Engineering Journal issn: 0125-8281

Article

Verification-Based Decoding for Rateless Codes in the Presence of Errors and Erasures

Usana Tuntoolavest[a,*], Nabeela Shaheen[b], and Visuttha Manthamkarn[c]

Department of Electrical Engineering, Faculty of Engineering, Kasetsart University, Bangkok, 10900, Thailand
E-mail: [a,*]usana.t@ku.th (Corresponding author), [b]nabeela.s@ku.th, [c]visuttha.m@ku.th

Abstract. In this paper, verification-based decoding is proposed for the correction and filling-in of lost/erased packets for multicast service in data networks, which employs Rateless codes. Patterns of preferred parity-check equations are presented for the reduction of the average number of parity-check symbols required. Since the locations of unverified symbols are known, the effect of erasures and errors is the same in terms of the overhead required for successful decoding. Simulation results show that for an error-only, an erasure-only or a combination of both at 10% error/erasure probability, 78% of the messages can be recovered with a 50% overhead, whereas 99% of the messages can be recovered with a 100% overhead.

Keywords: Verification-based decoding, multicast, erroneous packets, lost packets, Rateless code.

ENGINEERING JOURNAL Volume 26 Issue 4
Received 17 November 2021
Accepted 24 March 2022
Published 30 April 2022
Online at https://engj.org/
DOI:10.4186/ej.2022.26.4.37

DOI:10.4186/ej.2022.26.4.37

1. Introduction

Real-time multimedia broadcasting/multicasting, such as audio/video streaming via wireless networks, are used more widely in numerous applications nowadays. In particularly, Lightweight User Datagram Protocol (UDP-Lite) network is designed for both broadcasting and multicasting services [1] as well as various multimedia such as H.263+, H.264 and MPEG-4 [2]. Furthermore, unlike other networks, the UDP-lite network does not drop corrupted data packets due to its feature [2]. As a result, both error and erasure packets, caused by either wireless bursty channels [3][4][5][6] or congested networks, can occur in this network. Conventionally, to guarantee the transfer in a communication link, Forward Error Correction codes (FEC) and Automatic Repeat Request (ARQ) are standard methods to correct errors and recover erasures, respectively. In wireless broadcasting/multicasting networks, the use of ARQ is not suitable because each of the numerous receivers may have different channel conditions and limited feedback channel capabilities. ARQ also cause significant delay [4] and the retransmission from the sender is becoming less practical when the number of receivers increases as the sender needs to maintain one feedback channel for each receiver. Therefore, recovering the lost and erroneous packets by ARQ is not practical in wireless broadcasting/multicasting networks with large number of receivers. The fountain code, named Raptor codes can be used to solve this problem as shown in [7].

Fountain codes are suitable "for channels with erasures such as the internet" [8]. They are also used as reliable multicasts in data networks in cases where a sender must send the same message to a large number of receivers [8]. Since packets can be lost during their transmission in the network from many reasons such as congestion or lack of high-speed backbones [5], noises and interferences [3], different set of packets may be received by each receiver. The advantage of Fountain codes is that the decoder at each receiver is capable of always performing the decoding process, even in the case of a very-low quality channel. Ideally, there is no need for ARQ since FEC will always be sufficient if there is no limit on the redundancy.

Fountain codes are also called "Rateless codes" because there is no fixed code rate for any of the receivers [8]. In good channel conditions, a receiver has a higher effective code rate than a receiver in bad channel conditions. Therefore, each receiver may have a different effective code rate. Rateless codes were originally proposed for packet-erasure channels. However, since then, a significant amount of research on various noisy channels has been conducted [9][10][11]. These codes have been found to be more robust than conventional fixed-rate codes. Moreover, it has been proved that Raptor codes can achieve capacity on the Symmetric Channel [10].

In data networks, the fundamental unit of transmission is the packet. Thus, it is more appropriate to treat a packet as the fundamental unit or a symbol over GF(2^r) in the coding scheme. Thus, each packet is viewed as a non-binary symbol over GF(2^r) in this paper. Luby and Mitzenmacher stated that a large value of r makes the probability of linearly dependent erroneous symbols negligible [12]. Another work on Vector symbol decoding (VSD), which provided an inspiration for the proposed algorithm, also showed that $r \geqslant 32$ bits are enough to satisfy the linearly independent error pattern assumption [13]. This assumption is necessary to avoid false verifications (FV).

