Prioritized LT Codes

Simon S. Woo and Michael K. Cheng
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109
{Simon.S.Woo, Michael.K.Cheng}@jpl.nasa.gov

Abstract—**It is common in data transmissions that some information is more important than others. This is especially true in space communications where mission critical information or science data are high priority. In this work, we propose a simple yet constructive scheme to send high priority data reliably and efficiently using Luby Transform (LT) codes. The new proposed scheme modifies the conventional LT encoder to send high priority data as a degree 1 and 2 so that high priority data can be quickly resolved and very likely recovered before a decoder stops. Preliminary results show that a carefully designed degree distribution of high priority data increases the likelihood of receiving high priority information while having negligible performance impact on data with lower priority. The performance of the proposed scheme is evaluated and compared with the conventional LT approach with the same parameters under a range of erasure error rates.**

I. Introduction

Currently, the Constellation Program [1] in NASA is underway to assist humans to return to the moon by 2020 and establish a base for exploration of the lunar surface. In order to meet the complex mission requirements and challenges, there is a strong need to optimize data transmissions over space communications links. High priority data that contain mission critical information or science data need to be recovered with high reliability. Currently, strong forward error correction (FEC) codes have been used to provide the extra protection for high priority data. Here, transmission rates have to be sacrificed to provide better protection. Also, protocols based on the automatic repeat request (ARQ) strategy [2] have been used for several successful space missions systems in the past. However, throughput inefficiency arises from extra hand-shake requirements and additional propagation delay between sender and receiver. To compensate for drawbacks of FEC and ARQ schemes, several variations of a hybrid ARQ scheme [3] have been widely adapted, where the hybrid ARQ consists of an FEC subsystem in an ARQ system.

In this work, we consider using LT codes [4] in the application layer as a part of the hybrid ARQ system. With relatively small overhead, it has been shown that LT codes can achieve extremely good performance over the erasure channel as information size increases. The benefit of LT codes over Reed-Solomon codes is the low decoding complexity. Moreover, LT codes can be generated on the fly. Block LT codes can be effective in deep space communications, where propagation delay is significant, thus requiring minimal use of the reverse channel.

Karp *et al.* and Maneva *et al.* analyzed the error probability of the *belief propagation* decoder applied to LT codes in [5] and [6] respectively. Also, Rahnavard *et al.* considered rateless codes to provide more protection of high priority data in [7]. Furthermore, unequal error protection for video streaming using Growth codes have been recently studied in [8]. However, in [7] and [8], low priority data reception performance has to be sacrificed due to sending high priority data more frequently.

In this work, we explore a novel way of sending high priority data using block LT codes, where high and low priority data packets are combined in LT code blocks. By examining the degree distributions of the entire data as well as high priority data, we found that careful selection of degrees for high priority bits can accelerate decoding the majority of high priority information with minimal modifications in the LT encoder. The motivation and objective of this work is to maximize the likelihood of receiving high priority data, even in cases where a decoder fails to decode all information.

II. Background

LT codes are efficient for encoding and decoding of a large size of information bits with relatively small overhead. They have a great potential for point-to-point, broadcast, and multicast communication applications. The following parameters are defined for the LT encoding process:

- k : information bits
- n : encoded bits
- $r = n - k$: overhead
- $p_1, p_2, \ldots, p_k$: discrete probabilities describing the degrees so that $\sum_{i=1}^{k} p_i = 1$

Each encoded bit is generated as follows:

1) Select a random degree, d, according to the degree distribution $\{ p_1, p_2, \ldots, p_k \}$
2) Choose a d-element subset uniformly from $\{1,2,\ldots, k\}$
3) XOR the information bits in positions specified by 2)

The encoded bit is a linear combination of information bits that are chosen from the degree distribution. The decoding

This research is supported under the title "Improving In Situ Communications using Network Coding" by the Research and Technology Development fund from Jet Propulsion Laboratory.

process is based on *belief propagation*, where degree 1 encoded bits are immediately recovered in Tanner graph [4]. The neighbors of recovered bits are XORed with recovered bits and the degree of each connected neighbor is decreased by one. Details of the decoding process are explained in [4].

