely, it defines: the public variables `cXor`, `cHead` and `cTail`; the XOR canary setup function, the function `canReduce`; and a new `longjmp` function redefining the original one of the C library. For setting up a random value for the XOR canary this module uses the Linux special file `/dev/urandom`.

The second auxiliary module defines a new `main` function and all the checking and aborting functions for the development policy described in Section IV-C.2. The new `main` simply defines the initial boundary canary (`canary`$_0$ in Fig. 3), initiates the value of the variables in the previous module and calls the original `main` function of the application. None of the stack variables below the current stack frame are bounded with canaries; this means that program arguments and environment variables are still vulnerable. We decided this way because we believe that those variables can be protected more efficiently by other means, like memory protection mechanisms (see Section VIII).

The third auxiliary module defines a new `main` function and all the checking and aborting functions for the production policy described in Section IV-C.2. The new `main` first creates a child process for running the application and the initial process stays as the watchdog of the new one. The watchdog process will catch all system calls of the application process using the `ptrace` facility, typically used by debuggers and available in all Unix systems. The `main`

function, when executing in the child process, is basically identical to the one of the second module. Again, stack variables below the current stack frame are not protected by boundary canaries.

For the watchdog action the third module includes a function for catching the system calls of the child process. This function triggers the call to `canWalk` before allowing the execution of any requested I/O system call. The `canWalk` function is similar to the one in the second module but, because it runs in a different process, canary values must be loaded from the target process with the system call `ptrace` and the request `PEEK_DATA`.

VI. EFFECTIVENESS EVALUATION

We tested the effectiveness of StackFences with the test suite developed by J. Wilander and M. Kambar, described in [9] and kindly provided by the authors. The results were the best possible: StackFences detected and halted *all* 12 attacks overflowing stack variables. We where also able to do so using either of the canary checking policies described in Section IV-C.2.

The empirical results obtained with the same test suite and using 4 protection tools — StackGuard, StackShield, `propolice` and Libsafe/Libverify – showed that the tools where able to handle, in the best case, 10 out of the 12 attacks (with `propolice`) [9, Tables 4 and 5]. Note, however, that:

- the test suite is not complete, it only overflows character arrays, which are exactly the vulnerable variables considered by `propolice`. But StackFences is more powerful, being able to detect overflows in local variables other than character arrays. Therefore, StackFences is much better suited for assessing the correctness of applications in run-time than `propolice` but that cannot be fully demonstrated with this test suite.
- `propolice` does not detect any overflows within consecutive character arrays, as StackFences does, and such vulnerability is also not explored by the test suite.

There are other forms of attacks overflowing stack locations that are not performed by the test suite neither handled by StackFences. We are talking about overflows actually *using function parameters*, and not only considering them as targets. We believe that the general reason for researchers not considering such attacks is that overflows usually happen in arrays, typically for character strings, and C programmers usually pass arrays to functions by reference, not by value[5]. But we think that function parameters are as vulnerable as function variables and should be checked likewise, though that is not currently the case with StackFences.

VII. PERFORMANCE EVALUATION

The performance penalties introduced by StackFences depend on several factors, namely: (i) the number of vulnerable local variables; (ii) the number of function calls; (iii) the length of the list of canaries when `canWalk` is called; and (iv) the number of I/O operations that trigger the call to `canWalk` (if using the production policy). We could have devised microbenchmarks to study the influence of all these factors, but that would not help us to extrapolate the results in order to foresee the overheads introduced in real applications.

We decided to evaluate StackFences with both micro and macro benchmarks. The microbenchmarks provide upper bounds to the overheads caused by setting and checking canaries in each function's prologue and epilogue, respectively. The macro benchmarks help us to have an idea about the relative cost of each checking policy, because they control the calls to `canWalk`, and also to get an idea about the space occupied by canaries.

All benchmarks ran in a Red Hat 8.0 Linux box, with the kernel 2.4.18-27.8.0, a Intel Pentium IV CPU at 2.4 GHz, 256 Mbytes RAM and 512 Kbytes cache.