Belief-Propagation-based decoders for non-binary codes are computationally ineffective when the field size '2^r' increases. Other approaches for decoding these codes have been presented in [12][14][15]. One of these approaches (i.e., the Verification-based (VB) decoding [12]), has been applied to Low-density parity-check (LDPC) codes and other block codes in various decoding schemes. VB schemes, which are called "Majority-logic-like decoding," were developed by Metzner in [16][17] for block codes. Luby and Mitzenmacher [12] presented an iterative VB decoder for LDPC codes for the q-ary symmetric channel (q-SC). Regarding VB decoding of Rateless schemes, an elegant algorithm for Raptor codes on the same q-SC was presented in [14], where another alternative for VB decoding of Rateless codes was presented.

This paper proposed an alternative decoding scheme for wireless broadcasting/multicasting networks with no ARQ that can still guarantee reliable transmission. The main difference of the proposed scheme from other works is that it is capable of handling both errors and erasures by itself in the same algorithm instead of relying on other layers of the OSI model. This is possible because the decoding algorithm can verify the correct symbol positions. The unverified symbols are previously treated as errors in [12], but they can be treated as erasures instead since their positions are known while their values are unknown. This makes errors and erasures alike to the decoder. Therefore, the proposed decoder requires the same amount of overhead to treat errors, erasures and combinations of errors and erasures. This allows for more flexible scenarios of errors only, erasures only and a combination of errors and erasures.

In contrast, the method proposed in [12] assumes errors only, whereas other works on Rateless codes assume erasures only [8][18]. Some Rateless codes that assume erasures only may leave the error correction to another decoding technique such as Reed-Solomon decoder. The proposed decoding technique shares some of the verification rules with the technique presented in [12]. The difference is that the proposed VB decoding scheme is capable of correctly verifying codes, which have both short and long cycles in the Tanner graph, whereas the rules described in [12] suffer from FV in the presence of short cycles.

In [19], the research conducted on VB decoding for packet based LDPC codes aimed at reducing the likelihood of FV by increasing the number of checks

required to verify each symbol. However, by applying the proposed VB decoding, FV is found to be negligible for large symbols in consideration. Therefore, only one check is enough to verify any symbol.

In many protocols, packets usually have either a simple built-in error-correction or error-detection code. Regarding error-correction codes, a small number of erroneous bits due to noises can be corrected, whereas larger numbers of erroneous bits will result in erroneous packets. For error-detection codes, any number of erroneous bits will result in packet erasure or loss. Other conditions may result in packet loss such as in a congested network. In this case, low-priority packets may be dropped. Thus, in this paper, the probability of error is considered as the probability of an erroneous packet, whereas the probability of erasure is considered as the probability of a lost packet.

The background on VB decoding is explained in Section 2. Fountain codes and the framework of the Rateless codes are also considered in this section. The code specifications and preferred parity-check patterns are described in Section 3. The proposed method is presented in Section 4, whereas its computation complexity is presented in Section 5. Simulation results are presented in Section 6. Finally, a discussion on the results is presented in Section 7, and the conclusions are presented in Section 8.

2. VB Decoding Background

2.1. A Note on Verification-based (VB) Decoding

The basic concept in VB decoding is that a check node can verify its neighbors if the check constraint is satisfied, given the erroneous symbols are not linearly dependent. The Exclusive-OR (XOR) sum of all check-node neighbors being equal to zero is the check constraint most used. If only one of the symbols during a check is unverified at any stage during decoding, it is calculated using this check constraint.

Apart from these basic rules for verification and correction, individual decoders employ some additional ways for verification and/or correction [12][20].

Example 1: Consider a nonzero check-node ($\boldsymbol{c}$) in Eq. (1) and suppose the received symbols, y_1 and y_2 are both verified as v_1 and v_2, where v_i is the i^{th} codeword symbol and y_j is the j^{th} received symbol.

$$\boldsymbol{c} = \boldsymbol{v}_1 \oplus \boldsymbol{v}_2 \oplus \boldsymbol{y}_3 \quad (1)$$

where $\oplus$ is the vector modulo-2 sum.

