16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

ROBUST TRANSMISSION OF HTML FILES: ITERATIVE JOINT SOURCE-CHANNEL DECODING OF DEFLATE CODES

Zied Jaoua[*+], *Anissa Mokraoui-Zergaïnoh*[+], *Pierre Duhamel*[*]

[*]LSS/CNRS, SUPELEC, Plateau de Moulon, 91 192 Gif sur Yvette, France
[+]L2TI, Institut Galilée, Université Paris 13, Avenue Jean Baptiste Clément, 93 430 Villetaneuse, France
{jaoua,duhamel}@lss.supelec.fr, anissa.mokraoui@galilee.univ-paris13.fr

ABSTRACT

This paper considers the transmission problem over a noisy mobile radio channel of encoded html files according to the http1.1 protocol specifications. The deflate algorithm encodes the html files using the combination of two entropy coding algorithms: Lempel-Ziv and Huffman. The developed receiver is based on an iterative joint source channel decoding approach using the turbo principle built from two serial concatenated codes. The proposed Soft-Input Soft-Output inner decoder is based on the traditional sequential M-algorithm which is modified in order to improve the performance of the decoder. The proposed algorithm exploits the specific grammatical rules of Huffman codes, Lempel-Ziv codes and the syntax of the html language. This decoder is combined to a Soft-Input Soft-Output channel decoder of convolutional codes. Simulation results, over an additive white gaussian noise channel, show that the proposed receiver reduces the number of errors occurring in the transmitted html file compared to any conventional channel decoding.

1. INTRODUCTION

Wireless Internet is being used more and more in many applications. A huge amount of data is handled such as to transfer or to download html pages from a web server to a mobile client. In the recent http release standard, i.e. http1.1 ([1]), an additional functionality is proposed to encode the html (or xml) files to be downloaded or transferred before their transmission over the communication channel. The specified encoding method is the deflate method which is the combination of some variant of Lempel-Ziv (LZ) plus Huffman entropy coding algorithms. The deflate codes are then encapsulated according to the specifications given either in Gzip ([2]) or Zip ([3]) standards.

Since the deflate bitstream is represented by variable length codes, a loss or an alteration of a single bit in the received bitstream leads to propagation of errors and synchronization losses which could be catastrophic for the decoded html file. The lack of error resilience in the deflate codes remains a real problem since, to our knowledge, no investigation has been devoted to this topic.

Compared to robust Huffman codes, only few papers have been developed in the framework of robust LZ codes. The authors of reference [4] proposed a joint source channel coding algorithm to protect the universal Lempel-Ziv-77 (LZ-77) codes against a number of errors. They modified the traditional LZ-77 encoder in order to organize the inherent remaining redundancy (even if this redundancy is small) in such a way as to achieve error-resilience. Their approach requires specific encoders and decoders. In our previous work, we proposed a joint source channel decoding approach of LZ-77 codes ([5]). However the html files are compressed using a different variant of LZ-77 algorithm, meaning that some investigations are required to propose an efficient error resilience method according to the specified version of the LZ codes described in the http1.1 standard.

The problem of decoding Huffman codes transmitted via a noisy communication channel has received an increasing attention during the recent years, because the Huffman entropy coding algorithm is often applied as a final step for source compression in many multimedia applications. At the same time, the interest for joint source channel coding and decoding has considerably increased ([6], [7]). In joint source channel decoding approach, the Huffman code structure is exploited in order to perform efficient decoding of the corrupted bitstream ([7]). The Most developed Huffman decoding algorithms take advantage of the encoders properties. Indeed, the Huffman encoding operation is generally modelled as a discrete time finite-state Markov process. In our specific context, the situation is more complex since the Huffman coding algorithm is applied directly on the LZ codes. Unfortunately the modelling of the source by the Markov process is not sufficient. The decoding problem becomes more complicated.

The main objective of this paper concerns the protection of the deflate codes against transmission errors over a mobile channel. This paper is organized as follows. The next section introduces the deflate entropy coding algorithm. Section 3 presents the proposed Soft-Input Soft-Output decoding algorithm. Section 4 proposes a receiver based on an iterative joint source channel decoding approach. Section 5 provides simulation results over an additive white noise channel.

