IOP Conf. Series: Materials Science and Engineering **231** (2017) 012061 doi:10.1088/1757-899X/231/1/012061

Design of convolutional tornado code

Hui Zhou[1,2], Yao Yang[2], Hongmin Gao[2*], Lu Tan[1]

[1]College of Computer and software, Nanjing Institute of Industry Technology, Nanjing, 210023, China
[2]College of Computer and Information Engineering, Hohai University, Nanjing, 211100, China
*Corresponding Author

Abstract. As a linear block code, the traditional tornado (tTN) code is inefficient in burst-erasure environment and its multi-level structure may lead to high encoding/decoding complexity. This paper presents a convolutional tornado (cTN) code which is able to improve the burst-erasure protection capability by applying the convolution property to the tTN code, and reduce computational complexity by abrogating the multi-level structure. The simulation results show that cTN code can provide a better packet loss protection performance with lower computation complexity than tTN code.

1. Introduction

Tornado code is a type of erasure block code based on irregular sparse graphs. It has linear-time encoding and decoding algorithms and is able to be transmitted over lossy channels at the rate extremely close to capacity [1][2][3]. The tornado code is expected to be promising for applications such as reliable distribution of bulk data over the Internet (including video, news and financial distribution), database replication, and military communications [4]. However, in the burst-erasure environment, such as internet and wireless channel, the traditional tornado (tTN) code has very limited applications because it is highly possible that the burst loss in one block may exceed the error-correction capability of the tTN code in such environment.

This paper proposes a new tornado code, which improves the burst-erasure protection capability by adding the convolution property [5] to the tTN code. The rest of the paper is organized as follows. In Section II, the tTN code is introduced and Section III describes the cTN code in details. Finally, the simulation results are presented in Section IV and Section V draws the conclusion.

2. Traditional tornado code

The tTN is usually described by a multiple-level bipartite graph with a carefully chosen degree sequence. More specifically, a tTN code, $C(B_0, B_1, ..., B_k, L)$, is composed of a cascaded bipartite sub-graph chain $B_0, B_1, ..., B_k$ and a conventional erasure code L as shown in Fig. 1. Each node in the figure represents either a bit or a packet. Without loss of generality, this paper assumes each node is a bit. The sub-graph B_0 takes m message bits as its input and produces βm check bits, which are then fed into B_1 to generate $\beta^2 m$ new check bits. Similarly, the sub-graph B_i, has $\beta^i m$ input bits (from B_{i-1}) and produces $\beta^{i+1} m$ check bits. The conventional erasure code L is then applied to the output bits of the last sub-graph B_k. The final output codeword consists of the m message bits and all

the check bits produced at each stage of the cascade. In each subgraph, the values of the nodes on the right are computed by performing the XOR operation upon their input nodes on the left. At the decoder, the similar XOR operation is performed to recover the nodes of each stage in reverse order.

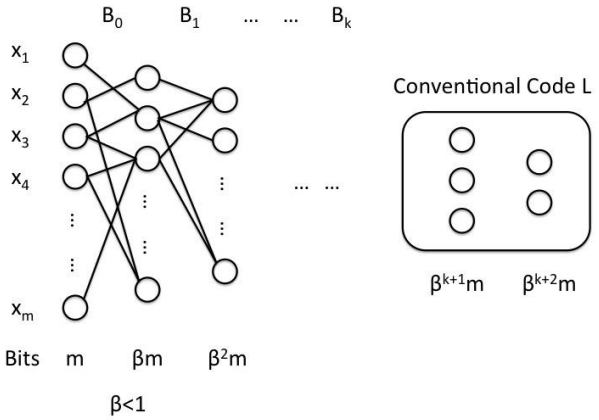

Fig. 1 Encoding Structure of the tTN Code

3. Convolutional tornado code

Although the tTN code has strong error correction capability against random packets loss, its performance suffers under the burst packet loss condition. In order to address this issue, a novel cTN code is proposed to improve the burst-erasure protection capability by incorporating the convolution property and reduce the computation complexity by abrogating the multi-level structure.

