Compressão de Dados em Redes de Sensores sem Fio Usando Codificação de Huffman

Marcos Costa Maciel, Henry Ponti Medeiros, Richard Demo Souza, Marcelo Eduardo Pellenz

***Resumo*— Esse artigo apresenta um método simples de compressão de dados para redes de sensores sem fio. Ao invés de se criar um novo mecanismo *adhoc* para resolver o problema, tenta-se mostrar que, dado o conhecimento geral prévio de algumas características dos parâmetros a serem monitorados, pode-se implementar uma codificação de Huffman convencional para comprimir dados de um mesmo fenômeno em diferentes localidades e períodos de tempo. As probabilidades dos parâmetros podem ser inferidas, por exemplo, através de conjuntos de dados públicos. O dicionário Huffman calculado usando essas probabilidades é utilizado para aproximar a entropia das diferenças das medições. Os resultados experimentais com conjuntos de dados de temperatura mostram que o método proposto supera um dos mecanismos de compressão mais populares projetados para este fim.**

***Palavras-Chave*— Redes de Sensores sem Fio, Compressão de dados.**

***Abstract*— This work presents a lightweight data compression method for wireless sensor networks. Instead of attempting to devise novel ad-hoc mechanisms to solve the problem, we attempt to show that, given previous general knowledge of the parameters that must be monitored, it is possible to efficiently employ conventional Huffman coding to represent the same phenomenon when measured at a different location and time period. The statistics of the parameter which must be monitored can be inferred, for example, from public datasets. The Huffman dictionary computed using those statistics is shown to approach the entropy of the data difference. Experimental results with temperature datasets show that the proposed method outperforms one of the most popular compression mechanisms for sensor networks.**

***Keywords*— Wireless Sensor Networks, Data Compression.**

I. INTRODUÇÃO

Um dos maiores desafios para a utilização em larga escala de redes de sensores sem fio (RSSFs) com aplicação prática é o desenvolvimento de mecanismos que possibilitem as redes operar por períodos prolongados de tempo considerando a limitação de energia dos nós sensores [1]. Sabendo-se que a comunicação de dados é geralmente um dos principais fatores responsáveis pelo consumo das reservas de energia da rede, o desenvolvimento de técnicas para reduzir a quantidade de informação transmitida pelo nó sensor, através da compressão dos dados, é de grande interesse.

Embora a área de pesquisa sobre compressão de dados esteja bem estabelecida, existem algoritmos que não podem ser utilizados de forma embarcada no nó sensor devido aos recursos limitados de *hardware*. Entretanto, uma quantidade de métodos de compressão de dados especificamente desenvolvidos para RSSFs foram propostos nos últimos anos [2]–[8]. O que muitos desses métodos tem em comum é o fato de que eles fazem uso da correlação dos dados coletados pelos nós sensores para atingir alta taxa de compressão empregando algoritmos computacionalmente simples.

O algoritmo proposto por Marcelloni e Vecchio [2], [3] é um dos métodos mais eficientes desenvolvidos de acordo com esses princípios. Ele processa as diferenças das medições consecutivas realizadas pelos sensores e divide-as em grupos de tamanhos incrementados exponencialmente. Esses grupos são então codificados por entropia usando uma tabela de compressão fixa baseada no algoritmo JPEG para compressão de coeficientes DC de imagens. Os autores registraram altas taxas de compressão sobre dados coletados por RSSFs reais.

As RSSFs são geralmente implantadas para monitorar um fenômeno particular de interesse [1]. Partindo deste fato, e de que em geral é possível ter um bom conhecimento *a priori* do comportamento estatístico deste fenômeno, será mostrado que usando um esquema de compressão baseado em uma tabela de Huffman fixa é possível atingir taxas de compressão maiores do que as obtidas em [2], [3]. Especificamente, será mostrado que construindo um dicionário de Huffman para codificar as diferenças entre duas amostras consecutivas de um conjunto de dados geral, a taxa de compressão obtida em outros conjuntos de dados relativos ao mesmo fenômeno, mas medido em diferentes localizações e períodos de tempo, pode ser muito próxima do limite teórico e muito melhor do que aquelas relatadas em [2], [3].