2. DEFLATE ENTROPY CODING ALGORITHM

This section presents the deflate entropy coding algorithm imposed in the http1.1 standard. The deflate is the combination of two well known entropy coding algorithms: a variant of the universal LZ-77 and Huffman algorithms. This section begins with the outline of the LZ algorithm adopted by the http1.1.

2.1 LZ entropy coding algorithm in deflate

A large family of LZ coding algorithms has been proposed. This paper focuses on the LZ algorithm implemented in the http1.1 protocol . This version is closely related to the LZ-77 universal lossless encoding algorithm developed in [5].

The LZ coding algorithm exploits the redundant nature of the html (xhtml, xml) file considered as an ASCII document as follows. Denote by T the html text of length n over a finite alphabet. The i-th symbol in T is denoted by $T[i]$. $T[i,j]$ represents the substring composed by the following symbols $T[i]T[i+1]T[i+2]...T[j]$. The LZ algorithm parses the text sequentially left to right and processes the data on line as it is read in order to adaptively build a dictionary.

Suppose that $i-1$ symbols have been parsed providing $h-1$ substrings composing the dictionary denoted by

Thanks to French National Research Agency for funding this work under the DITEMOI Project http://rnrt.ditemoi.free.fr/

$T[1, i-1] = s_1 s_2 ... s_{h-1}$, where s_k represents the k-th substring. At this step, the algorithm searches in the dictionary (i.e. $T[1, i-1]$) the largest h-th string that matches with the longest string available in $T[i, i+l_h-1]$ with $l_h \leq L$ where L is the fixed size of the search window. If some match is selected during the encoding process, the retained substring is encoded using a codeword denoted by $< p_i, l_i >$ where p_i is the pointer toward the dictionary indicating the beginning of the substring to be encoded and l_i the length of the new substring to be included in the dictionary. Otherwise, if no match occurred, the codeword is given by $< c_j >$ corresponding to the symbol $T[j]$ to be included in the dictionary. Therefore the text to be encoded by LZ is represented by two kinds of codewords as follows: $..., < p_i, l_i >, ..., < c_j >, ...$ In the deflate normalization ([8]), the size of the dictionary and the search window are respectively fixed to 256 bytes and 32 K-bytes. Consequently the words p_i, l_i and c_i are respectively encoded on 15 bits (i.e. $p_i = p_i^{14}...p_i^k...p_i^0$), 8 bits (i.e. $l_i = l_i^7...l_i^k...l_i^0$) and 8 bits (i.e. $c_j = c_j^7...c_j^k...c_j^0$).

2.2 Huffman entropy coding algorithm in deflate

Deflate offers three different operating modes ([8]). The first one concerns the segmentation of either compressed or uncompressed big file. This mode is not considered in this paper. Two other modes are proposed in order to compress the LZ codewords. The main difference between these modes is that one is sometimes more efficient in terms of compression ratio than the other one but is much more time consuming.

The sophisticated compression method, referenced as mode 3 in deflate, proposes the construction of two dynamic Huffman code tables according to a specific reorganization of the traditional Huffman tree structure. These tables are adapted to the statistics of the LZ codewords. Moreover, the Huffman tables are also compressed with entropy run length encoding algorithm and then inserted in the compressed LZ bitstream. For more details one can refer to [8]. The second compression mode, referenced as mode 2 in deflate standard, proposes two predefined static Huffman code tables. These tables are directly used by the deflate encoder and decoder. This strategy speeds up both the compression and decompression processes. Moreover, no Huffman table need to be transmitted to the deflate decoder, since they are predefined. Obviously, the cost to be paid is a (hopefully slightly) decreased efficiency of the compression, incase the predefined tables do not match the actual source statistics. The first table contains 256 "edoc" which are used to encode the LZ lengths and characters. For a given "edoc" in the table, the Huffman code is constructed by the concatenation of the corresponding prefix and suffix codes leading to variable length codes. For LZ length, the Huffman codeword can be encoded between 8 and 13 bits, while the Huffman LZ characters are encoded between 8 and 9 bits. The second table is designed to encode LZ pointers. The table contains 32768 "edoc". For each LZ pointers, the table associates an "edoc". From this "ecdoc", the Huffman codeword is constructed using the concatenation of the corresponding prefix code written on 5 bits, and its suffix code of at most 13 bits.

