Esquema de Proteção Desigual de Erros para Dados Compactados usando LZSS

Marcos A. Siqueira, Marcelo E. Pellenz, Altair O. Santin e Richard Demo Souza

***Resumo* — Neste artigo propomos uma estratégia de codificação de canal para minimizar o impacto de surtos de erros na transmissão ou armazenagem de arquivos compactados. A estratégia proposta é para o algoritmo de compressão baseado em dicionário dinâmico LZSS, amplamente utilizado para compressão de dados. Investigamos através de uma extensa análise computacional as classes de informação mais sensíveis a erros e propomos uma estratégia para proteção desigual que minimiza a propagação de erros no processo de descompressão.**

***Palavras-Chave* — LZSS, compressão de dados, códigos de controle de erros, proteção desigal de erros, armazenamento de dados.**

***Abstract* — In this paper a channel coding strategy is proposed for minimizing the impact of error bursts in the transmission or storage of compressed files. The strategy is proposed for the dynamic dictionary compression algorithm LZSS, which is widely used for data compression. We investigate through extensive computer simulations which information classes are more sensitive to errors, and propose an unequal error protection strategy that minimizes the error propagation during the decompression process.**

***Index terms* — LZSS, data compression, error control coding, unequal error protection, data storage.**

I. INTRODUÇÃO

Os algoritmos de compressão sem perdas baseados na codificação de Lempel-Ziv são amplamente usados para compressão de textos [1-4] e também em determinados esquemas de compressão com perdas para imagem e vídeo [5][6]. Em sistemas multimídia, sempre é necessário manipular, armazenar e transmitir um grande volume de dados compactados. Contudo, dados compactados são muito sensíveis a qualquer evento de erro, devido ao fato que estes erros podem se propagar durante o processo de descompressão. Portanto, a utilização de técnicas de codificação de canal otimizadas para transmissão e armazenagem de dados digitais compactados através de canais ruidosos pode minimizar significativamente estes efeitos. Estas técnicas conjuntas de codificação fonte/canal permitem a recuperação parcial de dados em arquivos compactados corrompidos e também o aumento de eficiência de transmissão em esquemas de controle de erros híbridos FEC/ARQ, onde apenas os segmentos de informação corrompidos são retransmitidos.

Marcos A. Siqueira, Marcelo E. Pellenz e Altair O. Santin, Programa de Pós-Graduação em Informática Aplicada, Pontifícia Universidade Católica do Paraná, Curitiba, Brasil, E-mails: {msiqueira, marcelo,santin}@ppgia.pucpr.br. Richard Demo Souza, Centro Federal de Educação Tecnológica, Paraná, Curitiba, Brasil. E-mail: richard@cpgei.cefetpr.br

Para minimizar a degradação de qualidade da informação devido ao processo de propagação de erros na descompressão, é necessário aplicar estratégias para contenção e recuperação de erros [7]. Contudo, algumas partes da informação compactada são mais sensíveis em termos de propagação de erros que outras. Neste caso, um esquema de proteção desigual pode ser aplicado, protegendo partes da seqüência contra um número maior de erros [8]. O uso de estratégias otimizadas de proteção desigual para dados compactados pode encontrar também grandes aplicações em redes sem fio [9], dado que estas estratégias podem minimizar a retransmissão de informação, melhorando o *throughput* do sistema.

Em [10] os autores propõem uma estratégia de controle de erros para códigos aritméticos usando proteção desigual de erros. Uma estratégia similar é proposta em [11] para o algoritmo de compressão LZ77. Neste artigo investigamos o impacto de erros em dados compactados usando o algoritmo LZSS [3][4]. O algoritmo LZSS é derivado da amplamente utilizada família LZ77 [1][2]. Simulações computacionais foram realizadas de maneira a se identificar o impacto de surtos de erros nos dados compactados. Baseado nestes resultados numéricos, uma estratégia para contenção de erros usando um esquema de proteção desigual de erros foi proposto para proteger a classe mais sensível dos dados compactados.

Este artigo está organizado da seguinte maneira. Na Seção II é apresentada uma breve descrição das técnicas de compressão adaptativas LZ77 e LZSS. Na Seção III investigamos através de simulações computacionais o impacto dos erros no arquivo compactado e a sua propagação no procedimento de descompressão. Na Seção IV é proposta uma estratégia de proteção desigual de erros e sua implementação no algoritmo LZSS. As conclusões são apresentadas na Seção V.