From Eq. (1), there is only one unverified symbol, y_3. As a result, the check-node can correct y_3 by the Eq. (2).

$$\boldsymbol{y}_3 = \boldsymbol{v}_1 \oplus \boldsymbol{v}_2 \oplus \boldsymbol{c} \quad (2)$$

2.2. Rateless Codes

Rateless codes are codes that has no pre-fixed rate for the code; the encoder can generate as many encoded symbols as those required for decoding. The well-known Rateless codes are fountain codes. Fountain codes are based on the fountain and bucket analogy [18]. Just like a bucket can be filled by collecting a certain number of droplets from a fountain, a fountain code can be decoded if a certain number of encoded units/symbols is received.

Example 2: Suppose a sender sends packets #1 to #8. Assume that the decoder can decode successfully if it receives 5 different packets.

Table 1. Sets of received packets for 3 different receivers.

Rx	1st received packet	2nd received packet	3nd received packet	4th received packet	5th received packet
1	#1	#3	#4	#6	#7
2	#2	#3	#4	#5	
3	#1	#2	#5	#6	#8

From Table 1, the first and the third receivers receive 5 packets, so they can decode successfully. The second receiver receives only 4 packets, so it cannot decode at this time.

3. Code Specifications and Preferred Check Patterns

The framework of the Rateless code used for testing the proposed algorithm is described in this section. For the proof of concept, a code with a degree distribution chosen by trial and error is generated using the software developed by Neal and MacKay for the LDPC codes [21]. The code structure is systematic. This allows quick data recovery in good channel conditions. Optimizations to find good-degree distributions, which are mostly suited to the decoding algorithm, are not conducted here.

This paper proposes preferred check-symbol patterns based on "a divide and conquer" technique, which has been submitted for a petty patent [22]. Good results on the decoding overhead are obtained when preferred check-symbol patterns are transmitted right after the data symbols in good channel conditions. These patterns for 20 data symbols are shown in Fig. 1. The first check symbol is designed to be the sum of all data symbols. Thus, if all data symbols plus this check symbol are received correctly, the overhead is minimized to only one check symbol. The next two check symbols are designed to be the sum of the first half and the second half of the data symbols, respectively. In this way, they can verify half of the data symbols when errors or erasures occur in the other half. The 4th check symbol is then the sum of the first 25% and the third 25% of the data symbols. Finally, the 5th one is the sum of the second 25% and the fourth 25% of the data symbols. Basically, half of the data symbols are included with new check patterns until the last two check symbols of the preferred patterns are 101010….10 and 010101…01.

1st check pattern	11111111111111111111
2nd check pattern	11111111110000000000
3rd check pattern	00000000001111111111
4th check pattern	11111000001111100000
5th check pattern	00000111110000011111
	…
	…

Fig. 1. Preferred parity-check patterns. The positions of 1's indicate the positions of data symbols included in the check equations.

4. Description of the Proposed Decoding Algorithm

When decoding starts, the symbols are either unverified or marked as erased due to lost packets. The decoder runs greedily, as the check symbols arrive. The use of systematic codes allows data symbols to be transmitted before check symbols. The decoder starts its decoding process each time it receives a new check symbol. It then applies the following rules, which are also depicted in the flowchart shown in Fig. 2.