O restante deste artigo está organizado como segue. Na Seção II é definido o problema a ser resolvido. Na Seção III é descrito o algoritmo LEC, enquanto na Seção IV é apresentado o método proposto. Na Seção V serão mostrados os resultados obtidos com a abordagem proposta comparados com os obtidos quando processados pelo algoritmo LEC [2], [3]. Finalmente, a Seção VI conclui o artigo.

II. DEFINIÇÃO DO PROBLEMA

Considera-se um nó sensor monitorando temperaturas cujos valores são números inteiros [1]. Seja o dado adquirido pelo nó sensor no instante i representado depois da conversão analógica-digital pelo símbolo $x_i \in \mathcal{X}$, em que $i = 0, 1, 2, \ldots$. O conjunto $\mathcal{X} = \{\S, \S_\infty \ldots \S_{\mathcal{M}-\infty}\}$ é dito alfabeto fonte.

M. C. Maciel, Instituto Federal do Amazonas (IFAM), Manaus, Amazonas, mcm@ifam.edu.br

H. P. Medeiros, Purdue University, West Lafayette, Indiana, EUA. hmedeiro@purdue.edu

R. D. Souza, Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, Paraná. richard@utfpr.edu.br

M. E. Pellenz, Pontifícia Universidade Católica - Paraná (PUC-PR), Curitiba, Paraná. marcelo@ppgia.pucpr.br

[1] O esquema será apresentado supondo-se temperaturas inteiras, mas ele pode ser facilmente estendido para outras resoluções.

Se cada símbolo $x_j \in \chi$ tem uma representação binária com l_j bits de comprimento, então o número médio de bits usados para representar cada símbolo fonte é dado por $L = \sum_{j=0}^{M-1} p_j \ l_j$, onde p_j é a probabilidade de que $m_i = x_j$. Quando não existe um codificador de fonte, a representação dos símbolos da fonte tem geralmente tamanho $L_u = \lceil \log_2 M \rceil$. O limite teórico para o mínimo número de bits por símbolo para uma fonte discreta é dado pela entropia da fonte $H(\chi)$ [9], sendo:

$$H(\chi) = \sum_{j=0}^{M-1} p_j \ I_j = -\sum_{j=0}^{M-1} p_j \ \log_2(p_j), \qquad (1)$$

onde I_j é a medida de informação para cada símbolo x_j definido por $\log_2(\frac{1}{p_j})$. A eficiência do algoritmo de compressão é usualmente medida pela relação entre o comprimento médio das palavras que representam os símbolos e a entropia da fonte.

Para o projeto do método proposto considera-se o conjunto de medições de temperatura denominado *Set 1* proveniente de medições realizadas em Hargestown, MD, EUA, no período de 01/01/09 à 08/07/11 com 26.843 amostras conforme descrito na Tabela I. As temperaturas inteiras no *Set 1* variam entre $-16°$C e $+37°$C. Portanto, sem compressão necessita-se usar $L_u = 6$ bits/símbolo para representar os 54 símbolos diferentes do alfabeto. A entropia da fonte nesse caso é de $H = 5,29$ bits/símbolo. Implementando um código Huffman para essa fonte em particular, após a compressão obtem-se $L = 5,31$ bits/símbolo. Contudo, nota-se que não se obtém um bom ganho fazendo-se esta compressão, apenas consegue-se uma redução do tamanho médio do símbolo de 0,69 bits ou seja $11,5\%$ em relação ao caso não codificado. Isso ocorre porque a distribuição de probabilidade dos valores de temperatura é relativamente uniforme, enquanto que um esquema de compressão como o Huffman tem maior impacto no caso de uma distribuição fortemente não uniforme.