II. TÉCNICAS DE COMPRESSÃO USANDO DICIONÁRIOS ADAPTATIVOS

Em muitas aplicações a saída de uma fonte consiste de padrões recorrentes. Existem padrões que aparecem constantemente e outros que aparecem mais raramente. Uma abordagem bastante razoável para codificar estas fontes é manter uma lista dos padrões de recorrência mais freqüentes. Quando estes padrões aparecem na saída da fonte eles são codificados com uma referência à lista de padrões. Na seqüência são descritas duas estratégias clássicas de codificação por referência.

A. Algoritmo LZ77

Na abordagem do algoritmo LZ77 [1][2], a referência é simplesmente uma fração da seqüência previamente codificada. O codificador examina a seqüência de entrada através de uma janela deslizante de *n* caracteres (símbolos), com *c* caracteres formando o texto a ser compactado (*look-ahead buffer*) e *n-c* caracteres formando a seqüência recentemente codificada (*search buffer*). Para codificar uma seqüência no *look-ahead buffer*, o codificador move um ponteiro de busca através do *search buffer* até encontrar uma correspondência com o primeiro símbolo do *look-ahead buffer*. A distância do ponteiro de referência do *look-ahead buffer* é chamada de *offset* (o_i). O codificador examina os símbolos seguintes ao símbolo referenciado pela localização do ponteiro para verificar se existe correspondência para outros símbolos consecutivos no *look-ahead buffer*. O número de símbolos consecutivos no *search buffer* que são idênticos aos símbolos consecutivos no *look-ahead buffer*, iniciando com o primeiro símbolo referenciado pelo ponteiro, é denominado de *match length* (f_i). O codificador procura no *search buffer* pela maior seqüência idêntica. Uma vez que a maior seqüência idêntica foi encontrada, o sistema codifica uma tripla (o_i, f_i, u_i), onde u_i é o símbolo subseqüente ao *match length* no *look-ahead buffer*.

B.Algoritmo LZSS

Existem diversas variações do algoritmo LZ77 que foram propostas com o objetivo de torná-lo mais eficiente. A maioria das modificações propostas trata da representação mais eficiente da tripla codificada. Uma modificação do algoritmo LZ77 usada pela maioria de suas variantes é eliminar o caso onde a tripla é usada para codificar apenas um caractere, pois neste caso o uso da tripla é altamente ineficiente, especificamente se um grande número de caracteres aparece raramente no texto a ser codificado. Uma estratégia eficiente é a adição de um bit de *flag* para indicar se o que segue é uma palavra código ou um único símbolo. Utilizando este bit de *flag* torna-se desnecessário a utilização do terceiro elemento da tripla codificada. Esta versão modificada do LZ77 foi denominada de algoritmo LZSS [3][4]. A estrutura dos dados compactados é mostrada na Figura 1.

O arquivo compactado pode ser transmitido ou armazenado de acordo com a estrutura mostrada na Figura 1, onde *F* denota os *bits de flag*, *C* denota os *bits de caractere*, *M* corresponde aos *bits de match length*, e *O* aos *bits de offset*. Esta estrutura permite a descompressão instantânea do arquivo compactado conforme os dados vão sendo recebidos.

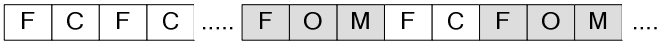

Fig. 1. Estrutura dos dados compactados no algoritmo LZSS.

III. AVALIAÇÃO DO IMPACTO DO SURTO DE ERROS NO PROCESSO DE DESCOMPRESSÃO

No processo de descompressão, o bit de *flag* indica se a informação subseqüente é um texto não codificado ou uma palavra código de referência ao dicionário. No caso de uma palavra código, o algoritmo de descompressão copia o símbolo indicado pela posição do ponteiro e todos os símbolos subseqüentes referenciados pelo *match length* [3][4]. Se erros forem introduzidos durante a transmissão ou armazenagem dos dados compactados, os seus efeitos no processo de descompressão dependem do tipo de informação afetada: bits de *flag*, bits de *caractere*, bits de *match length* ou bits de *offset*. Por exemplo, se os bits afetados forem os bits de *match length*, o tamanho do arquivo descompactado será alterado, aumentando ou reduzindo o tamanho do arquivo de dados original [11].