As first step in the robust transmission of deflate codes, this paper concentrates on the second deflate mode given in the standard. The main reason is a good tradeoff between simplicity and efficiency: its implementation is not complex and is much faster than the third mode, while its performance in terms of compression ratio is reasonable compared to mode 2. In the 4 examples of html files that are processed in the simulation section (standard examples from the real world), the average compression ratio for mode 3 is 2.95, while in mode 2 it is 2.5. It is clearly seen that mode 2 is still efficient, while its complexity is largely reduced. Furthermore, it is likely that robustification of mode 3 towards transmission errors is very difficult. Its use when transmitting files to mobiles remains also to be seen, due to lack of robustness and computational load.

3. SISO INNER DECODER BASED ON MODIFIED SEQUENTIAL DECODING M-ALGORITHM

This section focuses on the Soft-Input Soft-Output (SISO) inner decoder used by the developed iterative receiver. The proposed SISO decoder, denoted SD-deflate, is based on a modified version of the traditional decoding M-algorithm. This SISO decoder processes in two main steps. The first one estimates the deflate sequence using a modified version of the generic sequential decoding M-algorithm and the second one derives the soft values which will be discussed in the next section. First briefly describe the principe of the traditional sequential decoding M-algorithm.

Denote by $\mathbf{x} = (x_1, ..., x_t, ..., x_N)$ the sequence to be transmitted over an additive white gaussian noise (AWGN) channel and $\mathbf{y} = (y_1, ..., y_t, ..., y_N)$ the sequence of the received symbols. The M-algorithm, processes on a tree structure using at each depth of the tree the best M memorized paths. This algorithm belongs to the breadth first approaches ([9]). At each iteration, the best M memorized paths at the same depth in the tree are extended one step forward. For each extended path, the cumulated metric based on the maximum likelihood criterion is computed as follows:

$$\widehat{\mathbf{x}} = \arg\max_{x} \log(P(\mathbf{y}/\mathbf{x})) \qquad (1)$$

where $P(\mathbf{y}/\mathbf{x})$ is the maximum likelihood. The channel being memoryless, the maximum likelihood becomes:

$$P(\mathbf{y}/\mathbf{x}) = \prod_{k=1}^{N} P(y_k/x_k) = \prod_{k=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y_k - x_k)^2}{2\sigma^2}) \qquad (2)$$

where σ^2 represents the variance of the zero-mean Gaussian white noise. The estimated sequence is thus given by:

$$\widehat{\mathbf{x}} = \arg\min_{x} \sum_{k=1}^{N} (x_k - y_k)^2 \qquad (3)$$

where $\mu(N) = \sum_{k=1}^{N} (x_k - y_k)^2$ corresponds to the cumulated metric allowing to find the best path. Among these paths, only M best paths are selected in accordance with the cumulated metric. When the algorithm reaches the maximum depth, the best stored path in terms of cumulated metric is considered as the best sequence estimate.

3.1 Previous works oriented to robust LZ-77 codes

The sequential decoding M-algorithm, as briefly described above, does not take into account any property the decoded sequence should meet. However, sequential algorithms easily allow to restrict the sequences to those meeting a given syntax or set of rules. This restriction allows to reduce not only the computational complexity but also to improve the decoding efficiency. Indeed, in reference [10], the authors exploit the LZ-77 grammatical rules and the syntax of the html language which are combined altogether during the decoding step of the received sequence. In the proposed strategy, the modified algorithm chooses the M best sequences, only after having read all the elements corresponding to one LZ-77 codeword such as length, pointer or character code. If the conditions for valid LZ-77 sequences are not met, the corresponding branches are dropped from the tree. Among the remaining ones, those which do not correspond to valid html syntax are also dropped. Simulation results showed that the performance is drastically improved compared to a generic sequential decoding M-algorithm applied to LZ-77 files.