The purpose of the degree distribution is to generate just enough different degree bits to XOR so that a decoder can decode all information. Lower degree encoded bits are required for initiating the decoding process. The degree distribution is designed in such a way that, probabilistically, most of the information bits that are in higher degrees can be decodable from the lower degree bits. Luby proposed the *Ideal-Soliton* [4] distribution. However, the performance of the *Ideal-Soliton* distribution is poor because the probability of generating degree 1 encoded bits approaches zero, as n increases. Hence, Luby proposed a more practical *Robust-Soliton* degree distribution [4] with additional parameters c and δ that specify the probability distribution of degrees. One of the reasons that the *Robust-Soliton* distribution performs better is that the probability of generating degree 1 does not vanish as much as in the *Ideal-Soliton* distribution as the packet size increases, while the probability of generating the other degrees in the *Robust-Soliton* degree distribution approaches those of the *Ideal-Soliton* degree distribution. In particular, the probability of generating degree 2 is close to 0.5, where it yields reasonably good decoding performance with minimal overhead.

III. Prioritized LT coding scheme

Here, we propose a simple scheme to accelerate high priority data receptions, by slightly modifying the conventional LT encoder. In our scheme, we use the *Robust-Soliton* distribution to decide the number of bits to XOR, but we non-uniformly choose some of lower degrees for high priority data while uniformly selecting the low priority data. The following parameters are defined for the prioritized LT coding scheme:

- h : the number of high priority bits
- $t = h/k$: fraction of high priority bits over total information bits
- $p_1, p_2, \ldots, p_k$: discrete probabilities describing the degrees so that $\sum_{i=1}^{k} p_i = 1$ for all bits using the *Robust-Soliton* distribution
- $q_1, q_2, \ldots, q_h$: discrete probabilities describing the degrees so that $\sum_{i=1}^{h} q_i = 1$ for high priority bits using the *Robust-Soliton* distribution

For ease of descriptions and analysis, we only consider two types of priority - high and low priority data. Furthermore, let high priority data be the first h bits of information. We refer to this as a high priority data group. Similarly, a low priority data group is defined as $\{h + 1, h + 2, \ldots, k\}$. The output of the encoder is defined as $\{x_1, x_2, \ldots, x_n\}$, where n is the size of encoded information and x_i is either 0 or 1. We assume that the number of degree 1 encoded bits are smaller than the size of high priority data h for practical considerations. Hence, without loss of generality, we define the following:

$$p_1 \cdot k << h. \tag{1}$$

Furthermore, we assume the size of high priority data is less than the half of total number of information bits in this work, where $p_2 \sim 0.5$. Therefore,

$$\begin{aligned} h &\le (p_1 + p_2) \cdot k, \\ t &\le (p_1 + p_2). \end{aligned} \tag{2}$$

From (1) and (2), t is in the following range:

$$p_1 < t \le p_1 + p_2. \tag{3}$$

Also let all degree 1 encoded bits be selected from a high priority data group. In addition, a certain number Ω of degree 2 encoded bits are solely chosen from a high priority data group and the rest of the degree 2 encoded bits are selected uniformly over all information bits as in conventional LT encoding.

The following pseudo code is given to describe the encoding steps to generate each encoded bit, x_i :

At the beginning,
    Initialize *count* = 1
For $i \in \{1,2,..., n\}$,
1) Select a random degree d according to the *Robust-Soliton* degree distribution, $\{p_1, p_2, \ldots, p_k\}$
   if $d = 1$,
      choose d uniformly from high priority data group $\{1,2, \ldots, h\}$
   else if $d = 2$,
      If *count* $\le \Omega$
         choose d uniformly from high priority data group $\{1,2,\ldots, h\}$
         and
         count = *count* + 1
      else
         choose d uniformly from entire set of information bits $\{1,2,\ldots, k\}$
   else if $d \ge 3$
      choose d uniformly from entire set of information bits $\{1,2,\ldots, k\}$
2) XOR information bits obtained from 1) to generate x_i

The key ideas behind our scheme are the following: the encoder imposes a constraint on the degree 1 and degree 2 encoded bits, since degree 1 and degree 2 critically affect the decoding performance. All degree 1 encoded bits, $p_1 \cdot k$, are selected from a high priority data group. This is because, in decoding LT codes, a degree 1 encoded bits are *necessary* for a decoder to initiate the decoding process. If there are no more degree 1 encoded bits, then the decoding process stops. Hence, if high priority data were sent as a degree 1, they could

be decoded immediately and would not have to wait for other bits to be decoded. Sending high priority bits as degree 1 removes the dependency requirement and increases the likelihood of receiving high priority bits first.