Com o objetivo de avaliar o impacto do surto de erros num arquivo compactado com o algoritmo LZSS, desenvolvemos um software de simulação usando MATLAB® e linguagem C++. O software permite analisar o efeito da propagação dos erros no arquivo descompactado em função da posição de ocorrência do surto de erros no arquivo compactado, cuja estrutura é mostrada na Figura 1. Para realizar a análise, utilizamos um arquivo texto de uma base de dados para teste de algoritmos de compressão [12]. Um arquivo da mesma base de dados foi utilizado para avaliar o esquema de proteção desigual apresentado em [11] para o algoritmo LZ77. Analisamos especificamente o arquivo texto "paper4.txt" de tamanho 13KB.

A Tabela I apresenta uma síntese dos parâmetros do arquivo compactado. O algoritmo de compressão foi implementado utilizando-se 8 bits para codificar *caractere*, 1 bit para codificar *flag*, 12 bits para codificar *match length* e 16 bits para codificar *offset*. As Figuras 2 e 3 mostram os valores de *match length* e de *offset*, respectivamente, gerados no arquivo compactado.

A estrutura do arquivo para o esquema de codificação de canal usando proteção desigual de erros é mostrada na Figura 4, onde os bits são agrupados em diferentes classes, onde cada classe pode receber um nível de proteção específico.

TABELA I
INFORMAÇÕES DO ARQUIVO COMPACTADO

ESPECIFICAÇÃO	TAMANHO
Tamanho do arquivo descompactado	13286 bytes
Tamanho do arquivo compactado	9014 bytes
Número de bits do arquivo compactado	72108 bits
Número de símbolos de *flag* no arquivo compactado	3772 símbolos
Número de símbolos de caractere no arquivo compactado	1864 símbolos
Número de símbolos de *offset* no arquivo compactado	1908 símbolos
Número de símbolos de *match* no arquivo compactado	1908 símbolos

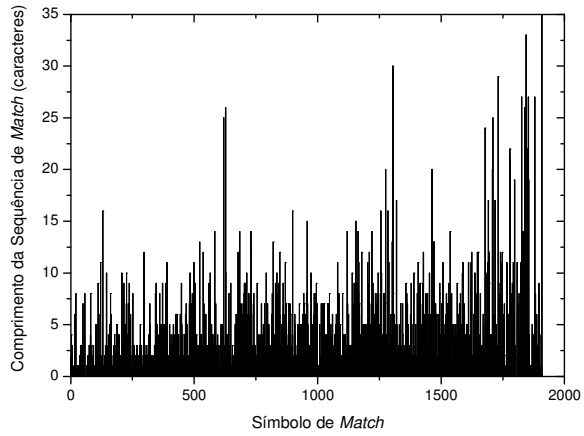

Fig. 2. Valores dos símbolos de *match length* no arquivo compactado.

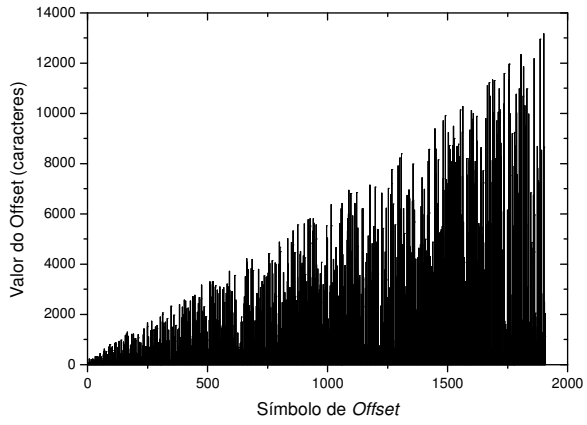

Fig. 3. Valores dos símbolos de *offset* no arquivo compactado.

Nesta estrutura o número de bits em cada classe deve ser transmitido com o arquivo compactado. Esta informação constitui o bloco denominado de cabeçalho (header) mostrado na Figura 4. Para efeitos de análise do esquema de proteção desigual, estamos considerando que a informação de cabeçalho é sempre recebida livre de erros.

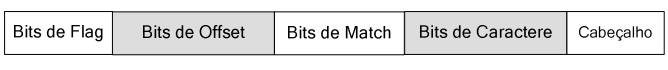

Fig. 4. Estrutura do arquivo compactado para proteção desigual de erros.