Based on this strategy, this paper concentrates on the modifications to be included in the above approach for applying M-algorithm to the concatenation of LZ plus Huffman met in deflate. The difficulty lies in the existence of Huffman codes, which do not allow to progress by blocks which have some signification in terms of syntax, thus preventing efficient pruning of the trees.

3.2 Modified sequential decoding M-algorithm

The proposed sequential decoding M-algorithm (Fig. 1), denoted SD-deflate, parses from left to right the received bitstream and checks the grammatical rules of Huffman and LZ, plus the syntax rules of the html language. In addition to the metric criterion, if the grammatical and syntax rules are not satisfied, the SD-deflate drops the corresponding branches from the tree structure. The proposed grammatical rules to be checked are listed as follows:

- The parsed deflate codes must be Huffman codes;
- The decoded LZ pointer does not exceed the number of symbols already read and inserted in the dictionary;
- The LZ length code is followed by a LZ pointer code;
- The decoded LZ character must belong to the specific range reserved to the ASCII codes;
- In addition, to these grammatical rules, the syntax of the decoded html text is progressively checked according to the specifications provided in the SGML [11].

According to these rules, the modified sequential M-algorithm processes as follows.

1. For each received information y_k, extend the M current branches of the tree until building $2^\tau M$ paths, where τ is a parameter which is tuned for a good tradeoff between complexity and performance of the algorithm;
2. Among these $2^\tau M$ paths, drop the branches which do not satisfy the Huffman grammatical rules using the provided Huffman tables;
3. For each retained branch, analyse the decoded Huffman bitstream as follows:
   (a) if the number of the decoded Huffman elements, is not sufficient but some bits seem to be part of a valid LZ prefix of any pointer codeword, then memorize the branch corresponding to the beginning of the prefix and do not take any decision at this stage, then follow the process described in step 7. Otherwise go to the following step.
   (b) if the number of the decoded Huffman elements, is sufficient to consider that the decoded codeword is a LZ candidate codeword then check the LZ grammatical rules and take a decision.
   If this LZ code is not valid drop the branch and go to 4. Otherwise continue the html syntax verification. If the syntax is not valid drop the corresponding branch and go to 4.
   (c) if the number of the decoded Huffman elements, is not sufficient to consider that the decoded codeword is a LZ candidate codeword, go to 1.
4. Compute the cumulated metrics of all retained branches, i.e. those satisfying the grammatical rules of the codes and the html syntax;
5. Sort these branches according to their metrics;
6. Select the M best paths according to the metrics then go to step 1;
7. Among the different LZ prefix pointers previously located and memorized, extend again each retained branch from the last detected prefix pointer using at this time 5 bits (i.e. 5 received information) since it corresponds to the number of bits required to represent any prefix pointer as specified by the deflate standard;
8. Compute the cumulated metrics of all retained branches;
9. Sort these branches according to their metrics;
10. Select the M best paths according to the metrics then go step 1.

The performance of the SD-deflate is closely related to the choice of the parameters τ and M. If these parameters take large values, the performance of the algorithm increases. Indeed with higher parameters, the research of the optimal solution tends to an exhaustive research. However, the algorithm becomes much more time consuming. Moreover, if the metric decision is taken at inappropriate depth in the tree, without any consideration, the performance of the SD-deflate may be affected considerably while reducing significantly its computational complexity. A tradeoff between complexity and performance must be considered. For equivalent performance but a reduction of the computational cost of the SD-deflate, a slight modification concerns step 7. Rather than using the last located prefix pointer, we propose to select the first located prefix pointer. At this location, before extending each branch, we add a metric decision reducing then the number of the retained branches. Steps 8, 9 and 10 are removed. Note that all these modifications keep the property of the M-algorithm to compare paths of same lengths only.