For the degree 2 encoded bits, a similar reasoning is applied. Degree 2 encoded bits can be directly decodable from the degree 1 encoded bit. Because we design all degree 1 encoded bits from high priority data group in the previous step, degree 2 encoded bits should include some of high priority bits to make use of decoded degree 1 high priority bits. Therefore, this is a main reason of choosing Ω number of degree 2 encoded bits among the high priority data group. Choosing too small or too large Ω impacts the performance drastically. A value around $\Omega \sim h/2$ seems to be a reasonable choice, since it is the number of degree 2 required for decoding packet size h so that it can decode high priority data first. The optimum value of Ω is explained in the next section. The update counter *count* is introduced to count the number of occurrences of degree 2 encoded bits so that the first Ω number of degree 2 encoded bits are chosen from the high priority data group uniformly. The rest of degree 2 and other higher degree bits are uniformly chosen from the entire span of information k. This provides room for initiating low priority data decoding simultaneously so that low priority data decoding performance is not sacrificed. In addition, extra protection and better decoding performance on high priority data can be added by uniformly selecting bits over k when the degree is greater than 2. High priority data portions can be again included in the higher degree encoded bits.

Hence, the prioritized LT scheme provides following benefits: high priority data can be decoded faster which directly impacts on the performance, since most of high priority data can be decodable before LT codes stop from not receiving enough overhead. Also, on average, prioritized LT scheme requires less number of XOR operations to decode high priority information.

To better illustrate our encoding algorithm, an example is given in Figure 1, where k = 10, n = 12, h = 3, and Ω = 2. Filled black circles indicate the high priority data and white empty dotted circles indicate the encoded bits with degree 2:

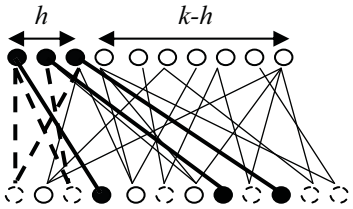

Fig. 1. Prioritized LT encoding process

For this specific example, the degrees of encoded bits are 2, 3, 2, 1, 3, 2, 3, 1, 2, 1, 2, and 2 respectively. This example illustrates that all degree 1 encoded bits are selected from three high priority data sources. Among six degree 2 encoded bits, first Ω encoded bits are selected uniformly over the high priority data group and the corresponding edge connections are shown as a dotted line. The rest of degree 2 and higher degree encoded bits are randomly chosen over k.

IV. Simple Analysis for Degree 2 Encoded Bits

Although it is difficult to analyze the complete decoding process of LT codes due to statistical variations in the degree distribution, the average degree 2 encoded bits can be categorized in terms of high priority bits, low priority bits, or combinations of both. This provides insights of how degree 2 encoded bits are distributed in terms of high and low priority data. Specifically, we want to characterize the degree 2 encoded bit distributions between the conventional LT codes and the prioritized LT scheme and check how the two different algorithms scale with different high priority data size.

The average number of degree 2 encoded bits generated from XORing of (high priority data, high priority data), (high priority data, low priority data), and (low priority data, low priority data) pair are

$$\{\frac{h}{k}\cdot\frac{h}{k}\cdot(p_2\cdot k)\},\ 2\cdot\{\frac{k-h}{k}\cdot\frac{h}{k}\cdot(p_2\cdot k)\},\text{ and}$$

$$\{\frac{k-h}{k}\cdot\frac{k-h}{k}\cdot(p_2\cdot k)\},$$

respectively for the conventional LT codes, where the sum of three terms is equal to $p_2\cdot k$. In the prioritized LT scheme, the average number of degree 2 encoded bits generated from XORing of (high priority data, high priority data), (high priority data, low priority data), and (low priority data, low priority data) pair are

$$\{\Omega+\frac{h}{k}\cdot\frac{h}{k}\cdot(p_2\cdot k-\Omega)\},\ 2\cdot\{\frac{k-h}{k}\cdot\frac{h}{k}\cdot(p_2\cdot k-\Omega)\},$$

$$\text{and }\{\frac{k-h}{k}\cdot\frac{k-h}{k}\cdot(p_2\cdot k-\Omega)\},$$

respectively. Hence, different number of degree 2 encoded bits are generated from different XOR combinations in two schemes. To accurately describe the behavior of each term, Figure 2 is given to compare the probability of generating each pair of degree 2 with the conventional LT encoding and the prioritized LT scheme as a function of t, while fixing Ω as a $h/2$. The exact value of Ω is determined through numerical simulation in the later section.