No procedimento de avaliação, inicialmente efetuamos a compressão do arquivo "paper4.txt" usando o algoritmo LZSS. Então o algoritmo compactado é processado pelo software, que realiza a segmentação dos dados compactados em bits de flag, bits de caractere, bits de offset e bits de match. Após a decomposição o efeito de propagação de um surto de erros é analisado de forma independente para cada parâmetro. Seguindo o procedimento adotado em [10][11], aplicamos um surto de erros de 48 bits no arquivo compactado. Através deste procedimento podemos identificar a influência dos erros sobre cada parâmetro do arquivo compactado e qual o seu impacto na reconstrução do arquivo original durante o processo de descompressão.

Na primeira análise mostrada na Figura 5, avaliamos o impacto de um surto de erros de 48 bits introduzido em diferentes posições do arquivo compactado. A posição do surto é deslocada em intervalos de 48 bits, ou seja, eles são aplicados a seqüências consecutivas de 48 bits no arquivo compactado (bits 1-48, 49-97, 98-146, e assim por diante). Os valores no eixo *x* representam cada um dos passos de 48 bits. Dado que o tamanho do arquivo é de 72108 bits, temos 72108/48=1500 seqüências de 48 bits. O eixo *y* representa o percentual de erros no arquivo descompactado em função da localização do surto de erros no arquivo compactado. É importante ressaltar que a propagação de erros é mais significativa se o surto ocorrer no início do arquivo compactado, pois nas posições iniciais se encontra a maior parte da informação de dicionário referenciada ao longo da compressão. Portanto a informação do *search buffer* torna-se inconsistente, gerando propagação de erros na descompressão.

A Figura 6 apresenta o efeito do surto de erros quando o arquivo compactado é entrelaçado antes da transmissão. A profundidade do entrelaçador é de 36056 bits. Podemos observar que o entrelaçador espalha o surto de erros ao longo do arquivo compactado e amplifica o efeito de propagação dos erros na descompressão.

A Figura 7 apresenta o efeito de propagação de erro no arquivo descompactado considerando a ocorrência de um surto apenas nos bits de flag. Note que o eixo *x* indica seqüências de bits de flag de 48 bits. Dado que o número total de bits de flag no arquivo compactado é de 3772 bits, o eixo *x* se estende até 78 valores na Figura 7. Conforme descrito na seção anterior, o bit de flag indica se a informação subseqüente corresponde a texto codificado ou não codificado. Se os bits de flag forem corrompidos teremos como conseqüência a substituição de texto claro por palavras códigos e vice-versa. Como resultado teremos bits de offset indicando posições iniciais erradas e bits de match indicando comprimento de cópia errada no dicionário. A propagação de erro gerada pelos bits de flag é similar aos resultados apresentados na Figure 6, onde os erros são introduzidos sequencialmente no arquivo compactado, de acordo com a estrutura da Figura 2. Erros nos bits de flag corrompem o tamanho do arquivo no processo de descompressão, gerando um arquivo descompactado com tamanho diferente do original.

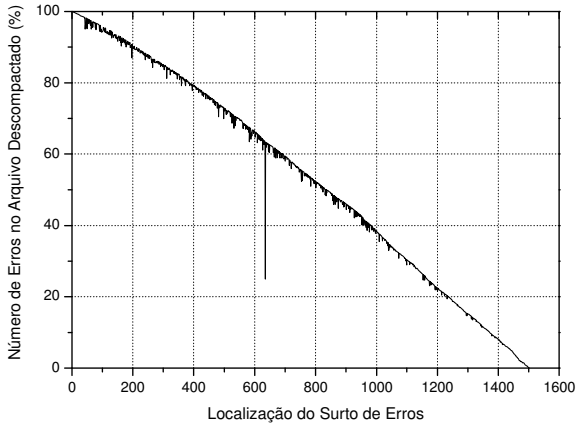

Fig. 5. Efeito do surto de erros no arquivo compactado em termos da localização do surto de erros.