4. ROBUST DEFLATE TRANSMISSION BASED ON ITERATIVE JOINT SOURCE CHANNEL DECODING

The communication system developed in this paper is illustrated by Fig.2. Based on this transmission model, the receiver estimates the deflate sequence of the transmitted bits from the sequence of the received symbols. This problem is solved using an iterative approach based on the turbo principle with serial concatenated codes as developed in [12].

Denote $\mathbf{d} = (d_1, ..., d_k, ..., d_N)$ the binary sequence generated by the deflate algorithm where N is the length of the sequence and d_k is the k-th coded bit. The coded bits $\{d_k\}$ are assumed to be uniformly distributed. This deflate binary sequence is first permuted by a pseudo-random interleaver to break the error burst at the receiver, then used as an input to the convolutional encoder with rate $\frac{1}{n}$. The convolutional encoder outputs are then modulated by a BPSK modulator and transmitted over memoryless AWGN channel. The transmitted data stream is denoted as $\mathbf{x}_1^N = (\mathbf{x}_1, ..., \mathbf{x}_k, ..., \mathbf{x}_N)$ where $\mathbf{x}_k = (x_{k,0}, x_{k,1}, ..., x_{k,n-1})$. The received data stream $\mathbf{y}$ is denoted by $\mathbf{y}_1^N = (\mathbf{y}_1, ..., \mathbf{y}_k, ..., \mathbf{y}_N)$ where $\mathbf{y}_k = (y_{k,0}, y_{k,1}, ..., y_{k,n-1})$.

Define two sets $\boldsymbol{R}$ and $\boldsymbol{D}$ of binary sequences $\mathbf{d}$ of length N. The first set $\boldsymbol{R} = \{[d_1, d_2, ..., d_N] \in \{0,1\}^N\}$ contains all possibles sequences, while the second one $\boldsymbol{D} = \{[d_1, d_2, ..., d_N] \in \{0,1\}^N\}$ contains only the sequences satisfying the deflate structure and the syntax of the html language.

The first block corresponds to the channel outer decoder and the second block concerns the source inner decoder (i.e. SISO SD-deflate) described in the previous section (see Fig.2). Each decoding block is fed with soft inputs and can deliver soft outputs. This soft information is exchanged, in an iterative process, between the channel decoder and the source decoder. Classically, this soft information is the so-called *extrinsic probability* of each bit (i.e. the *a posteriori* probability of the bit, divided by its *a priori* probability). The iterative receiver estimates the transmitted message $\mathbf{b}$ so as to iteratively optimize the maximum a posteriori (MAP) criterion within each block, and considering the extrinsic information provided by the other block as an *a priori* probability. Thus, at a given iteration I, the decoding algorithm proceeds in two steps as described below.

4.1 Outer BCJR channel decoder

In the first step, the BCJR channel decoder computes the a posteriori probability $P_R(\mathbf{d}/\mathbf{y})$ for for every $\mathbf{d} \in \mathbf{R}$ (all possible sequences) which is also equivalent to the product of the marginals a posteriori probabilities (APP) $\prod_{k=1}^{N} P^I_{BCJR}(d_k/y_k)$ since the channel transmission is assumed to be memoryless and the $\{d_k\}$ are assumed independent. At any iteration I, the output of the outer BCJR channel decoder is given by the extrinsic probability associated to the coded bit as follows:

$$E^I_{BCJR}(d_k) = K_{BCJR} \frac{P^I_{BCJR}(d_k/y_k)}{P^I_{BCJR}(d_k)} \qquad (4)$$

where K_{BCJR} is the normalization factor such as $E^I_{BCJR}(d_k = 0) + E^I_{BCJR}(d_k = 1) = 1$; $P^I_{BCJR}(d_k)$ is the a priori probability associated to the coded bit corresponding at the interleaved extrinsic information $(E^{I-1}_{DSLZ}(d_k))$ computed by the SD-deflate decoder at the previous iteration (i.e. $I-1$). The de-interleaved extrinsic information $E^I_{BCJR}(d_k)$ is then sent as an a priori input to the SD-deflate decoder.