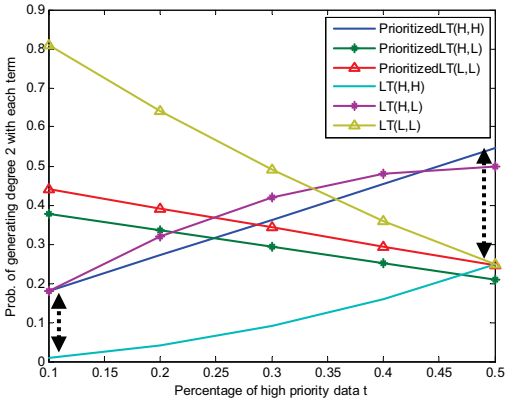

Fig. 2. Probability of generating degree 2 encoded bits with varying t

Solid lines indicate the probability of generating degree 2 encoded bits with the prioritized LT scheme. Dotted lines

indicate the result from the conventional LT scheme. In the prioritized LT scheme, it is shown that the probability of generating degree 2 encoded bits solely from high priority data increases. It is roughly offset by $\Omega \cdot (1 - \left(\frac{h}{k}\right)^2)$ compared to the conventional LT codes. In conventional LT codes, the probability of generating a bit from XORing two high priority bits is low over $t < 0.5$. Although slopes are about the same in both cases, the gain of high priority data protection comes from the gap which is shown with vertical arrows. From Figure 2, the prioritized LT scheme is scalable well over all different priority sizes, since the probability of generating encoded bits from the three different combinations almost reside mostly between 0.2 to 0.5 over all t. In conventional LT codes, however, this range is much wider; therefore, diluting the protection of high priority data. This is because LT codes are designed for decoding an entire information bit rather than decoding certain portion of data, whereas our scheme focuses on decoding the high priority data portion more effectively.

V. Experimental Results

A. Simulation Setup

Simulation is conducted with the following parameters: (n, k, h) = (150, 100, 40) and (n, k, h) = (1200, 1000, 400) to capture the performance difference using different packet sizes. The *Robust-Soliton* distribution parameters [4], $c = 0.04$ and $\delta = 0.5$ are used since these values yield good overhead performance from previous experiments. Due to the stochastic nature of the LT encoding, simulation is run 1,000 times to compute the average rate of successful 1) entire packet reception, 2) high priority data reception, and 3) low priority data reception.

The simulation is conducted in three different aspects. First, Ω values that yield the highest high priority data decoding performance are determined over independently generated erasures. Once we determine Ω, information bits that are decoded at each decoding process are plotted in order to capture the decoded information bit sequences. Histograms are plotted of decoder failures, which capture the packet failure distribution of the percentage of original information being decoded for the given failure. Performance analysis between the conventional LT codes and the prioritized LT scheme are compared side-by-side under different erasure error conditions.

B. Selecting Ω

To find the Ω that achieves good overall performance as well as good high priority data reception performance, numerical simulation is performed. We vary Ω while fixing (n, k, h, e) = (150, 100, 40, 0.1) and (1200, 1000, 400, 0.05) for 1,000 runs, where e is erasure error rate. The performance is measured based on the following matrices:

- P (successful decoding of all data) = $P(S)$
- P (decoding failure) = 1 - $P(S)$ = $P(F)$
- P(successful decoding of high priority data) = $P(S)$ + P(successful decoding of more than 90 percent of high priority data | decoding failures) = $P(S) + P(H \mid F)$
- P(successful decoding of low priority data) = $P(S)$ + P(successful decoding of more than 90 percent of low priority data | decoding failures) = $P(S) + P(L \mid F)$

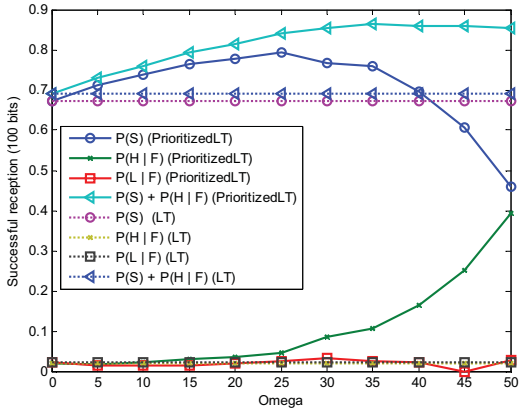

Fig. 3. Success rate with varying Ω with (n, k, h, e) = (150, 100, 40, 0.1)