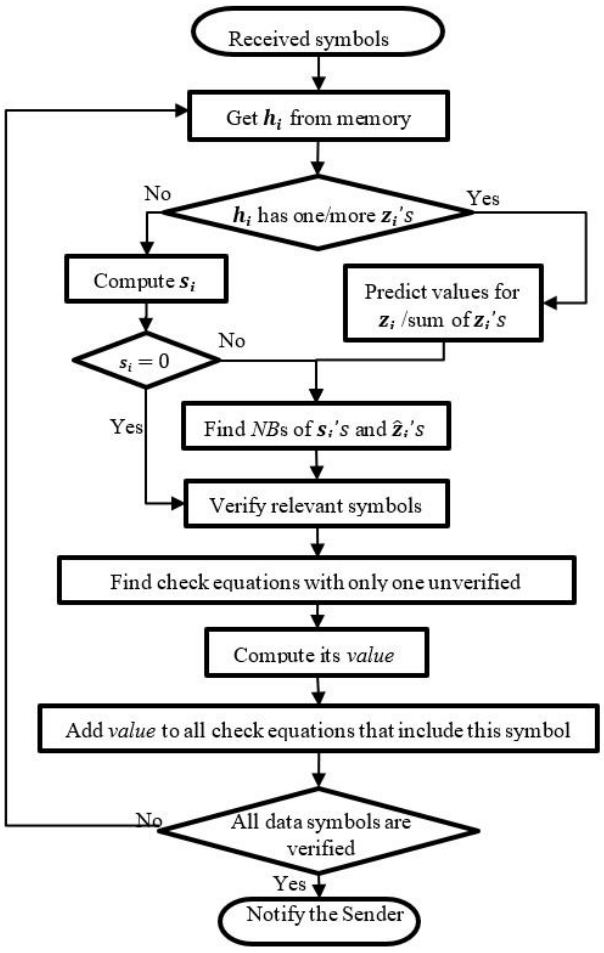

Fig. 2. Flowchart of the proposed decoding algorithm. Abbreviations: NB = null combination; h_i = check equation; z_i = erased symbol, z_i = predicted value.

Equations (3) and (4) are given to show the calculation of syndrome matrix when the decoder receives a new symbol.

$$\boldsymbol{S} = \boldsymbol{H} \cdot \boldsymbol{Y} \tag{3}$$

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ \vdots \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ z_4 \\ y_5 \\ \vdots \end{bmatrix} \tag{4}$$

where $\boldsymbol{S}$ is the syndrome matrix.

$\boldsymbol{H}$ is the parity-check matrix.

$\boldsymbol{Y}$ is the received symbol matrix.

1) For the case of no erased symbols (z_i) in the check equation (h_i), if the vector modulo-2 sum of the received values (syndrome, s_i) is a zero vector, all symbols in h_i are verified.
2) For the case of only one unverified symbol in h_i, that symbol value is set to make s_i a zero vector.
3) For the case s_i is non-zero, its value is recorded in a list. This is potentially an erroneous symbol value or a sum of more erroneous symbols. Such combinations of s_i's, which are found to sum to a zero vector with modulo-2 addition, are called "null combinations (NB)". This term was first used by Metzner in [23]. An NB indicates the canceling out of the same erroneous symbols from the s_i's involved. Thus, all symbols, which do not occur in pairs in the corresponding h_i's, are verified.
4) For the case one or more z_i's in h_i exist, a value for z_i or the sum of z_i's is predicted by assuming that the rest of the data and check symbols are correct. This value is recorded in a separate list. The prediction of the same erased symbols from different checks is summed with modulo-2 addition. If there is an NB, then, similar to the argument in Rule #3, the symbols, which do not occur in pairs in the corresponding h_i's, are verified. Generally, any number of predicted values expected to result in NB can be checked if they cause NB. However, it can be observed that up to five values are enough for the least amount of overhead.

5. Computational Complexity

Since the code uses symbols over GF(2^r), all computations are performed using modulo-2 arithmetic. Therefore, the complexity is dominated by symbol-level operations, especially for large symbol sizes. Regarding Rule #1 and #2 of the decoding algorithm, the vector XOR is used to calculate syndromes and the values of unverified symbols. Regarding Rule # 3, a Gauss-Jordan reduction is performed to facilitate the search for NB's. The corresponding list elements are arranged into a matrix, where the list elements are the rows of the matrix. Since modern processors provide 128-bit words [24], a

whole column in the matrix can be represented by a single word. The XOR of two 128-bit words is equal to a vector XOR of 128-bit symbols. This greatly reduces the number of operations. Considering that the erasure positions are already known, the application of Rule #4 can be made simpler. Thus, the step of searching for NBs can be omitted.

Therefore, the complexity of the decoder is dominated by the operation described in Rule #3. An effort is made to keep the entries on the $\boldsymbol{s}_i$'s list small. Once there is an NB between two $\boldsymbol{s}_i$'s, one of them is removed from the list because it will not be useful for verifying any more symbols. As soon as the erroneous symbols are corrected, the related entries are also removed. Implementation of these measures keeps the list size small throughout the decoding process. The number of operations depends on the number of check symbols used in the decoding process. Hence, the complexity also depends on this same number.