4.2 Inner SD-deflate source decoder

The second step, related to the inner source decoder SD-deflate, consists in a projection of the distribution of the APP evaluated on $\boldsymbol{R}$, on the set of the APP distribution compatible with the deflate structure codes and the syntax of the html language. The distribution of the required APP, denoted by $P^+_{SDLZ}(\mathbf{d})$, is that which minimizes the Kullback-Leibler distance given by:

$$P^+_{SD-deflate}(\mathbf{d}) = \arg \min_{P_{\boldsymbol{D}}(\mathbf{d})} dist(P_{\boldsymbol{D}}(\mathbf{d}), P_{\boldsymbol{R}}(\mathbf{d}/\mathbf{y})) \qquad (5)$$

It has been shown in [12] that the required APP distribution is given by :

$$P^+_{SD-deflate}(\mathbf{d}) = \begin{cases} \frac{P_{\boldsymbol{R}}(\mathbf{d}/\mathbf{y})}{\sum_{d \in \mathbf{D}} P_{\boldsymbol{R}}(\mathbf{d}/\mathbf{y})} & \text{if } \mathbf{d} \in \mathbf{D} \\ 0 & \text{if } \mathbf{d} \in \mathbf{R}\backslash\mathbf{D} \end{cases} \qquad (6)$$

At any iteration I, the output of the inner source decoder (SD-deflate) is given by the extrinsic probability associated to the coded bit as follows:

$$E^I_{SD-deflate}(d_k) = K_{SD-deflate} \frac{P^I_{SD-deflate}(d_k/y_k)}{P^I_{SD-deflate}(d_k)} \qquad (7)$$

where $K_{SD-deflate}$ is the normalization factor such as $E^I_{SD-deflate}(d_k = 0) + E^I_{SD-deflate}(d_k = 1) = 1$; $P^I_{SD-deflate}(d_k)$ is the a priori probability associated to the coded bit corresponding to the interleaved extrinsic information $(E^{I-1}_{BCJR}(d_k))$ computed by the BCJR decoder at the previous iteration (i.e. $I-1$). The marginal APP is deduced from equation (5) as follows:

$$P^I_{SD-deflate}(d_k/y_k) = \sum_{d_k:0,1} P^+_{SD-deflate}(\mathbf{d}) \qquad (8)$$

The de-interleaved extrinsic information $E^I_{SD-deflate}(d_k)$ is then sent as an input of the BCJR decoder. The SD-deflate decoder focuses on the distribution of the APP that maximize $P^+_{SDLZ}(\mathbf{d})$ which is equivalent to preserve the optimal solutions with respect to the complete sequences and the individual bits as follows:

$$\widehat{d_k} = \arg \max_{d_k:0,1} \log(\sum_{\mathbf{d} \in \mathbf{D}} P^+_{SD-deflate}(\mathbf{d})) \qquad (9)$$

Most of the required work is performed by a slight change in the modified M-algorithm described in section 3: the probabilities computed from the channel outputs are replaced by the extrinsic values provided by the other blocks. As an output, this version provides the set of M best paths which are compatible with the deflate and html syntax. The only additional computation is to marginalize the corresponding probabilities on these M paths rather than on all feasible sequences. Since (hopefully) these M paths are the likeliest, this a rather valid approximation.

Rather than using the soft-outputs, as described in the SOMA algorithm ([13], i.e. max-log-approximation of the LLR), the SD-deflate M-Algorithm estimates the soft-outputs using the LLR computed on all M previously extended paths. This method overcomes the problem of overestimating the LLR avoiding therefore the degradation of the performance of the system,

5. SIMULATION RESULTS

The simulations are carried out on two html files. The first one is a WAP file available in [14] which is a small file of size 413 bytes. It contains 60.5 % of tags compared to the total number of characters in the html file. The second one is a WEB file available in [15] which corresponds to a medium file of size 3031 bytes, containing only 11.5 % of tags compared to the total number of characters in the html file. The performance of the proposed receiver is measured in terms of symbol error ratio (SER) for various (E_b/N_0). Obviously, since we aim at transmitting files to mobiles with small to medium size screens, the file is likely to be not too large in practical situations. Note also that, due to the nature of the files, the percentage of tags increases when the file size decreases. This impacts the efficiency of our method. The simulations are carried out in an environment reminiscent of the IEEE 802.11.a standard using the first operating mode [16], which is intended to be used around 6dB. Therefore, the specified convolutional encoder has the following characteristics: a constraint length equal to 7, with a polynomial generator presented by its octal representation [171,133] and a code rate equal to 1/2.