O desempenho pode melhorar se, como em [2], [3], forem consideradas as diferenças das medições consecutivas de temperatura. Por exemplo, no caso do *Set 1* a entropia das diferenças de temperatura é somente $H_d = 2,13$ bits/símbolo, tendo neste caso uma redução de $59,7\%$ comparado com a entropia das temperaturas do mesmo *Set 1*. Tal redução é devido ao fato de que a distribuição de probabilidade das diferenças é fortemente não uniforme. Assim, é muito mais promissor considerar a compressão das diferenças consecutivas de temperatura do que a compressão das próprias temperaturas. Tal fato foi explorado em [2], [3], onde os autores comprimem as diferenças usando um esquema similar à compressão JPEG de coeficientes DC para imagens digitais.

Aplicando o método de [2] para as temperaturas do *Set 1* obtém-se $L = 3,24$ bits/símbolo, uma redução de $46,0\%$ considerando o valor original de 6 bits/símbolo necessário para o *Set 1*, mas ainda $18,5\%$ a mais do que o limite teórico dado pela entropia das diferenças de temperaturas ($H_d = 2,13$ bits/símbolo). O limite teórico pode ser aproximado pelo código de Huffman, produzindo $L = 2,16$ bits/símbolo, uma redução de $64,0\%$ considerando o valor original de 6 bits/símbolo. Mesmo assim, note que o esquema proposto em [2] usa um dicionário fixo, que pode ser aplicado em qualquer fonte. Uma técnica de codificação por entropia como Huffman tem que ser combinada com a distribuição da fonte para atingir um desempenho otimizado [9]. No entanto, existe um problema de causalidade, pois não se pode conhecer a distribuição exata *a priori*. Além disso, o alfabeto de Huffman calculado para uma dada fonte pode não ter um bom desempenho se usado para comprimir outra fonte com distribuição diferente. A fim de contornar esse problema pode-se fazer uso da técnica de codificação Huffman adaptativa (ou dinâmica) [10]. A principal desvantagem dessa abordagem é que ela requer uma manutenção no dicionário a cada nova conexão entre os pares de nós vizinhos [7]. Devido à severa restrição de memória nos nós sensores sem fio, esse método é impraticável no nosso contexto que considera nós sensores com arquitetura bastante simples.

III. O ALGORITMO LEC

O algoritmo LEC (do inglês *Lossless Entropy Compression*) proposto por Marcelloni e Vecchio [2], [3] explora uma versão modificada do código Golomb-exponencial [11] de ordem zero, que é uma espécie de código universal. A ideia básica é dividir o alfabeto em grupos de tamanho com crescimento exponencial. Cada palavra código é formada por um código unário e um binário: em particular, o código unário (um código de tamanho variável) especifica o grupo, enquanto o código binário (código de tamanho fixo) representa a indexação do grupo. No LEC, mantém-se a abordagem da divisão do alfabeto em grupos de tamanho incrementado exponencialmente, mas os grupos são codificados por entropia, ao invés de um código unário. Essa modificação introduz a possibilidade de especificar códigos livre de prefixo para os grupos.

Cada medida m_i adquirida pelo sensor é convertida para uma representação binária com l_i = R bits, onde R é a resolução do conversor analógico digital (ADC). Para cada nova aquisição m_i, o LEC processa as diferenças $d_i = m_i - m_{i-1}$ (diferença entre duas medições consecutivas), que são entradas para um codificador por entropia. O codificador por entropia executa uma compressão sem perdas das diferenças d_i e codifica-as com base nas suas estatísticas. Cada valor d_i diferente de zero é representado por uma sequência de bits bs_i composta de duas partes $s_i|a_i$, onde s_i codifica o número n_i de bits necessários para representar d_i (que é o grupo ao qual d_i pertence) e a_i que é a representação de d_i (o índice de posição no grupo). Quando d_i é igual a 0, o grupo correspondente tem tamanho igual a 1 e por isso não existe a necessidade de codificar o índice de posição no grupo, isso significa que a_i não é representado.