A. Microbenchmarks

As microbenchmarks we used a set of functions with a variable number of variables, all of them vulnerable, and an expression per variable for triggering their vulnerability (see Figure 5). We measured the minimum CPU clock cycles taken by 100 consecutive calls of those functions.

```
void null (int * x) {}
```

```
void f0 () {}
```

```
void f1 ()
{
  int A1;

  // Turn A1 vulnerable
  null ( &A1 );
}
```

...

```
void f32 ()
{
  int A1, ..., A32;

  // Turn A1, ... and A32 vulnerable
  null ( &A1 ); ...; null ( &A32 );
}
```

Figure 5 - Example of functions used in microbenchmarks.

The microbenchmarks where compiled in 3 different ways — with `gcc`, `tcc`, and `tcc` with StackFences and the production policy. The results of the evaluation are presented in Table I: the first two columns show function names and their number of vulnerable local variables; the remaining columns show the minimum CPU clock cycles observed in 10000 consecutive runs of the benchmark. The values were measured using the Pentium's RDTSC instruction after a proper serialization with a CPUID instruction (as suggested by Intel [18]) and corrected by subtracting

[5]To pass an array to a function by value the array must be member of a structure an the structure must be passed to the function by value.

function	vulnerable variables	elapsed time (ms) gcc	tcc orig.	tcc with StackFences	
f0()	0	3,520	3,528		(+0%)
f1()	1	4,524	4,532	6,732	(+49%)
f2()	2	5,724	5,736	8,960	(+56%)
f3()	3	6,940	7,004	10,556	(+51%)
f4()	4	8,136	8,136	12,576	(+55%)
f5()	5	9,340	9,332	14,388	(+54%)
f6()	6	10,536	10,824	16,512	(+53%)
f7()	7	12,148	11,932	21,212	(+78%)
f8()	8	12,928	12,928	26,656	(+106%)
f16()	16	22,756	22,536	47,988	(+113%)
f32()	32	41,724	41,768	92,956	(+123%)

Table I
RESULTS OF THE EXECUTION OF MICROBENCHMARKS FOR EVALUATING THE MAXIMUM OVERHEADS INTRODUCED BY STACKFENCES'S PROLOGUES AND EPILOGUES.

the minimum time taken by the measurement code. Therefore, they give a accurate notion of the maximum overheads imposed by StackFences in functions' prologues and epilogues. Note that these overheads are independent of StackFences's checking policies but the total overhead of StackFences per function used in the microbenchmark is not independent, because with the development policy we would call `canWalk` before actually calling the dummy function. To avoid such call we compiled the benchmarks only with production policy, as previously referred.

The microbenchmarks show that the maximum base overhead for protecting local variables is significant (approximately 50%) but the maximum cost per variable is relatively reduced. The base overhead is mostly due to the epilogue, namely to the `canReduce` function, because it implements a costly validation loop. StackFences's prologues are faster because they have no loops and no tests. The microbenchmarks also show that the overheads introduced by StackFences, when comparing with `gcc`, are not imputable to `tcc`. Thus, since StackFences's prologues and epilogues are almost compiler-independent[6], a similar overhead could be obtained with `gcc`.

B. Macrobenchmarks

For the evaluation of StackFences with macro benchmarks we chose 3 tools with moderate file I/O and high CPU activity: `ctags`, `tcc` and `bzip2`. These 3 tools where compiled in 4 different ways — with `gcc`, `tcc`, and `tcc` with the two StackFences checking policies — and executed with 3 different parameters — the sources of each of the 3 tools. For `ctags` and `tcc` we used a list of source files; for `bzip2` we used a `tar` file with the sources.