The simulation results corresponding to the two html files are respectively illustrated by the graphs given by Fig. 3 and Fig. 4. The iterative receiver is implemented with the following values: $M = 10$, $\tau = 5$. Three iterations are performed. Initially, the SISO SD-deflate is implemented as described in section 3. For a given E_b/N_0 equal to $6dB$ (i.e. in the operating zone defined in the IEEE 802.11a), the proposed iterative receiver reduces the SER by a factor 9 compared to the results achieved by a conventional decoder using only a hard decision at the output of the BCJR channel decoder (see Fig. 2). For this particular example, increasing the parameter M to 20 does not improve the performance of the iterative receiver (see Fig. 3). However the computational time increases approximatively by a factor equal to 2. As commented at the end of section 3, the SD-deflate decoder is slightly modified in order to reduce the computational time of the decoding process using also $M = 10$ and $\tau = 5$. For the first html file, the simulation results provided Fig 5, show that around $6dB$, the performance remains unchanged. However the computational time is reduced by a factor of 1/3 compared to the initial version of the SISO SD-deflate decoder.

6. CONCLUSION

This paper proposes an iterative joint source channel decoding approach allowing the error correction of corrupted deflate compressed html pages during their transmission via a noisy mobile channel. The proposed receiver is based on the turbo principle where the inner SISO decoder is based on a modified version of the generic sequential decoding M-algorithm. This new decoding algorithm exploits the grammatical rules of the second specified compression mode given in the deflate standard combined to the syntax rules of the html language. The provided simulation results, in a mobile environment, show that the proposed receiver improves the SER performance without introducing additional redundant information in the transmitted deflate bitstream.

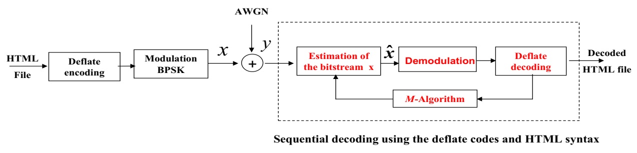

Figure 1: Sequential decoding M-algorithm adapted to the Deflate codes and the syntax of the html language (SD-deflate)

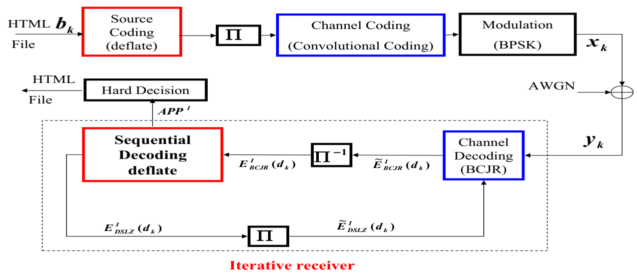

Figure 2: Communication system using iterative joint source channel decoding

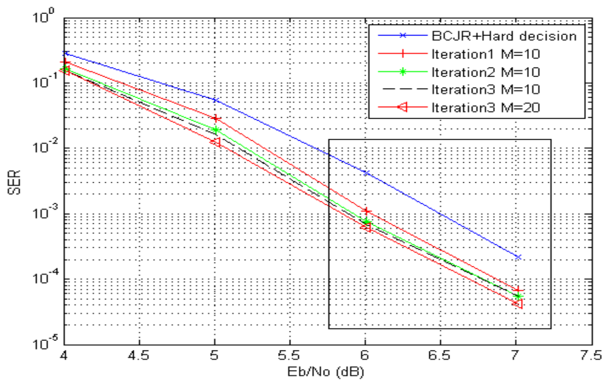

Figure 3: SER performance of the iterative receiver applied to the first html file