Para d_i diferente de zero, n_i é processado como $\lceil \log_2(|d_i|) \rceil$, sendo n_i no máximo igual a R. Assim, para codificar n_i uma tabela de $R + 1$ entradas livres de prefixo é especificada. Essa tabela depende da distribuição das diferenças d_i: diferenças com maior frequência estão associadas a códigos pequenos. Geralmente em dados coletados por RSSFs, verifica-se que as diferenças com maiores frequências estão próximas de 0. Assim, para evitar o custo de processar as frequências dos nós sensores, adota-se a Tabela II em que

TABELA I
PRINCIPAIS CARACTERÍSTICAS DOS CONJUNTOS DE TEMPERATURAS

Localização	Nome	Range (°C)	Qtd. Amostras	Data (dd/mm/aa)	Taxa Amost.
Hagerstown, MD, USA [14]	Set 1	-16 a +37	26.843	01/01/09 a 08/07/11	10 min
Manaus, AM, Brasil [14]	Set 2	+21 a +36	3.676	01/07/11 a 30/11/11	60 min
Jonesboro, AR, USA [14]	Set 3	-1 a +41	3.738	01/07/11 a 20/11/11	60 min
Le Génépi, Suíça [15]	Set 4	-11 a +16	42.141	28/08/07 a 31/10/07	2 min
Morges, Suíça [15]	Set 5	+5 a +29	14.527	06/08/07 a 02/09/07	2 min
Bern, Suíça [15]	Set 6	-14 a +6	4.851	13/03/07 a 15/03/07	0.5 min

TABELA II
DICIONÁRIO USADO NO LEC

n_i	s_i	d_i
0	00	0
1	010	-1, +1
2	011	-3,-2,+2,+3
3	100	-7,...,-4,+4,...,+7
4	101	-15,...,-8,+8,...,+15
5	110	-31,...,-16,+16,...,+31
6	1110	-63,...,-32,+32,...,+63
7	11110	-127,...,-64,+64,...+127
8	111110	-255,...,-128,+128,...,+255
9	1111110	-511,...,-256,+256,...,+511
10	11111110	-1023,...,-512,+512,...,+1023
11	111111110	-2047,...,-1024,+1024,...,+2047
12	1111111110	-4095,...,-2048,+2048,...,+4095
13	11111111110	-8191,...,-4096,+4096,...,+8191
14	111111111110	-16383,...,-8192,+8192,...,+16383

as primeiras 11 linhas coincidem com a tabela usada como base do algoritmo JPEG para compressão de coeficientes DC [12]. Observa-se que se a resolução do ADC é maior do que R = 14 bits, a tabela tem que ser apropriadamente expandida.

Para gerenciar os possíveis valores negativos de d_i, o LEC mapeia as diferenças de entrada para valores não negativos $\widetilde{d}_i$ (valor não negativo de d_i), usando as seguintes relações:

$$\widetilde{d}_i \quad = \begin{cases} d_i, & \text{se} \quad d_i \geq 0, \\ 2^{n_i} - 1 - |d_i|, & \text{se} \quad d_i < 0. \end{cases}$$

Além disso, s_i é o valor correspondente a n_i na tabela do código e a_i é a representação binária de $\widetilde{d}_i$ através de n_i bits.

O procedimento usado para gerar a_i garante que todos os possíveis valores tenham códigos diferentes. Sendo $m_1 = 30$ e $m_0 = 27$ por exemplo, tem-se $d_1 = m_1 - m_0 = 30 - 27 = +3$, então: $n_1 = \lceil \log_2(|d_1|) \rceil = 2$. Na Tabela II com $n_1 = 2$ temos $s_1 = 011$, e com $d_1 = +3$ tem-se $\widetilde{d}_1 = +3$, obtém-se $a_1 = 11$ que é a representação binária de $\widetilde{d}_1$ através de n_1 bits, logo a codificação $bs_1 = s_1|a_1 = 011|11$. Considera-se agora $m_2 = 18$, sendo assim $d_2 = m_2 - m_1 = 18 - 30 = -12$, então: $n_2 = \lceil \log_2(|d_2|) \rceil = 4$. Na Tabela II com $n_2 = 4$ temos s_2 = 101, e com $d_2 = -12$ tem-se $\widetilde{d}_2 = 2^{n_2} - 1 - |d_2| = 2^4 - 1 - |-12|$ = 3 , obtêm-se $a_2 = 0011$ que é a representação binária de $\widetilde{d}_2$ através de n_2 bits, logo a codificação $bs_2 = s_2|a_2 = 101|0011$. Uma vez gerado bs_i, ele é concatenado ao *bitstream* que forma a versão comprimida da sequência de medidas m_i [3].