Just as its name implies, the proposed cTN code produces the check nodes based on the message nodes from not only the current block but also the previous blocks. The basic idea of the cTN code is to rearrange the order of the check nodes and message nodes in a block, such that all the check nodes will be treated as message nodes for the subsequent encoding. In this way, new check nodes are calculated according to the check nodes obtained previously and message nodes.

The key to cTN code is how to construct the so-called convolution bipartite graph. Fig.2 demonstrates an example of constructing the convolution bipartite graph. At first, the multi-level conventional bipartite graph given in Fig. 1 is simplified to a single-level one shown in Fig. 2(A) for coding convenience. It should be noted that such simplification needs to preserve the code redundancy to maintain a similar correction capability. Next, convolution property is added into the graph by changing all the backward edges to forward edges, such as changing (c_{n+1}^1, x_{n+1}^5) in Fig. 2(B) to (c_{n+1}^1, x_n^5) in Fig. 2(C). A backward edge here means the edge connecting the current check node c and one of its corresponding message node m, while the corresponding forward edge connects c and m' which locates the same position in the previous block as m. At last, the order of the check nodes and the message nodes are rearranged, that is, all the check nodes are uniformly inserted into the message node sequence to involve them in the subsequent coding, as illustrated in Fig. 2(D). The rearrangement process helps to enhance the correlation between the check nodes and message nodes, such that the convolution property can be fully applied to compensate the performance degradation caused by the single-level simplification. Please note that the transmission order of the cTN code is different from the tTN code. The tTN code outputs in blocks with the whole message nodes followed by the check nodes, say $x_n^1, x_n^2, x_n^3, x_n^4, x_n^5, c_n^1, c_n^2$ as in Fig. 2(A). In contrast, the cTN code interleaves the message nodes and check nodes together, i.e., $...x_n^1, x_n^2, c_n^1, x_n^3, c_n^2, x_n^4, x_n^5...$ as in Fig. 2(D).

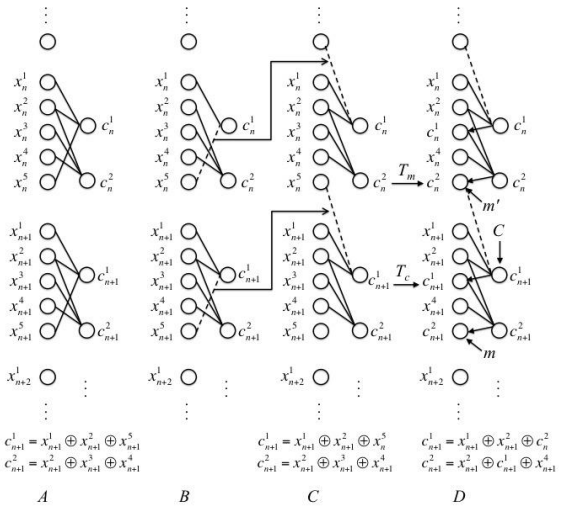

Fig. 2 Convolution Bipartite Graph Generation

At the decoder, the original messages are recovered block by block according to the convolution bipartite graph. Sometimes, the decoder may fail to recover the message nodes due to the burst packets loss. When this happens, the decoder waits for more check nodes for decoding until all the message nodes are successfully recovered or the decoder buffer overflows. It is self-evident that the decoder buffer size is of great importance to the performance of cTN code. The larger the decoder buffer size M, the higher the recovery probability of missing nodes, however, the higher the decoding delay at the same time. On the contrary, the performance of the block FEC codes, such as the RS code [6] and the tTN code, will not affected by the size of the decoding buffer. In practice, the size of M for the cTN code should be carefully chosen by considering the tradeoff between the correction capability and the decoding delay.

4. Simulation results

A comparison of the cTN code against the tTN code and RS code, over the Gilbert channel model [7] with an average burst length of 3, was simulated on a Pentium IV CPU at 3.0 GHz with 1 GB RAM running Windows XP. All three codes have 900 message nodes and 80 check nodes. The tTN code has three levels of the bipartite sub-graphs and each level has 900, 72 and 8 nodes respectively.