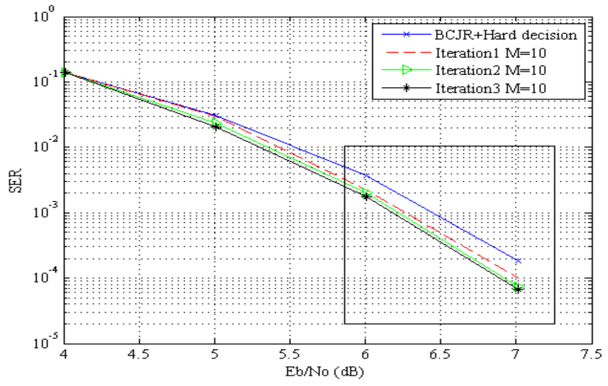

Figure 4: SER performance of the iterative receiver applied to the second html file

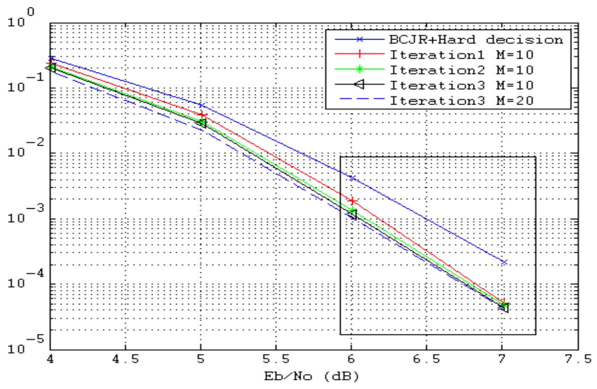

Figure 5: SER performance of the iterative receiver applied to the first html file using modified SD-deflate

REFERENCES

[1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T.Berners-Lee, "Hypertext transfer protocol– http/1.," June 1999.

[2] L.P. Deutsch, "GZIP Compressed data format specification," in *rfc1952*, May 1996.

[3] L.P. Deutsch, "ZLIB Compressed data format specification," in *rfc1950*, May 1996.

[4] S. Lonardi, W. Szpankowski, and M. D. Ward, "Error resilient lz'77 data compression: Algorithms, analysis, and experiments," in *IEEE Transactions on Information Theory 53(5)*, May 2007, pp. 1799–1813.

[5] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," in *IEEE Transactions on Information Theory, 23(3)*, May 1977, pp. 337–343.

[6] V.B. Balakirsky, "Joint source-channel coding with variable length codes," in *Proc. Intl. Conf. Inform. Theory*, 1997, p. 419.

[7] R. Bauer and J. Hagenauer, "Turbo-fec/vlc-decoding and its application to text compression," in *Proc. of the Conference on Information Sciences and Systems*, 2000.

[8] L.P. Deutsch, "DEFLATE Compressed data format specification," in *rfc1951*, May 1996.

[9] J.B. Anderson and S. Mohan, "Source and channel coding: an algorithmic approach," in *Kluwer Academic Publishers,Norwell*, MA 1991.

[10] Z. Jaoua, A. Zergaïnoh-Mokraoui, and P. Duhamel, "Robust transmission of html files : Iterative joint source-channel decoding of lempel ziv-77 codes," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2008.

[11] "http://www.w3.org/markup/sgml/" .

[12] P. Magniez, B. Muquet, P. Duhamel, V. Buzenac, and M. deCourville, "Optimal decoding of bit-interleaved modulations: Theoretical aspects and practical algorithms," in *2nd Intl. Symposium on Turbo Codes and Related Topics*, Sept. 2000, pp. 284–287.

[13] K. K. Y. Wong and P. J. Mclane, "Bi-directional soft-output M-algorithm for iterative decoding," in *IEEE International conference on Communications,2*, Juin 2004, pp. 792–797.

[14] "Samples database of nokia mobile internet toolkit www.forum.nokia.com/tools, worldcup.html" .

[15] "ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html" .

[16] "http://www.ieee802.org/11/" .