IV. MÉTODO PROPOSTO

Depois de comparar a distribuição de probabilidade das temperaturas e das diferenças consecutivas de temperatura para conjuntos de medidas realizadas em diferentes localidades, foi notado que as distribuições das diferenças eram bastante similares para todos os conjuntos, apesar da distribuição dos valores de temperatura variarem significativamente. Isso pode ser observado na Figura 1, que mostra a distribuição de probabilidade das diferenças entre medições consecutivas para cada conjunto de dados da Tabela I. Como mostra a figura, todas as distribuições são aproximadamente normais com média zero. Assim, a princípio um alfabeto de Huffman fixo, projetado especificamente para um dos conjuntos de dados, pode ter bom desempenho para outros com características semelhantes.

Neste trabalho propõe-se a construção de um alfabeto Huffman fixo obtido pela aplicação do algoritmo de Huffman para um dado conjunto de medidas de temperaturas. Considerou-se o *Set 1* como nosso conjunto de referência, sem nenhuma razão particular além do fato de que o número de amostras e a faixa de temperaturas medidas são grandes. Esse alfabeto mostrado na Tabela III, é então usado para comprimir outros dados reais de temperatura presentes nos *Sets* da Tabela I. Como o alfabeto é fixo, a complexidade desta abordagem é baixa, sendo não mais complexa do que o apresentado em [2]. Por exemplo, uma implementação de codificação e decodificação Huffman para um Microcontrolador AVR, usado num nó sensor, utiliza somente 468 bytes de memória de programa [13].

TABELA III
ALFABETO DE HUFFMAN PROPOSTO

d_i	**Códigos**
-10	0010101000101110
-9	00101010001010
-8	001010100010110
-7	001010100011
-6	001010100111
-5	00101010010
-4	001010101
-3	0010100
-2	00100
-1	01
0	1
+1	000
+2	0011
+3	001011
+4	00101011
+5	00101010000
+6	001010100110
+7	00101010001001
+8	00101010001000
δ	0010101000101111

No esquema de compressão proposto, como em qualquer um baseado na compressão de diferenças, dois casos especiais devem ser considerados: i) A primeira amostra m_0 será trans-

mitida sem compressão já que não existe valor anterior para computar a diferença d_0. A partir da segunda amostra coletada teremos as diferenças $d_i = m_i - m_{i-1}$; ii) A Tabela III possui um alfabeto limitado de diferenças, consequência dos dados disponíveis no conjunto de referência. Caso ocorra um símbolo não presente no dicionário, uma solução viável é tratá-lo como uma medição inicial e enviá-lo sem compressão. Em ambos os casos acima, um símbolo sem compressão pode ser transmitido enviando-se primeiramente um marcador especial presente no dicionário de Huffman denotado por δ.