6. Simulation Results

For each value of the probability of error or probability of erasure, 50,000 decoding runs were performed. The number of check symbols required for a complete decoding of k data symbols was recorded each time. The number of total messages recovered by increasing the amount of overhead was accumulated and plotted against the $(1+\varepsilon)k$ overhead, which is given as a multiple of k. Therefore, an overhead of ε, for example, means that symbols are received in total. These include k data symbols and εk check symbols.

The performance of the proposed decoding algorithm is presented in Fig. 3 and 4. The number of data symbols is $k = 20$ with 32 bits/symbol. From Fig. 3, it can be observed that the decoder requires the same amount of overhead for errors and erasures. This effect is obvious when a combination of errors and erasures also provides the same result.

From Fig. 3, it can be observed that 78% of decoding runs complete the decoding process for a $0.5k$ overhead, and 99% of the decoding process is completed before the overhead of k symbols with an average of $0.4k$ for a given probability value. The results obtained for various probabilities for a combination of errors and erasures are shown in Fig. 4. The numbers shown are the sum of both, and it is assumed that $P_e = P_{era}$. For example, a probability of error/erasure = 0.08 means that P_e = 0.04 and P_{era} = 0.04 as well. Assuming no error and erasure ($P_{e/era}$ = 0), the decoder is successful with an overhead of 0.05 or 1 check symbol/20 data symbols. When the ratio $P_{e/era}$ increases, the fraction of messages recovered with the same overhead decreases, as expected. However, all messages are eventually recovered when the overhead is $1.5k$. A comparison among three different code lengths for 20, 50, and 100 data symbols is presented in Fig. 5. In the case of 20 data symbols, the amount of overhead required to recover the whole message is the largest.

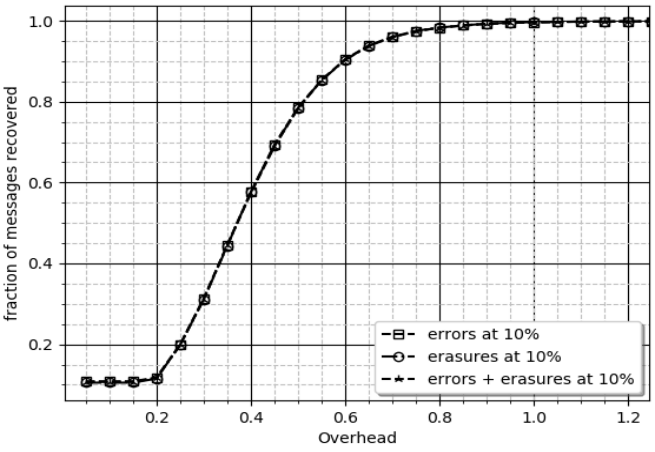

Fig. 3. Fraction of successful decoding completions vs. overhead for errors only, erasures only, and a combination of both.

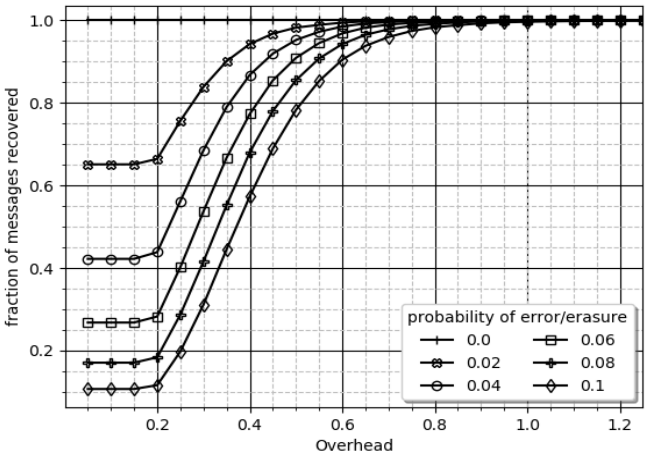

Fig. 4. Fraction of successful decoding completions vs. overhead for the cases that $P_e = P_{era}$, and their sum is in the range $0 - 0.1$.