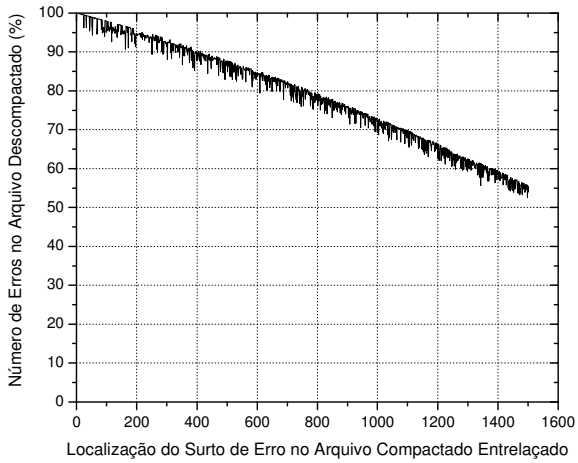

Fig. 6. Efeito do surto de erros no arquivo compactado em termos da localização do surto de erros, no arquivo compactado entrelaçado.

As Figuras 8, 9 e 10 apresentam resultados para erros inseridos nos bits de *match*, bits de *offset* e bits de *caractere*, respectivamente. A escala do eixo *x* segue o mesmo critério usado para as Figuras 5, 6 e 7. Podemos verificar nas Figuras 8 e 9 que os bits de *match* são mais relevantes que os bits de *offset* em termos de propagação de erros durante o processo de descompressão. Podemos também observar na Figura 11 que a propagação de erros é menor que 5% se o surto afetar apenas bits de caractere. Neste caso os dados podem ser recuperados com um pequena fração de erros. Baseado nesta análise podemos otimizar a estratégia de propagação de erros que será aplicada aos dados compactados. Erros nos bits de offset e de caractere causam menos impacto no processo de descompressão. Contudo, erros nos bits de flag causam um efeito severo na descompressão, ainda mais se eles ocorrerem no início do arquivo compactado.

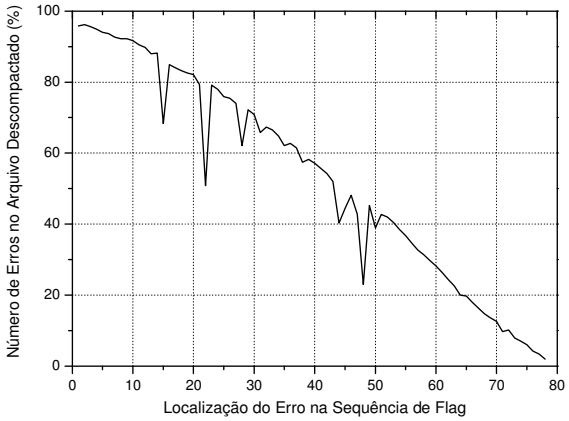

Fig. 7. Efeito do surto de erros nos bits de *flag* do arquivo compactado.

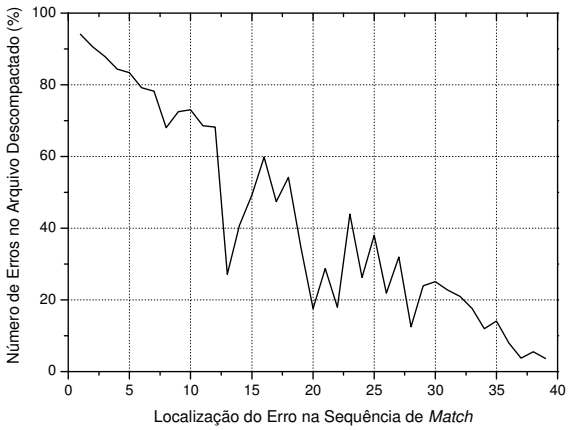

Fig. 8. Efeito do surto de erros nos bits de *match length* do arquivo compactado.

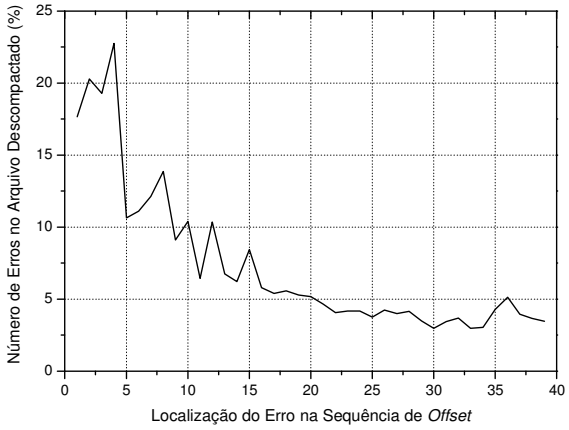

Fig. 9. Efeito do surto de erros nos bits de *offset* do arquivo compactado.