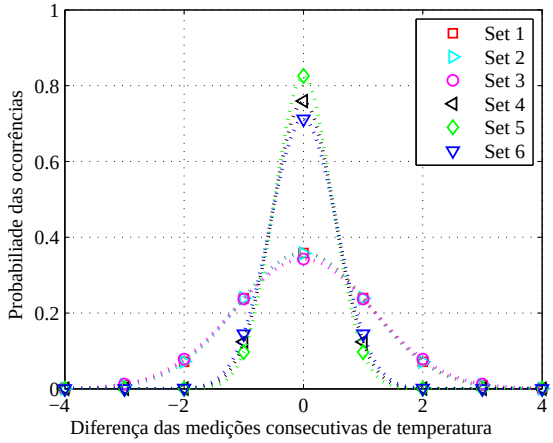

Fig. 1. Distribuição das probabilidades de ocorrência das diferenças

V. Resultados

Nessa seção investigou-se o desempenho do esquema proposto, quando o alfabeto de Huffman fixo da Tabela III é usado para comprimir os conjuntos de temperaturas da Tabela I. Note que, como mostrado na Tabela I, os conjuntos de dados em teste, *Set 2* até *Set 6*, foram coletados em diferentes locais e datas do *Set 1*, que foi usado como referência para construir o alfabeto da Tabela III. O desempenho do esquema proposto é comparada com o limite teórico dado pela entropia da fonte (considerando ambas temperaturas e diferenças de temperaturas) e com a performance do LEC [2], [3].

A Figura 2 mostra as entropias (H) das medições e das diferenças consecutivas (H_d), e o tamanho médio do símbolo sem compressão (L_u), para cada conjunto de dados mostrado na Tabela I. A Figura 3 mostra o tamanho médio por símbolo (L) depois da compressão, a taxa de compressão (C_r) e a eficiência do código (η) usando a abordagem proposta, bem como para o LEC. A taxa de compressão é calculada como:

$$C_r = 100 \times \left(1 - \frac{L}{L_u}\right) \%, \tag{2}$$

enquanto a eficiência do código em relação ao respectivo limite teórico é dada por:

$$\eta = 100 \times \left(\frac{H_d}{L}\right) \%. \tag{3}$$

Como o resultado demonstra, o método proposto não só supera o LEC mas também atinge uma alta taxa de compressão. Além disso, a eficiência do código no método proposto se aproxima de 100 % para alguns conjuntos de dados (*Sets 2, 3 e 6*). Isso significa que apesar de se usar um dicionário fixo, o esquema proposto atingiu um excelente desempenho para os outros conjuntos de dados. Ainda pode-se observar que a redução do tamanho médio dos símbolos foi de aproximadamente 50% comparado com o LEC.

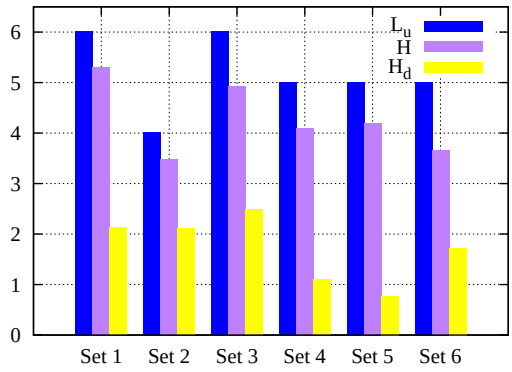

Fig. 2. Tamanho médio do símbolo (L_u) sem compressão, entropia das temperaturas medidas (H) e entropia das diferenças consecutivas de temperaturas (H_d), todos expressos em [bits / símbolo]

A. Validação do método usando dicionários alternativos

Investigou-se também qual seria o impacto no uso dos demais *Sets*, além do *Set 1*, como referência na geração do dicionário fixo. Na Tabela IV é apresentado o valor dos tamanhos médios dos símbolos (L) aplicando o método proposto utilizando como referência os *Sets* de cada coluna aplicados aos *Sets* por linha. Na última linha apresenta-se o tamanho médio dos símbolos considerando todos os *Sets* e usando um dicionário baseado no *Set* do topo da coluna. Foi observado que os *Sets 1,2,4,5 e 6* reproduziram valores de tamanho médio dos símbolos com certa uniformidade, apenas o *Set 3* reproduziu valores de tamanho médio dos símbolos com valores 17% acima dos outros, conforme observado na linha que representa as médias das colunas. Isso ocorre pois este *Set* possui o menor valor de probabilidade de ocorrência da média zero, o que gerou códigos maiores para representação dos símbolos. Apesar das variações registradas, nenhuma das referências levou a valores maiores de L comparado ao LEC.