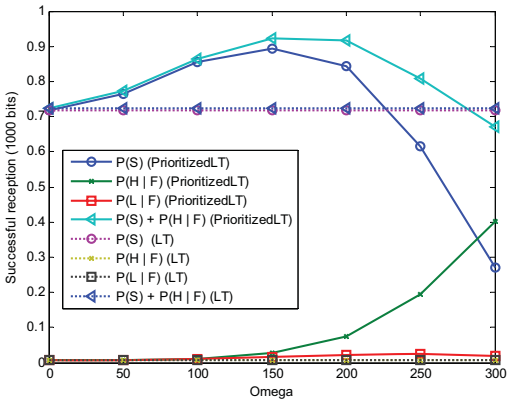

Fig. 4. Success rate with varying Ω with (n, k, h, e) = (1200, 1000, 400, 0.05)

For this work, we assumed that 90 percent or more of decoding high and low priority data is adequate enough to capture differences between two different schemes and underlying channel codes can provide protections.

In Figure 3 and 4, the x-axis indicates the different Ω values. As one can see, Ω values between 0 to 40 and 0 to 200 bits achieve better overall packet reception performance as well as good high priority data reception performance for k = 100 and k = 1000 bits respectively. When k = 100 bits, specifically Ω values around 25 bits achieve good overall and high priority data decoding performance, where 25 is close to the average number of generating degree 2 bits for high priority data h, $q_2 \cdot h$. For k = 1000 bits, Ω values either 150 or 200 bits yield the highest reception performance, which is close to $q_2 \cdot h$. Therefore, we verified that the optimum Ω that yields good performance is closely matched with $q_2 \cdot h$. As Ω increases, the overall packet decoding performance is rapidly decreased due to inefficiency of selecting too many degree 2 encoded bits from high priority data group in both cases. The low priority data decoding performance is about the same in both schemes.

C. Decoding Comparison

Figure 5 thru 8 capture a snapshot of decoded information sequences for the conventional LT encoding and prioritized scheme as the decoding process progresses with (n, k, h, e) = (150, 100, 40, 0.01) and (n, k, h, e) = (1200, 1000, 400, 0.01). The x-axis is the iteration step that the *belief propagation* decoder decodes information bits with XOR operations. The value of y-axis identifies the information bit at the iteration time it is decoded at each decoding step. The y values from 1 to 40 indicates the high priority data bits and from 41 to 100 are the low priority data. Similarly, for (n, k, h, e) = (1200, 1000, 400, 0.01) case, y values from 1 to 400 are high priority data and values from 400 to 1000 are the low priority data.

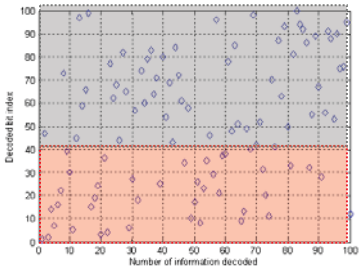

Fig. 5. Sequentially decoded information bits for the conventional LT codes with (n, k, h, e) = (150, 100, 40, 0.01)

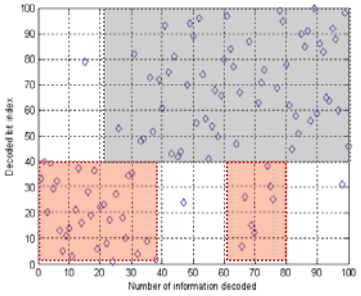

Fig. 6. Sequentially decoded information for the prioritized LT scheme with (n, k, h, e) = (150, 100, 40, 0.01)

Both schemes successfully decode all information bits. However, the conventional LT scheme decodes information bits uniformly over the entire data as decoding process progresses. On the other hand, the majority of high priority data is being decoded at the beginning of decoding process in the prioritized LT scheme. After recovering high priority data at the beginning, the decoder starts decoding low priority data. Hence, decoding process occurs in a cluster manner, jumping from decoding the high priority group to the low priority data group. This is a direct consequence of sending high priority data as a degree 1 and 2. Therefore, it is observed that the orders of bits that are decoded over each iteration step are clearly different.