Figure 6 illustrates the performance comparison of the proposed VB decoder with other decoders that can handle packet-sized symbols. Tornado code decoding is the best in recovering messages given the same overhead, but it assumes that only erasures can occur. Vector symbol decoding (VSD) is another nonbinary decoding for packet-sized symbol, but it can correct errors only [25][26][27]. The performance of the proposed VB decoding is better than VSD when the overhead is less than 0.45 and VSD is better when the overhead is at least 0.45. However, both Tornado code decoding and VSD are not designed to handle a mixture of errors and erasures.

7. Discussion

The simulation results verify that the decoder requires a similar amount of overhead, regardless of the presence of erroneous or erased symbols. This is expected because Rule #2, which is the only correction rule used, makes no distinction between erroneous and erasure symbols. When checking is performed, a single unverified data symbol is set to a value that satisfies the

DOI:10.4186/ej.2022.26.4.37

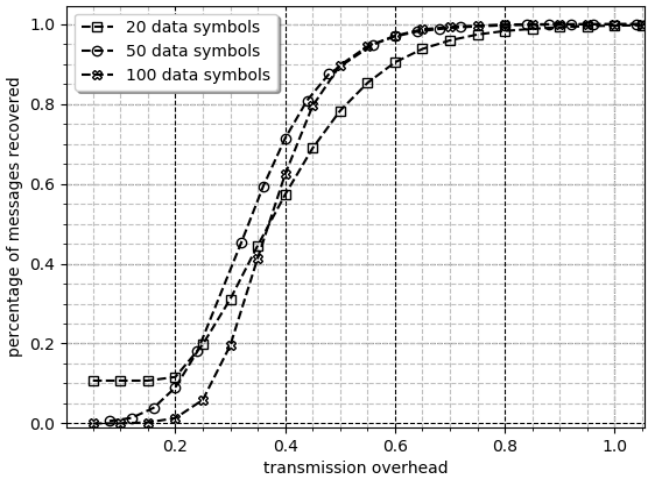

Fig. 5. Effect of code length on overhead.

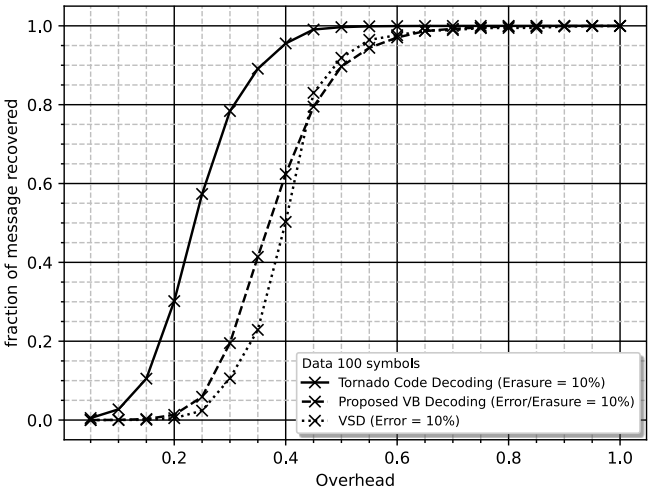

Fig. 6. Performance comparison of Tornado code (erasure only), the proposed algorithm (mixture of errors and erasures) and VSD (error only).

check constraint. The difference between the effect of errors and erasures is that the step performed to find NBs is omitted for erasures, as explained in Section 5. The complexity is directly related to the list size. The list size (and hence, the complexity) can be kept low by removing redundant data from the lists whenever possible. Hence, the size of these lists keeps changing and does not necessarily increase with the increase in the overhead.

From the graphs in Fig. 3, 4, and 5, three regions can be roughly observed. The first region starts at the fraction of complete decoding runs with a single received check symbol. The rising region represents the chain recovery of data symbols. A recovered symbol increases the probability of recovering more data symbols. The saturation region is the region, where approximately 5% of the decoding attempts require a higher amount of overhead to complete the decoding process.

In comparison with other decoding techniques that can handle packet-sized symbols and are suitable for Rateless codes, Tornado code decoding is better and VSD is quite similar to the proposed VB decoding. However, Tornado code decoding is for erasure only, while VSD is for error only. The main advantage of the proposed decoding is that it can handle a mixture of errors and erasures at the same time.