Na Tabela V é apresentado o impacto da inserção de marcadores especiais proveniente da geração de símbolos não presentes no dicionário Huffman fixo gerado pelo conjunto de dados de referência. Nas colunas temos os conjuntos de referência e nas linhas os impactos correspondentes a cada conjunto comprimido, ou seja, o percentual de marcadores especias que seriam necessários no bloco de dados para transmissão. Pode-se constatar que os piores casos ocorrem quando os *Sets 4 e 5* são as referências na geração do dicionário Huffman, fazendo com que exista uma ocorrência um pouco maior de símbolos fora do dicionário. Mas mesmo assim a ocorrência de símbolos fora do dicionário é pequena, causando pouco impacto na fase de compressão. Assim, conclui-se que o desempenho do esquema não é fortemente restrito à escolha do conjunto de referência.

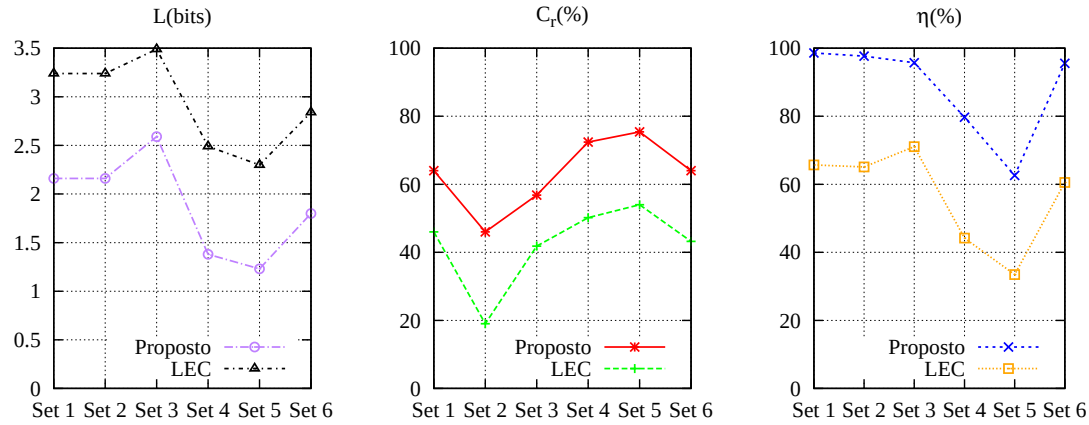

Fig. 3. Tamanho médio do símbolo após a compressão (L), taxa de compressão (C_r) e eficiência do código (η) para os diferentes conjuntos de temperatura.

TABELA IV
TAMANHO MÉDIO DOS SÍMBOLOS APÓS COMPRESSÃO (L) NO CRUZAMENTO ENTRE AS BASES DE DADOS

	Set 1	***Set 2***	***Set 3***	***Set 4***	***Set 5***	***Set 6***
Set 1	2,16	2,17	2,31	2,19	2,25	2,20
Set 2	2,16	2,15	2,31	2,17	2,21	2,19
Set 3	2,59	2,59	2,56	2,58	2,72	2,65
Set 4	1,38	1,38	2,02	1,38	1,38	1,38
Set 5	1,23	1,23	2,00	1,22	1,22	1,22
Set 6	1,80	1,80	2,19	1,78	1,83	1,80
Média	1,89	1,89	2,31	1,89	1,94	1,91

TABELA V
IMPACTO DA INSERÇÃO DE MARCADORES ESPECIAIS PROVENIENTE DA GERAÇÃO DE SÍMBOLOS NÃO PRESENTES NO DICIONÁRIO HUFFMAN FIXO GERADO PELO CONJUNTO DE DADOS DE REFERÊNCIA(%)

	Set 1	***Set 2***	***Set 3***	***Set 4***	***Set 5***	***Set 6***
Set 1	0,00	0,03	0,03	0,79	1,77	0,03
Set 2	0,00	0,00	0,03	0,98	1,17	0,03
Set 3	0,00	0,03	0,00	2,84	3,18	0,03
Set 4	0,00	0,00	0,00	0,00	0,01	0,00
Set 5	0,00	0,00	0,00	0,04	0,00	0,00
Set 6	0,00	0,00	0,00	0,80	1,05	0,00