Hence, if even a decoder stops due to low overhead sent or high erasures in the channel, the prioritized LT scheme can guarantee a certain level of rapidly successful high priority data reception. The prioritized LT scheme achieves the decoding of high priority data as fast as possible at the beginning.

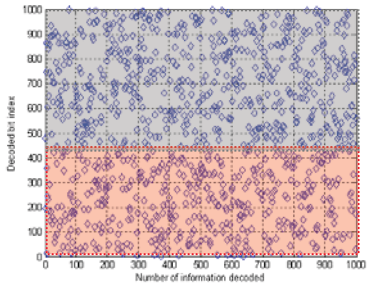

Fig. 7. Sequentially decoded information for the conventional LT codes with (n, k, h, e) = (1200, 1000, 400, 0.01)

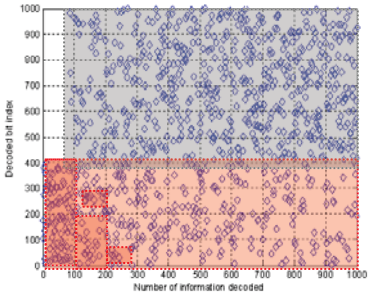

Fig. 8. Sequentially decoded information for the prioritized LT scheme with (n, k, h, e) = (1200, 1000, 400, 0.01)

D. Failure distributions over different erasure error rates

We characterize how much high priority data are decoded correctly, when a decoder fails to decode the entire message. We plot histograms with the number of occurrences of successful high priority packet reception for 1,000 runs with (n, k, h) = (150, 100, 40) with two different erasure rates.

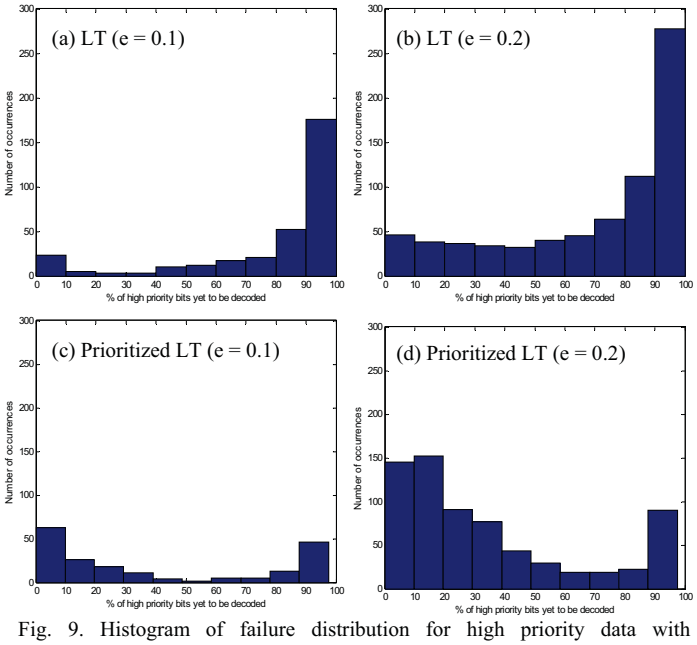

Fig. 9. Histogram of failure distribution for high priority data with conventional LT codes in (a) and (b) and prioritized LT scheme in (c) and (d)

The y-axis represents the number of occurrences and the x-axis is the percentages of high priority information bits yet to be decoded. Specifically, in Figure 9(a), among the total

packet failures, about 25 failures (the far left bar in the histogram) successfully decode 90 percent or more of high priority data. About 170 of the failures (the far right bar in the histogram) decode 10 percent or less of the high priority information bits. Erasure error rates from 10 and 20 percent are given to show differences in both schemes.

Comparing the two schemes, we clearly see that the packet failure distributions are different. In the prioritized LT scheme, many of failures are skewed towards the left side of the histograms, which implies many of high priority data are decoded correctly as compared to the conventional LT codes. This shows that when a decoder fails to decode entire information, it decodes the majority of the high priority data, which is consistent with the decoding examples given in Figure 5 thru 8.

Finally, to capture the overall performance on how much improvement or degradation can be seen between two schemes we plot the overall percentage of successful packet reception using same parameters in section V. *B*.