The simulated node recovery performance is summarized in Table 1, where the recovery rate indicates the percentage of the corrected nodes in the lost nodes. From the table, it is obvious that the proposed cTN code greatly outperforms the tTN code and it performs better and better as M increases. It can also be observed that both the cTN and the tTN code are less competitive than RS code in terms of recovery performance. This makes sense because the tTN code has a decoding inefficiency of $(1+\varepsilon)$ while RS code has 1 [3].

Table 1 nodes recovery performance comparison (average burst length of 3)

Recovery Rate(%)	Average Packet Loss Rate						
	1%	2%	3%	4%	5%	6%	7%
cTN(*M*=980)	100	100	100	100	99.84	99.85	98.23
cTN(*M*=490)	100	100	100	100	100	99.33	92.99
tTN	99.98	99.16	95.56	85.6	67.71	52.25	33.46
RS	100	100	100	100	100	100	99.98

The comparison of the computation complexity (in terms of encoding/decoding time) among the cTN, tTN code and fast RS code (fRS) [8] are shown in Table. 2. In the simulation, 675Mbytes data with node length of 150 bytes is encoded and decoded. As shown in the table, the encoding and

decoding time of the cTN code is 73% and 78.3% shorter than the tTN code. Moreoever, the fRS code consumes far more encoding and decoding time than the tTN code and cTN code because fRS code has approximately quadratic complexity while TN code has linear complexity [3][8].

Table 2 complexity comparison

FEC Code	Encoding time(seconds)	Decoding time(seconds)
cTN	24.53	37.97
tTN	90.69	175.10
fRS	197.96	306.82

From the above simulations, we can conclude that the cTN code demands much less encoding and decoding time though it suffers a slight recovery performance loss than the RS. In addition, compared with the tTN, the cTN code is more robust in burst loss environment with lower complexity.

5. Conclusions

In this paper, we propose a convolution tornado code, which incorporates the convolution concept into the traditional tornado code. By doing so, the recovery performance is greatly improved in a burst-erasure environment with reduced computation complexity. The simulations results also confirm such improvement. Therefore, the proposed cTN code is more appropriate for real-time data communication.

Acknowledgements

This study is supported by the Projects in the National Science & Technology Pillar Program during the Twelfth Five-year Plan Period (2015BAB07B01), Fundamental Research Funds for the Central Universities (No. 2015B26914), Innovation and Entrepreneurship Training Program of Hohai University for students (No. 2016102941103).

References

[1] M. Luby, M. Mitzenmacher, A. Shokrollahi and D. Spielman, “Efficient Erasure Correcting Codes,” IEEE Trans. Information Theory, vol. 47, no. 2, pp. 569-584, Feb. 2001.
[2] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman and V. Stemann, “Practical Loss-Resilient Codes,” In Proceedings of the 29th ACM Symposium on Theory of Computing, pp. 150-159, 1997.
[3] J. Byers, M. Luby, M. Mitzenmacher and A. Rege, “A Digital Fountain Approach to Reliable Distribution of Bulk Data,” In Proc. of ACM SIGCOMM’98, pp. 56-67, 1998.
[4] J. Zhang, Q. Cui and X. Liu, “Broadcast Authentication Scheme Based on Tornado Code for Wireless Sensor,” Multimedia Information Networking and Security, MINES '09, pp. 365-369, 2009.
[5] Y. Xiang and F. Li, “Coding via random convolution,” Image and Signal Processing (3rdCISP), pp. 3263-3267, 2010.
[6] P. Wang, L. Song, S. Yu and L. Yang, “Analysis and comparison of FEC and FEC-ARQ protection schemes based on RS and Raptor code,” Wireless Communications and Signal Processing (WCSP2010), pp. 1-6, 2010.
[7] Gilbert, E. N., “Capacity of burst-noise channel,” Bell Syst. Tech. J., vol 39, no. 9, pp. 1253-1266, 1960.
[8] Feng, G. L., Deng, R. H., Bao, F., “Packet-loss resilient coding scheme with only XOR operations,” IEE E Proc. Communications, vol 151, no. 4, pp. 322-328, 2004.