VI. CONCLUSÕES

Este artigo apresenta um mecanismo de compressão sem perda que utiliza um dicionário fixo de Huffman. O esquema proposto tem uma baixa complexidade computacional, sendo facilmente implementado na prática. Foram calculadas as taxas de compressão obtidas de vários conjuntos de dados coletados em diferentes localidades e durante períodos de tempo distintos. As taxas de compressão obtidas usando a abordagem proposta variaram entre 46% e 75%. Finalmente, o esquema proposto, extremamente simples, superou o método [2], [3] para todos os conjuntos de dados considerados.

No futuro pretende-se investigar de forma mais aprofundada a relação entre a taxa de amostragem e a faixa de valores contidos nos conjuntos de dados com o propósito de desenvolver técnicas que permitam amenizar a variabilidade do desempenho do método proposto. Seria possível, por exemplo, evitar o pior desempenho quando o *Set 3* é utilizado para gerar o dicionário se estivesse disponível previamente a informação de que a dispersão dos valores é maior do que a esperada.

AGRADECIMENTOS

Este trabalho foi parcialmente financiado pela CAPES, CNPq e FAPEAM.

REFERÊNCIAS

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A Survey on Sensor Networks,” *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102–114, Aug. 2002.
[2] F. Marcelloni, M. Vecchio, ”A Simple Algorithm for Data Compression in Wireless Sensor Networks,” *IEEE Communications Letters*, vol. 12, no. 6, pp. 411-413, June 2008.
[3] F. Marcelloni, M. Vecchio, ”An Efficient Lossless Compression Algorithm for Tiny Nodes of Monitoring Wireless Sensor Networks”, *The Computer Journal*, vol. 52, no. 8, pp 969-987, April 2009.
[4] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow, D. Estrin, ”Lightweight Temporal Compression of Microclimate Datasets,” *In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pp. 516- 524, 16-18 Nov. 2004.
[5] C. M. Sadler and M. Martonosi, ”Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks”, *In Proceedings of the 4th international conference on Embedded networked sensor systems (SenSys ’06)*. pp. 265-278, ACM, New York, NY, USA, 2006.
[6] A. Reinhardt, M. Hollick, R. Steinmetz, ”Stream-oriented Lossless Packet Compression in Wireless Sensor Networks”, *In Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. SECON ’09*, pp. 1-9, Jul. 2009.
[7] A. Reinhardt, D. Christin, M. Hollick, J. Schmitt, P. Mogre, R. Steinmetz, J. Silva, B. Krishnamachari, F. Boavida, ”Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks” *Wireless Sensor Networks*, Springer Berlin, vol. 5970, pp. 33-48, 2010.
[8] E. P. Capo-Chichi, H. Guyennet, J.-M. Friedt, ”K-RLE: A New Data Compression Algorithm for Wireless Sensor Network”, *In Proceedings of the Third International Conference on Sensor Technologies and Applications, 2009. SENSORCOMM ’09*, pp. 502-507, Aug. 2009.
[9] K. Sayood, *Introduction to Data Compression*, Third Edition, Morgan Kaufman, 2005.
[10] J. S. Vitter, ”Design and Analysis of Dynamic Huffman Codes,” *Journal of the ACM*, vol. 34, no. 4, pp. 825-845, Oct. 1987.
[11] S. W. Golomb, ”Run-length Encodings”, *IEEE Transactions Information Theory*, vol. 12, pp. 399-401, 1966.
[12] W. Pennebaker, J. Mitchell, ”JPEG Still Image Data Compression Stardard”, *Kluwer Academic Publishers*, 1992.
[13] AVR. [Online]. http://www.das-labor.org/wiki/AVR–Huffman/en
[14] Weather Underground [Online]. http://www.wunderground.com
[15] SensorScope. [Online]. http://sensorscope.epfl.ch