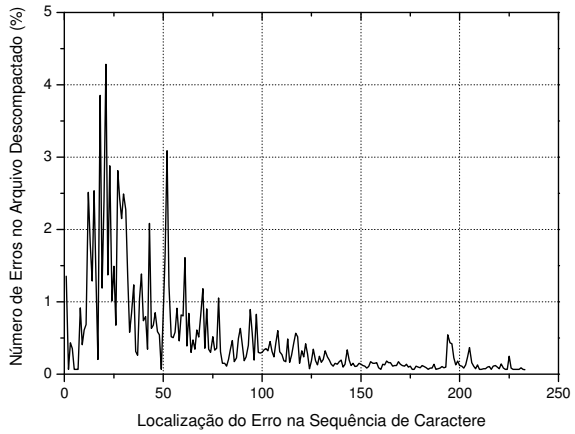

Fig. 10. Efeito do surto de erros nos bits de *caractere* do arquivo compactado.

IV. Estratégia de Proteção Desigual de Erros

A estratégia básica para prover proteção desigual de erros para transmissão ou armazenagem é a utilização de esquemas de codificação diferenciados para as diferentes classes de informação. Para o caso de dados compactados usando o algoritmo LZSS, uma estratégia natural é aplicar mais proteção para os bits de flag.

Na estratégia proposta neste artigo utilizamos um código Reed-Solomon [13], que é apropriado para a correção de surtos de erros em canais com memória. Objetivando manter uma boa eficiência de compressão, aplicamos um código com alta taxa para adicionar pouca redundância ao arquivo compactado. Os parâmetros do código Reed-Solomon utilizado foram (n,k,t)=(255,247,4) sobre GF(256). Neste caso o codificador opera com blocos de informação de k=247 símbolos e produz palavras código de comprimento n=255 símbolos. A capacidade de correção é de $t=4$ símbolos, que é equivalente a seqüência de erros de 32 bits. O número de bits de redundância adicionados ao arquivo compactado é de 2368 bits, o que equivale a um acréscimo de apenas 3,28% no tamanho do arquivo compactado.

Numa análise preliminar aplicamos o código Reed-Solomon para proteger todos os bits de dados compactados, ou seja, sem proteção desigual de erros. O resultado é mostrado na Figura 11 para um surto de erro de 48 bits introduzido em diferentes posições ao longo do arquivo compactado. O eixo x das Figuras 11 e 12 são definidos da mesma maneira que na Figura 5. Comparando as Figuras 5 e 11 podemos observar que apenas para algumas posições específicas de localização do erro que a estratégia de codificação é significativamente eficiente para reduzir a propagação de erros.

Na segunda análise aplicamos o código Reed-Solomon apenas nos bits de flag, numa estratégia de proteção desigual de erros. Os bits de offset, bits de match e bits de caractere do arquivo compactado não foram codificados. O resultado é mostrado na Figura 12, onde podemos observar uma redução significativa do efeito de propagação de erros no arquivo descompactado. Note também que a ocorrência de um surto de erros nos bits iniciais de flag é mais severa em termos de propagação. Contudo, para erros nos bits de flag subseqüentes, a propagação é reduzida de forma significativa. Isso ocorre porque a maioria das referências ao dicionário durante o processo de compressão estão localizadas no início do arquivo compactado. Quando utilizamos a estratégia de proteção desigual, a propagação média de erros reduz de 6016 para 956 símbolos. Isso mostra claramente o impacto da proteção desigual para este tipo de informação.

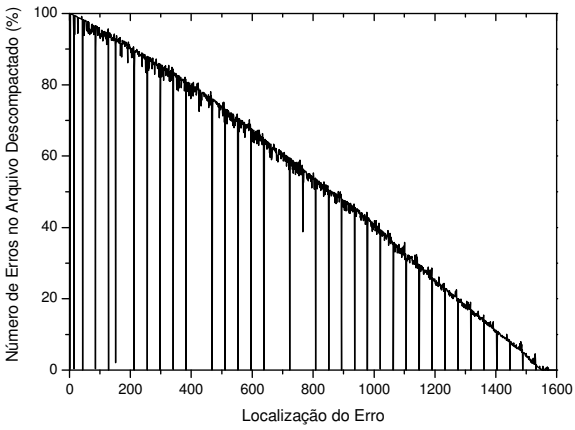

Fig. 11. Efeito do surto de erro no arquivo compactado com proteção igual de erros.