The results of the evaluation are presented in Table II: the second and third columns show the number of local variables used by the tools and the number and percentage of them that are vulnerable and checked by StackFences; the fourth column shows the arguments used with the tools; the fifth column shows the protection policy used with StackFences. In the rest of the columns we have execution results: the sixth and seventh columns show the elapsed time observed with the tools compiled with `gcc` (with maximum optimization) and the normal `tcc` (that has no optimizations); the eighth column shows the elapsed time observed with the tools compiled with `tcc` and StackFences, using both security policies, and the overhead in percentage comparing with the results of the previous column; the last six columns show the number of calls to `canReduce` and `canWalk` and the average and maximum number of canaries checked per call. StackFences' auxiliary modules were written in C and compiled with `gcc` and maximum optimisation. The elapsed times are the minimum observed in 100 consecutive runs of each test.

The values provided for `gcc` are only indicative, because many dynamic solutions for dealing with buffer overflows were implemented with it. But it doesn't make sense to extrapolate the overhead of StackFences comparing with `gcc` because some of the optimizations used by the latter would also reduce the overhead of StackFences if it was part of `gcc`. For instance, `tcc` has many small `inline` functions. Since our version of TCC ignores `inline` qualifiers, that greatly increases the number of `canReduce` calls and the average number of canaries checked by `canWalk`. Such overhead would not happen if we had integrated StackFences with `gcc`.

The results in Table II show that the overheads introduced by StackFences are acceptable. With the development policy, overheads are between 22% and 189% of the elapsed time with `tcc` and without StackFences. With the production policy, overheads are lower, as desired, between 3% and 41%. The production policy greatly reduces the number of calls to `canWalk`, as expected. But in some cases, namely with `ctags`, results show that it is almost irrelevant to use either checking policy. That happens because the average number of canaries checked by `canWalk` is low, making more relevant the cost of the process switching between the application and its watchdog in the production policy.

Considering the applications tested, the extra space occupied by canaries in the stack is not an issue. In the worst case there is a maximum of 252+10 canaries (when `tcc` is compiled by itself using the development policy), which represents a memory overhead of about 1 KB.

A small comparison can be established between the performance of `ctags` with StackGuard and with StackFences. According to [2], `ctags` with StackGuard has an overhead of 80% when processing 78 files, 37,000 lines of code. Using the same number of files and an approximated number of lines of code (37,188) we got for StackFences a lower performance penalty: 31% with the production policy and 41% with the development policy.

According to Table II, `gcc` and `tcc` generate equally fast `ctags` executables. Therefore, for this particular experience we can compare the overheads of the two protection mechanisms independently of the compilers implementing them. And the conclusion is that StackFences is faster than

[6]Actually StackFences's prologues/epilogues use the frame pointer, which is the common case and the only one supported by `tcc`. Thus, in compilations omitting the frame pointer, that are possible with `gcc`, they must be generated differently.

tool	local variables		arguments	StackFences policy	execution									
					elapsed time (ms)				StackFences statistics					
					gcc -O3	tcc			canReduce			canWalk		
	total	vulnerable				orig.	with StackFences		calls	length avg.	length max.	calls	length avg.	length max.
bzip2	462	302 (65%)	bzip2 sources	development	70	122	149	(+22%)	1.667	5.9	11	437.946	21.8	24
				production			126	(+3%)				48	7.4	13
			tcc sources	development	207	324	396	(+22%)	1.970	6.0	11	1.127.521	22.1	24
				production			334	(+3%)				62	8.4	13
			ctags sources	development	230	360	439	(+22%)	1.683	5.9	11	1.207.621	22.3	24
				production			371	(+3%)				60	8.5	13
tcc	978	603 (62%)	bzip2 sources	development	89	167	419	(+151%)	411.256	2.4	10	2.927.026	30.8	165
				production			217	(+30%)				1.250	14.6	65
			tcc sources	development	97	179	517	(+189%)	464.208	2.5	10	3.737.023	33.5	252
				production			222	(+24%)				928	15.9	60
			ctags sources	development	304	541	1.149	(+112%)	1.231.803	2.3	10	8.297.317	24.1	163
				production			724	(+34%)				4.682	13.9	102
ctags	911	178 (20%)	bzip2 sources	development	36	36	49	(+36%)	3.338	1.4	3	1.334.837	2.1	5
				production			44	(+22%)				137	2.3	4
			tcc sources	development	91	92	132	(+43%)	13.899	1.4	3	3.599.858	2.2	5
				production			116	(+26%)				207	2.3	4
			ctags sources	development	91	98	138	(+41%)	16.530	1.4	3	3.480.356	2.2	5
				production			138	(+41%)				391	2.2	4