As expected, no FVs were observed during the simulations. By employing a large enough symbol size, false verification is avoided with high confidence [12]. The algorithm presented here does not suffer from FV due to cycles in the code as described in [12].

For the range of probabilities considered, the maximum amount of overhead observed is k. For higher probabilities, the messages will be eventually recovered but the amount of the overhead may be higher. This situation could be managed by keeping an upper limit on the overhead allowed before reporting failure. For a higher number of data packets, the required overhead for a given fraction of recovered messages decreases after a certain amount of overhead. Moreover, as shown in [28] for Raptor codes in Binary Symmetric channels, the overhead decreases with the code length for a fixed residual bit error rate.

8. Conclusion

In this paper, an alternative approach for the decoding of Rateless codes was introduced. The aim was to present a decoding scheme for multicasting service in data networks. Various sources of data corruption exist in these networks. Instead of focusing on a single-channel model, the overall network conditions are considered in terms of the probability of lost/erased and erroneous packets. The ability of the proposed algorithm to handle errors as well as erasures makes for a relaxed decoding constraint, as the assumptions of no erroneous or no lost packet reception may not be practical in many cases. The proposed decoder utilizes a similar amount of overhead for the recovery of erroneous and erased packets. Recovering from erasures usually requires fewer computations than errors, as their locations are known beforehand. The proposed algorithm is very flexible, as it is capable of handling network conditions with errors only, erasures only or a combination of both.

Varied performances were observed for different degree distributions. Optimizing for the best-degree distribution suited for the coding scheme can greatly improve the performance. This can be investigated in a future work.

Acknowledgement

The authors would like to thank Kasetsart University Research and Development Institute for support with proof reading and comments.

References

[1] G. Fairhurst, B. Trammell, and M. Kuehlewind, "Services provided by IETF transport protocols

and congestion control mechanisms," *RFC 8095*, 2004.

[2] L.-A. Larzon, M. Degermark, S. Pink, L.-E. Jonsson, and G. Fairhurst, "The lightweight user datagram protocol (UDP-Lite)," *RFC 3828*, 2004.

[3] P. Ostovari and J. Wu, "Reliable broadcast with joint forward error correction and erasure codes in wireless communication networks," in *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, Dallas, USA, 2015.

[4] I.-H. Hou, "Broadcasting delay-constrained traffic over unreliable wireless links with network coding," *IEEE ACM Trans. Netw.*, vol. 23, no. 23, pp. 728-740, 2015.

[5] D. Chen, B. Rong, N. Shayan, M. Bennani, J. Cabral, M. Kadoch, and A. K. Elhakeem, "Interleaved FEC/ARQ coding for QoS multicast over the internet," *Can. J. Electr. Comput. Eng.*, vol. 29, no. 3, pp. 159-166, 2004.

[6] A. Majumda, D. G. Sachs, I. V. Kozintsev, K. Ramchandran, and M. M. Yeung, "Multicast and unicast real-time video streaming over wireless LANs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 6, pp. 524-534, 2002.

[7] T. Mladenov, S. Nooshabadi and K. Kim, "Efficient GF(256) raptor code decoding for multimedia broadcast/multicast services and consumer terminals," *IEEE Trans. Consum. Electron.*, vol. 58, no. 2, pp. 356-363, 2012.

[8] J. Byers, M. Luby, M. Mitzenmacher and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *ACM SIGCOMM Comput. Commun. Rev*, vol. 28, no. 4, pp. 56-67, 1998.

[9] R. Palanki and J. S. Yedidia, "Rateless codes on noisy channels," in *IEEE International Symposium on Information Theory*, Chicago, USA, 2004.

[10] O. Etesami, M. Molkaraie, and A. Shokrollahi, "Raptor codes on symmetric channel," in *IEEE International Symposium on Information Theory*, Chicago, USA, 2004.

[11] J. Castura and Y. Mao, "Rateless coding over fading channels," *IEEE Commun. lett.*, vol. 10, no. 1, pp. 46-48, 2006.

[12] M. Luby and M. Mitzenmacher, "Verification-based decoding for packet-based low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 120-127, 2005.