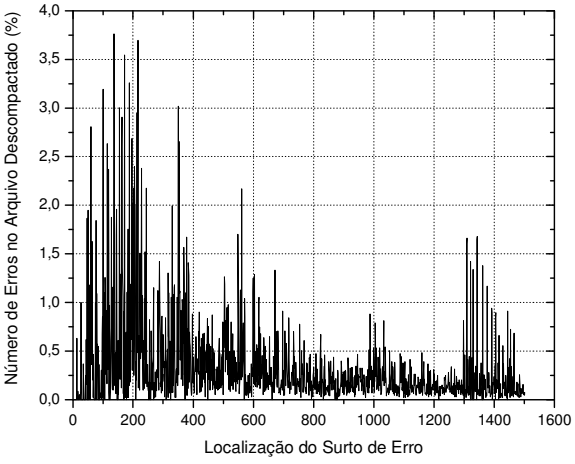

Fig. 12. Efeito do surto de erro no arquivo compactado com proteção desigual de erros.

V. CONCLUSÕES

Neste trabalho apresentamos uma análise detalhada do efeito do surto de erros sobre dados compactados usando o algoritmo LZSS. Um software foi desenvolvido para decompor a estrutura do arquivo compactado e identificar as classes de informação que possuem maior impacto na propagação de erros durante o processo de descompressão.

Como estudo de caso, um arquivo texto do *Calgary Corpus* foi analisado e os resultados para a ocorrência de erros nos bits de *caractere*, *match*, *flag* e *offset* foram apresentados. Baseado nestes resultados, um esquema de proteção desigual de erros usando um código Reed-Solomon foi proposto para os bits de *flag*. O esquema proposto se mostrou eficiente em termos da redução do efeito de propagação de erros, reduzindo em mais de seis vezes o número médio de erros propagados, conforme observado no processo de simulação. A estratégia proposta pode ser utilizada com outros códigos corretores de erro e também aplicada para os bits de *match* e *offset*.

REFERÊNCIAS

[1] J. Ziv and A. Lempel, "A universal algorithm for data compression," *IEEE Trans. Inform. Theory,* vol. 23, no. 3, pp. 337-343, May 1977.

[2] K. Sayood, *Introduction to Data compression,* 2nd Edition, Morgan Kaufmann Publishers, 2000.

[3] T. Bell, "Better OPM/L text compression," *IEEE Trans. Commun*. vol. 34, no. 12, pp. 1176-1182, Dec. 1986.

[4] G. Held, *Data and Image Compression – Tools and Techniques,*New York: John Wiley and Sons Ltd, 1996.

[5] S. W. Chiang and L. M. Po, "Adaptive lossy LZW algorithm for palettised image compression," *IEE Electronics Letters*, vol. 33, no. 10, pp. 852-854, May 1997.

[6] D. Greene, M. Vishwanath, F. Yao and T. Zhang, "A progressive Ziv-Lempel algorithm for image compression", Xerox Palo Alto Research Center, Computer Science Department, Stanford University.

[7] S. Lin and D. J. Costello Jr., *Error Control Coding*, Prentice-Hall, 1983.

[8] B. Masnick and J. Wolf, "On linear unequal error protection codes," *IEEE Trans. Inform. Theory,* vol. 13, no. 4, pp. 600-607, Oct. 1967.

[9] M. Budagavi, W. R. Heinzelman, J. Webb, and R. Talluri, "Wireless MPEG-4 video communication on DSP chips ," *IEEE Signal Proc. Mag.*, vol. 17, no. 1, pp. 36-53, Jan. 2000.

[10] E. Fujiwara, H. Chen and M. Kitakami, "Burst error recovery method for VF arithmetic codes," *IEEE ITW'99*, June 1999.

[11] E. Fujiwara, H. Chen and M. Kitakami, "Error recovery for Ziv-Lempel coding by using UEP Schemes," in *IEEE ISIT'00*, June 2000.

[12] http://www.data-compression.info/Corpora/CalgaryCorpus/

[13] T. Berman and J. Freedman, "Non-interleaved Reed-Solomon coding over a bursty channel," *IEEE Milcom'92*, Oct. 1992.