Table II

RESULTS OF THE EXECUTION OF MACRO BENCHMARKS WITH EXISTING APPLICATIONS FOR EVALUATING THE TOTAL OVERHEADS INTRODUCED BY STACKFENCES AND STATISTIC DATA REGARDING THE CHECKING OF STACKFENCES' CANARIES.

StackGuard, which makes less validation actions! This paradox is probably explained by the fact that the performance figures presented in [2] were obtained with a *non-optimized version of StackGuard, that added canaries and code for checking them to all functions*, instead of doing it only for functions with vulnerable local variables. This fact is mentioned in the performance optimisations for StackGuard described in [2]. StackFences, on the contrary, was optimized to add canaries and for checking them locally with `canReduce` *only in functions with vulnerable variables*.

We believe that the overhead introduced by StackFences can be further reduced. For instance, different `canReduce` functions may be used for different lengths of local canary lists, allowing more effective loop unrolling optimizations. For reducing the overhead of the production policy we can keep the current watchdog model but use a thread, or Linux clone of the monitored process, to execute the function `canWalk`, or we can develop a kernel module or patch for tracing only the relevant system calls (see [19] for a detailed analysis and examples of this approach), instead of tracing all system calls from the watchdog process with `ptrace`.

VIII. CONCLUSIONS AND FUTURE WORK

We have presented StackFences, a run-time solution for detecting buffer overflows affecting variables allocated in stack frames. StackFences detects overflows in all potentially vulnerable local variables instead of detecting only overflows affecting known attack targets, like return addresses. To the best of our knowledge this is the first solution to perform such a detailed run-time stack analysis. For this purpose StackFences extends two canary mechanisms to detect overflows: the canary mechanism introduced in StackGuard and the XOR canary mechanism introduced in StackGhost. StackFences uses canaries for monitoring all local variables susceptible to overflow, either affecting highly sensible values, like return addresses, or not.

For balancing detection accuracy and performance we conceived two checking policies for StackFences: a development policy, more detailed and allowing a more accurate and timely detection of overflow occurrences, suitable for development scenarios; and a production policy, lazier, checking only when absolutely necessary, thus more suitable for production environments.

For testing StackFences we developed a prototype for Linux systems using TCC (Tiny C Compiler). C modules compiled with TCC and StackFences are fully compatible with the standard C libraries and with modules compiled with other compilers or compilation options. The modifications introduced by StackFences affect only the code inside each function.

In terms of effectiveness, StackFences detected and halted *all* the 12 attacks of the test suite developed by Wilander and Kambar [9]. Although it may appear that StackFences is minutely different from `propolice`, which avoids 10 of those attacks, that is not true because StackFences detects other overflows that are not considered by the test suite and neither tackled by `propolice`. Namely, (i) `propolice` only tries to reduce the impact of overflows in character arrays, while StackFences detects all overflows in other kinds of vulnerable stack variables (ii) `propolice` may not detect overflows within consecutive character arrays, as StackFences does. Therefore, StackFences is much better suited for assessing the correctness of applications in run-time than `propolice`, though that is not properly demonstrated by the test suite.

The performance of StackFences is acceptable but depends a lot on the compiled application. Microbenchmarks showed that StackFences's prologues and epilogues have a significant maximum base overhead of about 50% in the elapsed time, but a small overhead per each vulnerable local variable. Macrobenchmarks with 3 tools showed a overhead in the elapsed time between 22% and 189%, when us-