[13] U. Tuntoolavest, "A simple method to improve the performance of convolutional vector symbol decoding with small symbol size," in *2004 IEEE Region 10 Conference TENCON 2004*, Chiang Mai, Thailand, 2004.

[14] R. Karp, M. Luby, and A. Shokrollahi, "Verification decoding of raptor codes," in *International Symposium on Information Theory*, Adelaide, Australia, 2005.

[15] M. Fossorier and D. Declercq, "Decoding algorithms for nonbinary LDPC codes over GF(*q*)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633-643, 2007.

[16] J. J. Metzner, "Majority-logic-like decoding of vector symbols," *IEEE Trans. Commun.*, vol. 40, no. 10, pp. 1227-1230, 1996.

[17] J. J. Metzner, "Majority-logic-like vector symbol decoding with alternative symbol value lists," *IEEE Trans. Commun.*, vol. 48, no. 12, pp. 2005-2013, 2000.

[18] A. Shokrollahi, "Raptor codes," *IEEE ACM Trans. Netw.*, vol. 14, no. SI, pp. 2551-2567, 2006.

[19] B. Zhu, D. Huang, and S. Nordholm, "Enhanced verification-based decoding for packet-based LDPC codes," *IEEE Commun. Lett.*, vol. 12, no. 2, pp. 136-138, 2008.

[20] F. Zhang and H. D. Pfister, "List-message passing achieves capacity on the q-ary symmetric channel for large q," in *IEEE Global Telecommunications Conference*, Washington, USA, 2007.

[21] D. MacKay and R. Neal, *Software for Low Density Parity Check Codes*. 2012. [Online]. Available: http://www.cs.toronto.edu/~radford/ftp/LDPC-2012-02-11/index.html [Accessed: 15 October 2019].

[22] U. Tuntoolavest and N. Shaheen, "Encoding and decoding method of systematic Rateless Codes with the predefined first set of parity symbols structure by the Divide and Conquer technique to resolve erasures and errors," (in Thai) submitted for petty patent no. 2003003424, 2020.

[23] J. J. Metzner and E. J. Kapturowski, "A general decoding technique applicable to replicated file disagreement location and concatenated code decoding," *IEEE Trans. Inf. Theory*, vol. 36, no. 4, pp. 911-917, 1990.

[24] J. Ledin, "The RISC-V Architecture and Instruction Set," in *Modern Computer Architecture and Organization*, 1st ed. Birmingham, United Kingdom: Packt Publishing, 2020, ch. 11, pp. 293.

[25] U. Tuntoolavest, *Vector Symbol Decoding for Wireless Fading Channels*. Saarbrücken, German: VDM publishing, 2009.

[26] N. Shaheen and U. Tuntoolavest, "Effect of weight distribution on vector symbol decoder performance," in *4th IEEE international Women in Engineering (WIE) Conference on Electrical and Computer Engineering 2018*, Pattaya, Thailand, 2018, pp. 1-4.

[27] J. J. Metzner, "Vector symbol decoding with list inner symbol decisions," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 371-380, 2003.

[28] A. Shokrollahi and O. Etesami, "Raptor codes on binary memoryless symmetric channels," *IEEE Trans. Inf. Theory*, vol. 52, no. 5, pp. 2033-2051, 2006.

Usana Tuntoolavest received the B.S. degree in electrical engineering from Chulalongkorn University in 1995, and the M.S. and Ph.D. degrees from the Pennsylvania State University, PA, USA in 1997 and 2002, respectively. She is currently an Associate Professor at Kasetsart University, Bangkok, Thailand. She has received many awards including the Excellence Lecturer of the 5th ASAIHL-Thailand Award (2017) from the Association of Southeast Asian Institute of higher learning, Thailand.

Nabeela Shaheen received the B.S degree in electronic engineering in 2011 and The M.Eng Degree in 'Information and Communication Technology' in 2021. She currently works as an Embedded Software Architect.

Visuttha Manthamkarn received the B.S. degree in electrical engineering from Kasetsart University in 2019. He is now working as a researcher at Kasetsart University, Bangkok, Thailand. His current research interests include channel coding and implementation for the LDPC encoding/decoding.