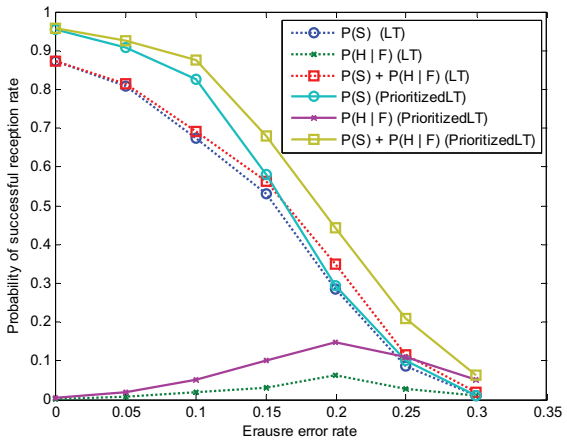

Fig. 10. High Priority Data Decoding performance of (n, k, h) = (150, 100, 40) over various erasure error rates

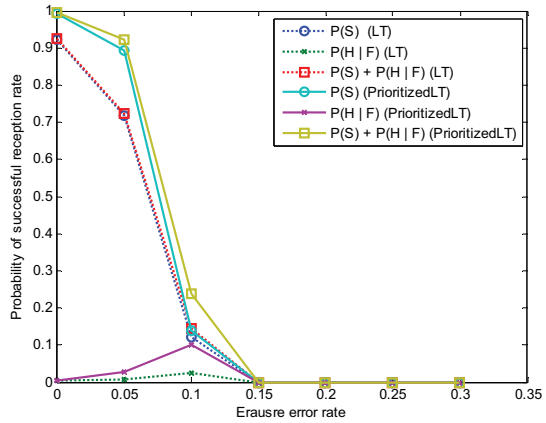

Fig. 11. High Priority Data Decoding performance of (n, k, h) = (1200, 1000, 400) over various erasure error rates

As we can see in Figure 10 and 11, more than 10 to 15 percent enhancement can be achieved on decoding high priority using the prioritized LT scheme over different erasure error rates without degrading overall decoding performance when $t < 0.5$ for different packet sizes. Moreover, high priority data decoding performance of our scheme scales well as the erasure error rate increases.

VI. Conclusion

In this work, we propose a scheme to control LT codes to send high priority data more effectively. Our scheme achieves better performance on decoding high priority data as well as overall information without penalizing the decoding of low priority data assuring high priority data is no more than the half of information bits. The cost is in some added complexity in the encoder. Potential benefits are in the protecting of high priority data in image, voice, and video transmissions in terrestrial and space communications.

VII. Acknowledgments

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology and was sponsored by the National Aeronautics and Space Administration. The authors would like to thank Matt Klimesh, Kar-Ming Cheung, Sam Dolinar, Jon Hamkins, Loren Clare, and Philip Tsao in the Communications Research Section at the Jet Propulsion Laboratory for their suggestions and feedback, which greatly clarified and improved this work.

References

[1] NASA Constellation Program, http://www.nasa.gov/mission_pages/constellation/main/index.html

[2] Consultative Committee for Space Data Systems (CCSDS), *CCSDS File Delivery Protocol* (*CFDP*), *Part 1*: *Introduction and Overview*, Consultative Committee for Space Data Systems, CCSDS 720.1-G-2, Green Book, Issue 2, Washington, DC, Sept. 2003.

[3] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 2004.

[4] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS)*, Vancouver, BC, Canada, Nov. 2002, pp. 271–280.

[5] R. M. Karp, M. Luby and A. Shokrollahi, "Finite-Length Analysis of LT codes," in *Proc. IEEE Int. Symp. Information Theory*, Chicago, IL, Jun./Jul. 2004, pp. 39.

[6] E. Maneva, and A. Shokrollahi, "New Model for Rigorous Analysis of LT codes," in *Proc. IEEE Int. Symp. on Information Theory (ISIT)*, Seattle, WA, Jul. 2006, pp. 2677-2679.

[7] N. Rahnavard, B. N. Vellambi, and F. Fekri, "Rateless Codes with Unequal Error Protection Property," in *Proc. IEEE Trans. on Information Theory,* vol. 53, no. 4, pp.1521-1532, Apr. 2007

[8] A. G. Dimakis, J. Wang, K. and Ramchandran, "Unequal Growth Codes: Intermediate Performance and Unequal Error Protection for Video Streaming," in *Proc. Int. Workshop on Multimedia Signal Processing (MMSP)*, October 2007, pp. 107-110.